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

M. Choudary Gorantla, Colin Boyd, and Juan Manuel González 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

Abstract. Key establishment between two parties that uses only one message transmission is referred to as one-pass key establishment (OPKE). OPKE provides the opportunity for very efficient constructions, even though they will typically provide a lower level of security than the corresponding multi-pass variants. In this paper, we explore the intuitive connection between signcryption and OPKE. By establishing a formal relationship between these two primitives, we show that with appropriate security notions, OPKE can be used as a signcryption KEM and vice versa. In order to establish the connection we explore the definitions of security for signcryption (KEM) and give new and generalised definitions. By making our generic constructions concrete we are able to provide new examples of signcryption KEMs and an OPKE protocol.

Keywords. One Pass Key establishment, Signcryption, Signcryption KEM

1 Introduction

One of the fundamental problems of cryptography is to establish a secret key between two parties with no prior shared state. In a groundbreaking paper, Diffie and Hellman [20] provided an elegant, yet simple way of solving this problem using public key cryptography. Later, many protocols have been proposed with improved security properties [29, 24, 28, 26, 13]. Although most of these protocols are interactive, many such protocols also provide simplified *one-pass* versions which use only one message. One-pass key establishment (OPKE) provides the opportunity for very efficient constructions, even though they will typically offer a lower level of security than the corresponding multi-pass variants. OPKE protocols can be deployed in systems that do not require the parties to participate interactively, such as E-mail or over networks with intermittent connectivity.

Another variant of two-party key establishment is non-interactive key establishment. The session key in a non-interactive protocol is established using the long-term private keys of the parties and optionally some session specific public information, like timestamps. As the secret values used in computing the session key are only the long-term keys, non-interactive protocols cannot offer forward secrecy i.e. revealing the long-term key of any of the parties would compromise the session keys established in the past. On the other hand, forward secrecy for the sender is a desired security property for OPKE protocols. Hence, OPKE protocols offer balanced security-efficiency trade off when compared to the non-interactive and multi-pass variants. For example, when a message needs to be encrypted with a session key, they require two less message flows than two-pass protocols and at the same time provide better security properties than non-interactive protocols.

1.1 OPKE and Signcryption

Zheng [34] introduced *signcryption* as a public-key cryptographic primitive that provides both privacy and authenticity at greater efficiency than the naive composition of signature and encryption

* This is the full version of the paper appeared in IMA Conference on Cryptography and Coding 2007. Research funded by the Australian Research Council through Discovery Project DP0773348.

schemes. Zheng [35] later observed that a signcryption scheme could be used in a key transport protocol by simply choosing a new key for each session and sending it in a signcrypted message. This intuitively gives the desired properties for key establishment since 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 appropriate notions of security for signcryption and show how signcryption and OPKE can be related under those notions.

Since the introduction of signcryption, different definitions of security have emerged. An, Dodis and Rabin [2] divided security notions for signcryption into two types: *outsider security* assumes that the adversary is not one of the parties communicating while *insider security* allows the adversary to be one of the communicating parties. Insider security for signcryption 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 [1].

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. For key establishment there is no need for non-repudiation as it is never required for a single party to claim 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 in this paper 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 [19].

Cramer and Shoup [16] formalised the concept of hybrid encryption which securely uses public key encryption technique to encrypt a session key, and symmetric key encryption technique 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 and is used to generate a random key and its encryption. In a series of papers, Dent [17, 19, 18] 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 to provide OPKE as suggested by Zheng [35], signcryption KEM seems better suited to the purpose. 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.

1.2 Related Work

An et al. [2] defined security notions for signcryption schemes as insider and outsider security in a two-user setting. They also described how to extend these notions to multi-user setting. Baek et al. [4] independently provided similar notions of security for signcryption, but their model was not completely adequate. Recently, the same authors [5] 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 3.

Zheng [35] informally showed how a signcryption scheme can be used as a key transport protocol. Dent [17, 19] and Bjørstad and Dent [11] discussed how a signcryption KEM can be used as an OPKE protocol. Bjørstad and Dent [11] proposed the concept of signcryption tag-KEM and claimed that better key establishment mechanisms could 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 [2]. However, this has been ignored or its absence has been treated as a positive feature called "Past Message Recovery" in earlier work [34, 5].

1.3 Contributions

We provide new definitions of insider confidentiality and outsider unforgeability for signcryption. These notions complement Baek et al.'s [5] notions of outsider confidentiality and insider unforgeability. We also extend security notions for signcryption KEM defined by Dent [18, 19] to multi-user setting. Furthermore, security notions for both signcryption and signcryption KEM are revised by allowing the adversary to reveal the random coins used in computing signcryptext and encapsulation respectively.

We then show the suitability of the new/revised notions in deriving OPKE from signcryption KEMs. Generic constructions of OPKE protocols from signcryption KEMs and vice versa are proposed. These constructions are instantiated using existing schemes. In particular, we use existing OPKE protocols [26, 32, 22] to derive new signcryption KEMs with stronger security properties than the current signcryption KEMs. 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 OPKE to signcryption KEM and vice versa;
- new signcryption KEMs with forward secrecy;
- an attack on a signcryption KEM of Dent [19].

1.4 Organization

Section 2 reviews the security goals and formal models for OPKE protocols. We present new notions of security for both signcryption and signcryption KEMs in Section 3. Section 4 examines the outsider secure signcryption KEMs designed by Dent [19]. The generic construction of signcryption KEM from OPKE is covered in Section 5, while the reverse construction is covered in Section 6.

2 Security Model for One-pass Key Establishment

Before introducing the formal notion of security for OPKE protocols, we first discuss security goals desired of a key establishment protocol in general and then identify how these goals can be adapted to OPKE protocols.

2.1 Security Goals

Blake-Wilson et al. [12] defined the desirable security goals of a two-party key establishment protocol as below:

known key security: The knowledge of a session key should not enable an adversary to compromise other session keys.

(perfect) forward secrecy: A protocol is said to have perfect forward secrecy if the compromise of long-term keys of one or more entities does not lead to compromise of past session keys established using those long-term keys.

unknown-key share: If a protocol resists unknown-key share attacks, an entity A cannot be coerced into sharing a key with entity B without A's knowledge, i.e., when A believes the key is shared with some entity $C \neq B$.

key compromise impersonation: A protocol is resilient to key compromise impersonation attacks, if compromising an entity A's long-term private key does not enable the adversary to impersonate other entities to A.

key control: Neither entity should be able to force the session key to be a preselected value

Out of the above goals, OPKE protocols can provide neither perfect forward secrecy nor key compromise impersonation resilience. An adversary with the knowledge of a receiver B's long-term private key can always compute all the session keys for the sessions in which it was just passively observing the communication to B. Moreover, if the ephemeral public key from A is not explicitly authenticated, the adversary can always impersonate other users to B. These attacks are inevitable due to lack of ephemeral contribution to the session key from B. However, a form of partial forward secrecy described below is desirable for OPKE protocols [30].

sender forward secrecy: A protocol is said to have sender forward secrecy if the compromise of long-term private key of a sender A does not lead to compromise of past session keys established using that key.

Krawczyk [26] identified a generic attack, which shows that no two-pass key establishment protocol authenticated via public keys and with no prior secret shared state can provide perfect forward secrecy. Instead, this type of protocols can only achieve *weak forward secrecy*, where the adversary remains passive during the protocol execution while the long-term private keys of both the parties can be revealed. This restriction equally applied to OPKE protocols and by sender forward secrecy, we mean weak forward secrecy with respect to sender's long-term key exposure.

Okamoto et al. [30] identified sender key compromise impersonation as a desired security goal for OPKE protocols. However, we observe that mounting a key compromise impersonation attack on the sender in a OPKE protocol does not make sense as there is only one message flowing from the sender to the receiver i.e., there is no incoming message for the sender.

2.2 Security Model for Authenticated Key Exchange

Bellare and Rogaway [9] initiated the formal security treatment of key establishment protocols in the complexity theoretic framework. Later models [10, 12, 6, 8, 15] extended this model to capture different attack scenarios. Although the model proposed by Canetti and Krawczyk [15](CK model) can be regarded adequate to analyse the security of a key establishment protocol, it is known that the CK model does not capture weak forward secrecy and resilience to key compromise impersonation (KCI).

Krawczyk [26] proposed extensions to the CK model which capture the properties of weak forward secrecy, KCI resilience and resilience to ephemeral private key leakage. Earlier models [9, 15] assumed strong corruption model, where the adversary is allowed to reveal session specific information in addition to long-term private key through a single corrupt query. On the other hand, the extensions proposed by Krawczyk assumed a weak corruption model in which the adversary is allowed to compromise only the long-term private key of a party by issuing corrupt query. Later, LaMacchia et al. [27] extended the CK model and proposed a unified model (called eCK model) which captures most of the desired security properties of key establishment protocols. We now briefly review the eCK model using some well known notations from earlier models.

A protocol π is modelled as a collection of programs running at n different parties. Let $\mathcal{U}$ be the set of n parties. Each party is assumed to have a pair of long-term public and private keys, (PK_U, SK_U) generated during an initialization phase prior to the protocol run. Each instance of π within a party is defined as a session and each party may have multiple such sessions running concurrently. Let π_U^i be the i-th run of the protocol π at party $U \in \mathcal{U}$.

Each protocol instance at a party is identified by a unique session ID. The session ID of an instance π_U^i is denoted by sid_U^i. We assume that the session ID is derived during the run of the protocol and also that each party knows who the other participant is for each protocol instance. The partner ID pid_U^i of an instance π_U^i, is a set containing the identity of the party U and that of the intended peer to the session.

An instance π_U^i enters an *accepted* state when it computes a session key sk_U^i. Note that an instance may terminate without ever entering into an accepted state. The information of whether an instance has terminated with acceptance or without acceptance is assumed to be public. Two instances π_U^i and $\pi_{U'}^j$ at two different parties U and U' respectively are considered *partnered* iff (1) both the instances have accepted, (2) $\mathsf{sid}_U^i = \mathsf{sid}_{U'}^j$ and (3) $\mathsf{pid}_U^i = \mathsf{pid}_{U'}^j$.

The communication network is assumed to be fully controlled by an adversary $\mathcal{A}$, which schedules and mediates the sessions among all the parties. $\mathcal{A}$ is allowed to insert, delete or modify the protocol messages. If the adversary honestly forwards the protocol messages among all the participants, then all the instances are partnered and output identical session keys. In addition to controlling message transmission, $\mathcal{A}$ is empowered with the ability to ask the following queries:

- $\mathsf{Send}(\pi_U^i, m)$: sends a unique message m to the instance π_U^i and the response is returned to $\mathcal{A}$. If m contains only the partner ID pid_U^i, this query activates π_U^i with $\mathsf{pid}_U^i \setminus \{U\}$ as peer.

– $\mathsf{RevealKey}(\pi_U^i)$: If π_U^i has accepted, the session key established at π_U^i is returned.
– $\mathsf{LongTermKeyReveal}(U)$: The long-term private key of the party U is returned.
– $\mathsf{EphemeralKeyReveal}(\pi_U^i)$: The ephemeral private key used during the execution of π_U^i is returned to $\mathcal{A}$. This includes the random coins generated during the protocol. We assume that the ephemeral private information is erased from the party's memory once the session is accepted.
– $\mathsf{Test}(\pi_U^i)$: On this query a random bit b is chosen. If $b = 0$, the session key established at π_U^i is returned to $\mathcal{A}$, otherwise a random value from the session key distribution is returned.

$\mathcal{A}$ is allowed to continue with its execution by issuing the above queries even after the Test query. Finally, it terminates by outputting its guess b' on distinguishing the session key from a random value.

Freshness Let π_U^i be an instance completed at party U and let $\pi_{U'}^j$ be the partnered instance at $U' \in \mathsf{pid}_U^i$ (there may be no such $\pi_{U'}^j$). An instance π_U^i is said to be not fresh if any of the following conditions holds:

– $\mathcal{A}$ issues $\mathsf{RevealKey}(\pi_U^i)$ **or** $\mathsf{RevealKey}(\pi_{U'}^j)$ (if there exists such a $\pi_{U'}^j$)
– A partnered instance $\pi_{U'}^j$ exists and
  - $\mathcal{A}$ issues [$\mathsf{LongTermKeyReveal}(U)$ **and** $\mathsf{EphemeralKeyReveal}(\pi_U^i)$] **or** [$\mathsf{LongTermKeyReveal}(U')$ **and** $\mathsf{EphemeralKeyReveal}(\pi_{U'}^j)$]
– A partnered instance $\pi_{U'}^j$ does not exist and
  - $\mathcal{A}$ issues [$\mathsf{LongTermKeyReveal}(U)$ **and** $\mathsf{EphemeralKeyReveal}(\pi_U^i)$] **or** $\mathsf{LongTermKeyReveal}(U')$

In all other cases the instance is said to be fresh. $\mathcal{A}$ wins the game if the selected test session remains fresh till the end of its execution and its guess $b' = b$. The advantage of $\mathcal{A}$ is defined as below:

$$Adv_\pi^{\mathcal{A}} = |2 \cdot \Pr[\mathcal{A} \text{ wins }] - 1|$$

A protocol π is said to be secure in the above model if for all PPT adversaries $\mathcal{A}$, $Adv_\pi^{\mathcal{A}}$ is negligible in the security parameter.

2.3 Security Model for OPKE

The above model can be easily adapted to capture the goals of OPKE protocols. The initial setup phase, the adversarial and communication models remain the same as defined for multi-pass key establishment protocols in the traditional public key setting. In independent work, Ustaoglu [32] defined a notion of freshness for OPKE protocols. We now present a slightly different notion.

Freshness Let π_U^i be an instance completed at party U and let $\pi_{U'}^j$ be the partnered instance at U' (there may be no such $\pi_{U'}^j$). An instance π_U^i is said not be opke-fresh if any of the following conditions holds:

– $\mathcal{A}$ issues $\mathsf{RevealKey}(\pi_U^i)$ **or** $\mathsf{RevealKey}(\pi_{U'}^j)$ (if there exists such a $\pi_{U'}^j$)
– A partnered instance $\pi_{U'}^j$ exists and
  - If π_U^i is an initiator session, $\mathcal{A}$ issues [$\mathsf{LongTermKeyReveal}(U)$ **and** $\mathsf{EphemeralKeyReveal}(\pi_U^i)$] **or** [$\mathsf{LongTermKeyReveal}(U')$]

- If π_U^i is a responder session, $\mathcal{A}$ issues [$\mathsf{LongTermKeyReveal}(U)$] **or** [$\mathsf{LongTermKeyReveal}(U')$ **and** $\mathsf{EphemeralKeyReveal}(\pi_{U'}^j)$]

– A partnered instance $\pi_{U'}^j$ does not exist and
  - $\mathcal{A}$ issues [$\mathsf{LongTermKeyReveal}(U)$] **or** $\mathsf{LongTermKeyReveal}(U')$

In all other cases π_U^i is said to be opke-fresh. The advantage of $\mathcal{A}$ in winning the above game is defined in the same way as defined for multi-pass protocols above.

The adapted model for OPKE protocols explicitly takes into the account the fact that there is no ephemeral contribution from the responder. At a high level, the difference between this model and the model for two-pass protocols is that an Ephemeral Key Reveal to a session at the responder returns no information in an OPKE protocol.

2.4 Replay Attacks and Receiver's Forward Secrecy

As mentioned in the security model above, we have defined session-ID as the concatenation of the identities of the peers and the message sent in that session. Since the security model insists that each session should have a unique session-ID, this formulation of session-IDs fails to model replay attacks for OPKE protocols. If the adversary replays the single message of a successfully completed past session, it will be accepted as that of the particular past session. This may be seen as an artificial way of preventing replay attacks on OPKE protocols. The situation could be addressed by including a time-stamp to uniquely identify a session. However, such an approach requires the parties to have access to a reliable universal time oracle [31] or a trusted time stamping authority which have to be online. Furthermore, as discussed earlier, OPKE protocols cannot provide forward secrecy for the receiver.

Both the above problems can be addressed in an elegant way using a key evolving scheme for the receiver. In a key evolving scheme [3, 7, 14] the private key of a party *evolves* over time; private keys for future time periods can be generated from the current private key but it is unconditionally/computationally infeasible to recover private keys of elapsed time periods from the current one. If we use a key evolving scheme to generate the private key for the receiver, we can design an OPKE protocol with receiver's forward secrecy such that session keys established during elapsed time periods cannot be recovered from the knowledge of the private key corresponding to the current time period. However it should be noted that compromising the private key for a time period τ_i allows the adversary to recover all the session keys already established during τ_i.

By employing the key evolving approach, replay attacks can also be thwarted to some extent. The adversary in this case can only successfully replay messages exchanged during the current time period, but not the ones from the sessions in the elapsed time period. If the parties execute only one session per time period, replay attacks can be completely eliminated.

3 Security Model for Signcryption

We now present new definitions of security for signcryption based on the those of Baek et al. [5]. We introduce Open query into the security model of signcryption. Although session state reveal query has been known in the security models for key establishment protocols, so far it has not been considered for signcryption. The Open query effectively allows an adversary against a signcryption scheme to selectively open a valid signcryptext. Informally, a signcryption scheme is considered secure under a model which considers open queries, if opening a signcryptext does not effect the security of other unopened signcryptexts.

3.1 New security notions for signcryption

A signcryption scheme $\mathcal{SC}$ is specified by four polynomial-time algorithms: common-key-gen, 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 to be used in the scheme.

key-gen: is a PPT algorithm that takes params as input and outputs a public-private key pair (pk, sk) used for signcryption and/or unsigncryption.

signcryption: is a PPT algorithm that takes params, a private key sk_s of a sender, 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), receiver key pairs (pk_r, sk_r) and any message m.

In a signcryption scheme, the users need private keys for two purposes: for signcrypting messages to be sent and for unsigncrypting messages received from other users. The private keys used for each of these purposes, along with their corresponding public keys, can be the same or different. This is much the same as the option to use the same, or different, keys for decryption and signing. In this paper, we assume that the users have only a single key pair for both signcryption and unsigncryption. At first look, this may seem to deviate from the common practice. However, we explain in Section 3.4 why our assumption is appropriate, particularly for the case of signcryption.

We define insider and outsider security notions for signcryption in multi-user setting based on the discussion given by An et al. [2]. It is natural to consider the security of a signcryption scheme in a multi-user setting where an honest party can generate signcryptexts for any user and also receive signcryptexts from any user in the system. In the security model, the adversary is given the ability to obtain signcryptions of an honest party that are created for any user through a flexible signcryption oracle (FSO). Similarly, it is also given access to a flexible unsigncryption oracle (FUO) that unsigncrypts a given signcryptext created for an honest party by any user. Because of these additional adversarial powers, security in the two-user setting does not imply security in multi-user setting [5].

The adversary is also empowered with Open queries, which allow it to reveal the random coins used while generating a signcryption by an honest party. This query can be seen as a way of modelling faulty random number generators. As we treat the security of signcryption schemes in multi-user setting, we explicitly allow the adversary to issue Corrupt queries to parties through which it can obtain the long-term private keys. Note that after corrupting a party the adversary can act on behalf of that user and interact with other honest users in the system. Intuitively, this query models the attack scenario where an adversary tries to break the confidentiality/unforgeability of a signcryption between two honest users with the knowledge of the signcryptions generated/obtained by interacting with honest users. The Corrupt query is also crucial in modelling the insider security notions for signcryption.

Let n be the number of users in the system, where n is polynomial in the security parameter k. The key pairs of the honest parties are generated by the challenger according to the specified key generation algorithm. The adversary is given the public keys of all the honest users initially. It is also given access to FSO, FUO, Open and Corrupt oracles corresponding to all the honest parties.

FSO: On the input (pk_s, pk_r, m), FSO returns a signcryptext C generated using sk_s corresponding to pk_s and the public key pk_r on the message m.

FUO: On the input (pk_s, pk_r, C), FUO returns a plaintext m or a $\perp$ symbol after performing unsigncryption on C using sk_r corresponding to pk_r and pk_s.

Open: On the input (pk_s, pk_r, C), Open oracle returns $\perp$ if C was not an output of an earlier FSO query. Otherwise, the random coins used while generating C under sk_s and pk_r are returned.

Corrupt: On input U_i, this oracle returns the private key of the user U_i.

Baek et al. [5] defined the notions of outsider security for confidentiality and insider security for unforgeability in multi-user setting. Note that they implicitly considered Corrupt queries, while Open queries were not considered at all. In this section, we extend the desired security notions for signcryption by introducing insider security for confidentiality and outsider security for unforgeability in multi-user setting.

Insider confidentiality Informally, the goal of $\mathcal{A}^{CCA}$ is to break the confidentiality of messages signcrypted for an honest target user by any user (including insiders).

- *Phase 1:* $\mathcal{A}^{CCA}$ is allowed to issue FSO, FUO, Open and Corrupt queries.
- *Challenge:* At the end of phase 1, $\mathcal{A}^{CCA}$ outputs two equal length messages m_0, m_1 and a pair of public keys (pk_s^*, pk_r^*) and submits them to the challenger. The challenger chooses a random bit b and gives $\mathcal{A}^{CCA}$ a challenge signcryptext C^* created on m_b using the private key sk_s^* (corresponding to pk_s^*) and pk_r^*.
- *Phase 2:* $\mathcal{A}^{CCA}$ can continue executing as in phase 1, except asking the trivial FUO query on the input (pk_s^*, pk_r^*, C^*).
- *Guess:* Finally, $\mathcal{A}^{CCA}$ outputs a bit b'. Let U_A and U_B be the owners of the public keys pk_s^* and pk_r^* respectively. $\mathcal{A}^{CCA}$ wins the game if $b' = b$ and if the following conditions hold.
  1. $\mathcal{A}^{CCA}$ has not issued Corrupt(U_B) query,
  2. Both Open(pk_s^*, pk_r^*, C^*) and Corrupt(U_A) have not been issued.

The advantage of $\mathcal{A}^{CCA}$ in winning the insider confidentiality game is:

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

Outsider confidentiality An adversary against outsider confidentiality of $\mathcal{SC}$ operates in the same way as the adversary against insider confidentiality described above except that it is not allowed to issue Corrupt(U_A) query. Informally, the goal of the adversary in this notion is to break the confidentiality of messages signcrypted by an honest user U_A for another honest user U_B. The advantage of the adversary in winning the outsider confidentiality game is defined in the same way as above.

Outsider unforgeability Informally, the goal of an outsider adversary $\mathcal{A}^{CMA}$ against unforgeability of $\mathcal{SC}$ is to forge a valid signcryptext created by an honest user U_A for another honest user U_B. $\mathcal{A}^{CMA}$ is allowed to issue FSO, FUO, Open and Corrupt queries. At the end of its execution, $\mathcal{A}^{CMA}$ outputs $(pk_s^*, pk_r^*, m^*, C^*)$ as its forgery. Let U_A and U_B be the owners of the public keys pk_s^* and pk_r^* respectively. $\mathcal{A}^{CMA}$ wins the outsider unforgeability game if the following conditions hold:

1. C^* is a valid signcryption of m^* under (pk_s^*, pk_r^*)
2. there has been no Corrupt query with the input U_A or U_B.
3. C^* is not an output of an earlier FSO query

The above game models strong unforgeability notion for signcryption. The advantage of $\mathcal{A}^{CMA}$ in the outsider unforgeability game is the same as its probability of success in winning the game.

Insider unforgeability An adversary against insider unforgeability of $\mathcal{SC}$ operates in the same way as an adversary against outsider unforgeability described above except that it is now allowed to issue $\mathsf{Corrupt}(\mathsf{U_B})$ query. Informally, the goal of adversary in this notion is to produce a valid forgery of a signcryptext created by an honest user U_A for any other user (including insiders). The advantage of an adversary in the insider unforgeability game is the same as its probability of success in winning the game.

3.2 New security notions for signcryption KEM

A signcryption KEM $\mathcal{SK}$ is specified by four polynomial-time algorithms: common-key-gen, key-gen, encapsulation and decapsulation. The algorithms common-key-gen and key-gen are identical to those defined in Section 3.1. The public-private key pair generated by the key-gen algorithm are now used for encapsulation and/or decapsulation.

encapsulation: is a PPT algorithm that takes params, a sender private key sk_s and a receiver 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 public key pk_s, a receiver 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 considered 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 [17, 19, 18] defined both insider and outsider security notions for signcryption KEM in the two-user setting. He also provided an informal description of how to define security for signcryption KEMs in multi-user setting [17]. Recently, Yoshida and Fujiwara [33] defined security notion for signcryption Tag-KEMs in multi-user setting. Here, we present new notions of security for signcryption KEMs in multi-user setting building on the definitions in Section 3.1.

In our security model, the adversary is given the power to obtain encapsulations created by an honest party for any user through a flexible encapsulation oracle (FEO). The adversary is also allowed to issue queries to a flexible decapsulation oracle (FDO) that decapsulates a given encapsulation created for an honest party by any user. It can also issue Open and Corrupt queries as explained earlier.

Let n be the number of users in the system, where n is polynomial in the security parameter k. The key pairs of the honest parties are generated by the challenger according to the specified key generation algorithm. The adversary is given the public keys of all the honest users initially. It is given access to FEO, FDO, Open and Corrupt oracles corresponding to the honest parties as described below.

FEO: On the input (pk_s, pk_r), FEO returns a pair (K, C), where C is an encapsulation of K generated using sk_s corresponding to pk_s and the public key pk_r

FDO: On the input (pk_s, pk_r, C), FDO returns a symmetric key K or a $\perp$ symbol after performing decapsulation on C using sk_r and pk_s.

Open: On the input (pk_s, pk_r, C), Open oracle returns $\perp$ if C was not an output of an earlier FEO query. Otherwise, the random coins used while generating C under sk_s and pk_r are returned.

$\mathsf{Corrupt}$: On input U_i, this oracle returns the private key of the user U_i.

Insider confidentiality Informally, the goal of $\mathcal{A}^{CCA}$ is to break the confidentiality of encapsulations created for an honest target user by any user (inlcuding insiders).

- *Phase 1:* $\mathcal{A}^{CCA}$ is allowed to issue FEO, FDO, Open and Corrupt queries.
- *Challenge:* At the end of phase 1, $\mathcal{A}^{CCA}$ outputs a pair of public keys (pk_s^*, pk_r^*). The challenger generates a valid symmetric key, encapsulation pair (K_0, C^*) using the private key sk_s^* corresponding to pk_s^* and public key pk_r^*. 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 to $\mathcal{A}^{CCA}$.
- *Phase 2:* $\mathcal{A}^{CCA}$ can continue executing as in phase 1, except asking the trivial FDO queries on the input (pk_s^*, pk_r^*, C^*).
- *Guess:* Finally, $\mathcal{A}^{CCA}$ outputs a bit b'. Let U_A and U_B be the owners of the public keys pk_s^* and pk_r^* respectively. $\mathcal{A}^{CCA}$ wins the game if $b' = b$ and if the following conditions hold.
  1. $\mathcal{A}^{CCA}$ has not issued $\mathsf{Corrupt}(U_B)$ query.
  2. Both $\mathsf{Open}(pk_s^*, pk_r^*, C^*)$ and $\mathsf{Corrupt}(U_A)$ have not been issued.

The advantage of $\mathcal{A}^{CCA}$ in winning the insider confidentiality game is:

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

Outsider confidentiality An adversary against outsider confidentiality of $\mathcal{SK}$ operates in the same way an adversary against insider confidentiality described above except that it is not allowed to issue $\mathsf{Corrupt}(U_A)$ query. Informally, the goal of the adversary in this notion is to break the confidentiality of encapsulations created by an honest user U_A for another honest user U_B. The advantage of the adversary in winning the outsider confidentiality game is defined in the same way as above.

Outsider unforgeability Informally, the goal of an outsider adversary $\mathcal{A}^{CMA}$ against unforgeability of $\mathcal{SK}$ is to forge a valid encapsulation created by an honest user U_A for another honest user U_B. $\mathcal{A}^{CMA}$ is allowed to issue FEO, FDO, Open and Corrupt queries. At the end of its execution, $\mathcal{A}^{CMA}$ outputs $(pk_s^*, pk_r^*, K^*, C^*)$ as its forgery. Let U_A and U_B be the owners of the public keys pk_s^* and pk_r^* respectively. $\mathcal{A}^{CMA}$ wins the outsider unforgeability game if the following conditions hold:

1. C^* is a valid encapsulation of K^* under (pk_s^*, pk_r^*)
2. there has been no Corrupt query with the input U_A or U_B.
3. (K^*, C^*) is not an output of an earlier FEO query

The above game models strong unforgeability notion for signcryption KEM. The advantage of $\mathcal{A}^{CMA}$ in the outsider unforgeability game is the same as its probability of success in winning the game.

Insider unforgeability An adversary against insider unforgeability of $\mathcal{SK}$ operates in the same way as an adversary against outsider unforgeability described above except that it is now allowed to issue $\mathsf{Corrupt}(\mathsf{U_B})$ query. Informally, the goal of adversary in this notion is to produce a valid forgery of an encapsulation created by an honest user U_A for any other user (including insiders). The advantage of an adversary in the insider unforgeability game is the same as its probability of success in winning the game.

3.3 On the unforgeability notion for signcryption KEMs

Dent [19] defined a different notion, called Left-or-Right (LoR) security, for outsider unforgeability of signcryption KEM. He showed that an adversary that can output a valid forgery (K^*, C^*) under the notion described in the previous section could be efficiently turned into another adversary that can win the LoR game. Dent pointed out that LoR security was 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 [19] 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. However, in our generic constructions we show that our notion of outsider unforgeability is enough when relating OPKE protocols with signcryption KEMs. One may still use an LoR secure signcryption KEM to derive an OPKE protocol, as LoR security guarantees security under our notion of unforgeability.

We emphasise that a hybrid signcryption scheme with insider security cannot be guaranteed using a signcryption KEM secure under the insider unforgeability notion described in the previous section. Dent [18] 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 OPKE protocols. A signcryption KEM that is insider confidential in our definition can be used to derive a OPKE protocol with sender forward secrecy. However, it is unclear what property is implied by our definition of insider unforgeability, when a signcryption KEM secure under this notion is used to derive an OPKE protocol.

3.4 On using a single key pair for both signcryption and unsigncryption

We have assumed that the users employ only a single key pair for both signcryption and unsigncryption in a signcryption scheme and similarly for both encapsulation and decapsulation in a signcryption KEM. As signcryption aims to provide both confidentiality and authentication simultaneously, our assumption may seem to contravene the practice of using a separate key pair each for encryption and signing. It is known in the folklore that if we use a single key pair for both encryption and signing, a conflict between confidentiality and authentication may arise. The signing key has to be under the sole control of the signer to guarantee non-repudiation, while the decryption key may have to be backed up by a trusted authority for enabling it with key escrow capability or for recovering the encrypted data in case the decryption key is lost.

Although signcryption aims to provide both confidentiality and authentication, note that the definition of signcryption does not automatically guarantee non-repudiation. In particular, a signcryption scheme with only outsider unforgeability does not imply non-repudiation as a receiver can always forge a valid signcryption. Hence, even if we have two separate key pairs, backing up users' unsigncryption keys enables the trusted authority to forge the signcryption of a party intended to any other party. Note that this can be done even if the signcryption keys are under the sole possession of the corresponding parties. Since we concentrate on signcryption schemes with only outsider unforgeability it is reasonable to assume that the users have only a single key pair for both signcryption and unsigncryption. The above argument equally applies to signcryption KEM.

4 Outsider Secure Signcryption KEMs

Dent [19] proposed an outsider secure signcryption KEM called elliptic curve integrated signcryption scheme KEM (ECISS-KEM1) based on the ECIES-KEM [25]. A potential problem with the ECISS-KEM1 was identified by Dent who then proposed another signcryption KEM that was claimed to overcome this problem, but without a proof of security. Both the schemes are described below.

4.1 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 $\mathcal{H}$ that outputs a symmetric key of desired length. ECISS-KEM1 is proven secure by Dent in the two-user setting against outsider security (for both confidentiality and unforgeability) assuming the hardness of the computational Diffie-Hellman problem.

– Encapsulation	– Decapsulation
1. Choose an element $t \in_R Z_q^*$	1. Set $K = \mathcal{H}(rP_s + C)$
2. Set $K = \mathcal{H}(sP_r + tP)$	2. Output K
3. Set $C = tP$	
4. Output (K, C)	

Fig. 1. ECISS-KEM1

4.2 Potential problems with ephemeral data

Dent discussed a potential weakness with 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 allows the adversary to indefinitely impersonate the sender.

It is interesting that Dent identified this as a problem although it is not recognized 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. Hence, the plausibility of an attack by the adversary by revealing ephemeral data, as well as its consequences, are equally valid for a signcryption KEM as for a key establishment protocol. The Open query we have introduced into the security notions of signcryption KEM models ephemeral data leakage and enables us to achieve a useful stronger notion of security.

The feasibility of an adversary revealing ephemeral data 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 also be argued that applications such as signcryption or OPKE are of limited vulnerability to ephemeral data leakage 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.

In our security model for signcryption KEM, Open query reveals only the random coins used while generating encapsulation, but not any other session specific information. This query can be seen as a way of modelling faulty random number generators. We assume that the operation of computing encapsulation to be atomic and also that the local values are securely erased immediately once they are used. Thus ECISS-KEM1 would still be secure under our new model. In fact we prove a slightly modified version of ECISS-KEM1 to be secure under our new outsider security notions.

Note that leaking the value $sP_r + tP$ allows the adversary to compute the static Diffie-Hellman key between the parties P_s and P_r with which the adversary can compromise the communication between these two parties forever. Hence, we emphasize that the static Diffie-Hellman key between two parties in this scheme should be protected with the same care as the long term private key would be protected. We leave it an open problem to construct a Diffie-Hellman based OPKE protocol/signcryption KEM even secure when the static Diffie-Hellman shared key is revealed.

4.3 ECISS-KEM2

Dent proposed another signcryption KEM (ECISS-KEM2) which was aimed to address the ephemeral data exposure problem. 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 $K = \mathcal{H}(sP_r + tP_r)$ and its encapsulation is $C = tP$. Given an encapsulation, the symmetric key can be recovered using the deterministic decapsulation algorithm as $K = \mathcal{H}(rP_s + rC)$.

Dent argued that even if an attacker discovered the value $sP_r + tP_r$, it would help in recovering only a single message for which the hashed material was 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 was stated informally, Dent claimed that a proof could be given with a non-standard security assumption. However, the attack below enables an active adversary to impersonate a 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 public key P_s. This forces the receiver to compute the symmetric key as $K = \mathcal{H}(rP_s + rC^*) = \mathcal{H}(rsP + r(P - sP)) = \mathcal{H}(rsP + rP - rsP) = \mathcal{H}(P_r)$, 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. This attack directly violates the Left or Right security defined by Dent for outsider unforgeability.

4.4 ECISS-KEM1 in multi-user setting

We slightly modify ECISS-KEM1 to work in multi-user environment as shown in Figure 2. As suggested by An et al. [2], the identities of the sender and receiver, $\hat{S}$ and $\hat{R}$ respectively, are now embedded in the encapsulation and decapsulation processes. We also make an efficiency improvement by directly choosing the encapsulation C uniformly at random from the group G. Note that in the scheme described in Figure 1, the same effect has been achieved at an additional exponentiation.

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

Fig. 2. ECISS-KEM1 in the multi-user setting

We show that the modified ECISS-KEM1 satisfies the outsider notions confidentiality and unforgeability defined in Section 3.2. Before proceeding to security proofs, we briefly describe the computational assumptions on which the security of ECISS-KEM1 in multi-user setting is based.

Computational Diffie-Hellman (CDH) problem: The CDH problem in a group G is to compute abP, given an instance $\langle P, aP, bP\rangle$, where P is an arbitrary generator of G and $a, b \in \mathbb{Z}_q^*$. It is assumed that CDH problem is hard to solve in G if for any polynomial time adversary the probability of computing the correct CDH, given a uniformly random CDH instance, is negligible in the security parameter.

Decisional Diffie-Hellman (DDH) problem: Given an instance $\langle P, aP, bP, cP\rangle$, the DDH problem is to output whether cP is the CDH of aP and bP or not. It is assumed that DDH problem is hard to solve in G if for any polynomial time adversary the probability of solving the DDH problem, given a uniformly random DDH instance from G, is negligibly close to $\frac{1}{2}$.

Gap Diffie-Hellman (GDH) problem: The GDH problem in a group G is to solve CDH in G given access to an oracle $\mathcal{O}_{DDH}$ which solves the DDH problem for a given instance. It is assumed that GDH problem is hard to solve in G if for any polynomial time adversary the probability of solving GDH, given a uniformly random CDH instance, is negligible in the security parameter.

Theorem 1. *ECISS-KEM1 in multi-user setting is secure in the outsider unforgeability notion in the random oracle model assuming hardness of the GDH problem in G. The advantage of a polynomial adversary $\mathcal{A}^{CMA}$ against the outsider unforgeability notion of the scheme is bounded by $n(n-1)\epsilon$, where ϵ is the advantage of a polynomial time GDH solver $\mathcal{A}^{GDH}$ and n is the number of users in the multi-user setting.*

Proof. Let $A = aP$, $B = bP$ be the problem instance given to $\mathcal{A}^{GDH}$. $\mathcal{A}^{GDH}$ injects the values A and B as public keys P_s^* and P_r^* of two users with identities $\hat{S}^*$ and $\hat{R}^*$ respectively. By selecting these two users, $\mathcal{A}^{GDH}$ guesses that $\mathcal{A}^{CMA}$ will output a forgery between $\hat{S}^*$ and $\hat{R}^*$. For all the other users $\mathcal{A}^{GDH}$ chooses the private-public keys pairs. It also maintains two lists $L_{\mathcal{H}}$ and $L_{\mathcal{E}}$ which are used to answer the random oracle $\mathcal{H}$ and the FEO queries. These lists are initially empty and will be filled in while $\mathcal{A}^{GDH}$ simulates answers to $\mathcal{A}^{CMA}$'s queries as below:

– $\mathcal{H}$ Queries: On input $(X, \hat{S}, \hat{R})$, $\mathcal{A}^{GDH}$ first checks to see if there is an existing entry $(X, \hat{S}, \hat{R}, K)$ 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 encapsulation list $L_{\mathcal{E}}$ and does the following:

if $(C, K, \hat{S}, \hat{R}) \in L_{\mathcal{E}}$ *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
  else
    return K to $\mathcal{A}^{CMA}$
    update $L_{\mathcal{H}} = L_{\mathcal{H}} \| (X, \hat{S}, \hat{R}, K)$
  end
else
  Select $K \xleftarrow{R} \{0,1\}^k$ and return it to $\mathcal{A}^{CMA}$
  update $L_{\mathcal{H}} = L_{\mathcal{H}} \| (X, \hat{S}, \hat{R}, K)$
end
else
Select $K \xleftarrow{R} \{0,1\}^k$ and return it to $\mathcal{A}^{CMA}$
update $L_{\mathcal{H}} = L_{\mathcal{H}} \| (X, \hat{S}, \hat{R}, K)$
end

– FEO: $\mathcal{A}^{GDH}$ starts with 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 $(X, \hat{S}, \hat{R}, K) \in L_{\mathcal{H}}$ 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 $K \xleftarrow{R} \{0,1\}^k$. It returns (K, C) to $\mathcal{A}^{CMA}$. Finally, $L_{\mathcal{E}}$ is updated to $L_{\mathcal{E}} = L_{\mathcal{E}} \| (C, K, \hat{S}, \hat{R})$.
– FDO: On input (P_s, P_r, C), $\mathcal{A}^{GDH}$ first checks to see if there is an entry $(C, K, \hat{S}, \hat{R}) \in L_{\mathcal{E}}$. In case of a match it returns the corresponding symmetric key K. Otherwise, it does the following:
if $(X, \hat{S}, \hat{R}, K) \in L_{\mathcal{H}}$ *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
  else
    fetch corresponding K from $L_{\mathcal{H}}$ and return it to $\mathcal{A}^{CMA}$
    update $L_{\mathcal{E}} = L_{\mathcal{E}} \| (C, K, \hat{S}, \hat{R})$
  end
else
  Select $K \xleftarrow{R} \{0,1\}^k$ and return it to $\mathcal{A}^{CMA}$
  update $L_{\mathcal{E}} = L_{\mathcal{E}} \| (C, K, \hat{S}, \hat{R})$
end
else
Select $K \xleftarrow{R} \{0,1\}^k$ and return it to $\mathcal{A}^{CMA}$
update $L_{\mathcal{E}} = L_{\mathcal{E}} \| (C, K, \hat{S}, \hat{R})$
end
– Corrupt: $\mathcal{A}^{GDH}$ aborts on a Corrupt query with input S^* or R^*. For all the other users this query can be trivially answered since $\mathcal{A}^{GDH}$ generated the key pairs for them.
– Open: This query returns the random tape used during the encapsulation process.

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, C^* must be a valid encapsulation of K^*. If C^* is a valid encapsulation of K^* then $\mathcal{A}^{CMA}$ must have queried the $\mathcal{H}$ with corresponding keying material, in which case $\mathcal{A}^{GDH}$ would have answered the GDH challenger already. The probability $\mathcal{A}^{CMA}$ selecting S^* as sender and R^* as recipient from a set of n users is $\frac{1}{n(n-1)}$. Hence, the advantage of $\mathcal{A}^{GDH}$, ϵ in solving the GDH problem is $\geq \frac{\epsilon'}{n(n-1)}$, where ϵ' is the advantage of $\mathcal{A}^{CMA}$. □

Theorem 2. *ECISS-KEM1 in multi-user setting is secure in the outsider confidentiality notion in the random oracle model assuming hardness of the GDH problem in G. The advantage of a polynomial adversary $\mathcal{A}^{CCA}$ against the outsider confidentiality notion of the scheme is upper bounded by $\left(2n(n-1)\epsilon + \frac{2(q_e+q_d)}{q}\right)$, where ϵ is the advantage of a polynomial time GDH solver $\mathcal{A}^{GDH}$, n is the number of users in the multi-user setting and q_e and q_d are the maximum number of encapsulation and decapsulation queries respectively that $\mathcal{A}^{CCA}$ is allowed to ask.*

Proof. Let $A = aP$, $B = bP$ be the problem instance given to $\mathcal{A}^{GDH}$. $\mathcal{A}^{GDH}$ injects the input values as the public keys of two users and simulates answers to all $\mathcal{H}$, FEO, FDO, $\mathsf{Corrupt}$ and Open queries of $\mathcal{A}^{CCA}$ in the same way as described above.

Answering the GDH challenger: In the *Challenge* phase, $\mathcal{A}^{GDH}$ aborts if $\mathcal{A}^{CCA}$ does not output (pk_s^*, pk_r^*). Otherwise, it randomly selects an encapsulation $C^* \in G$, a random key $K^* \xleftarrow{R} \{0,1\}^*$ and returns (K^*, C^*) as the challenge to $\mathcal{A}^{CCA}$.

If $\mathcal{A}^{CCA}$ can distinguish the random key K^* from the real key corresponding to the encapsulation C^* with non-negligible advantage, either (1) it must have queried the random oracle $\mathcal{H}$ with the input $(X^*, \hat{S}^*, \hat{R}^*)$ such that $(X^* - C^*)$ is the CDH of P_s^* and P_r^* or (2) C^* was an output of an earlier FEO query or input to an earlier FDO query. In case 1, $\mathcal{A}^{GDH}$ would have answered the GDH challenger correctly as explained in the simulation above.

Let E_1 and E_2 be the events that the cases 1 and 2 occur respectively. Note that if the event E_1 happens, the success probability of $\mathcal{A}^{GDH}$, $\epsilon \geq \frac{\Pr[E_1]}{n(n-1)}$, while it does not have any advantage when the event E_2 occurs. Since C^* was chosen at randomly from the group G, the probability of occurrence for E_2 is $\frac{q_e+q_d}{q}$. The probability of success for $\mathcal{A}^{CCA}$ is given as:

$$\begin{aligned}
\Pr[\mathsf{Succ}_{\mathcal{A}^{CCA}}] &= \Pr[\mathsf{Succ}_{\mathcal{A}^{CCA}}|E_2]\Pr[E_2] + \Pr[\mathsf{Succ}_{\mathcal{A}^{CCA}}|\neg E_2]\Pr[\neg E_2] \\
&\leq \Pr[E_2] + \Pr[\mathsf{Succ}_{\mathcal{A}^{CCA}}|\neg E_2] \\
&\leq \frac{q_e+q_d}{q} + \Pr[\mathsf{Succ}_{\mathcal{A}^{CCA}}|\neg E_2] \qquad (1)
\end{aligned}$$

We also have

$$\begin{aligned}
\Pr[\mathsf{Succ}_{\mathcal{A}^{CCA}}|\neg E_2] &= \Pr[\mathsf{Succ}_{\mathcal{A}^{CCA}}|E_1 \wedge \neg E_2]\Pr[E_1] + \Pr[\mathsf{Succ}_{\mathcal{A}^{CCA}}|\neg E_1 \wedge \neg E_2]\Pr[\neg E_1] \\
&\leq \Pr[E_1] + \Pr[\mathsf{Succ}_{\mathcal{A}^{CCA}}|\neg E_1 \wedge \neg E_2] \\
&\leq \Pr[E_1] + \frac{1}{2} \qquad (2)
\end{aligned}$$

From equations 1 and 2 we have

$$\Pr[E_1] \geq \frac{1}{2}\left(\mathsf{Adv}_{\mathcal{A}^{CCA}} - \frac{2(q_e+q_d)}{q}\right)$$

Hence,

$$\epsilon \geq \frac{1}{2n(n-1)}\left(\mathsf{Adv}_{\mathcal{A}^{CCA}} - \frac{2(q_e+q_d)}{q}\right)$$

By rearranging the above equation, we have the claimed advantage for $\mathcal{A}^{CCA}$. □

5 From OPKE to Signcryption KEM

Let π be an OPKE protocol. We show that π can be directly used as a signcryption KEM $\mathcal{SK}$. The session key computed in π serves as the symmetric key output of the signcryption KEM, while the outgoing message of π becomes the encapsulation of the key in $\mathcal{SK}$. The session key computation process at the receiver end in π can be applied as the decapsulation algorithm in $\mathcal{SK}$ to retrieve the symmetric key.

We now examine how the properties of an OPKE protocol transfer into the context of a signcryption KEM scheme. Note that if an OPKE protocol has sender forward secrecy, the session key remains uncompromised even if the long-term private key of the sender is revealed. This property essentially is the same as the insider confidentiality notion defined for signcryption KEM, which protects the confidentiality of symmetric key even when the sender's private key is compromised. Similarly, the implicit authentication property guarantees the parties in an OPKE protocol that only the intended peers can compute the session key. This property is identical to the outsider unforgeability notion of signcryption KEM, which guarantees that only the sender or the receiver can produce valid encapsulation-symmetric key pairs between themselves. We now give a formal proof for our generic construction.

Theorem 3. *If π is an OPKE protocol secure as per the model in Section 2.3, then it can be used as a signcryption KEM $\mathcal{SK}$ secure in the insider confidentiality and outsider unforgeability notions.*

Proof. We prove the theorem by showing that if $\mathcal{SK}$ is not secure in the insider confidentiality **or** outsider unforgeability notion, then π is also not secure as per the security model in Section 2.3. 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 π, which 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.

Let $\mathcal{U} = \{U_1, \ldots, U_n\}$ be a set of n users, where n is polynomial in the security parameter k. $\mathcal{A}^{\pi}$ obtains the public keys of all the users from its challenger and forwards them to $\mathcal{A}^{CMA}$. $\mathcal{A}^{\pi}$ answers the queries asked by $\mathcal{A}^{CMA}$ as below:

– *FEO Queries*: For an FEO query asked by $\mathcal{A}^{CMA}$ with input (pk_s, pk_r), $\mathcal{A}^{\pi}$ initiates a session by issuing a $\mathsf{send}(\pi^i_{U_s}, \mathsf{pid}^i_{U_s})$ query, where $\mathsf{pid}^i_{U_s} = \{U_s, U_r\}$ and U_s and U_r are the owners of the public keys pk_s and pk_r respectively. Let C be the outgoing message returned by π^i_U. $\mathcal{A}^{\pi}$ then issues a $\mathsf{RevealKey}(\pi^i_U)$ query and receives the session key K accepted in that session. The pair (K, C) is returned as the output of the FEO query to $\mathcal{A}^{CMA}$.
– *FDO Queries*: On an FDO query with the input (pk_s, pk_r, C), $\mathcal{A}^{\pi}$ issues a $\mathsf{send}(\pi^j_{U_r}, (\mathsf{pid}^j_{U_r}, C))$, where $\mathsf{pid}^j_{U_r} = \{U_s, U_r\}$. If the session is accepted, it issues a $\mathsf{RevealKey}(\pi^j_{U_r})$ query and returns the session key K to $\mathcal{A}^{CMA}$. Otherwise, it returns a $\perp$ symbol.
– *Corrupt Queries*: On a $\mathsf{Corrupt}$ query with the input U, $\mathcal{A}^{\pi}$ issues a $\mathsf{LongTermKeyReveal}(U)$ and forwards the output to $\mathcal{A}^{CMA}$.
– *Open Queries*: On an Open query with the input π^i_U, $\mathcal{A}^{\pi}$ issues a $\mathsf{EphemeralKeyReveal}(\pi^i_U)$ and forwards the output to $\mathcal{A}^{CMA}$.

Answering the challenger: $\mathcal{A}^{CMA}$ finally outputs $(pk^*_s, pk^*_r, K^*, C^*)$ as its forgery such that C^* is a valid encapsulation of K^* created by U^*_s for U^*_r. $\mathcal{A}^{\pi}$ now establishes a fresh session between U^*_s and U^*_r, by issuing $\mathsf{send}(\pi^j_{U^*_r}, (\mathsf{pid}^j_{U^*_r}, C))$. It chooses this session 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^*_s created for U^*_r such that U^*_s and U^*_r have not been corrupted. As explained above, $\mathcal{A}^{\pi}$ 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

$$Adv^{\pi}_1(k) = \epsilon_u \tag{3}$$

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 $\mathsf{session\text{-}key}$ 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 $\frac{\epsilon_c}{n(n-1)}$. 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 $\mathsf{session\text{-}key}$ queries respectively.
Let $\mathcal{U} = \{U_1, \ldots, U_n\}$ be a set of n users, where n is polynomial in the security parameter k. $\mathcal{A}^{\pi}$ obtains the public keys of all the users from its challenger and forwards them to $\mathcal{A}^{CCA}$. $\mathcal{A}^{\pi}$ selects two users $U^*_s, U^*_r \in \mathcal{U}$. By selecting these users, $\mathcal{A}^{\pi}$ guesses that U^*_s and U^*_r will be the chosen as the sender and receiver respectively by $\mathcal{A}^{CCA}$ in the challenge phase. $\mathcal{A}^{\pi}$ now initiates a session $\pi^t_{U^*_s}$ by issuing a $\mathsf{send}(\pi^t_{U^*_s}, \mathsf{pid}^t_{U^*_s})$, where $\mathsf{pid}^t_{U^*_s} = \{U^*_s, U^*_r\}$. It obtains the outgoing message C^* and establishes a matching session by issuing a $\mathsf{send}(\pi^{t'}_{U^*_r}, \mathsf{pid}^{t'}_{U^*_r})$ query, where $\mathsf{pid}^{t'}_{U^*_r} = \mathsf{pid}^t_{U^*_s}$. $\mathcal{A}^{\pi}$ now chooses either $\pi^t_{U^*_s}$ or $\pi^{t'}_{U^*_r}$ 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 $\mathcal{A}^{\pi}$.

– *FDO, FEO, Corrupt and Open Queries:* These queries are answered by $\mathcal{A}^{\pi}$ in the same as way described above. Note that $\mathcal{A}^{CCA}$ is allowed to issues a $\mathsf{Corrupt}$ with the input U^*_s.

Answering the challenger: After adaptively asking the queries $\mathcal{A}^{CCA}$ outputs a pair of public keys (pk'_s, pk'_r). If $pk_{s'} \neq pk^*_s$ and $pk'_r \neq pk^*_r$, $\mathcal{A}^{\pi}$ aborts its execution. Otherwise, it gives (K_b, C^*) as the challenge to $\mathcal{A}^{CCA}$. $\mathcal{A}^{CCA}$ may continue to ask the queries in Phase 2 with trivial restrictions stated in Section 3.2. It finally returns a bit θ as its guess with an advantage ϵ_c. In case $\theta = 0$, $\mathcal{A}^{\pi}$ 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.
Note that $\mathcal{A}^{\pi}$ wins its game with non-negligible advantage only if its guess for the public keys output by $\mathcal{A}^{CCA}$ is correct i.e. only if U^*_s and U^*_r are chosen by $\mathcal{A}^{CCA}$ as sender and receiver respectively in the challenge phase. This occurs with the probability $\frac{1}{n(n-1)}$. Hence, the advantage of $\mathcal{A}^{\pi}$ when constructed from $\mathcal{A}^{CCA}$ is

$$Adv^{\pi}_2(k) \geq \frac{\epsilon_c}{n(n-1)} \tag{4}$$

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 (3) and (4), the advantage of $\mathcal{A}^{\pi}$ when constructed from $\mathcal{A}^{CMA}$ or $\mathcal{A}^{CCA}$ is $Adv^{\pi}(k) \geq \min\{Adv^{\pi}_1(k), Adv^{\pi}_2(k)\}$, 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 $Adv^{\pi}(k)$ 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.

5.1 New signcryption KEMs

OPKE protocols proposed by Krawczyk [26], Ustaoglu [32] and Gorantla et al. [22] can be used as signcryption KEMs secure in the insider confidentiality and outsider unforgeability notions. The new signcryption KEMs between the parties A and B in multi-user setting are presented in Figure 3. The private-public key pairs of A and B are (s, P_s) and (r, P_r) respectively.

5.2 Security of the new KEMs

Krawczyk [26] gave a sketch of the proof of security for one-pass HMQV in the random oracle model. The security model considers session-state reveal query instead of EphemeralKeyReveal query. It can be easily shown that one-pass HMQV is secure under the model in Section 2.3. By combining this result with Theorem 3, it follows that the new signcryption KEM obtained from one-pass HMQV is secure in the insider confidentiality and outsider unforgeability notions.

We can also obtain signcryption KEMs from the OPKE protocols proposed by Ustaoglu [32] and Gorantla et al. [22]. The one-pass CMQV protocol of Ustaoglu was proven secure in the eCK model described in Section 2.3 assuming random oracles. On the other hand, the OPKE protocol of Gorantla et al. [22] was proven in a weaker model that does not consider session-state reveal or EphemeralKeyReveal queries. Therefore, the signcryption KEM derived from this OPKE protocol does not guarantee security when the adversary is allowed to ask Open queries. However, unlike one-pass HMQV/CMQV this protocol was proven secure in the standard model. In the earlier version

Signcryption KEM based on one-pass HMQV

params: $(G, P, q, \mathcal{H}_1, \mathcal{H}_2)$, $\mathcal{H}_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q, \mathcal{H}_2 : \{0,1\}^* \rightarrow \{0,1\}^k$

- Encapsulation
  1. Choose $t \in_R Z_q^*$
  2. Set $C = tP$
  3. Set $h = \mathcal{H}_1(C, (\hat{A}||\hat{B}))$
  4. Set $K = \mathcal{H}_2((t + sh)P_r)$
  5. Output (K, C)
- Decapsulation
  1. Set $h = \mathcal{H}_1(C, (\hat{A}||\hat{B}))$
  2. Set $K = \mathcal{H}_2\left(r(C + hP_s)\right)$
  3. Output K

Signcryption KEM based on one-pass CMQV

params: $(G, P, q, \mathcal{H}_0, \mathcal{H}_1, \mathcal{H}_2)$, $\mathcal{H}_0 : \{0,1\}^k \times \mathbb{Z}_q^* \rightarrow \mathbb{Z}_q^*, \mathcal{H}_1, \mathcal{H}_2$: as above

- Encapsulation
  1. Choose $\tilde{t} \in_R \{0,1\}^k$
  2. Compute $t = \mathcal{H}_0(\tilde{t}, s)$, $C = tP$
  3. Set $h = \mathcal{H}_1(C, \hat{A}, \hat{B})$
  4. Set $K = \mathcal{H}_1((t + sh)P_r, C, \hat{A}, \hat{B})$
  5. Output (K, C)
- Decapsulation
  1. Set $h = \mathcal{H}_1(C, \hat{A}, \hat{B})$
  2. Set $K = \mathcal{H}_2(r(C + hP_s), C, \hat{A}, \hat{B})$
  3. Output K

Signcryption KEM based on the OPKE protocol of Gorantla et al.

params: $(G, P, q, \mathcal{M}, \mathcal{H}_2)$, $\mathcal{M}$: a MAC scheme, $\mathcal{H}_2$: as above

- Encapsulation
  1. Compute $K_{sr} = sP_r$
  2. Choose $u \in_R Z_q^*$
  3. Set $U = uP$
  4. Compute $\tau = \mathcal{M}_{K_{sr}}(U, \hat{A}, \hat{B})$
  5. Set $C = (U, \tau)$
  6. Set $K = \mathcal{H}_2(uP_r, \hat{A}, \hat{B})$
  7. Output (K, C)
- Decapsulation
  1. Compute $K_{rs} = rP_s$
  2. Check $\tau \stackrel{?}{=} \mathcal{M}_{K_{rs}}(U, \hat{A}, \hat{B})$
  3. Set $K = \mathcal{H}_2(rU, \hat{A}, \hat{B})$
  4. Output K

Fig. 3. New Signcryption KEM

of this paper [23], we defined security for signcryption KEMs without considering Open queries and also established connection between signcryption KEM and OPKE protocols using that model. Hence, the signcryption KEM resulting from the OPKE protocol of Gorantla et al. [22] would be the first signcryption KEM secure in the standard model under the security notions defined in the earlier version of this paper.

	Confidentiality		Unforgeability		Efficiency		Model
	Outsider	Insider	Outsider	Insider	Encap.	Decap.	
ECISS-KEM1 [19]	Y	N	Y	N	2 Exp	1 Exp	ROM
ECISS-KEM2 [19]	broken						ROM
Dent [18]	Y	N	Y	Y	1 Exp	2 Exp	ROM
Bjørstad and Dent [11]	Y	N	Y	Y	1 Exp	2 Exp	ROM
One-pass HMQV [26]	Y	Y	Y	N	2 Exp	1.5 Exp	ROM
One-pass CMQV [32]	Y	Y	Y	N	2 Exp	1.5 Exp	ROM
Gorantla et al. [22]	Y	Y	Y	N	1 Exp	1 Exp	Std.

Table 1. Security and efficiency comparisons with existing signcryption KEMs

Table 1 compares the new signcryption KEMs 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. Unlike the previous schemes, all the new signcryption KEMs achieve insider security for confidentiality. The schemes based on one-pass

HMQV/CMQV achieve this additional property only at the cost of an extra half-length exponentiation [1] compared to the ECISS-KEM1 in the decapsulation algorithm. Moreover, these schemes also have a proof of security when the adversary leaks session state/ephemeral key. Although the scheme based on Gorantla et al.'s [22] protocol does not have a proof of security when the adversary has access to ephemeral data, it is more efficient than the other signcryption KEMs, when the static Diffie-Hellman key between the sender and the receiver is pre-computed. It should also be noted that the security of the new signcryption KEMs is treated in multi-user setting.

Note that we have not considered ephemeral pubic key validation when comparing efficiency of signcryption KEMs. In each of the schemes in Table 3 schemes, it requires an additional exponentiation for the recipient to perform this operation.

6 From signcryption KEM to OPKE

We now consider the generic construction in the other direction. We first discuss how a signcryption KEM $\mathcal{SK}$ can be used as an OPKE protocol π. When $\mathcal{SK}$ is used as π, the encapsulation algorithm of $\mathcal{SK}$ becomes the session key computation process for the sender in π. The generated symmetric key serves as the session key, while 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 an OPKE protocol secure in the model defined in Section 2.3, it should be secure in the insider confidentiality and outsider unforgeability notions. Informally, security under insider confidentiality and outsider unforgeability enables the resulting protocol to have sender forward secrecy and implicit authentication respectively. If $\mathcal{SK}$ is secure when considering open queries, the resulting π will be secure under against compromise of ephemeral data.

Theorem 4. *If a signcryption KEM is secure in the insider confidentiality and outsider unforgeability notions, then it can be used as an OPKE π that is secure under the model defined in Section 2.3.*

Proof. The truth value of the above theorem is the same as the statement: if π is not secure in the model defined in Section 2.3, 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.

[1] Krawczyk [26] explained that choosing the output length of the hash function $\mathcal{H}_1$ as $\frac{q}{2}$ provides the right performance-security trade-off.

The input to $\mathcal{A}^{CMA}$ consists of a set of n users $\{U_1, \ldots, U_n\}$ and their public keys. Its aim is to produce (K^*, C^*) where C^* is a valid encapsulation of K^* under the key pairs of two honest users from $\mathcal{U}$, using $\mathcal{A}^\pi$ as subroutine. $\mathcal{A}^{CMA}$ wins its game with non-negligible advantage only if the event Forgery occurs. All the queries from $\mathcal{A}^\pi$ can be answered by $\mathcal{A}^{CMA}$ using the oracles available as part of the outsider unforgeability notion. The simulation is described below:

- Send: When a $\mathsf{Send}(\pi^i_{U_s}, \mathsf{pid}^i_{U_s})$ query is asked, where $\mathsf{pid}^i_{U_s} = \{U_s, U_r\}$, $\mathcal{A}^{CMA}$ queries its FEO with input (pk_s, pk_r), where pk_s and pk_r are the public keys of the users U_s and U_r respectively. It obtains (K, C) and returns C to $\mathcal{A}^\pi$ as the outgoing message. $\mathcal{A}^{CMA}$ also keeps an entry (i, U_s, U_r, K, C) in its encapsulation list $L_{\mathcal{E}}$.
  If $\mathcal{A}^\pi$ issues $\mathsf{send}(\pi^j_{U_r}, (\mathsf{pid}^j_{U_r}, C))$, $\mathcal{A}^{CMA}$ queries its FDO with the input (pk_s, pk_r, C). If it obtains a symmetric key K from the challenger, $\mathcal{A}^{CMA}$ stores the value (j, U_r, U_s, K, C) in $L_{\mathcal{E}}$. If the output of FDO is $\perp$ then the session is not accepted and the entry (j, U_r, U_s, $\perp$, C) is stored in $L_{\mathcal{E}}$. The result of whether the session is accepted or not is made known to $\mathcal{A}^\pi$.
- RevealKey: For a RevealKey query with the input $(\pi^i_{U_s})$ query, it returns the key held in that session as follows: Since a RevealKey key reveal query is issued only on a session that has accepted, 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, U_s, U_r, $*$, $*$) in $L_{\mathcal{E}}$ and returns the corresponding key K in case of a match. If there is no key stored in $L_{\mathcal{E}}$ for the record (s, U_s, U_r, $*$, $*$), $\mathcal{A}^{CMA}$ query returns $\perp$.
- LongTermKeyReveal: When $\mathcal{A}^\pi$ wishes to reveal the long term key of a party U_i, $\mathcal{A}^{CMA}$ issues a Corrupt query to its challenger with the input U_i. The value returned is given to $\mathcal{A}^\pi$.
- EphemeralKeyReveal: On an EphemeralKeyReveal query with the input π^i_U, $\mathcal{A}^{CMA}$ first checks that there is an entry for the instance π^i_U and if there is a match it extracts the encapsulation C from the corresponding entry and issues an $\mathsf{Open}(pk_s, pk_s, C)$ to its challenger. The value returned by the challenger is forwarded to $\mathcal{A}^{CMA}$.
- Test: A Test query with the input $\pi^t_{U_{s^*}}$ can be trivially answered by $\mathcal{A}^{CMA}$ since it has access to all the session keys in the list $L_{\mathcal{E}}$.

Let Forgery be the event that $\mathcal{A}^\pi$ issues a $\mathsf{send}(\pi^j_{U_{r^*}}, (\mathsf{pid}^j_{U_{r^*}}, C^*))$ such that C^* is a valid encapsulation under the key pairs of U^*_s and U^*_r such that C^* is not a response to an earlier $(\pi^i_{U_{s^*}}, \mathsf{pid}^i_{U_{s^*}})$ query and both U^*_s and U^*_r are uncorrupted. If Forgery occurs, $\mathcal{A}^{CMA}$ fetches the corresponding K^* from $L_{\mathcal{E}}$ and outputs (K^*, C^*) as its forgery. Note that $\mathcal{A}^{CMA}$ can check the validity of C^* under the key pairs of U^*_s and U^*_r by issuing a $\mathrm{FDO}(U^*_s, U^*_r, C^*)$ query.
Clearly, $\mathcal{A}^{CMA}$ wins its game if and only if the event Forgery occurs. Thus, the advantage of $\mathcal{A}^{CMA}$ is given as:

$$Adv^{CMA}_{\mathcal{A}}(k) = \Pr[\mathsf{Forgery}] \tag{5}$$

For each Send query to the instance π^i_U, $\mathcal{A}^{CMA}$ 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 that $\mathcal{A}^\pi$ 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 a 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.

Let $\mathcal{U} = \{U_1, \ldots, U_n\}$ be a set of n users, where n is polynomial in the security parameter k. $\mathcal{A}^{CCA}$ obtains the public keys of all the users from its challenger and forwards them to $\mathcal{A}^{\pi}$. $\mathcal{A}^{CCA}$ selects two users $U_s^*, U_r^* \in \mathcal{U}$. By selecting these users, $\mathcal{A}^{\pi}$ guesses that U_s^* and U_r^* will be the chosen as the peers in the test session. Let pk_s^* and pk_r^* be the public keys of the users U_s^* and U_r^* respectively. In the *Challenge* phase, $\mathcal{A}^{CCA}$ outputs the public keys (pk_s, pk_r^*) to its challenger. The challenger in turn gives (K_b, C^*) to $\mathcal{A}^{CCA}$ as described in Section 3.2.
$\mathcal{A}^{CCA}$ chooses $t \in_R \{1, \ldots, m\}$. With these choices $\mathcal{A}^{CCA}$ is trying to guess $\mathcal{A}^{\pi}$'s choice of the test session. 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. It is now ready to simulate the view of $\mathcal{A}^{\pi}$.

- Send: These queries are answered in the same way explained in the first part of the proof except the t-th instantiation at the party U_s^*, which is handled as follows: If U_r^* is not the peer in that session $\mathcal{A}^{CCA}$ aborts. Otherwise, it returns C^* as the outgoing parameter.
- RevealKey, EphemeralKeyReveal: These queries are handled in the same way as explained above.
- LongTermKeyReveal: $\mathcal{A}^{CCA}$ aborts on a LongTermKeyReveal query with input U_r^*. All other queries are handled in the same as explained above.
- Test: If $\pi^t_{U_{s^*}}$ or its matching session $\pi^{t'}_{U_{r^*}}$ is not the test session $\mathcal{A}^{CCA}$ aborts. Otherwise, it returns K_b to $\mathcal{A}^{\pi}$.

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 $\frac{1}{mn(n-1)}$. Hence, The advantage of $\mathcal{A}^{CCA}$ is given as

$$Adv_{\mathcal{A}}^{CCA}(k) \geq \frac{(Adv_{\mathcal{A}}^{\pi}(k)|\overline{\mathsf{Forgery}})}{mn(n-1)} \tag{6}$$

For each $\mathsf{send}(\pi^s_{B,i})$ query, $\mathcal{A}^{CCA}$ has to issue an FDO query. The maximum possible number of such queries involving 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 5 and 6, $\Pr(\mathsf{Forgery})$ 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 the OPKE protocol constructed from such an $\mathcal{SK}$ is also negligible. □

A new OPKE Protocol. The ECISS-KEM1 in multi-user setting described in Figure 2 can be used as an OPKE protocol. However, as the ECISS-KEM1 is secure only in the outsider security model it does not provide sender forward secrecy. The security of the resulting OPKE protocol follows from Theorems 1, 2 and 4.

7 Conclusion

We have explored the connection between OPKE protocols and signcryption KEMs. In the process, we have defined new notions of security for both signcryption and signcryption KEMs in multi-user setting. Inspired by existing models for key establishment protocols, the security models for signcryption and signcryption KEMs have been strengthened by considering ephemeral data leakage. We have then shown that there exists a duality between OPKE protocols and signcryption KEMs.

By instantiating our generic construction, we have been able to use existing OPKE protocols to derive new signcryption KEMs with stronger properties than those known before. However, even though the new signcryption KEMs are stronger in terms of confidentiality, they do not provide insider secure authentication (non-repudiation). It is an open question to investigate what properties of an OPKE protocol imply insider unforgeability for signcryption KEM. In the other direction, we have derived an OPKE protocol with weaker security than those already known. It is also interesting to see if one can construct OPKE protocols from insider secure signcryption KEMs [18].

Our generic constructions can be easily adapted to the ID-based setting. To the best of our knowledge, there exists no ID-based signcryption KEM in the literature. Hence, by applying our generic construction of deriving signcryption KEMs from OPKE protocols, we can obtain the first ID-based signcryption KEM from the ID-based OPKE protocol of Gorantla et al. [21].

References

1. 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`
2. An, J., Dodis, Y., Rabin, T.: On the Security of Joint Signature and Encryption. In: Advances in Cryptology–EUROCRYPT'02, *LNCS*, vol. 2332, pp. 83–107. Springer (2002)
3. Anderson, R.: Two Remarks on Public Key Cryptology. Tech. Rep. 549, Computer Laboratory, University of Cambridge (2002). Available at `http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-549.pdf`
4. Baek, J., Steinfeld, R., Zheng, Y.: Formal Proofs for the Security of Signcryption. In: Public Key Cryptography–PKC'02, *LNCS*, vol. 2274. Springer (2002)
5. Baek, J., Steinfeld, R., Zheng, Y.: Formal Proofs for the Security of Signcryption. Journal of Cryptology **20**(2), 203–235 (2007)
6. 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, pp. 419–428 (1998)
7. Bellare, M., Miner, S.K.: A Forward-Secure Digital Signature Scheme. In: Advances in Cryptology–CRYPTO '99, *LNCS*, vol. 1666, pp. 431–448. Springer (1999)
8. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated Key Exchange Secure against Dictionary Attacks. In: Advances in Cryptology–EUROCRYPT'00, *LNCS*, vol. 1807, pp. 139–155. Springer (2000)
9. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Advances in Cryptology–CRYPTO'93, *LNCS*, vol. 773, pp. 232–249. Springer (1993)
10. Bellare, M., Rogaway, P.: Provably secure session key distribution: the three party case. In: Proc. of the 27th Annual ACM Symposium on Theory of Computing–STOC'95, pp. 57–66. ACM (1995)
11. Bjørstad, T., Dent, A.: Building Better Signcryption Schemes with Tag-KEMs. In: Public Key Cryptography–PKC'06, *LNCS*, vol. 3958, pp. 491–507. Springer (2006)
12. Blake-Wilson, S., Johnson, D., Menezes, A.: Key Agreement Protocols and Their Security Analysis. In: Proc. of the 6th IMA Intl. Conf. on Cryptography and Coding'97, *LNCS*, vol. 1355, pp. 30–45. Springer (1997)
13. Boyd, C., Cliff, Y., González Nieto, J., Paterson, K.G.: Efficient One-Round Key Exchange in the Standard Model. In: Y. Mu, W. Susilo, J. Seberry (eds.) Information Security and Privacy – ACISP 2008, *LNCS*, vol. 5107. Springer (2008)
14. Canetti, R., Halevi, S., Katz, J.: A Forward-Secure Public-Key Encryption Scheme. In: E. Biham (ed.) Advances in Cryptology–EUROCRYPT 2003, *LNCS*, vol. 2656, pp. 255–271. Springer (2003)

15. Canetti, R., Krawczyk, H.: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In: Advances in Cryptology–EUROCRYPT'01, *LNCS*, vol. 2045, pp. 453–474. Springer (2001)
16. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. Tech. rep., http://shoup.net/ (2002)
17. Dent, A.: Hybrid Cryptography. Cryptology ePrint Archive, Report 2004/210 (2004). `http://eprint.iacr.org/2004/210`
18. Dent, A.: Hybrid Signcryption Schemes with Insider Security. In: Information Security and Privacy, 10th Australasian Conference–ACISP'05, *LNCS*, vol. 3574, pp. 253–266. Springer (2005)
19. Dent, A.: Hybrid Signcryption Schemes with Outsider Security. In: Information Security, 8th International Conference–ISC'05, *LNCS*, vol. 3650, pp. 203–217. Springer (2005)
20. Diffie, W., Hellman, M.: New Directions in Cryptography. IEEE Transactions on Information Theory **IT-22**(6), 644–654 (1976)
21. Gorantla, M.C., Boyd, C., Gonzalez Nieto, J.M.: Id-based one-pass authenticated key establishment. In: L. Brankovic, M. Miller (eds.) Sixth Australasian Information Security Conference (AISC 2008), *CRPIT*, vol. 81, pp. 39–46. ACS, Wollongong, NSW, Australia (2008)
22. Gorantla, M.C., Boyd, C., González Nieto, J.M.: Strong designated verifier signature in a multi-user setting. In: L. Brankovic, W. Susilo (eds.) Seventh Australasian Information Security Conference (AISC 2009), *CRPIT*, vol. 98, pp. 21–31. ACS, Wellington, New Zealand (2009)
23. Gorantla, M.C., Boyd, C., Nieto, J.M.G.: On the connection between signcryption and one-pass key establishment. In: S.D. Galbraith (ed.) IMA Int. Conf., *LNCS*, vol. 4887, pp. 277–301. Springer (2007)
24. IEEE: P1363 Standard Specifications for Public Key Cryptography. Tech. rep., IEEE (2000). IEEE Std. 1363–2000
25. International Organization for Standardization: ISO/IEC CD 18033-2, Information technology - Security techniques - Encryption Algorithms - Part 2: Asymmetric Ciphers (2003)
26. Krawczyk, H.: HMQV: A High-Performance Secure Diffie-Hellman Protocol. In: Advances in Cryptology–CRYPTO'05, *LNCS*, vol. 3621, pp. 546–566. Springer (2005)
27. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger Security of Authenticated Key Exchange. Cryptology ePrint Archive, Report 2006/073 (2006). `http://eprint.iacr.org/2006/073` To appear in ProvSec 2007
28. Law, L., Menezes, A., Qu, M., Solinas, J., Vanstone, S.: An Efficient Protocol for Authenticated Key Agreement. Des. Codes Cryptography **28**(2), 119–134 (2003)
29. Matsumoto, T., Takashima, Y., Imai, H.: On Seeking Smart Public-Key-Distribution Systems. IEICE TRANSACTIONS **E69**(2), 99–106 (1986)
30. Okamoto, T., Tso, R., Okamoto, E.: One-Way and Two-Party Authenticated ID-Based Key Agreement Protocols Using Pairing. In: Modeling Decisions for Artificial Intelligence, 2nd International Conference–MDAI'05, *LNCS*, vol. 3558, pp. 122–133. Springer (2005)
31. 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, *LNCS*, vol. 3108, pp. 358–369. Springer (2004)
32. Ustaoglu, B.: Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. Des. Codes Cryptography **46**(3), 329–342 (2008)
33. Yoshida, M., Fujiwara, T.: On the Security of Tag-KEM for Signcryption. Electr. Notes Theor. Comput. Sci. **171**(1), 83–91 (2007)
34. Zheng, Y.: Digital Signcryption or How to Achieve Cost(Signature & Encryption) << Cost(Signature) + Cost(Encryption). In: Advances in Cryptology–CRYPTO'97, *LNCS*, vol. 1294, pp. 165–179. Springer (1997)
35. Zheng, Y.: Shortened Digital Signature, Signcryption and Compact and Unforgeable Key Agreement Schemes. Tech. rep., http://grouper.ieee.org/groups/1363/StudyGroup/Hybrid.html (1998). A submission to IEEE P1363 Standard Specifications for Public Key Cryptography