Chapter 5
Signcryption Schemes Based on Bilinear Maps

Paulo S.L.M. Barreto, Benoît Libert, Noel McCullagh, and Jean-Jacques Quisquater

5.1 Introduction

As has been established in the previous chapters, signcryption is a cryptographic primitive which combines the message integrity, message origin authentication, and (if possible) signature non-repudiation properties of a traditional digital signature with the privacy-preserving property of a public key encryption scheme.

The last chapter discussed the construction of signcryption schemes based on the Diffie–Hellman problem. In this chapter we look at signcryption schemes resulting from bilinear maps, also commonly called "pairings." Since computing a bilinear map can be significantly slower than computing an exponentiation in a group of the same order, the development of signcryption schemes based on bilinear maps only makes sense if these schemes can provide an advantage over the simpler and potentially more efficient Diffie–Hellman-based schemes. This can be in the form of an improved security analysis or some extra property of the scheme. In this section, we discuss some of these advantages such as ciphertext anonymity and detachable signatures, and give examples of schemes that enjoy these properties.

Pairings were first brought to the attention of the cryptographic community when Menezes, Okamoto, and Vanstone described an attack using the Weil pairing to efficiently convert the Elliptic Curve Discrete Logarithm Problem (EC-DLP) to the Discrete Logarithm Problem in a finite field, which can be solved in sub-exponential time [138]. This is referred to as the MOV attack in the literature.

In 2000, Joux used bilinear maps in the construction of the first pairing-based cryptographic protocol [109]. This was a tripartite Diffie–Hellman key agreement protocol, and as such was subject to the man-in-the-middle attack. Importantly, it was the first non-destructive use of pairings in the literature.

A third paper, by Boneh and Franklin [45, 46], really got cryptographers excited by the new possibilities afforded by bilinear maps. It closed a long-standing open problem in cryptography. The problem of constructing an efficient, secure

B. Libert (✉)
Crypto Group, Microelectronics Laboratory, Université catholique de Louvain, Louvain, Belgium
e-mail: benoit.libert@uclouvain.be

A.W. Dent, Y. Zheng (eds.), *Practical Signcryption*, Information Security and Cryptography, DOI 10.1007/978-3-540-89411-7_5,

identity-based encryption (IBE) scheme was proposed by Shamir in 1984 [177]. In his paper Shamir proposed the first identity-based signature scheme, but left the construction of an identity-based encryption scheme as an open problem. Seventeen years later, in 2001, an efficient solution was proposed by Boneh and Franklin. This solution made use of bilinear maps.

Since the original paper by Boneh and Franklin there have been many identity-based and, indeed, non-identity-based protocols based on pairings. In addition to IBE [45, 46], we also have many flavors of identity-based signatures [18, 57], key agreement schemes [59, 137], and, as we shall see in Chap. 10, identity-based signcryption schemes.

5.2 Bilinear Map Groups

Definition 5.1 Let k be a security parameter and p be a k-bit prime number. Let us consider groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of order p and let g_1, g_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. We say that $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ are *bilinear map groups* if there exists a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties:

1. Bilinearity: $\forall (u, v) \in \mathbb{G}_1 \times \mathbb{G}_2 \; \forall a, b \in \mathbb{Z}$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: $\forall u \in \mathbb{G}_1, e(u, v) = 1 \; \forall v \in \mathbb{G}_2$ if and only if $u = 1_{\mathbb{G}_1}$.
3. Computability: $\forall (u, v) \in \mathbb{G}_1 \times \mathbb{G}_2$, $e(u, v)$ is efficiently computable.

In addition to the above general properties, the constructions described in this chapter additionally need an efficient, publicly computable (but not necessarily invertible) isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ such that $\psi(g_2) = g_1$. Many other pairing-based protocols (such as short signatures [47, 48]) require the availability of such an isomorphism, either in the implementation of schemes themselves or in their security proofs.

Such bilinear map groups are known to be instantiable with ordinary elliptic curves such as MNT curves [144] and the kind of curves studied by Barreto and Naehrig [20]. In practice, $\mathbb{G}_1$ is a p-order cyclic subgroup of such a curve $E(\mathbb{F}_r)$ while $\mathbb{G}_2$ is a subgroup of $E(\mathbb{F}_{r^\alpha})$, where α is the "embedding degree of the group" (i.e., the smallest integer α for which the order p of the group divides $r^\alpha - 1$). The group $\mathbb{G}_T$ is the set of p-th roots of unity in the finite field $\mathbb{F}_{r^\alpha}$. In this case, the trace map can be used as an efficient isomorphism ψ as long as $\mathbb{G}_2$ is properly chosen [183] within $E(\mathbb{F}_{r^\alpha})$.

The property of *computability* is ensured by Miller's famous algorithm [140, 141]—the detail of which is beyond the scope of this chapter. In p-order cyclic subgroups of curves of embedding degree α, the complexity of Miller's algorithm is dominated by $O(\log\ p)$ operations in the extension field $\mathbb{F}_{r^\alpha}$ containing the group $\mathbb{G}_T$. Computing a pairing is generally significantly more expensive than computing an elliptic curve scalar multiplication. Using a naïve implementation of Miller's algorithm, a pairing computation is more than α^2 times slower than a scalar multiplication on $E(\mathbb{F}_r)$. On the other hand, a recent paper by Scott [174]

estimates that most optimized algorithms for an embedding degree $\alpha = 2$ end up with a running time which is from two to four times as long as an RSA decryption. Regardless, pairing-based cryptographic protocols usually strive to minimize the number of pairing calculations they involve.

Some specific cryptographic protocols require the use of symmetric pairings, where $\mathbb{G}_1 = \mathbb{G}_2$ and ψ is the identity mapping. Such symmetric pairings have the additional commutativity property: for any pair $u, v \in \mathbb{G}_1^2$, $e(u, v) = e(v, u)$. Admissible mappings of this kind can be derived from the Weil and Tate pairings using special endomorphisms called "distortion maps" [194] that are known to only exist on a particular kind of curve termed "supersingular" in the literature.[1] Supersingular curves may be more susceptible to attacks than ordinary curves. Indeed, several optimization tricks for them [17] require the use of fields of small characteristic. The problem is that the MOV and Frey–Rück reductions [82, 138] reduce the discrete logarithm problem over the elliptic curve to the discrete logarithm problem in a finite field, and the discrete logarithm problem in a finite field is much easier to solve in fields of small characteristic [65] than in fields of large characteristic and similar overall size. Since this threat is well known, it is usually thwarted by increasing field sizes to maintain a sufficient level of security. Therefore, protocols where bandwidth requirements have to be minimized (e.g., [47, 48]) usually avoid supersingular curves whenever possible.

5.3 Assumptions

The security of the first scheme described in this chapter relies on a natural variant of the Diffie–Hellman problem introduced in [47, 48].

Definition 5.2 The *co-Diffie–Hellman (co-CDH)* problem in bilinear map groups $(\mathbb{G}_1, \mathbb{G}_2)$ is to compute $g_1^{ab} \in \mathbb{G}_1$ given $(g_1, g_2, g_1^a, g_2^b) \in (\mathbb{G}_1 \times \mathbb{G}_2)^2$ for random values $a, b \xleftarrow{R} \mathbb{Z}_p^*$. The advantage of a co-CDH solver is defined as the probability of finding g_1^{ab} taken over the random choice of a, b and the solver's coin tosses. In the following, we call $\mathrm{Adv}^{\mathrm{co-CDH}}(t, k)$ the maximal probability, taken over the random choice of $a, b \in \mathbb{Z}_p^*$ and the adversary's coin tosses, of solving a random co-CDH instance within time t when the security parameter is $k = \lfloor \log p \rfloor$.

The security properties of the second scheme described in the chapter rest on the intractability of the following problems introduced in [41, 42] which extend ideas from [143, 170].

Definition 5.3 Consider a set of bilinear map groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$.

[1] In fact, a curve $E(\mathbb{F}_r)$ is said to be supersingular if its number of points $\#E(\mathbb{F}_r)$ is such that $t = r + 1 - \#E(\mathbb{F}_r)$ is a multiple of the characteristic of $\mathbb{F}_r$.

- The *q-Diffie–Hellman Inversion problem* (q-DHI) consists of computing the value $g_1^{1/x} \in \mathbb{G}_1$ given a tuple $(g_1, g_2, g_2^x, g_2^{(x^2)}, \ldots, g_2^{(x^q)}) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$ for a randomly drawn $x \xleftarrow{R} \mathbb{Z}_p^*$.
- The *q-Strong Diffie–Hellman problem* (q-SDH) consists of computing a pair $(c, g_1^{1/(x+c)}) \in \mathbb{Z}_p \times \mathbb{G}_1$, given elements $(g_1, g_2, g_2^x, g_2^{(x^2)}, \ldots, g_2^{(x^q)}) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$ for a random $x \xleftarrow{R} \mathbb{Z}_p^*$.

Again, the advantage of solvers is defined as their probability, taken over the random choice of x and their own coin tosses, of finding the appropriate group element. In the following, we denote by $\mathrm{Adv}^{q\text{-DHI}}(t, k)$ (resp. $\mathrm{Adv}^{q\text{-SDH}}(t, k)$) the maximal probability of solving a random q-DHI (resp. q-SDH) instance within time t when the security parameter is $k = \lfloor \log p \rfloor$.

It should be emphasized that the strength of these assumptions grows with the parameter q (which will be the number of random oracle queries allowed for adversaries in games modeling their security). Since this parameter must be reasonably large—the upper bound $q \approx 2^{60}$ is frequently used in the literature—for proofs to be meaningful, those assumptions are notably less trustworthy than the standard computational Diffie–Hellman assumption.

However, despite recent concerns [61] regarding the hardness of the above problems, it still seems reasonable to use this scheme with an appropriate adjustment of key size. For instance, $|p| \approx 256$ seems to suffice if we settle for a security level equivalent to AES implemented with 128-bit keys.

5.4 Signcryption for Anonymous Communications

The schemes that we present in this chapter are anonymous and have "detachable signatures." The term "detachable signature" means that the output of the unsigncryption algorithm is a plaintext and some authentication material that can be forwarded to third parties who can check its validity using publicly available information. This is clearly equivalent to the notion of non-interactive non-repudiation proposed by Malone-Lee and discussed in Sect. 4.6.2.

Similar to certain identity-based signcryption schemes [51, 60], the constructions described in this chapter are meant to provide anonymous ciphertexts which do not reveal information on the identity of their author or recipient, much in the fashion of *key-private* public key cryptosystems [21].

We therefore begin by presenting a new syntax for a signcryption scheme and new security definitions. In particular, we will assume that there exists a single key derivation algorithm, which produces keys that can be used for both signcryption and unsigncryption (see Sect. 3.2.3). We present models for confidentiality, unforgeability, and anonymity.

For our purposes, a signcryption scheme consists of tuple of algorithms (`Setup`, `KeyGen`, `Signcrypt`, `Unsigncrypt`, `Verify`). The syntax of the first three algorithms (`Setup`, `KeyGen`, `Signcrypt`) remain the same as before, except that

the key generation algorithm KeyGen produces keys that can be used for both signcryption and unsigncryption. The unsigncryption algorithm takes as input a ciphertext C, a sender public key pk_S, and a receiver private key sk_R; it outputs either a message m and a detachable signature σ or an error symbol $\perp$. The verify algorithm is used to verify detached signatures. It takes as input a message m, a signature σ, and a receiver public key pk_R, and outputs either a valid symbol $\top$ or an invalid symbol $\perp$.

5.4.1 Message Privacy

The next definition captures message privacy: it is the equivalent of the multi-user insider confidentiality security model (FSO/FUO-IND-CCA2) presented in Sect. 3.2.1, except that we consider a single key generation algorithm.

Definition 5.4 We say that a signcryption scheme ensures *message privacy against chosen-ciphertext attacks* (we call this security notion insider FSO/FUO-IND-CCA2) if no PPT adversary has a non-negligible advantage in this game:

1. The challenger generates a private/public key pair (sk_U, pk_U). The private key sk_U is kept secret, while the public key pk_U is given to the adversary $\mathcal{A}$.
2. $\mathcal{A}$ first performs series of queries of the following kinds:

   - Signcryption queries: the adversary $\mathcal{A}$ produces a message $m \in \mathcal{M}$ and an arbitrary public key pk_R (which may differ from pk_U) and acquires the result $\texttt{Signcrypt}(param, sk_U, pk_R, m)$.
   - Unsigncryption queries: $\mathcal{A}$ produces a ciphertext C, a sender's public key pk_S, and acquires the result of $\texttt{Unsigncrypt}(param, pk_S, sk_U, C)$, which consists of a signed plaintext (m, σ) if the obtained signed plaintext is valid for the sender's public key pk_S and the $\perp$ symbol otherwise.

   These queries can be asked adaptively: each query may depend on the answers to previous ones. After a number of queries, $\mathcal{A}$ produces equal-length messages m_0, m_1 and an arbitrary private key sk_S.
3. The challenge flips a coin $b \xleftarrow{R} \{0, 1\}$ and computes the challenge signcryption $C \xleftarrow{R} \texttt{Signcrypt}(param, sk_S, pk_U, m_b)$ of m_b with the sender's private key sk_S and the public key pk_U. The ciphertext C is given to $\mathcal{A}$.
4. $\mathcal{A}$ performs new queries as in step 2 but he/she may not ask for the unsigncryption of the challenge ciphertext C with respect to the public key pk_S (i.e., the public key that corresponds to sk_S). At the end of the game, he/she outputs a bit b' and wins if $b' = b$.

$\mathcal{A}$'s advantage is defined to be $Adv_{\mathcal{A}}^{IND}(k) := |\Pr[b' = b] - 1/2|$.

5.4.2 Ciphertext Unforgeability and Signature Unforgeability

We define notions for message integrity (unforgeability). The main notion of unforgeability is the same as in Chap. 3 with the exception that it is suitable for signcryption schemes with a single key generation algorithm.

Definition 5.5 We say that a signcryption scheme is *strongly existentially ciphertext unforgeable* against insider chosen message attacks (FSO/FUO-sUF-CMA) if no PPT adversary $\mathcal{F}$ has a non-negligible advantage in the following game:

1. The challenger generates a key pair (sk_U, pk_U) and pk_U is given to the forger $\mathcal{F}$.
2. The forger $\mathcal{F}$ queries the signcryption and unsigncryption oracles in an adaptive fashion as in Definition 5.4.
3. $\mathcal{F}$ eventually outputs a ciphertext C and a key pair (sk_R, pk_R). The forger wins if the result $\texttt{Unsigncrypt}(param, pk_U, sk_R, C)$ is a pair (m, σ) such that (m, σ) is a valid signature with respect to the public key pk_U and C was not the output of a signcryption query $\texttt{Signcrypt}(param, sk_U, pk_R, m)$ during the game.

As in Sect. 3.2.2, the forger is allowed to have obtained the forged ciphertext as the result of a signcryption query for a different receiver's public key to which the one that the claimed forgery pertains.

Since we are concerned with specific constructions which allow receivers to extract authentication material (such as an ordinary digital signature) from a ciphertext and to forward it to a third party, non-repudiation with respect to this embedded authentication material may be sufficient in many contexts. This requirement is captured by the notion of *signature unforgeability* which was introduced for the first time by Boyen [51] and is recalled below.

Definition 5.6 A scheme is *existentially signature unforgeable* against chosen message attacks (or has the FSO/FUO-ESUF-CMA security) if no PPT adversary $\mathcal{F}$ has a non-negligible advantage against a challenger in this game:

1. The challenger generates a key pair (sk_U, pk_U) and pk_U is given to the forger $\mathcal{F}$.
2. $\mathcal{F}$ adaptively performs a series of queries to the signcryption and unsigncryption oracles as in Definition 5.4.
3. $\mathcal{F}$ outputs a ciphertext C and a key pair (sk_R, pk_R) and wins if the result of $\texttt{Unsigncrypt}(param, pk_U, sk_R, C)$ is a pair (m, σ) such that the pair (m, σ) is valid with respect to the public key pk_U and no signcryption query involving the message m and some receiver's public key pk'_R resulted in a ciphertext C' for which the output of $\texttt{Unsigncrypt}(param, pk_U, sk'_R, C')$ is (m, σ).

Of course, considering non-repudiation with respect to underlying signatures (instead of ciphertexts) only makes sense for schemes where the receiver extracts a signature from the ciphertext.

A potential incentive to settle for signature unforgeability is that it may reduce the amount of data that receivers have to forward to third parties coping with non-repudiation disputes. For instance, the scheme described in Sect. 5.6 allows receivers to extract short signatures from ciphertexts.

In settings where signature unforgeability suffices, a complementary notion was also introduced in [51]. It was called *ciphertext authentication* and assures that a receiver is always convinced that a ciphertext was jointly signed and encrypted by the same person and was not subject to a kind of man-in-the-middle attack. The resulting model shares many similarities with the multi-user outsider unforgeability model described in Sect. 3.2.2, with the exception that it only applies to signcryption schemes with one key generation algorithm.

Definition 5.7 A signcryption scheme has the *ciphertext authentication* property (FSO/FUO-AUTH-CMA) if no PPT adversary $\mathcal{F}$ has a non-negligible advantage in the next game:

1. The challenger generates two key pairs (sk_S, pk_S) and (sk_R, pk_R); pk_S and pk_R are given to the forger.
2. The forger $\mathcal{F}$ performs queries to the signcryption oracles $\texttt{Signcrypt}(param, sk_U, \cdot, \cdot)$ and the unsigncryption $\texttt{Unsigncrypt}(param, \cdot, pk_U, \cdot)$, for both $U = S$ and $U = R$, as in previous definitions.
3. $\mathcal{F}$ produces a ciphertext C and wins if the result of $\texttt{Unsigncrypt}(param, pk_S, sk_R, C)$ is a pair (m, σ) such that (m, σ) is a valid signature for the public key pk_S and no signcryption query involving the message m and the receiver's public key pk_R produced in the ciphertext C.

We emphasize that the latter definition is only useful in complement to the signature unforgeability property. These properties should not be considered if one is merely concerned with the ciphertext unforgeability in the sense of Definition 5.5.

5.4.3 Anonymity

In [51], Boyen suggested other security properties for signcryption schemes. One of them was called *ciphertext anonymity* and can be thought of as extending the notion of *key privacy* as considered by Bellare et al. [21] for public key encryption schemes. Intuitively, a public key encryption scheme is anonymous if ciphertexts convey no information about the public key that was used to create them.

In the signcryption setting, the ciphertext anonymity property is satisfied if ciphertexts reveal no information about who created them nor about whom they are intended to. This intuition is captured by the definition below which transposes the one given in [51] to a traditional public key setting.

Definition 5.8 A signcryption scheme is said to be *ciphertext anonymous* (FSO/FUO-ANON-CCA) if no PPT distinguisher $\mathcal{D}$ has a non-negligible advantage in the following game:

1. The challenger generates two distinct key pairs (sk_{R_0}, pk_{R_0}) and (sk_{R_1}, pk_{R_1}). The distinguisher $\mathcal{D}$ is provided with pk_{R_0} and pk_{R_1}.
2. $\mathcal{D}$ adaptively performs queries to the signcryption and unsigncryption oracles for the key pairs (sk_{R_0}, pk_{R_0}) and (sk_{R_1}, pk_{R_1}) as in previous definitions. $\mathcal{D}$ eventually outputs two private keys sk_{S_0} and sk_{S_1} and a plaintext m.

3. The challenger then flips coins $b, b' \stackrel{R}{\leftarrow} \{0, 1\}$ and computes a challenge ciphertext $C \stackrel{R}{\leftarrow} \texttt{Signcrypt}(param, sk_{S_b}, pk_{R_{b'}}, m)$.
4. $\mathcal{D}$ adaptively issues new queries as in step 2 with the restriction that $\mathcal{D}$ may not ask for the unsigncryption of pairs (C, pk_{S_j}), where $j \in \{0, 1\}$ and C is the challenge ciphertext, under either of the private keys sk_{R_0} or sk_{R_1}. Eventually, $\mathcal{D}$ outputs bits d, d' and wins if $(d, d') = (b, b')$.

The adversary's advantage is defined as $Adv_{\mathcal{D}}^{ANON}(k) := |\Pr[(d, d') = (b, b')] - \frac{1}{4}|$.

Again, this notion captures the security against insider attacks as the distinguisher is allowed to choose a pair of private keys among which the one used to create the challenge ciphertext is picked by the challenger.

5.5 A Tightly Secure Scheme

This section describes a signcryption scheme whose security is tightly related to the hardness of a natural variant of the Diffie–Hellman problem in bilinear map groups. It was originally proposed, in a slightly modified form, by Libert and Quisquater [123]. This method relies on the digital signature algorithm of Boneh et al. [47, 48]. In this scheme, private keys consist of an integer $x \in \mathbb{Z}_p^*$ and public keys consist of a group element $Y = g_2^x \in \mathbb{G}_2$. A signature on a message m has the shape $\sigma = H(m)^x \in \mathbb{G}_1$ (where the hash function H maps arbitrary messages onto the cyclic group $\mathbb{G}_1$). This signature can be verified by checking that $e(\sigma, g_2) = e(H(m), Y)$. In order to enhance the concrete security of the reduction in the proof of ciphertext unforgeability, a random quantity U that is used for encryption purposes also acts as a random salt to provide a tighter security reduction in the random oracle model [29].

The scheme may be viewed as a composition of a digital signature scheme which is existentially unforgeable against chosen message attacks (UF-CMA) [91] with a public key encryption scheme that is only secure against chosen plaintext attacks. In [10], it was already observed in the outsider security model that a sequential composition in the "sign-then-encrypt" order can amplify rather than simply preserve the security properties of the underlying building blocks—see Theorem 2.3. This construction gives another example showing that a CCA-secure signcryption system (in the sense of Definition 5.4) may be obtained from weaker building blocks. Here, in some sense, the redundancies needed to achieve CCA security are embedded in the signature.

5.5.1 The Scheme

For the security of the scheme depicted in Fig. 5.1, it is crucial that the underlying symmetric encryption scheme be deterministic and one-to-one: for a given plaintext and symmetric key, there should be a single possible ciphertext. If it were possible

$\texttt{Setup}(1^k)$
Pick a set of bilinear map groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ whose order is a prime p such that $2^{k-1} < p < 2^k$ and generators $(g_1, g_2) \in \mathbb{G}_1 \times \mathbb{G}_2$ such that $g_1 = \psi(g_2)$
Let ℓ_1 be the bitlength of elements of $\mathbb{G}_1$
Choose an one-time symmetric-key encryption scheme $SE = (\texttt{Enc}, \texttt{Dec})$ with key space $\mathcal{K}$ and ciphertext space $\mathcal{C}$
Pick cryptographic hash functions:
$H_1 : \{0,1\}^* \to \mathbb{G}_1^*$
$H_2 : \mathbb{G}_1^2 \to \{0,1\}^{\ell_1}$
$H_3 : \mathbb{G}_1^3 \to \mathcal{K}$
$param \leftarrow (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, SE, H_1, H_2, H_3, \ell_1)$
Return $param$

$\texttt{KeyGen}(param)$
$x_U \xleftarrow{R} \mathbb{Z}_p^*$; $y_U \leftarrow g_2^{x_U}$
$sk_U \leftarrow x_U$; $pk_U \leftarrow y_U$
Return (sk_U, pk_U)

$\texttt{Signcrypt}(m, sk_S, pk_R)$
Parse sk_S as (x_S, y_S); Parse pk_R as y_R
If $y_S, y_R \notin \mathbb{G}_2 \setminus \{1\}$ then return $\perp$
$r \xleftarrow{R} \mathbb{Z}_p^*$; $u \leftarrow g_1^r$
$v \leftarrow H_1(m \| u \| pk_S \| pk_R)^{x_S}$
$w \leftarrow v \oplus H_2(u \| \psi(y_R)^r)$
$\tau \leftarrow H_3(u \| v \| \psi(y_R)^r)$
$z \leftarrow \texttt{Enc}_\tau(m)$
$C \leftarrow (u, w, z)$
Return C

$\texttt{Unsigncrypt}(C, pk_S, sk_R)$:
Parse pk_S as y_S; Parse sk_R as (x_R, y_R)
If $y_S, y_R \notin \mathbb{G}_2 \setminus \{1\}$ then return $\perp$
Parse C as (u, w, z)
If $u \notin \mathbb{G}_1$ or $w \notin \{0,1\}^{\ell_1}$ or $z \notin \mathcal{C}$ Return $\perp$
$v \leftarrow w \oplus H_2(u \| u^{x_R})$
If $v \notin \mathbb{G}_1$ then return $\perp$
$\tau \leftarrow H_3(u \| v \| u^{x_R})$
$m \leftarrow \texttt{Dec}_\tau(z)$
If $e(v, g_2) \neq e(H_1(m \| u \| pk_S \| pk_R), y_S)$ Return $\perp$
Return (m, pk_R, v, u)

Fig. 5.1 The co-CDH-based scheme

to compute another encryption of the same plaintext m given $c = \texttt{Enc}(m)$, it would be possible to generate another ciphertext (u, w, z') given (u, w, z) and thus defeat the chosen ciphertext security of the whole scheme.

The receiver has to forward m, v, pk_R, and u to a third party to convince her that the message actually comes from the sender. Together with u and pk_R, the value v acts as a "detachable signature" that the receiver can extract from the ciphertext and transmit it to third parties. This signature is verified by checking that $e(v, g_2) = e(H_1(m \| u \| pk_S \| pk_R), y_S)$.

We note that the recipient's public key must be hashed together with the pair (m, u) in order to achieve the *strong* unforgeability according to Definition 5.5.

5.5.2 *Efficiency*

Three exponentiations in $\mathbb{G}_1$ are required in the signcryption algorithm, while one multiplication and two pairings must be performed at unsigncryption. The scheme is at least as efficient and more compact than most sequential compositions of the BLS signature [47, 48] with any CCA-secure Diffie–Hellman-based encryption scheme [11, 14, 68, 83, 84, 161, 181]. For example, a sequential combination of the BLS signature scheme [47, 48] with an ElGamal [81] encryption padded with

the Fujisaki–Okamoto conversion [83] would involve an additional exponentiation at decryption because of the "re-encryption phase" which checks the validity of the ciphertext. With $\ell_1 \approx k \geq 171$, this construction saves about 171 bits of overhead (i.e., the difference between ciphertext and plaintext sizes) with respect to a composition of the BLS signature scheme with Fujisaki–Okamoto/ElGamal.

The scheme looks like a sequential composition of the BLS signature scheme [47, 48] with the hybrid KEM/DEM ElGamal encryption scheme proven secure by Cramer and Shoup [68]. Actually, the hybrid ElGamal scheme *must* be implemented with an IND-CCA2 symmetric encryption while the above system only needs a symmetric scheme that meets the very weak requirement of being semantically secure against passive attacks (that is an attack where the adversary has no encryption or decryption oracle in an indistinguishability scenario). Here, for fixed-length messages, the symmetric encryption could simply be a "one-time pad" of the message with a hash value of $u \| v \| \psi(Y_R)^r$.

5.5.3 Security

The original version [123] of this system (where τ was obtained by hashing v alone) was found [186, 198] not to meet its intended security properties. Although a chosen ciphertext attack was also given [187] against the modification suggested in [198], its variant detailed in Fig. 5.1 is immune to these attacks (and the countermeasures do not incur any significant additional cost).

The scheme is proven secure in the random oracle model (with a tight reduction) assuming the hardness of the co-CDH problem. The proof of the next theorem features a tight reduction to the co-CDH problem using the property (pointed out for the first time in [111] for a specific kind of pairing-friendly groups) that its decisional counterpart is easy: it can be easily tested whether a given tuple $(g_1, g_2, g_1^a, g_2^b) \in (\mathbb{G}_1 \times \mathbb{G}_2)^2$ satisfies $a = b$ by checking if $e(g_1, g_2^b) = e(g_1^a, g_2)$.

Theorem 5.1 *The scheme is FSO/FUO-IND-CCA2 secure in the random oracle model assuming that the co-CDH problem is hard and that the symmetric encryption scheme is IND-CPA secure. For any adversary $\mathcal{A}$ running in time t_A and making at most q_{sc} signcryption queries, q_{usc} unsigncryption queries, and q_{H_i} queries to random oracles H_i ($i = 1, 2, 3$), we have*

$$\mathrm{Adv}_{\mathcal{A}}(t_A, k) \leq \frac{q_{usc}}{2^{k-2}} + \mathrm{Adv}_{\mathcal{B}}^{co\text{-}CDH}(t', k) + \mathrm{Adv}_{\mathcal{B}}^{ind\text{-}cpa\text{-}sym}(t', |\mathcal{K}|)$$

where

- $\mathrm{Adv}_{\mathcal{B}}^{co\text{-}CDH}(t', k)$ *stands for the maximal probability of solving the co-CDH problem in time* $t' \leq t_A + O(q_{usc} + q_{H_2} + q_{H_3})t_p + O(q_{H_1})t_{exp}$ *when the security parameter is* k *and* t_p, t_{exp} *stand for the time complexity of a pairing evaluation and an exponentiation, respectively.*

- $\mathrm{Adv}_{\mathcal{B}}^{ind\text{-}cpa\text{-}sym}(t', |\mathcal{K}|)$ *is the maximal advantage*[2] *of any adversary mounting a chosen plaintext attack on* $(\mathtt{Enc}, \mathtt{Dec})$ *within time* t' *when the key size is* $|\mathcal{K}|$.

Proof The proof consists of a sequence of games where the first game is the real attack game, and in the last game the adversary is essentially a passive attacker against the symmetric encryption scheme $(\mathtt{Enc}, \mathtt{Dec})$. In the sequence, the event that the adversary $\mathcal{A}$ wins in Game i is denoted S_i.
Game 1 is the real attack game detailed in Definition 5.4. The adversary is given public parameters comprising $g_2 \in \mathbb{G}_2$ and $g_1 = \psi(g_2)$ and the receiver's public key is defined as $pk_u = g_2^b \in \mathbb{G}_2$ for some random $b \overset{R}{\leftarrow} \mathbb{Z}_p^*$ chosen by the simulator $\mathcal{B}$. The latter uses $sk_U = b$ to answer all signcryption/unsigncryption queries. Random oracle queries are dealt with in the standard way, by returning random values in the appropriate range. To maintain consistency and return identical outputs if the same random oracle query is made more than once, $\mathcal{B}$ keeps track of all these queries and their outputs in lists L_1, L_2, and L_3. In the challenge phase, $\mathcal{A}$ outputs a pair of messages m_0, m_1 and a sender's private key $sk_S^* = x_S^*$. The simulator $\mathcal{B}$ flips a fair coin $d \overset{R}{\leftarrow} \{0, 1\}$. It also chooses a random exponent $a \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and successively computes ciphertext elements $u^* = g_1^a$, $v^* = H_1(m_d \| u^* \| g_2^{x_S^*} \| pk_u)^{x_S^*}$, $w^* = v^* \oplus H_2(u^* \| \psi(pk_u)^a)$, $\tau^* = H_3(u^* \| v^* \| \psi(pk_u)^a)$, and $z^* = \mathtt{Enc}_{\tau^*}(m_d)$. The challenge ciphertext (u^*, w^*, z^*) is given to $\mathcal{A}$ who eventually outputs a bit d' and wins if $d' = d$. The adversary $\mathcal{A}$'s advantage is thus $|\Pr[S_1] - 1/2|$.
Game 2: In this game, the first ciphertext component $u^* = g_1^a$ is calculated at the beginning of the game. This change is purely conceptual and $\Pr[S_2] = \Pr[S_1]$.
Game 3 is the same as Game 2 but the simulator $\mathcal{B}$ aborts if the adversary ever queries the unsigncryption of a ciphertext (u, w, z) such that $u = u^* = g_1^a$ *before* the challenge phase. We call F_3 the latter event. Game 3 and Game 2 are clearly identical until it occurs and we have $|\Pr[S_3] - \Pr[S_2]| \leq \Pr[F_3]$. Since u^* is independent of $\mathcal{A}$'s view until the challenge phase, we have $\Pr[F_3] \leq q_{usc}/p \leq q_{usc}/2^{k-1}$ so that $|\Pr[S_3] - \Pr[S_2]| \leq q_{usc}/2^{k-1}$.
Game 4: We modify the treatment of random oracle queries as well as that of signcryption/unsigncryption queries. A difference with earlier games is that H_2 and H_3 queries are now handled using four lists L_2, L_2' and L_3, L_3'.

- H_1 queries: If a hash query $H_1(m_i \| u_i \| pk_{S,i} \| pk_{R,i})$ is made, $\mathcal{B}$ first checks if the value of H_1 was previously defined for that input. If it was, the previously defined hash value $h_{1,i}$ is returned. Otherwise, $\mathcal{B}$ picks a random $t_i \overset{R}{\leftarrow} \mathbb{Z}_p^*$, returns $h_{1,i} \leftarrow g_1^{t_i} \in \mathbb{G}_1$, and inserts the tuple $(m_i, u_i, pk_{S,i}, pk_{R,i}, t_i)$ into L_1.
- H_2 queries: If a hash query $H_2(u_i \| R_i)$ is made, for inputs $(u_i, R_i) \in \mathbb{G}_1^2$, $\mathcal{B}$ first scans list L_2 to see if there exists a record $(u_i, R_i, h_{2,i}, \beta)$ for some

[2] This advantage is usually defined as $|\Pr[d = d'] - 1/2|$ when the adversary chooses a pair equal-length plaintexts m_0, m_1, obtains $c = \mathtt{Enc}_\tau(m_d)$ for a random key $\tau \overset{R}{\leftarrow} \mathcal{K}$ and a randomly drawn bit $d \overset{R}{\leftarrow} \{0, 1\}$, and outputs $d' \in \{0, 1\}$.

bit β. If so, the previously defined value $h_{2,i}$ is returned. Otherwise, $\mathcal{B}$ checks if (g_2, u_i, pk_u, R_i) is a valid co-Diffie–Hellman tuple (in our notation, we write $R_i = \text{co-DH}_{g_2}(u_i, pk_u)$) by checking whether $e(R_i, g_2) = e(u_i, pk_u)$.

- If yes, then $\mathcal{B}$ checks if L_2' contains an entry of the shape $(u_i, ?, h_{2,i})$ for some string $h_{2,i} \in \{0,1\}^{\ell_1}$. If yes, $h_{2,i}$ is returned and a record $(u_i, R_i, h_{2,i}, 1)$ is stored in L_2. If no entry $(u_i, ?, h_{2,i})$ exists in L_2', $\mathcal{B}$ returns a random string $h_{2,i} \stackrel{R}{\leftarrow} \{0,1\}^{\ell_1}$ and inserts $(u_i, R_i, h_{2,i}, 1)$ in L_2.
- If (g_2, u_i, pk_u, R_i) is not a co-DH tuple, $\mathcal{B}$ picks $h_{2,i} \stackrel{R}{\leftarrow} \{0,1\}^{\ell_1}$ at random and stores the tuple $(u_i, R_i, h_{2,i}, 0)$ in L_2.

- H_3 queries: If a hash query $H_3(u_i \| v_i \| R_i)$ is made, then $\mathcal{B}$ proceeds as for answering H_2 queries, using lists L_3 and L_3' to maintain the consistency and checking if (g_2, u_i, pk_u, R_i) is a co-Diffie–Hellman tuple, namely, L_3 contains entries of the form $(u_i, v_i, R_i, h_{3,i}, \beta)$, with $\beta \in \{0,1\}$. If $\beta = 1$, then $H_3(u_i \| v_i \| R_i) = h_{3,i}$ and it holds that $R_i = \text{co-DH}_{g_2}(u_i, pk_u)$. If $\beta = 0$, then $H_3(u_i \| v_i \| R_i) = h_{3,i}$ and $R_i \neq \text{co-DH}_{g_2}(u_i, pk_u)$. The auxiliary list L_3' contains entries $(u_i, v_i, ?, h_{3,i})$ such that a subsequent query $H_3(u_i \| v_i \| R_i)$ for which $R_i = \text{co-DH}_{g_2}(u_i, pk_u)$ should receive the answer $h_{3,i} \in \mathcal{K}$.
- Signcryption queries: If a signcryption query on a plaintext m and a recipient's public key pk_R is made, then $\mathcal{B}$ picks a random $r \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$, computes $u = g_1^r \in \mathbb{G}_1$, and checks if L_1 contains a tuple (m, u, pk_u, pk_R, t) indicating that $h_1(m \| u \| pk_u \| pk_R)$ was previously set to be g_1^t. If no such tuple is found, $\mathcal{B}$ picks $t \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and stores the entry (m, u, pk_u, pk_R, t) in L_1. It then computes $v = \psi(pk_u)^t = (g_1^b)^t \in \mathbb{G}_1$. The rest follows as in the signcryption process: $\mathcal{B}$ computes $\psi(pk_R)^r$ (for the pk_R specified by the adversary), simulates H_2 and H_3 to obtain $h_2 = H_2(u \| \psi(pk_R)^r)$ and $\tau = H_3(u \| v \| \psi(pk_R)^r)$, and then computes $w = v \oplus h_2$ and $z = \mathtt{Enc}_\tau(m)$. The ciphertext (u, w, z) is returned to $\mathcal{A}$.
- Unsigncryption queries: For an unsigncryption query on a ciphertext $C = (u, w, z)$ and a sender's public key $pk_S \in \mathbb{G}_2$, $\mathcal{B}$ checks if list L_2 contains the sole possible tuple $(u, R, h_2, 1)$ for some $R \in \mathbb{G}_1$ and $h_2 \in \{0,1\}^{\ell_1}$ (meaning that $R = \text{co-DH}_{g_2}(u, pk_u)$ and that $H_2(u \| R)$ was set to $h_2 \in \{0,1\}^{\ell_1}$):

  - If such an entry exists, $\mathcal{B}$ obtains $v = w \oplus h_2$ and rejects C if $v \notin \mathbb{G}_1$. Otherwise, $\mathcal{B}$ obtains the secret key $\tau = H_3(u \| v \| R) \in \mathcal{K}$ (by simulating H_3) and sets $m = \mathtt{Dec}_\tau(z)$. Then, $\mathcal{B}$ computes $H = H_1(m \| u \| pk_S \| pk_u) \in \mathbb{G}_1$ (by simulating H_1) and checks whether $e(v, g_2) = e(H, pk_S)$. If so, the information (m, pk_u, v, u) is returned. Otherwise, C is rejected.
  - If such an entry does not exist, then $\mathcal{B}$ checks if L_2' contains an entry $(u, ?, h_2)$. If such an entry does not exist either, then $\mathcal{B}$ chooses $h_2 \stackrel{R}{\leftarrow} \{0,1\}^{\ell_1}$ and stores $(u, ?, h_2)$ in L_2', so as to preserve consistency and answer h_2 to a subsequent H_2 query on the input $u \| \text{co-DH}_{g_2}(u, pk_u)$. In either case, $\mathcal{B}$ obtains the h_2 value. It sets $v = w \oplus h_2 \in \{0,1\}^{\ell_1}$ and rejects C if $v \notin \mathbb{G}_1$. Then, $\mathcal{B}$ scans lists L_3 and L_3', in search for an entries of the shape $(u, v, R, \tau, 1)$

and $(u, v, ?, \tau)$, respectively. If no such entries exist, $\mathcal{B}$ picks $\tau \overset{R}{\leftarrow} \mathcal{K}$ at random and inserts a record $(u, v, ?, \tau)$ in L'_3 to make sure that a future hash query $H_3(u\|v\|\text{co-DH}_{g_2}(u, pk_u))$ will get the answer τ. Finally, $\mathcal{B}$ computes $m = \mathtt{Dec}_\tau(z)$. The ciphertext C is declared invalid if $e(v, g_2) \neq e(H, pk_S)$ where $H = H_1(m\|u\|pk_S\|pk_u)$. If the ciphertext is deemed valid, $\mathcal{A}$ is returned the information (m, pk_u, v, u).

It can be checked that the above simulation of the various oracles is consistent and $\mathcal{A}$'s view is not altered by these changes. It comes that $\Pr[S_4] = \Pr[S_3]$. We also note that the private key $sk_U = b$ is not explicitly used to answer signcryption or unsigncryption queries.
Game 5 modifies the way to answer unsigncryption queries and add a special rule that applies to *post-challenge* unsigncryption queries, namely, if $\mathcal{A}$ queries the unsigncryption of a ciphertext (u, w, z) such that $(u, w) = (u^*, w^*)$ after the challenge phase, $\mathcal{B}$ returns $\perp$. Two situations must be distinguished to see that this change does not significantly alter $\mathcal{A}$'s view.

- If the query pertains to the same sender's public key as in the challenge phase (i.e., $pk_S = pk_S^*$), we necessarily have $z \neq z^*$ (as the query is illegal otherwise). For such a ciphertext, the underlying $v = w^* \oplus H_2(u^*\|u^{*b})$ must be the same as the value v^* calculated in the challenge phase. Also, the same symmetric key $\tau^* = H_3(u^*\|v^*\|u^{*b})$ must be used to decipher z when the unsigncryption operation is carried out normally. Since $z \neq z^*$ and given that the encryption/decryption algorithms $(\mathtt{Enc}, \mathtt{Dec})$ are bijections, the plaintext $m = \mathtt{Dec}_{\tau^*}(z)$ must be different from the plaintext m_d that was encrypted in the challenge phase. Hence, unless we have a collision $H_1(m\|u^*\|pk_S^*\|pk_u) = H_1(m_d\|u^*\|pk_S^*\|pk_u)$ (which occurs with probability smaller than $1/|p| < 1/2^{k-1}$ when H_1 is modeled as a random oracle), v^* cannot be a valid signature for the message $m\|u^*\|pk_S^*\|pk_u$ and the unsigncryption algorithm would certainly reject it.
- If the query is made for a different sender $pk_S \neq pk_S^*$ (and we may thus have $z = z^*$ or not), the unsigncryption algorithm would still reveal the same v^* as in the challenge phase and the same symmetric key $\tau^* = H_3(u^*\|v^*\|u^{*b})$ would be used to decipher z. If we denote by $m = \mathtt{Dec}_{\tau^*}(z)$ the symmetric decryption of z under the key τ^*, the ciphertext (u^*, w^*, z) would only be accepted in earlier games in the event that $H_1(m_d\|u^*\|pk_S^*\|pk_u)^{sk_S^*} = H_1(m\|u^*\|pk_S\|pk_u)^{\log_g(pk_S)}$ (i.e., if the outputs of the random oracle H_1 are correlated in a very specific way for two different inputs). Since we are working in the random oracle model, this situation only occurs with probability $1/p < 1/2^{k-1}$.

Throughout all queries, the overall probability that the new rule causes $\mathcal{B}$ to reject a ciphertext that would have been deemed valid in earlier games is at most $q_{usc}/2^{k-1}$. We thus have $|\Pr[S_5] - \Pr[S_4]| \leq q_{usc}/2^{k-1}$.

Game 6 makes some last changes to the simulation. First, we modify the generation of the challenge ciphertext (u^*, w^*, z^*). The first element u^* is still set as $u_1^* = g_1^a$ ($\mathcal{B}$ does not explicitly know a—only g_1^a) but $w^* \stackrel{R}{\leftarrow} \{0,1\}^{\ell_1}$ is now chosen at random and z^* is generated as an encryption $z^* = \mathtt{Enc}_{\tau^*}(m_d)$ under a perfectly random key $\tau^* \stackrel{R}{\leftarrow} \mathcal{K}$. The other change is that we define an event E and let the simulator halt if it ever occurs. Event E is the occurrence of one of the following situations.

E.1 $\mathcal{A}$ queries oracle H_2 on the input $(u^* \| R)$ such that $R = \text{co-DH}_{g_2}(u^*, pk_u)$.
E.2 $\mathcal{A}$ queries oracle H_3 on an input $(u^* \| . \| R)$ such that $R = \text{co-DH}_{g_2}(u^*, pk_u)$.

We see that $\mathcal{B}$ is able to detect occurrences of E.1 and E.2, which both reveal the value $R = \text{co-DH}_{g_2}(g_1^a, g_2^b) = g_1^{ab}$. Since $\mathcal{B}$ never knows exponents a or b, it would solve an instance of the co-CDH problem if E.1 $\vee$ E.2 happens. We thus have $\Pr[E.1 \vee E.2] \leq \text{Adv}_{\mathcal{B}}^{co\text{-}CDH}(t', k)$, where $t' \leq t_A + O(q_{usc} + q_{H_2} + q_{H_3})t_p + O(q_{H_1})t_{exp}$ is an upper bound on $\mathcal{B}$'s computation time that takes into account the unsigncryption queries as well as H_2 and H_3 queries, each requiring two pairing evaluations.

If event E does not occur, the symmetric key τ^* is completely independent of $\mathcal{A}$'s view. Guessing $d \in \{0,1\}$ then amounts to carry out a chosen plaintext attack on the symmetric encryption scheme $(\mathtt{Enc}, \mathtt{Dec})$. Indeed, the only way for $\mathcal{A}$ to observe symmetric decryptions under τ^* would be to query the unsigncryption of ciphertexts of the shape (u^*, w^*, z). Such ciphertexts are precisely rejected by the oracle due to the rules introduced in Game 3 and Game 5. It comes that $|\Pr[S_6] - 1/2| = \text{Adv}_{\mathcal{B}}^{ind\text{-}cpa\text{-}sym}(t')$. □

We observe that the reduction is very tight. Up to negligible terms and assuming that $\text{Adv}_{\mathcal{B}}^{ind\text{-}cpa\text{-}sym}(t')$ is negligible, algorithm $\mathcal{B}$ has the essentially same probability to solve the co-CDH problem as the adversary's advantage in breaking the scheme. The cost of the reduction is also bounded by an expression which is linear in the number of adversarial queries. That is the reason why u is included among the arguments of H_2. The scheme remains secure if v is concealed by a hash value of $\psi(pk_R)^r$ alone but the reduction entails a number of pairing evaluations that are quadratic in the number of adversarial queries.

Having a tight reduction to a computational problem is a notable feature of the scheme. It contrasts with other Diffie–Hellman-based constructions which are also CCA secure under tight reductions but rely on gap assumptions involving oracles that do not exist. This can be lifted by implementing those schemes over pairing-friendly groups (where some gap problems become equivalent to computational problems) but most of them keep relatively loose reductions with respect to unforgeability (the only notable exception being Bjørstad and Dent's CM scheme [37] discussed in Sect. 4.7). In the present system, the reductions are also efficient in the proof of ciphertext unforgeability.

Theorem 5.2 *Assume that an adversary $\mathcal{F}$ has advantage $\text{Adv}_{\mathcal{F}}(t, k)$ over the FSO/FUO-sUF-CMA security of the scheme when running in time t, making q_{sc}*

signcryption queries, q_{usc} unsigncryption queries, and q_{H_i} queries on random oracles H_i (for $i = 1, 2, 3$). Then, there is an algorithm $\mathcal{B}$ which solves the co-CDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$ with probability

$$\mathrm{Adv}_{\mathcal{B}}^{co\text{-}CDH}(t', k) \geq \mathrm{Adv}_{\mathcal{F}}(t, k) - \frac{q_{sc}(q_{H_1} + q_{sc} + q_{usc}) + 1}{2^{k-1}}$$

and within running time $t' \leq t + O(q_{H_2} + q_{H_3} + q_{usc})t_p + O(q_{sc})t_{exp}$, where t_p and t_{exp} stand for the time required for a pairing evaluation and an exponentiation in $\mathbb{G}_1$.

Proof The simulator $\mathcal{B}$ receives a random co-Diffie–Hellman instance (g_1^a, g_2^b). It uses $\mathcal{F}$ as a subroutine to solve that instance and plays the role of $\mathcal{F}$'s challenger. The forger $\mathcal{F}$ is initialized with the input $pk_u = g_2^b$ and performs adaptive queries that are handled by $\mathcal{B}$ as explained below (using lists as in the proof of Theorem 5.1):

- H_1 queries: If a hash query on a tuple $m\|u\|pk_S\|pk_R$ is made, then $\mathcal{B}$ checks to see if the latter was previously queried. If so, then $\mathcal{B}$ returns the same value. For a query on a new tuple $m\|u\|pk_S\|pk_R$, $\mathcal{B}$ picks $t \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and defines $H_1(m\|u\|pk_S\|pk_R) = (g_1^a)^t \in \mathbb{G}_1$. The list L_1 is updated accordingly.
- H_2 queries and H_3 queries are dealt with as in Game 4, the proof of Theorem 5.1.
- Signcryption queries: If a signcryption query on a message m and a receiver's public key pk_R is made, $\mathcal{B}$ picks $r \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and computes $u = g_1^r \in \mathbb{G}_1$. If H_1 is already defined on $m\|u\|pk_u\|pk_R$, $\mathcal{B}$ declares "failure" and halts. Otherwise, $\mathcal{B}$ picks $t \overset{R}{\leftarrow} \mathbb{Z}_p^*$, sets $H_1(m\|u\|pk_u\|pk_R) = g_1^t \in \mathbb{G}_1$, and updates L_1 accordingly. It then computes $v = \psi(pk_u)^t \in \mathbb{G}_1$, $h_2 = H_2(u\|\psi(pk_R)^r) \in \{0,1\}^{\ell_1}$, $w = v \oplus h_2$, $\tau = H_3(u\|v\|\psi(pk_R)^r) \in \mathcal{K}$, and $z = \mathtt{Enc}_\tau(m) \in \mathcal{C}$. The ciphertext (u, w, z) is then returned to $\mathcal{F}$.
- Unsigncryption queries are handled exactly as in the proof of Theorem 5.1.

At the end of the game, $\mathcal{F}$ produces a ciphertext (u^*, w^*, z^*) and a recipient's public/private key pair (sk_R^*, pk_R^*). At that moment, $\mathcal{B}$ can unsigncrypt the ciphertext using sk_R^* and, if the ciphertext is a valid forgery for the sender's public key pk_u, $\mathcal{B}$ can extract the message m^* and the signature v^*. If the hash value $H_1(m^*\|u^*\|pk_u\|pk_R^*)$ was not explicitly defined by a query to the H_1 oracle during the simulation, then $\mathcal{B}$ reports "failure" and stops. Otherwise, $\mathcal{B}$ can extract v^* and the hash value $H_1(m^*\|u^*\|pk_u\|pk_R^*)$ must have been defined to be $(g_1^a)^{t^*}$, for some known $t^* \in \mathbb{Z}_p^*$. This implies that v^* must be equal to $(g_1^{ab})^{t^*}$, which yields the co-Diffie–Hellman value.

It is easy to see that the probability for $\mathcal{B}$ to fail in answering a signcryption query is not greater than $q_{sc}(q_{H_1} + q_{sc} + q_{usc})/p \leq q_{sc}(q_{H_1} + q_{sc} + q_{usc})/2^{k-1}$ (since at each signcryption query, there is at most $q_{H_1} + q_{sc} + q_{usc}$ elements in L_1). The probability that $\mathcal{F}$ succeeds without explicitly making the $H_1(m^*\|u^*\|pk_u\|pk_R^*)$ query can be bounded by considering the following three possibilities:

- If the value of $H_1(m^*\|u^*\|pk_u\|pk_R^*)$ is undefined by the simulation, then the probability that $\mathcal{F}$ wins is bounded by $1/2^{k-1}$ (since $\mathcal{F}$ must output a valid ciphertext).
- If the value of $H_1(m^*\|u^*\|pk_u\|pk_R^*)$ was defined by the signcryption oracle for some query on a message m and receiver public key pk_R, then we must have $m = m^*, pk_R = pk_R^*$, and that the associated value $u = u^*$. Since both ciphertexts must be valid, we have that $v = v^*$, and so $w = w^*$ and $\tau = \tau^*$. This means that $z = z^*$ by the deterministic nature of the symmetric encryption scheme. Hence, we conclude that $\mathcal{F}$ outputs the result of a signcryption query, which is a contradiction to the fact that $\mathcal{F}$ wins the game.
- If the value of $H_1(m^*\|u^*\|pk_u\|pk_R^*)$ was defined by the unsigncryption oracle, then $\mathcal{B}$ is still able to solve the co-CDH problem as the simulation of the unsigncryption oracle calls the simulated H_1 oracle for all its computations.

Hence, we can conclude that the probability that the algorithm fails is bounded by $q_{sc}(q_{H_1} + q_{sc} + q_{usc})/2^{k-1} + 1/2^{k-1}$ and the result follows. □

A similar theorem to Theorem 5.1 links the anonymity property of the scheme to the co-CDH assumption.

5.6 A Scheme with Short Detachable Signatures

Figure 5.2 describes a signcryption scheme by Libert and Quisquater [124] with a shorter detachable signature. The construction relies on a signature scheme independently proposed by Zhang et al. [202] and Boneh-Boyen [42]. In the latter work, this scheme was shown to efficiently produce 160-bit signatures without requiring the use of a special hash function mapping messages to be signed onto an elliptic curve subgroup, unlike the original BLS short signature proposed in [47, 48]. In [42], it was also shown that this scheme has a more efficient security reduction in the random oracle model under the q-strong Diffie–Hellman assumption than the reduction given by Zhang et al. [202] under the q-Diffie–Hellman inversion assumption.

The protocol makes use of a (masked) signature as an ElGamal-like ephemeral key as well as a checksum showing that a message was properly encrypted. The sender first computes an exponent $r \leftarrow \gamma/(h_1(b_m\|m\|pk_S) + sk_S) \in \mathbb{Z}_p^*$ where γ is randomly chosen from $\mathbb{Z}_p^*$, $m \in \{0,1\}^*$ is the message to sign and encrypt, and b_m is a message-dependent bit computed as a function of m and the private key sk_S according to Katz and Wang's proof technique [113]—this helps to achieve tight security reductions without introducing random "salts" in signatures. This exponent r is then used to compute an ephemeral Diffie–Hellman key g_1^r as in the ElGamal cryptosystem [81] and to scramble the secret γ using a hash value of $\psi(pk_R)^r$, while a digest of γ, $\psi(pk_R)^r$, and other elements is used to conceal the message m using a deterministic and one-to-one symmetric encryption scheme.

$\texttt{Setup}(1^k)$
Pick a set of bilinear map groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ whose order is a prime p such that $2^{k-1} < p < 2^k$ and generators $(g_1, g_2) \in \mathbb{G}_1 \times \mathbb{G}_2$ such that $g_1 = \psi(g_2)$ with $g_2 \xleftarrow{R} \mathbb{G}_2$
Choose a one-time deterministic symmetric-key encryption scheme $SE = (\texttt{Enc}, \texttt{Dec})$ with key space $\mathcal{K}$ and ciphertext space $\mathcal{C}$
Pick cryptographic hash functions:
$H' : \{0,1\}^* \to \{0,1\}$
$H_1 : \{0,1\}^* \to \mathbb{Z}_p$
$H_2 : \mathbb{G}_1^3 \to \{0,1\}^{k+1}$
$H_3 : \{0,1\}^k \to \mathcal{K}$
$param \leftarrow (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, SE, H', H_1, H_2, H_3)$
Return $param$

$\texttt{KeyGen}(param)$
$x_U \xleftarrow{R} \mathbb{Z}_p^*$; $y_U \leftarrow g_2^{x_U}$
$sk \leftarrow x_U$; $pk \leftarrow y_U$
Return (sk, pk)

$\texttt{Signcrypt}(param, sk_S, pk_R, m)$
Parse sk_S as (x_S, y_S); Parse pk_R as y_R
If $y_S, y_R \notin \mathbb{G}_2 \setminus \{1\}$ then return $\perp$
$\gamma \xleftarrow{R} \mathbb{Z}_p^*$
$b_m \leftarrow H'(sk_S, m)$
$r \leftarrow \frac{\gamma}{H_1(b_m \| m \| pk_S) + x_S} \bmod p$
$c_1 \leftarrow g_1^r$
$c_2 \leftarrow (\gamma \| b_m) \oplus H_2(c_1 \| y_R \| \psi(y_R)^r)$
$\tau \leftarrow H_3(\gamma \| b_m \| y_R \| \psi(y_R)^r)$
$c_3 \leftarrow \texttt{Enc}_\tau(m)$
$C \leftarrow (c_1, c_2, c_3)$
Return C

$\texttt{Unsigncrypt}(param, pk_S, sk_R, C)$
Parse pk_S as y_S; Parse sk_S as (x_S, y_S)
Parse C as (c_1, c_2, c_3)
If $c_1 \notin \mathbb{G}_1$, $c_2 \notin \{0,1\}^{k+1}$ or $c_3 \notin \mathcal{C}$
Return $\perp$
$(\gamma \| b_m) \leftarrow c_2 \oplus H_2(c_1 \| y_R \| c_1^{x_R})$
If $\gamma \notin \mathbb{Z}_p^*$ then return $\perp$
$\tau \leftarrow H_3(\gamma \| b_m \| y_R \| c_1^{x_R})$;
$m \leftarrow \texttt{Dec}_\tau(c_3)$
$\sigma \leftarrow c_1^{\gamma^{-1}}$
If $e(\sigma, y_S \cdot g_2^{H_1(b_m \| m \| pk_S)}) \neq e(g_1, g_2)$
Return $\perp$
Return (m, b_m, σ)

Fig. 5.2 The SDH-based scheme

The use of a masked signature as a "one-time" Diffie–Hellman key allows the sparing of one exponentiation (actually an elliptic curve scalar multiplication) with respect to a sequential signature/encryption composition.

When computing the second component of the ciphertext, the receiver's public key and the first component (which is an embedded signature as well as a Diffie–Hellman ephemeral key) are hashed together with the "one-time" Diffie–Hellman key $\psi(pk_R)^r$ in order to simplify the security proof.

In order to convince a third party that a recovered message m originates from the sender S, the receiver reveals the detached signature σ, the message m, and the associated bit b_m to the third party who can run the regular signature verification algorithm (i.e., check that $e(g_1, g_2) = e(\sigma, y_S \cdot g_2^{H_1(b_m \| m \| pk_S)})$). The scheme thus provides detachable signatures that cannot be linked to their original ciphertext: the signature is masked by a randomly chosen factor γ and anyone observing a valid message–signature pair can use his/her private key to build a signcryption of that message–signature pair under his/her public key. The scheme thus provides *ciphertext unlinkability* in the sense of [51].

5.6.1 Efficiency

Besides a modular inversion, the sender only computes two exponentiations in $\mathbb{G}_1$. The receiver's workload is dominated by one pairing computation (as $e(g_1, g_2)$ can be included among the common public parameters `param`), two exponentiations in $\mathbb{G}_1$, and one exponentiation in $\mathbb{G}_2$.

The scheme is described in terms of asymmetric pairings and requires the existence of a publicly computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$. It does not require the hashing of arbitrary strings onto cyclic elliptic curve subgroups. Hence, the kind of groups suggested in Sect. 4 of [183] may be employed here as they provide an asymmetric pairing configuration $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with an efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$. It has been reported that hashing onto $\mathbb{G}_2$ may be somewhat slow in such configurations. Such parameters allow the performance of the last step of the `Unsigncrypt` algorithm at a reasonable speed using the specific techniques for ordinary curves given by Barreto et al. [19].

Note that the two exponentiations that are the bulk of the sender's workload can be computed off-line (i.e., before knowing the message to be sent). Indeed, in an off-line phase, the sender can pick a random $r \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$, compute $c_1 \leftarrow g_1^r$ and $\omega \leftarrow \psi(pk_R)^r$, store them in memory, and, then, once the message m is known, compute $\gamma \leftarrow r(H_1(b_m \| m \| pk_S) + sk_S) \bmod p$, $c_2 \leftarrow (\gamma \| b_m) \oplus H_2(c_1 \| pk_R \| \omega)$ and $c_3 \leftarrow \mathtt{Enc}_{H_3(\gamma \| b_m \| y_R \| \omega)}(m)$. In this case, care must be taken not to re-use the same r to sign and encrypt distinct messages because this would expose the private key.

From a bandwidth point of view, the scheme allows receivers to extract short signatures from a ciphertext when they wish to convince a judge of the sender's authorship of the message (e.g., signatures of length 256 bits, using the pairing-friendly groups suggested by Barreto and Naehrig [20]).

5.6.2 Anonymous Communications

Like the co-CDH-based scheme of Fig. 5.1, this scheme is meant to provide anonymous communications, where ciphertexts do not reveal the identity of the sender and the receiver. In such a situation, it may be the case that the receiver himself/herself does not know who the sender is upon receiving a ciphertext. We nevertheless remark that the sender's public key is only needed in the final step of the unsigncryption algorithm. A simple solution to the above problem is to have the sender append a short string ID_S that identifies him/her (or even his public key) to the message that is being signcrypted. The receiver then has to perform an online lookup in a public repository to fetch the appropriate public key pk_S.

The syntax of the unsigncryption algorithm is then slightly modified as this algorithm does not take the sender's public key pk_S as input any longer. A similar modification can be made to the scheme of Fig. 5.1.

5.6.3 Security

The original version of the scheme [124] was shown by Tan [188] to be vulnerable to a chosen ciphertext attack taking advantage of a key substitution attack on the underlying signature scheme [42, 202]. However, protecting the scheme against the attack of Tan [188] is rather straightforward using a standard countermeasure to immunize signature schemes from key substitution attacks: it suffices to hash the signer's public key along with the message to be signed. The resulting scheme is even more efficient than the improvement of Ma [128] (subsequently also broken by Tan [189]) as it allows for shorter detachable signatures.

The message confidentiality and existential signature unforgeability, respectively, rely (in the random oracle model) on the intractability of the q-Diffie–Hellman Inversion and q-strong Diffie–Hellman assumptions.

For convenience, the message confidentiality is proved under an equivalent formulation of the q-DHI assumption, which we call the $(q+1)$-exponent problem in $(\mathbb{G}_1, \mathbb{G}_2)$. This consists of computing $g_1^{(x^{q+1})} \in \mathbb{G}_1$ given $(g_1, g_2, g_2^x, \ldots, g_2^{(x^q)})$. A proof of the following result can be found in [41] but we give it for completeness.

Lemma 5.1 *The q-Diffie–Hellman Inversion problem can be formulated as the problem of computing $g_1^{(x^{q+1})}$ on input of $(g_1, g_2, g_2^x, g_2^{(x^2)}, \ldots, g_2^{(x^q)}) \in \mathbb{G}^{q+1}$.*

Proof Given a sequence of elements $(g_1, g_2, g_2^x, \ldots, g_2^{(x^q)})$, where x is uniformly chosen in $\mathbb{Z}_p^*$, for which $g_1^{(x^{q+1})}$ should be found, one can easily construct a q-DHI instance $(y_1, y_2, y_2^A, y_2^{(A^2)}, \ldots, y_2^{(A^q)})$ by setting

$$y_1 = \psi(y_2) \qquad y_2 = g_2^{(x^q)} \qquad y_2^A = g_2^{(x^{q-1})} \qquad \ldots \qquad y_2^{(A^q)} = g_2$$

This implicitly defines the value $A = 1/x$. Any algorithm that computes the q-DHI solution $y_1^{1/A}$ thus reveals the value $g_1^{(x^{q+1})}$. □

Theorem 5.3 *The scheme is FSO/FUO-IND-CCA2 secure in the random oracle model assuming that the q-DHI problem is intractable and that the symmetric encryption scheme is IND-CPA secure. For any adversary $\mathcal{A}$ running in time $t_{\mathcal{A}}$ and making at most q_{sc} signcryption queries, q_{usc} unsigncryption queries, q_{H_i} queries to random oracles H_i $(i = 1, 2, 3)$, and $q_{H'}$ queries to H', we have*

$$\mathrm{Adv}_{\mathcal{A}}(t_{\mathcal{A}}, k) \leq \frac{q_{usc}}{2^{k-2}} + \frac{(q_{sc} + q_{usc})(q_{sc} + q_{usc} + q_{H_3}) + q}{2^{k-1}} + \mathrm{Adv}_{\mathcal{B}}^{q\text{-}DHI}(t', k) + \mathrm{Adv}_{\mathcal{B}}^{ind\text{-}cpa\text{-}sym}(t', |\mathcal{K}|)$$

where $\mathrm{Adv}_{\mathcal{B}}^{ind\text{-}cpa\text{-}sym}(t', |\mathcal{K}|)$ is defined as in Theorem 5.1 while $\mathrm{Adv}_{\mathcal{B}}^{q\text{-}DHI}(t', k)$ stands for the maximal probability of solving the q-DHI problem within running

time $t' \leq t_{\mathcal{A}} + O(q_{H_2} + q_{H_3} q_{usc}) t_p + O(q^2 + q_{H'} + q_{sc} + q_{usc}) t_{exp}$ *when the security parameter is* k *and* t_p *and* t_{exp} *denote the time complexity of a pairing evaluation and that of an exponentiation in* $\mathbb{G}_2$*, respectively.*

Proof The proof proceeds with a sequence of games. The first game is the real attack game, while in the final game the adversary has no better advantage than a passive adversary against the symmetric encryption scheme (`Enc`, `Dec`). Throughout the sequence, we call S_i the event that the adversary $\mathcal{A}$ wins (by correctly guessing the bit $d \in \{0, 1\}$ chosen by the challenger $\mathcal{B}$ in the challenge phase) in Game i.

Game 1 is the real attack game. The adversary $\mathcal{A}$ is provided with a random generator $h \stackrel{R}{\leftarrow} \mathbb{G}_2$, its image $g = \psi(h) \in \mathbb{G}_1$, as well as a receiver's public key $pk_U = X = h^x$. Throughout the game, the simulator $\mathcal{B}$ uses the private key $sk_U = x$ to answer signcryption and unsigncryption queries. Random oracle queries are answered by outputting random values in the appropriate range. This is done consistently in that, if an oracle is queried twice on the same input, the same output is returned by $\mathcal{B}$. To keep track of those queries, lists L', L_1, L_2, and L_3 are used to bookkeep all inputs and the matching outputs for oracles H', H_1, H_2, H_3. In the middle point of the game, $\mathcal{A}$ outputs a pair of messages (m_0, m_1) and a sender's private key $sk_S^* = x_S^*$. To generate the challenge ciphertext (c_1^*, c_2^*, c_3^*), the simulator $\mathcal{B}$ flips a fair binary coin $d \stackrel{R}{\leftarrow} \{0, 1\}$, randomly chooses $\gamma^* \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$, and generates (c_1^*, c_2^*, c_3^*) as $c_1^* = g^{r^*}$, $c_2^* = (\gamma^* \| b_m^*) \oplus H_2(c_1^* \| X \| \psi(X)^{r^*})$, and $c_3^* = \mathtt{Enc}_{\tau^*}(m_d)$ where $r^* = \gamma^* / (x_S^* + H_1(b_m^* \| m_d \| pk_S^*))$, $\tau^* = H_3(\gamma^* \| b_m^* \| X \| \psi(X)^{r^*}) \in \mathcal{K}$, and the bit $b_m^* \in \{0, 1\}$ is determined as $b_m^* = H'(x_S^*, m_d)$. In this game, the adversary has advantage $|\Pr[S_1] - 1/2|$.

Game 2 modifies the generation of the public generator $h \in \mathbb{G}_2$ and the public key X, namely, $\mathcal{B}$ defines values $(g_1, g_2, g_2^x, g_2^{(x^2)}, \ldots, g_2^{(x^q)}) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$ for a randomly chosen $x \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$. To generate $h \in \mathbb{G}_2$, $g = \psi(h) \in \mathbb{G}_1$ and a public key $X = h^x \in \mathbb{G}_2$, $\mathcal{B}$ picks $w_1, \ldots, w_{q-1} \stackrel{R}{\leftarrow} \mathbb{Z}_p$, expands the polynomial $f(z) = \prod_{i=1}^{q-1} (z + w_i) = \sum_{i=0}^{q-1} c_i z^i$, and uses it to obtain a random generator $h \in \mathbb{G}_2$ and a public key X. These can be obtained by first computing

$$h' = \prod_{i=0}^{q-1} (g_2^{(x^i)})^{c_i} = g_2^{f(x)} \quad \text{and} \quad X' = \prod_{i=1}^{q} (g_2^{(x^i)})^{c_{i-1}} = g_2^{x f(x)} = h'^x$$

We call F_2 the event that h' is the identity element of $\mathbb{G}$ (i.e., because $f(x) = 0$ when one of the w_i happens to be the exponent x). Since there are $q - 1$ values w_i and x is chosen at random, we have that the probability that $f(x) = 0$ is bounded by $(q - 1)/p$. It is easy to see that Game 1 and Game 2 are identical unless event F_2 occurs. Hence, $|\Pr[S_1] - \Pr[S_2]| \leq \Pr[F_2] \leq (q - 1)/p \leq (q - 1)/2^{k-1}$.

Game 3 changes the game so that part of the challenge ciphertext (c_1^*, c_2^*, c_2^*) is computed at the beginning of the game, namely, $\mathcal{B}$ chooses $a \stackrel{R}{\leftarrow} \mathbb{Z}_p$ at random and sets the first ciphertext component as $c_1^* = g^{(x+a)} = \psi(X) \cdot g^a \in \mathbb{G}_1$, which

implicitly defines the encryption exponent as $r^* = a + x$. When $\mathcal{A}$ chooses messages m_0, m_1 and a sender's private key sk_S^* in the challenge phase, $\mathcal{B}$ computes $\gamma^* = r^*(sk_S^* + H_1(b_m^* \| m_d \| pk_S^*))$ after having determined the message-dependent bit $b_m^* = H'(sk_S^*, m_d)$ and computes parts c_2^* and c_3^* of the ciphertext as specified by the description of the scheme. This change is only conceptual and we have $\Pr[S_2] = \Pr[S_3]$.

Game 4 is the same as game 3 but $\mathcal{B}$ aborts if a pre-challenge unsigncryption query involves a ciphertext (c_1, c_2, c_3) such that $c_1 = c_1^*$. Unless this event, that we call F_4, occurs, Game 3 and Game 4 proceed identically and we have $\Pr[S_3 \wedge \neg F_4] = \Pr[S_4 \wedge \neg F_4]$ and $|\Pr[S_4] - \Pr[S_3]| \leq \Pr[F_4]$. Since c_1^* is independent of $\mathcal{A}$'s view until the challenge phase, we must have $\Pr[F_4] \leq q_{usc}/p \leq q_{usc}/2^{k-1}$.

Game 5 modifies the treatment of random oracle queries and the way to handle signcryption and unsigncryption queries. We note that, for the values $w_i \in \mathbb{Z}_p$ that are roots of the polynomial $f(z)$, $\mathcal{B}$ can compute $q_{sc} = q - 1$ pairs $(w_i, g^{\frac{1}{w_i + x}})$ using only $(g_1, g_2, \ldots, g_2^{x^q})$ (and without using the underlying x). Using the technique of [42], it obtains these pairs $(w_i, g^{\frac{1}{w_i+x}})$ by expanding $f_i(z) = f(z)/(z + w_i) = \prod_{i=0}^{q-2} d_i z^i$ and computing

$$\tilde{g}_i = \prod_{j=0}^{q-2} \psi(g_2^{(x^j)})^{\theta d_j} = g_1^{\theta f_i(x)} = g_1^{\theta \frac{f(x)}{x+w_i}} = g^{\frac{1}{x+w_i}}$$

for $i = 1, \ldots, q-1$. Queries to random oracles H', H_1 and signcryption queries are now processed as follows:

- H' queries: When a query $H'(\alpha, m_i)$ is made, $\mathcal{B}$ checks if a tuple (α, m_i, b_{m_i}) appears in L'. If so, it returns the previously defined $b_{m_i} \in \{0, 1\}$. If no such tuple is found,
  - if $\alpha = x$ (which $\mathcal{B}$ can test by checking if $g_2^x = g_2^\alpha$ if it does not explicitly know x, as will be the case in later games), $\mathcal{B}$ looks up entries of the form $(?, m_i, b_{m_i})$ in L', returns the matching b_{m_i} if such an entry is found, and replaces the entry by (α, m_i, b_{m_i}). If no entry $(?, m_i, b_{m_i})$ is found, $\mathcal{B}$ responds with a random $b_{m_i} \xleftarrow{R} \{0, 1\}$ and stores (x, m_i, b_{m_i}) in L'.
  - if $\alpha \neq x$, $\mathcal{B}$ returns a random $b \xleftarrow{R} \{0, 1\}$ and stores (α, m_i, b) in L'.
- H_1 queries: These queries are indexed by a counter t that is initially set to 1. When a triple $\delta \| m \| X$ (involving the challenge public key X) is submitted in a H_1 query for a new message m, $\mathcal{B}$ looks up L' for a record $(?, m, b_m)$ (where ? is a placeholder for a currently unknown value). If no such record is found, $\mathcal{B}$ picks a random bit $b_m \xleftarrow{R} \{0, 1\}$ and stores $(?, m, b_m)$ in L' so that b_m will be associated with the message m from this point forward. The treatment of the hash query $H_1(\delta \| m \| X)$ then depends on $\delta \in \{0, 1\}$, namely, if $\delta = b_m$, $\mathcal{B}$ returns w_t, stores (δ, m, X, w_t) in L_1, and increments the counter t (in such a way that $\mathcal{B}$

is able to create a valid signature on m). Otherwise (i.e., if $\delta \neq b_m$), $\mathcal{B}$ returns a random $c \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and stores (δ, m, X, c) in L_1. Should H_1 be queried again on the same $\delta \| m \| X$ later on, $\mathcal{B}$ will look up L_1 and output the value that was defined at the first occurrence of the query.

- H_2 queries are now processed using two lists L_2 and L_2'. On input $Y_{1,i} \| Y_{2,i} \| Y_{3,i}$: $\mathcal{B}$ first checks if H_2 was previously queried on the same input and, if so, returns the previously defined value. Otherwise, $\mathcal{B}$ checks if the tuple $(h, Y_{1,i}, Y_{2,i}, Y_{3,i})$ is a valid co-Diffie–Hellman tuple (in our notation, we write $Y_{3,i} = \text{co-DH}_h(Y_{1,i}, Y_{2,i})$) by verifying if

$$e(Y_{1,i}, Y_{2,i}) = e(Y_{3,i}, h).$$

  If it is, $\mathcal{B}$ checks if L_2' contains a record $(Y_{1,i}, Y_{2,i}, ?, \zeta_i)$, for some $\zeta_i \in \{0,1\}^{k+1}$ and where ? is a placeholder for a currently undetermined value. In this case, ζ_i is returned and an entry $(Y_{1,i}, Y_{2,i}, Y_{3,i}, \zeta_i, 1)$ is added in L_2. If no entry of the shape $(Y_{1,i}, Y_{2,i}, ?, \zeta_i)$ is in L_2', $\mathcal{B}$ returns a string $\zeta_i \stackrel{R}{\leftarrow} \{0,1\}^{k+1}$ and inserts $(Y_{1,i}, Y_{2,i}, Y_{3,i}, \zeta_i, 1)$ in L_2. If $(h, Y_{1,i}, Y_{2,i}, Y_{3,i})$ is not a co-Diffie–Hellman tuple, $\mathcal{B}$ returns a random $\zeta_i \stackrel{R}{\leftarrow} \{0,1\}^{k+1}$ and the entry $(Y_{1,i}, Y_{2,i}, Y_{3,i}, \zeta_i, 0)$ is added in L_2.
- H_3 queries: When a query $H_3(\gamma \| b_m \| pk_R \| Y)$ is made, $\mathcal{B}$ looks up the history L_3 of H_3 queries. If it already contains a record $(*, pk_R, \gamma, b_m, Y, \tau)$ for any value of $*$, then $\mathcal{B}$ returns $\tau \in \mathcal{K}$. Otherwise, $\mathcal{B}$ checks all entries of the form $(c_1, pk_R, \gamma, b_m, ?, \tau)$, for some $c_1 \in \mathbb{G}_1$, and tests whether one of them satisfies $Y = \text{co-DH}_h(c_1, pk_R)$. If so, it returns the matching $\tau \in \mathcal{K}$ and replaces the record by $(c_1, pk_R, \gamma, b_m, Y, \tau)$ in L_3. Otherwise, $\mathcal{B}$ returns a random $\tau \stackrel{R}{\leftarrow} \mathcal{K}$ and stores $(?, pk_R, \gamma, b_m, Y, \tau)$ in L_3.
- Signcryption queries on a plaintext m, for an arbitrary receiver's key pk_R: We assume that m was previously submitted in an H_1 query and that the message-dependent bit b_m was previously defined. Since $H_1(b_m \| m \| X)$ was (or will be) defined to be w_j for some $j \in \{1, \ldots, t\}$, $\mathcal{B}$ knows that $\tilde{g}_j = g^{1/(w_j + x)}$ appears as a valid signature on m from $\mathcal{A}$'s view. So, it computes $c_1 = \tilde{g}_j^{\gamma} \in \mathbb{G}_1$ for some $\gamma \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$. It then checks if L_2 contains an entry $(c_1, pk_R, Y_3, \zeta, 1)$ (indicating that $Y_3 = \text{co-DH}_h(c_1, pk_R)$) or if L_2' contains a record of the form $(c_1, pk_R, ?, \zeta)$. If so, $\mathcal{B}$ sets $c_2 = (\gamma \| b_m) \oplus \zeta \in \{0,1\}^{k+1}$. Otherwise, it sets $c_2 \stackrel{R}{\leftarrow} \{0,1\}^{k+1}$ and inserts $\big(c_1, pk_R, ?, (\gamma \| b_m) \oplus c_2\big)$ in L_2'. If L_3 happens to already contain an entry $(*, pk_R, \gamma, b_m, ., \tau)$ comprising this particular γ, $\mathcal{B}$ fails (we call F_5 this event). Otherwise, it picks a random symmetric key $\tau \stackrel{R}{\leftarrow} \mathcal{K}$, sets $c_3 = \texttt{Enc}_\tau(m)$, and stores a record $(c_1, pk_R, \gamma, b_m, ?, \tau)$ in L_3 (in such a way that a subsequent $H_3(\gamma \| b_m \| pk_R \| \text{co-DH}_h(c_1, pk_R))$ obtains the answer τ). The resulting triple (c_1, c_2, c_3) is then returned to $\mathcal{A}$.
- Unsigncryption queries: When $\mathcal{A}$ submits a ciphertext $C = (c_1, c_2, c_3)$ together with a sender's public key pk_S, $\mathcal{B}$ checks whether list L_2 contains the unique entry $(c_1, X, Y, \zeta, 1)$ for some elements $Y \in \mathbb{G}_1$ and $\zeta \in \{0,1\}^{k+1}$ (indicating

that $Y = \text{co-DH}_h(c_1, X)$) or whether L'_2 contains the entry $(c_1, X, ?, \zeta)$ for some $\zeta \in \{0,1\}^{k+1}$:

- If it does, $\mathcal{B}$ obtains $(\gamma \| b_m) = c_2 \oplus \zeta \in \{0,1\}^{k+1}$, $\tau = H_3(\gamma \| b_m \| X \| Y)$ (by simulating H_3) and finally $m = \mathtt{Dec}_\tau(c_3)$. Finally, $\mathcal{B}$ extracts $\sigma = c_1^{1/\gamma}$ and returns the plaintext m and the signature σ if the verification equation $e(\sigma, pk_S \cdot h^{H_1(b_m \| m \| pk_S)}) = e(g,h)$ is satisfied.
- If it does not, $\mathcal{B}$ randomly picks $\zeta \xleftarrow{R} \{0,1\}^{k+1}$, $\tau \xleftarrow{R} \mathcal{K}$, and inserts $(c_1, X, ?, \zeta)$ into the list L'_2 (so that a subsequent H_2 query on $(c_1, X, \text{co-DH}_h(c_1, X))$ will be assigned the output ζ). It computes $(\gamma \| b_m) = c_2 \oplus \zeta \in \{0,1\}^{k+1}$ and aborts in the unlikely event, which we call F'_5, that the obtained γ already appears somewhere in L_3 (since γ is almost uniform in $\mathbb{Z}_p$, event F'_5 only happens with negligible probability). Otherwise, it stores $(c_1, X, \gamma, b_m, ?, \tau)$ in L_3 (in such a way that a subsequent query $H_3(\gamma \| b_m \| X \| \text{co-DH}_h(c_1, X))$ will receive the answer τ. The latter is used to compute $m \leftarrow \mathtt{Dec}_\tau(c_3)$. The signature $\sigma = c_1^{1/\gamma}$ is checked as above. If the verification succeeds, $\mathcal{B}$ returns (m, σ). Otherwise, it outputs $\perp$.

Unless event F_5 or F'_5 occurs at some query, $\mathcal{A}$'s view is not affected by the above modifications. We thus have $|\Pr[S_5] - \Pr[S_4]| \leq \Pr[F_5 \vee F'_5]$. It is easy to see that $\Pr[F_5 \vee F'_5] \leq (q_{sc} + q_{usc})(q_{sc} + q_{usc} + q_{H_3})/2^{k-1}$ since list L_3 never contains more than $q_{sc} + q_{usc} + q_{H_3}$ entries.

Game 6 changes the simulation of the unsigncryption oracle again and adds the following rule. *After* the challenge phase, if $\mathcal{A}$ queries the unsigncryption of a ciphertext (c_1, c_2, c_3) such that $(c_1, c_2) = (c_1^*, c_2^*)$, $\mathcal{B}$ returns $\perp$. We consider two cases:

- If the query is made for the same sender $pk_S = pk_S^*$, we must have $c_3 \neq c_3^*$. It is easy to see that for such a ciphertext the underlying values (γ, b_m) must be the same as the pair (γ^*, b_m^*) of the challenge ciphertext. Moreover, the same symmetric key $\tau^* = H_3(\gamma^* \| b_m^* \| X \| \psi(X)^{r^*})$ must be used to decipher c_3 when executing the unsigncryption operation. Since $c_3 \neq c_3^*$ and given that the symmetric encryption algorithm $(\mathtt{Enc}, \mathtt{Dec})$ is a bijection, the underlying plaintext $m = \mathtt{Dec}_{\tau^*}(c_3)$ must be different from m_d. Therefore, unless we have a collision $H_1(b_m^* \| m \| pk_S^*) = H_1(b_m^* \| m_d \| pk_S^*)$ (which occurs with probability at most $1/|p| < 1/2^{k-1}$ when H_1 is viewed as a random oracle), the underlying $c_1^{*\,1/\gamma^*}$ cannot be a valid signature for m.
- If the query is made for a different sender's public key $pk_S \neq pk_S^*$ (in which case, we may have $c_3 = c_3^*$ or $c_3 \neq c_3^*$), the unsigncryption operation would still reveal the same values $(\gamma, b_m) = (\gamma^*, b_m^*)$ as in the challenge phase and the same symmetric key $\tau^* = H_3(\gamma^* \| b_m^* \| X \| c_1^{*x})$ must be used to decipher c_3. However, if we denote by $m = \mathtt{Dec}_{\tau^*}(c_3)$ the symmetric decryption of c_3 under that symmetric key τ^*, the ciphertext (c_1^*, c_2^*, c_3) would only be accepted in

previous games if $\log_g(pk_S) + H_1(b_m^* \| m \| pk_S) = x_S^* + H_1(b_m^* \| m_d \| pk_S^*)$. In the random oracle model, this only occurs with probability at most $1/p < 1/2^{k-1}$.

Throughout all queries, the probability that the new rule causes the rejection of a ciphertext that would not have been rejected in earlier games is at most $q_{usc}/2^{k-1}$. We thus have $|\Pr[S_6] - \Pr[S_5]| \leq q_{usc}/2^{k-1}$.

Game 7 brings two changes to the simulation and first modifies the generation of the challenge ciphertext $C^* = (c_1^*, c_2^*, c_3^*)$ again. When $\mathcal{A}$ outputs messages m_0, m_1 together with a sender's private key $sk_S^* \in \mathbb{Z}_p^*$ in the challenge phase, $\mathcal{B}$ still computes c_1^* by choosing $a \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and setting $c_1^* = g^{(x+a)} = \psi(X) \cdot g^a \in \mathbb{G}_1$. However, sk_S^* is no longer used to compute c_2^* and neither are the private key $sk_U = x$ and the encryption exponent $r^* = x + a$. Elements (c_2^*, c_3^*) are generated by drawing $c_2^* \stackrel{R}{\leftarrow} \{0,1\}^{k+1}$ at random and computing $c_3^* = \mathtt{Enc}_{\tau^*}(m_d)$, for a random bit $d \stackrel{R}{\leftarrow} \{0,1\}$, using a random key $\tau^* \stackrel{R}{\leftarrow} \mathcal{K}$. The other change is that the simulator $\mathcal{B}$ immediately halts in the event, which we call E, that either of the following situations occurs:

- E.1 $\mathcal{A}$ queries oracle H' on a pair $(x, *)$, where $x = sk_U = \log_g(X)$ is the private key. This can be tested by checking whether $X = h^\alpha$ at each query $H'(\alpha, *)$.
- E.2 $\mathcal{A}$ queries oracle H_2 on an input $(c_1^* \| X \| Y)$ such that $Y = \text{co-DH}_h(c_1^*, X)$.
- E.3 $\mathcal{A}$ queries oracle H_3 on an input $(\gamma \| b_m \| X \| Y)$ such that $Y = \text{co-DH}_h(c_1^*, X)$.

We observe that $\mathcal{B}$ can detect E.1, E.2, and E.3 without knowing the private key $sk_U = x$. Event E.1 directly allows solving a given instance of the q-DHI problem and so do events E.2 and E.3 as we will see.

Let us assume for now that event E does not occur. Since the proof takes place in the random oracle model, without knowing the value $H_2\big(c_1^* \| X \| \text{co-DH}_h(c_1^*, X)\big)$, $\mathcal{A}$ has no information on $c_2^* \oplus H_2\big(c_1^* \| X \| \text{co-DH}_h(c_1^*, X)\big)$ and cannot realize that the challenge ciphertext was not properly generated. Game 7 is then identical to Game 6, which allows writing $\Pr[S_7 \wedge \neg E] = \Pr[S_6 \wedge \neg E]$ and $|\Pr[S_7] - \Pr[S_6]| \leq \Pr[E]$. Moreover, as long as event E.3 does not happen, the key τ^* that is used to compute c_3^* is perfectly independent of $\mathcal{A}$'s view and guessing the bit $d \in \{0,1\}$ boils down to mounting a chosen plaintext attack on the symmetric cipher $(\mathtt{Enc}, \mathtt{Dec})$. Indeed, before Game 6, the only situation where $\mathcal{A}$ could possibly manage to see the result of a symmetric decryption under the key τ^* would be by creating a valid ciphertext of the form (c_1^*, c_2^*, c_3) and such ciphertexts are always rejected by the unsigncryption oracle from Game 6 onwards. We thus have $|\Pr[S_7] - 1/2| = \text{Adv}_{\mathcal{B}}^{\textit{ind-cpa-sym}}(t')$, where t' is a bound on $\mathcal{B}$'s running time (which will be determined below).

We still have to explain how $\mathcal{B}$ can solve a q-DHI instance if event E.2 or event E.3 (as defined above) occurs. When, via H_2 or H_3 queries, $\mathcal{B}$ obtains the co-Diffie–Hellman value $Y = \text{co-DH}_h(c_1^*, X) = g^{x(x+a)} = g_1^{\theta x f(x)(x+a)}$, it expands $f'(z) = f'(z)(z+a) = \sum_{j=0}^{q} f_j z^j \in \mathbb{Z}_p[z]$ and, since $Y = \prod_{j=0}^{q} \psi(g_2^{(x^{j+1})})^{\theta f_j}$, $\mathcal{B}$ can compute

$$g_1^{(x^{q+1})} = \left[Y^{1/\theta} \cdot \prod_{j=0}^{q-1} \psi(g_2^{(x^{j+1})})^{-f_j}\right]^{\frac{1}{f_q}} \in \mathbb{G}_1$$

and solve the $(q+1)$-exponent instance.

From a computational point of view, $\mathcal{B}$'s running time is dominated by $q+2$ multi-exponentiations with q elements that reach an overall cost of $O(q^2)$ exponentiations. Computing $f(z)$ also involves a cost in $O(q^2)$ while computing each $f_i(z)$ implies $O(q)$ operations like the computation of the product $f(z)(z+a)$. When handling H_2 and H_3 queries, $\mathcal{B}$ also has to compute $O(q_{H_2} + q_{H_3} q_{usc})$ pairings. Finally, answering H' queries, signcryption queries, and unsigncryption queries also implies exponentiations.

The probability that event E occurs can thus be bounded by

$$\Pr[E] \leq \mathrm{Adv}_{\mathcal{B}}^{q\text{-}DHI}(t')$$

where $t' \leq t_{\mathcal{A}} + O(q_{H_2} + q_{H_3} q_{usc}) t_p + O(q^2 + q_{H'} + q_{sc} + q_{usc}) t_{exp}$. □

The scheme does not necessarily provide ciphertext unforgeability; however, it can be shown to give signature unforgeability (see Sect. 5.4.2). This means that, while it might be possible for an attacker to produce a new valid ciphertext from a particular sender, it is not possible for an attacker to produce a ciphertext that gives rise to a new message/signature pair. In other words, the resulting signature must have been originally produced by the legitimate sender.

Theorem 5.4 *If an ESUF-CMA adversary $\mathcal{F}$ has non-negligible advantage in the game of Definition 5.6, we can break the q-Strong Diffie–Hellman assumption in the random oracle model. More precisely, for any forger $\mathcal{F}$ running in time t, making q_{H_i} queries to oracles H_i $(i = 1, 2, 3)$, q_{usc} unsigncryption queries, and q_{sc} signcryption queries, there exists an algorithm $\mathcal{B}$ solving the q-SDH problem for $q = q_{H_1}$ such that*

$$\mathrm{Adv}_{\mathcal{F}}(t_{\mathcal{F}}, k) \leq \left(1 + 2 \cdot \left(1 - \frac{q-1}{2^{k-1}}\right)^{-1}\right) \cdot \mathrm{Adv}_{\mathcal{B}}^{q\text{-}SDH}(t', k) + \frac{(q_{sc} + q_{usc})(q_{sc} + q_{usc} + q_{H_3}) + 2q - 1}{2^{k-1}}$$

where $t' \leq t_{\mathcal{F}} + O(q_{H_2} + q_{H_3} q_{usc}) t_p + O(q^2 + q_{H'} + q_{sc} + q_{usc}) t_{exp}$ and t_p and t_{exp} stand for the same quantities as in Theorem 5.3.

Proof The proof consists of a sequence of five games. The first one is the real attack game described by Definition 5.6. In the last game, the simulator will be able to extract a solution to a q-SDH instance from its interaction with the adversary. In each game of the sequence, we call W_i the event that the adversary wins.

Game 1 is the real attack game. The adversary $\mathcal{F}$ is given a random generator $h \xleftarrow{R} \mathbb{G}_2$, $g = \psi(h) \in \mathbb{G}_1$ and a sender's public key $pk_U = X = h^x$. Throughout the

game, the simulator $\mathcal{B}$ uses the private key $sk_U = x$ to answer signcryption and unsigncryption queries. Random oracle queries are handled in the standard way, by returning random values in the appropriate set and producing the same output if the same input is queried several times. The forger $\mathcal{F}$ eventually outputs a ciphertext $C^* = (c_1^*, c_2^*, c_3^*)$ and a key pair (sk_R^*, pk_R^*). He/she wins if the unsigncryption of C^* under sk_R^* and pk_u is a valid triple (m^*, b^*, σ^*) with respect to pk_u and if this triple was not trivially obtained by querying the signcryption oracle (as described in Step 3 of Definition 5.6). The advantage of $\mathcal{F}$ is defined as $Adv_{\mathcal{F}} = \Pr[W_1]$.

Game 2 modifies the generation of the generator h and the sender's public key X which are now calculated in the same way as in Game 2 in the previous proof. The generators $h \in \mathbb{G}_2$, $g = \psi(h) \in \mathbb{G}_1$, that are given to the forger $\mathcal{F}$ as a part of the output of the common key generation algorithm, and the public key $X = h^x$ are generated as in Game 2 in the proof of Theorem 5.3, namely, $\mathcal{B}$ chooses a random polynomial $f(z) = \prod_{i=1}^{q-1}(z+w_i)$, with $w_1, \ldots, w_{q-1} \stackrel{R}{\leftarrow} \mathbb{Z}_p$. Given values $(g_1, g_2, g_2^x, \ldots, g_2^{(x^q)})$, it computes $h' = g_2^{f(x)}$, $X' = h'^x$ and finally $h = h'^\theta$, $X = X'^\theta$ where $\theta \stackrel{R}{\leftarrow} \mathbb{Z}_p$. If $f(x)$ happens to be zero, $\mathcal{B}$ can directly solve the problem as in the proof of Theorem 5.3. At the beginning of the game, $\mathcal{B}$ hands the public key $pk_u = X = h^x$ to $\mathcal{F}$. It is easy to see that Game 1 and Game 2 are identical unless the polynomial $f(z)$ accidentally cancels in x. This event, which we call F_2, occurs with probability bounded by $(q-1)/p$. Hence, we have that $|\Pr[W_1] - \Pr[W_2]| \leq \Pr[F_2] \leq (q-1)/2^{k-1}$.

Game 3 modifies the treatment of all queries and handles them exactly in the same way as in game 5 in the proof of Theorem 5.3. In this game, the private key $sk_u = x$ is not explicitly used to answer queries. In particular, signcryption queries can be dealt with without it since, after having calculated h and the public key, $\mathcal{B}$ knows $q-1$ pairs of the form $(w_i, g^{\frac{1}{w_i+x}})$. We denote by F_3 and F_3' the events that $\mathcal{B}$ fails when answering a signcryption and an unsigncryption query, respectively (i.e., the same events as F_5 and F_5' in the proof of Theorem 5.3). Unless either F_3 or F_3' occurs at some query, $\mathcal{A}$'s view will not be affected by these changes and we have $|\Pr[W_3] - \Pr[W_2]| \leq \Pr[F_3 \vee F_3']$. As in the proof of Theorem 5.3, we have $\Pr[F_3 \vee F_3'] \leq (q_{sc} + q_{usc})(q_{sc} + q_{usc} + q_{H_3})/p$ since list L_3 always contains at most $q_{sc} + q_{usc} + q_{H_3}$ records.

Game 4 introduces a failure event F_4 which is the same as event E.1 in the proof of Theorem 5.3 (namely $\mathcal{A}$ queries oracle H' on a pair $(x, *)$). Clearly, if event F_4 occurs, $\mathcal{B}$ can detect it and directly solve the q-SDH instance. We thus find that $\Pr[F_4] \leq \mathrm{Adv}^{q\text{-}SDH}(\mathcal{B})$ and $|\Pr[W_4] - \Pr[W_3]| \leq \mathrm{Adv}^{q\text{-}SDH}(\mathcal{B})$.

Game 5 raises a new failure event F_5 and makes $\mathcal{B}$ abort when it occurs. When the adversary $\mathcal{F}$ halts and outputs a ciphertext $C^* = (c_1^*, c_2^*, c_3^*)$ and an arbitrary recipient's key pair $(sk_R^*, pk_R^* = h^{sk_R^*})$, $\mathcal{B}$ looks up the values $H_2(c_1^* \| pk_R^* \| c_1^{*sk_R^*})$ and $\tau^* = H_3(\gamma^* \| b_{m^*} \| pk_R^* \| c_1^{*sk_R^*})$, where $(\gamma^* \| b_{m^*}) = c_2^* \oplus H_2(c_1^* \| pk_R^* \| c_1^{*sk_R^*})$, in the history of random oracle queries (and simulates random oracles for itself if necessary). We define F_5 to be the event that the hash value $H_1(b_{m^*} \| m^* \| X)$ was never defined by the simulation. Since H_1 is modeled as a random oracle, the probability

that $\mathcal{A}$ wins without having forced the simulation to define $H_1(b_{m^*}\|m^*\|X)$ is at most $1/p < 1/2^{k-1}$. We thus have $|\Pr[W_5] - \Pr[W_4]| \leq \Pr[F_5] \leq 1/2^{k-1}$.

We will now give an upper bound for $\Pr[W_5]$. When $\mathcal{F}$ outputs a fake signature embedded in a ciphertext $C^* = (c_1^*, c_2^*, c_3^*)$ and the key pair $(sk_R^*, pk_R^* = h^{sk_R^*})$, $\mathcal{B}$ can recover the fake triple $\left(m^*, b_{m^*}, \sigma^* = g^{1/(H_1(b_{m^*}\|m^*\|X)+x)}\right)$ that must be contained in C^* by successively computing $(\gamma^*\|b_{m^*}) = c_2 \oplus H_2(c_1^*\|pk_R^*\|c_1^{*sk_R^*})$, $\tau^* = H_3(\gamma^*\|b_{m^*}\|pk_R^*\|c_1^{*sk_R^*})$, $m^* = \mathtt{Dec}_{\tau^*}(c_3^*)$, and $\sigma^* = c_1^{*1/\gamma^*}$.

Let us first assume that m^* was never the input of a signcryption query. Then, with probability $1/2$, b_{m^*} differs from the message-dependent bit $b_{m^*}^*$ (that indicates how m^* should be signed with the private key corresponding to $pk_u = X$ in the underlying signature scheme) since the latter is independent of $\mathcal{F}$'s view. Recall that we have $\sigma^* = g^{1/(x+h_1^*)} = g_1^{\theta f(x)/(x+h_1^*)}$, where $h_1^* = H_1(b_{m^*}\|m^*\|X)$. As long as $b_{m^*} \neq b_{m^*}^*$, we have $h_1^* \notin \{w_1, \ldots, w_{q-1}\}$ with probability at least $1 - (q-1)/p$ (since H_1 is a random oracle) and $(x + h_1^*)$ does not divide $f(x)$. In this case, a q-SDH pair (h_1^*, g^*) can then be extracted by expanding $f(z)/(z + h_1^*)$ into

$$\frac{\gamma_{-1}}{z + h_1^*} + \sum_{i=0}^{q-2} \gamma_i z^i$$

and computing $g^* = \left[\sigma^{*1/\theta} \cdot \prod_{i=0}^{q-2} \psi(g_2^{(x^i)})^{-\gamma_i}\right]^{\frac{1}{\gamma_{-1}}}$.

In the event that m^* was the input of some signcryption query, the latter was answered by generating the underlying signature as $\sigma = g^{1/(x+H_1(b_{m^*}^*\|m^*\|X))}$ using the message-dependent bit $b_{m^*}^*$. It comes that the bit b_{m^*} must necessarily be different from $b_{m^*}^*$ as, according to Definition 5.6, the triple (m^*, b_{m^*}, σ^*) would not be a forgery otherwise. Since $b_{m^*} \neq b_{m^*}^*$, $\mathcal{B}$ can extract an SDH pair as in the first case. In either situation, we find that $\Pr[W_5] \leq 2 \cdot \mathrm{Adv}^{q\text{-}SDH}(\mathcal{B}) + (q-1)/2^{k-1}$. □

In comparison with other schemes, the disadvantage of this one is a security reduction relying on somewhat strong assumptions as the value q must be fairly large.