Order-Revealing Encryption: Definitions, Constructions, and Challenges

David Wu

Searching on Encrypted Data

Entries	Database	Detected Hashing Algorithms	Category	Dump Date
301,086,279	Twitter.com (Scraped Emails)	no passwords	Scraped Data	2017-12
288,584,667	NetEase (126.com & 163.com)	plaintext & MD5	Technology	2015-10
153,802,030	Harvested Marketing Data	no passwords	Marketing	2015
153,004,874	Adobe.com	3DES - ECB	Software	2013-10
143,090,412	North American Numbering Plan	no passwords	Government Records	2014-04
126,558,846	Badoo.com	MD5	Dating & Social Media	
121,385,316	Onliner Spambot Email Addresses	no passwords	Miscellaneous	
117,046,470	LinkedIn.com	SHA-1	Social Media	2012
100,544,934	VK.com	plaintext	Social Media	2013
99,873,194	Youku.com	MD5	Entertainment	2016

Database breaches have become the norm rather than the exception

[Data taken from Vigilante.pw]

Searching on Encrypted Data

The New York Times

Border Agency's Images of Travelers Stolen in Hack

Customs and Border Protection agency security cameras scanning license plates as vehicles cross the border from Tijuana, Mexico. John Moore/Getty Images

By Zolan Kanno-Youngs and David E. Sanger

June 10, 2019 **2 days ago!**

WASHINGTON — Tens of thousands of images of travelers and license plates stored by the Customs and Border Protection agency have been stolen in a digital breach, officials said Monday, prompting renewed questions about how the federal government secures and shares personal data.

Database breaches have become the norm rather than the exception

Why Not Encrypt?

The New York Times

Border Agency's Images of Travelers Stolen in Hack

Customs and Border Protection agency security cameras scanning license plates as vehicles cross the border from Tijuana, Mexico. John Moore/Getty Images

By Zolan Kanno-Youngs and David E. Sanger

June 10, 2019 **2 days ago!**

WASHINGTON — Tens of thousands of images of travelers and license plates stored by the Customs and Border Protection agency have been stolen in a digital breach, officials said Monday, prompting renewed questions about how the federal government secures and shares personal data.

Database breaches have become the norm rather than the exception

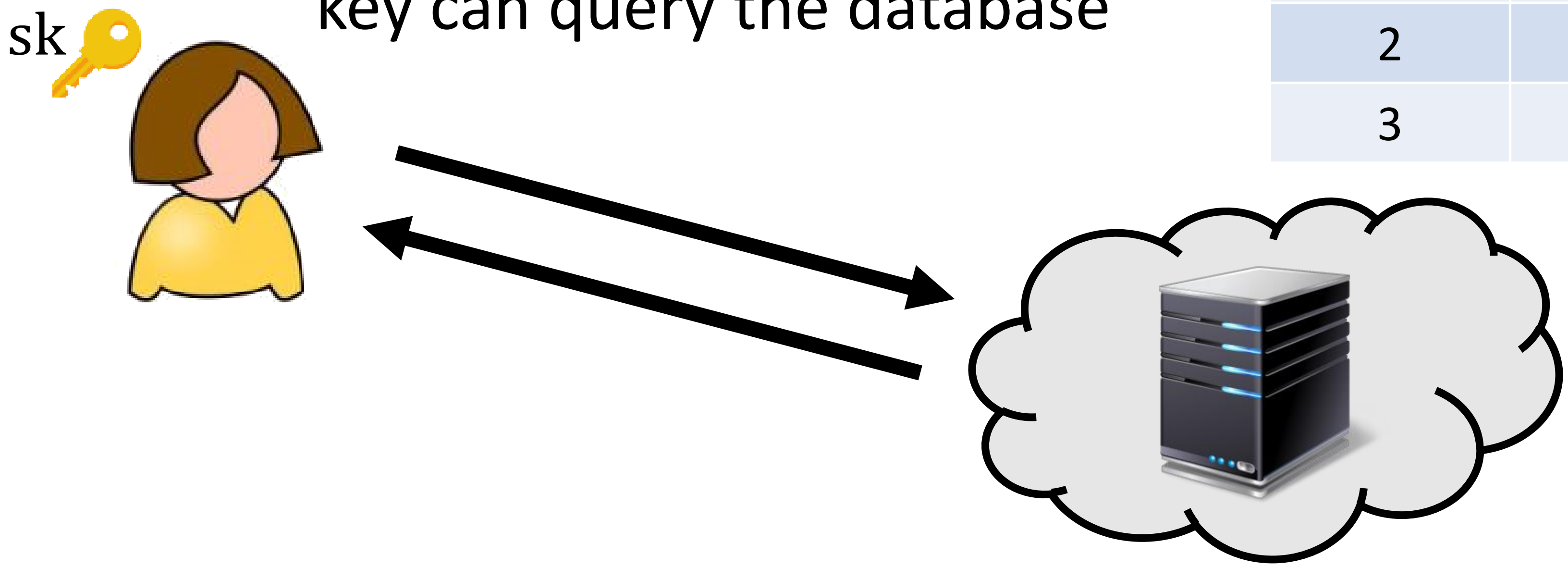
"Because it would have hurt Yahoo's ability to index and search messages to provide new user services"

– Jeff Bonforte (Yahoo SVP)

Searching on Encrypted Data

Any client (e.g., web client, employee) who hold a secret key can query the database

ID	Name	Age	Zip Code
0	Alice	31	68107
1	Bob	47	60015
2	Emily	41	38655
3	Jeff	45	46304



encrypted database

Can we construct an <u>encryption scheme</u> that still supports searching over encrypted data?

Searching on Encrypted Data

Any client (e.g., web client, employee) who hold a secret key can query the database

ID	Name	Age	Zip Code
0	Alice	31	68107
1	Bob	47	60015
2	Emily	41	38655
3	Jeff	45	46304

encrypted database

This talk: focus will be on *range queries*

Can we construct an encryption scheme that still supports searching over encrypted data?

Order-Preserving Encryption (OPE)

[BCLO09, BCO11]

Secret-key encryption scheme

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

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

$$x \geq y \iff \mathrm{ct}_x \geq \mathrm{ct}_y$$

Impose additional structural requirement on ciphertexts: ciphertexts themselves preserve the ordering

Searching on Encrypted Data

ID	Name	Age	Zip Code
0	Alice	31	68107
1	Bob	47	60015
2	Emily	41	38655
3	Jeff	45	46304

ID	Name	Age	Zip Code
0	Alice	31	68107
1	Bob	47	60015
2	Emily	41	38655
3	Jeff	45	46304

Encrypt each column with an OPE scheme (with different keys)

Encrypted values preserve the ordering, so server can still sort and perform range queries on encrypted values

Defining Security

Starting point: Semantic security (IND-CPA)

$b \in \{0,1\}$

sk

$m_0^{(i)}, m_1^{(i)} \in \mathcal{M}$

$\mathrm{Enc}\left(\mathrm{sk}, m_b^{(i)}\right)$

Challenger

Adversary

Semantic security: Adversary cannot guess b
(except with probability negligibly close to 1/2)

Best-Possible Security for OPE

[BCLO09, BCO11]

$b \in \{0,1\}$

sk

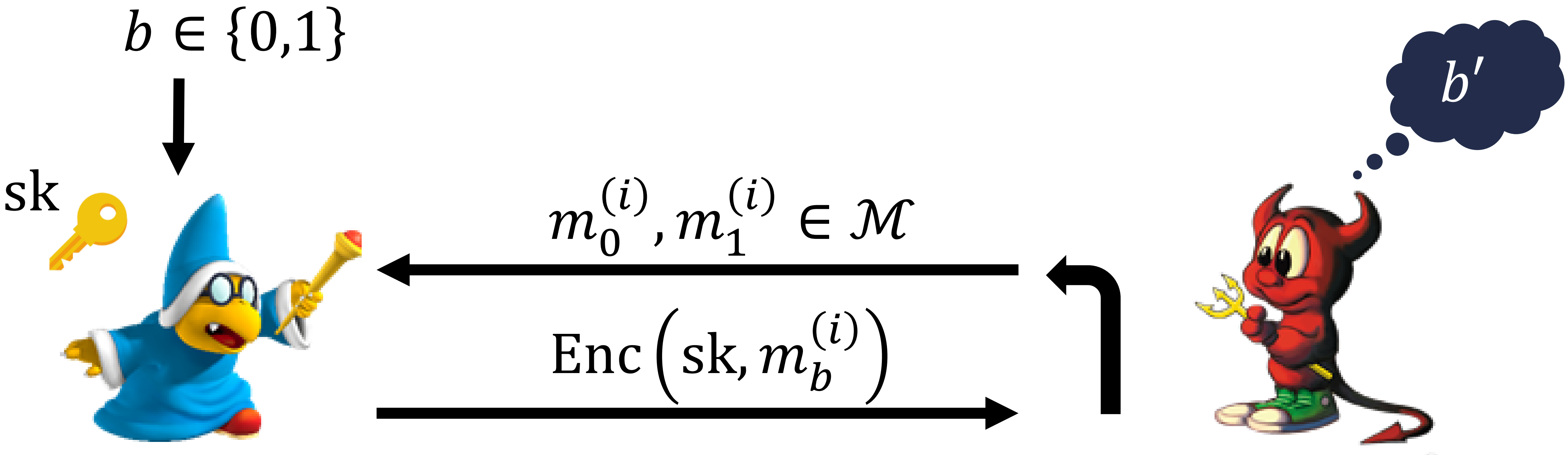
$m_0^{(i)}, m_1^{(i)} \in \mathcal{M}$

$\mathrm{Enc}\left(\mathrm{sk}, m_b^{(i)}\right)$

Must impose restriction on messages: otherwise trivial to break semantic security using comparison operator

Best-Possible Security for OPE

[BCLO09, BCO11]



$$\forall i, j: \quad m_0^{(i)} < m_0^{(j)} \Leftrightarrow m_1^{(i)} < m_1^{(j)}$$

Best-Possible Security for OPE

[BCLO09, BCO11]

$b \in \{0,1\}$

sk

$m_0^{(i)}, m_1^{(i)} \in \mathcal{M}$

$\mathrm{Enc}\left(\mathrm{sk}, m_b^{(i)}\right)$

b'

Order of “left” set of messages same as order of “right” set of messages

Best-Possible Security for OPE

[BCLO09, BCO11]

Best-possible notion of security is difficult to achieve for OPE

- **[BCLO09]:** If message space is $[M]$ and ciphertext space is $[N]$, then best-possible security requires $N > 2^{\Omega(M)}$

  ciphertext <u>length</u> scales linearly in the <u>size</u> of plaintext space

- **[LW16]:** If message space is $[M]$ for $M > 3$ and ciphertext space is $[N]$, then best-possible security requires $N > 2^{2^{\omega(\log \lambda)}}$

  ciphertext <u>length</u> is super-polynomial in security parameter

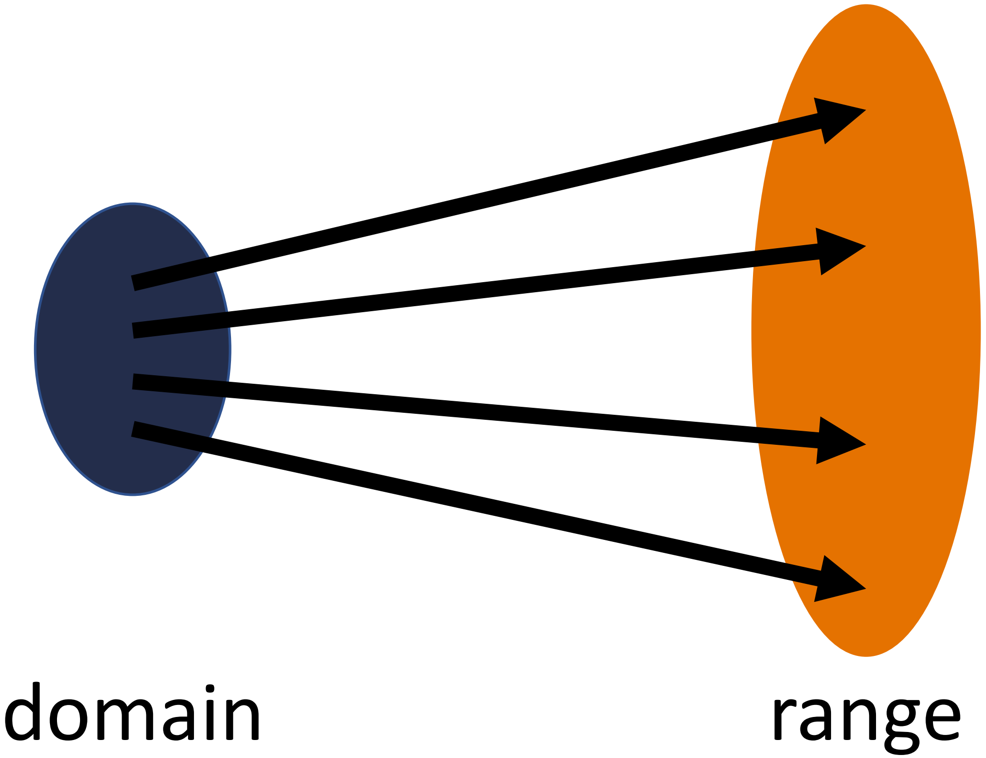
Both lower bounds exploit the fact that ciphertexts preserve the natural ordering over the integers

Alternative Security Definitions

Order-preserving encryption (OPE) [BCLO09, BCO11]:

- No "best-possible" security, so instead, compare with <u>random</u> order-preserving function (ROPF)

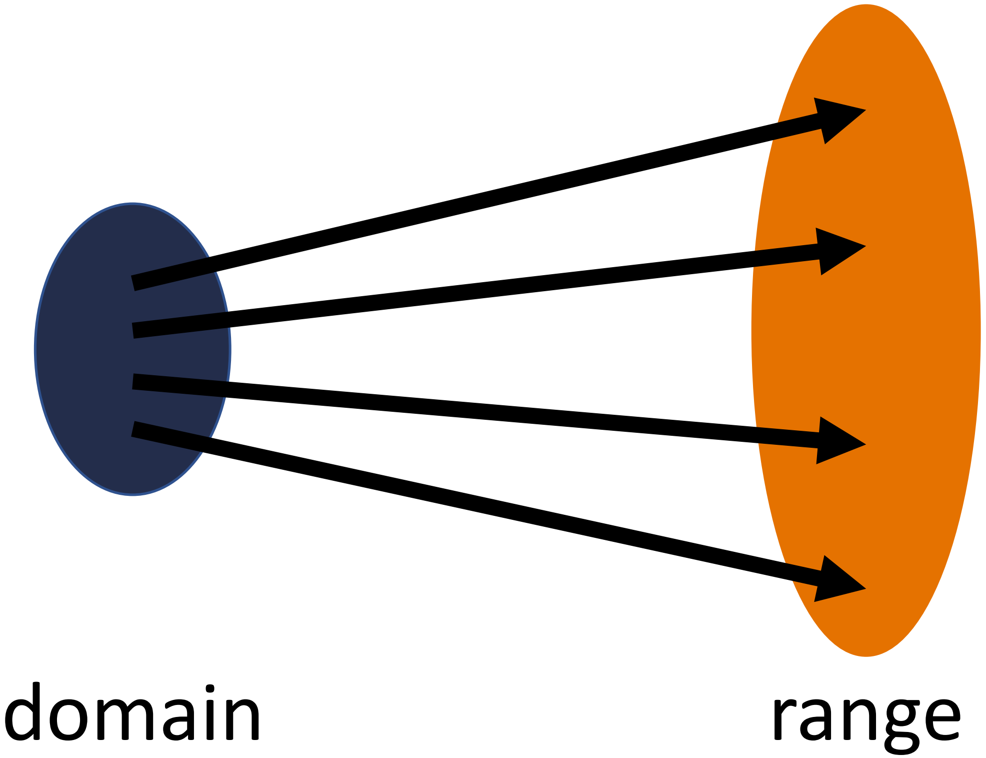
Encryption function implements a <u>random</u> order-preserving function



Alternative Security Definitions

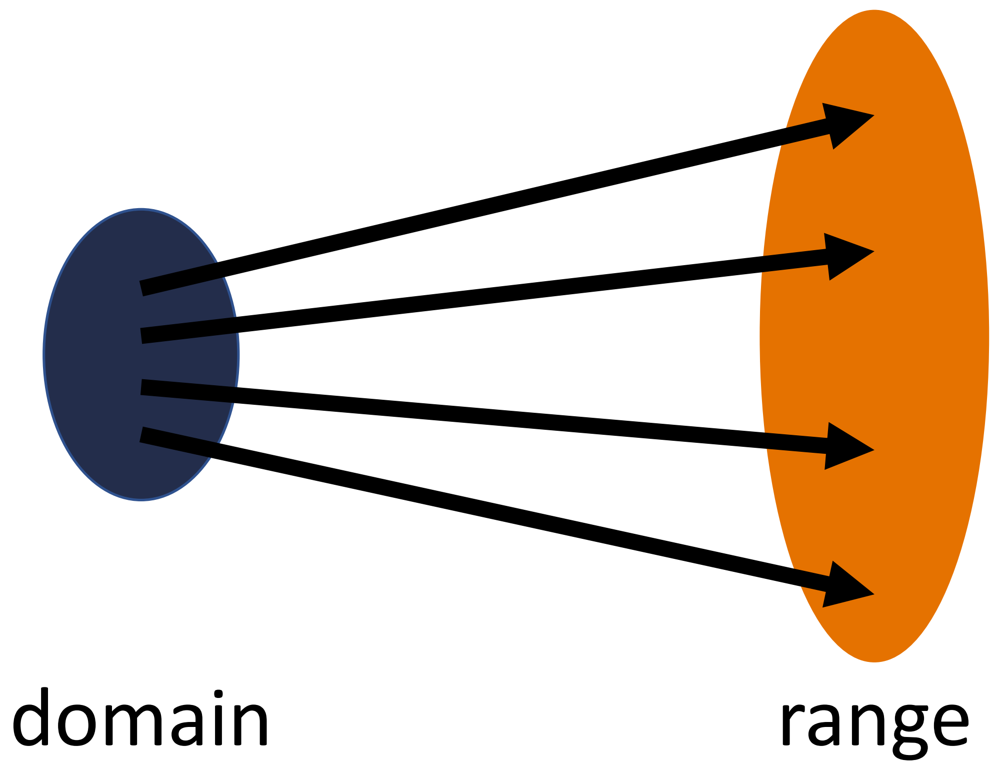
ROPF is an “ideal” order-preserving primitive – security definition similar in flavor to PRF security

Encryption function implements a <u>random</u> order-preserving function



OPE Security

[BCLO09, BCO11]



Advantage: Meaningful security definition that admits efficient constructions (based on just PRFs)

Disadvantage: Difficult to completely characterize what is hidden by a random order-preserving function

- Each ciphertext roughly reveals half of the most significant bits
- Each pair of ciphertexts roughly reveals half of the most significant bits of their difference

Big gap compared to best-possible security!

Order-Revealing Encryption (ORE)

(also called *efficiently orderable encryption*) [BCO11, BLRSZZ15]

Lower bounds on best-possible security leverage the fact that ciphertexts preserve the natural ordering over the integers

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

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

$x > y$

Public comparison function for ciphertexts

Insight: Allow ciphertexts to have arbitrary structure and just require a "comparison" function (e.g., functional encryption)

Order-Revealing Encryption (ORE)

(also called *efficiently orderable encryption*) [BCO11, BLRSZZ15]

Lower bounds on best-possible security leverage the fact that ciphertexts preserve the natural ordering over the integers

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

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

$x > y$

Public comparison function for ciphertexts

Server can still use public comparison function to compare ciphertexts and support range queries

Order-Revealing Encryption (ORE)

(also called *efficiently orderable encryption*) [BCO11, BLRSZZ15]

Lower bounds on best-possible security leverage the fact that ciphertexts preserve the natural ordering over the integers

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

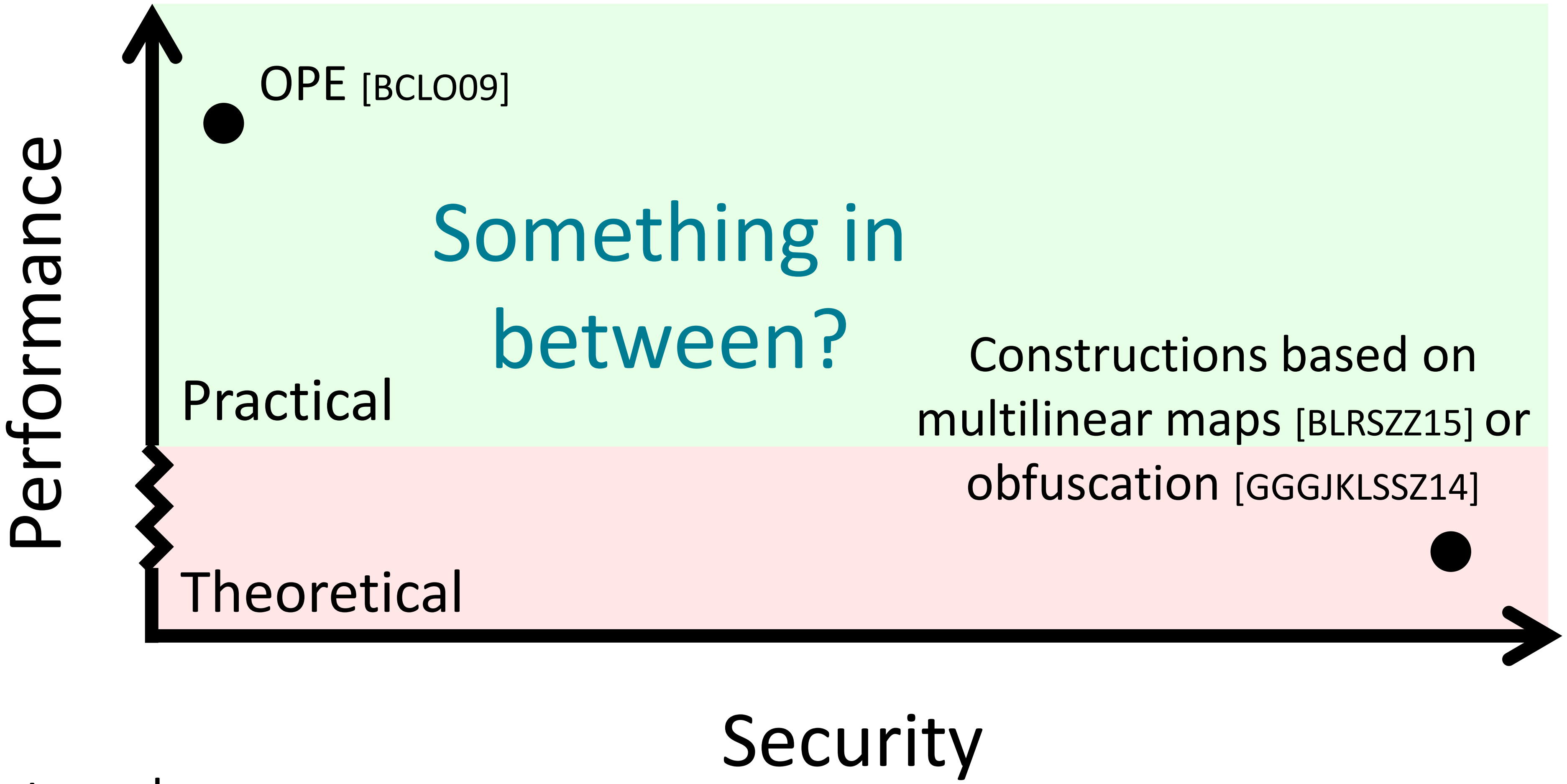
$\mathrm{ct}_2 = \mathrm{Enc}(\mathrm{sk}, y)$

$x > y$

Possible to achieve best-possible security, but constructions rely on multilinear maps or obfuscation…

Server can still use public comparison function to compare ciphertexts and support range queries

The Landscape of ORE

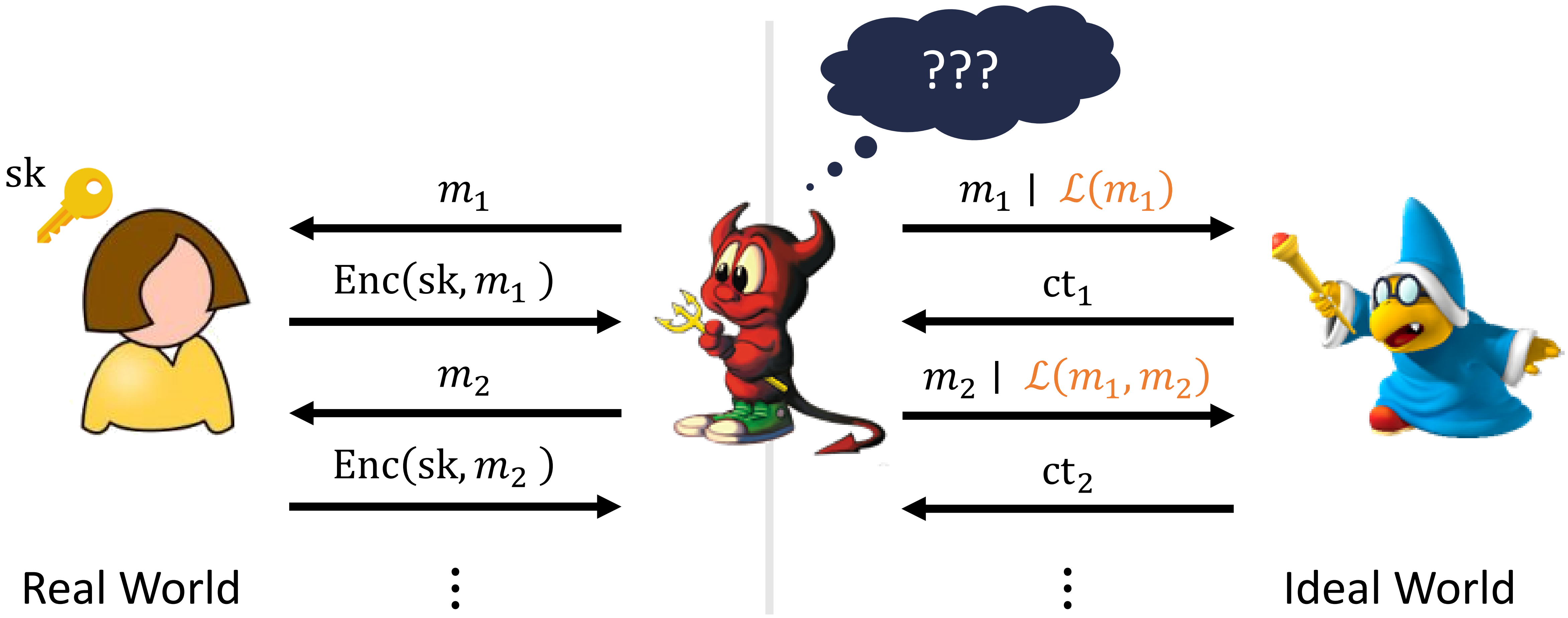


Not drawn to scale

A New Security Notion: SIM-ORE

[CLWW16]

Idea: Augment "best-possible" security with a leakage function $\mathcal{L}$



A New Security Notion: SIM-ORE

[CLWW16]

Idea: Augment "best-possible" security with a leakage function $\mathcal{L}$

Similar to SSE definitions [CGKO06, CK10]

Leakage function specifies <u>exactly</u> what is leaked by the encryption scheme

Real World ⋮ ⋮ Ideal World

A Simple ORE Construction

[CLWW16]

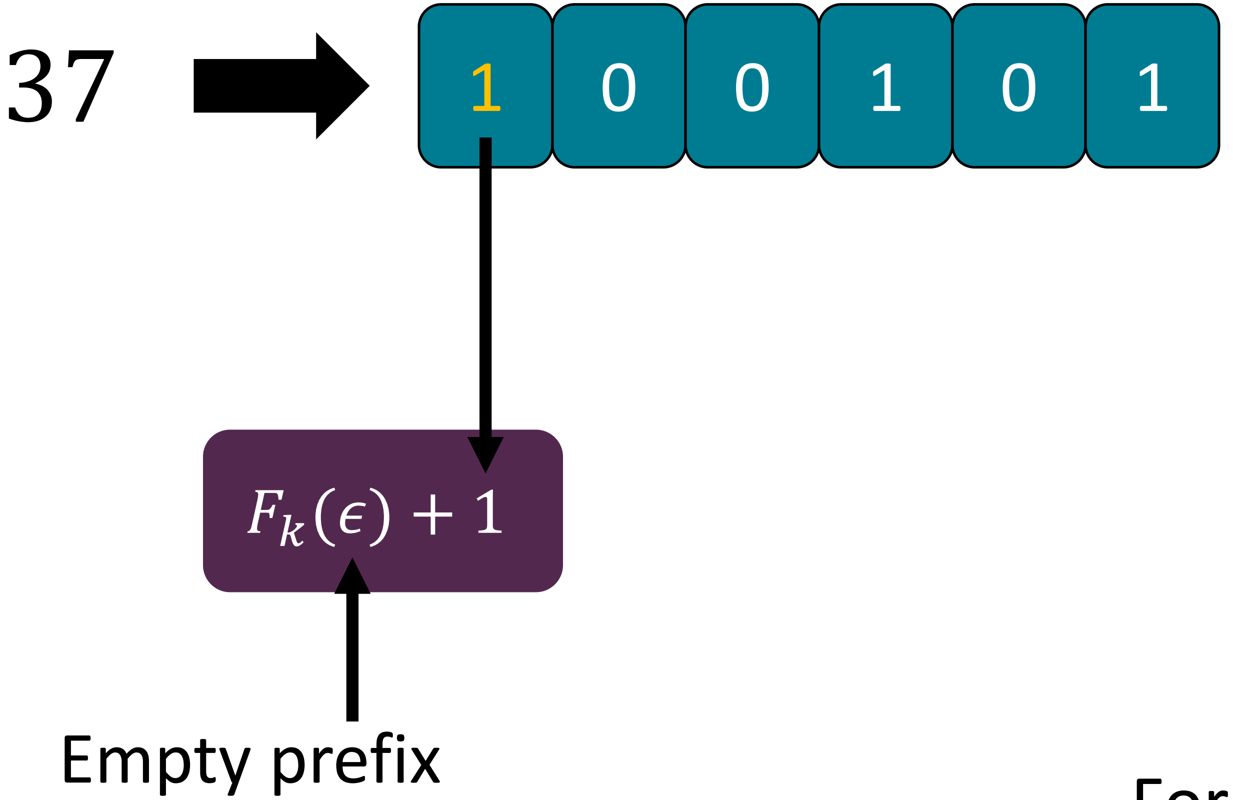
$37 \rightarrow$ | 1 | 0 | 0 | 1 | 0 | 1 |

For each index i, apply a PRF (e.g., AES) to the first $i-1$ bits, then add $b_i \pmod 3$

$F_k\colon \{0,1\}^* \rightarrow \{0,1,2\}$

A Simple ORE Construction

[CLWW16]

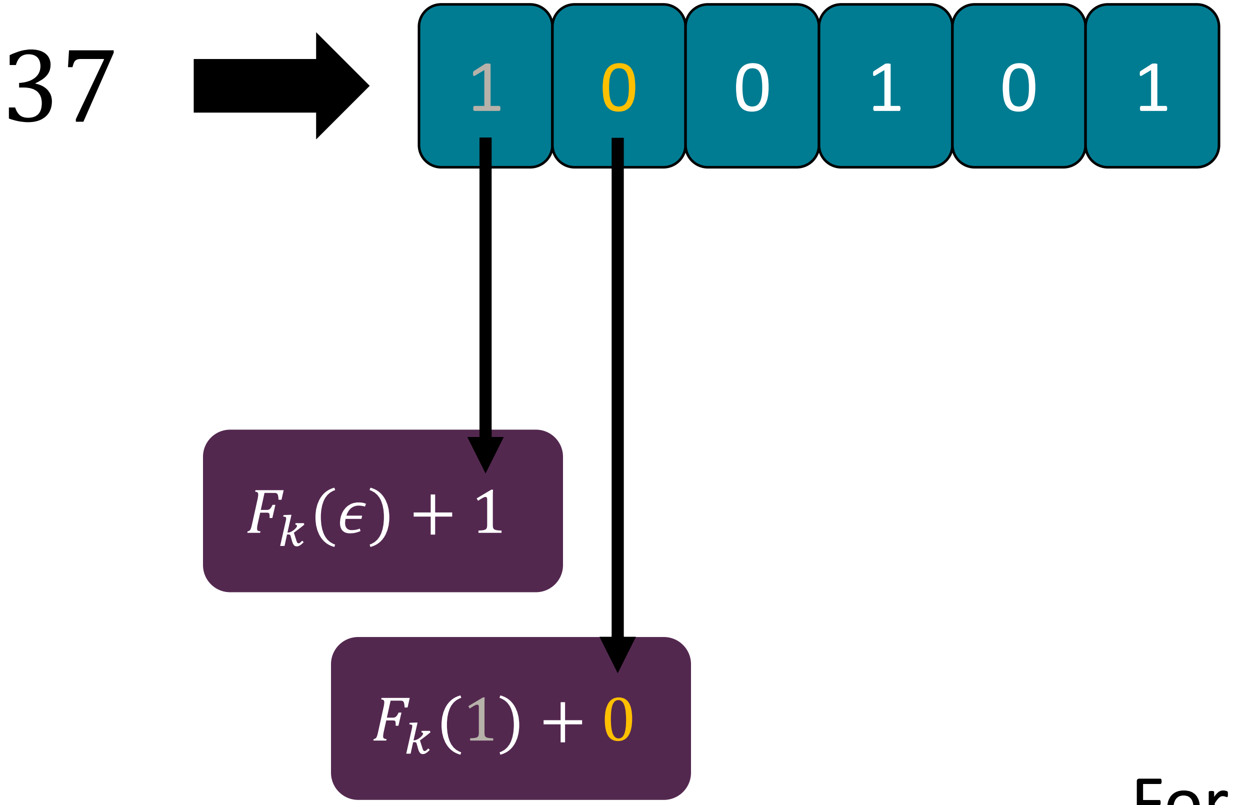


For each index i, apply a PRF (e.g., AES) to the first $i - 1$ bits, then add $b_i \pmod 3$

$F_k\colon \{0,1\}^* \to \{0,1,2\}$

A Simple ORE Construction

[CLWW16]

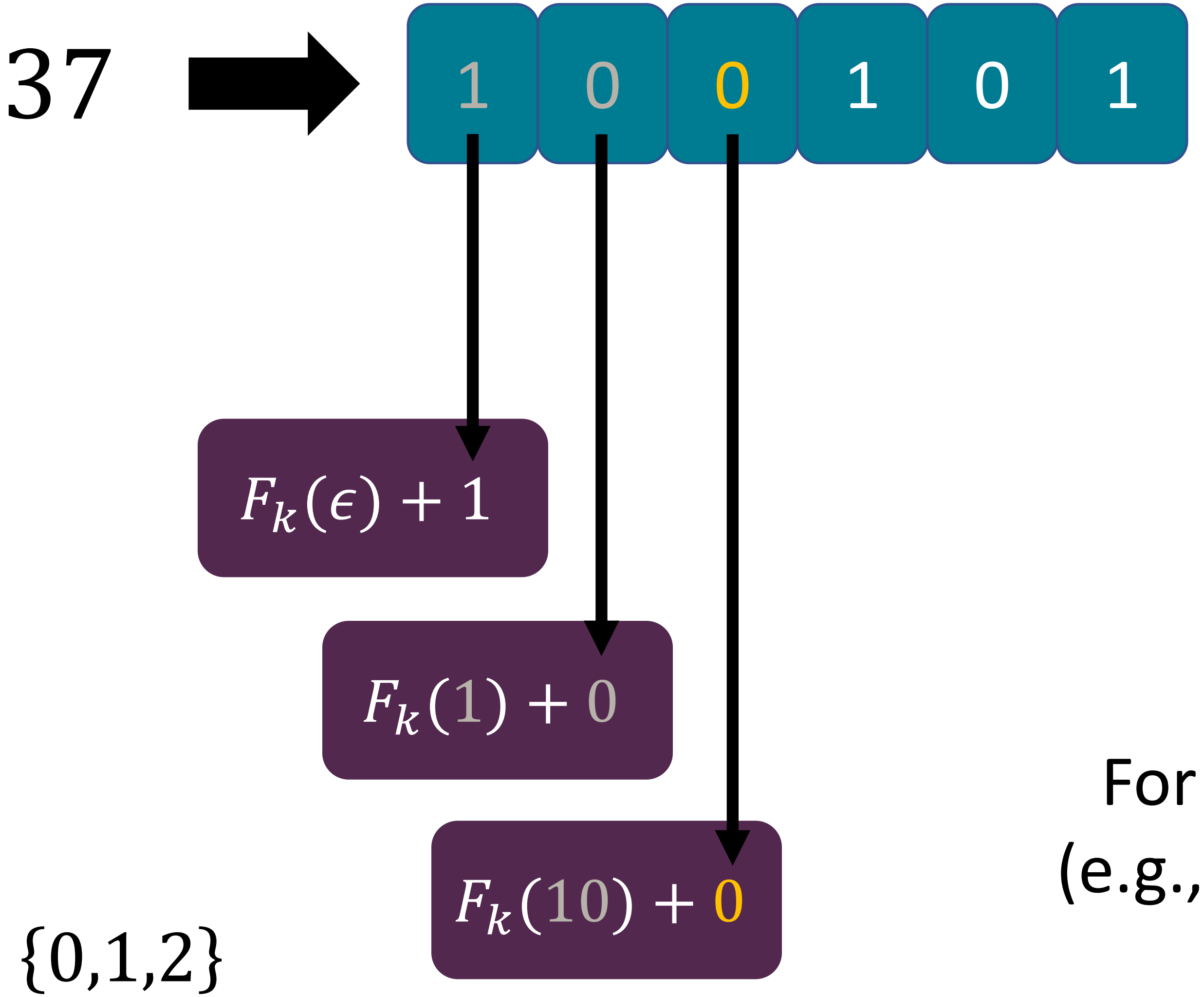


For each index i, apply a PRF (e.g., AES) to the first $i - 1$ bits, then add $b_i \pmod 3$

$F_k\colon \{0,1\}^* \to \{0,1,2\}$

A Simple ORE Construction

[CLWW16]

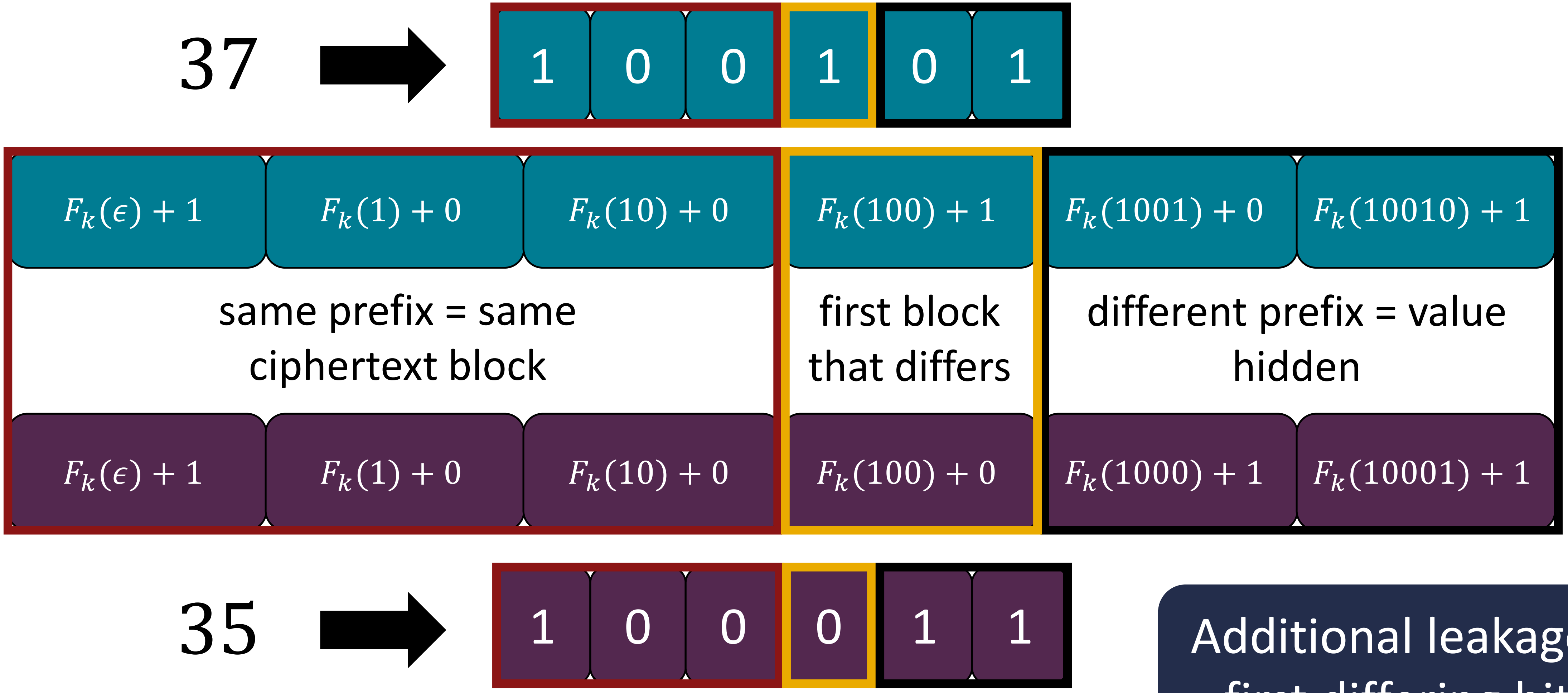


$F_k\colon \{0,1\}^* \to \{0,1,2\}$

For each index i, apply a PRF (e.g., AES) to the first $i-1$ bits, then add $b_i \pmod 3$

A Simple ORE Construction

[CLWW16]



Additional leakage: first differing bit

Recall: All additions happen modulo 3

A Simple ORE Construction

[CLWW16]

37 → 1 0 0 1 0 1

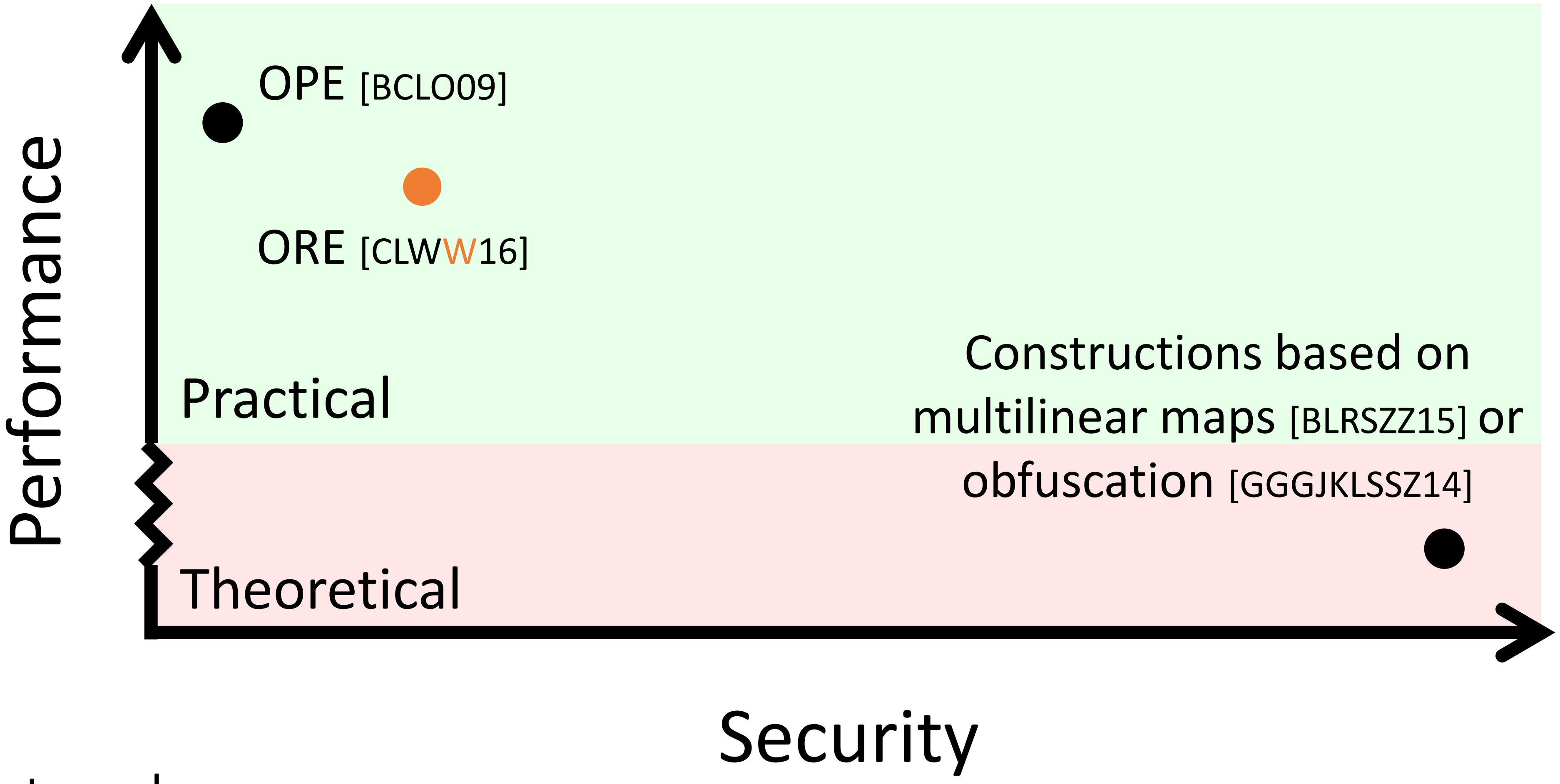
$F_k(\epsilon) + 1$	$F_k(1) + 0$	$F_k(10) + 0$	$F_k(100) + 1$	$F_k(1001) + 0$	$F_k(10010) + 1$
same prefix = same ciphertext block			first block that differs	different prefix = value hidden	
$F_k(\epsilon) + 1$	$F_k(1) + 0$	$F_k(10) + 0$	$F_k(100) + 0$	$F_k(1000) + 1$	$F_k(10001) + 1$

35 → 1 0 0 0 1 1

Additional leakage: first differing bit

Key insight: Embed comparisons into $\mathbb{Z}_3$

The Landscape of ORE



Not drawn to scale

Inference Attacks and Database Reconstruction

[NKW15, DDC16, KKNO16, GSBNR17, LMP18, GLMP19]

ID	Name	Age	Zip Code
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	Zip Code
???	Alice	30-35	68???
???	Bob	45-50	60???
???	Emily	40-45	38???
???	Jeff	40-45	46???

Plaintext recovery

Inference Attacks and Database Reconstruction

[NKW15, DDC16, KKNO16, GSBNR17, LMP18, GLMP19]

ID	Name	Age	Zip Code
wpjOos	2wzXW8	SqX9I9	KqLUXE
XdXdg8	y9GFpS	gwilE3	MJ23b7
P6vKhW	EgN0Jn	S0pRJe	aTaeJk
orJRe6	KQWy9U	tPWF3M	4FBEO0

Encrypted database

+

Public information

Frequency and statistical analysis

ID	Name	Age	Zip Code
???	Alice	30-35	68???
???	Bob	45-50	60???
???	Emily	40-45	38???
???	Jeff	40-45	46???

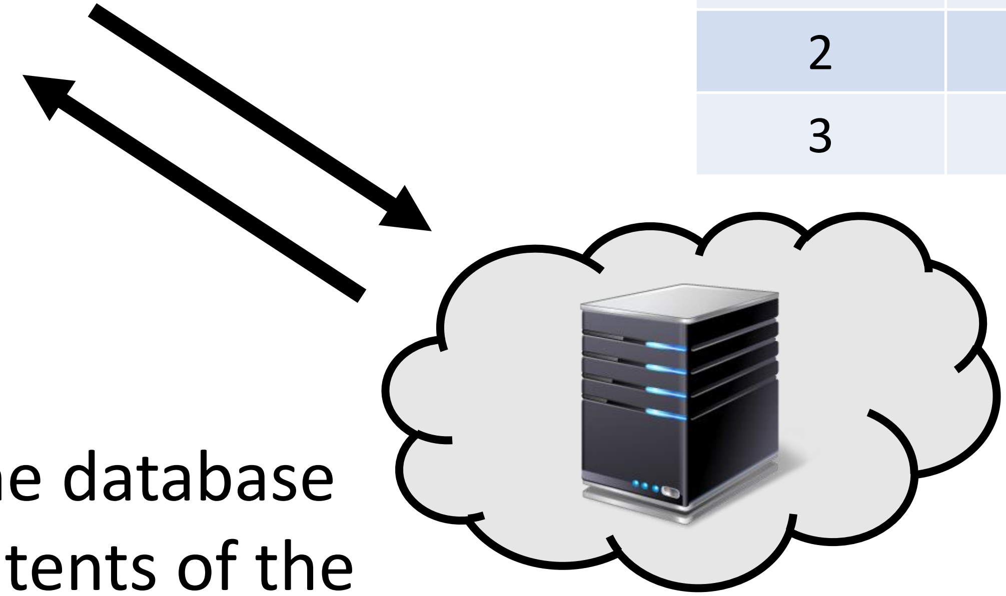
Plaintext recovery

ORE schemes reveal order of ciphertexts and thus, are vulnerable to offline inference attacks

Can we extend ORE to defend against offline inference attacks?

Snapshot Adversaries

ID	Name	Age	Zip Code
0	Alice	31	68107
1	Bob	47	60015
2	Emily	41	38655
3	Jeff	45	46304



Adversary breaks into the database server and steals the contents of the database on <u>disk</u> (i.e., obtains a "snapshot" of the database)

Database server

Snapshot Adversaries

Here, we assume the "snapshot" just contains the encrypted database contents and nothing more (e.g., no query caches, etc.)

ID	Name	Age	Zip Code
0	Alice	31	68107
1	Bob	47	60015
2	Emily	41	38655
3	Jeff	45	46304

Adversary breaks into the database server and steals the contents of the database on <u>disk</u> (i.e., obtains a "snapshot" of the database)

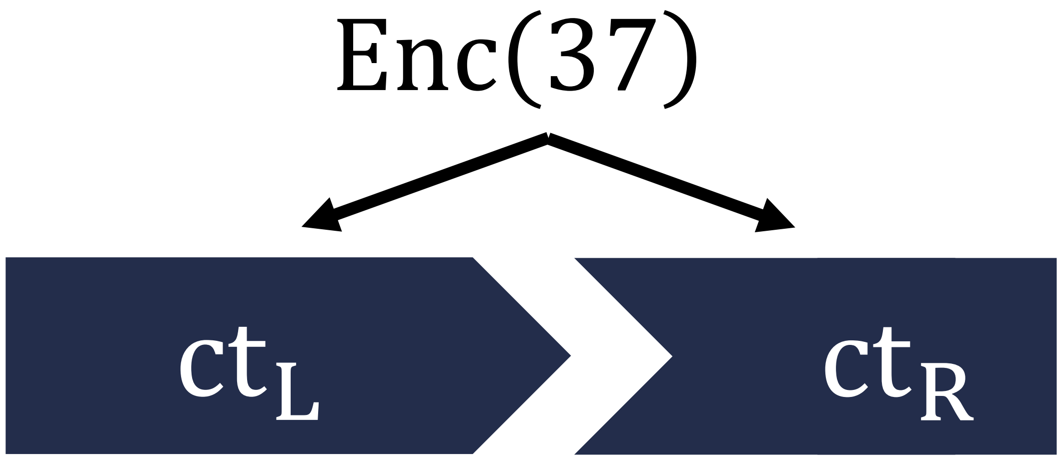
Database server

Defending Against Snapshot Adversaries

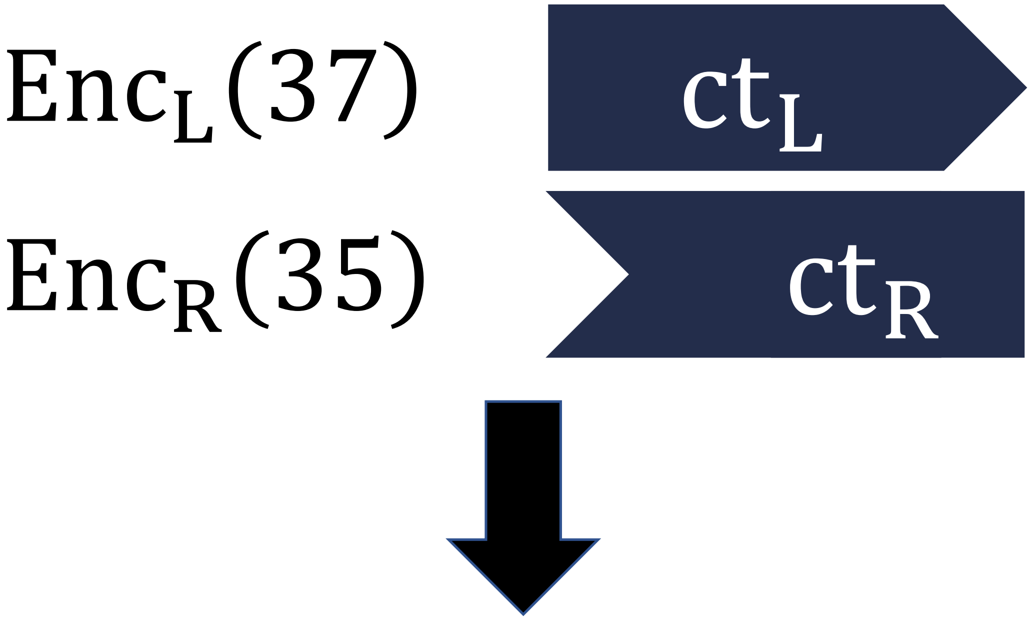
[LW16]

Approach: Require additional properties from the underlying ORE scheme

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



Ciphertexts naturally split into two components ("left-right" ORE)



Greater than

Defending Against Snapshot Adversaries

[LW16]

$\mathrm{Enc_L}(37)$ → $\mathrm{ct_L}$

$\mathrm{Enc_R}(35)$ → $\mathrm{ct_R}$

Comparison can be performed between left ciphertext and right ciphertext

Right ciphertexts reveal nothing about underlying messages!

Robustness against offline inference attacks!

But will require different protocol to implement range queries

Range Queries on Encrypted Data

[LW16]

ID	Name	Age	Zip Code
0	Alice	31	68107
1	Bob	47	60015
2	Emily	41	38655
3	Jeff	45	46304

Build encrypted index

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

Store right ciphertexts in sorted order

Record IDs encrypted under independent key

Zip Code	ID
$\mathrm{Enc}_R(38655)$	$\mathrm{Enc}(2)$
$\mathrm{Enc}_R(46304)$	$\mathrm{Enc}(3)$
$\mathrm{Enc}_R(60015)$	$\mathrm{Enc}(1)$
$\mathrm{Enc}_R(68107)$	$\mathrm{Enc}(0)$

Separate index for each searchable column, and using different ORE keys

Range Queries on Encrypted Data

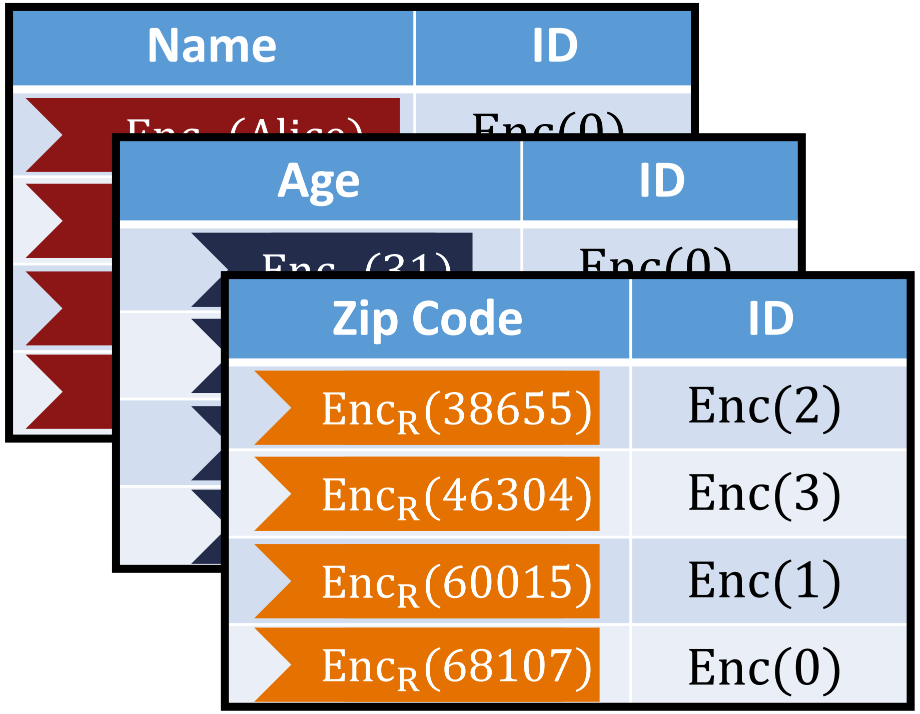
[LW16]

Encrypted database:

ID	Name	Age	Zip Code
0	Alice	31	68107
1	Bob	47	60015
2	Emily	41	38655
3	Jeff	45	46304

Columns (other than ID) are encrypted using standard encryption scheme

To perform range query, client provides <u>left ciphertexts</u> corresponding to its range

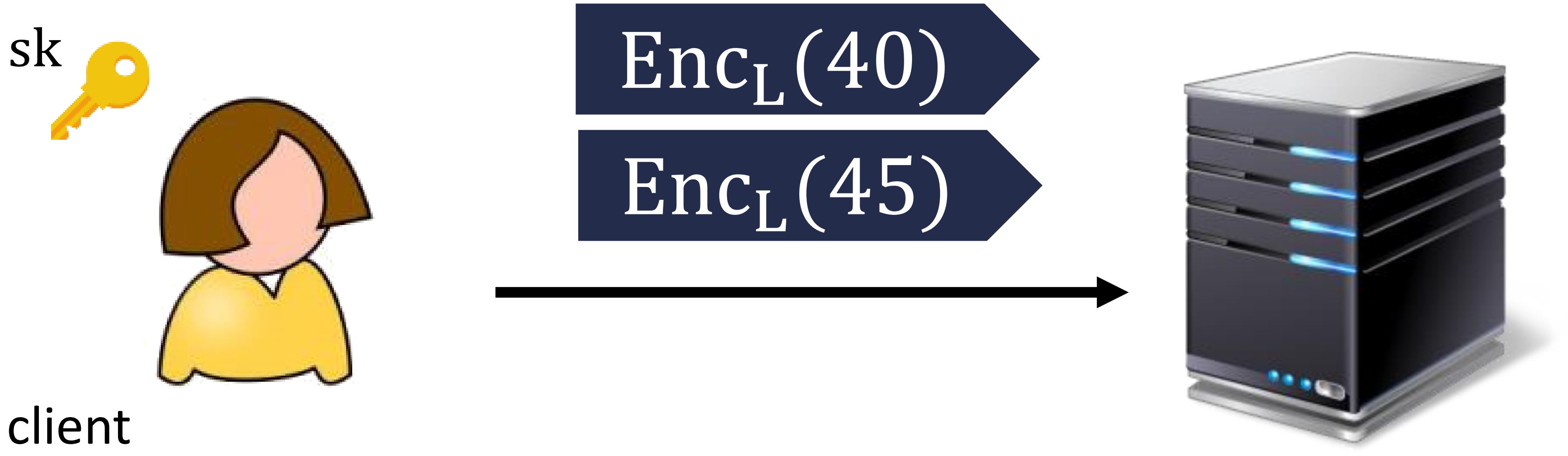


Encrypted search indices

Range Queries on Encrypted Data

[LW16]

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



Range Queries on Encrypted Data

[LW16]

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

$\text{Enc}_\text{L}(40)$

$\text{Enc}_\text{L}(45)$

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

Range Queries on Encrypted Data

[LW16]

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

$\mathrm{Enc}_{\mathrm{L}}(40)$

$\mathrm{Enc}_{\mathrm{L}}(45)$

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

Use binary search to determine endpoints (comparison via ORE)

Range Queries on Encrypted Data

[LW16]

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

$\text{Enc}_L(45)$

$\text{Enc}_L(40)$

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)

Range Queries on Encrypted Data

[LW16]

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

$\text{Enc}_\text{L}(40)$

$\text{Enc}_\text{L}(45)$

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

Return encrypted indices that match query

Use binary search to determine endpoints (comparison via ORE)

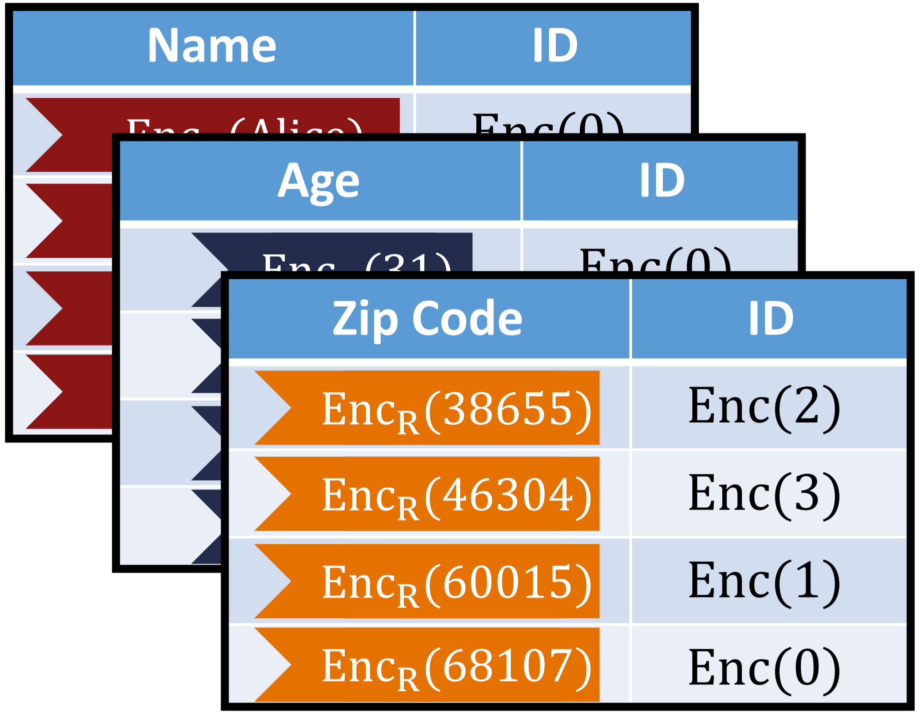
Range Queries on Encrypted Data

[LW16]

Encrypted database:

ID	Name	Age	Zip Code
0	Alice	31	68107
1	Bob	47	60015
2	Emily	41	38655
3	Jeff	45	46304

Encrypted database hides the contents!



Encrypted search indices

Left-Right ORE Construction

[LW16]

"Small-domain" ORE with best-possible security

+

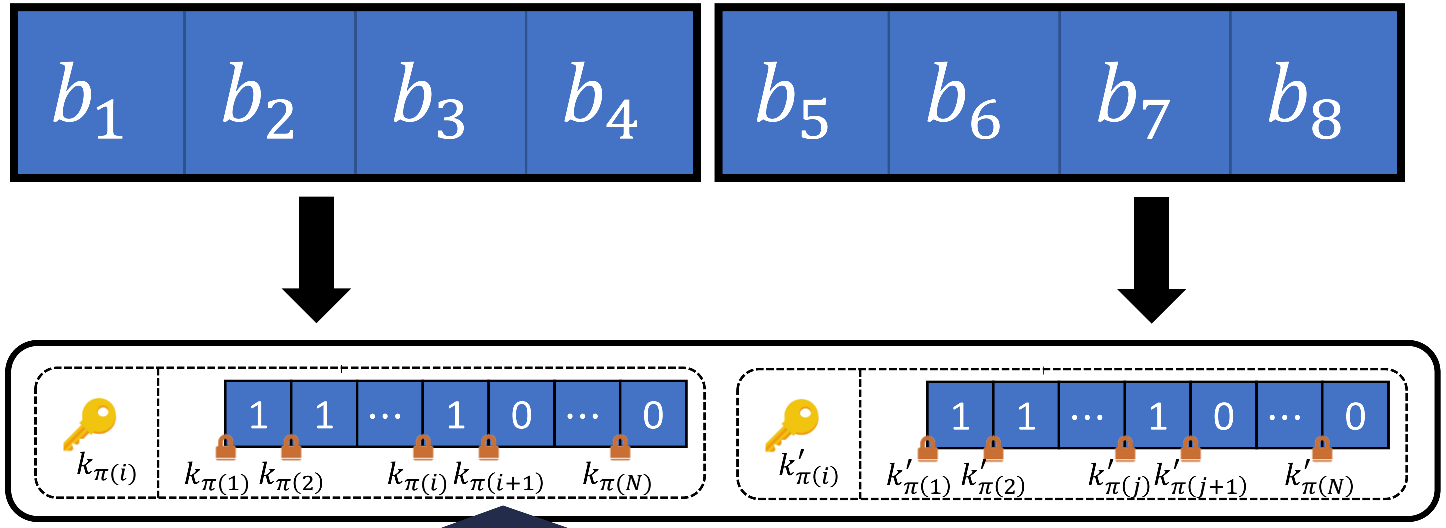
Block-by-block extension similar to previous construction

→

"Large-domain" ORE with leakage

Left-Right ORE Construction

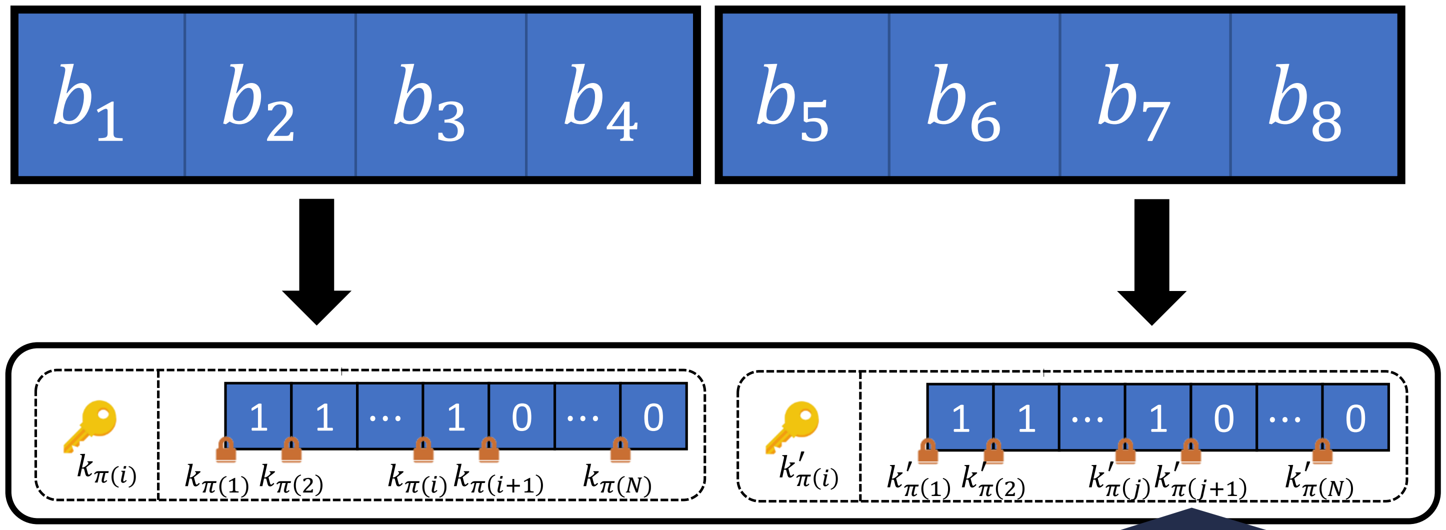
[LW16]



Small-domain left-right ORE that provides best-possible security

Left-Right ORE Construction

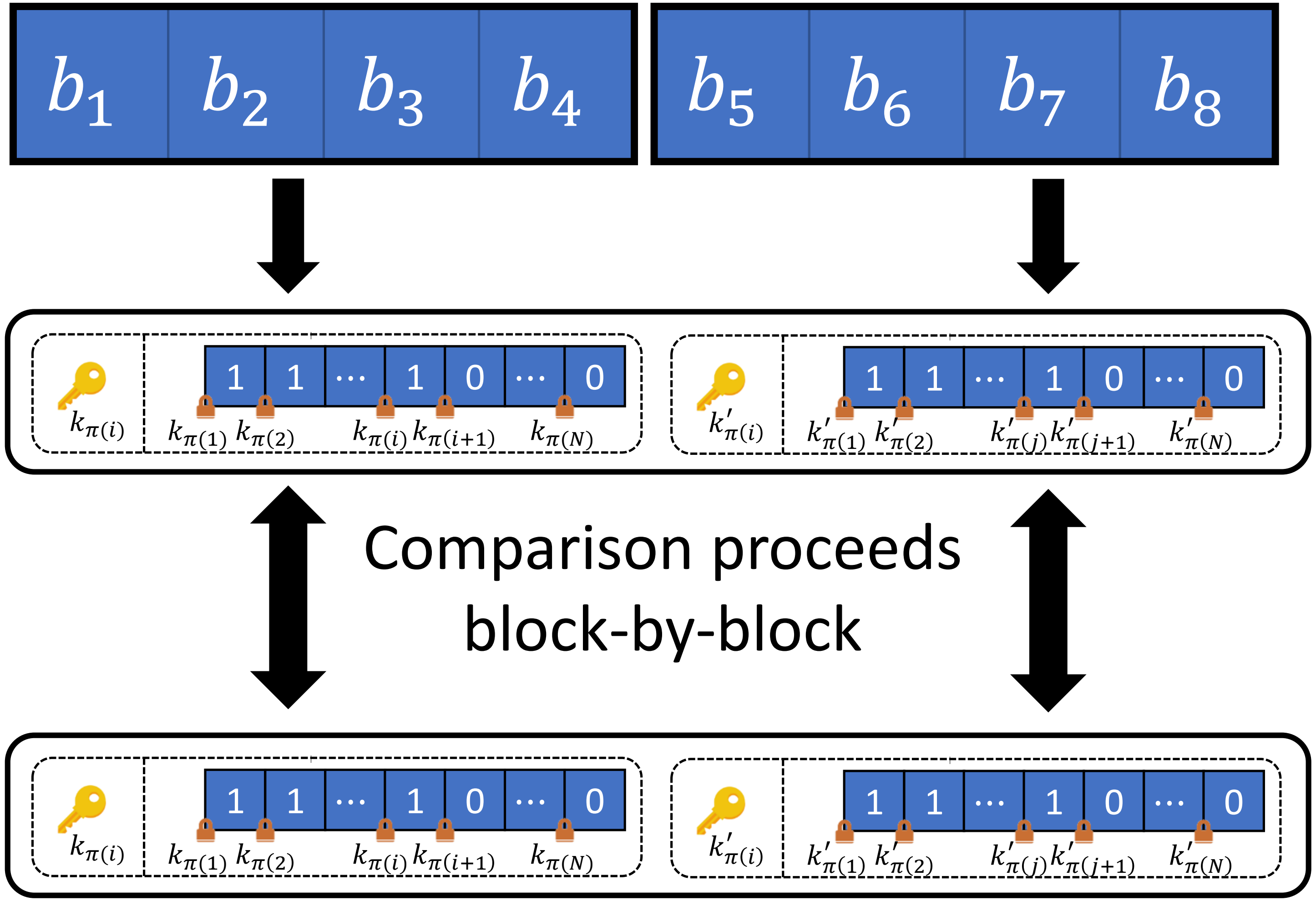
[LW16]



Each block encrypted with key derived from prefix (domain extension)

Left-Right ORE Construction

[LW16]

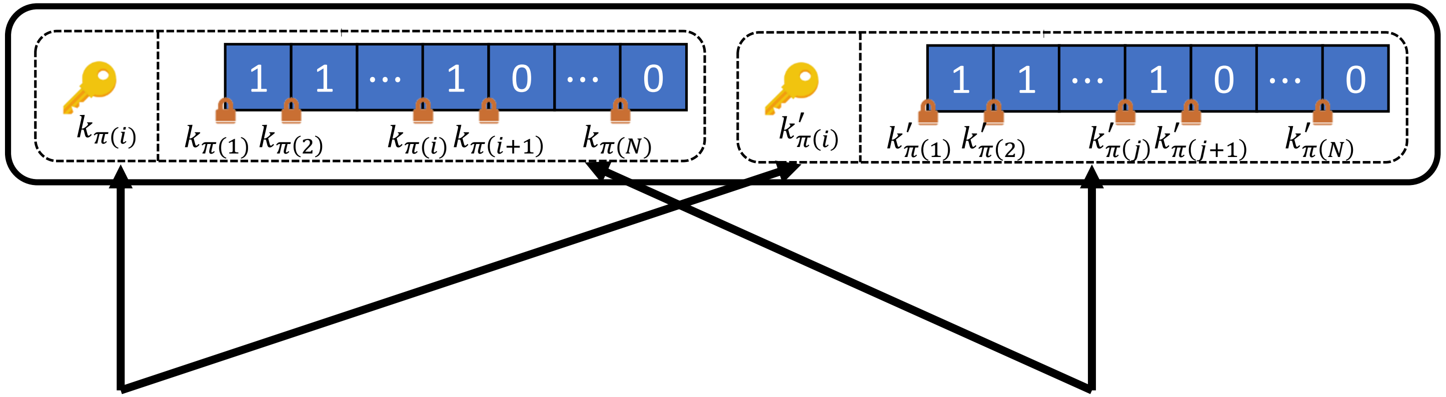


Overall leakage: First **block** that differs

Domain Extension for ORE

[LW16]

Same decomposition into left and right ciphertexts:



Left ciphertext

Right ciphertext

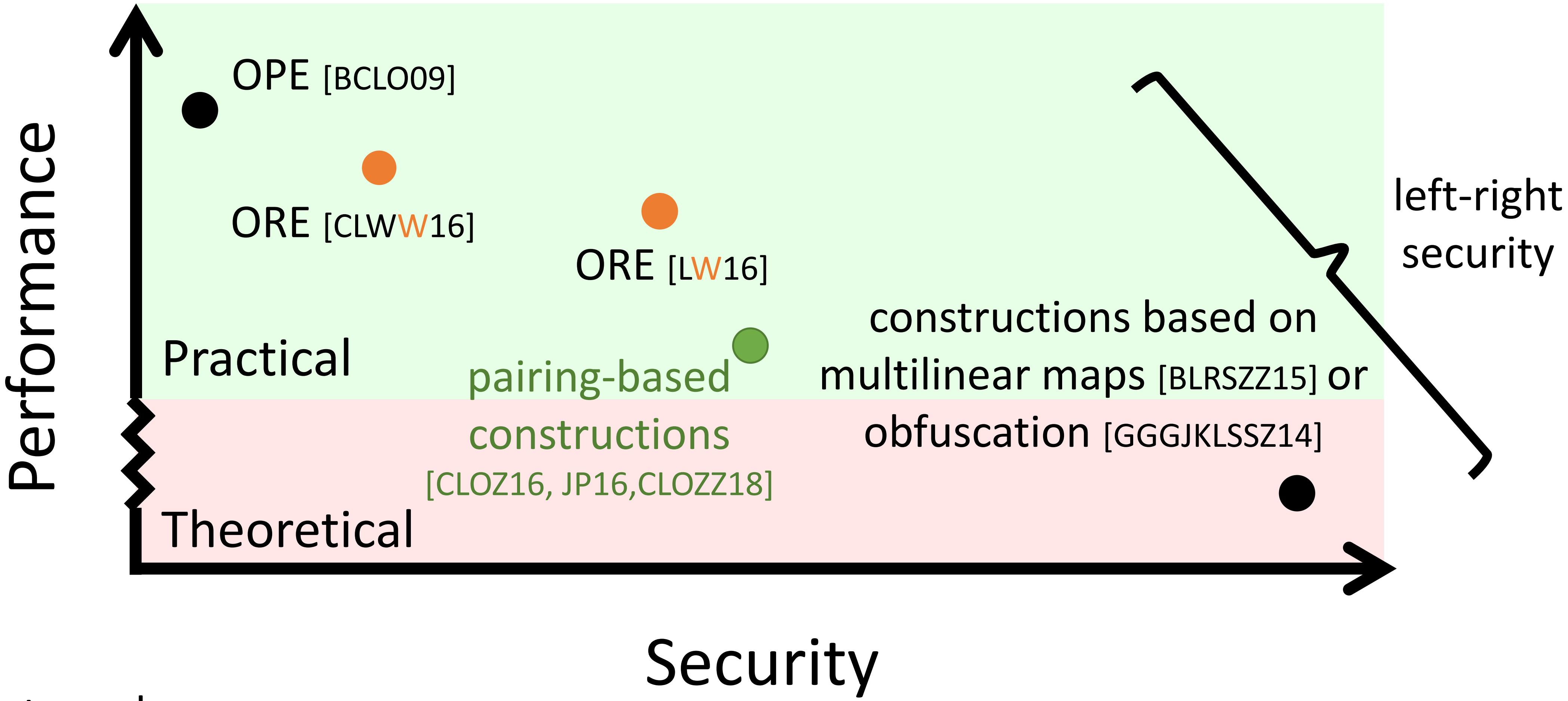
Right ciphertexts are <u>semantically secure</u>
(inherited from underlying small-domain left-right ORE)

Performance Measurements

Scheme	Encrypt (μs)	Compare (μs)	|ct| (bytes)
OPE [BCLO'09]	3601.82	0.36	8
Bit-by-Bit ORE	2.06	0.48	8
Left-Right (4-bit blocks)	16.50	0.31	192
Left-Right (8-bit blocks)	54.87	0.63	224

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

The Landscape of ORE



Not drawn to scale

Challenges in Using ORE

ID	Name	Age	Zip Code
0	Alice	31	68107
1	Bob	47	60015
2	Emily	41	38655
3	Jeff	45	46304

Motivates search for <u>stronger</u> notions of ORE

Real databases will cache query-processing data, so in practice, snapshots will contain query information

Can we construct a left-right ORE that achieves best-possible security if adversary only sees a small number of left ciphertexts?

Challenges in Using ORE

Attacks motivate design of new kinds of cryptographic primitives that better capture practical requirements

- New notions of ORE: *parameter-hiding ORE* [CLOZZ18]

ORE as a building block: direct application of ORE to construct encrypted databases has limitations, but perhaps can combine with other cryptographic tools (e.g., MPC) for better security

Conclusions

Searching on encrypted data is an important problem

Role of cryptography: Identify and construct useful cryptographic building blocks to enable and facilitate new designs of encrypted databases

OPE → ORE → Left-Right ORE → Parameter-Hiding ORE

Better attacks and security analysis motivate new cryptographic notions and raise interesting questions both for theory and for practice!

Thank you!