PURDUE UNIVERSITY

Privacy-Preserving Complex Query Evaluation over Semantically Secure Encrypted Data

Bharath K. Samanthula, Wei Jiang, and ***Elisa Bertino***

CERIAS
Center for Education and Research in Information Assurance and Security

Outline

- Motivation
- Problem Statement
- Related Work & Background
- Proposed Solution
- Complexity Analysis
- Conclusions/Future work

Outline

- Motivation
- Problem Statement
- Related Work & Background
- Proposed Solution
- Complexity Analysis
- Conclusions/Future work

Cloud Computing

- Need for outsourcing
  - data and computations as well
  - useful for data owners' with limited/no resources
- Key challenges
  - data are typically *encrypted* before outsourcing
  - efficiency of data management is a major requirement

Query Processing over Encrypted Data

- Privacy Requirements:
  - user's query should not be disclosed
  - confidentiality of outsourced data
- The important question is: "how can the cloud perform searches over encrypted data without ever decrypting them or compromising the user's privacy"
- Lead to new research: *privacy-preserving query evaluation over encrypted data* (PPQED)

Three Possible Approaches

1. Download the entire encrypted database
   – not practical, incurs heavy costs on user
2. Secure Co-processors (e.g., IBM’s 4764)
   – expensive, may not be meant for clouds
   – needs verification by users or a trusted third party
   – may not be affordable for small businesses
3. Custom-designed cryptographic methods
   – problem-specific cryptographic solutions
   – our work is based on this approach

Processing Complex Queries

- Existing PPQED methods are too specific (e.g., range and aggregate queries)
- Recent approaches: try to support complex queries, but are insecure / not feasible
- ***Our focus: A PPQED framework that can securely evaluate complex queries and is efficient from the user's perspective***

Outline

- Motivation
- Problem Statement
- Related Work & Background
- Proposed Solution
- Complexity Analysis
- Conclusions/Future work

System Model

- Three Entities:
  - The data owner (Alice)
  - The cloud service provider
  - The data consumer (Bob)
- *Alice* wants to outsource its database T and query processing services to the cloud
- *Bob* wants to retrieve the data records of T stored in the cloud that satisfy its query Q

Problem Definition

- Alice holds $T = <t_1,\ldots, t_n>$, where each t_i,$1 \le i \le n$, is a database record and consists of m attributes
- Alice encrypts T attribute-wise and sends it to a cloud
- Bob issues a complex query Q to the cloud and wants to retrieve t_i's that satisfy Q.

Problem Definition (contd.)

- Q is defined as a query with arbitrary number of sub-queries where each sub-query consists of conjunctions and/or disjunctions of an arbitrary number of relational predicates
- $Q : G_1 \vee G_2 \vee \ldots \vee G_{l-1} \vee G_l \rightarrow \{0,1\}$
- G_j is a clause with a number b_j of predicates and is given by $P_{j,1} \wedge P_{j,2} \wedge \ldots \wedge P_{j,b_j-1} \wedge P_{j,b_j}$
- Eg: Q = ((Age ≥ 40) ∧ (Disease = Diabetes)) ∨

  ((Sex = M) ∧(Marital Status = Married) ∧

  (Disease = Diabetes))

Problem Definition (contd.)

- *Main goal of PPQED*: Facilitate Bob in efficiently retrieving from T' (encrypted version of T) the data records that satisfy Q in a privacy-preserving manner:

  $PPQED(T', Q) \rightarrow S$

  where $S \subseteq T$ denotes the output set of records that satisfy Q, $\forall\ t' \in S, Q(t') = 1$

Privacy Goals

- **Data confidentiality** of T (for Alice) at all times
- **Query Privacy** (for Bob)
  - S should be disclosed only to Bob
- T-S should never be disclosed to Bob and Alice
- **Privacy of data access patterns**: access patterns to data for any two queries Q and Q' should be indistinguishable to Cloud

Outline

- Motivation
- Problem Statement
- Related Work & Background
- Proposed Solution
- Complexity Analysis
- Conclusions/Future work

PURDUE UNIVERSITY

Comparison with Related work

Method	Low Cost On Bob	Data Confidentiality	Query Privacy	Hide Data Access Patterns	CNF and DNF Query Support
Golle et al. [1]	✘	✔	✔	✘	✘
Boneh and Waters [2]	✘	✔	✔	✘	✘
Popa et al. [3]	✔	✘	✘	✘	✔
This paper	✔	✔	✔	✔	✔

[1] Golle, P., Staddon, J., Waters, B., Secure conjunctive keyword search over encrypted data, In: ANCS, pp. 31-45, Springer (2004)
[2] Boneh, D., Water, B., Conjunctive, subset, and range queries on encrypted data, In: TCC, pp. 535-554, Springer (2007)
[3] Popa, R.A., Redfield, C.M.S., Zeldovich, N., Balakrishnan, H., Cryptdb: protecting confidentiality with encrypted query processing, In: SOSP, pp. 85-100, ACM (2011)

Adversarial Model

- Secure Multi-party Computation (SMC):
  - semi-honest model
  - malicious model
- Our work assumes the semi-honest model (existing approaches are also based on this model)
- Future Work: Extend our solutions to the malicious setting

PURDUE UNIVERSITY

The Paillier Cryptosystem

- Additive homomorphic and probabilistic encryption scheme
- (E_{pk}, D_{sk}): encryption and decryption functions
- Homomorphic addition: $D_{sk}(E_{pk}(x+y)) = D_{sk}(E_{pk}(x) * E_{pk}(y) \bmod N^2)$
- Homomorphic multiplication: $D_{sk}(E_{pk}(x * y)) = D_{sk}(E_{pk}(x)^y \bmod N^2)$
- Semantic security: Given a ciphertext, the adversary cannot deduce any information about the corresponding plaintext

Outline

- Motivation
- Problem Statement
- Related Work & Background
- Proposed Solution
- Complexity Analysis
- Conclusions/Future work

Federated Cloud Model

- Two non-colluding semi-honest cloud service providers, denoted by C_1 and C_2 (they together form a federated cloud)
- Alice generates (pk,sk), computes T' using pk and outsources it to C_1, where $T'_{i,j} = E_{pk}(t_{i,j})$, for $1 \leq i \leq n$ and $1 \leq j \leq m$
- She also outsources sk to C_2

Basic idea

- Divide and Conquer:
  - securely evaluate each predicate
  - securely combine the predicate results
- Key challenge:
  - to perform the above two tasks over encrypted data in a privacy-preserving manner

PURDUE UNIVERSITY

Secure Primitives

- **Secure Multiplication (SM):** C_1 holds $E_{pk}(a)$, $E_{pk}(b)$ and C_2 holds sk, it computes $E_{pk}(a*b)$
- **Secure Bit-OR (SBOR):** C_1 holds $E_{pk}(o_1)$, $E_{pk}(o_2)$ and C_2 holds sk, it computes $E_{pk}(o_1 \vee o_2)$
- **Secure Comparison (SC):** C_1 holds $E_{pk}(a)$, $E_{pk}(b)$ and C_2 holds sk, it computes $E_{pk}(c)$, where $c = 1$ if $a > b$ and $c=0$ otherwise. Here we assume $0 \leq a,b < 2^w$
- **Note**: the outputs are revealed only to C_1

PURDUE UNIVERSITY

Secure Multiplication

Require: C_1 has $E_{pk}(a)$ and $E_{pk}(b)$; C_2 has sk

1. **C_1:**
   (a). Pick two random numbers $r_a, r_b \in Z_N$
   (b). $a' \leftarrow E_{pk}(a) * E_{pk}(r_a)$
   (c). $b' \leftarrow E_{pk}(b) * E_{pk}(r_b)$; send a', b' to C_2
2. **C_2:**
   (a). Receive a' and b' from C_1
   (b). $h_a \leftarrow D_{sk}(a')$
   (c). $h_b \leftarrow D_{sk}(b')$
   (d). $h \leftarrow h_a * h_b \bmod N$
   (e). $h' \leftarrow E_{pk}(h)$; send h' to C_1
3. **C1:**
   (a). Receive h' from C_2
   (b). $s \leftarrow h' * E_{pk}(a)^{N-rb}$
   (c). $s' \leftarrow s * E_{pk}(b)^{N-ra}$
   (d). $E_{pk}(a * b) \leftarrow s' * E_{pk}(N - r_a * r_b)$

Evaluation of a Predicate

Let P: (k, α, op) be a predicate, where α denotes the search input, k denotes the attribute index, and op denotes the relational operator

> t_i satisfies the predicate P (i.e., $P(t_i)=1$) iff the relational comparison operation op on $t_{i,k}$ and α returns the Boolean value True.

Secure Evaluation of Individual Predicates (SEIP)

- For a given P (where the search input is in encrypted format), C_1 and C_2 have to securely compute $E_{pk}(P(t_i))$
- Two approaches:
  - – Homomorphic Encryption (HE)
  - – Garbled Circuits (GC)

HE based Solution ($SEIP_h$)

- Given $E_{pk}(t_{i,k})$ and $E_{pk}(\alpha)$, C_1 and C_2 need to compute $E_{pk}(c)$, where c =1 if $t_{i,k} > \alpha$, and c=0 otherwise
- Existing solution [4] leaks c to at least one party
- We extend the solution in [4] to compute $E_{pk}(c)$, without leaking c or any other information

[4] Blake, I.F., Kolesnikov, V., One round secure comparison of integers, Journal of Mathematical Cryptology, 37-38 (2009)

HE-based SC Protocol [4]

- C_1:
  - Compute the difference $E_{pk}(d_i) = E_{pk}(x_i - y_i)$
  - Compute the XOR $E_{pk}(z_i) = E_{pk}(x_i \text{ XOR } y_i)$
  - Compute encrypted vector γ such that $\gamma_i = 2y_{i-1} + z_i$, where $y_0 = 0$
  - Compute encrypted vector δ such that $\delta_i = d_i + r_i * (\gamma_i - 1)$
  - ***Observation:*** exactly one of the values of δ is 1 (denoting x>y) and the remaining are random numbers
  - Permute the encrypted vector and send it to C_2
- C_2:
  - Decrypt the vector and check whether any of the values is 1
  - If so, $x > y$. Otherwise, $x \leq y$
- **Note:** The comparison result is revealed to C_2

$SEIP_h$ (contd.)

- C_1 randomly selects a functionality F: $t_{i,k} > \alpha$ or $t_{i,k} \leq \alpha$
- C_1 and C_2 together run the SC protocol of [4] and the (oblivious) comparison result c' is known only to C_2
- C_2 encrypts c' and sends it to C_1
- Depending on F, C_1 computes $E_{pk}(c)$ from $E_{pk}(c')$

PURDUE UNIVERSITY

SEIP$_h$ (contd.) - details

C_1:

- chooses F randomly and proceeds as follows.
  - If $F : x > y$, compute $E_{pk}(d_i) = E_{pk}(x_i - y_i)$.
  - Otherwise, compute $E_{pk}(d_i) = Epk(y_i - x_i)$, for $1 \leq i \leq w$.
- computes the encrypted vector δ using the similar steps (as discussed above) in protocol [4].
- permutes the encrypted vector (denoted as v) and sends v to C_2.

C_2:

- decrypts the encrypted vector component-wise and finds the index k.
  - If $D_{sk}(v_k) = 1$, then compute $U = E_{pk}(1)$.
  - Else, i.e., when $D_{sk}(v_k) = -1$, compute $U = E_{pk}(0)$
- sends U to C1.

C_1:

- computes the output $E_{pk}(c)$ as follows.
  - If $F : x > y$, then $E_{pk}(c) = U$.
  - Else, $E_{pk}(c) = E_{pk}(1) * U^{N-1}$.

GC based solution ($SEIP_g$)

- C_1 (circuit generator) and C_2 (circuit evaluator) convert $E_{pk}(t_{i,k})$ and $E_{pk}(\alpha)$ into garbled values (as a part of circuit)
- C_1 and C_2 compare $t_{i,k}$ and α using the SC technique given in [5].
- The result is randomized (as part of the circuit) by a value known only to C_1. The randomized result (revealed to C_2) is encrypted and sent to C_1
- Finally, C_1 removes the random factor to get $E_{pk}(c)$

[5] Kolesnikov, V., Sadeghi, A.R., Schneider, T., Improved garbled circuit building blocks and applications to auctions and computing minima, CANS, 1-20, Springer (2009)

Proposed PPQED Protocol

- Stage 1 – Secure Evaluation of Predicates (SEP)
  - $SEIP_h$ or $SEIP_g$(depending on the domain size)
- Stage 2 – Secure Retrieval of Output Data (SROD)

PURDUE UNIVERSITY

A naïve solution ($SROD_b$)

- Use SM to evaluate each clause G_j
  - Given $E_{pk}(P_{j,h}(t_i))$, compute $E_{pk}(G_j(t_i)) = E_{pk}(P_{j,1}(t_i) \wedge \ldots \wedge P_{j,b_j}(t_i))$ using SM
- Use SBOR to compute final query result
  - Given $E_{pk}(G_j(t_i))$, compute $E_{pk}(Q(t_i)) = E_{pk}(G_1(t_i) \vee \ldots \vee G_l(t_i))$
- Expensive for large number of predicates and clauses

Our Solution ($SROD_s$)

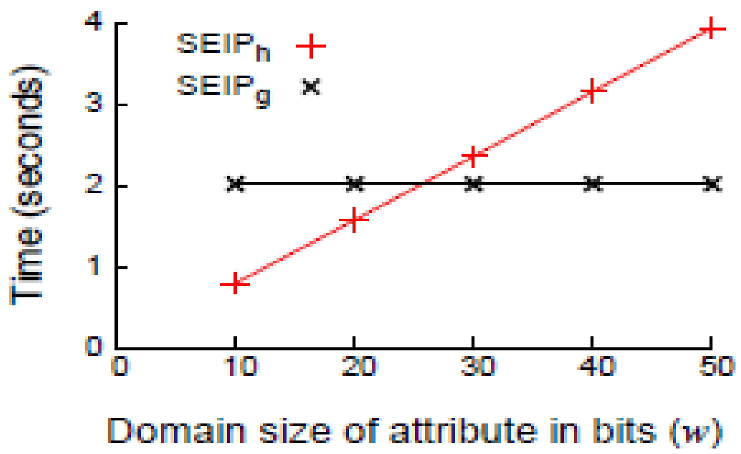
- To compute $E_{pk}(G_j(t_i))$:
  - Compute $E_{pk}(\Sigma_h P_{j,h}(t_i))$
  - Compare it with b_j using SC
  - **Key Observation**: $G_j(t_i) = 1$ iff $\Sigma_h P_{j,h}(t_i) = b_j$
- To compute $E_{pk}(Q(t_i))$:
  - Compute $E_{pk}(\Sigma_j G_j(t_i))$
  - Compare it with 0 using SC
  - **Key Observation**: $Q(t_i) = 1$ iff $\Sigma_j G_j(t_i) > 0$

Outline

- Motivation
- Problem Statement
- Related Work & Background
- Proposed Solution
- Complexity Analysis
- Conclusions/Future work

SEIP$_h$ vs. SEIP$_g$

- Implemented both using the Paillier Scheme
- Linux machine with Intel™ Xeon™ Six-Core® CPU 3.07 GHz processor, with 12 GB RAM, running Ubuntu 10.04 LTS



Encryption key size (*K*) is set to 1024 bits

$SROD_b$ Vs. $SROD_s$

- For any given data record t_i

Method	Computations	Communications
$SROD_b$	$O(l * s)$ encryptions	$O(K * l * s)$ bits
$SROD_s$	$O(l * \log_2 s)$ encryptions	$O(K * l * \log_2 s)$ bits

- *l*: number of clauses, *s*: upper bound on the number of predicates in each clause
- Our approach for SROD clearly outperforms the basic solution if *s* is large

Outline

- Motivation
- Problem Statement
- Related Work & Background
- Proposed Solution
- Complexity Analysis
- Conclusions/Future work

Summary

- A federated cloud framework that can support evaluations of complex queries in a privacy-preserving manner
- Hybrid solution: homomorphic encryption or garbled circuits
- Systematic approach to efficiently aggregate the predicate results
- Our approach guarantees data confidentiality and privacy of the user’s query

Future Work

- Implementation with MapReduce framework
- Extension to malicious setting
- In current work, we considered basic relational operators {<, >, ≤, ≥,=}
- Focus on other SQL queries, such as JOIN and GROUP BY, and evaluate their complexities

Thank You ☺

ANY QUESTIONS !!!

Appendix

Semantically Secure Encrypted Data

- Why semantic security?
  - data indistinguishability from cloud's perspective
  - ensures privacy of the user's data
  - users have more control over their data
- Example: the Paillier's encryption scheme

PURDUE UNIVERSITY

HE-based SC Protocol [4]

- ***Goal of SC***: Given that C_1 holds two integers $E_{pk}(x)$ and $E_{pk}(y)$, C_1 and C_2 jointly want to evaluate whether $x > y$.
- Existing SC protocols require encrypted bit representations as input rather than simple integers
- For this, we use secure bit-decomposition (SBD) [6,7]
  - convert $E_{pk}(x)$ to $<E_{pk}(x_1),\ldots, E_{pk}(x_w)>$
  - convert $E_{pk}(y)$ to $<E_{pk}(y_1),\ldots, E_{pk}(y_w)>$
  - x_1, x_w denote the most and least significant bits of x

[6] Samanthula, B.K., Jiang, W., An efficient and probabilistic secure bit-decomposition, ASIACCS, 541-546 (2013)
[7] Schoenmakers, B., Tuyls, P., Efficient binary conversion for Paillier encrypted values, Eurocrypt, 522-537 (2006)

PURDUE UNIVERSITY

HE-based SC Protocol [4]

- C_1:
  - Compute the difference $E_{pk}(d_i) = E_{pk}(x_i - y_i)$
  - Compute the XOR $E_{pk}(z_i) = E_{pk}(x_i \text{ XOR } y_i)$
  - Compute encrypted vector γ such that $\gamma_i = 2y_{i-1} + z_i$, where $y_0 = 0$
  - Compute encrypted vector δ such that $\delta_i = d_i + r_i * (\gamma_i - 1)$
  - ***Observation:*** exactly one of the values of δ is 1 (denoting x>y) and the remaining are random numbers
  - Permute the encrypted vector and send it to C_2
- C_2:
  - Decrypt the vector and check whether any of the values is 1
  - If so, $x > y$. Otherwise, $x \leq y$
- **Note:** The comparison result is revealed to C_2

PURDUE UNIVERSITY

GC-based SC Protocol [5]

- The basic idea is to build a garbled circuit (by one party) that can perform bit-wise comparisons (i.e., between x_i and y_i) and outputs a carryout bit which is fed as an input to the next iteration (along with x_{i+1} and y_{i+1}).
- The second party evaluates this circuit using oblivious transfer protocols and gets the comparison result of $x > y$ as the final output.