Privacy Preserving Shortest Path Queries on Directed Graph

Sara Ramezanian, Tommi Meskanen, Valtteri Niemi
University of Helsinki and Helsinki Institute for Information Technology
Helsinki, Finland
{[sara.ramezanian, tommi.meskanen, valtteri.niemi]@helsinki.fi}

Abstract—Trust relation in this work refers to permission that is given to a user at source-host to access another user at target-host through an authentication key with a unique fingerprint. We form a directed graph out of these trust relations, such that user-host pairs are considered as nodes and fingerprints as arrows. We present a novel protocol to query the shortest path from node A to node B, in a privacy preserving manner. We would like to use a cloud to perform such queries, but we do not allow the cloud to learn any information about the graph, nor the query. Also the database owner is prevented from learning any information about the query, except that it happened.

II. PRELIMINARIES

The necessary background on the technologies that our protocol is based on are explained in this section.

A. Private Information Retrieval

Private Information Retrieval (PIR) protocol [5] is a two party protocol between a server who stores a database of n items, and a client who holds an index i, 1 ≤ i ≤ n. At the end of protocol, the client retrieves nothing except the i
th item while the server also learns nothing about which item was queried by the client.

B. Blind Signature with RSA

The notion of blind signature (BS) was introduced by Chaum [6] in 1983 to protect the privacy of users of an electronic cash system. BS protocol is a two-party protocol between a user U who holds a message m and a signer S who has access to a secret signing key sk. At the end of protocol, U receives the signature of m (ϕ), while S learns nothing about ϕ or m. In [7], Bellare et al. proposed a BS scheme based on RSA [8] assumption.

C. Paillier Cryptosystem

The Paillier’s homomorphic encryption scheme [9] works as follow. The private key consists of two safe prime numbers p, q such that gcd(pq, (p − 1)(q − 1)) = 1. The public key is the pair (N, g) where N = pq and g is randomly chosen from \( \mathbb{Z}_N^* \) such that the order of g is a non-zero multiple of N. The encryption of m (0 ≤ m < N) is computed as follows: \( \text{Enc}_g(m, r) = g^m \cdot d^N \mod N^2 \), where r is randomly selected from \( \mathbb{Z}_N^* \). Let \( \lambda = \text{lcm}(p − 1, q − 1) \) and \( L(u) = (u − 1)/N \) for \( u \in \mathbb{Z}_N^* \). Then, the decryption functions is defined as follows:

\[
\text{Dec}_g(e) = \frac{L(c^\lambda \mod N^2)}{L(g^{\phi} \mod N^2)} \mod N
\]

D. A Private Information Retrieval Protocol by Chang

Chang’s PIR protocol [10] is a specialized PIR protocol in which the database is a two-dimensional matrix. Accordingly, the client holds the indices \( i^*, j^* \) and wants to retrieve the item located on the \( i^* \)th row and the \( j^* \)th column, denoted by \( x(i^*, j^*) \). Let the identity matrix \( I \) be as follows:

\[
I(t, t') = \begin{cases} 
1 & \text{if } t = t' \\
0 & \text{otherwise.} 
\end{cases}
\]
A summary of this protocol is presented in Fig. 1.

III. PROBLEM STATEMENT

In this paper, we study a scenario where a database
administrative has a database \(D\) that determines the
permission access rules between users based on their public
and private keys. If user A has a public key corresponding to
a private key of user B, we define a trust relation from user
B to user A.

The database \(D\) consists of quintuplets (source-user,
source-host, fingerprint, target-user, target-host). In this
database each source-host has a private key that corresponds to
a fingerprint that is in the quintuplet. In this database the
hash of public keys (fingerprints) are stored, which makes
them shorter. However, it is still impossible to find two public
keys with the same fingerprint. Any target-user provides
remote access to the source-user, if can show that he/she has
a private key corresponding to the public key that corresponds
to the fingerprint. This shows a trust relation from source-user
to target-user by the fingerprint that is identical between them.

We construct a directed graph with the quintuplets in the
database \(D\). The graph of these trust relations is constructed
as follows: the nodes are (user,host)-pairs and arrows are
labelled by fingerprints. For example, the quintuple (user \(\alpha\),
host A, fingerprint \(f\), user \(\beta\), host B) form an arrow from
node (user \(\alpha\), host A), to (user \(\beta\), host B) that is labelled with
the fingerprint \(f\).

In this work we aim to design a protocol to privately
query this graph to determine the shortest path between node
A and node B. The size of the graph of trust relations is big.
Therefore, we would like to use a cloud to perform privacy
preserving path queries on this directed graph. The goal is to
keep the structure of the graph private. Please note that our
protocol is not limited to trust relation databases, and it can
be used to privately query paths in any directed graph.

After executing the protocol, the client knows whether
there is a path from A to B, and if there is a path, he/she
learns the shortest path between these two nodes. Client is
prevented from learning anything else about the graph. Cloud
and database owner do not learn anything about the query
than that it occurred. Also, the cloud is not allowed to learn
the graph.

IV. RELATED WORK

Finding a suitable data structure to store information has
been in the interest of many researchers. One of the well-
known data structures to store and manage data in an efficient
way is a graph.

Graph structures can be used in many applications such as
the GPS systems [11], the web algorithm e.g. [12] and [13],
and on-line social networks [14]. The tend towards utilizing
graph structures also raises the importance of designing a
privacy-preserving mechanism to query graph data structures.

There have been a number of studies to compute a shortest
path in a directed graph in privacy-preserving manner such as
[15] and [16]. In the following subsection, we present a
protocol to privately retrieve whether there is a path from A
to B. We use this protocol as a building block in our protocol.

A. A Protocol by Ramezanian et al.

Ramezanian et al. presented a privacy-preserving protocol
in [17] that consists of three parties; a database owner, a
cloud and a client. The database owner has a database of
trust relations and forms a directed graph with them. In
this graph nodes are users and arrows are fingerprints. The
database owner wants to encrypt this reachability graph and
use a cloud to perform queries on the graph. The aim of the
protocol is to determine if node A can access node B, without
revealing any extra information about the path or the graph
to the clients.

In this protocol, Ramezanian et al. first form the transitive
closure graph from the database, then insert this graph into a
matrix. They encrypt the matrix bit by bit utilizing Meskanen
et al.’s protocol [18]. The database owner then ships the
crypted matrix to the cloud.

---

**Table: Private Information Retrieval Protocol by Chang**

<table>
<thead>
<tr>
<th>Server</th>
<th>Client</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input: a (l \times l) matrix (M)</td>
<td>Input: ((i^<em>, j^</em>))</td>
</tr>
<tr>
<td>For (i = 1, 2, ..., h) Computes:</td>
<td>With Paillier [9] scheme computes</td>
</tr>
<tr>
<td>(\sigma_i = \prod_{t=1}^{h} (\beta_t)^{2(i,t)} \mod N^2) (u_i, v_i \in \mathbb{Z}_N) such that (\sigma_i = u_iN + v_i)</td>
<td>(\alpha_t = E_g(I(t, i^<em>), r_t)) and (\beta_t = E_g(I(t, j^</em>), s_t)) where (t \in {1, 2, ..., h}), random (r_t) and (s_t) are in (\mathbb{Z}_N^*)</td>
</tr>
<tr>
<td>(u = \prod_{t=1}^{h} (\alpha_t)^{u_t}) and (v = \prod_{t=1}^{h} (\alpha_t)^{v_t} \mod N^2)</td>
<td>(\alpha_t) and (\beta_t)</td>
</tr>
<tr>
<td>(\leftarrow) The value of the item in ((i^<em>, j^</em>)) is been retrieved by computing (D_g(D_g(u)N + D_g(v)))</td>
<td>(\rightarrow u) and (v)</td>
</tr>
</tbody>
</table>

---

Fig.1. An overview of the protocol by Chang[10]
V. OUR PROTOCOL

In this section, we present the details of our privacy-preserving path queries protocol on directed graph.

There are three parties involved in this protocol; a database owner \( D \) who possesses a set of quintuples that defines trust relations between users at hosts, a cloud that is not allowed to learn anything about the database nor the queries, and a client who wants to privately perform a query on this graph to retrieve a) whether there is a path from node A to node B, and b) if there is such an access, what is the path between these two nodes.

Now, we first describe the off-line phase (phase 1) of our protocol where the encryption of the data structure takes place. Then, we explain the on-line phases (phase 2, 3 and 4), where the privacy-preserving path queries take place.

A. Phase 1

As explained before, the database owner first forms a directed graph, where the nodes are users at different hosts that are connected with their fingerprints. Fig. 2 shows an example of the graph of trust relations.

The database owner constructs a matrix \( M \) to store the trust relation directed graph in such a way that if there is a trust relation between users \( i \) and \( j \) then \( M_{ij} = 1 \) and otherwise \( M_{ij} = 0 \).

If user A has a permission to access user B, and user B has a permission to access user C, then there is a trust between user A and user C. Thus we need to calculate the transitive closure graph.

Fig. 3 shows the transitive closure graph of the graph in Fig. 2 and Fig. 4 shows the transitive closure matrix of that graph.

Instead of the transitive closure matrix \( T \), we are more interested in the structure of the transitive closure graph. Especially, what are the shortest paths in this graph.

For our purposes we extend Floyd-Warshall algorithm [19]. Originally this protocol just calculates the lengths of the shortest paths in a graph, but our extended version also stores the penultimate node in a shortest path. It generates a matrix \( P \), where \( P_{ij} \) is the penultimate node in a shortest path from \( i \) to \( j \).

Extended Floyd-Warshall algorithm to calculate the penultimate nodes of the shortest paths from all nodes to

\[
B = \begin{bmatrix}
0 & k_{ab} & k_{ac} & k_{ad} \\
k_{ba} & 0 & k_{bc} & k_{bd} \\
k_{ca} & k_{cb} & 0 & k_{cd} \\
k_{da} & k_{db} & k_{dc} & 0
\end{bmatrix}
\]

Fig.6. Matrix \( B \) of AES keys

\[
\mathcal{P} = \begin{bmatrix}
a & b & c & d \\
a & 1 & a & 0 & b & k_{ab} & c & k_{ac} \\
b & c & k_{bc} & 1 & b & 0 & c & k_{bc} \\
c & c & 0 & k_{ca} & a & 0 & c & 0 & 1 \\
d & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

Fig.7. Matrix \( \mathcal{P} \)
all nodes of a directed graph where the vertices are named 
1,...,v is as follows. In this algorithm W is a matrix where 
$W[i][j]$ is the length of the shortest path from i to j.

Extended Floyd-Warshall algorithm:
for i from 1 to v :
for j from 1 to v :
if i == j :
$W_{ij} = 0$;
$P_{ij} = 0$
else if (i,j) is an arrow in the graph :
$W_{ij} = 1$;
$P_{ij} = i$
else :
$W_{ij} = \infty$;
$P_{ij} = \perp$
for k from 1 to v :
for i from 1 to v :
for j from 1 to v :
if $W_{ij} > W_{ik} + W_{kj}$ :
$W_{ij} = W_{ik} + W_{kj}$
$P_{ij} = P_{kj}$

The element $P_{ij}$ of the matrix P tells us what was the last 
node before j on the shortest path from node i to j. In other 
words, if the shortest path between x and y is (x,k,g,f,c,y), 
then $P_{xy} = c$. If there is no path from i to j, then $P_{ij} = \perp$. 
An example of such matrix P is shown in Fig. 5.

The database owner generates matrix B as follows. The 
database owner chooses a symmetric encryption method, such 
as AES [20], and a different key $k_{ij}$ for each node pair i and j. 
All elements of the main diagonal of this matrix are obtained by a dummy key 0. Fig. 6 shows an example of 
matrix B.

The database owner generates matrix P from matrices B and 
P as follows:

$$
P_{ij} = \begin{cases} 
1 & \text{if } i = j \\
\{P_{ij}, k_i, P_j\} & \text{if } i \neq j \text{ and } P_{ij} \neq \perp \\
(0,0) & \text{if } i \neq j \text{ and } P_{ij} = \perp.
\end{cases}
$$

An example of matrix P is shown in Fig. 7.

Next the database owner creates an encrypted matrix A 
where $A_{ij} = E_{k_{ij}}(P_{ij})$ where $i \neq j$, and $A_{ii} = 1$. Thus if 
$\ell$ was the penultimate node on the path from i to j, someone 
who has the key $k_{ij}$ and the element $A_{ij}$ can by decrypting 
find out $\ell$ and how to decrypt $A_{\ell\ell}$. Furthermore, this entity can 
find out the penultimate node on the path from i to $\ell$, if he 
also knows $A_{i\ell}$.

Fig. 8 shows an example of this matrix A.

The owner of the database chooses RSA keys (e, d, n) and 
uses these keys to encrypt each key $k_{ij}$ to create the matrix 
$K_{ij}$, where $K_{ij} = k_{ij}^e \mod n$. Fig. 9 shows an example of 
this matrix.

The owner now gives both matrices A and K and the RSA 
public key (e, n) to the cloud.

B. Phase 2

Let us assume that the client is interested in finding out 
what is the path from node i to node j, if it exists.

By using a PIR protocol he retrieves the element $K_{ij}$ of 
matrix K from the cloud. The cloud also reveals the client 
the public RSA key (e, n).

C. Phase 3

In the third phase of the protocol, the client utilizes blind 
RSA decryption by choosing a random integer r and asking 
the database owner to decrypt the cryptotext 

$$x = r^e K_{ij} \mod n.$$

When the database owner replies with the value $x^d \mod n$, 
the client can calculate $x^d/r = r^{ed} K_{ij}^d/r = k_{ij} \mod n$.

D. Phase 4

In the fourth phase of our protocol, the client retrieves 
element $A_{ij}$ of matrix A from the cloud.

Now the client can use the key $k_{ij}$ obtained in the previous 
phase to decrypt $A_{ij}$ and to find out the penultimate node on 
the shortest path from i to j, say $\ell$, and the key $k_{\ell\ell}$.

If $\ell = \perp$, it means that there is no path from node i to node 
$j$ and the client can stop the protocol.

If $\ell \neq i$ then the client uses again the PIR protocol to 
retrieve the element $A_{i\ell}$ of matrix A from the cloud.

Now the client can use key $k_{i\ell}$ to decrypt $A_{i\ell}$ and to find 
out the penultimate node on the shortest path from i to $\ell$, and 
a new key.

The client continues this way until the complete path has 
been found between i and j.

Fig. 10 shows an overview of our protocol.
Fig.9. Matrix $K$ of RSA encryption of elements in Matrix $B$

\[
K = \begin{pmatrix}
\begin{array}{cccc}
a & b & c & d \\
\bar{0}^e \mod n & k_{ab}^e \mod n & k_{ac}^e \mod n & k_{ad}^e \mod n \\
 k_{ba}^e \mod n & \bar{0}^e \mod n & k_{bc}^e \mod n & k_{bd}^e \mod n \\
 k_{ca}^e \mod n & k_{cb}^e \mod n & \bar{0}^e \mod n & k_{cd}^e \mod n \\
 k_{da}^e \mod n & k_{db}^e \mod n & k_{dc}^e \mod n & \bar{0}^e \mod n
\end{array}
\end{pmatrix}
\]

Fig.10. An overview of our protocol

VI. PERFORMANCE ANALYSIS

In this section, we present the performance analysis of our protocol. First we analyse the time complexity of each phase and then we give an overview of the communication complexity of each phase.

We denote the number of nodes by $l$. We also denote the length of the shortest path that the client is interested in, by $k$. Please note that $k$ is smaller than $l$.

A. Time Complexity

The database owner uses our extended Floyd-Warshall algorithm to create the matrix $P$. The time complexity of this algorithm is $O(l^3)$.

The database owner needs time to create $l^2 - l$ RSA encryptions and as the AES key is longer than the block size, $2(l^2 - l)$ AES encryptions. Also please note that the main diagonal of $P$ does not need to be encrypted because they are known to be 0.

In phase 2, according to Chang’s protocol [10], client needs to calculate $2l$ modular exponentiations and cloud needs to calculate $l^2 + l$ modular exponentiations.

In phase 3, client needs to perform one RSA encryption. Database owner does one decryption. At last, client performs one inverse and one multiplication to get the AES key blindly.

In phase 4, elements in the matrix $K$ are of size 256 bits. Cloud concatenates every 8 elements and obtains $l \times 1/8$ matrix $K'$. Now, client first calculates $l + 1/8$ modular exponentiations and sends them to the cloud. Then, based on the number of nodes in the path that the client is interested in, he/she calculates up to $(k - 1) \times 1/8$ modular exponentiations and sends them to the cloud. The cloud needs to calculate $l^2 + l$ modular exponentiations up to $k$ times.

B. Communication Complexity

In phase 1, the database owner sends matrices $A$ and $B$ to the cloud. If RSA key is of the size 2048 bits then the size of matrix $B$ is $2048(l^2 - l)$ bits. If the size of the key in AES is 256, then the size of the matrix $A$ is $2(128)(l^2 - l)$ bits.

In phase 2, if $p$ and $q$ in Paillier are of size 1024 bits, utilizing Chang’s protocol, client sends $4096 \times 2l$ bits to the

---

**PROCEEDING OF THE 22ND CONFERENCE OF FRUCT ASSOCIATION**

---------------------------------------------------------------------------- 221 ------------------------------------------------ ----------------------------
cloud and the cloud sends two results of size 1 KB to the client.

In phase 3, client needs to send one RSA encryption of size 2048 bits. Database owner does one decryption and sends 2048 bits to the client.

In phase 4, utilizing Chang’s protocol on the matrix $K'$, client first sends $4096 \times (l + l/8)$ bits of encrypted values to the cloud and later up to $(k-1) \times l/8 \times 4096$ bits of information to the cloud. The cloud needs to send the PIR result of 1 KB, up to $k$ times. So $k$ KB is the amount of bandwidth usage for this phase.

VII. SECURITY AND PRIVACY ANALYSIS

In this section, we present the security and privacy analysis of our protocol. We analyse our protocol against semi-honest and malicious user, semi-honest and malicious cloud and semi-honest and malicious database owner.

All the discussions here are formulated with the assumption that the cryptosystems that are using (RSA, AES and Paillier cryptosystems) are secure.

We first show that any semi-honest or malicious cloud or database owner does not gain any information about the clients search nor the outcome of the query. However, the cloud will learn an upper limit for the length of the possible path.

This is due to the fact that, the protocol by Chang is based on Pailler cryptosystem. The cloud only receives encrypted zeroes and ones from the client. Therefore any cloud, malicious or semi-honest, can not learn anything about the choice of the client. The cloud will learn how many times the Chang’s protocol is initiated and thus an upper limit for the length of a possible path.

The database owner decrypts blindly the key that the client sends him. Because the decryption is done blindly the database owner does not learn anything about the result of the decryption.

Our protocol has the property that: the client only learns the shortest path between nodes A and B upon each query.

We can justify this by the following: The client can ask the database owner to decrypt one key $k_i$, This key can be used to decrypt one element from the matrix $A$. Thus the client can learn one new key and one node, the penultimate node in the path from $i$ to $j$. This new key again can be used only to decrypt one more element from the matrix $A$. This can be repeated until the revealed node is $j$ in which case no new key is learned. Thus all the keys the client learns are only good for revealing the path from node $i$ to node $j$.

Another property of our protocol is that the cloud does not learn anything about the structure of the reachability graph except an upper limit for the size of the graph and the length of the path that was queried.

This is due to the fact that the matrices that the cloud gets from the database owner are encrypted using RSA or AES. The size of the matrices only gives an upper bound for the number of user-host pairs and the number of nodes in the trust relation graph. The PIR between the client and the cloud prevents the client from learning what entries the client is interested in. The cloud only learns how many times the client initiates the PIR protocol and thus an upper limit for the length of the path.

VIII. CONCLUSION

In this paper, we presented a novel protocol that enables privacy preserving path queries on directed graphs. We used a realistic scenario to motivate our research problem, where the database where the graph derived from the database consists of trust relations. We extended the Floyd-Warshall algorithm to generate a matrix that holds the penultimate nodes of the shortest paths between each pair of nodes in our directed graph. We used Paillier homomorphic encryption scheme and Chang’s protocol to hide the queries.

Our protocol consists of three parties. The database owner sends the matrices $A$ and $K$ to the cloud. The client query the cloud to privately retrieve the shortest path between the nodes if there is a path and if a path exists, what the path is.

ACKNOWLEDGMENT

We thank the anonymous reviewers of 22nd Fruct Conference for their helpful comments. This work was supported in part by Tekes project “Cloud assisted Security Services”.

REFERENCES


