Pair-wise Cryptographic Models for Secure Data Exchange in P2P Database Management Systems

Sk. Md. Mizanur Rahman  
SITE, uOttawa, Ottawa, ON, K1N6N5 Canada  
mrahman@site.uottawa.ca

Mehedi Masud  
SITE, uOttawa, Ottawa, ON, K1N6N5 Canada  
mmasud@site.uottawa.ca

Carlisle Adams  
SITE, uOttawa, Ottawa, ON, K1N6N5 Canada  
cadams@site.uottawa.ca

Khalil El-Khatib  
FBIT, UOIT, Oshawa, ON, L1H7K4 Canada  
khalil.el-khatib@uoit.ca

Hussein Mouftah  
SITE, uOttawa, Ottawa, ON, K1N6N5 Canada  
mouftah@site.uottawa.ca

Eiji Okamoto  
SIE, University of Tsukuba, Tsukuba, 305-8573 Japan  
okamoto@risk.tsukuba.ac.jp

ABSTRACT

A peer-to-peer database management system (P2PDBMS) is a collection of autonomous data sources, called peers. In this system each peer augments a conventional database management system with an inter-operability layer (i.e. mappings/policies) for sharing data and services. Peers exchange data in a pair-wise fashion on-the-fly in response to a query without any centralized control. Generally, the communication link between two peers is insecure and peers create a temporary session while exchanging data. When peers exchange highly confidential data between them over an insecure communication network, such as the Internet, the data might be trapped and disclosed by the intruders. In a P2PDBMS there is no centralized control for data exchange, hence we cannot assume any central third party security infrastructure (e.g. PKI) to protect confidential data. So far, there is currently no available/existing security protocol for secured data exchange in P2PDBMS. In this paper we propose three models for secure data exchange in P2PDBMSs and the corresponding security protocols. The proposed protocol allows the peers to compute their secret session keys dynamically during data exchange based on the policies between them. Our proposed protocol is robust against the man-in-the-middle attack, the masquerade attack, and the reply attack.

Keywords

P2PDBMS Security Model, Secure Data Exchange, Pairing-based cryptography, Authentication

1. INTRODUCTION

A peer-to-peer database management systems (P2PDBMS) is a collection of autonomous data sources, called peers, where each peer augments a conventional database management system with an inter-operability layer (i.e. mappings) for sharing data and services. The local databases on peers are called peer databases. In a P2PDBMS, each peer chooses its own database schema and maintains data independently. Although peer databases are created independently, data in one peer may semantically relate with data in another peer. Therefore, each peer specifies pair-wise mappings with other peers to share and exchange related data.

Contrary to the traditional data integration systems where a global mediated schema is required for data exchange, in P2PDBMS, semantic relationships exist between peers, or among a small set of peers, for exchanging data. The data are accessed globally by any peer by traversing the network of peers. In the last few years, significant progress has been made in research on various issues related to P2PDBMSs, such as peer data exchange settings [1], data integration models [3], mediation methods [4], coordination mechanisms [5, 6], and mappings [7] among the peer databases.

There is an increasing interest in the creation of peer-to-peer database management systems, which includes establishing and maintaining mappings between peers, processing queries using appropriate propagation techniques, and exchanging data between peers. While there is a rich body of research concerning frameworks and mapping issues among peers, the aspect of sharing data between trusted or acquainted peers in a secured way is given less attention. In many collaborative data sharing efforts, particularly in biological and health sciences, confidential data between sources are exchanged for sharing and coordinating information with each other. Generally, in collaborative data sharing, independent researchers or groups with different goals, schemas, and data agree to share data with one another. Each group independently curates, revises, and extends this shared data. At some point sources need to exchange data which may be confidential (e.g. new research results on genes in a biological system; patients’ personal information to a health information network) by establishing a temporary data exchange session. In a peer-to-peer system, we cannot assume a fixed secure channel for data exchange between each pair of peers since peers are dynamic and may leave the network anytime, or acquaintances between peers are temporary. Moreover, it would be highly expensive and not feasible to maintain a secure link
over an insecure network such as the internet.

about patients’ treatments, medications, and test results

macists, and other stakeholders) are willing to share data

cians; walk-in clinics; hospitals; medical laboratories; phar-

mation network’, where different parties (e.g. family physi-

peration of IBC presented in [14], has only become truly practical

with the advent of Pairing-Based Cryptography (PBC) [16, 15]. In most circumstances the points on an elliptic curve

form a simple cyclic group. Therefore, it is suitable to im-

plement pairing-based cryptography on such elliptic curves.

In order to achieve secured data exchange in a P2PDBMS
dynamic network, this paper presents a protocol based on
Identity Based Cryptography (IBC) [14] is a type of public-
key cryptography where a single piece of information that
uniquely identifies a user (e.g. IP or email address) can be
used both to exchange keys and to encrypt data. The no-
tion of IBC presented in [14], has only become truly practical
with the advent of Pairing-Based Cryptography (PBC) [16, 15]. In most circumstances the points on an elliptic curve
form a simple cyclic group. Therefore, it is suitable to im-
plement pairing-based cryptography on such elliptic curves.

In brief, our protocol has the following properties:

(1) flexible message-oriented secure data exchange between
peers (2) exchange of data between peers without any third
party certificates (3) communication between peers could
be as simple as a single TCP connection (4) both parties
(i.e. source and target) authenticate each other during data
exchange.

Organization of the paper: The next section introduces
the primitives of cryptography that is necessary to describe
our proposed protocol. Section 3 describes how the secure
data exchange policy/mapping is established between two
peers and the threats that can occur when peers exchange
their data in an unsecured channel. In Section 4, the pa-
er presents our cryptography solution and describes the
proposed protocol for exchanging data between peers. In
section 5 we discuss issues of cryptographic implementation
and prevention of different attacks in our secure data ex-
change protocol. Section 6 describes related work, and fi-
nally Section 7 concludes and points out avenues for further
research.

2. CRYPTOGRAPHIC PRIMITIVES

In this section, we describe some basic cryptographic prim-
ities and mathematical properties which are useful to un-

Figure 1: An example model of a peer to peer database
management system

for each pair of peers. When data are exchanged through
an unsecured link between acquainted peers, data are no
longer secured despite the assumption that each source pro-
tects its own data from malicious tampering and accessing
by external intruders.

The following example illustrates the needs to use a pairing-
based dynamic security policy for exchanging confidential
data between peers. This scenario relates to a ‘health infor-
mation network’, where different parties (e.g. family physi-
cians; walk-in clinics; hospitals; medical laboratories; phar-
macists, and other stakeholders) are willing to share data
about patients’ treatments, medications, and test results
over an insecure network such as the internet.

EXAMPLE 1. Consider the scenario of a P2PDBMS in
Figure 1. The figure illustrates a collaborative ‘health in-
formation network’. In the system, family doctors (FDDB),
hospitals (HDB), medical laboratories (LABDB), pharma-
cists (PHDB), and other stakeholders (e.g. medical research

cells (RDB)) are willing to exchange or coordinate infor-
mation about patients’ treatments, medications, test results,
and diseases. In the system, data in a database of a peer
may need to be exchanged with other related peers accord-
ing to established policies between them. For example, fam-
ily doctors may want to keep track of patients’ medications
for some specific diseases. Therefore, family doctors should
have a link with the pharmacist database (PHDB) and any
patient in PHDB diagnosed with a disease that is of inter-
est to family doctors may need to be exchanged with PDDB.
Moreover, family doctors may be interested in collecting test
results of their patients from laboratories and the medica-
tions that their patients take while staying at hospitals. The
links from HDB and PHDB to RDB show that research cell
database (RDB) is interested in information about certain
diseases for research purposes. The links between peers in
the figure are formally a set of mappings or mapping con-
straints. For example, \( m_{12} \) represents mappings from peer
\( P_1 \) to \( P_2 \).

Note that the acquaintances between peers are established
with predefined policies and trust relationships without hav-
ing a centralized security policy. The existing conventional
public key Infrastructure (PKI) is not suitable to apply
in P2PDBMS since a centralized-trusted control system is
needed for the PKI. Recent progress of Elliptic Curve Cryp-
tography (ECC) shows that it is feasible to implement ECC.
Studies have shown that ECC consumes considerably less
resources than conventional public key cryptography (PKC)
for a given security level [10, 11, 12].

However, in order to effectively use ECC, it is necessary
to authenticate the public keys; otherwise, the network will
be vulnerable to man-in-the-middle attacks. Public key au-
thentication requires a Public Key Infrastructure to issue
and revoke certificates and it requires users to store, ex-
change, and verify these certificates [13].

Identity-Based Cryptography (IBC) [14] is a type of public-
key cryptography where a single piece of information that
uniquely identifies a user (e.g. IP or email address) can be
used both to exchange keys and to encrypt data. The no-
tion of IBC presented in [14], has only become truly practical
with the advent of Pairing-Based Cryptography (PBC) [16, 15]. In most circumstances the points on an elliptic curve
form a simple cyclic group. Therefore, it is suitable to im-
plement pairing-based cryptography on such elliptic curves.

In order to achieve secured data exchange in a P2PDBMS
dynamic network, this paper presents a protocol based on
Identity Based Encryption (IBE) and pairing-based cryptog-
ography. Using pairing-based and IBE properties, each peer
in the network generates a dynamic secret session key based
on the attributes mentioned in the query and the predefined
data exchange policy. In this protocol, peers authenticate
each other in a pair-wise fashion without a centralized au-
thentication policy.

In brief, our protocol has the following properties:

(1) flexible message-oriented secure data exchange between
peers (2) exchange of data between peers without any third
party certificates (3) communication between peers could
be as simple as a single TCP connection (4) both parties
(i.e. source and target) authenticate each other during data
exchange.

Organization of the paper: The next section introduces
the primitives of cryptography that is necessary to describe
our proposed protocol. Section 3 describes how the secure
data exchange policy/mapping is established between two
peers and the threats that can occur when peers exchange
their data in an unsecured channel. In Section 4, the pa-
er presents our cryptography solution and describes the
proposed protocol for exchanging data between peers. In
section 5 we discuss issues of cryptographic implementation
and prevention of different attacks in our secure data ex-
change protocol. Section 6 describes related work, and fi-
nally Section 7 concludes and points out avenues for further
research.

2. CRYPTOGRAPHIC PRIMITIVES

In this section, we describe some basic cryptographic prim-
ities and mathematical properties which are useful to un-
understand our proposed protocol.

2.1 Elliptic curves
Elliptic curves are considered interesting primarily as an alternative group structure. In regard to implementation of common cryptographic protocols, certain advantages come with the elliptic curve families, \( E(F_q) \): \( y^2 = x^3 + Ax + B \) [20]. The main advantage is that much smaller keys can be used, as there is no known polynomial-time algorithm for the discrete logarithm (DL) problem for the great majority of such curves. Given a point \( P \) on a curve \( E \) defined over a finite field \( F_q \), where \( q = p^m \), and \( p \) is a large prime, the problem is to determine “\( a \)” for given “\( aP \)”. In most circumstances the points on such a curve form a simple cyclic group.

At the foundation of many cryptosystems is a hard mathematical problem that is computationally infeasible to solve. The DL problem is the basis for the security of many cryptosystems, including the elliptic curve cryptography. More specifically, the ECC relies upon the difficulty of the elliptic curve discrete logarithm problem (ECDLP) [21].

2.2 Bilinear maps
Let \( G_1 \) be an additive group and \( G_2 \) be a multiplicative group of the same prime order \( q \). Let \( P \) be an arbitrary generator of \( G_1 \). Note that \( aP \) denotes \( P \) added to itself \( a \) times. Assume that the discrete logarithm (DL) problem is hard in both \( G_1 \) and \( G_2 \). We can think of \( G_1 \) as a group of points on an elliptic curve over \( F_q \), and \( G_2 \) as a subgroup of the multiplicative group of a finite field \( F_q \), for some \( k \in Z_q^* \), where \( Z_q^* = \{ \xi \in \xi \leq q-1 \} \). A mapping \( e : G_1 \times G_1 \rightarrow G_2 \), satisfying the following properties, is called a cryptographic bilinear map.

- **Bilinearity:** \( e(aP,bQ) = e(P,Q)^{ab} \) for all \( P,Q \in G_1 \) and \( a,b \in Z_q^* \). This can be restated in the following way. For \( P, Q, R \in G_1 \), \( e(P+Q,R) = e(P,R)e(Q,R) \) and \( e(P,Q+R) = e(P,Q)e(P,R) \).
- **Non-degeneracy:** If \( P \) is a generator of \( G_1 \), then \( e(P,P) \) is a generator of \( G_2 \). In other words, \( e(P,P) \neq 1 \).
- **Computable:** A mapping is efficiently computable if \( e(P,Q) \) can be computed in polynomial-time for all \( P,Q \in G_1 \).

Modified Weil Pairing [15] and Tate Pairing [16] are examples of cryptographic bilinear maps.

2.3 Diffie-Hellman problems
In this section, we recall the properties of Diffie-Hellman [17] gap families. Before defining the Diffie-Hellman gap families, we assume the following:

- \( P \) is a point on an elliptic curve \( E \) given by \( Y^2 = X^3 + aX + \beta \mod T \) where \( T \) is a prime number.
- \( \langle P \rangle \) is a subgroup of \( E \) generated by \( P \).
- \( | \langle P \rangle | = q \).

The group \( G_1 \) represents the group of points on the elliptic curve \( E \). Using the group \( G_1 \), we can define the following hard cryptographic problems applicable to our proposed protocol.

- **Computational Diffie-Hellman (CDH) Problem:** Given a triple \( (aP,bP) \in G_1 \) for \( a,b \in Z_q^* \), find the element \( abP \in E \).
- **Decision Diffie-Hellman (DDH) problem:** Given a quadruple \( (P,aP,bP,cP) \in G_1 \) for \( a,b,c \in Z_q^* \), decide whether \( c = ab \mod q \) or not.
- **Gap Diffie-Hellman (GDH) Problem:** A class of problems where the CDH problem is hard but DDH problem is easy.
- **Bilinear Diffie-Hellman (BDH) Problem:** Given a quadruple \( (P,aP,bP,cP) \in G_1 \) for some \( a,b,c \in Z_q^* \), compute \( e(P,P)^{abc} \).

Groups where the CDH problem is hard but DDH problem is easy are called GAP Diffie-Hellman (GDH) groups. Details about GDH groups can be found in [15, 17].

3. SECURE DATA EXCHANGE SETTINGS
In this section first we introduce the concept of data exchange settings in a P2PDBMS. We then discuss different security threats that may occur while two peers exchange data through an insecure channel.

Attributes are symbols taken from a given finite set \( U = \{ A_1, \cdots, A_k \} \) called the universe. Each attribute \( A_i \) is associated with a finite set of values called the domain of \( A_i \) and is denoted by \( dom(A_i) \). Suppose \( X = \{ A_1, A_2, \cdots, A_k \} \subseteq U \), with the elements \( A_i(1 \leq i \leq k) \) taken in the order shown, then \( dom(X) \subseteq dom(A_1) \times dom(A_2) \times \cdots \times dom(A_k) \). A non-empty subset of \( U \) is called a relation schema \( R \). A database schema is a finite collection \( R = (R_1, \cdots, R_m) \) of relation schemas.

The authors in [1] introduced a setting of data exchange between peers. In a data exchange setting, one peer is a source peer (data provider), an "authoritative" or "trusted" peer that can contribute new data, while the other peer, called the target peer (data receiver) accepts data from the source. The ultimate goal of a data exchange setting is to exchange data from a source to a target according to the mappings between the source and the target. Let \( S \) be a schema at a peer \( P_i \) and \( T \) be a schema at another peer \( P_j \). If mappings are specified from \( S \) to \( T \), then \( S \) is called a source schema and \( T \) is called a target schema. Generally, in data exchange settings mappings are constituted by a set of assertions of the forms

\[
\Sigma_{st} = q_S \rightarrow q_T
\]

where \( q_S \) and \( q_T \) are two formulas, respectively over the source schema \( S \), and over the target schema \( T \). Intuitively,
an assertion $q_s \rightarrow q_T$ specifies that the concept represented by the formula $q_s$ over the source schema corresponds to the concept in the target schema represented by the formula $q_T$. The assertions are basically tuple-generating dependencies [2]. An example of assertions can be specified as logical expressions of the form:

$$\forall x \exists w \phi(x, w) \rightarrow \exists y \psi(x, y)$$

where the left hand side (LHS) of the implication, $\phi$, is a conjunction of relation atoms over the schema of $S$ and the right hand side (RHS) of the implication, $\psi$, is a conjunction of relation atoms over the schema $T$. The mapping expresses a constraint about the appearance of a tuple in the instance satisfying the constraint of the RHS, given a particular combination of tuples satisfying the constraint of the LHS.

Basically, mappings perform structural relationship of data between source and target as well as data to be exchanged from source to target. Through the mappings, a source also exports part of its schema accessible to the target. The following is a simple example of a data exchange setting. 

**Example 2.** Consider a family physician database (FDB) with the schema $S$ consisting of two relations $R_1$(OHIP, Name, Address, Illness, DOB) and $R_2$(OHIP, TestName, Result, Date). Also consider a database in a research cell database (RDB) with the schema $T$ consisting of a relation $R_3$(OHIP, Name, Illness, DOB, TestName, Result). Assume that a mapping is established between $S$ and $T$ as follows:

$$\forall \text{ohip, name, dob, testname, result} \exists \text{name, address, illness, dob, result} R_1(\text{ohip, name, address, illness, dob, testname, result}) \rightarrow R_3(\text{ohip, illness, dob, testname, result})$$

The mapping expresses that patient data (ohip, name, dob, testname, result) are exchanged from FDB to RDB by a query request $q$ from FDB. It also shows that FDB shares the attributes \{OHIP, Illness, DOB, TestName, Result\} with RDB. Although, the attributes are shared for RDB they also contain some confidential attributes e.g. \{OHIP, DOB\}. Hence, during data exchange the data of the confidential attributes should not be exposed to others by any means. We can say that these attributes are more confidential compared to the attributes \{TestName, Result\}, since the values of these attributes do not have any meaning unless one knows corresponding ohip and date of birth. Note that only the source knows which attributes are confidential attributes among the shared attributes. The administrator of the source is responsible to distinguish shared and confidential attributes. Note that in this paper we only consider the schema-level mappings between a source and a target. We assume that when the mappings are created only the source and the corresponding target know the structural relationship between their schemas (i.e., correspondences between the attributes and relations). The structural relationship is not known to other peers.

We now formally define shared attributes, confidential attributes, Non-confidential attributes, and private attributes.

**Definition 1 (Shared attributes).** Consider two peers $P_i$ and $P_j$ in a P2PDBMS. Let $S$ be a schema with a set of attributes $U_s$ in $P_i$ and $T$ be a schema with a set of attributes $U_t$ in $P_j$. Assume a mapping $\Sigma_{st} = q_s \rightarrow q_T$ between $P_i$ and $P_j$. Let $\text{att}(/Sigma_st/)$ denote the set of attributes exposed by $P_i$ using the mapping $\Sigma_{st}$. Therefore, the shared attributes, denoted by $SA$, is $SA \subseteq U_s = \text{att}(/Sigma_st/)$. 

**Definition 2 (Confidential attributes).** Consider the mapping $\Sigma_{st} = q_s \rightarrow q_T$ between two peers $P_i$ and $P_j$. Let $SA$ be the set of shared attributes. Therefore, the confidential attributes, denoted by $CA$, are $CA \subseteq SA$.

**Definition 3 (Non-confidential attributes).** Considering the mapping $\Sigma_{st} = q_s \rightarrow q_T$ between two peers $P_i$ and $P_j$, where $SA$ is the set of shared attributes, the private attributes, denoted by $PA$, is $U_s \setminus SA$.

**Definition 4 (Private attributes).** Considering the mapping $\Sigma_{st} = q_s \rightarrow q_T$ between two peers $P_i$ and $P_j$, where $SA$ is the set of shared attributes, the private attributes, denoted by $PA$, is $U_s \setminus SA$.

**Example 3.** Consider the example 2. Based on the mappings, we see that the shared attributes are \{OHIP, Illness, DOB, TestName, Result\}, the confidential attributes are \{OHIP, DOB\}, and the non-confidential attributes are \{Illness, TestName, Result\}. Note that administrators of the peers define the attributes that are confidential implicitly during the creation of mappings.

We now describe a scenario to justify the need of a protocol that secures confidential information of shared attributes during exchange of data in an unsecured channel.

Assume that a user at RDB submits the following query $q$. 

**SELECT ohip, name, dob, illness, result**

**FROM R_{1}**

**WHERE testname=“whitebloodcount”**

Since RDB is connected with FDB, the query is forwarded to RDB after transformation [9, 8] with respect to the schema of FDB. Suppose the transformed query for FDB is as follows:

**SELECT ohip, name, dob, illness, result**

**FROM R_{1}, R_{2}**

**WHERE (R_{1}.ohip=R_{2}.ohip) and (testname=“whitebloodcount”)**

When the query is received by FDB it realizes that the target is requesting some confidential data, for example \{ohip, dob\}. It is now the responsibility of FDB to provide the requested data in a secured way because FDB is the “trusted” or “authoritative” source according the data exchange setting. We observe that there are several security
threats that can occur during data exchange from a source to a target.

In the next subsection we discuss these threats.

### 3.1 Security threats for data exchange in P2PDBMSs

#### Man-in-the-middle attack (MITM): In MITM, intruders can make independent connections with the source and the target and relay messages between them, making them believe that they are exchanging data directly with each other over a private connection (e.g., TCP connection). In fact, the entire data exchange session is controlled by the intruder. Once the TCP connection is intercepted, the intruder acts as a proxy. Thus, the intruder becomes another node on the communication channel and is able to read, insert, and modify the data in the intercepted communication. The scenario is shown in Figure 2.

#### Masquerade attack: In this attack, a malicious peer may pretend to be a valid target of a source by stealing the identity of the real target. Thus, a malicious peer may gain access to the data of the source. The easiest point of entry for a masquerade peer is provided by a weak authentication between the source and the target. Once the malicious node passes the authentication process, it may be authorized by the source as a target to access its data. Similarly, a malicious node may falsely act as a target for a request. Therefore, a malicious node may also be able to tamper with both exchanged data and the mappings.

#### Replay attack: A replay attack is an active network attack in which a valid data transmission is maliciously or fraudulently repeated or delayed. Suppose Alice is a target who wants to authenticate her identity to a source, Bob. For valid identification of Alice, Bob requests her password as a proof of identity, which Alice provides to Bob (possibly after some transformation using a hash function). Meanwhile, an intruder, Eve, is eavesdropping on the conversation and is recording the password. After the verification phase is over, Eve connects to Bob as Alice. Now, if Bob asks Alice for proof of identity, Eve sends Alice’s password that was recorded in the verification phase.

### 3.2 Our contribution

In a P2PDBMS, a peer may act as a source and/or a target. For secure data exchange, source and target peers are responsible for generating the secret session key for a specific data exchange session. For exchanging data from a source peer $P_i$ to a target peer $P_j$, source-to-target data exchange mappings are constituted. Thus if the target $P_j$ requests data from the source $P_i$, by a query then the source provides data to the target depending on the query request. In order to provide data for the query request on-the-fly, a security mechanism is needed between the source $P_i$ and the target $P_j$. Since there is no pre-existing security mechanism between peers, there may be an attack on the data exchange session (see Section 3.1). Based on different trust relationships between a source and a target we define three secured data exchange models. The models are as follows:

#### Model 1: In this model the source and the target fully trust each other and the target peer explicitly knows which are the confidential and non-confidential attributes that are defined by the source through the mapping. This model is applicable if the source and the target mutually agree about which attributes are to be used to generate the session key for a query request. Hence when the target requests data through a query the source can identify which attributes are used to compute the session key for data exchange.

#### Model 2: In this model the source and the target fully trust each other but the target peer is not aware of the confidential and non-confidential attributes defined by the source. This model is particularly designed to make the source, the only authoritative peer to initiate the process for generating the session key in response to the query received from the target. Hence, when a source receives a query from a target the source alone dynamically selects the confidential and non-confidential attributes for generating the session key. However, the target is later informed about the attributes that are used to generate the session key by the source through some exchange of messages. In Section 4.3, we describe the process how the target is informed the confidential and non-confidential attributes that are used to generate the session key.

#### Model 3: In this model the source and the target may not fully trust each other and a target is unaware of the confidential and non-confidential attributes. This model is explicitly designed to prevent a situation when there could be some sort of social engineering of the exchanged data from the target as well as from the source. In order to prevent social engineering, a source has to consider some private parameters from the target and the target has to consider some private parameters from the source during exchange of data. These parameters are generated dynamically without any previous agreement. Hence, if the source or the target performs a social engineering attack during data exchange it is identified. The query result is encrypted by the source in such a way that only the valid target can decipher the query result. The target deciphers the query result by using some defined operations with its private parameters.

In this paper, we propose three secure data exchange models in a data exchange setting and the corresponding security protocols. Our proposed protocols are based on the one way cryptographic hash function and the cryptographic hard properties of IBE and pairing over elliptic curves. In the pro-
4. DESCRIPTION OF THE PROPOSED PROTOCOL

In this section, we describe the methods to generate secure session keys and how data is securely exchanged between the peers considering the above models. At first we describe the parameters that are used to generate the session key and message authentication code; later we describe the security mechanisms of our proposed models.

4.1 Parameters for the proposed models

In order to generate a secret session key and the message authentication code for data exchange in three models, different parameters are required. The parameters are discussed as follows.

**System parameters:** System parameters (e.g. group, bilinear map, hash function) are used for generating a secret session key for data exchange between peers. Depending on the mutual agreement between peers, system parameters may be fixed for each data exchange session or they may be changed for each session. Depending on the situation the system parameters may be private or public.

**Session parameters:** Session parameters (e.g. dynamically generated id of peers, random number in \( \mathbb{Z}_p^* \) random numbers) are used for a specific data exchange session in order to generate the secret session key. These parameters are dynamic for each session of data exchange. Depending on the situation the session parameters may be private or public.

**Private secret parameters:** Private parameters (e.g. dynamically generated random number in \( \mathbb{Z}_p^* \) that are based on the private attributes of the peers) are defined and only known to the peer itself.

4.2 Parameters for Model 1

Assume a source peer \( P_i \) with schema \( S \) and a target peer \( P_j \) with schema \( T \). Also assume that based on the data exchange policy between \( P_i \) and \( P_j \) the shared attributes are classified as follows:

- **Confidential attributes (CA):** \( \{CA_1, CA_2, \ldots, CA_m\} \)
- **Non-confidential attributes (NCA):** \( \{NCA_1, NCA_2, \ldots, NCA_p\} \)

The purpose of the security protocol is to ensure secure data exchange when \( P_j \) requests data from \( P_i \) through a query \( Q \) that contains confidential attributes as well as non-confidential attributes. Assume a query \( Q_t \) at any instance time \( t \) is requested from \( P_j \) to \( P_i \).

In order to request data from \( P_i \), peer \( P_j \) generates the following system and session parameters.

**System parameters:**

- \( G_1 \), an additive group of prime order \( q \).
- \( H_1 : \{0,1\}^* \to G_1 \), a collision resistant cryptographic hash function which maps from arbitrary-length strings to an element in \( G_1 \).

**Session parameters:**

- A dynamically generated id of peer \( P_j \), \( ID_{P_j} = H_1(P_j) \in G_1 \), where \( \gamma \) is a random number.
- A random number \( R_{j-session} \) which is used for generating the authentication and verification codes for the target \( P_j \).

After generating the parameters \( < G_1, H_1, ID_{P_j}, R_{j-session} > \), peer \( P_i \) sends the parameters with the query \( Q_t \) to \( P_j \). When \( P_j \) receives the parameters and the query, \( P_j \) identifies the confidential and non-confidential attributes. Assume \( P_j \) identifies the following confidential and non-confidential attributes from the query \( Q_t \):

- **Confidential attributes in \( Q_t \):** denoted by \( CA_{Q_t} = \{QCA_1, QCA_2, \ldots, QCA_m'\} \subseteq CA \)
- **Non-confidential attributes in \( Q_t \):** denoted by \( NCA_{Q_t} = \{QNCA_1, QNCA_2, \ldots, QNCA_p'\} \subseteq NCA \)
When $P_i$ receives the parameters from $P_j$, it also generates system and session parameters for computing a secret session key for the authentication of $P_i$ and for encryption of the query result, $Q_i^t$. The generated parameters are given below. System parameters:

- $H_2 : \{0,1\}^* \rightarrow Z_p^*$, a collision resistant cryptographic hash function which maps from arbitrary-length strings to elements in $Z_p^*$.
- $H_3 : \{0,1\}^* \rightarrow \{0,1\}^{\lambda}$, a collision resistant cryptographic hash function mapping from arbitrary-length strings to $\lambda$-bit fixed length strings.

Session parameters:

- An ID $ID_{P_i} = H_1(P_i^s) \in G_1$, where $\zeta$ is a random number.
- A random number $R_{i-\text{SESSION}}$ which is used for generating the authentication and verification codes for the source $P_i$.

Depending on the confidential and non-confidential attributes, the source $P_i$ and the target $P_j$ generates their secret session key $K_{S_i}$ and $K_{S_j}$, and authentication code $Aut_0$ and $Aut_1$, respectively. Let $\text{SESSION\_KEY}_{\text{Attribute}}$ be the set of confidential and non-confidential attributes that are used to generate a session key. Therefore, in Model 1, $\text{SESSION\_KEY}_{\text{Attribute}} = CA_{Q_i} \cup NCA_{Q_i}$. To generate a session key and an authentication code, source and target use their own parameters and the parameters received from each other. The generation and the purpose of the session keys and authentication codes for Model 1 and Model 2 are discussed in Section 4.4.

4.3 Parameters for Model 2

In this model the source peer $P_i$ selects shared attributes from its schema $S$ depending on the query $Q_i$ requested by the target peer $P_j$ at time $t$. The target $P_j$ is not aware of the confidential and non-confidential attributes which are used for generating the session keys $K_{S_j}$. Source $P_i$ determines the confidential and non-confidential attributes after receiving the query from target $P_j$.

In addition to the parameters of the Model 1, the target $P_j$ and the source $P_i$ generate the following session parameters:

Session parameters:

- Target $P_j$ and source $P_i$ generate random numbers $R_{i-\text{ATTRIBUTE}}$ and $R_{i-\text{ATTRIBUTE}}$, respectively, that are used for computing the authentication codes of the attributes.

In every session, $P_i$ randomly selects confidential and non-confidential attributes from the query $Q_i$ for computing a secret session key. The procedure of selecting the confidential and non-confidential attributes is presented below.

4.3.1 Attributes selection in Model 2

After receiving $Q_i$, from $P_j$, $P_i$ randomly selects one confidential attribute $QCA_1 \in CA_{Q_i}$ and another non-confidential attribute $QNC_{A_1} \in NCA_{Q_i}$. In addition, $P_i$ generates attribute authentication codes $S_{\text{AttrAut}}_{i} = H_3(QCA_1||R_{i-\text{ATTRIBUTE}})$ and $S_{\text{AttrAut}}_{i} = H_3(QNC_{A_1}||R_{i-\text{ATTRIBUTE}})$ and sends them to $P_j$. The selection of the attributes that are used to generate the session key by the source $P_i$ is discussed as follows.

Source $P_i$ sends the system parameters $< H_2, H_3 >$ including the session parameters $< ID_{P_i}, R_{i-\text{ATTRIBUTE}}, R_{i-\text{SESSION}} >$ and $< S_{\text{AttrAut}}_{i}, S_{\text{AttrAut}}_{i} >$ to the target $P_j$. Target $P_j$ computes authentication codes of its shared attributes $SA_Q$, where $SA_Q \subseteq U_s = att(\Sigma_{s_\text{Aut}})$.

Let, the shared attributes of the target be $SA_Q = \{QCA_1, QCA_2, \ldots, QCA_t, QNC_{A_1}, QNC_{A_2}, \ldots, QNC_{A_t}\}$. Target $P_j$ computes authentication codes of its shared attributes as follows:

$$T_{\text{AttrAut}}_{i} = H_3(QCA_1||R_{i-\text{ATTRIBUTE}})$$
$$T_{\text{AttrAut}}_{i} = H_3(QCA_2||R_{i-\text{ATTRIBUTE}})$$
$$\vdots$$
$$T_{\text{AttrAut}}_{i} = H_3(QNC_{A_t}||R_{i-\text{ATTRIBUTE}}).$$

Target $P_j$ also computes verification codes of its shared attributes as follows:

$$T_{\text{AttrVer}}_{i} = H_3(QCA_1||R_{i-\text{ATTRIBUTE}})$$
$$T_{\text{AttrVer}}_{i} = H_3(QCA_2||R_{i-\text{ATTRIBUTE}})$$
$$\vdots$$
$$T_{\text{AttrVer}}_{i} = H_3(QNC_{A_t}||R_{i-\text{ATTRIBUTE}}).$$

Target $P_i$ defines a matched attributes set denoted as $\text{MatchAtt}$ and initializes it with null. $\text{MatchAtt}$ is used to collect the authentication codes of the attributes that are matched between the source and the target. Also, target $P_j$ defines an un-matched attributes set denoted as $\text{UnMatchAtt}$ and initializes the set with all of the verification codes. Target $P_j$ compares $< S_{\text{AttrAut}}_{i}, S_{\text{AttrAut}}_{i} >$ with the verification codes. If $< S_{\text{AttrAut}}_{i} >$ or $< S_{\text{AttrAut}}_{i} >$ matches with a verification code then the corresponding share attribute $QCA_i$ or $QNC_{A_i}$ is collected, and a new attribute authentication code $NEW_{T_{\text{Aut}}} = H_3(QCA_i||R_{i-\text{ATTRIBUTE}})$ or $NEW_{T_{\text{Aut}}} = H_3(QNC_{A_i}||R_{i-\text{ATTRIBUTE}})$ is computed. If $< S_{\text{AttrAut}}_{i} >$ and $< S_{\text{AttrAut}}_{i} >$ both match with verification codes then the corresponding share attributes $QCA_i$ and $QNC_{A_i}$ are collected as well as $NEW_{T_{\text{Aut}}} = H_3(QCA_i||R_{i-\text{ATTRIBUTE}})$ or $NEW_{T_{\text{Aut}}} = H_3(QNC_{A_i}||R_{i-\text{ATTRIBUTE}})$ is computed. Finally, $\text{MatchAtt} = \{NEW_{T_{\text{Aut}}}\}$ or $\text{MatchAtt} = \{NEW_{T_{\text{Aut}}} \}$. Procedure attributes finding describes the formation of $\text{MatchAtt}$ and $\text{UnMatchAtt}$ which we present below.
PROCEDURE attributes_finding (Parameters: \{authentication code\}, \{verification code\})
\{\}, represents the empty set.
\(\text{TEMP}_{\text{MatchAttr}}\) is used to collect the verification codes.
\(\text{TEMP}_{\text{VerifierAttr}}\) is used to collect the shared attributes.

STEP 1. \(\text{MatchAttr} := \{\}\); \(\text{TEMP}_{\text{MatchAttr}} := \{\}\) and \(\text{TEMP}_{\text{VerifierAttr}} := \{\}\).

STEP 2. \(\text{UnMatchAttr} := \{\text{AttrVer}_{1}, \text{AttrVer}_{2}, \ldots, \text{AttrVer}_{1+n}, \ldots, \text{AttrVer}_{m-1}, \text{AttrVer}_{m}\}\).

STEP 3. Compare \(< \text{SAttr}_{i} > \in \text{UnMatchAttr}\).
3.a. IF \(< \text{SAttr}_{i} > \in \text{UnMatchAttr}\).
   \(\text{SAttr}_{i} := \{\}\).
3.a.2. \(\text{UnMatchAttr} := \text{UnMatchAttr} - \text{TEMP}_{\text{MatchAttr}}\).

STEP 4. \(\text{MatchAttr} := \{\}\); \(\text{UnMatchAttr} := \{\}\).
4.a. IF \(< \text{SAttr}_{i} > \in \text{UnMatchAttr}\) THEN
   4.a.1. \(\text{TEMP}_{\text{MatchAttr}} := \text{TEMP}_{\text{MatchAttr}} + \text{SAttr}_{i}\).
4.a.2. \(\text{UnMatchAttr} := \text{UnMatchAttr} - \text{TEMP}_{\text{MatchAttr}}\).

STEP 5. IF \(\text{TEMP}_{\text{MatchAttr}} \neq \{\}\) THEN
5.a. \(\text{TEMP}_{\text{VerifierAttr}} := \{\text{SESSION}_{\text{VerifierAttr}}\}\); where \("\text{SESSION}_{\text{VerifierAttr}}\) is the share attribute of the corresponding verification code.
5.b. \(\text{MatchAttr} := \text{MatchAttr} \cup \{\text{Element}_k \in \text{TEMP}_{\text{VerifierAttr}}\}\) where \("\text{Element}_k \in \text{TEMP}_{\text{VerifierAttr}}\) is the elements of \(\text{TEMP}_{\text{VerifierAttr}}\).

Target \(P_j\) sends session parameters \(< \text{MatchAttr}, \text{UnMatchAttr}, \text{R}_j\text{-attribute}>\) to the source \(P_i\).

Source \(P_i\) checks \(\text{MatchAttr}\); if \(\text{MatchAttr}\) is empty then source realizes that \(< \text{QCA}_1, \text{QCA}_2 >\) are not available to the target \(P_i\). If \(\text{MatchAttr}\) is not empty then source computes new verification codes \(\text{NEW}_{\text{VerifierAttr}} = \text{H}_3(\text{QCA}_1||\text{R}_j\text{-attribute})\) and \(\text{NEW}_{\text{VerifierAttr}} = \text{H}_3(\text{QCA}_2||\text{R}_j\text{-attribute})\) and \(\text{Compare}_{\text{MatchAttr}}\) and \(\text{Compare}_{\text{NEW}_{\text{VerifierAttr}}}\) with the elements of \(\text{MatchAttr}\). If \(\text{NEW}_{\text{VerifierAttr}}\) and/or \(\text{NEW}_{\text{VerifierAttr}}\) matches with the elements of \(\text{MatchAttr}\) then \(< \text{QCA}_1 >\) and/or \(< \text{QCA}_2 >\) are collected by \(P_i\) in its temporary random attributes set \(\text{TEMP}_{\text{RandAttr}}\).

Thus, \(\text{TEMP}_{\text{RandAttr}} = \{\text{QCA}_1\}, \{\text{QCA}_2\}\), or \(\{\text{QCA}_1, \text{QCA}_2\}\).

Source \(P_i\) generates verification codes of its confidential attributes \(\text{CA}_{Q_i}\) and non-confidential attributes \(\text{NCA}_{Q_i}\) as follows:

\[\text{S}_{\text{CA}_{Q_i}-\text{Verifier}} = \text{H}_3(\text{QCA}_1||\text{R}_j\text{-attribute})\]
\[\text{R}_j\text{-attribute}\]
\[\text{S}_{\text{CA}_{Q_i}-\text{Verifier}} = \text{H}_3(\text{QCA}_2||\text{R}_j\text{-attribute})\]
\[\text{R}_j\text{-attribute}\]...
Target $P_j$ collects $\ell$ number of verification codes from $\{T_{\text{SessionAttrVer}}\}$ for all $i$, $i = 1 \cdots (r + \ell)$, where the set of the collected verification codes, denoted $T_{\text{SessionCollectAttrVer}}$, is $T_{\text{SessionCollectAttrVer}} = \{T_{\text{SessionAttrVer}}\} \forall j, j = 1 \cdots \ell$. Finally, target $P_j$ collects the shared attributes corresponding to the verification codes of the set $T_{\text{SessionCollectAttrVer}}$ that are used to generate the session key $K_{S_j}$ for the target $P_j$. Thus the shared attributes corresponding to the verification codes of the set $T_{\text{SessionCollectAttrVer}}$, denoted $\text{SESSIONKEY}_{\text{Attribute}} = \{\text{QCA}_1, \text{QCA}_3, \text{QNA}_A, \cdots \}$, are used to verify the attributes set $\text{SESSIONKEY}_{\text{Attribute}}$ with the source $P_i$.

$T_{\text{SessionAttrAut}} = H_3(\text{QCA}_1||\text{R}_i\text{-attribute}||)$

$T_{\text{SessionAttrAut2}} = H_3(\text{QCA}_3||\text{R}_i\text{-attribute}||)$

$T_{\text{SessionAttrAut3}} = H_3(\text{QNA}_A||\text{R}_i\text{-attribute}||)$

$T_{\text{SessionAttrAut4}} = H_3(\text{QNA}_A||\text{R}_j\text{-attribute}||)$

$T_{\text{SessionAttrAut5}} = H_3(\text{QNA}_A||\text{R}_j\text{-attribute}||)$

$T_{\text{SessionAttrAut6}} = H_3(\text{QNA}_A||\text{R}_j\text{-attribute}||)$

$T_{\text{SessionAttrAut7}} = H_3(\text{QNA}_A||\text{R}_i\text{-attribute}||)$

Target $P_j$ further generates the following verification codes of the shared attribute set $\text{SESSIONKEY}_{\text{Attribute}}$ for cross-authentication checking of the attributes set $\text{SESSIONKEY}_{\text{Attribute}}$ with the source $P_i$.

$S_{\text{SessionAttrAut1}} = H_3(\text{QCA}_1||\text{R}_i\text{-attribute}||)$

$S_{\text{SessionAttrAut2}} = H_3(\text{QCA}_3||\text{R}_i\text{-attribute}||)$

$S_{\text{SessionAttrAut3}} = H_3(\text{QNA}_A||\text{R}_i\text{-attribute}||)$

$S_{\text{SessionAttrAut4}} = H_3(\text{QNA}_A||\text{R}_j\text{-attribute}||)$

$S_{\text{SessionAttrAut5}} = H_3(\text{QNA}_A||\text{R}_j\text{-attribute}||)$

$S_{\text{SessionAttrAut6}} = H_3(\text{QNA}_A||\text{R}_j\text{-attribute}||)$

$S_{\text{SessionAttrAut7}} = H_3(\text{QNA}_A||\text{R}_i\text{-attribute}||)$

Target $P_j$ sends $\{S_{\text{SessionAttrAut}}\}$ for all $i$, $i = 1 \cdots \ell$ to the source $P_i$. After receiving $\{S_{\text{SessionAttrAut}}\}$, the source $P_i$ generates the following verification codes:

$s_{\text{SessionAttrVer1}} = H_3(\text{QCA}_1||\text{R}_i\text{-attribute}||)$

$s_{\text{SessionAttrVer2}} = H_3(\text{QCA}_3||\text{R}_i\text{-attribute}||)$

$s_{\text{SessionAttrVer3}} = H_3(\text{QNA}_A||\text{R}_i\text{-attribute}||)$

$s_{\text{SessionAttrVer4}} = H_3(\text{QNA}_A||\text{R}_j\text{-attribute}||)$

$s_{\text{SessionAttrVer5}} = H_3(\text{QNA}_A||\text{R}_j\text{-attribute}||)$

$s_{\text{SessionAttrVer6}} = H_3(\text{QNA}_A||\text{R}_j\text{-attribute}||)$

$s_{\text{SessionAttrVer7}} = H_3(\text{QNA}_A||\text{R}_i\text{-attribute}||)$

Finally, source $P_i$ compares $\{s_{\text{SessionAttrVer}}\}$ for all $i$, $i = 1 \cdots \ell$ with $\{S_{\text{SessionAttrAut}}\}$, if $\{S_{\text{SessionAttrVer}}\}$ matches $\{S_{\text{SessionAttrAut}}\}$, then the session key $K_{S_i}$ is generated by using the confidential and non-confidential attributes $\text{SESSIONKEY}_{\text{Attribute}} = \{\text{QCA}_1, \text{QCA}_3, \text{QNA}_A, \cdots \}$.

In the next section we discuss the generation of session key and authentication code for Model 1 and Model 2.

### 4.4 Generation of secret session key and authentication code for Model 1 and Model 2

The source $P_i$ computes a secret element in $Z_q^*$, called shared secret parameter, denoted by $\sigma$, that is based on the session key attribute set $\text{SESSIONKEY}_{\text{Attribute}}$, as follows:

$$\sigma = H_2(\text{SESSIONKEY}_{\text{Attribute}}) \in Z_q^*$$

This shared secret parameter $\sigma$ is used as a shared secret session key $K_{S_j}$ (i.e. $K_{S_j} = \sigma$) for exchanging data between the source $P_i$ and the target $P_j$.

Source $P_i$ also generates authentication code $\text{Auth}_0$ as follows:

$$\text{Auth}_0 = H_3(K_{S_j}||\text{ID}_{P_i}||\text{ID}_{P_j}||R_{\text{session}}||0)$$

where $R_{\text{session}}$ is a random number generated by the source $P_i$ to distinguish every session from others so that replay attacks cannot take place on the communication.

Finally, source $P_i$ sends the system parameters $< H_2, H_3 >$ including the session parameters $< \text{ID}_{P_i}, R_{\text{session}}, \text{Auth}_0 >$ to the target $P_j$.

After receiving the system parameters as well as session parameters form the source $P_i$, target $P_j$ generates a session key $K_{S_j}$ as follows:

$$\sigma = H_2(\text{SESSIONKEY}_{\text{Attribute}}) \in Z_q^*$$

Therefore, $K_{S_j} = K_{S_i}$

Verification code $\text{Ver}_{0}$ is computed by the target $P_j$ as follows:

$$\text{Ver}_0 = H_3(K_{S_j}||\text{ID}_{P_i}||\text{ID}_{P_j}||R_{\text{session}}||0)$$

The verification code $\text{Ver}_0$ is computed to verify the authentication code $\text{Auth}_0$ of $P_i$.

Target $P_j$ compares $\text{Ver}_0$ with $\text{Auth}_0$; if $\text{Ver}_0 = \text{Auth}_0$ then target generates another authentication code $\text{Auth}_1$ as follows:

$$\text{Auth}_1 = H_3(K_{S_j}||\text{ID}_{P_i}||\text{ID}_{P_j}||R_{\text{session}}||1)$$

Finally, $P_j$ sends $< \text{Auth}_1 >$ to source $P_i$. Upon receiving $< \text{Auth}_1 >$ from target $P_j$, source $P_i$ generates another verification code $\text{Ver}_1$ as follows, and compares it with $\text{Auth}_1$.

$$\text{Ver}_1 = H_3(K_{S_j}||\text{ID}_{P_i}||\text{ID}_{P_j}||R_{\text{session}}||1)$$

If $\text{Ver}_1$ matches with $\text{Auth}_1$, i.e. $\text{Ver}_1 = \text{Auth}_1$ then source $P_i$ sends the data of the query result $Q^R$ after encrypting it with the session key $K_{S_j}$.
We use “0” and “1” in the generation of $Aut_0$ and $Aut_1$ for distinguishing the computation (session keys and authentication codes) as well as communication (authentication codes) between source $P_i$ and target $P_j$. $P_i$ uses “0” for computing the authentication code ($Aut_0$) and for sending to $P_j$, Hence, the source performs both computation and communication by using “0”; on the other hand, “0” is used only for verification purpose by the target. Therefore, the target performs computation only with “0”. The target uses “1” in computing the authentication code ($Aut_1$) and for sending to the source. Hence, the target performs both computation and communication by using “1”; on the other hand, “1” is used only for verification purposes by the source. Hence, the source performs only computation with “1”.

4.5 Secure authenticated data exchange in Model 1 and Model 2

After authentication between the source and the target, source $P_i$ generates a message authentication code, denoted by $MAC_{message}$ of the query result $Q_i^R$, which is computed as $MAC_{message} = H_3(Q_i^R)$. The source also encrypts $Q_i^R$ with its secret session key $K_{S_i}$, denoted by $CIPHER_{QR}$, which is computed as $CIPHER_{QR} = E_{K_{S_i}}(Q_i^R)$, where $E_{K_{S_i}}$ means encryption using the session key $K_{S_i}$. Finally, $P_i$ sends the following packet to $P_j$.

$$<ID_{P_j}, CIPHER_{QR}, MAC_{message}, ID_{P_j}>$$

After receiving the packet, $P_j$ decrypts $CIPHER_{QR}$ with the session key $K_{S_j}$ denoted as $D_{K_{S_j}}(CIPHER_{QR})$ and generates the verification message authentication code, denoted by $VER_{message}$, which is computed as follows:

$$VER_{message} = H_3(D_{K_{S_j}}(CIPHER_{QR}))$$

Finally, $P_j$ compares $VER_{message}$ with $MAC_{message}$. If $VER_{message} = MAC_{message}$ then the data is accepted.

The whole process is illustrated in Figure 4 and described in the following steps.

The step-by-step procedure of the proposed protocol goes as follows:

Key generation and message authentication code of the query result for Model 1 and Model 2 is described in the following steps. Attribute selection steps for Model 2 are not included within these steps.

STEP 1: A query $Q_t$ is generated at the target $P_j$.

STEP 2: Target $P_j$ determines group $G_1$, hash function $H_1$ and performs the following steps:

2.a: Generates an ID $ID_{P_j}$.
2.b: Sends $<G_1, H_1, Q_t, ID_{P_j}, R_{j-session}>$ to the source $P_i$.

STEP 3: Source $P_i$ executes the query $Q_t$ on its local database and performs the following steps:

3.a: Determines cryptographic hash functions $H_2$ and $H_3$.

3.b: Generates an ID $ID_{P_i}$, a random number $R_{i-session}$.****: (For Model 2: Assume $SESSION_{Key,Attribute}$ is already collected)
3.c: Generates secret session key $K_{S_i}$, authentication code $Aut_0$.
3.d: Sends $<H_2, H_3, ID_{P_i}, R_{i-session}, Aut_0>$ to the target $P_j$.

STEP 4: Target $P_j$ generates secret session key $K_{S_j}$, verification code $Ver_0$.
4.a: Generates random $R_{j-session}$.
4.b: Compares $Ver_0$ with $Aut_0$ if $Ver_0 = Aut_0$ then generates $Aut_1$.
4.c: Sends $<Aut_1, ID_{P_i}>$ to the source $P_i$.

STEP 5: Source $P_i$ generates verification code $Ver_1$.
5.a: Compares $Ver_1$ with $Aut_1$ if $Ver_1 = Aut_1$ then generates message authentication code $MAC_{message}$.

5.b: Encrypts query result $Q_i^R$, with session key $K_{S_i}$ denoted as $CIPHER_{QR}$.
5.c: Sends $<ID_{P_i}, CIPHER_{QR}, MAC_{message}, ID_{P_i}>$ to the target $P_j$.

STEP 6: Target decrypts $CIPHER_{QR}$ with session key $K_{S_j}$, generates verification message authentication code $VER_{message}$; compares $VER_{message}$ with $MAC_{message}$; if $VER_{message} = MAC_{message}$ then data has been exchanged successfully.

4.6 Parameters for Model 3

In identity-based crypto there is generally a private key generator (PKG) which entities use in order to obtain their private keys. This is a trusted authority (like a CA in a PKI). In our proposed protocol there is no PKG but still our protocol
works properly. In this proposed security protocol, responsibilities of a PKG is mutually performed by the source and the target.

In this model the target $P_j$ requests the source $P_i$ to send query result $Q_i^R \in \{0,1\}^{n-\ell_0}$ in such a way that no one can access $Q_i^R$ without having the private attributes of the target. Therefore, at the beginning source $P_i$ verifies target’s query request. To this end the source and the target do the same sequence of tasks as in Model 2 to select the session key attribute set $SESSION_{KEYAttribute} = \{QCA_1, QCA_3, \ldots, QNCA_1, QNCA_4\}$.

Source $P_i$ computes $\sigma$ as discussed in Section 4.4 and an identification authentication code $Aut_{ID_i}$ as follows.

$$Aut_{ID_i} = H_3(\sigma||ID_{P_i}||ID_{P_j}||R_i||\text{SESSION}||nonce_i||0);$$

where $nonce_i$ is a random number.

Source $P_i$ sends $<Aut_{ID_i}, nonce_i>$ to the target $P_j$. After receiving $<Aut_{ID_j}, nonce_j>$ from the source, target $P_j$ computes $\sigma$ similar to the source, generates verification code denoted $Ver_{ID_j}$ as follows and compares $Ver_{ID_j}$ with $Aut_{ID_j}$.

$$Ver_{ID_j} = H_3(\sigma||ID_{P_i}||ID_{P_j}||R_j||\text{SESSION}||nonce_j||0);$$

If $Ver_{ID_j} = Aut_{ID_j}$ then target $P_j$ generates randomly a **private secret parameter** denoted $\beta$ from the target’s private attribute $P_{\text{privateAttr}}$, generates an identity authentication code $Aut_{ID_j}$, and public key $P_{j PUB - K_{\text{sym}}}$ as follows:

$$\beta = H_2(P_{j \text{PUB - Attr}}) \in Z_q^*$$

$$P_{j PUB - K_{\text{sym}}} = \beta ID_{P_j};$$

where $ID_{P_j} \in G_1$, identity of the source $P_i$.

$$Aut_{ID_j} = H_3(\sigma||ID_{P_i}||ID_{P_j}||R_j||\text{SESSION}||P_{j PUB - K_{\text{sym}}}||nonce_j||nonce_j||1);$$

where $nonce_j$ is a random number.

Target $P_j$ sends $<P_{j PUB - K_{\text{sym}}}, Aut_{ID_j}, nonce_j>$ to the source $P_i$. After receiving $<P_{j PUB - K_{\text{sym}}}, Aut_{ID_j}, nonce_j>$ form the target $P_j$, source $P_i$ generates an identity verification code $Ver_{ID_j}$ as follows and compares with $<Aut_{ID_j}$.

$$Ver_{ID_j} = H_3(\sigma||ID_{P_i}||ID_{P_j}||R_j||\text{SESSION}||P_{j PUB - K_{\text{sym}}}||nonce_j||nonce_j||1);$$

If $Ver_{ID_j} = Aut_{ID_j}$ then source $P_i$ generates the following parameters in addition to the parameters of the Model 1 and Model 2.

**System parameters:**

- $G_2$, a multiplicative group of the same prime order $q$ as the order of the additive group $G_1$.
- $\tilde{\varepsilon}$: $G_1 \times G_1 \rightarrow G_2$ is the bilinear map.
- $H_4$, a cryptographic collision resistant hash function defined as $H_4: \{0,1\}^{n-\ell_0} \times \{0,1\}^{\ell_0} \rightarrow Z_q^*$, where $Z_q^*$ = $\{\mu|1 \leq \mu \leq q - 1\}$, integer $n$ and $l_0$, $0 < l_0 < n$.
- $H_4$, a cryptographic collision resistant hash function defined as $H_5: G_1 \times G_2 \times G_1 \rightarrow \{0,1\}^n$, where $n > 0$.

### 4.6.1 Secure data exchange in the Model 3

Assume source $P_i$ has at least one **private attribute** and it is denoted $P_{i \text{PrivateAttr}}$. $P_i$ generates **private secret parameters** $\omega$ and $P_{i PUB}$ based on the private attribute $P_{i \text{PrivateAttr}}$. $P_i$ also generates a public parameter $P_{j PUB}$ as follows:

$$\omega = H_2(P_{i \text{PrivateAttr}}) \in Z_q^*$$

$$P_{j PUB} = \omega ID_{P_j};$$

$P_{j PUB}$ is a random number. $P_j$ requests the source $P_i$ to send a **private secret parameter** $s$ based on the query result $Q_i^R \in \{0,1\}^{n-\ell_0}$; and generates a random number $\delta$, where $\delta \in \{0,1\}^{\ell_0}$.

$$s = H_4(Q_i^R, \delta) \in Z_q^*$$

**Encryption and authentication code generation of the query result $Q_i^R \in \{0,1\}^{n-\ell_0}$ by the source:**

To encipher the query result $Q_i^R$, source $P_i$ generates the parameters $T \in G_2$ and $\Omega \in G_1$ as follows:

$$\Gamma = \tilde{\varepsilon}(P_{i PUB}, ID_{P_j})^* = \tilde{\varepsilon}(\omega ID_{P_j}, ID_{P_j})^*;$$

$[s$ and $P_{i PUB}$ are the private parameters of the source $P_i]$

$$\Omega = sP_{j PUB - K_{\text{sym}}};$$

The source encrypts the query result $Q_i^R$, denoted $\text{Cipher}(Q_i^R)$, and computes as follows:

$$\text{Cipher}(Q_i^R) = \Gamma, \Delta >;$$

where $\Gamma = sID_{P_i}$; $\Delta = (Q_i^R||\delta) \oplus H_3(\Gamma, T, \Omega)$; and it is simplified as:

$$\text{Cipher}(Q_i^R) = \Gamma, \Delta > = < \Gamma, \tilde{\varepsilon}(\Gamma, T, \Omega);$$

$$H_3(sID_{P_i}, \tilde{\varepsilon}(P_{i PUB}, ID_{P_j})^*, sP_{j PUB - K_{\text{sym}}}) >$$

The source computes a message authentication code $MAC_{Q_i^R}$ of the query result $Q_i^R$ as follows:
**MAC**$_i = H_3(Q_i || \sigma || nonce_i || 0)$

Finally, the source sends $Cipher\{Q_i^R\}, P_i, \text{MAC}_{Q_i^R}$ with the parameters $< G_2, \hat{e}, H_4, H_5 >$ to the target:

**Decryption and verification code generation of the query result** $Q_i^R \in \{0, 1\}^{^n-10}$ by the target:

At first the target splits $\Gamma$ from $Cipher\{Q_i^R\} = < \Gamma, \Delta >$ and computes $\Delta = H_5(\Gamma, \tilde{T}, \tilde{\Omega})$; where, $\tilde{T}$ and $\tilde{\Omega}$ are computed as follows:

$\tilde{T} = \hat{e}(\Gamma, P_i - H_5(\Gamma, \tilde{T}, \tilde{\Omega})) = \hat{e}(sID_{P_i}, \omega ID_{P_i});$ [$s$ and $\omega$ are the private parameters of the source $P_i$]

$\tilde{T} = \hat{e}(ID_{P_i}, ID_{P_i})^\omega$

$\tilde{\Omega} = \hat{e}(\omega ID_{P_i}, ID_{P_i})^s$

$\tilde{\Omega} = \hat{e}(P_i - P_{TV}, ID_{P_i})^s$

$\tilde{\Omega} = \Omega$

The target computes $\tilde{\Omega}$ as follows:

$\tilde{\Omega} = \beta \Gamma; \ \ [\beta$ is the private parameter for the target $P_i]$  

$\beta \Gamma = \beta sID_{P_i};$ [$s$ is a private parameter for the source $P_i$]

$\tilde{\Omega} = s\beta ID_{P_i}$

$\tilde{\Omega} = sP_i - P_{TV} - \kappa_{sv}$

$\tilde{\Omega} = \Omega$

Finally, target evaluates the the following expression:

$\Delta = H_5(\Gamma, \tilde{T}, \tilde{\Omega})$

$\Delta = (Q_i^R || \delta) \oplus H_5(\Gamma, \tilde{T}, \tilde{\Omega})$

$\Delta = (Q_i^R || \delta) \oplus H_5(\Gamma, \tilde{T}, \tilde{\Omega})$

$\Delta = (Q_i^R || \delta)$

Target splits $Q_i^R$ and $\delta$ from $(Q_i^R || \delta)$; and computes a parameter $\delta$ from $Q_i^R$, where, $\delta$ is used for generating the verification code $Ver_{Q_i^R}$ of $Q_i^R$ as follows:

$\delta = H_4(Q_i^R, \delta) \in Z_5$

$\delta = H_4(Q_i^R, \delta) \in Z_5$

$\delta = H_4(Q_i^R, \delta) \in Z_5$

$Ver_{Q_i^R} = H_3(Q_i^R || \delta || \sigma || nonce_i || 0)$

If $(Ver_{Q_i} = MAC_{Q_i^R})$ then target generates another authentication code $MAC_{j-Q_i^R}$ as follows and sends to the source:

$MAC_{j-Q_i^R} = H_3(Q_i^R || \delta || \sigma || nonce_i || nonce_j || 1)$

Upon receiving $MAC_{j-Q_i^R}$ from the target $P_j$, source $P_i$ generates another verification code $MAC_{i-Q_j^R}$ as follows, and compares it with $MAC_{i-Q_j^R}$:

$MAC_{i-Q_j^R} = H_3(Q_j^R || \delta || \sigma || nonce_j || nonce_i || 1)$

If $MAC_{i-Q_j^R}$ matches with $MAC_{j-Q_i^R}$, (i.e $MAC_{i-Q_j^R} = MAC_{j-Q_i^R}$) then the data has been exchanged successfully.

**5. SECURITY ANALYSIS**

In this section we discuss one cryptographic implementation of the proposed protocols and the mechanism for preventing different attacks.

**5.1 Cryptographic implementation**

The bilinear map $\hat{e}$ can be the Tate pairing with some of the modifications and performance improvements described in [18] and the elliptic curve $E$ can be $y^2 = x^3 + x$. The aforementioned group order $q$ is a large 160-bit prime based on another 512-bit prime $p = 2^{16} - 1$ (for some $t$, large enough to make $p$ be the correct size) such bit-length configurations of $p$ and $q$ provide a level of security comparable to RSA cryptography with a key size of 1024 bits [19]. Then $G_1$ is a cyclic subgroup of the additive group of points on the elliptic curve $E$ over the finite field $F_p$, while $G_2$ is a cyclic subgroup of the multiplicative group associated with the finite field $F_p^*$.

If an intruder node captures public messages containing sensitive information $Aut_0$, and/or $Aut_1$, and/or $MAC_{Q_i^R}$ and/or $MAC_{j-Q_i^R}$, still intruder cannot compute any secret key. This is because the key generation is a pairing function operation over an elliptic curve with a secret point. Thus, an outsider node cannot be authenticated as it is not capable of generating shared keys.

**5.2 Prevention of attacks**

**Masquerade attack:** In our proposed protocol, peers authenticate each other before exchanging data. Furthermore, in every session of data exchange between peers, parameters (session/system) are generated dynamically. The session parameters $< R_{session}, Aut_1, Aut_0, R_{j-session} >$ are completely different in each session. Hence, by storing these session parameters, an intruder node cannot pass the authentication process. Therefore, the intruder cannot pretend to be a valid peer in the data exchange. Thus, a masquerade attack is not a threat to our proposed protocol.

**Reply attack:** In our proposed protocol, a malicious peer cannot pass the authentication process. We use an example to illustrate the situation. Consider a scenario with two peers $P_i$ as a source and $P_j$ as a target in a P2PDBMS, and a malicious peer $P_k$ wants to mount a replay attack. Suppose that $P_i$ sends a query $Q_i$ to $P_j$ for data exchange and the session/system parameters generated during the data exchange session are $< G_1, H_1, ID_{P_i}, >, < G_2, \hat{e}, H_4, H_5 >, < ID_{P_j}, R_{j-session}, Aut_0 >$, and $< Aut_1, R_{j-session} >$. The generation of parameters is discussed in Section 4. Assume that when $P_i$ sends $Q_i$ to $P_j$, $P_k$ makes a copy of $Q_i$ and the session/system parameters during the data exchange session for replay attack. Later, $P_k$ sends the query $Q_i$ to the target by using the last session parameters $< G_1, H_1, ID_{P_i} >$ for the replay attack. After receiving these parameters, $P_j$ generates a new session and system parameters, and sends them to $P_k$. Now the random number $R_{j-session}$ is newly generated by source $P_i$ to compute a new authentication code $Aut_0$, denoted $Aut_{new}$, and a new verification code $Ver_1$, denoted $Ver_{1-new}$. Note that after the
session is over. If the verification code stored by the server is different from the verification code computed by the client, they are not the same. This is because the verification code is generated by a one-way hash function, and thus cannot be reversed. Therefore, every session is completely independent of the previous session parameters.

If malicious peer $P_k$ collects $\text{Cipher}_k(Q^k)$, it cannot decrypt the query result $Q^k$. Because $P_k$ cannot compute the secret session key $K_{S_i}$, it cannot complete the authentication process.

If malicious peer $P_k$ collects $\text{Cipher}_k(Q^k)$ then $P_k$ can not decipher $\text{Cipher}_k(Q^k)$ to find query result $Q^k$, because $P_k$ can not generates $\Omega$. Thus, our proposed protocol is robust against replay attack.

Since our protocol is robust against reply attack, the man-in-the-middle attack will also be prevented.

6. RELATED WORK
To the best of the knowledge of the authors, our proposal is the first work for query-based secure session key generation for secure data exchange between peers in P2PDBMS. The only work that is close to the proposal is the work of [22], where the authors claim secure data propagation among multiple nodes by using pre-existing friendship relationships among the nodes in the network. It is assumed that the nodes are friends with each other in real life, thus they have a pre-existing trust relationship. Furthermore, the authors also assume that the nodes have secure keys beforehand; using these fixed security agreements regarding private keys, nodes in the network can set up secure connections and exchange data. This assumption is not realistic in a peer-to-peer database environment. Therefore, the assumption is eliminated from our protocol which does not require any pre-existing security agreement between the peers. The security setup is completely based on query, initiated by a target peer.

7. CONCLUSION
In this paper, we have presented novel secure data exchange protocols for P2PDBMS. The protocols are designed using one way hash function and IBE with pairing-based cryptography. Any two peers that need to exchange data over an insecure medium can generate on-the-fly a secret session key by exchanging some system and session parameters. An important feature of the proposed protocols is that peers always generate a new session key for every new data exchange session; therefore, every session is completely independent with respect to the session key generation, and hence the proposed protocol successfully prevents different attacks. To the best of our knowledge, this is the first security protocol in the literature for secure data exchange between dynamic peers in a P2PDBMS.

8. REFERENCES


