On the Connection Between Signcryption and One-pass Key Establishment*

M. Choudary Gorantla, Colin Boyd and Juan Manuel Gonzàlez Nieto
Information Security Institute, Queensland University of Technology
GPO Box 2434, Brisbane, QLD 4001, Australia.
mc.gorantla@isi.qut.edu.au, {c.boyd,j.gonzaleznieto}@qut.edu.au

November 1, 2007

Abstract

There is an intuitive connection between signcryption and one-pass key establishment. Although this has been observed previously, up to now there has been no formal analysis of this relationship. The main purpose of this paper is to prove that, with appropriate security notions, one-pass key establishment can be used as a signcryption KEM and vice versa. In order to establish the connection we explore the definitions for signcryption (KEM) and give new and generalised definitions. By making our generic construction concrete we are able to provide new examples of a signcryption KEM and a one-pass key establishment protocol.

Keywords. Key establishment, Signcryption, Signcryption KEM.

Contents

1 Introduction **2**
1.1 Related Work 4
1.2 The Canetti–Krawczyk Model 4

2 Security Definitions **6**
2.1 New security notions for signcryption 6
2.2 New security notions for signcryption KEM 9
2.3 On the unforgeability notion for signcryption KEMs 10

3 Outsider secure signcryption KEMs **11**
3.1 ECISS-KEM1 11
3.2 Potential problems with ephemeral data 11
3.3 ECISS-KEM2 12
3.4 ECISS-KEM1 in multi-user setting 13

*To appear at the 11th IMA International Conference on Cryptography and Coding, volume 4887 of LNCS, Springer-Verlag, pp. 277–301, 2007.

4 Signcryption KEM from one-pass key establishment **13**
4.1 New signcryption KEM from the one-pass HMQV 16
4.2 Security of the new KEM 16

5 One-pass key establishment from signcryption KEM **17**

6 Conclusion **20**

A Proof of Theorem 1 **22**

1 Introduction

Zheng [1] introduced the notion of *signcryption* as an asymmetric cryptographic primitive that provides both privacy and authenticity at greater efficiency than the generic composition of signature and encryption schemes. A seemingly unrelated cryptographic primitive is *key establishment* which aims to allow parties to establish a shared key that can be used to cryptographically protect subsequent communications. Most key establishment protocols are interactive, but many such protocols provide simplified *one-pass* versions which only use a single message. One-pass key establishment provides the opportunity for very efficient constructions, even though they will typically provide a lower level of security than interactive protocols.

Zheng [2] later observed that a signcryption scheme can be used as a key transport protocol by simply choosing a new key and sending it in a signcrypted message. This intuitively gives the desired properties for key establishment since the signcryption gives assurance to the sender that the key is available only to the recipient, and assurance to the recipient that the key came from the sender. However, this work contains neither a security model nor a proof for this construction and there remains currently no formal treatment. Since key establishment is notoriously tricky to get right, it is important to decide exactly what security properties such a construction can provide. The main purpose of this paper is to define the appropriate notions of security and show how signcryption and one-pass key establishment can be related under those notions.

Security for signcryption. Since the introduction of signcryption, different definitions of security have emerged. An, Dodis and Rabin [3] divided security notions for signcryption into two types: *outsider security* assumes that the adversary is not one of the participants communicating while *insider security* allows the adversary to be one of the communicating parties. Insider security is a stronger notion than outsider security as it protects the authenticity of a sender from a malicious receiver and privacy of a receiver from a malicious sender. Therefore insider security implies the corresponding notion for outsider security. The notions of outsider security and insider security with respect to authenticity are similar to third-person unforgeability and receiver unforgeability defined for asymmetric authenticated encryption [4].

Because signcryption is intended to include functionality similar to that of digital signatures, it is natural that non-repudiation is a desirable property. Non-repudiation requires insider security since the sender of a signcrypted message must be prevented from showing that it could have been formed by the recipient of that message. A signcryption scheme with only outsider security cannot provide non-repudiation [3, 5]. For key establishment there is no need for non-repudiation — it is never required for a single party to take responsibility for the shared key. On the other hand, a commonly required property for key establishment is *forward secrecy* which ensures that if the long-term key of a participant in the protocol is compromised then previously established keys will

remain secure. We can regard forward secrecy as analogous to insider security in signcryption with respect to confidentiality. Compromise of the sender's private key should not allow an adversary to obtain previously signcrypted messages.

In addition to forward secrecy, another common security requirement for key establishment is security against compromise of ephemeral protocol data. This is not considered in the existing models for signcryption schemes and so it is not possible, in general, to convert from a signcryption scheme to a key establishment protocol with this stronger security notion. We will argue later that there is good reason that signcryption schemes should consider security against compromise of ephemeral data. In particular this observation allows us to explain a potential weakness observed by Dent in one of his own constructions [6].

Signcryption KEMs. Cramer and Shoup [7] formalised the concept of hybrid encryption schemes which securely use public key encryption techniques to encrypt a session key, and symmetric key encryption techniques to encrypt the actual message. This hybrid construction has a key encapsulation mechanism (KEM) and a data encapsulation mechanism (DEM) as its underlying tools. A KEM is similar to a public key encryption scheme except that it is used to generate a random key and its encryption. In a series of papers, Dent [8, 6, 9] extended this hybrid paradigm to signcryption, resulting in the construction of signcryption KEM and signcryption DEM with different security notions.

Although we could use plain signcryption schemes to provide one-pass key establishment as suggested by Zheng [2], signcryption KEMs seem better suited to the job. This is because they do just what we need by providing a new random key, yet have the potential to be more efficient than plain signcryption. Note, however, that the remarks regarding the differences in security models between signcryption and key establishment apply equally when signcryption KEMs are considered in place of plain signcryption.

Contributions. We provide new definitions of security for signcryption and subsequently for signcryption KEMs. We then show the suitability of these new notions in deriving one-pass key establishment protocols from signcryption KEMs. Generic constructions of signcryption KEMs from one-pass key establishment protocols and vice versa are proposed. These constructions are instantiated using existing schemes; in particular we use HMQV [10], the most efficient currently known key agreement protocol with a proof of security, to derive a new signcryption KEM with strong security properties. One of the main observations of our paper is that the security models for key establishment are stronger than those normally accepted for signcryption. Moreover, the stronger security seems to be just as appropriate for signcryption as it is for key establishment. Specific contributions of the paper are:

- new definitions for signcryption (KEM)s;
- generic construction from one-pass key establishment to signcryption and vice versa;
- the first secure signcryption KEM with forward secrecy;
- an attack on a signcryption KEM of Dent [6].

The remainder of this introduction briefly surveys related work and outlines the Canetti–Krawczyk model for key establishment. Section 2 then considers the current definitions of security for signcryption and how they can be strengthened. Section 3 examines the outsider secure signcryption KEMs designed by Dent [6]. The generic construction of signcryption KEM from one-pass key establishment is covered in Section 4 while the reverse construction is covered in Section 5.

1.1 Related Work

An et al. [3] defined security notions for signcryption schemes as insider and outsider security in the two-user setting. They also described how to extend these notions to the multi-user setting. Baek et al. [11] independently attempted to provide similar notion of security for signcryption, but their model was not completely adequate. Recently, the same authors [12] extended these notions to match the corresponding definitions given by An et al. However, their security notions are still not complete, as discussed in Section 2.1.

Zheng [2] informally showed how a signcryption scheme can be used as a key transport protocol. Dent [8, 6] and Bjørstad and Dent [13] discussed how a signcryption KEM can be used as a one-pass key establishment protocol. Bjørstad and Dent [13] proposed the concept of signcryption tag-KEM and claimed that better key establishment mechanisms can be built with this. However, none of these papers formally defined security in a model that is suitable for key establishment protocols. Moreover, the confidentiality notion defined for all these KEMs does not offer security against insider attacks. Insider security for confidentiality enables achieving *forward secrecy* i.e. the compromise of the sender's private key does not compromise the confidentiality of signcryptions created using that key [3]. However, this has been ignored or its absence is treated as a positive feature called "Past Message Recovery" in the earlier work [1, 12].

1.2 The Canetti–Krawczyk Model

To analyse the security of key establishment protocols, we use the Canetti-Krawczyk (CK) model [14, 15], which we briefly describe here. The CK model has primarily been used for multi-pass key establishment protocols, but it can also be used to analyse one-pass key establishment protocols without modification as shown by Krawczyk [10].

In the CK model a protocol π is modelled as a collection of n programs running at different parties, $P_1, \ldots, P_n$. Each invocation of π within a party is defined as a *session*, and each party may have multiple sessions running concurrently. The communications network is controlled by an adversary $\mathcal{A}^{\pi}$, which schedules and mediates all sessions between the parties. When first invoked within a party, π calls an initialization function that returns any information needed for the bootstrapping of the cryptographic authentication functions (e.g. private keys and authentic distribution of other parties' public keys). After this initialization stage, the party waits for activation. $\mathcal{A}^{\pi}$ may activate a party P_i by means of a $\mathsf{send}(\pi^s_{i,j}, \lambda)$ request where λ is an empty message[1]. This request instructs P_i to commence a session with party P_j. In response to this request P_i outputs a message m intended for party P_j. s is a session identifier unique amongst all sessions between P_i and P_j. In this paper, where we only consider one-pass key establishment, we define the session-id as the tuple (P_i, P_j, m), where m is the unique message sent by P_i to P_j. The adversary activates the receiver P_j with an incoming message using the request $\mathsf{send}(\pi^s_{j,i}, m)$.

$\mathcal{A}^{\pi}$ is responsible for transmitting messages between parties, and may fabricate or modify messages when desired. Upon activation, the parties perform some computations and update their internal state. Two sessions are said to be *matching sessions* if their session-ids are identical.

In addition to the activation of parties, $\mathcal{A}^{\pi}$ can perform the following queries.

1. $\mathsf{corrupt}(P_i)$. With this query $\mathcal{A}^{\pi}$ learns the entire current state of P_i including long-term secrets, session internal state and unexpired session keys. From this point on, $\mathcal{A}^{\pi}$ may issue

[1] Here we use the notation of Bellare and Rogaway [16] rather than the original notation of CK model, which uses a query named establish-session to achieve the same result.

any message in which P_i is specified as the sender and play the role of P_i. The adversary is allowed to replace the public key of a corrupted user by any value of its choice.

2. session-key($\pi^s_{i,j}$). This query returns the unexpired session key (if any) accepted by P_i during a given session s with P_j.

3. session-state($\pi^s_{i,j}$). This query returns all the internal state information of party P_i associated to a particular session s with P_j; the state information does not include the long term private key.

4. session-expiration($\pi^s_{i,j}$). This query can only be performed on a completed session. It is used for defining *forward secrecy* and ensures that the corresponding session key is erased from P_i's memory. The session is thereafter said to be expired;

5. test-session($\pi^s_{i,j}$). To respond to this query, a random bit b is selected. If $b = 0$ then the session key is output. Otherwise, a random key is output chosen from the probability distribution of keys generated by the protocol. This query can only be issued to a session that has not been *exposed*. A session is exposed if the adversary performs any of the following actions:

   - a session-state or session-key query to this session or to the matching session, or
   - a corrupt query to either partner before the session expires at that partner.

Security is defined based on a game played by the adversary. In this game $\mathcal{A}^\pi$ interacts with the protocol. In the first phase of the game, $\mathcal{A}^\pi$ is allowed to activate sessions and perform corrupt, session-key, session-state and session-expiration queries as described above. The adversary then performs a test-session query to a party and session of its choice. The adversary is not allowed to expose the test-session. $\mathcal{A}^\pi$ may then continue with its regular actions with the exception that no other test-session query can be issued. Eventually, $\mathcal{A}^\pi$ outputs a bit b' as its guess on whether the returned value to the test-session query was the session key or a random value, then halts. $\mathcal{A}^\pi$ wins the game if $b = b'$. The definition of security is as follows.

Definition 1. *A key establishment protocol π is called* session key (SK-) secure *with* forward secrecy *if the following properties are satisfied for any adversary $\mathcal{A}^\pi$.*

1. *If two uncorrupted parties complete matching sessions then they both output the same key.*

2. *The probability that $\mathcal{A}^\pi$ guesses correctly the bit b is no more than $\frac{1}{2}$ plus a negligible function in the security parameter.*

We define the advantage of $\mathcal{A}^\pi$ to be twice the probability that $\mathcal{A}^\pi$ wins, minus one. Hence the second requirement will be met if the advantage of $\mathcal{A}^\pi$ is negligible.

As discussed by Krawczyk [10] it is impossible to achieve forward secrecy in less than three rounds with any protocol authenticated via public keys and without previously established shared state between the parties. With one-pass key establishment, using public keys and with no previous shared secret state, the best that can be achieved is *sender forward secrecy*, whose definition is the same as above, except that sessions can only be expired at the sender. Canetti and Krawczyk also provide a definition of *SK-security without forward secrecy*, where the adversary is not allowed to expire sessions at all.

As mentioned above, for the one-pass key establishment protocols in this paper we define the session-id to be the concatenation of the identities of the peers and the unique message sent in that session. This formulation of session-id prevents the adversary from replaying the message from one protocol in a different session since the model insists that each session has a unique session-id. This may be seen as an artificial way of preventing replay attacks which are an inherent limitation of one-pass key establishment protocols: in reality an adversary can simply replay the single message of the protocol and it will be accepted by the recipient unless it includes some time-varying value. This situation could be addressed by including a time-stamp to uniquely identify a session, and assuming that all parties have access to a universal time oracle [17]. We do not explore that approach further in this paper.

2 Security Definitions

We now present definitions of security for signcryption schemes that complement those of Baek et al. [12]. We then extend these definitions to arrive at new notions of security for signcryption KEMs in the multi-user setting.

2.1 New security notions for signcryption

A signcryption scheme $\mathcal{SC}$ is specified by five polynomial-time algorithms: common-key-gen, sender-key-gen, receiver-key-gen, signcryption and unsigncryption.

common-key-gen: is a probabilistic polynomial time (PPT) algorithm that takes the security parameter k as input and outputs the common/public parameters params used in the scheme. These parameters include description of the underlying groups and hash functions used.

sender-key-gen: is a PPT algorithm that takes params as input and outputs the sender's public-private key pair (pk_s, sk_s) used for signcryption.

receiver-key-gen: is a PPT algorithm that takes params as input and outputs the receiver's public-private key pair (pk_r, sk_r) used for unsigncryption.

signcryption: is a PPT algorithm that takes params, a sender's private key sk_s, a receiver's public key pk_r and message m to be signcrypted as input. It returns a signcryptext C.

unsigncryption: is a deterministic polynomial-time algorithm that takes params, a sender's public key pk_s, a receiver's private key sk_r and a signcryptext C as input. It outputs either a plaintext m or an error symbol $\perp$.

For $\mathcal{SC}$ to be considered valid it is required that $\mathsf{unsigncryption}(pk_s, sk_r, \mathsf{signcryption}(sk_s, pk_r, m)) = m$ for all sender key pairs (pk_s, sk_s) and receiver key pairs (pk_r, sk_r).

For all the security notions defined in this paper we distinguish two users Alice and Bob as sender and receiver respectively. Depending on the notion of security one of these users, or both, will be adversary's target.

We define insider and outsider security for signcryption schemes in the multi-user setting based on the discussion given by An et al. [3]. It is natural to consider the security of a signcryption scheme in the multi-user setting where Alice can signcrypt messages for any user including Bob, and any user including Alice can signcrypt messages for Bob. Hence, in this model the adversary

is given the power to obtain signcryptions of Alice created for any user through a flexible signcryption oracle (FSO). Similarly, the adversary is also given access to a flexible unsigncryption oracle (FUO) that unsigncrypts a given signcryptext created for Bob by any user. Because of these additional adversarial powers, security in the two-user setting does not imply security in the multi-user setting [12].

In any signcryption scheme users employ a private key for two purposes: for signcrypting messages sent to other users and for unsigncrypting messages received from other users. The private keys used for each of these purposes, along with their corresponding public keys, may be the same or they may be different. This is much the same as the option to use the same, or different, keys for decryption and for signing. In keeping with common practice we will assume that each key pair is used for a single purpose. Therefore we specify that all users have two different key pairs for signcryption and for unsigncryption. In contrast, An et al. [3] assumed that a single key pair is used for both signcryption and unsigncryption. Their security model therefore requires giving the adversary access to additional oracles for unsigncrypting by Alice and signcrypting by Bob. To the same end, our model allows the receiver key pair of Alice and sender key pair of Bob to be given to the adversary.

Baek et al. [12] defined only outsider security for confidentiality and insider security for unforgeability in the multi-user setting. In this section we make the desired security notions for signcryption complete by defining insider security for confidentiality and outsider security for unforgeability in the multi-user setting. We assume that the challenger fixes the set of users $\{U_1, \ldots, U_n\}$ and their key pairs before its interaction with the adversary[2]. Note that the security notions defined by Baek et al. [12] implicitly assume the same. The adversary is given all the public keys of the users initially so that it can choose the public keys from the given set when accessing the oracles. One can relax this restriction in our definitions by allowing the adversary to query the FSO and/or FUO with arbitrarily chosen public keys.

Let (pk_A, sk_A) be the public-private key pair used by Alice for signcryption and let (pk_B, sk_B) be the public-private key pair used by Bob for unsigncryption. The behaviour of Alice's FSO and Bob's FUO is described below:

FSO: On the input (pk_r, m), FSO returns a signcryptext C generated using sk_A and the public key pk_r on the message m. The adversary may choose Bob's public key pk_B as the receiver public key, i.e., $pk_r = pk_B$.

FUO: On the input (pk_s, C), FUO returns a plaintext m or a $\perp$ symbol after performing unsigncryption on C using sk_B and the public key pk_s. The adversary may choose pk_s to be the public key of Alice i.e. $pk_s = pk_A$.

For the sake of completeness, we first present our new definition for insider confidentiality and then briefly describe the definition for outsider confidentiality given by Baek et al. [12]. Similarly, we present our new definition for outsider unforgeability and then strengthen the definition for insider unforgeability given by Baek et al. [12].

Insider confidentiality An insider adversary $\mathcal{A}^{CCA}$ against confidentiality of $\mathcal{SC}$ is assumed to have knowledge of all key pairs except Bob's private key used for unsigncryption. The goal of

[2]This can be seen as a way of preventing the adversary from registering replaced public keys with the certifying authority for a particular user.

$\mathcal{A}^{CCA}$ is to break the confidentiality of messages signcrypted for Bob by any user. $\mathcal{A}^{CCA}$ can issue FUO(pk_s, C) for any sender's public key pk_s and a signcryptext C. Alice's FSO can be simulated by $\mathcal{A}^{CCA}$ itself with its knowledge of the corresponding private key. We define this security notion as FUO-IND-CCA2 that simulates chosen ciphertext attacks against $\mathcal{SC}$.

- *Challenge Phase:* After adaptively asking the FUO queries, $\mathcal{A}^{CCA}$ outputs two equal length messages m_0, m_1 and a public key $pk_{s'}$ and submits them to the challenger. The challenger chooses $b \in_R \{0, 1\}$ and gives $\mathcal{A}^{CCA}$ a challenge signcryptext C^* created on m_b using the private key $sk_{s'}$ corresponding to $pk_{s'}$ and Bob's public key pk_B.

$\mathcal{A}^{CCA}$ can continue asking the FUO queries except the trivial FUO$(pk_{s'}, C^*)$. However, an FUO query on C^* using a public key $pk_s \neq pk_{s'}$ is still allowed.

- *Guess Phase:* Finally, $\mathcal{A}^{CCA}$ outputs a bit b' and wins the game if $b' = b$.

The advantage of $\mathcal{A}^{CCA}$ in winning the FUO-IND-CCA2 game is:

$$Adv_{\mathcal{A}^{CCA},\mathcal{SC}} \stackrel{def}{=} 2 \cdot \Pr[b' = b] - 1$$

Outsider confidentiality An adversary against outsider confidentiality of $\mathcal{SC}$ is assumed to know all private keys except Alice's private key used for signcryption and Bob's private key used for unsigncryption. The goal of the adversary in this notion is to break the confidentiality of messages signcrypted by Alice for Bob. The outsider confidentiality notion FSO/FUO-IND-CCA2 [12] gives the adversary access to both FSO and FUO. After adaptively asking the FSO and FDO queries, the challenge and guess phases are carried on as described above. The advantage of the adversary in winning the FSO/FUO-IND-CCA2 game is also defined in the same way as above.

Outsider unforgeability An outsider adversary $\mathcal{A}^{CMA}$ against unforgeability of $\mathcal{SC}$ is assumed to know all private keys except Alice's private key used for signcryption and Bob's private used for unsigncryption. The goal of $\mathcal{A}^{CMA}$ is to forge a valid signcryptext created by Alice for Bob. $\mathcal{A}^{CMA}$ is given access to both FSO and FUO.

After querying the FSO and FUO adaptively, $\mathcal{A}^{CMA}$ outputs a forgery C^*. A weak (conventional) notion of unforgeability requires C^* to be a valid signcryption created by Alice for Bob on a new message m^* i.e. m^* was never queried to FSO. For the notion of strong unforgeability C^* has to be a valid signcryption of Alice created for Bob on a message m^* such that C^* was never an output of FSO, although FSO(m^*, pk_r) might have been issued earlier, even for $pk_r = pk_B$. We call this notion FSO/FUO-sUF-CMA for strong unforgeability against chosen message attacks by $\mathcal{A}^{CMA}$. The advantage of $\mathcal{A}^{CMA}$ in winning the FSO/FUO-sUF-CMA game is the probability of $\mathcal{A}^{CMA}$ outputting such a C^*.

Insider unforgeability An adversary against insider unforgeability of $\mathcal{SC}$ is assumed to know all the private keys except Alice's private key used for signcryption. The goal of adversary in this notion is to produce a valid forgery of a signcryptext created by Alice for any other user. The insider unforgeability notion FSO-UF-CMA [12] gives the adversary access to FSO, as FUO can be simulated by the adversary itself. After querying the FSO adaptively, the adversary outputs a forgery (C^*, pk_r^*) for any receiver's public key pk_r^*. The FSO-UF-CMA notion does not require the message m^* to be new although only FSO(m^*, pk_r), for $pk_r \neq pk_r^*$ queries are allowed. We strengthen this notion to FSO-sUF-CMA by allowing FSO(m^*, pk_r) even for $pk_r = pk_r^*$.

2.2 New security notions for signcryption KEM

A signcryption KEM $\mathcal{SK}$ is specified by five polynomial-time algorithms: common-key-gen, sender-key-gen, receiver-key-gen, encapsulation and decapsulation. The algorithms common-key-gen, sender-key-gen and receiver-key-gen are the same as those defined in Section 2.1. The sender's key pair (pk_s, sk_s) and the receiver's key pair (pk_r, sk_r) are now used for encapsulation and decapsulation respectively.

encapsulation: is a PPT algorithm that takes params, a sender's private key sk_s and a receiver's public key pk_r as input. It returns the pair (K, C), where, K is a symmetric key and C is its encapsulation.

decapsulation: is a deterministic polynomial-time algorithm that takes params, a sender's public key pk_s, a receiver's private key sk_r and an encapsulation C. It outputs either a symmetric key K or an error symbol $\perp$.

For $\mathcal{SK}$ to be valid it is required that if $(K, C) = \mathsf{encapsulation}(sk_s, pk_r)$, then $\mathsf{decapsulation}(pk_s, sk_r, C) = K$ for all sender key pairs (pk_s, sk_s) and receiver key pairs (pk_r, sk_r).

Dent [8, 6, 9] defined both insider and outsider security notions for signcryption KEMs in the two-user setting. He also provided an informal description of how to define security for signcryption KEMs in the multi-user setting [8]. Recently, Yoshida and Fujiwara [18] defined security notion for signcryption Tag-KEMs in the multi-user setting. Here, we present new notions of security for signcryption KEMs in the multi-user setting building on the definitions in Section 2.1.

In the security model for a signcryption KEM in the multi-user setting the adversary is given the power to obtain encapsulations of Alice created for any user through a flexible encapsulation oracle (FEO). The adversary is also given access to a flexible decapsulation oracle (FDO) that decapsulates a given encapsulation created for Bob by any user. Let (pk_A, sk_A) be the public-private key pair used by Alice for encapsulation and let (pk_B, sk_B) be the public-private key pair used by Bob for decapsulation.

The challenger initially fixes the set of users $\{U_1, \ldots, U_n\}$ and their key pairs. The adversary is given all the public keys of the users initially so that it can choose the public keys from the given set when accessing the oracles. The behaviour of Alice's FEO and Bob's FDO is described below.

FEO: On receiving pk_r, FEO returns a pair (K, C), where C is an encapsulation of K generated using sk_A and pk_r. The adversary may choose pk_B as the receiver's public key, i.e., $pk_r = pk_B$.

FDO: On receiving (pk_s, C), FDO returns a symmetric key K or a $\perp$ symbol after performing decapsulation on C using sk_B and pk_s. The adversary may choose pk_A as the sender's public key i.e. $pk_s = pk_A$.

Insider confidentiality An insider adversary $\mathcal{A}^{CCA}$ against confidentiality of $\mathcal{SK}$ is assumed to have knowledge of all key pairs except Bob's private key used for decapsulation. The goal of $\mathcal{A}^{CCA}$ is to break the confidentiality of encapsulations created for Bob by any user. It is given access only to FDO as the oracle FEO can be simulated with the knowledge of Alice's private key used for encapsulation. We call this notion of security FDO-IND-CCA2.

- *Challenge Phase:* After adaptively asking the FDO queries, $\mathcal{A}^{CCA}$ outputs a public key $pk_{s'}$. The challenger generates a valid symmetric key, encapsulation pair (K_0, C^*) using the private

key $sk_{s'}$ corresponding to $pk_{s'}$ and Bob's public key pk_B. It selects a key K_1 randomly from the symmetric key distribution. It then chooses $b \in_R \{0,1\}$ and gives (K_b, C^*) as the challenge.

$\mathcal{A}^{CCA}$ can continue its execution except asking the FDO$(pk_{s'}, C^*)$ query that trivially decides the guess. However, an FDO query on C^* using a public key $pk_s \neq pk_{s'}$ is still allowed.

- *Guess Phase:* Finally, $\mathcal{A}^{CCA}$ outputs a bit b' and wins the game if $b' = b$.

The advantage of $\mathcal{A}^{CCA}$ in winning the FDO-IND-CCA2 game is

$$Adv_{\mathcal{A}^{CCA},\mathcal{SK}} \stackrel{def}{=} 2 \cdot \Pr[b' = b] - 1$$

Outsider confidentiality For outsider confidentiality the adversary is assumed to know all the private keys except Alice's private key used for encapsulation and Bob's private key used for decapsulation. The goal of the adversary in this notion is to break the confidentiality of encapsulations created by Alice for Bob. The adversary must be given access to both FEO and FDO. We call this notion FEO/FDO-IND-CCA2. After adaptively asking the FEO and FDO queries, the challenge and guess phases are carried on as described above. The advantage of an adversary in winning the FEO/FDO-IND-CCA2 game is also defined in the same way as above.

Outsider unforgeability An outsider adversary $\mathcal{A}^{CMA}$ against unforgeability of $\mathcal{SK}$ is assumed to know all private keys except Alice's private key used for encapsulation and Bob's private used for decapsulation. The goal of $\mathcal{A}^{CMA}$ is to forge a valid symmetric key and encapsulation pair (K^*, C^*) such that C^* is an encapsulation of K^* created by Alice for Bob. It is given access to both FEO and FDO.

After querying FEO and FDO adaptively, $\mathcal{A}^{CMA}$ produces a forgery (K^*, C^*). It wins the game if $\mathsf{decapsulation}(pk_A, sk_B, C^*) = K^* \neq \perp$. The trivial restriction for (K^*, C^*) to be considered valid is that (K^*, C^*) was never an output of FEO. We call this notion FEO/FDO-sUF-CMA for strong unforgeability against outsider attacks. The advantage of $\mathcal{A}^{CMA}$ in winning the FEO/FDO-sUF-CMA game is the probability of $\mathcal{A}^{CMA}$ outputting such a (K^*, C^*).

Insider unforgeability An insider adversary against unforgeability of $\mathcal{SK}$ is assumed to know all the private keys except Alice's private key used for encapsulation. The goal of the adversary in this notion is to forge a valid encapsulation created by Alice to any other user. It is given access only to FEO as all keys for decapsulation are known to the adversary. We call this notion FEO-sUF-CMA for strong unforgeability against insider attacks. After adaptively querying the FEO, the adversary outputs a forgery (K^*, C^*, pk_r^*). It wins the FEO-sUF-CMA game if $\mathsf{decapsulation}(pk_A, sk_r^*, C^*) = K^* \neq \perp$. The trivial restriction for (K^*, C^*, pk_r^*) to be considered valid is that (K^*, C^*) was never an output of FEO. The advantage of the adversary in winning the FEO-sUF-CMA game is the probability of outputting such a (K^*, C^*, pk_r^*).

2.3 On the unforgeability notion for signcryption KEMs

Dent [6] defined a different notion, called Left-or-Right (LoR) security, for outsider unforgeability of signcryption KEMs. He showed that an adversary that can output a valid forgery (K^*, C^*) under the notion described in the previous section can be efficiently turned into another adversary that

can win the LoR game. Dent pointed out that LoR security is a strong requirement for outsider secure signcryption KEMs. An outsider secure signcryption KEM under our definition can be combined with an outsider secure signcryption DEM in the notion defined by Dent [6] to achieve an outsider secure hybrid signcryption scheme. However, this composition may not yield a tight reduction when compared to the hybrid signcryption scheme that is composed of an LoR secure signcryption KEM and an outsider secure signcryption DEM. In our generic constructions we show that our notion FEO/FDO-sUF-CMA is enough when relating one-pass key establishment protocols with signcryption KEMs. One may still use an LoR secure signcryption KEM to derive a one-pass key establishment protocol, as LoR security guarantees security under the FEO/FDO-sUF-CMA notion.

We emphasise that an insider secure hybrid signcryption scheme can never be guaranteed using the definition of insider unforgeability for signcryption KEM described in the previous section. Dent [9] described the impossibility of achieving an insider secure hybrid signcryption scheme by generic composition of such a signcryption KEM and an insider secure signcryption DEM. The difficulty is that a signcryption KEM that generates symmetric keys and encapsulations independent of the message to be signcrypted cannot provide the non-repudiation service and thus cannot be insider secure. But our definitions of security for insider security of signcryption KEMs are still useful when one observes their connection with one-pass key establishment protocols. A signcryption KEM that is insider confidential in our definition can be used to derive a one-pass key establishment protocol that provides sender forward secrecy. Similarly, we speculate that a signcryption KEM that is insider unforgeable in our definition can be turned into a one-pass key establishment protocol that provides resistance to key compromise impersonation (KCI) attacks even when the receiver's private key is compromised. The latter observation is yet to be explored formally.

3 Outsider secure signcryption KEMs

Dent [6] proposed an outsider secure signcryption KEM called elliptic curve integrated signcryption scheme KEM (ECISS-KEM1) based on the ECIES-KEM [19]. A potential problem with the ECISS-KEM1 was identified by Dent who then proposed an improvement that is claimed to overcome this problem, but without a proof of security. Both the schemes are described below.

3.1 ECISS-KEM1

Let (G, P, q) be the system parameters, where G is a large additive cyclic group of prime order q and P is an arbitrary generator of G. Let $(P_s = sP, s)$ and $(P_r = rP, r)$ be the public-private key pairs of the sender and receiver respectively, where $s, r \in_R Z_q^*$. ECISS-KEM1 is described in Figure 1. The scheme uses a hash function that outputs a key of desired length. ECISS-KEM1 is proven secure by Dent in the two-user setting against outsider security (for both confidentiality and integrity) assuming the hardness of CDH problem.

3.2 Potential problems with ephemeral data

Dent discussed a potential weakness with the scheme ECISS-KEM1 as follows. If an attacker is ever to obtain $sP_r + tP$ (through a temporary break-in), the component sP_r can be recovered easily. This means that the adversary can indefinitely impersonate the sender.

- Encapsulation
  1. Choose an element $t \in_R Z_q^*$
  2. Set $K = \text{Hash}(sP_r + tP)$
  3. Set $C = tP$
  4. Output (K, C)

- Decapsulation
  1. Set $K = \text{Hash}(rP_s + C)$
  2. Output K

Figure 1: ECISS-KEM1

It is interesting that Dent identified this as a problem even though it is not recognised as one by any of the current security models for signcryption. On the other hand, we can see that this capability of the adversary to obtain ephemeral protocol data has already been known in the key establishment models for many years. If the KEM is used to build a one-pass key agreement protocol, then in the Canetti–Krawczyk model described in Section 1.2 the session-state query allows the adversary to obtain $sP_r + tP$ and hence break the protocol.

We would argue that the plausibility of such an attack, as well as its consequences, are equally valid for a signcryption KEM as for a key establishment protocol. Therefore we suggest that session-state queries can be usefully added to the signcryption security model to give a useful stronger notion of security. The feasibility of session-state queries will certainly vary according to the application scenario. Factors that might influence the feasibility include the security of storage during processing and the quality of practical random number generators. It may be argued that applications such as signcryption or one-pass key establishment are of limited vulnerability to session-state queries since local values can be erased immediately once they are used. In contrast, two-pass protocols often require some ephemeral values to be stored until interaction with a protocol peer are completed.

3.3 ECISS-KEM2

Having recognised the problem with ECISS-KEM1, Dent proposed another signcryption KEM (ECISS-KEM2). The system parameters, key pairs and hash function are the same as those in ECISS-KEM1. The symmetric key in the encapsulation algorithm of ECISS-KEM2 is computed as $K = \text{Hash}(sP_r + tP_r)$ and its encapsulation is $C = tP$. Given an encapsulation, the symmetric key can be recovered using the deterministic decapsulation algorithm as $K = \text{Hash}(rP_s + rC)$.

Dent argued that even if an attacker discovers the value $sP_r + tP_r$, it would help in recovering only a single message for which the hashed material is used to produce the symmetric key. This is because it is not easy to compute sP_r from the discovered value and C. Although the security of the scheme is stated informally, Dent claimed that a proof can be given with a non-standard security assumption. However, the attack below enables an active adversary to impersonate the sender to any receiver indefinitely.

Attack on ECISS-KEM2 An active adversary calculates C^* as $P - P_s$ and sends it to the receiver as a message from a sender with the public key P_s. This forces the receiver to compute

- Encapsulation
  1. Choose $C \in_R G$
  2. Set $K = \text{Hash}(\hat{S}, \hat{R}, sP_r + C)$
  3. Output (K, C)
- Decapsulation
  1. Set $K = \text{Hash}(\hat{S}, \hat{R}, rP_s + C)$
  2. Output K

$\hat{S}$, $\hat{R}$ represent the identities of the sender S and receiver R respectively.

Figure 2: ECISS-KEM1 in the multi-user setting

shared key as $K = \text{Hash}(rP_s + rC^*) = \text{Hash}(rsP + r(P - sP)) = \text{Hash}(rsP + rP - rsP) = \text{Hash}(P_r)$, which can easily be computed by the adversary. Now, the adversary can use a DEM along with ECISS-KEM2 and *signcrypt* messages as having come from the original sender. The attack is possible because the random element chosen in the encapsulation algorithm and the static private key of the sender are not combined in a way that makes eliminating them from the hashed material difficult. This attack directly violates the Left or Right security defined by Dent for outsider unforgeability.

3.4 ECISS-KEM1 in multi-user setting

We slightly modify ECISS-KEM1 to work in a multi-user environment as shown in Figure 2. This new version has the same potential problems described in Section 3.2. As suggested by An et al. [3], the identities of the users are now embedded in the encapsulation and decapsulation processes.

Theorem 1. *ECISS-KEM1 in the multi-user setting is secure in the outsider unforgeability notion in the random oracle model assuming hardness of the Gap Diffie Hellman (GDH) problem in G.*

Theorem 2. *ECISS-KEM1 in the multi-user setting is secure in the outsider confidentiality notion in the random oracle model assuming hardness of the GDH problem in G.*

The proof of Theorem 1 is provided in Appendix A. The proof of Theorem 2 is very similar to that of Theorem 1 and we omit the details.

4 Signcryption KEM from one-pass key establishment

This section first discusses how a one-pass key establishment protocol π can be used as a signcryption KEM $\mathcal{SK}$. A proof of the generic construction is also provided.

The session key computed by a sender in π can be used as the symmetric key of $\mathcal{SK}$. The outgoing message of π becomes the encapsulation of the key. The session key computation process at the receiver end in π can be used as the decapsulation algorithm to retrieve the symmetric key.

Theorem 3. *If π is a one-pass key establishment protocol SK-secure with sender forward secrecy in the CK model, then it can be used as a signcryption KEM that is secure in the insider confidentiality and outsider unforgeability notions.*

Proof. To prove the theorem it is enough if we show that if $\mathcal{SK}$ is not secure in the insider confidentiality **or** outsider unforgeability notion, then π is also not secure in the CK model. Given an adversary $\mathcal{A}^{CCA}$ against insider confidentiality or $\mathcal{A}^{CMA}$ against outsider unforgeability with non-negligible advantage, we construct an adversary $\mathcal{A}^{\pi}$ against SK-security of π in the CK model that can distinguish a real session key from a random number in polynomial time.

Constructing $\mathcal{A}^{\pi}$ from $\mathcal{A}^{CMA}$: We start by assuming the existence of $\mathcal{A}^{CMA}$ against outsider unforgeability, with a non-negligible advantage ϵ_u. Then, we prove that $\mathcal{A}^{\pi}$ can distinguish a real session key from a random value with the same advantage using $\mathcal{A}^{CMA}$ as subroutine. The running time of $\mathcal{A}^{\pi}$ is $t_1 \leq t_u + (n_{feo} + n_{fdo})(t_s + t_k)$, where t_u is the time required for $\mathcal{A}^{CMA}$ to forge $\mathcal{SK}$, n_{feo} and n_{fdo} are the number of FEO and FDO queries issued by $\mathcal{A}^{CMA}$ respectively and t_s and t_k are the response times for send and session-key queries respectively. The view of $\mathcal{A}^{CMA}$ is simulated as below:

$\mathcal{A}^{\pi}$ allows $\mathcal{A}^{CMA}$ to choose two users U_A and U_B from a set of users $\{U_1, \ldots, U_n\}$. It then corrupts all the parties except U_A and U_B and gives their key pairs to $\mathcal{A}^{CMA}$. This enables $\mathcal{A}^{CMA}$ to choose public keys from the given set when accessing U_A's FEO and U_B's FDO. The queries asked by $\mathcal{A}^{CMA}$ are answered as below:

- *FEO Queries:* For an FEO query asked by $\mathcal{A}^{CMA}$ with input pk_j, the adversary $\mathcal{A}^{\pi}$ initiates a session by issuing a $\mathsf{send}(\pi^s_{A,j}, \lambda)$ query and obtains the parameter C computed by the oracle $\pi^s_{A,j}$. It then issues a $\mathsf{session\text{-}key}(\pi^s_{A,j})$ query to get the session key K accepted in that session. The pair (K, C) is returned to $\mathcal{A}^{CMA}$.
- *FDO Queries:* On an FDO query with the input (pk_i, C), $\mathcal{A}^{\pi}$ issues a $\mathsf{send}(\pi^s_{B,i}, C)$. It then issues a $\mathsf{session\text{-}key}(\pi^s_{B,i})$ query and returns the accepted session key K to $\mathcal{A}^{CMA}$. It returns a $\perp$ symbol if there is no key accepted in the session.

Answering the challenger: $\mathcal{A}^{CMA}$ finally outputs a forgery (K^*, C^*) such that C^* is a valid encapsulation of K^* created by U_A for U_B. $\mathcal{A}^{\pi}$ now establishes a fresh session between U_A and U_B, by issuing $\mathsf{send}(\pi^t_{B,A}, C^*)$ and chooses it as the test session. The challenger computes the session key K_0 of the test session and selects a random value K_1 from session key distribution. It then chooses $b \in_R \{0, 1\}$ and gives K_b to $\mathcal{A}^{\pi}$. $\mathcal{A}^{\pi}$ outputs its guess as 0 if $K_b = K^*$ or 1 otherwise.

For $\mathcal{A}^{CMA}$ to be successful it has to forge a valid encapsulation of U_A created for U_B i.e. between any two users that were chosen initially. As explained above, $\mathcal{A}^{\pi}$ always wins whenever $\mathcal{A}^{CMA}$ outputs such a forgery by establishing a test session between those two users. Hence, the advantage of $\mathcal{A}^{\pi}$ constructed from $\mathcal{A}^{CMA}$ is

$$Adv^{\pi}_1(k) = \epsilon_u \tag{1}$$

For each FEO or FDO query, $\mathcal{A}^{\pi}$ has to establish a session through a send query and retrieve the session key through a session-key query. Hence, the running time of $\mathcal{A}^{\pi}$ is bounded by $t_1 \leq t_u + (n_{feo} + n_{fdo})(t_s + t_k)$.

Constructing $\mathcal{A}^{\pi}$ from $\mathcal{A}^{CCA}$: Now, we assume that there exists $\mathcal{A}^{CCA}$ against insider confidentiality with a non-negligible advantage ϵ_c. Using $\mathcal{A}^{CCA}$ as subroutine, we construct an adversary $\mathcal{A}^{\pi}$ that can distinguish real session key from a random value with an advantage of

at least $\frac{\epsilon_c}{(n-1)}$. The running time of $\mathcal{A}^{\pi}$ is $t_2 \leq t_c + n_{fdo}(t_s + t_k)$, where t_c is the running time of $\mathcal{A}^{CCA}$, n_{fdo} is the number of FDO queries issued by $\mathcal{A}^{CCA}$ and t_s and t_k are the response times for send and session-key queries respectively.

$\mathcal{A}^{\pi}$ allows $\mathcal{A}^{CCA}$ to select a user U_B from the set of users $\{U_1, \ldots, U_n\}$. The aim of $\mathcal{A}^{CCA}$ is to break the confidentiality of an encapsulation created for U_B by any other user.

$\mathcal{A}^{\pi}$ now initiates a session $\pi^t_{A,B}$ between U_B and any other user U_A, by issuing a $\mathsf{send}(\pi^t_{A,B}, \lambda)$ query. It obtains the outgoing parameter C^* and establishes a matching session by issuing a $\mathsf{send}(\pi^t_{B,A}, C^*)$ query. $\mathcal{A}^{\pi}$ chooses either $\pi^t_{A,B}$ or $\pi^t_{B,A}$ as the test session. The challenger selects $b \in_R \{0,1\}$ and gives real session key computed in the test session if $b = 0$ or a random value chosen from session key distribution otherwise. Let K_b be the value returned to $\mathcal{A}^{\pi}$. $\mathcal{A}^{\pi}$ now issues a $\mathsf{session\text{-}expiration}(\pi^t_{A,B})$ query, which ensures that the key computed in that session is erased.

$\mathcal{A}^{\pi}$ corrupts all the users (including U_A) except U_B and gives the key pairs to $\mathcal{A}^{CCA}$. It is now ready to answer the queries asked by $\mathcal{A}^{CCA}$:

- *FDO Queries:* When a decapsulation query is asked with the input (pk_i, C), $\mathcal{A}^{\pi}$ initiates a session through $\mathsf{send}(\pi^s_{B,i}, C)$ query. It then issues a $\mathsf{session\text{-}key}(\pi^s_{B,i})$ and obtains the session key K generated in that session and returns it to $\mathcal{A}^{CCA}$. It returns a $\perp$ symbol if there is no key accepted in the session.

Answering the challenger: After adaptively asking the FDO queries $\mathcal{A}^{CCA}$ outputs a public key $pk_{s'}$. If $pk_{s'} \neq pk_A$, $\mathcal{A}^{\pi}$ aborts its execution. Otherwise, it gives (K_b, C^*) as the challenge to $\mathcal{A}^{CCA}$. $\mathcal{A}^{CCA}$ may continue to ask the FDO queries except the trivial one with input (pk_A, C^*). It finally returns a bit θ as its guess with an advantage ϵ_c. Incase $\theta = 0$, $\mathcal{A}^{\pi}$ outputs $b = 0$, which implies C^* is a valid encapsulation of K_b and thus K_b is a real session key. $\mathcal{A}^{\pi}$ outputs $b = 1$ otherwise.

$\mathcal{A}^{CCA}$ becomes successful if it can break the confidentiality of an encapsulation created for the initially chosen U_B by any other user. $\mathcal{A}^{\pi}$ wins its game with non-negligible advantage only if $\mathcal{A}^{CCA}$ outputs $pk_{s'} = pk_A$ in the challenge phase i.e. the public key of the user U_A selected by $\mathcal{A}^{\pi}$. This occurs with the probability $\frac{1}{(n-1)}$. Hence, the advantage of $\mathcal{A}^{\pi}$ when constructed from $\mathcal{A}^{CCA}$ is

$$Adv_2^{\pi}(k) \geq \frac{\epsilon_c}{(n-1)} \tag{2}$$

For each FDO query asked by $\mathcal{A}^{CCA}$, $\mathcal{A}^{\pi}$ has to establish a session through a send query and retrieve the session key through a session-key query. Hence, the running time of $\mathcal{A}^{\pi}$ is bounded by $t_2 \leq t_c + n_{fdo}(t_s + t_k)$.

From (1) and (2), the advantage of $\mathcal{A}^{\pi}$ when constructed from $\mathcal{A}^{CMA}$ or $\mathcal{A}^{CCA}$ is $Adv^{\pi}(k) \geq \min\{Adv_1^{\pi}(k), Adv_2^{\pi}(k)\}$, which is non-negligible. The running time of such $\mathcal{A}^{\pi}$ with the advantage $Adv^{\pi}(k)$ is $t_{\pi} \leq \max\{t_1, t_2\}$. But, as the protocol π is secure in the CK model $Adv^{\pi}(k)$ must be negligible. This is a contradiction to the construction of $\mathcal{A}^{\pi}$ from $\mathcal{A}^{CMA}$ or $\mathcal{A}^{CCA}$. Hence, there exists no such $\mathcal{A}^{CMA}$ or $\mathcal{A}^{CCA}$ that has non-negligible advantage against $\mathcal{SK}$ □

Note that, if π does not provide sender forward secrecy, then the resulting $\mathcal{SK}$ will be outsider secure for both confidentiality and unforgeability notions.

- Encapsulation
  1. Choose $t \in_R Z_q^*$
  2. Set $C = tP$
  3. Set $h = \mathcal{H}(C, (\hat{A}||\hat{B}))$
  4. Set $K = \text{Hash}((t + sh)P_r)$
  5. Output (K, C)

- Decapsulation
  1. Set $h = \mathcal{H}(C, (\hat{A}||\hat{B}))$
  2. Set $K = \text{Hash}\,(r(C + hP_s))$
  3. Output K

Figure 3: New Signcryption KEM

	Confidentiality		Unforgeability		Efficiency	
	Outsider	Insider	Outsider	Insider	Encap.	Decap.
ECISS-KEM1 [6]	Y	N	Y	N	2 Exp	1 Exp
ECISS-KEM2 [6]	broken					
Dent [9]	Y	N	Y	Y	1 Exp	2 Exp
Bjørstad and Dent [13]	Y	N	Y	Y	1 Exp	2 Exp
Our new KEM	Y	Y	Y	N	2 Exp	1.5 Exp

Table 1: Security and efficiency comparisons with existing signcryption KEMs

4.1 New signcryption KEM from the one-pass HMQV

The one-pass HMQV protocol proposed by Krawczyk [10] can be used as a signcryption KEM secure in the insider confidentiality and outsider unforgeability notions. This new signcryption KEM between the parties A and B in the multi-user setting is presented in Figure 3. Apart from the system parameters used for ECISS-KEM1 described in Section 3.1, a new hash function $\mathcal{H}$ defined as $\mathcal{H} : G \times \{0,1\}^* \rightarrow Z_q^*$ is used.

4.2 Security of the new KEM

Krawczyk [10] proved the one-pass HMQV secure in the CK model. Its security is based on the XCR signature, whose security was also proven by Krawczyk in the random oracle model assuming the hardness of the CDH problem. By combining this result with Theorem 3, it follows that the new signcryption KEM is secure in the insider confidentiality and outsider unforgeability notions.

Table 1 compares the new signcryption KEM with existing signcryption KEMs in terms of security and efficiency. The security notions considered are insider and outsider security for both confidentiality and unforgeability. The efficiency is measured by number of group exponentiations required in encapsulation and decapsulation algorithms. The new signcryption KEM is the only one that has insider security for confidentiality. It achieves this forward secrecy with an additional half-length exponentiation[3] compared to the ECISS-KEM1 in the decapsulation algorithm. Unlike the ECISS-KEM1, the discovery of ephemeral data by an adversary in the new signcryption KEM

[3]Krawczyk [10] showed that the length of $h = \frac{q}{2}$ provides the right performance-security trade-off.

leads to compromise of only one particular communication. Moreover, the notions of security considered for all other signcryption KEMs are in the two-user setting, whereas the security of the new signcryption KEM is treated in the multi-user setting.

5 One-pass key establishment from signcryption KEM

We now consider the generic construction in the other direction. We first discuss how a signcryption KEM $\mathcal{SK}$ can be used as a one-pass key establishment protocol π. The security requirements of $\mathcal{SK}$ that can be used are first stated and a formal construction of π from $\mathcal{SK}$ is then presented.

When $\mathcal{SK}$ is used as π, the encapsulation algorithm of $\mathcal{SK}$ becomes the session key computation process by the sender in π. The generated symmetric key serves as the session key and the encapsulation of the symmetric key as the outgoing message to the receiver. The receiver can compute the same session key by executing the decapsulation algorithm on the incoming message.

For $\mathcal{SK}$ to be suitable to be used as a one-pass key establishment protocol it should be secure in the insider confidentiality and outsider unforgeability notions. Security in these notions enables the resulting protocol to have SK-security with sender forward secrecy in the CK model. For the reasons discussed in Section 3.2 security against compromise of ephemeral data is not guaranteed for π. Therefore the adversary is not allowed to have access to the session-state query.

Theorem 4. *If a signcryption KEM is secure in the insider confidentiality and outsider unforgeability notions, then it can be used as a one-pass key establishment protocol π that is SK-secure with sender forward secrecy in the CK model (without* session-state *queries).*

Proof. The truth value of the above theorem is the same as the statement: if π is not secure in the CK model, then $\mathcal{SK}$ is not secure in either insider confidentiality **or** outsider unforgeability notion. Hence, it is enough to show that given an adversary $\mathcal{A}^{\pi}$ against π that can distinguish a real session key from a random number with advantage ϵ, then either $\mathcal{A}^{CMA}$ or $\mathcal{A}^{CCA}$ against $\mathcal{SK}$ can be constructed with advantage $\epsilon' \geq \epsilon$ in polynomial time.

The proof is divided into two parts. In the first part $\mathcal{A}^{CMA}$ is constructed with non-negligible advantage only if an event Forgery (explained later) occurs. In the second part $\mathcal{A}^{CCA}$ is constructed from $\mathcal{A}^{\pi}$ with non-negligible advantage if the event Forgery does not occur.

Let $\{U_1, U_2, ..., U_n\}$ be set of n users and assume each user is activated at most m times by $\mathcal{A}^{\pi}$, where n and m are polynomials in the security parameter.

Constructing $\mathcal{A}^{CMA}$ from $\mathcal{A}^{\pi}$: We assume the existence of $\mathcal{A}^{\pi}$ that can distinguish a real session key from a random value in time t_f. We then construct $\mathcal{A}^{CMA}$ within time $t_1 \leq t_f + m(n-1)(t_{feo} + 2 \cdot t_{fdo})$, where t_{feo} and t_{fdo} are the response times for the FEO and FDO queries respectively.

The input to $\mathcal{A}^{CMA}$ consists of the sender and receiver public keys pk_A and pk_B of two users U_A and U_B from the set of n users $\{U_1, \ldots, U_n\}$ respectively. Its aim is to produce (K^*, C^*) where C^* is a valid encapsulation of K^* under sk_A and pk_B, using $\mathcal{A}^{\pi}$ as subroutine. The input of $\mathcal{A}^{CMA}$ also contains key pairs of each of the n parties in the protocol, except the sender's private key sk_A of U_A and receiver's private key sk_B of U_B for whom only the corresponding public keys are given. $\mathcal{A}^{CMA}$ wins its game only if the target session chosen by $\mathcal{A}^{\pi}$ is a session between U_A and U_B. All the queries from $\mathcal{A}^{\pi}$ that do not concern U_A and U_B can be answered directly by $\mathcal{A}^{CMA}$ which knows the private keys of the users. For

queries that require the knowledge of sk_A and sk_B, $\mathcal{A}^{CMA}$ uses its own oracles and returns the messages produced to $\mathcal{A}^{\pi}$ as described below.

- send: When a $\mathsf{send}(\pi^s_{A,j}, \lambda)$ query is asked, $\mathcal{A}^{CMA}$ queries its FEO with the input pk_j and obtains (K, C). It then returns C to $\mathcal{A}^{\pi}$ as the outgoing message and keeps (s, A, j, K, C) in its encapsulation list $L_{\mathcal{E}}$. If $\mathcal{A}^{\pi}$ issues $\mathsf{send}(\pi^s_{B,i}, C)$, $\mathcal{A}^{CMA}$ queries its FDO with the input (pk_i, C). If it obtains a symmetric key K from the challenger, $\mathcal{A}^{CMA}$ marks the oracle $\pi^s_{B,i}$ as accepted and stores the value (s, B, i, K, C) in $L_{\mathcal{E}}$. If the output of the FDO is $\perp$ then the session is not accepted and the entry $(s, B, i, \perp, C)$ is stored in $L_{\mathcal{E}}$. The result of whether the session is accepted or not is made known to $\mathcal{A}^{\pi}$.
- session-key: For a $\mathsf{session\text{-}key}(\pi^s_{i,j})$ query, it returns the key held in the session with identifier s as follows: Since a session-key key reveal query is issued only on a session that has an accepted session key, the session id s must have an entry in $L_{\mathcal{E}}$. $\mathcal{A}^{CMA}$ checks to see if there is an entry for (s, i, j) in $L_{\mathcal{E}}$ and returns the corresponding key K incase of a match. If there is no key stored in $L_{\mathcal{E}}$ along with s, the session-key is not a valid query.
- session-expiration: On the input $\pi^s_{i,j}$, $\mathcal{A}^{CMA}$ deletes the entry with s from $L_{\mathcal{E}}$. There must have been an entry in $L_{\mathcal{E}}$ because session-expiration can be issued only on a completed session.
- corrupt: When $\mathcal{A}^{\pi}$ wishes to corrupt a party U_i (for $i \neq A, B$), $\mathcal{A}^{CMA}$ fetches session keys from $L_{\mathcal{E}}$ that are generated by the oracles at U_i. It returns all these keys (if any) along with U_i's long term private key. $\mathcal{A}^{CMA}$ outputs "fail" for a corrupt query on U_A or U_B. Note that $\mathcal{A}^{CMA}$ cannot return the internal state information for a corrupt query for reasons discussed in Section 3.2.

Whenever, a $\mathsf{send}(\pi^s_{B,A}, C)$ is issued, $\mathcal{A}^{CMA}$ first checks to see if there exists an entry (s, A, B, K, C) in $L_{\mathcal{E}}$ for some s and K. If there is an entry it just returns the message that the session is accepted. Otherwise, it queries its FDO with the input (pk_A, C). If the output of FDO is $\perp$ it returns the message that the session is not accepted. If $\text{FDO}(pk_A, C) = K^* \neq \perp$, $\mathcal{A}^{CMA}$ outputs (K^*, C^*) as its forgery with $C^* = C$.

Let Forgery be the event that $\mathcal{A}^{\pi}$ issues a $\mathsf{send}(\pi^s_{B,A}, C^*)$ such that C^* is a valid encapsulation under sk_A and pk_B such that it was not the response of an earlier $\mathsf{send}(\pi^s_{A,B}, \lambda)$ query. Clearly, $\mathcal{A}^{CMA}$ wins its game only if the event Forgery occurs. If $\mathcal{A}^{\pi}$ ends its run without choosing a test session between U_A and U_B or if the event Forgery does not occur $\mathcal{A}^{CMA}$ outputs "fail". The probability of $\mathcal{A}^{\pi}$ choosing a test session that has U_A as initiator and U_B as responder is $\frac{1}{n(n-1)}$. Thus, the non-negligible advantage of $\mathcal{A}^{CMA}$ is given as:

$$Adv^{CMA}_{\mathcal{A}}(k) \geq \frac{\Pr[\mathsf{Forgery}]}{n(n-1)} \tag{3}$$

For each send query to the oracle $\pi^s_{A,j}$, $\mathcal{A}^{CMA}$ has to query the FEO and FDO oracles. The maximum number of such queries involving U_A can be $m(n-1)$. Similarly, for each send query to $\pi^s_{B,i}$ a query to FDO is made. The maximum possible number of such queries involving U_B is $m(n-1)$. Hence, $\mathcal{A}^{CMA}$ can forge $\mathcal{SK}$ with the above advantage in time $t_1 \leq t_f + m(n-1)(t_{feo} + 2 \cdot t_{fdo})$.

Constructing $\mathcal{A}^{CCA}$ from $\mathcal{A}^{\pi}$: Now, we assume the existence of $\mathcal{A}^{\pi}$ that can distinguish a real session key from a random value in time t_d when the event Forgery does not occur. Using $\mathcal{A}^{\pi}$ as subroutine, we construct $\mathcal{A}^{CCA}$ within time $t_2 \leq t_d + (m(n-1)-1)t_{fdo}$, where t_{fdo} is the time required to get a response from FDO.

The input to $\mathcal{A}^{CCA}$ consists of a receiver's public key pk_B of a user U_B and the key pairs of rest of users from the set $\{U_1, \ldots, U_n\}$. The aim of $\mathcal{A}^{CCA}$ is to break the confidentiality of encapsulations created for U_B by any other user using $\mathcal{A}^{\pi}$ as subroutine.

$\mathcal{A}^{CCA}$ returns a sender's public key pk_A of a user U_A to its challenger. The challenger gives (K_b, C^*) to $\mathcal{A}^{CCA}$ as the challenge computed as described in Section 2.2. $\mathcal{A}^{CCA}$ chooses $t \in_R \{1, \ldots, m\}$. With these choices $\mathcal{A}^{CCA}$ is trying to guess $\mathcal{A}^{\pi}$'s choice of the target session. It is now ready to simulate the view of $\mathcal{A}^{\pi}$.

Except for U_B, the actions of rest of the uncorrupted users are performed by $\mathcal{A}^{CCA}$ with its knowledge of the corresponding private keys. For queries that require the knowledge of receiver's private key of U_B, $\mathcal{A}^{CCA}$ uses its own oracle.

- send: When $\mathcal{A}^{\pi}$ issues a $\mathsf{send}(\pi^s_{i,B}, \lambda)$ query, $\mathcal{A}^{CCA}$ generates (K, C), where C is encapsulation of K created by U_i as it knows the private key sk_i. It then returns C to $\mathcal{A}^{\pi}$ as the outgoing value and keeps (s, i, B, K, C) in its encapsulation list $L_{\mathcal{E}}$. If $\mathcal{A}$ issues $\mathsf{send}(\pi^s_{B,i}, C)$, $\mathcal{A}^{CCA}$ queries its FDO with the input (pk_i, C). If it obtains a symmetric key K from the challenger, the session is accepted and the entry (s, B, i, K, C) is stored in $L_{\mathcal{E}}$. If the output of FDO is $\perp$, the session is not accepted and the entry $(s, B, i, \perp, C)$ is stored. The result of whether the session is accepted or not is made known to $\mathcal{A}^{\pi}$. The t-th instantiation between U_A and U_B is handled in a special way as explained later.
- session-key: When a $\mathsf{session\text{-}key}(\pi^s_{i,j})$ is issued, $\mathcal{A}^{CCA}$ first checks to see if there is an entry for (s, i, j) in $L_{\mathcal{E}}$ and returns the corresponding key incase of a match. Otherwise, the session-key is not a valid query.
- session-expiration:On the input $\pi^s_{i,j}$, $\mathcal{A}^{CCA}$ deletes the entry with s from $L_{\mathcal{E}}$.
- corrupt: When $\mathcal{A}^{\pi}$ wishes to corrupt a party U_i (for $i \neq A, B$), $\mathcal{A}^{CCA}$ fetches session keys from $L_{\mathcal{E}}$ that are generated by the oracles at U_i. It returns all these keys (if any) along with U_i's long term private key. It aborts the simulation on a corrupt query on U_A or U_B.

If a $\mathsf{send}(\pi^t_{A,B}, \lambda)$ query is issued, $\mathcal{A}^{CCA}$ returns C^* as the outgoing parameter. Now, $\mathcal{A}^{\pi}$ can choose the t-th session between U_A and U_B as its target session in one of following ways

- The session $\pi^t_{A,B}$ itself or
- A matching session established by issuing $\mathsf{send}(\pi^t_{B,A}, C^*)$ query.

$\mathcal{A}^{\pi}$ can issue a $\mathsf{corrupt}(U_A)$ only after a $\mathsf{session\text{-}expiration}(\pi^t_{A,B})$. If it chooses the t-th session between A and B as the test session as expected by $\mathcal{A}^{CCA}$, K_b is returned. Eventually, $\mathcal{A}^{\pi}$ halts with its guess θ. If $\theta = 0$, $\mathcal{A}^{CCA}$ outputs $b = 0$ implying that K_b is a real session key and thus C^* is an encapsulation of K_b. Otherwise $b = 1$ is returned.

If Forgery occurs $\mathcal{A}^{\pi}$ may win its game without choosing the session in which the challenge encapsulation C^* is injected, as the test session. In this case $\mathcal{A}^{CCA}$ gets no advantage. Hence,

if the event Forgery occurs or if $\mathcal{A}^{\pi}$ chooses a different session other than the one expected by $\mathcal{A}^{CCA}$ as test session, $\mathcal{A}^{CCA}$ outputs a random bit b with a probability $\frac{1}{2}$.

The probability of $\mathcal{A}^{\pi}$ choosing a session t that has U_A as initiator and U_B as responder is $\frac{1}{mn(n-1)}$. The non-negligible advantage of $\mathcal{A}^{CCA}$ is given as

$$Adv_{\mathcal{A}}^{CCA}(k) \geq \frac{(Adv_{\mathcal{A}}^{\pi}(k)|\overline{\mathsf{Forgery}})}{mn(n-1)} \tag{4}$$

For each $\mathsf{send}(\pi_{B,i}^{s})$ query, $\mathcal{A}^{CCA}$ has to issue an FDO query. The maximum possible number of such queries invloving the user U_B is $m(n-1)-1$; excluding one in the test session. Hence, the running time of $\mathcal{A}^{CCA}$ with the above advantage is $t_2 \leq t_d + (m(n-1)-1)t_{fdo}$.

By the theorem of total probability, the advantage of $\mathcal{A}^{\pi}$ is given by

$$\begin{aligned} \mathcal{A}dv_{\mathcal{A}}^{\pi} &= (\mathcal{A}dv_{\mathcal{A}}^{\pi}|\mathsf{Forgery}) \times \Pr(\mathsf{Forgery}) + (\mathcal{A}dv_{\mathcal{A}}^{\pi}|\overline{\mathsf{Forgery}}) \times \Pr(\overline{\mathsf{Forgery}}) \\ &\leq \Pr(\mathsf{Forgery}) + (\mathcal{A}dv_{\mathcal{A}}^{\pi}|\overline{\mathsf{Forgery}}) \end{aligned}$$

However, from Equations 3 and 4, $\Pr(\mathsf{Forgery})$ and $(\mathcal{A}dv_{\mathcal{A}}^{\pi}|\overline{\mathsf{Forgery}})$ are negligible when $\mathcal{SK}$ is secure in the insider confidentiality and outsider unforgeability notions. Hence, the advantage of an adversary $\mathcal{A}^{\pi}$ against one-pass key establishment protocol constructed from such an $\mathcal{SK}$ is also negligible. □

New key establishment protocol: The ECISS-KEM1 in the multi user setting described in Figure 2 can be used as a one-pass key establishment protocol. However, as the ECISS-KEM1 is secure only in the outsider security model it does not provide sender forward secrecy. Moreover, as discussed in Section 3.2 it does not have security against session-state reveal queries. One advantage it does have over the one-pass HMQV is that its overall efficiency is better.

6 Conclusion

We have shown that there exists a duality between signcryption KEMs and one-pass key establishment protocols. However, the models typically used for defining security of key establishment are stronger than those for signcryption. Hence it has turned out that starting from signcryption KEMs we can only derive one-pass key establishment protocols with weaker security than those already known (such as HMQV).

In the other direction we have been able to use a strong one-pass key establishment protocol (HMQV) to derive a signcryption KEM with stronger properties than those known before. However, even though our signcryption KEM is stronger in terms of confidentiality, it does not provide insider secure authentication (non-repudiation). It might be possible to obtain a signcryption KEM that is insider secure with respect to both confidentiality and authentication from a one-pass key establishment protocol that is sender forward secure and resilient to KCI attacks. The feasibility of doing this is still not clear.

It remains an open question to derive hybrid signcryption schemes with insider security for both confidentiality and authentication even without using our generic constructions. Providing more signcryption schemes secure when the adversary has access to a session-state query also remains an interesting challenge.

Acknowledgements

The authors thank Alex Dent for his extensive comments and expert advice. This work was supported by the Australian Research Council under grant DP0773348.

References

[1] Zheng, Y.: Digital Signcryption or How to Achieve Cost(Signature & Encryption) << Cost(Signature) + Cost(Encryption). In: Advances in Cryptology–CRYPTO'97. Volume 1294 of LNCS., Springer (1997) 165–179

[2] Zheng, Y.: Shortened Digital Signature, Signcryption and Compact and Unforgeable Key Agreement Schemes. Technical report, http://grouper.ieee.org/groups/1363/StudyGroup/Hybrid.html (1998) A submission to IEEE P1363 Standard Specifications for Public Key Cryptography.

[3] An, J., Dodis, Y., Rabin, T.: On the Security of Joint Signature and Encryption. In: Advances in Cryptology–EUROCRYPT'02. Volume 2332 of LNCS., Springer (2002) 83–107

[4] An, J.: Authenticated Encryption in the Public-Key Setting: Security Notions and Analyses. Cryptology ePrint Archive, Report 2001/079 (2001) `http://eprint.iacr.org/2001/079`.

[5] Dodis, Y.: Signcryption (Short Survey). Encyclopedia of Cryptography and Security (2005) Available at `http://theory.lcs.mit.edu/~yevgen/surveys.html`.

[6] Dent, A.: Hybrid Signcryption Schemes with Outsider Security. In: Information Security, 8th International Conference–ISC'05. Volume 3650 of LNCS., Springer (2005) 203–217

[7] Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. Technical report, http://shoup.net/ (2002)

[8] Dent, A.: Hybrid Cryptography. Cryptology ePrint Archive, Report 2004/210 (2004) `http://eprint.iacr.org/2004/210`.

[9] Dent, A.: Hybrid Signcryption Schemes with Insider Security. In: Information Security and Privacy, 10th Australasian Conference–ACISP'05. Volume 3574 of LNCS., Springer (2005) 253–266

[10] Krawczyk, H.: HMQV: A High-Performance Secure Diffie-Hellman Protocol. In: Advances in Cryptology–CRYPTO'05. Volume 3621 of LNCS., Springer (2005) 546–566

[11] Baek, J., Steinfeld, R., Zheng, Y.: Formal Proofs for the Security of Signcryption. In: Public Key Cryptography–PKC'02. Volume 2274 of LNCS., Springer (2002)

[12] Baek, J., Steinfeld, R., Zheng, Y.: Formal Proofs for the Security of Signcryption. Journal of Cryptology **20** (2007) 203–235

[13] Bjørstad, T., Dent, A.: Building Better Signcryption Schemes with Tag-KEMs. In: Public Key Cryptography–PKC'06. Volume 3958 of LNCS., Springer (2006) 491–507

[14] Bellare, M., Canetti, R., Krawczyk, H.: A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols (Extended Abstract). In: Proc. of the 30th Annual ACM Symposium on Theory of Computing–STOC'98. (1998) 419–428

[15] Canetti, R., Krawczyk, H.: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In: Advances in Cryptology–EUROCRYPT'01. Volume 2045 of LNCS., Springer (2001) 453–474

[16] Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Advances in Cryptology–CRYPTO'93. Volume 773 of LNCS., Springer (1993) 232–249

[17] Tin, Y.S.T., Vasanta, H., Boyd, C., González-Nieto, J.M.: Protocols with Security Proofs for Mobile Applications. In: Information Security and Privacy, 9th Australasian Conference–ACISP'04. Volume 3108 of LNCS., Springer (2004) 358–369

[18] Yoshida, M., Fujiwara, T.: On the Security of Tag-KEM for Signcryption. Electr. Notes Theor. Comput. Sci. **171** (2007) 83–91

[19] International Organization for Standardization: ISO/IEC CD 18033-2, Information technology - Security techniques - Encryption Algorithms - Part 2: Asymmetric Ciphers. (2003)

[20] Okamoto, T., Pointcheval, D.: The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. In: Public Key Cryptography–PKC'01. Volume 1992 of LNCS., Springer (2001) 104–118

A Proof of Theorem 1

Proof. To prove this theorem we show that if there exists a polynomial time adversary $\mathcal{A}^{CMA}$ against the unforgeability of the KEM with non-negligible advantage ϵ, then a polynomial time algorithm $\mathcal{A}^{GDH}$ can be constructed that solves the Gap Diffie-Hellman (GDH) problem with the same advantage as $\mathcal{A}^{CMA}$. Recall that the GDH problem entails solving the Computational Diffie-Hellman (CDH) with the assistance of a decisional Diffie-Hellman oracle $\mathcal{O}_{DDH}$ [20].

Let $A = aP$, $B = bP$ be the problem instance given to $\mathcal{A}^{GDH}$ with the goal to find the value abP. $\mathcal{A}^{GDH}$ runs $\mathcal{A}^{CMA}$ and simulates the answers to the queries made by $\mathcal{A}^{CMA}$ as shown below.

- Hash: For Hash queries , $\mathcal{A}^{GDH}$ initially starts with an empty list $L_{\mathcal{H}}$. On input $(\hat{S}, \hat{R}, X)$, $\mathcal{A}^{GDH}$ first checks to see if there is an existing entry $(\hat{S}, \hat{R}, X, K)$ for some K in $L_{\mathcal{H}}$ that stores the past returned hash values. If so, it returns the corresponding K; otherwise it accesses the global encapsulation list $L_{\mathcal{E}}$ and does the following:

if $(\hat{S}, \hat{R}, C, K) \in L_{\mathcal{E}}$ *for some* K *and* C *values* **then**
compute $Y = (X - C)$
if $\mathcal{O}_{DDH}(P_s, P_r, Y) = True$ **then**
if $P_s = A$ *and* $P_r = B$ **then**
return Y as solution to the GDH challenger and exit
else
return K to $\mathcal{A}^{CMA}$
update $L_{\mathcal{H}} = L_{\mathcal{H}} \| (\hat{S}, \hat{R}, X, K)$
end
else
Select K randomly from the key distribution and return it to $\mathcal{A}^{CMA}$
update $L_{\mathcal{H}} = L_{\mathcal{H}} \| (\hat{S}, \hat{R}, X, K)$
end
else
Select K randomly from the key distribution and return it to $\mathcal{A}^{CMA}$
update$L_{\mathcal{H}} = L_{\mathcal{H}} \| (\hat{S}, \hat{R}, X, K)$
end

- FEO: Initially $\mathcal{A}^{GDH}$ has an empty encapsulation list $L_{\mathcal{E}}$. On input (P_s, P_r), $\mathcal{A}^{GDH}$ first selects $C \in_R G$. It then checks each entry $(\hat{S}, \hat{R}, X, K)$ in $L_{\mathcal{H}}$ to see if $\mathcal{O}_{DDH}(P_s, P_r, X - C) = True$ for the same (P_s, P_r) as in the input to FEO. If so, it fetches the corresponding K from $L_{\mathcal{H}}$; otherwise it selects K randomly from the symmetric key distribution. It returns (K, C) to $\mathcal{A}^{CMA}$. Finally, $L_{\mathcal{E}}$ is updated to $L_{\mathcal{E}} = L_{\mathcal{E}} \| (\hat{S}, \hat{R}, K, C)$.

- FDO: On input (P_s, P_r, C), $\mathcal{A}^{GDH}$ first checks to see if there is an entry$(\hat{S}, \hat{R}, C, K) \in L_{\mathcal{E}}$. In case of a match it returns the corresponding symmetric key K. Otherwise, it does the following:
if $(\hat{S}, \hat{R}, X, K) \in L_{\mathcal{H}}$ *for some* X **then**
compute $Y = (X - C)$
if $\mathcal{O}_{DDH}(P_s, P_r, Y) = True$ **then**
if $P_s = A$ *and* $P_r = B$ **then**
return Y as solution to the GDH challenger and exit
else
fetch corresponding K from $L_{\mathcal{H}}$ and return it to $\mathcal{A}^{CMA}$
update $L_{\mathcal{E}} = L_{\mathcal{E}} \| (\hat{S}, \hat{R}, K, C)$
end
else
Select K randomly from the key distribution and return it to $\mathcal{A}^{CMA}$
update $L_{\mathcal{E}} = L_{\mathcal{E}} \| (\hat{S}, \hat{R}, K, C)$
end
else
Select K randomly from the key distribution and return it to $\mathcal{A}^{CMA}$
update $L_{\mathcal{E}} = L_{\mathcal{E}} \| (\hat{S}, \hat{R}, K, C)$

end

Answering the GDH challenger: Eventually, $\mathcal{A}^{CMA}$ outputs a forgery (K^*, C^*) as an encapsulation created by S from R. For the forgery to be valid under the outsider unforgeability notion FEO/FDO-sUF-CMA, C^* must be a valid encapsulation of K^*. If C^* is a valid encapsulation of K^* then $\mathcal{A}^{CMA}$ must have queried the Hash with corresponding keying material, in which case $\mathcal{A}^{GDH}$ would have answered the GDH challenger already. Hence, the advantage of $\mathcal{A}^{GDH}$, ϵ' in solving the GDH problem is the same as the advantage of $\mathcal{A}^{CMA}$. □