The Construction of Privacy-Preserving Protocols for Applications

CSIRO Seminar

Dr. Russell Paulet

17th Jan 2017

Outline

Introduction

- The goal of this talk is to give an overview of my research
- Simplifications have been made to make the content as accessible as possible
- Once we understand the basic idea, then we are able to look into more of the mathematical details
- The applications will help motivate the underlying theme

Basic Public key Encryption Definitions

- A public key encryption scheme is defined by three algorithms: a key generation algorithm (KG), an encryption algorithm (E), and a decryption algorithm (D)
- These functions are defined as follows
    - $(pk, sk) = KG(k)$: Takes as input a security parameter k, and outputs a public key pk and a secret key sk
    - $c = E(m, pk)$: Takes as input a message and a public key pk, and outputs a ciphertext c
    - $m' = D(c, sk)$: Takes as input a ciphertext c and a secret key sk, and outputs a message m'
- The public key encryption scheme is said to be correct if $m = m'$

Homomorphic Encryption

- Informally, the homomorphic property of an encryption scheme preserves some underlying structure
- This is often expressed mathematically as

$$E_k(a) \star E_k(b) = E_k(a * b) \tag{1}$$

For some binary operators $\star$ and $*$

- The consequence of such property is that we are able to operate the message even if it is encrypted!

Examples

- Usually, public key encryption schemes are either additively or multiplicatively homomorphic
  - (Additive - Paillier)

$$\begin{aligned} E(m_1)E(m_2) &= (g^{m_1} r_1^n)(g^{m_1} r_2^n) \\ &= g^{m_1+m_2}(r_1 r_2)^n \\ &= E(m_1 + m_2) \end{aligned} \tag{2}$$

  - (Multiplicative - RSA)

$$E(m_1)E(m_2) = m_1^e m_2^e = (m_1 m_2)^e = E(m_1 m_2) \tag{3}$$

- A scheme that is both additively and multiplicatively homomorphic is called a *fully homomorphic encryption scheme*

Examples

- The public key encryption scheme by Boneh-Goh-Nissim (BGN, for short) was the first scheme to support both addition and multiplication
- Suppose we are given two BGN ciphertexts $c_1 = g^{m_1} \cdot h^{r_1}$ and $c_2 = g^{m_2} \cdot h^{r_2}$, then we can compute the encrypted sum $m_1 + m_2$ as $c_1 \cdot c_2$
- Furthermore, we are able to compute the encrypted product $m_1 \cdot m_2$ as $e(c_1, c_2)$
- Where e is a map that (at least) satisfies the bilinear property

$$e(P^a, Q^b) = e(P, Q)^{a \cdot b} \tag{4}$$

- Unfortunately we are only able to multiply once due to the operation of the bilinear map

Examples

- The first scheme that was first to support arbitrary computation on encrypted data was due to Gentry
- Followed a simple blueprint
  - Construct a somewhat homomorphic encryption scheme
  - Squash the decryption circuit
  - Bootstrap (create a self-sustaining process)
- A simple example of a somewhat encryption scheme has been introduced that uses only elementary number theory
  - Encryption is $c = m + 2r + pq$, where $m \in \{0, 1\}$
  - Decryption is $m' = (c \pmod p)(mod\ 2)$
- The message will decrypt correctly when the noise component is small enough

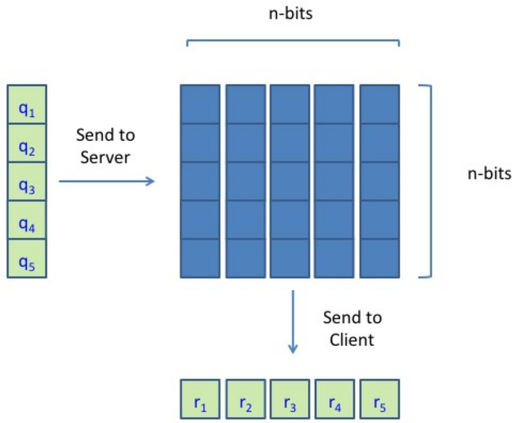
Private Information Retrieval

- Private Information Retrieval is a protocol where a client retrieves data from a server such that the server does not know which information was retrieved
- Private information retrieval protocols are subject to the constraint that the data communicated is strictly less than the total database size
- This constraint prevents the trivial solution of downloading the database and then searching it locally
- In these slides, we consider only computational private information retrieval as opposed to information theoretic private information retrieval

Example PIR

- The first computational PIR protocol to attain communication complexity less than the database size was given by Kushilevitz-Ostrovsky and is based on the Goldwasser-Micali encryption scheme
- The Goldwasser-Micali encryption scheme is known to be additively homomorphic (mod 2)
- In simple terms the Kushilevits-Ostrovsky PIR arranges the database into a $n \times n$ square
- The GM encryption scheme is used by the client to encrypt an n-bit array
- Then the server homomorphically adds this to the database and returns a new n-bit array to the client
- Based on this result the client is able to determine the bit at (i, j) is 0 or 1

Example PIR


n-bits
q1
q2
q3
q4
q5
Send to
Server
n-bits
Send to
Client
r1
r2
r3
r4
r5

Oblivious Transfer

- The concept of oblivious transfer was introduced by Rabin
- Oblivious transfer is similar in definition to PIR, as it requires that the privacy of the client is protected
- The definition of oblivious transfer also requires that the client can only retrieve one message (or record) from the server

Example

- The following example illustrates the general idea of oblivious transfer using the RSA cryptosystem
  1. Server sends the public key (e, N) to the client, along with two random numbers x_0 and x_1 to the client
  2. Client chooses a random number k and performs $v = k^e + x_b$, where $b \in \{0, 1\}$, and sends it to the server
  3. Server computes $k_0 = (v - x_0)^d$ and $k_1 = (v - x_1)^d$ and sends $m_0 + k_0$ and $m_1 + k_1$ to the client
  4. Client computes either $m_0 - k$ or $m_1 - k$, depending on his choice of b to receive the message
- The server is guaranteed that the client only received one message
- At the same time, the server is *oblivious* to which message was transferred

Applications

- Now we have reviewed the basic foundational theory we can review some examples of privacy-preserving applications
- We will consider two application domains
  - Location-based queries
  - Data warehouse queries

Private Location-Based Queries

- Location-based services provide a means for clients to access information about Points Of Interest (POIs)
- POIs can include restaurants, ATMs, hospitals and so on
- Disclosing the client's position can be very concerning as it gives a clear picture of where they have gone and how often
- So, we wish for the client to be able to interact with a LBS server, while concealing their location
- Likewise, the server wants to ensure that the client is appropriately accessing the data
- For simplicity, consider that the spatial domain for the user is arranged into a $n \times n$ grid (usually this is called the *Cloaking Region*)

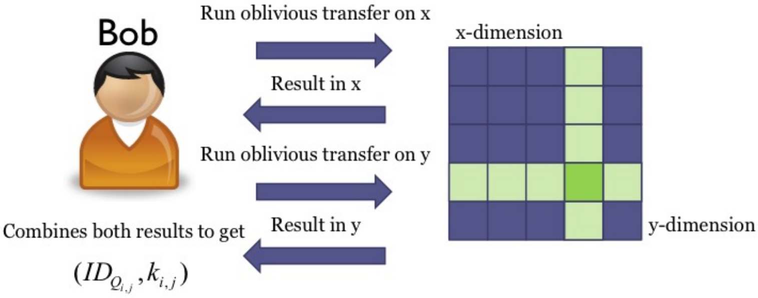
Private Location-Based Queries (Solution 1)

- This solution[1] is constructed in two phases
  - Oblivious transfer phase: This is used to obtain a key for an encrypted cell
  - Private information retrieval phase: This is used to retrieve the (encrypted) data, which can be decrypted using the key obtained from the previous phase

[1] R. Paulet, M. Golam Kaosar, X. Yi, E. Bertino, 'Privacy-Preserving and Content- Protecting Location Based Queries', IEEE Transactions on Knowledge and Data Engineering (TKDE), 2014

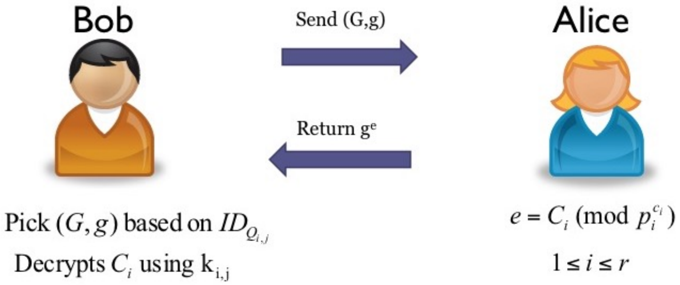
Private Location-Based Queries (Solution 1)

Oblivious transfer phase



Private Location-Based Queries (Solution 1)

Private information retrial phase



Private Location-Based Queries (Solution 2)

- This solution[2] uses an improved construction to the previous solution
- The improved construction allows greater integration, such that the two phases can be combined into one very naturally

[2] X. Yi, R. Paulet, E. Bertino, V. Varadharajan, 'Practical Approximate k Nearest Neighbor Queries with Location and Query Privacy', IEEE Transactions on Knowledge and Data Engineering (TKDE), 2016

Private Location-Based Queries (Solution 2)

- We will look at the simplest version of the protocol to give an idea of the general approach
- Suppose that the client wants to retrieve cell (i, j)
  - **Query Generation**: For $\ell \in [1, n]$ compute

$$c_\ell = \begin{cases} E(1, pk) = g^1 \cdot r_\ell^N \ (mod \ \ N^2) & if \, \ell = i \\ E(0, pk) = g^0 \cdot r_\ell^N \ (mod \ \ N^2) & otherwise \end{cases} \quad (5)$$

  - **Response Generation**: For $\gamma = [1, n]$ compute

$$C_\gamma = \prod_{\ell=i}^{n} c_\ell^{d_{\ell,\gamma}} \ (mod \ \ N^2) \quad (6)$$

  - **Response Retrieval**: Using the sk, the client can retrieve the data as

$$d = D(C_j, sk) \quad (7)$$

Private Data Warehouse Queries

- The primary purpose of a data warehouse is to support making decisions
- The data is organised into a multi-dimensional hypercube, where every cell contains one or more measures
- There are a number of operations that can be performed on this multi-dimensional data cube
  - **Roll-up** Example: aggregate from month to year
  - **Drill-down** Example: drill-down from year to month
  - **Slice** Select one dimension of the data cube
  - **Dice** Select two or more dimensions of the data cube
  - **Pivot** Change the orientation of the data cube
- Fundamentally, we need to support these in a privacy-preserving solution

Private Data Warehouse Queries (Solution 1)

- This solution[3] uses the BGN cryptosystem
- Suppose that the data warehouse is represented as $D(x_1, x_2, \ldots, x_m)_{y_1, y_2, \ldots, y_n}$
- Then, we encrypt using the BGN cryptosystem as

$$z = E(x, PK) = g^x \cdot h^r \tag{8}$$

  which is sent to the client
- By using the homomorphic properties of the BGN cryptosystem we can support Roll-up, Drill-down, Slice, Dice, and Pivot

[3] X. Yi, R. Paulet, G. Xu, E. Bertino, 'Private Data Warehouse Queries', 18th ACM Symposium on Access Control Models and Technologies (SACMAT), June 12-14, 2013, Amsterdam, The Netherlands

Private Data Warehouse Queries (Solution 2)

- One problem with the BGN cryptosystem is that decryption is slow
- This is because decryption requires the computation of the discrete logarithm
- We overcome this limitation by employing Paillier instead[4], which has a deterministic decryption procedure

[4] X. Yi, R. Paulet, E. Bertino, G. Xu, 'Private Cell Retrieval From Data Warehouses', IEEE Transactions on Information Forensics and Security, 2016

Conclusions

- In summary, we have built privacy-preserving solutions on the properties of homomorphic encryption
- This allows us to manipulate data without complete access
- Ideally, we want a complete set of operations on encrypted data, but this is presently known to be expensive in general
- We must balance practicality and utility (based on desired security level)

Questions

- Any questions?