An Efficient Multiple PKG Compatible Identity Based Key Agreement for Vehicular Networks

Harish Karthikeyan[2], Suvradip Chakraborty[1], Kunwar Singh[2] and C. Pandu Rangan[1]

[1] Department of Computer Science and Engineering,
Indian Institute of Technology Madras, India
suvradip@cse.iitm.ac.in, rangan@cse.iitm.ac.in
[2] Department of Computer Science and Engineering,
National Institute of Technology,Tiruchirapalli, India,
harishk93@live.com, kunwar@ntt.edu

Abstract. In this paper we propose an efficient single-round, two-party identity based authenticated key agreement protocol in the setting of multiple Private Key Generators (PKGs). One of the major advantages of our construction is that it does not involve any pairing operations. To date, existing protocols in the Identity Based Key Agreement domain revolves around a single PKG environment. Efforts to exploit the multiple PKGs paradigm have placed excessive reliance on Elliptic Curve Cryptography and bilinear pairings. These are computationally intensive and cannot be used when computation is premium, such as in a Vehicular Ad-Hoc Network (VANET), specially when the vehicles in a VANET need to perform a large of key agreement. Previous attempts to model identity based key agreement in multiple PKG scenario by Chen and Kundla, McCullagh have very limited scope and provide weak security guarantees. We propose a new security model for identity based key agreement protocols involving multiple PKGs based on the eCK security model which is much more stronger than the existing models and captures additional properties like Key Compromise Impersonation and forward secrecy that were not captured by the previous models. Our protocol is proven secure in this new security model under the Gap Diffie Hellman (GDH) assumption in the Random Oracle (RO) model.

Keywords: Identity Based Authenticated Key agreement (IDKA), Provable Security, Random Oracle Model, eCK model, Vehicular Ad-Hoc Network (VANET), Multiple PKG, Pairing-Free

1 Introduction

Key Exchange Protocols are designed and conceived for the exchange of communication between two parties in a secure manner in an insecure environment with the help of the established key. The established key is a key shared between the two parties in question, calculated from the values exchanged. As can be inferred, the security pitfall of this concept is the lack of authentication of the received messages. This is remedied with the notion of Authenticated Key Exchange (AKE) which allows the parties to authenticate each other before the key is established.

The first Key Agreement Protocol was proposed in [DH76]. This was a public key based system where every party has a pair of keys, the public key and the private key. These keys were used to establish the key. This was the most primitive model with susceptibility to an active adversary for a man-in-the-middle-attack. Security was guaranteed with the adoption of a certificate based model to authenticate the two parties marking the advent of Authenticated Key Exchange. Certificates are signed documents by a trusted party. The parties involved are mandated to obtain and verify certificates whenever key establishment process happens. This brings in the issue of public key certificate management with regards to revocation and updation. The constant need for communication with the trusted authority to regenerate certificates could also give room for increased communication cost. These drawbacks were addressed in the identity-based key agreement model.

The identity based model uses the identity of the party as a certificate which provides the binding between the key and the party. The gain in using the model is that there is no requirement of centralized certification authorities. The parties need not cross-verify each other and there is no need for a communication to relay revocation information. To generate a shared key, the parties need to have in its possession its own secret key, the public parameters of the KGC and the identity of the peer party with which the key needs to be established. Identity Based Key Agreement thus have their own relative merits over its Public Key version but is bogged down by a severely crippling cap on the number of users a KGC can deal with. This limitation curtails the versatility of the protocol and often diminishes the benefits of an Identity Based model. This stumbling block is remedied by the notion of multiple PKGs or KGCs which guarantees scalability with a little additional overhead that might be considered negligible when compared to benefits that it offers.

In this world of "smartness", every entity that makes a man's life easier has a degree of inherent smartness. This ranges from the ubiquitous mobile phones to the vehicles that ply on the road. The "smart" capability of vehicles can be exploited to create a network among them. These networks are called Vehicular Ad-Hoc Networks (VANETs). VANETs are being looked at for transferring information between two vehicles. With the VANETs in place, a car A can inform another car B about an accident, or traffic congestion in an area. The point of this communication is to effectively manage traffic and at the same time pass on critical information about life threatening events on the road for speedy recovery attempts. VANETs have enormous security requirements owing to two factors. The first factor being the Ad-Hoc, wireless and mobile communication which imply security threats. The second factor is the reliability of the message as a single error could precipitate catastrophic events. The vehicle ordering on the road is also dynamic in nature. Since vehicles on the road come from the manufacturing plants of several different companies, for security purposes, these companies would each want to have a PKG for vehicles from their plants. This is a very valid expectation and thus we cannot have communication between only vehicles of the same manufacturer. This provides the impetus for research into the domain of multiple PKGs.

2 Related Work and Our Contribution

Following Shamir's proposal for the paradigm of identity based cryptography, a number of key agreement protocols were developed in the identity based paradigm. Chen et al present a comparison of these protocols in [CCS07]. But most of them involved pairing and hence their practical implementation was not efficient. For applications, where computation is premium, usage of bilinear pairings offers very little scope. Gunther [Gün90] and Saeednia [Sae00] proposed protocols which did not involve pairing; Fiore [FG10] proposed a key agreement protocol without pairing which was an improvement over them. The first two protocols namely [Gün90] and [Sae00] lacked formal proof of security; whereas the protocol in [FG10] was formally proved in the Canetti-Krawczyk [CK01] model and is also more efficient in terms of computational complexity than the protocols in [Gün90] and [Sae00]. However the same was analysed and vulnerabilities identified in [MM13] and in [CM09]. An improvement on this was done by [VSVR13]. Though identity based protocols overcome the need for certificate based authentication, reducing communication overhead and complex key management, they too have their limitations. These limitations are that single PKG model, by themselves are not scalable. It is unrealistic to expect a single PKG to cater to the requirements of large organizations which often transcend geographical barriers. This provided the catalyst for multiple PKG research. One of the pioneering work was undertaken by Chen and Kudla in [CK03] but it had the issue of assuming all the different PKGs to have the same system parameters and differ only in their master secret keys. The work undertaken by Lee et al in [LKKO05] was a documentation of exclusive research in the multiple PKG domain. However there was no concrete security proof and the security analysis was done heuristically based on knowledge of existing attacks. This was also pairing based. Another pairing-based protocol was proposed by McCullagh in [MB05]. This provided for communication between different domains and was proven secure in the BJM Security Model proposed in [BWJM97], a modification of the BR model proposed by Bellare and Rogaway for symmetric key settings in [BR94]. Yet another heuristically proved protocol was proposed in [VYK13]. A pairing-free protocol for multiple PKG was proposed by Farash and Attari in [FAA14] but it stemmed from certain vulnerabilities as demonstrated in [MM13]. The security of the protocol was also analyzed in the BJM security model which is a much weaker security model compared to the more advanced security models such as the CK model proposed in [CK01] and the eCK model proposed in [LLM07]. Existing literature has sought to use Public Key Infrastructure for Key Agreement for communication. Such protocols suffer from Man-in-the-Middle attack susceptibility. An active adversary could intercept key communication and tamper with the same leading to catastrophic consequences. The solution proposed by Dolev et al in [DKPS13] was to certify the key and the attributes together. Even with certificates, out-of-band sensing is needed to ensure that there are no signed certificate thefts. The paper [DKPS13] requires attribute verification by a camera, microphone and other devices. The issue with regards to the Certificate based system would be the overhead needed to reissue certificate post expiry or revocation. Though this is not involved as an overhead during the actual session-key establishment, this is still an additional overhead.

2.1 Our Contribution

Through this paper, we aim to make contributions along these lines.

1. **New Security Model.** We propose a new security model based on the eCK model exclusively for multiple PKG. The adversary has greater power in this model. Hitherto security proofs have been based on heuristic proving or the weaker security model of BR and BJM model.
2. **Pairing-Free.** Our protocol is pairing-free and single round. This increases the efficiency and maintains its utility in applications where computation and time is premium. Bilinear pairings have markedly higher computation cost when compared to regular group arithmetic computations. Our protocol is proven secure in the Gap-Diffie Hellman complexity assumption in the Random Oracle model.
3. **Secure Against Active Attacks.** Some of the standard protocols such as the ones in [FG10] and [FAA14] are vulnerable to active attacks such as Key-Offset and Forgery. We overcome this limitation by including a term which is a signature of the ephemeral keys. This provides enhanced security for a little extra overhead.

Table 1 compares our scheme with existing protocols. It compares based on the number of exponentiation, the number of pairing operations, the security model under the schemes are proven secure and the resistance to active adversary. All of the protocols listed in the table have been designed for the multiple PKG model. From the table we can see that our protocol has been proven secure in a much stronger model when compared to existing protocols. It is also noteworthy that our protocol offers resistance to active adversary by incorporating a signature in the message being sent. This guarantees origin authentication, a feature missing in the other protocols. Another major advantage of our protocol is that it does not involve any pairing operations. In general it is always desirable to have a protocol that involves simple group theoretic operations than pairing as it is slightly inefficient to find too much pairing-friendly curves. Our protocol also removes all the attacks of the scheme presented in [FAA14] as reported in [MM13] and is also proven secure in a much stronger model. But for this we have to pay an extra cost of 3 more exponentiations compared to [FAA14].

Scheme	**Exp (each)**	**Pairing (each)**	**Security Model**	**Active Adversary**
SCK-3 [CK03]	1	2	BJM	$\times$
Lee et al. [LKKO05]	2	2	*N.A.*	$\times$
Farash and Attari [FAA14]	7	*N.A.*	BJM	$\times$
Ours	$7+3^{\#}$	*N.A.*	eCK	$\surd$

#: This cost is for the correctness check.

Table 1 : Comparison with the existing schemes

3 Notations.

In this section we describe the notations we will be using throughout our paper. We denote the security parameter by κ. The set of integers is denoted by $\mathbb{Z}$. $[n]$ denotes the set $\{1,\ldots,n-1\}$ for $n \geq 2$. We denote by $x \in_R X$ the fact that x is chosen uniformly at random from the set X. $\mathbb{G}$ denotes a group of order q (where q is a prime of length κ) with generator g. $\mathbb{G}^*$ denotes the non-zero elements of $\mathbb{G}$. $\mathbb{Z}_p^*$ denotes the multiplicative group of integers modulo p, where p is a prime and $p|(q-1)$.

4 Our Security Model For Multiple PKG Identity Based Key Exchange

In this section we describe our security model for Identity based key agreement protocol in the multiple PKG scenario. We assume that the number of PKGs is fixed at start up. Let us denote the number of PKGs by n. A protocol with the multiple PKG support is made up of two or more roles for the parties involved. The most common of these roles being the the initiator $\mathcal{I}$ and the responder $\mathcal{R}$. The parties involved are defined by their identity $ID_i^{(k)}$ which denotes that the party ID_i is registered with the k^{th} KGC. We further assume that the identifiers of the users are unique globally, i.e., identifiers of different users registered under different PKGs are also unique among themselves. Also we assume that there is a binding between an identifier ID and the PKG under which it is registered with. We can assume this information comes hard-coded along with each identifier. This in turn provides uniqueness among the identities among the users even if two users in two different PKGs have the same identifier. Each of the protocol may have several sub protocols which are initiated by the parent protocol and these can happen multiple times leading to several instances. Each instance of the protocol is called a *session*. With each session we associate a *owner* who initiates the session and the other party involved in the session is called as a *peer*. Messages are transmitted between the owner of the session and a peer of the session in the process of key establishment. These messages are from the message space μ. These messages are based on local computations with some element of preprocessing. The messages that form the basis of communication constitutes the the *session state*. The fundamental requirement of a key exchange protocol is that the parties involved have to compute the same session key. Once the *session key* has been computed, it is given as output and the *session state* is deleted. If the establishment is unsuccessful, the session is in *abort* state, meaning there is no *session key* is generated. The session sid is defined by $sid = \left(ID_i^{(k)}, ID_j^{(l)}, out, in, \zeta\right)$, where $ID_i^{(k)}$ begins this session with its peer $ID_j^{(l)}$ and out and in are respectively the components or messages sent to $ID_j^{(l)}$ and received by $ID_i^{(k)}$ and ζ indicates whether the party is the initiator ($\mathcal{I}$) of the session or responder ($\mathcal{R}$).

In an Identity based Key Agreement (IBKA) protocol, each entity i is defined by a unique identity, ID_i. Each PKG_k maintains its own public parameters $params_k$ which is available publicly and its own master secret key msk_k which is its own secret, and generates the private key S_i for each user i registered under the same PKG. A protocol consists of the following sub-functions:

Setup: Each PKG chooses the public parameters and the master secret key. The public parameters are published and the master secret key is held confidential. The set up algorithm also outputs the number of PKGs which is fixed at start up.

Key Generation: The user i submits its identity $ID_i^{(k)}$ to the PKG_k and the PKG constructs the private key $S_i^{(k)}$ for the user with identity $ID_i^{(k)}$.

Key Agreement: To establish the shared secret key between two users A and B with identities $ID_A^{(k)}$ and $ID_B^{(l)}$ registered with private key generators PKG_k and PKG_l respectively and possessing secret keys $S_A^{(k)}$ and $S_B^{(l)}$ respectively, the users engage in a session by exchanging components and eventually set up the shared secret key. Either user A or B could initiate the protocol.

Definition 1. ***(Partnered Sessions.)*** *Let Π be a protocol and $sid = \left(ID_i^{(k)}, ID_j^{(l)}, out, in, \zeta\right)$ and $sid' = \left(ID_a^{(y)}, ID_b^{(z)}, out', in', \zeta'\right)$ be the identifier of two sessions. Then sid and sid' are called partnered sessions if:*

- $b = i$ *and* $a = j$
- $z = k$ *and* $y = l$
- $in' = out$ *and* $out' = in$ *and*
- $\zeta' \neq \zeta$

It is prudent to note that k can be equal to l. This would effectively be a single PKG scenario.

Adversary. We model the adversary as PPTM (Probabilistic Polynomial Turing Machine). In addition to passive eavesdropping, the adversary can control the communication channel by modifying the outgoing messages and their sequence too. The modification can be altering, tampering or injecting new messages. In order to model real life attacks the adversary is allowed to obtain more information such as the long term secret key of parties or ephemeral key of parties (to model *Key Compromise Impersonation* attack) or session keys of completed sessions between two parties (to model *Known Session Key* attack). The amount of information that the adversary gets is modeled by a set of oracle queries as follows:

1. *Send*($ID_i^{(k)}$,$ID_j^{(l)}$,m)*:* This query models the ability of an adversary to perform active attacks, i.e., to intercept all communication, send, receive, alter, delete and reschedule messages. It sends a message m to $ID_i^{(k)}$ on behalf of $ID_j^{(l)}$ and returns $ID_i^{(k)}$'s response to this message to the adversary. If $m = 0$, this query makes party $ID_i^{(k)}$ to start an AKE session with $ID_j^{(l)}$ and to provide communication from $ID_j^{(l)}$ to $ID_i^{(k)}$. Else it will send the message m from party $ID_i^{(k)}$ to party $ID_j^{(l)}$ and makes $ID_j^{(l)}$ respond to the supposed session $\left(ID_i^{(lk}, ID_j^{(l)}, m, \star, \mathcal{I}\right)$.
2. *Corrupt*($ID_i^{(k)}$)*:* This query allows an adversary to obtain the long term key of user ID_i registered with private key generator PKG_k. Here $k \in [n+1]$ where n is the number of PKGs fixed at setup.
3. *EphemeralKeyReveal(sid):* This query allows the adversary to get the ephemeral keys of the session *sid*.
4. *SessionKeyReveal(sid):* This query allows an adversary to learn the session key of a completed session with identifier *sid*.
5. *MasterReveal(k,z):* This query allows the adversary to learn the master secret key of PKGs k and z. Note that if the communication takes place between parties registered with same PKG, then $k = z$.
6. *EstablishParty*($ID_i^{(k)}$)*:* This query allows an adversary to register a party with identity $ID_i^{(k)}$ to the system. A party against which this query is not issued is said to be *honest*, otherwise it is at the complete control of the adversary.
7. *Test(sid):* The adversary chooses a test session among all the completed and fresh sessions. The challenger tosses a random bit $b \in_R \{0,1\}$. If $b = 0$ the challenger will give the adversary the correctly computed session key K_0 of the test session. Otherwise the challenger will select a random shared secret key K_1 from the distribtuion of session keys and provide the adversary with K_1.

We now state the definition of freshness of a session.

Definition 2. ***(Session Freshness).*** *Let Π be a protocol, and sid $= (ID_i^{(k)}, ID_j^{(l)}, out, in, \zeta)$ be the identifier of a completed session. The session sid is said to be locally-exposed if any of the following conditions holds:*

- *$\mathcal{A}$ issued a SessionKeyReveal query on sid or its matching session sid$'$ (if one exists).*

– *A matching session sid$'$ exists and $\mathcal{A}$ issued Corrupt on $ID_i^{(k)}$ and EphemeralKeyReveal on sid.*
– *A matching session sid$'$ exists and $\mathcal{A}$ issued Corrupt on $ID_j^{(l)}$ and EphemeralKeyReveal on sid$'$.*
– *$\mathcal{A}$ issued an EstablishParty($ID_i^{(k)}$) or EstablishParty($ID_j^{(l)}$).*
– *A matching session sid$'$ does not exist and $\mathcal{A}$ issued Corrupt on $ID_j^{(l)}$*
– *A matching session sid$'$ does not exist and $\mathcal{A}$ issued both Corrupt on $ID_i^{(k)}$ EphemeralKeyReveal on sid.*

A session sid is said to be exposed if (a) it is locally exposed, or (b) its matching session sid$'$ exists and is locally exposed. A session that is not exposed is called fresh.

Note that we exclude *MasterReveal(k,z)* query from the freshness definition. This is because if the adversary issues *MasterReveal(k,z)*, it is equivalent to issuing *Corrupt($ID_i^{(k)}$)* and *Corrupt($ID_i^{(z)}$)* queries simultaneously for the involved parties. Since our freshness allows for both simultaneous corrupt queries on the two parties, the security guarantee of our protocol does not break in the presence of Master Reveal query.

Definition 3. ***(IBKA- multiple(m) PKG security)****. Let Π be a protocol with the property that if two honest parties complete matching sessions, the two parties compute the same session key. This is defined as the correctness of the protocol. The protocol Π is said to be IBKA-mPKG-secure, if no polynomially bounded adversary can distinguish a fresh session key from a random value, chosen from the distribution of session keys, with probability significantly greater than 1/2.*
The formal definition of a test query has been defined above. It is essential to note that only one query of this form is allowed. After the Test query has been issued, the adversary can adaptively query the oracles like before provided the test session remains fresh in the sense of Definition 2. Finally, $\mathcal{A}$ outputs his guess b' in the test session. An adversary wins the game if he guesses the challenge correctly i.e., $b' = b$. The advantage of $\mathcal{A}$ against Π in the IBKA-mPKG model is defined as

$$\mathsf{Adv}_{\Pi}^{\mathcal{IBKA}-m\mathcal{PKG}}(\mathcal{A}) = |\Pr[b' = b] - \frac{1}{2}|$$

The IBKA-mPKG security of Π is defined as follows:

1. *If two honest parties complete matching sessions, then with overwhelming probability they both compute the same session key.*
2. *For any probabilistic polynomial-time adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\Pi,\mathcal{IBKA}-m\mathcal{PKG}}$ is negligible.*

5 Complexity Assumptions.

In this section, we present the hard problem assumptions that we will be using to prove the security of our protocol.

Definition 4. ***Computation Diffie-Hellman Problem (CDH) -*** *Given $(g, g^a, g^b) \in_R \mathbb{G}^3$ for unknown $a, b \in \mathbb{Z}_q^*$, where $\mathbb{G}$ is a cyclic prime order multiplicative group with g as a generator and q the order of the group, the CDH problem in $\mathbb{G}$ is to compute g^{ab}.*

The advantage of any probabilistic polynomial time algorithm $\mathcal{A}$ in solving the CDH problem in $\mathbb{G}$ is defined as

$$Adv_{\mathcal{A}}^{CDH} = Pr\left[\mathcal{A}(g, g^a, g^b) = g^{ab} \mid a, b \in \mathbb{Z}_q^*\right]$$

The CDH Assumption is that, for any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage $Adv_{\mathcal{A}}^{CDH}$ is negligibly small.

Definition 5. ***Decisional Diffie-Hellman Problem (DDH) -*** *Given $(g, g^a, g^b, h) \in \mathbb{G}^4$ for unknown $a, b \in \mathbb{Z}_q^*$, where $\mathbb{G}$ is a cyclic prime order multiplicative group with g as a generator and q the order of the group, the DDH problem in $\mathbb{G}$ is to check whether $h \overset{?}{=} g^{ab}$.*

The advantage of any probabilistic polynomial time algorithm $\mathcal{A}$ in solving the DDH problem in $\mathbb{G}$ is defined as

$$Adv_{\mathcal{A}}^{DDH} = |Pr\left[\mathcal{A}(g, g^a, g^b, g^{ab}) = 1\right] - Pr\left[\mathcal{A}(g, g^a, g^b, h) = 1\right]| \mid a, b \in \mathbb{Z}_q^*$$

The DDH Assumption is that, for any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage $Adv_{\mathcal{A}}^{DDH}$ is negligibly small.

Definition 6. ***Gap Diffie Hellman Assumption (GDH).*** *Given $(g, g^a, g^b) \in_R \mathbb{G}^3$ and access to a Decision Diffie Hellman (DDH) oracle $\mathcal{DDH}(\cdot,\cdot,\cdot)$ which on input g^a, g^b and g^c outputs* `True` *if and only if $c = ab$, the Gap Diffie Hellman problem is to compute $g^{ab} \in \mathbb{G}$.*
The advantage of an adversary $\mathcal{A}$ in solving the Gap Diffie Hellman problem is defined as

$$\mathsf{Adv}_{\mathcal{A}}^{GDHP} = \Pr\left[\mathcal{A}^{\mathcal{DDH}(\cdot,\cdot,\cdot)}(g, g^a, g^b) = g^{ab}\right]$$

The Gap Diffie Hellman assumption holds in $\mathbb{G}$ if for all polynomial time adversaries $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{GDHP}$ is negligible.

6 Our Construction

In this section we present the construction of our protocol. The following protocol is modified from the protocol proposed in [FAA14]. As demonstrated in [MM13], the protocol proposed by Farash and Attari suffers from attacks mounted by active adversary who has the power to intercept and modify the messages being exchanged. Ours construction fixes all these flaws that is pointed out by [MM13]. This is effectively a Man-in-the-middle attack. Two attacks of this kind are the *Key Offset Attack* and the *Forgery Attack*. The intuitive idea of how our protocol avoids all these and other attacks is discussed in section 7.

We now present our protocol. Let ID_A and ID_B be the identities of two vehicles belonging to the VANET and participating in the IDKA protocol. They may be registered under a single PKG or under two different PKGs also.

Setup: On input 1^{κ}, this algorithm outputs *params*, a set of system parameters. Each KGC$_i$ ($i \in [n]$) configure its parameters as follows:

- Choose a group $\mathbb{G}^{(i)}$ of order $q^{(i)}$ where $q^{(i)}$ is a prime and a generator $g^{(i)} \in \mathbb{G}^{(i)}$. Also choose a multiplicative subgroup $\mathbb{Z}^*_{p^{(i)}}$ where $p^{(i)}$ is a prime and $p^{(i)}|(q^{(i)}-1)$.
- Choose two hash functions, ${H_1}^{(i)} : \{0,1\}^* \to \mathbb{Z}^*_{q^{(i)}}$ and ${H_2}^{(i)} : \{0,1\}^* \to \mathbb{Z}^*_{q^{(i)}}$
- Then it picks a random $x^{(i)} \xleftarrow{\$} \mathbb{Z}^*_{q^{(i)}}$ and sets $y^{(i)} = \left(g^{(i)}\right)^{x^{(i)}}$
- Publishes the parameters $\langle y^{(i)}, H_1^{(i)}, H_2^{(i)}, \mathbb{G}, q^{(i)}, g^{(i)} \rangle$.

Key Generation: A vehicle with identity ID_U and registered under KGC$_i$ sends its attributes and a hash of the attributes to the KGC$_i$. The KGC verifies if these attributes are indeed bona-fide through out of band sensing. If the attributes are indeed valid, it checks if the hashed value is correct. After verifying it proceeds as follows:

- Chooses $r_U \xleftarrow{\$} \mathbb{Z}^*_{p^{(i)}}$ and compute $R_U = \left(g^{(i)}\right)^{r_U}$
- Compute $S_U = r_U + H_1^{(i)}\left(ID_U, R_U\right).x^{(i)}$

U's long term private key is $\langle S_U, R_U \rangle$ and is transmitted via a secure channel.

Key Agreement: When a user A with identity ID_A registered with PKG_k wants to communicate with a user B with identity ID_B registered with PKG_l, it sends a message to B initiating conversation. To generate this message, A chooses two ephemeral secret keys, one each from the group corresponding to the public *params* of PKG_k and PKG_l. It then generates the message components which are the ephemeral public keys. It also sends a signature of the ephemeral values being sent to guard against active adversary. It also raises the generator of PKG_l to its long term secret key for key component. Once the components are in place, A sends the message to B. B checks for the correctness of the message and responds with a similar message. Once the message transfer is complete the users compute the session key. This is outlined in detail in 1.

Key Sanity Check: After receiving the private key from the PKG in the Key Generation phase, the user performs the following check to ensure the correctness of the components of the private key.

$\left(g^{(i)}\right)^{S_U} \stackrel{?}{=} R_U \cdot \left(y^{(i)}\right)^{{H_1}^{(i)}(ID_U, R_U)}$.

Vehicle A	**Vehicle B**
1. Chooses $a^{(1)} \in_R Z^*_{q^{(1)}}$ and $a^{(2)} \in_R Z^*_{q^{(2)}}$	**1.** Chooses $b^{(1)} \in_R Z^*_{q^{(1)}}$ and $b^{(2)} \in_R Z^*_{q^{(2)}}$
2. Compute:	**2.** Compute:
(a) $T_A^{(1)} = (g^{(1)})^{a^{(1)}}$	(a) $T_B^{(1)} = (g^{(1)})^{b^{(1)}}$
(b) $T_A^{(2)} = (g^{(2)})^{a^{(2)}}$	(b) $T_B^{(2)} = (g^{(2)})^{b^{(2)}}$
3. Compute:	**3.** Compute:
(a) $V_A^{(1)} = S_A + H_2^{(1)}\left(ID_A, T_A^{(1)}, T_A^{(2)}\right).a^{(1)}$	(a) $V_B^{(1)} = (g^{(1)})^{S_B}$
(b) $V_A^{(2)} = (g^{(2)})^{s_A}$	(b) $V_B^{(2)} = S_B + H_2^{(2)}\left(ID_B, T_B^{(1)}, T_B^{(2)}\right).b^{(2)}$

$$\xrightarrow{\langle ID_A, T_A^{(1)}, T_A^{(2)}, R_A, V_A^{(1)}, V_A^{(2)} \rangle}$$

$$\xleftarrow{\langle ID_B, T_B^{(1)}, T_B^{(2)}, R_B, V_B^{(1)}, V_B^{(2)} \rangle}$$

Correctness Check:	***Correctness Check:***
4. Computes and Checks:	**4.** Computes and Checks:
(a) $C_B = R_B.(y^{(2)})^{H_1{}^{(2)}(ID_B, R_B)}$	(a) $C_A = R_A.(y^{(1)})^{H_1{}^{(1)}(ID_A, R_A)}$
(b) $(g^{(2)})^{V_B^{(2)}} \overset{?}{=} C_B.(T_B^{(2)})^{H_2^{(2)}(ID_B, T_B^{(1)}, T_B^{(2)})}$	(b) $(g^{(1)})^{V_A^{(1)}} \overset{?}{=} C_A.(T_A^{(1)})^{H_2^{(1)}(ID_A, T_A^{(1)}, T_A^{(2)})}$
If Check 4 is correct *proceed*:	If Check 4 is correct, *proceed*:
Session Key Generation:	***Session Key Generation:***
(i) $K_A^{(1)} = (V_B^{(1)}.T_B^{(1)})^{(a^{(1)} + S_A)}$	(i) $K_B^{(1)} = (C_A.T_A^{(1)})^{(b^{(1)} + S_B)}$
(ii) $K_A^{(2)} = (C_B.T_B^{(2)})^{(a^{(2)} + S_A)}$	(ii) $K_B^{(2)} = (V_A^{(2)}.T_B^{(2)})^{(b^{(2)} + S_B)}$
(iii) $K_A^{(3)} = \left(T_B^{(1)}\right)^{(a^{(1)})}$	(iii) $K_B^{(3)} = \left(T_A^{(1)}\right)^{(b^{(1)})}$
(iv) $K_A^{(4)} = (T_B^{(2)})^{(a^{(2)})}$	(iv) $K_B^{(4)} = (T_A^{(2)})^{(b^{(2)})}$

Final Session Key Generation

A: $K_{AB} = H\{ID_A, ID_B, T_A^{(1)}, T_A^{(2)}, T_B^{(1)}, T_B^{(2)}, K_A^{(1)}, K_A^{(2)}, K_A^{(3)}, K_A^{(4)}\}$

B: $K_{BA} = H\{ID_A, ID_B, T_A^{(1)}, T_A^{(2)}, T_B^{(1)}, T_B^{(2)}, K_B^{(1)}, K_B^{(2)}, K_B^{(3)}, K_B^{(4)}\}$

Algorithm 1: Description of the Key Agreement protocol

This can be verified as:
$\left(g^{(i)}\right)^{S_U} = \left(g^{(i)}\right)^{r_U + H_1^{(i)}(ID_U, R_U).x^{(i)}} \cdots\cdots\cdots\cdots (i)$ and
$R_U \cdot \left(y^{(i)}\right)^{{H_1}^{(i)}(ID_U, R_U)} = g^{r_U} \cdot \left(g^{(i)}\right)^{x^{(i)} \cdot {H_1}^{(i)}(ID_U, R_U)} = g^{r_U + x^{(i)} \cdot {H_1}^{(i)}(ID_U, R_U)} \ldots (ii)$
Hence $(i) = (ii)$ and if the secret key was properly constructed by the PKG the check goes through.

Remark 1. The protocol is synchronous and consists of only one send per user per session. There is an initiator and a peer in this model.

Remark 2. The values of ID_i, R_i $(i \in \{A, B\})$ are time invariant and they need to be sent only on the first communication between the parties. From the next exchange onwards the parties can send only T_i and V_i values.

Remark 3. ***Correctness Check*** ensures safeguard against a dynamic adversary. A dynamic adversary can tamper the components exchanged which affects session key generation.

Lemma 1. *The shared secret key computed by both the parties are identical.*

Proof. The key components have been constructed as given below:

$$K_A^{(1)} = K_B^{(1)} = \left(g^{(1)}\right)^{\left(a^{(1)} + S_A\right)\left(b^{(1)} + S_B\right)}$$
$$K_A^{(2)} = K_B^{(2)} = \left(g^{(2)}\right)^{\left(a^{(2)} + S_A\right)\left(b^{(2)} + S_B\right)}$$
$$K_A^{(3)} = K_B^{(3)} = \left(g^{(1)}\right)^{\left(a^{(1)}\right)\left(b^{(1)}\right)}$$
$$K_A^{(4)} = K_B^{(4)} = (g^{(2)})^{(a^{(2)})(b^{(2)})}$$

Thus the session keys K_{AB} and K_{BA} are equal.

□

7 Security Proof

In this section we present the formal security proof for our protocol described in the previous section.

But before proceeding with the formal security proof we provide an informal discussion how our construction presented in section 6 counters the attacks presented in [MM13] for the protocol [FAA14].

1. Key Offset Attack: In this attack an adversary modifies the message being sent by multiplying the Ephemeral Public Key with its own exponent λ. This offsets the agreed session key by the same exponent unknown to either of the parties. This happens in protocols where key confirmation is absent. In the protocol proposed by Farash and Attari, the absence of origin authentication causes a vulnerability. The same is remedied in our protocol by adding an extra term in the message being sent. The extra term is effectively a Schnorr signature of the original terms being sent. The signature is verified in the correctness check of Steps 5, 8 in Algorithm 1. For an active adversary to successfully mount the attack in this model, it should have the ability to generate a new signature, to go with the message being sent. This cannot be achieved as the signature generation involves the long term secret key of the user, which the adversary is unaware of.

2. Forgery Attack: In this attack, an eavesdropper intercepts the message that a user A broadcasts through the public channel. The eavesdropper uses the intercepted message to assume the identity of A and masquerade as A to other users. By a reasoning similar to that for Key Offset Attack, it is clear that the protocol of Farash and Attari has a vulnerability to this kind of attack and the extra terms being sent in our protocol remedies this vulnerability.

In addition to these attacks, as demonstrated in [MM13], the protocol in [FAA14] is vulnerable to *Known Session Specific Temporary Information Attack* (KSTIA) and *Key Compromise Impersonation Attack* (KCI).

3. KSTIA: In this attack the knowledge of the ephemeral secrets of both the parties gives the adversary the power to compute the session key. Our protocol completely overhauls the key computation aspect of the protocol proposed in [FAA14] to overcome this susceptibility. This is achieved by ensuring that the final key value is dependent on $CDH(g^{S_A}, g^{S_B})$ which has been incorporated in key components K_λ^μ where $\lambda \in \{A, B\}$ and $\mu \in \{1, 2\}$.

4. KCI: It is an attack where the adversary is able to use the compromised long-term key of a party A to masquerade to A as another party B. It is obvious that when the long-term secret key of A is compromised, the adversary can masquerade as A to other users. For the converse to happen, the adversary intercepts the message that B sends to A. The intercepted message is then modified with its own chosen values for the ephemeral secrets and the message is sent to A, effectively adversary masquerades as B to A. It then computes the session key. This attack works in [FAA14] since there is no correctness check. Our protocol avoids this problem by two ways - adding the signature and then checking the correctness and by making the key components dependent on S_B. For the adversary to be able to compute the session key it would need to successfully compute the values of $K_A^{(1)}$ and $K_A^{(2)}$. This requires the knowledge of S_B which the adversary is not aware of. This shows our protocol's resilience to KCI.

We now present the formal security proof of our protocol. The proof is based on the security model described for Identity Based Key Agreement in the Multiple PKG paradigm, IBKA-mPKG which is based on the eCK model. The scheme is proved secure under the Gap Diffie-Hellman (GDH) assumption in the random oracle model. The security proof is modeled as a game between the challenger and the adversary.

Theorem 1. *Under the GDH assumption in $\mathbb{G}$ and the RO model,the protocol in section 6 is $\mathcal{IBKA}-m\mathcal{PKG}$-secure.*

Proof. As empowered by our security model, the adversary $\mathcal{A}$ is allowed to make session activation queries.

– A query of the form $\left(ID_i^{(k)}, ID_j^{(l)}\right)$ makes user (or vehicle) i registered with PKG_k perform **Steps 1-3** of our protocol, and create a session with identifier $\left(ID_i^{(k)}, ID_j^{(l)}, \langle T_i^{(1)}, T_i^{(2)}, R_i, V_i^{(1)}, V_i^{(2)}\rangle, \star, \mathcal{I}\right)$.
– On a query $\left(ID_i^{(k)}, ID_j^{(l)}, \langle T_i^{(1)}, T_i^{(2)}, R_i, V_i^{(1)}, V_i^{(2)}\rangle\right)$, B performs **Steps 4-7** of our protocol and creates a session with identifier $\left(ID_j^{(l)}, ID_i^{(k)}, \langle T_j^{(1)}, T_j^{(2)}, R_j, V_j^{(1)}, V_j^{(2)}\rangle, \langle T_i^{(1)}, T_i^{(2)}, R_i, V_i^{(1)}, V_i^{(2)}\rangle, \mathcal{R}\right)$.
– The query $\left(ID_i^{(k)}, ID_j^{(l)}, \langle T_i^{(1)}, T_i^{(2)}, R_i, V_i^{(1)}, V_i^{(2)}\rangle, \langle T_j^{(1)}, T_j^{(2)}, R_j, V_j^{(1)}, V_j^{(2)}\rangle, \mathcal{R}\right)$ makes A update the session identifier $\left(ID_i^{(k)}, ID_j^{(l)}, \langle T_i^{(1)}, T_i^{(2)}, R_i, V_i^{(1)}, V_i^{(2)}\rangle, \star, \mathcal{I}\right)$ (if any) to $\left(ID_i^{(k)}, ID_j^{(l)}, \langle T_i^{(1)}, T_i^{(2)}, R_i, V_i^{(1)}, V_i^{(2)}\rangle, \langle T_j^{(1)}, T_j^{(2)}, R_j, V_j^{(1)}, V_j^{(2)}\rangle, \mathcal{I}\right)$ and perform **Step 8** and then both the parties involve in session key generation phase of our protocol.

  The adversary is also allowed to make the following queries: *EphemeralKeyReveal, Corrupt, SessionKeyReveal, EstablishParty*, and *MasterReveal* (to obtain the master secret keys msk_l and msk_k)

Let us assume that an adversary $\mathcal{A}$ can win the game, i.e, distinguish a fresh session key from a random session key chosen from the distribution of session keys, with probability significantly greater than $\frac{1}{2}$. $\mathcal{A}$ can do this in one of the following ways:

– **Guessing attack:** $\mathcal{A}$ guesses the test session key correctly.
– **Key Replication attack:** $\mathcal{A}$ succeeds in making two non-matching sessions compute the same session key and then $\mathcal{A}$ simply issues a session key reveal query on one of the sessions and uses that key in the other session.
– **Forging attack:** $\mathcal{A}$ computes the hash digest components correctly and queries the H digest query to get the session key.

Since are proving in the random oracle model, the first two probability of the success of the first two attacks is negligible. This is due to the collision resistance property of the hash function H. Thus it is enough to analyze the event E defined as the event "$\mathcal{A}$ succeeds in forging the session key of a fresh session denoted by $sid_0 = \left(ID_i^{(k)}, ID_j^{(l)}, \langle T_i^{(k)}, T_i^{(l)}, R_i, V_i^{(k)}, V_i^{(l)}\rangle, \langle T_j^{(k)}, T_j^{(l)}, R_j, V_j^{(k)}, V_j^{(l)}\rangle, \zeta\right)$". The event E is divided in two subcases – E.1: "$\mathcal{A}$ succeeds in forging the session key of a fresh session that has a matching session" and E.2: "$\mathcal{A}$ succeeds in forging the session key of a fresh session without a matching session". So if suffices to show that neither E.1 nor E.2 can happen with non-negligible probability.

Analysis of E.1. If event E.1 occurs with non-negligible probability, using $\mathcal{A}$ we can build a GDH solver that succeeds with non-negligible probability. Event E.1 can be further analyzed in each of these sub-cases:

– E.1.1: E.1 $\wedge$ $\mathcal{A}$ issues $Corrupt(ID_i^{(k)})$ and $Corrupt(ID_j^{(l)})$.

– E.1.2: E.1: $\wedge$ $\mathcal{A}$ issues an *EphemeralKeyReveal* query on both sid_0 and sid'_0.

– E.1.3: E.1: $\wedge$ $\mathcal{A}$ issues $Corrupt(ID_i^{(k)})$ and an *EphemeralKeyReveal* query on sid'_0.

– E.1.4: E.1: $\wedge$ $\mathcal{A}$ issues $Corrupt(ID_j^{(l)})$ and an *EphemeralKeyReveal* query on sid_0.

Analysis of E.1.1. Suppose that E.1.1 occurs with non-negligible probability, using $\mathcal{A}$ we can build a polynomial time GDH solver $\mathcal{S}$, which succeeds with non-negligible probability. The solver interacts with $\mathcal{A}$ as follows:
$\mathcal{S}$ simulates $\mathcal{A}$'s environment with $n_{kgc}(\kappa)$ separate KGCs, each KGC can support $n_u(\kappa)$ users sessions where κ is the security parameter. Since $\mathcal{A}$ is polynomial (in $|q|$), we suppose that each party may be involved in $n_s(\kappa)$ sessions ($n_{kgc}(\kappa), n_u(\kappa), n_s(\kappa) \leq \mathcal{L}(|q|)$ for some polynomial $\mathcal{L}$). We refer to $ID_i^{(k)}$ registered with KGC_k as $\hat{P}_i$. $\mathcal{S}$ chooses $\hat{P}_l, \hat{P}_m$, and $t \in_R [n_s(\kappa)+1]$ (with these choices, $\mathcal{S}$ is guessing the test session). The challenger is given the CDH problem instance $\langle \mathbb{G}, g, q, p, C = g^a, D = g^b \rangle$ of the corresponding GDH instance. The challenger simulates the hash oracles in the following way:

H_1 *Oracle* : The challenger is queried by the adversary for the hash value of the identity $ID_i^{(k)}$ and R_i. If the H_1 *Oracle* was already queried with the same input, the challenger returns the value computed before which is stored in the hash list L_{h1} described below. Otherwise chooses $k_i \in_R \mathbb{Z}^*_{pk}$, computes $h_i = \left(g^{(i)}\right)^{k_i}$, adds the tuple $\langle h_i, ID_i^{(k)}, k_i, R_i \rangle$ to the L_{h1} list.

Similarly, for oracle H_2 the solver $\mathcal{S}$ looks up its corresponding list L_{h2} to see if the hash value corresponding to the query is already listed in the table.

Party corruption: The adversary presents the challenger with an identity $ID_i^{(k)}$ and the challenger should return the private key of that entity. The challenger proceeds as follows:

The challenger chooses $r_i^{(k)} \in_R \mathbb{Z}^*_{pk}$. It then computes $R_i = (g^i)^{r_i^{(k)}}$.

The challenger checks if the H_1 *Oracle* was already queried for $ID_i^{(k)}$ and R_i. If yes, it extracts k_i, h_i from the list L_{h1} and proceeds to the next step. If not queried before, the challenger chooses $k_i \in_R \mathbb{Z}^*_{pk}$, computes $h_i = \left(g^{(i)}\right)^{k_i}$, adds the tuple $\langle h_i, ID_i^{(k)}, k_i, R_i \rangle$ to the L_{h1} list.
The challenger proceeds as per protocol to generate the long term keys.

Lemma 2. *The private key returned by the challenger $\mathcal{S}$ during the party corruption query are consistent with the system and it passes the* ***Key Sanity Check*** *as it is generated as per the protocol only.*

The solver $\mathcal{S}$ now interacts with $\mathcal{A}$ as follows:

- $\mathcal{S}$ takes as input $C = T_{i0}^{(1)}$ and $D = T_{j0}^{(1)} \in \mathbb{G}$. Here we assume that the $params_K$ has the group $\mathbb{G}$ and generator as g. Note that $\mathcal{S}$ does not know the values of $i0^{(1)}$ and $j0^{(1)}$ and the values of $i0^{(1)}$ and $j0^{(1)}$ is already set to a and b respectively, implicitly.
- On a session activation query of the form $(\hat{P_l}, \hat{P_m})$, $\mathcal{S}$ does the following:
  - Choose i_1, i_2 as the ephemeral secrets.
  - Create session identifier $sid' = \Big(\hat{P_l}, \hat{P_m}, \langle R_i, T_i^{(1)}, T_i^{(2)}, V_i^{(1)}, V_i^{(2)}\rangle, \star, \mathcal{I}\Big)$ and
  - Provide $\mathcal{A}$ with $\Big(\hat{P_l}, \hat{P_m}, \langle R_i, T_i^{(1)}, T_i^{(2)}, V_i^{(1)}, V_i^{(2)}\rangle\Big)$.
- On a session activation query of the form, $\Big(\hat{P_m}, \hat{P_l}, \langle R_j, T_j^{(1)}, T_j^{(2)}, V_j^{(1)}, V_j^{(2)}\rangle\Big)$, $\mathcal{S}$ does the following:
  - Choose i_1, i_2 as the ephemeral secrets.
  - Create session identifier $sid' = \Big(\hat{P_l}, \hat{P_m}, \langle R_i, T_i^{(1)}, T_i^{(2)}, V_i^{(1)}, V_i^{(2)}\rangle, \langle R_j, T_j^{(1)}, T_j^{(2)}, V_j^{(1)}, V_j^{(2)}\rangle, \mathcal{R}\Big)$
  - Provide $\mathcal{A}$ with $\Big(\hat{P_l}, \hat{P_m}, \langle R_i, T_i^{(1)}, T_i^{(2)}, V_i^{(1)}, V_i^{(2)}\rangle\Big)$
  - Completes the session $\Big(\hat{P_l}, \hat{P_m}, \langle R_i, T_i^{(1)}, T_i^{(2)}, V_i^{(1)}, V_i^{(2)}\rangle, \langle R_j, T_j^{(1)}, T_j^{(2)}, V_j^{(1)}, V_j^{(2)}\rangle, \mathcal{R}\Big)$.
- On the session activation query $\Big(\hat{P_l}, \hat{P_m}, \langle R_i, T_i^{(1)}, T_i^{(2)}, V_i^{(1)}, V_i^{(2)}\rangle, \langle R_j, T_j^{(1)}, T_j^{(2)}, V_j^{(1)}, V_j^{(2)}\rangle\Big)$, $\mathcal{S}$ does the following:
  - $\mathcal{S}$ updates the session identifier $\Big(\hat{P_l}, \hat{P_m}, \langle R_i, T_i^{(1)}, T_i^{(2)}, V_i^{(1)}, V_i^{(2)}\rangle, \star, \mathcal{I}\Big)$ (if any) to $sid = \Big(\hat{P_l}, \hat{P_m}, \langle R_i, T_i^{(1)}, T_i^{(2)}, V_i^{(1)}, V_i^{(2)}\rangle, \langle R_j, T_j^{(1)}, T_j^{(2)}, V_j^{(1)}, V_j^{(2)}\rangle, \mathcal{I}\Big)$.
  - If the sid' session exists and is already completed, $\mathcal{S}$ sets the sid session key to that of sid'.
  - Else, if a digest query was previously issued on some $\psi = \Big(\sigma, \hat{P_l}, \hat{P_m}, R_i, T_i^{(1)}, T_i^{(2)}, T_j^{(1)}, T_j^{(2)}, K_{\lambda 1}, K_{\lambda 2}, K_{\lambda 3}, K_{\lambda 4},\Big)$ (in this case $\lambda \in \{l, m\}$) $\mathcal{S}$ sets the session key to $H(\psi)$.
  - Else $\mathcal{S}$ chooses K_{ij} from the distribution of session keys and set the sid session key to K_{ij} and updates L_h.
- If $\mathcal{A}$ issues a *Corrupt*, an *EphemeralKeyReveal*, a *SessionKeyReveal*, or an *EstablishParty* query at a party $\mathcal{S}$ answers faithfully.
- At the activation of the t^{th} session at at $ID_{i0}^{(k)}$, if the peer is not $ID_{j0}^{(l)}$, $\mathcal{S}$ aborts; otherwise, it provides $\mathcal{A}$ with $(ID_{i0}^{(k)}, ID_{j0}^{(l)}, T_{i0}^{(1)}, T_{i0}^{(2)}, V_{i0}^{(1)}, V_{i0}^{(2)}, R_{i0})$ (recall that the solver takes as input $T_{i0}^{(1)}$ and $T_{j0}^{(1)}$).

– When $\mathcal{A}$ activates the session matching the t^{th} session at $(ID_{j0}^{(l)}$, $\mathcal{S}$ provides $\mathcal{A}$ with $(ID_{j0}^{(l)}, ID_{i0}^{(k)}, T_{j0}^{(1)}, T_{j0}^{(2)}, V_{j0}^{(1)}, V_{j0}^{(2)}, R_{j0})$.
– In any of the following situations, $\mathcal{S}$ aborts.
  – $\mathcal{A}$ halts with a test session different from the t^{th} session at $ID_{i0}^{(k)}$.
  – $\mathcal{A}$ issues a *SessionKeyReveal* query or an *EphemeralKeyReveal* query on $ID_{i0}^{(k)}$ or $ID_{j0}^{(l)}$.
  – $\mathcal{A}$ issues an *EstablishParty* query on $ID_{i0}^{(k)}$ or $ID_{j0}^{(l)}$.
– If $\mathcal{A}$ halts with a guess, $\mathcal{S}$ outputs $K_{\lambda 3}$ as $CDH(C, D)$. Otherwise $\mathcal{S}$ aborts.

The simulation remains perfect, except with negligible probability; the solver $\mathcal{S}$ guesses correctly the test session with probability $\left((n_{kgc} \cdot n_u)^2 n_s\right)^{-1}$. If $\mathcal{A}$ succeeds under this simulation, and $\mathcal{S}$ guesses correctly the test session, $\mathcal{S}$ outputs $CDH(T_{i0}^{(1)}, T_{j0}^{(1)})$. Hence if $\mathcal{A}$ succeeds with nonnegligible probability in E.1.1, $\mathcal{S}$ outputs with nonnegligible probability $CDH(T_{i0}^{(1)}, T_{j0}^{(1)})$,contradicting the GDH assumption.

Analysis of E.1.2. Suppose that E.1.2 occurs with non-negligible probability, using $\mathcal{A}$ we can build a polynomial time GDH solver $\mathcal{S}$, which succeeds with non-negligible probability. We modify the interaction of event E.1.1 as follows:

– $\mathcal{S}$ takes as input $C = R_{i0}$ and $D = V_{j0}^{(1)} \in \mathbb{G}$. So it doesn't know the values of r_{i0} and s_{j0}. We have set r_{i0} and S_{j0} implicitly as a and b respectively.
– On a session activation query of the form $(\hat{P_m}, \hat{P_l}, \langle R_j, T_j^{(1)}, T_j^{(2)}, V_j^{(1)}, V_j^{(2)} \rangle)$, with $\hat{P_l} = ID_i$ or ID_j, $\mathcal{S}$ does the following:
– On a session activation query of the form, $\left(\hat{P_m}, \hat{P_l}, \langle R_j, T_j^{(1)}, T_j^{(2)}, V_j^{(1)}, V_j^{(2)} \rangle\right)$, $\mathcal{S}$ does the following:
  – Choose i_1, i_2 as the ephemeral secrets.
  – Create session identifier
    $sid' = \left(\hat{P_l}, \hat{P_m}, \langle R_i, T_i^{(1)}, T_i^{(2)}, V_i^{(1)}, V_i^{(2)} \rangle, \langle R_j, T_j^{(1)}, T_j^{(2)}, V_j^{(1)}, V_j^{(2)} \rangle, \mathcal{R}\right)$
  – Provide $\mathcal{A}$ with $\left(\hat{P_l}, \hat{P_m}, \langle R_i, T_i^{(1)}, T_i^{(2)}, V_i^{(1)}, V_i^{(2)} \rangle\right)$
– On the session activation query $\left(\hat{P_l}, \hat{P_m}, \langle R_i, T_i^{(1)}, T_i^{(2)}, V_i^{(1)}, V_i^{(2)} \rangle, \langle R_j, T_j^{(1)}, T_j^{(2)}, V_j^{(1)}, V_j^{(2)} \rangle, \mathcal{R}\right)$, with $\hat{P_l} = ID_i$ or ID_j, $\mathcal{S}$ does the following:
  – $\mathcal{S}$ updates the session identifier $\left(\hat{P_l}, \hat{P_m}, \langle R_i, T_i^{(1)}, T_i^{(2)}, V_i^{(1)}, V_i^{(2)} \rangle, \star, \mathcal{I}\right)$ (if any) to $sid = \left(\hat{P_l}, \hat{P_m}, \langle R_i, T_i^{(1)}, T_i^{(2)}, V_i^{(1)}, V_i^{(2)} \rangle, \langle R_j, T_j^{(1)}, T_j^{(2)}, V_j^{(1)}, V_j^{(2)} \rangle, \mathcal{I}\right)$
  – If the sid' session exists and is already completed, $\mathcal{S}$ sets the sid session key to that of sid'.
  – Else, if a digest query was previously issued on some $\psi = \left(\sigma, \hat{P_l}, \hat{P_m}, R_i, T_i^{(1)}, T_i^{(2)}, T_j^{(1)}, T_j^{(2)}, K_{\lambda 1}, K_{\lambda 2}, K_{\lambda 3}, K_{\lambda 4},\right)$ (in this case

$\lambda \in \{l, m\}$) and if $K_{\lambda 1} = CDH\left(R_i \cdot \left(Y_k{}^{H_1{}^{(k)}(ID_i, R_i)}\right) \cdot T_i^{(1)}, V_j^{(1)} \cdot T_j^{(1)}\right)$ and $K_{\lambda 2} = CDH\left(V_i^{(2)} \cdot T_i^{(2)}, R_j \cdot \left(Y_l{}^{H_1{}^{(l)}(ID_j^{(l)}, R_j)}\right) \cdot T_j^{(2)}\right)$ (using the $\mathcal{DDH}$ oracle), $\mathcal{S}$ sets the session key to $H(\psi)$.

  – Else $\mathcal{S}$ chooses K_{ij} from the distribution of session keys and set the *sid* session key to K_{ij} and updates L_h.

– At $\mathcal{A}$'s digest query on some $\psi = \left(\sigma, \hat{P_l}, \hat{P_m}, R_i, T_i^{(1)}, T_i^{(2)}, T_j^{(1)}, T_j^{(2)}, K_{\lambda 1}, K_{\lambda 2}, K_{\lambda 3}, K_{\lambda 4},\right)$ (in this case $\lambda \in \{l, m\}$) with $\hat{P_l} = ID_i$ or ID_j or with $\hat{P_m} = ID_i$ or ID_j, $\mathcal{S}$ checks whether a value K_{ij} already exists for the queried ψ, then it returns K_{ij} as the completed session key, else

  (i) if there is an already completed session with identifier $sid = \left(\hat{P_l}, \hat{P_m}, \langle R_i, T_i^{(1)}, T_i^{(2)}, V_i^{(1)}, V_i^{(2)} \rangle, \langle R_j, T_j^{(1)}, T_j^{(2)}, V_j^{(1)}, V_j^{(2)} \rangle, \mathcal{I}\right)$or sid', and and if $K_{\lambda 1} = CDH\left(R_i. \left(Y_k{}^{H_1{}^{(k)}(ID_i, R_i)}\right) \cdot T_i^{(1)}, V_j^{(1)} \cdot T_j^{(1)}\right)$ and $K_{\lambda 2} = CDH\left(V_i^{(2)}.T_i^{(2)}, R_j. \left(Y_l{}^{H_1{}^{(l)}(ID_j, R_j)}\right).T_j^{(2)}\right)$, then $\mathcal{S}$ returns the completed session key, else

  (ii) It chooses K_{ij} from the distribution of session keys and set the *sid* session key to K_{ij} and provides $\mathcal{A}$ with K_{ij}.

– At the activation of the t^{th} session at at $ID_{i0}^{(k)}$, if the peer is not $ID_{j0}^{(l)}$, $\mathcal{S}$ aborts; otherwise, it chooses i_1, i_2 as the ephemeral secrets and provide $\mathcal{A}$ with $\left(ID_{i0}^{(k)}, ID_{j0}^{(l)}, T_{i0}^{(1)}, T_{i0}^{(2)}, V_{i0}^{(1)}, V_{i0}^{(2)}, R_i\right)$. Note that R_{i0} is set as C, the input of the CDH problem instance.

– When $\mathcal{A}$ activates the session matching the t^{th} session at $ID_{j0}^{(l)}$, $\mathcal{S}$ chooses j_1, j_2 and provides $\mathcal{A}$ with $(ID_{j0}^{(l)}, ID_{i0}^{(k)}, T_{j0}^{(1)}, T_{j0}^{(2)}, V_{j0}^{(1)}, V_{j0}^{(2)}, R_j)$. Note that $V_{j0}^{(1)}$ is set as D, the input of the CDH problem instance.

– In any of the following situations, $\mathcal{S}$ aborts.
  – $\mathcal{A}$ halts with a test session different from the t^{th} session at $ID_{i0}^{(k)}$.
  – $\mathcal{A}$ issues a *SessionKeyReveal* at $ID_{i0}^{(k)}$ or its matching session.
  – $\mathcal{A}$ issues an *EstablishParty* or *Corrupt* query on $ID_{i0}^{(k)}$ or $ID_{j0}^{(l)}$.

– When $\mathcal{A}$ halts with a guess σ_0 for the test session, $\mathcal{S}$ outputs a guess $CDH(C, D)$ from $K_{\lambda 1}, T_{i1}, T_{j1}$ among other values known and determined by the challenger.

The simulation remains perfect, except with negligible probability. If $\mathcal{A}$ succeeds and $\mathcal{S}$ guesses correctly the test session (this happens with probability $\left((n_{kgc} \cdot n_u)^2 n_s\right)^{-1} \cdot \Pr(\text{E.1.2})$ $\mathcal{S}$ outputs $CDH(C, D)$. Under the GDH assumption and RO model, this cannot happen, except with negligible probability.

Analysis of E.1.3 and E.1.4 Events E.1.3 and E.1.4 are symmetrical to each other. So it suffices to analyze event E.1.3. Suppose that E.1.3 occurs with non-negligible probability, using $\mathcal{A}$ we can build a polynomial time GDH solver $\mathcal{S}$, which succeeds with non-negligible probability. The solver interacts with $\mathcal{A}$ as follows:

- $\mathcal{S}$ takes as input $C = T_{i0}^{(1)}$ and $D = V_{j0}^{(1)} \in \mathbb{G}$. Note that $\mathcal{S}$ does not know the private key of j .
- On a session activation query of the form $\Big(ID_i, ID_j, \langle R_i, T_{i0}^{(1)}, T_{i0}^{(2)}, V_{i0}^{(1)}, V_{i0}^{(2)} \rangle\Big)$, $\mathcal{S}$ does the following:
  - Choose $r_j \in_R \mathbb{Z}^*_{p^{(l)}}$ and compute $R_j = (g^{(l)})^{r_j}$.
  - Create a session with identifier $sid' = \Big(ID_j, ID_i, \langle R_j, T_j^{(1)}, T_j^{(2)}, V_j^{(1)}, V_j^{(2)} \rangle, \langle R_i, T_i^{(1)}, T_i^{(2)}, V_i^{(1)}, V_i^{(2)} \rangle, \mathcal{R}\Big)$ and provide $\mathcal{A}$ with $\Big(ID_j, ID_i, \langle R_j, T_j^{(1)}, T_j^{(2)}, V_j^{(1)}, V_j^{(2)} \rangle\Big)$.
  - Choose K_{ij} from the distribution of session keys and set the session key to K_{ij}.
- On the query $\Big(ID_j, ID_i, \langle R_j, T_j^{(1)}, T_j^{(2)}, V_j^{(1)}, V_j^{(2)} \rangle, \langle R_i, T_i^{(1)}, T_i^{(2)}, V_i^{(1)}, V_i^{(2)} \rangle\Big)$, $\mathcal{S}$ does the following:
  - Update the session identifier $\Big(ID_j, ID_i, \langle R_j, T_j^{(1)}, T_j^{(2)}, V_j^{(1)}, V_j^{(2)} \rangle, \star, \mathcal{I}\Big)$ (if any) to $\Big(ID_j, ID_i, \langle R_j, T_j^{(1)}, T_j^{(2)}, V_j^{(1)}, V_j^{(2)} \rangle, \langle R_i, T_i^{(1)}, T_i^{(2)}, V_i^{(1)}, V_i^{(2)} \rangle, \mathcal{I}\Big)$.
  - If a value is already assigned to the *sid'* session key, set the *sid* session key to that of *sid'*. Else, if a digest query was previously issued on some $\psi = \Big(\sigma, \hat{P}_l, \hat{P_m}, R_i, T_i^{(1)}, T_i^{(2)}, T_j^{(1)}, T_j^{(2)}, K_{\lambda 1}, K_{\lambda 2}, K_{\lambda 3}, K_{\lambda 4},\Big)$ (in this case $\lambda \in \{l, m\}$) and if $K_{\lambda 1} = CDH\Big(R_i \cdot \Big({Y_k}^{{H_1}^{(k)}(ID_i, R_i)}\Big) \cdot T_i^{(1)}, V_j^{(1)} \cdot T_j^{(1)}\Big)$ and $K_{\lambda 2} = CDH\Big(V_{i2} \cdot T_i^{(2)}, R_j \cdot \Big({Y_l}^{{H_1}^{(l)}(ID_j, R_j)}\Big) \cdot T_j^{(2)}\Big)$, then set the *sid* session key to $H(\psi)$.
  - Else, K_{ij} from the distribution of session keys and set the *sid* session key to K_{ij}.
- At $\mathcal{A}$'s digest query on some $\psi = \Big(\sigma, \hat{P}_l, \hat{P_m}, R_i, T_i^{(1)}, T_i^{(2)}, T_j^{(1)}, T_j^{(2)}, K_{\lambda 1}, K_{\lambda 2}, K_{\lambda 3}, K_{\lambda 4},\Big)$ (in this case $\lambda \in \{l, m\}$) with $P_l = ID_j$ or $P_m = ID_i$, $\mathcal{S}$ checks if the same query was made previously, if so it returns the previously returned value. Else it follows the steps as mentioned above.
- When $\mathcal{A}$ activates the t-th session at $ID_{i0}^{(k)}$, if the peer is not $ID_{j0}^{(l)}$, $\mathcal{S}$ aborts; otherwise, it provides $\mathcal{A}$ with $\Big(ID_i, ID_j, \langle R_i, T_i^{(1)}, T_i^{(2)}, V_i^{(1)}, V_i^{(2)} \rangle\Big)$ (recall that the solver takes as input $T_{i0}^{(1)}$ and $V_{j0}^{(1)}$).
- When $\mathcal{A}$ activates the session matching the t-th session at $ID_{i0}^{(k)}$, $\mathcal{S}$ chooses $r_j \in_R \mathbb{Z}^*_{p^{(l)}}$ if not chosen before and provides $\mathcal{A}$ with $\Big(ID_j, ID_i, \langle R_j, T_j^{(1)}, T_j^{(2)}, V_j^{(1)}, V_j^{(2)} \rangle\Big)$.

- If $\mathcal{A}$ issues an *EphemeralKeyReveal* query on the session matching the t-th session at $ID_{i0}^{(k)}$, $\mathcal{S}$ answers faithfully.
- In any of the following situations, $\mathcal{S}$ aborts.
  - $\mathcal{A}$ halts with a test session different from the t-th session at $ID_{i0}^{(k)}$.
  - $\mathcal{A}$ issues a $Corrupt(ID_{j0}^{(l)})$ query or an *EstablishParty* query on $ID_{i0}^{(k)}$ or $ID_{j0}^{(l)}$.
  - $\mathcal{A}$ issues an *EphemeralKeyReveal* query on the t-th session at $ID_{i0}^{(k)}$.
- If $\mathcal{A}$ halts with a guess, $\mathcal{S}$ produces $\left(K_{i1} \cdot \left(T_{i0}^{(1)} \cdot R_i \cdot (y^{(i)})^{H_1{}^{(k)}(ID_i,R_i)}\right)^{-j_1} \cdot \left(V_{j0}^{(1)}\right)^{-s_i}\right)$ as $CDH(C,D)$.

The simulation remains perfect, except with negligible probability; the solver $\mathcal{S}$ guesses correctly the test session with probability $\left((n_{kgc} \cdot n_u)^2 \, n_s\right)^{-1}$. If $\mathcal{A}$ succeeds under this simulation, and $\mathcal{S}$ guesses correctly the test session, $\mathcal{S}$ outputs $CDH(C,D)$.

So from the analysis none of the events E.1.1, E.1.2, E.1.3,E.1.4 can occur with non-negligible probability,and hence we conclude that event E.1.1 cannot occur (except with negligible probability).

Analysis of E.2. Event E.2 can be further analyzed as follows:

- E.2.1: E.2 $\wedge$ $\mathcal{A}$ issues a *Corrupt* query on ID_i and
- E.2.2: E.2 $\wedge$ $\mathcal{A}$ issues a *EphemeralKeyReveal* query on the test session.

These are the strongest query that $\mathcal{A}$ can issue in event E.2.

Event E.2.1: To show that E.2.1 cannot happen with non-negligible probability, the simulation proceeds similar to the analysis of E.1.3. $\mathcal{S}$ takes as input $T_{i0}^{(1)}$ and $V_{j0}^{(1)} \in_R \mathbb{G}$ (for $ID_i^{(k)}$, $\mathcal{S}$ chooses $k_i \in_R \mathbb{Z}_{pk}^*$, computes $h_i = \left(g^{(i)}\right)^{k_i}$). The simulation environment remains perfect except with negligible probability. If $\mathcal{A}$ succeeds with a forgery and $\mathcal{S}$ guesses correctly the test session ($\mathcal{S}$ guesses correctly the test session with probability $\left((n_{kgc} \cdot n_u)^2 \, n_s\right)^{-1}$), $\mathcal{S}$ outputs $CDH(C,D)$ which contradicts the GDH assumption.

Event E.2.2: We modify the analysis of event E.1.2 by aborting when $\mathcal{A}$ activates a session matching the t^{th} session at $ID_{i0}^{(k)}$. The simulation environment remains perfect except with negligible probability. If $\mathcal{A}$ succeeds and $\mathcal{S}$ guesses correctly the test session then $\mathcal{S}$ outputs $CDH(R_{i0}, V_{j0}^{(1)})$. $\mathcal{S}$ is polynomial, and if E.2.2 occurs with non-negligible probability, $\mathcal{S}$ yields a polynomial time CDH solver, which succeeds with non-negligible probability; contradicting the GDH assumption.

8 Conclusion

Traditional notions of key agreement in the identity based paradigm is constrained by a single PKG with an upward ceiling on the number of users it can serve. The issue has larger ramifications in the context of organizations which are situated in diverse locations with large number of users. The same issue plagues the key agreement for communication in VANET. The solution to overcome the constraint is to embrace the idea of multiple PKG. We have proposed an efficient, pairing-free, single round identity based key agreement protocol (IDKA) which supports communication between parties registered with distinct PKGs. The protocol offers scalability and enhanced utility in several applications including in Vehicular Networks. We have also proposed a new security model based on the eCK model exclusively for multiple PKGs. Our protocol also provides security against active adversary. We show that our protocol is secure in our new security model under the Gap Diffie-Hellman complexity assumption in the RO model. We leave open the problem of constructing a single round IDKA protocol in multiple PKG scenario in standard model.

References

[BR94] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *Advances in CryptologyCRYPTO93*, pages 232–249. Springer, 1994.

[BWJM97] Simon Blake-Wilson, Don Johnson, and Alfred Menezes. *Key agreement protocols and their security analysis*. Springer, 1997.

[CCS07] Liqun Chen, Zhaohui Cheng, and Nigel P Smart. Identity-based key agreement protocols from pairings. *International Journal of Information Security*, 6(4):213–241, 2007.

[CK01] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *Advances in CryptologyEUROCRYPT 2001*, pages 453–474. Springer, 2001.

[CK03] Liqun Chen and Caroline Kudla. Identity based authenticated key agreement protocols from pairings. In *Computer Security Foundations Workshop, 2003. Proceedings. 16th IEEE*, pages 219–233. IEEE, 2003.

[CM09] Qingfeng Cheng and Chuangui Ma. Ephemeral key compromise attack on the ib-ka protocol. *IACR Cryptology ePrint Archive*, 2009:568, 2009.

[DH76] Whitfield Diffie and Martin E Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, 1976.

[DKPS13] Shlomi Dolev, Lukasz Krzywiecki, Nisha Panwar, and Michael Segal. Certificating vehicle public key with vehicle attributes a (periodical) licensing routine, against man-in-the-middle attacks and beyond. In *SAFECOMP 2013-Workshop ASCoMS (Architecting Safety in Collaborative Mobile Systems) of the 32nd International Conference on Computer Safety, Reliability and Security*, 2013.

[FAA14] Mohammad Sabzinejad Farash and Mahmoud Ahmadian-Attari. A pairing-free id-based key agreement protocol with different pkgs. *IJ Network Security*, 16(2):143–148, 2014.

[FG10] Dario Fiore and Rosario Gennaro. Making the diffie-hellman protocol identity-based. In *Topics in Cryptology-CT-RSA 2010*, pages 165–178. Springer, 2010.

[Gün90] Christoph G Günther. An identity-based key-exchange protocol. In *Advances in CryptologyEurocrypt89*, pages 29–37. Springer, 1990.

[LKKO05] Hoonjung Lee, Donghyun Kim, Sangjin Kim, and Heekuck Oh. Identity-based key agreement protocols in a multiple pkg environment. In *Computational Science and Its Applications–ICCSA 2005*, pages 877–886. Springer, 2005.

[LLM07] Brian LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In *Provable Security*, pages 1–16. Springer, 2007.

[MB05] Noel McCullagh and Paulo SLM Barreto. A new two-party identity-based authenticated key agreement. In *Topics in Cryptology–CT-RSA 2005*, pages 262–274. Springer, 2005.

[MM13] Dheerendra Mishra and Sourav Mukhopadhyay. Cryptanalysis of pairing-free identity-based authenticated key agreement protocols. In *Information Systems Security*, pages 247–254. Springer, 2013.

[Sae00] S. Saeednia. Improvement of gunther's identity-based key exchange protocol. *Electronics Letters*, 36(18):1535–1536, 2000.

[VSVR13] S Sree Vivek, S Sharmila Deva Selvi, Layamrudhaa Renganathan Venkatesan, and C Pandu Rangan. Efficient, pairing-free, authenticated identity based key agreement in a single round. In *Provable Security*, pages 38–58. Springer, 2013.

[VYK13] Thokozani Felix Vallent, Eun-Jun Yoon, and Hyunsung Kim. An escrow-free two-party identity-based key agreement protocol without using pairings for distinct pkgs. *IEIE Transactions on Smart Processing & Computing*, 2(3):168–175, 2013.