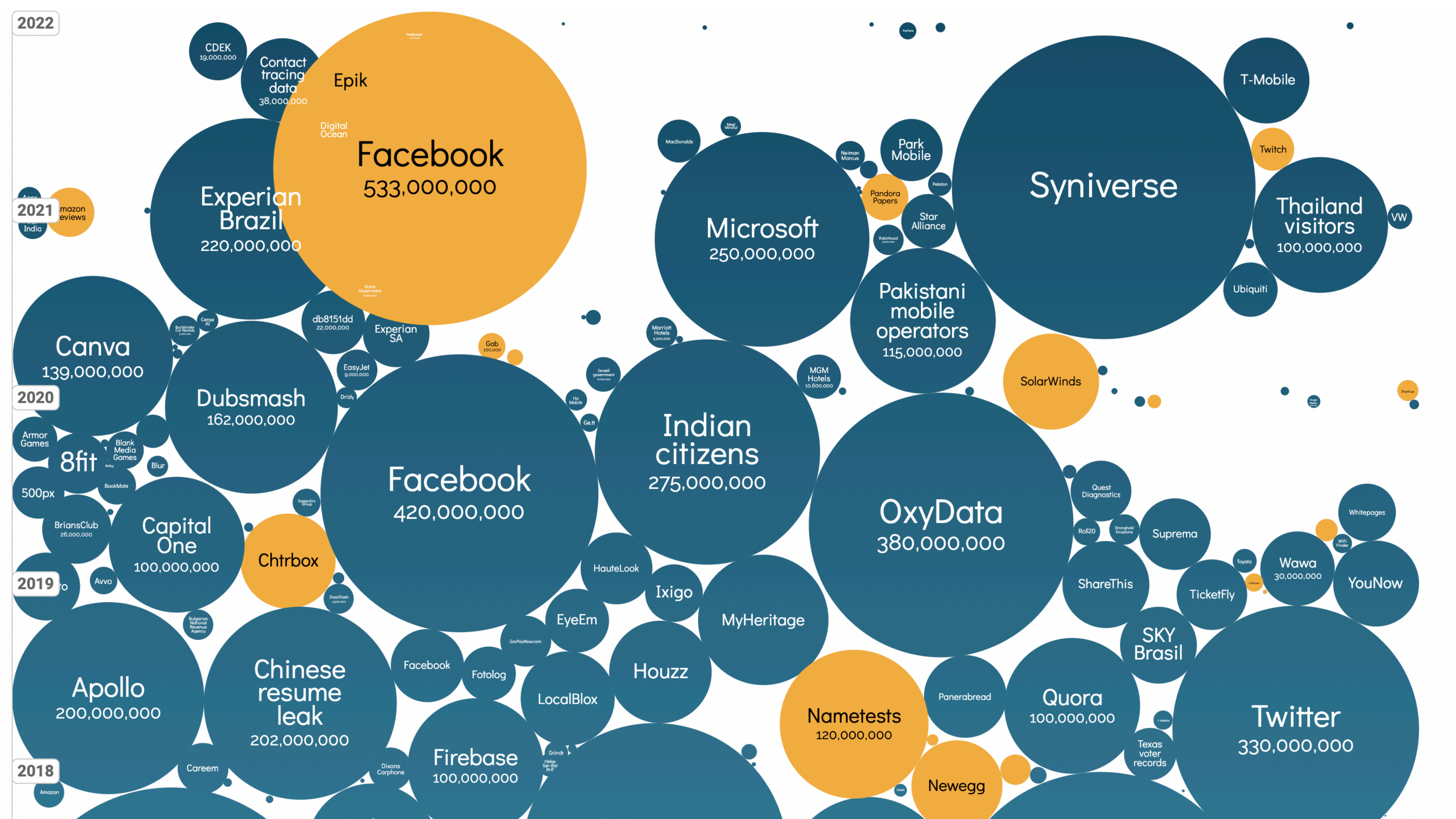
Encrypted Distributed Systems

Seny Kamara

with Archita Agarwal

MongoDB®

BROWN


2022
2021
2020
2019
2018
CDEK
19,000,000
Contact tracing data
38,000,000
Epik
Digital Ocean
Facebook
533,000,000
Experian Brazil
220,000,000
Amazon reviews
India
Canva
139,000,000
db8151dd
22,000,000
Experian SA
EasyJet
9,000,000
Drizly
Dubsmash
162,000,000
Armor Games
8fit
Blank Media Games
Blur
500px
BookMate
BriansClub
26,000,000
Capital One
100,000,000
Chtrbox
Avvo
DoorDash
Facebook
420,000,000
Gab
100,000
MacDonalds
Park Mobile
Neiman Marcus
Pandora Papers
Peloton
Star Alliance
Robinhood
Microsoft
250,000,000
Syniverse
T-Mobile
Twitch
Thailand visitors
100,000,000
VW
Ubiquiti
Pakistani mobile operators
115,000,000
Marriott Hotels
Israeli government
Ge.tt
Indian citizens
275,000,000
MGM Hotels
10,600,000
SolarWinds
OxyData
380,000,000
Quest Diagnostics
Roll20
Stronghold Kingdoms
Suprema
Whitepages
WiFi Finder
ShareThis
Toyota
TicketFly
Wawa
30,000,000
YouNow
HauteLook
Ixigo
EyeEm
MyHeritage
Houzz
GovPayNow.com
Facebook
Fotolog
LocalBlox
Apollo
200,000,000
Bulgarian National Revenue Agency
Chinese resume leak
202,000,000
Careem
Amazon
Dixons Carphone
Firebase
100,000,000
Grindr
Nametests
120,000,000
Panerabread
Newegg
Quora
100,000,000
SKY Brasil
Texas voter records
Twitter
330,000,000

14,717,618,286*

4%

* since 2013

Why so Few?

Incompetence?

Lazyness?

Cost?

"...because it would have hurt Yahoo's ability to index and search message data..."

— J. Bonforte in NY Times

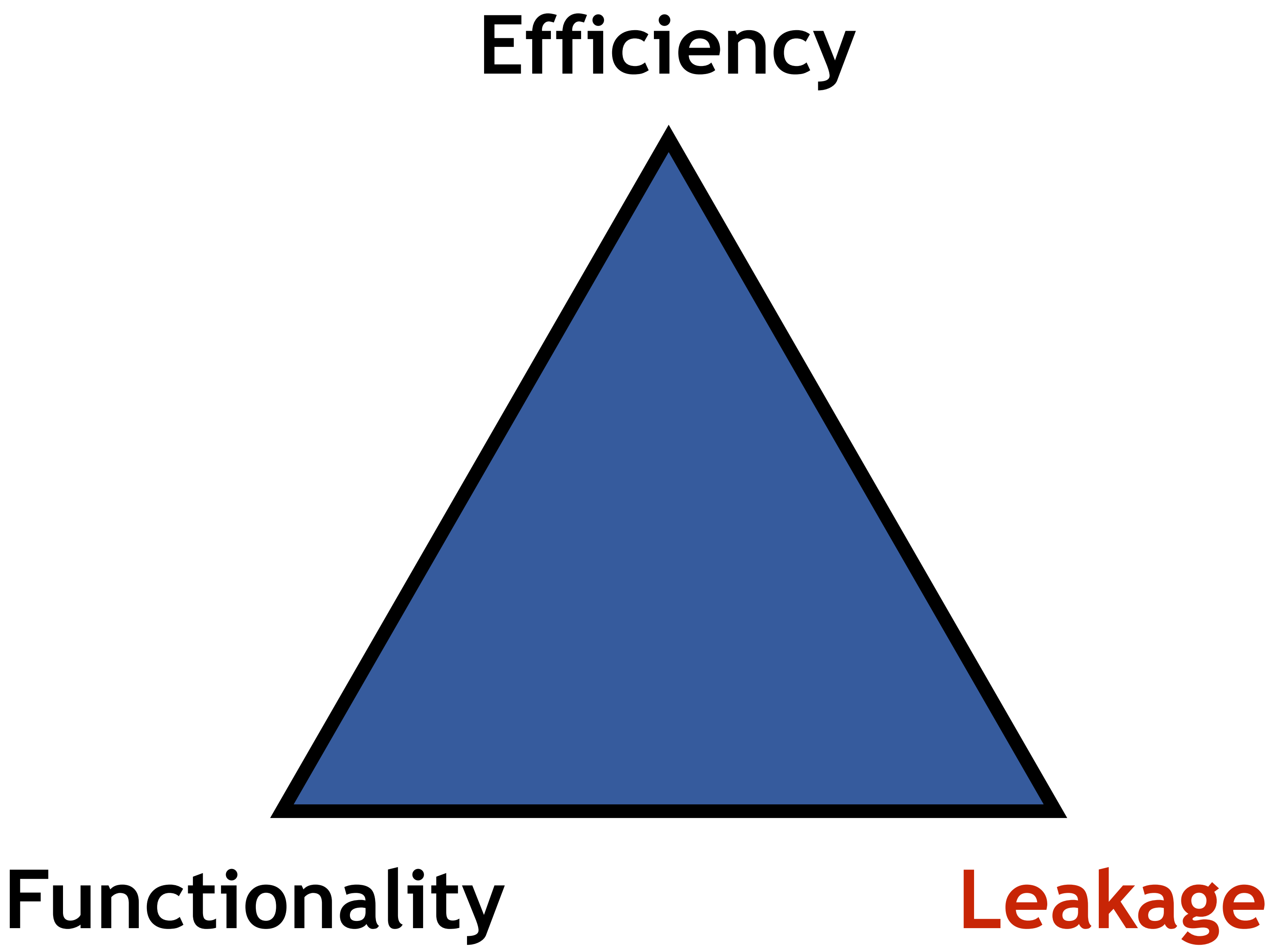
Q: can we search on encrypted data?

Encrypted Search Algorithms

- Major companies
  - MongoDB, Google
  - Meta, Microsoft
  - Amazon, Cisco
  - Hitachi, Fujitsu
  - more…
- Funding agencies
  - NSF
  - IARPA
  - DARPA
- Startups
  - Aroki Systems (acquired)
  - too many to list…

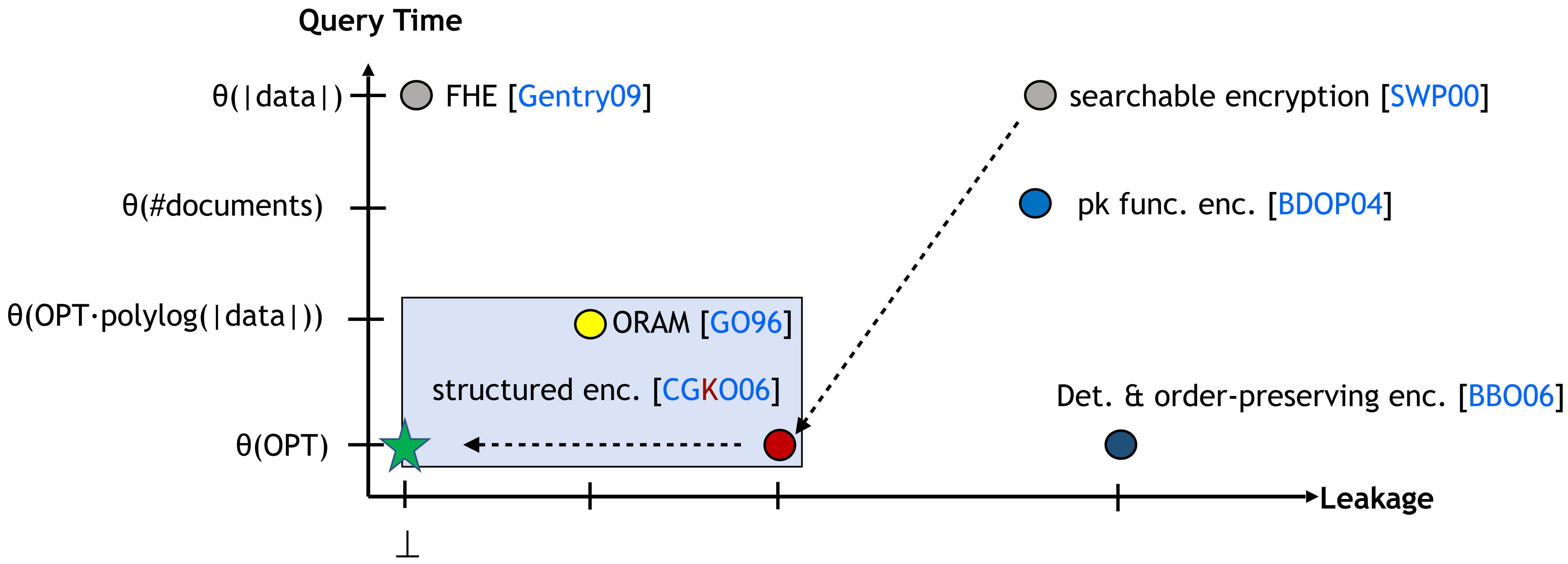
Encrypted Search Algorithms

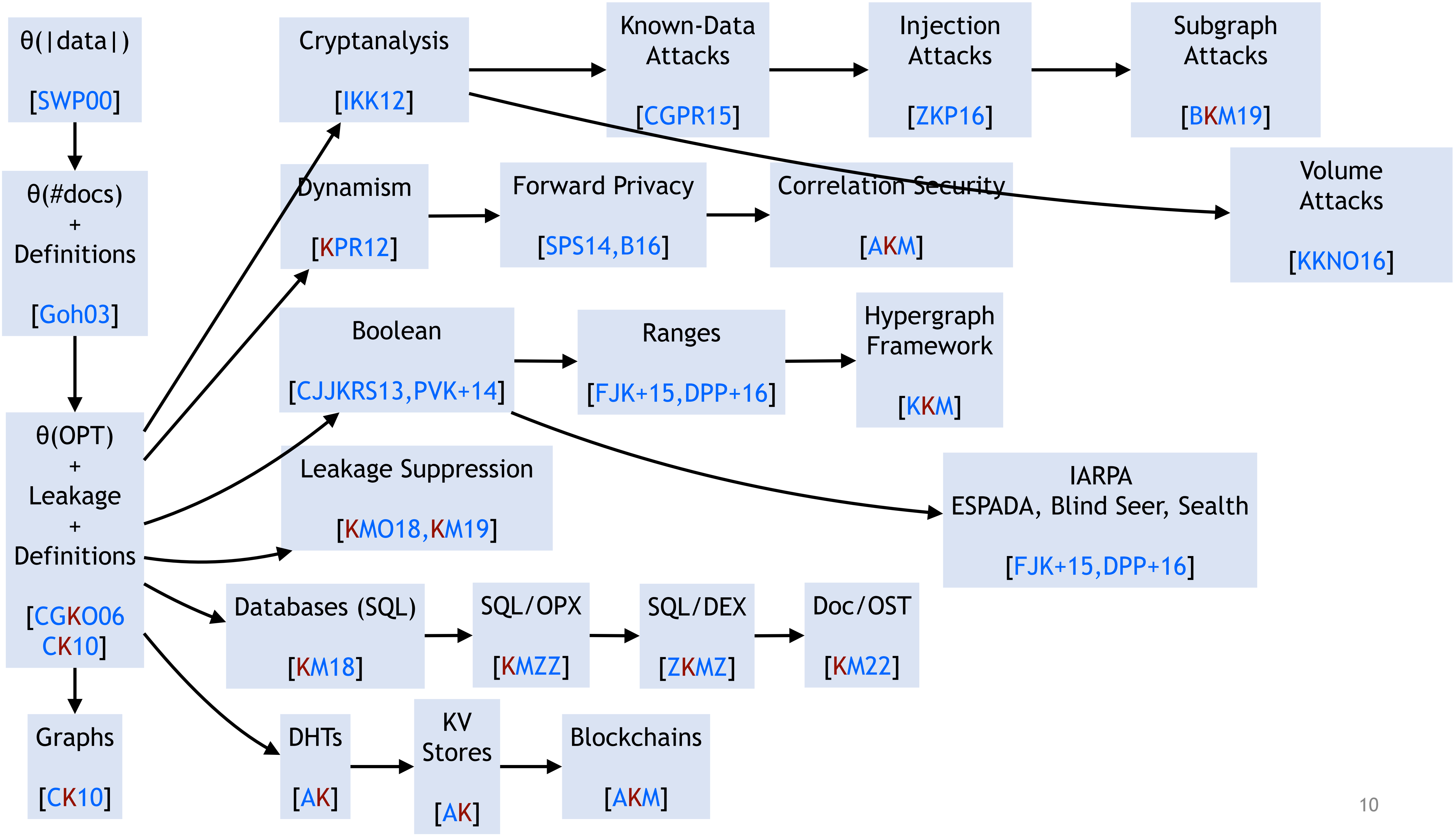
Property-Preserving Encryption (PPE) [BBO06]	Functional Encryption [BSW11]	Structured Encryption (STE) [CGKO06,CK10]
Fully-Homomorphic Encryption (FHE) [Gentry09]	Oblivious RAM (ORAM) [GO96]	Multi-Party Computation [Yao86,GMW87]


Efficiency
Functionality
Leakage

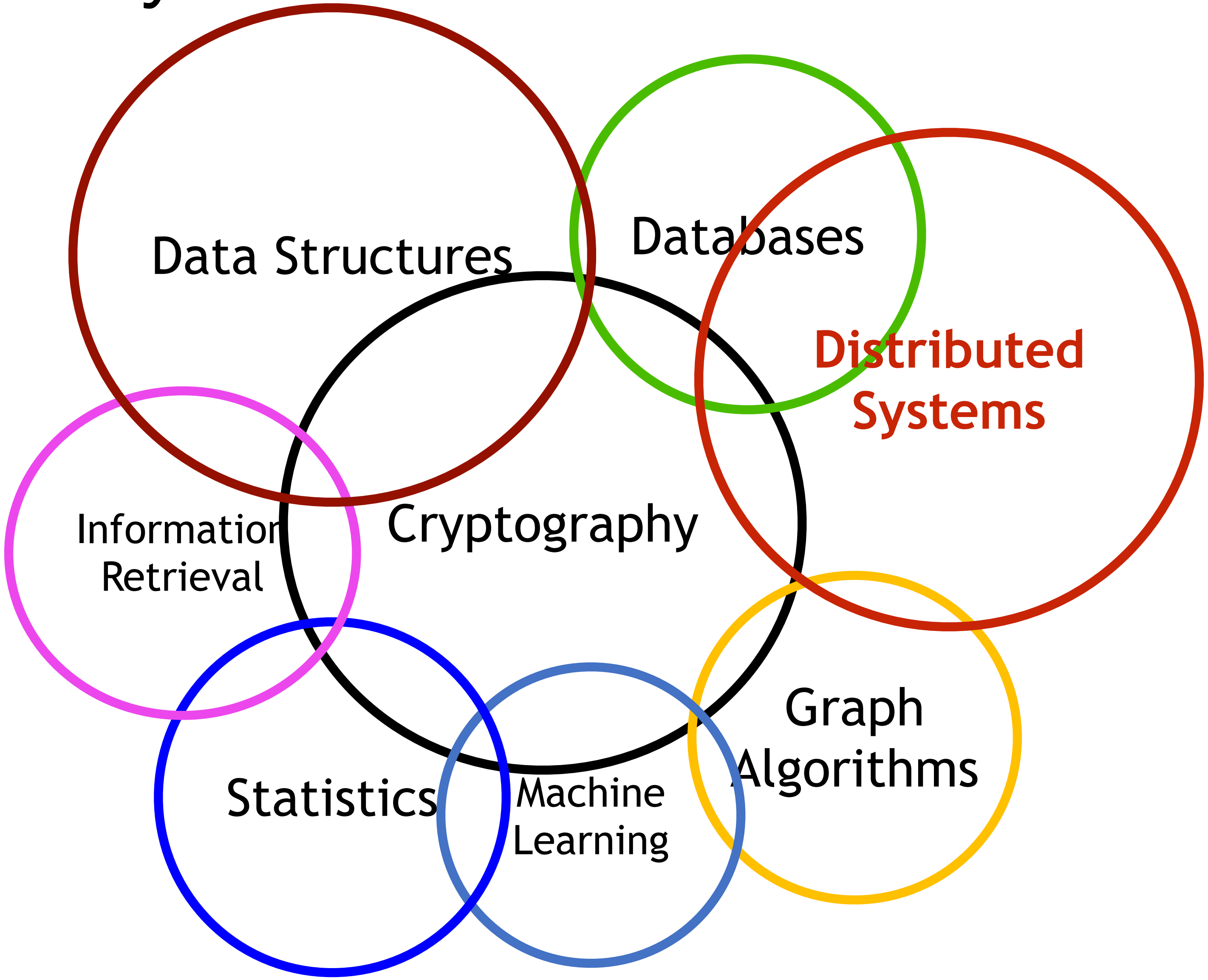
Not Scientific!

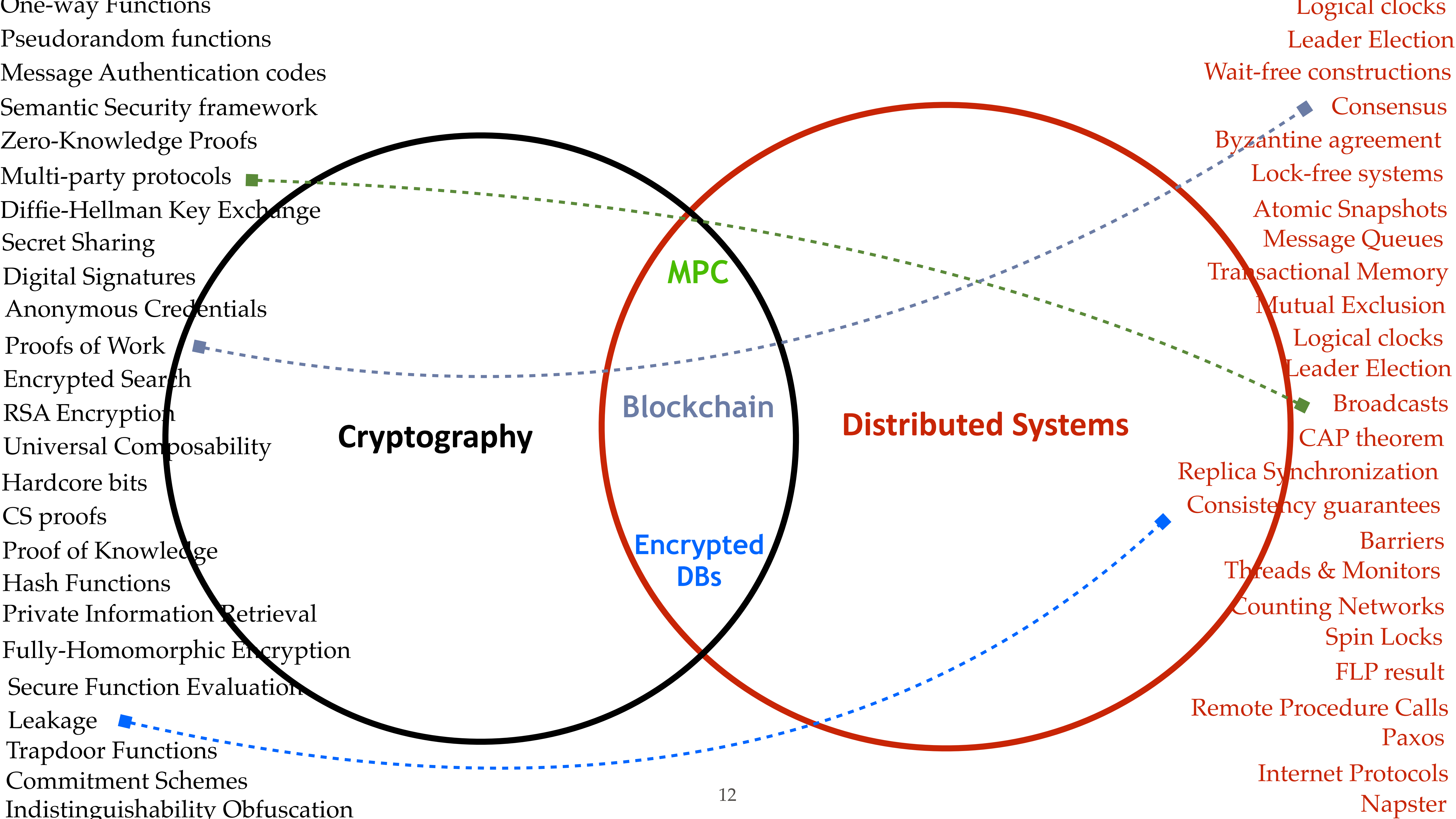
Efficiency vs. Security




θ(|data|)
[SWP00]
θ(#docs) + Definitions
[Goh03]
θ(OPT) + Leakage + Definitions
[CGKO06 CK10]
Graphs
[CK10]
Cryptanalysis
[IKK12]
Known-Data Attacks
[CGPR15]
Injection Attacks
[ZKP16]
Subgraph Attacks
[BKM19]
Volume Attacks
[KKNO16]
Dynamism
[KPR12]
Forward Privacy
[SPS14,B16]
Correlation Security
[AKM]
Boolean
[CJJKRS13,PVK+14]
Ranges
[FJK+15,DPP+16]
Hypergraph Framework
[KKM]
IARPA
ESPADA, Blind Seer, Sealth
[FJK+15,DPP+16]
Leakage Suppression
[KMO18,KM19]
Databases (SQL)
[KM18]
SQL/OPX
[KMZZ]
SQL/DEX
[ZKMZ]
Doc/OST
[KM22]
DHTs
[AK]
KV Stores
[AK]
Blockchains
[AKM]

Interdisciplinary

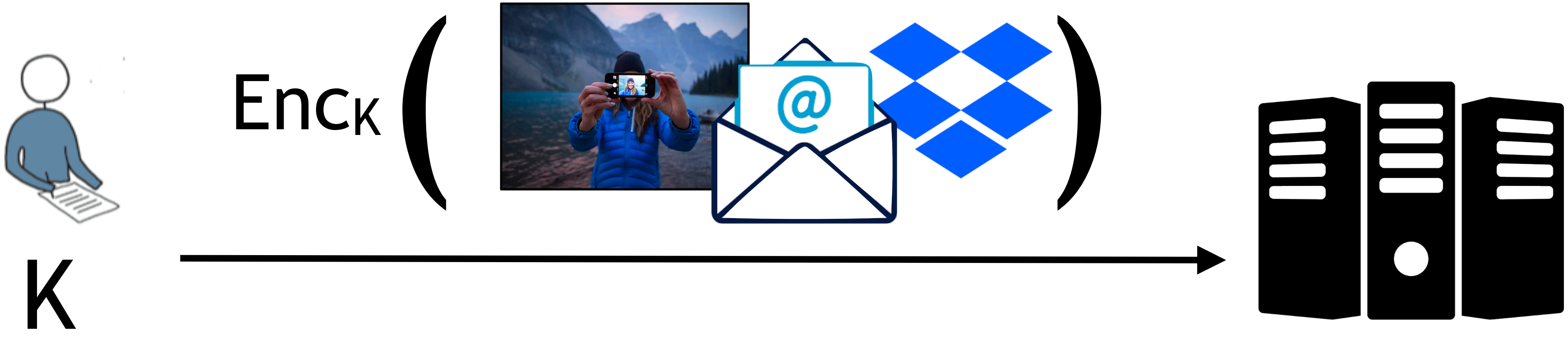




One-way Functions
Pseudorandom functions
Message Authentication codes
Semantic Security framework
Zero-Knowledge Proofs
Multi-party protocols
Diffie-Hellman Key Exchange
Secret Sharing
Digital Signatures
Anonymous Credentials
Proofs of Work
Encrypted Search
RSA Encryption
Universal Composability
Hardcore bits
CS proofs
Proof of Knowledge
Hash Functions
Private Information Retrieval
Fully-Homomorphic Encryption
Secure Function Evaluation
Leakage
Trapdoor Functions
Commitment Schemes
Indistinguishability Obfuscation
Cryptography
MPC
Blockchain
Encrypted DBs
Distributed Systems
Logical clocks
Leader Election
Wait-free constructions
Consensus
Byzantine agreement
Lock-free systems
Atomic Snapshots
Message Queues
Transactional Memory
Mutual Exclusion
Logical clocks
Leader Election
Broadcasts
CAP theorem
Replica Synchronization
Consistency guarantees
Barriers
Threads & Monitors
Counting Networks
Spin Locks
FLP result
Remote Procedure Calls
Paxos
Internet Protocols
Napster

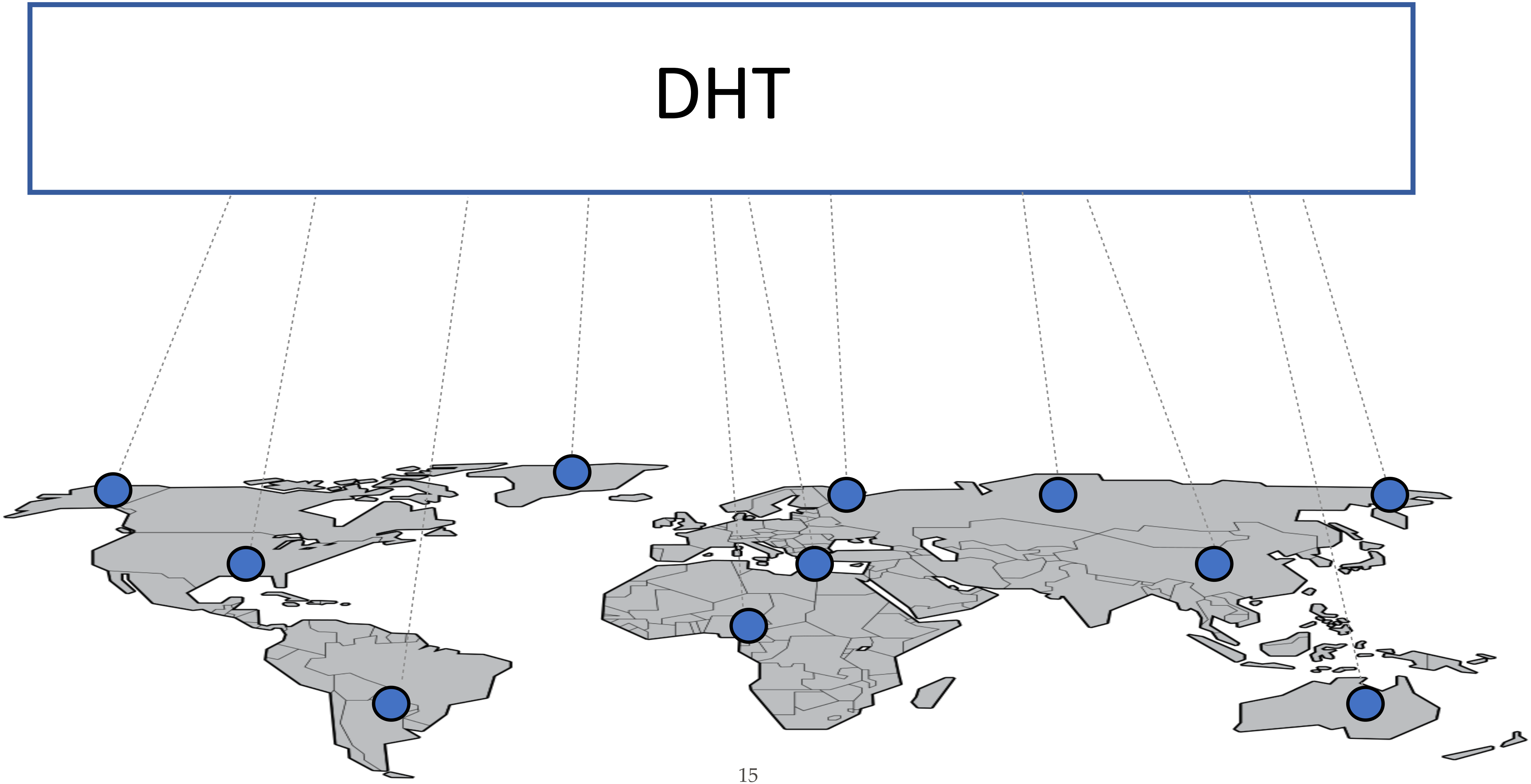
Encrypted Algorithms & Encrypted Systems

- **Q:** can we design algorithms that operate on encrypted data?
- **Q:** can we build systems that run on encrypted data?
  - databases, key-value stores, blockchains, …



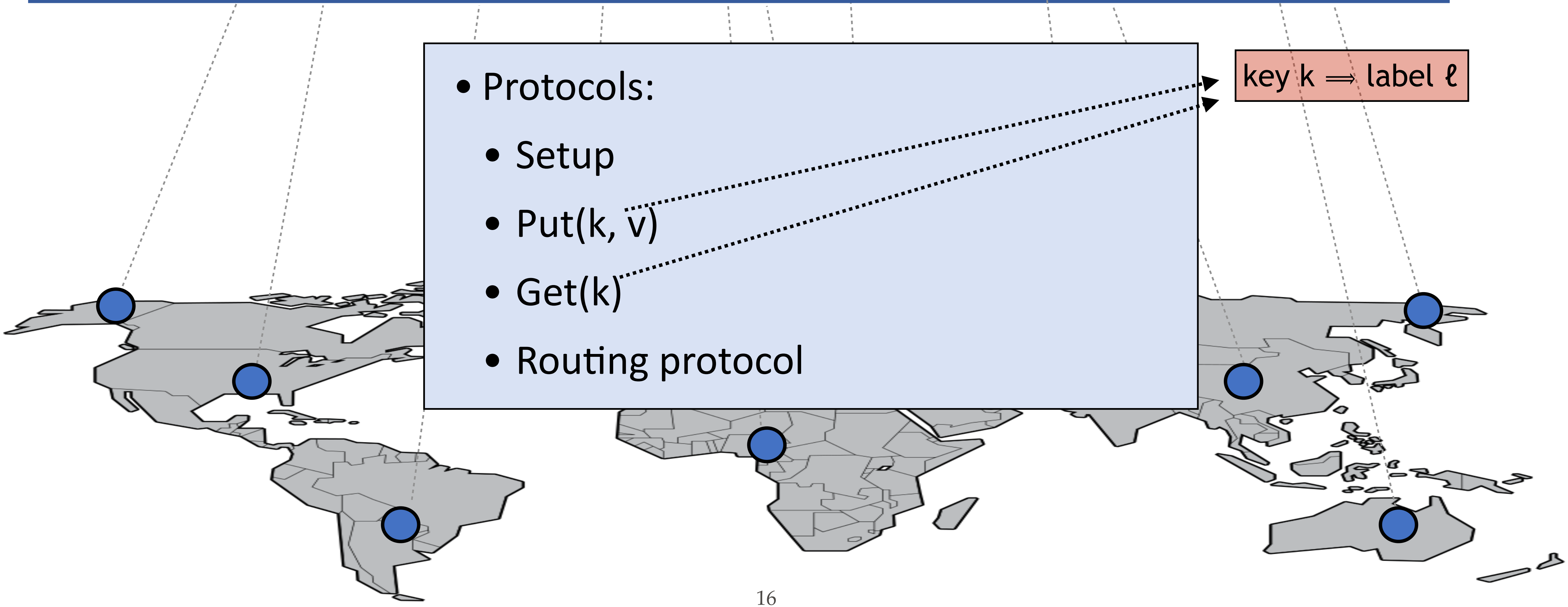
- search
- answer DB queries
- train ML models
- …

Q: what’s the simplest distributed data structure?


DHT

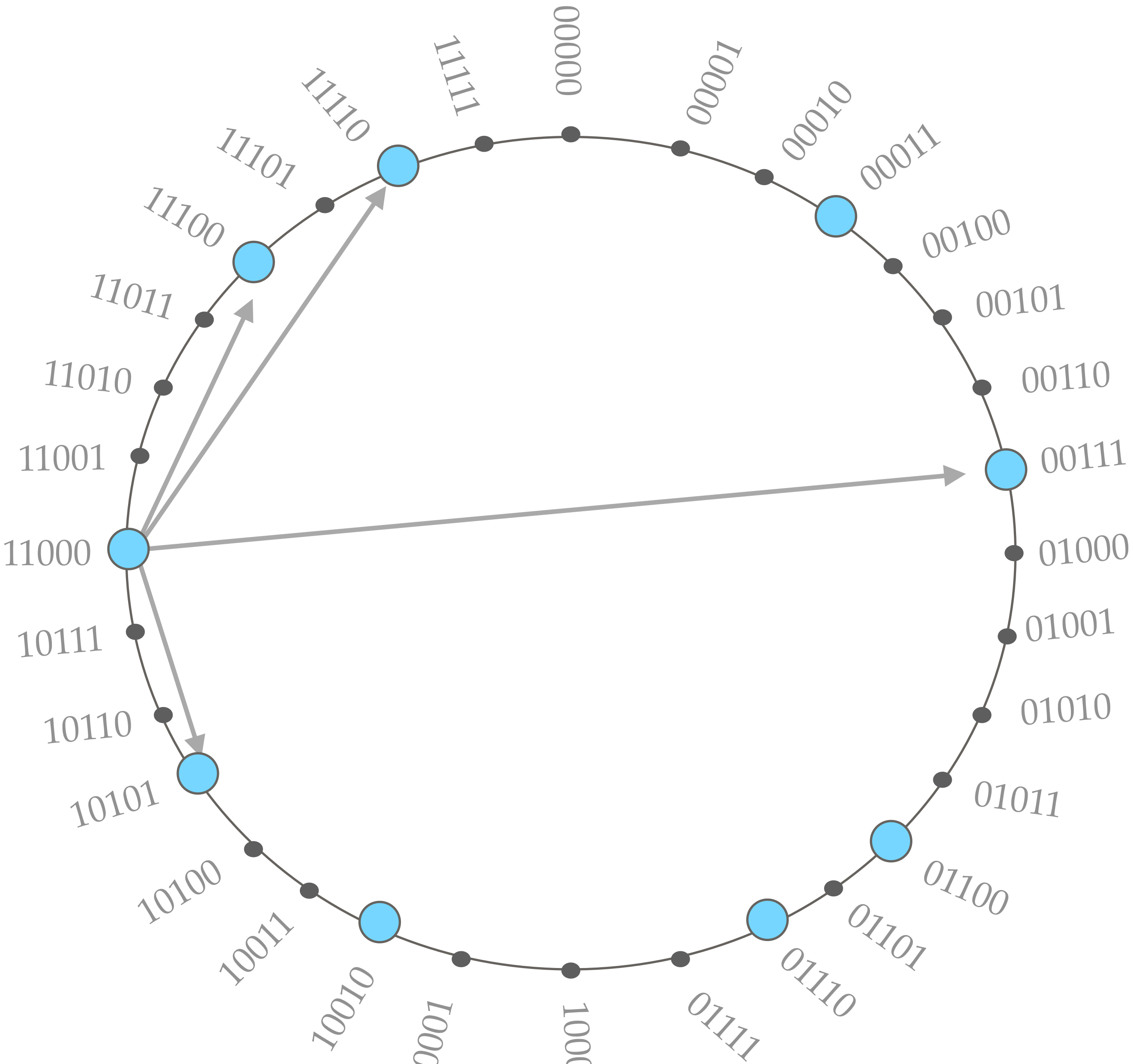
DHT

- Protocols:
  - Setup
  - Put(k, v)
  - Get(k)
  - Routing protocol

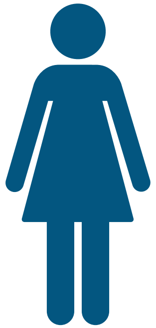


Chord DHT

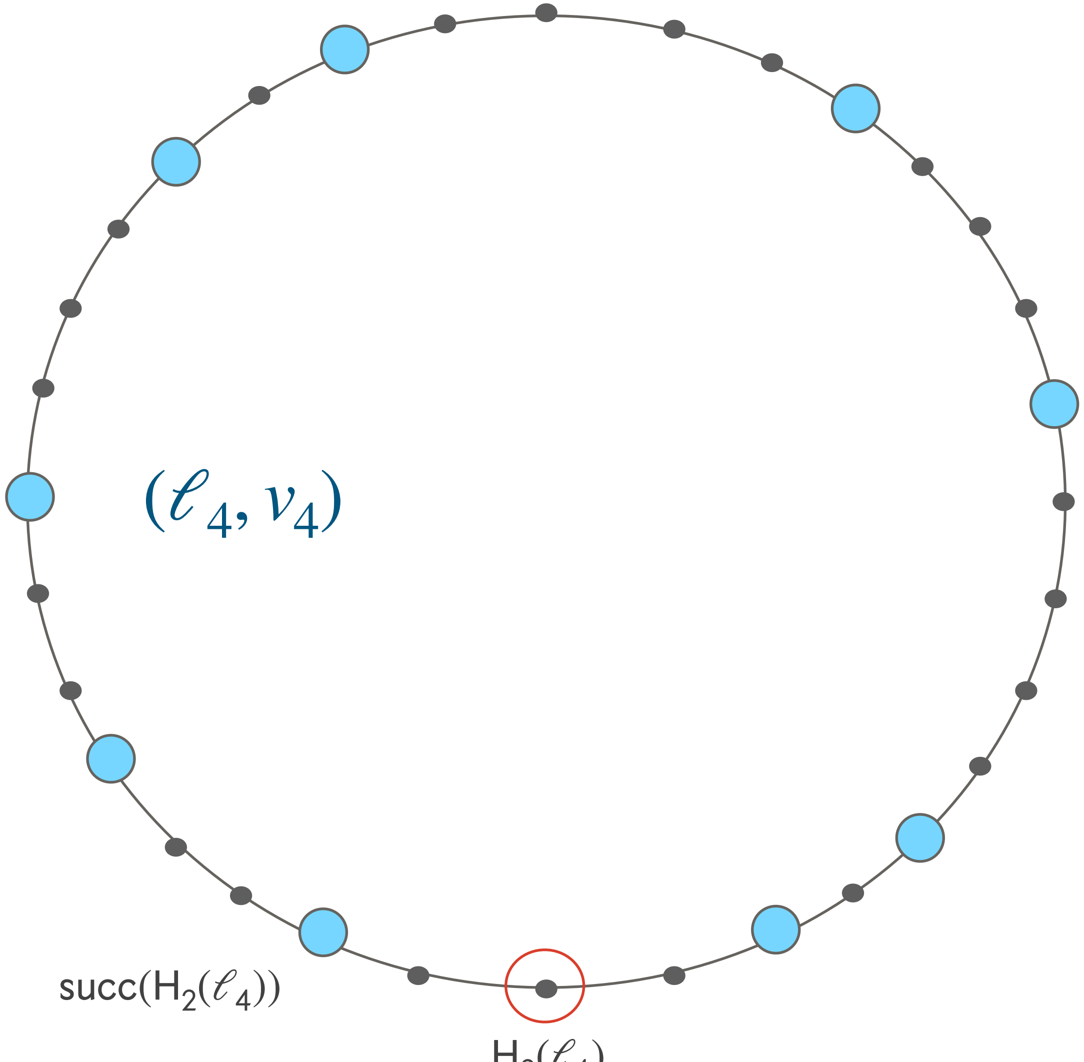
- Logical Address Space : **A**
- $(H_1, H_2) \leftarrow \mathsf{Setup}()$
  - H_1 : hashes node ids to addresses
  - H_2: hashes labels to addresses
- Routing
  - Logarithmic sized routing tables
  - Logarithmic sized paths



Chord DHT : **Put()**



Put(ℓ_4, v_4)



Abstraction of DHTs

$succ \circ H_2$

Where should labels be stored?

Where should clients send their requests

H_3

- addr : $\mathbf{N} \rightarrow \mathbf{A}$
- server : $\mathbf{L} \rightarrow \mathbf{A}$
- route : $\mathbf{A} \times \mathbf{A} \rightarrow 2^{\mathbf{A}}$
- fe : $\mathbf{L} \rightarrow \mathbf{A}$

All nodes are assigned logical addresses

H_1

How should messages be routed?

Fixed

OUTLINE

(I) Encrypted DHTs

(III) Takeaways & Conclusion

- What are DHTs
  - Abstraction of core components
- **Formalize EDHTs**
  - Syntax & Security defn
- Construction
- Analysis of EDHTs
  - Main security theorem

Formalizing EDHTs

- Define the **syntax** of EDHTs
- Define the **security** of EDHTs

Formalizing EDHTs

- Define the **syntax** of EDHTs
- Define the **security** of EDHTs

Formalizing EDHTs : Syntax

$$\text{EDHT} = (\text{Gen}, \text{Setup}, \text{Put}, \text{Get})$$

- Executed by user
- Generates cryptographic keys

- Executed by trusted party
- sets up system

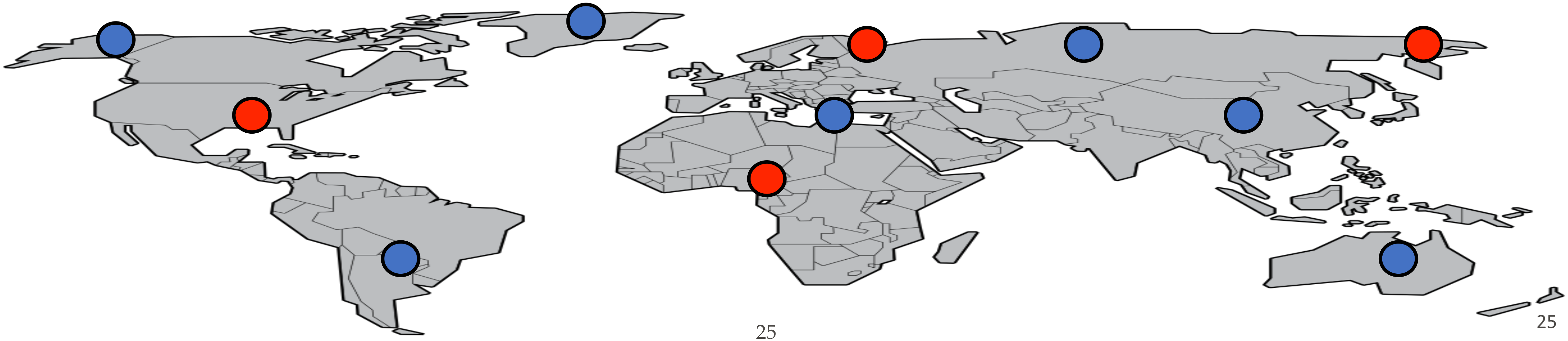
- Executed by user
- Put(K, ℓ, v): stores (ℓ, v)
- Get(K, ℓ): retrieves (ℓ, v)

Formalizing EDHTs

- Define the **syntax** of EDHT
- Define the **security** of EDHTs

Adversarial Model

- Static
- Semi-honest



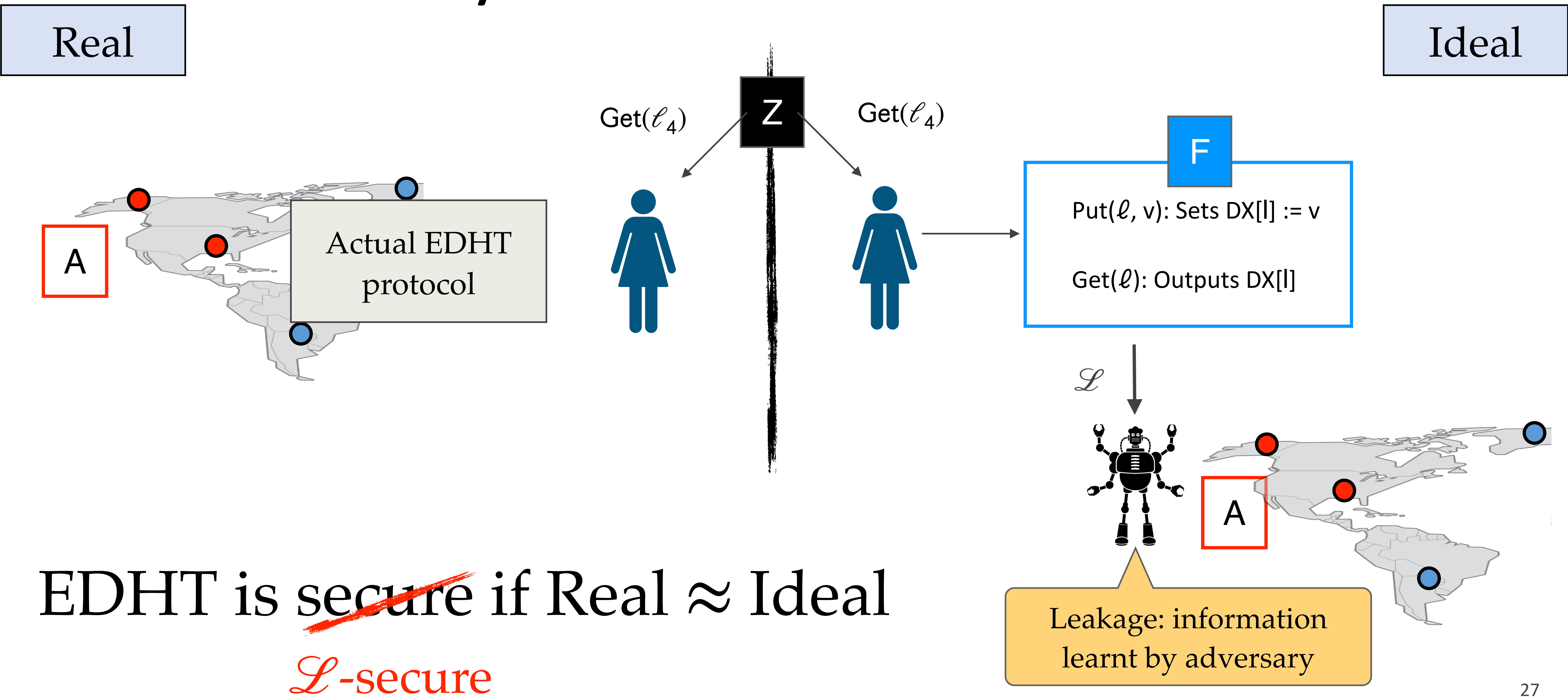
25

EDHTs Security

Real

Ideal

EDHT Security



EDHT is ~~secure~~ $\mathscr{L}$-secure if Real ≈ Ideal

OUTLINE

(I) Encrypted DHTs

(III) Takeaways & Conclusion

- What are DHTs
  - Abstraction of core components
- Formalize EDHTs
  - Syntax & Security defn
- **Construction**
- Analysis of EDHTs
  - Main security theorem

EDHT Construction

Setup()

- DHT.Setup()

Gen(1^k)

- Sample $K_1 \leftarrow \{0, 1\}^k$
- $K_2 \leftarrow$ SKE.Gen(1^k)
- Output (K_1, K_2)

Put(K, ℓ, v)

- $K = (K_1, K_2)$
- $t = F_{K1}(\ell)$
- e = SKE.Enc$_{K2}$(v)
- DHT.Put(t, e)

Get(K, ℓ)

- $K = (K_1, K_2)$
- $t = F_{K1}(\ell)$
- e $\leftarrow$ DHT.Get (t)
- v $\leftarrow$ SKE.Dec$_{K2}$(e)

OUTLINE

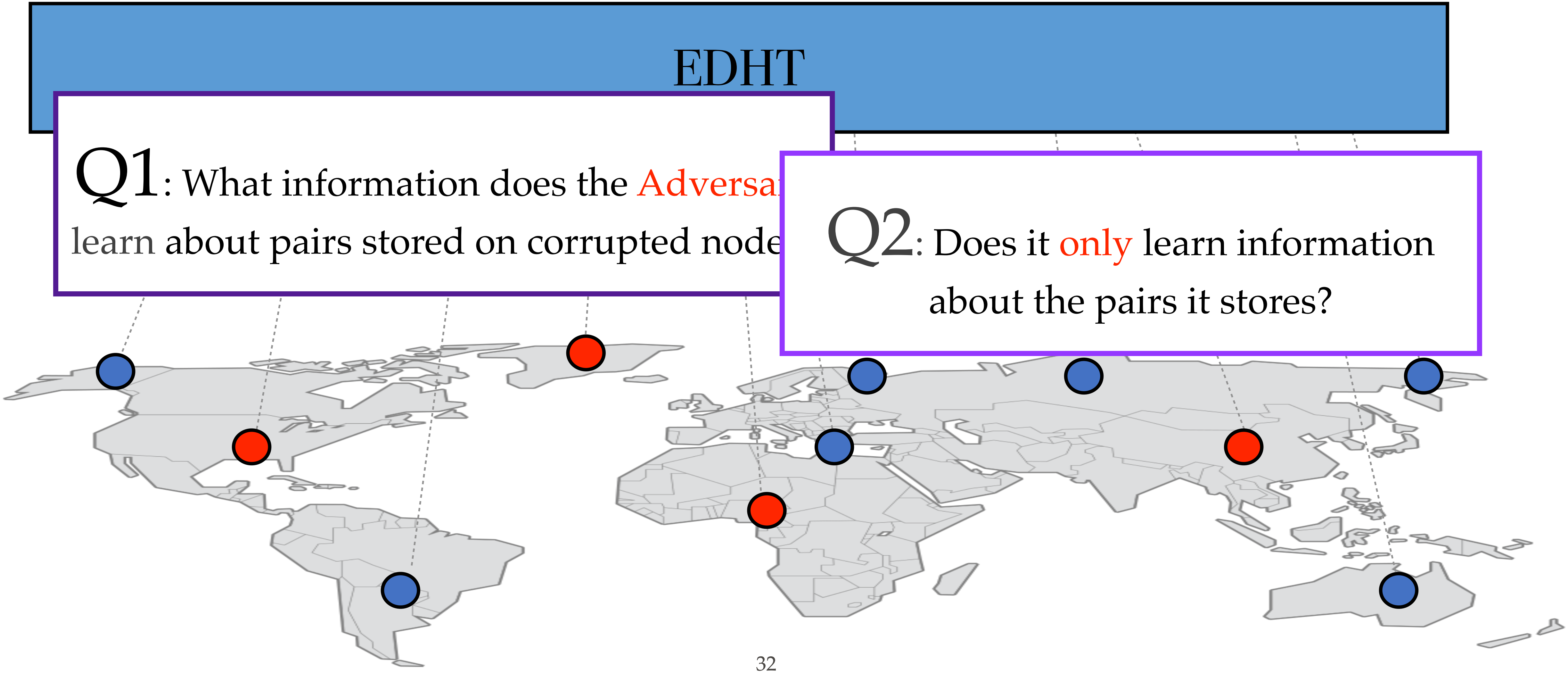
(I) Encrypted DHTs

(III) Takeaways & Conclusion

- What are DHTs
  - Abstraction of core components
- Formalize EDHTs
  - Syntax & Security defn
- Construction
- **Analysis of EDHTs**
  - **Main security theorem**

Q: What kind of security are we getting?

What does the Adversary learn?



Q1: What information does the Adversa
learn about pairs stored on corrupted node

Q2: Does it only learn information about the pairs it stores?

What does the Adversary learn?

Example:

Infer a good approximation of total number of pairs!

- Total pairs adv. holds : m
- Total expected pairs : ~ mn/t
  - if DHTs are load balanced

Q2: Does it only learn information about the pairs it stores?

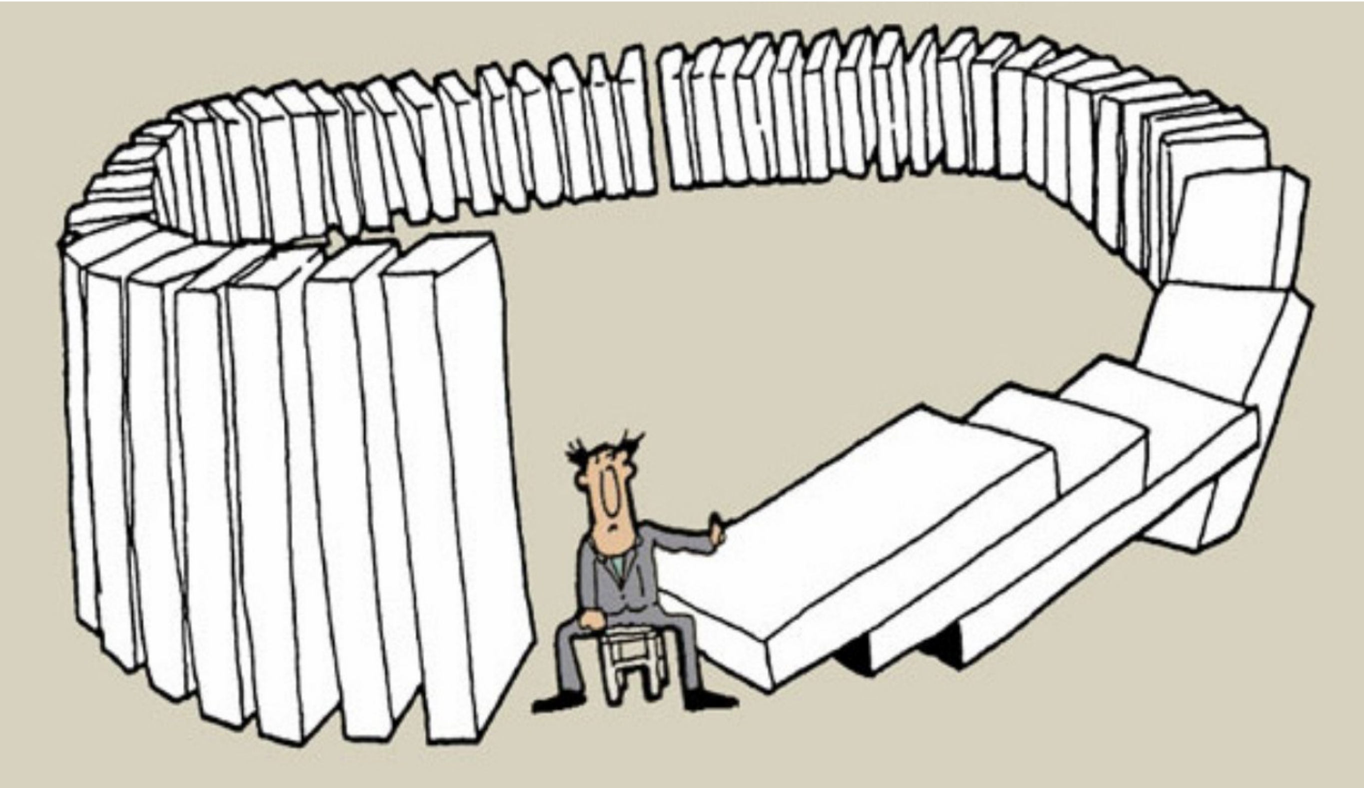
NO

System architecture

Security

Properties of DHTs

P1: Balance



P2: Non-committing allocations

Properties of DHTs

P1: Balance

whp, the probability of
any θ-bounded adversary
seeing a label
should not be more than ε

- addr : $\mathbf{N} \rightarrow \mathbf{A}$
- server : $\mathbf{L} \rightarrow \mathbf{A}$
- route : $\mathbf{A} \times \mathbf{A} \rightarrow 2^{\mathbf{A}}$
- fe : $\mathbf{L} \rightarrow \mathbf{A}$

P2: Non-committing allocations

Properties of DHTs

P1: Balance

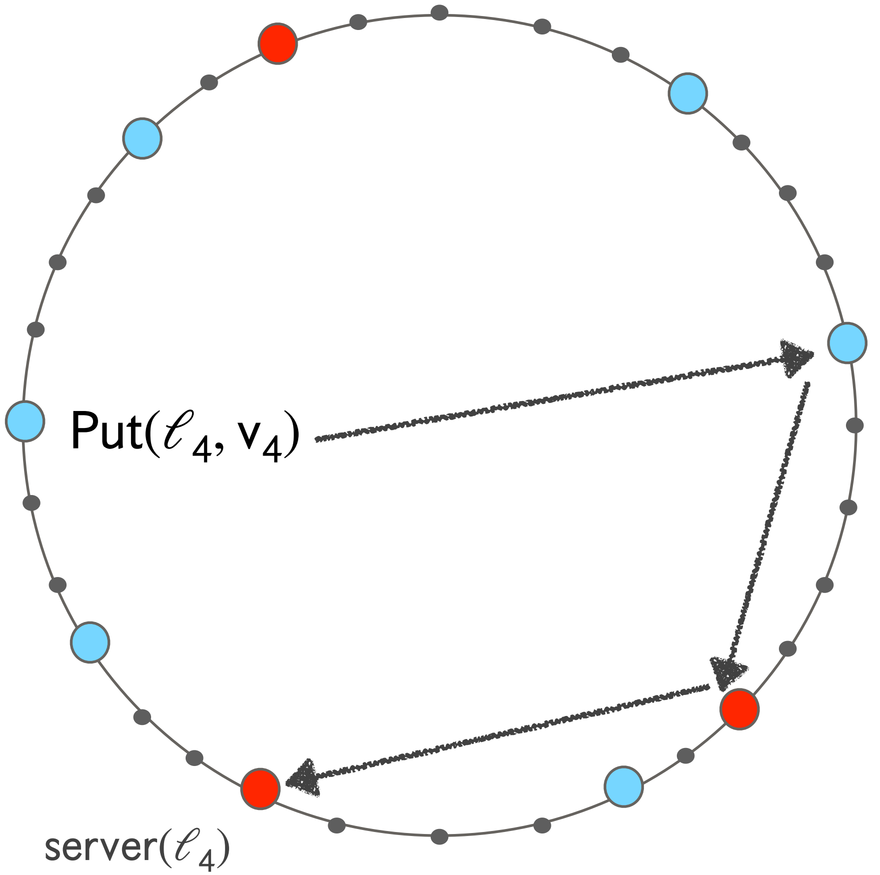
whp, the probability of
any θ-bounded adversary
seeing a label
should not be more than ε

- addr : **N** → **A**
- server : **L** → **A**
- route : **A** X **A** → $2^{\mathbf{A}}$
- fe : **L** → **A**

P2: Non-committing allocations

When does an adversary see a label?

- When it **stores** the label or **routes** the label



Properties of DHTs

P1: Balance

whp, the probability of
any θ-bounded adversary
seeing a label
should not be more than ε

Storing or routing a label

- addr : **N** → **A**
- server : **L** → **A**
- route : **A** X **A** → $2^{\mathbf{A}}$
- fe : **L** → **A**

P2: Non-committing allocations

much more technical!

Leakage

L_ε:

leaks the repetition pattern (when a query for the same label is repeated) for an ε-fraction of queries

affected by balance ε of DHT

Main Security Theorem

Th : If DHT is $(\varepsilon, \theta, \delta)$-balanced and has non-committing allocations, then EDHT is $\mathcal{L}_\varepsilon$-secure with prob at least $1 - \delta - \mathrm{negl}(k)$

Balance of Chord

Th : Chord is $(\varepsilon, \theta, \delta)$-balanced for

$$\varepsilon = \frac{\theta}{n}\left(\log n + 6\log\left(\frac{n}{\theta}\right)\right), \quad \delta = \frac{1}{n^2} \text{ and } \theta \le \frac{n}{e\log n}$$

Balance of Chord

Th : Chord is $(\varepsilon, \theta, \delta)$-balanced for

$$\varepsilon = \frac{\theta}{n}\left(\log n + 6\log\left(\frac{n}{\theta}\right)\right), \quad \delta = \frac{1}{n^2} \text{ and } \theta \leq \frac{n}{e\log n}$$

↓

$$\varepsilon = O\left(\frac{\theta}{n}\log n\right) \quad \text{vs} \quad \varepsilon = O\left(\frac{\theta}{n}\right)$$

optimal

OUTLINE

(I) Encrypted DHTs

Transient DHTs

(III) Takeaways & Conclusion

- What are DHTs
  - Abstraction of core components
- Formalize EDHTs
  - Syntax & Security defn
- Construction
- Analysis of EDHTs
  - Main security theorem

Outline

(0) Introduction ✓

(I) Encrypted DHTs ✓

(II) Encrypted Key-Value Stores

(III) Future Directions

What are Key-Value Stores?

Same as DHTs

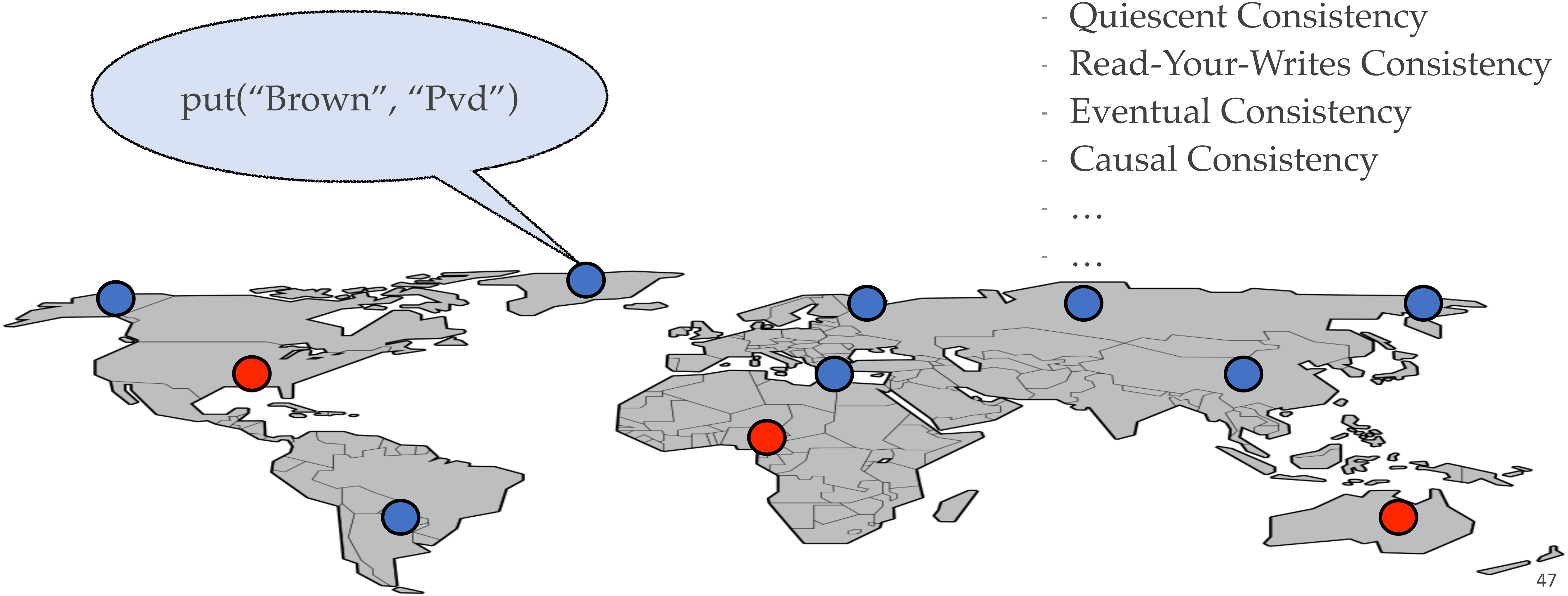
+

Replication

KVS

CONSISTENCY ??

- Linearizability
- Sequential Consistency
- Quiescent Consistency
- Read-Your-Writes Consistency
- Eventual Consistency
- Causal Consistency
- …
- …



Abstraction of KVS

- addr : $\mathbf{N} \rightarrow \mathbf{A}$
- ~~server : $\mathbf{L} \rightarrow \mathbf{A}$~~ replicas : $\mathbf{L} \rightarrow 2^{A}$
- route : $\mathbf{A} \times \mathbf{A} \rightarrow 2^{\mathbf{A}}$
- fe : $\mathbf{L} \rightarrow \mathbf{A}$

Construction of EKVS

SAME AS BEFORE

Put(K, ℓ, v)

- $t = F_{K1}(\ell)$
- $e = SKE.Enc_{K2}(v)$
- ~~DHT.Put(t, e)~~
  KVS.Put(t, e)

Security of EKVS

Single user setting

Clients do not share data

Multi user setting

Clients can share data

concurrent operations on same piece of data possible

Properties of KVSs

P1: Balance

whp, the probability of
any θ-bounded adversary
seeing a label
should not be more than ε

P2: Non-committing

much more
technical!

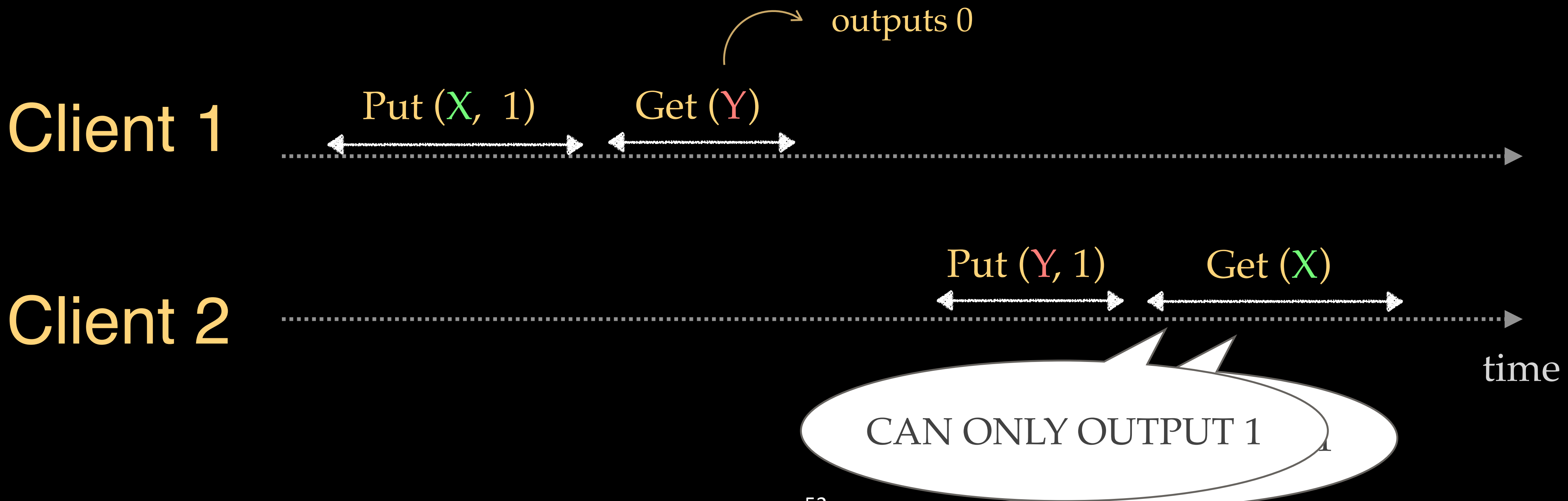
P3: Consistency

"I know my grades are straight 'Cs,' but don't I get some credit for consistency?"

Security of EKVS

good nodes ⟶ Label X, Label Y ⟵ bad nodes

KVS is Sequentially Consistent



Security of EKVS

Single user setting

If KVS is $(\varepsilon, \theta, \delta)$-balanced, and **RYW consistent**, then EKVS is L_ε-secure with prob at least $1 - \delta - \text{negl}(k)$

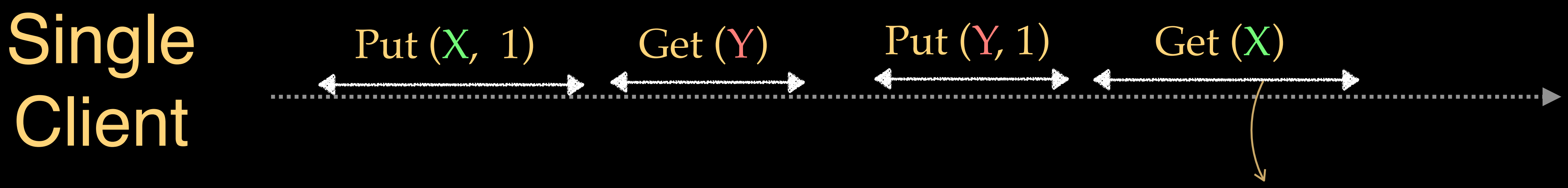
repetition pattern on pairs visible to the adversary

Multi user setting

Clients can share data

concurrent operations on same piece of data possible

Security of EKVS



will always output 1
because in *single-user* setting
RYW guarantees
Get(X) reads last Put(X) independently
of operations on Y

Security of EKVS

Single user setting

If KVS is $(\varepsilon, \theta, \delta)$-balanced, and **RYW consistent**, then EKVS is L_ε-secure with prob at least $1 - \delta - negl(k)$

repetition pattern on pairs visible to the adversary

Multi user setting

EKVS is L-secure with prob at least $1 - negl(k)$

repetition pattern on all the pairs

Outline

(0) Introduction

(I) Encrypted DHTs

(II) Encrypted Key Value Stores

(III) Future Directions

Security of EKVS

Single user setting

If KVS is $(\varepsilon, \theta, \delta)$-balanced, and RYW consistent, then EKVS is L_ε-secure with prob at least $1 - \delta - negl(k)$

Q1: What happens w/ other consistency guarantees?

with prob at least $1 - negl(k)$

Q2: Are stronger notions of consistency better for privacy?

Security of EKVS

If K…

RY…

EKVS…

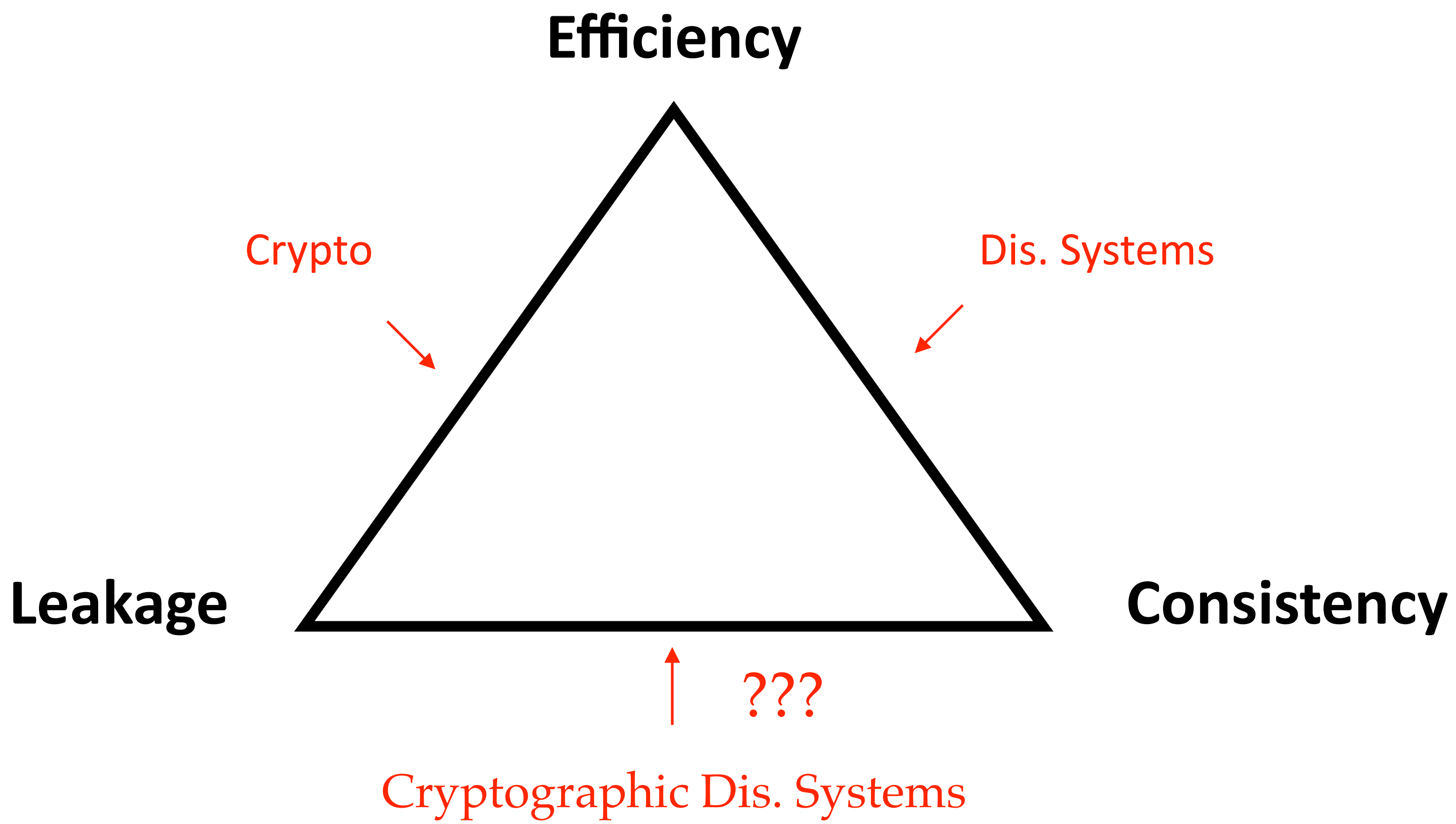
with p…

Q3: Can we improve security by assuming some consistency guarantees?

Q4: If no, can we show a lower bound on the leakage?

Multi user setting

EKVS is L-secure
with prob at least 1 - negl(k)


Efficiency
Crypto
Dis. Systems
Leakage
Consistency
???
Cryptographic Dis. Systems

- Acknowledgements
  - Archita Agarwal, MongoDB
  - Tarik Moataz, MongoDB

- References
  - *Encrypted Distributed Storage Systems (thesis)*, A. Agarwal
  - *Encrypted Distributed Hash Tables*, A. Agarwal, S. Kamara
  - *Encrypted Key Value Stores*, A. Agarwal, S. Kamara


Thank You
Vinaka
köszönöm
Kiitos
Maake
Asante
Shukria
Dhanyavadagalu
감사합니다
Dankscheen
спасибо
Kam Sah Hammida
Manana
Dankon
Terima Kasih
Matondo
Dank Je
Mauruuru
Biyan
Diolch i Chi
Taiku
Tack
Blagodaram
Ngiyabonga
Dziekuje
Chokrane
Grazie
Juspaxar
Arigato
Mochchakkeram
Gracias
Khap
Tingki
நன்றி
Bedankt
Dakujem
धन्यवाद
cảm ơn bạn
Paldies
Gratias Tibi
Ua Tsaug Rau Koj
Grazas
Obrigado
Děkuji
Nirringrazzjak
Di Ou Mèsi
Kia Ora
Kop Khun
ありがとう
Suksama
Rahmat
Matur Nuwun
谢谢
хвала
Hvala
Welalin
Djiere Dieuf
Misaotra
Danke
Merci
Go Raibh Maith Agat
Eskerrik Asko
Salamat
ขอบคุณ
Najis Tuke