Int. J. Inf. Secur. (2017) 16:541–557
DOI 10.1007/s10207-016-0346-9

REGULAR CONTRIBUTION

Generic construction of an eCK-secure key exchange protocol in the standard model

Janaka Alawatugoda[1]

Published online: 5 August 2016

Abstract LaMacchia, Lauter and Mityagin presented a strong security model for authenticated key agreement, namely the eCK model. They also constructed a protocol, namely the NAXOS protocol, that enjoys a simple security proof in the eCK model. However, the NAXOS protocol uses a random oracle-based technique to combine the long-term secret key and the per session randomness, so-called NAXOS trick, in order to achieve the eCK security definition. For NAXOS trick-based protocols, the leakage of per session randomness modeled in the eCK model is somewhat unnatural, because the eCK model leaks per session randomness, while the output of the NAXOS trick computation remains safe. In this work, we present a standard model eCK-secure protocol construction, eliminating the NAXOS trick. Moreover, our protocol is a generic construction, which can be instantiated with arbitrary suitable cryptographic primitives. Thus, we present a generic eCK-secure, NAXOS-free, standard model key exchange protocol. To the best of our knowledge this is the first paper on generic transformation of a CCA2-secure public-key encryption scheme to an eCK-secure key exchange protocol in the standard model.

Keywords Public-key cryptography · Key exchange protocols · eCK model · Standard model

1 Introduction

In 1976, Diffie and Hellman introduced a key exchange primitive [10], which enables two parties to exchange a secret key (session key) by communicating over a public channel. Users *Alice* and *Bob* agree on a group $\mathbb{G}$ of prime order q and on a generator g of this group. This is done before executing the rest of the protocol, and g and q are assumed to be public. Alice picks a random integer $a \xleftarrow{\$} \mathbb{Z}_q$ and computes $A \leftarrow g^a$ and sends it to Bob. Then Bob picks a random integer $b \xleftarrow{\$} \mathbb{Z}_q$ and computes $B \leftarrow g^b$ and sends it to Alice. After that, Alice computes $B^a = (g^b)^a = s \in \mathbb{G}$ and Bob computes $A^b = (g^a)^b = s \in \mathbb{G}$. Thus, both Alice and Bob end up with the same value $s \in \mathbb{G}$. An eavesdropper who watches this communication can see A and B values, but should be unable to determine the values of s (assuming CDH holds).

Many key exchange protocols have been created based on the Diffie–Hellman key exchange primitive [7,11,14]. In these key exchange protocols, different types of keys may be used to compute session keys: long-term secret keys are the static secrets belonging to the protocol participants which are often used to add authentication to the session key, and ephemeral keys are the session-specific secrets belonging to protocol participants which are used to add freshness to the session key. There are a number of known security features for key exchange protocols:

Implicit key authentication If a protocol provides a guarantee that no party apart from the protocol participants can compute the session key, that key exchange protocol is said to provide *implicit key authentication*. If a key exchange protocol provides implicit key authentication, that protocol is said to be an *authenticated key exchange* protocol.

Key confirmation If a key exchange protocol provides a guarantee that each party is assured that all other participants possess the session key, that key exchange protocol is said to provide *key confirmation*.

✉ Janaka Alawatugoda
janaka@ce.pdn.ac.lk

[1] Department of Computer Engineering, University of Peradeniya, Peradeniya 20400, Sri Lanka

Known key security The knowledge of a session key should not enable the adversary to learn the session keys in other sessions; all session keys should not depend on the session keys of other sessions.

Unknown key share (UKS) security It should not happen that a party A shares a session key with some party B, but believes that it is sharing the session key with some one else C. That means public keys and identities of the parties should be certified and confirmed or incorporated into protocol execution.

Key compromise impersonation (KCI) resistance Knowing the long-term secret key of a party A should not enable the adversary to impersonate other honest parties to A.

Forward secrecy An adversary who knows the long-term secret keys of parties should not be able to compute the session keys of past sessions between those two particular parties.

1.1 Key exchange security models

In order to analyze the security of key exchange protocols, a formal methodology is needed. Therefore, key exchange security models have been created. A security model is a formal security statement of certain security features. Generally, security models are designed to reflect real-world adversarial capabilities, addressing the known security features (mentioned earlier). It is natural to design security models with theoretical adversaries which have more capabilities than real-world adversaries, because that way it is possible to address more powerful attacks which may exist in the future. Following is the general structure of a security model.

- *Definition of the algorithm:* Inputs, outputs and abstract description of the algorithm.
- *Adversary capabilities:* How the adversary can interact with the system and which information the adversary is allowed to learn, usually in the form of queries. As a usual practice the adversary is made as strong as possible by giving more capabilities to the adversary.
- *Security game:* The way in which the adversary performs queries.
- *Security goal:* The requirement for the adversary to win the security game.

In a security model, there is a predefined list of queries that an adversary can perform (adversary capabilities). Those queries reveal information such as session keys, ephemeral keys and long-term secret keys. Even after performing the queries, within the constraints defined in the security model, if the adversary's advantage of distinguishing the real session key from a random key chosen from the same distribution is negligible, the protocol is said to be secure in the particular security model. The session in which the adversary tries to distinguish the real session key from a random key is known as the *target session.*

Table 1 Security features of different security models

Security feature	BR 93	BR 95	CK	eCK
Implicit key authentication	Yes	Yes	Yes	Yes
Known key security	Yes	Yes	Yes	Yes
Key confirmation	Yes	Yes	Yes	Yes
UKS	Yes	Yes	Yes	Yes
KCI	No	No	No	Yes
Forward secrecy	No	No	Yes	Weak forward secrecy

The Bellare–Rogaway models (BR93 [4], BR95 [6]), the Canetti–Krawczyk (CK) model [8] and the extended Canetti–Krawczyk (eCK) model [17] are a few such security models, and protocol designers use them to analyze the security of key exchange protocols. Security features like implicit key authentication, key confirmation, known key security and UKS security are addressed in the models such as BR models, CK model and the eCK model.

In the BR models and the CK model, the adversary is not allowed to learn the long-term secret key of the owner of the target session, before it expires. Therefore, those models are not capable of addressing the key compromise impersonation attacks, whereas the eCK model allows the adversary to learn the long-term secret key of the owner of the target session. Therefore the eCK model addresses the KCI attacks. Moreover, the BR models and the CK model do not allow the adversary to reveal the session states or ephemeral keys of the target session or its partner session. Therefore, those models are not capable of addressing the ephemeral key leakage attacks, whereas the eCK model allows the adversary to reveal both of the ephemeral keys of the target session, as long as the owner and the partner principals to the target session are not corrupted. Therefore the eCK model addresses the ephemeral key reveal attacks. In the CK model, after the target session has expired, the adversary is allowed to learn the long-term secret keys of the protocol participants of the target session, regardless of whether the adversary actively interfered with the target session, whereas the eCK model only allows the adversary to learn the long-term secret keys of both protocol participants of the target session when the adversary has not actively interfered with the target session. Therefore, the CK model addresses the perfect forward secrecy, while the eCK model only addresses the weak perfect forward secrecy. Table 1 summarizes the security features of above discussed security models.

Table 2 Basic characteristics of few eCK-secure key exchange protocols

Protocol	NAXOS trick	Proof model	DH assumption	Generic/concrete
NAXOS [17]	Yes	Random oracle	GDH	Concrete
CMQV [21]	Yes	Random oracle	GDH	Concrete
MO [18]	No	Standard	DDH	Concrete
KFU P1 [16]	No	Random oracle	GDH	Concrete
KFU P2 [16]	No	Random oracle	CDH	Concrete
Yang P1 [22]	No	Standard	Bilinear DDH	Concrete
Yang GC-KKN [22]	No	Standard	DDH	Generic
ASB [2]	No	Random oracle	GDH	Concrete
Protocol P1 (this paper)	No	Standard	DDH	Generic

Likewise, the eCK model is clearly defined to capture most of the demanding security features of key exchange protocols and thus widely used as a strong security model to analyze the security of key exchange protocols. We explain the eCK model in detail in Sect. 3.

1.2 eCK-secure key exchange protocols

The initial effort of constructing the eCK-secure key exchange protocols is combining the long-term secret key and the ephemeral secret key using a random oracle function [5] to obtain a pseudo-ephemeral value. This trick is first introduced by LaMacchia et al. [17] in their protocol, named NAXOS, and now it is widely known as the NAXOS trick. A "pseudo"-ephemeral key $\widetilde{esk}$ is computed as the random oracle function of the long-term key lsk and the actual ephemeral key esk: $\widetilde{esk} \leftarrow H(esk, lsk)$. The value $\widetilde{esk}$ is never stored, and thus, in the eCK model the adversary must learn both esk and lsk in order to be able to compute $\widetilde{esk}$. Note however, that in the NAXOS protocol, the initiator must compute $\widetilde{esk} = H(esk, lsk)$ twice: once when sending its Diffie–Hellman ephemeral public key $g^{\widetilde{esk}}$ and once when computing the Diffie–Hellman shared secrets from the received values. This is to avoid storing a single value that, when compromised, can be used to compute the session key. There are some key exchange protocols created using the NAXOS trick [17,21].

Recently, some researchers worked on constructing eCK-secure key exchange protocols without NAXOS trick [2,16,18,22]. The motivation for such research can be explained as follows: the eCK model addresses the leakage of the ephemeral secret key. It is unnatural to assume that the ephemeral secret key is leaked, while the exponent of the ephemeral public key (e.g., the pseudo- ephemeral value in the NAXOS protocol) remains safe, without leaking. Therefore, it seems that there is an unnatural and indirect assumption of a leakage-free exponentiation computation or leakage-free random source, in the eCK-security proof of the NAXOS-style key exchange protocols. Therefore, eliminating the NAXOS trick and still preserving the eCK security would be more realistic. Moreover, the NAXOS trick is a random oracle-based technique. Favorable things on random oracle-based constructions are that, the schemes are efficient, proofs are clean and the random oracles can be replaced with suitable hash functions in the real- world implementations. On the other hand, random oracle proofs are considered as ideal-world proofs, rather than real-world proofs. Therefore, perhaps cryptographers tend to construct cryptographic schemes which are proven secure in the standard model.

1.3 Our contribution

In this paper our aim is to present a generic eCK-secure, NAXOS-free, standard model key exchange protocol, namely the protocol P1. Thus, our generic protocol is a strongly secure and realistic framework for real-world instantiations. Our protocol is a Diffie–Hellman-style key exchange protocol, and we assume on the hardness of the decisional Diffie–Hellman (DDH) problem. Moreover, our protocol uses an arbitrary CCA2-secure public-key encryption scheme to encrypt Diffie–Hellman public ephemeral values and exchange them between the protocol principals. An arbitrary pseudo-random function is used to derive the secret session key using the ephemeral Diffie–Hellman shared key, long-term Diffie–Hellman shared key and the message flow. Since our protocol is a generic protocol, this can be instantiated with an arbitrary CCA2 public-key encryption scheme and an arbitrary pseudo-random function. Therefore, it is possible to instantiate our protocol with more efficient CCA2-secure public-key encryption schemes and pseudo-random functions in future and achieve better performance. In Table 2, we look at the basic characteristics of few eCK-secure key exchange protocols in the literature, comparing with our new protocol. Table 2 shows that our protocol captures all of the desired features that we discussed here.

Table 3 Cost analysis of few standard model eCK-secure key exchange protocols

Protocol	Overall cost	Instantiation details
MO [18]	3Exp, 1Multi-exp, 1CR, 1πPRF	Concrete
Yang P1 [22]	2Exp, 4Multi-exp, 4Pair, 1PRF, 2TCR	Concrete
Yang GC-KKN [22]	7Exp, 2Multi-exp, 3PRF, 2TCR	Factoring-based KEM
Protocol P1 (this paper)	6Exp, 4Multi-exp, 2PRF	Cramer–Shoup PKE

The protocol execution cost of our protocol is one encryption, one decryption, three exponentiations and two pseudo-random operations. Table 3 shows the protocol execution costs of the standard model protocols that mentioned in Table 2. The generic GC-KKN protocol is instantiated with a factoring-based key encapsulation mechanism as mentioned by Yang [22], and our protocol is instantiated with Cramer–Shoup public-key encryption scheme [9]. In Table 3, CR denotes collision-resistant hash functions, TCR denotes target collision-resistant hash functions, Exp denotes exponentiations, Multi-exp denotes multi-exponentiations, Pair denotes pairings, PRF denotes pseudo-random functions, and π-PRF denotes pseudo-random function with pairwise independent random source. Compared to other protocols mentioned in Table 3, Cramer–Shoup-based instantiation of our protocol needs relatively simple multi-exponentiations and additionally two pseudo-random functions.

To the best of our knowledge, this is the *first paper on generic transformation of a* CCA2*-secure public-key encryption scheme to an* eCK *-secure key exchange protocol in the standard model*.

2 Preliminaries

In this section we review the preliminaries that we use in this paper.

2.1 Pseudo-random functions

We now describe the security definition of pseudo-random functions according to Katz and Lindell [15].

Definition 1 (Pseudo-random functions) Let $F : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$ be an efficient, length preserving, keyed function. F is a *pseudo-random function* if for all PPT adversaries $\mathcal{B}$, there is a negligible function $\text{Adv}_{\text{PRF}}(\mathcal{B})$ in k such that:

$$\left| \Pr[\mathcal{B}^{F(key,\cdot)}(1^k) = 1] - \Pr[\mathcal{B}^{f_{\text{rnd}}(\cdot)}(1^k) = 1] \leq \text{Adv}_{\text{PRF}}(\mathcal{B}) \right|,$$

where the first probability is taken over uniform choice of $key \in \{0,1\}^k$ and the randomness of $\mathcal{B}$, and the second probability is taken over uniform choice of f_{rnd} and randomness of $\mathcal{B}$, and $\mathcal{B}$ is not given a key key.

2.2 Indistinguishability against adaptive chosen ciphertext attacks (CCA2)

A public-key encryption scheme consists of three algorithms as follows:

- KG: This is a PPT algorithm that takes as input the security parameter and outputs a public/secret key pair (pk, sk). This also specifies the message (plaintext) space $\mathcal{M}$ and the ciphertext space $\mathcal{C}$.
- Enc: This is a PPT algorithm that takes as input $m \in \mathcal{M}$ and a public-key pk, and outputs a ciphertext $c \in \mathcal{C}$.
- Dec: This is a deterministic algorithm that takes as input a ciphertext $c \in \mathcal{C}$ and a secret key sk, and outputs either a message $m \in \mathcal{M}$ or the error symbol $\perp$.

A public-key encryption scheme must satisfy the correctness property: for all valid key pairs (pk, sk), if $c = \text{Enc}_{pk}(m)$ for any $m \in \mathcal{M}$, then $\text{Dec}_{sk}(c) = m$.

We now review a strong security notion for public-key encryption schemes: *indistinguishability against adaptive chosen ciphertext attacks* (CCA2), referring Bellare et al. [3].

Definition 2 (Indistinguishability against adaptive chosen ciphertext attacks (CCA2)) Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be any PPT adversary in the security parameter k, against a public-key encryption scheme PKE $=$ (KG, Enc, Dec). The CCA2 security experiment for the public-key encryption scheme PKE, $\text{Exp}^{\text{CCA2}}_{\text{PKE},\mathcal{A}}(1^k)$, is defined as follows:

1. $(pk, sk) \xleftarrow{\$} \text{KG}(1^k)$
2. $(m_0, m_1, state) \leftarrow \mathcal{A}_1^{\text{Dec}(sk,c)}(pk)$ *such that* $|m_0| = |m_1|$
3. $b \xleftarrow{\$} \{0,1\}$
4. $c^* \leftarrow \text{Enc}(pk, m_b)$
5. $b' \leftarrow \mathcal{A}_2^{\text{Dec}(sk,c)_{c^* \neq c}}(pk, c^*, state)$
6. $\mathcal{A}$ wins if $b' = b$

Decryption Oracle

- $\text{Dec}(sk, c) \rightarrow m$ where m is the corresponding plaintext c.
- returns m to $\mathcal{A}$

The public-key encryption scheme PKE is CCA2-secure, if for every PPT adversary $\mathcal{A}$ the advantage of winning the security experiment $\text{Exp}^{\text{CCA2}}_{\text{PKE}}(\mathcal{A})$: $\text{Adv}^{\text{CCA2}}_{\text{PKE}}(\mathcal{A})$, is negligible in the security parameter k.

2.3 Diffie–Hellman assumptions

We now describe two Diffie–Hellman assumptions which form the basis of security for many cryptographic primitives. Let k be the security parameter, $\mathcal{G}$ be a group generation algorithm and $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^k)$, where $\mathbb{G}$ is a cyclic group of prime order q and g is an arbitrary generator of $\mathbb{G}$.

Definition 3 (Computational Diffie–Hellman (CDH) assumption) We say that computational Diffie–Hellman assumption holds in $\mathbb{G}$ if for all PPT algorithms $\mathcal{A}$, the probability of solving the CDH problem in $\mathbb{G}$ given as:

$$\text{Pr}^{\text{CDH}}_{g,q}(\mathcal{A}) = \text{Pr}\left(\mathcal{A}(\mathbb{G}, g, q, g^a, g^b) = g^{ab}\right)$$

is negligible for a given security parameter k.

Definition 4 (Decisional Diffie–Hellman (DDH) assumption) Consider the following two distributions: $\mathcal{DH}_{\mathbb{G}} = \{(g, g^a, g^b, g^{ab}); a, b \xleftarrow{\$} \mathbb{Z}_q\}$ and $\mathcal{R}_{\mathbb{G}} = \{(g, g^a, g^b, g^c); a, b, c \xleftarrow{\$} \mathbb{Z}_q\}$. It is said that DDH assumption holds in $\mathbb{G}$ if for all PPT algorithms $\mathcal{A}$, the advantage in distinguishing the two distributions $\mathcal{DH}$ and $\mathcal{R}$ given as:

$$\text{Adv}^{\text{DDH}}_{g,q}(\mathcal{A}) = \left|\text{Pr}[\mathcal{A}(\mathcal{DH}_{\mathbb{G}}) = 1] - \text{Pr}[\mathcal{A}(\mathcal{R}_{\mathbb{G}}) = 1]\right|$$

is negligible for a given security parameter k.

3 Extended Canetti–Krawczyk model (eCK)

The motivation of LaMacchia et al. [17] in designing the eCK model was that an adversary should have to compromise both the long-term and ephemeral secret keys of a party in order to recover the session key.

Parties and long-term keys Let $\mathcal{U} = \{U_1, \ldots, U_{N_P}\}$ be a set of N_P parties. Each party U_i where $i \in [1, N_P]$ has a pair of long-term public and secret keys, (pk_{U_i}, sk_{U_i}). Each party U_i owns at most N_S number of protocol sessions.

Sessions Each party may run multiple instances of the protocol concurrently or sequentially; we use the term *principal* to refer a party involved in a protocol instance and the term *session* to identify a protocol instance at a principal. The notation $\Pi^s_{U,V}$ represents the sth session at the owner principal U, with intended partner principal V. The principal which sends the first protocol message of a session is the *initiator* of the session, and the principal which responds to the first protocol message is the *responder* of the session. A session $\Pi^s_{U,V}$ enters an *accepted* state when it computes a session key. Note that a session may terminate without ever entering into the accepted state. The information of whether a session has terminated with or without acceptance is public.

Partnering Legitimate execution of a key exchange protocol between two principals U and V makes two partnering sessions owned by U and V, respectively. Two sessions $\Pi^s_{U,V}$ and $\Pi^{s'}_{U',V'}$ are said to be partners if all of the following hold:

1. both $\Pi^s_{U,V}$ and $\Pi^{s'}_{U',V'}$ have computed session keys;
2. messages sent from $\Pi^s_{U,V}$ and messages received by $\Pi^{s'}_{U',V'}$ are identical;
3. messages sent from $\Pi^{s'}_{U',V'}$ and messages received by $\Pi^s_{U,V}$ are identical;
4. $U' = V$ and $V' = U$;
5. Exactly one of U and V is the initiator, and the other is the responder.

The protocol is said to be *correct* if two partner sessions compute identical session keys.

Adversarial powers The adversary $\mathcal{A}$ is a probabilistic polynomial time algorithm in the security parameter k that has the control over the whole network. $\mathcal{A}$ interacts with set of sessions which represent protocol instances. $\mathcal{A}$ can adaptively ask following queries.

- `Send` (U, V, s, m) query—This query allows $\mathcal{A}$ to run the protocol. It sends the message m to the session $\prod^s_{U,V}$ as coming from the session $\prod^{s'}_{V,U}$. $\prod^s_{U,V}$ will return to $\mathcal{A}$ the next message according to the protocol conversation so far or decision on whether to accept or reject the session. $\mathcal{A}$ can also use this query to initiate a new protocol instance with blank m. This query captures capabilities of active adversary, who can initiate sessions and modify or delay protocol messages.
- `SessionKeyReveal` (U, V, s) query—If a session $\prod^s_{U,V}$ has accepted and holds a session key, $\mathcal{A}$ gets the session key of $\prod^s_{U,V}$. A session can only accept a session key once. This query captures the known key attacks.
- `EphemeralKeyReveal` (U, V, s) query—Gives all the ephemeral keys (per session randomness) of the session $\prod^s_{U,V}$ to $\mathcal{A}$.
- `Corrupt` (U) query—$\mathcal{A}$ gets all the long-term secrets of the principal U. But this query does not reveal any session keys to $\mathcal{A}$. This query captures the key compromise impersonation (KCI) attacks, unknown key share (UKS) attacks and forward secrecy.

– `Test` (U, s) query—Once a session $\prod_{U,V}^{s}$ has accepted and holds a session key, $\mathcal{A}$ can attempt to distinguish it from a random key. When $\mathcal{A}$ asks the `Test` query, the session $\prod_{U,V}^{s}$ first chooses a random bit $b \in \{0, 1\}$ and if $b = 1$, the actual session key is returned to $\mathcal{A}$; otherwise, a random session key is chosen uniformly at random from the same session-key distribution and is returned to $\mathcal{A}$. This query is only allowed to be asked once.

Freshness A session $\prod_{U,V}^{s}$ is said to be *fresh* if and only if all of the following hold:

1. The session $\prod_{U,V}^{s}$ and its partner (if it exists), $\prod_{V,U}^{s'}$ have not been asked the `Session- Key reveal` query.
2. If partner $\prod_{V,U}^{s'}$ exists none of the following combinations have been asked:
   (a) `Corrupt`(U) and `EphemeralKeyReveal` (U, V, s)
   (b) `Corrupt`(V) and `EphemeralKeyReveal` (V, U, s')
3. If partner $\prod_{V,U}^{s'}$ does not exist, none of the following combinations have been asked
   (a) `Corrupt`(V)
   (b) `Corrupt`(U) and `EphemeralKeyReveal` (U, V, s)

Security game

– *Stage 0:* The challenger generates the keys by using the security parameter k.
– *Stage 1:* $\mathcal{A}$ is executed and may ask any of `Send`, `SessionKeyReveal`, `EphemeralKeyReveal`, `Corrupt` queries to any session at will.
– *Stage 2:* At some point $\mathcal{A}$ chooses a fresh session and asks the `Test` query.
– *Stage 3:* $\mathcal{A}$ continue asking `Send`, `SessionKeyReveal`, `EphemeralKeyReveal`, `Corrupt` queries. The only condition is that $\mathcal{A}$ cannot violate the freshness of the test session.
– *Stage 4:* At some point $\mathcal{A}$ outputs the bit $b' \in \{0, 1\}$ which is its guess of the value b on the test session. $\mathcal{A}$ wins if $b' = b$.

Definition of security Let $\text{Succ}_{\mathcal{A}}$ be the event that the adversary $\mathcal{A}$ wins the eCK game.

Definition 5 A protocol (π) is said to be secure in the eCK model if there is no PPT adversary $\mathcal{A}$ who can win the eCK game with non-negligible advantage in the security parameter k. The advantage of an adversary $\mathcal{A}$ is defined as:

$$\text{Adv}_{\pi}^{\text{eCK}}(\mathcal{A}) = |2\text{Pr}(\text{Succ}_{\mathcal{A}}) - 1| \ .$$

4 Generic eCK-secure key exchange in the standard model

In this work we construct a generic eCK-secure key exchange protocol, in the standard model, using an arbitrary CCA2-secure public-key encryption scheme and an arbitrary pseudo-random function. We prove the security of our protocol in the standard model, assuming the hardness of the DDH problem.

4.1 Construction of the generic protocol P1

The protocol P1 shown in Table 4 is a Diffie–Hellman-style [10] key agreement protocol. Let k be the security parameter and group $\mathbb{G}$ be generated using a group generation algorithm which takes k as an input, where $\mathbb{G}$ be a group of prime order q with generator g. We use an arbitrary CCA2-secure public-key encryption scheme PKE $=$ (KG, Enc, Dec) to encrypt protocol messages. Given the security parameter k, KG computes a pair of secret/public keys. Let sk_{Alice}, pk_{Alice} be the secret/public encryption keys of Alice and sk_{Bob}, pk_{Bob} be the secret/public encryption keys of Bob. Let a, A and b, B are the Diffie–Hellman long-term secret and public keys of Alice and Bob, respectively, while x, X and y, Y are the Diffie–Hellman ephemeral secret and public keys of Alice and Bob, respectively. After exchanging the protocol messages ($\bar{X} \xleftarrow{\$} \text{Enc}_{pk_{\text{Bob}}}(X)$ and $\bar{Y} \xleftarrow{\$} \text{Enc}_{pk_{\text{Alice}}}(Y)$), both principals decrypt the incoming messages and compute a Diffie–Hellman-style shared secrets (Y^x, B^a and X^y, A^b) and then compute the session key using a pseudo-random function PRF.

4.2 Security analysis of the protocol P1

Theorem 1 *If $\mathbb{G}$ is a group of a prime order q and a generator g, where the Diffie–Hellman* (DDH) *assumption holds, the underlying public-key encryption scheme* PKE *is* CCA2*-secure and* PRF *is a pseudo-random function, then the protocol* P1 *is secure in the* eCK *model.*

Let $\mathcal{U} = \{U_1, \ldots, U_{N_P}\}$ be a set of N_P parties. Each party U_i owns at most N_s number of protocol sessions. Let $\mathcal{A}$ be any adversary against the eck challenger of the protocol P1*. Then, the advantage of $\mathcal{A}$ against the* eCK *security challenge of the protocol* P1*, $\text{Adv}_{\text{P1}}^{\text{eCK}}$ is:*

$$\text{Adv}_{\text{P1}}^{\text{eCK}}(\mathcal{A}) \leq N_P^2 N_s^2 \max\Big(\big(\text{Adv}_{q,g}^{\text{DDH}}(\mathcal{C}) + \text{Adv}_{\text{PRF}}(\mathcal{B})\big), \big(\text{Adv}_{\text{PRF}}(\mathcal{B}) + \text{Adv}_{\text{PKE}}^{\text{CCA2}}(\mathcal{D})\big)\Big) \ .$$

where $\mathcal{C}$ is the algorithm against a DDH *challenger, $\mathcal{B}$ is the algorithm against the underlying pseudo-random function* PRF *and $\mathcal{D}$ is the algorithm against the* CCA2 *challenger of the underlying public-key encryption scheme* PKE*.*

Table 4 Protocol P1

Alice (initiator)		*Bob* (responder)
	Initial setup	
$a \xleftarrow{\$} \mathbb{Z}_q^*, A \leftarrow g^a$		$b \xleftarrow{\$} \mathbb{Z}_q^*, B \leftarrow g^b$
$sk_{\text{Alice}}, pk_{\text{Alice}} \xleftarrow{\$} \text{KG}(1^k)$		$sk_{\text{Bob}}, pk_{\text{Bob}} \xleftarrow{\$} \text{KG}(1^k)$
	Protocol execution	
$x \xleftarrow{\$} \mathbb{Z}_q^*, X \leftarrow g^x$		$y \xleftarrow{\$} \mathbb{Z}_q^*, Y \leftarrow g^y$
$\bar{X} \xleftarrow{\$} \text{Enc}_{pk_{\text{Bob}}}(X)$	$\xrightarrow{Alice, \bar{X}}$	$\bar{Y} \xleftarrow{\$} \text{Enc}_{pk_{\text{Alice}}}(Y)$
	$\xleftarrow{Bob, \bar{Y}}$	
$Y \leftarrow \text{Dec}_{sk_{\text{Alice}}}(\bar{Y})$		$X \leftarrow \text{Dec}_{sk_{\text{Bob}}}(\bar{X})$
$Z_1 \leftarrow Y^x, Z_2 \leftarrow B^a$		$Z_1' \leftarrow X^y, Z_2' \leftarrow A^b$
$K \leftarrow \text{PRF}(Z_1, Alice\|\bar{X}\|Bob\|\bar{Y}) \oplus$		$K \leftarrow \text{PRF}(Z_1', Alice\|\bar{X}\|Bob\|\bar{Y}) \oplus$
$\text{PRF}(Z_2, Alice\|\bar{X}\|Bob\|\bar{Y})$		$\text{PRF}(Z_2', Alice\|\bar{X}\|Bob\|\bar{Y})$
	K is the session key	

Proof We split the proof of Theorem 1 into two main cases: when the partner to the test session exists and when it does not. □

1. A partner to the test session exists.
   (a) Adversary corrupts both the owner and the partner principals to the test session—Case **1a**
   (b) Adversary corrupts neither the owner nor the partner principal to the test session—Case **1b**
   (c) Adversary corrupts the owner to the test session, but does not corrupt the partner to the test session—Case **1c**
   (d) Adversary corrupts the partner to the test session, but does not corrupt the owner to the test session—Case **1d**
2. A partner to the test session does not exist: the adversary is not allowed to corrupt the peer to the target session.
   (a) Adversary corrupts the owner to the test session—Case **2a**
   (b) Adversary does not corrupt the owner to the test session- Case **2b**

Case 1a Adversary corrupts both the owner and partner principals to the test session.

Game 1: This is the original game. When `Test` query is asked the Game 1 challenger will choose a random bit $b \xleftarrow{\$} \{0, 1\}$. If $b = 1$, the real session key is given to $\mathcal{A}$; otherwise, a random value chosen from the same session-key space is given. Hence,

$$\text{Adv}_{\text{Game 1}}(\mathcal{A}) = \text{Adv}^{\text{eCK}}_{\text{P1,Case 1a}}(\mathcal{A}). \quad (1)$$

Game 2: Same as Game 1 with the following exception: before $\mathcal{A}$ begins, two distinct random principals $U^*, V^* \xleftarrow{\$} \{U_1, \ldots, U_{N_P}\}$ are chosen and two random numbers $s^*, t^* \xleftarrow{\$} \{1, \ldots N_s\}$ are chosen, where N_P is the number of protocol principals and N_s is the number of sessions on a principal. The session $\Pi^{s^*}_{U^*,V^*}$ is chosen as the target session, and the session $\Pi^{t^*}_{V^*,U^*}$ is chosen as the partner to the target session. If the test session is not the session $\Pi^{s^*}_{U^*,V^*}$ or partner to the session is not $\Pi^{t^*}_{V^*,U^*}$, the Game 2 challenger aborts the game. Unless the incorrect choice happens, the Game 2 is identical to the Game 1. Hence,

$$\text{Adv}_{\text{Game 2}}(\mathcal{A}) = \frac{1}{{N_P}^2 N_s^2} \text{Adv}_{\text{Game 1}}(\mathcal{A}). \quad (2)$$

Game 3: Same as Game 2 with the following exception: the Game 3 challenger randomly chooses $z \xleftarrow{\$} \mathbb{Z}^{*q}$ and computes $K \leftarrow \text{PRF}(g^z, \cdot\|\bar{X}\|\cdot\|\bar{Y}) \oplus \text{PRF}(\text{CDH}(U, V), \cdot\|\bar{X}\|\cdot\|\bar{Y})$. When the adversary asks the $\texttt{Test}(U^*, V^*, s^*)$ query, Game 3 challenger will answer with K ($\cdot$ is used as a placeholder since either U^* or V^* can be put there depending on the initiator and responder roles).

Note 1 Let U, V be the two long-term Diffie–Hellman public keys of the protocol principals U^*, V^*, respectively, such that $U = g^u$, $V = g^v$ and $\text{CDH}(U, V) = g^{uv}$.

We construct an algorithm $\mathcal{C}$ against a DDH challenger, using the adversary $\mathcal{A}$ as a subroutine. $\mathcal{C}$ sets all the long-term secret/public key pairs (Diffie–Hellman and encryption key pairs) to all protocol principals. The algorithm $\mathcal{C}$ runs a copy of $\mathcal{A}$ and interacts with $\mathcal{A}$ such that $\mathcal{A}$ is interacting with either Game 2 or Game 3. The DDH challenger sends values (g^x, g^y, g^z) such that either $z = xy$ or $z \xleftarrow{\$} \mathbb{Z}_q^*$, as the

inputs to the algorithm $\mathcal{C}$. Algorithm $\mathcal{C}$ simulates answers to the adversarial queries as follows:

- $\texttt{Send}(U, V, s, m)$ query:
  - If U^* is the initiator, $\mathcal{C}$ sends the ciphertext $\bar{X} \xleftarrow{\$} (pk_{V^*}, X)$ to $\mathcal{A}$ as the first message of the test session. Upon receiving the second protocol message ($\bar{Y} \xleftarrow{\$} (pk_{U^*}, Y)$) from V^* to U^*, $\mathcal{C}$ computes the session key $K \leftarrow \text{PRF}(g^z, U^*\|\bar{X}\|V^*\|\bar{Y}) \oplus \text{PRF}(\text{CDH}(U, V), U^*\|\bar{X}\|V^*\|\bar{Y})$.
  - If U^* is the responder, upon receiving the first protocol message ($\bar{X} \xleftarrow{\$} (pk_{U^*}, X)$) from V^* to U^*, $\mathcal{C}$ sends $\bar{Y} \xleftarrow{\$} (pk_{V^*}, Y)$ to $\mathcal{A}$ as the second protocol message of the test session and computes the session key $K \leftarrow \text{PRF}(g^z, V^*\|\bar{X}\|U^*\|\bar{Y}) \oplus \text{PRF}(\text{CDH}(U, V), V^*\|\bar{X}\|U^*\|\bar{Y})$.

Note 2 For clarity we consider $\bar{X} \xleftarrow{\$} (pk., X)$ as the first protocol message and $\bar{Y} \xleftarrow{\$} (pk., Y)$ as the second protocol message, in the target session.

- For all the other cases of `Send` queries, $\mathcal{C}$ can decrypt incoming protocol messages and execute the protocol normally.
- $\texttt{SessionKeyReveal}(U, V, s)$ query: `SessionKeyReveal` query is not allowed to the target session or the partner of the target session. $\mathcal{C}$ can compute all the other session keys by executing the protocol normally.
- $\texttt{EphemeralKeyReveal}(U, V, s)$ query: $U = U^*, V = V^*, s = s^*$ and $U = V^*, V = U^*, s = t^*$ are prohibited since the adversary is allowed to corrupt both the owner and the partner to the target session. For all other `EphemeralKeyReveal` queries $\mathcal{C}$ can answer correctly, because $\mathcal{C}$ has the ephemeral keys.
- $\texttt{Corrupt}(U)$ query: Algorithm $\mathcal{C}$ can answer all the `Corrupt` queries, since $\mathcal{C}$ has all the long-term keys.
- $\texttt{Test}(U, V, s)$ query: When $U = U^*, V = V^*, s = s^*$, answers with the K which is computed as explained in the `Send` query. Otherwise aborts the game.

If $\mathcal{C}$'s input is a Diffie–Hellman triple, simulation constructed by $\mathcal{C}$ is identical to Game 2; otherwise, it is identical to Game 3. If $\mathcal{A}$ can distinguish the difference between games, then $\mathcal{C}$ can answer the DDH challenge. Hence,

$$|\text{Adv}_{\text{Game 2}}(\mathcal{A}) - \text{Adv}_{\text{Game 3}}(\mathcal{A})| \leq \text{Adv}^{\text{DDH}}_{q,g}(\mathcal{C}). \quad (3)$$

Game 4: Same as Game 3 with the following exception: the Game 4 challenger randomly chooses $K \xleftarrow{\$} \{0, 1\}^k$ and sends it to the adversary $\mathcal{A}$ as the answer to the $\texttt{Test}(U^*, V^*, s^*)$ query.

If $\mathcal{A}$ can distinguish the difference between Game 3 and Game 4, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the session-key value K is computed using the real PRF with a hidden key, or using a random function. The adversary $\mathcal{A}$ is given a K, such that it is computed using the PRF or randomly chosen from the session-key space. The following describes $\mathcal{B}$?s procedure of answering queries.

- $\texttt{Send}(U, V, s, m)$ query:
  - If U^* is the initiator, upon receiving the second protocol message ($\bar{Y}$) computes the session key $K \leftarrow \text{Oracle}^{\text{PRF}}(U^*||\bar{X}||V^*||\bar{Y}) \oplus \text{PRF}(\text{CDH}(U, V), U^*||\bar{X}||V^*||\bar{Y})$.
  - If U^* is the responder, upon receiving the first protocol message ($\bar{X}$) computes the session key $K \leftarrow \text{Oracle}^{\text{PRF}}(V^*||\bar{X}||U^*||\bar{Y}) \oplus \text{PRF}(\text{CDH}(U, V), V^*||\bar{X}||U^*||\bar{Y})$.
  - For all the other cases of `Send` queries, $\mathcal{B}$ can execute the protocol normally.
- $\texttt{SessionKeyReveal}(U, V, s)$ query: `SessionKeyReveal` query is not allowed to the target session or its partner. $\mathcal{B}$ can compute all the session keys by executing the protocol normally.
- $\texttt{EphemeralKeyReveal}(U, V, s)$ query: $U = U^*, V = V^*, s = s^*$) and $U = V^*, V = U^*, s = t^*$ are prohibited since the adversary is allowed to corrupt both the owner and the partner to the target session. For all other `EphemeralKeyReveal` queries $\mathcal{B}$ can answer correctly, because $\mathcal{B}$ has the ephemeral keys.
- $\texttt{Corrupt}(U)$ query: $\mathcal{B}$ can answer all other `Corrupt` queries, since $\mathcal{B}$ has all the long-term secret keys.
- $\texttt{Test}(U, V, s)$ query: When $U = U^*, V = V^*, s = s^*$, answers with the K which is computed as explained in the `Send` query. Otherwise aborts the game.

If the oracle is using the real PRF with a hidden key, the simulation is identical to Game 3, whereas if the oracle is using a random function, the simulation constructed is identical to Game 4. If $\mathcal{A}$ can distinguish the difference between Game 3 and Game 4, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the PRF challenger is real or random. Hence,

$$|\text{Adv}_{\text{Game 3}}(\mathcal{A}) - \text{Adv}_{\text{Game 4}}(\mathcal{A})| \leq \text{Adv}_{\text{PRF}}(\mathcal{B}). \quad (4)$$

Semantic security of the session key in Game 4: Since the session key K of $\Pi^{s^*}_{U^*,V^*}$ is chosen randomly and independently from all other values, $\mathcal{A}$ does not have any advantage in Game 4. Hence,

$$\mathrm{Adv}_{\mathrm{Game}\ 4}(\mathcal{A}) = 0. \tag{5}$$

Using Eqs. (1)–(5) we find,

$$\mathrm{Adv}^{\mathrm{eCK}}_{\mathrm{P1,Case\ 1a}}(\mathcal{A}) \leq {N_P}^2 {N_s}^2 \Big(\mathrm{Adv}^{\mathrm{DDH}}_{q,g}(\mathcal{C}) + \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{B})\Big).$$

Case 1b Adversary corrupts neither the owner nor the partner principals to the test session.

Game 1: This is the original game. When `Test` query is asked the Game 1 challenger will choose a random bit $b \xleftarrow{\$} \{0, 1\}$. If $b = 1$, the real session key is given to $\mathcal{A}$; otherwise, a random value chosen from the same session-key space is given. Hence,

$$\mathrm{Adv}_{\mathrm{Game}\ 1}(\mathcal{A}) = \mathrm{Adv}^{\mathrm{eCK}}_{\mathrm{P1,Case\ 1b}}(\mathcal{A}). \tag{6}$$

Game 2: Same as Game 1 with the following exception: before $\mathcal{A}$ begins, two distinct random principals $U^*, V^* \xleftarrow{\$} \{U_1, \ldots, U_{N_P}\}$ are chosen and two random numbers $s^*, t^* \xleftarrow{\$} \{1, \ldots N_s\}$ are chosen, where N_P is the number of protocol principals and N_s is the number of sessions on a principal. The session $\Pi^{s^*}_{U^*,V^*}$ is chosen as the target session, and the session $\Pi^{t^*}_{V^*,U^*}$ is chosen as the partner to the target session. If the test session is not the session $\Pi^{s^*}_{U^*,V^*}$ or partner to the session is not $\Pi^{t^*}_{V^*,U^*}$, the Game 2 challenger aborts the game. Unless the incorrect choice happens, the Game 2 is identical to the Game 1. Hence,

$$\mathrm{Adv}_{\mathrm{Game}\ 2}(\mathcal{A}) = \frac{1}{{N_P}^2 N_s^2} \mathrm{Adv}_{\mathrm{Game}\ 1}(\mathcal{A}). \tag{7}$$

Game 3: Same as Game 2 with the following exception: the Game 3 challenger randomly chooses $c \xleftarrow{\$} \mathbb{Z}^*_q$ and computes $K \leftarrow \mathrm{PRF}(\mathrm{CDH}(X, Y), \cdot \|\bar{X}\| \cdot \|\bar{Y}) \oplus \mathrm{PRF}(g^c, \cdot \|\bar{X}\| \cdot \|\bar{Y})$. When the adversary asks the $\mathtt{Test}(U^*, V^*, s^*)$ query, Game 3 challenger will answer with K.

Note 3 Let X, Y be the two ephemeral Diffie–Hellman public keys (unencrypted) of the protocol principals in a session, such that $X = g^x$, $Y = g^y$ and $\mathrm{CDH}(X, Y) = g^{xy}$.

We construct an algorithm $\mathcal{C}$ against a DDH challenger, using the adversary $\mathcal{A}$ as a subroutine. The DDH challenger sends values (g^a, g^b, g^c) such that either $c = ab$ or $c \xleftarrow{\$} \mathbb{Z}^*_q$, as the inputs to the algorithm $\mathcal{C}$. $\mathcal{C}$ sets $U \leftarrow g^a$ as the long-term Diffie–Hellman public key of U^*, and $V \leftarrow g^b$ as the long-term Diffie–Hellman public key of V^*. Moreover, $\mathcal{C}$ sets all the other long-term Diffie–Hellman secret/public key pairs and all the encryption key pairs of protocol principals. The algorithm $\mathcal{C}$ runs a copy of $\mathcal{A}$ and interacts with $\mathcal{A}$ such that $\mathcal{A}$ is interacting with either Game 2 or Game 3. Algorithm $\mathcal{C}$ simulates answers to the adversarial queries as follows:

- $\mathtt{Send}(U, V, s, m)$ query:
  - If U^* is the initiator, $\mathcal{C}$ can start the protocol normally. Upon receiving the second protocol message from V^* to U^*, $\mathcal{C}$ computes the session key $K \leftarrow \mathrm{PRF}(\mathrm{CDH}(X, Y), U^*\|\bar{X}\|V^*\|\bar{Y}) \oplus \mathrm{PRF}(g^c, U^*\|\bar{X}\|V^*\|\bar{Y})$ (consider X is from the initiator and Y is from the responder, and $\bar{X}$ and $\bar{Y}$ are the encrypted X and Y, respectively, which are computed in normal protocol run).
  - If U^* is the responder, upon receiving the first protocol message from V^* to U^*, $\mathcal{C}$ executes the protocol normally, sends the second protocol message of the test session and computes the session key $K \leftarrow \mathrm{PRF}(\mathrm{CDH}(X, Y), V^*\|\bar{X}\|U^*\|\bar{Y}) \oplus \mathrm{PRF}(g^c, V^*\|\bar{X}\|U^*\|\bar{Y})$.
  - For all the cases of `Send` queries, $\mathcal{C}$ can decrypt incoming protocol messages and execute the protocol normally. In the places where both U^* and V^* are involved, $\mathcal{C}$ uses g^c in key derivation.
- $\mathtt{SessionKeyReveal}(U, V, s)$ query: `SessionKeyReveal` query is not allowed to the target session or the partner of the target session. $\mathcal{C}$ can compute all the other session keys by executing the protocol normally. In the places where both U^* and V^* are involved, $\mathcal{C}$ uses g^c in key derivation.
- $\mathtt{EphemeralKeyReveal}(U, V, s)$ query: Algorithm $\mathcal{C}$ can answer all the other `EphemeralKeyReveal` queries, since $\mathcal{C}$ has all the ephemeral keys.
- $\mathtt{Corrupt}(U)$ query: $\mathtt{Corrupt}(U^*)$ and $\mathtt{Corrupt}(V^*)$ are prohibited since the adversary is allowed to reveal the ephemeral keys of the test session and its partner. Algorithm $\mathcal{C}$ can answer all the other `Corrupt` queries, since $\mathcal{C}$ has all the long-term keys.
- $\mathtt{Test}(U, V, s)$ query: When $U = U^*$, $V = V^*$, $s = s^*$, answers with the K which is computed as explained in the `Send` query. Otherwise aborts the game.

If $\mathcal{C}$'s input is a Diffie–Hellman triple, simulation constructed by $\mathcal{C}$ is identical to Game 2; otherwise, it is identical to Game 3. If $\mathcal{A}$ can distinguish the difference between games, then $\mathcal{C}$ can answer the DDH challenge. Hence,

$$|\mathrm{Adv}_{\mathrm{Game}\ 2}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game}\ 3}(\mathcal{A})| \leq \mathrm{Adv}^{\mathrm{DDH}}_{q,g}(\mathcal{C}). \tag{8}$$

Game 4: Same as Game 3 with the following exception: the Game 4 challenger randomly chooses $K \xleftarrow{\$} \{0, 1\}^k$ and sends it to the adversary $\mathcal{A}$ as the answer to the $\mathtt{Test}(U^*, V^*, s^*)$ query.

If $\mathcal{A}$ can distinguish the difference between Game 3 and Game 4, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the session-key value K is computed using the real PRF with a hidden key or using a random function. The adversary $\mathcal{A}$ is given a K, such that it is computed using the PRF or randomly chosen from the session-key space. The following describes $\mathcal{B}$?s procedure of answering queries.

- $\texttt{Send}(U, V, s, m)$ query:
  - If U^* is the initiator, upon receiving the second protocol message computes the session key $K \leftarrow \text{PRF}(\text{CDH}(X, Y), U^*||\bar{X}||V^*||\bar{Y}) \oplus \text{Oracle}^{\text{PRF}}(U^*||\bar{X}||V^*||\bar{Y})$.
  - If U^* is the responder, upon receiving the first protocol message computes the session key $K \leftarrow \text{PRF}(\text{CDH}(X, Y), V^*||\bar{X}||U^*||\bar{Y}) \oplus \text{Oracle}^{\text{PRF}}(V^*||\bar{X}||U^*||\bar{Y})$.
  - When both U^* and V^* involve in a session query the $\text{Oracle}^{\text{PRF}}$ to compute the session key upon receiving protocol messages.
  - For all the other cases of `Send` queries, $\mathcal{B}$ can execute the protocol normally.
- $\texttt{SessionKeyReveal}(U, V, s)$ query: `SessionKeyReveal` query is not allowed to the target session or its partner. $\mathcal{B}$ can compute all the session keys as explained under the `Send` query description.
- $\texttt{EphemeralKeyReveal}(U, V, s)$ query: $\mathcal{B}$ can answer all `EphemeralKeyReveal` queries correctly, because $\mathcal{C}$ has the ephemeral keys.
- $\texttt{Corrupt}(U)$ query: $\texttt{Corrupt}(U^*)$ and $\texttt{Corrupt}(V^*)$ are prohibited since the adversary is allowed to reveal the ephemeral keys of the test session and its partner. Algorithm $\mathcal{C}$ can answer all the other `Corrupt` queries, since $\mathcal{C}$ has all the long-term keys.
- $\texttt{Test}(U, V, s)$ query: When $U = U^*, V = V^*, s = s^*$, answers with the K which is computed as explained in the `Send` query. Otherwise aborts the game.

If the oracle is using the real PRF with a hidden key, the simulation is identical to Game 3, whereas if the oracle is using a random function, the simulation constructed is identical to Game 4. If $\mathcal{A}$ can distinguish the difference between Game 3 and Game 4, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the PRF challenger is real or random. Hence,

$$|\text{Adv}_{\text{Game 3}}(\mathcal{A}) - \text{Adv}_{\text{Game 4}}(\mathcal{A})| \leq \text{Adv}_{\text{PRF}}(\mathcal{B}). \tag{9}$$

Semantic security of the session key in Game 4: Since the session key K of $\Pi^{s^*}_{U^*,V^*}$ is chosen randomly and independently from all other values, $\mathcal{A}$ does not have any advantage in Game 4. Hence,

$$\text{Adv}_{\text{Game 4}}(\mathcal{A}) = 0. \tag{10}$$

Using Eqs. (6)–(10) we find,

$$\text{Adv}^{\text{eCK}}_{\text{P1,Case 1b}}(\mathcal{A}) \leq {N_P}^2 {N_s}^2 \Big(\text{Adv}^{\text{DDH}}_{q,g}(\mathcal{C}) + \text{Adv}_{\text{PRF}}(\mathcal{B})\Big).$$

Case 1c Adversary corrupts the owner to the test session, but does not corrupt the partner.

Game 1: This is the original game. When `Test` query is asked the Game 1 challenger will choose a random bit $b \xleftarrow{\$} \{0, 1\}$. If $b = 1$, the real session key is given to $\mathcal{A}$; otherwise, a random value chosen from the same session-key space is given. Hence,

$$\text{Adv}_{\text{Game 1}}(\mathcal{A}) = \text{Adv}^{\text{eCK}}_{\text{P1,Case 1c}}(\mathcal{A}). \tag{11}$$

Game 2: Same as Game 1 with the following exception: before $\mathcal{A}$ begins, two distinct random principals $U^*, V^* \xleftarrow{\$} \{U_1, \ldots, U_{N_P}\}$ are chosen and two random numbers $s^*, t^* \xleftarrow{\$} \{1, \ldots N_s\}$ are chosen, where N_P is the number of protocol principals and N_s is the number of sessions on a principal. The session $\Pi^{s^*}_{U^*,V^*}$ is chosen as the target session, and the session $\Pi^{t^*}_{V^*,U^*}$ is chosen as the partner to the target session. If the test session is not the session $\Pi^{s^*}_{U^*,V^*}$ or partner to the session is not $\Pi^{t^*}_{V^*,U^*}$, the Game 2 challenger aborts the game. Unless the incorrect choice happens, Game 2 is identical to Game 1. Hence,

$$\text{Adv}_{\text{Game 2}}(\mathcal{A}) = \frac{1}{{N_P}^2 N_s^2} \text{Adv}_{\text{Game 1}}(\mathcal{A}). \tag{12}$$

Game 3: Same as Game 2 with the following exception: the Game 3 challenger randomly chooses C from the ciphertext space as encryption of the public ephemeral value X of the session $\Pi^{s^*}_{U^*,V^*}$ and sends it to the session $\Pi^{t^*}_{V^*,U^*}$ as having come from the session $\Pi^{s^*}_{U^*,V^*}$.

We introduce an algorithm $\mathcal{D}$ which is constructed using the adversary $\mathcal{A}$. If $\mathcal{A}$ can distinguish the difference between Game 2 and Game 3, then $\mathcal{D}$ can be used against the CCA2 challenger of underlying public-key cryptosystem, PKE. The algorithm $\mathcal{D}$ uses the public key of the CCA2 challenger as the public key of the protocol principal V^* and generates all other public/secret key pairs (Diffie–Hellman and encryption keys) for protocol principals. $\mathcal{D}$ runs a copy of $\mathcal{A}$ and interacts with $\mathcal{A}$, such that it is interacting with either Game 2 or Game 3. $\mathcal{D}$ picks two random strings, $X_0, X_1 \xleftarrow{\$} \mathbb{Z}^*_q$ and passes them to the CCA2 challenger. From the CCA2

challenger, $\mathcal{D}$ receives a challenge ciphertext C such that $C \xleftarrow{\$}(pk_{V^*}, X_\theta)$ where $X_\theta = X_0$ or $X_\theta = X_1$. $\mathcal{D}$ uses X_1 as the decryption of C when answering queries. The following describes the procedure of answering queries:

- `Send`(U, V, s, m) query:
  - $U = U^*, V = V^*, s = s^*$:
    - If U^* is the initiator, $\mathcal{D}$ sends the ciphertext C to $\mathcal{A}$ as the first message of the test session. Upon receiving the second protocol message computes the session key $K \leftarrow \text{PRF}(\text{CDH}(X_1, Y), U^* \| C \| V^* \| \bar{Y}) \oplus \text{PRF}(\text{CDH}(U, V), U^* \| C \| V^* \| \bar{Y})$ (consider Y is from the responder, and $\bar{Y}$ is the encrypted Y).
    - If U^* is the responder, upon receiving the first protocol message sends C to $\mathcal{A}$ and computes the session key $K \leftarrow \text{PRF}(\text{CDH}(X_1, Y), V^* \| \bar{Y} \| U^* \| C) \oplus \text{PRF}(\text{CDH}(U, V), V^* \| \bar{Y} \| U^* \| C)$ (consider Y is from the initiator, and $\bar{Y}$ is the encrypted Y).
  - $U = U^*, V = V^*, s \neq s^*$: Executes the protocol normally.
  - $U = U^*, V \neq V^*$: Executes the protocol normally.
  - $U = V^*$:
    - If this is the initiator and it is the first message, then it executes the protocol normally.
    - If this is the initiator and the second protocol message, or the responder:
      - If C has come as the incoming message uses X_1 as the decryption of the incoming message.
      - Else uses the decryption oracle to decrypt incoming messages.
  - $U, V \neq U^*$ or V^*: Executes the protocol normally.
- `SessionKeyReveal`(U, V, s) query: `SessionKeyReveal` query is not allowed to the target session or the partner of the target session. $\mathcal{D}$ can compute all the session keys by executing the protocol.
  - For sessions involving the principal V^*, and the incoming message to V^* is the same message which has come to V^* in the target session, uses X_1 as the decryption.
  - For other sessions involving the principal V^*, $\mathcal{D}$ can decrypt the incoming messages to V^* by using the decryption oracle.
  - Otherwise, $\mathcal{D}$ can decrypt all the other incoming messages to protocol principals by its own.

  Then compute the session key using the PRF.
- `EphemeralKeyReveal`(U, V, s) query: $\mathcal{D}$ can answer all `EphemeralKeyReveal` queries allowed in the freshness condition correctly, because $\mathcal{D}$ has the ephemeral keys.
- `Corrupt`(U) query: Except for `Corrupt`(V^*), algorithm $\mathcal{D}$ can answer all other `Corrupt` queries. In this case we consider the situation in which the adversary is not allowed to corrupt the partner principal of the target session, so in fact, $\mathcal{D}$ can answer all legitimate `Corrupt` queries.
- `Test`(U, V, s) query: When $U = U^*, V = V^*, s = s^*$, answers with the K which is computed as explained in the `Send` query. Otherwise aborts the game.

If the value C is the encryption of the value X_1, the simulation constructed by $\mathcal{D}$ is identical to the Game 2; otherwise, it is identical to Game 3. If $\mathcal{A}$ can distinguish the difference between games, then $\mathcal{D}$ can answer the CCA2 challenge successfully. Hence,

$$|\text{Adv}_{\text{Game 2}}(\mathcal{A}) - \text{Adv}_{\text{Game 3}}(\mathcal{A})| \leq \text{Adv}^{\text{CCA2}}_{\text{PKE}}(\mathcal{D}). \tag{13}$$

Game 4: Same as Game 3 with the following exception: the Game 4 challenger randomly chooses $K \xleftarrow{\$}\{0, 1\}^k$ and sends it to the adversary $\mathcal{A}$ as the answer to the `Test`(U^*, V^*, s^*) query.

If $\mathcal{A}$ can distinguish the difference between Game 3 and Game 4, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the session-key value K is computed using the real PRF with a hidden key or using a random function. The adversary $\mathcal{A}$ is given a K, such that it is computed using the PRF or randomly chosen from the session-key space. The following describes $\mathcal{B}$'s procedure of answering queries.

- `Send`(U, V, s, m) query:
  - $U = U^*, V = V^*, s = s^*$:
    - If U^* is the initiator, upon receiving the second protocol message computes the session key $K \leftarrow \text{Oracle}^{\text{PRF}}(U^* || \bar{X} || V^* || \bar{Y}) \oplus \text{PRF}(\text{CDH}(U, V), U^* || \bar{X} || V^* || \bar{Y})$ (consider X is from the initiator and Y is from the responder, and $\bar{X}$ and $\bar{Y}$ are the encrypted X and Y, respectively, which are computed in normal protocol run).
    - If U^* is the responder, upon receiving the first protocol message computes the session key $K \leftarrow \text{Oracle}^{\text{PRF}}(V^* || \bar{X} || U^* || \bar{Y}) \oplus \text{PRF}(\text{CDH}(U, V), V^* || \bar{X} || U^* || \bar{Y})$.
  - $U = U^*, V = V^*, s \neq s^*$: Executes the protocol normally.
  - $U = U^*, V \neq V^*$: Executes the protocol normally.
  - $U = V^*$:
    - If this is the initiator and it is the first message, then it executes the protocol normally.

- If this is the initiator and the second protocol message, or the responder:
  - If the same message that came to V^* in the test session has come as the incoming message computes the session key using the $\text{Oracle}^{\text{PRF}}$.
  - Otherwise, executes the protocol normally.

– $U, V \neq U^*$ or V^*: Executes the protocol normally.

– `SessionKeyReveal`(U, V, s) query: `SessionKeyReveal` query is not allowed to the target session or its partner. $\mathcal{B}$ can compute all the session keys by executing the protocol.
  – For sessions involving the principal V^*, and the incoming message to V^* is the same message which has come to V^* in the target session, $\mathcal{B}$ uses $\text{Oracle}^{\text{PRF}}$ to compute the session key.
  – For all other sessions, $\mathcal{B}$ computes the session key by using the PRF.

– `EphemeralKeyReveal`(U, V, s) query: $\mathcal{B}$ can answer all `EphemeralKeyReveal` queries, which are allowed by the freshness condition, because $\mathcal{B}$ has the ephemeral keys.

– `Corrupt`(U) query: Except for V^*, algorithm $\mathcal{B}$ can answer all other `Corrupt` queries. In this case we consider the situation in which the adversary is not allowed to corrupt the partner principal of the target session, so in fact, $\mathcal{B}$ can answer all legitimate `Corrupt` queries.

– `Test`(U, V, s) query: When $U = U^*$, $V = V^*$, $s = s^*$, answers with the K which is computed as explained in the `Send` query. Otherwise aborts the game.

If the oracle is using the real PRF with a hidden key, the simulation is identical to Game 3, whereas if the oracle is using a random function, the simulation constructed is identical to Game 4. If $\mathcal{A}$ can distinguish the difference between Game 3 and Game 4, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the PRF challenger is real or random. Hence,

$$|\text{Adv}_{\text{Game 3}}(\mathcal{A}) - \text{Adv}_{\text{Game 4}}(\mathcal{A})| \leq \text{Adv}_{\text{PRF}}(\mathcal{B}). \tag{14}$$

Semantic security of the session key in Game 4: Since the session key K of $\Pi^{s^*}_{U^*,V^*}$ is chosen randomly and independently from all other values, $\mathcal{A}$ does not have any advantage in Game 4. Hence,

$$\text{Adv}_{\text{Game 4}}(\mathcal{A}) = 0. \tag{15}$$

Using Eqs. (11)–(15) we find,

$$\text{Adv}^{\text{eCK}}_{\text{P1,Case 1c}}(\mathcal{A}) \leq N_P^2 {N_s}^2 \Big(\text{Adv}_{\text{PRF}}(\mathcal{B}) + \text{Adv}^{\text{CCA2}}_{\text{PKE}}(\mathcal{D})\Big).$$

Case 1d Adversary corrupts the partner to the test session, but does not corrupt the owner.

Game 1: This is the original game. When `Test` query is asked the Game 1 challenger will choose a random bit $b \xleftarrow{\$} \{0, 1\}$. If $b = 1$, the real session key is given to $\mathcal{A}$; otherwise, a random value chosen from the same session-key space is given. Hence,

$$\text{Adv}_{\text{Game 1}}(\mathcal{A}) = \text{Adv}^{\text{eCK}}_{\text{P1,Case 1d}}(\mathcal{A}). \tag{16}$$

Game 2: Same as Game 1 with the following exception: before $\mathcal{A}$ begins, two distinct random principals $U^*, V^* \xleftarrow{\$} \{U_1, \ldots, U_{N_P}\}$ are chosen and two random numbers $s^*, t^* \xleftarrow{\$} \{1, \ldots N_s\}$ are chosen, where N_P is the number of protocol principals and N_s is the number of sessions on a principal. The session $\Pi^{s^*}_{U^*,V^*}$ is chosen as the target session, and the session $\Pi^{t^*}_{V^*,U^*}$ is chosen as the partner to the target session. If the test session is not the session $\Pi^{s^*}_{U^*,V^*}$ or partner to the session is not $\Pi^{t^*}_{V^*,U^*}$, the Game 2 challenger aborts the game. Unless the incorrect choice happens, Game 2 is identical to Game 1. Hence,

$$\text{Adv}_{\text{Game 2}}(\mathcal{A}) = \frac{1}{{N_P}^2 N_s^2} \text{Adv}_{\text{Game 1}}(\mathcal{A}). \tag{17}$$

Game 3: Same as Game 2 with the following exception: the Game 3 challenger randomly chooses C from the ciphertext space as encryption of the public ephemeral value X of the session $\Pi^{s^*}_{U^*,V^*}$, and sends it to the session $\Pi^{s^*}_{U^*,V^*}$ as having come from the session $\Pi^{t^*}_{V^*,U^*}$.

We introduce an algorithm $\mathcal{D}$ which is constructed using the adversary $\mathcal{A}$. If $\mathcal{A}$ can distinguish the difference between Game 2 and Game 3, then $\mathcal{D}$ can be used against the CCA2 challenger of underlying public-key cryptosystem, PKE. The algorithm $\mathcal{D}$ uses the public key of the CCA2 challenger as the public key of the protocol principal U^* and generates all other public/secret key pairs (Diffie–Hellman and encryption keys) for protocol principals. $\mathcal{D}$ runs a copy of $\mathcal{A}$ and interacts with $\mathcal{A}$, such that it is interacting with either Game 2 or Game 3. $\mathcal{D}$ picks two random strings, $X_0, X_1 \xleftarrow{\$} \mathbb{Z}^*_q$ and passes them to the CCA2 challenger. From the CCA2 challenger, $\mathcal{D}$ receives a challenge ciphertext C such that $C \xleftarrow{\$} (pk_{V^*}, X_\theta)$ where $X_\theta = X_0$ or $X_\theta = X_1$. $\mathcal{D}$ uses X_1 as the decryption of C when answering queries. The following describes the procedure of answering queries:

– `Send`(U, V, s, m) query:
  – $U = V^*, V = U^*, s = t^*$:

- If V^* is the initiator, $\mathcal{D}$ sends the ciphertext C to $\mathcal{A}$ as the first message of the test session. Upon receiving the second protocol message computes the session key $K \leftarrow \mathrm{PRF}(\mathrm{CDH}(X_1, Y), V^* \| C \| U^* \| \bar{Y}) \oplus \mathrm{PRF}(\mathrm{CDH}(U, V), V^* \| C \| U^* \| \bar{Y})$ (consider Y is from the responder, and $\bar{Y}$ is the encrypted Y).
- If V^* is the responder, upon receiving the first protocol message sends C to $\mathcal{A}$ and computes the session key $K \leftarrow \mathrm{PRF}(\mathrm{CDH}(X_1, Y), U^* \| \bar{Y} \| V^* \| C) \oplus \mathrm{PRF}(\mathrm{CDH}(U, V), U^* \| \bar{Y} \| V^* \| C)$ (consider Y is from the initiator, and $\bar{Y}$ is the encrypted Y).

– $U = V^*, V = U^*, s \neq t^*$: Executes the protocol normally.
– $U = V^*, V \neq U^*$: Executes the protocol normally.
– $U = U^*$:
  - If this is the initiator and it is the first message, then executes the protocol normally.
  - If this is the initiator and the second protocol message, or the responder:
    · If C has come as the incoming message uses X_1 as the decryption of the incoming message.
    · Else uses the decryption oracle to decrypt incoming messages.
– $U, V \neq U^*$ or V^*: Executes the protocol normally.

– `SessionKeyReveal`(U, V, s) query: `SessionKeyReveal` query is not allowed to the target session or the partner of the target session. $\mathcal{D}$ can compute all the session keys by executing the protocol.

  – For sessions involving the principal U^*, and the incoming message to U^* is the same message which has come to U^* in the target session, uses X_1 as the decryption.
  – For other sessions involving the principal U^*, $\mathcal{D}$ can decrypt the incoming messages to U^* by using the decryption oracle.
  – Otherwise, $\mathcal{D}$ can decrypt all the other incoming messages to protocol principals by its own.

  Then compute the session key using the PRF.
– `EphemeralKeyReveal`(U, V, s) query: $\mathcal{D}$ can answer all `EphemeralKeyReveal` queries allowed in the freshness condition correctly, because $\mathcal{D}$ has the ephemeral keys.
– `Corrupt`(U) query: Except for `Corrupt`(U^*), algorithm $\mathcal{D}$ can answer all other `Corrupt` queries. In this case we consider the situation in which the adversary is not allowed to corrupt the partner principal of the target session, so in fact, $\mathcal{D}$ can answer all legitimate `Corrupt` queries.
– `Test`(U, V, s) query: When $U = U^*, V = V^*, s = s^*$, answers with the K which is computed as explained in the `Send` query. Otherwise aborts the game.

If the value C is the encryption of the value X_1, the simulation constructed by $\mathcal{D}$ is identical to the Game 2; otherwise, it is identical to Game 3. If $\mathcal{A}$ can distinguish the difference between games, then $\mathcal{D}$ can answer the CCA2 challenge successfully. Hence,

$$|\mathrm{Adv}_{\mathrm{Game}\,2}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game}\,3}(\mathcal{A})| \leq \mathrm{Adv}^{\mathrm{CCA2}}_{\mathrm{PKE}}(\mathcal{D}). \quad (18)$$

Game 4: Same as Game 3 with the following exception: the Game 4 challenger randomly chooses $K \xleftarrow{\$} \{0, 1\}^k$ and sends it to the adversary $\mathcal{A}$ as the answer to the `Test`(U^*, V^*, s^*) query.

If $\mathcal{A}$ can distinguish the difference between Game 3 and Game 4, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the session-key value K is computed using the real PRF with a hidden key, or using a random function. The adversary $\mathcal{A}$ is given a K, such that it is computed using the PRF or randomly chosen from the session-key space. The following describes $\mathcal{B}$'s procedure of answering queries.

– `Send`(U, V, s, m) query:
  – $U = V^*, V = U^*, s = t^*$:
    - If V^* is the initiator, upon receiving the second protocol message computes the session key $K \leftarrow \mathrm{Oracle}^{\mathrm{PRF}}(V^* || \bar{X} || U^* || \bar{Y}) \oplus \mathrm{PRF}(\mathrm{CDH}(U, V), V^* || \bar{X} || U^* || \bar{Y})$ (consider X is from the initiator and Y is from the responder, and $\bar{X}$ and $\bar{Y}$ are the encrypted X and Y, respectively, which are computed in normal protocol run).
    - If V^* is the responder, upon receiving the first protocol message computes the session key $K \leftarrow \mathrm{Oracle}^{\mathrm{PRF}}(U^* || \bar{X} || V^* || \bar{Y}) \oplus \mathrm{PRF}(\mathrm{CDH}(U, V), U^* || \bar{X} || V^* || \bar{Y})$.
  – $U = V^*, V = U^*, s \neq t^*$: Executes the protocol normally.
  – $U = V^*, V \neq U^*$: Executes the protocol normally.
  – $U = U^*$:
    - If this is the initiator and it is the first message, then it executes the protocol normally.
    - If this is the initiator and the second protocol message, or the responder:
      · If the same message that came to U^* in the test session has come as the incoming message, computes the session key using the $\mathrm{Oracle}^{\mathrm{PRF}}$.
      · Otherwise, executes the protocol normally.
  – $U, V \neq U^*$ or V^*: Executes the protocol normally.

- `SessionKeyReveal`(U, V, s) query: `SessionKeyReveal` query is not allowed to the target session or its partner. $\mathcal{B}$ can compute all the session keys by executing the protocol.
  - For sessions involving the principal U^*, and the incoming message to U^* is the same message which has come to U^* in the target session, $\mathcal{B}$ uses Oracle$^{\text{PRF}}$ to compute the session key.
  - For all other sessions, $\mathcal{B}$ computes the session key by using the PRF.
- `EphemeralKeyReveal`(U, V, s) query: $\mathcal{B}$ can answer all `EphemeralKeyReveal` queries, which are allowed by the freshness condition, because $\mathcal{B}$ has the ephemeral keys.
- `Corrupt`(U) query: Except for U^*, algorithm $\mathcal{B}$ can answer all other `Corrupt` queries. In this case we consider the situation in which the adversary is not allowed to corrupt the partner principal of the target session, so in fact, $\mathcal{B}$ can answer all legitimate `Corrupt` queries.
- `Test`(U, V, s) query: When $U = U^*$, $V = V^*$, $s = s^*$, answers with the K which is computed as explained in the `Send` query. Otherwise aborts the game.

If the oracle is using the real PRF with a hidden key, the simulation is identical to Game 3, whereas if the oracle is using a random function, the simulation constructed is identical to Game 4. If $\mathcal{A}$ can distinguish the difference between Game 3 and Game 4, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the PRF challenger is real or random. Hence,

$$|\text{Adv}_{\text{Game 3}}(\mathcal{A}) - \text{Adv}_{\text{Game 4}}(\mathcal{A})| \leq \text{Adv}_{\text{PRF}}(\mathcal{B}). \tag{19}$$

Semantic security of the session key in Game 4: Since the session key K of $\Pi^{s^*}_{U^*,V^*}$ is chosen randomly and independently from all other values, $\mathcal{A}$ does not have any advantage in Game 4. Hence,

$$\text{Adv}_{\text{Game 4}}(\mathcal{A}) = 0. \tag{20}$$

Using Eqs. (16)–(20) we find,

$$\text{Adv}^{\text{eCK}}_{\text{P1,Case 1d}}(\mathcal{A}) \leq {N_P}^2 {N_s}^2 \Big(\text{Adv}_{\text{PRF}}(\mathcal{B}) + \text{Adv}^{\text{CCA2}}_{\text{PKE}}(\mathcal{D})\Big).$$

Case 2a Adversary corrupts the owner to the test session.

Game 1: This is the original game. When `Test` query is asked the Game 1 challenger will choose a random bit $b \xleftarrow{\$} \{0, 1\}$. If $b = 1$, the real session key is given to $\mathcal{A}$; otherwise, a random value chosen from the same session-key space is given. Hence,

$$\text{Adv}_{\text{Game 1}}(\mathcal{A}) = \text{Adv}^{\text{eCK}}_{\text{P1,Case 2a}}(\mathcal{A}). \tag{21}$$

Game 2: Same as Game 1 with the following exception: before $\mathcal{A}$ begins, two distinct random principals $U^*, V^* \xleftarrow{\$} \{U_1, \ldots, U_{N_P}\}$ are chosen and two random number $s^*, t^* \xleftarrow{\$} \{1, \ldots N_s\}$ are chosen, where N_P is the number of protocol principals and N_s is the number of sessions on a principal. The session $\Pi^{s^*}_{U^*,V^*}$ is chosen as the target session, and the session $\Pi^{t^*}_{V^*,U^*}$ is chosen as the peer session. If the test session is not the session $\Pi^{s^*}_{U^*,V^*}$, the Game 2 challenger aborts the game. Unless the incorrect choice happens, Game 2 is identical to Game 1. Hence,

$$\text{Adv}_{\text{Game 2}}(\mathcal{A}) = \frac{1}{{N_P}^2 N_s^2} \text{Adv}_{\text{Game 1}}(\mathcal{A}). \tag{22}$$

Game 3: Same as Game 2 with the following exception: the Game 3 challenger randomly chooses C from the ciphertext space as encryption of the public ephemeral value X of the session $\Pi^{s^*}_{U^*,V^*}$, and sends it to the session $\Pi^{t^*}_{V^*,U^*}$ as having come from the session $\Pi^{s^*}_{U^*,V^*}$.

We introduce an algorithm $\mathcal{D}$ which is constructed using the adversary $\mathcal{A}$. If $\mathcal{A}$ can distinguish the difference between Game 2 and Game 3, then $\mathcal{D}$ can be used against the CCA2 challenger of underlying public-key cryptosystem, PKE. The algorithm $\mathcal{D}$ uses the public key of the CCA2 challenger as the public key of the protocol principal V^* and generates all other public/secret key pairs (Diffie–Hellman and encryption keys) for protocol principals. $\mathcal{D}$ runs a copy of $\mathcal{A}$ and interacts with $\mathcal{A}$, such that it is interacting with either Game 2 or Game 3. $\mathcal{D}$ picks two random strings, $X_0, X_1 \xleftarrow{\$} {\mathbb{Z}^*}_q$ and passes them to the CCA2 challenger. From the CCA2 challenger, $\mathcal{D}$ receives a challenge ciphertext C such that $C \xleftarrow{\$} (pk_{V^*}, X_\theta)$ where $X_\theta = X_0$ or $X_\theta = X_1$. $\mathcal{D}$ uses X_1 as the decryption of C when answering queries. The way of answering the queries is the same as in the Game 3 of Case 1c.

If the value C is the encryption of the value X_1, the simulation constructed by $\mathcal{D}$ is identical to the Game 2; otherwise, it is identical to Game 3. If $\mathcal{A}$ can distinguish the difference between games, then $\mathcal{D}$ can answer the CCA2 challenge successfully. Hence,

$$|\text{Adv}_{\text{Game 2}}(\mathcal{A}) - \text{Adv}_{\text{Game 3}}(\mathcal{A})| \leq \text{Adv}^{\text{CCA2}}_{\text{PKE}}(\mathcal{D}). \tag{23}$$

Game 4: Same as Game 3 with the following exception: the Game 4 challenger randomly chooses $K \xleftarrow{\$} \{0, 1\}^k$ and sends it to the adversary $\mathcal{A}$ as the answer to the `Test`(U^*, V^*, s^*) query.

If $\mathcal{A}$ can distinguish the difference between Game 3 and Game 4, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the session-key value K is computed using the real PRF with a hidden key, or using a random function. The adversary $\mathcal{A}$ is given a K, such that it is computed using the PRF or randomly chosen from the session-key space. The way of answering the queries is the same as in the Game 4 of Case 1c.

If the oracle is using the real PRF with a hidden key, the simulation is identical to Game 3, whereas if the oracle is using a random function, the simulation constructed is identical to Game 4. If $\mathcal{A}$ can distinguish the difference between Game 3 and Game 4, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the PRF challenger is real or random. Hence,

$$|\mathrm{Adv}_{\text{Game 3}}(\mathcal{A}) - \mathrm{Adv}_{\text{Game 4}}(\mathcal{A})| \leq \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{B}). \tag{24}$$

Semantic security of the session key in Game 4: Since the session key K of $\Pi^{s^*}_{U^*,V^*}$ is chosen randomly and independently from all other values, $\mathcal{A}$ does not have any advantage in Game 4. Hence,

$$\mathrm{Adv}_{\text{Game 4}}(\mathcal{A}) = 0. \tag{25}$$

Using Eqs. (21)–(25) we find,

$$\mathrm{Adv}^{\mathrm{eCK}}_{\text{P1,Case 2a}}(\mathcal{A}) \leq {N_P}^2 {N_s}^2 \Big(\mathrm{Adv}_{\mathrm{PRF}}(\mathcal{B}) + \mathrm{Adv}^{\mathrm{CCA2}}_{\mathrm{PKE}}(\mathcal{D})\Big).$$

Case 2b Adversary does not corrupt the owner to the test session.

Game 1: This is the original game. When `Test` query is asked the Game 1 challenger will choose a random bit $b \xleftarrow{\$} \{0, 1\}$. If $b = 1$, the real session key is given to $\mathcal{A}$; otherwise, a random value chosen from the same session-key space is given. Hence,

$$\mathrm{Adv}_{\text{Game 1}}(\mathcal{A}) = \mathrm{Adv}^{\mathrm{eCK}}_{\text{P1,Case 2b}}(\mathcal{A}). \tag{26}$$

Game 2: Same as Game 1 with the following exception: before $\mathcal{A}$ begins, two distinct random principals $U^*, V^* \xleftarrow{\$} \{U_1, \ldots, U_{N_P}\}$ are chosen and two random numbers $s^*, t^* \xleftarrow{\$} \{1, \ldots N_s\}$ are chosen, where N_P is the number of protocol principals and N_s is the number of sessions on a principal. The session $\Pi^{s^*}_{U^*,V^*}$ is chosen as the target session, and the session $\Pi^{t^*}_{V^*,U^*}$ is chosen as the peer session. If the test session is not the session $\Pi^{s^*}_{U^*,V^*}$, the Game 2 challenger aborts the game. Unless the incorrect choice happens, the Game 2 is identical to the Game 1. Hence,

$$\mathrm{Adv}_{\text{Game 2}}(\mathcal{A}) = \frac{1}{{N_P}^2 N_s^2} \mathrm{Adv}_{\text{Game 1}}(\mathcal{A}). \tag{27}$$

Game 3: Same as Game 2 with the following exception: the Game 3 challenger randomly chooses $c \xleftarrow{\$} \mathbb{Z}^*_q$ and computes $K \leftarrow \mathrm{PRF}(\mathrm{CDH}(X, Y), \cdot \| \bar{X} \| \cdot \| \bar{Y}) \oplus \mathrm{PRF}(g^c, \cdot \| \bar{X} \| \cdot \| \bar{Y})$. When the adversary asks the $\mathtt{Test}(U^*, V^*, s^*)$ query, Game 3 challenger will answer with K.

Note 4 Let X, Y be the two ephemeral Diffie–Hellman public keys (unencrypted) of the protocol principals in a session, such that $X = g^x$, $Y = g^y$ and $\mathrm{CDH}(X, Y) = g^{xy}$.

We construct an algorithm $\mathcal{C}$ against a DDH challenger, using the adversary $\mathcal{A}$ as a subroutine. The DDH challenger sends values (g^a, g^b, g^c) such that either $c = ab$ or $c \xleftarrow{\$} \mathbb{Z}^*_q$, as the inputs to the algorithm $\mathcal{C}$. $\mathcal{C}$ sets $U \leftarrow g^a$ as the long-term Diffie–Hellman public key of U^* and $V \leftarrow g^b$ as the long-term Diffie–Hellman public key of V^*. Moreover, $\mathcal{C}$ sets all the other long-term Diffie–Hellman secret/public key pairs and all the encryption key pairs of protocol principals. The algorithm $\mathcal{C}$ runs a copy of $\mathcal{A}$ and interacts with $\mathcal{A}$ such that $\mathcal{A}$ is interacting with either Game 2 or Game 3. The way of answering the queries is the same as in the Game 3 of Case 1b.

If $\mathcal{C}$'s input is a Diffie–Hellman triple, simulation constructed by $\mathcal{C}$ is identical to Game 2; otherwise, it is identical to Game 3. If $\mathcal{A}$ can distinguish the difference between games, then $\mathcal{C}$ can answer the DDH challenge. Hence,

$$|\mathrm{Adv}_{\text{Game 2}}(\mathcal{A}) - \mathrm{Adv}_{\text{Game 3}}(\mathcal{A})| \leq \mathrm{Adv}^{\mathrm{DDH}}_{q,g}(\mathcal{C}). \tag{28}$$

Game 4: Same as Game 3 with the following exception: the Game 4 challenger randomly chooses $K \xleftarrow{\$} \{0, 1\}^k$ and sends it to the adversary $\mathcal{A}$ as the answer to the $\mathtt{Test}(U^*, V^*, s^*)$ query.

If $\mathcal{A}$ can distinguish the difference between Game 3 and Game 4, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the session-key value K is computed using the real PRF with a hidden key, or using a random function. The adversary $\mathcal{A}$ is given a K, such that it is computed using the PRF or randomly chosen from the session-key space. The way of answering the queries is the same as in the Game 4 of Case 1b.

If the oracle is using the real PRF with a hidden key, the simulation is identical to Game 3, whereas if the oracle is using a random function, the simulation constructed is identical to Game 4. If $\mathcal{A}$ can distinguish the difference between Game 3 and Game 4, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the PRF challenger is real or random. Hence,

$$|\mathrm{Adv}_{\text{Game 3}}(\mathcal{A}) - \mathrm{Adv}_{\text{Game 4}}(\mathcal{A})| \leq \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{B}). \tag{29}$$

Semantic security of the session key in Game 4: Since the session key K of $\Pi^{s^*}_{U^*,V^*}$ is chosen randomly and independently from all other values, $\mathcal{A}$ does not have any advantage in Game 4. Hence,

$$\mathrm{Adv}_{\text{Game 4}}(\mathcal{A}) = 0. \tag{30}$$

Using Eqs. (26)–(30) we find,

$$\mathrm{Adv}^{\mathrm{eCK}}_{\mathrm{P1,Case\ 2b}}(\mathcal{A}) \leq N_P^2 {N_s}^2 \Big(\mathrm{Adv}^{\mathrm{DDH}}_{q,g}(\mathcal{C}) + \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{B})\Big).$$

Combining all the above cases.

According to the analysis we can see the adversary $\mathcal{A}$'s advantage of winning against the eCK challenger of the protocol P1 is:

$$\mathrm{Adv}^{\mathrm{eCK}}_{\mathrm{P1}}(\mathcal{A}) \leq N_P^2 N_s^2 \max\Big(\big(\mathrm{Adv}^{\mathrm{DDH}}_{q,g}(\mathcal{C}) + \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{B})\big), \big(\mathrm{Adv}_{\mathrm{PRF}}(\mathcal{B}) + \mathrm{Adv}^{\mathrm{CCA2}}_{\mathrm{PKE}}(\mathcal{D})\big)\Big).$$

5 Conclusion and future works

In this paper we presented a generic eCK-secure, NAXOS-free, standard model key exchange protocol, namely the protocol P1. Thus, our generic protocol is a strongly secure and realistic framework for real-world instantiations. The protocol execution cost of our protocol is one encryption, one decryption, three exponentiations and two pseudo-random operations. Cramer–Shoup-based instantiation of our protocol needs relatively simple multi-exponentiations and additionally two pseudo-random functions.

As a future work authors would like to focus on leakage-resilient improvements on the protocol P1. The essential modification would be replacing the CCA2 public-key encryption scheme with a suitable leakage-resilient CCA2-secure public-key encryption scheme and using a leakage-resilient mechanism to compute the exponentiation operations, in places where the long-term Diffie–Hellman secret keys are used as exponents. Several strong eCK-style leakage-resilient security models have been introduced in the literature [1,19], which would be useful to analyze the leakage-resilient security of the improved protocol. There are several standard model leakage-resilient CCA2-secure public-key encryption schemes in the literature [13,20], which can be used to replace the CCA2 public-key encryption scheme. In order to compute the Diffie–Hellman exponentiations in leakage-resilient manner, the mechanism used by Alawatugoda et al. [2] would be appropriate, which is influenced by the leakage-resilient storage scheme of Dziembowski and Faust [12]. Thus, it is possible to improve this generic protocol to achieve leakage resiliency.

Acknowledgements I would like to acknowledge Colin Boyd, Douglas Stebila and Tatsuaki Okamoto for valuable discussions on authenticated key exchange protocols. Moreover, I am grateful to the handling editor Sherman S. M. Chow and the two anonymous reviewers for their valuable comments to polish-up the paper. Further, I am supported by the National Research Council (NRC), Sri Lanka Postdoctoral Fellowship grant NRC 16-020.

References

1. Alawatugoda, J., Stebila, D., Boyd, C.: Modelling after-the-fact leakage for key exchange. In: 9th ACM Symposium on Information, Computer and Communications Security (ASIA CCS'14), Kyoto, Japan, June 03–06, 2014, pp. 207–216 (2014)
2. Alawatugoda, J., Stebila, D., Boyd, C.: Continuous after-the-fact leakage-resilient eck-secure key exchange. In: Proceedings of the Cryptography and Coding—15th IMA International Conference (IMACC 2015), Oxford, UK, December 15–17, 2015, pp. 277–294 (2015)
3. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In: CRYPTO, pp. 26–45 (1998)
4. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: CRYPTO, pp. 232–249 (1993)
5. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 93, pp. 62–73. ACM Press, New York City (1993)
6. Bellare, M., Rogaway, P.: Provably Secure Session Key Distribution—The Three Party Case. ACM Press, New York City (1995)
7. Boyko, V., MacKenzie, P., Patel, S.: Provably secure password-authenticated key exchange using Diffie–Hellman. In: Proceedings of the 19th International Conference on Theory and Application of Cryptographic Techniques (EUROCRYPT'00), pp. 156–171. Springer, Berlin (2000)
8. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: EUROCRYPT, pp. 453–474 (2001)
9. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO'98, volume 1462 of LNCS, pp. 13–25. Springer, Berlin (1998)
10. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. Inf. Theory **22**, 644–654 (1976)
11. Diffie, W., van Oorschot, P.C., Wiener, M.J.: Authentication and authenticated key exchanges. Des. Codes Cryptogr. **2**(2), 107–125 (1992)
12. Dziembowski, S., Faust, S.: Leakage-resilient cryptography from the inner-product extractor. In: ASIACRYPT, pp. 702–721 (2011)
13. Halevi, S., Lin, H.: After-the-fact leakage in public-key encryption. In: Theory of Cryptology Conference, pp. 107–124 (2011)
14. Jablon, D.P.: Strong password-only authenticated key exchange. SIGCOMM Comput. Commun. Rev. **26**(5), 5–26 (1996)
15. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. Chapman and Hall/CRC Press, Boca Raton (2007)
16. Kim, M., Fujioka, A., Ustaoglu, B.: Strongly secure authenticated key exchange without naxos' approach. In: Proceedings of the Advances in Information and Computer Security, 4th International Workshop on Security, IWSEC 2009, Toyama, Japan, October 28–30, 2009, pp. 174–191 (2009)
17. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: ProvSec, pp. 1–16 (2007)
18. Moriyama, D., Okamoto, T.: An eck-secure authenticated key exchange protocol without random oracles. In: Proceedings of the

Provable Security, Third International Conference, ProvSec 2009, Guangzhou, China, November 11–13, 2009, pp. 154–167 (2009)
19. Moriyama, D., Okamoto, T.: Leakage resilient eCK-secure key exchange protocol without random oracles. In: ASIACCS, pp. 441–447 (2011)
20. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: CRYPTO, pp. 18–35 (2009)
21. Ustaoglu, B.: Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. Des. Codes Cryptogr. **46**(3), 329–342 (2008)
22. Yang, Z.: Efficient eck-secure authenticated key exchange protocols in the standard model. In: Proceedings of the Information and Communications Security—15th International Conference (ICICS 2013), Beijing, China, November 20–22, 2013, pp. 185–193 (2013)