eCK Secure Single Round ID-based Authenticated Key Exchange Protocols with Master Perfect Forward Secrecy (Extended Version)*

Tapas Pandit[1]†, Rana Barua[1], and Somanath Tripathy[2]
[1]*Indian Statistical Institute, Kolkata, India*
{tapasgmmath, ranabarua.isi}@gmail.com
[2]*Indian Institute of Technology, Patna, India*
som@iitp.ac.in

Abstract

Recently, LaMacchia, Lauter and Mityagin proposed the extended Canetti-Krawczyk (eCK) model for Authenticated Key Exchange (AKE) protocols that covers many attacks on existing models. An ID-based AKE protocol with *Perfect Forward Secrecy* (PFS) (respectively *Master Perfect Forward Secrecy* (MPFS)) ensures that revelation of the *static keys* of the parties (respectively the *master secret key* of the private key generator), must not compromise even a single bit of the session keys of the past sessions between the parties. Currently, to the best of our knowledge, there is no ID-based eCK secure single round AKE protocol with either PFS or MPFS. In our preliminary version [18] we have proposed, *without proofs*, ID-based eCK secure single round AKE protocols with PFS and MPFS in the random oracle model. To achieve this, we also have constructed ID-based eCK secure single round AKE protocols, one without Master Forward Secrecy (MFS) and one with MFS, almost at the same computational cost as the existing efficient ID-based eCK Secure Single Round AKE protocols. In this full version, we provide proofs to show that all of our protocols are secure under the Gap Bilinear Diffie-Hellman (GBDH) problem.

Keywords: Authenticated Key Exchange, ID-based cryptography, eCK-secure, perfect forward secrecy

1 Introduction

Authenticated key exchange is a cryptographic primitive that plays an important role in secure communication. Key establishment (KE) is a primitive which allows two or more parties to establish a common key called a session key, which they can use for secure communication. If a common session key is established by two parties U_i and U_j and no other parties learn the established session key, then the KE protocol is called an *authenticated key exchange protocol*.

Diffie and Hellman in [8] introduced peer-to-peer key exchange protocol. But it suffers from the man-in-the-middle attack because of lack of user authentication. This issue is addressed by combining key agreement protocol and certificate based digital signature to achieve authenticated key agreement. But the weakness of the certificate based key agreement protocol is to keep all the desired certificates in a secure place, namely, with the certificate authority. In 1984, Shamir [20] proposed an alternative idea of ID-based primitives, where the public key is the identity. The corresponding private key is generated by

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, volume: 5, number: 4, pp. 65-85

*This paper is an extended version of the work originally presented at the 2014 Network and System Security (NSS'14), Xi'an, China, October 15-17, 2014 ([18]).

†Corresponding author: Stat-Math Unit, Indian Statistical Institute, Kolkata, Pin - 700108, India, Tel: +91-33-2575-3461

a trusted third party designated as private key generator (PKG). Thus, the identity-based cryptosystems [1, 19] simplify the process of key management.

Currently, many identity-based key agreement protocols using pairings have been proposed [13, 21, 16, 4, 23, 3, 5, 22, 10, 9]. A few have been proposed [12, 17, 10, 9] that are claimed to be eCK Secure.

The eCK model proposed by LaMacchia, Lauter and Mityagin does not cover the Perfect Forward Secrecy and the Master Perfect Forward Secrecy. An AKE protocol with PFS [11, 7], ensures that even if the static keys of parties are revealed by the active adversary (unlike weak PFS), the session keys of the past sessions between the parties are not compromised. In other words, the adversary is allowed to reveal the static keys of the parties but after the completion of the test session by the parties. Similarly, an ID-based AKE with MPFS protects the session keys of the past sessions even if the master secret key is exposed to the adversary. Here, the adversary is actively involved in choosing the message of its own choice (unlike MFS). The adversary is not permitted to query the ephemeral keys for the test session and its matching session.

Using some authentication mechanism, e.g., signature scheme, one round eCK secure AKE protocol with PFS may be possible as was pointed out by Cas Cremers et al. [7]. Huang et al. in [11] stated, following the observation of LaMacchia et al. in [14], that the generic signature scheme is not adequate for one round eCK secure AKE protocol with PFS. In fact, an adversary against one round AKE protocol using randomized signature scheme may learn the static key if the adversary reveals these random coins and impersonates the honest party. This requires that the signature scheme must be deterministic for this purpose. Another requirement is that the static key of the AKE protocol should commute with the static key of the hired signature scheme. All these are in the PKI setting.

On the other hand, in the ID-based setting, managing one round eCK security with either PFS or MPFS is harder since static keys of the parties are connected via the master secret key. Till date, to the best our knowledge, there is no ID-based eCK secure single round AKE protocol with either PFS or MPFS. Therefore, any ID-based eCK secure single round AKE protocol with either PFS or MPFS will be a welcome addition to the list of the existing schemes.

1.1 Our Contribution

In our preliminary version [18], we have proposed eCK secure ID-based single round AKE protocols, Π_2 with PFS and Π_4, Π_5, Π_6 with MPFS from the GBDH problem. For achieving this, we have also constructed eCK secure ID-based single round AKE protocols, Π_1 without MFS and Π_3 with MFS, at almost the same computational cost as the existing efficient eCK secure ID-based AKE protocols. (For detailed comparison, see Table 1). In [18] no proofs were provided. In this full version, we have provided all the proofs to show that all these AKE protocols are secure under the GBDH problem. The main challenging task seems to be to define the ephemeral public key and pre-session key components (i.e., the part to be hashed to compute the final session key), so that the hard problem can be solved by these pre-session key components if a PPT adversary breaks this protocol.

Table 1: Efficiency Comparison

Protocol	Pre-Comp	Post-Comp	eCK Security	MFS	PFS	MPFS	Assmpn
$Chow-Choo_1$[5]	–	1P+3SM+2A	X	X	X	X	BDH
$Chow-Choo_2$[5]	–	1P+5SM+2A	X	✓	X	X	MBDH
$Huang-Cao$ [12]	–	2P+3SM+4A	✓	✓	X	X	BDH
A. Fujioka et al.[10, 9]	IP	1P+3SM+2A	✓	✓	X	X	GBDH
Our Scheme Π_1	1P	1P+1SM+2E	✓	X	X	X	GBDH
Our Scheme Π_2	1P	1P+1SM+2E	✓	X	✓	X	GBDH
Our Scheme Π_3	1P	1P+3SM+2E	✓	✓	X	X	GBDH
Our Scheme Π_4	2P	2P+3SM+2E+1$\mathscr{O}$	✓	✓	✓	✓	GBDH
Our Scheme Π_5	1P	2P+3SM+1E+1$\mathscr{O}$	✓	✓	✓	✓	GBDH
Our Scheme Π_6	1P	3P+3SM+1E	✓	✓	✓	✓	GBDH

To explain the computational cost we use some notation: P for bilinear pairing, SM for scalar multiplication on $\mathbb{G}$, A for addition of two points of the bilinear group $\mathbb{G}$, E for exponentiation in $\mathbb{G}_T$ and $\mathscr{O}$ stands for decisional bilinear Diffie-Hellman test. Pre-Comp stands for the pre-computation i.e., the computations before choosing the ephemeral key (i.e., independent of ephemeral key). Post-Comp denotes the rest of the computations. Note that in PFS and MPFS, the adversary is actively involved, whereas, in wPFS and MFS, it is passive. Assmpn stands for Hardness Assumption.

1.2 Related Work

Two efficient ID-based AKE protocols based on their challenge response signature technique were proposed by Chow et al. [5]. They claimed that their protocol supports Session Key Reveal queries in all cases and Ephemeral Key Reveal queries in almost all cases, except for the sessions owned by the peer of the test session. Therefore, their schemes neither support the CK model nor the eCK model, as in both the models, the adversary is allowed to make the Ephemeral Key Reveal queries for all sessions except for test session and its matching session.

Huang et al. [12] first proposed an ID-based AKE protocol using pairing which is provably secure in the eCK model (that includes MFS) under the BDH assumption. The main non-trivial task in simulation of the eCK model is to consistently answer the Session Key Reveal queries, final hash oracle queries and Ephemeral Secret Reveal queries without knowing the static key i.e. the long term secret key. They used a technique called the TRAPDOOR Test to handle the above queries rather than using the Gap-CDH assumption.

An eCK secure ID-Based AKE protocol with MFS was proposed by A. Fujioka et al. [10] under the GBDH problem. Performance wise, it is almost the same as [12] except, in [12], 4 addition operations are involved and the static key consists of 2 group elements, whereas in [10], 2 addition operations are involved and the static key consists of a single group element. Later, A. Fujioka et al. in [9] extend this result using asymmetric pairing.

A provably eCK secure ID-based AKE protocol based on the same technique as Huang et al was constructed by Ni et al. [17]. However, they claimed that by some pre-computation (or off-line computation), the session key computation time can be reduced. But the total computation cost is very high as it requires six pairing computations.

1.3 Organization

This paper is organized as follows. We provide a brief background on Bilinear Pairing and eCK security model in Section 2. The proposed ID-based AKE protocols and their security are discussed in Section 3. Finally, we conclude the work in Section 4.

2 Preliminaries

Notation For a set X, $x \xleftarrow{R} X$ denotes that x is randomly picked from X according to the distribution R. Likewise, $x \xleftarrow{U} X$ indicates x is uniformly selected from X. *poly* stands for polynomial.

2.1 Bilinear Pairing

A prime order bilinear pairing groups are a tuple $(q, \mathbb{G}, \mathbb{G}_T, e)$, where q is prime, $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of prime order q and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is an efficiently computable map such that

1. (Bilinear) $\forall P, Q \in \mathbb{G}, a, b \in \mathbb{Z}_q, e(aP, bQ) = e(P, Q)^{ab}$
2. (Non-degenerate) $\exists P \in \mathbb{G}$ such that $e(P, P)$ has order q in $\mathbb{G}_T$

2.2 Diffie-Hellman Problems

Computational Diffie-Hellman (CDH) Problem: Given P, aP and $bP \in \mathbb{G}$, it is hard to compute $abP \in \mathbb{G}$.

Computational Bilinear Diffie-Hellman (BDH) Problem: Given $P, aP, bP, cP \in \mathbb{G}$, it is hard to compute $e(P, P)^{abc}$

Decisional Bilinear Diffie-Hellman (DBDH) Problem: It is hard to distinguish the *BDH* tuple $(P, aP, bP, cP, e(P, P)^{abc})$ from a tuple $(P, aP, bP, cP, e(P, P)^{r})$, where r is randomly chosen.

Gap Bilinear Diffie-Hellman (GBDH) Problem [15] : Given a tuple (P, aP, bP, cP) as input, it is hard to compute $e(P, P)^{abc}$ with the help of a DBDH oracle, $\mathcal{O}_{DBDH}$ which for given tuple $(P, xP, yP, zP, \kappa) \in \mathbb{G}^4 \times \mathbb{G}_T$, answers "true" if $\kappa = e(P, P)^{xyz}$ else "false".

The advantage of an adversary, $\mathscr{A}$ breaking this GBDH problem is defined as

$$Adv_{\mathscr{A}}^{\mathsf{GBDH}}(\lambda) := Pr[a, b, c \xleftarrow{U} \mathbb{Z}_q,\ \mathscr{A}^{\mathcal{O}_{DBDH}(.,.,.)}(P, aP, bP, cP) = BDH(P, aP, bP, cP)]$$

where the security parameter λ defines the size of the bilinear groups.

2.3 Security Model ([18])

In the Canetti-Krawczyk [2] security model (CK-model) for AKE, the adversary is allowed to reveal the session state information but not the test session and its matching session. In the extended-CK (eCK) [14] security model (defined in PKI-setting), the adversary is given full power in revealing both static and ephemeral keys without trivially breaking the session. The eCK model captures many security features that are not covered by any single model, viz, weak perfect forward security (wPFS), key-compromise impersonation (KCI) attack, leakage of ephemeral keys attack etc. However, Cas Cremers first showed in [6] that the CK model and the eCK model are formally and practically incomparable. They provide for each model some attacks on the protocols from the literature that are not captured by the other models.

Huang et al. [12] first, formalized the eCK model in the ID-Based setting, where it includes an additional attack, viz, master forward secrecy. Here, we separate out the MFS part from the eCK model due to Huang et al. [12] and we handle it separately, i.e., our eCK model is inspired by the original eCK model of [14].

Let $\mathscr{U} = \{U_i : \ i = 1, \cdots, n\}$ be the set of parties with each party U_i having an identity ID_i being a probabilistic polynomial-time (PPT) Turing machine. The protocol may run between any two of these parties. Each party may execute a polynomial number of protocol instances (sessions) in parallel with other parties. For each party U_i, there exists a public key Q_i that can be derived from its identity ID_i using hash function H_1. Let $Comm_i$ and $Comm_j$ be the outgoing messages, consist of ephemeral public key(s) and/or authentication tag from U_i and U_j respectively.

Let Π_{ij}^t be a completed session run between the parties U_i and U_j. Let *sid* stand for session identifier. It is defined as $sid := (Comm_i, Comm_j, ID_i, ID_j)$, where $Comm_i$ and $Comm_j$ are defined as earlier, ID_i is the owner of the session, ID_j is peer. A session Π_{ji}^ι is said to be a matching session of Π_{ij}^t if Π_{ji}^ι is completed and has *sid* of the form $(Comm_j, Comm_i, ID_j, ID_i)$.

The adversary $\mathscr{A}$ modelled here, is a PPT Turing machine which has full control on the communication network over which protocol messages can be altered, injected or eavesdropped at any time. The security of a protocol Π is defined as an adaptive game between the parties U_i and the adversary $\mathscr{A}$. This game executes in two phases. In the first phase to capture possible leakage of private information, the adversary is provided with the capability of asking the following additional oracle queries in any order.

EphemeralSecretReveal(Π_{ij}^t): The adversary $\mathscr{A}$ is given the ephemeral secret used in session Π_{ij}^t. This could be possible if the session-specific secret information is stored in insecure memory, or if the random number generator of the party be guessed.

SessionKeyReveal(Π_{ij}^t): $\mathscr{A}$ is given the session key for Π_{ij}^t, provided that the session holds a session key.

Long-termSecretReveal(U_i): $\mathscr{A}$ obtains the long term secret key of U_i.

EstablishParty(U_i): The adversary $\mathscr{A}$ can register ID_i on behalf of the party U_i. In this case, $\mathscr{A}$ obtains the long term secret key of U_i.

Send(Π_{ij}^t, m): The adversary's ability of controlling the communication network is modelled by the *Send* query. Here, the adversary sends a message m to party U_i in the t^{th} session Π_{ij}^t on behalf of party U_j and gets responses from U_i according to the protocol specification.

The adversary begins the second phase of the game by choosing a fresh session Π_{ij}^t and issuing a $Test(\Pi_{ij}^t)$ query, where the fresh session and test query are defined as follows:

Definition 2.1. (Fresh session) A session Π_{ij}^t executed by an honest party U_i with another party U_j is said to be fresh if none of the following conditions hold:

1. U_j is engaged in session Π_{ji}^ι matching to Π_{ij}^t and the adversary $\mathscr{A}$ reveals the session key of Π_{ij}^t or Π_{ji}^ι.
2. U_j is engaged in session Π_{ji}^ι matching to Π_{ij}^t and $\mathscr{A}$ issues either both the Long-term secret key of U_i and the ephemeral secret of Π_{ij}^t, or both the Long-term secret key of U_j and the ephemeral secret of Π_{ji}^ι.
3. No session matching to Π_{ij}^t exists and the adversary $\mathscr{A}$ reveals either the Long-term secret key of U_j or both the static keys of U_i and the ephemeral secret of Π_{ij}^t.

Definition 2.2. (Test Π_{ij}^t Query:) On the Test Query, a bit $b \in \{0,1\}$ is randomly chosen. The session key is given to the adversary $\mathscr{A}$, if $b = 0$, otherwise a uniformly chosen random value from the distribution of valid session keys is returned to $\mathscr{A}$. Only one query of this form is allowed for the adversary. Of course, after the $Test(\Pi_{ij}^t)$ query has been issued, the adversary can continue querying provided that the test session Π_{ij}^t is fresh. $\mathscr{A}$ outputs his guess b' in the test session. An adversary wins the game if the selected test session is fresh and if he guesses the challenge correctly i.e., $b' = b$. Thus the adversary's advantage in winning the game is defined as

$$Adv_{\mathscr{A}}^{\Pi}(\lambda) = |Pr[\mathscr{A}\ wins] - 1/2|$$

Definition 2.3. (eCK Security). An authenticated key exchange protocol is said to be secure (in the eCK model) if matching sessions compute the same session keys and for any PPT adversary $\mathscr{A}$ the advantage in winning the above game is negligible.

Definition 2.4. (Master Forward Secrecy). An authenticated key exchange protocol is said to be secure with MFS if the definition 2.3 still holds even after the adversary is allowed to learn the master secret key[1].

Definition 2.5. (Perfect Forward Secrecy). An authenticated key exchange protocol is said to be secure with PFS if the definition 2.3 still holds even when the adversary is allowed to learn the static keys of the owner and peer but after the completion of the test session.

Definition 2.6. (Master Perfect Forward Secrecy). An authenticated key exchange protocol is said to be secure with MPFS if the definition 2.3 still holds even when the adversary is allowed to learn the master secret key but after the completion of the test session.

3 Identity-Based Single Round Authenticated Key Exchange Protocols

We propose below eCK secure ID-based AKE protocols with different additional features, viz, Π_1 without MFS, Π_2 with PFS, Π_3 with MFS and Π_4, Π_5, Π_6 with Master PFS. Although, the protocols Π_4, Π_5, Π_6 achieve the same security, but they have different computational cost analysis as given in Table 1. The security of all these protocols rely on the GBDH problem. All the protocols presented here are based on the structure of Chow et al. [5].

3.1 eCK Secure ID-Based AKE Protocol without MFS (Π_1)

Here we present our basic ID-based eCK secure AKE protocol without MFS from the GBDH problem. The efficiency of the protocol is comparable to the existing protocols. A tabular representation of the protocol Π_1 is given in Table 2.

Setup(1^λ): Let $\mathbb{G}$ be a bilinear group of prime order q, and let P be a generator of $\mathbb{G}$. In addition, let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ denote the bilinear map. Let λ be the security parameter that will determine the size of the groups. Let $H_1 : \{0,1\}^\star \to \mathbb{G}$ and $H : \{0,1\}^\star \to \{0,1\}^\mu$, where $\mu = poly(\lambda)$, be hash functions. The PKG chooses $s \xleftarrow{U} \mathbb{Z}_q$ as MSK. Then, it declares the public parameters as $\mathsf{PP} := \{\mathbb{G}, \mathbb{G}_T, e, P, sP, H, H_1\}$
KeyGen(PP, MSK, ID_A): It first computes the public key of the party ID_A as $Q_A := H_1(ID_A)$. Then, it sets the long term secret key of the party ID_A as $\mathsf{SK}_A = sQ_A$.
KeyAgreement: The following is the description of a single round ID-based key exchange protocol between two parties with identities ID_A and ID_B.

[1] In definition 2.4, $\mathscr{A}$ is passive in the test session, otherwise $\mathscr{A}$ itself chooses an ephemeral key of the test session and trivially computes the session key

Table 2: Our eCK secure ID-Based AKE Protocol without MFS (Π_1)

ID_A	ID_B
$Q_A := H_1(ID_A), \mathsf{SK}_A := sQ_A$ Pre-Comp: $g_2^s := e(\mathsf{SK}_A, Q_B)$	$Q_B := H_1(ID_B), \mathsf{SK}_B := sQ_B$ Pre-Comp: $g_2^s := e(\mathsf{SK}_B, Q_A)$
$\eta_A \xleftarrow{U} \mathbb{Z}_q,\ \alpha_A := \eta_A Q_A$	$\eta_B \xleftarrow{U} \mathbb{Z}_q,\ \alpha_B := \eta_B Q_B$
$Comm_A := (\alpha_A) - - - - - - - - - - - - - - - - >$ $< - - - - - - - - - - - - - - - - Comm_B := (\alpha_B)$	
$g_2^{s\eta_B} := e(\mathsf{SK}_A, \alpha_B)$ $\kappa_{AB} := (g_2^s)^{\eta_A}.g_2^{s\eta_B},\ \sigma_{AB} := (g_2^{s\eta_B})^{\eta_A}$ $\mathsf{SN}_{AB} := H(\kappa_{AB},\ \sigma_{AB},\ P,\ sP,\ sid)$	$g_2^{s\eta_A} := e(\mathsf{SK}_B, \alpha_A)$ $\kappa_{BA} := g_2^{s\eta_A}.(g_2^s)^{\eta_B},\ \sigma_{BA} := (g_2^{s\eta_A})^{\eta_B}$ $\mathsf{SN}_{BA} := H(\kappa_{BA},\ \sigma_{BA},\ P,\ sP,\ sid)$
$sid := (Comm_A, Comm_B, ID_A, ID_B)$	

Pre-Computation : Let $g_2 := e(Q_A, Q_B)$. The parties ID_A and ID_B respectively compute $g_2^s := e(\mathsf{SK}_A, Q_B)$ and $g_2^s := e(\mathsf{SK}_B, Q_A)$. (This is independent of ephemeral key)

Post-Computation : The party ID_A picks an ephemeral key $\eta_A \xleftarrow{U} \mathbb{Z}_q$ and sends the ephemeral public key $\alpha_A := \eta_A Q_A$ to ID_B. Similarly, the party ID_B sends the ephemeral public key $\alpha_B := \eta_B Q_B$ to ID_A. Upon receiving the message $Comm_B = \alpha_B$ from ID_B, the party ID_A computes the pre-session key components as $g_2^{s\eta_A} := (g_2^s)^{\eta_A},\ g_2^{s\eta_B} := e(\mathsf{SK}_A, \alpha_B),\ \kappa_{AB} := g_2^{s\eta_A}.g_2^{s\eta_B} = g_2^{s(\eta_A+\eta_B)},\ \sigma_{AB} := (g_2^{s\eta_B})^{\eta_A} = g_2^{s\eta_A\eta_B}$. Finally, ID_A computes the session key as $\mathsf{SN}_{AB} := H(\kappa_{AB},\ \sigma_{AB},\ P,\ sP,\ sid)$, where the session identifier sid is given by $(Comm_A, Comm_B, ID_A, ID_B)$. Similarly, Upon receiving the message $Comm_A = \alpha_A$ from ID_A, the party ID_B computes the pre-session key components as $g_2^{s\eta_B} := (g_2^s)^{\eta_B},\ g_2^{s\eta_A} := e(\mathsf{SK}_B, \alpha_A),\ \kappa_{BA} := g_2^{s\eta_A}.g_2^{s\eta_B} = g_2^{s(\eta_A+\eta_B)},\ \sigma_{BA} := (g_2^{s\eta_A})^{\eta_B} = g_2^{s\eta_A\eta_B}$ and the session key is computed as $\mathsf{SN}_{BA} := H(\kappa_{BA},\ \sigma_{BA},\ P,\ sP,\ sid)$.

Theorem 3.1. *The proposed Protocol Π_1 is eCK secure, provided the* GBDH *assumption holds for $(\mathbb{G}, \mathbb{G}_T, e, q)$ and H, H_1 are treated as random oracles.*

Proof. Let $\mathscr{A}$ be an eCK PPT adversary against the given AKE protocol (Π_1). We establish a PPT algorithm $\mathscr{S}$, called Simulator, which takes an instance of GBDH problem from the challenger $\mathscr{C}$. Modeling H, H_1 as random oracles, $\mathscr{S}$ solves the GBDH problem by employing the advantage of an adversary $\mathscr{A}$ in breaking the AKE protocol. Since, the session key is computed as $\mathsf{SN} := H(\kappa, \sigma, P, sP, sid)$ and H is treated as random oracle, the adversary $\mathscr{A}$ has only two ways to distinguish the session key SN from a random string.

1. Forging Attack: At some point the adversary $\mathscr{A}$ queries the tuple $(\kappa, \sigma, P, sP, sid)$ on the oracle H.
2. Key-replication attack: The adversary $\mathscr{A}$ forces the establishment of another session with the same session key as the test session.

Since, each time ephemeral keys are chosen randomly, so, distinct AKE sessions must have distinct tuples on H. Therefore, any two different sessions never have the same session key unless the oracle H produces collisions. Hence, Key-replication attack is impossible if the oracle H produces no collisions.

The behavior of the simulation is based on the guess of the test session and the strategy that the adversary $\mathscr{A}$ adopts. Let λ denote the security parameters. Assume that the adversary can activate at most $\varepsilon := poly(\lambda)$ honest parties and at most $\delta := poly(\lambda)$ sessions in each party. $\mathscr{S}$ guesses that with

probability at least $1/\varepsilon^2$, the adversary $\mathscr{A}$ will choose one party with identity $ID_i = ID_A$ to be owner of the possible test session and other party $ID_j = ID_B$ as it's peer. With probability at least $1/\delta$, $\mathscr{S}$ always guesses that $\mathscr{A}$ will select Π_{ij}^t as the test session. During answering the test query, if the test session chosen by $\mathscr{A}$ does not match with the guess of $\mathscr{S}$, the simulator $\mathscr{S}$ aborts the game. Otherwise $\mathscr{S}$ chooses $r \xleftarrow{U} \{0,1\}^\mu$ and gives to the adversary $\mathscr{A}$.

To handle the oracles H and H_1, $\mathscr{S}$ maintains the lists H^{list} and H_1^{list} respectively. The entries in the lists H^{list} and H_1^{list} are of the form $(\kappa, \sigma, P, sP, sid)$ and $(ID_i, t_i, Q_i = H_1(ID_i))$ respectively, where $t_i \xleftarrow{U} \mathbb{Z}_q$ and $sid = (Comm_i, Comm_j, ID_i, ID_j)$. The simulator also maintains a list SN^{list} whose entries are of the form (sid, SN) and this list will be updated during the *Send* queries. Through out this simulation, we assume that ID_A is the initiator and ID_B is the responder, as the reverse case can be handled in a similar manner.

According to the definition of fresh session, the Forging Attack can be divided into the following complementary cases (using the proof strategy of [12]):

- Case 1: No honest party owns a matching session to the test session. According to the fresh session definition, $\mathscr{A}$ is not allowed to learn the static secret key SK_B of ID_B. So, two possible cases may arise.
  - Case 1.1: The adversary is not allowed to learn the ephemeral key of ID_A. In this case $\mathscr{A}$ may query for SK_A.
  - Case 1.2: The adversary is not permitted to learn the static key SK_A of ID_A. In this case $\mathscr{A}$ may query for the ephemeral key of ID_A.
- Case 2: There is a honest party who owns the matching session to the test session.
  - Case 2.1: The adversary is not allowed to learn both the ephemeral key of ID_A and the static key SK_B of ID_B. (This is easier as compared to Case 1.1, so it is left to the readers)
  - Case 2.2: The adversary is not allowed to learn both the static key SK_A of ID_A and the static key SK_B of ID_B. (This is easier as compared to Case 1.2, so it is left to the readers)
  - Case 2.3: The adversary is not permitted to learn both the static key SK_A of ID_A and the ephemeral key of ID_B. (This is Analogous to Case 2.1)
  - Case 2.4: The adversary is not permitted to learn both the ephemeral key of ID_A and the ephemeral key of ID_B.

Case 1.1: In this case, the simulator $\mathscr{S}$ receives an instance, Q, $U = aQ$, $V = bQ$, $W = sQ$ of GBDH problem for $(\mathbb{G}, \mathbb{G}_T, e, q)$ from the challenger $\mathscr{C}$. With the help of an adversary $\mathscr{A}$ against this AKE protocol, $\mathscr{S}$ solves (computes $e(Q,Q)^{abs}$) the GBDH problem. $\mathscr{S}$ sets implicitly $MSK = \{s\}$(s is unknown to $\mathscr{S}$). It chooses $p \xleftarrow{U} \mathbb{Z}_q$ and publishes $\mathsf{PP} := \{e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T, P = p\boxed{Q}, sP = p\boxed{W}, H, H_1\}$.

- $H_1(ID_i)$: As said earlier, $\mathscr{S}$ maintains a list H_1^{list} which is initially empty and whose entries are of the form (ID_i, t_i, Q_i). $\mathscr{S}$ searches ID_i in the list H_1^{list} to find a tuple (ID_i, t_i, Q_i) containing ID_i.
  - If it is already there, then $\mathscr{S}$ returns the stored value Q_i to $\mathscr{A}$.
  - Otherwise, if $ID_i = ID_A$, $\mathscr{S}$ responds $\boxed{Q}$ to the adversary $\mathscr{A}$ and inserts the new tuple $(A, null, Q_A = \boxed{Q})$ into the list H_1^{list}.
  - Otherwise, if $ID_i = ID_B$, $\mathscr{S}$ sends $\boxed{V}$ to the adversary $\mathscr{A}$ and inserts the new tuple $(B, null, Q_B = \boxed{V})$ into the list H_1^{list}.

- Otherwise, $\mathscr{S}$ picks a scalar $t_i \xleftarrow{U} \mathbb{Z}_q$ and sets $Q_i = H_1(ID_i) = t_i \boxed{Q}$ and then updates the list H_1^{list} by inserting (ID_i, t_i, Q_i).

- $H(\kappa, \sigma, P, sP, sid)$: The simulator $\mathscr{S}$ maintains a list H^{list} which is initially empty and whose entries are of the form $(\kappa, \sigma, P, sP, sid, \mathsf{SN})$. $\mathscr{S}$ can handle the usual queries on H (as he knows the static key $\mathsf{SK}_i (\neq \mathsf{SK}_B)$) except for the tuples of the form $(\kappa, \sigma, P, sP, sid)$, where $sid = (Comm_C, Comm_B, ID_B, ID_C)$, ID_C is ID_B's peer and may be a fictitious party fully controlled by $\mathscr{A}$. $\mathscr{S}$ searches the tuple $(\kappa, \sigma, P, sP, sid)$ in the list H^{list}.
  - If it is already there, then $\mathscr{S}$ returns the stored value SN to $\mathscr{A}$.
  - Otherwise, it checks the consistency of the pre-session key components κ and σ corresponding to sid. Note that $\mathscr{S}$ does not know the static key of ID_B, but still he can check the validity of κ and σ with the help of gap i.e., the decision oracle of GBDH as follows: let $Comm_C = \alpha_C = \eta_C Q_C$ and $Comm_B = \alpha_B = \eta_B Q_B$. It first, checks $\kappa / e(\mathsf{SK}_C, \alpha_B) \stackrel{?}{=} \sigma^{1/\eta_B}$ and then, checks $DBDH(Q_C, \alpha_B = \eta_B Q_B, \alpha_C = \eta_C Q_C, \mathsf{SK}_C = sQ_C, \sigma) \stackrel{?}{=} 1$
    - If both the relations hold, then it searches sid into the list SN^{list} to get a tuple (sid, SN).
      - If it is not in SN^{list}, then $\mathscr{S}$ chooses $\mathsf{SN} \xleftarrow{U} \{0,1\}^{\mu}$ and replies the value SN to $\mathscr{A}$ and updates the list H^{list} with the new entry $(\kappa, \sigma, P, sP, sid, \mathsf{SN})$.
      - If it is in the list SK^{list} as a tuple (sid, SN), then returns the stored value SN to the adversary $\mathscr{A}$ and inserts the new tuple $(\kappa, \sigma, P, sP, sid, \mathsf{SN})$ into the list H^{list}.
    - else, $\mathscr{S}$ chooses $\mathsf{SN} \xleftarrow{U} \{0,1\}^{\mu}$ and replies the value SN to $\mathscr{A}$ and updates the list H^{list} with the new entry $(\kappa, \sigma, P, sP, sid, \mathsf{SN})$.

- EstablishParty(ID_i): In this case, $\mathscr{S}$ registers the ID_i on behalf of $\mathscr{A}$. $\mathscr{S}$ searches the tuple of the form (ID_i, t_i, Q_i) in the list H_1^{list} and returns the long term secret key $\mathsf{SK}_i = t_i \boxed{W}$ to the adversary $\mathscr{A}$.

- Long-termSecretReveal(ID_i):
  - If $ID_i = ID_B$, $\mathscr{S}$ aborts.
  - Otherwise, if $ID_i = ID_A$, $\mathscr{S}$ returns $\mathsf{SK}_A := \boxed{W}$ to $\mathscr{A}$.
  - Otherwise, $\mathscr{S}$ searches ID_i in the list H_1^{list} to find the tuple (ID_i, t_i, Q_i). Then, $\mathscr{S}$ returns $\mathsf{SK}_i := t_i \boxed{W}$.

- EphemeralSecretReveal(Π_{ij}^t):
  - If Π_{ij}^t is test session, then $\mathscr{S}$ aborts.
  - Otherwise, it returns the ephemeral key to the adversary $\mathscr{A}$.

- Send($\Pi_{ij}^t, Comm_j$):
  - If Π_{ij}^t is test session, the $\mathscr{S}$ sets $Comm_i = \alpha_i := \boxed{U}$ and returns it to $\mathscr{A}$. (Note that in this case $ID_i = ID_A$ and $ID_j = ID_B$)
  - Otherwise, $\mathscr{S}$ computes α_i by the rule of protocol and sends it to $\mathscr{A}$. $\mathscr{S}$ searches the list H^{list} for a tuple $(\kappa, \sigma, P, sP, sid, \mathsf{SN})$ containing sid such that κ and σ are both valid pre-session key components with respect to sid (the validity can be checked by the decision oracle of GBDH problem).

  - If it is true, then it stores the new tuple (sid, SN) in SN^{list}.
  - else, $\mathscr{S}$ picks $\mathsf{SN} \xleftarrow{U} \{0,1\}^{\mu}$ and stores the new tuple (sid, SN) in SN^{list}.

- SessionKeyReveal(Π_{ij}^{t}):
  - If Π_{ij}^{t} is a test session, then $\mathscr{S}$ aborts.
  - Otherwise, $\mathscr{S}$ returns stored value SN in the list SN^{list}
- Test(Π_{ij}^{t}):
  - If Π_{ij}^{t} is a test session, then $\mathscr{S}$ aborts.
  - Otherwise, $\mathscr{S}$ chooses $r \xleftarrow{U} \{0,1\}^{\mu}$ and gives it to the adversary $\mathscr{A}$.

Let $g_2 := e(Q_A, Q_B)$. If the adversary $\mathscr{A}$ succeeds this test session, then $\mathscr{A}$ must have made a query to the oracle H by the tuple $(\kappa, \sigma, P, sP, sid)$, where $\kappa = g_2^{s(a+\eta_B)}$, $\sigma = g_2^{sa\eta_B}$, $sid = (Comm_A, Comm_B, ID_A, ID_B)$, $\alpha_B (= Comm_B) = \eta_B Q_B$. Now, $\mathscr{S}$ computes $g_2^{s\eta_B} := e(\mathsf{SK}_A, \alpha_B)$. Then, it computes the solution of the given GBDH as $BDH(Q, U = aQ, V = bQ, W = sQ) := \kappa / g_2^{s\eta_B} = g_2^{sa} = e(Q,Q)^{sab}$. So, the advantage $Adv_{\mathscr{S}}^{GBDH}(\lambda)$ of the simulator $\mathscr{S}$ in breaking GBDH problem is at least $\frac{\varsigma}{\varepsilon^2 \delta} Adv_{\mathscr{A}}^{\Pi_1}(\lambda)$, where ς is the probability of occurrence of this case.

Case 1.2: Similarly, the simulator is given an instance, Q, $U = aQ$, $V = bQ$, $W = sQ$ of GBDH problem for $(\mathbb{G}, \mathbb{G}_T, e, q)$. Using the advantage of an adversary $\mathscr{A}$ against this AKE protocol, $\mathscr{S}$ solves (computes $e(Q,Q)^{abs}$) the GBDH problem. $\mathscr{S}$ implicitly sets $\mathsf{MSK} := \{s\}$ (s is unknown to $\mathscr{S}$) and it picks $p \xleftarrow{U} \mathbb{Z}_q$. Then, it publishes $\mathsf{PP} := \{e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T, P := p\boxed{Q}, sP := p\boxed{W}, H, H_1\}$. Below, we discuss only those queries whose answers are different from Case 1.1.

- $H_1(ID_i)$: Similarly as above, $\mathscr{S}$ maintains a list H_1^{list} which is initially empty and whose entries are of the form (ID_i, t_i, Q_i). $\mathscr{S}$ searches ID_i in the list H_1^{list} to find a tuple (ID_i, t_i, Q_i) containing ID_i.
  - If ID_i is already there, then $\mathscr{S}$ gives the stored value Q_i to $\mathscr{A}$.
  - Otherwise, if $ID_i = ID_A$, $\mathscr{S}$ returns $\boxed{U}$ to the adversary $\mathscr{A}$ and inserts the new tuple $(A, null, Q_A := \boxed{U})$ into the list H_1^{list}.
  - Otherwise, if $ID_i = ID_B$, $\mathscr{S}$ replies $\boxed{V}$ to the adversary $\mathscr{A}$ and inserts the new tuple $(B, null, Q_B := \boxed{V})$ into the list H_1^{list}.
  - Otherwise, $\mathscr{S}$ picks a scalar $t_i \xleftarrow{U} \mathbb{Z}_q$ and sets $Q_i := H_1(ID_i) = t_i\boxed{Q}$ and then updates the list H_1^{list} by inserting (ID_i, t_i, Q_i).
- $H(\kappa, \sigma, P, sP, sid)$: Same as Case 1.1, except for the extra queries of the form $(\kappa, \sigma, P, sP, sid = (Comm_C, Comm_A, ID_A, ID_C))$, where ID_C is ID_A's peer and may be a fictitious party controlled by $\mathscr{A}$. $\mathscr{S}$ handles this by the same way as it is done for the queries of the form $(\kappa, \sigma, P, sP, sid = (Comm_C, Comm_B, ID_B, ID_C))$ in Case 1.1.
- Long-termSecretReveal(ID_i):
  - If $ID_i = ID_A$ or $ID_i = ID_B$, $\mathscr{S}$ aborts.
  - Otherwise, $\mathscr{S}$ searches ID_i in the list H_1^{list} to find the tuple (ID_i, t_i, Q_i). Then $\mathscr{S}$ returns $t_i\boxed{W}$.

- Send($\Pi^t_{ij}, Comm_j$): $\mathscr{S}$ computes α_i by the rule of protocol and sends it to $\mathscr{A}$. $\mathscr{S}$ picks $\mathsf{SN} \xleftarrow{U} \{0,1\}^\mu$ and stores the new tuple (sid, SN) in SN^{list}.

Let $g_2 := e(Q_A, Q_B)$. If the adversary $\mathscr{A}$ wins this game, then it must have made a query to the oracle H by the tuple $(\kappa, \sigma, sP, sid)$, where $\kappa = g_2^{s(\eta_A+\eta_B)}$, $\sigma = g_2^{s\eta_A\eta_B}$, $sid = (Comm_A, Comm_B, ID_A, ID_B)$, $\alpha_A = \eta_A Q_A, \alpha_B = \eta_B Q_B$. Now, $\mathscr{S}$ computes $g_2^{s\eta_B} := \sigma^{1/\eta_A}$ and then, calculates $g_2^{s\eta_A} := \kappa / g_2^{s\eta_B}$. Hence, it computes the solution of the given GBDH as $BDH(Q, U = aQ, V = bQ, W = sQ) := (g_2^{s\eta_A})^{1/\eta_A} = g_2^s = e(Q,Q)^{sab}$. So, the advantage $Adv^{GBDH}_{\mathscr{S}}(\lambda)$ of the simulator $\mathscr{S}$ in breaking GBDH problem is at least $\frac{\varsigma}{\varepsilon^2\delta} Adv^{\Pi_1}_{\mathscr{A}}(\lambda)$, where ς is the probability of occurrence of this case.

Case 2.4 (Forward Secrecy): As usual, $\mathscr{S}$ receives the parameters Q, $U = aQ$, $V = bQ$, $W = sQ$ of GBDH problem for $(\mathbb{G}, \mathbb{G}_T, e, q)$ from the challenger $\mathscr{C}$. $\mathscr{S}$ implicitly sets $\mathsf{MSK} := \{s\}$ (s is unknown to $\mathscr{S}$) and it picks $p \xleftarrow{U} \mathbb{Z}_q$. Then, it publishes $\mathsf{PP} := \{e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T, P := p\boxed{Q}, sP := p\boxed{W}, H, H_1\}$. For each party ID_i, $\mathscr{S}$ chooses $t_i \xleftarrow{U} \mathbb{Z}_q$. It sets the public key as $Q_i := t_i\boxed{Q}$ and static key as $\mathsf{SK}_i := sQ_i$. It can easily answer all the queries even including the long term secret keys of all the parties. $\mathscr{S}$ sets $\alpha_A := t_A\boxed{U} = aQ_A$, $\alpha_B := t_B\boxed{V} = bQ_B$. Let $g_2 := e(Q_A, Q_B)$. Similarly, on success of this game, $\mathscr{A}$ must have queried the oracle H by a tuple $(\kappa, \sigma, P, sP, sid)$, where $\kappa = g_2^{s(a+b)}$, $\sigma = g_2^{sab}$, $sid = (Comm_A, Comm_B, ID_A, ID_B)$. Then, it computes the solution of the given GBDH as $BDH(Q, U = aQ, V = bQ, W = sQ) := \sigma^{1/t_A t_B} = e(Q,Q)^{sab}$. So, the advantage $Adv^{GBDH}_{\mathscr{S}}(\lambda)$ of the simulator $\mathscr{S}$ in breaking GBDH problem is at least $\frac{\varsigma}{\varepsilon^2\delta} Adv^{\Pi_1}_{\mathscr{A}}(\lambda)$, where ς is the probability of occurrence of this case. □

3.2 eCK Secure ID-Based AKE Protocol with PFS (Π_2)

In this section, an ID-based eCK secure AKE protocol with PFS is presented. The extra feature, PFS, allows the adversary to learn the static key SK_A and SK_B with similar kind of security guarantee but after the completion of the test session. Here, the adversary is active and is allowed to choose the message of its own choice. The computational efficiency of this protocol is almost the same as existing efficient eCK secure AKE protocols. The protocol is presented in Table 3.

Setup(1^λ): This is almost the same as protocol Π_1, except there is an additional hash $H_2 : \{0,1\}^\star \to \{0,1\}^\mu$ in PP.

KeyAgreement: The following is the description of a single round ID-based key exchange protocol between two parties with identities ID_A and ID_B.

Pre-Computation : Same as protocol Π_1.

Post-Computation : The party ID_A picks an ephemeral key $\eta_A \xleftarrow{U} \mathbb{Z}_q$ and sets the ephemeral public key as $\alpha_A := \eta_A Q_A$. It also prepares a peer authentication tag γ_A as $\gamma_A := H_2((g_2^s)^{\eta_A}, g_2^s, ID_A, ID_B)$ and then sends $Comm_A := (\alpha_A, \gamma_A)$ to ID_B. Similarly, ID_B sends $Comm_B := (\alpha_B := \eta_B Q_B, \gamma_B := H_2((g_2^s)^{\eta_B}, g_2^s, ID_B, ID_A))$ to ID_A. Upon receiving the message $Comm_B = (\alpha_B, \gamma_B)$ from ID_B, the party ID_A computes $g_2^{s\eta_B} := e(\mathsf{SK}_A, \alpha_B)$ and checks $\gamma_B \stackrel{?}{=} H_2(g_2^{s\eta_B}, g_2^s, ID_B, ID_A)$. If the relation does not hold then, it aborts; otherwise, it computes the pre-session key components[2] as $\kappa_{AB} := g_2^{s\eta_A} . g_2^{s\eta_B} = g_2^{s(\eta_A+\eta_B)}$, $\sigma_{AB} := (g_2^{s\eta_B})^{\eta_A} = g_2^{s\eta_A\eta_B}$. Finally, ID_A computes the session key as $\mathsf{SN}_{AB} := H(\kappa_{AB}, \sigma_{AB}, P, sP, sid)$, where sid is given by $(Comm_A, Comm_B, ID_A, ID_B)$. Similarly, upon receiving the message $Comm_A = (\alpha_A, \gamma_A)$ from ID_A, the party ID_B computes $g_2^{s\eta_A} := e(\mathsf{SK}_B, \alpha_A)$

[2]The value $g_2^{s\eta_A}$ is computed by ID_A during the computation of peer authentication tag γ_A

and checks $\gamma_A \stackrel{?}{=} H_2(g_2^{s\eta_A}, g_2^s, ID_A, ID_B)$. If it is false, then it aborts; otherwise, it computes the pre-session key components as $\kappa_{BA} := g_2^{s\eta_A}.g_2^{s\eta_B} = g_2^{s(\eta_A+\eta_B)}$, $\sigma_{BA} := (g_2^{s\eta_A})^{\eta_B} = g_2^{s\eta_A\eta_B}$. The session key is computed as $\mathsf{SN}_{BA} := H(\kappa_{BA}, \sigma_{BA}, P, sP, sid)$.

Table 3: Our eCK secure ID-Based AKE Protocol with PFS (Π_2)

ID_A	ID_B
$Q_A := H_1(ID_A), \mathsf{SK}_A := sQ_A$ Pre-Comp: $g_2^s := e(\mathsf{SK}_A, Q_B)$	$Q_B := H_1(ID_B), \mathsf{SK}_B := sQ_B$ Pre-Comp: $g_2^s := e(\mathsf{SK}_B, Q_A)$
$\eta_A \xleftarrow{U} \mathbb{Z}_q,\ \alpha_A := \eta_A Q_A$ $\gamma_A := H_2((g_2^s)^{\eta_A}, g_2^s, ID_A, ID_B)$	$\eta_B \xleftarrow{U} \mathbb{Z}_q,\ \alpha_B := \eta_B Q_B$ $\gamma_B := H_2((g_2^s)^{\eta_B}, g_2^s, ID_B, ID_A)$
$Comm_A := (\alpha_A, \gamma_A) - - - - - - - - - - - - - - - - >$ $< - - - - - - - - - - - - - - - - Comm_B := (\alpha_B, \gamma_B)$	
$g_2^{s\eta_B} := e(\mathsf{SK}_A, \alpha_B)$ ID_A checks $\gamma_B \stackrel{?}{=} H_2(g_2^{s\eta_B}, g_2^s, ID_B, ID_A)$ If it is false, then aborts $\kappa_{AB} := g_2^{s\eta_A}.g_2^{s\eta_B},\ \sigma_{AB} := (g_2^{s\eta_B})^{\eta_A}$ $\mathsf{SN}_{AB} := H(\kappa_{AB}, \sigma_{AB}, P, sP, sid)$	$g_2^{s\eta_A} := e(\mathsf{SK}_B, \alpha_A)$ ID_B checks $\gamma_A \stackrel{?}{=} H_2(g_2^{s\eta_A}, g_2^s, ID_A, ID_B)$ If it is false, then aborts $\kappa_{BA} := g_2^{s\eta_A}.g_2^{s\eta_B},\ \sigma_{BA} := (g_2^{s\eta_A})^{\eta_B}$ $\mathsf{SN}_{BA} := H(\kappa_{BA}, \sigma_{BA}, P, sP, sid)$
$sid := (Comm_A, Comm_B, ID_A, ID_B)$	

Theorem 3.2. *The proposed Protocol Π_2 is eCK secure with PFS, if the* GBDH *assumption holds for $(\mathbb{G}, \mathbb{G}_T, e, q)$ and H, H_1, H_2 are random oracles*

Proof. The proof of eCK security is almost the same as that of protocol Π_1 (theorem 3.1), except for an extra component involved via the new hash function $H_2 : \{0,1\}^\star \to \{0,1\}^\mu$ that slightly modifies the proof. In the case analysis, we only briefly discuss those queries and issues whose answers are different from theorem 3.1.

Case 1.1: As usual, the simulator $\mathscr{S}$ receives an instance, $Q,\ U = aQ,\ V = bQ,\ W = sQ$ of GBDH problem for $(\mathbb{G}, \mathbb{G}_T, e, q)$ from the challenger $\mathscr{C}$. Using the advantage of an adversary $\mathscr{A}$ against this AKE protocol, $\mathscr{S}$ solves (computes $e(Q,Q)^{abs}$) the GBDH problem. $\mathscr{S}$ sets implicitly $MSK = \{s\}$(s is unknown to $\mathscr{S}$). It chooses $p \xleftarrow{U} \mathbb{Z}_q$ and publishes $\mathsf{PP} := \{e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T, P = p\boxed{Q}, sP = p\boxed{W}, H, H_1, H_2\}$.

- $H_1(ID_i)$: Same as Case 1.1 in theorem 3.1.
- $H_2(g_2^{s\eta_i}, g_2^s, ID_i, ID_j)$: $\mathscr{S}$ maintains a list H_2^{list} which is initially empty and whose entries are of the form $(g_2^{s\eta_i}, g_2^s, ID_i, ID_j, \gamma_i)$. It searches $(g_2^{s\eta_i}, g_2^s, ID_i, ID_j)$ in the list H_2^{list} to find the tuple $(g_2^{s\eta_i}, g_2^s, ID_i, ID_j, \gamma_i)$.
  - If it is already there, then $\mathscr{S}$ returns the stored value γ_i to $\mathscr{A}$.
  - Otherwise, $\mathscr{S}$ picks $\gamma_i \xleftarrow{U} \{0,1\}^\mu$ and returns it to $\mathscr{A}$ and inserts the new tuple $(g_2^{s\eta_i}, g_2^s, ID_i, ID_j, \gamma_i)$ into the list H_2^{list}.

- $H(\kappa,\sigma,P,sP,sid)$: $\mathscr{S}$ first checks the consistency of the messages in sid via their authentication[3] tags. If the messages are valid then, it responds in similar manner as in Case 1.1 in theorem 3.1 otherwise, $\mathscr{S}$ chooses $\mathsf{SN} \xleftarrow{U} \{0,1\}^{\mu}$ and replies the value SN to $\mathscr{A}$ and updates the list H^{list} with the new entry $(\kappa,\sigma,P,sP,sid,\mathsf{SN})$.
- EstablishParty(ID_i): Same as Case 1.1 in theorem 3.1.
- Long-termSecretReveal(ID_i): Same as Case 1.1 in theorem 3.1.
- EphemeralSecretReveal(Π^t_{ij}): Same as Case 1.1 in theorem 3.1.
- Send(Π^t_{ij},$Comm_j$): Let $g_2 := e(Q_i,Q_j)$. The simulator $\mathscr{S}$ first checks the consistency of the message $Comm_j = (\alpha_j,\gamma_j)$ using either static key SK_i or the decision oracle of GBDH problem (in case of without having SK_i). If it is not consistent, then $\mathscr{S}$ aborts.
  - If Π^t_{ij} is test session, the simulator $\mathscr{S}$ sets $\alpha_i := \boxed{U}$ and $\gamma_i := H_2(g_2^{sa},g_2^s,ID_i,ID_j) \xleftarrow{U} \{0,1\}^{\mu}$ and returns the message $Comm_i := (\alpha_i,\gamma_i)$ to $\mathscr{A}$. $\mathscr{S}$ updates the list H_2^{list} by the tuple $(null,g_2^s,ID_i,ID_j,\gamma_i)$. (Note that in this case $ID_i = ID_A$ and $ID_j = ID_B$). Also, the point to be noted that $\mathscr{S}$ could not compute $g_2^{sa} := e(Q_A,Q_B)^{sa}$ as it neither knows SK_B nor the ephemeral key a. If later, $\mathscr{A}$ queries with the tuple (g_2^*,g_2^s,ID_A,ID_B) to the oracle H_2, then it can recognize whether $g_2^* \stackrel{?}{=} g_2^{sa}$ via the decision oracle test $DBDH(Q,U,V,W,g_2^*) \stackrel{?}{=} g_2^{sa}$. If it is true, $\mathscr{S}$ returns the stored value γ_A otherwise, return $r \xleftarrow{U} \{0,1\}^{\mu}$ to $\mathscr{A}$ and updates the list H_2 with the new entry $(g_2^*,g_2^s,ID_A,ID_B,r)$.
  - Otherwise, $\mathscr{S}$ computes $Comm_i := (\alpha_i := \eta_i Q_i, \gamma_i := H_2(g_2^{s\eta_i},g_2^s,ID_i,ID_j) \xleftarrow{U} \{0,1\}^{\mu})$ by the rule of protocol, updates the H_2^{list} by the tuple $(g_2^{s\eta_i},g_2^s,ID_i,ID_j,\gamma_i)$ and sends it to $\mathscr{A}$. $\mathscr{S}$ picks $\mathsf{SN} \xleftarrow{U} \{0,1\}^{\mu}$ and stores the new tuple (sid,SN) in SN^{list}.
- SessionKeyReveal(Π^t_{ij}): Same as Case 1.1 in theorem 3.1.
- Test(Π^t_{ij}): Same as Case 1.1 in theorem 3.1.

The rest of Case 1.1 follows from that of Case 1.1 of theorem 3.1.

Case 1.2: As earlier, the simulator is given an instance, $Q,\ U = aQ,\ V = bQ,\ W = sQ$ of GBDH problem for $(\mathbb{G},\mathbb{G}_T,e,q)$. With the help of an adversary $\mathscr{A}$ against this AKE protocol, $\mathscr{S}$ solves (computes $e(Q,Q)^{abs}$) the GBDH problem. $\mathscr{S}$ implicitly sets $\mathsf{MSK} := \{s\}$ (s is unknown to $\mathscr{S}$) and it picks $p \xleftarrow{U} \mathbb{Z}_q$. Then, it publishes $\mathsf{PP} := \{e : \mathbb{G}\times\mathbb{G} \to \mathbb{G}_T, P := p\boxed{Q}, sP := p\boxed{W}, H, H_1, H_2\}$. Below, we discuss only those queries whose answers are different from Case 1.1.

- $H_1(ID_i)$: Same as Case 1.2 of theorem 3.1.
- $H(\kappa,\sigma,P,sP,sid)$: Same as Case 1.1, except for the extra queries of the form $(\kappa,\sigma,P,sP,sid = (Comm_C,Comm_A,ID_A,ID_C))$, where ID_C is ID_A's peer and may be a fictitious party controlled by $\mathscr{A}$. $\mathscr{S}$ handles this by the same way as it is done for the queries of the form $(\kappa,\sigma,P,sP,sid = (Comm_C,Comm_B,ID_B,ID_C))$ in Case 1.1.
- Long-termSecretReveal(ID_i):
  - If $ID_i = ID_A$ or $ID_i = ID_B$, $\mathscr{S}$ aborts.

[3]This can be checked either by the key(s) SK_i (and/or SK_j) or the decision oracle of GBDH problem

– Otherwise, $\mathscr{S}$ searches ID_i in the list H_1^{list} to find the tuple (ID_i, t_i, Q_i). Then $\mathscr{S}$ returns $t_i\boxed{W}$.

- Send($\Pi_{ij}^t, Comm_j$): $\mathscr{S}$ first checks the consistency of the message $Comm_j$. If it is not, then $\mathscr{S}$ aborts, otherwise, $\mathscr{S}$ computes $Comm_i := (\alpha_i := \eta_i Q_i, \gamma_i := H_2(g_2^{s\eta_i}, g_2^s, ID_i, ID_j) \xleftarrow{U} \{0,1\}^\mu)$ by the rule of protocol, updates the H_2^{list} by the tuple $(g_2^{s\eta_i}, g_2^s, ID_i, ID_j, \gamma_i)$ and sends it to $\mathscr{A}$. $\mathscr{S}$ picks $\mathsf{SN} \xleftarrow{U} \{0,1\}^\mu$ and stores the new tuple (sid, SN) in SN^{list}.

The rest of Case 1.2 follows from that of Case 1.2 of theorem 3.1.

Case 2.4 (Forward Secrecy): This is almost the same as Case 2.4 of theorem 3.1 except, here will be the peer authentication tags γ_A and γ_B respectively in $Comm_A$ and $Comm_B$.

Perfect Forward Secrecy: This is almost the same as Case 2.4 except, the adversary $\mathscr{A}$ is involved actively to choose the message of its own choice and is allowed to learn the static keys SK_A and SK_B respectively of ID_A and ID_B but after the completion of test session. As usual, the simulator $\mathscr{S}$ is given an instance, $Q,\ U = aQ,\ V = bQ,\ W = sQ$ of GBDH problem for $(\mathbb{G}, \mathbb{G}_T, e, q)$. With the help of adversary $\mathscr{A}$ against this AKE protocol, $\mathscr{S}$ solves (computes $e(Q,Q)^{abs}$) the *BDH* problem. $\mathscr{S}$ implicitly sets $\mathsf{MSK} := \{s\}$ (s is unknown to $\mathscr{S}$) and it picks $p \xleftarrow{U} \mathbb{Z}_q$. Then, it publishes $\mathsf{PP} := \{e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T, P := p\boxed{Q}, sP := p\boxed{W}, H, H_1, H_2\}$. For each party ID_i, $\mathscr{S}$ chooses $t_i \xleftarrow{U} \mathbb{Z}_q$. It sets the public key as $Q_i := t_i\boxed{Q}$ and static key as $\mathsf{SK}_i := sQ_i$. $\mathscr{S}$ can easily answer all the queries even including the static keys of all the parties but the keys SK_A and SK_B are given to $\mathscr{A}$ after the completion of test session.

Let $sid = (Comm_A, Comm_B, ID_A, ID_B)$ be the session chosen by $\mathscr{A}$ during the test phase. We claim that both the messages $Comm_A$ and $Comm_B$ could not be created by the adversary $\mathscr{A}$ itself as he is unaware of SK_A and SK_B. In fact, let $\alpha_B = \eta_B Q_B, \gamma_B = H_2(g_2^{s\eta_B}, g_2^s, ID_B, ID_A)$. Here, $\mathscr{A}$ essentially could not compute $g_2^s := e(Q_A, Q_B)^s = e(Q,Q)^{st_At_B}$ (it gives the solution of BDH instance, Q, t_AQ, t_BQ, sQ, which would contradict the hypothesis).

Therefore, both the messages could be computed by the simulator $\mathscr{S}$, itself. So, $\mathscr{S}$ sets $\alpha_A := t_A\boxed{U} = aQ_A, \gamma_A := H_2(g_2^{sa}, g_2^s, ID_A, ID_B) \xleftarrow{U} \{0,1\}^\mu, \alpha_B := t_B\boxed{V} = bQ_B, \gamma_B := H_2(g_2^{sb}, g_2^s, ID_B, ID_A) \xleftarrow{U} \{0,1\}^\mu$ during the send query by $\mathscr{A}$. If $\mathscr{A}$ succeeds this test session, then, it must have queried the oracle H by a tuple $(\kappa, \sigma, P, sP, sid)$, where $\kappa = g_2^{s(a+b)}$, $\sigma = g_2^{sab}$, $sid = (Comm_A, Comm_B, ID_A, ID_B)$. Then, it computes the solution of the given GBDH as $BDH(Q, U = aQ, V = bQ, W = sQ) := \sigma^{1/t_At_B} = e(Q,Q)^{sab}$, a contradiction. □

3.3 eCK Secure ID-Based AKE Protocol with MFS (Π_3)

An ID-based eCK secure AKE protocol with MFS from GBDH problem is described here. Note that we separate out the MFS part from the definition of eCK security. This protocol achieves the same computational cost in $\mathbb{G}$ as compared to the existing most efficient eCK secure AKE protocols in random oracle model.

Setup(1^λ): Same as protocol Π_1

KeyAgreement: The following is the description of a single round ID-based key exchange protocol between two parties with identities ID_A and ID_B.

Pre-Computation : Same as protocol Π_1.

Post-Computation : Almost the same as protocol Π_1, except there will be an additional component $\beta_A := \eta_A P$ (resp. $\beta_B := \eta_B P$) in $Comm_A$ (resp. $Comm_B$). The session key is computed as $\mathsf{SN}_{AB} := H(\kappa_{AB},\ \sigma_{AB},\ \tau_{AB},\ sP,\ sid)$, where $\tau_{AB} := \eta_A \beta_B$. Similarly, SN_{BA} is computed. For details, see the Table 4.

Table 4: Our eCK secure ID-Based AKE Protocol with MFS (Π_3)

ID_A	ID_B
$Q_A := H_1(ID_A), \mathsf{SK}_A := sQ_A$ Pre-Comp: $g_2^s := e(\mathsf{SK}_A, Q_B)$	$Q_B := H_1(ID_B), \mathsf{SK}_B := sQ_B$ Pre-Comp: $g_2^s := e(\mathsf{SK}_B, Q_A)$
$\eta_A \xleftarrow{U} \mathbb{Z}_q,\ \alpha_A := \eta_A Q_A,\ \beta := \eta_A P$	$\eta_B \xleftarrow{U} \mathbb{Z}_q,\ \alpha_B := \eta_B Q_B,\ \beta_B := \eta_B P$
$Comm_A := (\alpha_A, \beta_A) - - - - - - - - - - - - - - - - >$ $< - - - - - - - - - - - - - - - - Comm_B := (\alpha_B, \beta_B)$	
$g_2^{s\eta_B} := e(\mathsf{SK}_A, \alpha_B)$ $\kappa_{AB} := (g_2^s)^{\eta_A}.g_2^{s\eta_B},\ \sigma_{AB} := (g_2^{s\eta_B})^{\eta_A},\ \tau_{AB} := \eta_A\beta_B$ $\mathsf{SN}_{AB} := H(\kappa_{AB},\ \sigma_{AB},\ \tau_{AB},\ P,\ sP,\ sid)$	$g_2^{s\eta_A} := e(\mathsf{SK}_B, \alpha_A)$ $\kappa_{BA} := g_2^{s\eta_A}.(g_2^s)^{\eta_B},\ \sigma_{BA} := (g_2^{s\eta_A})^{\eta_B},\ \tau_{BA} := \eta_B\beta_A$ $\mathsf{SN}_{BA} := H(\kappa_{BA},\ \sigma_{BA},\ \tau_{BA},\ P,\ sP,\ sid)$
$sid := (Comm_A, Comm_B, ID_A, ID_B)$	

Theorem 3.3. *If the* GBDH *assumption holds for* $(\mathbb{G}, \mathbb{G}_T, e, q)$*, the* CDH *assumption holds for* $(\mathbb{G}, q)$ *and* H, H_1 *are random oracles, then the proposed Protocol* Π_3 *is eCK secure with MFS.*

Proof. The proof technique is almost the same as that of protocol Π_1 (Theorem 3.1), except that there will be additional message component and pre-session key component that can be handled in similar manner as in Theorem 3.1. Another exception is Case 2.4 which handles here the master forward secrecy from the CDH problem. Therefore, in the following, we only discuss the analysis of Case 2.4.

Case 2.4 (Master Forward Secrecy): The simulator $\mathscr{S}$ is given an instance, $P, U := aP, V := bP$ of the CDH problem for $(\mathbb{G}, q)$. With the help of an adversary $\mathscr{A}$ against this AKE protocol, $\mathscr{S}$ solves (computes abP) the CDH problem. $\mathscr{S}$ picks a scalar $s \xleftarrow{U} \{0,1\}$ as MSK and it publishes $\mathsf{PP} := \{e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T, \boxed{P}, s\boxed{P}, H, H_1\}$. For each party ID_i, $\mathscr{S}$ chooses $t_i \xleftarrow{U} \mathbb{Z}_q$. It sets the public key as $Q_i := t_i\boxed{P}$ and static key as $\mathsf{SK}_i := sQ_i$. $\mathscr{S}$ can easily answer all the queries even including the static keys of all the parties as well as MSK. $\mathscr{S}$ sets $\alpha_A := t_A\boxed{U} = aQ_A,\ \beta_A := \boxed{U},\ \alpha_B := t_B\boxed{V} = bQ_B,\ \beta_B := \boxed{V}$ during the send query by $\mathscr{A}$. If $\mathscr{A}$ succeeds this test session, then, it must have queried the oracle H by a tuple $(\kappa, \sigma, \tau, P, sP, sid)$, where $\kappa = g_2^{s(a+b)}$, $\sigma = g_2^{sab}$, $\tau = abP$, $sid = (Comm_A, Comm_B, ID_A, ID_B)$. Therefore, it computes the solution of the given CDH as $CDH(P, U = aP, V = bP) := \tau = abP$. □

3.4 eCK Secure ID-Based AKE Protocol with MPFS (Π_4)

We present here an ID-based eCK secure AKE protocol with Master PFS. The Master PFS guarantees the security of past sessions of the AKE protocol, even after the exposure of MSK to the adversary, i.e., the adversary may know the master secret key MSK but after the completion of test session. In this case, the adversary is forbidden to query the ephemeral keys of either side of the test session. This protocol is given in Table 5.

Setup(1^λ): Same as protocol Π_2.
KeyAgreement: The following is the description of a single round ID-based key exchange protocol between two parties with identities ID_A and ID_B.

Table 5: Our eCK secure ID-Based AKE Protocol with MPFS (Π_4)

ID_A	ID_B
$Q_A := H_1(ID_A), \mathsf{SK}_A := sQ_A$	$Q_B := H_1(ID_B), \mathsf{SK}_B := sQ_B$
Pre-Comp: $g_2^s := e(\mathsf{SK}_A, Q_B),\ f_A^s := e(Q_A, sP)$	Pre-Comp: $g_2^s := e(\mathsf{SK}_B, Q_A),\ f_B^s := e(Q_B, sP)$
$\eta_A \xleftarrow{U} \mathbb{Z}_q,\ \alpha_A := \eta_A Q_A,\ \beta_A := \eta_A P$	$\eta_B \xleftarrow{U} \mathbb{Z}_q,\ \alpha_B := \eta_B Q_B,\ \beta_B := \eta_B P$
$\gamma_A := H_2((g_2^s)^{\eta_A}, g_2^s, (f_A^s)^{\eta_A}, ID_A, ID_B)$	$\gamma_B := H_2((g_2^s)^{\eta_B}, g_2^s, (f_B^s)^{\eta_B}, ID_B, ID_A)$
$Comm_A := (\alpha_A, \beta_A, \gamma_A) - - - - - - - - - - - - - - - - >$	
$< - - - - - - - - - - - - - - - - Comm_B := (\alpha_B, \beta_B, \gamma_B)$	
$g_2^{s\eta_B} := e(\mathsf{SK}_A, \alpha_B),\ f_B^{s\eta_B} := e(\alpha_B, sP)$	$g_2^{s\eta_A} := e(\mathsf{SK}_B, \alpha_A),\ f_A^{s\eta_A} := e(\alpha_A, sP)$
ID_A checks two relations below	ID_B checks two relations below
$DBDH(P, sP, \beta_B = \eta_B P, Q_B, f_B^{s\eta_B}) \stackrel{?}{=} 1$	$DBDH(P, sP, \beta_A = \eta_A P, Q_A, f_A^{s\eta_A}) \stackrel{?}{=} 1$
$\gamma_B \stackrel{?}{=} H_2(g_2^{s\eta_B}, g_2^s, f_B^{s\eta_B}, ID_B, ID_A)$	$\gamma_A \stackrel{?}{=} H_2(g_2^{s\eta_A}, g_2^s, f_A^{s\eta_A}, ID_A, ID_B)$
If at least one of them is false, then aborts	If at least one of them is false, then aborts
$\kappa_{AB} := g_2^{s\eta_A}.g_2^{s\eta_B},\ \tau_{AB} := \eta_A \beta_B$	$\kappa_{BA} := g_2^{s\eta_A}.g_2^{s\eta_B},\ \tau_{BA} := \eta_B \beta_A$
$\mathsf{SN}_{AB} := H(\kappa_{AB},\ \tau_{AB},\ P,\ sP,\ sid)$	$\mathsf{SN}_{BA} := H(\kappa_{BA},\ \tau_{BA},\ P,\ sP,\ sid)$
$sid := (Comm_A, Comm_B, ID_A, ID_B)$	

Pre-Computation : Let $g_2 := e(Q_A, Q_B)$. Now the parties ID_A and ID_B respectively compute $g_2^s := e(\mathsf{SK}_A, Q_B), f_A^s := e(Q_A, sP)$ and $g_2^s := e(\mathsf{SK}_B, Q_A),\ f_B^s := e(Q_B, sP)$. (This is independent of ephemeral key)

Post-Computation : The party ID_A picks an ephemeral key $\eta_A \xleftarrow{U} \mathbb{Z}_q$ and sets the ephemeral public keys as $\alpha_A := \eta_A Q_A,\ \beta_A := \eta_A P$. It also prepares an authentication tag $\gamma_A := H_2((g_2^s)^{\eta_A}, g_2^s, (f_A^s)^{\eta_A}, ID_A, ID_B)$ and then sends $Comm_A := (\alpha_A, \beta_A, \gamma_A)$ to ID_B. Similarly, the party ID_B sends the message $Comm_B := (\alpha_B := \eta_B Q_B, \beta_B := \eta_B P, \gamma_B := H_2((g_2^s)^{\eta_B}, g_2^s, (f_B^s)^{\eta_B}, ID_B, ID_A))$ to ID_A. Upon receiving $Comm_B = (\alpha_B, \beta_B, \gamma_B)$ from ID_B, the party ID_A computes $g_2^{s\eta_B} := e(\mathsf{SK}_A, \alpha_B),\ f_B^{s\eta_B} := e(sP, \alpha_B)$ and checks the following two relations:

$$DBDH(P, sP, \eta_B P, Q_B, f_B^{s\eta_B}) \stackrel{?}{=} 1 \text{ and } \gamma_B \stackrel{?}{=} H_2(g_2^{s\eta_B}, g_2^s, f_B^{s\eta_B}, ID_B, ID_A)$$

If at least one of them does not hold, then, it aborts; otherwise, it computes the pre-session key components as $\kappa_{AB} := g_2^{s\eta_A}.g_2^{s\eta_B} = g_2^{s(\eta_A+\eta_B)},\ \tau_{AB} := \eta_A \beta_B = \eta_A \eta_B P$. Finally, ID_A computes the session key as $\mathsf{SN}_{AB} := H(\kappa_{AB},\ \tau_{AB},\ P,\ sP,\ sid)$, where *sid* has its usual meaning. Similarly, Upon receiving the message $Comm_A = (\alpha_A, \beta_A, \gamma_A)$ from ID_A, the party ID_B can compute the session key as $\mathsf{SN}_{BA} := H(\kappa_{BA},\ \tau_{BA},\ P,\ sP,\ sid)$.

Theorem 3.4. *If the* GBDH *assumption holds for* $(\mathbb{G}, \mathbb{G}_T, e, q)$*, the* CDH *assumption holds for* $(\mathbb{G}, q)$ *and* H, H_1, H_2 *are random oracles, then the proposed Protocol* Π_4 *is eCK secure with Master FPS.*

Proof. eCK security proof is similar to those of Theorems 3.1, 3.2 and 3.3, except, the Case 1.2 where the component σ is no more involved, is handled slightly differently.

Case 1.2: $\mathscr{S}$ is given an instance, $Q, U := aQ, V := bQ, W := sQ$ of the GBDH problem. It sets $Q_A := \boxed{U}, Q_B := \boxed{V}$. For other party with identity ID_i, it chooses $t_i \xleftarrow{U} \mathbb{Z}_q$ and sets $Q_i := t_i \boxed{Q}$ and $\mathsf{SK}_i := t_i \boxed{W}$. All the usual queries can be handled as in earlier case. During test phase, let

$sid = (Comm_A, Comm_B, ID_A, ID_B)$ be chosen by $\mathscr{A}$ as test session, where $\alpha_A = \eta_A Q_A, \beta_A = \eta_A P, \gamma_A = H_2(g_2^{s\eta_A}, g_2^s, f_A^{s\eta_A}, ID_A, ID_B)$ and similarly, the part of $Comm_B$. Now, we claim that neither $Comm_A$ nor $Comm_B$ are computed by $\mathscr{A}$ itself. Suppose $\mathscr{A}$ computes $Comm_A$, then the tuple $(g_2^{s\eta_A}, g_2^s, f_A^{s\eta_A}, ID_A, ID_B)$ must have been queried by $\mathscr{A}$ to H_2 oracle to get γ_A, i.e., $\mathscr{S}$ returns the solution $g_2^s := e(Q_A, Q_B)^s := e(Q,Q)^{sab}$ of the given GBDH problem, which is a contradiction. So, both the messages $Comm_A$ and $Comm_B$ are actually computed by $\mathscr{S}$, i.e., he knows the ephemeral keys η_A and η_B. If $\mathscr{A}$ succeeds this test session, then, it must have done a query to the oracle H by the tuple $(\kappa, \tau, P, sP, sid)$, where $\kappa = g_2^{s(\eta_A+\eta_B)}$, $\tau = \eta_A\eta_B P$, $sid = (Comm_A, Comm_B, ID_A, ID_B)$. Therefore, it computes the solution of the given GBDH as $BDH(Q, U := aQ, V := bQ, W := sQ) := \kappa^{1/(\eta_A+\eta_B)} = e(Q,Q)^{sab}$.

Master Perfect Forward Secrecy: This is almost the same as Mater Forward Secrecy (Case 2.4 of theorem 3.3) except that the adversary $\mathscr{A}$ is involved actively in choosing the message of its own choice and is allowed to learn the static keys SK_A, SK_B as well as the MSK but after the completion of the test session. The simulator $\mathscr{S}$ is given an instance, $P, U = aP, V = bP$ of the CDH problem for $(\mathbb{G}, \mathbb{G}_T, e, q)$. With the help of an adversary $\mathscr{A}$ against this AKE protocol, $\mathscr{S}$ solves (computes abP) the CDH problem. $\mathscr{S}$ picks a scalar $s \xleftarrow{U} \{0,1\}$ as MSK and it publishes $\mathsf{PP} := \{e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T, \boxed{P}, s\boxed{P}, H, H_1, H_2\}$. For each party ID_i, $\mathscr{S}$ chooses $t_i \xleftarrow{U} \mathbb{Z}_q$. It sets the public key as $Q_i := t_i\boxed{P}$ and static key as $\mathsf{SK}_i := sQ_i$. $\mathscr{S}$ can easily answer all the queries even including the static keys of all the parties and MSK but the keys SK_A, SK_B and MSK are given to $\mathscr{A}$ after the completion of test session.

Let $sid = (Comm_A, Comm_B, ID_A, ID_B)$ be the session chosen by $\mathscr{A}$ during test phase. We claim that both the messages $Comm_A$ and $Comm_B$ could not be created by the adversary $\mathscr{A}$ itself as he is unaware of $\mathsf{MSK} = s$. In fact, let $\alpha_B = \eta_B Q_B, \gamma_B = H_2(g_2^{s\eta_B}, g_2^s, f_B^{s\eta_B}, ID_B, ID_A)$. As earlier, $\mathscr{A}$ essentially could not compute $g_2^s := e(Q_A, Q_B)^s = e(P,P)^{st_At_B}$ (which is the solution of BDH instance, P, t_AP, t_BP, sP, a contradiction).

Therefore, both the messages must be computed by the simulator $\mathscr{S}$. So, it sets $\alpha_A := t_A\boxed{U} = aQ_A$, $\beta_A := \boxed{U}$, $\gamma_A := H_2(g_2^{sa}, g_2^s, f_A^{sa}, ID_A, ID_B) \xleftarrow{U} \{0,1\}^\mu$ for message $Comm_A$ and it sets $\alpha_B := t_B\boxed{V} = bQ_B$, $\beta_B := \boxed{V}$, $\gamma_B := H_2(g_2^{sb}, g_2^s, f_B^{sb}, ID_B, ID_A) \xleftarrow{U} \{0,1\}^\mu$ for $Comm_B$ during the send query by $\mathscr{A}$. If $\mathscr{A}$ succeeds this test session, then, it must have queried the oracle H by a tuple $(\kappa, \tau, P, sP, sid)$, where $\kappa = g_2^{s(a+b)}$, $\tau = abP$, $sid = (Comm_A, Comm_B, ID_A, ID_B)$. Then, it computes the solution of the given CDH as $CDH(P, U = aP, V = bP) := \tau = abP$. □

3.5 eCK Secure ID-Based AKE Protocol with MPFS (Π_5)

In this section, we propose an ID-based eCK secure AKE protocol with MPFS which requires less pairing computations than the previous eCK secure protocol with MPFS (Π_4).

Setup(1^λ): Same as protocol Π_2.
KeyAgreement: The following is the description of a single round ID-based key exchange protocol between two parties with identities ID_A and ID_B.

Pre-Computation : Same as protocol Π_2.

Post-Computation : For details, see the Table 6.

Theorem 3.5. *If the* GBDH *assumption holds for* $(\mathbb{G}, \mathbb{G}_T, e, q)$*, the* CDH *assumption holds for* $(\mathbb{G}, q)$ *and* H, H_1, H_2 *are random oracles, then the proposed Protocol* Π_5 *is eCK secure with Master FPS.*

Proof. The proof is similar to that of Theorem 3.4. □

Table 6: Our eCK secure ID-Based AKE Protocol with MPFS (Π_5)

ID_A	ID_B
$Q_A := H_1(ID_A), \mathsf{SK}_A := sQ_A$ Pre-Comp: $g_2^s := e(\mathsf{SK}_A, Q_B)$	$Q_B := H_1(ID_B), \mathsf{SK}_B := sQ_B$ Pre-Comp: $g_2^s := e(\mathsf{SK}_B, Q_A)$
$\eta_A \xleftarrow{U} \mathbb{Z}_q,\ \alpha_A := \eta_A Q_A,\ \beta_A := \eta_A P$ $\gamma_A := H_2((g_2^s)^{\eta_A}, g_2^s, ID_A, ID_B)$	$\eta_B \xleftarrow{U} \mathbb{Z}_q,\ \alpha_B := \eta_B Q_B,\ \beta_B := \eta_B P$ $\gamma_B := H_2((g_2^s)^{\eta_B}, g_2^s, ID_B, ID_A)$
$Comm_A := (\alpha_A, \beta_A, \gamma_A) - - - - - - - - - - - - - - - - >$ $< - - - - - - - - - - - - - - - - Comm_B := (\alpha_B, \beta_B, \gamma_B)$	
$g_2^{s\eta_B} := e(\mathsf{SK}_A, \alpha_B),\ f_B^{s\eta_B} := e(\alpha_B, sP)$ ID_A checks two relations below $DBDH(P, sP, \eta_B P, Q_B, f_B^{s\eta_B}) \stackrel{?}{=} 1$ $\gamma_B \stackrel{?}{=} H_2(g_2^{s\eta_B}, g_2^s, ID_B, ID_A)$ If at least one of them is false, then aborts $\kappa_{AB} := g_2^{s\eta_A}.g_2^{s\eta_B},\ \tau_{AB} := \eta_A \beta_B$ $\mathsf{SN}_{AB} := H(\kappa_{AB},\ \tau_{AB},\ P,\ sP,\ sid)$	$g_2^{s\eta_A} := e(\mathsf{SK}_B, \alpha_A),\ f_A^{s\eta_A} := e(\alpha_A, sP)$ ID_B checks two relations below $DBDH(P, sP, \eta_A P, Q_A, f_A^{s\eta_A}) \stackrel{?}{=} 1$ $\gamma_A \stackrel{?}{=} H_2(g_2^{s\eta_A}, g_2^s, ID_A, ID_B)$ If at least one of them is false, then aborts $\kappa_{BA} := g_2^{s\eta_A}.g_2^{s\eta_B},\ \tau_{BA} := \eta_B \beta_A$ $\mathsf{SN}_{BA} := H(\kappa_{BA},\ \tau_{BA},\ P,\ sP,\ sid)$
$sid := (Comm_A, Comm_B, ID_A, ID_B)$	

3.6 eCK Secure ID-based AKE Protocol with MPFS (Π_6)

The protocols Π_4 and Π_5 described above require one decisional oracle test to authenticate the peer. But, still we do not know the cost of that test, i.e., whether its cost is more than a pairing or not. Therefore, this leads the confusion regarding the computational comparison of the protocols, one which involves the decisional test and other which does not. To resolve that issue, we propose an eCK secure ID-based AKE protocol with Master Perfect Forward Secrecy without the decisional oracle test.

Setup(1^λ): Same as protocol Π_2.

KeyAgreement: The following is the description of a single round ID-based key exchange protocol between two parties with identities ID_A and ID_B.

Pre-Computation : Same as protocol Π_2.

Post-Computation : For details, see the Table 7.

Theorem 3.6. *If the* GBDH *assumption holds for* $(\mathbb{G}, \mathbb{G}_T, e, q)$, CDH *assumption holds for* $(\mathbb{G}, q)$ *and* H, H_1, H_2 *are random oracles, then the proposed Protocol* Π_6 *is eCK secure with Master FPS.*

Proof. The proof is similar to that of Theorem 3.4. □

Table 7: Our eCK secure ID-Based AKE Protocol with MPFS (Π_6)

ID_A	ID_B
$Q_A := H_1(ID_A), \mathsf{SK}_A := sQ_A$	$Q_B := H_1(ID_B), \mathsf{SK}_B := sQ_B$
Pre-Comp: $g_2^s := e(\mathsf{SK}_A, Q_B)$	Pre-Comp: $g_2^s := e(\mathsf{SK}_B, Q_A)$
$\eta_A \xleftarrow{U} \mathbb{Z}_q,\ \alpha_A := \eta_A Q_A,\ \beta_A := \eta_A P$	$\eta_B \xleftarrow{U} \mathbb{Z}_q,\ \alpha_B := \eta_B Q_B,\ \beta_B := \eta_B P$
$\gamma_A := H_2((g_2^s)^{\eta_A}, g_2^s, ID_A, ID_B)$	$\gamma_B := H_2((g_2^s)^{\eta_B}, g_2^s, ID_B, ID_A)$
$Comm_A := (\alpha_A, \beta_A, \gamma_A) - - - - - - - - - - - - - - - - >$	
$< - - - - - - - - - - - - - - - - Comm_B := (\alpha_B, \beta_B, \gamma_B)$	
$g_2^{s\eta_B} := e(\mathsf{SK}_A, \alpha_B)$	$g_2^{s\eta_A} := e(\mathsf{SK}_B, \alpha_A)$
ID_A checks two relations below	ID_B checks two relations below
$e(\alpha_B, P) \stackrel{?}{=} e(\beta_B, Q_B)$	$e(\alpha_A, P) \stackrel{?}{=} e(\beta_A, Q_A)$
$\gamma_B \stackrel{?}{=} H_2(g_2^{s\eta_B}, g_2^s, ID_B, ID_A)$	$\gamma_A \stackrel{?}{=} H_2(g_2^{s\eta_A}, g_2^s, ID_A, ID_B)$
If at least one of them is false, then aborts	If at least one of them is false, then aborts
$\kappa_{AB} := g_2^{s\eta_A}.g_2^{s\eta_B},\ \tau_{AB} := \eta_A\beta_B$	$\kappa_{BA} := g_2^{s\eta_A}.g_2^{s\eta_B},\ \tau_{BA} := \eta_B\beta_A$
$\mathsf{SN}_{AB} := H(\kappa_{AB},\ \tau_{AB},\ P,\ sP,\ sid)$	$\mathsf{SN}_{BA} := H(\kappa_{BA},\ \tau_{BA},\ P,\ sP,\ sid)$
$sid := (Comm_A, Comm_B, ID_A, ID_B)$	

4 Conclusion

We have proposed, in the ID-based setting, single round eCK secure AKE protocols with PFS and MPFS from the GBDH problem. These seem to be the first such schemes. To construct these protocols, we have used the *peer authentication mechanism*, whereby Bob can verify Alice's message with only his static key. Due to this authentication mechanism, our schemes seem to achieve good efficiency.

References

[1] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Proc. of CRYPTO'01, Santa Barbara, California, USA, LNCS*, volume 2139, pages 213–229. Springer-Verlag, August 2001.

[2] R. Canetti and H. Krawczyk. Universally composable notions of key exchange and secure channels. In *Proc. of the 21st Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'02), Amsterdam, Netherlands, LNCS*, volume 2332, pages 337–351. Springer-Verlag, April-May 2002.

[3] L. Chen, Z. Cheng, and N. P. Smart. Identity-based key agreement protocols from pairings. *Journal of Information Security*, 6(4):213–241, July 2007.

[4] L. Chen and C. Kudla. Identity based authenticated key agreement protocols from pairings. In *Proc. of the 16th IEEE computer security foundations workshop (CSFW'03), Pacific Grove, California, USA*, pages 219–233. IEEE, June-July 2003.

[5] S. S. M. Chow and K. K. R. Choo. Strongly-secure identity-based key agreement and anonymous extension. In *Proc. of the 10th International Conference on Information Security (ISC'07), Valparaíso, Chile, LNCS*, volume 4779, pages 203–220. Springer-Verlag, October 2007.

[6] C. Cremers. Examining indistinguishability-based security models for key exchange protocols: The case of ck, ck-hmqv, and eck. In *Proc. of the 6th ACM Symposium on Information, Computer and Communications Security (ASIACCS'11), Hong Kong, China*, pages 80–91. ACM, March 2011.

[7] C. Cremers and M. Feltz. One-round strongly secure key exchange with perfect forward secrecy and deniability. Cryptology ePrint Archive, Report 2011/300, 2011. http://eprint.iacr.org/2011/300.

[8] W. Diffie and M. E. Hellman. New directions in cryptography. *Journal of IEEE Transactions on Information Theory*, 22(6):644–654, November 1976.

[9] A. Fujioka, F. Hoshino, T. Kobayashi, K. Suzuki, B. Ustaoglu, and K. Yoneyama. id-eck secure id-based authenticated key exchange on symmetric and asymmetric pairing. *Journal of IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, 96(6):1139–1155, July 2013.

[10] A. Fujioka, K. Suzuki, and B. Ustaoğlu. Ephemeral key leakage resilient and efficient id-akes that can share identities, private and master keys. In *Proc. of the 4th International Conference on Pairing-Based Cryptography (PAIRING'10),Yamanaka Hot Spring, Ishikawa, Japan, LNCS*, volume 6487, pages 187–205. Springer-Verlag, December 2010.

[11] H. Huang. An eck-secure one round authenticated key exchange protocol with perfect forward security. In *Proc. of 5th International Conference on Provable Security (PROVSEC'11), Xi'an, China, LNCS*, volume 6980, pages 389–397. Springer-Verlag, October 2011.

[12] H. Huang and Z. Cao. An id-based authenticated key exchange protocol based on bilinear diffie-hellman problem. Cryptology ePrint Archive, Report 2008/224, 2008. http://eprint.iacr.org/2008/224.

[13] A. Joux. A one round protocol for tripartite diffie-hellman. In *Proc. of the 4th International Symposium on Algorithmic Number Theory (ANTS-IV'00), Leiden, The Netherlands, LNCS*, volume 1838, pages 385–394. Springer-Verlag, July 2000.

[14] B. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. In *Proc. of 1st International Conference on Provable Security (PROVSEC'07), Wollongong, Australia, LNCS*, volume 4784, pages 1–16. Springer-Verlag, November 2007.

[15] B. Libert and J. Quisquater. Identity based undeniable signatures. In *Proc. of the 5th International Conference on Cryptographers' Track at the RSA Conference (CT-RSA'04), San Francisco, California, USA, LNCS*, volume 2964, pages 112–125. Springer-Verlag, February 2004.

[16] B. N, McCullagh and B. PSLM. A new two-party identity-based authenticated key agreement. In *Proc. of the 6th International Conference on Cryptographers' Track at the RSA Conference (CT-RSA'05), San Francisco, California, USA, LNCS*, volume 3376, pages 262–274. Springer-Verlag, February 2005.

[17] L. Ni, G. Chen, J. Li, , and Y. Hao. Strongly secure identity-based authenticated key agreement protocols. *Journal of Computers & Electrical Engineering*, 37(2):205–217, March 2011.

[18] T. Pandit and R. Barua. eck secure single round id-based authenticated key exchange protocols with master perfect forward secrecy. In *Proc. of the 8th International Conference on Network and System Security (NSS'14), Xi'an, China, LNCS*, volume 8792, pages 435–447. Springer-Verlag, October 2014.

[19] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *Proc. of the 2000 Symposium on Cryptography and Information Security (SCIS'00), Okinawa, Japan*, pages 135–148, January 2000.

[20] A. Shamir. Identity based cryptosystems and signature schemes. In *Proc. of CRYPTO'84, Santa Barbara, California, USA, LNCS*, volume 196, pages 47–53. Springer-Verlag, August 1984.

[21] N. P. Smart. Identity-based authenticated key agreement protocol based on weil pairing. *Journal of Electronics Letters*, 38(13):630–632, 2002.

[22] S. Wang, Z. Cao, K. K. R. Choo, and L. Wang. An improved identity-based key agreement protocol and its security proof. *Journal of Information Sciences*, 179(3):307–318, January 2009.

[23] Y. Wang. Efficient identity-based and authenticated key agreement protocol. Cryptology ePrint Archive, Report 2005/108, 2005. http://eprint.iacr.org/2005/108.

Author Biography

Tapas Pandit received the M.Sc degree in Pure Mathematics from Calcutta University in 2005 and M.Tech degree in Computer Science from Indian Statistical Institute, Kolkata in 2008. Currently he is pursuing Ph.D in Cryptography under the supervision of Prof. Rana Barua at Indian Statistical Institute, Kolkata, India. His research interests include public key cryptography, attribute-based cryptography and functional cryptography.

Rana Barua received his Ph.D. under the supervision of Ashok Maitra in Descriptive Set Theory in 1987 from I.S.I. Since then he has been at I.S.I. in various capacities and is currently a Professor of Mathematics in the Stat-Math Unit. He is also the Acting Head of the R.C.Bose Centre of Cryptology and Security, I.S.I He has been a visiting faculty at the University of Miami, Coral Gables during 1999-2000. Apart from Descriptive Set Theory, he has worked in Recursion Theory, Automata Theory, DNA Computing and Simple Voting Games and has made significant contributions in these areas. His current research interest is in Cryptography, in general, and cryptographic protocols, in particular.

Somanath Tripathy is an Assistant Professor in the Computer Science and Engineering Department of IIT Patna. He joined the faculty in December 2008. Dr Tripathy received his PhD in Computer Science and Engineering from IIT Guwahati in 2007. His research Interest includes lightweight cryptography, Computer and Network security. He has published 37 research papers in peer reviewed journals and conferences. He is a member of IEEE and life member of CRSI (Cryptology Research Society of India)