An Ideal-Security Protocol for Order-Preserving Encoding

Klev

October 3, 2015

1 Abstract

The sort order of the ciphertexts matches the sort-order of the corresponding plaintexts. Reveals no information about the plaintext, other than the order. The main technique is mutable ciphertexts. This means that, over time, the ciphertexts for a small number of plaintext values change. The resulting protocol is somewhat interactive supporting a given set of interactions. 1-2 orders of magnitude securer than the state-of-the-art OPE schemes.

2 Introduction

Common operation: Order comparison, which is used for range checks, ranking, sorting, etc. To achieve this, many systems uses OPE where $Enc(x) > Enc(y) iff x > y$. OPE is appealing because applications can perform order procedures on the ciphertexts in the same was as on plaintexts. The security goal of OPE is IND-OCPA, where we reveal no information about the plaintext values besides the order.

First paper to present OPE where only the order of the encrypted values are revealed, in other terms: No leakage! Which is nice. Acceptable for the protocol to be mutable, in other terms change already encrypted ciphertext values as new plaintexts are encrypted.

Mutability is required to be able to obtain ideal security. Same-time OPE security (stOPE) where the application can learn the order of data items that are stored in the application at the same time. Stronger than IND-OCPA.

mOPE: Builds a balanced tree for searching which contains all the plaintexts values that are encrypted by the application. The OPE of a value is the path from root to that particular value in the search tree. If y is greater than x, then y will be located to the right of x in the search tree.

Paths are described using binary encoding where the length is equal to the depth of the three. Increases from left to right.

The search three is stored on the untrusted database server, the proxy server then encrypts values into the search tree by using an interactive protocol.

Transformation summary technique to update mutable ciphertexts in the search tree.

3 OPE Schemes

IND-OCPA equals ideal security for OPE schemes indicating that the adversay with a set of ciphertexts cannot learn anything about the corresponding plaintexts, except for the order. No existing schemes achieved it.

mOPE is the first to achieve IND-OCPA.

Previous assumption: Ciphertexts are immutable. Fuck that, let them be mutable (malleable).

4 Model

Client-server architecture that interact which each other. The client sits on the data that it wants to encrypt. Same structure as in CryptDB. Two problems: Same as CryptDB. Curious managers and adversaries with access.

5 Cryptographic preliminaries and notation

$d \leftarrow D$, where d is sampled from random distribution D.
$d \leftarrow S$, where d is sampled from an uniform distribution over a set S.
Security parameter is k.
RND is standard IND-CPA 2 symmetric key encryption scheme. KeyGen, ENC, DEC
DET is a standard PRF. KeyGen, ENC, DEC

6 Mutable Order-preserving Encoding (mOPE)

Example: 69, 32, 20, 10 and 25. Ideal order encryption: 5, 4, 2, 1, 3. The key is that we do not know anything about the future values, so encrypting 69 with 5 is not a good idea. Solution: Let the client read previously encrypted values

of its choosing stored at the database, and be able to modify a small portion of them if necessary.

Tree construction is a binary search tree. Values to the right is strictly larger than the value on its left. Implementation uses a b-ary B-tree. Many advantages, logarithmic worst-case runtime.

How to: Each node of the three contains the encryption of the value using DET and the client's secret key. Nodes are ordered based on the plaintext value. Stored in the database. The client will help the server in inserting the value into the search tree, as it is encrypted and impossible for the server to figure out on its own.

How to insert: Client gets root node, decrypts and request the node on next level if the server goes... left or right depending on the decrypted result. This little "game" goes all the way down until the client gets "no child". Insert new DET node. Path for root node is an empty string. Padding is added to all paths.

$OPEEncoding = [path]10..0$

Tree balancing to keep the tree from growing too deep. Split child node in two, and then proceed upwards if necessary. Must update storage containing OPE encodings also.This might take some time.

Transformation summary to the rescue. Describe tree balancing operations with a short O(log n) summary. Done completely by the server.

When inserting a new data item into the binary tree, the server rebalances the tree, and the OPE table is updated mapping the DET encryption of the value to a new or updated OPE encoding.

7 Syntax and correctness

Encoding of a value can be of two types: Permanent ciphertext c or a mutable encoding e that changes as the binary tree grows. Security definition of mOPE: Must not leak anything else than the order of the elements, definition is IND-OCPA presented in Boldyreva.

Security definition + proof

8 Impossibility of Non-Mutable Stateful OPE

This scheme achieves IND-OCPA, because of its use of ciphertext mutability. Boldyreva showed that for a OPE scheme to be IND-OCPA, the length of the ciperhtexts need to be exponential to the size of the plaintext, which is impractical. IND-OCPA stateless / stateful scheme is infeasible. *Proofs and theorems*

9 Storage-Aware Order-Preserving Encoding

IND-OCPA = Leaks only order. Ideally, the system should only leak the order of the items currently stored in the database. If one uses constant values in a query, the server should learn order of items stored with respect to the query, but it should not store the value to learn previous or future values stored in the database. We call this same-time OPE security. Reveals order of items stored in the database at that particular point of time.

Storage-aware OPE (stOPE) achieves this particular security. This idea is not captured in the original definition as it was something that was not achievable in the previous approaches. It was simply not possible to indicate when an data item was removed using the standard interfaces of the existing encryption scheme.

Initialization init: Client generates secret key s_k as part of initial state.
Insert(v): Client inserts encoding of $v \in D$ in the server. Obtains encryption c.
Remove(v): Value v removes from server if it exists. Client outputs c.
Query(v): Returns tightest interval around v. (v_1, v_2) and bit b true if $v = v_1$
Order(c): Takes a ciphertext as input, returns OPE e.

stOPE Scheme:

Ensure that the order of current values is preserved: delete values from the OPE tree when they are deleted from the database. Can not use DET encryption as it leaks information about equal, encrypted values. We only want to leak the order of the elements on the server right now. Uses RND encryption in stead.

10 Malicious (Active) Server

Malicious server: May alter responses to client. Not delete values from OPE tree. etc.

Force server to behave: add Merkle hashing on top of OPE tree and use it to check correctness based on server's response. Merkle tree: Each node is labelled with the hash of its children nodes. the server must "prove" using Merkle hashes

that the requested queries have been executed properly.

Proof consists of the old merkle information and the new merkle information.

Client checks by computing the root of the merkle tree based on its own view of the tree and the information delivered from the server and then compares the two results. Stores the correct result as its "view of the tree".

11 Using mOPE in a Database Application

Setup: Trusted client-side application using mOPE to encode values. Unmodified applications can use THE PROXY! UDFs in the database server that implement mOPs order function and update encodings.

The client application maintains separate mOPE client state(secret key) for every column that is encrypted with OPE.

Insert query: ¡INSERT INTO secret VALUES (5)¿
1) Encrypt 5 using mOPE.Enc
2) Obtains $c \leftarrow DET.Enc(5)$
3) INSERT INTO secret VALUES (MOPE_ORDER(c)), where MOPE_ORDER is a UDF-implementation of mOPE.Order
4) Transformation summary: If necessary, mOPE rebalances the tree. Must also update OPE encodings stored in the database.

Transformation summary: Sequence of split and merges.

Height of tree is bounded by $\log(N)$.
Magic: UPDATE secret SET item=MOPE_TRANSFORM(item, summary) WHERE item>=low AND item<=high, where MOPE_TRANSFORM is a UDF adjusting encodings based on transformation summary.

12 Conclusion

mOPE uses the idea of mutable ciphertexts.
Propose same-time OPE (stOPE) allows adversary to only learn order of elements that are currently in the database.