Order-Revealing Encryption:
How to Search on Encrypted Data

Kevin Lewi and David J. Wu
Stanford University

Searching on Encrypted Data

ars TECHNICA BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE FORUMS

EVENT VERIZON —

Yahoo admits it's been hacked again, and 1 billion accounts were exposed

That's a billion with a b—and is separate from the breach "cleared" in September.

SEAN GALLAGHER - 12/14/2016, 3:26 PM

The information accessed from potentially exposed accounts "may have included names, email addresses, telephone numbers, dates of birth, hashed passwords (using MD5) and, in some cases, encrypted or unencrypted security questions and answers…"

Searching on Encrypted Data

The database was discovered by MacKeeper researcher Chris Vickery on March 31, in the course of searching for random phrases on the domain s3.amazonaws.com.

“It's as bad as I expected,” he tweeted. “Bank-related. Plaintext passwords. Big name company. I've reached out to them.”

Searching on Encrypted Data


SECTIONS HOME SEARCH The New York Times SUBSCRIBE LOG IN

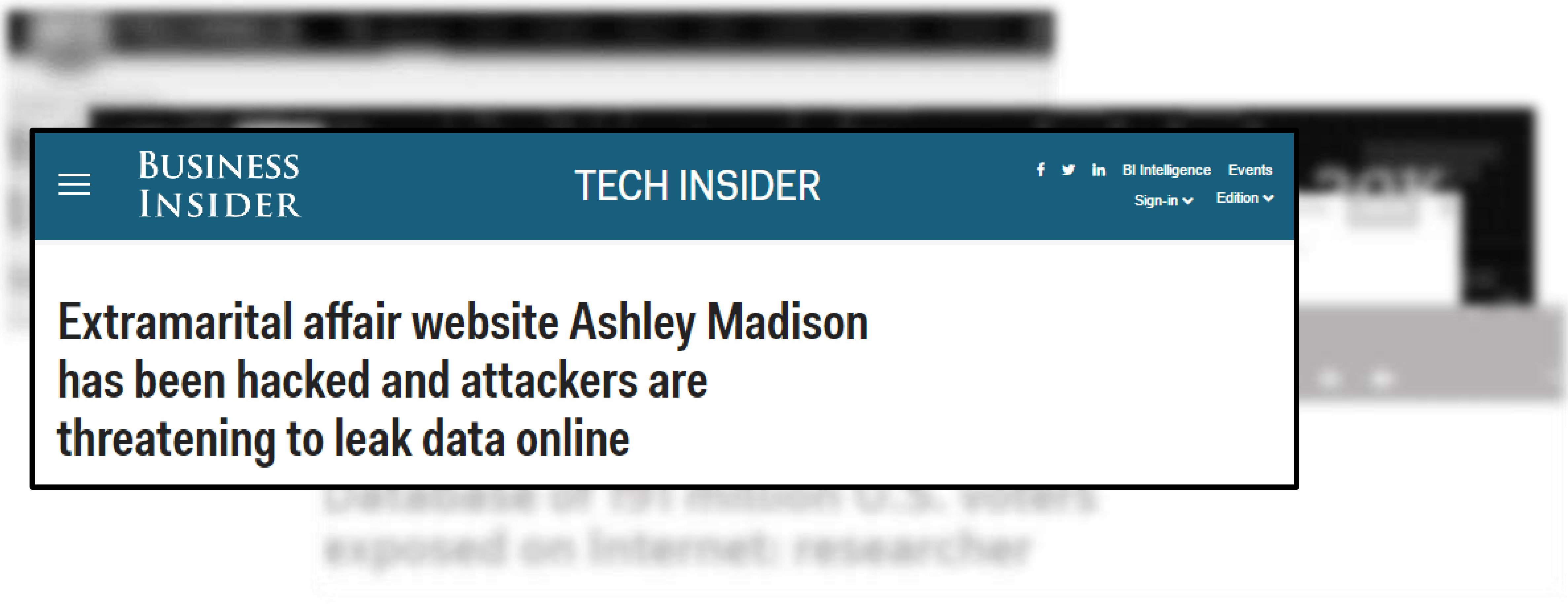
BUSINESS DAY

Data Breach at Anthem May Forecast a Trend

By REED ABELSON and JULIE CRESWELL FEB. 6, 2015

Searching on Encrypted Data

Searching on Encrypted Data


BUSINESS INSIDER

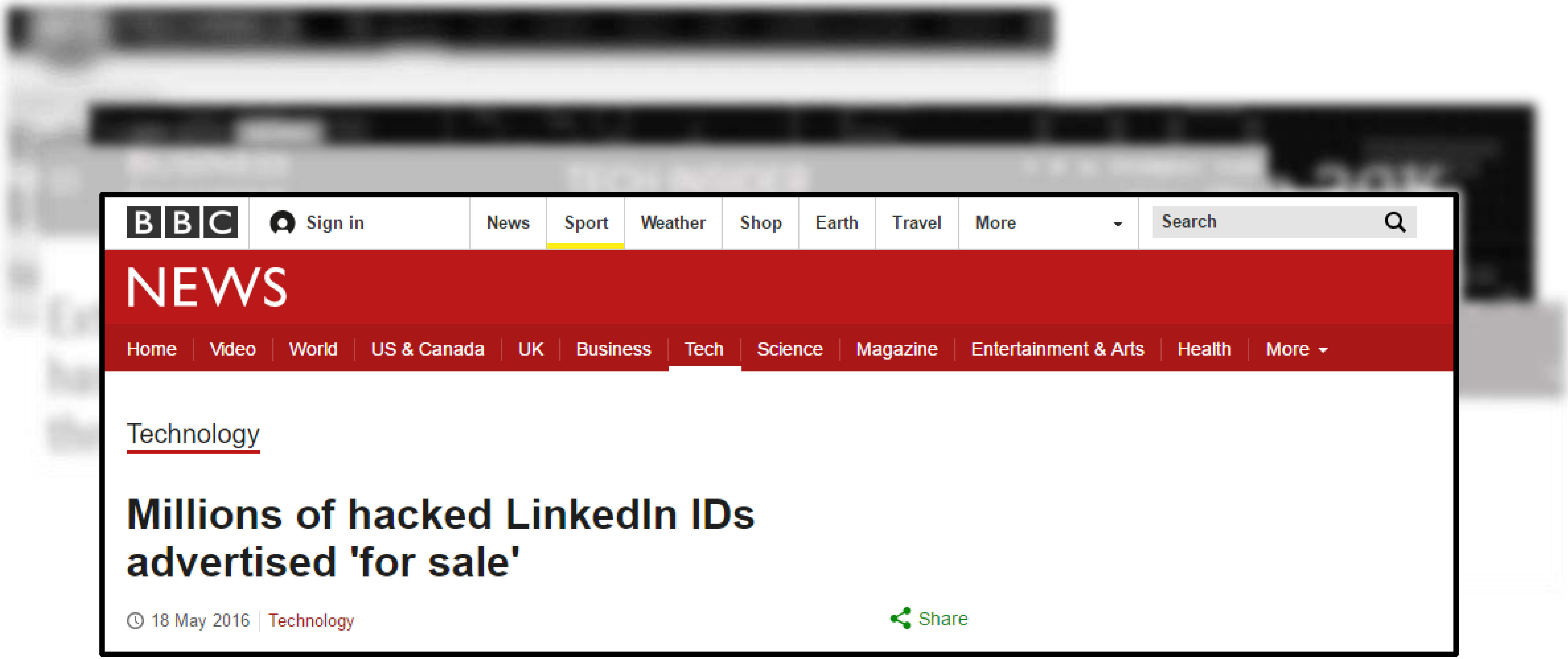
TECH INSIDER

BI Intelligence Events

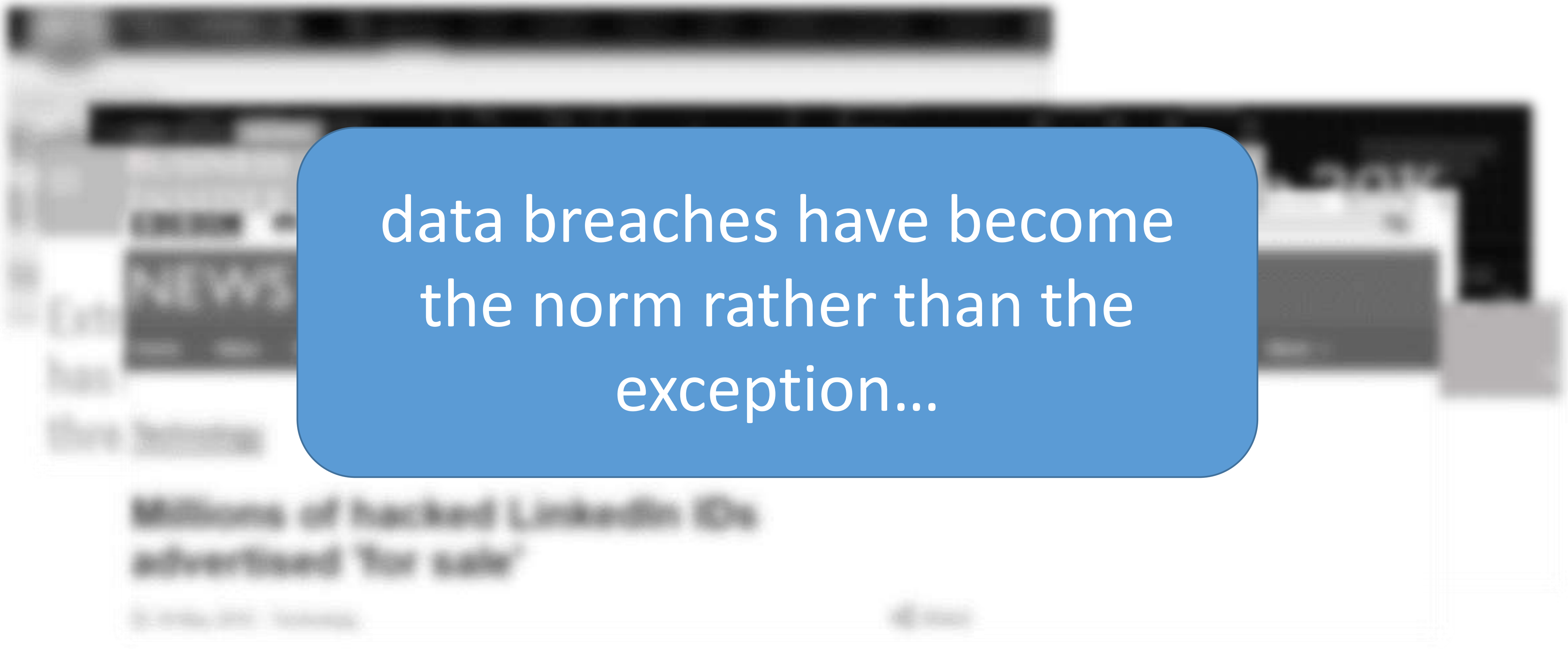
Sign-in Edition

Extramarital affair website Ashley Madison has been hacked and attackers are threatening to leak data online

Searching on Encrypted Data

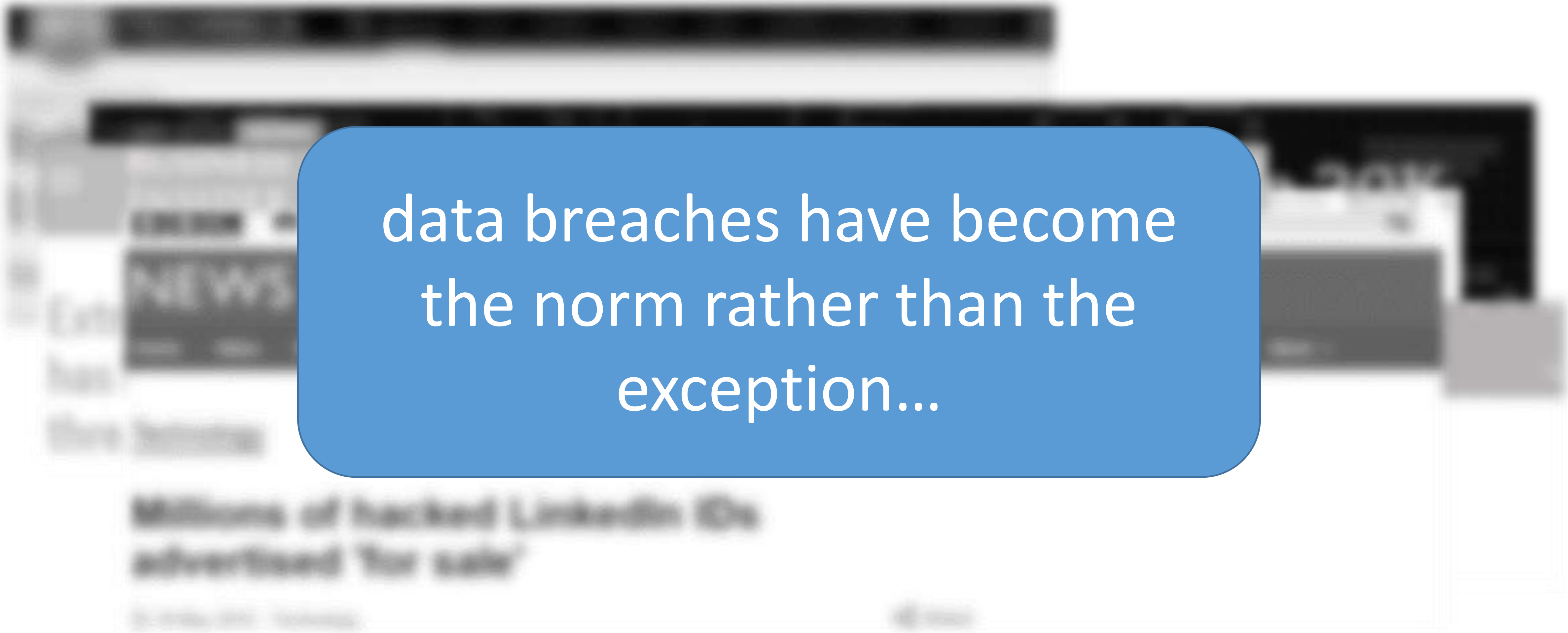


Searching on Encrypted Data



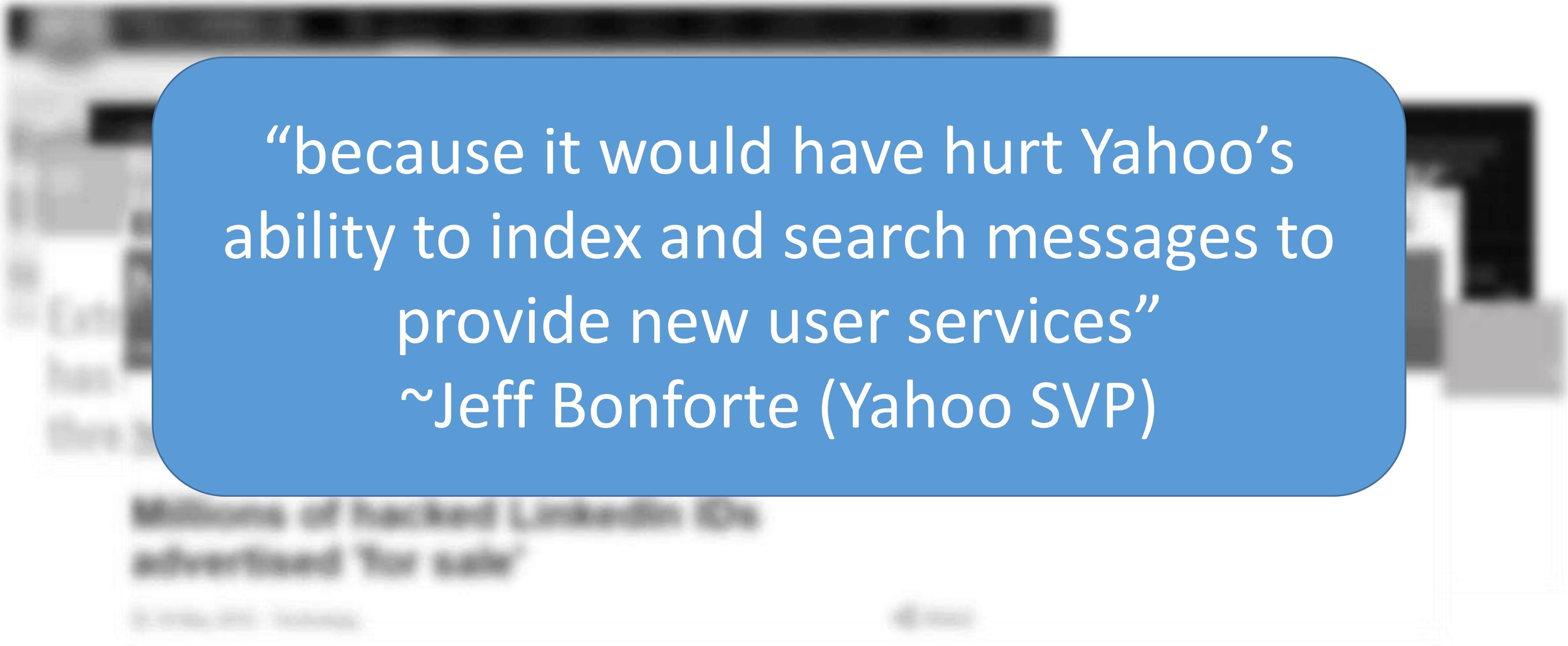
data breaches have become the norm rather than the exception...

Why Not Encrypt?



data breaches have become the norm rather than the exception…

Why Not Encrypt?



"because it would have hurt Yahoo's ability to index and search messages to provide new user services"
~Jeff Bonforte (Yahoo SVP)

Searching on Encrypted Data

database

ID	Name	Age	Diagnosis
0	Alice	31	2
1	Bob	47	3
2	Charlie	41	2
3	Inigo	45	4

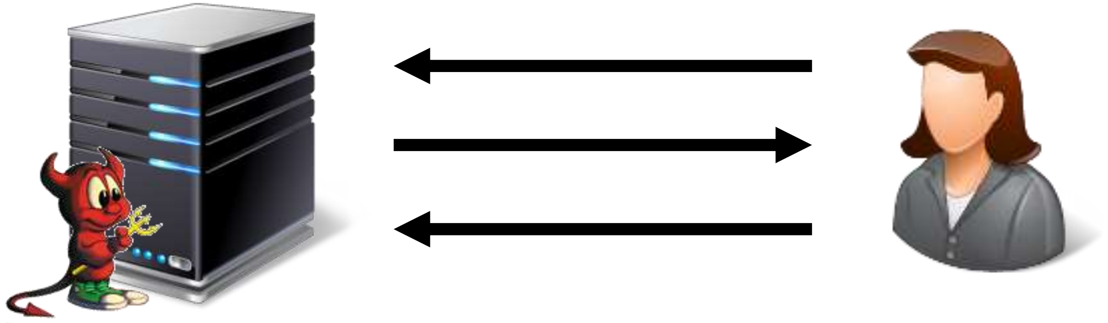
client

client holds a secret key (needed to encrypt + query the server)

server

server stores encrypted database

Security for Encrypted Search



adversary sees encrypted database +
queries and can interact with the database

active adversary

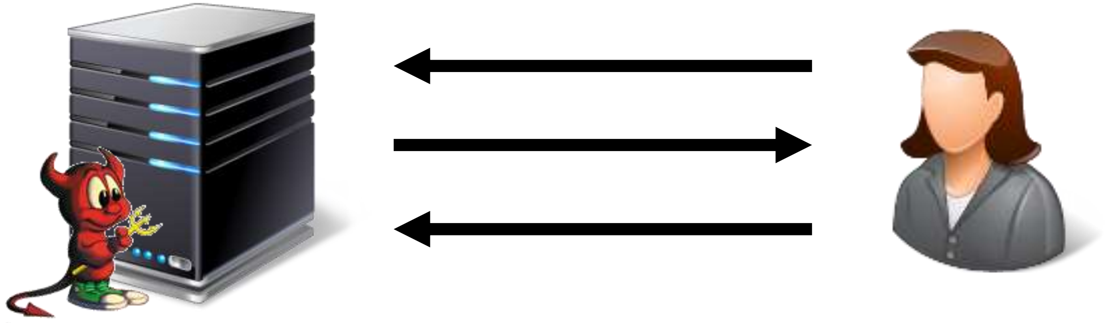
online attacks (e.g., active corruption)

offline attacks (e.g., passive snapshots)

adversary only sees contents
of encrypted database

snapshot adversary

Security for Encrypted Search



adversary sees encrypted database + queries and can interact with the database

online attacks (e.g., active corruption)

offline attacks (e.g., passive snapshots)

adversary only sees contents of encrypted database

typical database breach: contents of database are stolen and dumped onto the web

Order-Revealing Encryption [BLRSZZ'15]

secret-key encryption scheme

Which is greater: the value encrypted by ct_1 or the value encrypted by ct_2?

sk

client

$$\begin{aligned} \mathrm{ct}_1 &= \mathrm{Enc}(\mathrm{sk}, 123) \\ \mathrm{ct}_2 &= \mathrm{Enc}(\mathrm{sk}, 512) \\ \mathrm{ct}_3 &= \mathrm{Enc}(\mathrm{sk}, 273) \end{aligned}$$

server

(legacy-friendly) range queries on encrypted data

Order-Revealing Encryption [BLRSZZ'15]

given any two ciphertexts

$\mathrm{ct}_1 = \mathrm{Enc}(\mathrm{sk}, x)$

$\mathrm{ct}_2 = \mathrm{Enc}(\mathrm{sk}, y)$

$x > y$

there is a <u>public</u> function for performing comparisons

OPE [BCLO'09]: comparison function is <u>numeric comparison</u> on ciphertexts

Order-Revealing Encryption [BLRSZZ'15]

given any two ciphertexts

$\mathrm{ct}_1 = \mathrm{Enc}(\mathrm{sk}, x)$

$\mathrm{ct}_2 = \mathrm{Enc}(\mathrm{sk}, y)$

$x > y$

best-possible security: reveal just the ordering and nothing more

in practice: constructions reveal some additional information

Inference Attacks [NKW'15, DDC'16, GSBNR'16]

ID	Name	Age	Diagnosis
wpjOos	2wzXW8	SqX9l9	KqLUXE
XdXdg8	y9GFpS	gwilE3	MJ23b7
P6vKhW	EgN0Jn	S0pRJe	aTaeJk
orJRe6	KQWy9U	tPWF3M	4FBEO0

encrypted database

+

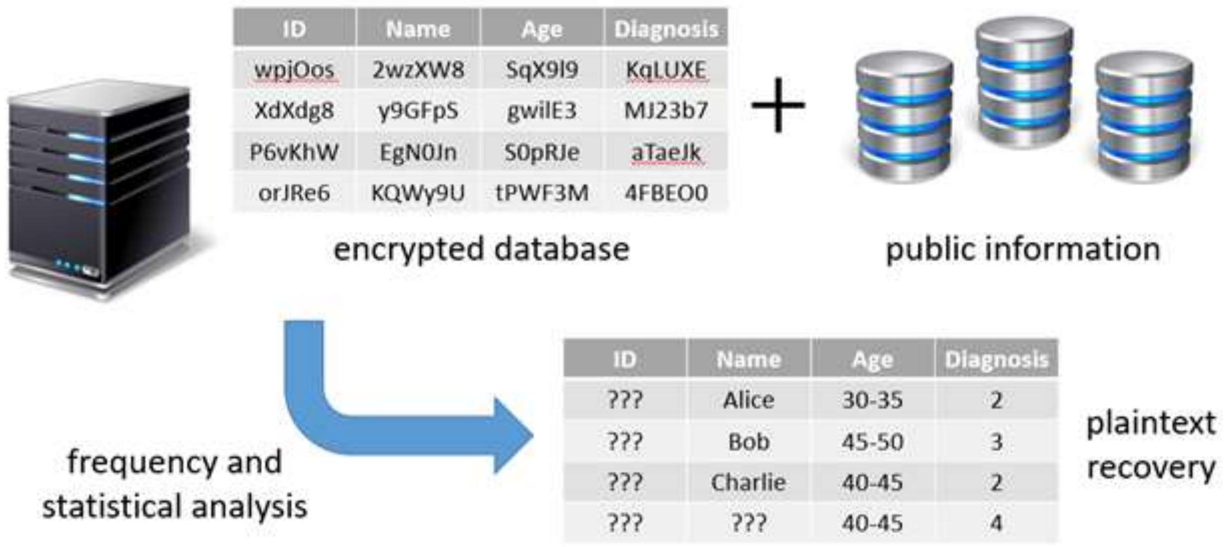
public information

frequency and
statistical analysis

ID	Name	Age	Diagnosis
???	Alice	30-35	2
???	Bob	45-50	3
???	Charlie	40-45	2
???	???	40-45	4

plaintext
recovery

Inference Attacks [NKW’15, DDC’16, GSBNR’16]



ID	Name	Age	Diagnosis
wpjOos	2wzXW8	SqX9l9	KqLUXE
XdXdg8	y9GFpS	gwilE3	MJ23b7
P6vKhW	EgN0Jn	S0pRJe	aTaeJk
orJRe6	KQWy9U	tPWF3M	4FBEO0

ID	Name	Age	Diagnosis
???	Alice	30-35	2
???	Bob	45-50	3
???	Charlie	40-45	2
???	???	40-45	4

PPE schemes <u>always</u> reveal certain properties (e.g., equality, order) on ciphertexts and thus, are vulnerable to <u>offline inference attacks</u>

Can we <u>fully</u> defend against offline inference attacks while remaining legacy-friendly?

This Work

Can we <u>fully</u> defend against offline inference attacks while remaining legacy-friendly?

Trivial solution: encrypt the entire database, and have client provide decryption key at query time

But zero online security!

Desiderata: an ORE scheme that enables:

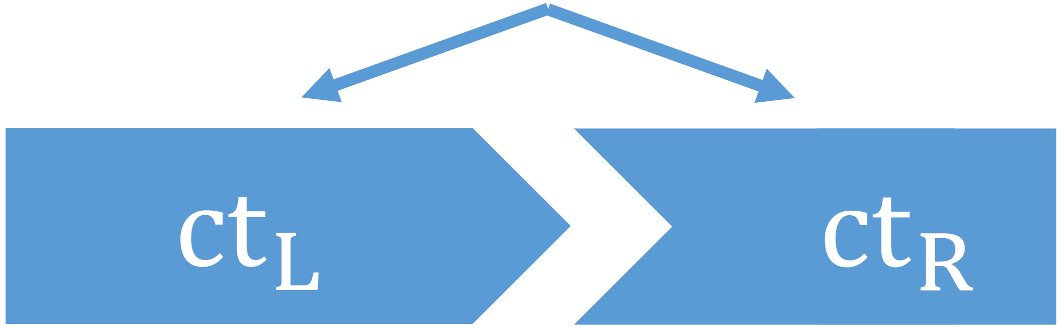
- perfect offline security
- limited leakage in the online setting

ORE with Additional Structure

Focus of this work: performing range queries on encrypted data

Key primitive: order-revealing encryption scheme where ciphertexts have a “decomposable” structure

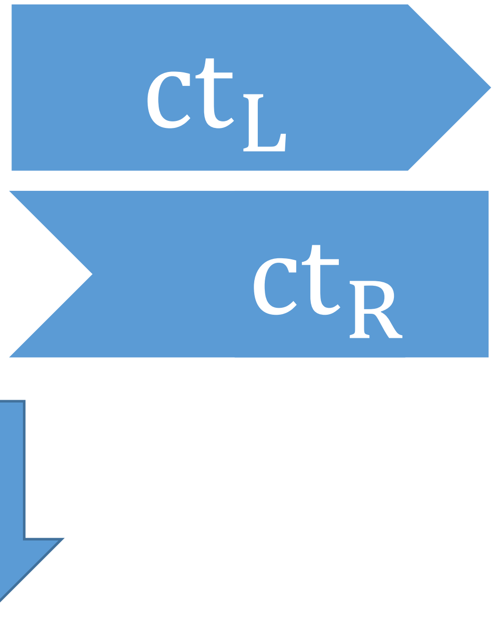
$\mathrm{Enc}(101)$



ciphertexts naturally split into two components

$\mathrm{Enc}_L(101)$

$\mathrm{Enc}_R(100)$



greater than

ORE with Additional Structure

$\mathrm{Enc}_{\mathrm{L}}(101)$ ct_{L}

$\mathrm{Enc}_{\mathrm{R}}(100)$ ct_{R}

comparison can be performed between left ciphertext and right ciphertext

right ciphertexts provide **semantic security**!

robustness against offline inference attacks!

Encrypted Range Queries

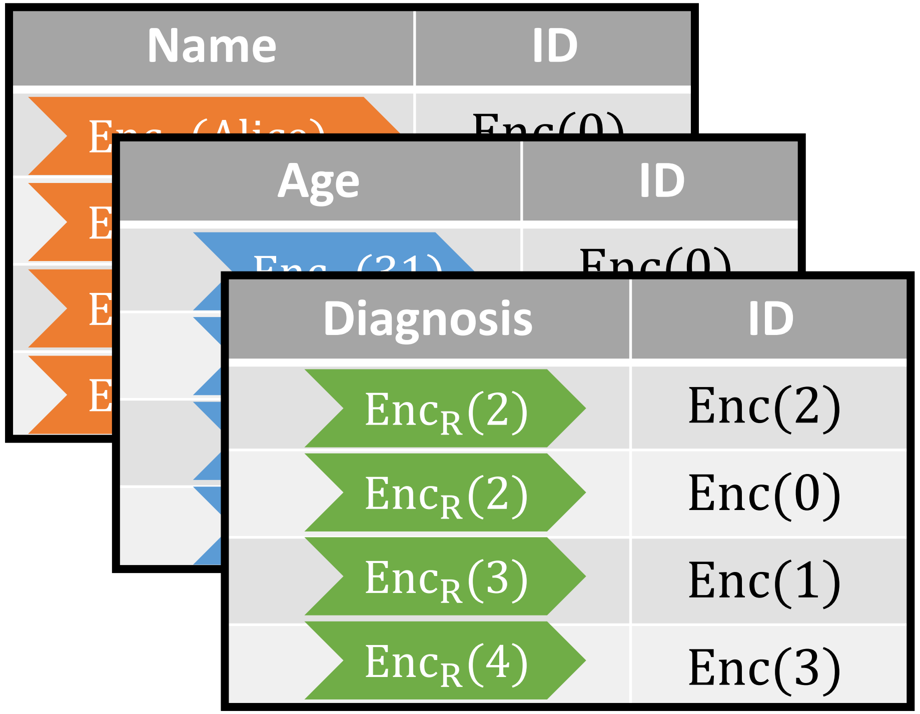
ID	Name	Age	Diagnosis
0	Alice	31	2
1	Bob	47	3
2	Charlie	41	2
3	Inigo	45	4

build encrypted index

store right ciphertexts in sorted order

Age	ID
$\mathrm{Enc_R}(31)$	$\mathrm{Enc}(0)$
$\mathrm{Enc_R}(41)$	$\mathrm{Enc}(2)$
$\mathrm{Enc_R}(45)$	$\mathrm{Enc}(3)$
$\mathrm{Enc_R}(47)$	$\mathrm{Enc}(1)$

record IDs encrypted under independent key



separate index for each searchable column, and using independent ORE keys

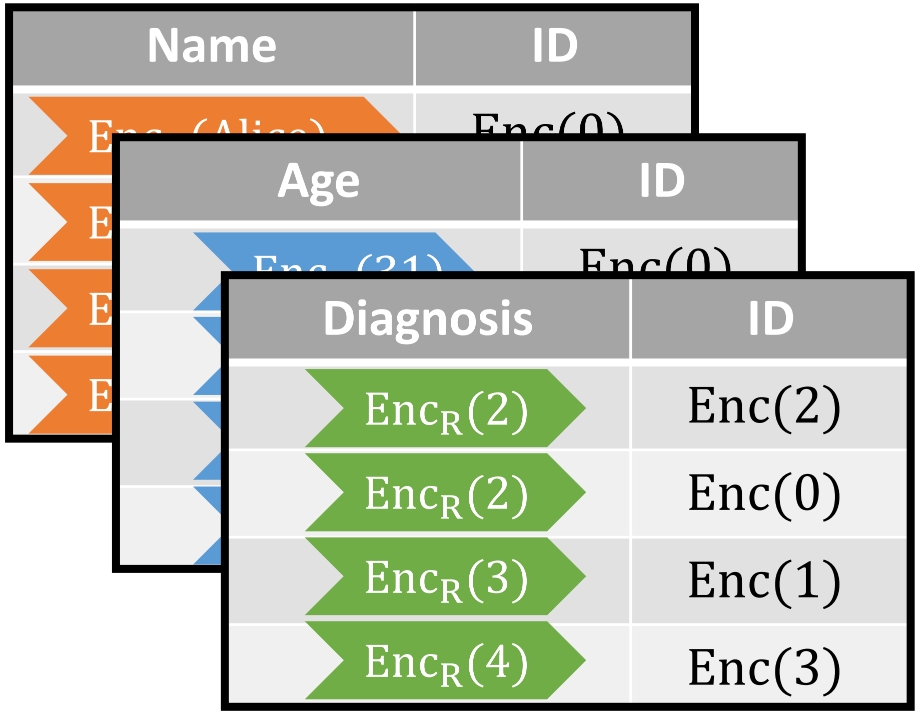
Encrypted Range Queries

Encrypted database:

ID	Name	Age	Diagnosis
0	Alice	31	2
1	Bob	47	3
2	Charlie	41	2
3	Inigo	45	4

columns (other than ID) are encrypted using a semantically-secure encryption scheme

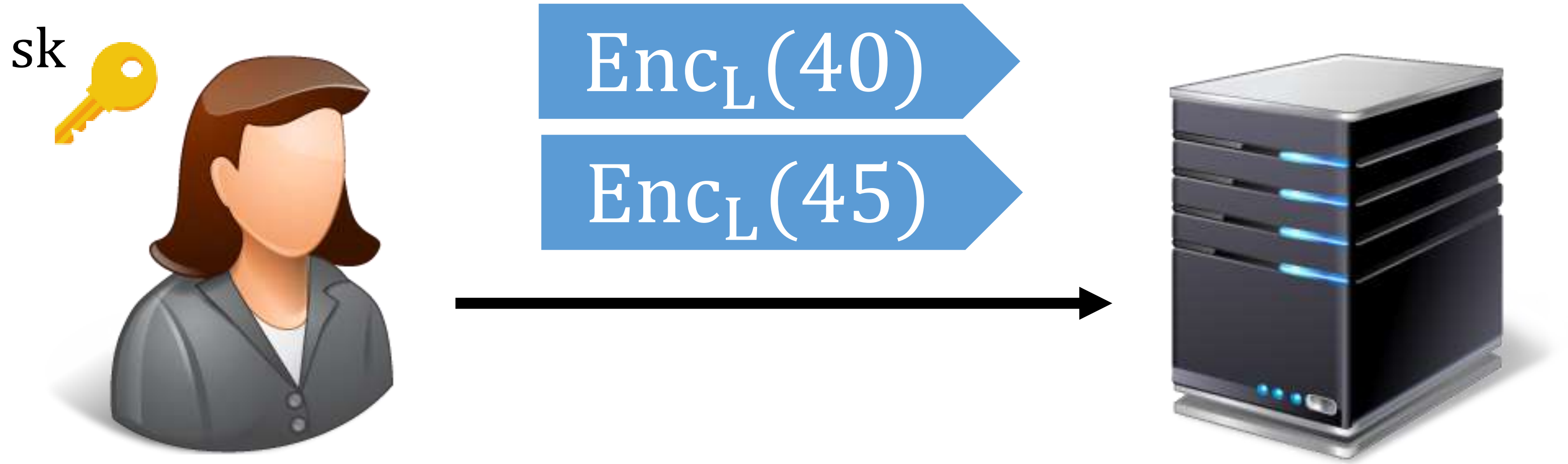
clients hold (secret) keys needed to decrypt and query database



encrypted search indices

Encrypted Range Queries

Query for all records where $40 \geq \text{age} \geq 45$:



Encrypted Range Queries

Query for all records where $40 \geq \text{age} \geq 45$:

$\mathrm{Enc}_L(40)$

$\mathrm{Enc}_L(45)$

Age	ID
$\mathrm{Enc}_R(31)$	$\mathrm{Enc}(0)$
$\mathrm{Enc}_R(41)$	$\mathrm{Enc}(2)$
$\mathrm{Enc}_R(45)$	$\mathrm{Enc}(3)$
$\mathrm{Enc}_R(47)$	$\mathrm{Enc}(1)$

Encrypted Range Queries

Query for all records where $40 \geq \text{age} \geq 45$:

$\text{Enc}_L(40)$

$\text{Enc}_L(45)$

Age	ID
$\text{Enc}_R(31)$	$\text{Enc}(0)$
$\text{Enc}_R(41)$	$\text{Enc}(2)$
$\text{Enc}_R(45)$	$\text{Enc}(3)$
$\text{Enc}_R(47)$	$\text{Enc}(1)$

use binary search to determine
endpoints (comparison via ORE)

Encrypted Range Queries

Query for all records where $40 \geq \text{age} \geq 45$:

$\mathrm{Enc}_L(40)$

$\mathrm{Enc}_L(45)$

Age	ID
$\mathrm{Enc}_R(31)$	$\mathrm{Enc}(0)$
$\mathrm{Enc}_R(41)$	$\mathrm{Enc}(2)$
$\mathrm{Enc}_R(45)$	$\mathrm{Enc}(3)$
$\mathrm{Enc}_R(47)$	$\mathrm{Enc}(1)$

use binary search to determine endpoints (comparison via ORE)

Encrypted Range Queries

Query for all records where $40 \geq \text{age} \geq 45$:

$\text{Enc}_L(40)$

$\text{Enc}_L(45)$

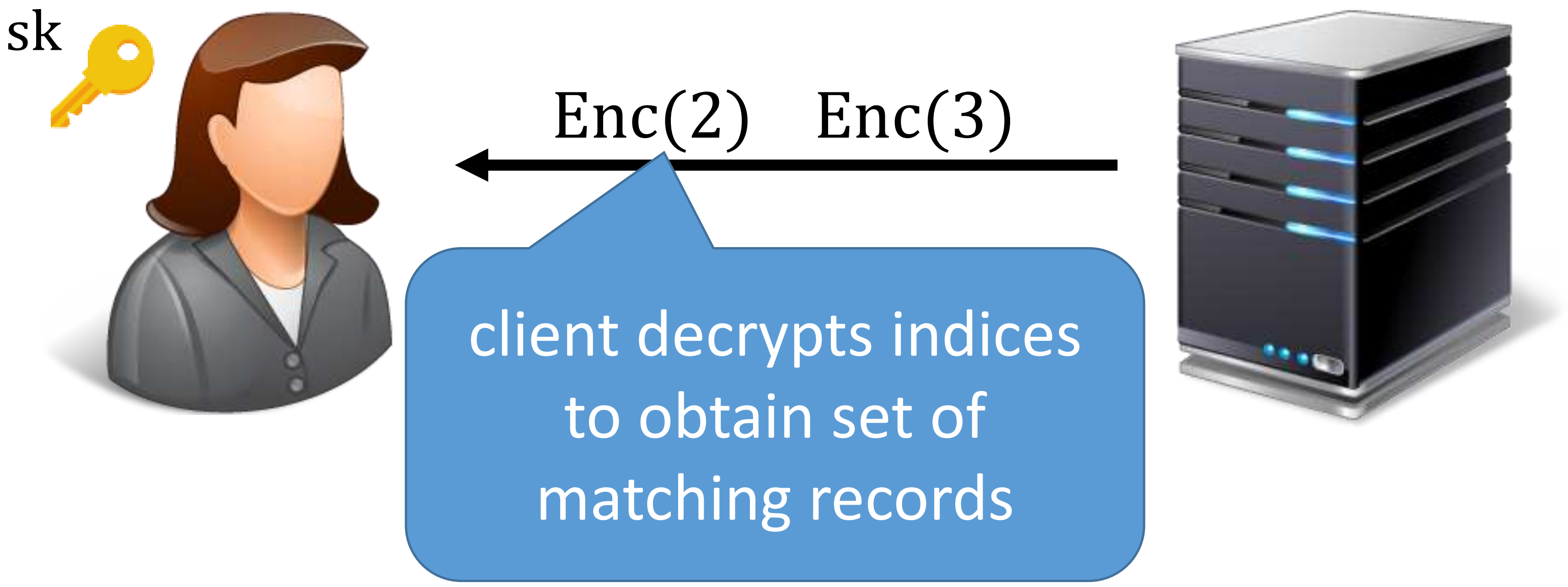
Age	ID
$\text{Enc}_R(31)$	$\text{Enc}(0)$
$\text{Enc}_R(41)$	$\text{Enc}(2)$
$\text{Enc}_R(45)$	$\text{Enc}(3)$
$\text{Enc}_R(47)$	$\text{Enc}(1)$

return encrypted indices that match query

use binary search to determine endpoints (comparison via ORE)

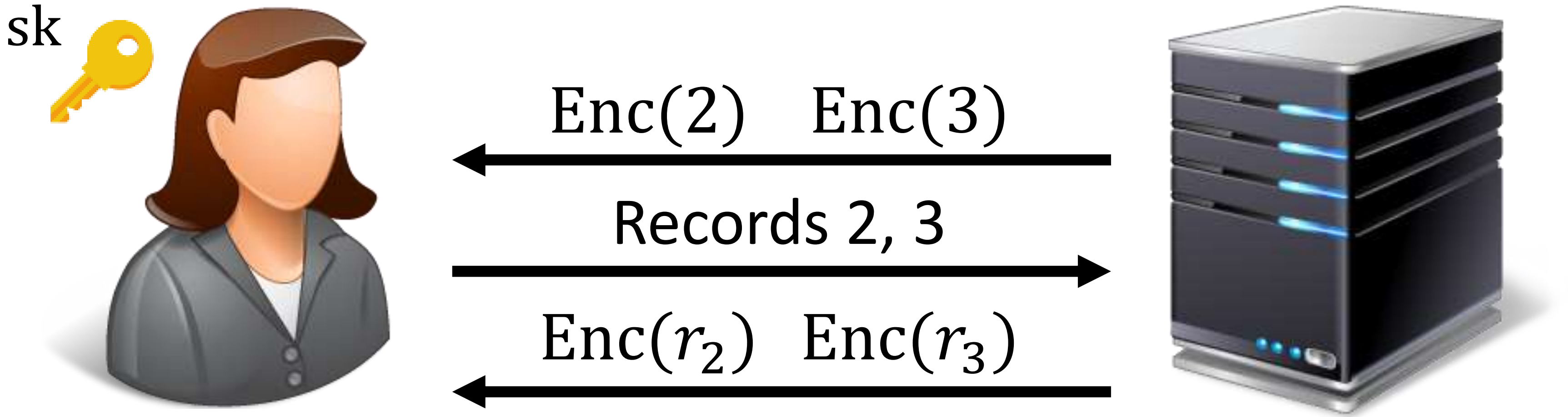
Encrypted Range Queries

Query for all records where $40 \geq \text{age} \geq 45$:



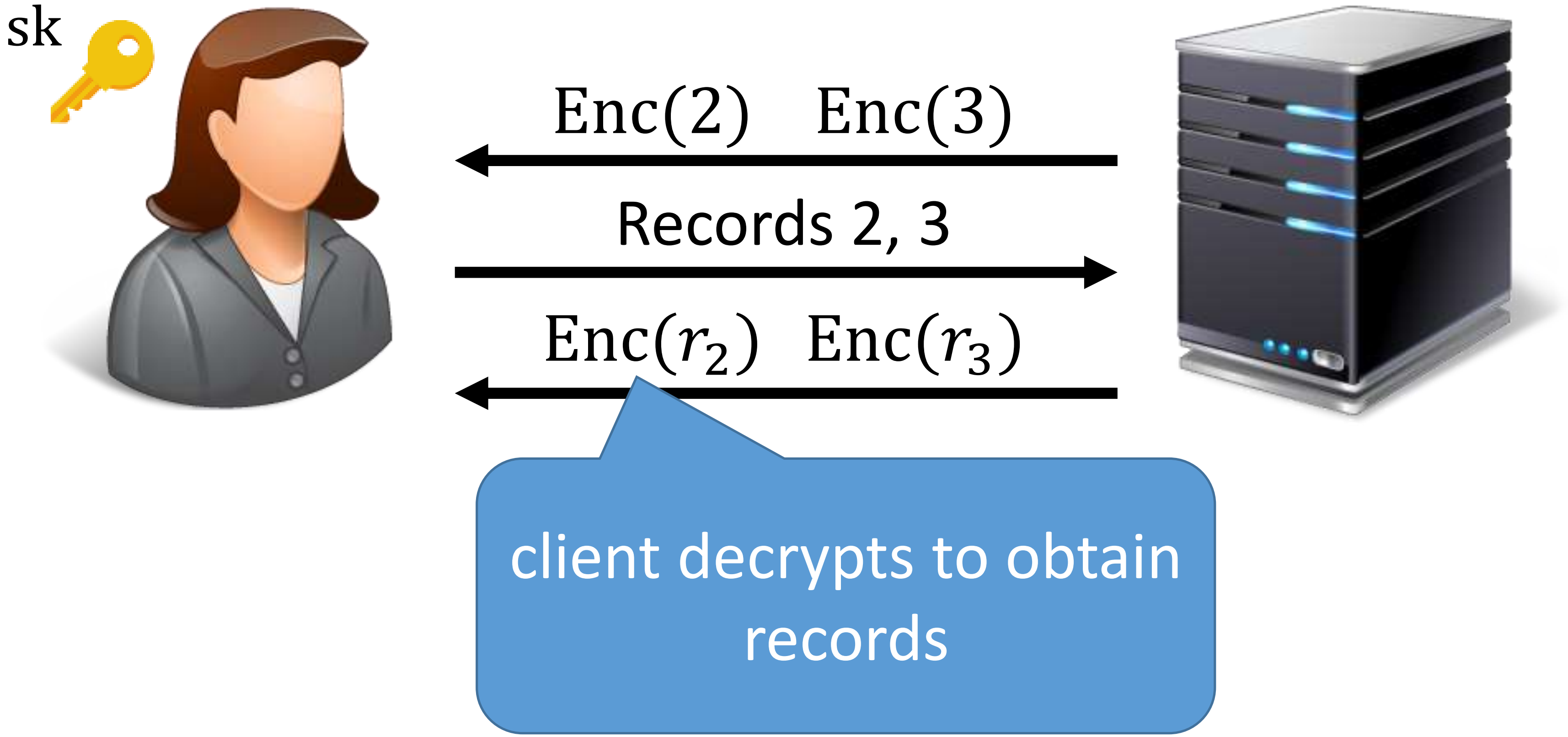
Encrypted Range Queries

Query for all records where $40 \geq \text{age} \geq 45$:



Encrypted Range Queries

Query for all records where $40 \geq \text{age} \geq 45$:



Encrypted Range Queries

Query for all records where $40 \geq \text{age} \geq 45$:

sk

$\mathrm{Enc}(2)$ $\mathrm{Enc}(3)$

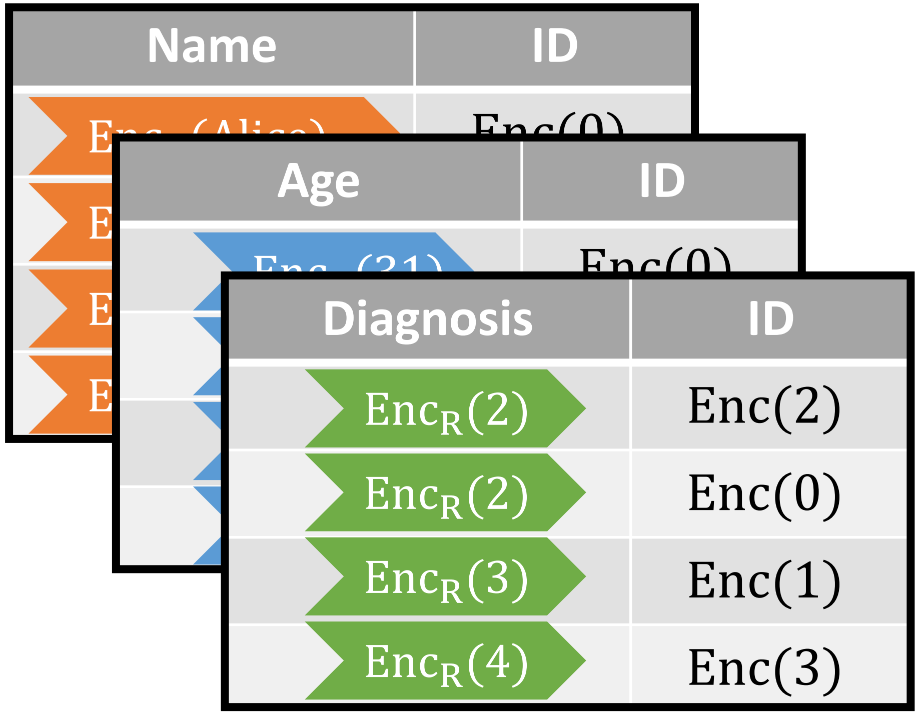
Records 2, 3

some online leakage: access pattern + ORE leakage

Encrypted Range Queries

Encrypted database (view of the snapshot adversary):

ID	Name	Age	Diagnosis
0	Alice	31	2
1	Bob	47	3
2	Charlie	41	2
3	Inigo	45	4



encrypted search indices

encrypted database is semantically secure!

<u>Perfect offline security</u>

Our New ORE Scheme

"small-domain" ORE with best-possible security

+

domain extension technique inspired by CLWW'16

→

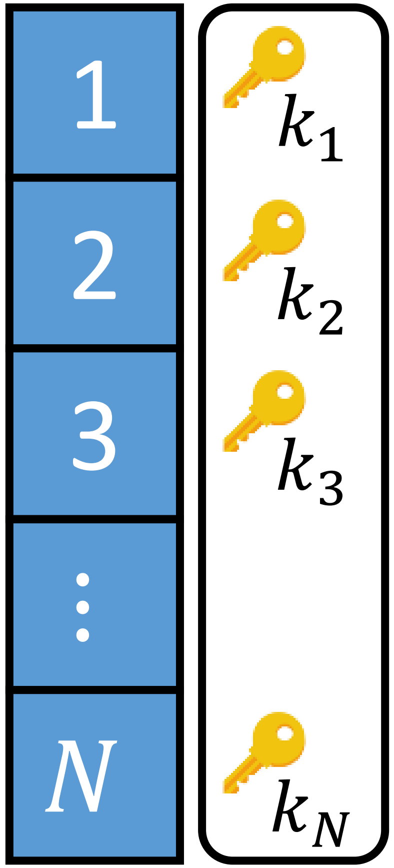
"large-domain" ORE with some leakage

first practical ORE construction that can provide best-possible offline security!

Small-Domain ORE with Best-Possible Security

Suppose plaintext space is small: $\{1,2,\ldots,N\}$

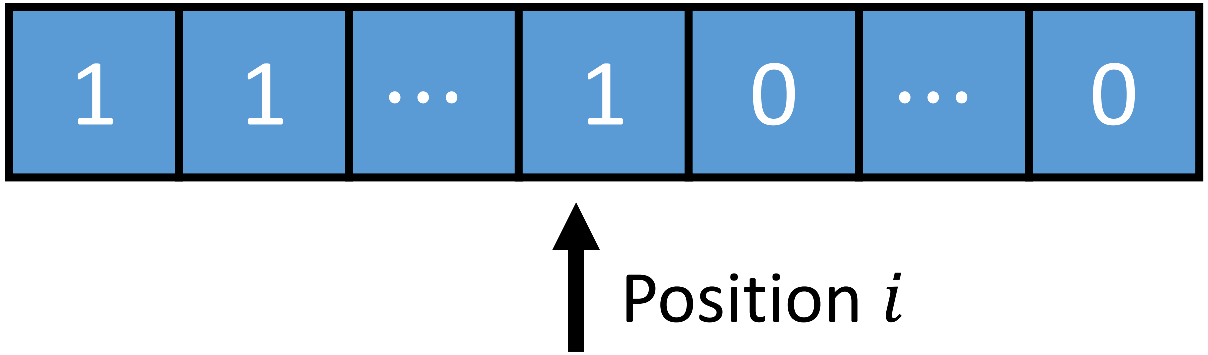
associate a key with each value



$(k_1,\ldots,k_N)$ is the secret key (can be derived from a PRF)

Small-Domain ORE with Best-Possible Security

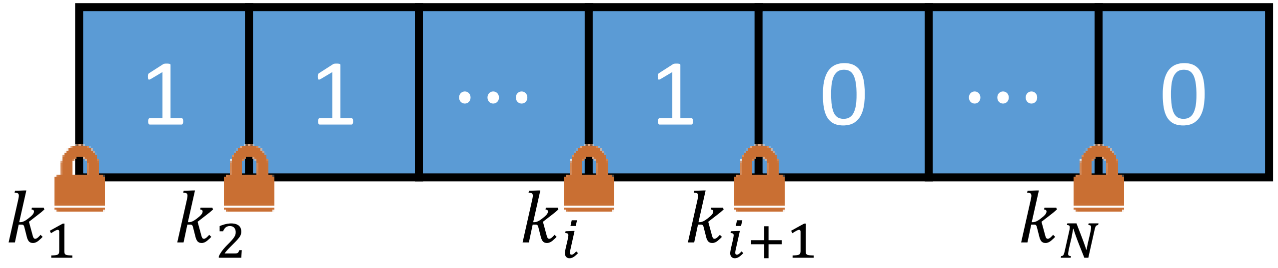
Encrypting a value i



Invariant: all positions $\leq i$ have value 1 while all positions $> i$ have value 0

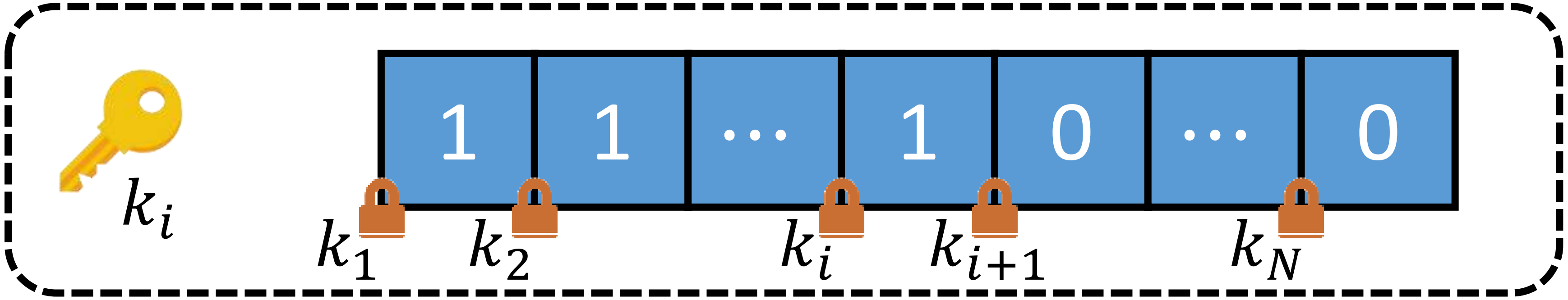
Small-Domain ORE with Best-Possible Security

Encrypting a value i



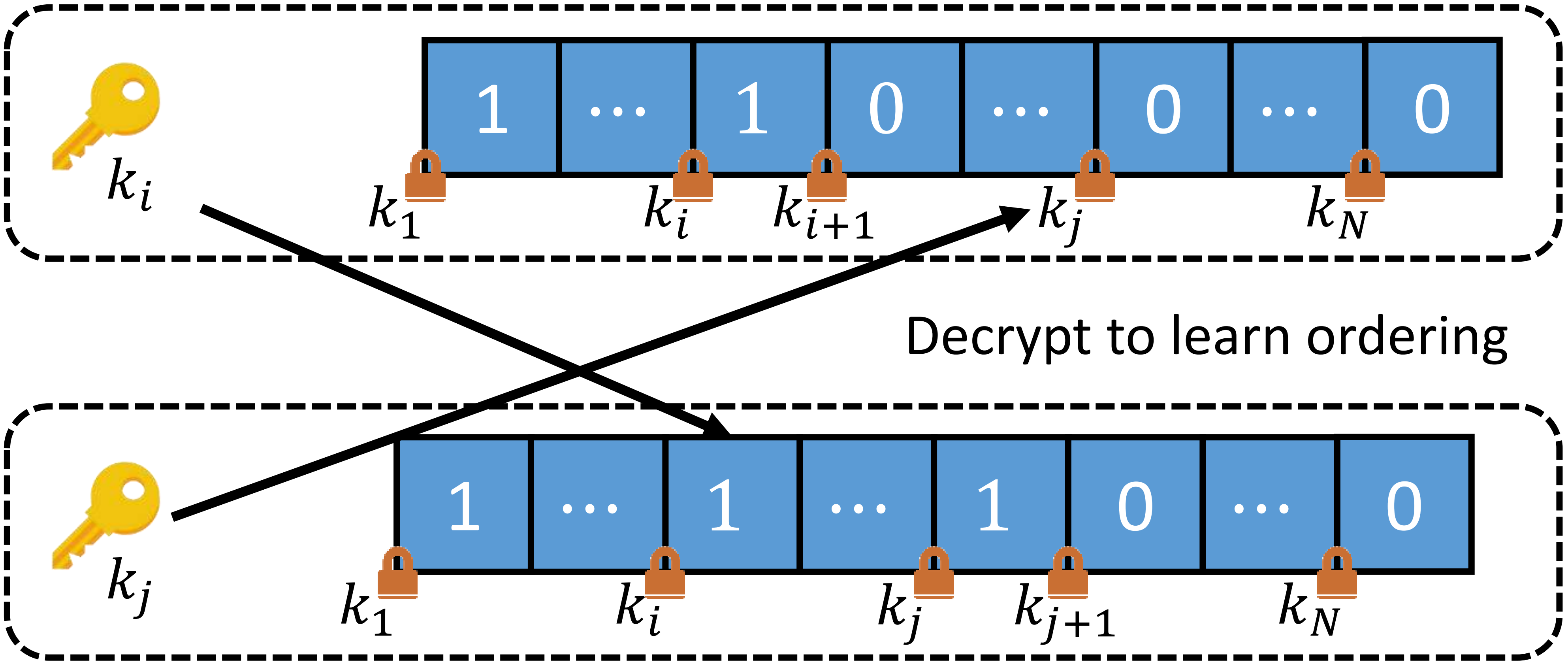
encrypt each slot with key for that slot

To allow comparisons, also give out key for slot i



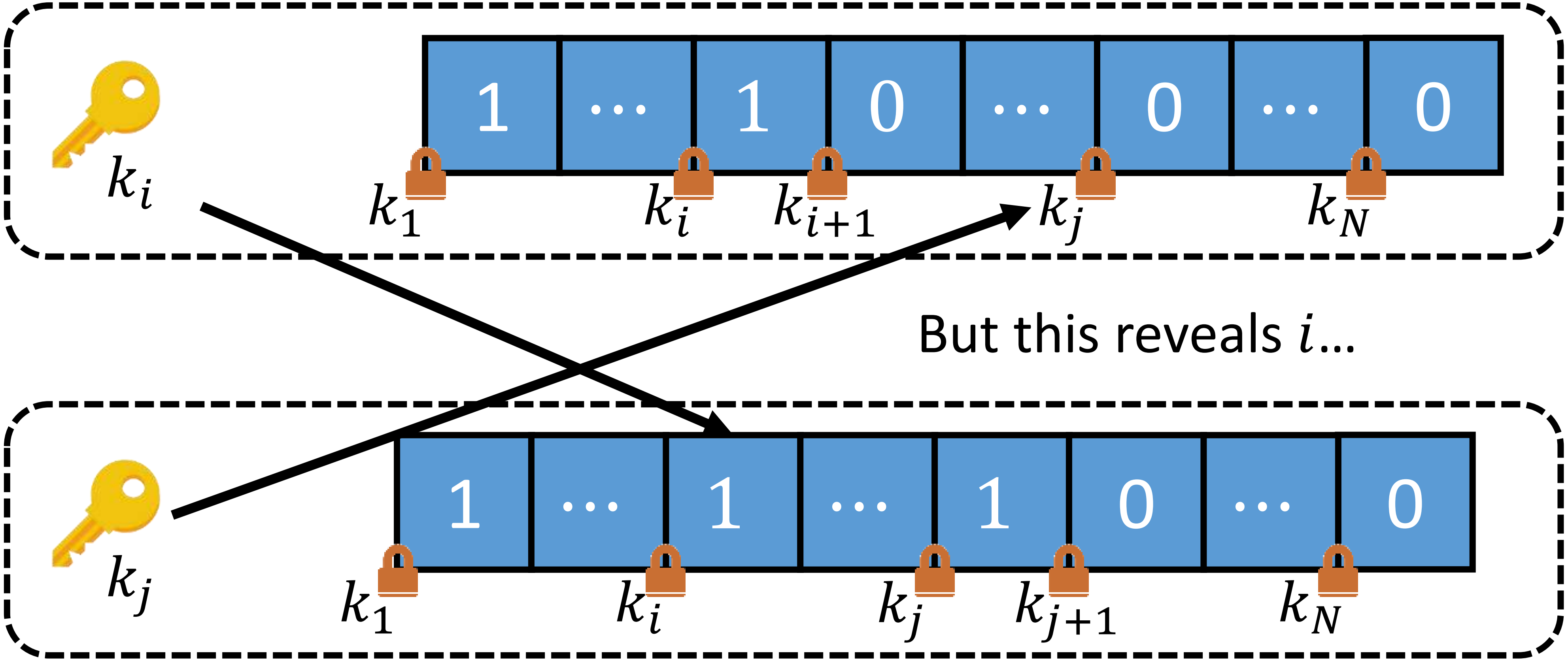
Small-Domain ORE with Best-Possible Security

Given two ciphertexts



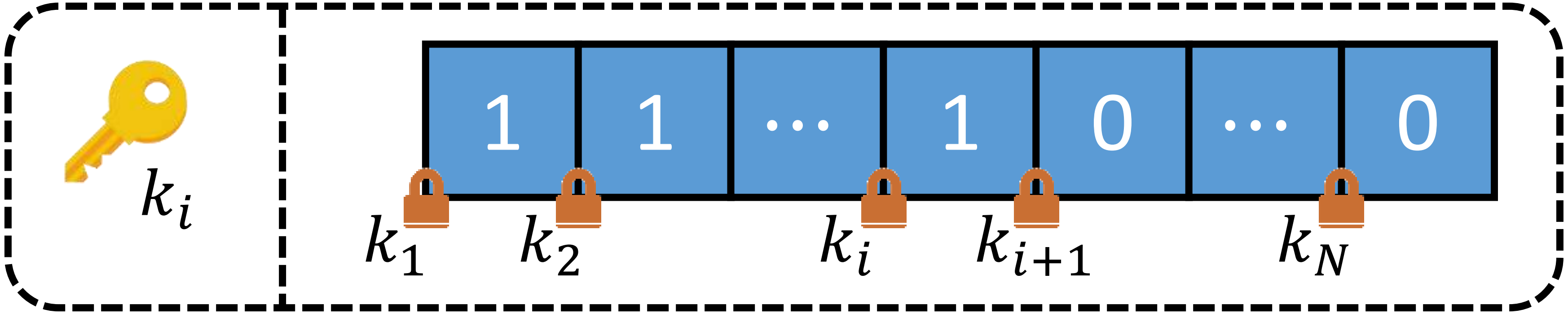
Small-Domain ORE with Best-Possible Security

Given two ciphertexts



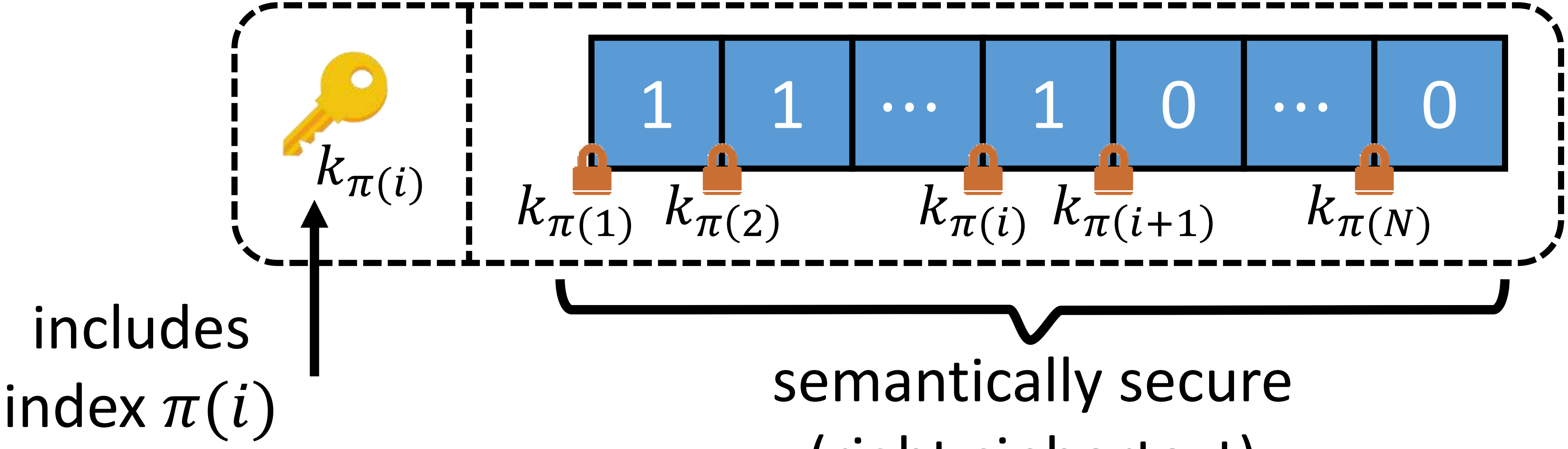
Small-Domain ORE with Best-Possible Security

Solution: apply random permutation π (part of the secret key) to the slots



Small-Domain ORE with Best-Possible Security

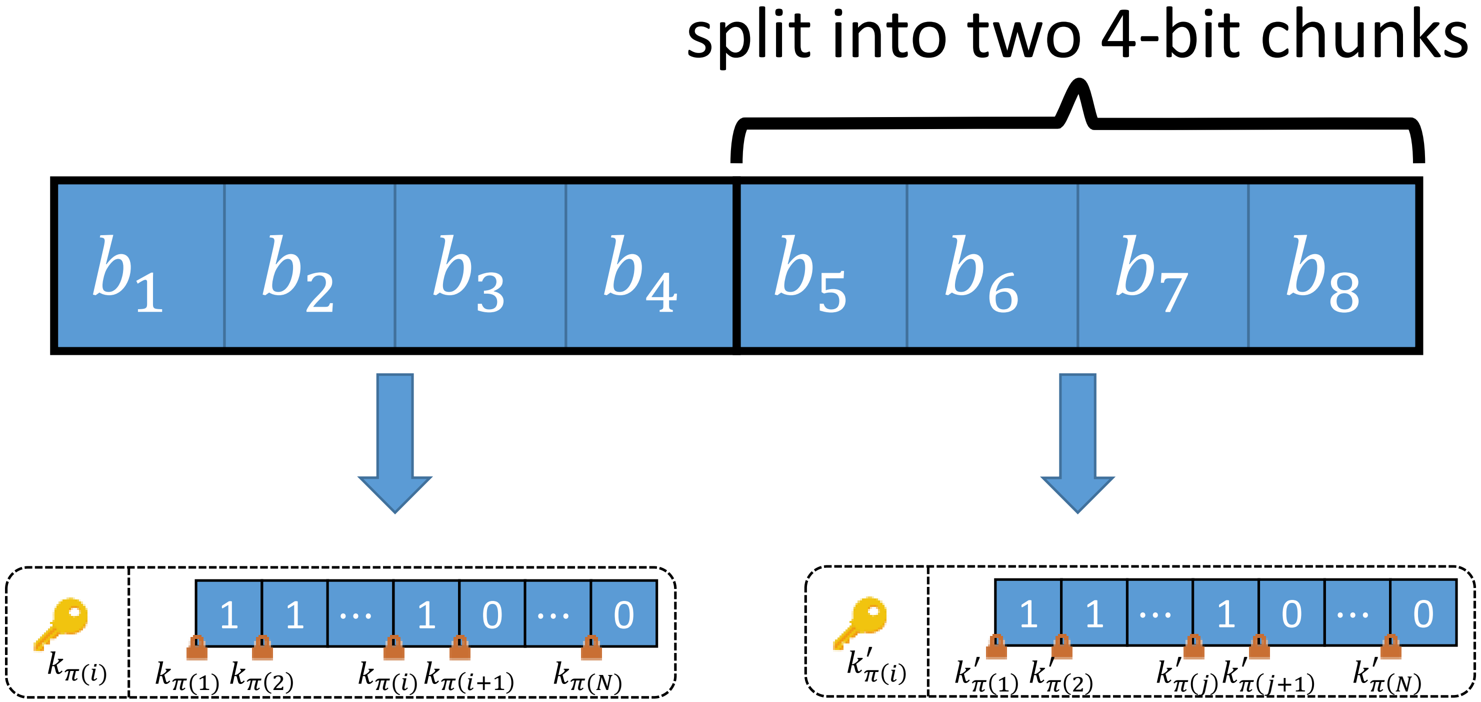
Solution: apply random permutation π (part of the secret key) to the slots



Achieves best-possible security, but ciphertexts are big

Domain Extension for ORE

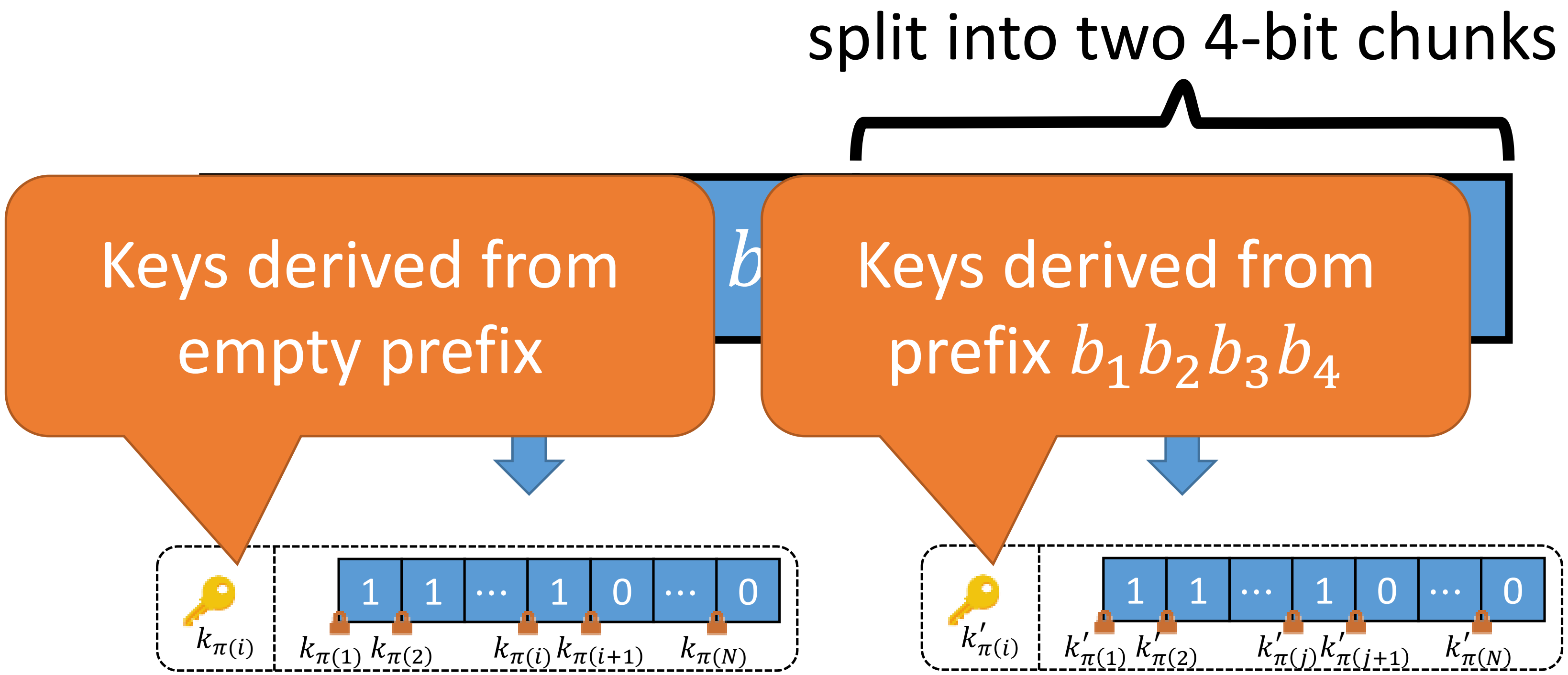
Key idea: decompose message into smaller blocks and apply small-domain ORE to each block



encrypt each chunk using an ORE instance with a secret key derived from the *prefix*

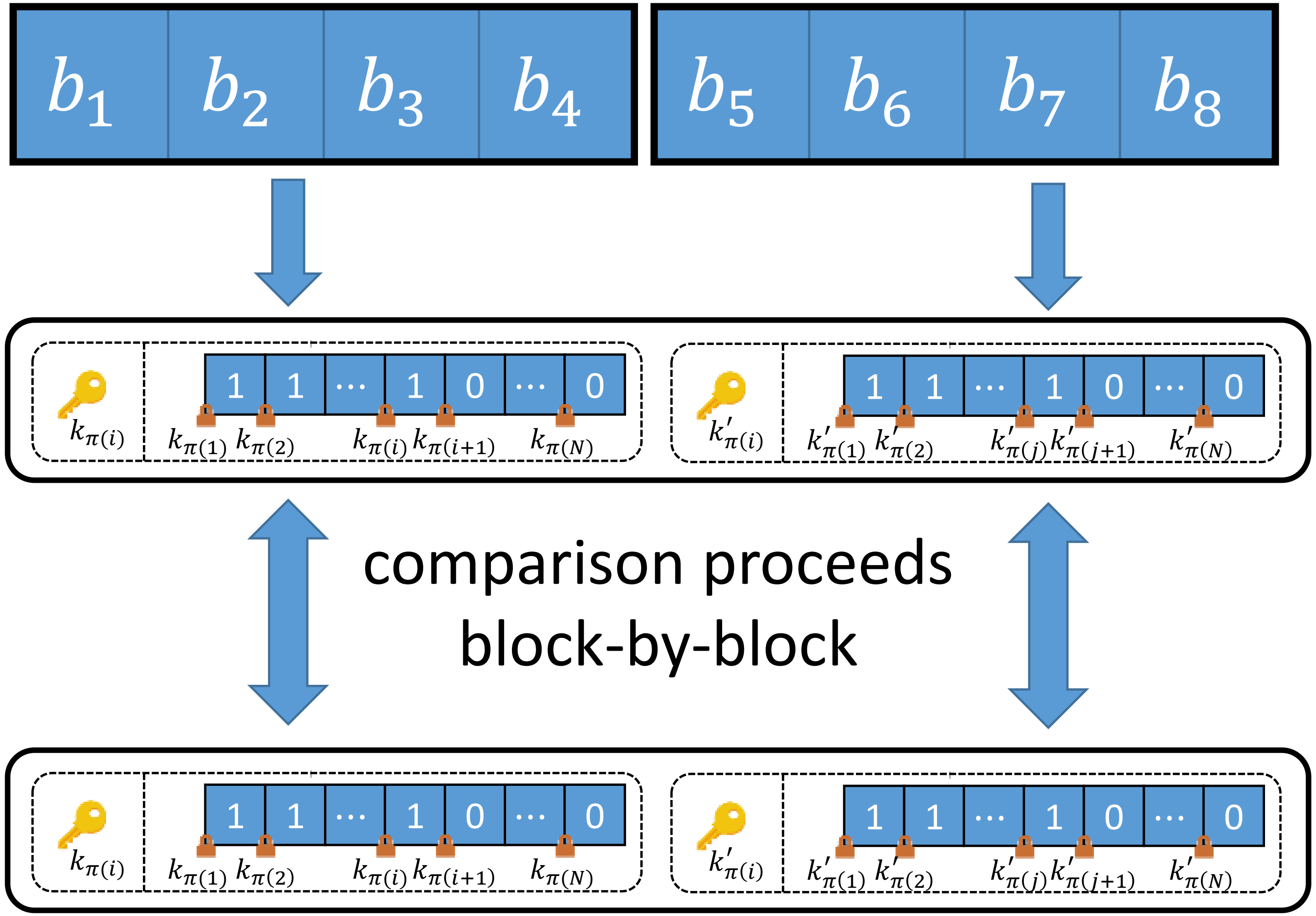
Domain Extension for ORE

Key idea: decompose message into smaller blocks and apply small-domain ORE to each block



encrypt each chunk using an ORE instance with a secret key derived from the *prefix*

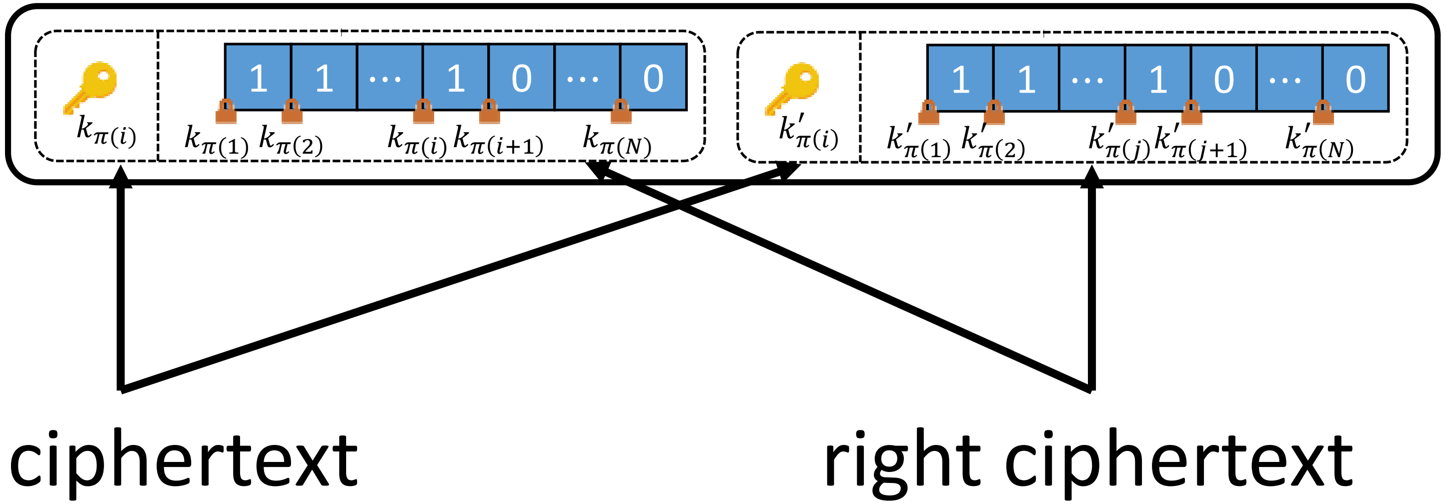
Domain Extension for ORE



Overall leakage: first **block** that differs

Domain Extension for ORE

Same decomposition into left and right ciphertexts:



Right ciphertexts provide semantic security!

Note: optimizations are possible if we apply this technique in a non-black-box way to the small-domain ORE. See paper for details.

Performance Evaluation

Scheme	Encrypt (µs)	Compare (µs)	|ct| (bytes)
OPE [BCLO'09]	3601.82	0.36	8
Practical ORE [CLWW'16]	2.06	0.48	8
This work (4-bit blocks)	16.50	0.31	192
This work (8-bit blocks)	54.87	0.63	224
This work (12-bit blocks)	721.37	2.61	1612

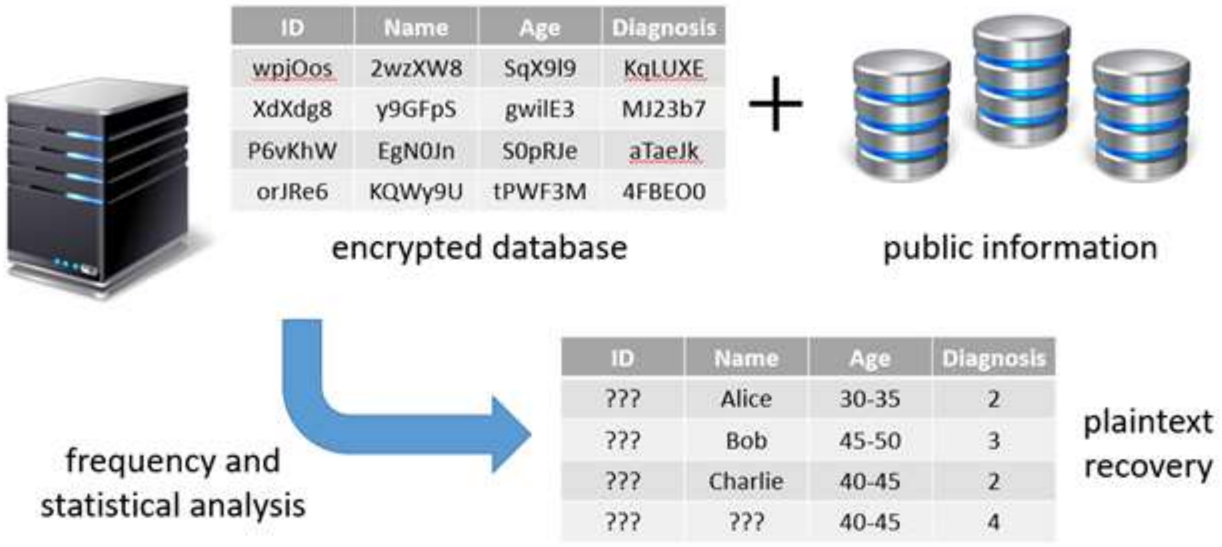
Benchmarks taken for C implementation of different schemes (with AES-NI). Measurements for encrypting 32-bit integers.

Performance Evaluation

Scheme	Encrypt (μs)	Compare (μs)	|ct| (bytes)
OPE [BCLO'09]	3601.82	0.36	8
Practical ORE [CLWW'16]	2.06	0.48	8
This work (4-bit blocks)	16.50	0.31	192
This work (8-bit blocks)	54.87	0.63	224
This work (12-bit blocks)	721.37	2.61	1612

Encrypting byte-size blocks is 65x faster than OPE, but ciphertexts are 30x longer. Security is substantially better.

Conclusions



- Inference attacks render most conventional PPE-based constructions insecure
- However, ORE is still a useful building block for encrypted databases

- Introduced new paradigm for constructing ORE that enables range queries in a way that is mostly <u>legacy-compatible</u> and provides <u>offline semantic security</u>
- New ORE construction that is concretely efficient with strong security

Questions?

Paper: `https://eprint.iacr.org/2016/612`
Website: `https://crypto.stanford.edu/ore/`
Code: `https://github.com/kevinlewi/fastore`