

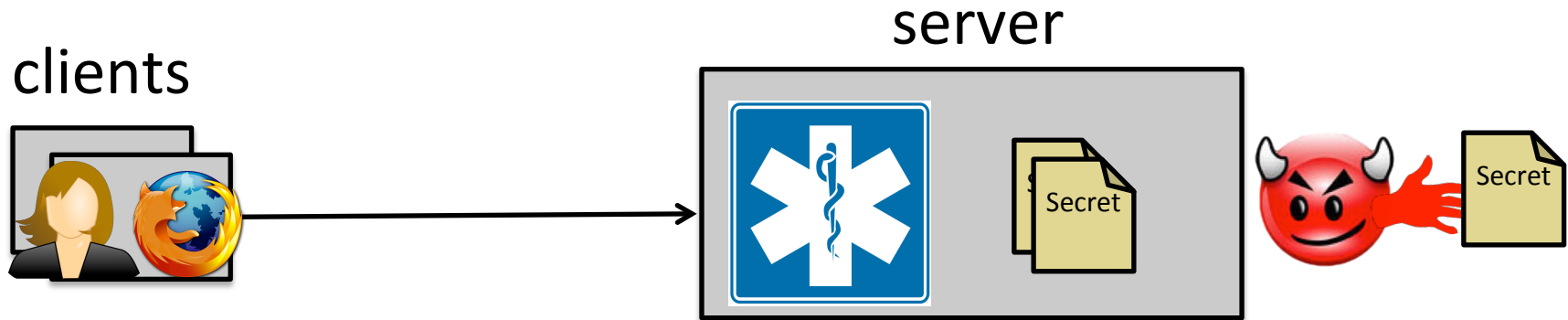
# Using cryptography in databases and web applications

Nickolai Zeldovich

MIT CSAIL

Joint work with Raluca Ada Popa, Stephen Tu,  
Emily Stark, Jonas Helfer, Steven Valdez,  
Hari Balakrishnan, Frans Kaashoek, Sam Madden

# Problem: private data breaches



no computation

storage



encryption

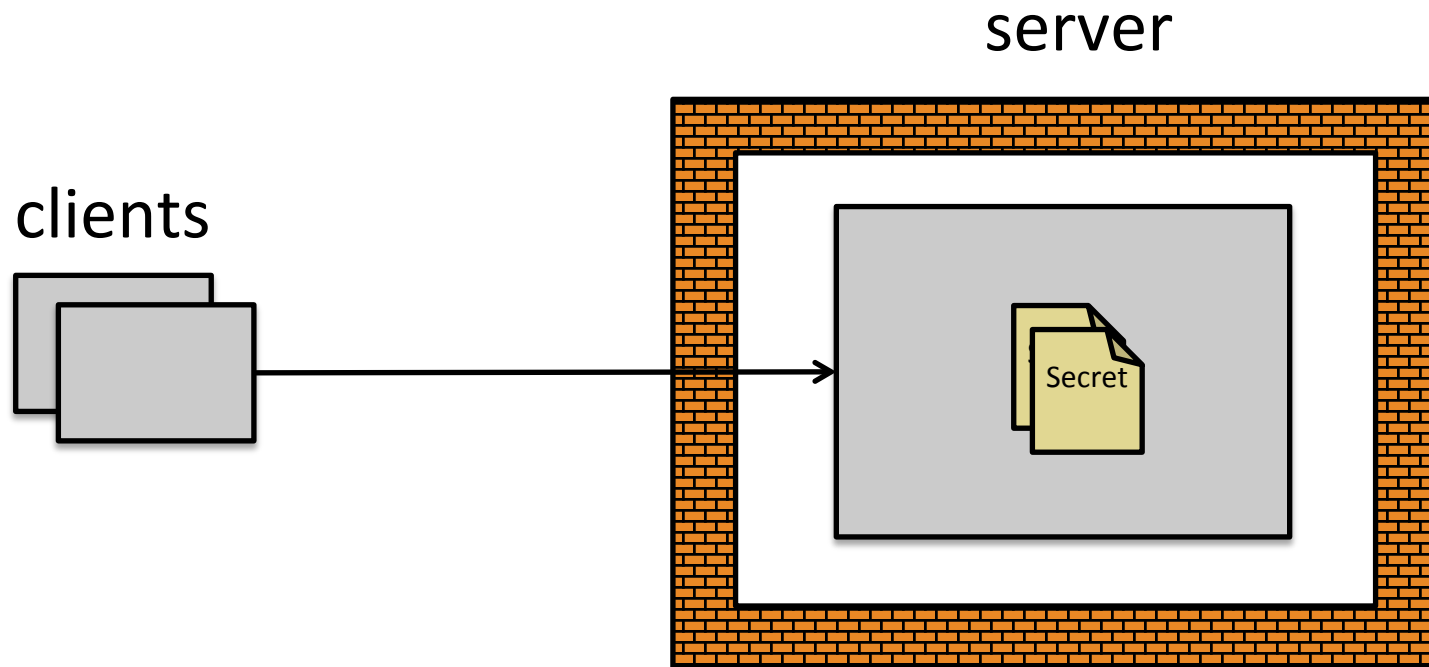
(encrypted files, email)

computation

databases, web applications, mobile applications, machine learning, etc.



# Common approach: prevent break-ins

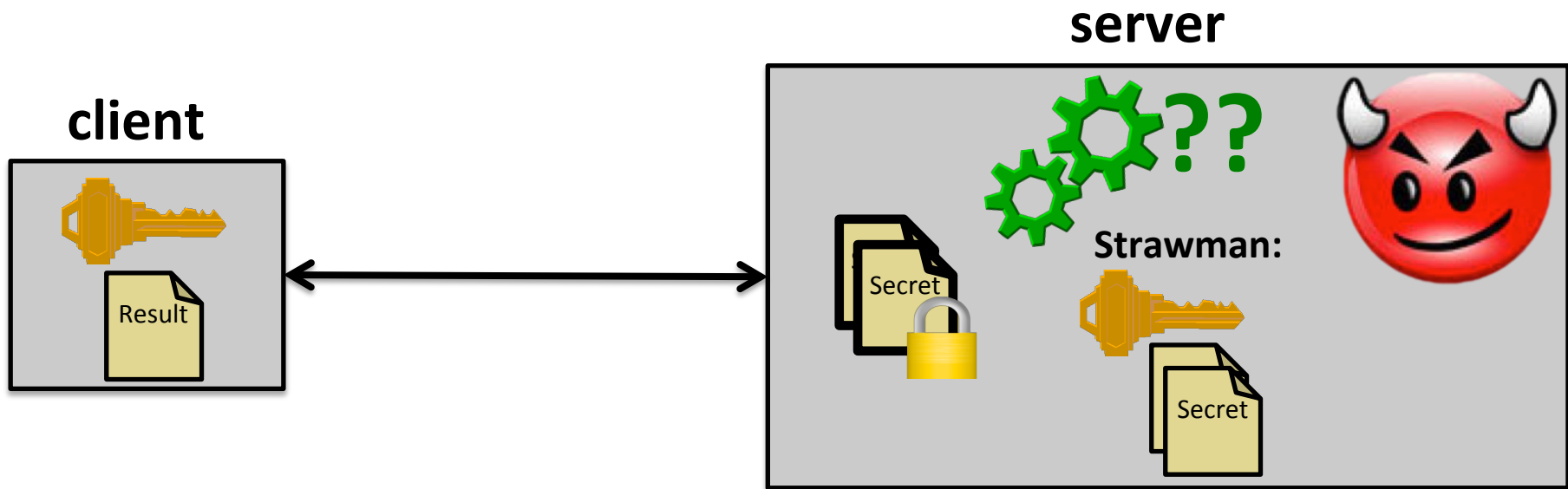


Enforced at many levels: operating system, hardware, network, programming language, ...

# Bad guys find ways to break in

- Complex software has bugs
  - Attackers find and exploit vulnerabilities
- Many people have access to infrastructure
  - Server administrators
  - Cloud / data center employees
  - Anyone that breaks into their accounts
- **Compromises are inevitable**

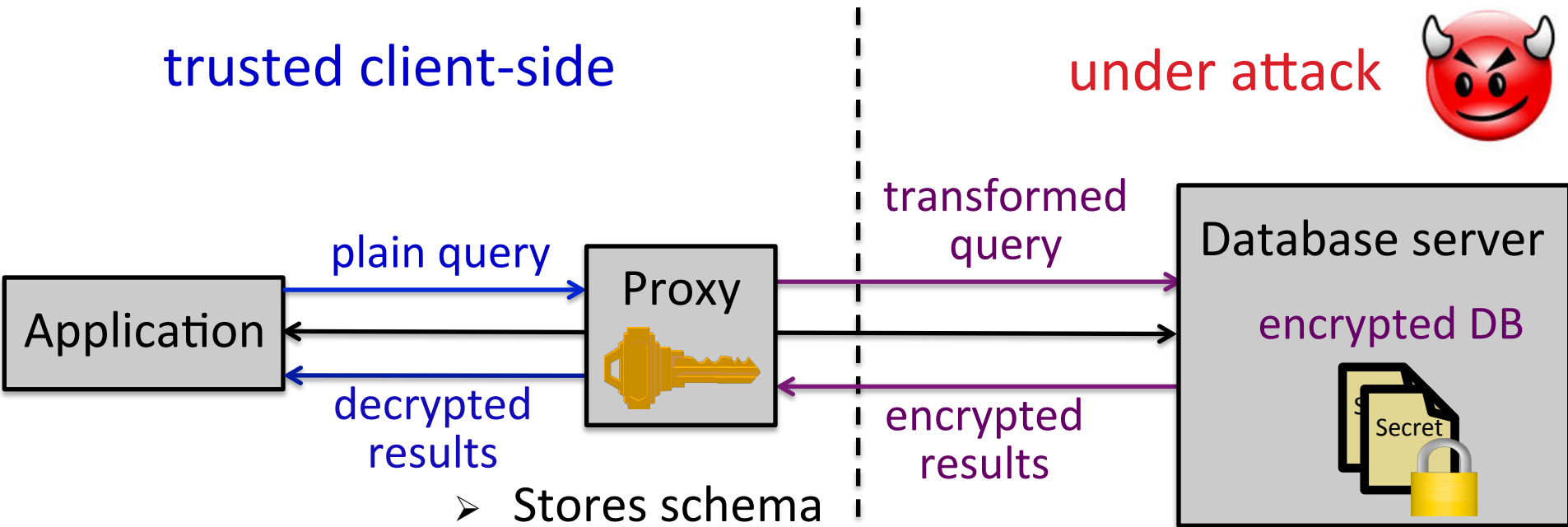
# New approach: practical processing of encrypted data



# CryptDB setup

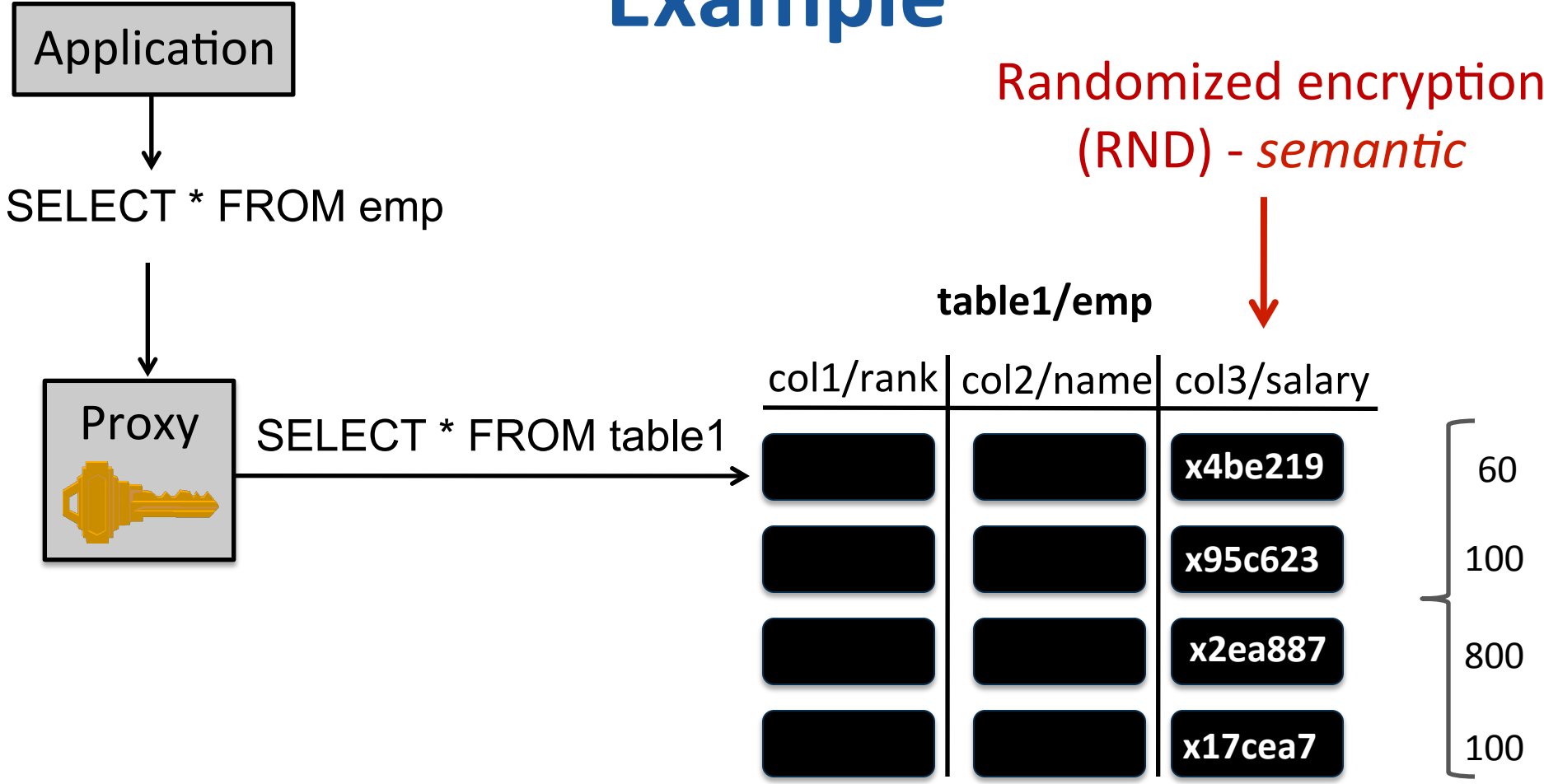
trusted client-side

under attack



- Stores schema and master key
- Minimal or no query execution

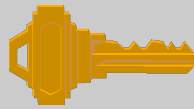
# Example



# Example

Application

SELECT \* FROM emp  
WHERE salary = 100

Proxy  


SELECT \* FROM table1  
WHERE col3 = x5a8c34

Randomization  
encryption (DET)

table1/emp

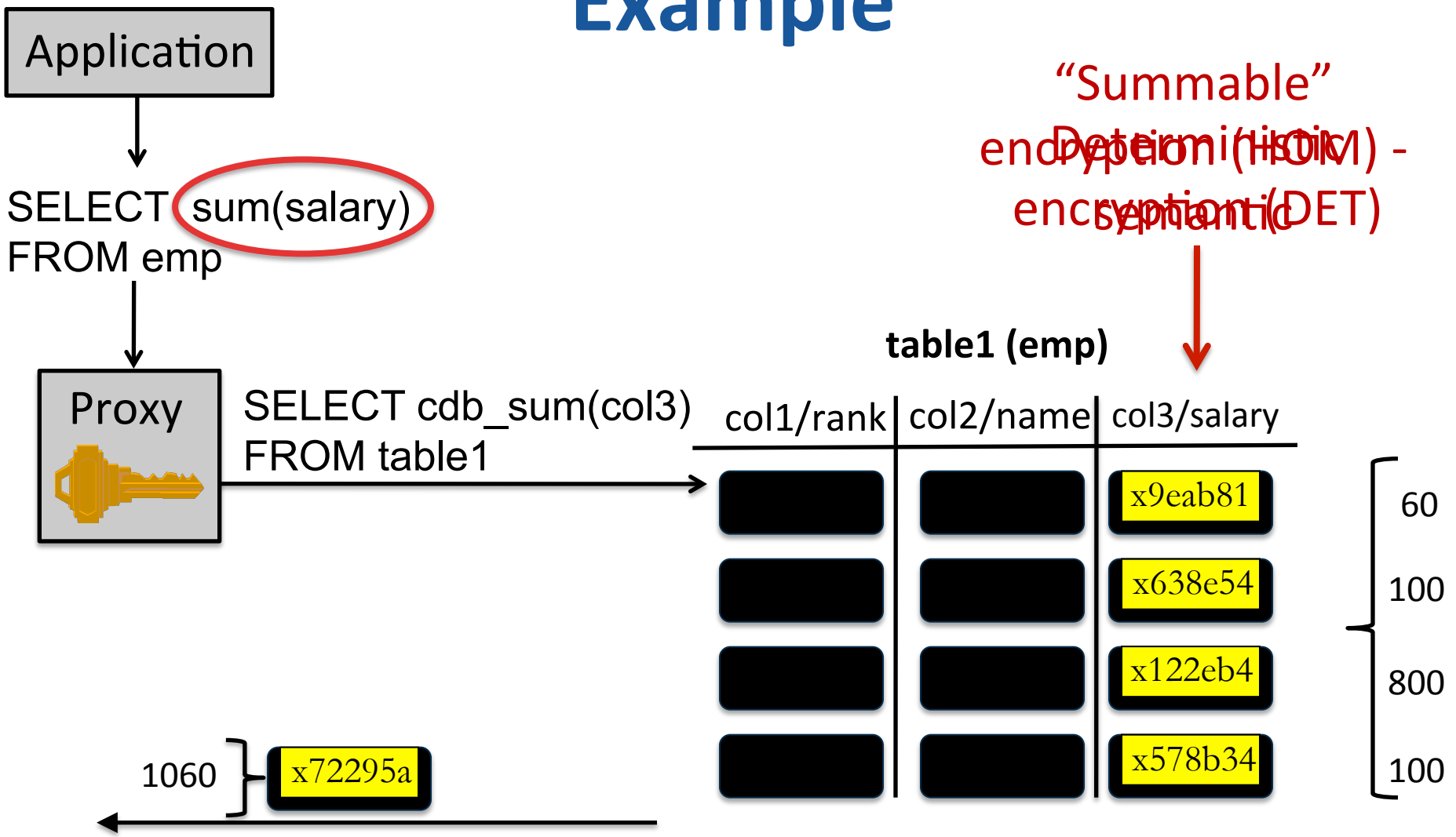
col1/rank	col2/name	col3/salary
		x934bc1
		x5a8c34 ✓
		x84a21c
		x5a8c34 ✓

60  
100  
800  
100

		x5a8c34
		x5a8c34



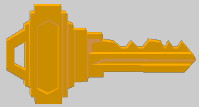
# Example



# Example

Application

SELECT **sqrt(sum(salary))**  
FROM emp

Proxy  


SELECT cdb\_sum(col3)  
FROM table1

table1 (emp)

col1/rank	col2/name	col3/salary
		x9eab81
		x638e54
		x122eb4
		x578b34

60  
100  
800  
100

1060 } x72295a

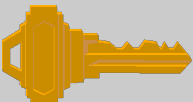
←

# Example

Application

SELECT sqrt(sum(salary))  
FROM emp

32.55

Proxy  


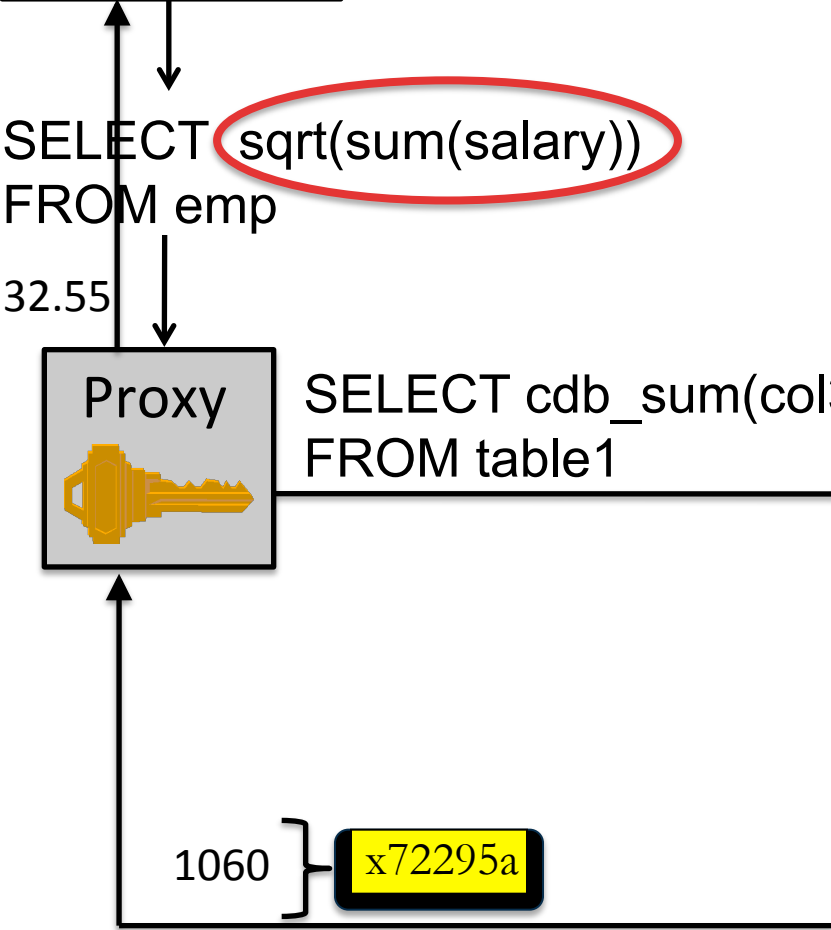
SELECT cdb\_sum(col3)  
FROM table1

table1 (emp)

col1/rank	col2/name	col3/salary
		x9eab81
		x638e54
		x122eb4
		x578b34

1060 } x72295a

60  
100  
800  
100



# Techniques

- Compute on encrypted data at the server
  - Use SQL-aware set of efficient encryption schemes
  - Adjust encryption of data based on queries
- Compute on decrypted data at the proxy
  - Can decrypt → can perform any computation
  - Choose optimal split to reduce bandwidth, proxy load

# SQL-aware encryption schemes

Security

Scheme	Construction	Function
--------	--------------	----------

≈ semantic security

RND	AES in UFE	data moving
HOM	Paillier	addition
SEARCH	Song et al., '00	word search

SQL operations:  
e.g., SELECT,  
UPDATE, DELETE,  
INSERT, COUNT  
e.g., SUM, +  
restricted ILIKE

reveals only repeat pattern

DET	AES in CMC	equality
JOIN	our new scheme	join

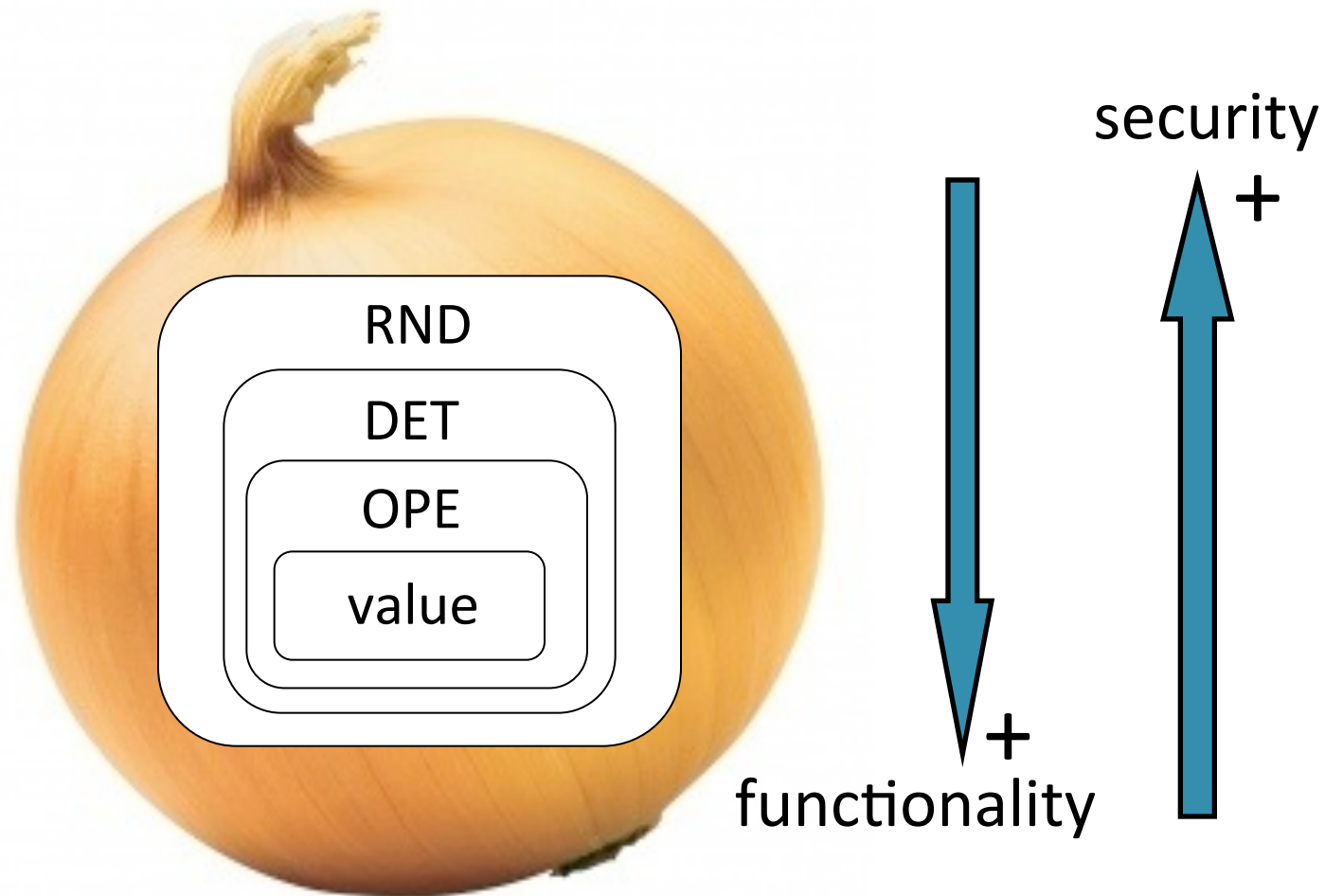
e.g., =, !=, IN,  
GROUP BY,  
DISTINCT

reveals only order

OPE	our new scheme	order
-----	----------------	-------

e.g., >, <, ORDER BY,  
ASC, DESC, MAX,  
MIN, GREATEST,  
LEAST

# Onion of encryptions

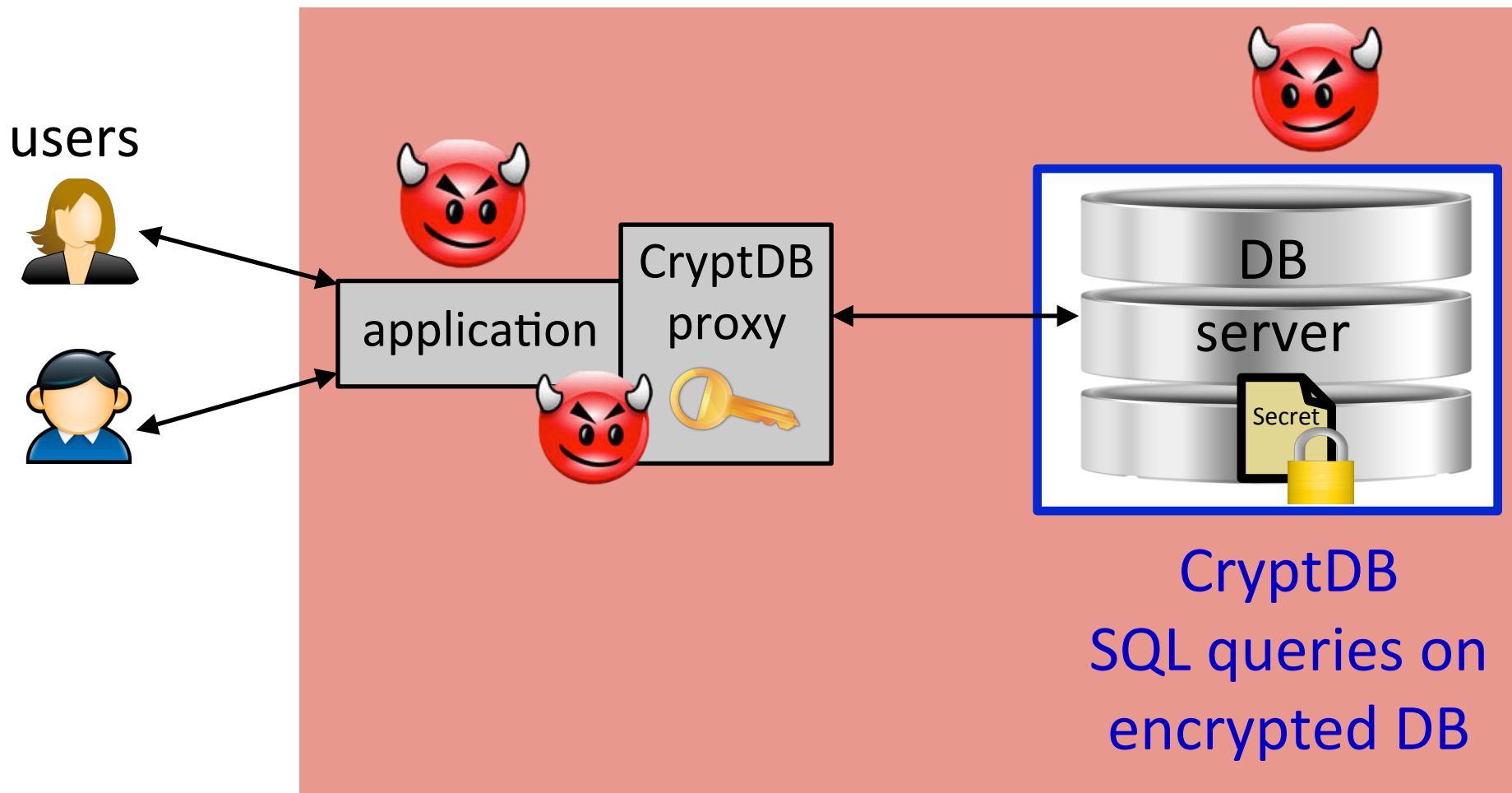


Adjust encryption: strip off layer of the onion

# CryptDB works well in practice

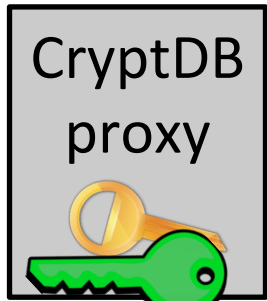
- Supports many database applications
  - Web sites, transactional processing, data analytics
  - Never reveals plaintext data on database server
- Modest performance overheads
  - 20-30% throughput loss for typical benchmarks
- Approach now used by Google (among others)
  - Encrypted BigQuery service

# Compromised app. server?

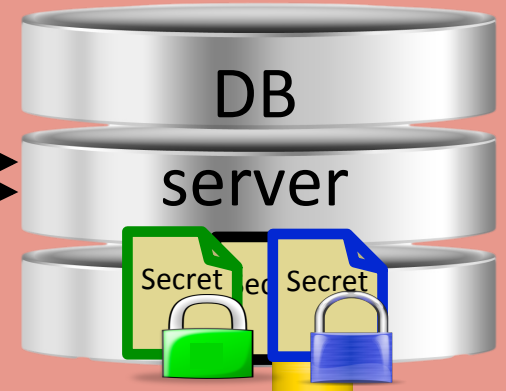
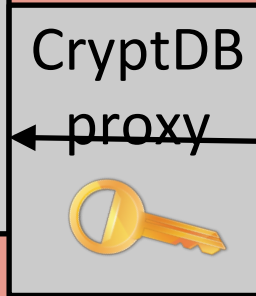
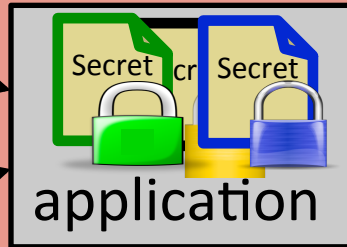




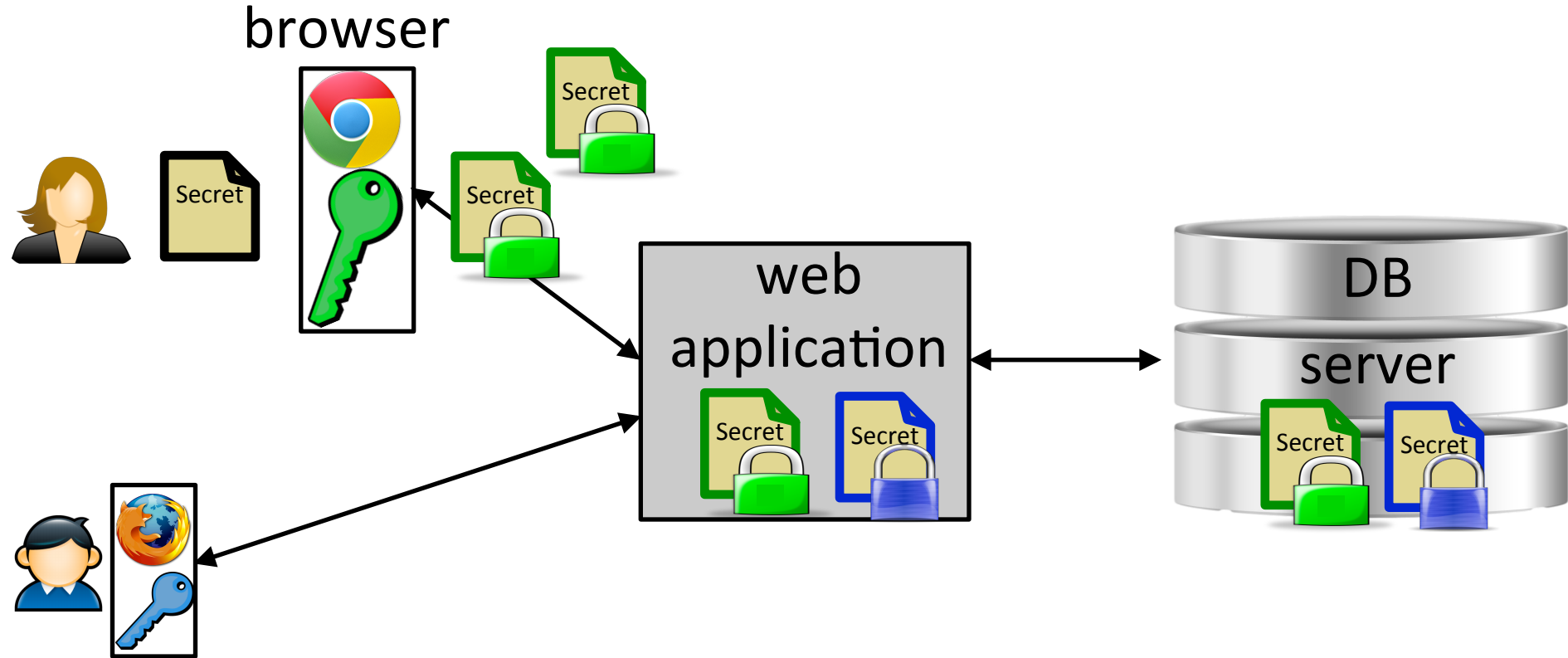
# Compromised app. server?



users



# Mylar: browser-side encryption



Decrypted data exists only in users' browsers

# Challenge: computation in web applications

## 1. Client-side application framework

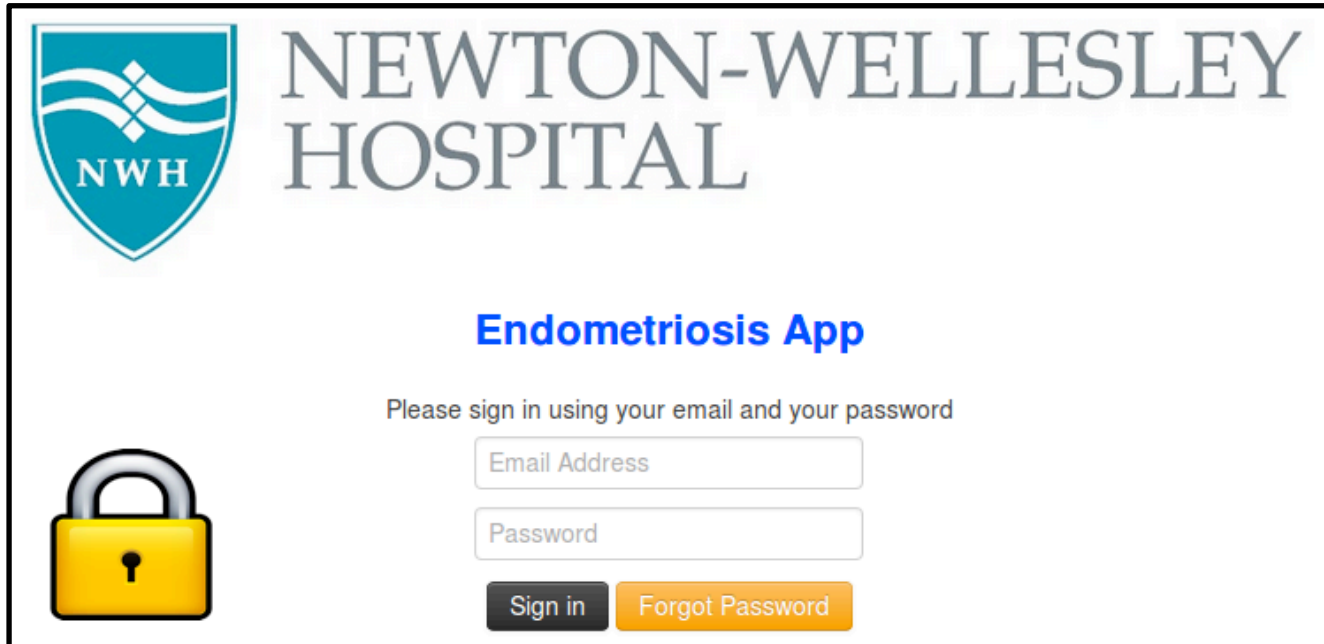
- Most computation happens in client's web browser (Javascript code)

## 2. Non client-side computation:

- Data sharing – need a way to manage keys
- Keyword search – need new cryptosystem: documents encrypted with *many keys*

# Mylar supports many applications

- Ported 6 applications to Mylar
- Performance overheads are modest
- Data privacy despite server compromises



The screenshot shows the login interface for the Endometriosis App. At the top left is the Newton-Wellesley Hospital (NWH) logo, a blue shield with white waves and the letters 'NWH'. To its right, the text 'NEWTON-WELLESLEY HOSPITAL' is displayed in a grey, serif font. Below the logo and hospital name, the title 'Endometriosis App' is written in a bold blue font. Underneath the title, a grey instruction reads 'Please sign in using your email and your password'. To the left of the input fields is a yellow padlock icon. There are two input fields: 'Email Address' and 'Password', both with light grey borders. At the bottom, there are two buttons: a dark grey 'Sign in' button and an orange 'Forgot Password' button.

# Future research directions

- Practical cryptography
  - Computing on data encrypted w/ many keys
  - Delegating limited functions over encrypted data
- Practical systems
  - Auditing for data disclosures
  - Protecting end-user computers