How Practical Is Computable Encryption?

Ken Smith, M. David Allen, Andrew Sillers, Hongying Lan

The MITRE Corporation

Ameet Kini

GeoEye Corporation

703-983-6115 • kps@mitre.org
Outline

- **Background**
- **DataStorm project at MITRE**
- **Practical challenges:**
  - Handling non-executable SQL
  - Configuring the encrypted DBMS
  - Performance Results
Outsourcing Data Management: The Promise And The Problem

- Outsourcing to 3rd party clouds promises:
  - Reduced need for internal IT & hardware
  - Cost efficiency of dealing with a high-volume provider
  - Novel services (e.g., AWS: S3, Elastic MapReduce, ...)

- However, outsourcing (sensitive) data means losing some control.
  - What if the cloud provider’s IT infrastructure is compromised?
  - What if their personnel leak or steal information?
How Do I Protect Sensitive Data In a Cloud?

- 1) Perform security engineering using COTS products and cloud-provider security services
   - Configure access controls, use at-rest encryption, etc.
   - E.g., Amazon Web Services’ “Virtual Private Cloud”

- 2) Use an encrypted blob

- 3) Using “computable encryption”
Hacking Into Oracle

Oracle Exploits / Exploit

This section "Oracle Exploits / Exploit" (or Proof-of-Concept code) contains information about Oracle security vulnerabilities in several products like database, webcache, tns listener, pls sql function, pls sql packages, forms, reports, isoqlplus, ora ... This is not illegal or dangerous. If your database or application server is hardened, all the exploits mentioned here are WITHOUT any effect. This page does not contain Oday exploits.

All exploit code on this website is already out there, e.g. in newsgroups, on websites (like bugtraq). Hacker and script kiddies are using such code every day. DBAs and security professionals like pentester or auditors should know how to escalate privileges, become DBA, become root, decrypt data, crash a database or doing a denial of service attack.

A lot of proof-of-concept code can be found in Metalink if you know how to search in Metalink. Red-Database-Security GmbH will soon publish a document how to find exploit code in the knowledge base of Oracle (Metalink).

Listener Exploits - Learn why it is important to protect your TNS Listener. With a few simple commands everyone (with listener access) can overtake the listener first and after that your database.

Oracle 8i Exploits - There are a still Oracle 8.1.7.4 instances out there (even if desupported). If you have an older version of 8i please try to update at least to 8.1.7.4 plus the latest security patchsets. Check the Critical Patch Updates on from secalert on a regular bases for additional information.

Oracle 9i Exploits - Many customers are still using 9.2.0.8. If you are not using the latest patchset / patchsets it is possible to become DBA with a single command (e.g. via CTXSYS.DRLOAD, DBMS_METADATA, DBMS_CDC_SUBSCRIBE).

Oracle 10g Exploits - More secure than 8i or 9i. Contains new features (like dbms_scheduler) with new security issues.

Oracle 11g Exploits. Latest version of the Oracle database

Oracle Application Server Exploits - Many software products like Oracle E-Business-Suite, Oracle Clinical, Oracle Collaboration Suite, custom development software ... are using OAS / IAS.

Oracle Application Express Exploits - The web application development tool APEX is used to develop and deploy applications that are hosted in the Oracle database.

Oracle Weblogic Exploits - WebLogic is a Java platform for developing, deploying, and integrating enterprise applications.

Other websites with Oracle exploit code
Oracle Metalink (Oracle Metalink account required)

New Features = New Exploits
German state buys Swiss bank data in tax evasion crackdown

The western German state of North-Rhine Westphalia has confirmed buying a CD with information on German tax evaders in Switzerland days before the finance ministers from both countries are set to meet on the issue.

Swiss banking secrecy has caused a spat with Germany

A spokeswoman for the North Rhine-Westphalia state government said it had received the data on Friday, February 26, but declined to reveal how much had been paid for the exchange.

German magazine Focus disclosed in early February that North Rhine-Westphalia state officials were negotiating a purchase of data on 1,500 German clients of Swiss banks from a whistleblower in France, who reportedly asked for some 2.5 million euros (US$3.4 million).
How Do I Protect Sensitive Data In a Cloud?

- 1) Perform security engineering using COTS products and cloud-provider security services
- 2) Use an encrypted blob
  - No plaintext data is visible in the cloud! AWS recommends this.
  - Good for a reasonably small corpus of documents.
  - But problematic in large databases if you only need to know the location of helicopter 21, or avg(age) in your company.
  - Returning too much data is worse still in a “Big Data” era!
- ... 3) Using “computable encryption”
What is “Computable Encryption”?

- Cryptosystems are valued primarily for securing information.
- As a side effect, some cryptosystems enable ciphertext operations corresponding to useful plaintext operations.
  - E.g., In the Paillier cryptosystem, modular multiplication of ciphertexts corresponds to the addition of plaintexts.
  - E.g., In any deterministic cryptosystem, equality tests on ciphertexts correspond to equality tests on plaintexts.
- Computable encryption is the use of such cryptosystems to compute (securely) directly over ciphertexts.

Diagram:

1. Data
2. $xzA4$ (Encrypt Sensitive Data)
3. $4Azx$ (Compute over the Encrypted Data)
4. NewData (Decrypt & Use the Result)
Craig Gentry’s Thesis

- Can a single cryptosystem enable all possible computation?
  - Arbitrary computations are possible (e.g., a Java program)
- This is known as the fully homomorphic encryption problem
  - Solved by Craig Gentry of Stanford (now IBM)
    - 2009 ACM dissertation of the year

- While possible and secure, Gentry’s work is about 10 billion times slower than computing over plaintexts
  - a 1 second execution would take 3 centuries
  - DARPA PROCEED focuses on speedup
Optimizing in 3D

Functionality (What functions can be computed)

Gentry’s Thesis is here

Security

Efficiency

If we accept less functionality (i.e., portions of SQL), can we achieve practical efficiency?
A Menagerie of Useful Cryptosystems Exist

- Probabilistic encryption (e.g., AES)
  - no computation enabled, but very secure and very efficient
- Deterministic encryption
  - =, ≠, COUNT, DISTINCT, IN, UNION, AND, …
- Order Preserving encryption (e.g., Aggrawal, Boldyreva)
  - >, <, <=, ORDER BY, MAX, MIN, …
- Additively homomorphic encryption (e.g., Paillier)
  - +, SUM
- And a few others …

This covers a large portion of SQL!
A Promising Architecture Exists

“Database as a Service” 2012 ACM Test of Time Award


100+ subsequent papers!

Important “flavors”:
- Which cryptosystems are used
- Bucketization, or not
- Post-processing, or not

Figure 1: The service-provider architecture.
Outline

- Background
- DataStorm project
- Practical challenges:
  - Handling non-executable SQL
  - Configuring the encrypted DBMS
  - Performance Results
DataStorm Project Objectives

1) Provide secure SQL* functionality by exploiting computable encryption

2) Without trusting the DBMS (or DBA) with keys or plaintext, and without altering the application or DBMS

3) With a “reasonable” performance hit

* Also key value store, and text repository …

Larger goal: surface & address barriers that keep computable encryption from being “shrink-wrapped”
DataStorm Architecture

- **Key components:**
  - IDEA
  - Migration tool
  - DataStorm Middleware
Outline

- Background
- DataStorm project
- Practical challenges:
  - Handling non-executable SQL
  - Configuring the encrypted DBMS
  - Performance Results
Barrier 1: Non-executable SQL Constructs

- We currently lack (efficient) cryptosystems to execute:
  - arbitrary substring matches
  - \( \sqrt{}, \cos(), /, \ln(), \ldots \)
  - fringe SQL: \texttt{today()}, bitwise or, etc.
  - many UDFs

- It is also impossible to execute:
  - “encryption type mismatch”
    - due to Raluca Popa (MIT – CryptDB)
  - operations disallowed by policy

SQL operations which cannot be implemented in computable ciphertext can be addressed by “post-processing”
DataStorm Post-Processing Architecture

- **Original Query (Q):**
  
  ```
  SELECT Name, (Salary*0.0833) 
  WHERE Salary < 100K AND Loc LIKE ‘%McLean%’
  ```

- **Query at encrypted DB (Q’):**
  
  ```
  SELECT *n*, (*s1* * *0.0833*),*l* 
  FROM *e* 
  WHERE *s2* < *100K* 
  ```

- **Post-processing Query (Q’’):**
  
  ```
  SELECT col1, col2 
  FROM Result 
  WHERE col3 LIKE ‘%McLean%’
  ```
DataStorm Execution Trace
(ICDE 2012 DemoTrack)
This is a “plaintext” query that an application would execute.
Get all flights departing from an airport starting with “MI” where the carrier is DAL.
This is the re-written “encrypted” query that the server actually executes. Note that the server doesn’t even know the names of the tables or columns. The value “DAL” will be passed encrypted.
The server executes the query and returns encrypted data.
Back on the client-side, the data is decrypted back to plaintext.
In post-processing, we must further filter the results to only those where the departing airport was like “MI%”
Finally, the application gets the correct answer to the query back, in decrypted form, with the proper metadata.
Outline

- Background
- DataStorm project
- Practical challenges:
  - Handling non-executable SQL
  - Configuring the encrypted DBMS
  - Performance Results
Barrier 2: DBA’s Should Not Need Cryptography Skills

- Many potential plans may exist to execute a given workload
  - ... each with a different security/performance profile
  - a complex “Pareto Frontier” of options exists

- Users who simply want more secure query execution can’t be expected to:
  - a) analyze database query workloads
  - b) understand cryptography

- Solution: help users quickly assess deployability!
  - and easily (or automatically) configure encryption choices
IDEA

- Interactive Database Encryption Advisor (IDEA)
  - examines plaintext schema and local workload
  - helps users understand & visualize tradeoffs in encryption strategies
  - recommends & enables interactive refinement
  - generates encryption map

- The database migration tool uses the encryption map to generate an encrypted database
IDEA Interface

- Plaintext Schema Information
- Query Workload Sandbox
- Auto-generate a Maximal Push Encryption Plan
- Auto-generate a All Client Encryption Plan
- Ciphertext Schema Generation
- View Impacts of Security Choices
- Working Query Display
- View Plan Graph for Working Query

Encryption Choices:
- Active Schema
  - Table DS_NFDC_APT_load
  - Table DS_ETMS2ASQP
  - Table DS_ETMS_FLIGHT
  - Table DS_ASQP_ADDEDVALUE
  - Table DS_ACFRAMEDATA_load
  - Table DS_ETMS_FLIGHT
  - Table DS_ETMS2ASQP_LOAD
  - Table DS_NFDC_APT
  - Table DS_ASQP_ADDEDVALUE_load
  - Table DS_ACFRAMEDATA

Choose Query
- Basic Query
- Column in Post
- Count [With Alias]
- Count(*) [With Count(*)]
- Distinct
- Equality
- Group By
- In Statement
- Inequality
- Like (Not Using %)
- Like (Using %)
- Numeric Range
- Order By
- Relational Join
- Select *
- Subquery
- Sum

Current Working Query:
```
SELECT f.flightid, f.carrier, f.flightno, f.deptairport, f.arrairport
FROM ds_etms_flight f;
```
Encryption Choices for Recommended “Maximal Push” Plan

DataStorm recommends DETERMINISTIC on column ds_etms_flight.carrier because this is the strongest available choice to support computing your query workload over ciphertext.

Current Working Query

```
SELECT f.flightid,
    f.carrier,
    f.flightno,
    f.deptairport,
    f.arrairport
FROM ds_etms_flight f
WHERE f.carrier = 'DAL';
```
IDEA Demo

Encryption Choices

- Active Schema
  - Table DS_NFDC_APT_LOAD
  - Table DS_ETMS2ASQP
  - Table DS_ETMS_FLIGHT_load
  - Table DS_ASQP_ADDEDVALUE
  - Table DS_ACFRAMEDATA_load
  - Table DS_ETMS_FLIGHT
    - FLIGHTID DOUBLE [Red]
      - Choices: Order-Preserving
    - ACID VARCHAR2 [Red]
    - CARRIER VARCHAR2 [Red]
      - Choices: Deterministic
    - FLIGHTNO VARCHAR2 [Red]
      - Choices: Probabilistic
    - DEPTAIRPORT VARCHAR2 [Red]
      - Choices: Probabilistic
    - ARRAIRPORT VARCHAR2 [Red]
      - Choices: Probabilistic
    - DEPTDATE DATETIME
    - ARRDATE DATETIME
    - STATUS CHAR
    - AC_TYPE VARCHAR2
      - Table DS_ETMS2ASQP_LOAD
      - Table DS_NFDC_APT
      - Table DS_ASQP_ADDEDVALUE_load
      - Table DS_ACFRAMEDATA

Choose Query

Reference Set
- Basic Query
- Column in Post
- Count [With Alias]
- Count(*) [With Distinct]
- Group By
- In Statement
- Inequality
- Like (Not Using %)
- Like (Using %)
- Order By
- Relational Join
- Select *
- Subquery
- Sum

Workload
- Equality
- Numeric Range

Current Working Query

```
SELECT f.flightid, f.carrier, f.flightno, f.deptairport, f.arrairport
FROM ds_etms_flight f
WHERE f.flightid >= 20090101000000
  AND f.flightid <= 20090101000020;
```

Inspect Query Plan

Display Plan: Maximal Push

Show Graph
IDEA Demo

Choose Query

Reference Set
- Basic Query
- Column in Post
- Count [With Alias]
- Count(*) [With
- Distinct
- Group By
- In Statement
- Inequality
- Like (Not Using %)
- Order By
- Relational Join
- Select *
- Subquery
- Sum

Workload
- Equality
- Like (Using %)
- Numeric Range

Display Query Plan
- Display Plan: Maximal Push
- Show Graph

Encryption Choices

- Active Schema
  - Table DS_NFDC_APT_Load
  - Table DS_ETMS2ASQP
  - Table DS_ETMS_FLIGHT_load
  - Table DS_ASQP_ADDEDVALUE
  - Table DS_ACFRAMEDATA_load
  - Table DS_ETMS_FLIGHT
    - FLIGHTID DOUBLE
      - Choices: Order-Preserving
      - Reasons
    - ACID VARCHAR2
    - CARRIER VARCHAR2
      - Choices: Deterministic
      - Reasons
    - FLIGHTNO VARCHAR2
      - Choices: Probabilistic
      - Reasons
    - DEPTAIRPORT VARCHAR2
      - Choices: Probabilistic
      - Reasons
    - ARRAIRPORT VARCHAR2
      - Choices: Probabilistic
      - Reasons
    - DEPTDATE DATETIME
    - ARRDATE DATETIME
    - STATUS CHAR
    - AC TYPE VARCHAR2

- Table DS_ETMS2ASQP_LOAD
- Table DS_NFDC_APT
- Table DS_ASQP_ADDEDVALUE_load
- Table DS_ACFRAMEDATA
Outline

- Background
- DataStorm project
- Practical challenges:
  - Handling non-executable SQL
  - Configuring the encrypted DBMS
  - (Some Initial) Performance Results
MP vs AC: Indexed Retrieval

- **All client** (all post-proc) affected by large data volumes
  - could be optimized (HW decryption, caching, …)
- **Maximal push** (maximizes work at server) is stable and very fast
  - exploits DBMS index; < 2ms for 1M records (low crypto overhead)

<table>
<thead>
<tr>
<th>Fast Network</th>
<th>10k MP</th>
<th>10k AC</th>
<th>100k MP</th>
<th>100k AC</th>
<th>1000k MP</th>
<th>1000k AC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Time</td>
<td>1.64</td>
<td>218.2616</td>
<td>1.83</td>
<td>2468.49</td>
<td>1.88</td>
<td>27095.43</td>
</tr>
<tr>
<td>Exec + Xfer</td>
<td>1.61</td>
<td>21.3709</td>
<td>1.79</td>
<td>196.73</td>
<td>1.83</td>
<td>2861.65</td>
</tr>
<tr>
<td>Decrypt</td>
<td>0.0087</td>
<td>72.39</td>
<td>0.0088</td>
<td>831.58</td>
<td>0.0094</td>
<td>8987.58</td>
</tr>
<tr>
<td>Populate</td>
<td>0</td>
<td>118.2172</td>
<td>0</td>
<td>1374.42</td>
<td>0</td>
<td>14980.54</td>
</tr>
<tr>
<td>Client</td>
<td>0</td>
<td>6.2026</td>
<td>0</td>
<td>65.67</td>
<td>0</td>
<td>265.56</td>
</tr>
<tr>
<td>Execute Client</td>
<td>0</td>
<td>6.2026</td>
<td>0</td>
<td>65.67</td>
<td>0</td>
<td>265.56</td>
</tr>
</tbody>
</table>

SELECT p_name FROM part WHERE p_brand='Brand#43'
**MP vs AC: Paillier Summations**

- *Maximal push comparable to All client*
  - due to 13 microsecs / homomorphic addition
- Massive speedup possible via parallelism (e.g., Hbase)
  - compute avg. age in US in ~4 secs with 1000 processors

<table>
<thead>
<tr>
<th>Fast Network</th>
<th>10k MP</th>
<th>10k AC</th>
<th>100k MP</th>
<th>100k AC</th>
<th>1000k MP</th>
<th>1000k AC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Time</td>
<td>207.56</td>
<td>142.63</td>
<td>1344.63</td>
<td>1560.67</td>
<td><strong>13212.11</strong></td>
<td>16637.66</td>
</tr>
<tr>
<td>Exec + Xfer</td>
<td>131.83</td>
<td>14.44</td>
<td>1270.34</td>
<td>132.62</td>
<td>13138.74</td>
<td>1581.81</td>
</tr>
<tr>
<td>Decrypt</td>
<td>75.69</td>
<td>39.02</td>
<td>74.2451</td>
<td>407.12</td>
<td>73.32</td>
<td>4097.24</td>
</tr>
<tr>
<td>Populate</td>
<td>0</td>
<td>86.00</td>
<td>0</td>
<td>1001.67</td>
<td>0</td>
<td>10785.33</td>
</tr>
<tr>
<td>Client</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Execute Client</td>
<td>0</td>
<td>3.091</td>
<td>0</td>
<td>19.1662</td>
<td>0</td>
<td>173.19</td>
</tr>
</tbody>
</table>

SELECT SUM(p_size) as SUM_SIZE FROM part
Computable Encryption vs Plaintext: What Do You Pay?

Equality test on index query; answer is 4% of database; run remotely; times in ms

<table>
<thead>
<tr>
<th></th>
<th>PT total</th>
<th>CT (MP) total</th>
<th>CT/PT ratio</th>
<th>CT server</th>
<th>CT decrypt</th>
<th>CT misc</th>
</tr>
</thead>
<tbody>
<tr>
<td>10K</td>
<td>17.5451445</td>
<td>31.9446655</td>
<td>1.82</td>
<td>0.5521111111</td>
<td>0.716081</td>
<td>30.67647339</td>
</tr>
<tr>
<td>100K</td>
<td>64.098557</td>
<td>199.638414</td>
<td>3.11</td>
<td>15.73873737</td>
<td>5.051751</td>
<td>178.8479256</td>
</tr>
<tr>
<td>1000K</td>
<td>594.0421615</td>
<td>1944.573251</td>
<td>3.27</td>
<td>211.590899</td>
<td>40.4716245</td>
<td>1692.510728</td>
</tr>
</tbody>
</table>

- Unstable ratio: 82-227% overhead for ciphertext
- Network part of “CT misc” is dominating (40K records returned for 1000K DB)
  - “big data” gets even bigger with ciphertext
- Smaller answers and/or faster link are wins
### Equality Query Times (No Network)

Equality test on index query; answer is 4% of database; run on “localhost” to eliminate network

<table>
<thead>
<tr>
<th></th>
<th>PT total</th>
<th>CT total</th>
<th>CT/PT ratio</th>
<th>CT server</th>
<th>CT decrypt</th>
<th>CT misc</th>
</tr>
</thead>
<tbody>
<tr>
<td>10K</td>
<td>0.6286115</td>
<td>1.0018130</td>
<td>1.59</td>
<td>0.68953535</td>
<td>0.1819360</td>
<td>0.130341646</td>
</tr>
<tr>
<td>100K</td>
<td>3.2718125</td>
<td>7.1076260</td>
<td>2.17</td>
<td>5.31842424</td>
<td>1.2889455</td>
<td>0.500256260</td>
</tr>
<tr>
<td>1000K</td>
<td>220.8141110</td>
<td>369.9193835</td>
<td>1.67</td>
<td>356.80302020</td>
<td>12.7611650</td>
<td>0.355198300</td>
</tr>
</tbody>
</table>

- Nice stable ratio! 59-117% overhead w.o. network
  - (High server times at 1000K likely artifact of Postgres on a Windows laptop, but ratios are stable.)
- CT misc < .5ms and stable! Confirms *main* cost contributors are: a) server execution, b) network transfer, c) decrypt (not huge, for AES)
- Speedups?
  - Server parallelism, where possible, is an obvious speedup for CT server cost
  - HW decrypt for AES & ciphertext caching *could* help decrypt
### Summation Query Times (in ms)

Summation query using Paillier (additively homomorphic cryptosystem); server execution time only; Paillier using

<table>
<thead>
<tr>
<th></th>
<th>PT server</th>
<th>CT server</th>
<th>CT/PT ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>10K</td>
<td>1.49811111</td>
<td>111.969676</td>
<td>74.74</td>
</tr>
<tr>
<td>100K</td>
<td>14.0107676</td>
<td>1153.161051</td>
<td>82.31</td>
</tr>
<tr>
<td>1000K</td>
<td>169.018767</td>
<td>11356.5945</td>
<td>67.19</td>
</tr>
</tbody>
</table>

- Stable ratio: 67X to 82X. (1 to 2 orders of magnitude slower).
- Paillier sums done via UDAF written in C using Open SSL (modular multiplication)
  - about 15 μsec each
- Note that Paillier decrypts (not shown) require ~50ms/integer!
  - OK for: sum, avg, or a small GROUP BY
  - Not OK for answers involving e.g., 10,000 integers (500 seconds!)
- Clear “sweet spot” for Paillier: parallel integer summations, averages, BI rollups into small sets of numbers
Conclusions

- **Sweet spots** (promising use cases):
  - Simple equality test, range test lookups
    - “Where is helicopter 21?”, “Retrieve Bob’s medical record”
    - 60-120% overhead in our recent (unoptimized) tests
  - Addition-based math:
    - Small sum() & avg(); at scale with parallelism and/or HW assist
  - Setup & configuration “wizard” valuable to reduce complexity
    - Encrypted DB to address local security policy, query mix
  - Post-processing is useful in *specific* situations
    - Coping with occasional non-computable operation
    - Industrial strength planner needed to help know “when”
Acknowledgements

- Project contributors:
  - Current: M. David Allen; Andrew Sillers; Hongying Lan; William Wang; Kerry McKay
  - Alumni: Doug Burdick (IBM); Christopher Wolf (Microsoft); Ameet Kini (GeoEye); Adam Young (Bloomberg)

- Funding:
  - MITRE Innovation Program “Mission Assurance” Portfolio (Vipin Swarup): 19128626-AA

- … & many valuable discussions with colleagues
Backup
## Accumulo: Basic Searching #1

### Point lookups
- **English:** Show me watchlist passengers whose nationality is Iran
- **Accumulo:** Get me all records where CQ='NATION' and VAL='IRAN'
  
  (uses bundled iterator RegExFilter)
- **Datastorm:** Uses deterministic encryption (DET)

### Plaintext Table

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>ADDRESS</th>
<th>NATION</th>
<th>PHONE</th>
<th>EXPENSES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Customer#0000001</td>
<td>23rd St. Tehran</td>
<td>IRAN</td>
<td>25-989-741-988</td>
<td>712</td>
</tr>
<tr>
<td>2</td>
<td>Customer#0000002</td>
<td>2nd St. Lyon</td>
<td>FRANCE</td>
<td>23-768-687-3665</td>
<td>122</td>
</tr>
</tbody>
</table>

### Ciphertext Table

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>PHONE</th>
<th>EXPENSES</th>
</tr>
</thead>
<tbody>
<tr>
<td>119</td>
<td>iasdufiosda[ias</td>
<td>lowjecnwc</td>
<td>...</td>
</tr>
<tr>
<td>275</td>
<td>uioasufisafu[ias</td>
<td>sfsk[fsdfsdf</td>
<td>...</td>
</tr>
</tbody>
</table>
Accumulo: Basic Searching #2

- **Range lookups**
  - Show me watchlist passengers who have spent between $9000 and $9010
  - Accumulo: Get me all records where CQ='EXPENSES' and 9000 < VAL < 9010
    (uses custom iterator called RangeFilter)
  - Datastorm: Uses order preserving encryption (OPES)

### Plaintext Table

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>…</th>
<th>EXPENSES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Customer#0000001</td>
<td>…</td>
<td>5000</td>
</tr>
<tr>
<td>2</td>
<td>Customer#0000002</td>
<td>…</td>
<td>9002</td>
</tr>
<tr>
<td>3</td>
<td>Customer#0000003</td>
<td>…</td>
<td>9008</td>
</tr>
<tr>
<td>4</td>
<td>Customer#0000004</td>
<td>…</td>
<td>9010</td>
</tr>
</tbody>
</table>

### Ciphertext Table

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>…</th>
<th>EXPENSES</th>
</tr>
</thead>
<tbody>
<tr>
<td>179</td>
<td>lowjecnwoiwnjdu</td>
<td>…</td>
<td>23</td>
</tr>
<tr>
<td>234</td>
<td>sfsklfjksldfwiw</td>
<td>…</td>
<td>23328</td>
</tr>
<tr>
<td>314</td>
<td>iuofdgbufoley</td>
<td>…</td>
<td>25684</td>
</tr>
<tr>
<td>-583</td>
<td>iwnfgksiuwider</td>
<td>…</td>
<td>26738</td>
</tr>
</tbody>
</table>
Accumulo: Simple Analytic

- Finding co-travelers
  - Get me watchlist passengers who have either flown into or out of the same airport on the same data

  e.g., Bill and Gary flew out of Paris on Jan 1, 2012.

- Incorporating cell-level sensitivity labels
- Also observing linear speedup with parallelism (on summations)
Speeding up the Paillier Sum via Parallelism

DataStorm

Ciphertext SQL Query With Sum()

Single Node Ciphertext RDBMS

Paillier SUM() UDAF

Compute Node

Compute Node

Compute Node

Compute Node

Compute Node

Compute Node

Compute Node

Compute Node

Ciphertext Answer
Impact Summary

- DataStorm prototype transparently encrypts sensitive data so:
  - Existing SQL queries can still be executed
    - Existing applications & DBMS require no changes
  - Untrusted databases never see your data
    - If your cloud provider is compromised, attackers can only see your encrypted data
- IDEA prototype helps configure DataStorm
  - enables users to balance speed and security
- Performance studies show “sweet spots”
  - Indexed retrievals; massive summations

- This technology could radically change the cost/benefit tradeoff for outsourcing data, and protecting secure enclaves.
Two Encryption Lattices Exist!

Focus of NIST Approval

Paillier Additively Homomorphic

AES 256 Probabilistic

AES (Code Book) Deterministic

Boldreva Order Preserving Deterministic

Plaintext

Security of Encryption Algorithms

Obfuscated

Functionality of Encryption Algorithms

Obfuscated

Probabilistic

Deterministic

Additively Homomorphic

Order Preserving Deterministic

Plaintext

Enables Computation
# Demo: Regression Test

## DataStorm Regression Tests

<table>
<thead>
<tr>
<th>Test Type</th>
<th>SQL Query</th>
<th>View Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basic Query</strong></td>
<td><code>SELECT F.FLIGHTID, F.CARRIER, F.FLIGHTNO, F.DEPTAIRPORT, F.ARRAIRPORT FROM DS_ETMS_FLIGHT F</code></td>
<td>Time: 0.210 seconds / 2.490 seconds</td>
</tr>
<tr>
<td><strong>Equality</strong></td>
<td><code>SELECT F.FLIGHTID, F.CARRIER, F.FLIGHTNO, F.DEPTAIRPORT, F.ARRAIRPORT FROM DS_ETMS_FLIGHT F WHERE F.CARRIER='DAL'</code></td>
<td>Time: 0.046 seconds / 1.167 seconds</td>
</tr>
<tr>
<td><strong>Inequality</strong></td>
<td><code>SELECT F.FLIGHTID, F.CARRIER, F.FLIGHTNO, F.DEPTAIRPORT, F.ARRAIRPORT FROM DS_ETMS_FLIGHT F WHERE F.CARRIER &lt;&gt; 'DAL'</code></td>
<td>Time: 0.150 seconds / 1.547 seconds</td>
</tr>
<tr>
<td><strong>Numeric Range</strong></td>
<td><code>SELECT F.FLIGHTID, F.CARRIER, F.FLIGHTNO, F.DEPTAIRPORT, F.ARRAIRPORT FROM DS_ETMS_FLIGHT F WHERE F.FLIGHTID &gt;= 20090101000000 AND F.FLIGHTID &lt;= 20090101000020</code></td>
<td>Time: 0.031 seconds / 0.844 seconds</td>
</tr>
<tr>
<td><strong>Order By</strong></td>
<td><code>SELECT F.FLIGHTID, F.CARRIER, F.FLIGHTNO, F.DEPTAIRPORT, F.ARRAIRPORT FROM DS_ETMS_FLIGHT F ORDER BY F.DEPTAIRPORT</code></td>
<td>Time: 0.188 seconds / 1.813 seconds</td>
</tr>
<tr>
<td><strong>Distinct</strong></td>
<td><code>SELECT DISTINCT F.CARRIER FROM DS_ETMS_FLIGHT F</code></td>
<td>Time: 0.016 seconds / 0.829 seconds</td>
</tr>
<tr>
<td><strong>Group By</strong></td>
<td><code>SELECT F.CARRIER, COUNT(F.CARRIER) AS COUNT FROM DS_ETMS_FLIGHT F GROUP BY F.CARRIER</code></td>
<td>Time: 0.032 seconds / 0.88 seconds</td>
</tr>
<tr>
<td><strong>Like (not Using %)</strong></td>
<td><code>SELECT F.FLIGHTID, F.CARRIER, F.FLIGHTNO, F.DEPTAIRPORT, F.ARRAIRPORT FROM DS_ETMS_FLIGHT F WHERE F.DEPTAIRPORT LIKE 'MIA'</code></td>
<td>Time: 0.032 seconds / 2.76 seconds</td>
</tr>
<tr>
<td><strong>Like (Using %)</strong></td>
<td><code>SELECT F.FLIGHTID, F.CARRIER, F.FLIGHTNO, F.DEPTAIRPORT, F.ARRAIRPORT FROM DS_ETMS_FLIGHT F WHERE F.CARRIER='DAL' AND F.DEPTAIRPORT LIKE 'MIA'</code></td>
<td>Time: 0.032 seconds / 0.922 seconds</td>
</tr>
<tr>
<td><strong>In Statement</strong></td>
<td><code>SELECT F.FLIGHTID, F.CARRIER, F.FLIGHTNO, F.DEPTAIRPORT, F.ARRAIRPORT FROM DS_ETMS_FLIGHT F WHERE F.DEPTAIRPORT IN ('MIA', 'DAL')</code></td>
<td>Time: 0.032 seconds / 2.76 seconds</td>
</tr>
</tbody>
</table>
Probabilistic Encryption Systems

- With overwhelming probability, two encryptions of the same plaintext will yield different ciphertexts, for example:
  - \( \text{encrypt}_1(“Bob”) = “#ta$3b7” \)
  - \( \text{encrypt}_2(“Bob”) = “9nbM&*” \)

- Probabilistic encryption is necessary for strong semantic security:
  - Indistinguishability against chosen plaintext attacks (IND-CPA)
  - i.e., stealing the encryptor can’t help you break ciphertexts

- Most probabilistic systems provide no computational properties.
  - but still valuable (protecting fields not involved in computation)
    - SELECT emp.image FROM emp WHERE emp.id = 35

- A high profile example is AES (e.g., 256 bit key, in CTR mode)
  - NSA-approved (FIPS 197) for protecting classified information
  - AES now implemented in HW on Intel & AMD chips
Deterministic Encryption Systems

- A plaintext always results in the same ciphertext
  - \(\text{encrypt}_1(“Bob”) = “#ta$3b7”\)
  - \(\text{encrypt}_2(“Bob”) = “#ta$3b7”\)

- Two attacks on stolen deterministic ciphertexts:
  - a) create a histogram of the ciphertexts
    - If ~365 buckets it’s likely a date! Bucket height may imply weekends
  - b) steal encryptor, encrypt all values in likely domain, match to ID
  - Harder to attack when data is sparse wrt full domain (e.g., real #’s)

- Deterministic encryption is extremely useful computationally
  - permits testing equality, fundamental to many SQL operations
    - Projection, Equijoin, Group by, Count, Distinct, Set Operators (IN, Exists, Union ...), Boolean Operands (AND, OR, NOT), Selection predicates (=, <>)

- Example: AES in ECB (electronic code book) mode
Order Preserving Encryption (OPE)

- Plaintext order is preserved in ciphertexts
  - “Bob” < “Frank” \(\Rightarrow\) encrypt(“Bob”) < encrypt(“Frank”)

- OPE should be deterministic for data with duplicates
  - but can be probabilistic when values are distinct (e.g., UNIQUE)

- Attack on stolen ciphertexts (for deterministic OPE)
  - create sorted histogram; if 365 buckets, exact dates evident

- OPE is very useful for implementing computations over ciphertext:
  - range join, max, min, selection by <, >, ORDER BY
  - without OPE you default to full table scans a lot!

- Two published algorithms: Aggrawal ‘04, Boldyreva ’09
  - encrypt/decrypt is relatively slow (Boldyreva 32 bit encrypt = ms)
  - it is possible to obscure the value distribution (e.g., gaussian)
Additively Homomorphic Encryption

- Ciphertext operations correspond to plaintext addition!
  - e.g., decrypt (encrypt(3) \textit{op} encrypt(4)) = 7
- Several algorithms, but the Paillier algorithm stands out
  - semantically secure! encrypt_1(3) \neq encrypt_2(3) ... yet it still works
  - reasonable performance: MITRE benchmark of one “addition” in several microsecs
  - ciphertext modular multiplication corresponds to plaintext addition
- Using Paillier, we can implement further SQL operations:
  - a + b, SUM()
  - must call “PaillierSUM()” UDF instead of SUM()
Literature Survey

- Surveyed over 50 research papers.
- Outcomes:
  - Interest around the world: US, Europe, Iran, China, ...
  - Much was published in 2010, the field is heating up
  - Convergence: Crypto & DB communities beginning to talk
  - Linked up with groups at MIT and ETH (Switzerland)
- Product: 10 page sponsor-friendly paper

References
A More Insidious Problem

- What if you have very sensitive (e.g., classified) data
  - and *your own* IT infrastructure is compromised?
  - or *your own* IT personnel leak or steal information?

- We live in an era of very invasive attacks:
  - Stuxnet nuclear centrifuge attack, attack on 100,000 Sony Playstation accounts, Swiss Bank DBA bribery, recent attacks by “Anonymous”, and many more …
What are the Prospects?

Craig Gentry’s 2009 ACM Dissertation of the Year proved any computation is possible over ciphertext. Practical use of Gentry’s system is unlikely. (An 8 bit multiplication takes about 15 minutes).

However, many SQL operations can be executed at near normal speeds!

Performing all possible computations via ciphertexts

Performing all SQL computations via ciphertexts

Computing much of SQL by manipulating ciphertexts is now practical (see recent Forbes article)
Where is this “Encrypted Box”?  

- **A public cloud**
  - cheaper place to operate, desired by new policies

- **Your own secure enclave**
  - when your IT infrastructure is subject to sophisticated cyber attack
  - when your IT insiders might not be trusted
DataStorm: Securing Databases Through Encryption

Research Objectives

DBMS’s are prime targets of cyberattacks. DataStorm aims to enable DBMS computations over ciphertext datasets. Specifically this will:

1. Provide full DBMS functionality (e.g., for database as a service in a cloud provider)
2. Without trusting the DBMS (or the human DBA) with any plaintext, even in memory.
3. With negligible performance overhead

Current Activities

- Complete architecture: DataStorm + IDEA + Migration tool
- Outreach to multiple sponsors (e.g., VA, Ft Meade, AF Intel/Cyber), PoCs
- ICDE 2012 demo
- Performance studies and CIDR submission
- Accumulo version

![Diagram of DataStorm system](image)
IDEA Demo

Choose Query
Reference Set
Basic Query
Column in Post
Count [With Alias]
Count(*) [With Distinct
Group By
In Statement
Inequality
Like (Not Using %)
Like (Using %)
Numeric Range
Order By
Relational Join
Select *
Subquery
Sum

Workload
Equality

Current Working Query
SELECT f.flightid,
f.carrier,
f.flightno,
f.departport,
f.arrairport
FROM ds_etms_flight f
WHERE f.carrier = 'DAL';
IDEA Demo

Encryption Choices

Active Schema
- Table DS_NFDC_APT_load
- Table DS_ETMS2ASQP
- Table DS_ETMS_FLIGHT_load
- Table DS_ASQP_ADDEDVALUE
- Table DS_ACFRAMEDATA_load
- Table DS_ETMS_FLIGHT
  - FLIGHTID DOUBLE
    - Choices: Order-Preserving
  - ACID VARCHAR2
  - CARRIER VARCHAR2
    - Choices: Deterministic
  - FLIGHTNO VARCHAR2
    - Choices: Probabilistic
  - DEPTAIRPORT VARCHAR2
    - Choices: Probabilistic
  - ARRAIRPORT VARCHAR2
    - Choices: Probabilistic
  - DEPTDATE DATETIME
  - ARRDATE DATETIME
  - STATUS CHAR
  - AC_TYPE VARCHAR2

Choose Query

Reference Set
- Basic Query
- Column in Post
- Count [With Alias]
- Count(*) [With Distinct Group By In Statement]
- Inequality
- Like (Not Using %)
- Like (Using %)
- Order By
- Relational Join
- Select *
- Subquery
- Sum

Workload
- Equality
- Numeric Range

Current Working Query

```
SELECT f.flightid, f.carrier, f.flightno, f.deptairport, f.arrairport
FROM ds_etms_flight f
WHERE f.flightid >= 20090101000000
AND f.flightid <= 20090101000020;
```

Inspect Query Plan

Display Plan: Maximal Push

Show Graph