Certificateless Signcryption

M. Barbosa[1] and P. Farshim[2]

[1] Departamento de Informática, Universidade do Minho,
Campus de Gualtar, 4710-057 Braga, Portugal.
mbb@di.uminho.pt

[2] Department of Computer Science, University of Bristol,
Merchant Venturers Building, Woodland Road,
Bristol BS8 1UB, United Kingdom.
farshim@cs.bris.ac.uk

Abstract. Certificateless cryptography achieves the best of the two worlds: it inherits from identity-based techniques a solution to the certificate management problem in public-key encryption, whilst removing the secret key escrow functionality inherent to the identity-based setting. Signcryption schemes achieve confidentiality and authentication simultaneously by combining public-key encryption and digital signatures, offering better overall performance and security. In this paper, we introduce the notion of certificateless signcryption and present an efficient construction which guarantees security under insider attacks, and therefore provides forward secrecy and non-repudiation. The scheme is shown to be secure using random oracles under a variant of the bilinear Diffie-Hellman assumption.

Keywords. Certificateless Cryptogrpahy, Signcryption, Insider Security, Non-Repudiation, Forward Secrecy, Randomness Reuse.

1 Introduction

Certificateless cryptography achieves the best of two worlds. It inherits from identity-based techniques a solution to the certificate management problem in public-key encryption, yet it eliminates the need for a trusted authority with key escrow capabilities.

In identity-based cryptography [18] an arbitrary bit-string representing a user's identity can be used as the encryption or verification public key. This means that public key certificates are not required. This feature, however, comes at the cost of introducing an all-powerful secret key issuing authority, which authenticates users and provides secret keys through a secure channel. The problem is that, not only must this authority be trusted to authenticate the users, but also not to take advantage of possessing the user's secret keys. This is known as the key escrow property of identity-based cryptosystems and it can be presented as a feature or a security problem, depending on the application scenario.

In certificateless cryptography key escrow is seen as an undesirable property, and user encryption and verification keys contain both a user identity and an unauthenticated public key. Similarly, user secret keys are constructed from two partial secrets: one coming from an identity-based trusted authority called the Key Generation Centre (KGC) and another one generated by the user. Certificateless security models capture scenarios where the attacker can be a system user or the KGC itself. To account for the fact that user public keys are not authenticated, attackers are allowed to replace users' public keys to attempt impersonation.

Since certificateless cryptography was introduced by Al-Ryiami and Paterson [1], numerous certificateless encryption and signature schemes, and variants thereof, have been proposed. However, the equivalent of public-key signcryption has not been considered in the certificateless setting.

Signcryption is a cryptographic primitive that captures a common practical scenario where one simultaneously requires confidentiality and non-repudiation of transmitted data. Ideally, this should allow for improvements in the overall security and efficiency of the resulting cryptosystems. The security goals associated with signcryption are stronger than those provided by authenticated encryption, where data authenticity suffices. For this reason the security model for signcryption should capture *insider attacks* where a dishonest receiver, should not be able to forge a valid signcryption originating from another user. In less common scenarios one may also require forward secrecy, where a message sent by a legitimate user, cannot be decrypted even by an adversary which later is able to get hold of the sender's secret key. This primitive has been extensively studied in the public-key and identity-based settings where many efficient and secure schemes have been proposed.

Our Contribution: In this paper we introduce the notion of certificateless signcryption, define appropriate security models, and propose an efficient scheme which is provably secure against insider attacks in the random oracle model [4]. The scheme presents stronger security properties than one might expect from its internal building blocks: by sharing randomness between encryption and signature modules not only we gain extra savings on computational and bandwidth load, but also we obtain strong insider security guarantees. As an additional contribution, we identify a problem in using Coron's technique for tighter security reductions in public key signature proofs which is specific to the certificateless setting and propose a technique to overcome it.

Paper Organisation: Section 2 describes related work available in the literature. In Section 3 we revise some background on bilinear groups, pairings and associated hard problems. Then, in Section 4 we introduce certificateless signcryption and define suitable security models for this primitive. In Section 5 we propose an efficient scheme and provide a security argument for it. Finally, in Section 6 we discuss the relevance and implications of our results.

2 Related Work

Signcryption in the public-key setting was introduced by Zheng in [20]. A systematic study of the properties of the signcryption schemes resulting from the black-box composition of encryption and signature schemes was later presented by An et al. [2]. Our certificateless signcryption scheme is based on the Encrypt-then-Sign paradigm discussed in [2]. More efficient generic constructions of signcryption schemes based on Tag-KEMs were introduced by Bjørstad et al. [5].

Malone-Lee in [17] formulated signcryption in the identity-based setting. This work was extended by Boyen [6] who proposed a more structured primitive definition and comprehensive security notions which captured different security guarantees that could be achieved by an identity-based signcryption scheme. Other efficient constructions were subsequently proposed by Libert et al. in [14] and Chen et al. in [7].

Together with the introduction of certificateless cryptography in [1], Al-Ryiami and Paterson proposed an encryption scheme and briefly outlined a signature scheme and extensions of other public-key primitives to the certificateless scenario. The encryption scheme was proven secure under a non-realistic security model where an adversary may replace users' public keys and demand decryptions without providing the corresponding secret key. In [15], Libert et al. generalised the encryption scheme in [1] by proposing a generic construction of certificateless encryption schemes from identity-based and public-key encryption in the random oracle model. Recently, Dent et al. [11] have proposed certificateless encryption schemes which provide the same security guarantees in the standard model. A good survey of certificateless public-key encryption schemes and security models can be found in [10].

The certificateless signature scheme proposed in [1], which lacked a security proof, was later found to be vulnerable to key replacement attacks [13]. Other certificateless signature schemes have been proposed in recent years. We refer to the work of Zhang et al. [19], which proposes a certificateless signature scheme closely based on the identity-based signature scheme of Libert et al. in [16]. We base our certificateless signcryption construction on this signature scheme, but it is worthwhile to mention that the proof of security presented in [19] for this scheme is flawed. Despite this, the scheme is secure, but in a slightly weaker security model later proposed informally in [12] together with a generic construction for certificateless signatures. We will discuss this issue further in Section 6.

To the best of our knowledge, the concept of certificateless signcryption has not been previously addressed in literature. However, a closely related construction, which offers authenticated certificateless encryption functionality was proposed in [8][3]. The difference between authenticated encryption and signcryption is a subtle but significant one: insider attacks are not considered in the unforge-

[3] Another paper titled "Authenticated Certificateless Public Key Encryption" by Lee and Lee has appeared on IACR ePrint archive in 2004. However, the security models in this scheme do not capture the certificateless setting.

ability game, which means that non-repudiation is not guaranteed. The security models adopted in [8] are significantly weaker than the ones considered in this paper, as they do not consider decryption on replaced public keys and impose unreasonable restrictions on the adversary's adaptive behaviour. In particular the forward-secrecy argument presented for the scheme in [8] considers only chosen-plaintext attacks. We base our signcryption scheme on this certificateless encryption scheme, but our Encrypt-then-Sign construction with randomness reuse permits obtaining improved security properties.

3 Bilinear Groups

Our scheme relies on bilinear groups and we briefly recall their definition below. We restrict our attention to the symmetric case where $G_1 \cong G_2$ and we may consider a common generator P for them.

Definition 1. *A bilinear group description Γ is a tuple $(p, G_1, G_2, G_T, e, P_1, P_2)$ where:*

- *G_1, G_2 and G_{T} are groups of order p with efficiently computable group laws.*
- *$e : G_1 \times G_2 \to G_T$ is an efficiently computable non-degenerate bilinear map.*
- *P_1 and P_2 are generators of G_1 and G_2 respectively.*

In practice G_1 and G_2 will be related to the (additive) group of points on an elliptic curve and G_T will be a subgroup of the (multiplicative) group of a finite field. Hence, we use additive notation for G_1 and G_2 and multiplicative notation for G_T.

Not only do we wish the discrete logarithm problem in the three groups to be intractable, but we also require the following problem to be hard too.

Definition 2. *Given a bilinear group description Γ, we say the gap bilinear Diffie-Hellman (GBDH) assumption holds if the advantage of any probabilistic polynomial time adversary as defined below is negligible.*

$$\mathrm{Adv}_{\Gamma}^{\mathtt{GBDH}}(A, q_{\mathtt{DBDH}}) := \Pr[T = e(P,P)^{abc} | a, b, c \leftarrow \mathbb{Z}_p; T \leftarrow A^{\mathcal{O}_\Gamma}(\Gamma, aP, bP, cP)].$$

Here $\mathcal{O}_\Gamma$ denotes a decision bilinear Diffie-Hellman oracle which on input a four-tuple (aP, bP, cP, T) outputs 1 if $T = e(P,P)^{abc}$ and 0 otherwise. By $q_{\mathtt{DBDH}}$ we denote the maximum number of queries that A asks its decision oracle.

The following weaker assumption is implied by the above.

Definition 3. *Given a bilinear group description Γ, we say the computational Diffie-Hellman assumption in the presence of a decision bilinear Diffie-Hellman oracle (GDH′) holds in G_1 if the advantage of any PPT adversary as defined below is negligible.*

$$\mathrm{Adv}_{\Gamma}^{\mathtt{GDH}'}(A, q_{\mathtt{DBDH}}) := \Pr[Q = abP | a, b \leftarrow \mathbb{Z}_p; Q \leftarrow A^{\mathcal{O}_\Gamma}(\Gamma, aP, bP)].$$

Here $\mathcal{O}_\Gamma$ and $q_{\mathtt{DBDH}}$ are as in the above definition.

This assumption in turn implies:

Definition 4. *Given a bilinear group description Γ, we say the computational Diffie-Hellman (CDH) assumption holds in G_1 if the advantage of any PPT adversary as defined below is negligible.*

$$\mathrm{Adv}_{\Gamma}^{\mathtt{CDH}}(A) := \Pr[Q = abP | a, b \leftarrow \mathbb{Z}_p; Q \leftarrow A(\Gamma, aP, bP)].$$

4 Certificateless Signcryption

A certificateless signcryption scheme is defined by a six-tuple of probabilistic polynomial-time algorithms. Four of these algorithms, the ones corresponding to key management operations, are identical to those defined for certificateless encryption:

1. $\mathtt{Setup}(1^\kappa)$. This is a global set-up algorithm, which takes as input the security parameter 1^κ and returns the KGC's secret key $\mathtt{Msk}$ and global parameters $\mathtt{params}$ including a master public key $\mathtt{Mpk}$ and descriptions of message space $\mathbb{M}_{\mathtt{CLSC}}(\mathtt{params})$, ciphertext space $\mathbb{C}_{\mathtt{CLSC}}(\mathtt{params})$ and randomness space $\mathbb{R}_{\mathtt{CLSC}}(\mathtt{params})$. This algorithm is executed by the KGC, which publishes $\mathtt{params}$.
2. $\mathtt{Extract\text{-}Partial\text{-}Private\text{-}Key}(\mathtt{ID}, \mathtt{Msk}, \mathtt{params})$. An algorithm which takes as input $\mathtt{Msk}$, $\mathtt{params}$ and an identifier string $\mathtt{ID} \in \{0,1\}^*$ representing a user's identity, and returns a partial secret key D. This algorithm is run by the KGC, after verifying the user's identity.
3. $\mathtt{Generate\text{-}User\text{-}Keys}(\mathtt{ID}, \mathtt{params})$ An algorithm which takes an identity and the public parameters and outputs a secret value x and a public key $\mathtt{PK}$. This algorithm is run by a user to obtain a public key and a secret value which can be used to construct a full private key. The public key is published without certification.
4. $\mathtt{Set\text{-}Private\text{-}Key}(D, x, \mathtt{params})$. A deterministic algorithm which takes as input a partial secret key D and a secret value x and returns the full private key S. Again, this algorithm is run by a user to construct the full private key.

The signcryption and de-signcryption algorithms are as follows:

5. $\mathtt{Sc}(\mathtt{m}, S_S, \mathtt{ID}_S, \mathtt{PK}_S, \mathtt{ID}_R, \mathtt{PK}_R, \mathtt{params}; r)$. This is the signcryption algorithm. On input of a message $\mathtt{m} \in \mathbb{M}_{\mathtt{CLSC}}(\mathtt{params})$, sender's full private key S_S, identity $\mathtt{ID}_S$ and public key $\mathtt{PK}_S$, the receiver's identity $\mathtt{ID}_R$ and public key $\mathtt{PK}_R$, the global parameters $\mathtt{params}$ and possibly some randomness $r \in \mathbb{R}_{\mathtt{CLSC}}(\mathtt{params})$, this algorithm outputs a ciphertext $\mathtt{c} \in \mathbb{C}_{\mathtt{CLSC}}(\mathtt{params})$ or an error symbol $\perp$.
6. $\mathtt{Dsc}(\mathtt{c}, S_R, \mathtt{ID}_R, \mathtt{PK}_R, \mathtt{ID}_S, \mathtt{PK}_S, \mathtt{params})$. The deterministic de-signcryption algorithm. On input of a ciphertext $\mathtt{c}$, receiver's full private key R_S, identity $\mathtt{ID}_R$ and public key $\mathtt{PK}_R$, the sender's identity $\mathtt{ID}_S$ and public key $\mathtt{PK}_S$ and the global parameters $\mathtt{params}$, this algorithm outputs a plaintext $\mathtt{m}$ or a failure symbol $\perp$.

Note that, as for identity-based signcryption, and unlike standard public-key signcryption, certificateless signcryption is intrinsically multi-user: there is no distinction between the simpler one-to-one scenario and a full-fledged multi-user scenario, as user's identities are explicitly associated with ciphertexts.

In defining the security of certificateless signcryption we follow the common approach in literature [7, 6] where one does not consider attacks targeting signcryptions where the sender and receiver identities are the same. In particular we disallow such queries to relevant oracles and do not accept this type of signcryption as a valid forgery. We discuss this issue further in Section 6.

The security game that captures the confidentiality requirement is based on the concept of ciphertext indistinguishability, and is defined as follows:

IND-atk-x
1. $(\mathtt{Msk}, \mathtt{params}) \leftarrow \mathtt{Setup}(1^\kappa)$
2. $(\mathtt{m}_0, \mathtt{m}_1, \mathtt{ID}_S^*, \mathtt{ID}_R^*, \mathtt{st}) \leftarrow A_1^{\mathcal{O}_1}(\mathtt{params}, \mathtt{aux})$
3. $b \leftarrow \{0, 1\}$
4. $\mathtt{c}^* \leftarrow \mathtt{Sc}(\mathtt{m}_b, S_S^*, \mathtt{ID}_S^*, \mathtt{PK}_S^*, \mathtt{ID}_R^*, \mathtt{PK}_R^*, \mathtt{params})$
5. $b' \leftarrow A_2^{\mathcal{O}_2}(\mathtt{c}^*, \mathtt{st})$

$$\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{IND-atk-x}}(A) := |2 \Pr[b' = b] - 1|.$$

Here $\mathtt{m}_0$ and $\mathtt{m}_1$ should be of equal length and $\mathtt{ID}_S^*$ and $\mathtt{ID}_R^*$ should be distinct. Parameter $\mathtt{aux}$ is the empty string when $\mathtt{x} = \mathtt{I}$ and it is the KGC's secret key $\mathtt{Msk}$ when $\mathtt{x} = \mathtt{II}$. Note it is possible that the challenger is not aware of the secret value corresponding to $\mathtt{ID}_S^*$, if the associated public key has been replaced. In this case, we require the adversary to provide this value[4]. Here, implicitly, the challenger continues to use $\mathtt{Msk}$ which could be unknown to the adversary.

The adversary has access to six oracles:

– **Request Public Key**: On input of an identity $\mathtt{ID}$, this oracle returns the corresponding public key. If such a key does not yet exist, it is constructed using the `Generate-User-Keys` algorithm.
– **Replace Public Key**: On input an identity $\mathtt{ID}$ and a valid $\mathtt{PK}$, this oracle replaces the public key associated with $\mathtt{ID}$ with $\mathtt{PK}$.
– **Extract Partial Secret Key**: On input of an identity $\mathtt{ID}$, this oracle returns a partial secret key $D_{\mathtt{ID}}$ for that identity, generated using the `Extract-Partial-Private-Key` algorithm.
– **Extract Private Key**: On input of an identity $\mathtt{ID}$, this oracle returns the (full) private key for that identity $S_{\mathtt{ID}}$. If such a key does not yet exist, it is constructed using the appropriate algorithms. The adversary is not allowed to query this oracle on any identity for which the corresponding public key has been replaced. This restriction is imposed due to the fact that it is unreasonable to expect that the challenger is able to provide a full secret key for a user for which it does not know the secret value. Additionally, the adversary is never allowed to call this oracle on the challenge identities $\mathtt{ID}_S^*$ and $\mathtt{ID}_R^*$. To capture *insider security*, this restriction applies only to $\mathtt{ID}_R^*$.

[4] A non-polynomial-time challenge oracle could be formulated as in [1].

– **De-signcrypt**: On input of a ciphertext, a sender's identity and a receiver's identity, this oracle returns the result of running the Dsc algorithm on the ciphertext, the sender's public parameters, and the receiver's full private key. Note that, it is possible that the challenger is not aware of the receiver's secret value, if the associated public key has been replaced. In this case, we require the adversary to provide it [5]. Of course, the adversary is not allowed to query this oracle in the second stage of the game on $\mathtt{c}^*$ under $\mathtt{ID}_S^*$ and $\mathtt{ID}_R^*$, unless the public key $\mathtt{PK}_S^*$ of the sender *or* that of the receiver $\mathtt{PK}_R^*$ used to signcrypt $\mathtt{m}_b$ has been replaced after the challenge was issued. We also disallow queries where $\mathtt{ID}_R = \mathtt{ID}_S$.

Next we describe the additional oracle restrictions in each attack scenario[6]:

Type I Adversary (IND-oCCA-I) This scenario models an attacker which is a common user of the system and is not in possession of the KGC's secret key. This type of adversary is not allowed to extract the partial secret key for $\mathtt{ID}_R^*$ or $\mathtt{ID}_S^*$ if the public key of this identity has been replaced before the challenge ciphertext was issued. To capture insider security (iCCA), this restriction is lifted from $\mathtt{ID}_S^*$.

In our security analysis we use a weaker formulation of insider Type I adversaries which we refer to as insider Type I′. Here the adversary is not allowed to extract the partial private key of $\mathtt{ID}_R^*$ at all.

Type II Adversary (IND-oCCA-II) This scenario models an honest-but-curious KGC, against which we want to preserve confidentiality. For this type of adversary, the partial secret key extraction oracle is not necessary, as the adversary can simply generate these keys itself using Msk. Additionally, this type of adversary is not allowed to replace the public key for $\mathtt{ID}_R^*$ or $\mathtt{ID}_S^*$ before the challenge is issued. To capture insider security (iCCA), this restriction is lifted from $\mathtt{ID}_S^*$.

In Appendix A we prove the following lemma which justifies our definition of Type I′ attackers.

Lemma 1. *If a certificateless signcryption scheme is IND-iCCA secure against Type II and Type I′ attackers then it is also IND-iCCA secure against Type I attackers:*

$$\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{IND-iCCA-I}}(A) \leq 2\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{IND-iCCA-I}'}(B_0) + 2\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{IND-iCCA-II}}(B_1).$$

The intuition behind the proof is that we guess if the adversary is going to replace the public key for the challenge identity or extract the partial private

[5] Note that, implicitly, the oracle answers continue to use Msk, which could be unknown to the adversary. A non-polynomial-time de-signcryption oracle could also be formulated as in [1].

[6] In all of the following scenarios, if the adversary does not make any decryption queries it is said to be an IND-oCPA/iCPA-x adversary.

key for it and, accordingly, we use the adversary to win a Type I′ or a Type II security game.

The authenticity property required for certificateless signcryption schemes is captured by the following (strong) existential unforgeability security model.

sUF-atk-x
1. $(\mathtt{Msk}, \mathtt{params}) \leftarrow \mathtt{Setup}(1^\kappa)$
2. $(\mathtt{c}^*, \mathtt{ID}_S^*, \mathtt{ID}_R^*) \leftarrow A^{\mathcal{O}}(\mathtt{params}, \mathtt{aux})$

where $\mathtt{aux}$ is the empty string when $\mathtt{x} = \mathtt{I}$ and it is the master secret key $\mathtt{Msk}$ when $\mathtt{x} = \mathtt{II}$. For the unforgeability game, we define the adversary's advantage as

$$\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{sUF-atk-x}}(A) := \Pr[\mathtt{m}^* \neq \perp \wedge (\mathtt{m}^*, \mathtt{ID}_S^*, \mathtt{PK}_S^*, \mathtt{ID}_R^*, \mathtt{PK}_R^*, \mathtt{c}^*) \notin L],$$

where $\mathtt{m}^* := \mathtt{Dsc}(\mathtt{c}^*, S_R^*, \mathtt{ID}_R^*, \mathtt{PK}_R^*, \mathtt{ID}_S^*, \mathtt{PK}_S^*, \mathtt{params})$ and $\mathtt{ID}_S^*$ and $\mathtt{ID}_R^*$ should be distinct. We denote by L the list of inputs and the corresponding outputs in queries to the signcryption oracle which is described below. The entries of this list are of the form $(\mathtt{m}, \mathtt{ID}_S, \mathtt{PK}_S, \mathtt{ID}_R, \mathtt{PK}_R, \mathtt{c})$ where $\mathtt{PK}_S$ and $\mathtt{PK}_R$ are the public keys corresponding to the queried identities at the time the query is placed. A weaker form of unforgeability can be defined by imposing that the signcryption oracle has not been used to obtain a different ciphertext on the same parameters associated with $\mathtt{c}^*$, namely $\mathtt{m}^*$, $\mathtt{ID}_S^*$, $\mathtt{ID}_R^*$ and the associated public keys at the time of A's termination.

The adversary has access to the same oracles as in the confidentiality game as well as an additional signcryption oracle. We describe the differences to the previous security game:

– **Extract Private Key**: Same as in the previous game, but an insider adversary is allowed to query this oracle only on $\mathtt{ID}_S^*$, rather than on $\mathtt{ID}_R^*$.
– **De-signcrypt**: We still disallow queries where $\mathtt{ID}_R = \mathtt{ID}_S$. Apart from this, there are no restrictions on calls to this oracle, although the adversary should provide the secret value for the receiver, in case the corresponding public key has been replaced.
– **Signcrypt**: On input of a message, a sender's identity and a receiver's identity, this oracle returns the result of running the signcryption algorithm on the message, the sender's full private key, and the receiver's public parameters. Note that, it is possible that the challenger is not aware of the sender's secret value, if the associated public key has been replaced. In this case, we require the adversary to provide it. We also disallow queries where $\mathtt{ID}_R = \mathtt{ID}_S$.

Various attack scenarios are as follows[7]:

Type I Adversary (sUF-oCMA-I) This type of adversary is not allowed to extract the partial secret keys for $\mathtt{ID}_S^*$ or $\mathtt{ID}_R^*$ if the public keys for these identities

[7] In all of the following scenarios, if the adversary does not make any signcryption queries it is said to be an UF-oNMA/iNMA-x (no message attack) adversary.

have been replaced. To capture insider security (iCMA), this restriction is lifted from $\mathtt{ID}_R^*$.

In our security analysis we use a weaker formulation of insider Type I adversaries which we refer to as insider Type I′. Here the adversary is not allowed to extract the partial private key of $\mathtt{ID}_S^*$ at all.

Type II Adversary (sUF-oCMA-II) This type of adversary is not allowed to replace the encrypt/verify key for $\mathtt{ID}_S^*$ or $\mathtt{ID}_R^*$. To capture insider security (iCMA), this restriction is lifted from $\mathtt{ID}_R^*$.

The following result, which relates the different authenticity adversarial models, is analogous to that presented for confidentially in Lemma 1.

Lemma 2. *If a certificateless signcryption scheme is sUF-iCMA secure against Type II and Type I′ attackers then it is also sUF-iCMA secure against Type I attackers:*

$$\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{sUF-iCMA-I}}(A) \leq 2\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{sUF-iCMA-I'}}(B_0) + 2\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{sUF-iCMA-II}}(B_1).$$

5 An Efficient Certificateless Signcryption Scheme

We now present our certificateless signcryption scheme which can be seen as an Encrypt-then-Sign construction where randomness is shared between signature and encryption schemes. Our scheme, which relies on a symmetric bilinear group description Γ, is as follows. We choose four cryptographic hash functions:

$$H_1 : \{0,1\}^* \to G_1, H_2 : \{0,1\}^* \to \{0,1\}^\kappa,$$
$$H_3 : \{0,1\}^* \to G_1, H_4 : \{0,1\}^* \to G_1.$$

We then select $\mathtt{Msk}$ uniformly at random from $\mathbb{Z}_p$, set $\mathtt{Mpk} := \mathtt{Msk} \cdot P$ and let $\mathtt{params} := (\Gamma, \mathtt{Mpk})$. The partial secret key extraction algorithm on input $(\mathtt{ID}, \mathtt{Msk})$ returns $D := \mathtt{Msk} \cdot H_1(\mathtt{ID})$. The user key generation algorithm returns a random element x from $\mathbb{Z}_p$ as the secret value, and $\mathtt{PK} := x \cdot P$ as the public key. The full private key is then set to be $S := (x, D)$. Message, ciphertext and randomness spaces are $\{0,1\}^\kappa$, $G_1 \times \{0,1\}^\kappa \times G_1$ and $\mathbb{Z}_p$ respectively. Signcryption and de-signcryption algorithms are given below.

$\mathtt{Sc}(\mathtt{m}, S_S, \mathtt{ID}_S, \mathtt{PK}_S, \mathtt{ID}_R, \mathtt{PK}_R, \mathtt{Mpk})$	$\mathtt{Dsc}(\mathtt{c}, S_R, \mathtt{ID}_R, \mathtt{PK}_R, \mathtt{ID}_S, \mathtt{PK}_S, \mathtt{Mpk})$
1. $r \leftarrow \mathbb{Z}_p$; $U \leftarrow rP$; $T \leftarrow e(\mathtt{Mpk}, Q_R)^r$	1. $(U, V, W) \leftarrow \mathtt{c}$
2. $h \leftarrow H_2(U, T, r\mathtt{PK}_R, \mathtt{ID}_R, \mathtt{PK}_R)$	2. $H \leftarrow H_3(U, V, \mathtt{ID}_S, \mathtt{PK}_S)$
3. $V \leftarrow \mathtt{m} \oplus h$	3. $H' \leftarrow H_4(U, V, \mathtt{ID}_S, \mathtt{PK}_S)$
4. $H \leftarrow H_3(U, V, \mathtt{ID}_S, \mathtt{PK}_S)$	4. If $e(\mathtt{Mpk}, Q_S)e(U, H)e(\mathtt{PK}_S, H') \neq$ $e(P, W)$ return $\perp$
5. $H' \leftarrow H_4(U, V, \mathtt{ID}_S, \mathtt{PK}_S)$	5. $(x_R, D_R) \leftarrow S_R$; $T \leftarrow e(D_R, U)$
6. $(x_S, D_S) \leftarrow S_S$	6. $h \leftarrow H_2(U, T, x_R U, \mathtt{ID}_R, \mathtt{PK}_R)$
7. $W \leftarrow D_S + rH + x_S H'$	7. $\mathtt{m} \leftarrow V \oplus h$
8. $\mathtt{c} \leftarrow (U, V, W)$	8. Return $\mathtt{m}$
9. Return $\mathtt{c}$	

We now turn to the security analysis of this scheme.

Theorem 1. *The certificateless signcryption scheme above is IND-iCCA-I/II secure, in the random oracle model, under the assumption that the gap bilinear Diffie-Hellman problem is intractable in the underlying bilinear group.*

This theorem follows from Lemmas 1, 3 and 4.

Lemma 3. *Under the GBDH assumption, no PPT attacker A has non-negligible advantage in winning the IND-iCCA-I′ game against the scheme proposed above, when all hash functions are modelled as random oracles. More precisely, there exists an algorithm B which uses A to solve the GBDH problem such that:*

$$\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{IND-iCCA-I'}}(A) \leq q_T \mathrm{Adv}_{\Gamma}^{\mathtt{GBDH}}(B, q_D^2 + 2q_D q_2 + q_2),$$

where $q_T = q_1 + q_X + q_{SK} + 2q_D + 2$. Here q_1, q_2, q_X, q_{SK} and q_D are the maximum number of queries that the adversary could place to H_1, H_2, partial private key extraction, private key extraction and de-signcryption oracles.

Proof. On receiving the GBDH challenge tuple (Γ, aP, bP, cP), where the generator is P, algorithm B sets $\mathtt{Mpk} := aP$ and $\mathtt{params} = (\Gamma, \mathtt{Mpk})$ and passes them on to A. Algorithm B chooses an index ℓ uniformly at random in $\{1, \ldots, q_T\}$, where q_T is as in the statement of the Lemma, and answers various oracle queries as follows:

H_1 **Queries**: On the i-th non-repeat query $\mathtt{ID}$, if $i \neq \ell$ algorithm B chooses $r \in \mathbb{Z}_p$ uniformly at random and sets $Q_{\mathtt{ID}} = rP$. It then adds $(i, \mathtt{ID}, r)$ to a list L_1 which is initially empty and returns $Q_{\mathtt{ID}}$. Otherwise, it returns $Q_{\mathtt{ID}_\ell} = bP$ and adds $(\ell, \mathtt{ID}, \perp)$ to L_1. From this point on on we denote the ℓ-th non-repeat identity queried to this oracle with $\mathtt{ID}_\ell$.

Extract Partial Secret Key Queries: For each new query $\mathtt{ID}$, algorithm B calls H_1 on $\mathtt{ID}$ and obtains $(i, \mathtt{ID}, r)$. If $i = \ell$ then B aborts the simulation. Otherwise, B returns $D = raP$.

Request Public Key Queries: For each query $\mathtt{ID}$, algorithm B checks in list L_K, which is initially empty, if there is a tuple $(\mathtt{ID}, \mathtt{PK}, x)$. If so, then B returns $\mathtt{PK}$. Otherwise, B generates a new key pair, updates the list L_K, and returns the public key.

Replace Public Key Queries: On input $(\mathtt{ID}, \mathtt{PK})$ algorithm B inserts/updates L_K with tuple $(\mathtt{ID}, \mathtt{PK}, \perp)$.

Extract Private Key Queries: For each new query $\mathtt{ID}$, algorithm B calls H_1 on $\mathtt{ID}$ and obtains $(i, \mathtt{ID}, r)$. If $i = \ell$ then B aborts the simulation. Otherwise, B searches L_K for the entry $(\mathtt{ID}, \mathtt{PK}, x)$, generating a new key pair if this does not exist, and returns (x, raP).

H_3 **Queries**: For each new query $(U, V, \mathtt{ID}, \mathtt{PK})$, algorithm B generates a random value t in $\mathbb{Z}_p$, updates an initially empty list L_3 with the input, t and tP and returns tP.

H_4 **Queries**: For each new query $(U, V, \texttt{ID}, \texttt{PK})$, algorithm B generates a random value s in $\mathbb{Z}_p$, updates an initially empty list L_4 with the input, s and sP and returns sP.

H_2 **Queries**: For each new query $(U, T, R, \texttt{ID}, \texttt{PK})$, algorithm B proceeds as follows:

1. It checks if the decision bilinear Diffie-Hellman oracle returns 1 when queried with the tuple (aP, bP, cP, T). If this is the case, algorithm B returns T and stops.
2. Algorithm B goes through the list L_2 with entries $(U, \star, R, \texttt{ID}, \texttt{PK}, h)$, for different values of h, such that the decision bilinear Diffie-Hellman oracle returns 1 when queried on the tuple (aP, bP, U, T). Note that in this case $\texttt{ID} = \texttt{ID}_\ell$. If such a tuple exists, it returns h (and replaces the symbol $\star$ with T).
3. If B reaches this point of execution, it returns a random h and updates the list L_2, which is initially empty, with a tuple containing the input and return values.

De-signcryption Queries: For each new query $(U, V, W, \texttt{ID}, \texttt{ID}')$ algorithm B proceeds as follows:

1. It executes the verification part of the de-signcryption algorithm obtaining $Q_{\texttt{ID}}$ and $\texttt{PK}$ by calling the H_1 and request public key oracles. It returns $\perp$ if verification does not succeed.
2. It calculates $R := x'U$, obtaining x' (and hence $\texttt{PK}'$) from either the adversary or by calling the request public key oracle.
3. If $\texttt{ID}' \neq \texttt{ID}_\ell$, it calculates $T = e(rU, \texttt{Mpk})$, where $(j, \texttt{ID}', r)$ is obtained by calling H_1 on $\texttt{ID}'$, and completes de-signcryption in the usual way placing a query on H_2.
4. If $\texttt{ID}' = \texttt{ID}_\ell$ then the pairing cannot be calculated. In order to return a consistent answer, B goes through L_2 and looks for a tuple $(U, T, R, \texttt{ID}_\ell, \texttt{PK}', h)$, for different values of T, such that the decision bilinear Diffie-Hellman oracle returns 1 when queried on (aP, bP, U, T). If such an entry exists, the correct pairing value is found and B decrypts using the hash value h.
5. If B reaches this point of execution, it places the entry $(U, \star, R, \texttt{ID}_\ell, \texttt{PK}', h)$ for a random h on list L_2 and decrypts using this h. The symbol $\star$ denotes an unknown value of pairing. Note that the identity component of all entries with a $\star$ is $\texttt{ID}_\ell$.

Eventually, A outputs two messages $(\texttt{m}_0, \texttt{m}_1)$ and two identities $\texttt{ID}_S^*$ and $\texttt{ID}_R^*$. Algorithm B places a query on H_1 with input $\texttt{ID}_R^*$. If the index of $\texttt{ID}_R^*$ is not ℓ, algorithm B fails. Otherwise it proceeds to construct a challenge as follows. It obtains from L_K the public key $\texttt{PK}$ corresponding to $\texttt{ID}_S^*$. Then it sets $U^* := cP$, selects a random bit σ and a random hash value h^* and sets $V^* := \texttt{m}_\sigma \oplus h^*$. The component W^* is set to be $D_S + rH + x_S H' = D_S + tcP + s\texttt{PK}$ where t is obtained from L_3, s is obtained from L_4 and D_S is calculated by calling the partial secret key extraction oracle on $\mathtt{ID}_S^*$. Note that, since $\mathtt{ID}_S^* \neq \mathtt{ID}_R^*$ the partial secret key extraction oracle simulation always give B the correct value of D_S.

In the second stage, A's queries are answered as before. Eventually, A will output its guess as to which message is signcrypted inside the challenge. Since ℓ is independent of adversary's view, and the list L_1 can be easily seen to have at most q_T elements, with probability $1/q_T$ the adversary will output an identity $\mathtt{ID}_\ell$ with index ℓ. If this event occurs, the simulation is perfect unless the adversary queries H_2 on the challenge-related tuple $(U^*, T^*, R^*, \mathtt{ID}_\ell, \mathtt{PK}^*)$. Since the hash function H_2 is modelled as a random oracle, the adversary will not have any advantage if this tuple does not appear on L_2. However, if this happens, B will win the game due to the first step in the simulation of H_2. The Lemma follows from this observation and the fact that the total number of decision bilinear Diffie-Hellman oracle calls that B makes is at most $q_D^2 + 2q_D q_2 + q_2$. □

Lemma 4. *Under the CDH assumption in G_1 no PPT attacker A has non-negligible advantage in winning the IND-iCCA-II game against the scheme proposed above, when all hash functions are modelled as random oracles. More precisely, there exists an algorithm B which uses A to solve the CDH problem such that:*

$$\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{IND-iCCA-II}}(A) \leq q_T \mathrm{Adv}_{\Gamma}^{\mathtt{CDH}}(B),$$

where $q_T = q_{PK} + q_{RPK} + q_{SK} + 2q_D + 2$. Here q_{PK} and q_{RPK} are the maximum number of queries that the adversary could place to request public key and replace public key oracles and q_{SK} and q_D are as before.

Proof. On receiving the CDH challenge tuple (Γ, aP, bP), with generator P, algorithm B generates a master key pair $(\mathtt{Msk}, \mathtt{Mpk})$ and sets $\mathtt{params} := (\Gamma, \mathtt{Mpk})$ and passes these on to A. Algorithm B chooses an index ℓ uniformly at random in $\{1, \ldots, q_T\}$, with q_T as in the statement of the Lemma, and answers various oracle queries as follows:

H_1 **Queries**: On the non-repeat query $\mathtt{ID}$ algorithm B chooses $r \in \mathbb{Z}_p$ uniformly at random and sets $Q_{\mathtt{ID}} = rP$. It then adds $(\mathtt{ID}, r)$ to a list L_1 which is initially empty and returns $Q_{\mathtt{ID}}$.

Request Public Key Queries: On the i-th non-repeat query $\mathtt{ID}$, if $i \neq \ell$, algorithm B generates a new key pair $(x, \mathtt{PK})$, updates the list L_K with $(i, \mathtt{ID}, x, \mathtt{PK})$. If $i = \ell$ algorithm B returns aP and adds $(\ell, \mathtt{ID}, aP, \perp)$ to L_K. From this point on on we denote the ℓ-th non-repeat identity queried to this oracle with $\mathtt{ID}_\ell$.

Replace Public Key Queries: On input $(\mathtt{ID}, \mathtt{PK})$ on the i-th non-repeat identity $\mathtt{ID}$ algorithm B inserts/updates L_K with tuple $(i, \mathtt{ID}, \mathtt{PK}, \perp)$. If $i = \ell$ then B aborts the simulation.

Extract Private Key Queries: For each new query $\mathtt{ID}$, B calls request public key on $\mathtt{ID}$ obtaining $(i, \mathtt{ID}, \mathtt{PK}, x)$. If $i = \ell$, algorithm B aborts the simulation. Otherwise, it calls H_1 on $\mathtt{ID}$ and gets $(\mathtt{ID}, r)$. It returns $(x, r\mathtt{Msk}P)$.

H_3 **Queries**: For each new query $(U, V, \mathtt{ID}, \mathtt{PK})$, algorithm B generates a random value t in $\mathbb{Z}_p$, updates an initially empty list L_3 with the input, t and tP and returns tP.

H_4 **Queries**: For each new query $(U, V, \mathtt{ID}, \mathtt{PK})$, algorithm B generates a random value s in $\mathbb{Z}_p$, updates an initially empty list L_4 with the input, s and sP and returns sP.

H_2 **Queries**: For each new query $(U, T, R, \mathtt{ID}, \mathtt{PK})$, algorithm B proceeds as follows:

1. It checks if $e(aP, bP) = e(P, R)$. If so, B returns R and stops.
2. Algorithm B goes through the list L_2 looking for entries $(U, T, \star, \mathtt{ID}, \mathtt{PK}, h)$, such that $e(U, bP) = e(P, R)$. Note that in this case $\mathtt{ID} = \mathtt{ID}_\ell$. If such a tuple exists, it returns h (and replaces the symbol $\star$ with R).
3. If B reaches this point of execution, it returns a random h and updates the list L_2, which is initially empty, with a tuple containing the input and return values.

De-signcryption Queries: For each new query $(U, V, W, \mathtt{ID}, \mathtt{ID}')$, algorithm B proceeds as follows:

1. It executes the verification part of the de-signcryption algorithm obtaining $Q_{\mathtt{ID}}$ and $\mathtt{PK}$ by calling H_1 and request public key oracles. It returns $\perp$ if the verification does not succeed.
2. It calculates $T = e(U, r'\mathtt{Mpk})$, where $(\mathtt{ID}', r')$ is obtained from H_1
3. If $\mathtt{ID}' \neq \mathtt{ID}_\ell$, it calculates $R := x'U$, where $(j, \mathtt{ID}', \mathtt{PK}', x')$ is obtained by calling the request public key oracle on $\mathtt{ID}'$, and x' is possibly received from the adversary. It completes de-signcryption in the usual way by placing a query on H_2.
4. If $\mathtt{ID}' = \mathtt{ID}_\ell$, the correct value of R cannot be computed. To answer the query consistently, B goes through L_2 and looks for a tuple $(U, T, R, \mathtt{ID}_\ell, \mathtt{PK}', h)$, for different values of R, such that $e(U, bP) = e(P, R)$. If such an entry exists, the correct value of R is found, and B decrypts using h.
5. If B reaches this point of execution, B places the entry $(U, T, \star, \mathtt{ID}', \mathtt{PK}', h)$ for a random h on list L_2 and decrypts using this h.

Eventually, A outputs two messages $(\mathtt{m}_0, \mathtt{m}_1)$ and two identities $\mathtt{ID}_S^*$ and $\mathtt{ID}_R^*$. Algorithm B queries the request public key oracle on $\mathtt{ID}_R^*$ and receives $(j, \mathtt{ID}_R^*, \mathtt{PK}^*, x^*)$. If $j \neq \ell$, it fails. Otherwise it proceeds to construct a challenge as follows. It obtains the public key $\mathtt{PK}$ for $\mathtt{ID}_S^*$ by calling the request public key oracle. It sets $U^* = bP$, selects a random bit σ and a random hash value h^* and sets $V^* = \mathtt{m}_\sigma \oplus h^*$. The component W^* is set to be $D_S + rH + x_S H' = D_S + tcP + s\mathtt{PK}$, where D_S is obtained by calling the extract partial secret key oracle[8], t is obtained from L_3 and s is obtained from L_4.

In the second stage, A's queries are answered as before. Eventually, A will output its guess as to which message is signcrypted inside the challenge. Since ℓ is

[8] Unlike Lemma 3 note that the simulation is possible for $\mathtt{ID}_S^* = \mathtt{ID}_R^*$.

independent of adversary's view, with probability $1/q_T$ the adversary will output an identity $\mathtt{ID}_R^*$ with index ℓ. If this event occurs, the simulation is perfect unless the challenge-related tuple $(U^*, T^*, R^*, \mathtt{ID}_\ell, \mathtt{PK}^*)$ is queried from H_2. However, since the hash function H_2 is modelled as a random oracle, the adversary will not have any advantage if this entry does not appear on L_2 list and, in this case, B will have won the game due to its simulation of H_2. The Lemma follows from this observation and the fact that the maximum length of the list L_K is q_T, as stated in the Lemma. $\square$

Theorem 2. *The certificateless signcryption scheme above is sUF-iCMA-I/II secure, in the random oracle model, under the GDH′ assumption in G_1.*

This theorem follows from Lemmas 2, 5 and 6.

Lemma 5. *Under the GDH′ assumption in G_1, no PPT attacker A has non-negligible advantage in winning the sUF-iCMA-I′ game against the scheme proposed above, when all hash functions are modelled as random oracles. More precisely, there exists an algorithm B which uses A to solve the GDH′ problem such that:*

$$\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{sUF-iCMA-I}'}(A) \leq q_T \mathrm{Adv}_{\Gamma}^{\mathtt{GDH}'}(B, q_D^2 + 2q_D q_2) + (q_{SC}(q_{SC} + q_D + q_3 + 1) + 2)/2^\kappa,$$

where $q_T = q_1 + q_X + q_{SK} + 2q_D + 2q_{SC} + 1$. Here q_3 and q_{SC} is the maximum number of queries that the adversary could place to the H_3 and signcryption oracles and q_1, q_X, q_{SK} and q_D are as before.

Proof. To prove this Lemma, we construct an algorithm B which uses A to solve the GDH′ problem over G_1. Algorithm B receives a GDH′ problem instance (Γ, aP, bP), with generator P, it sets $\mathtt{Mpk} := aP$ and provides $\mathtt{params} := (\Gamma, \mathtt{Mpk})$ to A. Algorithm B then chooses an index ℓ uniformly at random in $\{1, \ldots, q_T\}$, where q_T is as in the statement of the Lemma, and answers various oracle queries as follows:

H_1 **Queries**: On the i-th non-repeat query $\mathtt{ID}$, if $i \neq \ell$ algorithm B chooses $r \in \mathbb{Z}_p$ uniformly at random and sets $Q_{\mathtt{ID}} = rP$. It then adds $(i, \mathtt{ID}, r)$ to a list L_1 which is initially empty and returns $Q_{\mathtt{ID}}$. Otherwise, it returns $Q_{\mathtt{ID}_\ell} = bP$ and adds $(\ell, \mathtt{ID}, \perp)$ to L_1. From this point on on we denote the ℓ-th non-repeat identity queried to this oracle with $\mathtt{ID}_\ell$.

Extract Partial Secret Key Queries: For each new query $\mathtt{ID}$, algorithm B calls H_1 on $\mathtt{ID}$ and obtains $(i, \mathtt{ID}, r)$. If $i = \ell$ then B aborts the simulation. Otherwise, B returns $D = raP$.

Request Public Key Queries: For each query $\mathtt{ID}$, algorithm B checks in list L_K, which is initially empty, if there is a tuple $(\mathtt{ID}, \mathtt{PK}, x)$. If so, then B returns $\mathtt{PK}$. Otherwise, B generates a new key pair, updates the list L_K, and returns the public key.

Replace Public Key Queries: On input $(\mathtt{ID}, \mathtt{PK})$ algorithm B inserts/updates L_K with tuple $(\mathtt{ID}, \mathtt{PK}, \perp)$.

Extract Private Key Queries: For each new query $\mathtt{ID}$, algorithm B calls H_1 on $\mathtt{ID}$ and obtains $(i, \mathtt{ID}, r)$. If $i = \ell$ then B aborts the simulation. Otherwise, B searches L_K for the entry $(\mathtt{ID}, \mathtt{PK}, x)$, generating a new key pair if this does not exist, and returns (x, raP).

H_3 **Queries**: For each new query $(U, V, \mathtt{ID}, \mathtt{PK})$, algorithm B generates a random value t in $\mathbb{Z}_p$, updates an initially empty list L_3 with the input, t and tP and returns tP.

H_4 **Queries**: For each new query $(U, V, \mathtt{ID}, \mathtt{PK})$, algorithm B generates a random value s in $\mathbb{Z}_p$, updates an initially empty list L_4 with the input, s and sP and returns sP.

H_2 **Queries**: For each new query $(U, T, R, \mathtt{ID}, \mathtt{PK})$, algorithm B proceeds as follows:

1. It checks if $e(aP, bP) = e(P, R)$. If this is the case, algorithm B returns R and stops.
2. Algorithm B goes through the list L_2 with entries $(U, \star, R, \mathtt{ID}, \mathtt{PK}, h)$, for different values of h, such that the decision bilinear Diffie-Hellman oracle returns 1 when queried on the tuple (aP, bP, U, T). Note that in this case $\mathtt{ID} = \mathtt{ID}_\ell$. If such a tuple exists, it returns h (and replaces the symbol $\star$ with T).
3. It goes through list L_2 with entries entry $(U, T, \star, \mathtt{ID}, \mathtt{PK}, h)$, for different values of h, such that $e(U, \mathtt{PK}') = e(P, R)$. If such a tuple exists, it returns h (and replaces the symbol $\star$ with R).
4. If B reaches this point of execution, it returns a random h and updates the list L_2, which is initially empty, with a tuple containing the input and return values.

De-signcryption Queries: For each new query $(U, V, W, \mathtt{ID}, \mathtt{ID}')$ algorithm B proceeds as follows:

1. It executes the verification part of the de-signcryption algorithm obtaining $Q_{\mathtt{ID}}$ and $\mathtt{PK}$ by calling the H_1 and request public key oracles. It returns $\perp$ if verification does not succeed.
2. It checks if $\mathtt{ID} = \mathtt{ID}_\ell$ and if this is the case then B can solve the GDH$'$ problem as described below.
3. It calculates $R := x'U$, obtaining x' (and hence $\mathtt{PK}'$) from either the adversary or by calling the request public key oracle.
4. If $\mathtt{ID}' \neq \mathtt{ID}_\ell$, it calculates $T = e(rU, \mathtt{Mpk})$, where $(j, \mathtt{ID}', r)$ is obtained by calling H_1 on $\mathtt{ID}'$, and completes de-signcryption in the usual way placing a query on H_2.
5. If $\mathtt{ID}' = \mathtt{ID}_\ell$ then the pairing cannot be calculated. In order to return a consistent answer, B goes through L_2 and looks for a tuple $(U, T, R, \mathtt{ID}_\ell, \mathtt{PK}', h)$, for different values of T, such that the decision bilinear Diffie-Hellman oracle returns 1 when queried on (aP, bP, U, T). If such an entry exists, the correct pairing value is found and B decrypts using the hash value h.

6. If B reaches this point of execution, it places the entry $(U, \star, R, \mathtt{ID}_\ell, \mathtt{PK}', h)$ for a random h on list L_2 and decrypts using this h. The symbol $\star$ denotes an unknown value of pairing. Note that the identity component of all entries with a $\star$ is $\mathtt{ID}_\ell$.

Signcryption Queries: For each new query $(\mathtt{m}, \mathtt{ID}, \mathtt{ID}')$, algorithm B proceeds as follows:

1. It calls H_1 on $\mathtt{ID}$. If $\mathtt{ID} \neq \mathtt{ID}_\ell$, algorithm B simply signcrypts the message, getting the secret value x_S from the request public key or the adversary if necessary.
2. If $\mathtt{ID} = \mathtt{ID}_\ell$ (and hence $\mathtt{ID}' \neq \mathtt{ID}_\ell$), algorithm B generates two random values $u, v \in \mathbb{Z}_p$, sets $U = vaP$, calculates $T = e(U, r'\mathtt{Mpk})$, obtaining $(j, \mathtt{ID}', r')$ by calling H_1 on $\mathtt{ID}'$.
3. It goes through list L_2 looking for an entry $(U, T, R, \mathtt{ID}', \mathtt{PK}', h)$ for some R such that $e(U, \mathtt{PK}') = e(P, R)$, where $\mathtt{PK}'$ is obtained by calling the request public key oracle on $\mathtt{ID}'$. If such an entry exists, it calculates $V = \mathtt{m} \oplus h$. Otherwise it uses a random h and updates the list L_2 with $(U, T, \star, \mathtt{ID}', \mathtt{PK}', h)$.
4. Then B defines the hash value $H_3(U, V, \mathtt{ID}_\ell, \mathtt{PK})$ as $H = v^{-1}(uP - Q_{\mathtt{ID}_\ell})$, aborting the simulation if a such a hash queries has been responded with a different value before. This means that B updates list L_3 with tuple $(U, V, \mathtt{ID}_\ell, \mathtt{PK}, \perp, H)$. Finally, B sets $W = uaP + s\mathtt{PK}$, where s is the value obtained by querying H_4 on $(U, V, \mathtt{ID}_\ell, \mathtt{PK})$ and returns (U, V, W). Note that this is a valid signcryption.

Eventually, A outputs a signcryption (U^*, V^*, W^*) from sender $\mathtt{ID}_S^*$ to receiver $\mathtt{ID}_R^*$. Algorithm B now calls H_1 on $\mathtt{ID}_S^*$ and checks if $\mathtt{ID}_S^* = \mathtt{ID}_\ell$ and if this is not the case it aborts execution. Otherwise, it obtains $\mathtt{PK}^*$ by calling the request public key oracle on $\mathtt{ID}_S^*$ and retrieves t^* and s^* from lists L_3 and L_4 by querying H_3 and H_4 on $(U^*, V^*, \mathtt{ID}_\ell, \mathtt{PK}^*)$. Note that if A succeeded, then the verification condition holds:

$$
\begin{aligned}
e(P, W^*) &= e(\mathtt{Mpk}, Q_{\mathtt{ID}_\ell}) e(U^*, H^*) e(\mathtt{PK}^*, H'^*) \\
e(P, W^*) &= e(aP, bP) e(U^*, t^*P) e(\mathtt{PK}^*, s^*P) \\
e(P, abP) &= e(P, W^* - t^*U^* - s^*\mathtt{PK}^*),
\end{aligned}
$$

and thus B can recover

$$abP = W^* - t^*U^* - s^*\mathtt{PK}^*.$$

Let us now analyse the probability that B succeeds in solving the GDH$'$ problem instance. For this to happen, the simulation must run until the end of the game, the adversary must pick a specific identity as $\mathtt{ID}_S^*$, and it must query the hash functions H_3 and H_4 to properly construct the forgery. The probability that A is able to produce a forgery without querying both hash functions is upper bounded by $2/2^\kappa$.

The probability that B aborts the simulation is related with the following events:

– A places a partial key extraction on $\mathtt{ID}_\ell$.
– A places a full secret key extraction on $\mathtt{ID}_\ell$.
– B wants to simulate a signcryption query and this leads to an inconsistency in the H_3 simulation.

Note that if A places either of the first two fatal queries, then it could not possibly use $\mathtt{ID}_\ell$ as the sender identity in the forgery it produces at the end of the game, so we can pinpoint the probability that B does not abort the simulation due to these events and A picks the only useful case for solving GDH′ as $1/q_T$. Note that the maximum length of the list L_1 is q_T, as stated in the Lemma.

The latter fatal event occurs if B's simulation triggers a collision in its simulation of H_3. Since the maximum size of L_3 is $q_{SC} + q_D + q_3 + 1$, we can upper bound the probability that this occurs as $q_{SC}(q_{SC} + q_D + q_3 + 1)/2^\kappa$. The result follows by noting that B makes at most $q_D^2 + 2q_Dq_2$ queries to its decision bilinear Diffie-Hellman oracle. □

Lemma 6. *Under the CDH assumption in G_1, no PPT attacker A has non-negligible advantage in winning the sUF-iCMA-II game against the scheme proposed above, when all hash functions are modelled as random oracles. More precisely, there exists an algorithm B which uses A to solve the CDH problem such that:*

$$\mathrm{Adv}^{\mathtt{sUF-iCMA-II}}_{\mathtt{CLSC}}(A) \leq q_T \mathrm{Adv}^{\mathtt{CDH}}_{\Gamma}(B) + (q_{SC}(q_{SC} + q_D + q_3 + 1) + 2)/2^\kappa,$$

where $q_T = q_{PK} + q_{RPK} + q_{SK} + 2q_D + 2q_{SC} + 1$ and various q's are as before.

Proof. To prove this Lemma, we construct an algorithm B which uses A to solve the CDH problem over G_1. Algorithm B receives a CDH problem instance (Γ, aP, bP), with generator P, generates a master key pair $(\mathtt{Msk}, \mathtt{Mpk})$, sets $\mathtt{params} := (\Gamma, \mathtt{Mpk})$ and provides these to A. Algorithm B then chooses an index ℓ uniformly at random in $\{1, \ldots, q_T\}$, where q_T is as in the statement of the Lemma, and answers various oracle queries as follows:

H_1 **Queries**: On the non-repeat query $\mathtt{ID}$ algorithm B chooses $r \in \mathbb{Z}_p$ uniformly at random and sets $Q_{\mathtt{ID}} = rP$. It then adds $(\mathtt{ID}, r)$ to a list L_1 which is initially empty and returns $Q_{\mathtt{ID}}$.

Request Public Key Queries: On the i-th non-repeat query $\mathtt{ID}$, if $i \neq \ell$, algorithm B generates a new key pair $(x, \mathtt{PK})$, updates the list L_K with $(i, \mathtt{ID}, x, \mathtt{PK})$. If $i = \ell$ algorithm B returns aP and adds $(\ell, \mathtt{ID}, aP, \perp)$ to L_K. From this point on on we denote the ℓ-th non-repeat identity queried to this oracle with $\mathtt{ID}_\ell$.

Replace Public Key Queries: On input $(\mathtt{ID}, \mathtt{PK})$ on the i-th non-repeat identity $\mathtt{ID}$ algorithm B inserts/updates L_K with tuple $(i, \mathtt{ID}, \mathtt{PK}, \perp)$. If $i = \ell$ then B aborts the simulation.

Extract Private Key Queries: For each new query $\mathtt{ID}$, B calls request public key on $\mathtt{ID}$ obtaining $(i, \mathtt{ID}, \mathtt{PK}, x)$. If $i = \ell$, algorithm B aborts the simulation. Otherwise, it calls H_1 on $\mathtt{ID}$ and gets $(\mathtt{ID}, r)$. It returns $(x, r\mathtt{Msk}P)$.

H_3 **Queries**: For each new query $(U, V, \mathtt{ID}, \mathtt{PK})$, algorithm B generates a random value t in $\mathbb{Z}_p$, updates an initially empty list L_3 with the input, t and tP and returns tP.

H_4 **Queries**: For each new query $(U, V, \mathtt{ID}, \mathtt{PK})$, algorithm B generates a random value s in $\mathbb{Z}_p$, updates an initially empty list L_4 with the input, s and sbP and returns sbP.

H_2 **Queries**: For each new query $(U, T, R, \mathtt{ID}, \mathtt{PK})$, algorithm B proceeds as follows:

1. It checks if $e(aP, bP) = e(P, R)$. If so, B returns R and stops.
2. Algorithm B goes through the list L_2 looking for entries $(U, T, \star, \mathtt{ID}, \mathtt{PK}, h)$, such that $e(U, \mathtt{PK}) = e(P, R)$. If such a tuple exists, it returns h (and replaces the symbol $\star$ with R).
3. If B reaches this point of execution, it returns a random h and updates the list L_2, which is initially empty, with a tuple containing the input and return values.

De-signcryption Queries: For each new query $(U, V, W, \mathtt{ID}, \mathtt{ID}')$, algorithm B proceeds as follows:

1. It executes the verification part of the de-signcryption algorithm obtaining $Q_{\mathtt{ID}}$ and $\mathtt{PK}$ by calling H_1 and request public key oracles. It returns $\perp$ if the verification does not succeed.
2. It checks if $\mathtt{ID} = \mathtt{ID}_\ell$ and if this is the case then B can solve the CDH problem as described below.
3. It calculates $T = e(U, r'\mathtt{Mpk})$, where $(\mathtt{ID}', r')$ is obtained from H_1
4. If $\mathtt{ID}' \neq \mathtt{ID}_\ell$, it calculates $R := x'U$, where $(j, \mathtt{ID}', \mathtt{PK}', x')$ is obtained by calling the request public key oracle on $\mathtt{ID}'$, and x' is possibly received from the adversary. It completes de-signcryption in the usual way by placing a query on H_2.
5. If $\mathtt{ID}' = \mathtt{ID}_\ell$, the correct value of R cannot be computed. To answer the query consistently, B goes through L_2 and looks for a tuple $(U, T, R, \mathtt{ID}_\ell, \mathtt{PK}', h)$, for different values of R, such that $e(U, bP) = e(P, R)$. If such an entry exists, the correct value of R is found, and B decrypts using h.
6. If B reaches this point of execution, B places the entry $(U, T, \star, \mathtt{ID}', \mathtt{PK}', h)$ for a random h on list L_2 and decrypts using this h.

Signcryption Queries: For each new query $(\mathtt{m}, \mathtt{ID}, \mathtt{ID}')$, algorithm B calls the request public key oracle on $\mathtt{ID}$ and proceeds as follows:

1. If $\mathtt{ID} \neq \mathtt{ID}_\ell$, algorithm B simply signcrypts the message, getting the secret value x_S from the request public key or the adversary if necessary.
2. If $\mathtt{ID} = \mathtt{ID}_\ell$ algorithm B generates two random values $u, v \in \mathbb{Z}_p$, sets $U = vaP$ and calculates $T = e(U, \mathtt{Msk}Q_{\mathtt{ID}_j})$.

3. It goes through list L_2 looking for an entry $(U, T, R, \mathtt{ID}', \mathtt{PK}', h)$ for some R such that $e(U, \mathtt{PK}') = e(P, R)$, where $\mathtt{PK}'$ is obtained by calling the request public key oracle on $\mathtt{ID}'$. If such an entry exists, it calculates $V = \mathtt{m} \oplus h$. Otherwise it uses a random h and updates the list L_2 with $(U, T, \star, \mathtt{ID}', \mathtt{PK}', h)$.
4. Then B defines the hash value $H_3(U, V, \mathtt{ID}_\ell, \mathtt{PK})$ as $H = v^{-1}(uP - H_4)$, aborting the simulation if such a hash response has been given before, where H_4 is the output of $H_4(U, V, \mathtt{ID}_\ell, \mathtt{PK})$. This means that B updates list L_3 with tuple $(U, V, \mathtt{ID}_\ell, \mathtt{PK}, \perp, H)$. Finally, B sets $W = D_S + uaP$ and returns (U, V, W). Note that this is a valid signcryption.

Eventually, A outputs a valid signcryption (U^*, V^*, W^*) from sender $\mathtt{ID}_S^*$ to receiver $\mathtt{ID}_R^*$. Algorithm B now calls the request public key oracle on $\mathtt{ID}_S^*$, obtains $\mathtt{PK}^*$, and checks if $\mathtt{ID}_S^* = \mathtt{ID}_\ell$. If this is not the case it aborts the execution. Otherwise, it retrieves t^* and s^* from lists L_3 and L_4 by querying H_3 and H_4 on $(U^*, V^*, \mathtt{ID}_\ell, \mathtt{PK}^*)$. Note that if A succeeded, then the verification condition holds:

$$\begin{aligned} e(P, W^*) &= e(\mathtt{Mpk}, Q_{\mathtt{ID}_\ell}) e(U^*, H^*) e(\mathtt{PK}^*, H'^*) \\ e(P, W^*) &= e(\mathtt{Mpk}, Q_{\mathtt{ID}_\ell}) e(U^*, t^* P) e(aP, s^* bP) \\ e(P, s^* abP) &= e(P, W^* - D_{\mathtt{ID}_\ell} - t^* U^*) \end{aligned}$$

and thus B can recover

$$abP = (W^* - D_{\mathtt{ID}_\ell} - t^* U^*)/s^*.$$

Let us now analyse the probability that B succeeds in solving CDH. For this to happen, the simulation must run until the end of the game, the adversary must pick a specific identity as $\mathtt{ID}_S^*$, and it must query the hash functions H_3 and H_4 to properly construct the forgery. The probability that A is able to produce a forgery without querying both hash functions is upper bounded by $2/2^\kappa$. The probability that B aborts the simulation is related with the following events:

- A places a full secret key extraction on $\mathtt{ID}_\ell$.
- B wants to simulate a signcryption query and this leads to an inconsistency in the H_3 simulation.

Note that if A places the first fatal query, then it could not possible use $\mathtt{ID}_\ell$ as the sender identity in the forgery it produces at the end of the game, so we can pinpoint the probability that B does not abort the simulation due to this event and A picks the only useful case for solving CDH as $1/q_T$. Note that the maximum length of the list L_K is q_T, as stated in the Lemma.

The latter fatal event occurs if B's simulation triggers a collision in its simulation of H_3. Since the maximum size of L_3 is $q_{SC} + q_D + q_3 + 1$, we can upper bound the probability that this occurs as $q_{SC}(q_{SC} + q_D + q_3 + 1)/2^\kappa$. The result follows. □

6 Discussion

The security proof for the certificateless signcryption scheme presented in the previous section has several interesting aspects which we will now discuss.

Full Domain Hash: For the sake of clarity in the proof presentation, we chose not to adopt Coron's technique [9] to obtain tighter security reductions in the analysis of authenticity. Adaptation of this technique to the certificateless signcryption case can be achieved following the strategy introduced by Libert et al. in [16] for identity-based signature schemes. However, it is important to emphasise an issue specific to the certificateless setting which renders this adaptation less straightforward.

The adaptive power of a Type I attacker as defined in [1] allows the attacker to decide whether it replaces the public key for the challenge identity or it extracts the associated partial secret key. This means that a direct adaptation of the proof in [16], which embeds the hard problem instance in a fraction of the partial secret keys which arise in the security game, is meaningless for Type I adversaries that extract the partial secret key for the challenge identity.

This observation motivated the definition of the Type I$'$ attack model in this paper, and the lemmas relating Type I and Type II security with this new variant. The limited adaptive behaviour of Type I$'$ attackers permits applying Coron's technique directly in the certificateless scenario. As an example of why this is a relevant contribution, we refer to the certificateless signature proposed in [19], which is claimed to be secure against Type I attackers. The proof which is presented for this scheme is flawed, and actually establishes security against more limited Type I$'$ adversaries.

Randomness Reuse: The proposed signcryption scheme is structured internally as an Encrypt-then-Sign construction using algorithms from [8] and [19] and sharing randomness between the two schemes. The encryption algorithm can be shown to be IND-CPA secure, whereas the signature algorithm is sUF-CMA secure. The expected security of our construction, which follows from the work of An et al. [2], is therefore IND-CCA security against outsider adversaries and full insider sUF-CMA security. It is interesting to note, however, that our scheme presents full insider security for confidentiality. This is due to the reuse of randomness between the encryption and signature components which intuitively prevents an insider adversary from being able to forge a valid signcryption from another one for which it does not know the implicit randomness.

Randomness reuse also provides the usual efficiency gains. We are able to save a few exponentiations and one ciphertext element through this technique. Efficiency benefits also justify our choice of the GBDH problem in the security reduction. The gap oracle allows us to construct a consistent simulation without resorting to a generic transformation akin to that in [15] which would add an extra ciphertext element to the scheme and a costly consistency check in designcryption. As a final note on the efficiency of the scheme, we note that we could have based our construction on the certificateless encryption scheme in [1]. This would provide a small computational gain if one considered public key validity check could be pre-computed. However, this would imply reducing the scheme's security to the less standard variant of the GBDH problem used in [1].

Self-Signcryption: We note that, although our security models disallow attacks targeting signcryptions where the sender and receiver identities are the same, it is possible to modify our proof of security to account for this type of attacks. However, one would need less standard versions of the underlying hard problems to construct the security reduction. It is arguable whether this sort of security property is relevant in practice, although specific applications such as protecting one's files or previously sent encrypted e-mails may be used to justify it.

Malicious KGCs: Malicious KGC attacks have not been considered in this paper. However, the proposed scheme withstands the restricted attacks described in [3], which consist of allowing a malicious KGC to generate the $(\mathtt{Msk}, \mathtt{params})$ pair itself as long as it provides these to the challenger. We believe that a more realistic and stronger malicious KGC security model would only require that the adversary outputs the public parameters. We leave it as an open problem to find a certificateless signcryption scheme which can be proven secure in this stronger security model.

7 Acknowledgments

The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. The information in this document reflects only the authors' views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

References

1. S.S. Al-Riyami and K.G. Paterson. Certificateless Public-Key Cryptography. *Advances in Cryptology – ASIACRYPT 2003*, LNCS 2894:452–473. Springer-Verlag, 2003.
2. J.H. An, Y. Dodis and T. Rabin. On the Security of Joint Signature and Encryption. *Advances in Cryptology – EUROCRYPT 2002*, LNCS 2332:83–107. Springer-Verlag, 2002.
3. M.H. Au, J. Chen, J.K. Liu, Y. Mu, D.S. Wong and G. Yang. Malicious KGC Attacks in Certificateless Cryptography. *2007 ACM Symposium on Information, Computer and Communications Security*, pp. 302–311. March 2007.
4. M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. *Proceedings of the First Annual Conference on Computer and Communications Security*, pp. 62–73. 1993.
5. T.E. Bjørstad and A.W. Dent. Building Better Signcryption Schemes with Tag-KEMs. *Public Key Cryptography – PKC 2006*, LNCS 3958:491–507, Springer-Verlag 2006.

6. X. Boyen. Multipurpose Identity-Based Signcryption: A Swiss Army Knife for Identity-Based Cryptography. *Advances in Cryptology – CRYPTO 2003*, LNCS 2729:383–399. Springer-Verlag, 2003.
7. L. Chen and J. Malone-Lee. Improved Identity-Based Signcryption. *Public Key Cryptography – PKC 2005*, LNCS 3386:362–379. Springer-Verlag, 2005.
8. Z. Cheng and R. Comley. Efficient Certificateless Public Key Encryption. *Cryptology ePrint Archive*, Report 2005/012. 2005.
9. J.-S. Coron. On the Exact Security of Full Domain Hash. *Advances in Cryptology – CRYPTO 2000*, LNCS 1880:229–235. Springer-Verlag, 2000.
10. A.W. Dent. A Survey of Certificateless Encryption Schemes and Security Models. *Cryptology ePrint Archive*, Report 2006/211. 2006.
11. A.W. Dent, B. Libert and K.G. Paterson. Certificateless Encryption Schemes Strongly Secure in the Standard Model. *Cryptology ePrint Archive*, Report 2007/121. 2007.
12. B.C. Hu, D.S. Wong, Z. Zhang and X. Deng. Certificateless Signature: A New Security Model and an Improved Generic Construction. *Designs, Codes and Cryptography*, Vol. 42, Issue 2, pp. 109–126. Kluwer Academic Publishers, 2007.
13. X. Huang, W. Susilo, Y. Mu and F. Zhang. On the Security of Certificateless Signature Schemes from Asiacrypt 2003. *Cryptology and Network Security – CANS 2005*, LNCS 3810:13–25. Springer-Verlag, 2005.
14. B. Libert and J.-J. Quisquater. New Identity-Based Signcryption Schemes from Pairings. *IEEE Information Theory Workshop 2003*, pp. 155–158, January 2003.
15. B. Libert and J.-J. Quisquater. On Constructing Certificateless Cryptosystems from Identity-Based Encryption. *Public Key Cryptography 2006 – PKC 2006*, LNCS 3958:474–490. Springer-Verlag, 2006.
16. B. Libert and J.-J. Quisquater. The Exact Security of an Identity Based Signature and its Applications. *Cryptology ePrint Archive*, Report 2004/102. 2004.
17. J. Malone-Lee. Identity-Based Signcryption. *Cryptology ePrint Archive*, Report 2002/098. 2002.
18. A. Shamir. Identity-Based Cryptosystems and Signature Schemes. *Proceedings of CRYPTO '84 on Advances in Cryptology*, LNCS 196:47–53. Springer-Verlag, 1985.
19. Z. Zhang, D.S. Wong, J. Xu and D. Feng. Certificateless Public-Key Signature: Security Model and Efficient Construction. *4th International Conference on Applied Cryptography and Network Security – ACNS 2006*, LNCS 3989:293–308. Springer-Verlag, 2006.
20. Y. Zheng. Digital Signcryption or How to Achieve Cost(Signature & Encryption) $\ll$ Cost(Signature) + Cost(Encryption). *Advances in Cryptology – CRYPTO 97*, LNCS 1294:165–179, Springer-Verlag, 1997.

Appendix A – Proof of Lemma 1

Proof. Let A be an insider Type I adversary against the CLSC scheme. We construct an algorithm B which has non-negligible advantage against the insider Type I′ or the insider Type II security game as follows. Algorithm B receives the parameters from the Type I′ and Type II security games. It flips a coin c as to guess if A will be replacing the public key for $\mathtt{ID}_R^*$ (case $c = 0$) or extract the partial private key for it (case $c = 1$). If $c = 0$ then B passes the parameters from the Type I′ game to A and outputs a random bit in the Type II game, otherwise it passes the parameters from Type II game and returns a random bit in Type I′ game. Let us define the following two events:

– R: the event that A chooses to replace the public key of $\mathtt{ID}_R^*$ in the first stage.
– E: the event that A chooses to extract the partial private key of $\mathtt{ID}_R^*$ at some point.

We now describe how algorithm B answers various queries made by A in each case.

Case $c = 0$: Algorithm B answers the request public key, replace public key, partial private key extraction and decryption oracles using his equivalent oracles. When A outputs two identities $\mathtt{ID}_S^*$ and $\mathtt{ID}_R^*$ and two messages, algorithm B also returns these to its own challenge oracle. The simulation fails if A decides to extract the partial private key of $\mathtt{ID}_R^*$ and not replace its public key (event $\neg R \wedge E$). In this case B outputs a random bit and terminates. The second stage of the game is simulated as in the first case. Note that the simulation in the second stage fails if A ever decides to ask for the partial private key of $\mathtt{ID}_R^*$. This query is allowed if A did not replace the public key of $\mathtt{ID}_R^*$ in the first stage (event $\neg R \wedge E$). When A outputs a bit b', algorithm B also outputs this bit as his own guess.

Case $c = 1$: Algorithm B answers the request public key, replace public key, partial private key extraction and decryption oracles using his equivalent oracles. When A outputs two identities $\mathtt{ID}_S^*$ and $\mathtt{ID}_R^*$ and two messages, algorithm B also returns these to its own challenge oracle. The simulation fails if A decided to replace the public key of $\mathtt{ID}_R^*$ and not extract its partial private key (event $R \wedge \neg E$). In this case B outputs a random bit and terminates. The second stage of the game is simulated as in the first case. Note that the simulation in the second stage is perfect. When A outputs a bit b', algorithm B also outputs this bit as his own guess.

We now analyse the probably that algorithm B returns the correct answer in one of the games it plays. Let $b_{\mathtt{I}'}$ and $b_{\mathtt{II}}$ be the hidden bits in the Type I′ and Type II games respectively. Let also b_1 and b_2 denote the bits B outputs. Note that if the simulation does terminate unexpectedly, c remains hidden from adversary's view. Note also that A is not allowed to provoke the event $R \wedge E$. We have:

$$2\Pr[b_1 = b_{\mathtt{I}'}] = \Pr[b_1 = b_{\mathtt{I}'}|c = 0] + \Pr[b_1 = b_{\mathtt{I}'}|c = 1] = \Pr[b_1 = b_{\mathtt{I}'}|c = 0] + \frac{1}{2}.$$

And similarly:

$$2\Pr[b_2 = b_{\mathtt{II}}] = \Pr[b_2 = b_{\mathtt{II}}|c = 1] + \frac{1}{2}.$$

And hence:

$$\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{IND-iCCA-I}'}(B) = |\Pr[b_1 = b_{\mathtt{I}'}|c = 0] - \frac{1}{2}|,$$
$$\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{IND-iCCA-II}}(B) = |\Pr[b_2 = b_{\mathtt{II}}|c = 1] - \frac{1}{2}|.$$

Now:

$$\begin{aligned}\Pr[b_1 = b_{\mathtt{I}'}|c=0] &= \Pr[b_1 = b_{\mathtt{I}'} \wedge E_1|c=0] + \Pr[b_1 = b_{\mathtt{I}'} \wedge E_2|c=0] + \\ &\quad + \Pr[b_1 = b_{\mathtt{I}'} \wedge E_3|c=0] \\ &= \Pr[b_1 = b_{\mathtt{I}'} \wedge E_1|c=0] + 1/2 + \Pr[b_1 = b_{\mathtt{I}'} \wedge E_3|c=0],\end{aligned}$$

where $E_1 := R \wedge \neg E$, $E_2 := \neg R \wedge E$ and $E_3 := \neg R \wedge \neg E$. Similarly:

$$\Pr[b_2 = b_{\mathtt{II}}|c=1] = 1/2 + \Pr[b_2 = b_{\mathtt{II}} \wedge E_2|c=1] + \Pr[b_2 = b_{\mathtt{II}} \wedge E_3|c=1].$$

Now:

$$\begin{aligned}\Pr[b' = b_{\mathtt{I}}|c=0] &= \Pr[b_1 = b_{\mathtt{I}'} \wedge E_1|c=0] + \Pr[b' = b_{\mathtt{I}'} \wedge E_2|c=0] + \\ &\quad + \Pr[b_1 = b_{\mathtt{I}'} \wedge E_3|c=0].\end{aligned}$$

And similarly:

$$\begin{aligned}\Pr[b' = b_{\mathtt{I}}|c=1] &= \Pr[b' = b_{\mathtt{II}} \wedge E_1|c=1] + \Pr[b_2 = b_{\mathtt{II}} \wedge E_2|c=1] + \\ &\quad + \Pr[b_2 = b_{\mathtt{II}} \wedge E_3|c=1].\end{aligned}$$

Therefore adding up:

$$\begin{aligned}2\Pr[b' = b_{\mathtt{I}}] &= \Pr[b_1 = b_{\mathtt{I}'}|c=0] - \frac{1}{2} + \Pr[b' = b_{\mathtt{I}'} \wedge E_2|c=0] + \\ &\quad + \Pr[b_2 = b_{\mathtt{II}}|c=1] - \frac{1}{2} + \Pr[b' = b_{\mathtt{II}} \wedge E_1|c=1].\end{aligned}$$

Subtracting 1 from both sides, taking absolute signs and using the definitions of advantage we get:

$$\begin{aligned}\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{IND-iCCA-I}}(A) &\le \mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{IND-iCCA-I'}}(B) + \mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{IND-iCCA-II}}(B) + \\ &\quad + |\Pr[b' = b_{\mathtt{I}'} \wedge E_2|c=0] - \frac{1}{2}| + \\ &\quad + |\Pr[b' = b_{\mathtt{II}} \wedge E_1|c=1] - \frac{1}{2}|.\end{aligned}$$

It remains to bound the quantity in absolute signs above. Since c is independent of A's view until the event E_1 or E_2 occurs in each case, we have:

$$\begin{aligned}\Pr[b' = b_{\mathtt{I}} \wedge E_2|c=0] &= \Pr[b' = b_{\mathtt{II}} \wedge E_2|c=1] = \Pr[b_2 = b_{\mathtt{II}} \wedge E_2|c=1] \\ &\le \Pr[b_2 = b_{\mathtt{II}}|c=1], \\ \Pr[b' = b_{\mathtt{I}} \wedge E_1|c=1] &= \Pr[b' = b_{\mathtt{I}'} \wedge E_1|c=0] = \Pr[b_1 = b_{\mathtt{I}'} \wedge E_1|c=0] \\ &\le \Pr[b_1 = b_{\mathtt{I}'}|c=0].\end{aligned}$$

Therefore:

$$\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{IND-iCCA-I}}(A) \le 2\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{IND-iCCA-I'}}(B) + 2\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{IND-iCCA-II}}(B).$$

$\square$