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

Sk. Md. Mizanur Rahman
SITE, uOttawa,
Ottawa, ON, K1N6N5 Canada
mirahman@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, K1N 6N5 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

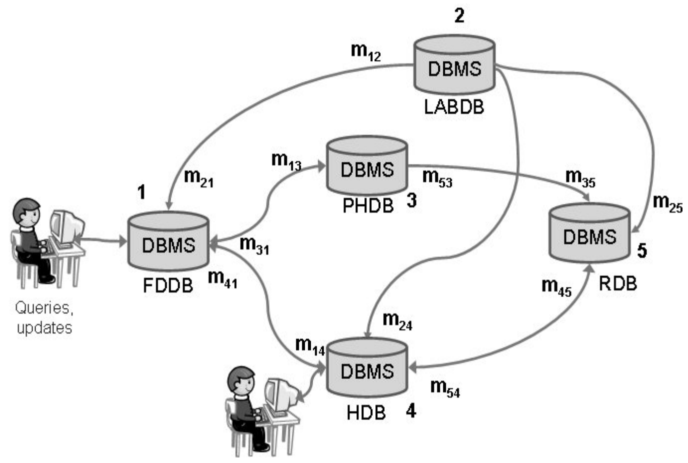


Figure 1: An example model of a peer to peer database management system

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 protects 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 information network', where different parties (e.g. family physicians; walk-in clinics; hospitals; medical laboratories; pharmacists, 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 information network'. In the system, family doctors (FDDB), hospitals (HDB), medical laboratories (LABDB), pharmacists (PHDB), and other stakeholders (e.g. medical research cells (RDB)) are willing to exchange or coordinate information 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 according to established policies between them. For example, family 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 interest to family doctors may need to be exchanged with FDDB. Moreover, family doctors may be interested in collecting test results of their patients from laboratories and the medications 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 constraints. 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 having 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 Cryptography (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 authentication requires a Public Key Infrastructure to issue and revoke certificates and it requires users to store, exchange, 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 notion 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 implement 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 cryptography. 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 authentication 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 paper 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 exchange protocol. Section 6 describes related work, and finally Section 7 concludes and points out avenues for further research.

2. CRYPTOGRAPHIC PRIMITIVES

In this section, we describe some basic cryptographic primitives and mathematical properties which are useful to 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 cryptosystem. 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^k} for some $k \in Z_q^*$, where $Z_q^* = \{\xi | 1 \leq \xi \leq q-1\}$. A mapping $e : G_1 \times G_1 \to G_2$, 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 $Y^2 = X^3 + \alpha X + \beta \bmod T$ where T is a prime number.
- $< P >$ is a subgroup of E generated by P.
- $| < P > | = q$.
- $a, b \in Z_q^*$

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 $(P, aP, bP) \in G_1$ 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 $c = ab \bmod q$ 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 $U = \{A_1, \cdots, A_q\}$ called the universe. Each attribute A_j is associated with a finite set of values called the *domain* of A_j and is denoted by $dom(A_j)$. 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 $\mathbf{R} = (R_1, \cdots, R_m)$ 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

$$\Sigma_{st} = q_S \to q_T$$

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:

$$\forall_{\mathbf{x}}[\exists_{\mathbf{w}}\phi(\mathbf{x},\mathbf{w}) \rightarrow \exists_{\mathbf{z}}\psi(\mathbf{x},\mathbf{z})]$$

where the left hand side (LHS) of the implication, ϕ, is a conjunction of relation atoms over the schema of S and the right hand side (RHS) of the implication, ψ, 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:*

$$\forall_{ohip,name,illness,dob,testname,result}\exists_{name,address}R_1(ohip,name,address,illness,dob), R_2(ohip,testname,result,date) \rightarrow R_3(ohip,illness,dob,testname,result)$$

The mapping expresses that patient data (ohip, name, illness, dob, testname, result) are exchanged from FDB to RDB by a query request q from RDB. 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 $att(\Sigma_{st})$ denote the set of attributes exposed by P_i using the mapping Σ_{st}. Therefore, the shared attributes, denoted by SA, is $SA \subseteq U_s = att(\Sigma_{st})$.*

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). *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 and CA be the set of confidential attributes. Hence, the non-confidential attributes, denoted by NCA, are $SA - CA$.*

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 $U_s - SA$.*

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_3
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

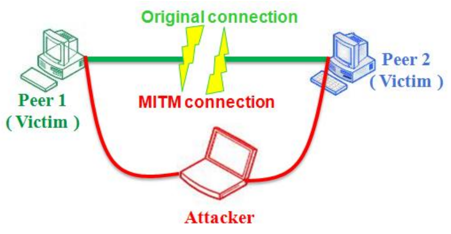


Figure 2: Illustration of man-in-the-middle attack in P2PDBMS.

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 intruders. 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 source for a target. Therefore, a malicious node may 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 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. Theses parameters are generated dynamically without any previous agrement. Hence, if the source or the target performs a social engineering attack during data exchange it is identified. The query result is enciphered 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-

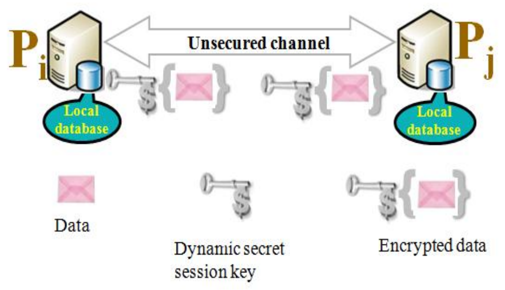


Figure 3: Security Architecture for P2PDBMS.

tocols when two peers want to exchange data, each of them generates its secret session key using the shared attributes between them. Peers generate session key on-the-fly for data exchange based on the requested query. Once the generation of the secret session key is complete, one peer sends a challenge to the other peer for its authentication; the other peer then sends a corresponding response as the answer to the challenge. If the challenge and response match with then the peers begin the data exchange by encrypting the data with their respective secret session key. This process is illustrated in Figure 3. In the proposed protocol, no malicious peer can take part in the communication as they are not authenticated among the peers and cannot self-generate the secret session key. Hence a man-in-the middle, masquerade, and reply attacks are prevented. In addition, the proposed protocols do not require other trusted third-party centralized control services for authenticated transactions between source and target. Peers generate their secret session key on-the-fly as well as authenticate each other. In the following section, we describe our protocols.

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 Z_q^*, 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 Z_q^* 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:

$$\text{Confidential attributes (CA)} = \{CA_1, CA_2, \cdots, CA_m\}$$
$$\text{Non-confidential attributes (NCA)} = \{NCA_1, NCA_2, \cdots, NCA_p\}$$

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.
- $H_1 : \{0,1\}^* \rightarrow G_1$, 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, $ID_{P_j} = H_1(P_j^{\gamma}) \in G_1$, where γ 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_j sends the parameters with the query Q_t to P_i. When P_i receives the parameters and the query, P_i identifies the confidential and non-confidential attributes. Assume P_i identifies the following confidential and non-confidential attributes from the query Q_t:

Confidential attributes in Q_t, denoted by $CA_{Q_t}=\{QCA_1, QCA_2, \cdots, QCA_{m'}\} \subseteq CA$
Non-confidential attributes in Q_t, denoted by $NCA_{Q_t} = \{QNCA_1, QNCA_2, \cdots, QNCA_{p'}\} \subseteq NCA$

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_j and for encryption of the query result, Q_t^R. The generated parameters are given below. *System parameters:*

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

Session parameters:

- An ID $ID_{P_i} = H_1(P_i^\zeta) \in G_1$, where ζ is a random number.
- A random number $R_{i-SESSION}$ 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 $SESSION_{KEY_{Attribute}}$ be the set of confidential and non-confidential attributes that are used to generate a session key. Therefore, in Model 1, $SESSION_{KEY_{Attribute}} = CA_{Q_t} \cup NCA_{Q_t}$. 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_t 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_i}. 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 $R_{j-ATTRIBUTE}$ and $R_{i-ATTRIBUTE}$, 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_t 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_t from P_j, P_i randomly selects one confidential attribute $QCA_1 \in CA_{Q_t}$ and another non-confidential attribute $QNCA_1 \in NCA_{Q_t}$. In addition, P_i generates attribute authentication codes $S_{AttrAut_1} = H_3(QCA_1 || R_{i-ATTRIBUTE})$ and $S_{AttrAut_2} = H_3(QNCA_1 || R_{i-ATTRIBUTE})$ 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 $< ID_{P_i}, R_{i-ATTRIBUTE}, R_{i-SESSION} >$ and $< S_{AttrAut_1}, S_{AttrAut_2} >$ to the target P_j. Target P_j computes authentication codes of its shared attributes SA_Q, where $SA_Q \subseteq U_s = att(\Sigma_{st})$.

Let, the shared attributes of the target be $SA_Q = \{QCA_1, QCA_2, ..., QCA_r, QNCA_1, QNCA_2, QNCA_t\}$. Target P_j computes authentication codes of its shared attributes as follows:
$T_{AttrAut_1} = H_3(QCA_1 || R_{j-ATTRIBUTE} || R_{i-ATTRIBUTE})$;
$T_{AttrAut_2} = H_3(QCA_2 || R_{j-ATTRIBUTE} || R_{i-ATTRIBUTE}); \cdots;$
$T_{AttrAut_r} = H_3(QCA_r || R_{j-ATTRIBUTE} || R_{i-ATTRIBUTE})$;
$T_{AttrAut_{r+1}} = H_3(QNCA_1 || R_{j-ATTRIBUTE} || R_{i-ATTRIBUTE})$;
$T_{AttrAut_{r+2}} = H_3(QNCA_2 || R_{j-ATTRIBUTE} || R_{i-ATTRIBUTE}); \cdots;$
$T_{AttrAut_{r+t}} = H_3(QNCA_t || R_{j-ATTRIBUTE} || R_{i-ATTRIBUTE})$.
Target P_j also computes verification codes of its shared attributes as follows:

$T_{AttrVer_1} = H_3(QCA_1 || R_{i-ATTRIBUTE})$;
$T_{AttrVer_2} = H_3(QCA_2 || R_{i-ATTRIBUTE}); \cdots;$
$T_{AttrVer_r} = H_3(QCA_r || R_{i-ATTRIBUTE})$;
$T_{AttrVer_{r+1}} = H_3(QNCA_1 || R_{i-ATTRIBUTE})$;
$T_{AttrVer_{r+2}} = H_3(QNCA_2 || R_{i-ATTRIBUTE}); \cdots;$
$T_{AttrVer_{r+t}} = H_3(QNCA_t || R_{i-ATTRIBUTE})$.

Target P_j defines a *matched attributes set* denoted as $MatchAttr$ and initializes it with null. $MatchAttr$ 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 $UnMatchAttr$ and initializes the set with all of the verification codes. Target P_j compares $< S_{AttrAut_1}, S_{AttrAut_2} >$ with the verification codes. If $< S_{AttrAut_1} >$ or $< S_{AttrAut_2} >$ matches with a verification code then the corresponding share attribute QCA_1 or $QNCA_1$ is collected, and a new attribute authentication code $NEW_{TAut_1} = H_3(QCA_1 || R_{i-ATTRIBUTE} || R_{j-ATTRIBUTE})$ or $NEW_{TAut_2} = H_3(QNCA_1 || R_{i-ATTRIBUTE} || R_{j-ATTRIBUTE})$ is computed. If $< S_{AttrAut_1} >$ and $< S_{AttrAut_2} >$ both match with verification codes then the corresponding share attributes QCA_1 and $QNCA_1$ are collected as well as NEW_{TAut_1} and NEW_{TAut_2} are computed. Finally, $MatchAttr = \{NEW_{TAut_1}\}$ or $MatchAttr = \{NEW_{TAut_2}\}$ or $MatchAttr = \{NEW_{TAut_1}, NEW_{TAut_2}\}$. Procedure attributes_finding describes the formation of $MatchAttr$ and $UnMatchAttr$ which we present below.

PROCEDURE attributes_finding (Parameters: {authentication code},{verification code})

```
\ { }, represents the empty set.
\ $TEMP_{MatchAttr}$ is used to collect the verification codes.
\ $TEMP_{VerAttr}$ is used to collect the shared attributes.

STEP 1. $MatchAttr := \{\}$; $TEMP_{MatchAttr} := \{\}$
        and $TEMP_{VerAttr} := \{\}$
STEP 2. $UnMatchAttr := \{T_{AttrVer_1}, T_{AttrVer_2}, ...,$
        $T_{AttrVer_r}, T_{AttrVer_{r+1}}, T_{AttrVer_{r+2}}, ..., T_{AttrVer_{r+t}}\}$
STEP 3. Compare $< S_{AttrAut_1} >$ in $UnMatchAttr$
        3.a. IF $< S_{AttrAut_1} >$ matches in $UnMatchAttr$
             THEN
               3.a.1. $TEMP_{MatchAttr} := \{S_{AttrAut_1}\}$
               3.a.2. $UnMatchAttr :=$
                          $UnMatchAttr - TEMP_{MatchAttr}$
STEP 4. Compare $< S_{AttrAut_2} >$ in $UnMatchAttr$
        4.a. IF $< S_{AttrAut_2} >$ matches in $UnMatchAttr$
             THEN
               4.a.1. $TEMP_{MatchAttr} :=$
                          $TEMP_{MatchAttr} \cup \{S_{AttrAut_2}\}$
               4.a.2. $UnMatchAttr :=$
                          $UnMatchAttr - TEMP_{MatchAttr}$
STEP 5. IF $TEMP_{MatchAttr} \neq \{\}$ THEN
        5.a. $TEMP_{VerAttr} := \{SHARED_{ATTR}\}$; where
             "$SHARED_{ATTR}$" is the share attribute of the
             corresponding verification code of
             $TEMP_{MatchAttr}$
        5.b. $MatchAttr := MatchAttr \cup$
               $\{H_3(ElementTemp_{VerAttr}$
                   $|| R_{j-ATTRIBUTE} || R_{i-ATTRIBUTE})\}$;
             where "$ElmentTemp_{VerAttr}$" is the elements
             of $TEMP_{VerAttr}$
```

Target P_j sends session parameters $< MatchAttr, UnMatchAttr, R_{j-ATTRIBUTE} >$ to the source P_i.

Source P_i checks $MatchAttr$; if $MatchAttr$ is empty then source realizes that $< QCA_1, QNCA_1 >$ are not available to the target P_j. If $MatchAttr$ is not empty then source computes new verification codes $NEW_{SVer_1} = H_3(QCA_1||R_{i-ATTRIBUTE}||R_{j-ATTRIBUTE})$ and $NEW_{SVer_2} = H_3(QCA_1||R_{i-ATTRIBUTE}||R_{j-ATTRIBUTE})$; and compares NEW_{SVer_1} and NEW_{SVer_2} with the elements of $MatchAttr$. If NEW_{SVer_1} and/or NEW_{SVer_2} matches with the elements of $MatchAttr$ then $< QCA_1 >$ and/or $< QNCA_1 >$ are collected by P_i in its temporary random attributes set $STemp_{RandATTR}$. Thus, $STemp_{RandATTR} = \{QCA_1\}$, or $\{QNCA_1\}$, or $\{QCA_1, QNCA_1\}$

Source P_i generates verification codes of its confidential attributes CA_{Q_t} and non-confidential attributes NCA_{Q_t} as follows:

$S_{CAttrVer_1} = H_3(QCA_1||R_{i-ATTRIBUTE}||R_{j-ATTRIBUTE})$;
$S_{CAttrVer_2} = H_3(QCA_2||R_{i-ATTRIBUTE}||R_{j-ATTRIBUTE}); \cdots$;
$S_{CAttrVer_m} = H_3(QCA_m||R_{i-ATTRIBUTE}||R_{j-ATTRIBUTE})$;
$S_{NCAttrVer_1} = H_3(QNCA_1||R_{i-ATTRIBUTE}||R_{j-ATTRIBUTE})$;
$S_{NCAttrVer_2} = H_3(QNCA_2||R_{i-ATTRIBUTE}||R_{j-ATTRIBUTE}); \cdots$;
$S_{NCAttrVer_p} = H_3(QNCA_p||R_{i-ATTRIBUTE}||R_{j-ATTRIBUTE})$.

Source P_i compares it's verification codes $\{S_{CAttrVer_i}|\ \forall i,\ i = 1 \cdots m\}$; and $\{S_{NCAttrVer_j}|\ \forall j,\ j = 1 \cdots p\}$; with the elements of $UnMatchAttr$, which is received from target P_j. Further, P_i separates the verification codes that are matched with the elements of $UnMatchAttr$ and makes a list. Source P_i collects the confidential and the non-confidential attributes corresponding to the verification codes that are in the separated list; and keeps the confidential and the non-confidential attributes in P_i's temporary random attributes set $STemp_{RandATTR}$. Assume that P_i keeps the matched attributes $\{QCA_1, QCA_2, QCA_3, QCA_4, QNCA_1, QNCA_2, QNCA_3, QNCA_4\}$ in $STemp_{RandATTR}$. Furthermore, P_i randomly collects some attributes from $STemp_{RandATTR}$ that are used for generating the session key for the current session. Let the set of the attributes randomly collected from $STemp_{RandATTR}$ by P_i, denoted $SESSION_{KEY_{Attribute}}$, and the cardinality of the set $SESSION_{KEY_{Attribute}}$, denoted ℓ; hence $\ell = |SESSION_{KEY_{Attribute}}|$. Assume P_i uses $SESSION_{KEY_{Attribute}} = \{QCA_1, QCA_3, QNCA_1, QNCA_4\}$ for generating the session key K_{S_i}, thus, in this case $\ell = |SESSION_{KEY_{Attribute}}| = 4$

Source P_i generates authentication codes of the attributes of the set $SESSION_{KEY_{Attribute}}$, denoted $S_{SESSION_{AttrAut_k}}$, where $k = 1, 2, \cdots, \ell$, as follows and sends to the target P_j.

$S_{SESSION_{AttrAut_1}} = H_3(QCA_1||R_{i-ATTRIBUTE}||R_{j-ATTRIBUTE}||0)$;
$S_{SESSION_{AttrAut_2}} = H_3(QCA_3||R_{i-ATTRIBUTE}||R_{j-ATTRIBUTE}||0)$;
$S_{SESSION_{AttrAut_3}} = H_3(QNCA_1||R_{i-ATTRIBUTE}||R_{j-ATTRIBUTE}||0)$;
$S_{SESSION_{AttrAut_4}} = H_3(QNCA_4||R_{i-ATTRIBUTE}||R_{j-ATTRIBUTE}||0)$;
After receiving $\{S_{SESSION_{AttrAut_i}}|\ \forall i,\ i = 1 \cdots \ell\}$ from source P_i, target P_j generates the following verification codes as follows:
$T_{SESSION_{AttrVer_1}} = H_3(QCA_1||R_{i-ATTRIBUTE}||R_{j-ATTRIBUTE}||0)$;
$T_{SESSION_{AttrVer_2}} = H_3(QCA_2||R_{i-ATTRIBUTE}||R_{j-ATTRIBUTE}||0); \cdots$;
$T_{SESSION_{AttrVer_r}} = H_3(QCA_r||R_{j-ATTRIBUTE}||R_{i-ATTRIBUTE}||0)$;
$T_{SESSION_{AttrVer_{r+1}}} = H_3(QNCA_1||R_{i-ATTRIBUTE}||R_{j-ATTRIBUTE}||0)$;
$T_{SESSION_{AttrVer_{r+2}}} = H_3(QNCA_2||R_{i-ATTRIBUTE}||R_{j-ATTRIBUTE}||0); \cdots$;
$T_{SESSION_{AttrVer_{r+t}}} = H_3(QNCA_t||R_{i-ATTRIBUTE}||R_{j-ATTRIBUTE}||0)$;

Target P_j collects ℓ number of verification codes from $\{T_{SESSION_{AttrVer_i}} | \forall i,\ i = 1 \cdots (r+t)\}$, where the set of the collected verification codes, denoted $T_{SESSION-COLLECT_{AttVer}}$, is $T_{SESSION-COLLECT_{AttVer}}$ $=\{T_{SESSION_{AttrVer_{j'}}} | \forall j,\ j = 1 \cdots \ell\}$=$\{S_{SESSION_{AttrAut_i}} | \forall i,\ i = 1 \cdots \ell\}$. Finally, target P_j collects the shared attributes corresponding to the verification codes of the set $T_{SESSION-COLLECT_{AttVer}}$ 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 $T_{SESSION-COLLECT_{AttVer}}$, denoted $SESSION_{KEY_{Attribute-J}}$, and $SESSION_{KEY_{Attribute-J}}$=$\{QCA_1, QCA_3, QNCA_1, QNCA_4\}$=$SESSION_{KEY_{Attribute}}$.

Target P_j further generates the following authentication codes of the shared attribute set $SESSION_{KEY_{Attribute-J}}$ for cross authentication checking of the attributes set $SESSION_{KEY_{Attribute}}$ with the source P_i.

$T_{SESSION_{AttrAut_1}} = H_3(QCA_1||R_{i-ATTRIBUTE}||R_{j-ATTRIBUTE}||1)$;
$T_{SESSION_{AttrAut_2}} = H_3(QCA_3||R_{i-ATTRIBUTE}||R_{j-ATTRIBUTE}||1)$;
$T_{SESSION_{AttrAut_3}} = H_3(QNCA_1||R_{i-ATTRIBUTE}||R_{j-ATTRIBUTE}||1)$;
$T_{SESSION_{AttrAut_4}} = H_3(QNCA_4||R_{i-ATTRIBUTE}||R_{j-ATTRIBUTE}||1)$;

Target sends $\{T_{SESSION_{AttrAut_i}} | \forall i,\ i = 1 \cdots \ell\}$ to the source P_i. After receiving $\{T_{SESSION_{AttrAut_i}} | \forall i,\ i = 1 \cdots \ell\}$ from target P_j, source P_i generates the following verification codes:

$S_{SESSION_{AttrVer_1}} = H_3(QCA_1||R_{i-ATTRIBUTE}||R_{j-ATTRIBUTE}||1)$;
$S_{SESSION_{AttrVer_2}} = H_3(QCA_3||R_{i-ATTRIBUTE}||R_{j-ATTRIBUTE}||1)$;
$S_{SESSION_{AttrVer_3}} = H_3(QNCA_1||R_{i-ATTRIBUTE}||R_{j-ATTRIBUTE}||1)$;
$S_{SESSION_{AttrVer_4}} = H_3(QNCA_4||R_{i-ATTRIBUTE}||R_{j-ATTRIBUTE}||1)$;

Finally, source P_i compares $\{S_{SESSION_{AttrVer_i}} | \forall i,\ i = 1 \cdots \ell\}$ with $\{T_{SESSION_{AttrAut_i}} | \forall i,\ i = 1 \cdots \ell\}$; *if* $\{S_{SESSION_{AttrVer_i}} | \forall i,\ i = 1 \cdots \ell\} = \{T_{SESSION_{AttrAut_i}} | \forall i,\ i = 1 \cdots \ell\}$, *then* session key K_{S_i} is generated by using the confidential and non-confidential attributes $SESSION_{KEY_{Attribute}}$ $=\{QCA_1, QCA_3, QNCA_1, QNCA_4\}$.

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 σ, that is based on the session key attribute set $SESSION_{KEY_{Attribute}}$, as follows.

$$\sigma = H_2(SESSION_{KEY_{Attribute}}) \in Z_q^*$$

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

Source P_i also generates authentication code Aut_0 as follows:

$$Aut_0 = H_3(K_{S_i}||ID_{P_i}||ID_{P_j}||R_{i-SESSION}||0)$$

where $R_{i-SESSION}$ 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 $< ID_{P_i}, R_{i-SESSION}, Aut_0 >$ 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:

$$\sigma = H_2(SESSION_{KEY_{Attribute}}) \in Z_q^*$$
$$K_{S_j} = \sigma$$

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

Verification code Ver_0 is computed by the target P_j as follows:

$$Ver_0 = H_3(K_{S_j}||ID_{P_i}||ID_{P_j}||R_{i-SESSION}||0)$$

The verification code Ver_0 is computed to verify the authentication code Aut_0 of P_i.

Target P_j compares Ver_0 with Aut_0; *if* $(Ver_0 = Aut_0)$ *then* target generates another authentication code Aut_1 as follows:

$$Aut_1 = H_3(K_{S_j}||ID_{P_i}||ID_{P_j}||R_{j-SESSION}||R_{i-SESSION}||1)$$

Finally, P_j sends $< Aut_1 >$ to source P_i. Upon receiving $< Aut_1 >$ from target P_j, source P_i generates another verification code Ver_1 as follows, and compares it with Aut_1.

$$Ver_1 = H_3(K_{S_i}||ID_{P_i}||ID_{P_j}||R_{j-SESSION}||R_{i-SESSION}||1)$$

If Ver_1 matches with Aut_1 , *i.e* $(Ver_1 = Aut_1)$ then source P_i sends the data of the query result Q_t^R after encrypting it with the session key K_{S_i}.

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_t^R, which is computed as $MAC_{MESSAGE} = H_3(Q_t^R)$. The source also encrypts Q_t^R with its secret session key K_{S_i}, denoted by $CIPHER_{Q_t^R}$, which is computed as $CIPHER_{Q_t^R} = E_{K_{S_i}}(Q_t^R)$, where $E_{K_{S_i}}$ means encryption using the session key K_{S_i}. Finally, P_i sends the following packet to P_j.

$$< ID_{P_i}, CIPHER_{Q_t^R}, MAC_{MESSAGE}, ID_{P_j} >$$

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

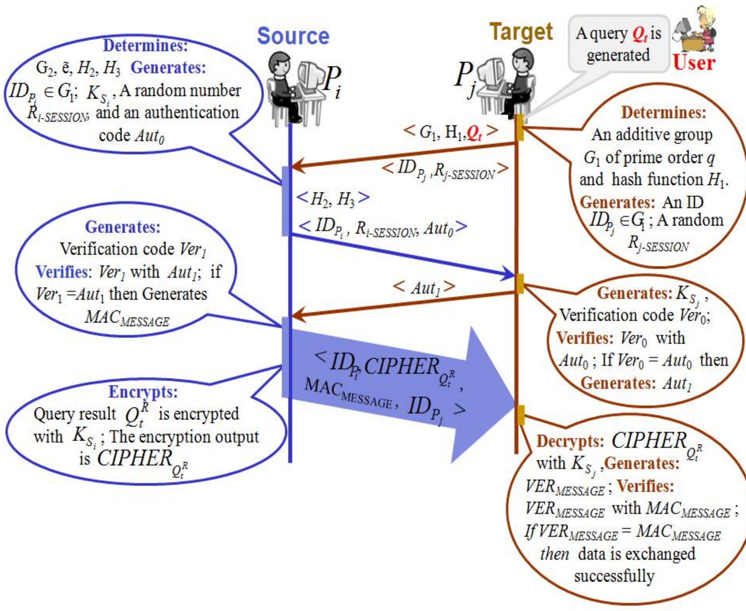
$$VER_{MESSAGE} = H_3(D_{K_{S_j}}(CIPHER_{Q_t^R}))$$

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 $< Aut_1 >$ 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 $Q_t^R$, with session
               key $K_{S_i}$ denoted as $CIPHER_{Q_t^R}$.
          5.c: Sends $< ID_{P_i}, CIPHER_{Q_t^R}, MAC_{MESSAGE},$
               $ID_{P_j} >$ to the target $P_j$.
STEP 6: Target decrypts $CIPHER_{Q_t^R}$ 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.
```



Source
Target
P_i
P_j
A query Q_t is generated
User
Determines: G_2, ẽ, H_2, H_3 Generates: $ID_{P_i} \in G_1$; K_{S_i}, A random number $R_{i\text{-}SESSION}$, and an authentication code Aut_0
$< G_1, H_1, Q_t >$
$< ID_{P_j}, R_{j\text{-}SESSION} >$
Determines: An additive group G_1 of prime order q and hash function H_1. Generates: An ID $ID_{P_j} \in G_1$; A random $R_{j\text{-}SESSION}$
$< H_2, H_3 >$
$< ID_{P_i}, R_{i\text{-}SESSION}, Aut_0 >$
Generates: Verification code Ver_1 Verifies: Ver_1 with Aut_1; if $Ver_1 = Aut_1$ then Generates $MAC_{MESSAGE}$
$< Aut_1 >$
Generates: K_{S_j}, Verification code Ver_0; Verifies: Ver_0 with Aut_0; If $Ver_0 = Aut_0$ then Generates: Aut_1
$< ID_{P_i}, CIPHER_{Q_t^R}, MAC_{MESSAGE}, ID_{P_j} >$
Encrypts: Query result Q_t^R is encrypted with K_{S_i}; The encryption output is $CIPHER_{Q_t^R}$
Decrypts: $CIPHER_{Q_t^R}$ with K_{S_j}, Generates: $VER_{MESSAGE}$; Verifies: $VER_{MESSAGE}$ with $MAC_{MESSAGE}$; If $VER_{MESSAGE} = MAC_{MESSAGE}$ then data is exchanged successfully

Figure 4: Illustration of Key Agreement and Secure Message Communication for P2PDBMS.

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 $Q_t^R \in \{0,1\}^{n-l_0}$ in such a way that no one can access Q_t^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 $SESSION_{KEY_{Attribute}}$ =$\{QCA_1, QCA_3,$-$QNCA_1, QNCA_4\}$.

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

$$Aut_{ID_i} = H_3(\sigma||ID_{P_i}||ID_{P_j}||R_{i-SESSION}||nonce_i||0);$$

where $nonce_i$ is a random number.

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

$$Ver_{ID_j} = H_3(\sigma||ID_{P_i}||ID_{P_j}||R_{i-SESSION}||nonce_i||0);$$

If $Ver_{ID_j} = Aut_{ID_i}$ *then* target P_j generates randomly a ***private secret parameter*** denoted β from the target's private attribute $P_{j_{PvtAttr}}$, generates an identity authentication code Aut_{ID_j}, and public key $P_{j_{PUB-Key}}$ as follows:

$$\beta = H_2(P_{j_{PvtAttr}}) \in Z_q^*$$

$$P_{j_{PUB-Key}} = \beta ID_{P_i}; \text{ where } ID_{P_i} \in G_1, \text{ identity of the source } P_i.$$

$$Aut_{ID_j} = H_3(\sigma||ID_{P_i}||ID_{P_j}||R_{i-SESSION}||P_{j_{PUB-Key}}||nonce_i||nonce_j||1)$$

where $nonce_j$ is a random number.

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

$$Ver_{ID_i} = H_3(\sigma||ID_{P_i}||ID_{P_j}||R_{i-SESSION}||P_{j_{PUB-Key}}||nonce_i||nonce_j||1);$$

If $Ver_{ID_i} = Aut_{ID_j}$ *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.
- $\tilde{e} : G_1 \times G_1 \rightarrow G_2$ is the bilinear map .
- H_4, a cryptographic collision resistant hash function defined as $H_4 : \{0,1\}^{n-l_0} \times \{0,1\}^{l_0} \rightarrow Z_q^*$, 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 $P_{i_{PvtAttr}}$. P_i generates ***private secret parameters*** ω and P_{i-PVT} based on the private attribute $P_{i_{PvtAttr}}$. P_i also generates a public parameter P_{i_j-PUB} as follows:

$\omega = H_2(P_{iPvtAttr}) \in Z_q^*$

$P_{i-PVT} = \omega ID_{P_i};$

$P_{i_j-PUB} = \omega ID_{P_j}$; where $ID_{P_j} \in G_1$, identity of the target P_j.

P_i again generates another ***private secret parameter*** s based on the query result $Q_t^R \in \{0,1\}^{n-l_0}$; and generates a random number $\aleph$, where $\aleph \in \{0,1\}^{l_0}$.

$$s = H_4(Q_t^R, \aleph) \in Z_q^*$$

Encryption and authentication code generation of the query result $Q_t^R \in \{0,1\}^{n-l_0}$ ***by the source:***

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

$$\begin{aligned}\Upsilon &= \tilde{e}(P_{i-PVT}, ID_{P_j})^s \\ &= \tilde{e}(\omega ID_{P_i}, ID_{P_j})^s;\end{aligned}$$

[s and P_{i-PVT} are the private parameters of the source P_i]

$\Omega = sP_{j_{PUB-Key}}$

The source encrypts the query result Q_t^R, denoted $Cipher\{Q_t^R\}$, and computed as follows:

$Cipher\{Q_t^R\}$ $=< \Gamma, \Delta >$; where $\Gamma = sID_{P_i}$; $\Delta = (Q_t^R||\aleph) \oplus H_5(\Gamma, \Upsilon, \Omega)$; and it is simplified as:

$$\begin{aligned}Ciph\{Q_t^R\} &=< \Gamma, \Delta > \\ &=< \Gamma, \Big((Q_t^R||\aleph) \oplus H_5(\Gamma, \Upsilon, \Omega)\Big) > \\ &=< \Big(sID_{P_i}\Big), \Big((Q_t^R||\aleph) \oplus H_5\big(sID_{P_i}, \tilde{e}(P_{i-PVT}, ID_{P_j})^s, sP_{j_{PUB-Key}}\big)\Big) >\end{aligned}$$

The source computes a message authentication code $MAC_{Q_t^R}$ of the query result Q_t^R as follows:

$MAC_{Q_t^R} = H_3(Q_t^R||s||\sigma||nonce_i||0)$

Finally the source sends $< Cipher\{Q_t^R\}, P_{i_j-PUB}, MAC_{Q_t^R} >$ with the parameters $< G_2, \tilde{e}, H_4, H_5 >$ to the target:

Decryption and verification code generation of the query result $Q_t^R \in \{0,1\}^{n-l_0}$ by the target:

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

$\tilde{\Upsilon} = \tilde{e}(\Gamma, P_{i_j-PUB})$
$= \tilde{e}(sID_{P_i}, \omega ID_{P_j})$; [$s$ and ω are the private parameters of the source P_i]
$= \tilde{e}(ID_{P_i}, ID_{P_j})^{s\omega}$
$= \tilde{e}(\omega ID_{P_i}, ID_{P_j})^{s}$
$= \tilde{e}(P_{i-PVT}, ID_{P_j})^{s}$
$= \Upsilon$

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

$\tilde{\Omega} = \beta\Gamma$; [$\beta$ is the private parameter for the target P_j]
$= \beta sID_{P_i}$; [s is a private parameter for the source P_i]
$= s\beta ID_{P_i}$;
$= sP_{j_{PUB-Key}}$
$= \Omega$

Finally, target evaluates the the following expression:

$$\Delta \oplus H_5(\Gamma, \tilde{\Upsilon}, \tilde{\Omega})$$
$$= \Big((Q_t^R||\aleph) \oplus H_5(\Gamma, \Upsilon, \Omega)\Big) \oplus H_5(\Gamma, \tilde{\Upsilon}, \tilde{\Omega})$$
$$= (Q_t^R||\aleph) \oplus H_5(\Gamma, \Upsilon, \Omega) \oplus H_5(\Gamma, \Upsilon, \Omega)$$
$$= (Q_t^R||\aleph)$$

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

$\partial = H_4(Q_t^R, \aleph) \in Z_q^*$
$VER_{Q_t^R} = H_3(Q_t^R||\partial||\sigma||nonce_i||0)$

if $(VER_{Q_t^R} = MAC_{Q_t^R})$ *then* target generates another authentication code $MAC_{j-Q_t^R}$ as follows and sends to the source:

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

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

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

If $MAC_{i-Q_t^R}$ matches with $MAC_{j-Q_t^R}$, (*i.e* $MAC_{i-Q_t^R} = MAC_{j-Q_t^R}$) 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 $\tilde{e}$ 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 $p = 2qr - 1$ (for some r 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 $F_{p^2}^*$.

If an intruder node captures public messages containing sensitive information AUT_0, and /or AUT_1, and/or $MAC_{Q_t^R}$ and/or $MAC_{j-Q_t^R}$, 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 can not 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 $< R_{i-SESSION}, Aut_0, Aut_1, R_{j-SESSION} >$ 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_j sends a query Q_t to P_i for data exchange and the sesssion/system parameters generated during the data exchange session are $< G_1, H_1, ID_{P_j} >$, $< G_2, \tilde{e}, H_2, H_3 >$, $< ID_{P_i}, R_{i-SESSION}, Aut_0 >$, and $< Aut_1, R_{j-SESSION} >$. The generation of parameters is discussed in Section 4. Assume that when P_j sends Q_t to P_i, P_k makes a copy of Q_t and the session/system parameters during the data exchange session for replay attack. Later, P_k sends the query Q_t to the source by using the last session parameters $< G_1, H_1, ID_{P_j} >$ for the replay attack. After receiving these parameters, P_i generates a new session and system parameters, and sends them to P_k. Now the random number $R_{i-SESSION}$ is newly generated by source P_i to compute a new authentication code Aut_0, denoted Aut_0^{new}, and a new verification code Ver_1, denoted Ver_1^{new}. Note that after the session is over P_i and P_j do not store Aut_0, Aut_1, Ver_0, and Ver_1. Since $Ver_1^{new} \neq Aut_1$, where Aut_1 is the old authentication code stored by P_k, P_i does not send the query result Q_t^R to P_k.

If $R_{i-SESSION}$ is generated repeatedly by the source P_i and all the previous session parameters are copied by P_k, still P_k cannot decrypt the query result Q_t^R. Because P_k cannot compute *secret session key* K_{S_i} or K_{S_j}, it cannot complete the authentication process.

If malicious peer P_k collects $Cipher\{Q_t^R\}$ then P_k can not decipher $Cipher\{Q_t^R\}$ to find query result Q_t^R, because P_k can not generates Ω. 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

[1] A. Fuxman, P. G. Kolaitis, R. J. Miller, and W. C. Tan, "Peer Data Exchange," In *ACM Trans. Database System*, 31(4):1454-1498, 2005.

[2] C. Beeri and M. Y. Vardi, "A Proof Procedure for Data Dependencies," In *Journal of the ACM*, 31(4):718-741, 1984.

[3] A. Y. Halevy, Z. G. Ives, D. Suciu, and I. Tatarinov, "Schema Mediation in Peer Data Management System," In *Proc. of the Int'l Conf. on Data Engineering*, pages 505-516, 2003.

[4] A. Y. Halevy, Z. G. Ives, J. Madhavan, P. Mork, D. Suciu, and I. Tatarinov, "The Piazza Peer-Data Management System," In *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 16(7):787-798, 2004.

[5] L. Serafini, F. Giunchiglia, J. Molopoulos, and P. Bernstein, "Local Relational Model:A Logocal Formalization of Database Coordination," Technical Report, Informatica e Telecomunicazioni, University of Trento, 2003.

[6] P. Rodriguez-Gianolli, M. Garzetti, L. Jiang, A. Kementsietsidis, I. Kiringa, M. Masud, R. Miller, and J. Mylopoulos, "Data Sharing in the Hyperion Peer Database System," In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB)*, pages 1291-1294, 2005.

[7] A. Kementsietsidis, M. Arenas, and R.J. Miller, "Mapping Data in Peer-to-Peer Systems: Semantics and Algorithmic Issues," In *Proc. of the Int'l Conf. on the Management of Data (ACMSIGMOD)*, pages 325-336, 2003.

[8] M. Masud, I. Kiringa, and A. Kementsietsidis, "Don't Mind Your Vocabulary: Data Sharing Across Heterogeneous Peers," In *Proc. of the Int'l Conf. on Cooperative Information Systems (CoopIS)*, pages 292-309, 2005.

[9] A. Kementsietsidis and M. Arenas, "Data Sharing Through Query Translation in Autonomous Sources," In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB)*, pages 468-479, 2004.

[10] V. Miller, "Uses of Elliptic Curves in Cryptography," Advances in Cryptology, in Crypto'85, Lecture Notes in Computer Science, Springer-Verlag, Vol. 218, 1986, pp. 417-426.

[11] N. Koblitz, "Elliptic Curve Cryptosystems," Mathematics of Computation, Vol.48, 1987, pp.203-209.

[12] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs," in Workshop on Cryptographic Hardware and Embedded Systems (CHES), 2004, pp.119-132.

[13] L. B. Oliveira, R. Dahab, "Pairing-Based Cryptography for Sensor Networks," in 5th IEEE International Symposium on Network Computing and Applications (NCA'06), MA, USA, Jul. 2006.

[14] A. Shamir, "Identity-based Cryptosystems and Signature Schemes," *CRYPTO'84*, on Advances in Cryptology, Springer-Verlag, 1984, pp. 47-53.

[15] D. Boneh and M. Franklin, "Identity-based Encryption from the Weil Pairing," *Proc. CRYPTO 2001, LNCS 2139*, Springer-Verlag, Berlin Heidelberg, 2001, pp. 213-229.

[16] R. Sakai, K. Ohgishi, and M. Kasahara, "Cryptosystems Based on Pairing," *Proc. Symposium on Cryptography and Information Security (SCIS2000)*, Jan. 2000, pp. 26-28.

[17] A. Joux and K. nguyen, "Separating Decision Diffie-Hellman from Diffie-Hellman in Cryptographic

Groups," *Cryptology ePrint Archive*, Report 2001/03, available at http://eprint.iacr.org/2001/03/.

[18] P.S.L. M. Barreto, H.Y. Kim, B. Bynn, and M. Scott, "Efficinet Algorithms for Pairing-Based Cryptosystems", In Proc. CRYPTO 02, Springler Verlag, August 2002.

[19] D. Balfanz, G. Durface and N. Shankar et al., "Secure Handshakes from Pairing-Based Key Agreements," IEEE Symposium on Security and Privacy, May 2003

[20] "The Tate Pairing", available at http://www.computing.dcu.ie/˜ mike/tate.html

[21] "Elliptic Curve Cryptography Tutorial," http://www.certicom.com/index.php/ecc

[22] Bogdan C. Popescu, Bruno Crispo, Andrew S. Tanenbaum, "Safe and Private Data Sharing with Turtle: Friends Team-Up and Beat the System," In Proc. of the 12th Cambridge Intl. Workshop on Security Protocols 2004,Lecture Notes in Computer Science(LNCS 3957), Springer-Verlag Berlin Heidelberg 2006; B. Christianson et al. (Eds.): Security Protocols 2004, pp. 213-220, 2006.