Efficient Similarity Search over Encrypted Data

Mehmet Kuzu, Mohammad Islam, Murat Kantarcioglu
Department of Computer Science, University of Texas at Dallas

Introduction

- Sensitive information is increasingly outsourced into the cloud in encrypted form.
- Encrypted storage complicates the computation on the data such as similarity search.
- Available searchable encryption schemes have been designed for only exact query matching .
- Secure multi-party computation based sophisticated cryptographic techniques enable similarity search but they are not practical.

Motivation

- Efficient similarity search have been extensively studied for plain data and one of the best solution is locality sensitive hashing (LSH).
- Appealing properties of LSH can be used in the context of encrypted data via a secure index.
- Secure index should provide some guarantees to ensure confidentiality of the sensitive data.

Security Definition

- Access Pattern (Ap): Identifiers of data items that are in the result set of a specific query.
- Similarity Pattern (Sp) : Relative similarity between distinct queries.
- History (H_n:) Data collection and query set for n consecutive queries.
- Trace(Y) : Maximum amount of information that data owner allows the leak to an adversary (encrypted data collection (C), Ap and Sp).
- View(V) : Information that is accessible to the adversary (C ,query trapdoors).

$$Pr[Dist(V(H_n)) = 1] - Pr[Dist(S(\gamma(H_n)) = 1] < \frac{1}{poly(s)}$$

Similarity Searchable Encryption

- Keygen(s): Output a private key K according to security parameter s.
- Enc(K, D_i): Encrypt D_i with key K to form C_i.
- Dec(K, D_i) Decrypt C_i with key K to obtain D_i.
- BuildIndex(K, D) Extract feature set for each D_i (F_i) in the data collection and form secure index I with the extracted features.
- Trapdoor(K, f) Generate a trapdoor for a specific feature f with key K and output T.
- FuzzySearch(I, T) Perform search on I with trapdoor of feature f (T) and output encrypted collection C with high probability:

$$C_j \in C \quad if \quad \exists (f_i \in F_j)\ [dist(f_i, f) \leq \alpha]$$
$$C_j \notin C \quad if \quad \forall (f_i \in F_j)\ [dist(f_i, f) \geq \beta]$$

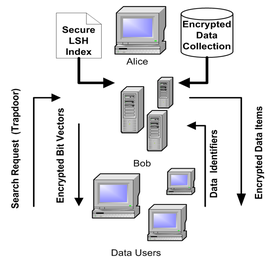
Secure LSH Index

- Feature Extraction : Extract features that characterizes sensitive data item D_i.
- Metric Space Translation: Map features to vectors in a metric space to apply LSH.
- Bucket Index Construction: Build a locality sensitive function g : (g_1, g_2, …, g_λ) from a family and apply g on feature vectors to form LSH buckets. Content of any bucket B_k is a bit vector (V_{B_k}) :

$$V_{B_k}[id(D_z)] = 1 \quad if \quad g_i(\vec{f_j}) = B_k \ for \ g_i \in g,\ f_j \in D_z$$
$$V_{B_k}[id(D_z)] = 0 \quad otherwise$$

- Bucket Index Encryption: Let Enc_{id} be a pseudorandom permutation, $Enc_{payload}$ be a secure encryption scheme, B_k be a bucket with content V_{B_k} and I be the secure index, then [$Enc_{id}(B_k)$, $Enc_{payload}(V_{B_k})$ ϵ I.

Secure Search Scheme



Shared Secret

- K_{coll} : Secret key of data collection encryption
- K_{id}, $K_{payload}$: Secret keys of index construction
- ρ: metric space translation function
- g: locality sensitive function

- Trapdoor Construction: Trapdoor for any feature f_i (T_{f_i}) is formed as follows:

$$T_{f_i} = \{Enc_{id}(g_1(\rho(f_i))), ..., Enc_{id}(g_\lambda(\rho(f_i)))\}$$

- Search: Retrieve encrypted bucket contents corresponding to components of the trapdoor, decrypt bit vectors and rank items:

$$V_{B_k} = Dec_{K_{payload}}(\sigma_{V_{B_k}}),\quad \pi_{B_k} \in T_{f_i}\ \ \&\&\ \ [\pi_{B_k}, \sigma_{V_{B_k}}] \in I$$
$$score(id(D_j)) = |Z_j| \quad s.t.\ \ V_{B_k} \in Z_j \ iff\ V_{B_k}[id(D_j)] = 1$$

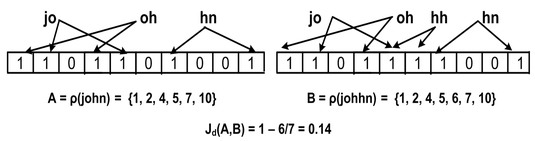
Ranking depends on the number of common buckets between compared features.

- Data Decryption: Retrieve encrypted data items with highest scores and decrypt them with K_{coll}.

Error Aware Keyword Search

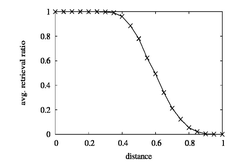
- Bloom filter encoding enables efficient space translation for approximate string matching.
- Elegant locality sensitive family has been designed for Jaccard distance (MinHash) that is [r_1, r_2 ,1-r_1, 1-r_2] sensitive.
- [r_1, r_2 , p_1, p_2] sensitive family can be formed from MinHash via AND-OR construction:

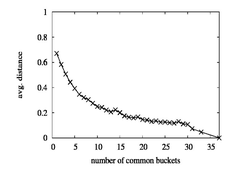
$$p_1 = 1 - (1 - (1 - r_1)^k)^\lambda, \quad p_2 = 1 - (1 - (1 - r_2)^k)^\lambda$$

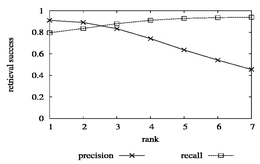


Experimental Evaluation

- A sample corpus of 5000 emails is constructed from publicly available Enron e-mail dataset.
- Words in e-mails are embedded into 500 bit Bloom filters with 15 hash functions.
- Common typos are introduced into the queries. (0.45, 0.8, 0.85, 0.01)-sensitive family is formed from MinHash to tolerate typos. LSH buckets are encrypted with AES in CTR mode.

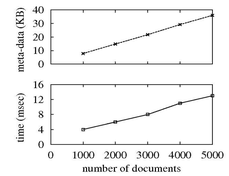


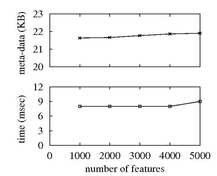




Default Parameters

- Number of documents: 5000
- Number of features: 5000
- LSH Parameters: k = 5, λ = 37





Conclusion

- We proposed LSH based secure index and search scheme to enable fast similarity search over encrypted data.
- We provided a rigorous security definition and proved the security of the scheme to ensure confidentiality.
- Efficiency of the proposed scheme is verified with empirical analysis.

This work has been published in IEEE ICDE 2012 Conference Proceedings.