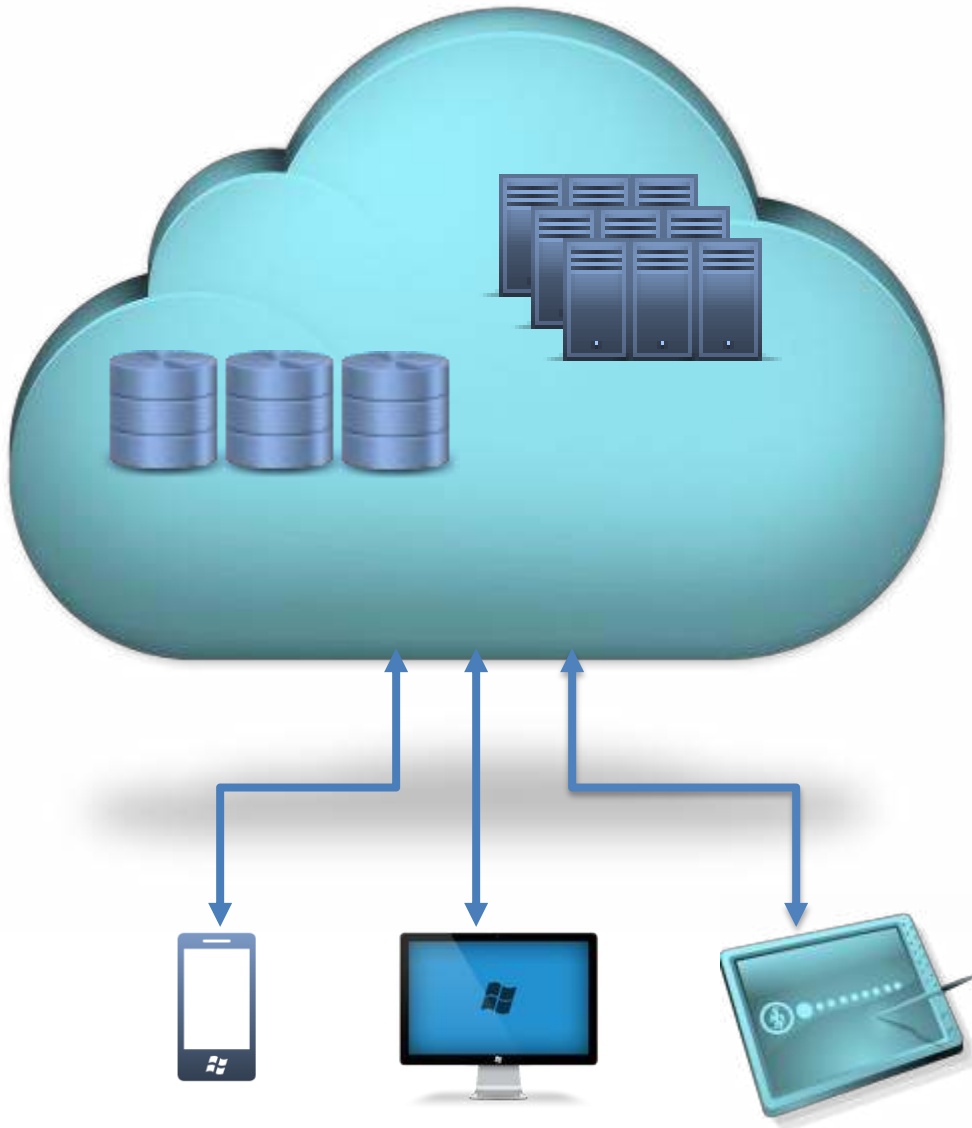


# Querying Encrypted Data

Arvind Arasu, Ken Eguro,  
Ravi Ramamurthy, Raghav Kaushik

Microsoft Research

# Cloud Computing



- Well-documented benefits
- Trend to move computation and data to cloud
- Database functionality
  - Amazon RDS
  - Microsoft SQL Azure
  - Heroku PostgreSQL
  - Xeround

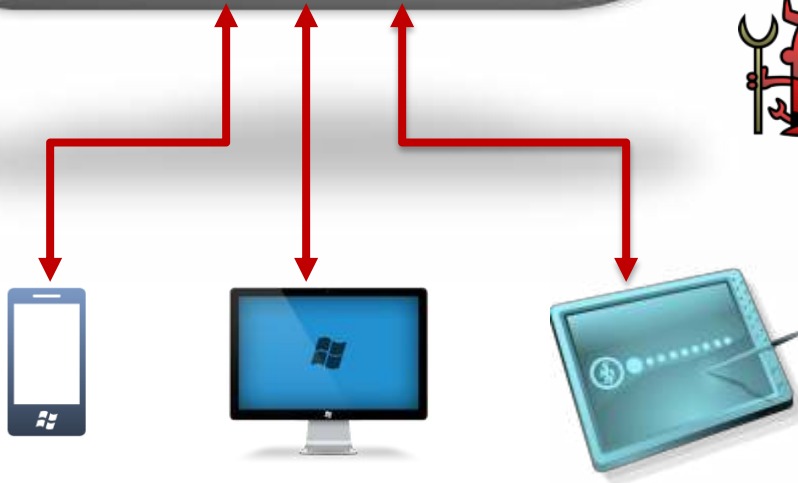
[AF+09, NIST09]

# Security Concerns



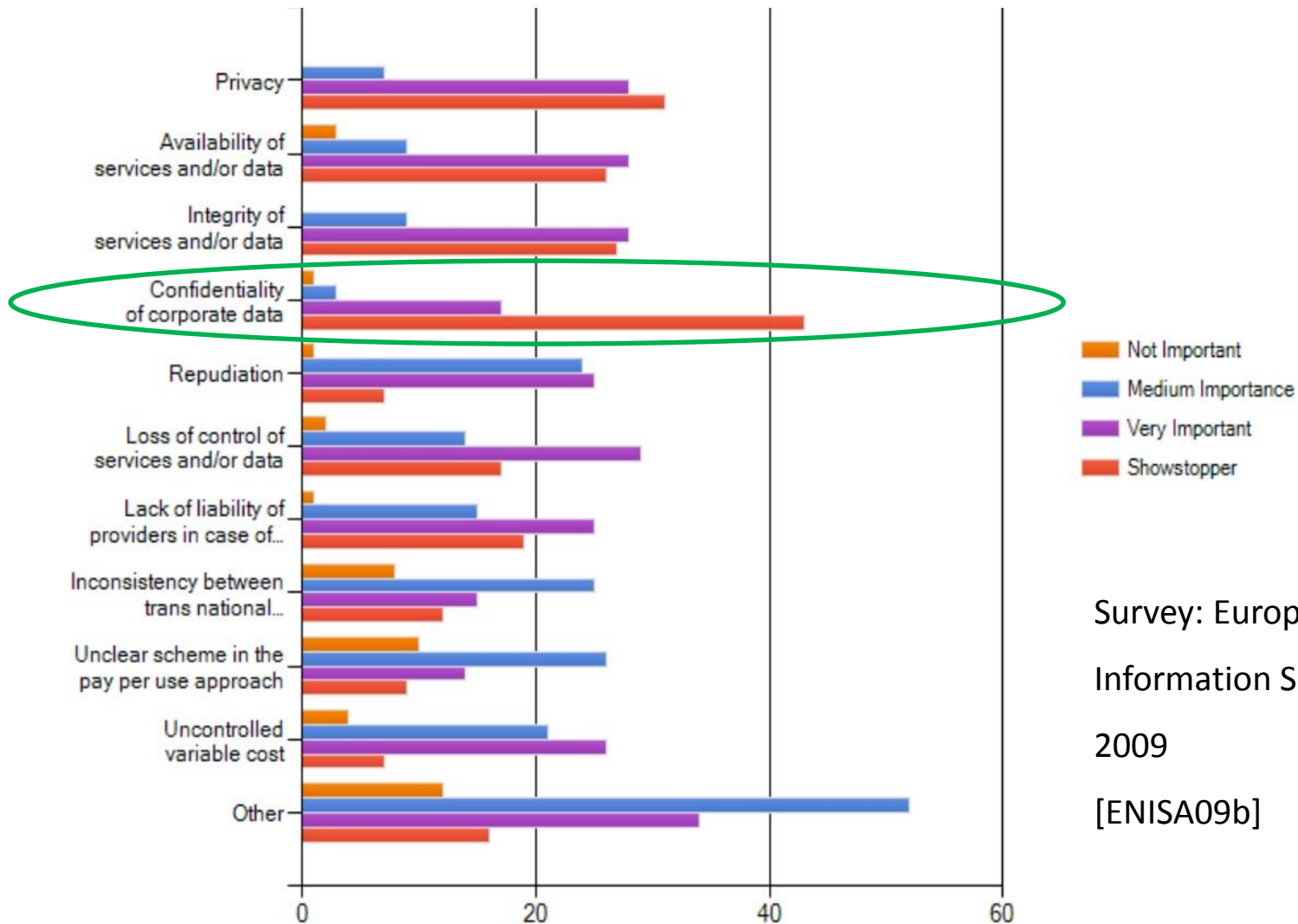
Data in the cloud vulnerable to:

- Snooping administrators
- Hackers with illegal access
  - Compromised servers



[CPK10, ENISA09a]

# What are your main cloud computing concerns?



Survey: European Network and Information Security Agency, Nov 2009  
[ENISA09b]

# Sensitive Data in the Cloud: Examples

## Software as a Service Applications

### Billing

- Aria Systems
- eVapt
- nDEBIT
- Redi2
- Zuora

### CRM

- 37 Signals
- Capsule
- Dynamics
- Intouchcrm
- LiveOps
- Oracle CRM
- Parature
- Responsys
- RO|Enablement
- Salesforce.com
- Save My Table
- Solve 360

### ERP

- Acumatica ERP
- Blue Link Elite
- Epicor Express
- NetSuite
- OrderHarmony
- Plex Online

### Health

- CECity
- SNO

### Personal Data

- Google Docs
- Microsoft Office
- Mint.com

Corporate data

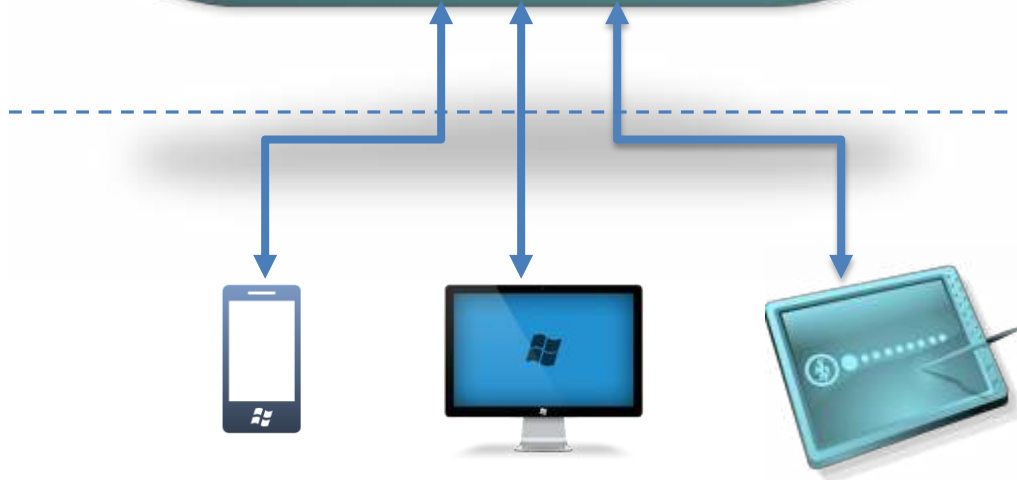
Personal data

Source: <http://cloudtaxonomy.opencrowd.com/taxonomy/>

# Data Encryption



```
a7be1a6997ad739bd8c9ca451f618b61  
b6ff744ed2c2c9bf6c590cbf0469bf41  
47f7f7bc95353e03f96c32bcfd8058df
```



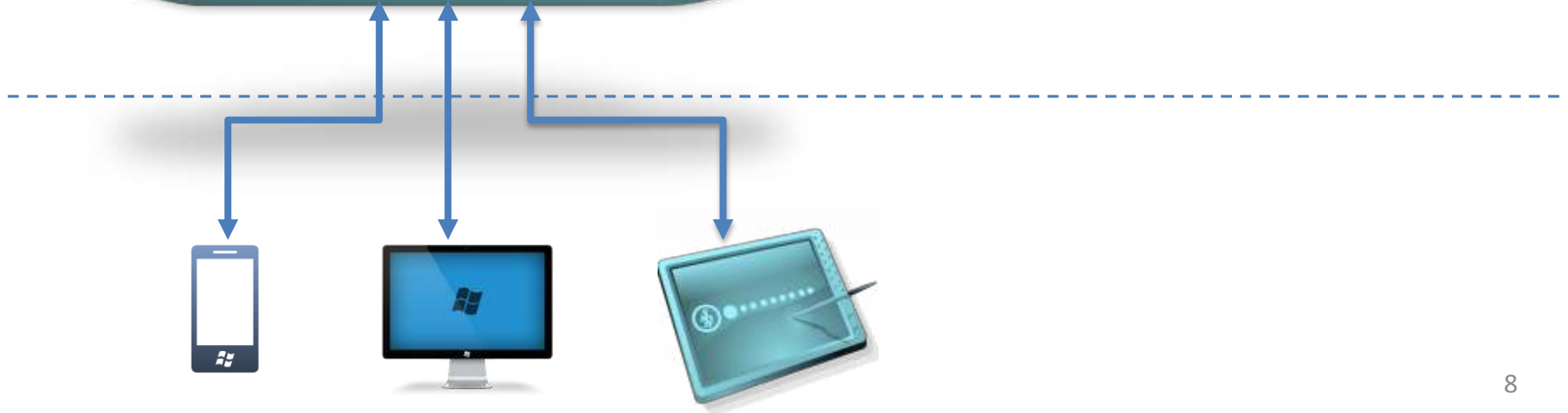
The quick brown fox jumps  
over the lazy dog

# AWS Security Advice

**7.2. Security.** We strive to keep Your Content secure, but cannot guarantee that we will be successful at doing so, given the nature of the Internet. Accordingly, without limitation to Section 4.3 above and Section 11.5 below, you acknowledge that you bear sole responsibility for adequate security, protection and backup of Your Content. We strongly encourage you, where available and appropriate, to use encryption technology to protect Your Content from unauthorized access and to routinely archive Your Content. We will have no liability to you for any unauthorized access or use, corruption, deletion, destruction or loss of any of Your Content.

Source: <http://aws-portal.amazon.com/gp/aws/developer/terms-and-conditions.html>

# Encryption and DbaaS: Functionality





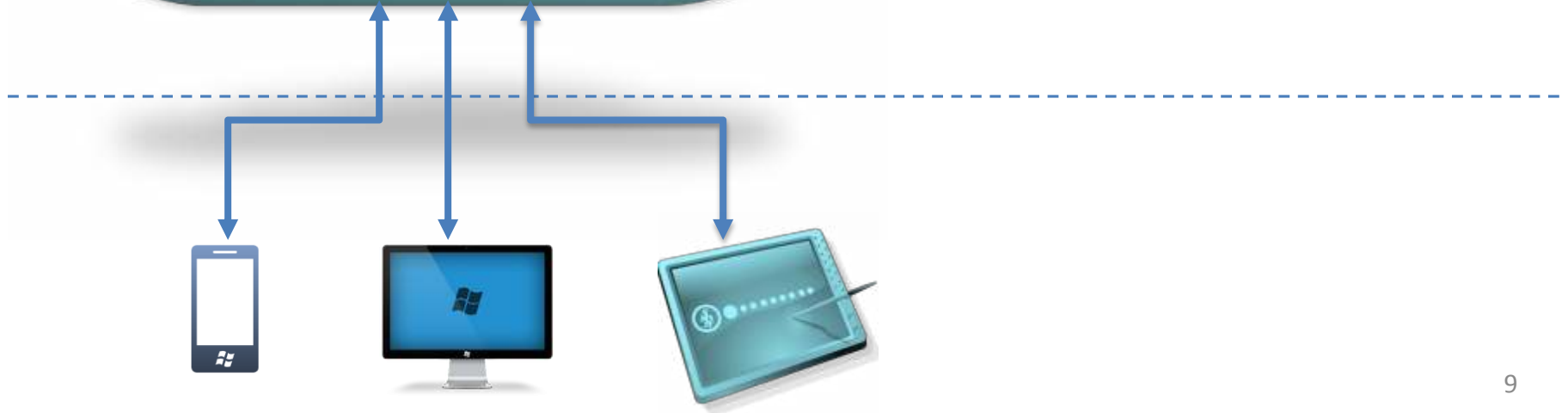
# Example: Online Course Database



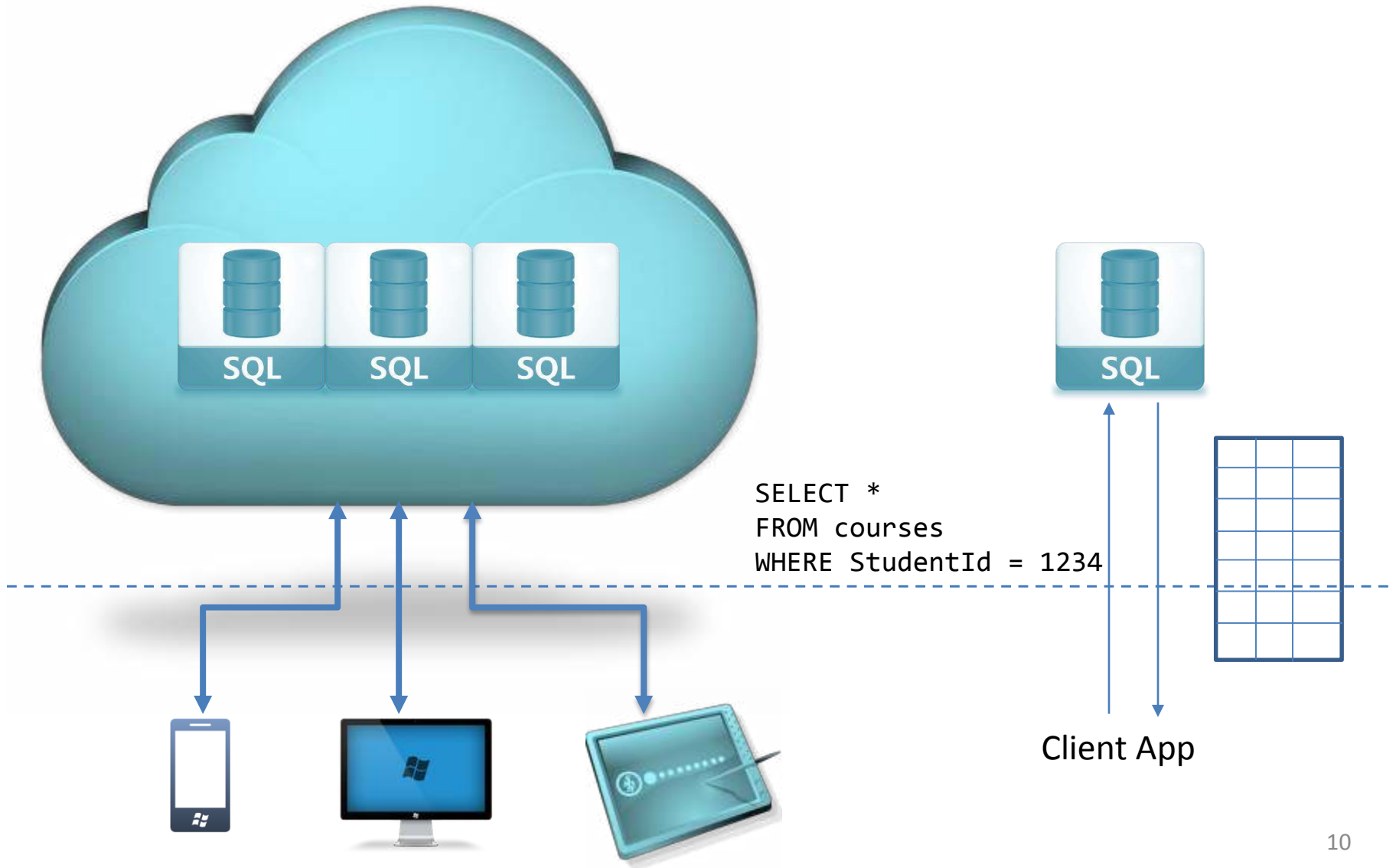
Student					
StudentId	Name	Addr	GPA	CreditCard	...

Course			
CourseId	Name	InstrId	...

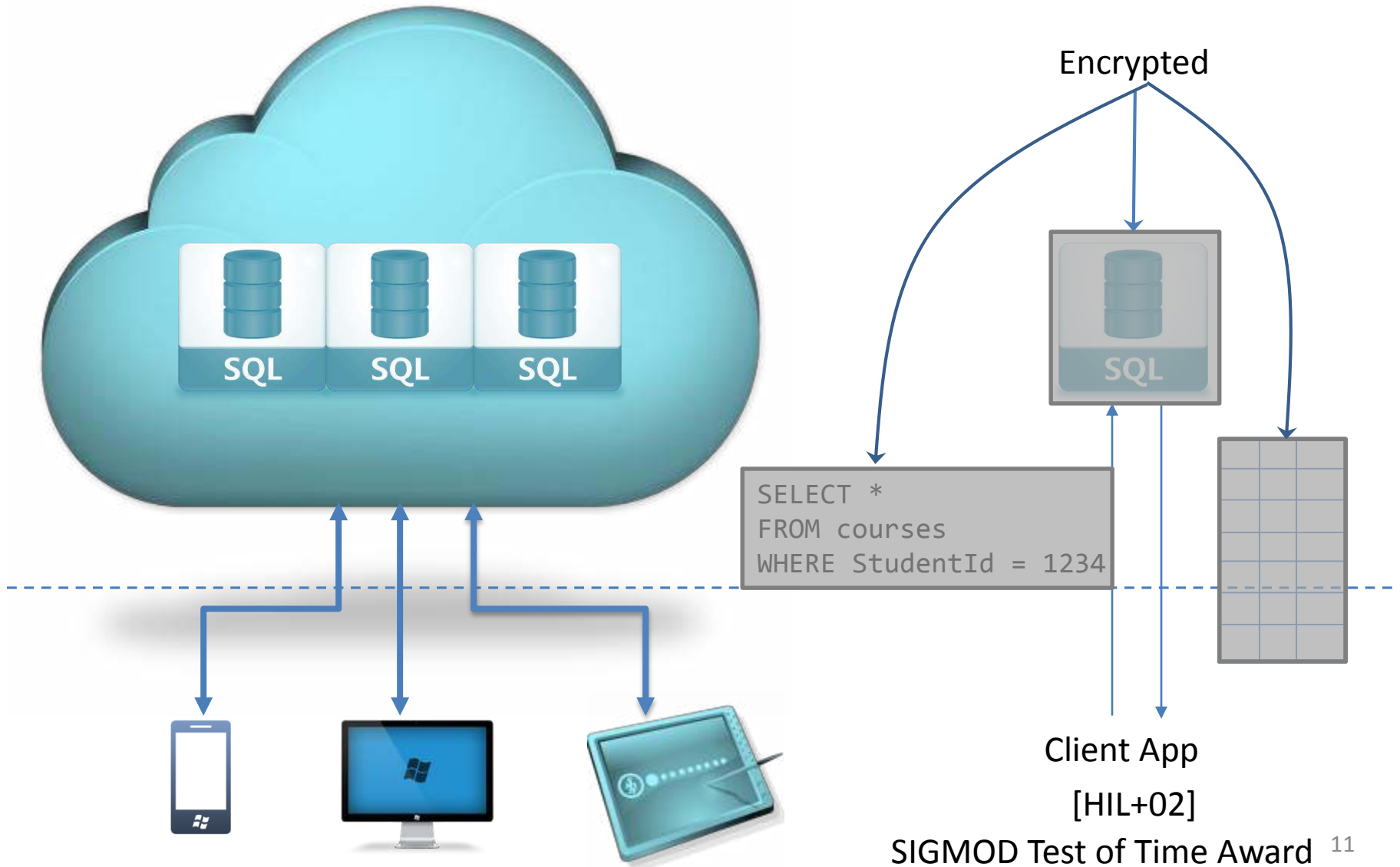
StudentCourse			
CourseId	StudentId	Grade	...



# Encryption and DbaaS: Functionality



# Encryption and DbaaS: Functionality



# Tutorial Overview

- Survey of existing work
  - Building blocks
  - End-to-end systems
    - Security-Performance-Generality tradeoff
    - Taxonomy, organization
- Open problems & Challenges
- Random pontifications

# Tutorial Goals & Non-Goals

- Takeaway goals:
  - Interesting & Important area
  - Lots of open (systems) problems
  - Multi-disciplinary
- Non-goals:
  - Latest advances Elliptic Curve Cryptography

Related tutorial:

Secure and privacy preserving Database Services in the Cloud

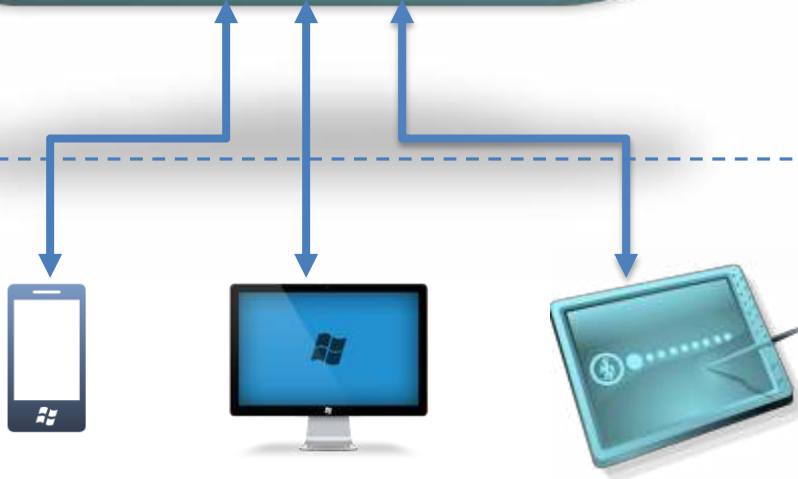
# Roadmap

- Introduction
- Overview
- Basics of Encryption
- Trusted Client based Systems
- Secure In-Cloud Processing
- Security
- Conclusion

# Passive Adversary

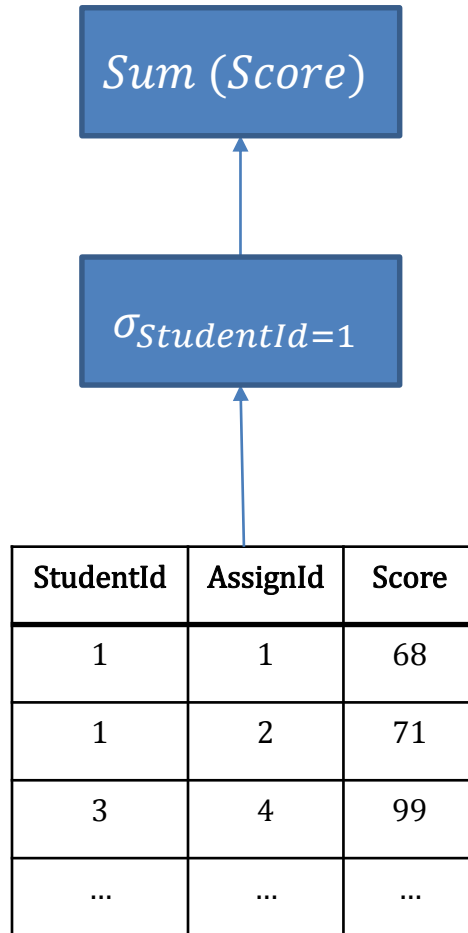


- Passive
- Honest but curious
- Does not alter:
  - Database
  - Results



Design systems for active adversary

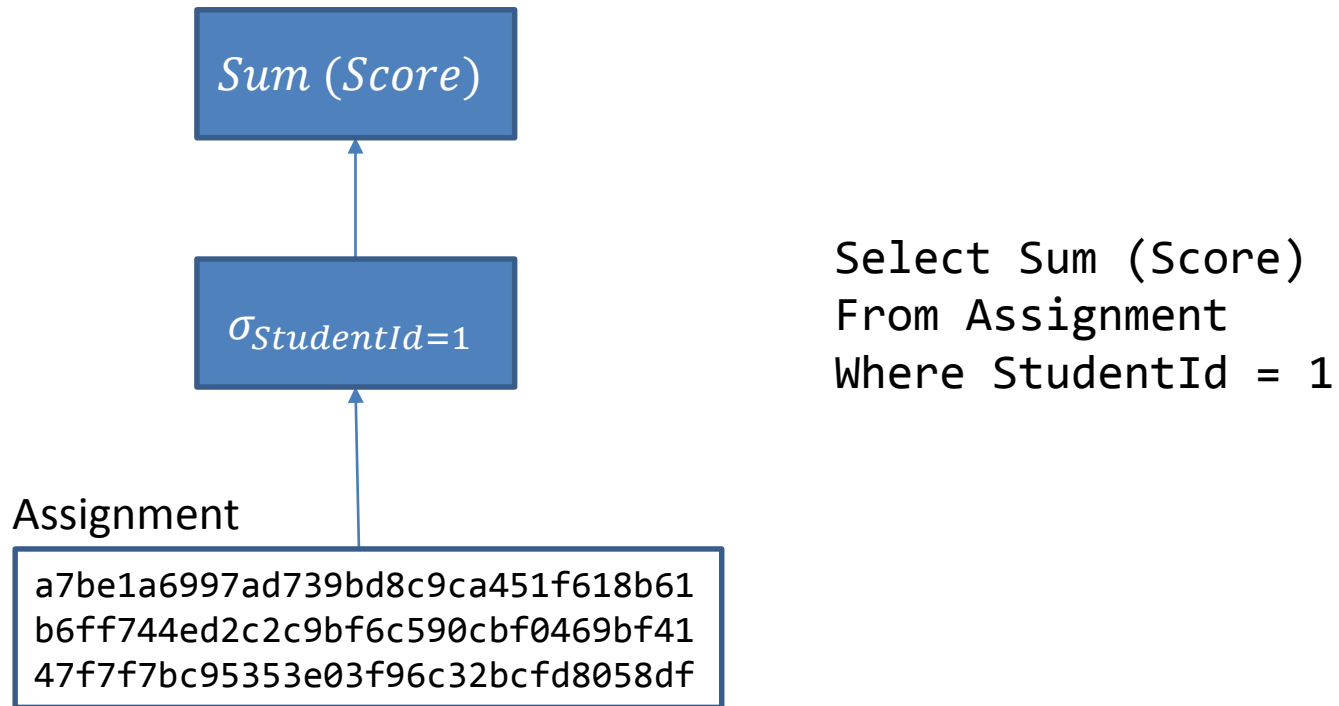
# Encryption: Fundamental Challenge



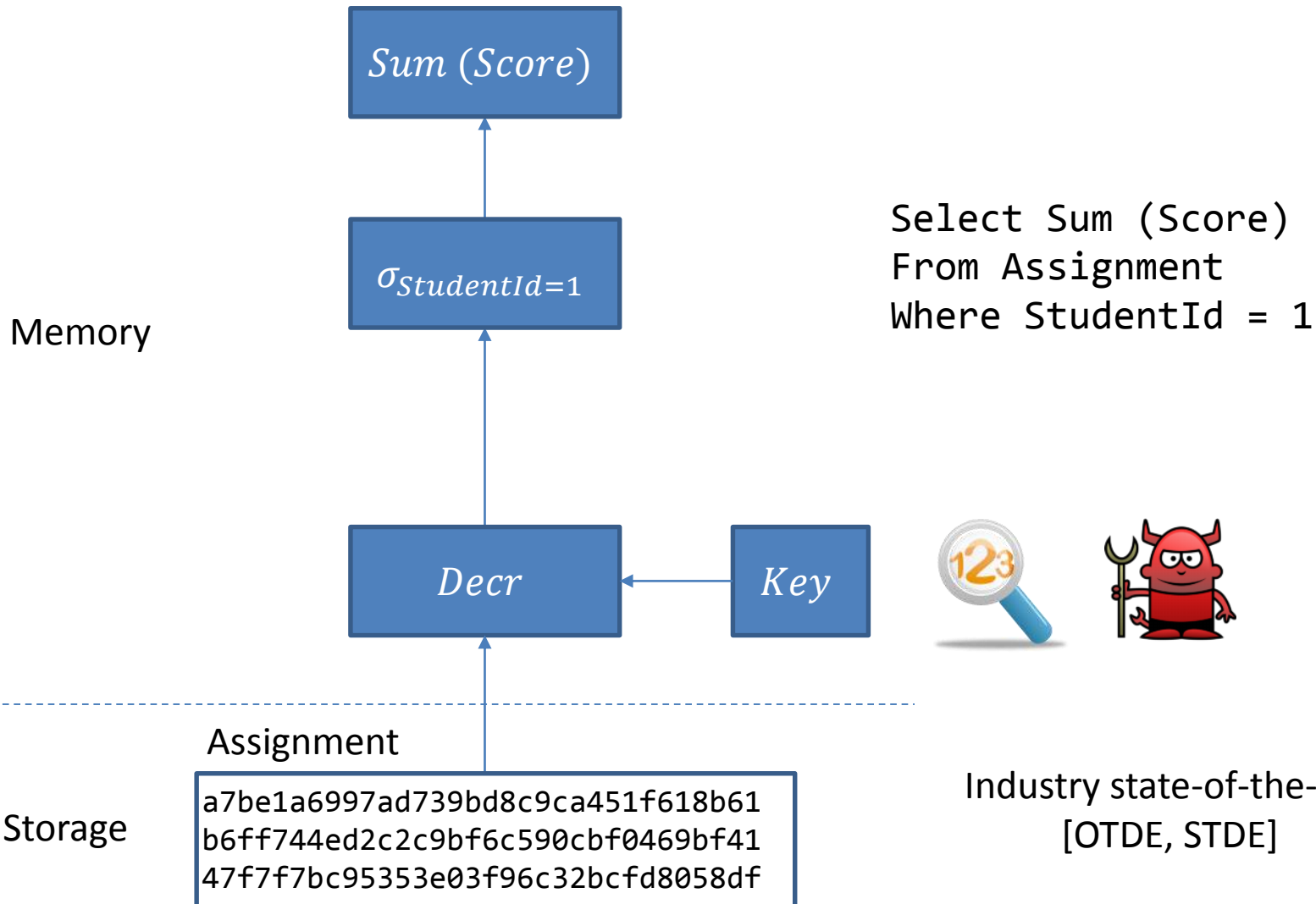
```
Select Sum (Score)
From Assignment
Where StudentId = 1
```



# Encryption: Fundamental Challenge



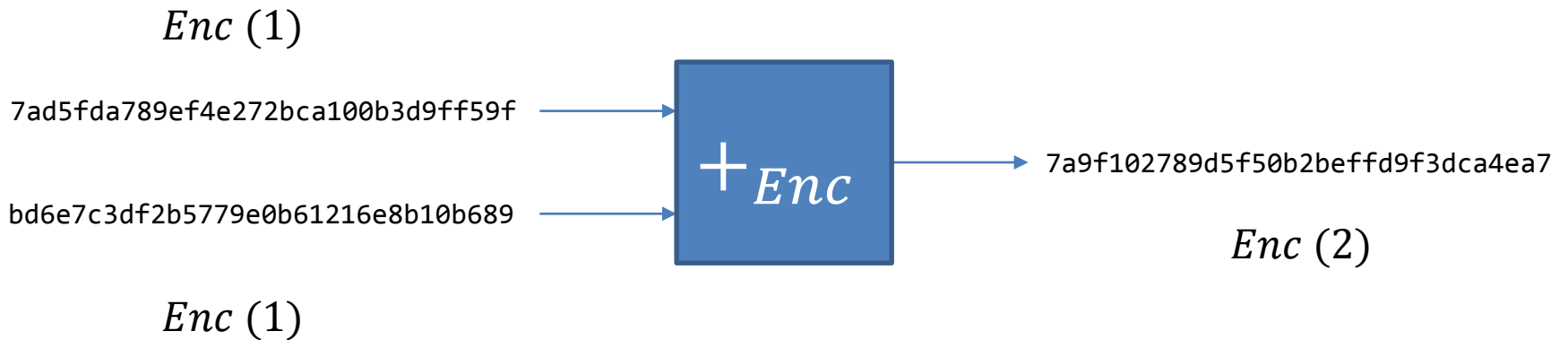
# Encryption: Fundamental Challenge



# Solution Landscape

- Two fundamental techniques
  - Directly compute over encrypted data
    - Special *homomorphic* encryption schemes

# Homomorphic Encryption

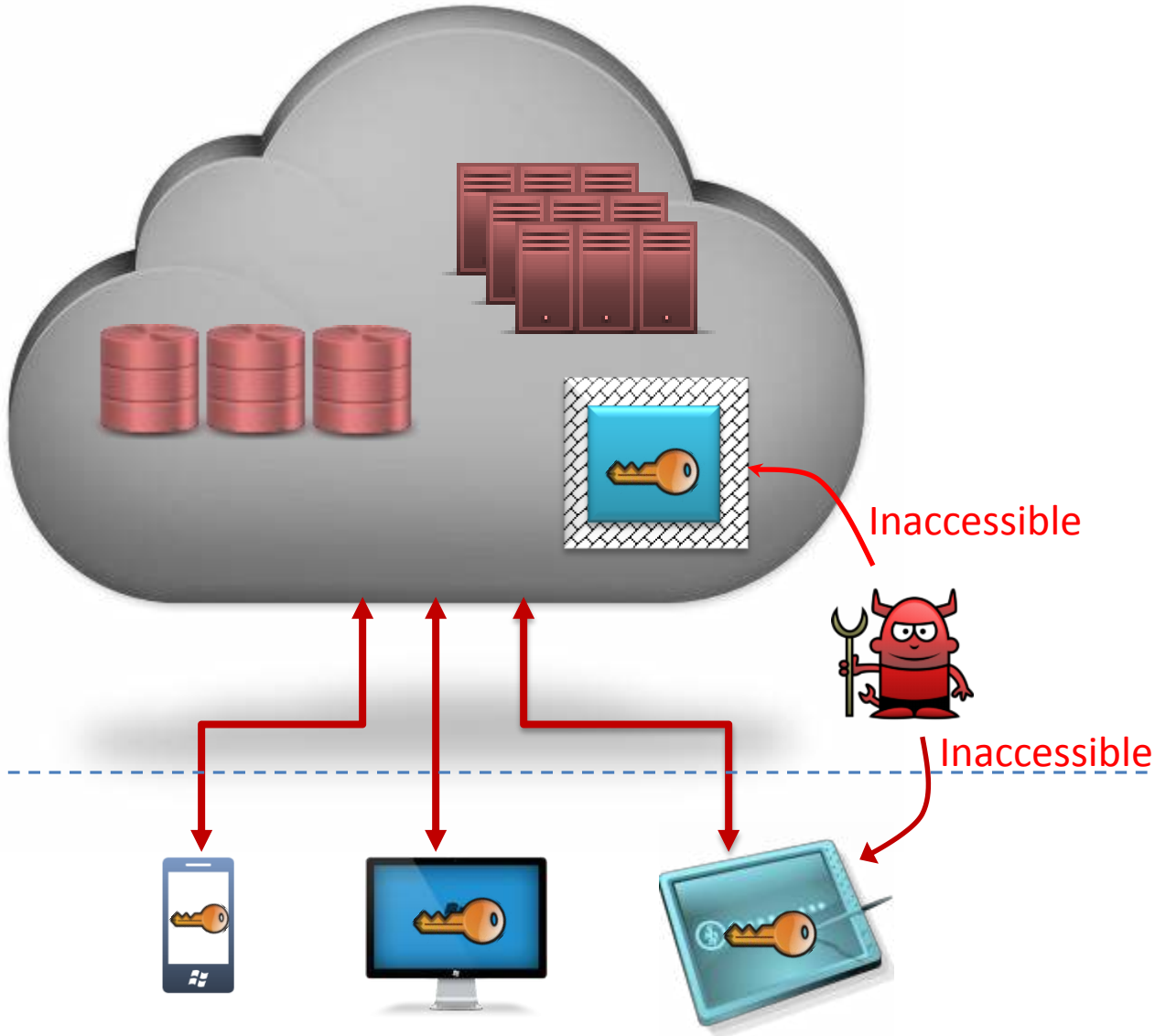


Encryption key is not an input

# Solution Landscape

- Two fundamental techniques
  - Directly compute over encrypted data
    - Special *homomorphic* encryption schemes
    - Challenge: limited class of computations
  - Use a “secure” location
    - Computations on plaintext

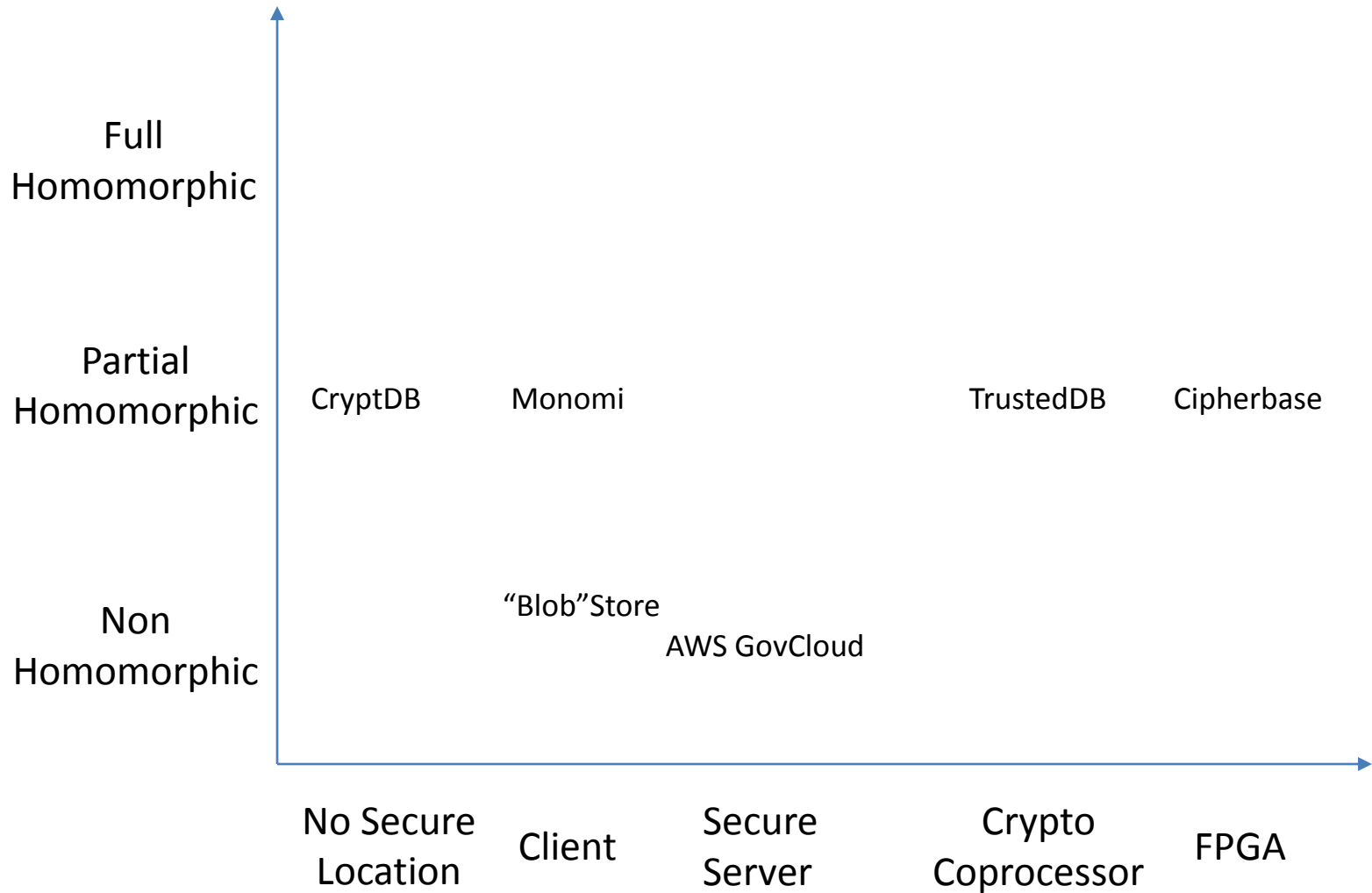
# Secure Location



# Solution Landscape

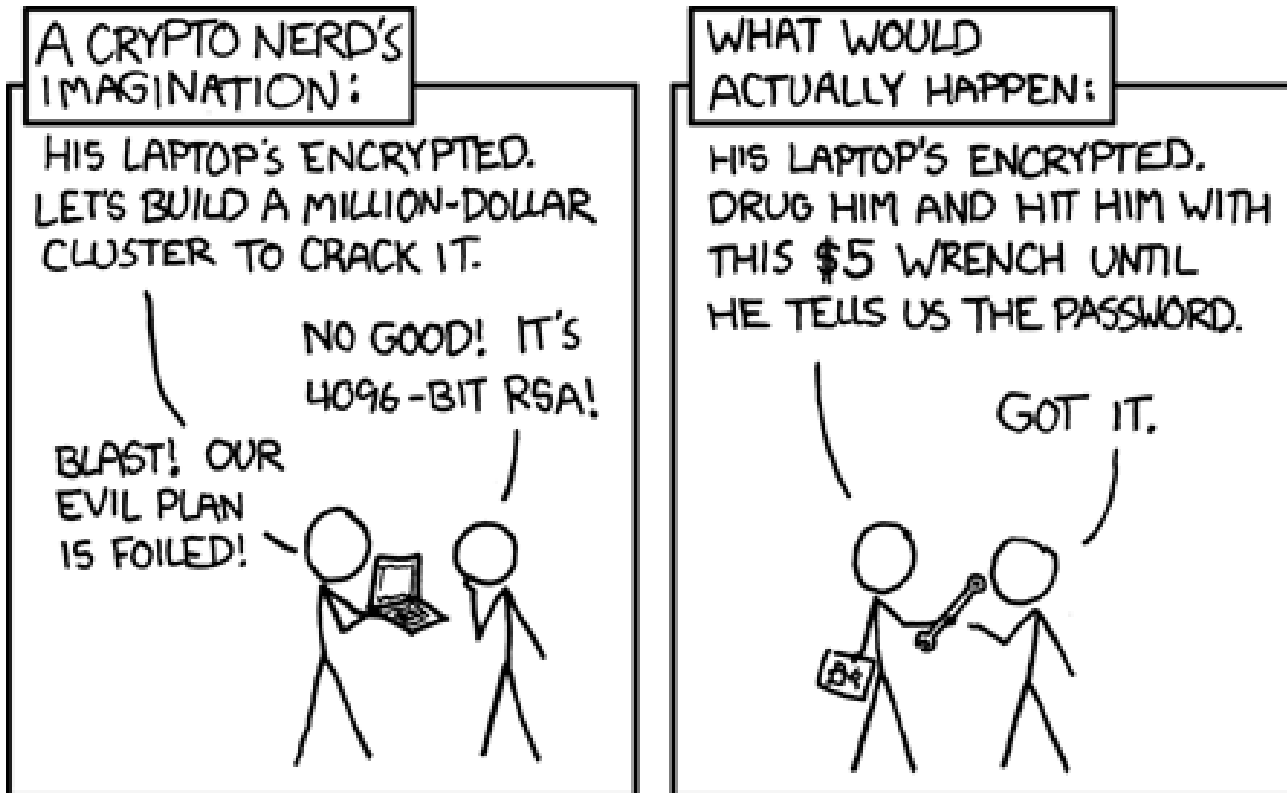
- Two fundamental techniques
  - Directly compute over encrypted data
    - Special *homomorphic* encryption schemes
    - Challenge: limited class of computations
  - Use a “secure” location
    - Computations on plaintext
    - Challenge: Expensive

# Systems Landscape





# Encryption == Security?



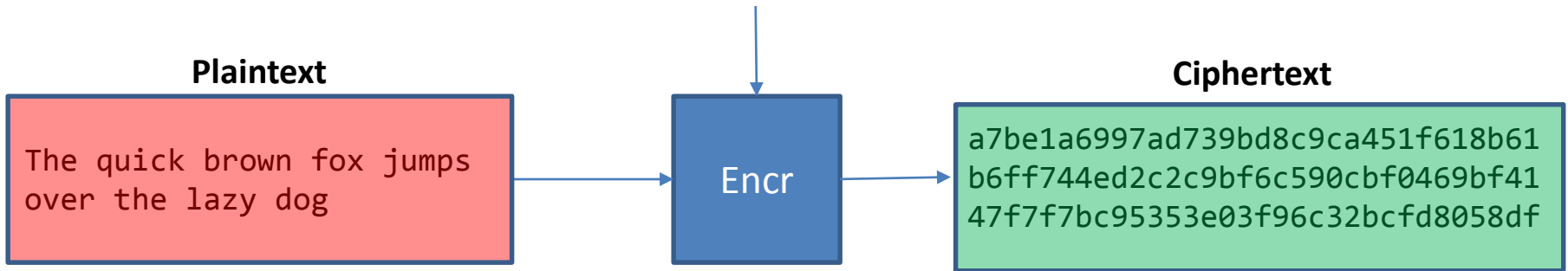
Source: <http://xkcd.com/538/>

# Roadmap

- Introduction
- Overview
- Basics of Encryption
- Trusted Client based Systems
- Secure In-Cloud Processing
- Security
- Conclusion

# Encryption Scheme

Key: 000102030405060708090a0b0c0d0e0f



Ciphertext

a7be1a6997ad739bd8c9ca451f618b61  
b6ff744ed2c2c9bf6c590cbf0469bf41  
47f7f7bc95353e03f96c32bcfd8058df

Decr

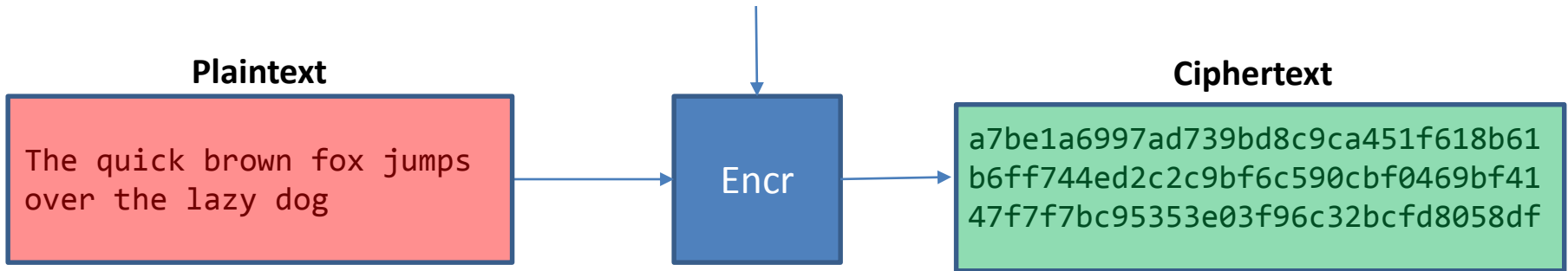
Plaintext

The quick brown fox jumps  
over the lazy dog

Key: 000102030405060708090a0b0c0d0e0f

# Encryption Scheme

Public Key: 000102030405060708090a0b0c0d0e0f



Ciphertext

a7be1a6997ad739bd8c9ca451f618b61  
b6ff744ed2c2c9bf6c590cbf0469bf41  
47f7f7bc95353e03f96c32bcfd8058df

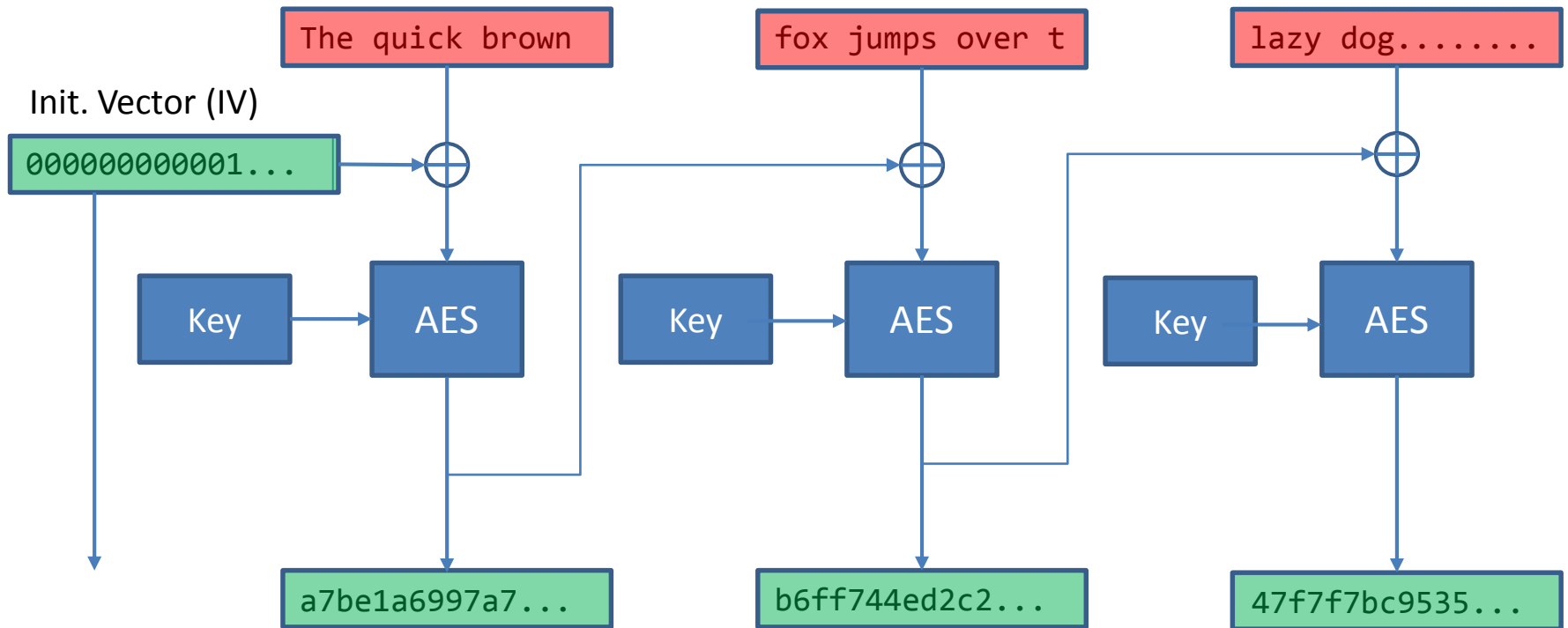


Plaintext

The quick brown fox jumps  
over the lazy dog

Private Key: 47b6ffedc2be19bd5359c32bcfd8dff5

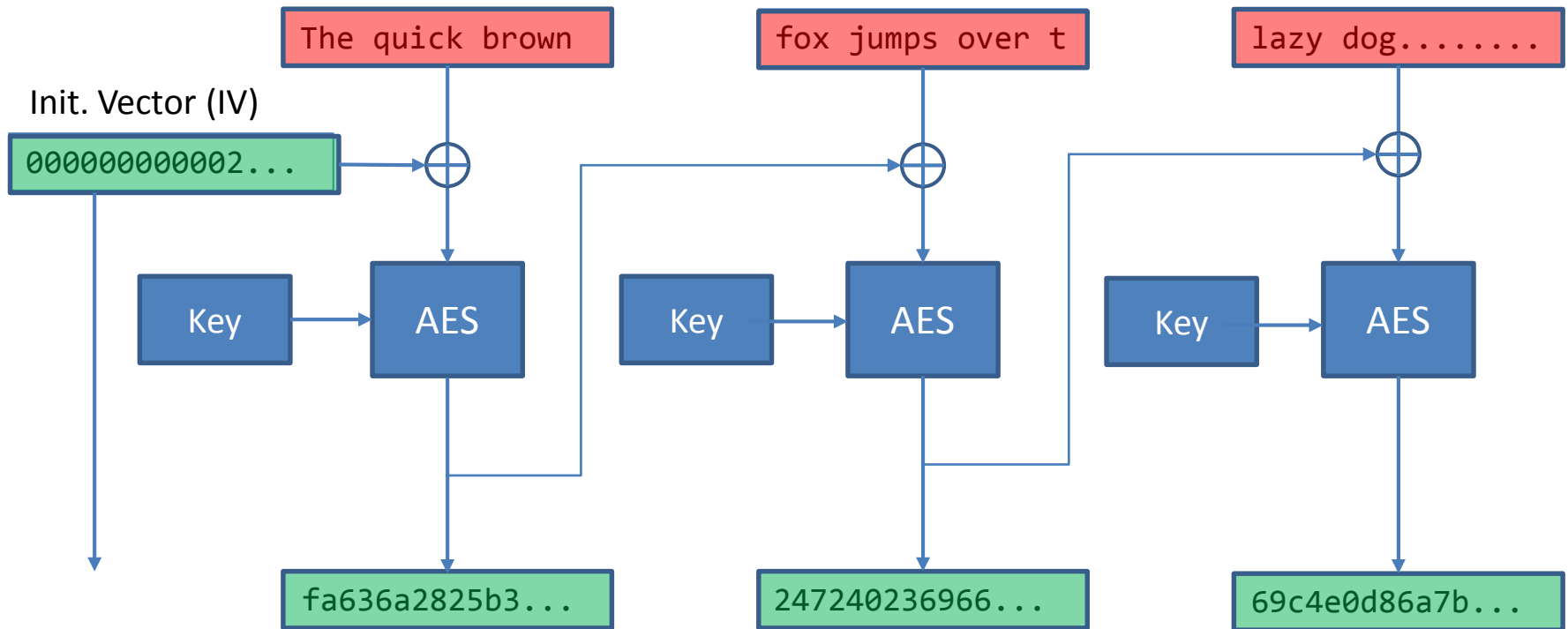
# AES + CBC Mode



Variable IV => Non-deterministic

[AES, KL 07]

# AES + CBC Mode

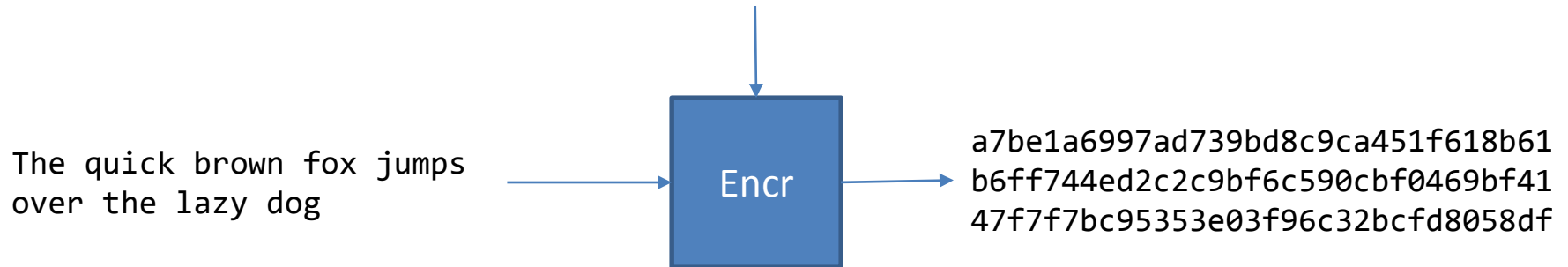


Variable IV => Non-deterministic

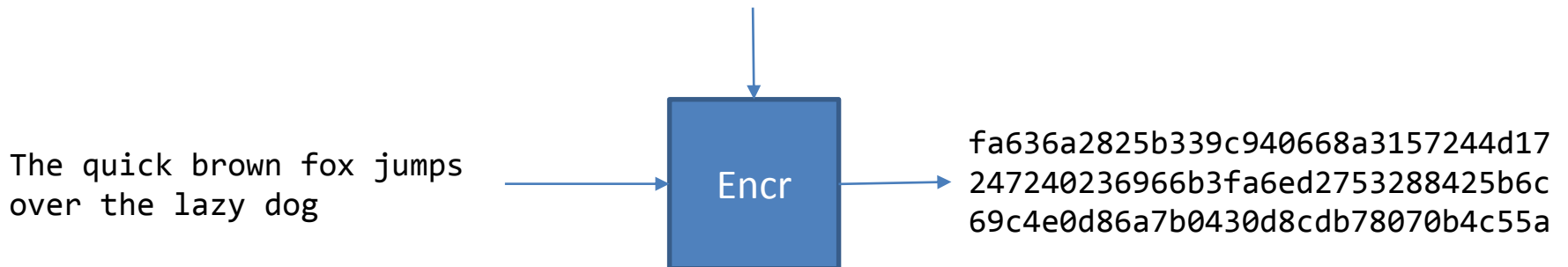
[AES, KL 07]

# Nondeterministic Encryption Scheme

Key: 000102030405060708090a0b0c0d0e0f

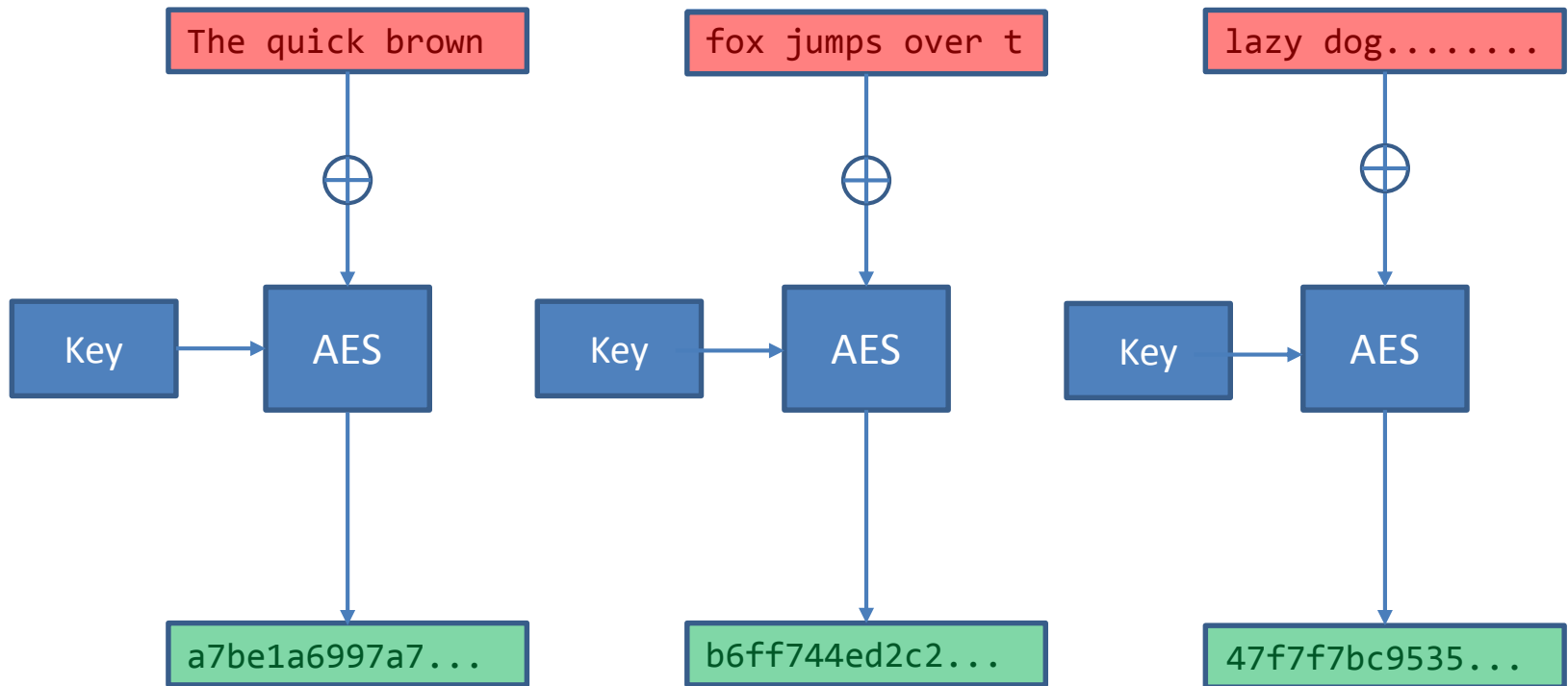


000102030405060708090a0b0c0d0e0f



Example: AES + CBC + variable IV

# AES + ECB Mode

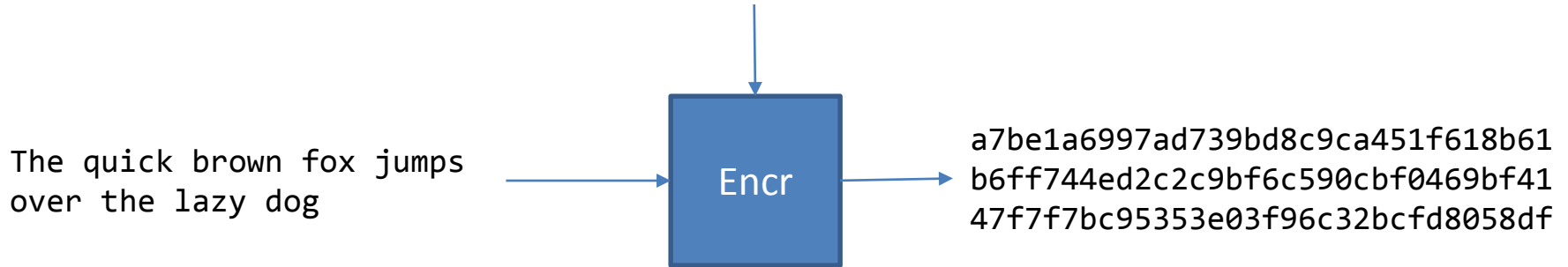


[AES, KL 07]

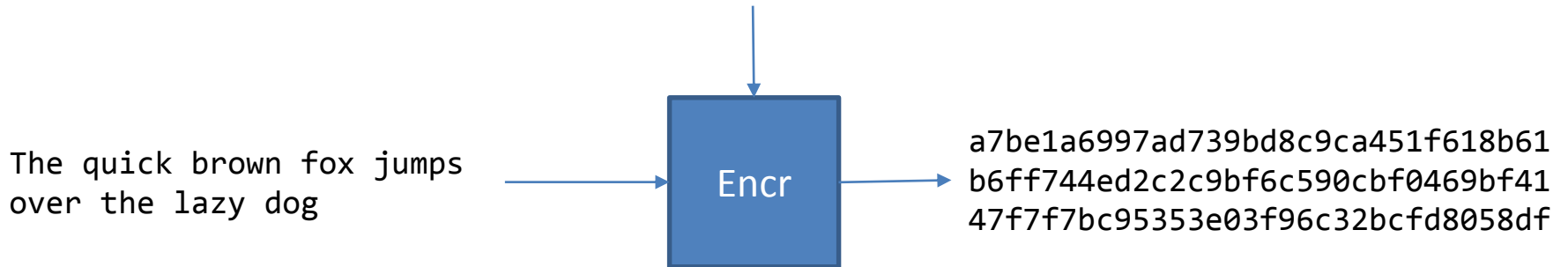


# Deterministic Encryption Scheme

Key: 000102030405060708090a0b0c0d0e0f



000102030405060708090a0b0c0d0e0f



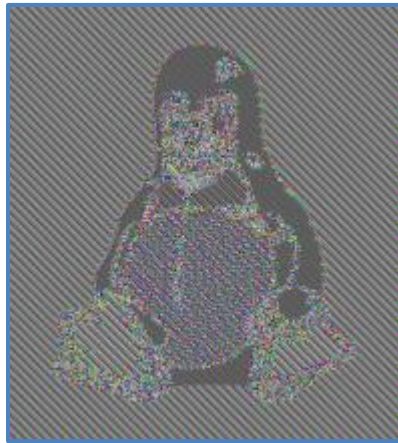
Example: AES + ECB

More secure deterministic encryption: [PRZ+11]

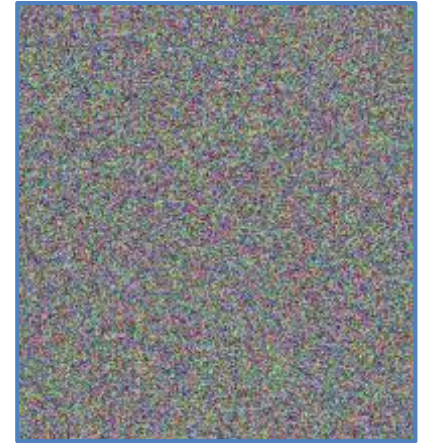
# Strong Security => Non-Deterministic



Original



Deterministic



Non-Deterministic

Source: [http://en.wikipedia.org/wiki/Block\\_cipher\\_modes\\_of\\_operation](http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation)

# Deterministic Encryption

```
select *  
from assignment  
where studentid = 1
```

$\sigma_{StudentId=1}$

StudentId	AssignId	Score
1	1	68
1	2	71
3	4	99
...	...	...

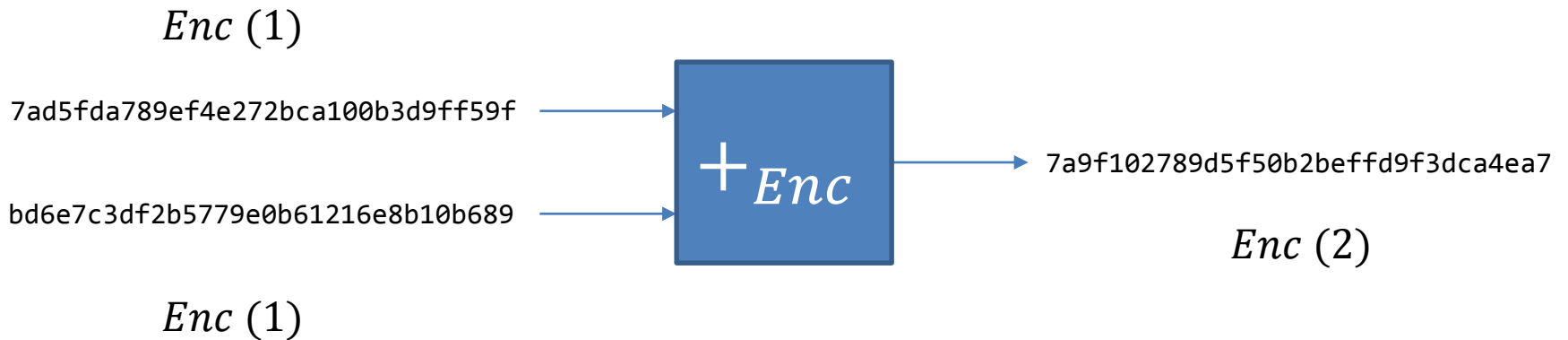
# Deterministic Encryption

```
select *  
from assignment  
where studentid_det = bd6e7c3df2b5779e0b61216e8b10b689
```

$\sigma_{StudentId\_det=bd6\dots}$

StudentId_DET	AssignId	Score
bd6e7c3df2b5779e0b61216e8b10b689	1	68
bd6e7c3df2b5779e0b61216e8b10b689	2	71
7ad5fda789ef4e272bca100b3d9ff59f	4	99
...	...	...

# Homomorphic Encryption



Encryption key is not an input

# Order Preserving Encryption

Value	Enc (Value)
1	0x0001102789d5f50b2beffd9f3dca4ea7
2	0x0065fda789ef4e272bcf102787a93903
3	0x009b5708e13665a7de14d3d824ca9f15
4	0x04e062ff507458f9be50497656ed654c
5	0x08db34fb1f807678d3f833c2194a759e

$$x < y \rightarrow Enc(x) < Enc(y)$$

[BCN11, PLZ13]

# Order-Preserving Encryption

```
select *  
from assignment  
where score >= 90
```

$\sigma_{Score \geq 90}$

StudentId	AssignId	Score
1	1	68
1	2	71
3	4	99
...	...	...

# Order-Preserving Encryption

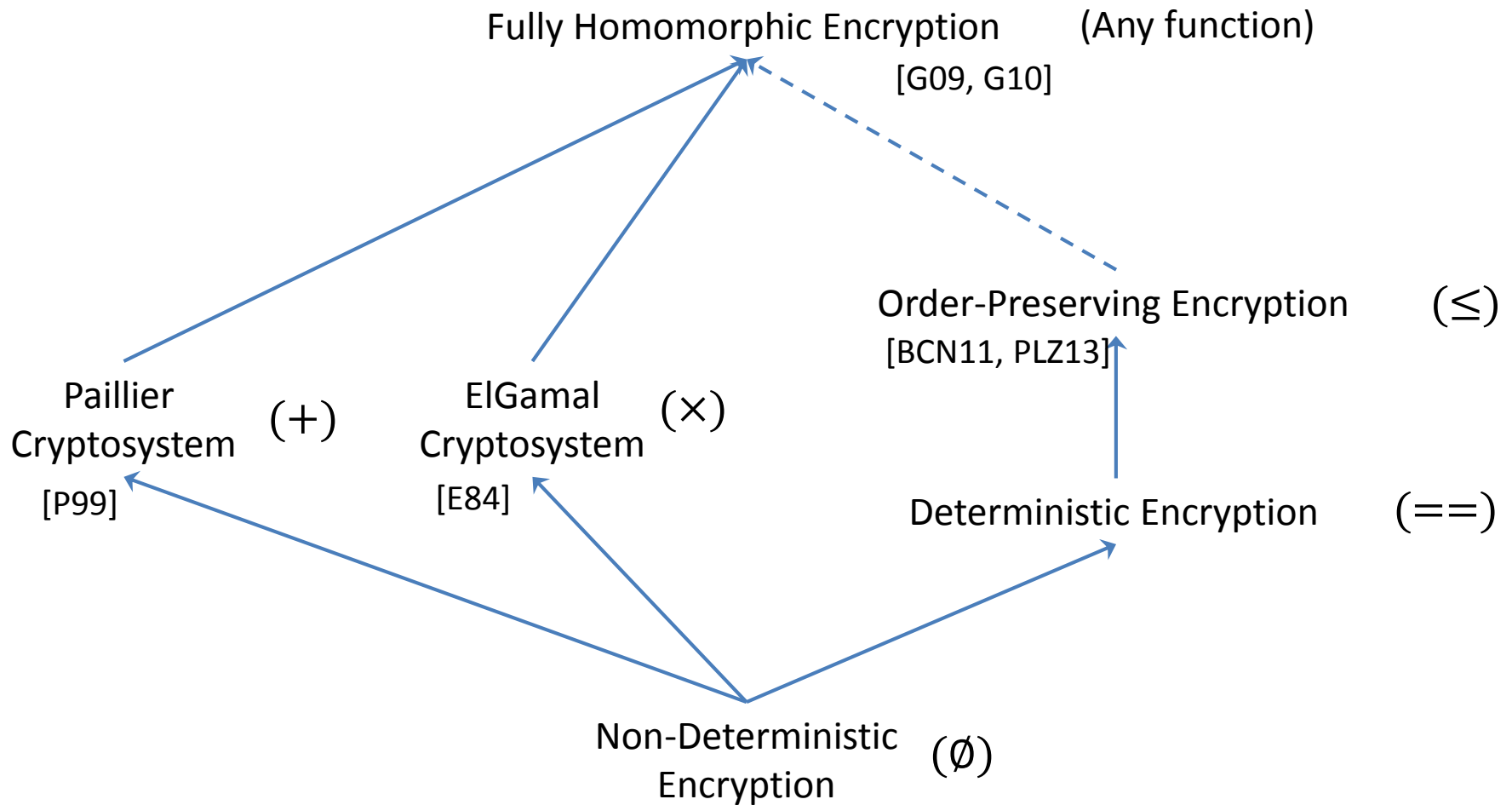
```
select *  
from assignment  
where score_OPE >= 0x04e062ff507458f9be50497656ed654c
```

$\sigma_{\text{score\_ope} \geq 04e0\dots}$

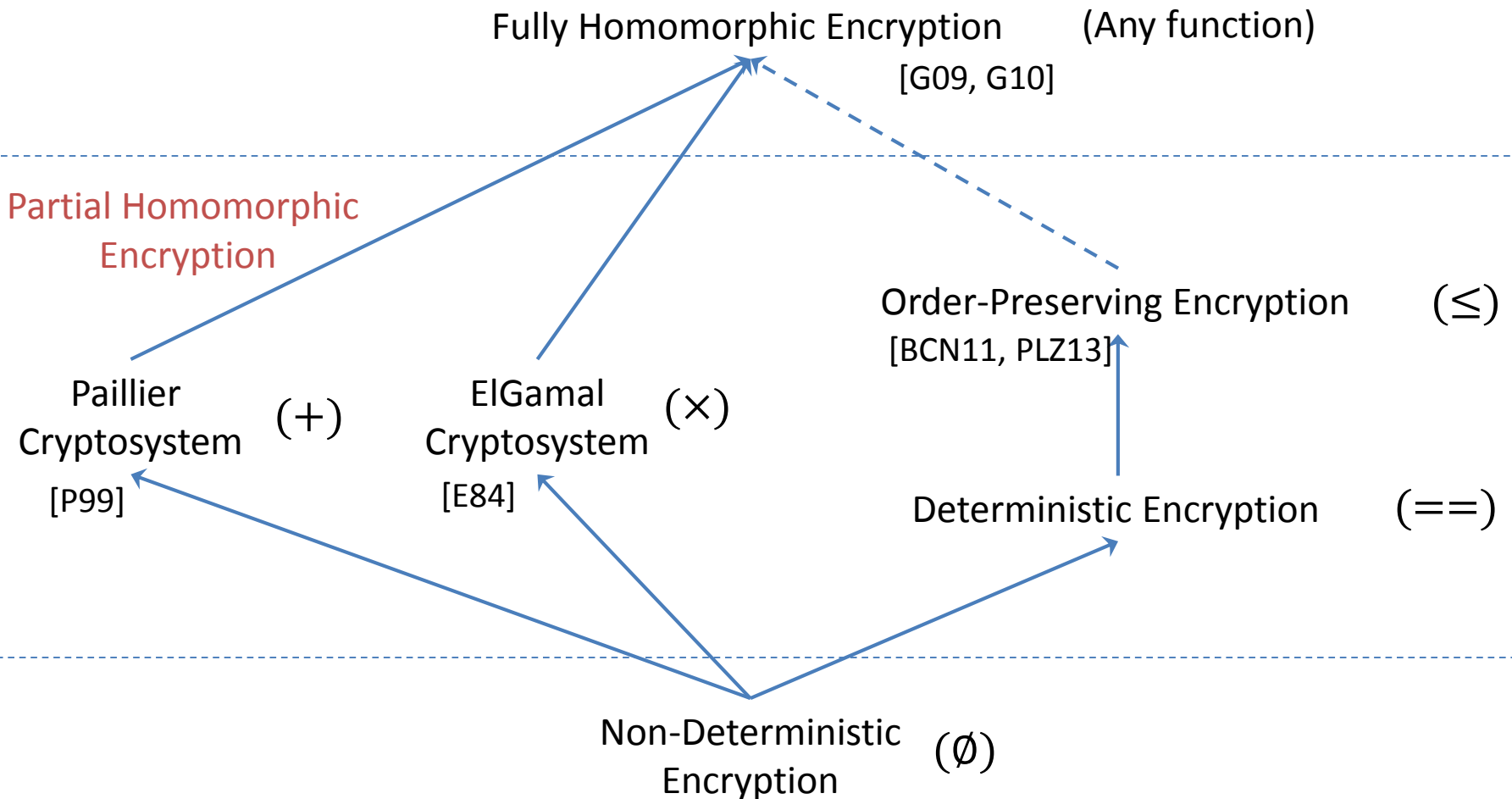
StudentId	AssignId	Score_OPE
1	1	0x0065fda789ef4e272bcf102787a93903
1	2	0x009b5708e13665a7de14d3d824ca9f15
3	4	0x08db34fb1f807678d3f833c2194a759e
...	...	...



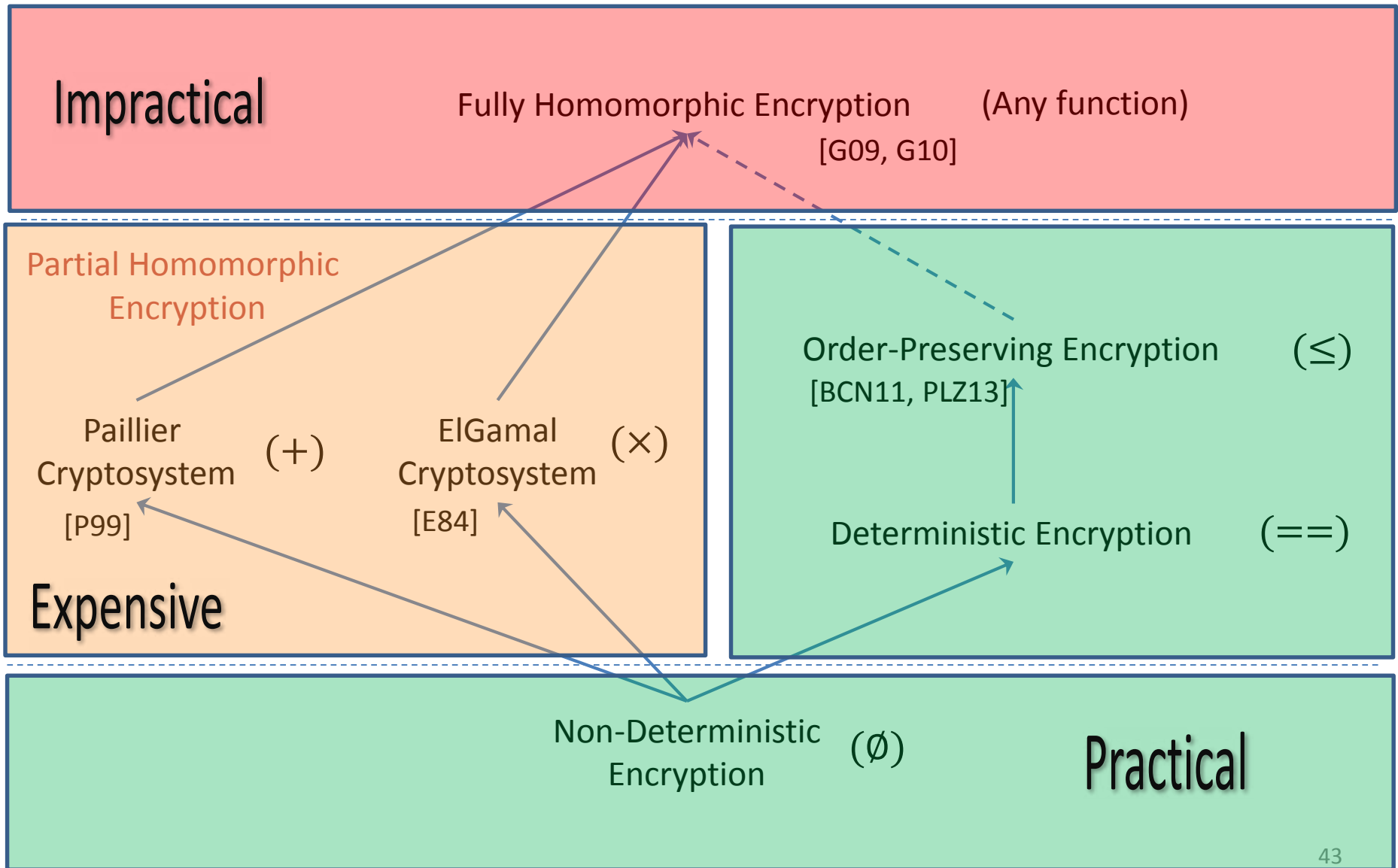
# Homomorphic Encryption Schemes



# Homomorphic Encryption Schemes



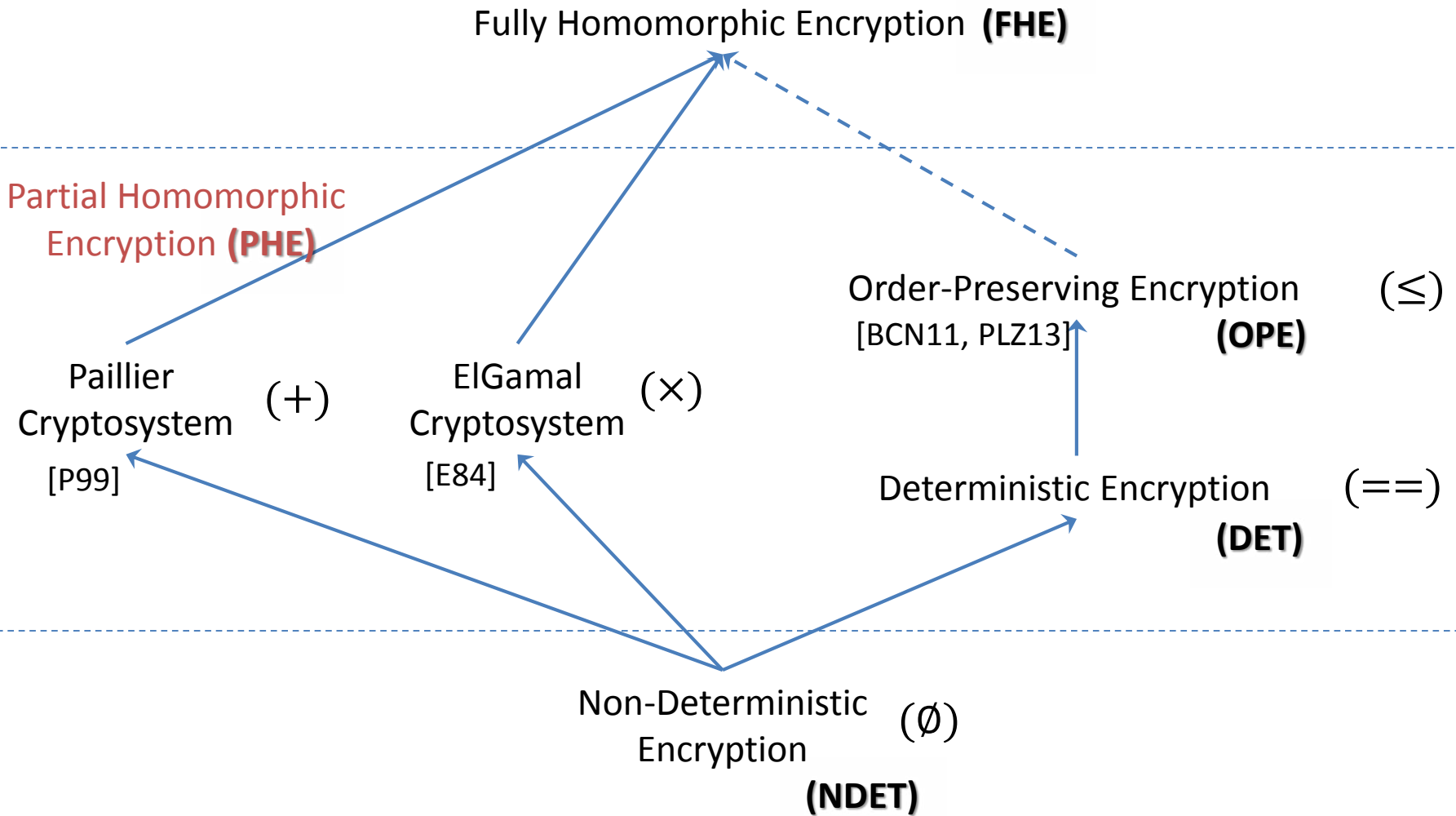
# Homomorphic Encryption Schemes



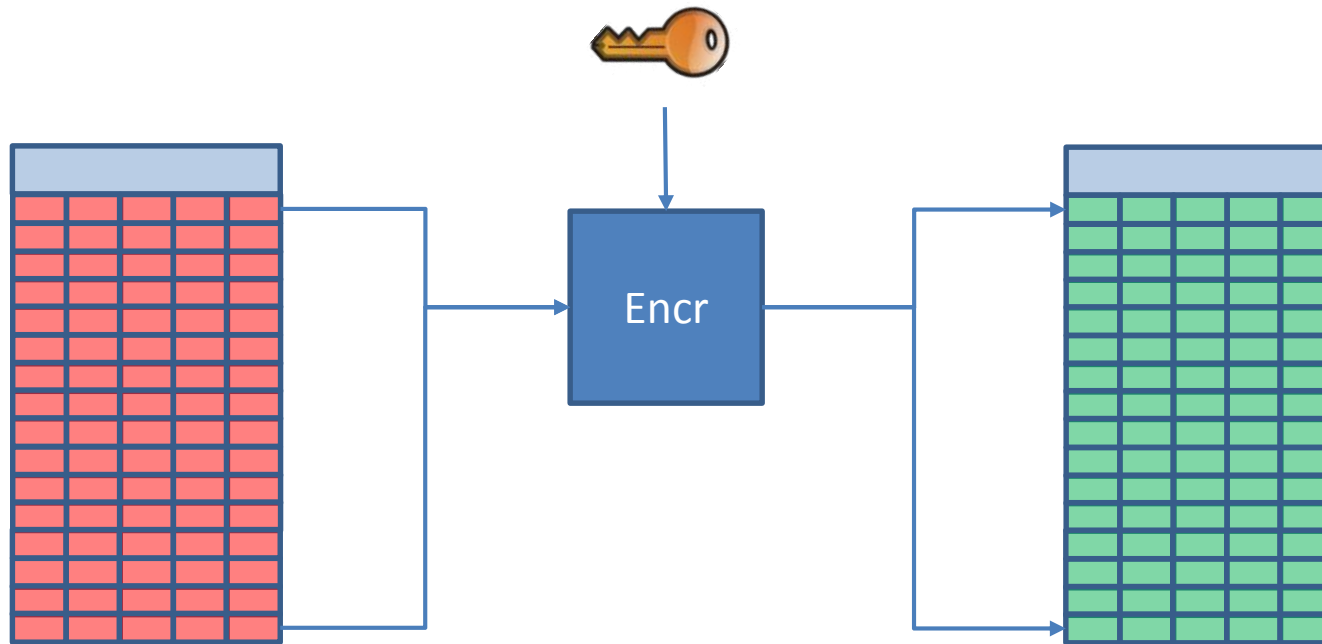
# Homomorphic Encryption Schemes: Performance

Scheme	Space for 1 integer (bits)	Time for 1 operation
Fully Homomorphic Encryption	$2^{14}$	Cosmic time scales
Paillier ElGamal	2048	$\approx$ ms
Deterministic Order-preserving	128	$\approx$ $\mu$ s

# Homomorphic Encryption Schemes: Notation



# How do I Encrypt a Database?

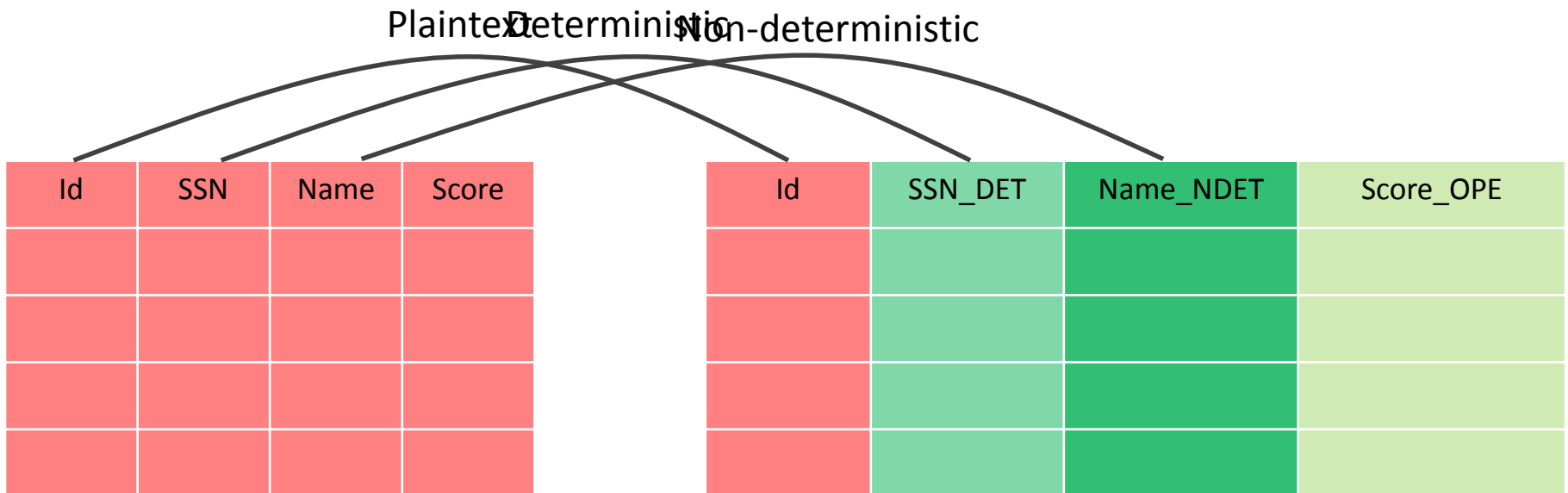


Cell granularity

Advantage:

- Random access to a cell contents
- Mix n Match encryption

# Mix n Match Encryption



Not covered: Deriving multiple keys. See [PRZ+11] for an example.

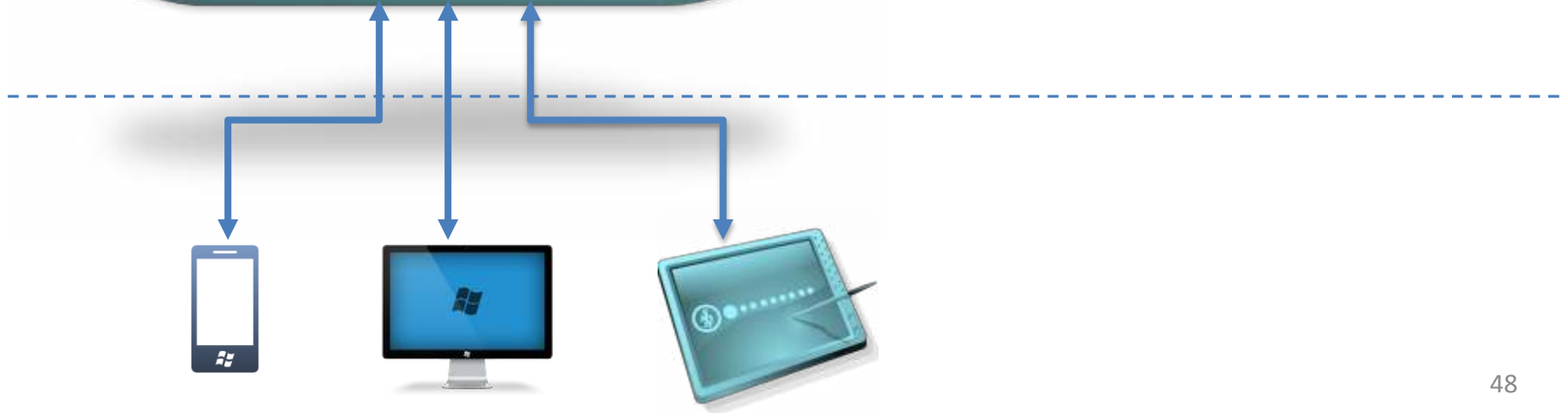
# Example: Online Course Database



Student					
StudentId	Name	Addr	GPA	CreditCard	...

Course			
CourseId	Name	InstrId	...

StudentCourse			
CourseId	StudentId	Grade	...

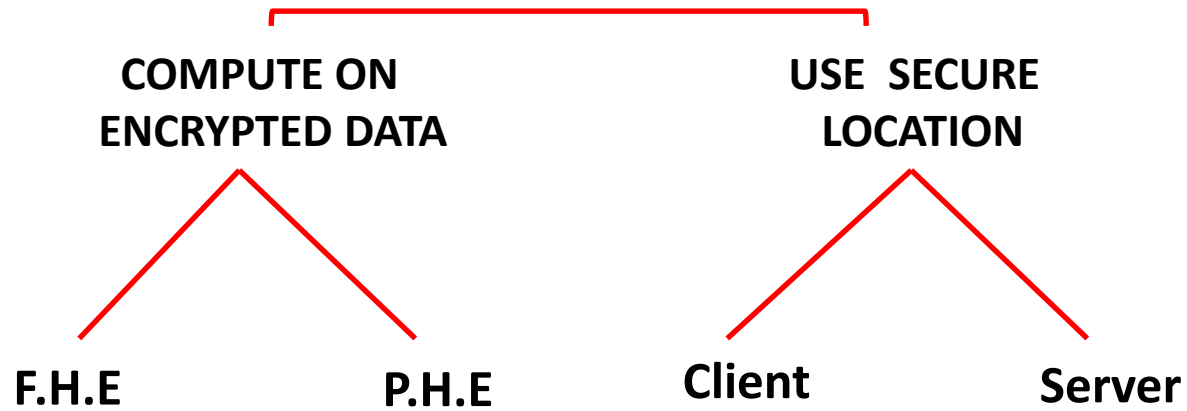




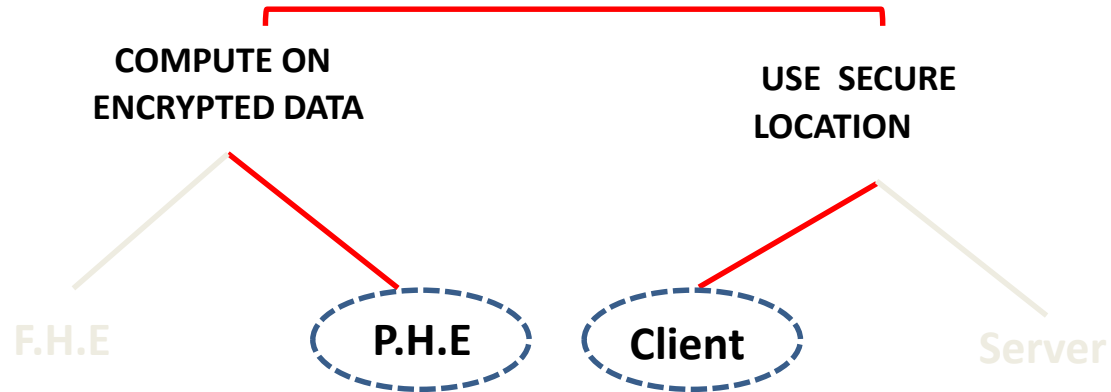
# Roadmap

- Introduction
- Overview
- Basics of Encryption
- Trusted Client based Systems
- Secure In-Cloud Processing
- Security
- Conclusion

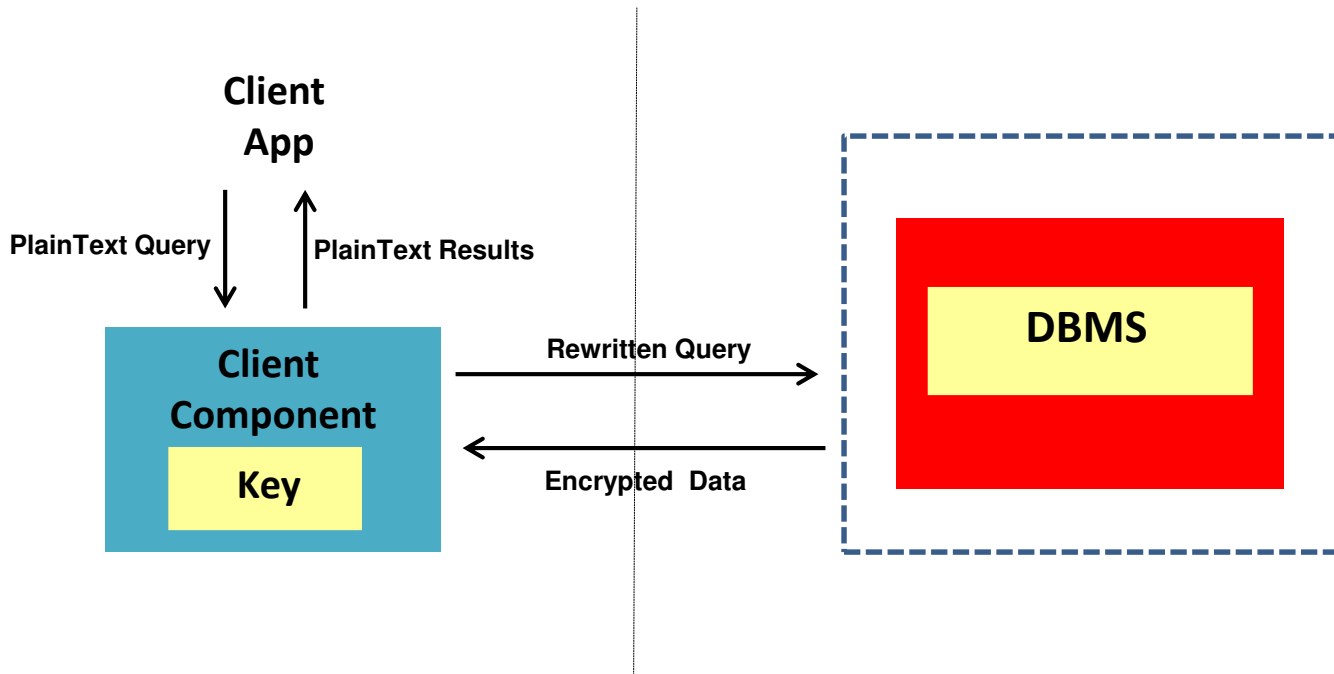
# Design Choices



# Trusted Client based Systems



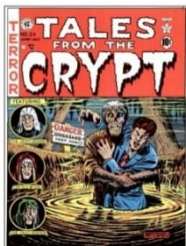
# Trusted Client Architecture



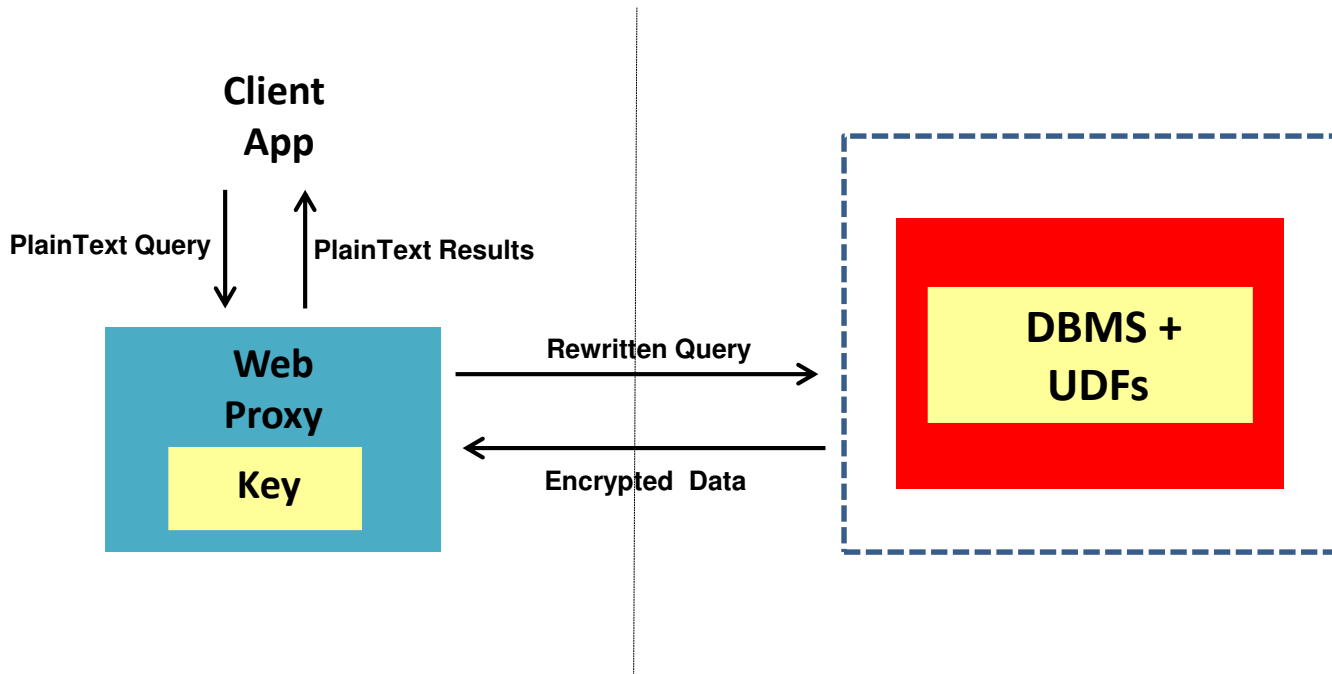
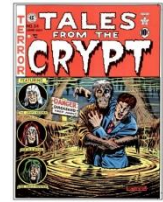
- Data not decrypted in DBMS
  - Only ciphertext seen in the DBMS
- No changes to DBMS/Client App

# Systems

- Minimal Client Computation
  - Use P.H.E (Cryptdb)
- Residual Query Processing in Client
  - Blob Store
  - Use in conjunction with P.H.E (Monomi)



# CryptDB Architecture

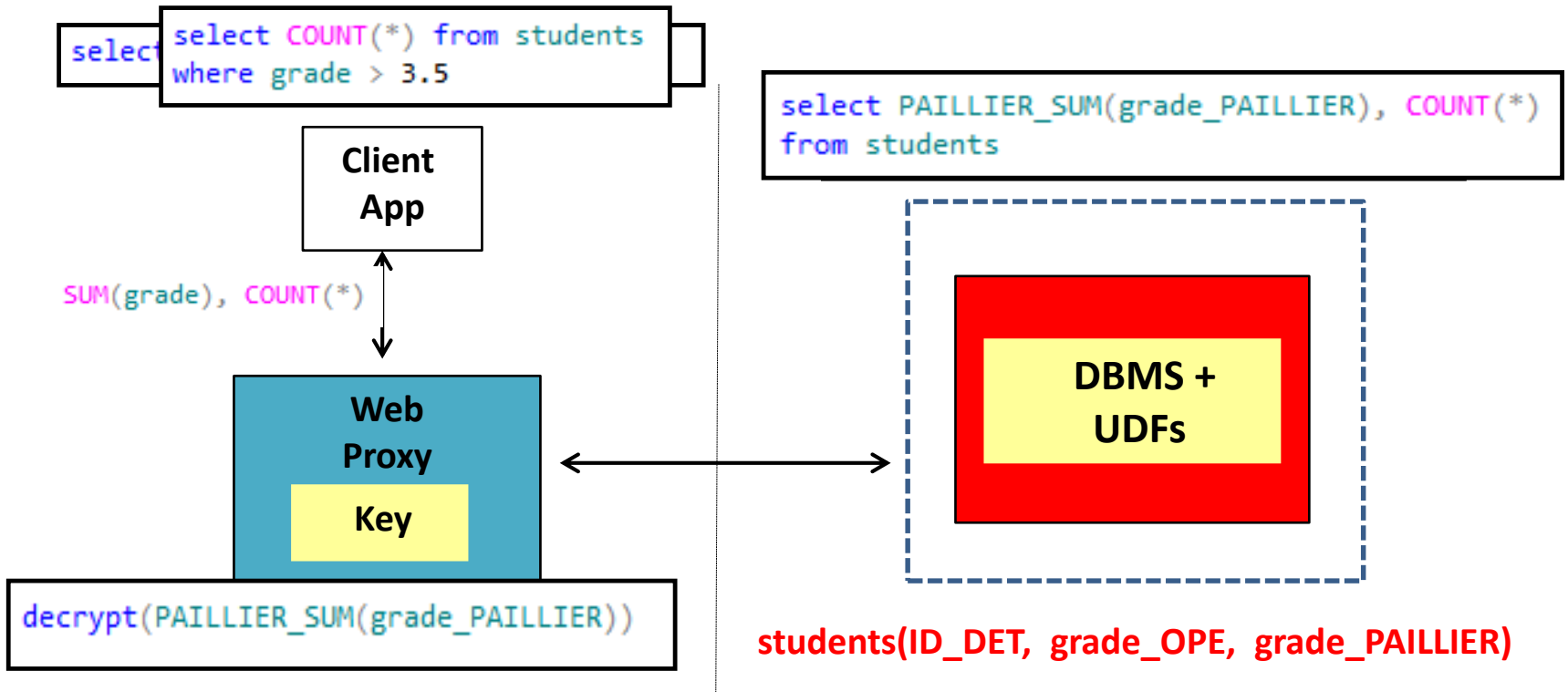


- Web proxy rewrites queries, decrypts result
- Leverage P.H.E techniques

# Database Design

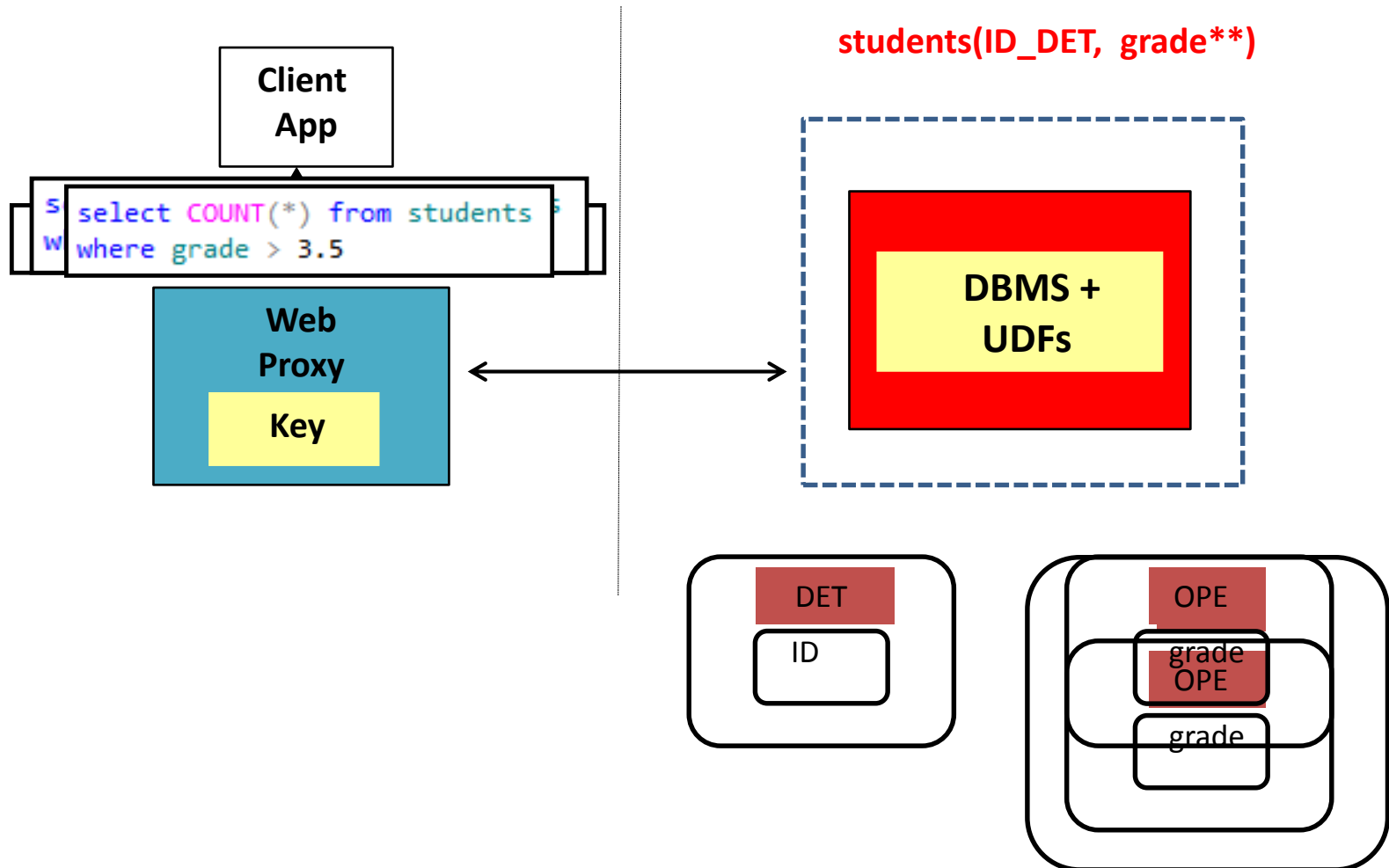
- `students(ID, grade)`
  - Point Lookups on ID column
  - SELECT and AGGREGATION queries on grade
- `students(ID_DET, grade_OPE)`
- `students(ID_DET, grade_OPE, grade_PAILLIER)`
  - Need to store columns encrypted in multiple ways
  - Static/Dynamic design based on workload

# Query Processing

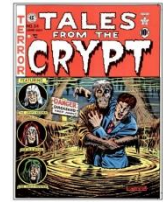




# Dynamic Database Design



# Summary



- P.H.E is not “free” – space overheads
  - For Paillier, to store one integer (32 bits), the ciphertext need to use 2048 bits!
  - Compact representation for paillier that is updatable – open problem.



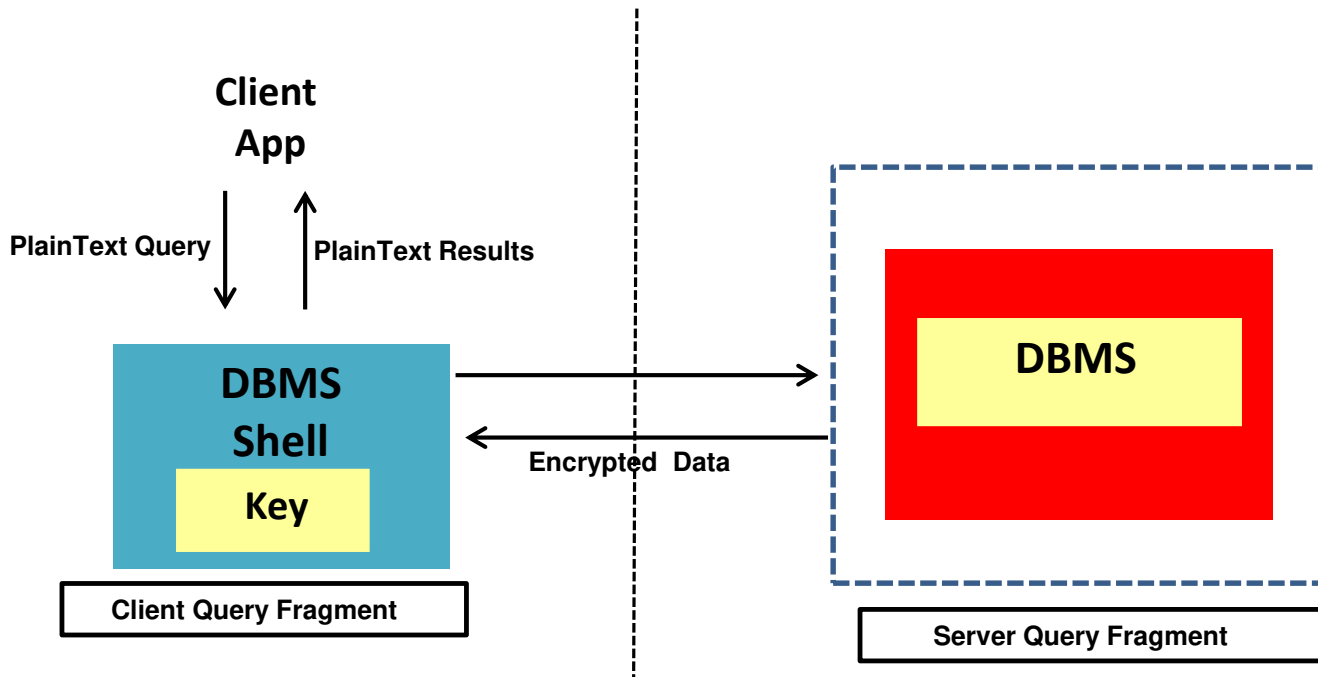
- P.H.E is inherently limited – cannot address all of SQL

```
select STDEV(grade) from students
```

# Systems

- No Client Computation
  - Leverage P.H.E
  - e.g., Cryptdb
- Residual Query Processing on Client
  - e.g., Blob store
  - Use in conjunction with P.H.E (e.g., Monomi)

# Computation in Trusted Client



- Distributed query processing between DBMS shell and untrusted DBMS

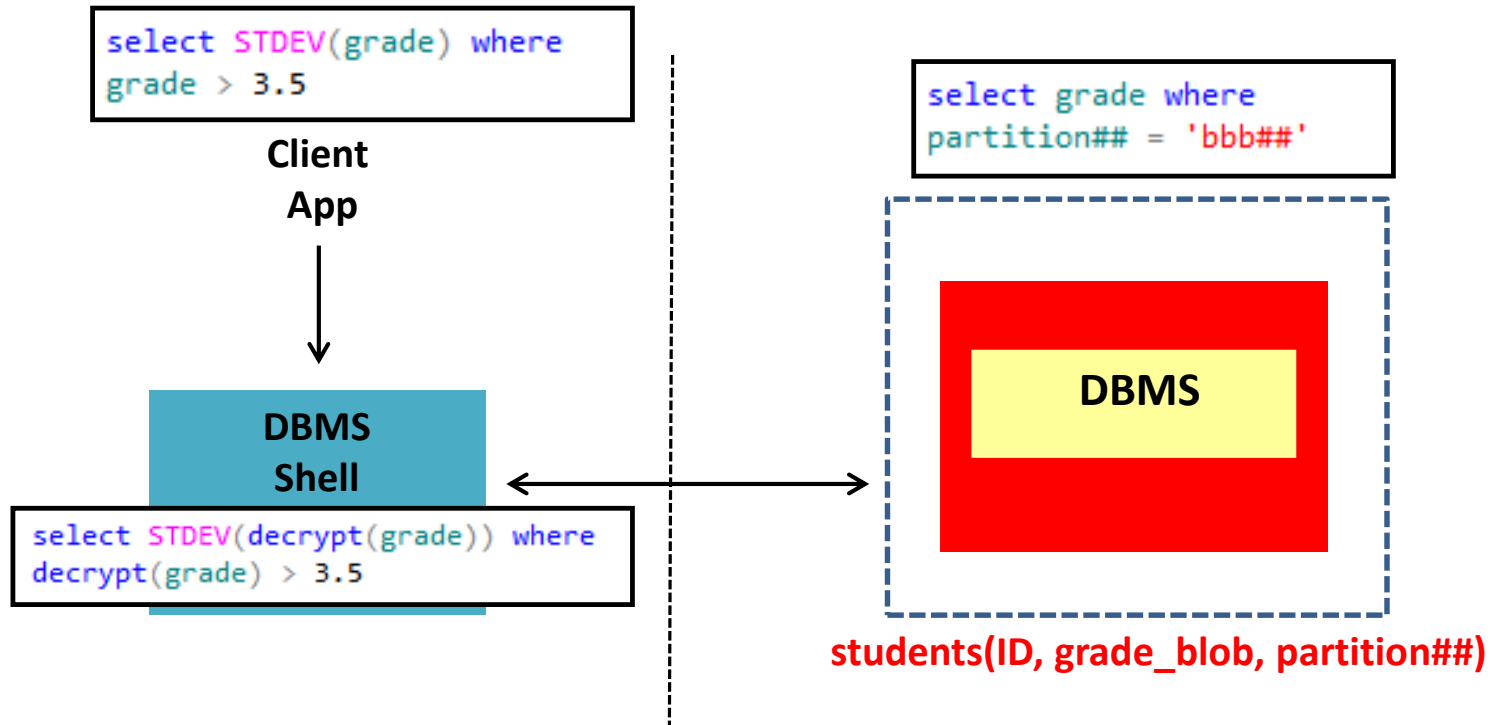
# Blob Store: Database Design



- Encrypted data stored as ‘blobs’ (No computation)
  - `students(ID, grade_blob)`
- Use additional “fake” partitions to index blobs
  - `students(ID, grade_blob, partition##)`

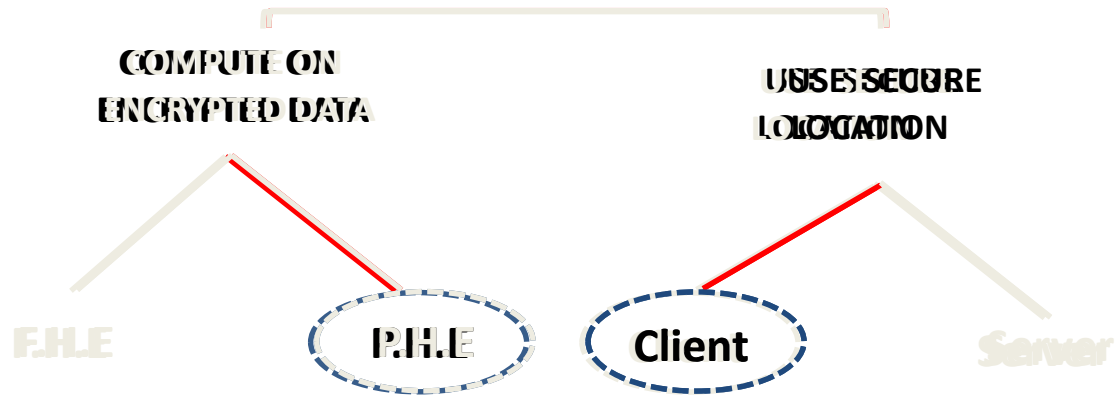
grade	partition##
0 - 1.0	<code>ccc##</code>
1.0 - 2.0	<code>aaa##</code>
2.0 - 3.0	<code>ddd##</code>
3.0 - 4.0	<code>bbb##</code>

# Blob Store: Query Processing

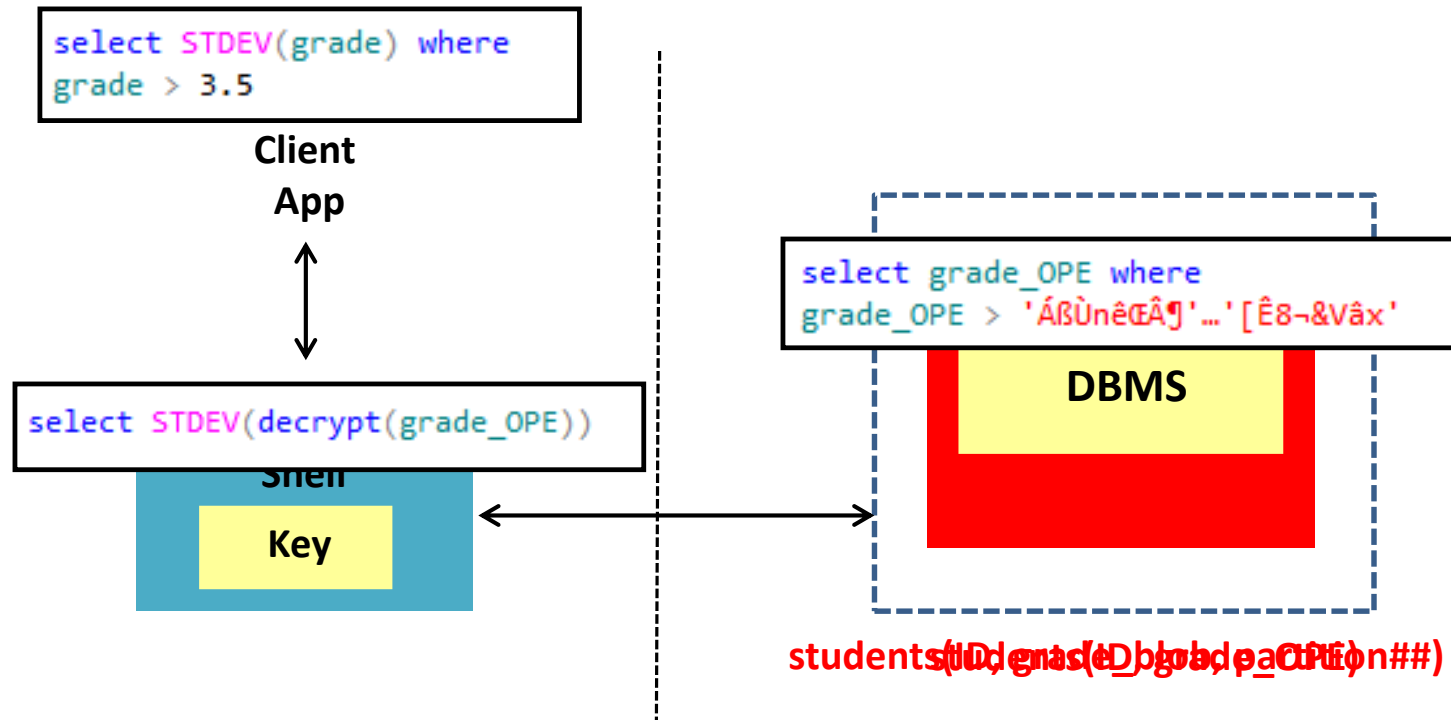
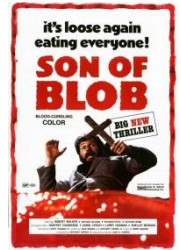


- Distributed query processing
  - Choosing appropriate partitioning
  - “Optimal” Query Splitting

# Trusted Client based Systems



# Augmenting Blob Store



- Use P.H.E to push more computation to DBMS
  - Monomi



# Pre-computation for complex queries

- Find student submissions that have been handled in a day late

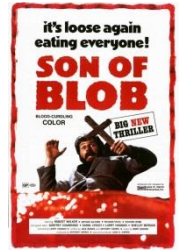
```
select id, count(*) from students
where submissiondate = deadline + 1
group by id
```

- Students(ID\_DET, submissiondate\_DET, **deadline\_DET**, **deadline\_plusone\_DET**)
  - Cannot “Mix and Match” different encryptions

```
select id, count(*) from students
where submissiondate = deadline_plusone
group by id
```



# Trusted Client: Summary



- No Server changes required to DBMS
- Works well for workloads where amount of data shipped is small
  - Physical Design is important for distributed queries
  - Pre-computation is not free
- Generality of approach is unproven
  - Integrity constraints, Triggers etc.
  - Automated tools to migrate database applications



# “Key” Limitation: Robustness

- Stored procedure to find student submissions that have been handed in late (with delay as a parameter)

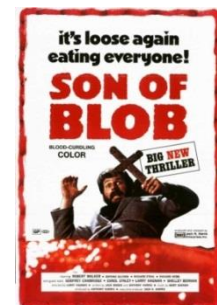
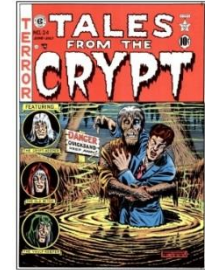
```
create procedure ComputeLateSubmissions (@delay)
as
  select id, count(*) from students
  where submissiondate = deadline + @delay
  group by id
```

- Cannot pre-compute all possible input values
  - why store the table in the cloud!

# TakeAway Quiz

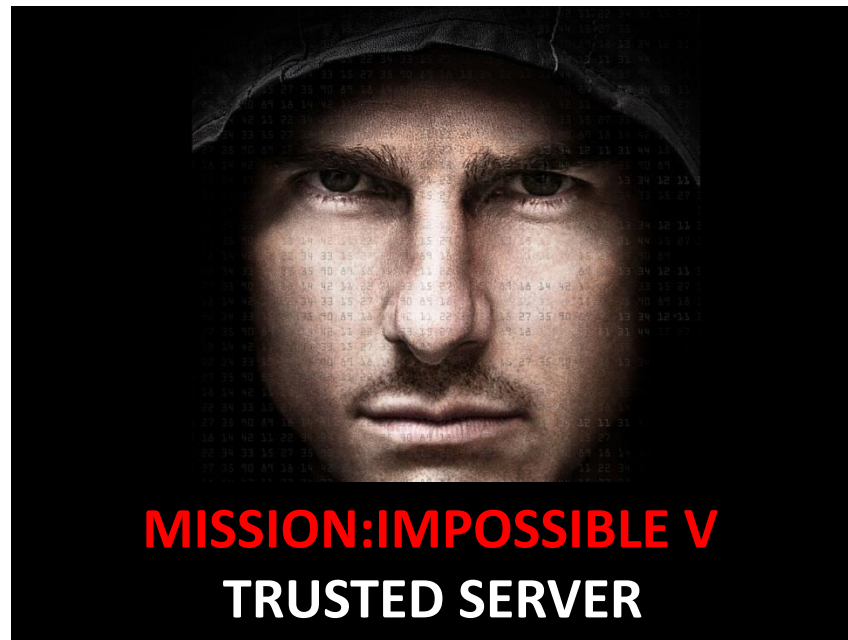
The Trusted Client approach:

- a) ~~Is “Dead” on arrival~~
- b) ~~Adding BLOB() keyword to SQL~~
- c) Is effective when most work can be offloaded to the server
- d) Is not a robust solution for general purpose queries



# Still to come ...

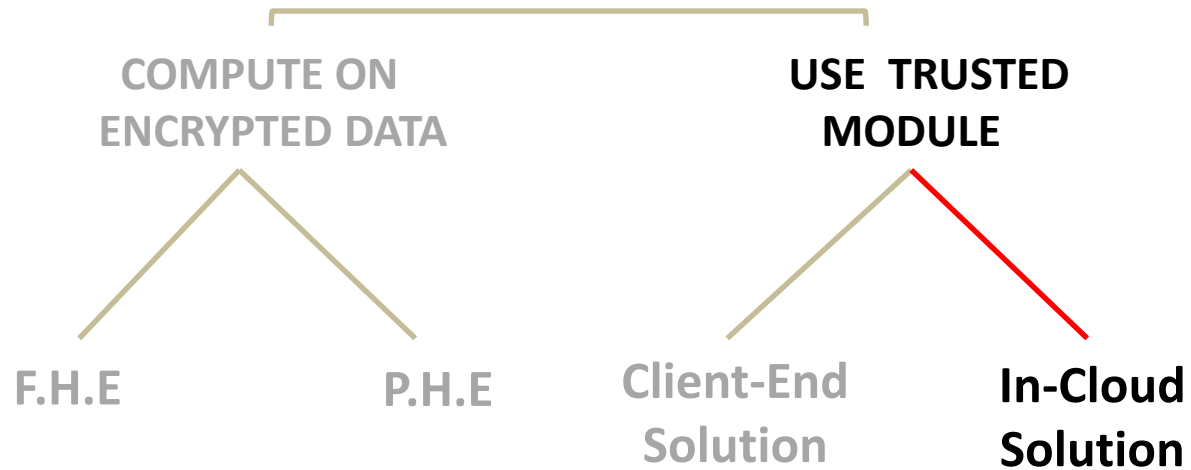
Is it possible to design a system where only the results are shipped to the client irrespective of query complexity ?



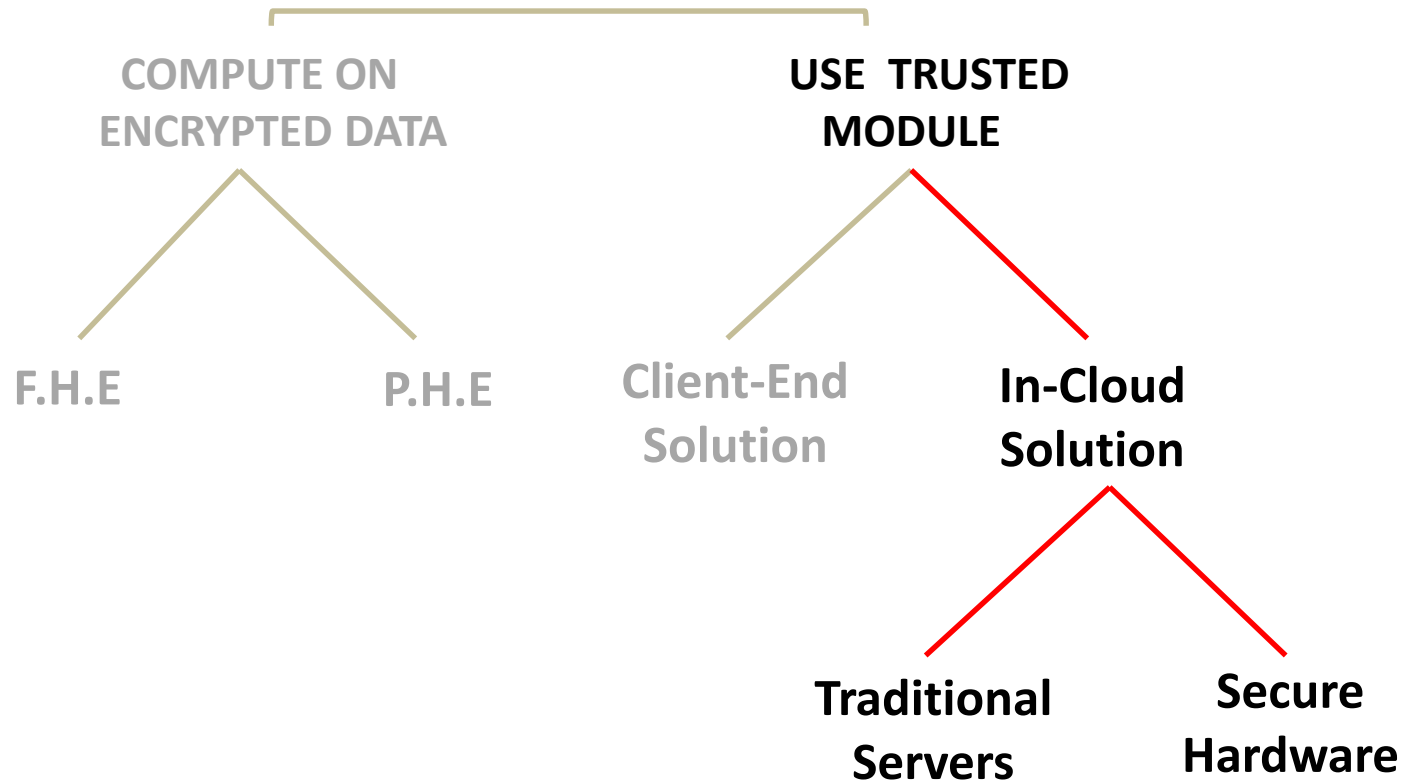
# Roadmap

- Introduction
- Overview
- Basics of Encryption
- Trusted Client based Systems
- **Secure In-Cloud Processing**
- **Security**
- **Conclusion**

# Secure In-Cloud Processing

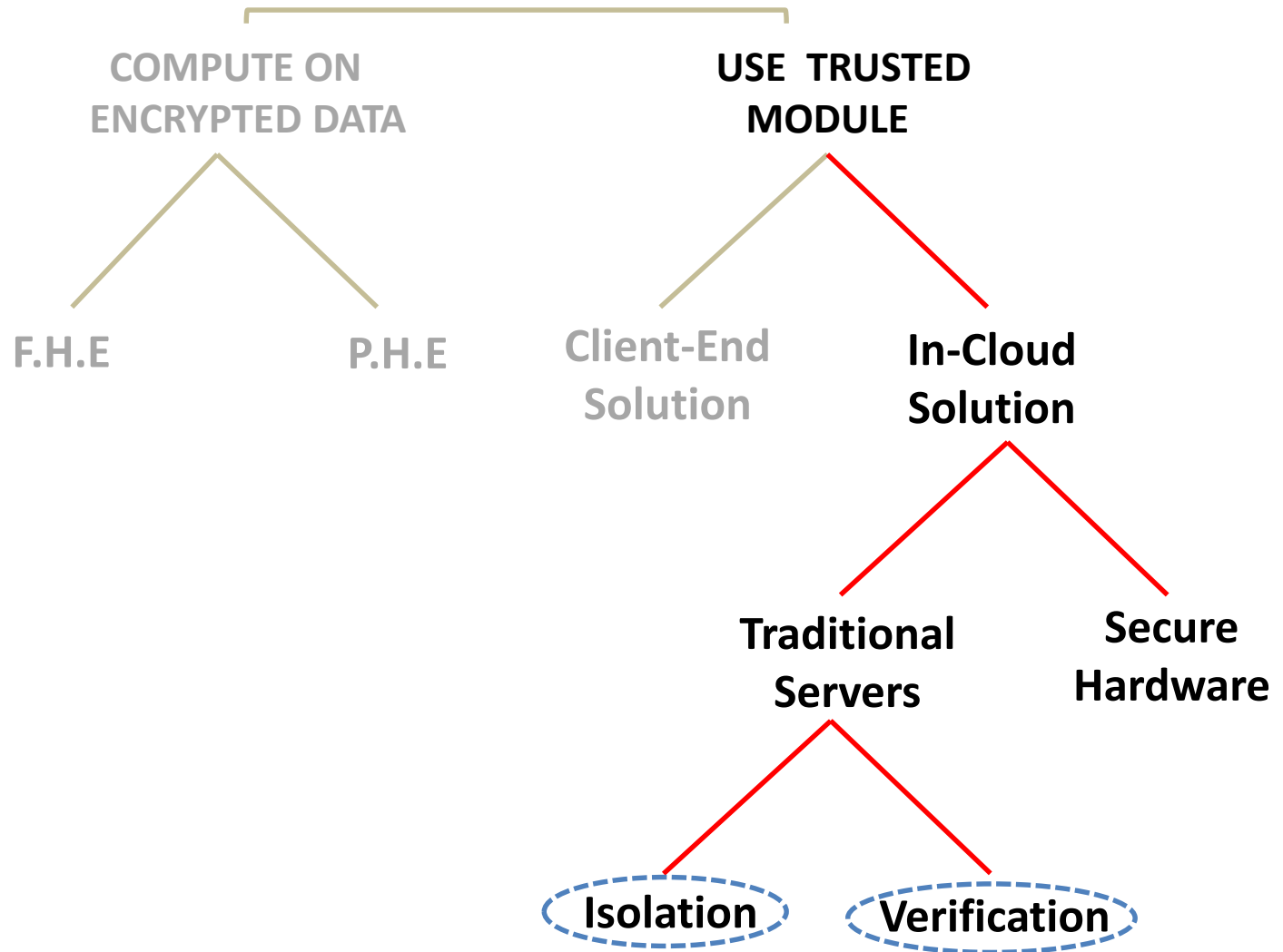


# Secure In-Cloud Processing





# Secure In-Cloud Processing



# Securing Traditional Servers

- Isolation
  - Physical (location, network)
  - Logical (hypervisor/VM, strict policies)
- Verification

Example: Amazon GovCloud [AWSGC]

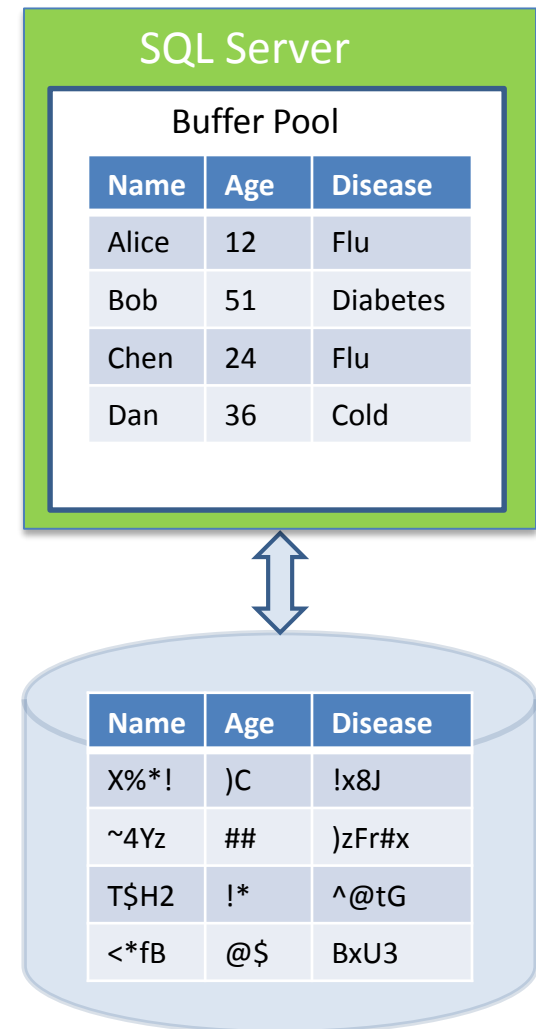
# Securing Traditional Servers

- Isolation
- Verification
  - Static (TPM authenticated boot)
  - Dynamic (malware detectors)

[TCGNotes, TPMspec]

# Secure Server Advantages

- Lowest impact solution
  - Use existing components, software and policies
  - We can all go home!



# Securing a Server is HARD!

TECHNOLOGY | June 2, 2011

## Google Mail Hack Blamed

Article Video Graphics

See how to reduce risk associated with F11 use with IBM's risk adv

## Universal Music Sites Hacked

E-mail addresses, user names and password hashes were

### More than 6 million LinkedIn passwords likely stolen



A screenshot of a LinkedIn profile page. The profile picture shows a woman with dark hair. The text on the page includes "Be great at what you do." and "Our vision - It's here".

## Sony Hack October 2011: Thousands Of PlayStation Network Accounts Targeted By Massive Attack



A photograph of a person in a white shirt standing in a server room. The room is filled with server racks and equipment.

## Hack

### Data Center

## US State Department hacked

Comments 0 Tweet 0 Like 0 +1 0 more

...taken is bre... identity.

# What Makes it Hard to Secure a Server?

- Built for flexibility & adaptability
  - General-purpose processors
    - A unified-memory space is a serious vulnerability
    - Security guarantees require 100% bug-free software

# What Makes it Hard to Secure a Server?

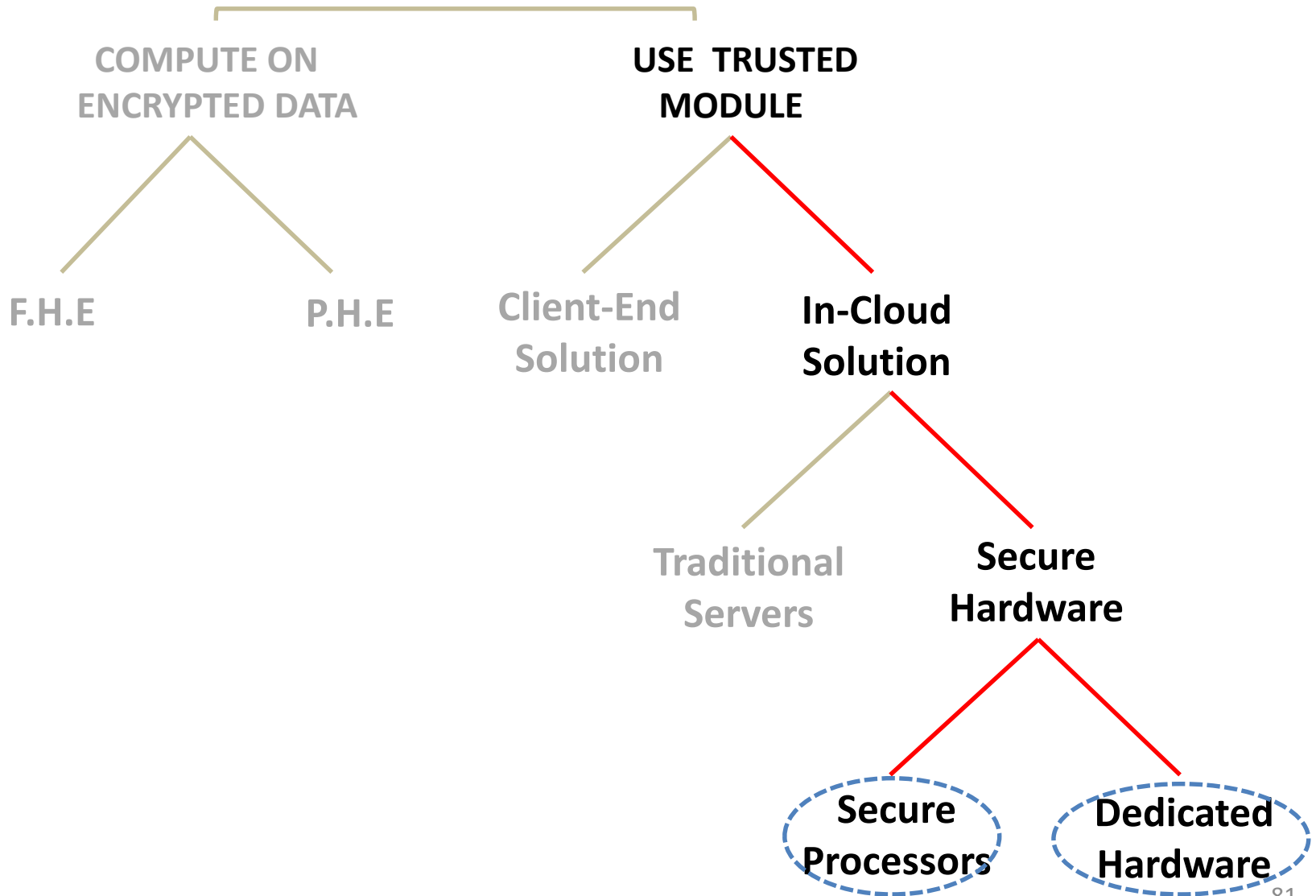
- Complex software stacks make rigorous security guarantees impossible
  - Largest formally verified OS kernel in literature is 8,700 lines of C and 600 lines of assembly
  - i.e. ‘Why don’t they make the whole plane out of that “black box” stuff?’

# What Makes it Hard to Secure a Server?

- We need to simplify the trusted computing platform!
- Better architectural isolation will also help!



# Secure In-Cloud Processing



# Previous Use of Secure Hardware

- Secure Co-Processor
  - ATM, smart cards
- Hardware Security Modules
  - Tamper-proof crypto acceleration
- FPGAs
  - Military use
- Limited Resources!

[TCGNotes]

**IBM 4764 PCI-X  
Cryptographic  
Coprorocessor**



**Secure FPGA**

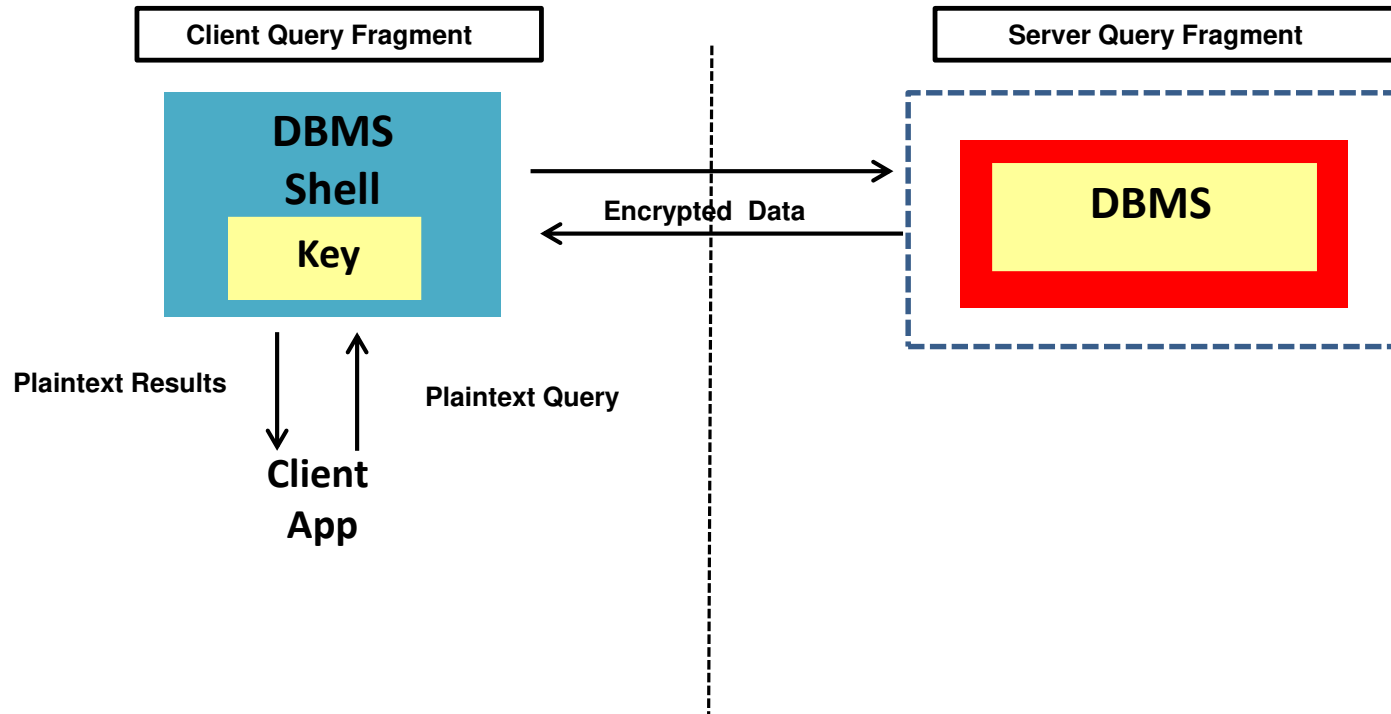
THE XILINX ADVANTAGE



- 3rd Generation Information Assurance
- 3rd Generation Anti-Tamper
- Government Agency Tested and Approved

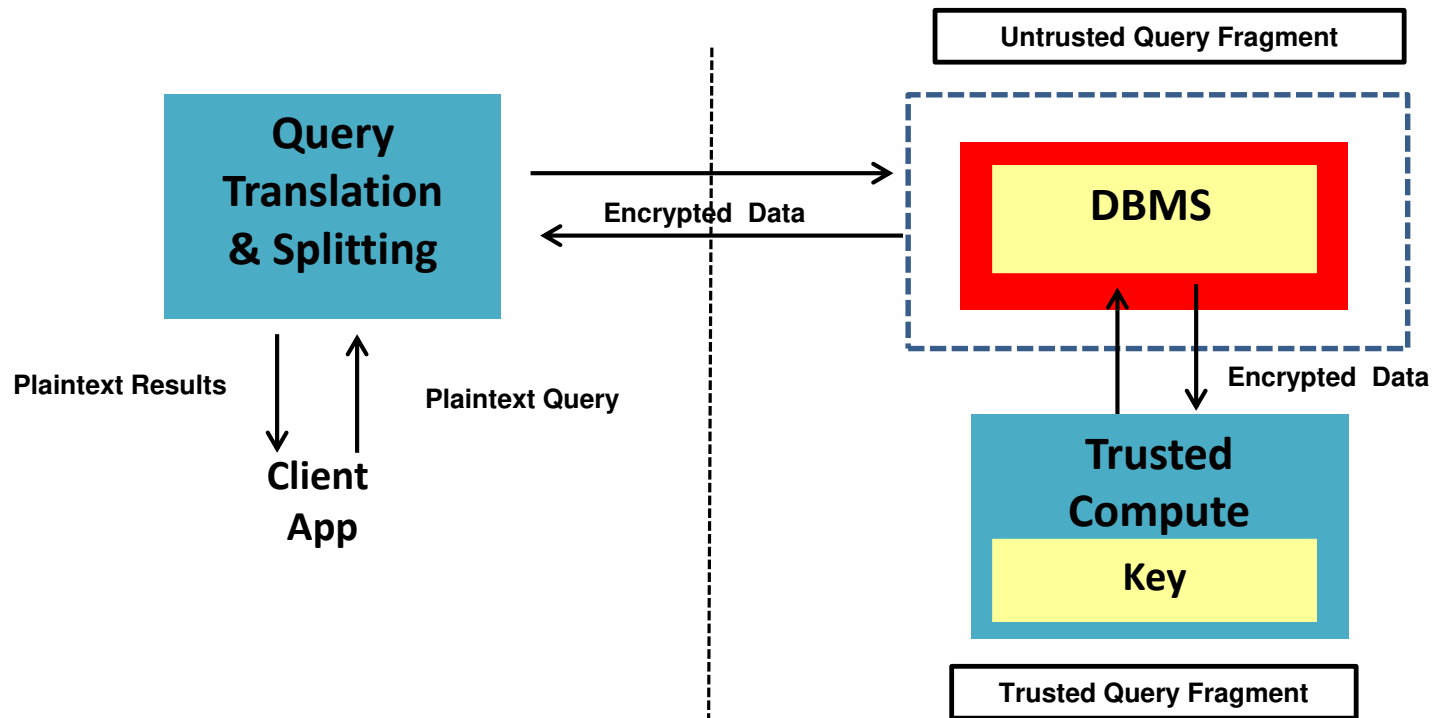


# Trusted Client Architecture



- Distributed query processing between untrusted DBMS and client-end DBMS shell

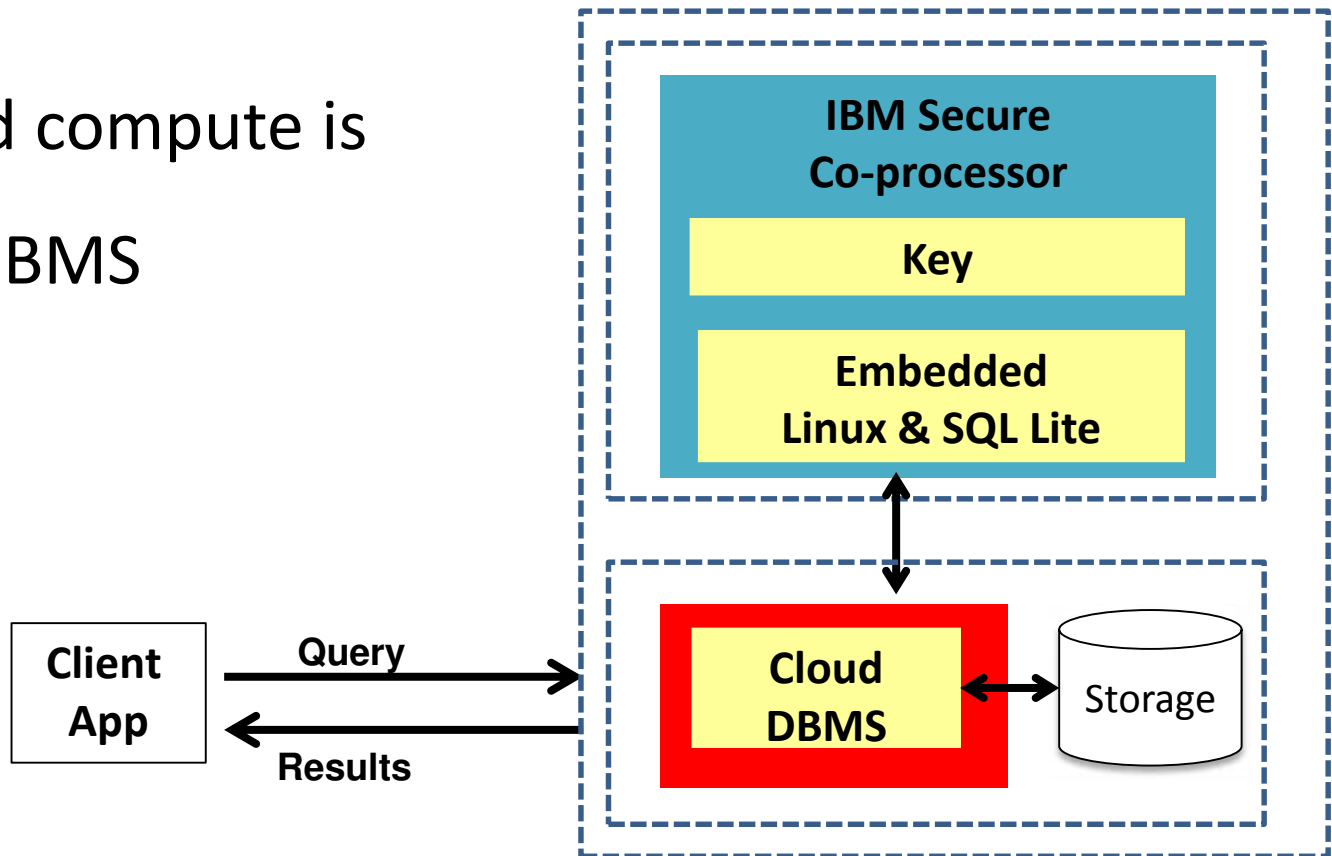
# Secure In-Cloud Compute Architecture



- Distributed query processing between untrusted DBMS and trusted cloud compute
- Solutions differ in granularity of integration

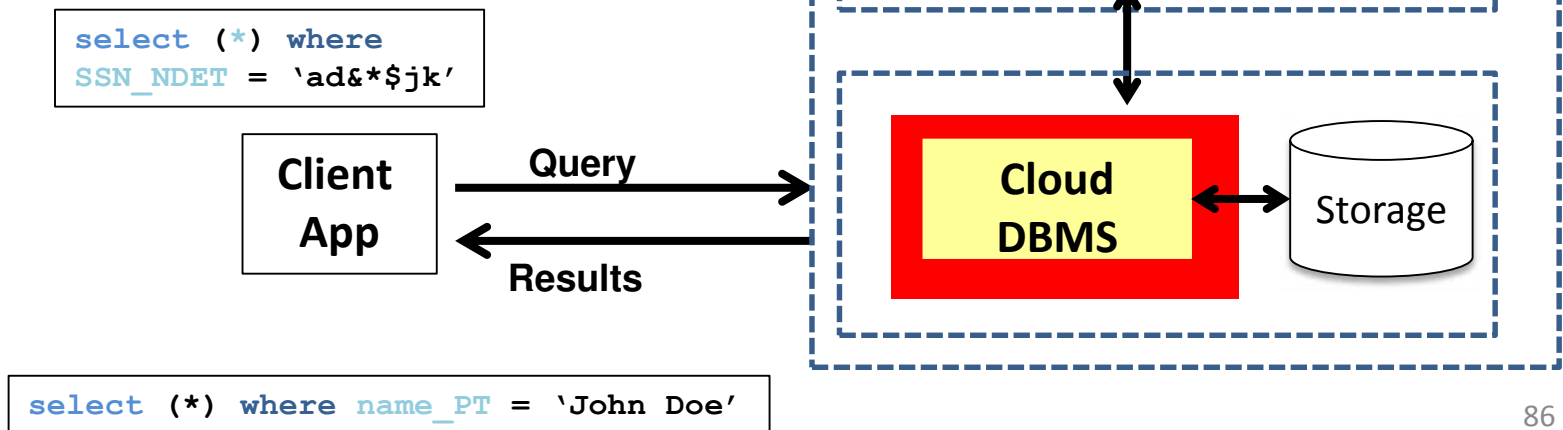
# Secure Processors

- TrustedDB
  - Trusted compute is a full DBMS



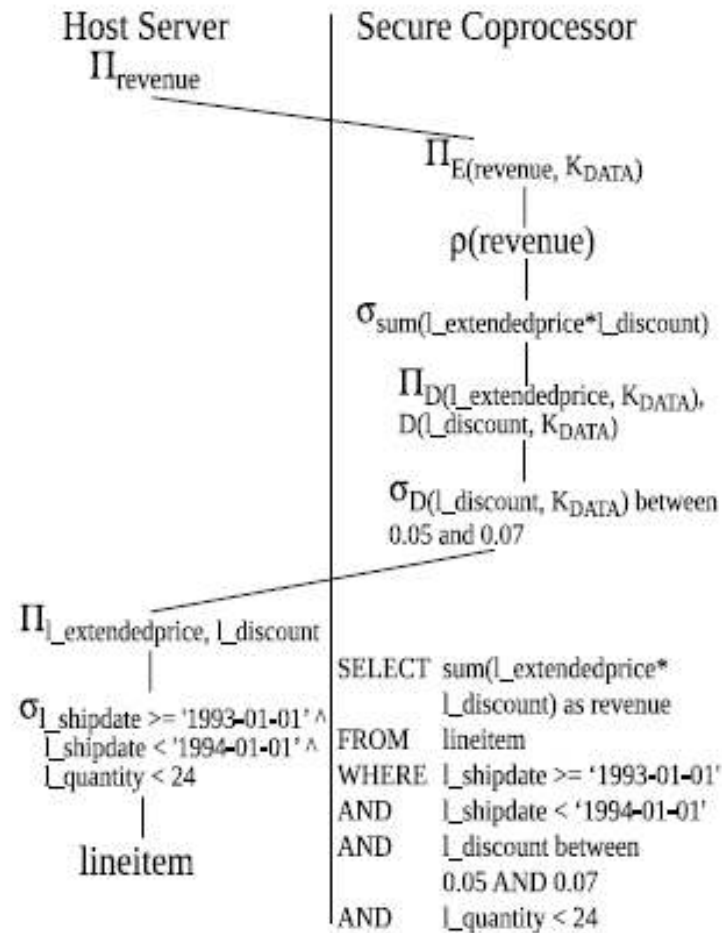
# Advantages of Loosely-Coupled Arch.

- Full trusted DBMS → loosely coupled system
  - Simple division of labor
  - Simple to build



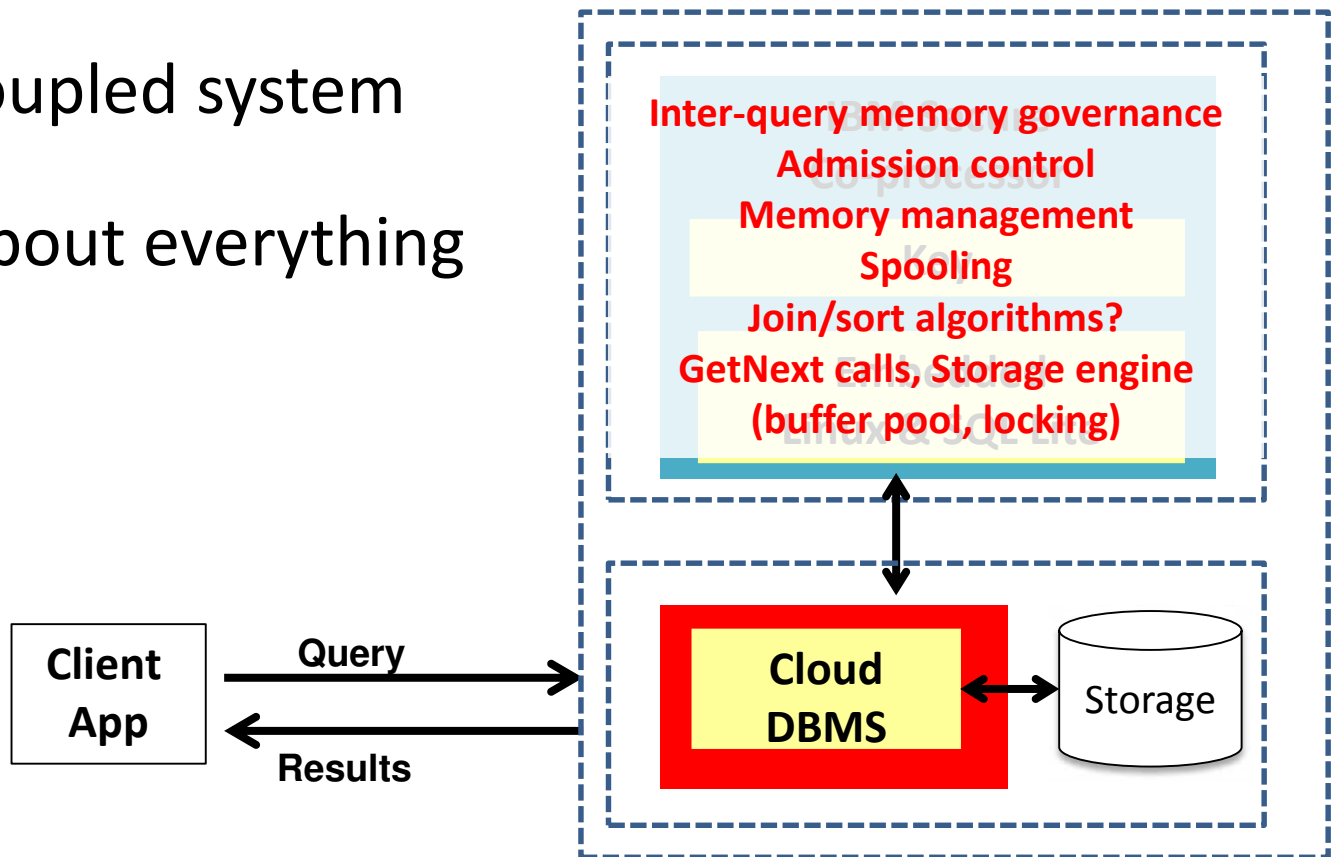
# TrustedDB Hybrid Example

LINEITEM	
ORDERKEY	
PARTKEY	
SUPPKEY	
LINENUMBER	
QUANTITY	
EXTENDEDPRICE	
DISCOUNT	
TAX	
RETURNFLAG	
LINESTATUS	
SHIPDATE	
COMMITDATE	
RECEIPTDATE	
SHIPINSTRUCT	
SHIPMODE	
COMMENT	



# Limitations of Loosely-Coupled Arch.

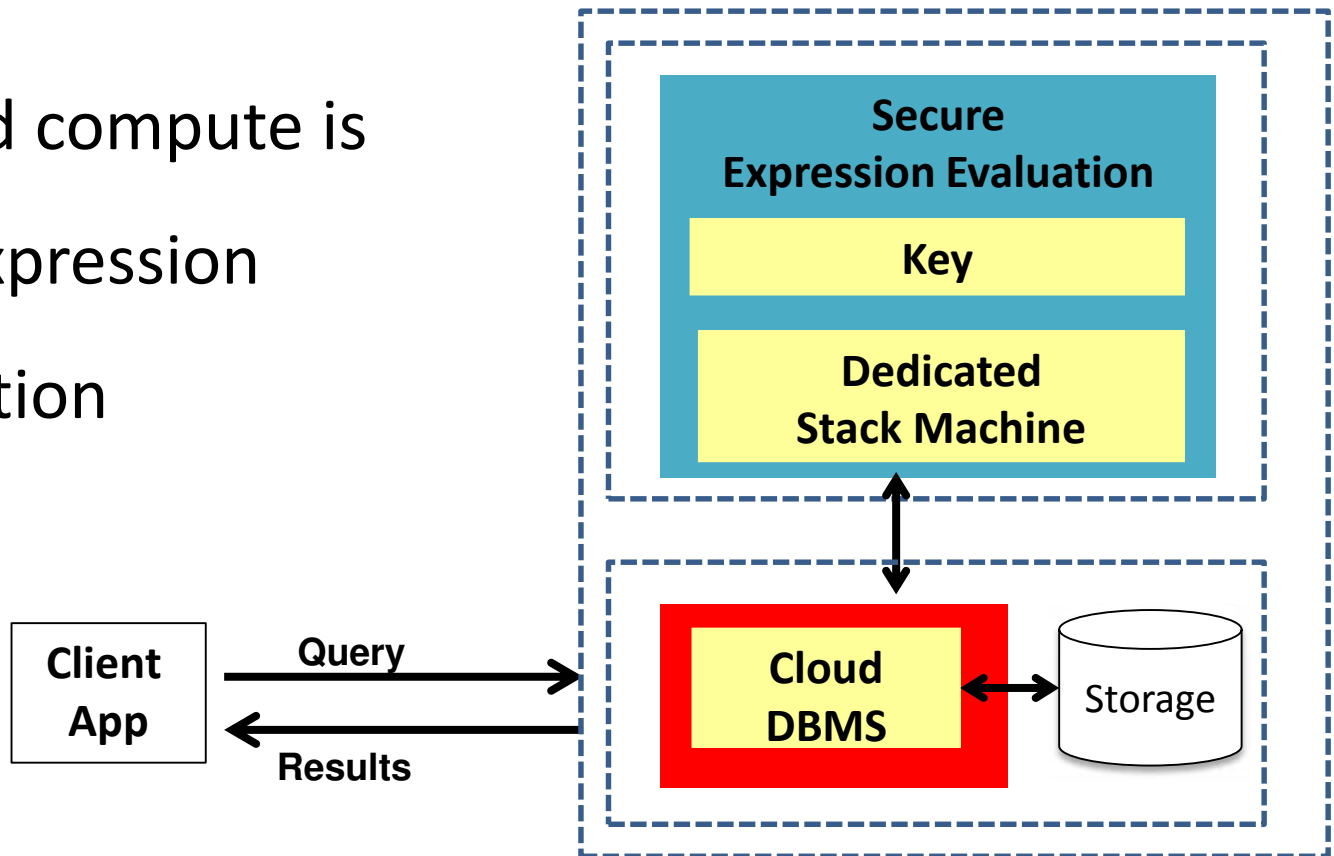
- Full trusted DBMS → loosely coupled system
  - How about everything else?



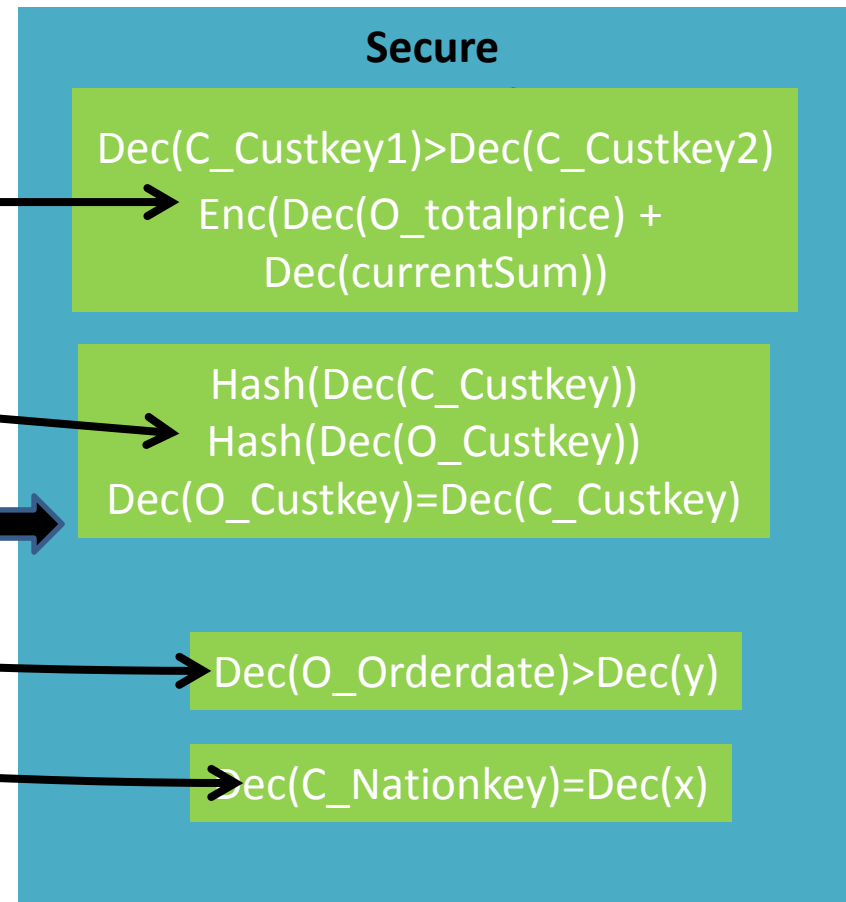
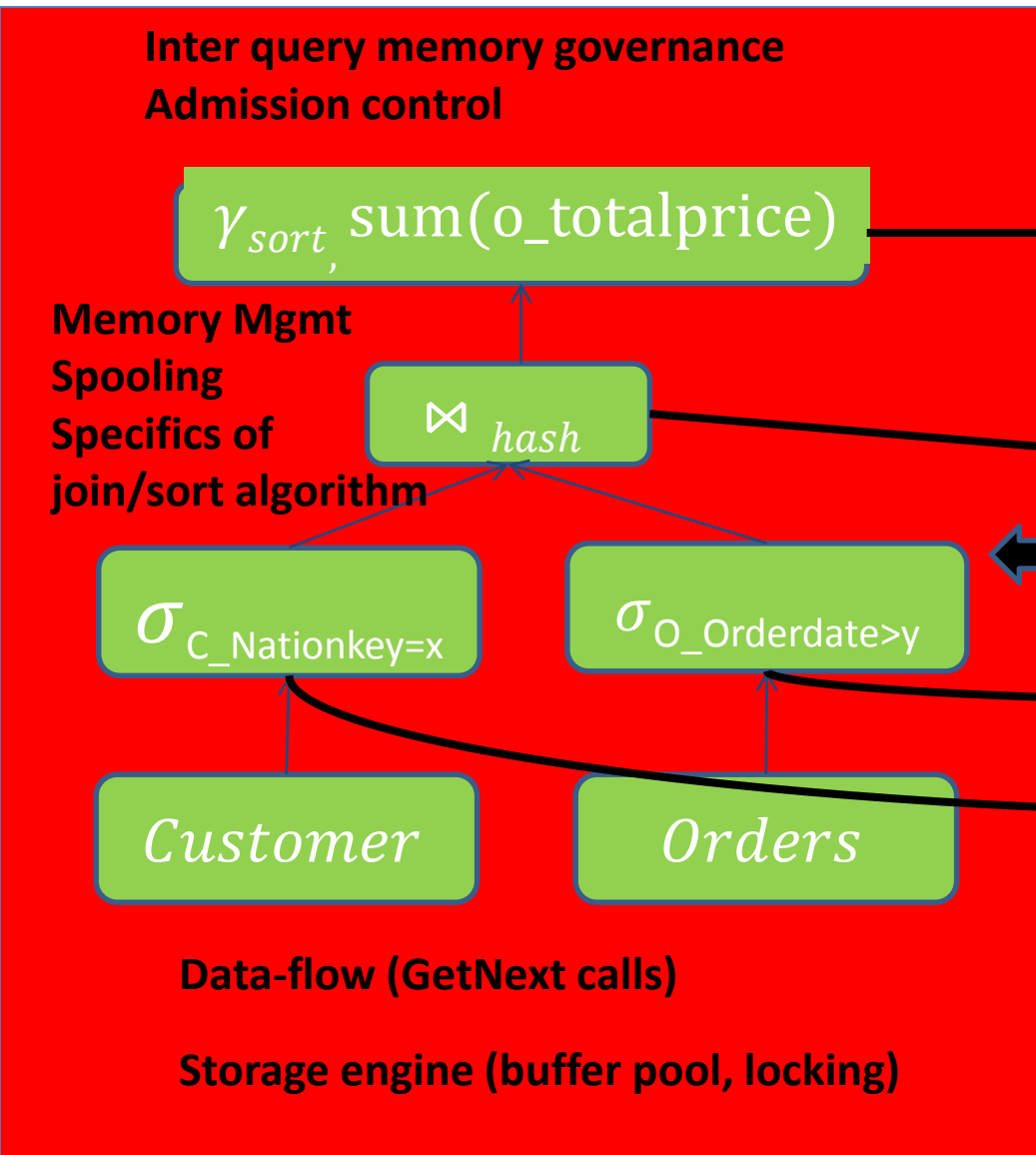


# Dedicated Expression Evaluation

- Cipherbase
  - Trusted compute is only expression evaluation



# Dedicated Expression Evaluation



# Dedicated Expression Evaluation

- Advantages
  - Efficiency of trusted compute resources
  - Dedicated circuits → virus-proof
  - Small footprint → formal verification
- Drawbacks
  - Fundamentally changes expression evaluation → non-trivial changes to host DBMS

# Summary

- Secure in-cloud trusted compute resources
- Open issues
  - Query optimization
    - e.g. Statistics on encrypted data, security-aware type matching
  - Execution engine
    - e.g. Data/computation reuse, masking latency to trusted computation
  - Physical Design
    - e.g. Leveraging stronger encryption



# Roadmap

- Introduction
- Overview
- Basics of Encryption
- Trusted Client based Systems
- Secure In-Cloud Processing
- **Security**
- **Conclusion**

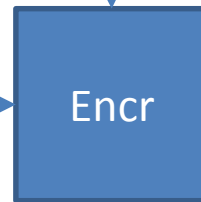


**SHOW  
ME THE  
SECURITY!**

# Encryption and Security

Key: 000102030405060708090a0b0c0d0e0f

The quick brown fox jumps  
over the lazy dog



a7be1a6997ad739bd8c9ca451f618b61  
b6ff744ed2c2c9bf6c590cbf0469bf41  
47f7f7bc95353e03f96c32bcfd8058df

# Encryption and Security

Key:





# Encryption and Security

Key:



- **Semantic security:**
  - No information leakage except input length
- Winner of this year's Turing Award



[KL07]

# Encryption and Security

Key:



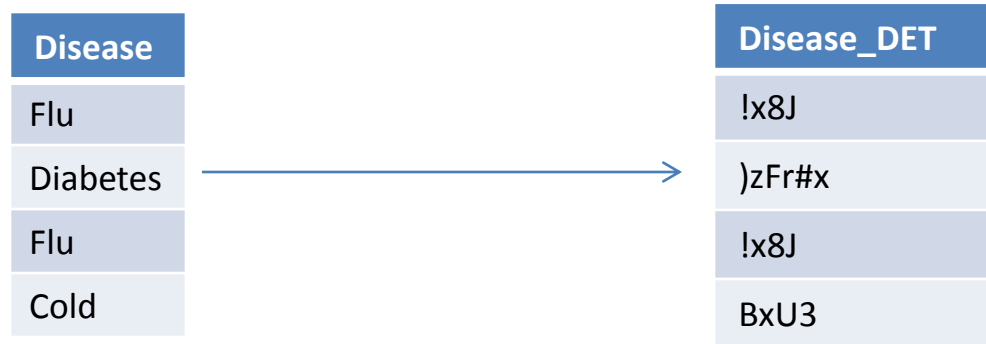
- Encryption schemes such as AES in CBC mode (non-deterministic) are believed to be semantically secure

# Security of Database Encryption



- Apply AES-CBC to every cell
- Leaks cell lengths

# Deterministic Encryption



- Leaks cell lengths
- Also, no. of distinct values + frequency distribution [BFO+08]

# Order-Preserving Encryption



- Leaks cell lengths
- Also, order of cell values [AKS+04, BCN11]



Design of order-preserving encryption

# Overall Security of Data Encryption

Name	Age	Disease
Alice	12	Flu
Bob	51	Diabetes
Chen	24	Flu
Dan	36	Cold



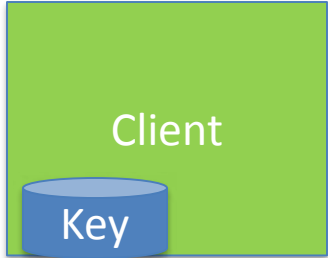
Name_NDET	Age	Disease_DET
X%*!	12	!x8J
~4Yz	51	)zFr#x
T\$H2	24	!x8J
<*fB	36	BxU3

- Name: AES-CBC non-deterministic
- Age: Clear-text
- Disease: AES deterministic
- Total information leaked = “sum” of column-level leakage

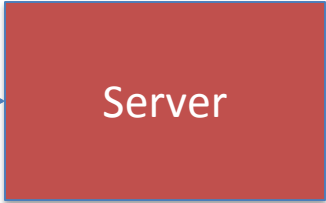
# Impact of Querying & Updating

## Trusted Client

## Untrusted Server



Update Employee  
Set Salary = \*&@#  
Where Name = 'Alice'

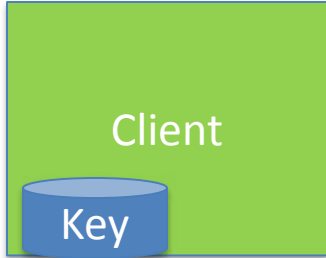


Name	Salary_NDET
Alice	X%*!
Bob	~4Yz
Chen	T\$H2
Dan	<*fB

# Impact of Querying & Updating

## Trusted Client

## Untrusted Server



Update Employee  
Set Salary = \*!-#  
Where Name = 'Alice'



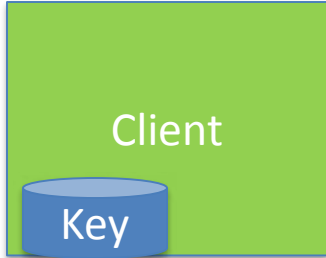
Name	Salary_NDET
Alice	X%*!
Bob	~4Yz
Chen	T\$H2
Dan	<*fB



# Impact of Querying & Updating

## Trusted Client

## Untrusted Server



Update Employee  
Set Salary = 23=\$<  
Where Name = 'Bob'

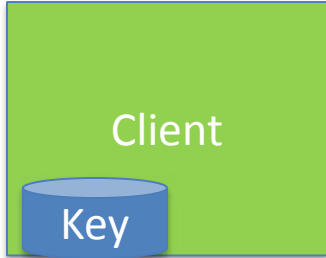


Name	Salary_NDET
Alice	X%*!
Bob	~4Yz
Chen	T\$H2
Dan	<*fB

# Impact of Querying & Updating

## Trusted Client

## Untrusted Server



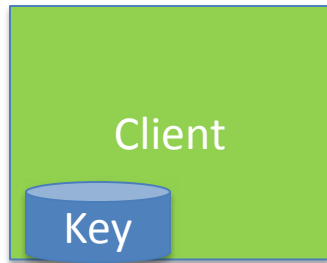
Update Employee  
Set Salary = +=\$<  
Where Name = 'Bob'



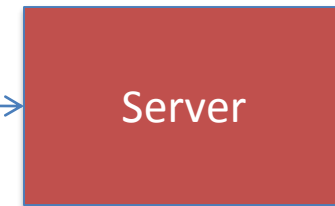
Name	Salary_NDET
Alice	X%*!
Bob	~4Yz
Chen	T\$H2
Dan	<*fB

# Impact of Querying & Updating

## Trusted Client



## Untrusted Server



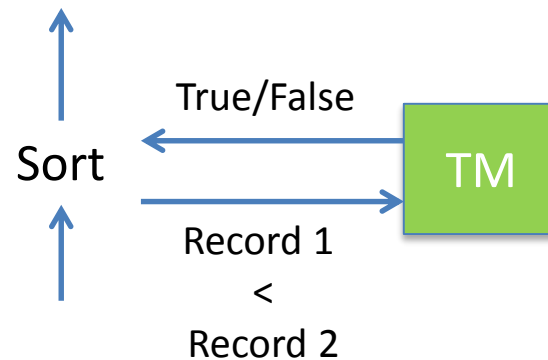
Update Employee  
Set Salary = #2\$^  
Where Name = 'Bob'

Name	Salary_NDET
Alice	X%*!
Bob	~4Yz
Chen	T\$H2
Dan	<*fB

- Background knowledge
  - Full-time employees earn more
  - Salaries of hourly-wage employees updated more
- Learn partial order of employee salary
  - Alice's salary > Bob's

**Query access patterns reveal information! [OS07]**

# Impact of Querying & Updating

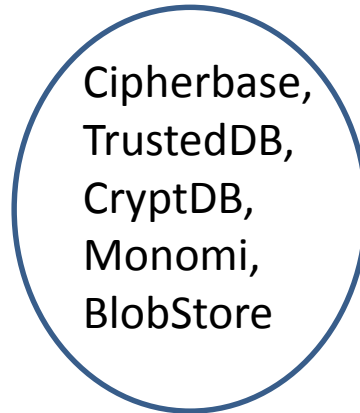


- Sort leaks ordering
- Encryption across the stack (disk + in-memory) does NOT imply no information leakage

The overall query workflow reveals information

**Dynamic security** (different from security of data at rest)

# Design Space



Stop with encryption

Can we bridge this gap?

Full Leakage

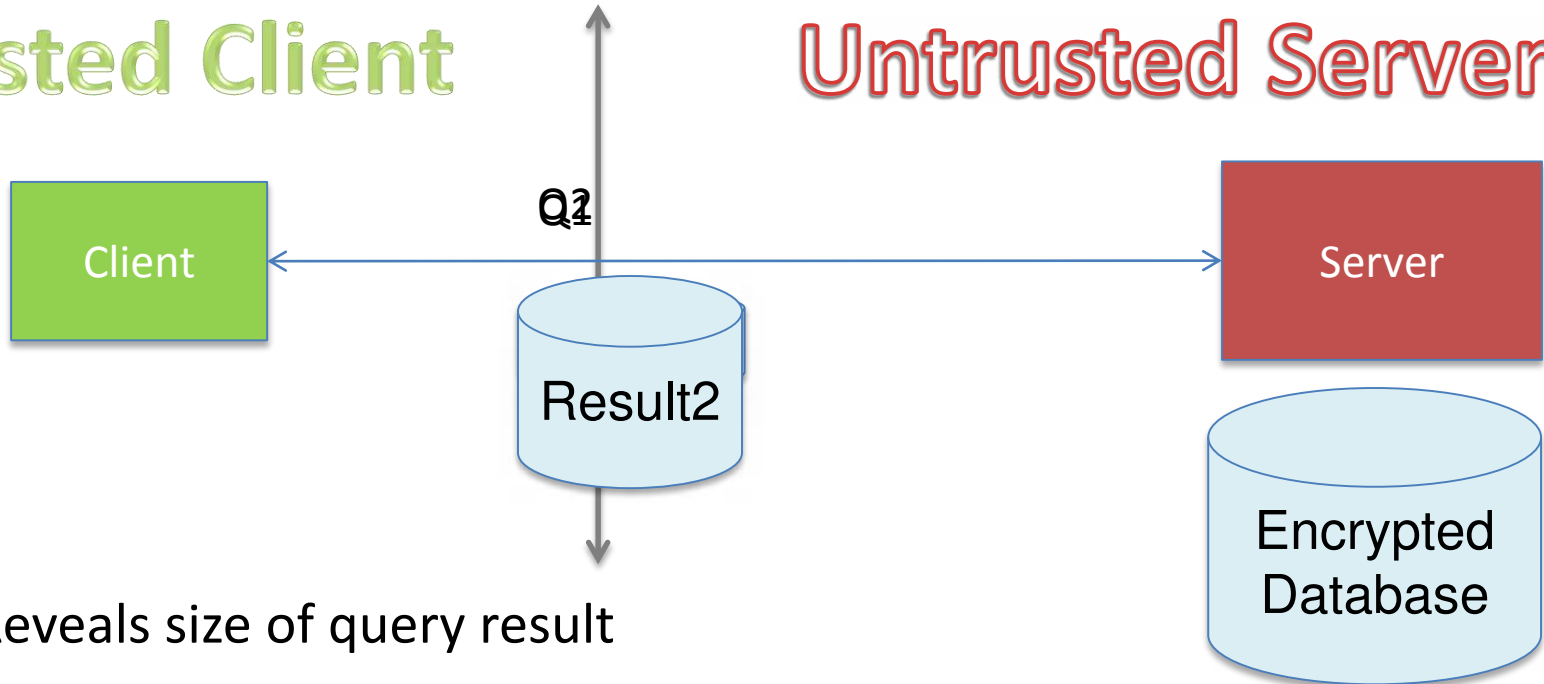
Operations on column	Leakage
Equality (including joins)	Frequency distribution
Indexing/Sorting /range predicates	Order

No Leakage

# No Leakage

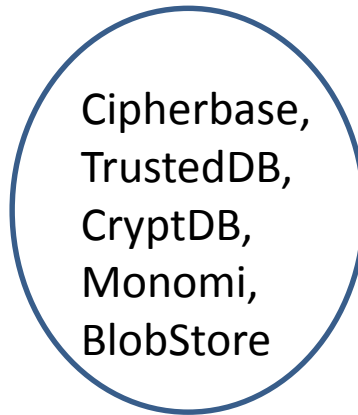
Trusted Client

Untrusted Server



- Reveals size of query result
- Hide query result size by making all query result sizes equal to maximum size
  - Joins reduce to cross products
  - Impractical

# Design Space



← Stop with encryption

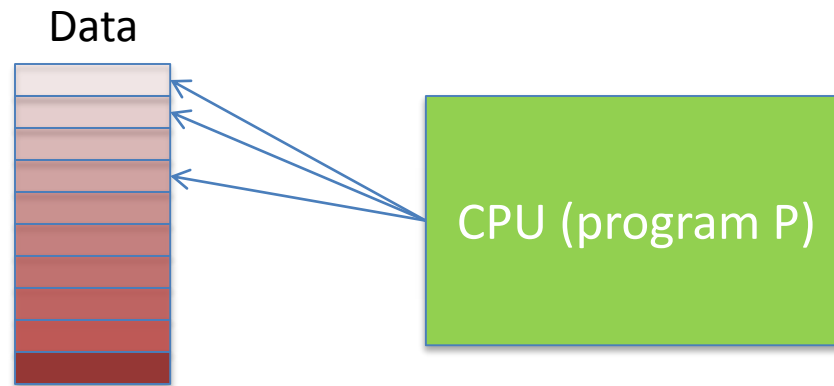
---

Full Leakage

Output Size,  
Running Time

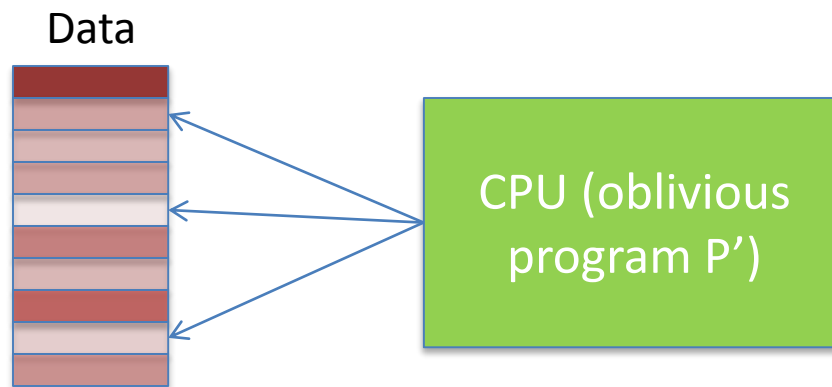
No Leakage  
**Impractical**

# Access Patterns Leak Information





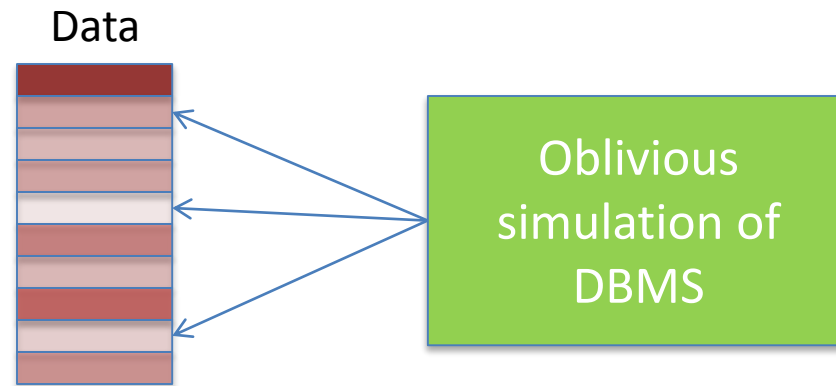
# Oblivious Simulation



- **Simulation:**  $P'$  equivalent to  $P$
- **Theoretically Efficient:** Running time of  $P'$  within polylog factor of running time of  $P$
- **Oblivious:** Access patterns of  $P'$  look random
- **Information leakage:** input size, output size, running time

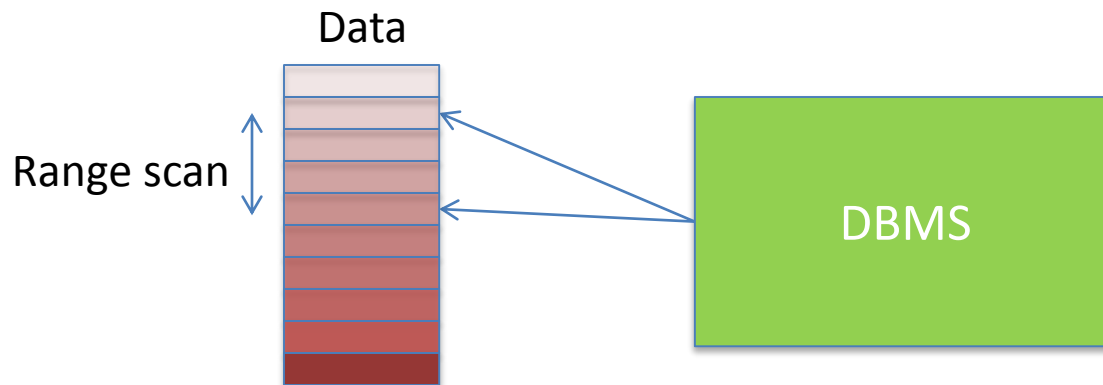
[GO96, W12, SS13]

# Application to DBMS



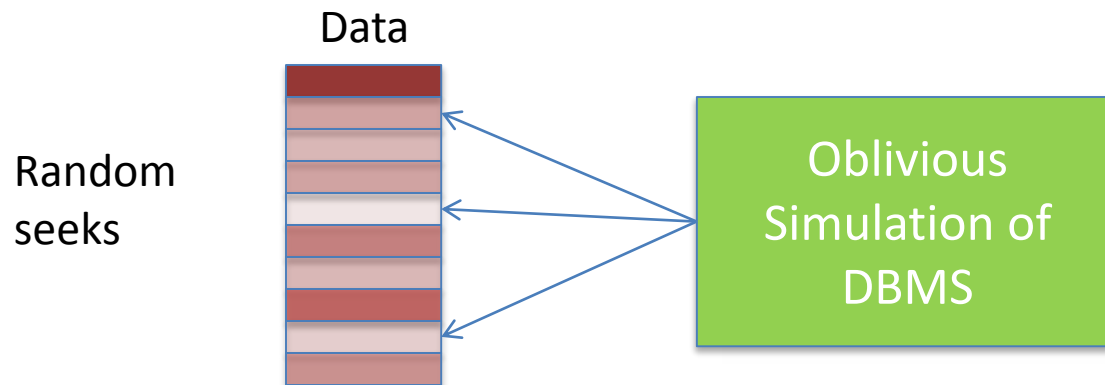
# But...

- Destroys spatial and temporal locality of reference



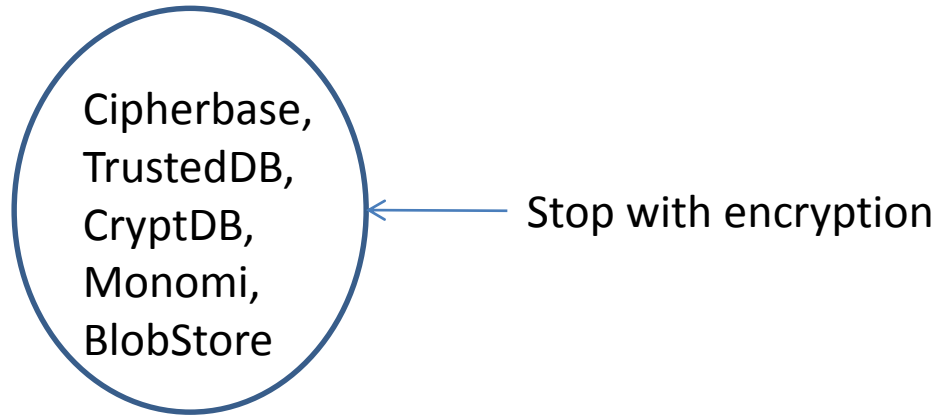
# But...

- Destroys spatial and temporal locality of reference



- Range scan of 100M records on hard disk → 100M seeks
  - $10^5$  seconds (~1 day)

# Design Space



---

Full Leakage

Output Size,  
Running Time  
**Impractical**

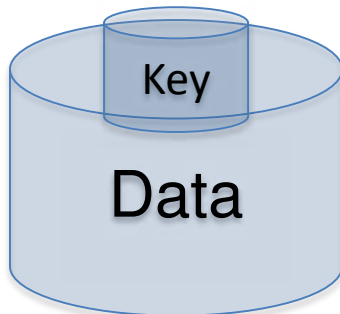
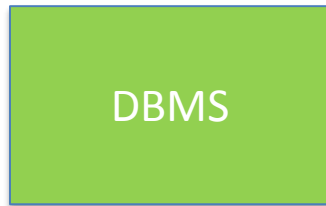
No Leakage  
**Impractical**



Is there a stronger and practically achievable security model?

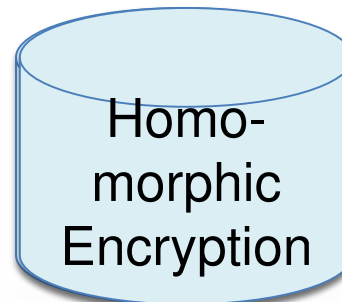
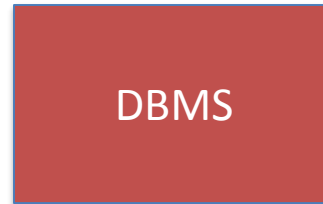
# Summary

## Trusted Client



Name	Age	Disease
Alice	12	Flu
Bob	51	Diabetes
Chen	24	Flu
Dan	36	Cold

## Untrusted Server



Name	Age	Disease
X%*!	)C	!x8J
~4Yz	##	)zFr#x
T\$H2	!*	^@tG
<*fB	@\$	BxU3



Cloud Admin

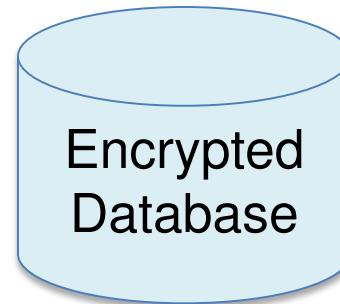
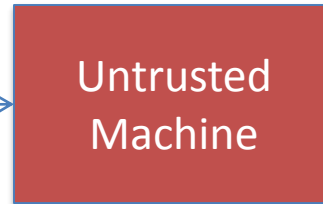
- Super-user with console access

# Summary

Trusted Client



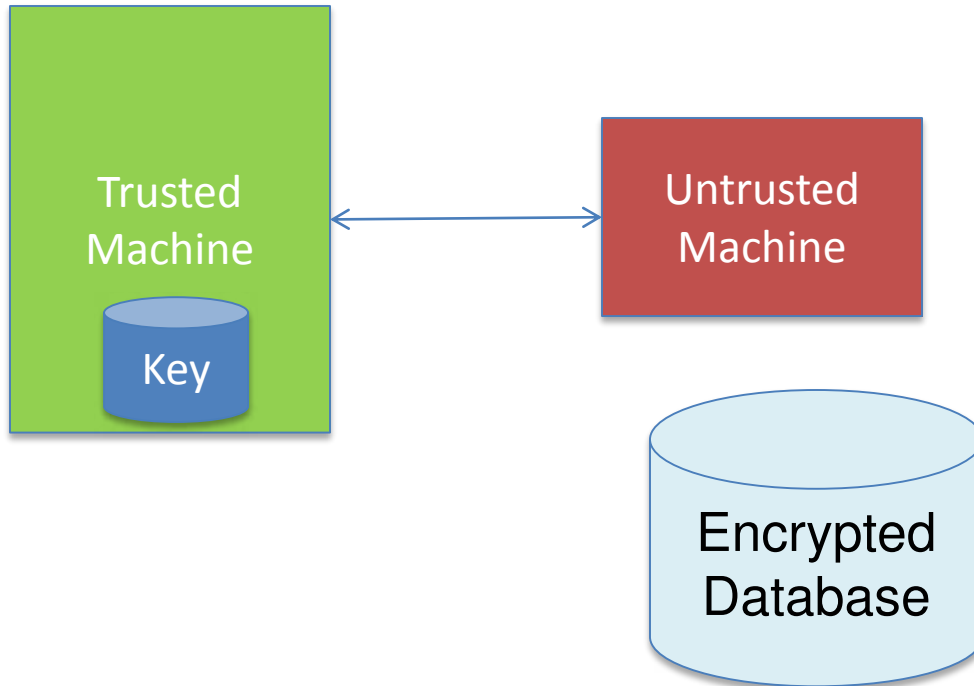
Untrusted Server



Name	Age	Disease
X%*!	)C	!x8J
~4Yz	##	)zFr#x
T\$H2	!*	^@tG
<*fB	@\$	BxU3

# Summary

## Untrusted Server



Name	Age	Disease
X%*!	)C	!x8J
~4Yz	##	)zFr#x
T\$H2	!*	^@tG
<*fB	@\$	BxU3



# Other Challenges

- Application Security
  - DBMS is only a part of the overall system stack
- Usability
  - Clients need tools and interpretable security models to navigate security-performance tradeoff
- Connections to other areas of security
  - Data privacy, access control, auditing

# Bibliography

1. [ABE+12] Arvind Arasu, Spyros Blanas, Ken Eguro, Manas Joglekar, Raghav Kaushik, Donald Kossmann, Ravishankar Ramamurthy, Prasang Upadhyaya, Ramarathnam Venkatesan: Engineering Security and Performance with Cipherbase. IEEE Data Eng. Bull. 35(4): 65-72 (2012).
2. [ABE+13] Orthogonal Security With Cipherbase. Arvind Arasu, Spyros Blanas, Ken Eguro, Raghav Kaushik, Donald Kossmann, Ravi Ramamurthy, and Ramaratnam Venkatesan. CIDR 2013.
3. [AES] AES Standard. FIPS 197. <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>
4. [AF+ 09] Above the Clouds: A Berkeley View of Cloud Computing. by Michael Armbrust, Armando Fox, and others. Tech Report EECS-2009-28, Univ. of Calif., Berkeley.
5. [AKS+04] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order-preserving encryption for numeric data. In SIGMOD 2004.
6. [AWSGC] Amazon GovCloud. <http://aws.amazon.com/govcloud-us/>.
7. [B68] K.E. Batcher, *Sorting networks and their applications*, Proceedings of the AFIPS Spring Joint Computer Conference 32, 307–314 (1968).
8. [BCL09] Order-Preserving Symmetric Encryption. Alexandra Boldyreva, Nathan Chenette, Younho Lee, Adam O'Neill. EUROCRYPT 2009.

# Bibliography

9. [BCN11] Order-Preserving Encryption Revisited: Improved Security Analysis and Alternative Solutions. Alexandra Boldyreva, Nathan Chenette, Adam O'Neill. CRYPTO 2011.
10. [BFO+08] M. Bellare, M. Fischlin, A. O'Neill, T. Ristenpart: Deterministic Encryption: Definitional Equivalences and Constructions without Random Oracles. CRYPTO 2008.
11. [BG11] Luc Bouganim, Yanli Guo: Database Encryption. Encyclopedia of Cryptography and Security (2nd Ed.) 2011.
12. [BOA] Buffer Overflow Attack. Lecture Notes.  
[http://www.cse.scu.edu/~tschwarz/coen152\\_05/Lectures/BufferOverflow.html](http://www.cse.scu.edu/~tschwarz/coen152_05/Lectures/BufferOverflow.html)
13. [BP02] Luc Bouganim, Philippe Pucheral: Chip-Secured Data Access: Confidential Data on Untrusted Servers. VLDB 2002
14. [BS11] Sumeet Bajaj, Radu Sion: TrustedDB: a trusted hardware based database with privacy and data confidentiality. SIGMOD Conference 2011.
15. [CPK 10] What's New About Cloud Computing Security?. Yanpei Chen, Vern Paxson and Randy H. Katz. Tech Report EECS-2010-5. Univ. of Calif., Berkeley.
16. [E84] A public key cryptosystem and a signature scheme based on discrete logarithms. Taher El Gamal. CRYPTO 1984.

# Bibliography

17. [ENISA 09a] Cloud Computing Risk Assessment. European Network and Information Security Agency. 2009.
18. [ENISA 09b] An SME perspective on cloud computing (survey). European Network and Information Security Agency, 2009.
19. [G09] Fully homomorphic encryption using ideal lattices. Craig Gentry. STOC 2009.
20. [G10] Computing arbitrary functions of encrypted data. Craig Gentry. CACM 2010.
21. [G11] Michael T. Goodrich. Data-oblivious external-memory algorithms for the compaction, selection, and sorting of outsourced data. In SPAA, pages 379–388, 2011.
22. [GO96] O. Goldreich, R. Ostrovsky: Software Protection and Simulation on Oblivious RAMs. J. ACM 43(3): 431-473 (1996)
23. [GZ07] Tingjian Ge, Stanley B. Zdonik. Answering Aggregation Queries in a Secure System Model. VLDB 2007.
24. [GZ07b] Tingjian Ge, Stanley B. Zdonik: Fast, Secure Encryption for Indexing in a Column-Oriented DBMS. ICDE 2007.

# Bibliography

25. [HIL+02] Executing SQL over Encrypted Data in the Database-Service-Provider Model. Hakan Hacigumus, Balakrishna R. Iyer, Chen Li, Sharad Mehrotra, SIGMOD 2002.
26. [HIM04] Hakan Hacigümüs, Balakrishna R. Iyer, Sharad Mehrotra: Efficient Execution of Aggregation Queries over Encrypted Relational Databases. DASFAA 2004.
27. [HIM05] Hakan Hacigümüs, Balakrishna R. Iyer, Sharad Mehrotra: Query Optimization in Encrypted Database Systems. DASFAA 2005.
28. [HIM05b] Efficient Execution of Aggregation Queries over Encrypted Relational Databases. Hakan Hacigümüs, Balakrishna R. Iyer, Sharad Mehrotra. DASFAA 2005.
29. [HMH08] Bijit Hore, Sharad Mehrotra, Hakan Hacigümüs: Managing and Querying Encrypted Data. Handbook of Database Security 2008
30. [HMI02] Providing Database as a Service. Hakan Hacigumus, Sharad Mehrotra, Balakrishna R. Iyer. ICDE 2002.
31. [HMT04] Bijit Hore, Sharad Mehrotra, Gene Tsudik: A Privacy-Preserving Index for Range Queries. VLDB 2004.
32. [K09] G. Klein et al, “seL4: formal verification of an OS kernel” SOSP 2009.

# Bibliography

33. [KL07] Introduction to Modern Cryptography. Jonathan Katz and Yehuda Lindell. Chapman & Hall/CRC Press. 2007.
34. [NIST 09] P. Mell and T. Grance. NIST definition of cloud computing. National Institute of Standards and Technology. October 7, 2009.
35. [OTDE] Oracle Transparent Data Encryption. <http://www.oracle.com/technetwork/database/options/advanced-security/index-099011.html>
36. [P99] Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. Pascal Paillier. EUROCRYPT 1999.
37. [PLZ13] An Ideal-Security Protocol for Order-Preserving Encoding. Raluca Ada Popa, Frank H Li, Nikolai Zeldovich. Symp on Security and Privacy, 2013.
38. [PRZ+11] CryptDB: protecting confidentiality with encrypted query processing. Raluca A. Popa, Catherine M. S. Redfield, Nikolai Zeldovich, Hari Balakrishnan. SOSP 2011.
39. [S96] Applied Cryptography. Bruce Schneier. John Wiley & Sons, 1996.
40. [SS05] Trusted Computing Platforms: Design and Applications. Sean W Smith. Springer. 2005.

# Bibliography

41. [SS13] E. Stefanov, E. Shi. ObliviStore: High Performance Oblivious Cloud Storage. IEEE S&P. 2013.
42. [STDE] Sql Server Transparent Data Encryption.  
<http://technet.microsoft.com/en-us/library/bb934049.aspx>
43. [TCGNotes] Trusted Computing Architecture and its applications. CS255 Lecture Notes. Stanford University.  
<http://crypto.stanford.edu/cs155old/cs155-spring11/lectures/08-TCG.pdf>
44. [TFM13] Stephen Tu, M. Frans Kaashoek, Samuel Madden et al. Processing Analytical Queries over Encrypted Data. VLDB 2013.
45. [TPMSpec] TPM Main Specification. [http://www.trustedcomputinggroup.org/resources/tpm\\_main\\_specification](http://www.trustedcomputinggroup.org/resources/tpm_main_specification)
46. [VYK12] Vaibhav Khadilkar, Kerim Yasin Oktay, Murat Kantarcioglu, Sharad Mehrotra: Secure Data Processing over Hybrid Clouds. IEEE Data Eng. Bull. 35(4): 46-54 (2012).
47. [W12] P. Williams. Oblivious Remote Data Access Made Parallel. PhD Thesis. 2012.