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

MITRE Innovation Program

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

red database security

Oracle-Anti-Hacker-Training
5 days click here

Services
Oracle Audit / Hardening
Security Training
Consulting

Information
Oracle Security Blog
Published Alerts
Upcoming Alerts
Patch Information
Whitepaper
Presentations
Oracle Fact Sheets
Exploits
Tutorials
Videos
Scripts

News & Events
Events
News

Company
Contact
People
Partner
Impressum
Sitemap

Search
Search

Search Red-Database-Security

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, plsql function, plsql packages, forms, reports, isqlplus, ohs, ... 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 not contain 0day 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.DRILOAD, 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)

Definition Exploit
An exploit is a common term in the computer security to refer to a piece of software that take advantage of a bug or vulnerability leading to a privilege escalation or d.o.s. on a computer system. Computer security experts are using exploit code to test if a patch is working properly.

New Features = New Exploits

Bribing the DBA …

DW-WORLD.DE
DEUTSCHE WELLE
YOUR LINK TO GERMANY

About us | Business & Sales | Contact
Press | DW-Akademie | Help | Site map
DW-STORE | Partner | Archive

20.01.2011

HOME

NEWS
- Germany
  - German Reunification
- Europe
- World
- Asia
- Business
- Science & Technology
- Environment & Development
- Culture & Lifestyle
- Sports

GERMANY INFO
- Visit Germany
- Facts about Germany
- Study in Germany

GERMAN COURSES
- Learning German
- Deutsch unterrichten

TAXES | 27.02.2010

German state buys Swiss bank data in tax evasion crackdown

Swiss banking secrecy has caused a spat with Germany

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.

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.

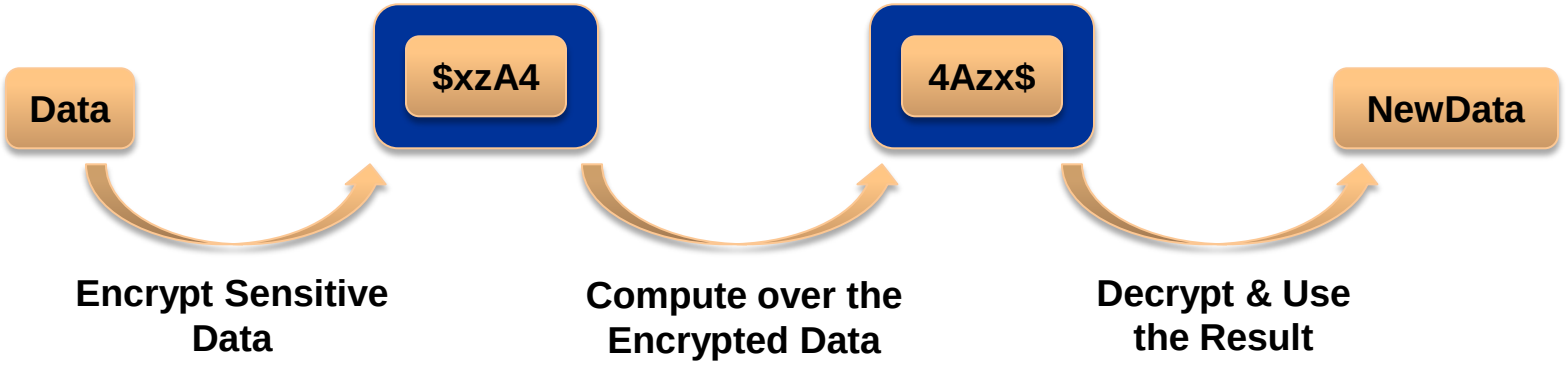
German magazine Focus disclosed in early February that North Rhine-Westphalia state officials were negotiating a purchase of data on 1,500 German clients of Swiss banks from a whistleblower in France, who reportedly asked for some 2.5 million euros (US$3.4 million).

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



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


Fully Homomorphic Encryption Using Ideal Lattices

Craig Gentry

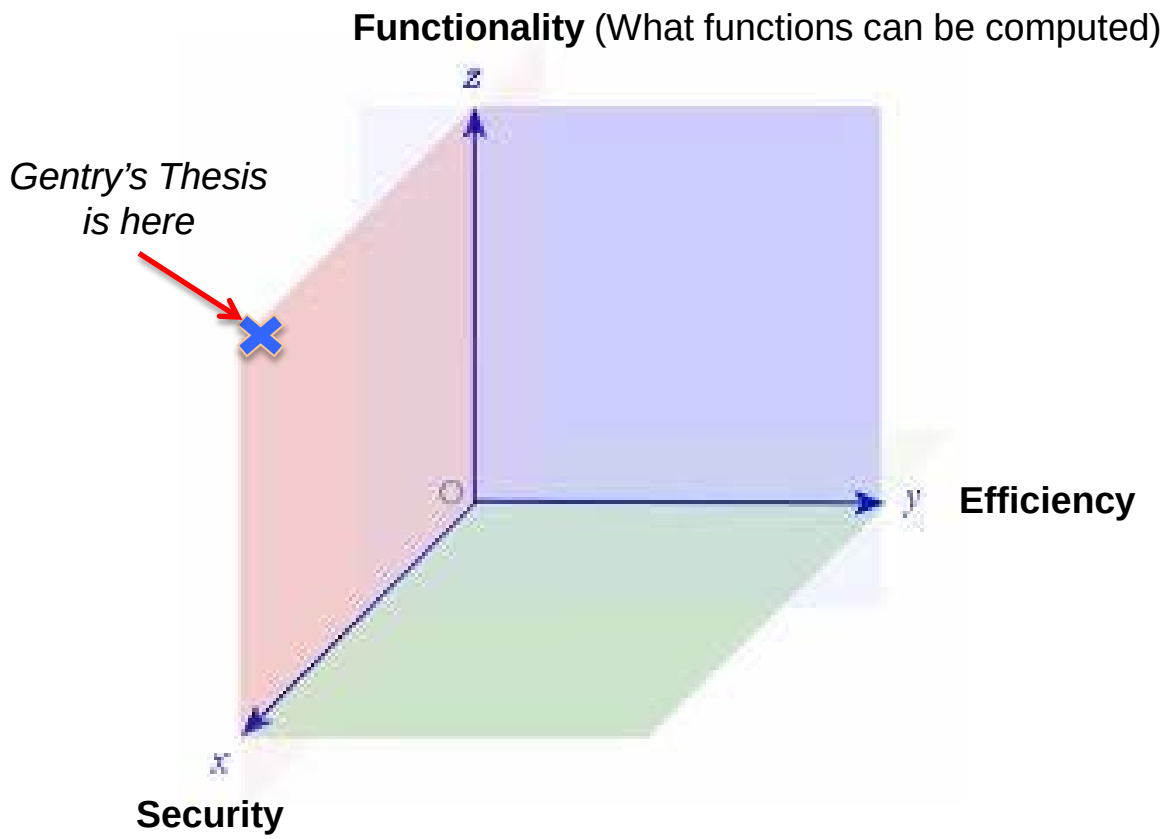
ABSTRACT

1. INTRODUCTION

1.3 Homomorphic Encryption

- 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

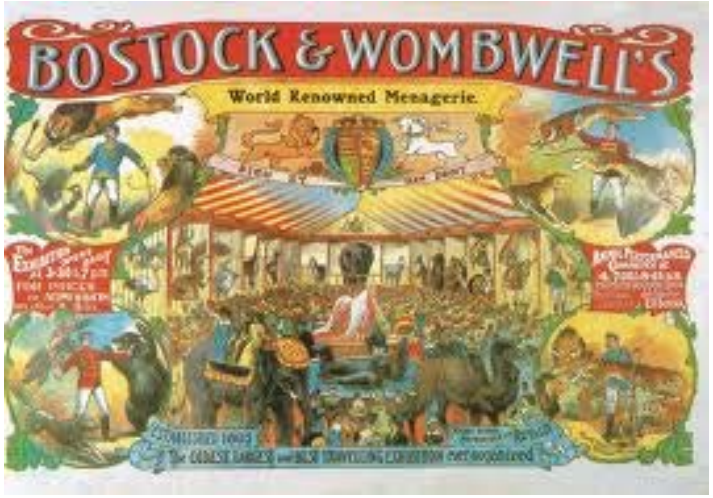


If we accept less functionality (i.e., portions of SQL), can we achieve practical efficiency?

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

Hakan Hacigümüş, Bala Iyer, Chen Li, and Sharad Mehrotra. 2002. Executing SQL over encrypted data in the database-service-provider model. In Proceedings of the 2002 ACM SIGMOD international conference on Management of data (SIGMOD '02). ACM, New York, NY, USA, 216-227.

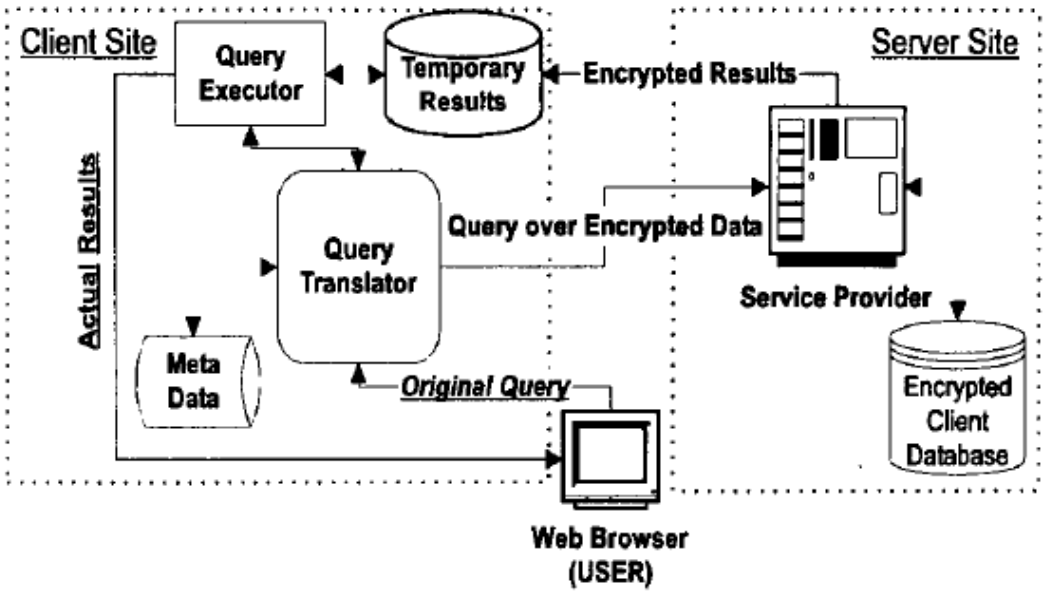


Figure 1: The service-provider architecture.

100+ subsequent papers!

Important "flavors":

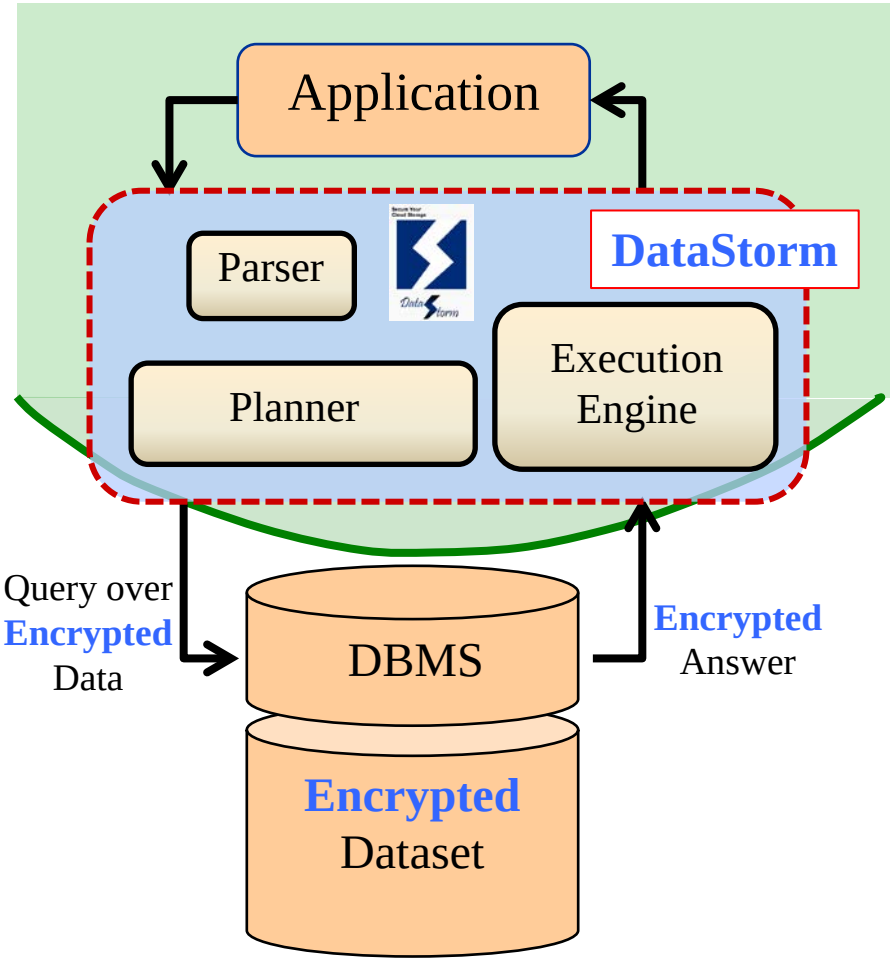
- Which cryptosystems are used
- Bucketization, or not
- Post-processing, or not

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**

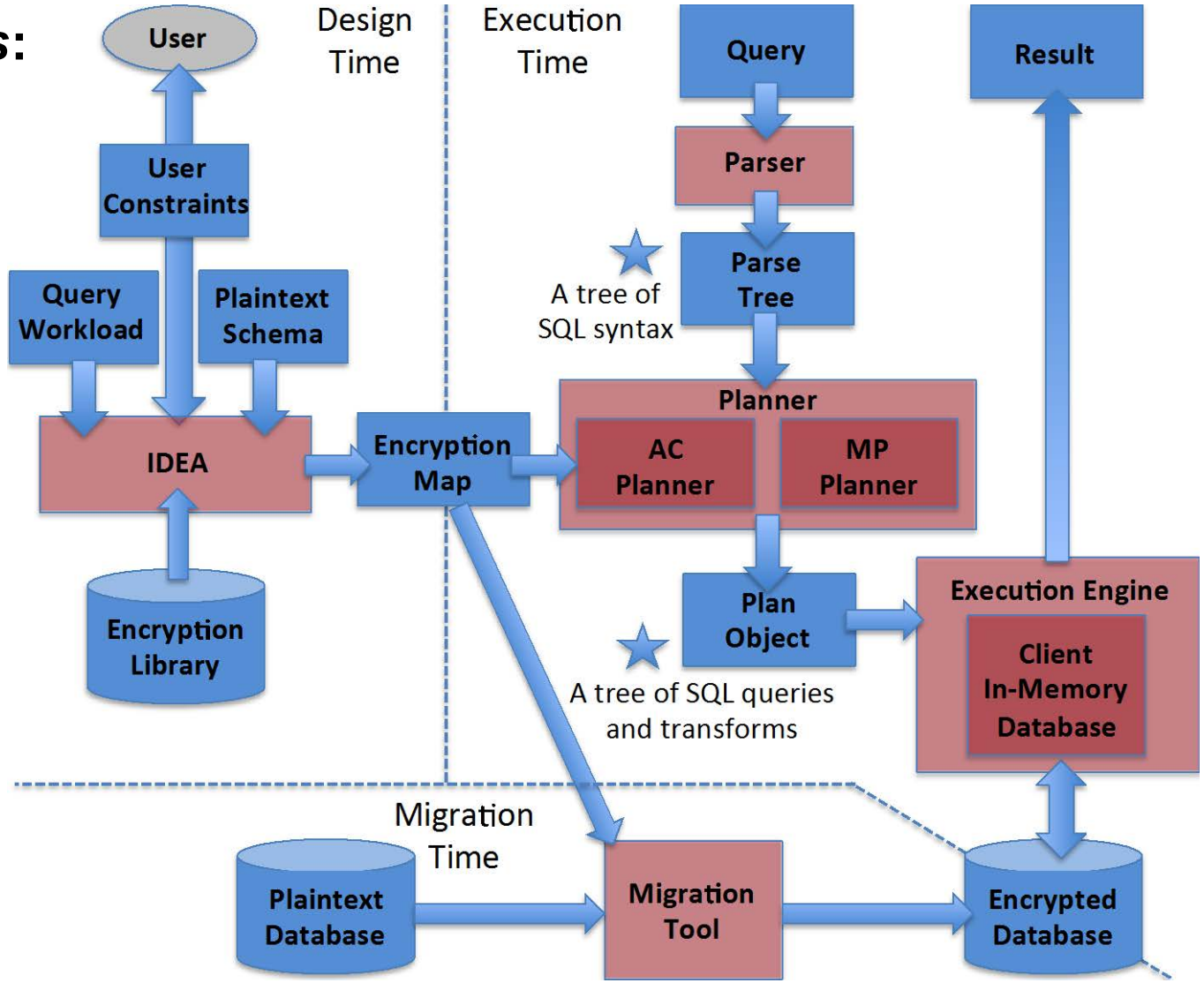


* Also key value store, and text repository …

***Larger goal**: surface & address barriers that keep computable encryption from being "shrink-wrapped"*

DataStorm Architecture

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



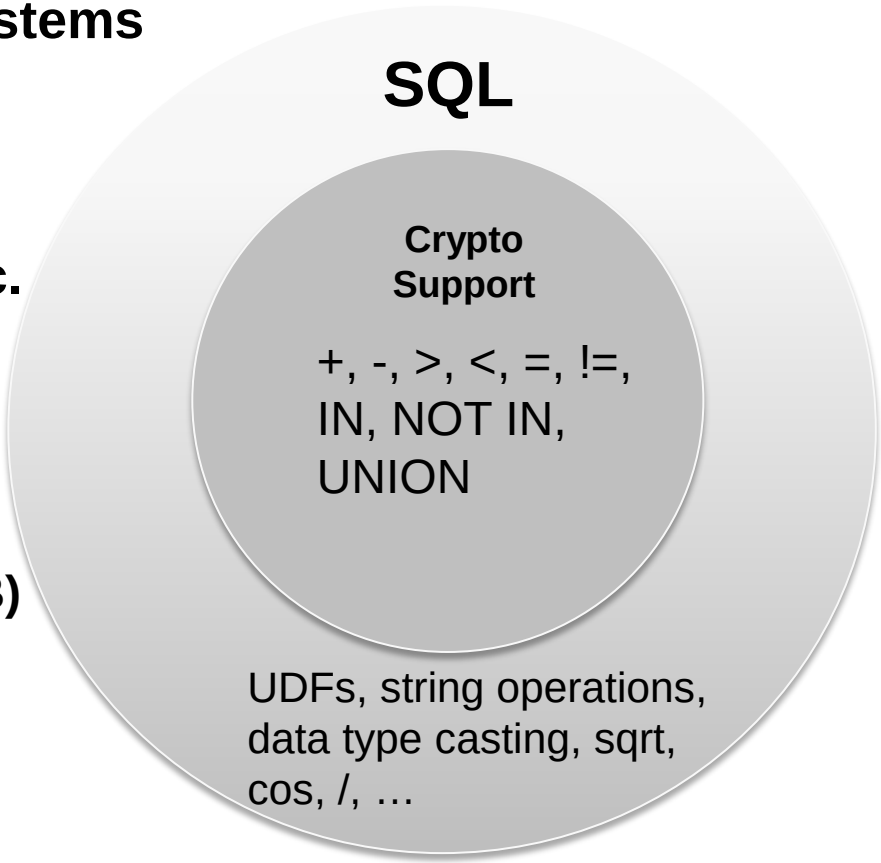
Outline

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

Barrier 1: Non-executable SQL Constructs

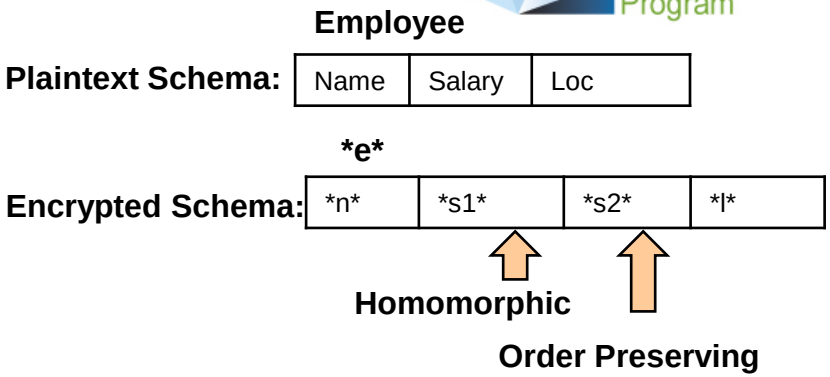
- **We currently lack (efficient) cryptosystems to execute:**
  - **arbitrary substring matches**
  - **sqrt(), cos(), /, ln(), …**
  - **fringe SQL: today(), bitwise or, etc.**
  - **many UDFs**

- **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):**

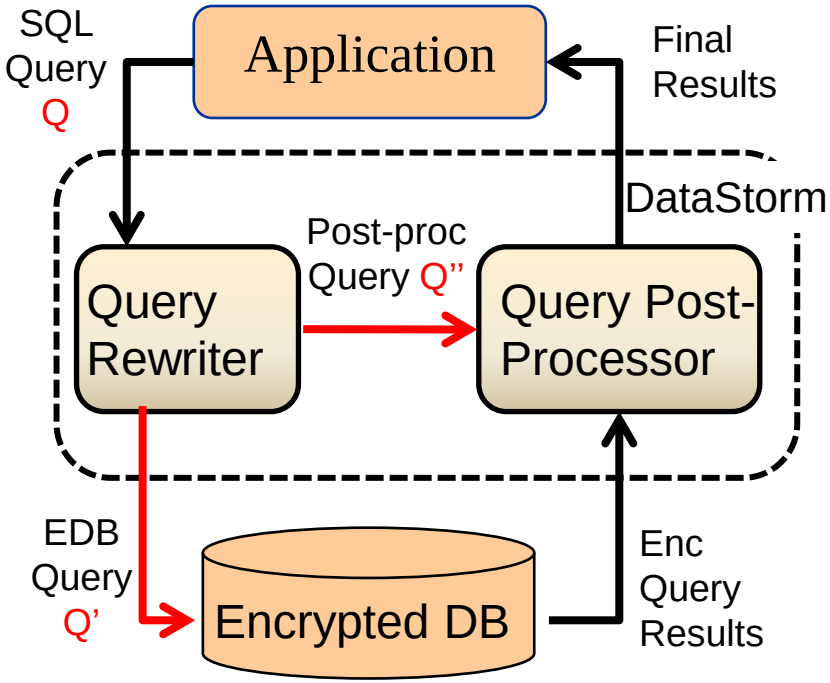
  SELECT Name, (Salary*0.0833)
  WHERE Salary < 100K AND Loc
  LIKE '%McLean%'

- **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)

DataStorm - Protection in a Turbulant Cloud

Application Layer

Create Query — 15.154 seconds

SELECT F.FLIGHTID, F.CARRIER, F.FLIGHTNO, F.DEPTAIRPORT, F.ARRAIRPORT FROM DS_ETMS_FLIGH...

```
SELECT F.FLIGHTID, F.CARRIER, F.FLIGHTNO, F.DEPTAIRPORT, F.ARRAIRPORT FROM
DS_ETMS_FLIGHT F WHERE F.CARRIER='DAL' AND F.DEPTAIRPORT LIKE 'MI%'
```

Display Results

FLIGHTID	CARRIER	FLIGHTNO	DEPTAIRPORT	ARRAIRPORT
2.0090101011476...	DAL	982	MIA	ATL
2.0090101017258...	DAL	1822	MIA	JFK
2.0090101017289...	DAL	990	MIA	ATL

DataStorm Layer

Encrypt Query — 0.015 seconds

```
SELECT F.sAnkeuHQfschZKSP, F.pbZnBNkalMcHHtSs, F.DJzEqluvBcnMAPSR, F.TyXpvWjBWBnfWurA,
F.WIJSvxEzPiKtdypJ FROM sVhCByVxAOcfqUzR F  WHERE F.pbZnBNkalMcHHtSs = ---param0--- AND  (1 =
1)
```

Perform Post-Processing — 1.436 seconds

```
SELECT COLUMN1, COLUMN2, COLUMN3, COLUMN4, COLUMN5 FROM RESULTS WHERE  (1 = 1)
AND column4  LIKE  'MI%'
```

Database Layer

Execute Query — 0.187 seconds

sAnkeuHQfschZKSP	pbZnBNkalMcHHtSs	DJzEqluvBcnMAPSR	TyXpvWjBWBnfWurA	WIJSvxEzPiKtdypJ
70f95cad808774cff...	e877e39aaec73ce...	56c56364393de15...	abcb12babebaf2e...	af98daaf38e0826e...
097ea5dcea43d46...	e877e39aaec73ce...	aa80d6e77a9c01b...	abcb12babebaf2e...	846aacf05a6950f3...
561407e3d9b539f...	e877e39aaec73ce...	41464320f0404f49...	abcb12babebaf2e...	28c9d5a99c7631d...
a25ddbbaa78655a...	e877e39aaec73ce...	b3eb47ea9e8956...	b25a945747615a8...	5cd322d7d493d58...
20a52a6eef3ab18f...	e877e39aaec73ce...	93ad8c905f7de6b...	b25a945747615a8...	a31e429e96c4f81...
8cc4b1055965f27...	e877e39aaec73ce...	9cf34c084a19bcd...	b25a945747615a8...	097e08c5c3704a7...
fadbe9ef0fa3a926...	e877e39aaec73ce...	90a480b1c45a9cb...	bc6f7eb82a94a19...	b25a945747615a...
42de5ba0b1bd7e0...	e877e39aaec73ce...	130289b5dd0165...	37c09840b1a3ec8...	885ddc3aeb43e71...
a52c00cf4b74056...	e877e39aaec73ce...	d53f9892f5a72cbc...	f9c920dce653d5a...	7760568fafea8174...
30b5cba1f58f2040...	e877e39aaec73ce...	d81384035d7bb4...	95d964e874b0e00...	9be2a711e11ac42...
f8ecc2b11c7a6922...	e877e39aaec73ce...	2635ed88a602e8...	f9c920dce653d5a...	99af73e6b67f7c92...

Decrypt Results — 0.063 seconds

FLIGHTID	CARRIER	FLIGHTNO	DEPTAIRPORT	ARRAIRPORT
2.0090101000012E...	DAL	1020	ATL	PHL
2.0090101000121...	DAL	1033	ATL	PHX
2.0090101000122...	DAL	1044	ATL	RIC
2.0090101000124...	DAL	1142	SLC	MSY
2.0090101000125...	DAL	1166	SLC	RDU
2.0090101000126...	DAL	1170	SLC	EWR
2.0090101000127...	DAL	1183	SEA	SLC
2.0090101000128...	DAL	137	JFK	LAX
2.0090101000129...	DAL	1402	FLL	BDL
2.009010100013E...	DAL	1440	MCO	BOS
2.0090101000131...	DAL	1462	FLL	LGA

2:57 PM
6/10/2011

Execution Trace

Create Query 15.154 seconds

SELECT F.FLIGHTID, F.CARRIER, F.FLIGHTNO, F.DEPTAIRPORT, F.ARRAIRPORT FROM DS_ETMS_FLIGH...

```
SELECT F.FLIGHTID, F.CARRIER, F.FLIGHTNO, F.DEPTAIRPORT, F.ARRAIRPORT FROM
DS_ETMS_FLIGHT F WHERE F.CARRIER='DAL' AND F.DEPTAIRPORT LIKE 'MI%'
```

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.

Execution Trace

Encrypt Query 0.015 seconds

```
SELECT F.sAnkeuHQfschZKSP, F.pbZnBNkaIMcHHtSs, F.DJzEqluvBcnMAPSR, F.TyXpvWjBWBnfWurA,
F.WIJSvxEzPiKtdypJ FROM sVhCByVxAOcfqUzR F  WHERE F.pbZnBNkaIMcHHtSs = ---param0--- AND  (1 =
1)
```

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.

Execution Trace

Execute Query 0.187 seconds

sAnkeuHQfschZKSP	pbZnBNkalMcHHtSs	DJzEqluvBcnMAPSR	TyXpvWjBWBnfWurA	WlJSvxEzPiKtdypJ
70f95cad808774cff...	e877e39aaec73ce...	56c56364393de15...	abcb12babebaf2e...	af98daaf38e0826e...
097ea5dcea43d46...	e877e39aaec73ce...	aa80d6e77a9c01b...	abcb12babebaf2e...	846aacf05a6950f3...
561407e3d9b539f...	e877e39aaec73ce...	41464320f0404f49...	abcb12babebaf2e...	28c9d5a99c7631d...
a25ddbbaa78655a...	e877e39aaec73ce...	b3eb47ea9e8956...	b25a945747615a8...	5cd322d7d493d58...
20a52a6eef3ab18f...	e877e39aaec73ce...	93ad8c905f7de6b...	b25a945747615a8...	a31e429e96c4f81...
8cc4b1055965f27...	e877e39aaec73ce...	9cf34c084a19bcd...	b25a945747615a8...	097e08c5c3704a7...
fadbe9ef0fa3a926...	e877e39aaec73ce...	90a480b1c45a9cb...	bc6f7eb82a94a19...	b25a945747615a...
42de5ba0b1bd7e0...	e877e39aaec73ce...	130289b5dd0165...	37c09840b1a3ec8...	885ddc3aeb43e71...
a52c00cf4b74056...	e877e39aaec73ce...	d53f9892f5a72cbc...	f9c920dce653d5a...	7760568fafea8174...
30b5cba1f58f2040...	e877e39aaec73ce...	d81384035d7bb4...	95d964e874b0e00...	9be2a711e11ac42...
f8ecc2b11c7a6922...	e877e39aaec73ce...	2635ed88a602e8...	f9c920dce653d5a...	99af73e6b67f7c92...

The server executes the query and returns encrypted data.

Execution Trace

Decrypt Results 0.063 seconds

FLIGHTID	CARRIER	FLIGHTNO	DEPTAIRPORT	ARRAIRPORT
2.009010100012E...	DAL	1020	ATL	PHL
2.00901010001 21...	DAL	1033	ATL	PHX
2.00901010001 22...	DAL	1044	ATL	RIC
2.00901010001 24...	DAL	1142	SLC	MSY
2.00901010001 25...	DAL	1166	SLC	RDU
2.00901010001 26...	DAL	1170	SLC	EWR
2.00901010001 27...	DAL	1183	SEA	SLC
2.00901010001 28...	DAL	137	JFK	LAX
2.00901010001 29...	DAL	1402	FLL	BDL
2.009010100013E...	DAL	1440	MCO	BOS
2.00901010001 31...	DAL	1462	FLL	LGA

Back on the client-side, the data is decrypted back to plaintext.

Execution Trace

Perform Post-Processing 1.436 seconds

```
SELECT COLUMN1, COLUMN2, COLUMN3, COLUMN4, COLUMN5 FROM RESULTS WHERE (1 = 1)
AND column4 LIKE 'MI%'
```

In post-processing, we must further filter the results to only those where the departing airport was like "MI%"

Execution Trace

Display Results

FLIGHTID	CARRIER	FLIGHTNO	DEPTAIRPORT	ARRAIRPORT
2.0090101011476...	DAL	982	MIA	ATL
2.0090101017258...	DAL	1822	MIA	JFK
2.0090101017289...	DAL	990	MIA	ATL

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

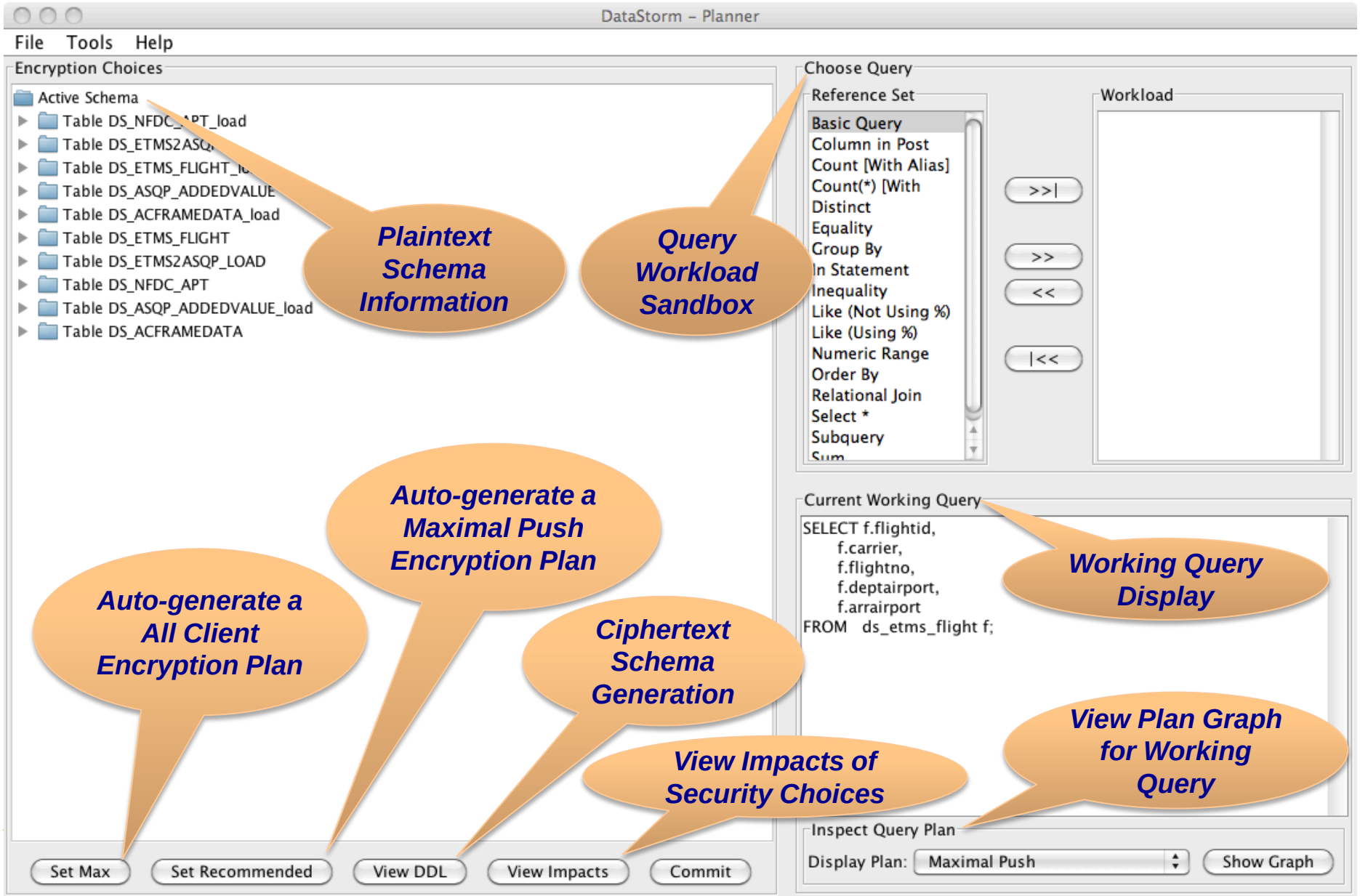
- Solution: help users quickly assess deployability!
  - and easily (or automatically) configure encryption choices

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



IDEA Demo

MITRE Innovation Program

DataStorm – Planner

File Tools Help

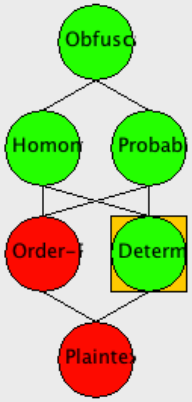
Encryption Choices

Active Schema
- Table DS_NFDC_APT_load
- Table DS_ETMS2ASQP
- Table DS_ETMS
- Table DS_ASQP
- Table DS_ACFR
- Table DS_ETMS
  - FLIGHTID DO
    - Choices:
    - Reasons
  - ACID VARCHA
  - CARRIER VAR
    - Choices:
    - Reasons
  - FLIGHTNO VA
    - Choices:
    - Reasons
  - DEPTAIRPORT
    - Choices:
    - Reasons
  - ARRAIRPORT
    - Choices:
    - Reasons
  - DEPTDATE DA
  - ARRDATE DA
  - STATUS CHAR
  - AC_TYPE VAR
- Table DS_ETMS
- Table DS_NFDC
- Table DS_ASQP
- Table DS_ACFR

Encryption Choices for Recommended "Maximal Push" Plan

Select a new type of Encryption

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



OK

Set Max | Set Recommended | View DDL | View Impacts | Commit

Choose Query

Choice Set
- 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.deptairport,
       f.arrairport
FROM   ds_etms_flight f
WHERE  f.carrier = 'DAL';
```

Inspect Query Plan

Display Plan: Maximal Push | Show Graph

IDEA Demo

DataStorm – Planner

File Tools Help

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: Probabilistic
      - 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

Set Max | Set Recommended | View DDL | View Impacts | Commit

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.deptairport,
       f.arrairport
FROM   ds_etms_flight f
WHERE  f.carrier = 'DAL';
```

Inspect Query Plan

Display Plan: Maximal Push | Show Graph

IDEA Demo

MITRE Innovation Program

DataStorm – Planner

File Tools Help

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

Set Max | Set Recommended | View DDL | View Impacts | Commit

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

MITRE Innovation Program

DataStorm – Planner

File Tools Help

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

Set Max | Set Recommended | View DDL | View Impacts | Commit

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

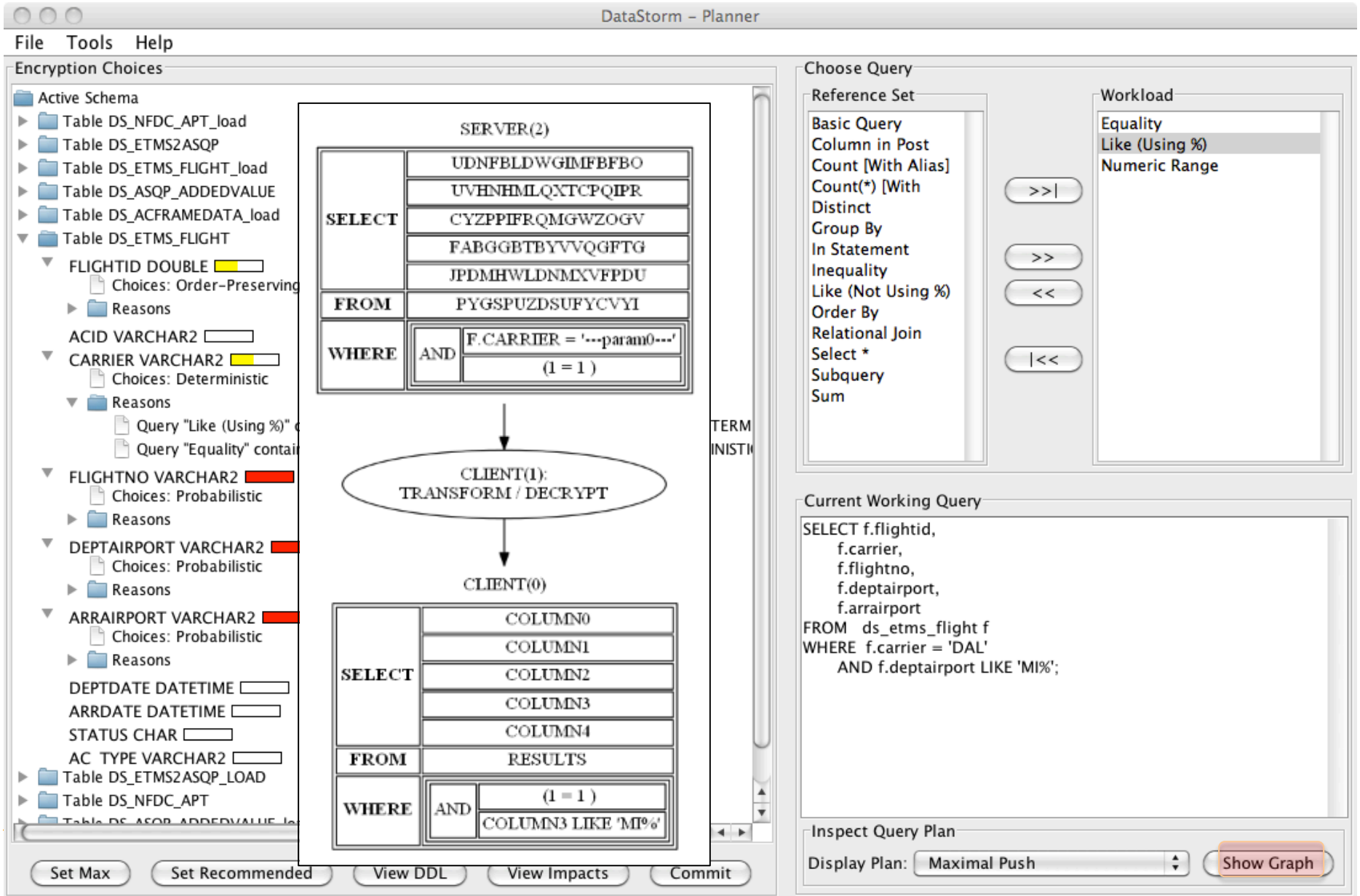
Current Working Query

```
SELECT f.flightid,
       f.carrier,
       f.flightno,
       f.deptairport,
       f.arrairport
FROM   ds_etms_flight f
WHERE  f.carrier = 'DAL'
       AND f.deptairport LIKE 'MI%';
```

Inspect Query Plan

Display Plan: Maximal Push | Show Graph

IDEA Demo


DataStorm – Planner
File Tools Help
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
Query "Like (Using %)"
Query "Equality" contai
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
SERVER(2)
SELECT
UDNFBLDWGIMFBFBO
UVHNHMLQXTCPQIPR
CYZPPIFRQMGWZOGV
FABGGBTBYVVQGFTG
JPDMHWLDNMXVFPDU
FROM
PYGSPUZDSUFYCVYI
WHERE
AND
F.CARRIER = '---param0---'
(1 = 1)
CLIENT(1):
TRANSFORM / DECRYPT
CLIENT(0)
SELECT
COLUMN0
COLUMN1
COLUMN2
COLUMN3
COLUMN4
FROM
RESULTS
WHERE
AND
(1 = 1)
COLUMN3 LIKE 'MI%'
Set Max
Set Recommended
View DDL
View Impacts
Commit
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
Current Working Query
SELECT f.flightid,
f.carrier,
f.flightno,
f.deptairport,
f.arrairport
FROM ds_etms_flight f
WHERE f.carrier = 'DAL'
AND f.deptairport LIKE 'MI%';
Inspect Query Plan
Display Plan:
Maximal Push
Show Graph

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)**

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

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**

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

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

	PT total	CT (MP) total	CT/PT ratio	CT server	CT decrypt	CT misc
10K	17.5451445	31.9446655	1.82	0.552111111	0.716081	30.67647339
100K	64.098557	199.638414	3.11	15.73873737	5.051751	178.8479256
1000K	594.0421615	1944.573251	3.27	211.590899	40.4716245	1692.510728

- **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

	PT total	CT total	CT/PT ratio	CT server	CT decrypt	CT misc
10K	0.6286115	1.0018130	1.59	0.68953535	0.1819360	0.130341646
100K	3.2718125	7.1076260	2.17	5.31842424	1.2889455	0.500256260
1000K	220.8141110	369.9193835	1.67	356.80302020	12.7611650	0.355198300

- **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

	PT server	CT server	CT/PT ratio
10K	1.498111111	111.9696768	74.74
100K	14.01076768	1153.161051	82.31
1000K	169.0187677	11356.59457	67.19

- **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

ID	NAME	ADDRESS	NATION	PHONE	EXPENSES
1	Customer#000000001	23rd St. Tehran	IRAN	25-989-741-988	712
~~2~~	~~Customer#000000002~~	~~2nd St. Lyon …~~	~~FRANCE~~	~~23-768-687-3665~~	~~122~~

Ciphertext Table

klasjfklsjfjksldfsk	wioruweioriweour	xizcopvizxopvuio	…
119	ioasdufiosdafuio	lowjecnwc	…
~~275~~	~~uioasufisafu[aias~~	~~sfsklfjsklfsdf~~	~~…~~

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

ID	NAME	…	EXPENSES
~~1~~	~~Customer#0000001~~	~~…~~	~~5000~~
2	Customer#0000002	…	9002
3	Customer#0000003	…	9008
~~4~~	~~Customer#0000004~~		~~9010~~

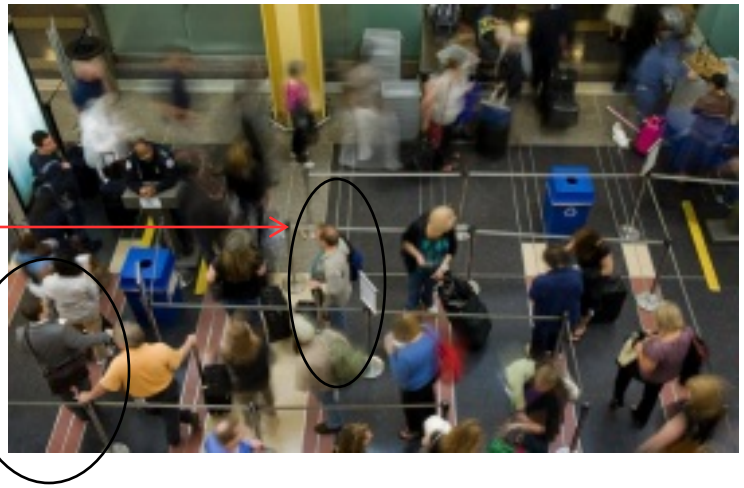
Ciphertext Table

j37fjhwify	yrbducvtbe	…	uxieh3b98f
~~179~~	~~lowjecnwceiwnjdu~~	~~…~~	~~23~~
234	sfsklfjsklfsdfiw	…	23328
314	iuofdgbiufdocyen	…	25684
~~583~~	~~iwnfgksienuwider~~		~~26738~~

Accumulo: Simple Analytic

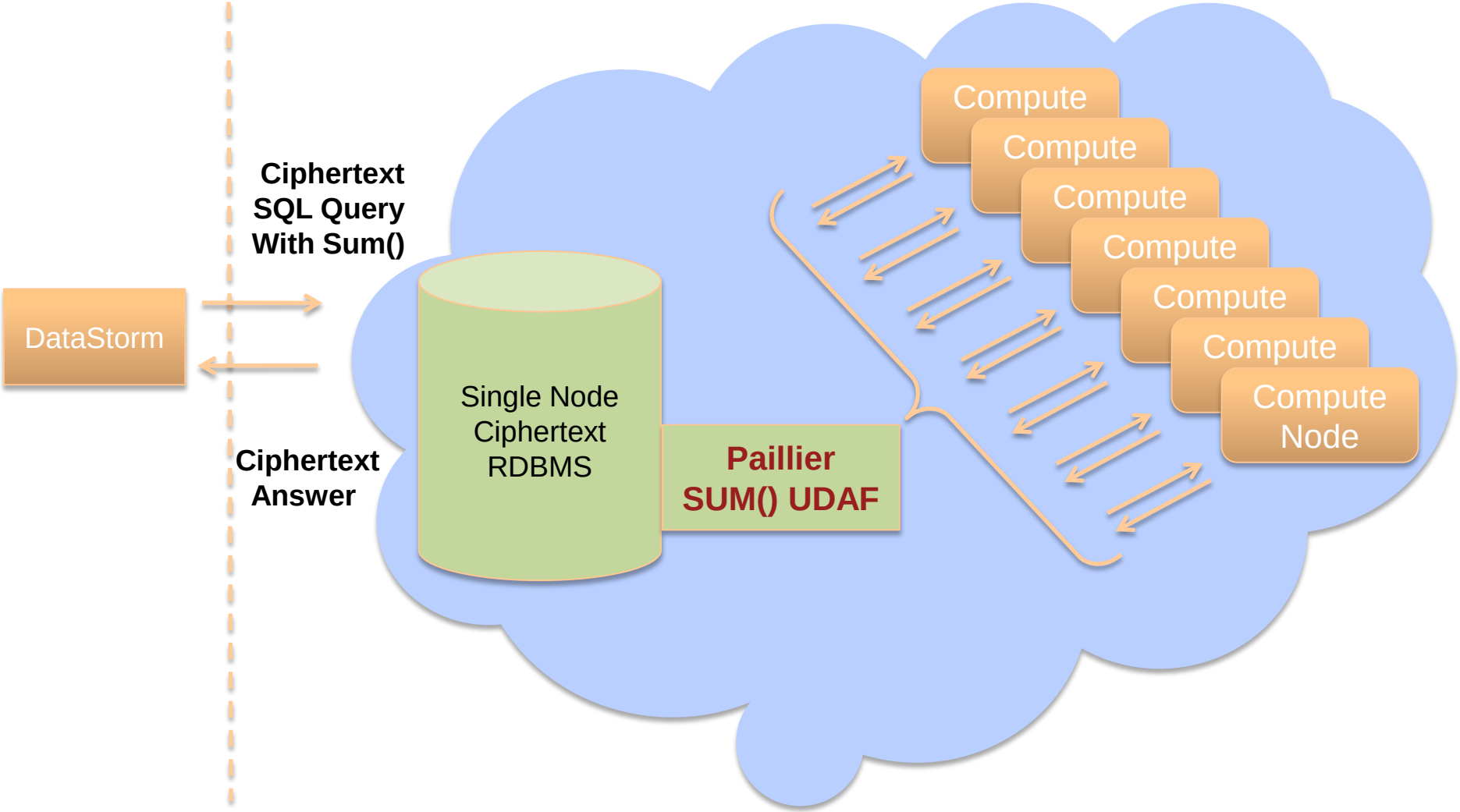
- **Finding co-travelers**
  - **Get me watchlist passengers who have either flown into or out of the same airport on the same data**

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 Palillier Sum via Parallelism

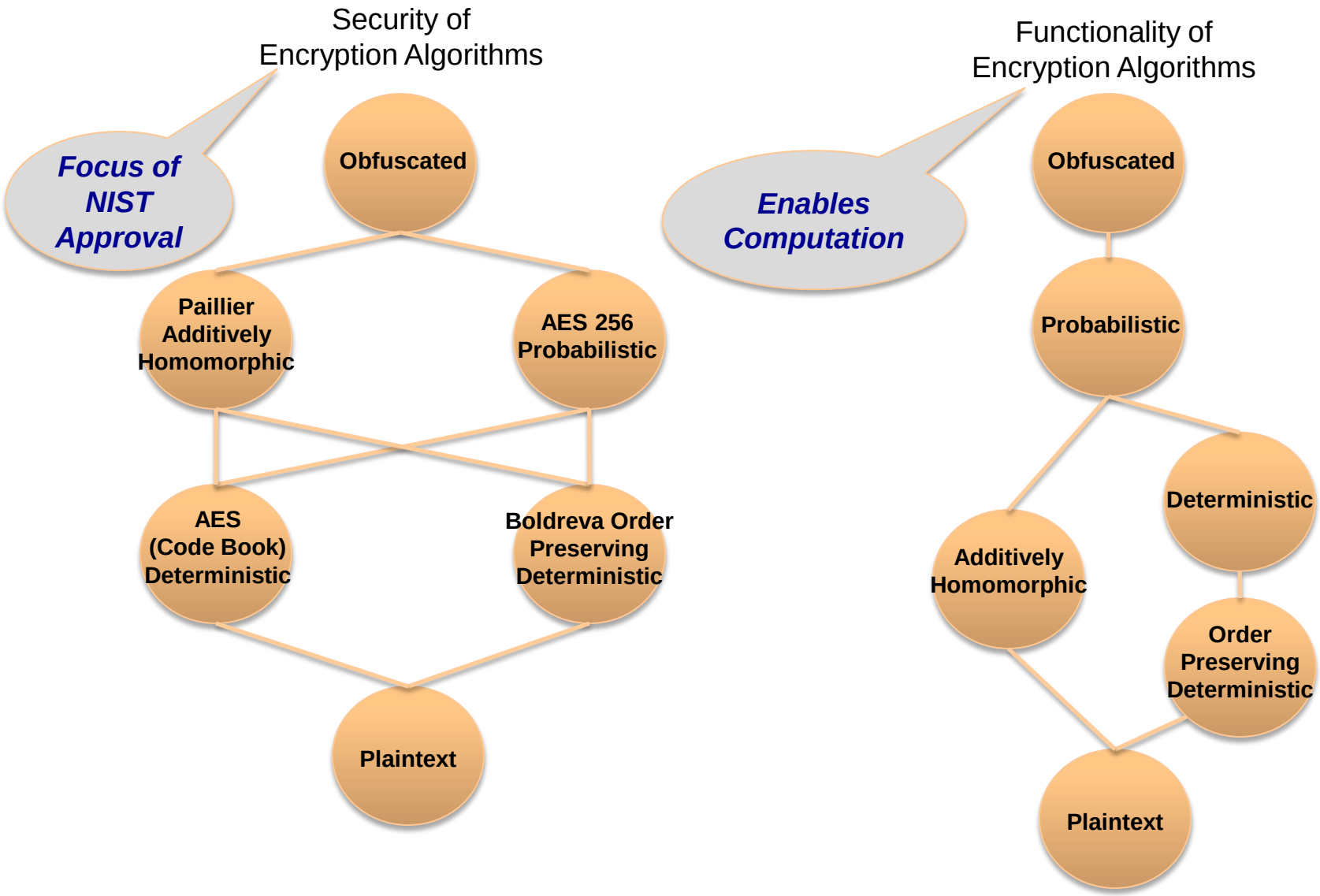


Impact Summary

- 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!



Demo: Regression Test

DataStorm Regression Tests

Basic Query

View Results

SELECT F.FLIGHTID, F.CARRIER, F.FLIGHTNO, F.DEPTAIRPORT, F.ARRAIRPORT FROM DS_ETMS_FLIGHT F

Time: 0.218 seconds / 2.438 seconds

Equality

View Results

SELECT F.FLIGHTID, F.CARRIER, F.FLIGHTNO, F.DEPTAIRPORT, F.ARRAIRPORT FROM DS_ETMS_FLIGHT F WHERE F.CARRIER='DAL'

Time: 0.046 seconds / 1.157 seconds

Inequality

View Results

SELECT F.FLIGHTID, F.CARRIER, F.FLIGHTNO, F.DEPTAIRPORT, F.ARRAIRPORT FROM DS_ETMS_FLIGHT F WHERE F.CARRIER<>'DAL'

Time: 0.156 seconds / 1.547 seconds

Numeric Range

View Results

SELECT F.FLIGHTID, F.CARRIER, F.FLIGHTNO, F.DEPTAIRPORT, F.ARRAIRPORT FROM DS_ETMS_FLIGHT F WHERE F.FLIGHTID >= 20090101000000 AND F.FLIGHTID <= 20090101000020

Time: 0.031 seconds / 0.844 seconds

Order By

View Results

SELECT F.FLIGHTID, F.CARRIER, F.FLIGHTNO, F.DEPTAIRPORT, F.ARRAIRPORT FROM DS_ETMS_FLIGHT F ORDER BY F.DEPTAIRPORT

Time: 0.188 seconds / 1.813 seconds

Distinct

View Results

SELECT DISTINCT F.CARRIER FROM DS_ETMS_FLIGHT F

Time: 0.016 seconds / 0.828 seconds

Group By

View Results

SELECT F.CARRIER, COUNT(F.CARRIER) AS COUNT FROM DS_ETMS_FLIGHT F GROUP BY F.CARRIER

Time: 0.032 seconds / 0.89 seconds

Like (Not Using %)

View Results

SELECT F.FLIGHTID, F.CARRIER, F.FLIGHTNO, F.DEPTAIRPORT, F.ARRAIRPORT FROM DS_ETMS_FLIGHT F WHERE F.DEPTAIRPORT LIKE 'MIA'

Time: 0.032 seconds / 2.75 seconds

Like (Using %)

View Results

SELECT F.FLIGHTID, F.CARRIER, F.FLIGHTNO, F.DEPTAIRPORT, F.ARRAIRPORT FROM DS_ETMS_FLIGHT F WHERE F.CARRIER='DAL' AND F.DEPTAIRPORT LIKE 'MI%'

Time: 0.015 seconds / 0.922 seconds

In Statement

Probabilistic Encryption Systems

- With overwhelming probability, two encryptions of the same plaintext will yield different ciphertexts, for example:
  - $encrypt_1$("Bob") = "#ta$3b7"
  - $encrypt_2$("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
  - encrypt$_1$(“Bob”) = “#ta$3b7”
  - encrypt$_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" ➔ 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) *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

[1] Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Order p
Proceedings of the 2004 ACM SIGMOD international con
ACM, Paris, France (2004)

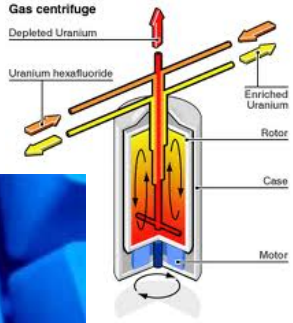
[2] Boldyreva, A., Chenette, N., Lee, Y., O'Neill, A.: Order-Pre
Proceedings of the 28th Annual International Conference on Adv
Theory and Applications of Cryptographic Techniques. Springer-V erlag, Cologne, Germany
(2009)

Research Trends in Encrypted Databases: A Literature Survey

Ken Smith – The MITRE Corporation– 12/31/10

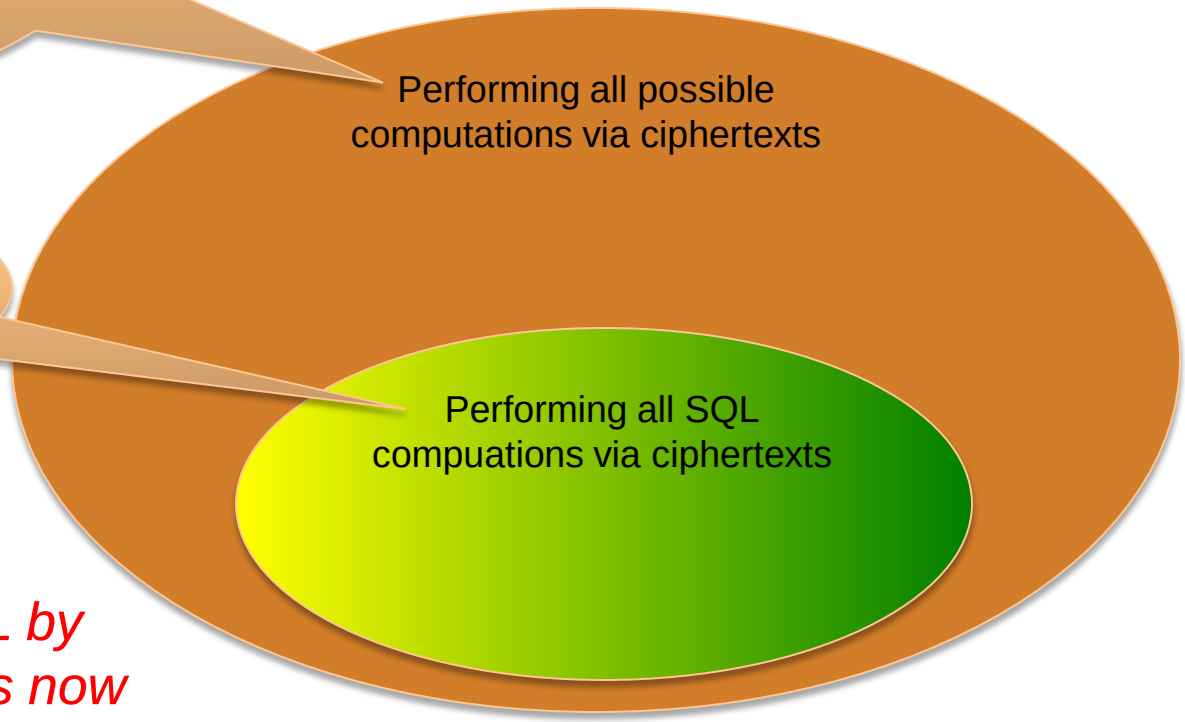
A More Insidious Problem

- **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 <u>any</u> 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!

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

Principal Investigator:	Ken Smith
Organization:	E540
MTS Level:	1.7 SY
Status:	Continuation
Mission Advocate:	John Marx
	ESC/HNCDI

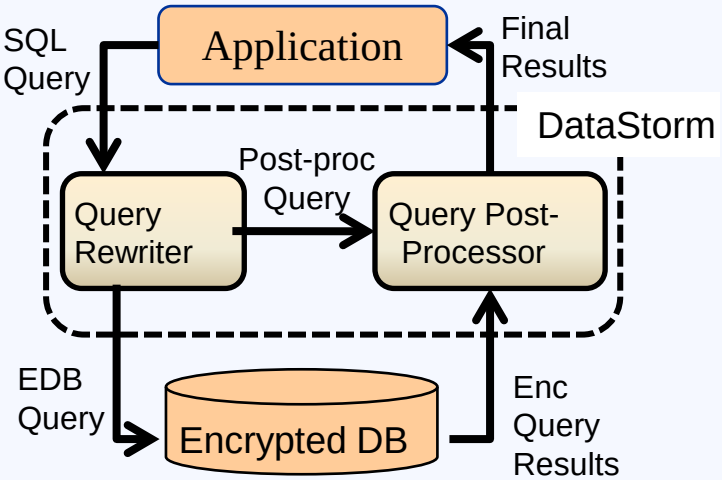
Research Objectives

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 negigible 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



IDEA Demo

DataStorm – Planner

File Tools Help

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: Probabilistic
      - 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

Set Max | Set Recommended | View DDL | View Impacts | Commit

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.deptairport,
       f.arrairport
FROM   ds_etms_flight f
WHERE  f.carrier = 'DAL';
```

Inspect Query Plan

Display Plan: Maximal Push | Show Graph

IDEA Demo

MITRE Innovation Program

DataStorm – Planner

File Tools Help

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

Set Max | Set Recommended | View DDL | View Impacts | Commit

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