# Privacy as a Service

Raymond Cheng

# Build practical cloud services that protect user privacy from powerful threats


Secure | https://www.cyberriskanalytics.com
Cyber Risk Analytics
ABOUT
PLATFORM
STATISTICS
FEATURES
CONTACT
Request Demo
Login
Data Breach Statistics
Cyber Risk Analytics is derived from a proprietary search engine and the thorough analysis of thousands of reported data breach incidents and the metrics driving cyber exposures.
124
Breaches YTD 2017
23,828
Breaches of All Time
1,446,678,984
Compromised Emails
2,477,186
Records YTD 2017
9,250,909,59
Records of All Time
84,120
Organizations Monitored

# Powerful Threats to User Privacy

**Organized Crime**

**Nation-State Actors**

# Powerful Threats to User Privacy

**Organized Crime**

**Nation-State Actors**

Gather Intelligence

Covert Surveillance

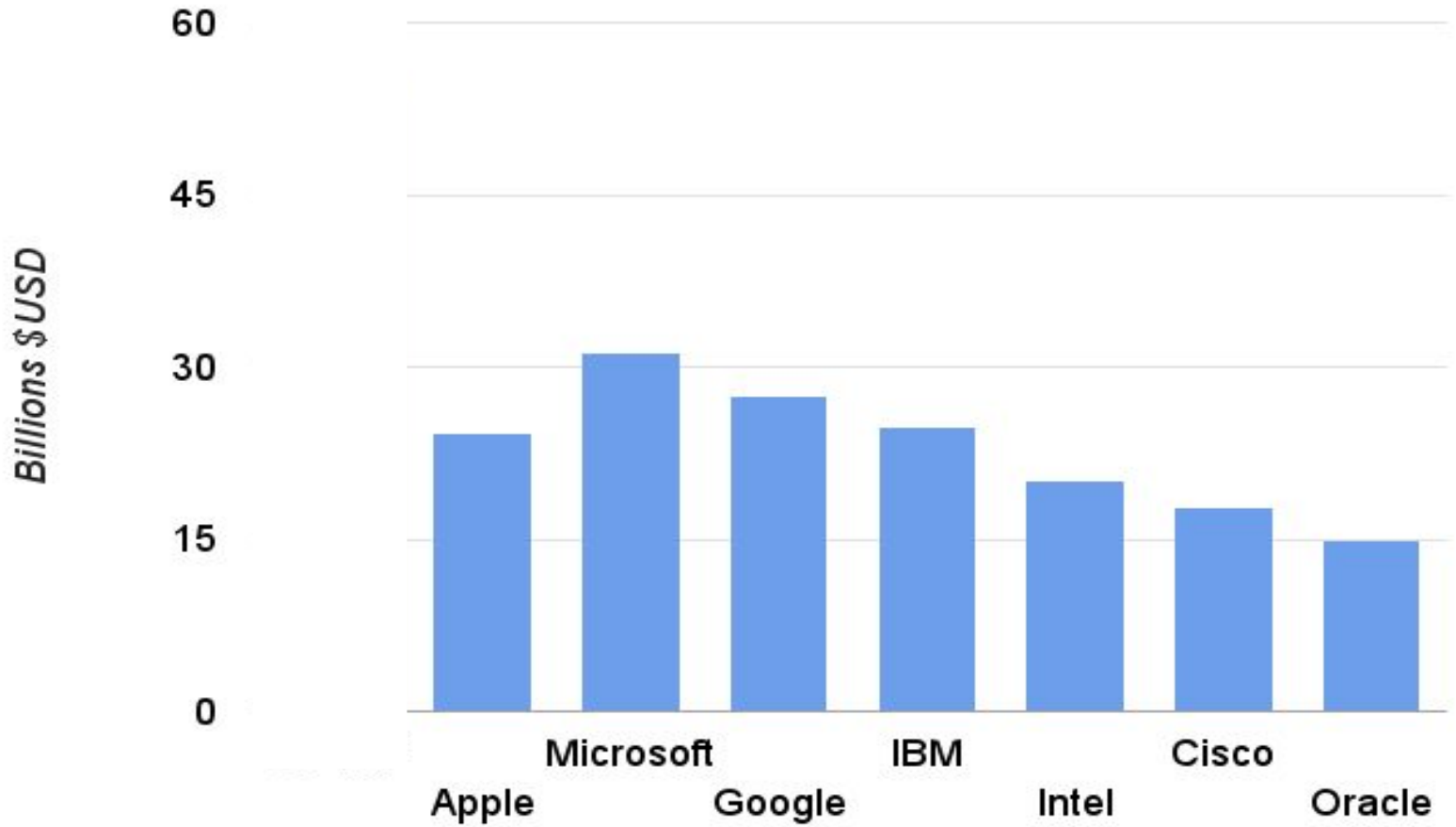
Cyberwarfare

Corporate Espionage

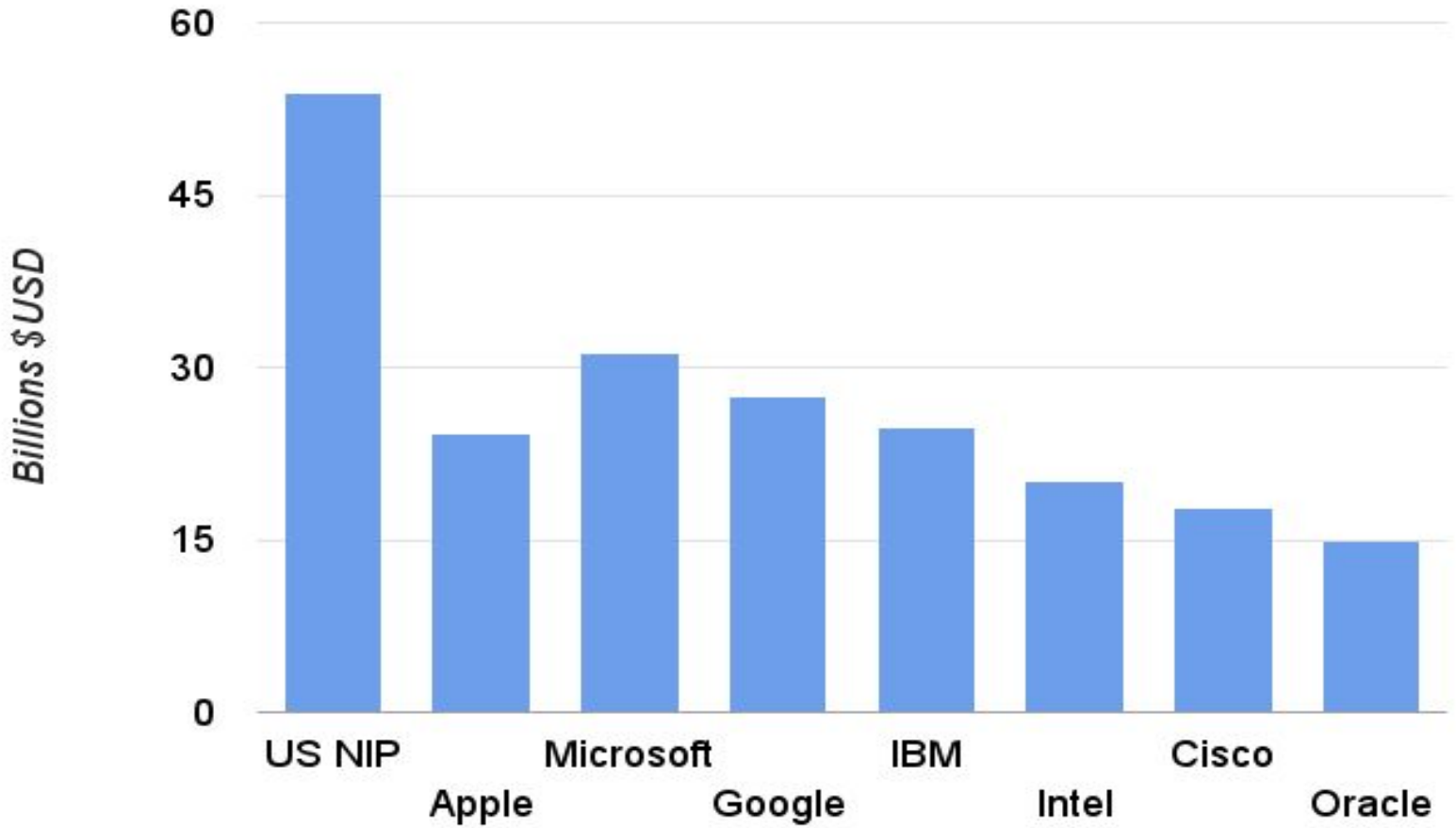
Influence Politics

Censor content . . .

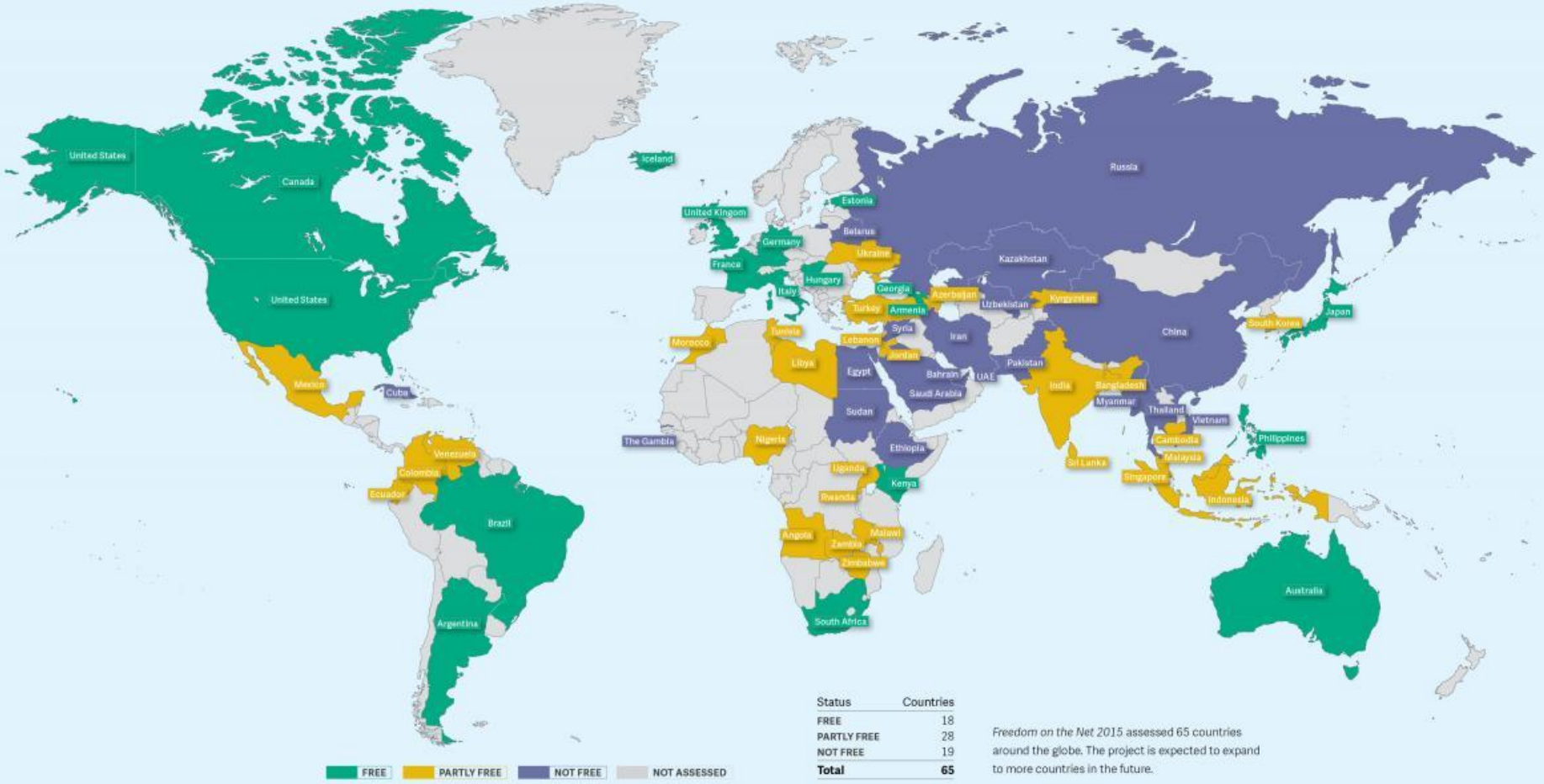
# Annual Operating Expenses



# Annual Operating Expenses

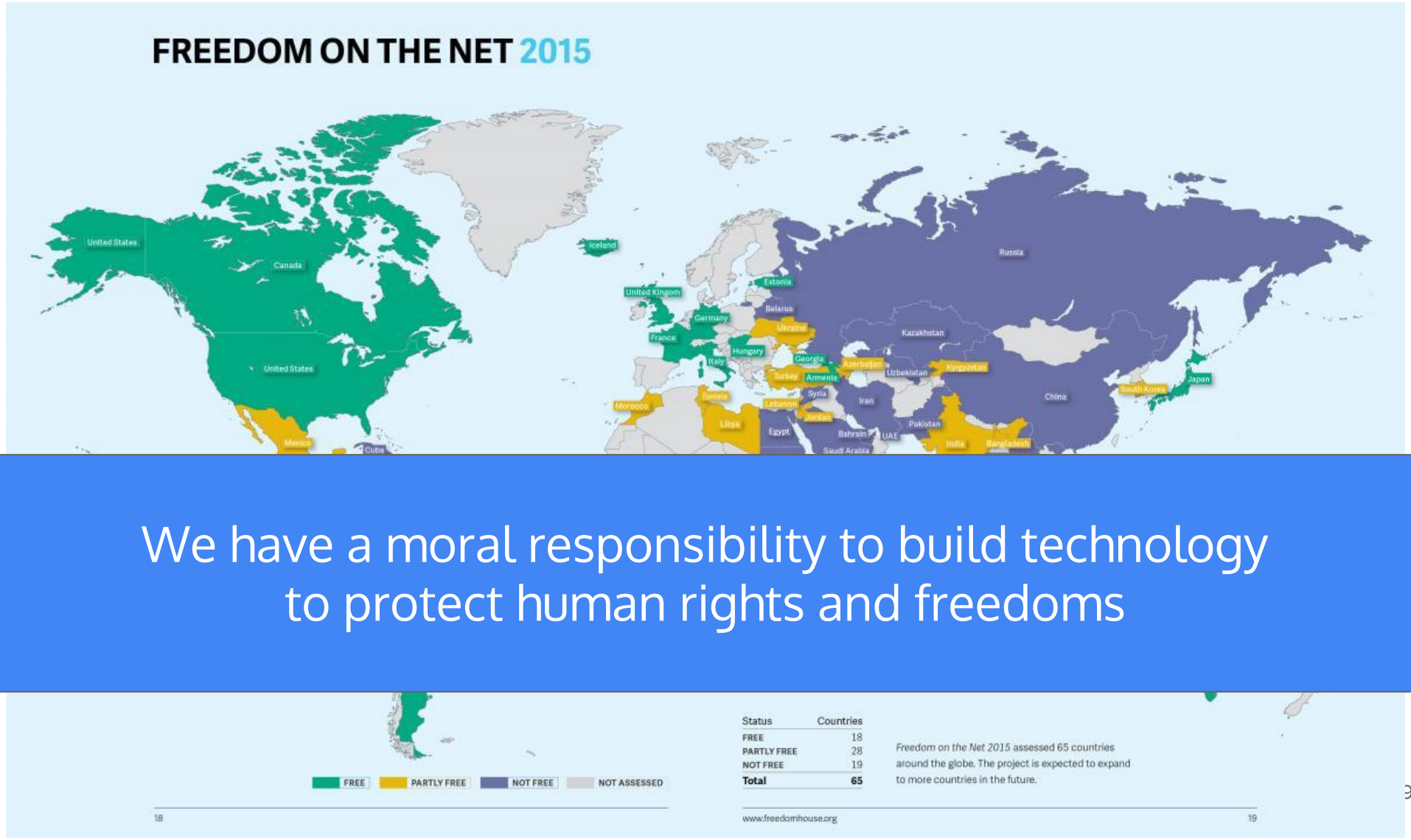


# FREEDOM ON THE NET 2015




| Status | Countries |
|---|---|
| FREE | 18 |
| PARTLY FREE | 28 |
| NOT FREE | 19 |
| **Total** | **65** |

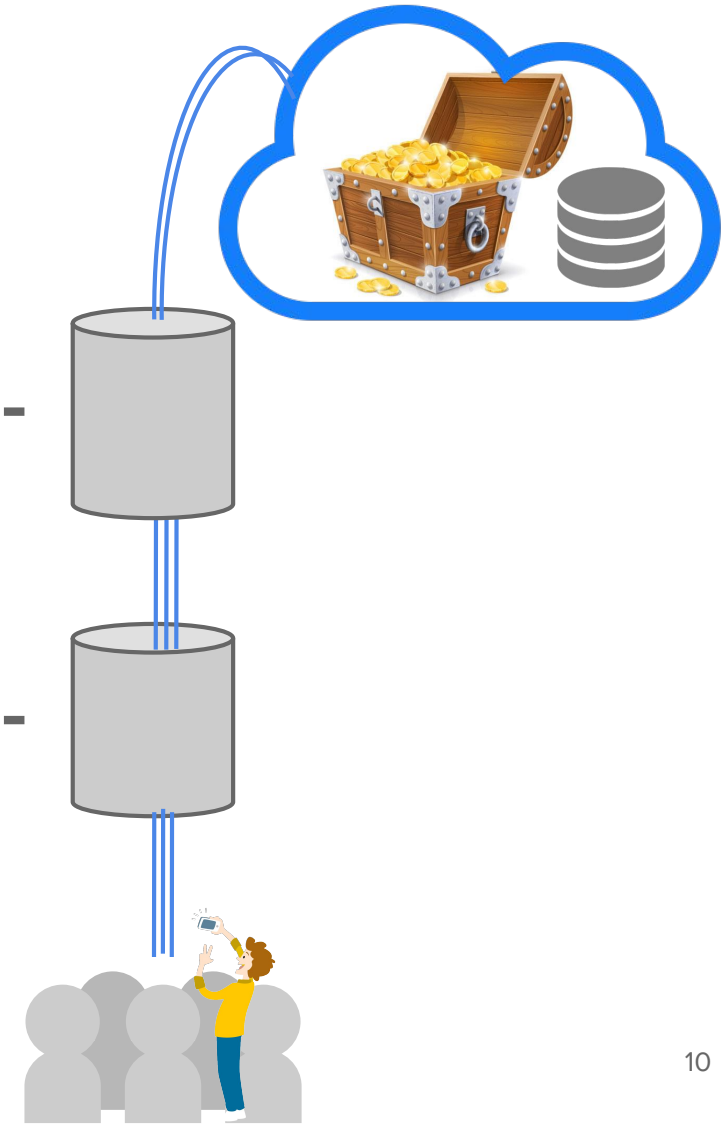
*Freedom on the Net 2015* assessed 65 countries around the globe. The project is expected to expand to more countries in the future.




We have a moral responsibility to build technology to protect human rights and freedoms

# Threat Model

**Cloud**



**Network**

**Clients**

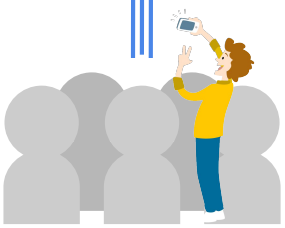
# Networks are vulnerable

**Cloud**

**Malicious Network**

Censorship, surveillance, misdirection

**Clients**



# Cloud services are routinely hacked

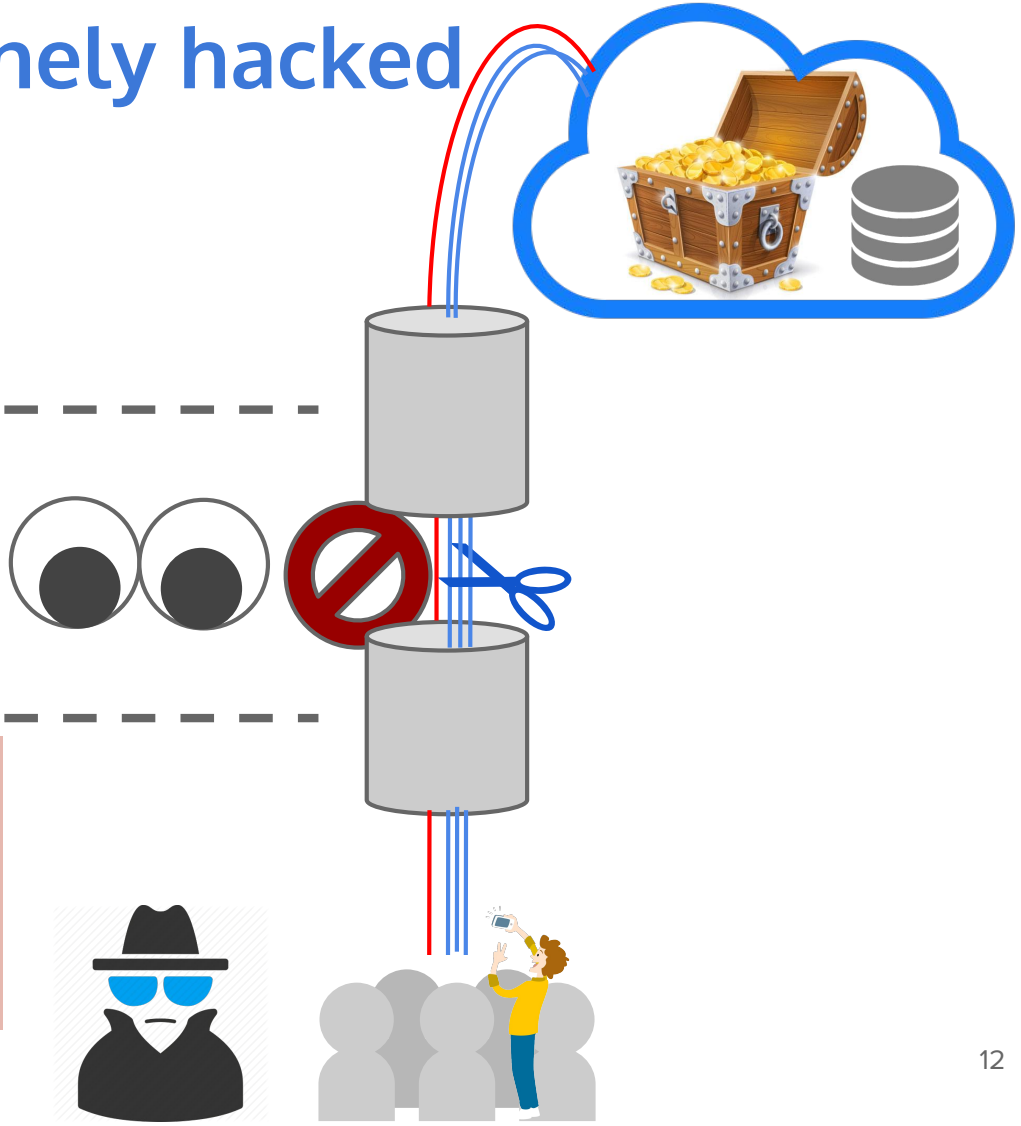
**Cloud**

**Malicious Network**

Censorship, surveillance, misdirection

**Malicious Clients**

Hackers



# Governments can compel cooperation

**Malicious Cloud**

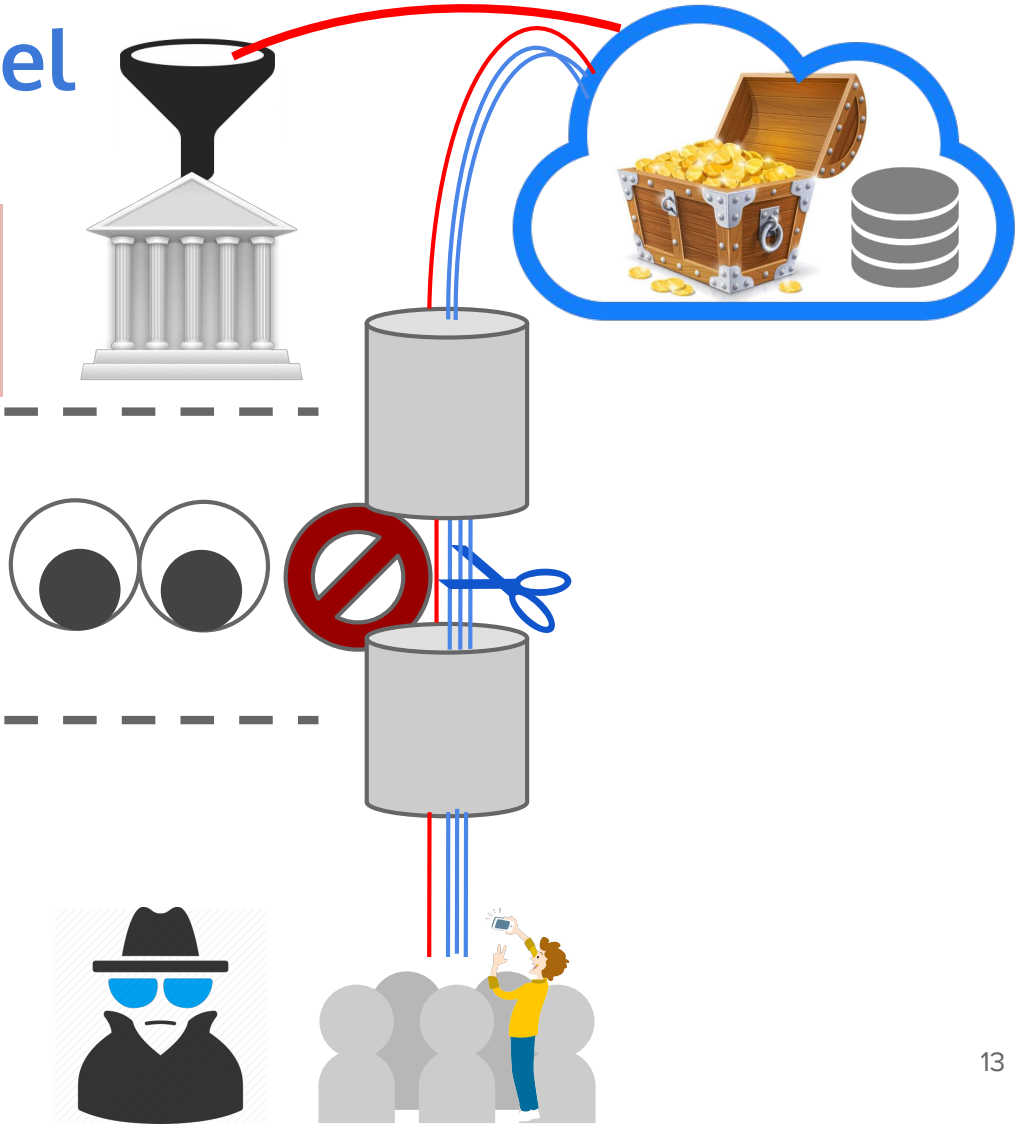
Data requests, surveillance, control

**Malicious Network**

Censorship, surveillance, misdirection

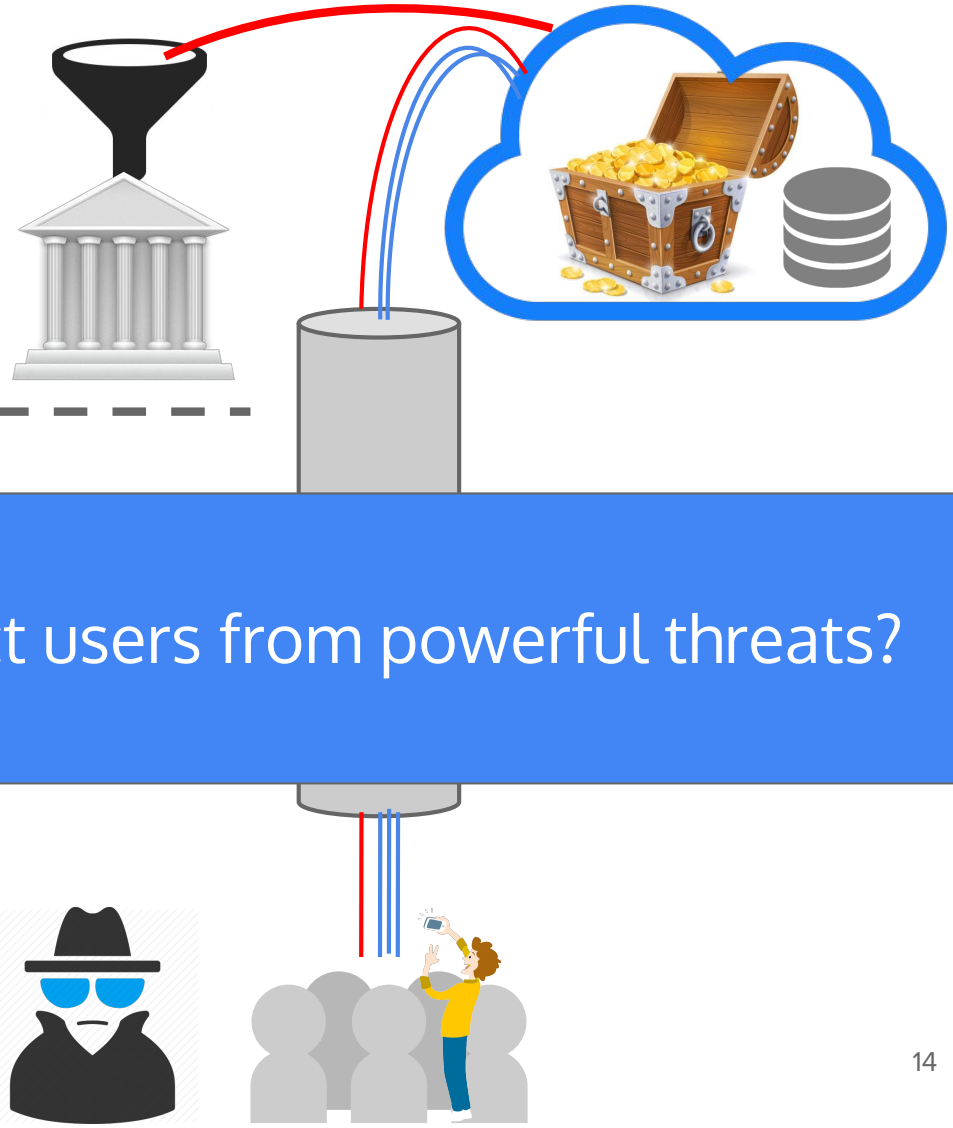
**Malicious Clients**

Hackers



Malicious Cloud

Malicious Network



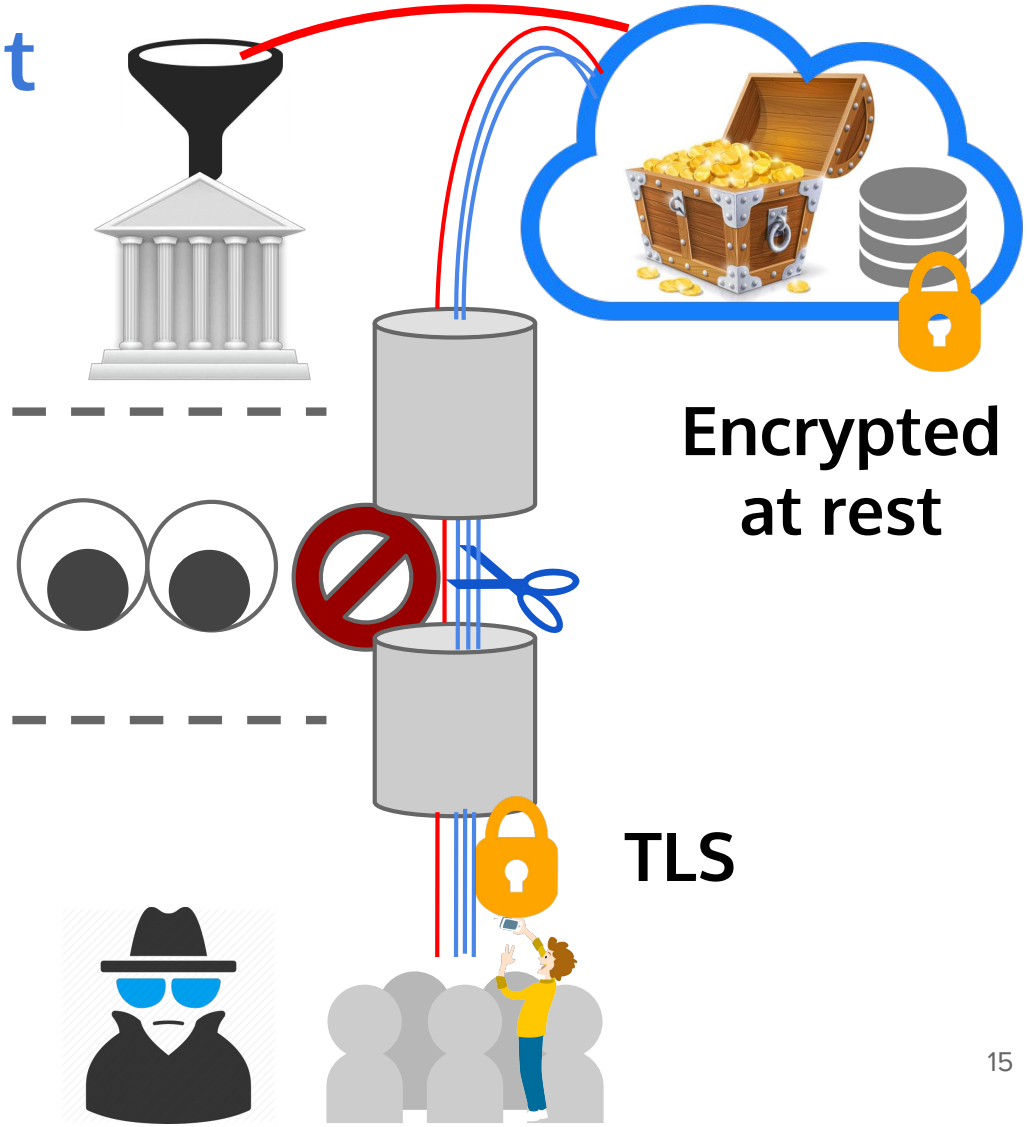
What security model can protect users from powerful threats?

# Encryption not sufficient

Malicious Cloud

Malicious Network

Malicious Clients



Encrypted at rest

TLS

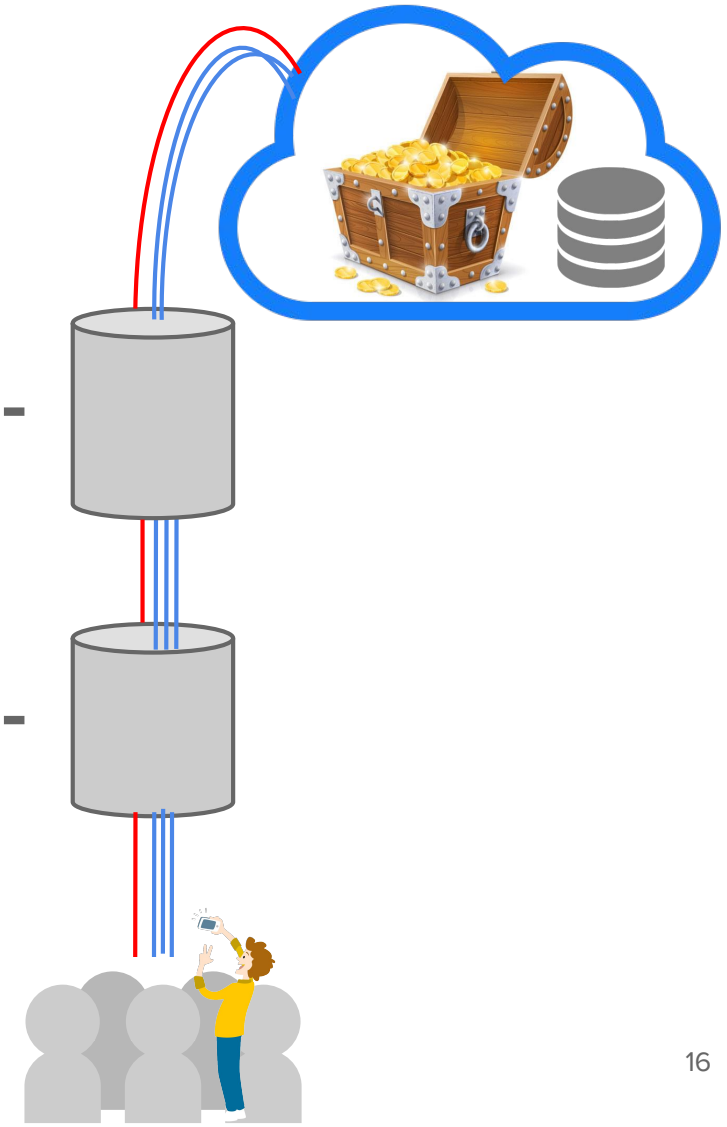
# Overview

**Malicious Network**

1. **uProxy** - censorship circumvention

**Malicious Clients**

2. **Radiatus** - harden web applications from external intrusion



# Overview

**Malicious Cloud**

3. **Oblivious Cloud Services**

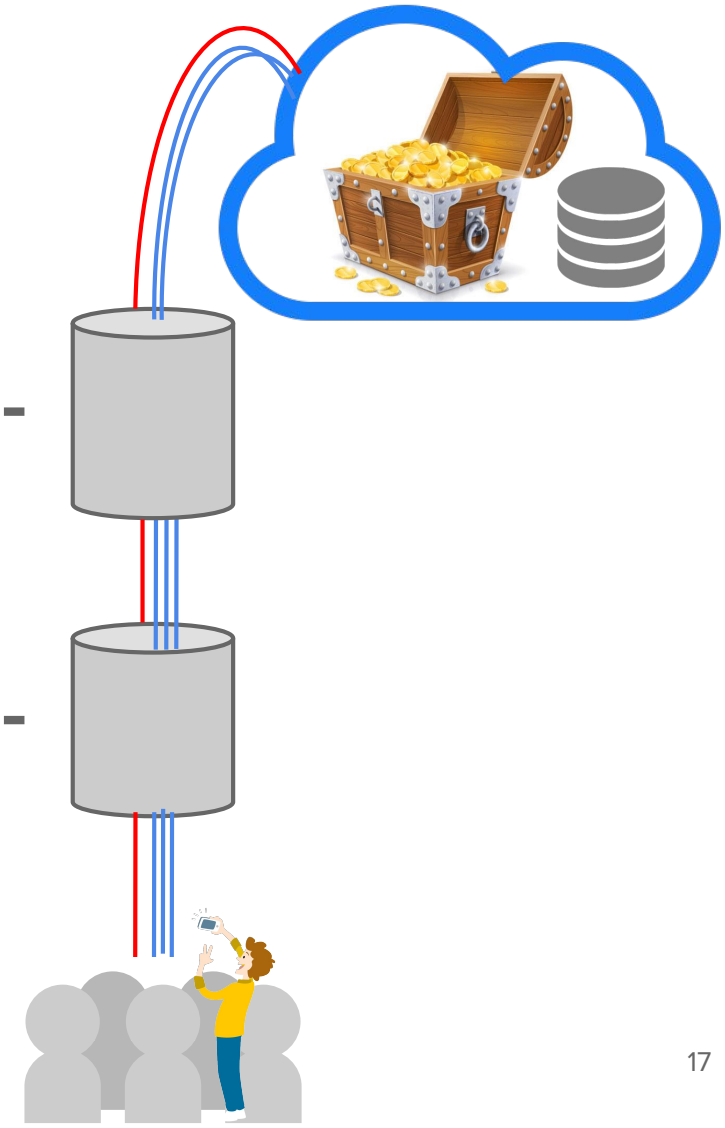
**Talek** - private publish-subscribe

**Malicious Network**

1. **uProxy** - censorship circumvention

**Malicious Clients**

2. **Radiatus** - harden web applications from external intrusion



# Overview

## Malicious Cloud

3. **Oblivious Cloud Services**

**Talek** - private publish-subscribe

*(Cheng, Scott, Parno, Zhang, Krishnamurthy, Anderson, 2016)*

## Malicious Network

1. **uProxy** - censorship circumvention

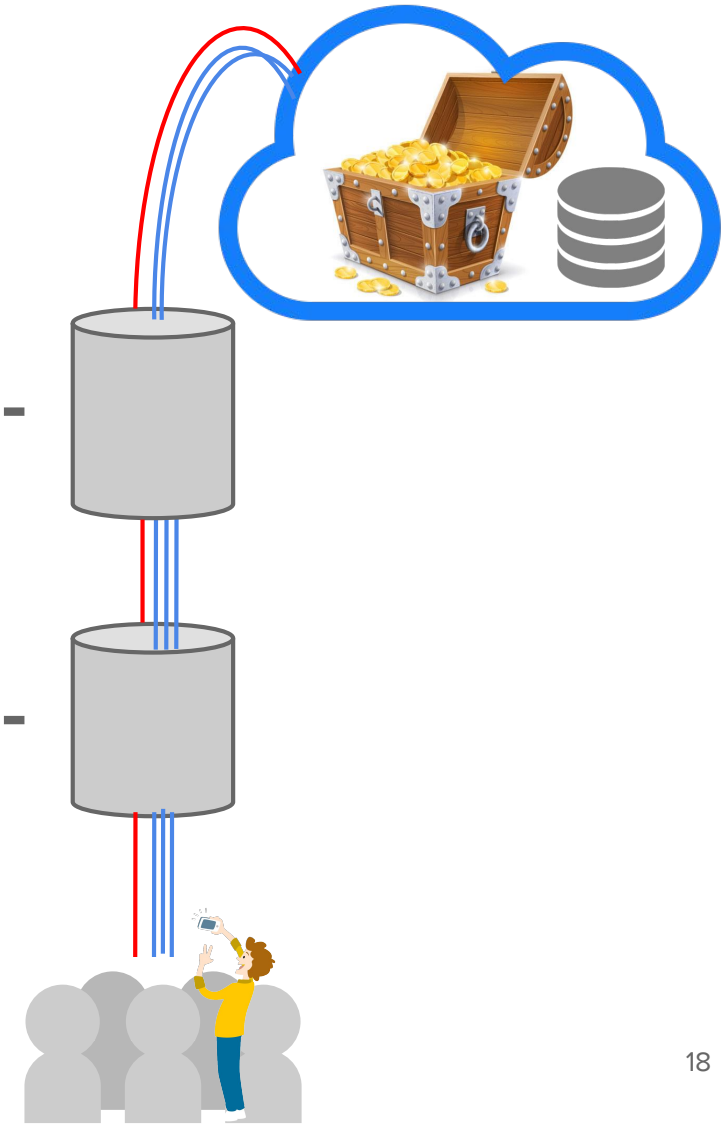
Deployed to thousands over the world

*(Cheng, Scott, Dixon, Krishnamurthy, Anderson, 2016)*

## Malicious Clients

2. **Radiatus** - harden web applications from external intrusion

*(Cheng, Scott, Ellenbogen, Howell, Roesner, Krishnamurthy, Anderson, 2016)*



# Collaborators

UNIVERSITY of WASHINGTON

Tom Anderson

Arvind Krishnamurthy

Franzi Roesner

Jon Howell — Google, Microsoft Research

Bryan Parno — Carnegie Mellon, Microsoft Research

Lucas Dixon — Google, Jigsaw

Will Scott — NYU, Tor

**Students:**

Irene Zhang
Paul Ellenbogen
Elizabeth Wei
Bonnie Pan
Nick Martindell
Tariq Yusuf
Caylan Lee
Nicholas Shahan

# Overview

**Malicious Cloud**

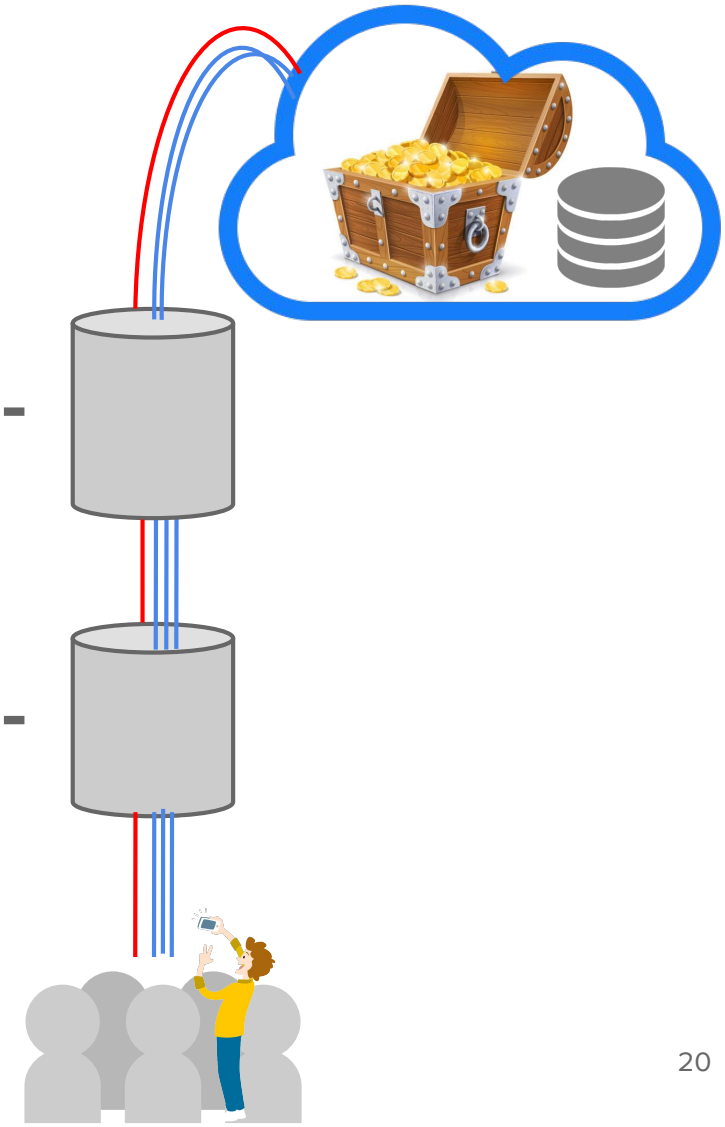
3. **Oblivious Cloud Services**
**Talek** - private publish-subscribe

**Malicious Network**

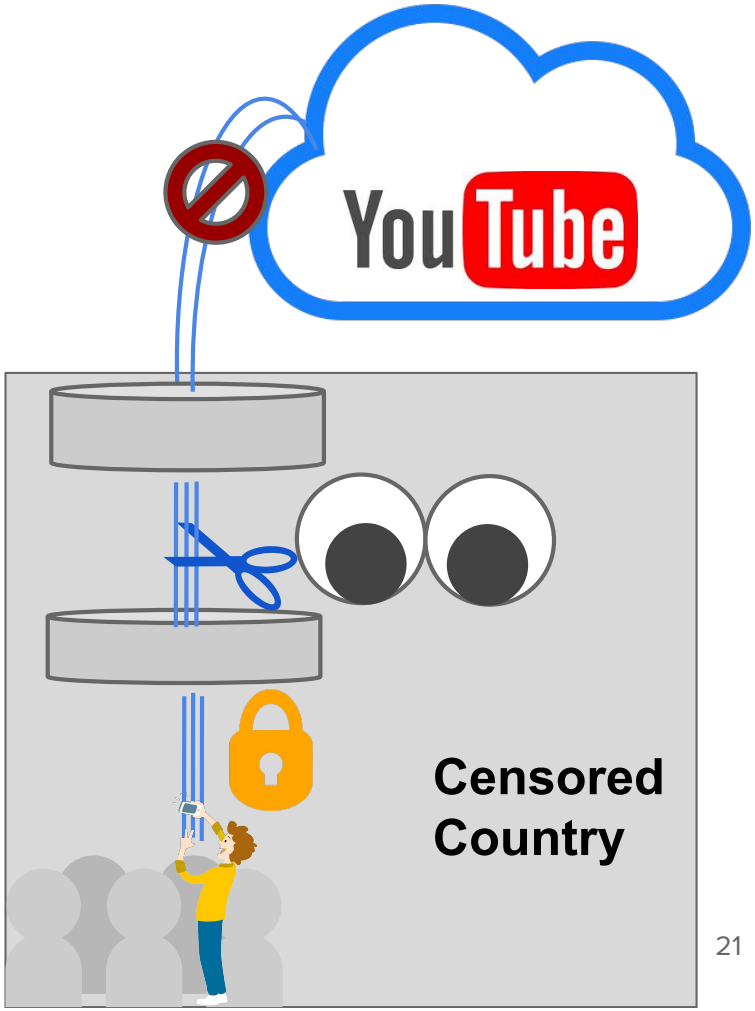
1. **uProxy** - censorship circumvention

**Malicious Clients**

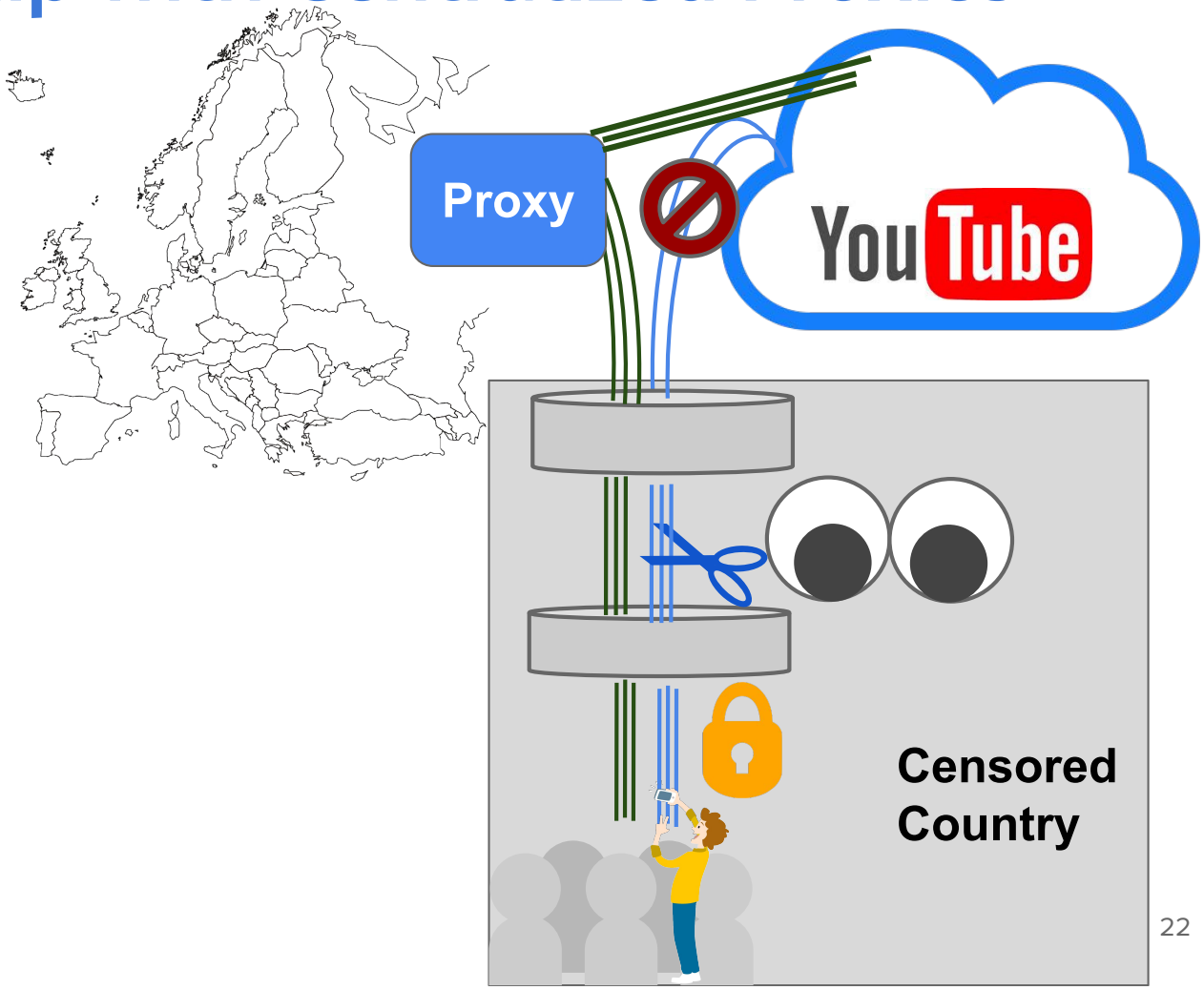
2. **Radiatus** - harden web applications from external intrusion



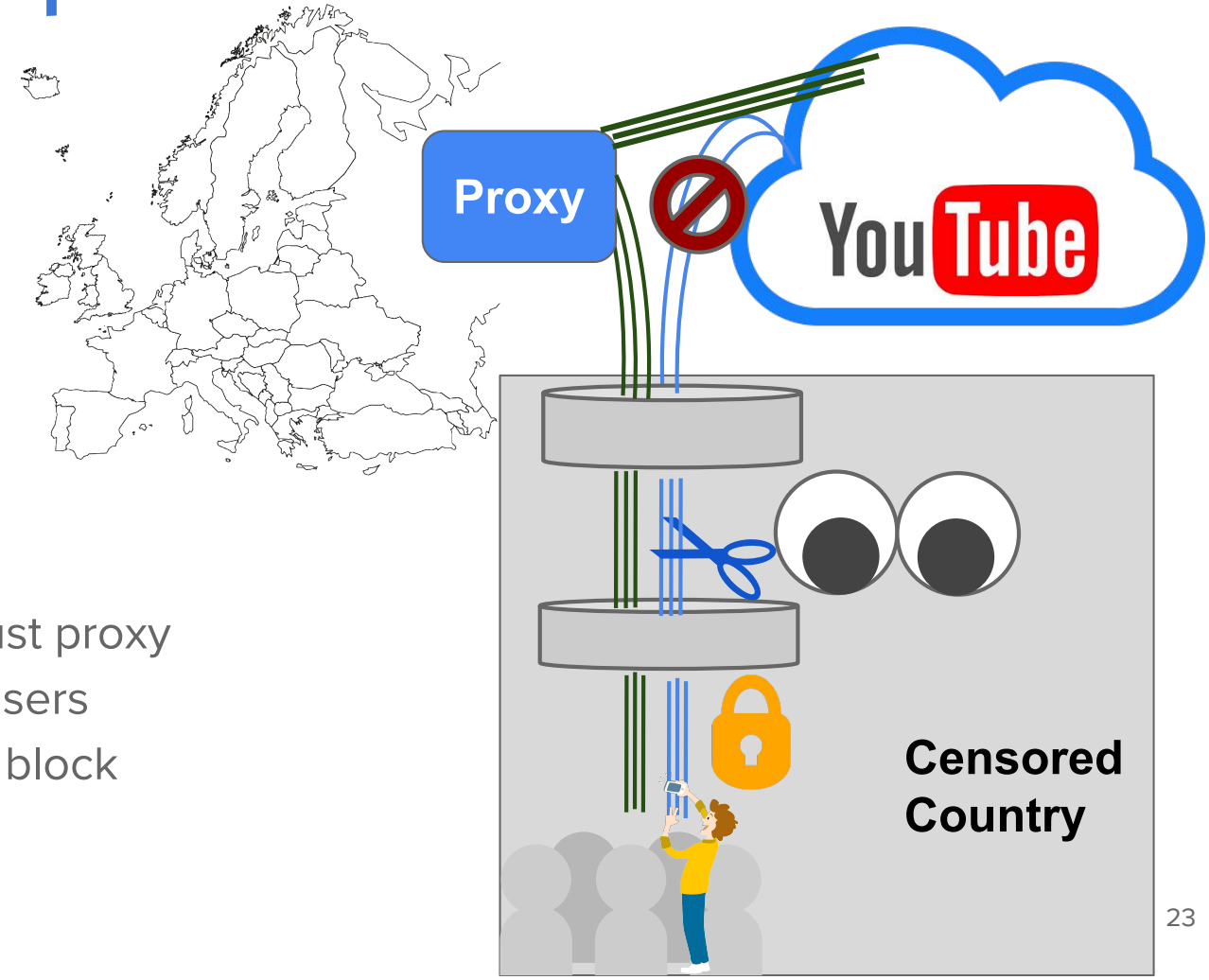
# Internet Censorship is a Pervasive Problem

# Evading Censorship with Centralized Proxies



# Evading Censorship with Centralized Proxies

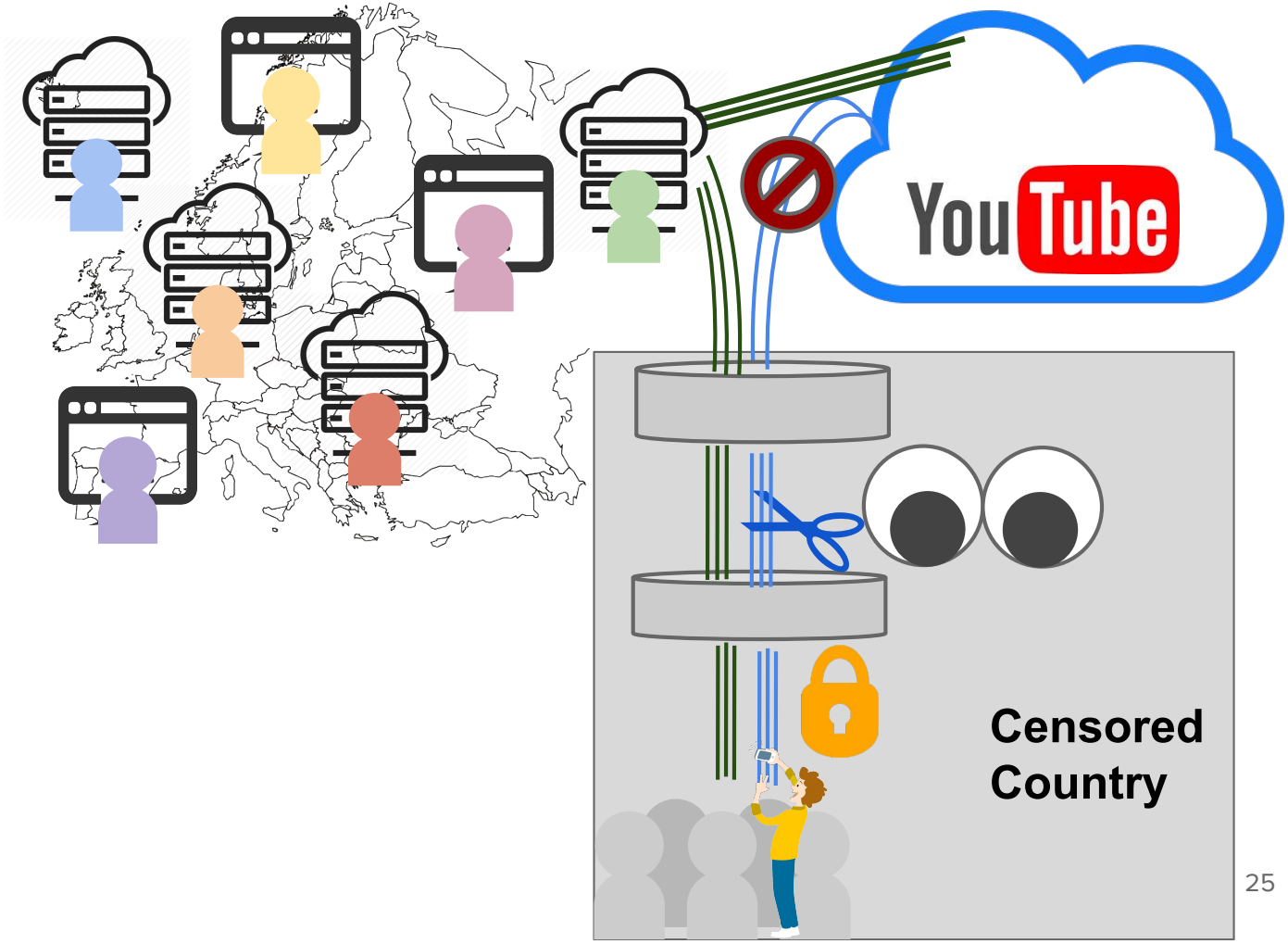



**Problem with Centralized Proxies**

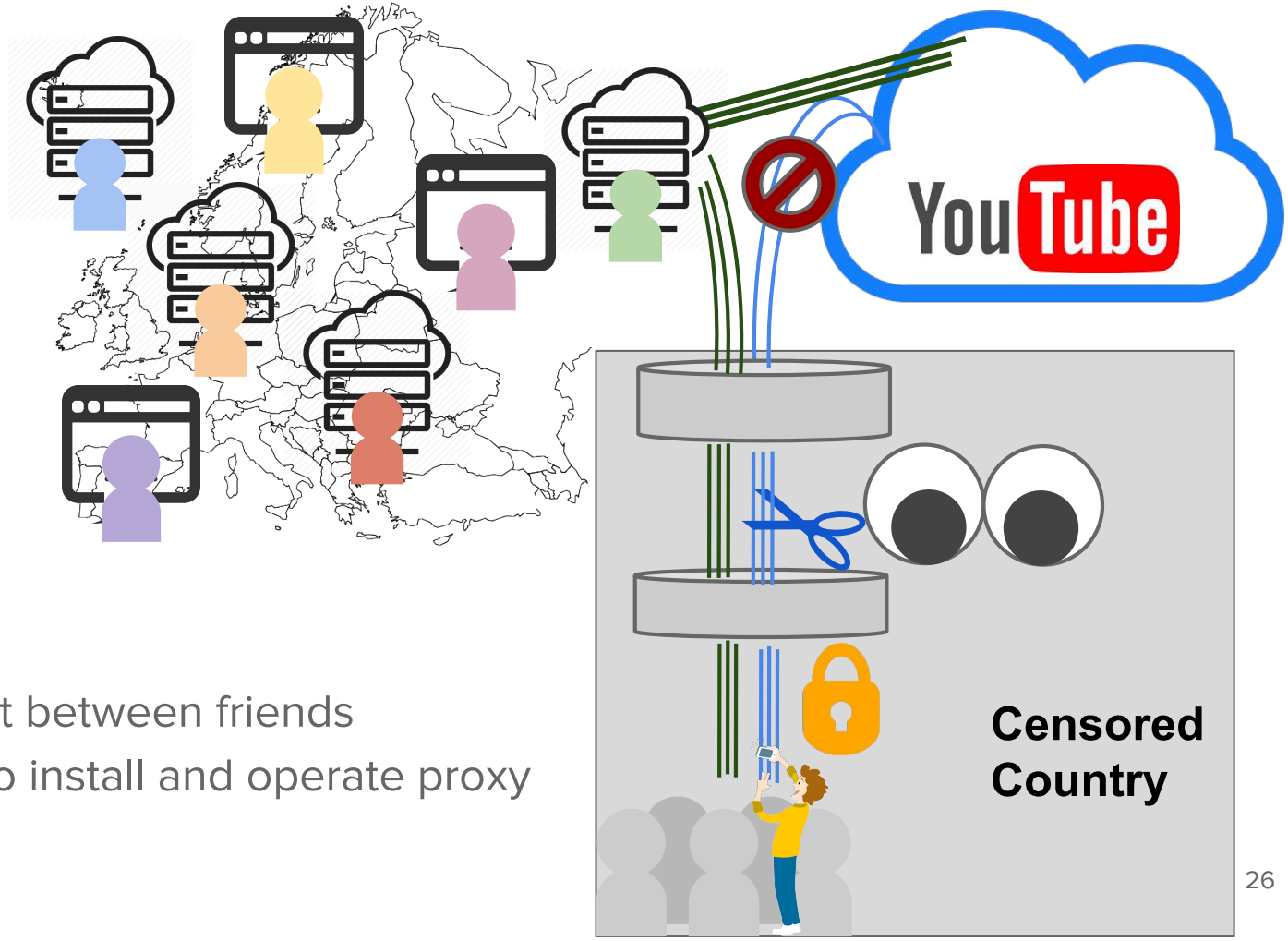
- Trust: users need to trust proxy
  proxy needs to trust users
- Scale: easy to find and block


RIP
34
13
17
MOLES ##!!!
?

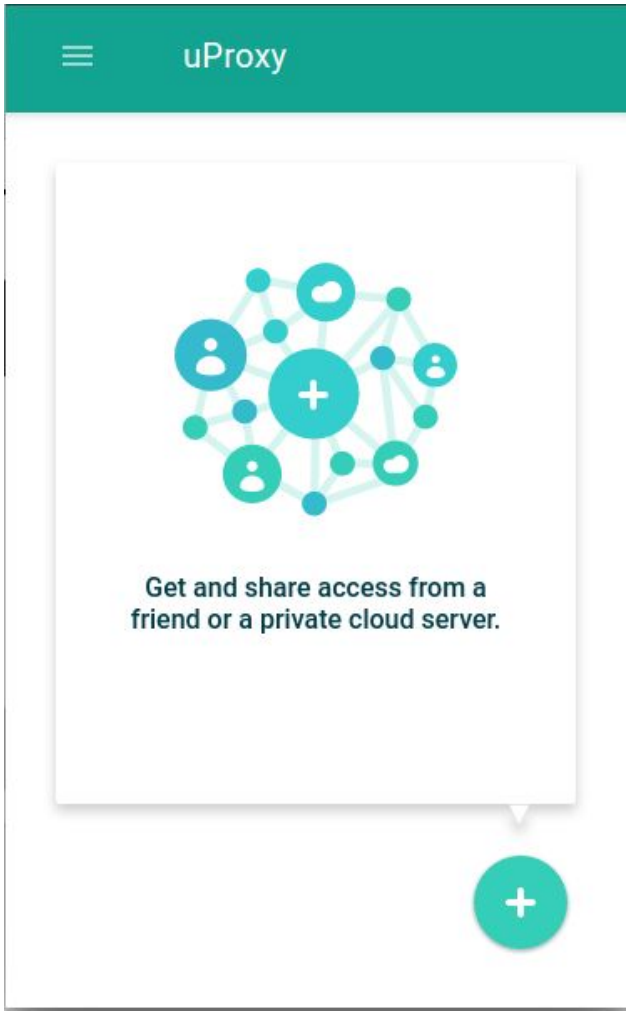
# Do-It-Yourself Censorship Circumvention

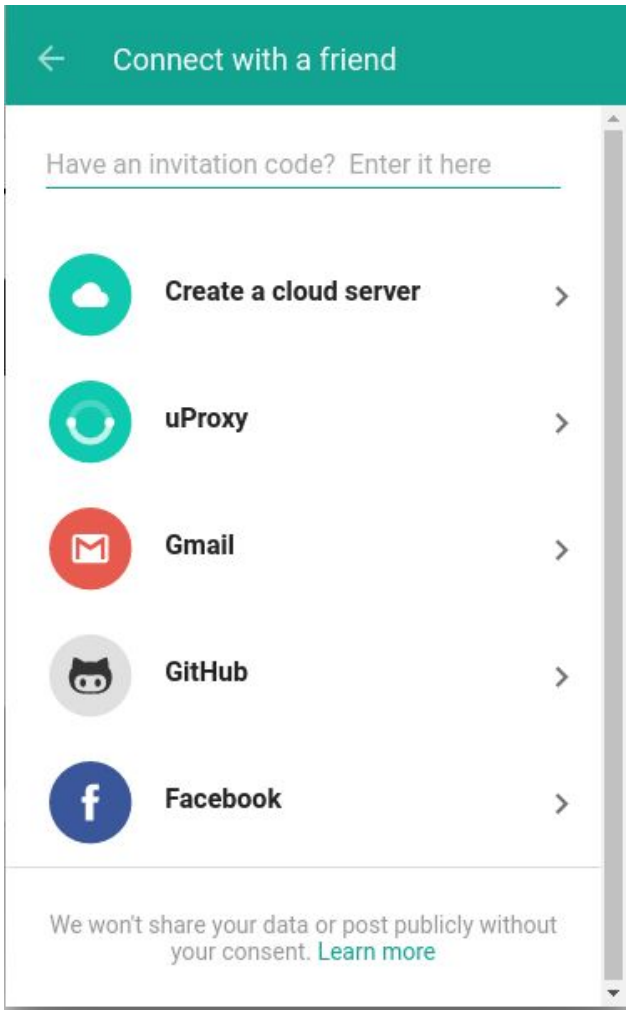


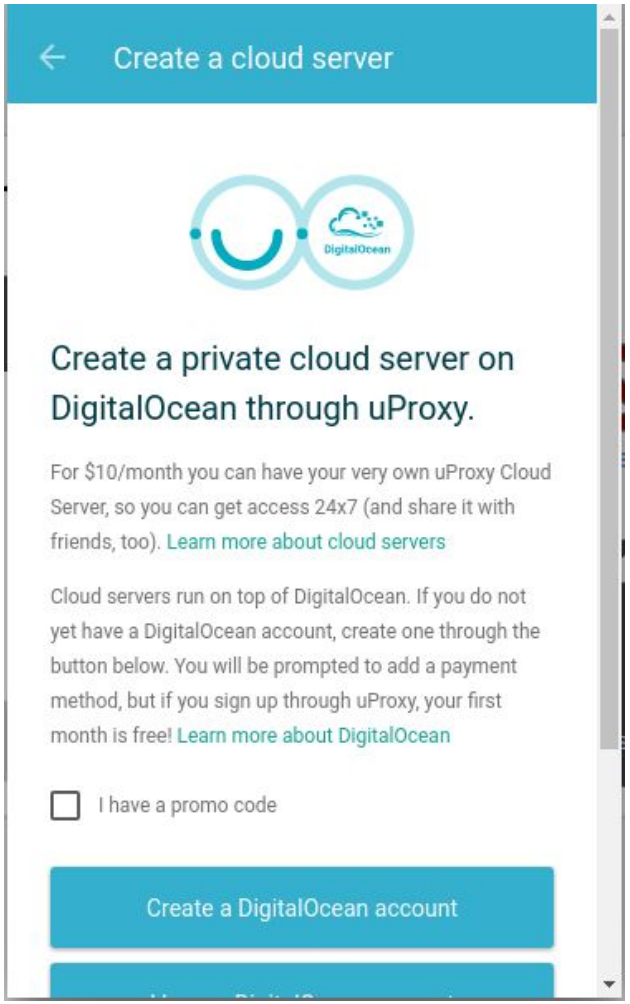
# Do-It-Yourself Censorship Circumvention




- Trust: Explicit consent between friends
- Scale: Trivially easy to install and operate proxy

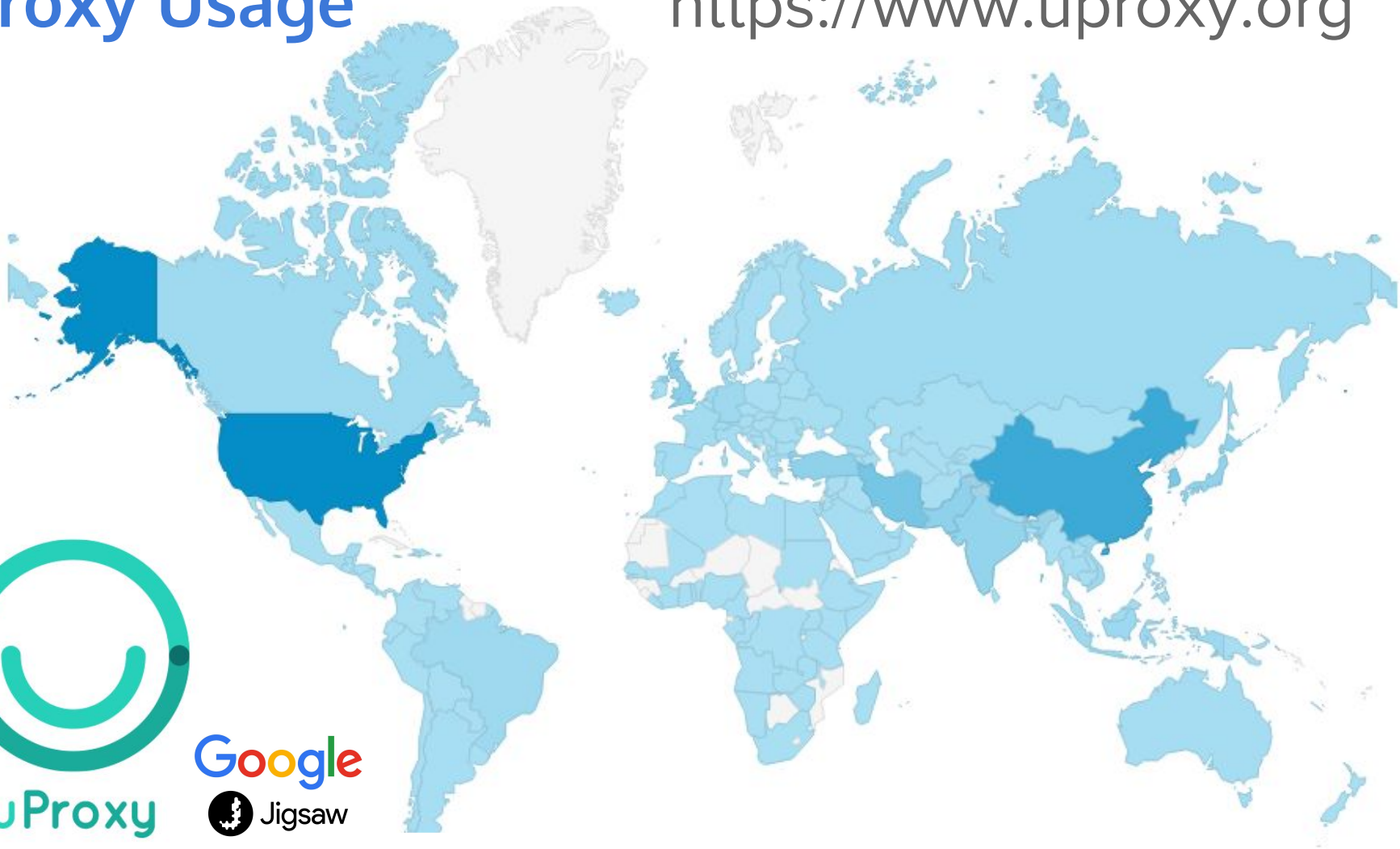

uProxy
Get and share access from a friend or a private cloud server.


Connect with a friend
Have an invitation code? Enter it here
Create a cloud server
uProxy
Gmail
GitHub
Facebook
We won't share your data or post publicly without your consent. Learn more


Create a cloud server
DigitalOcean
Create a private cloud server on DigitalOcean through uProxy.
For $10/month you can have your very own uProxy Cloud Server, so you can get access 24x7 (and share it with friends, too). Learn more about cloud servers
Cloud servers run on top of DigitalOcean. If you do not yet have a DigitalOcean account, create one through the button below. You will be prompted to add a payment method, but if you sign up through uProxy, your first month is free! Learn more about DigitalOcean
I have a promo code
Create a DigitalOcean account

# uProxy Usage

https://www.uproxy.org



uProxy

Google Jigsaw

# Overview

**Malicious Cloud**

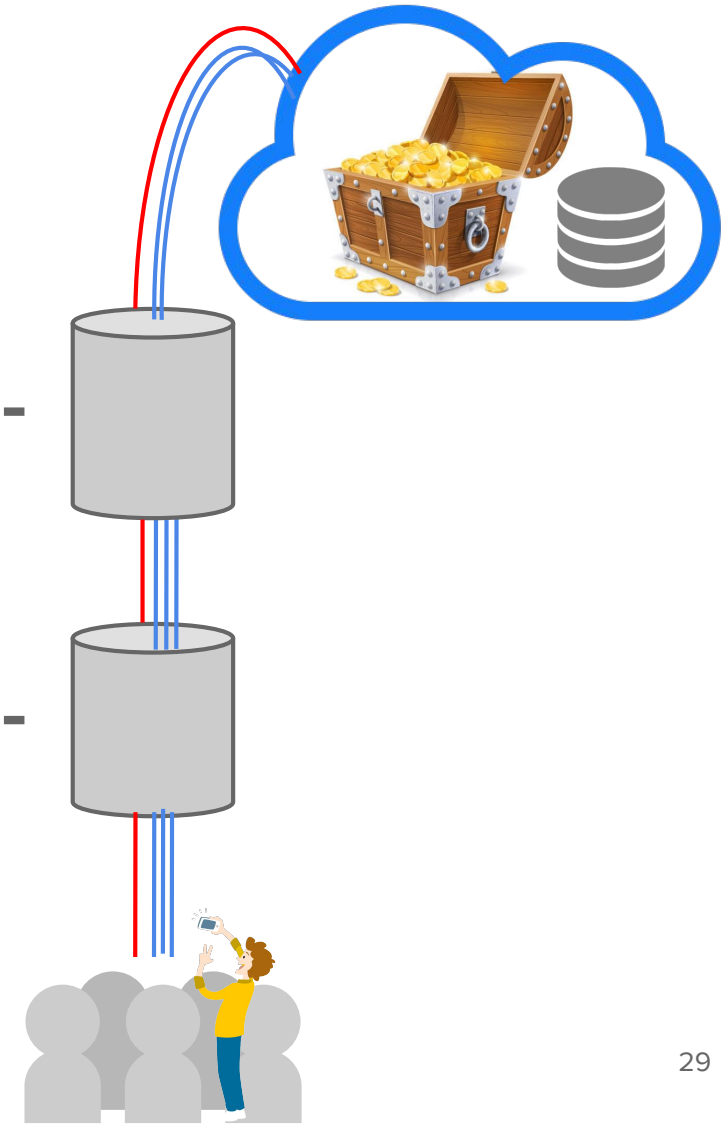
3. **Oblivious Cloud Services**
**Talek** - private publish-subscribe

**Malicious Network**

1. **uProxy** - censorship circumvention

**Malicious Clients**

2. **Radiatus** - harden web applications from external intrusion

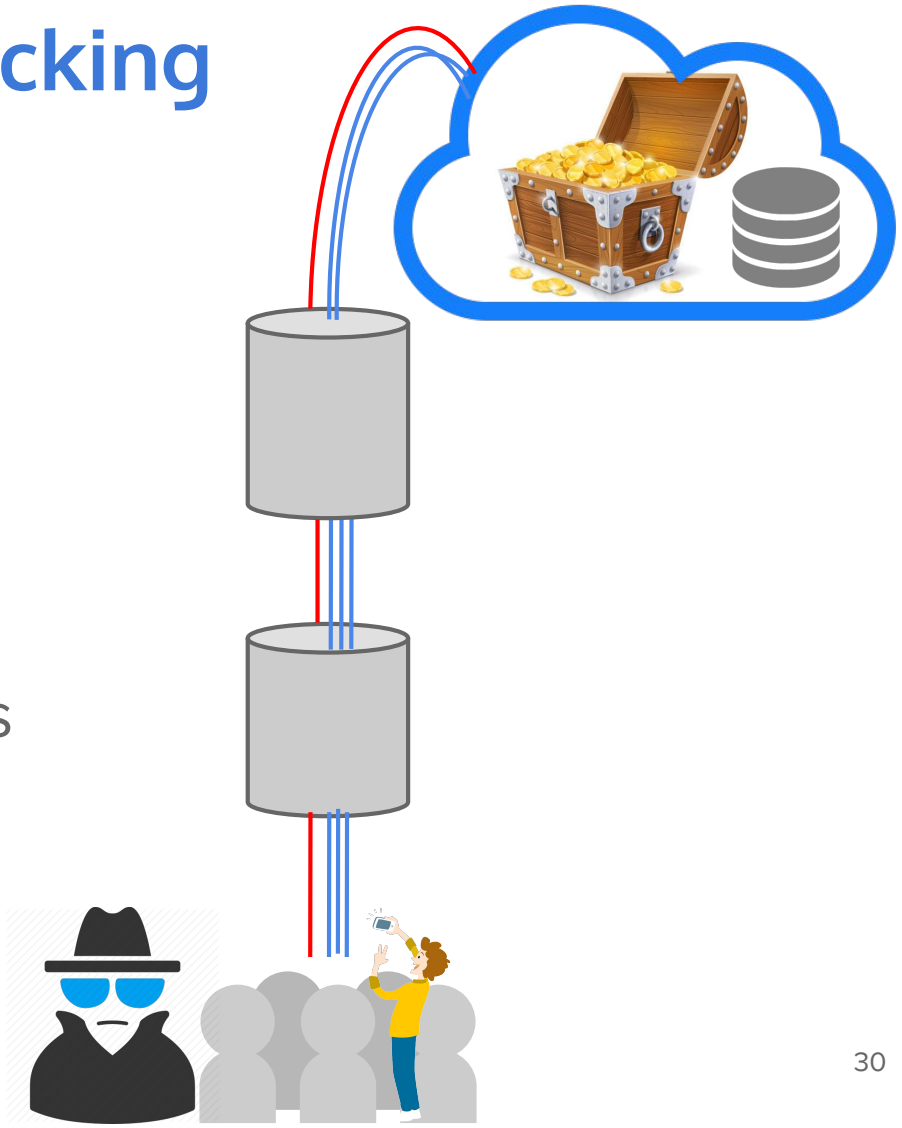


# Websites Vulnerable to Hacking

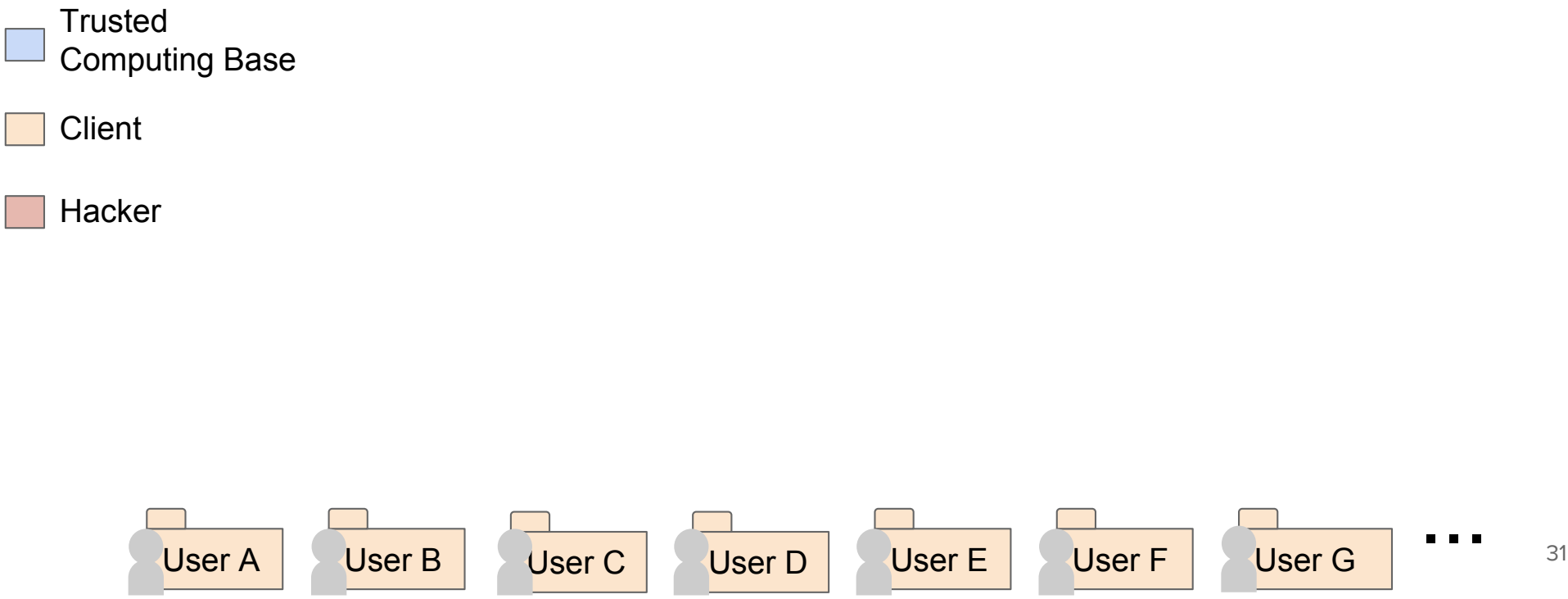
**Trust the cloud provider**

**Want to prevent external attacks**

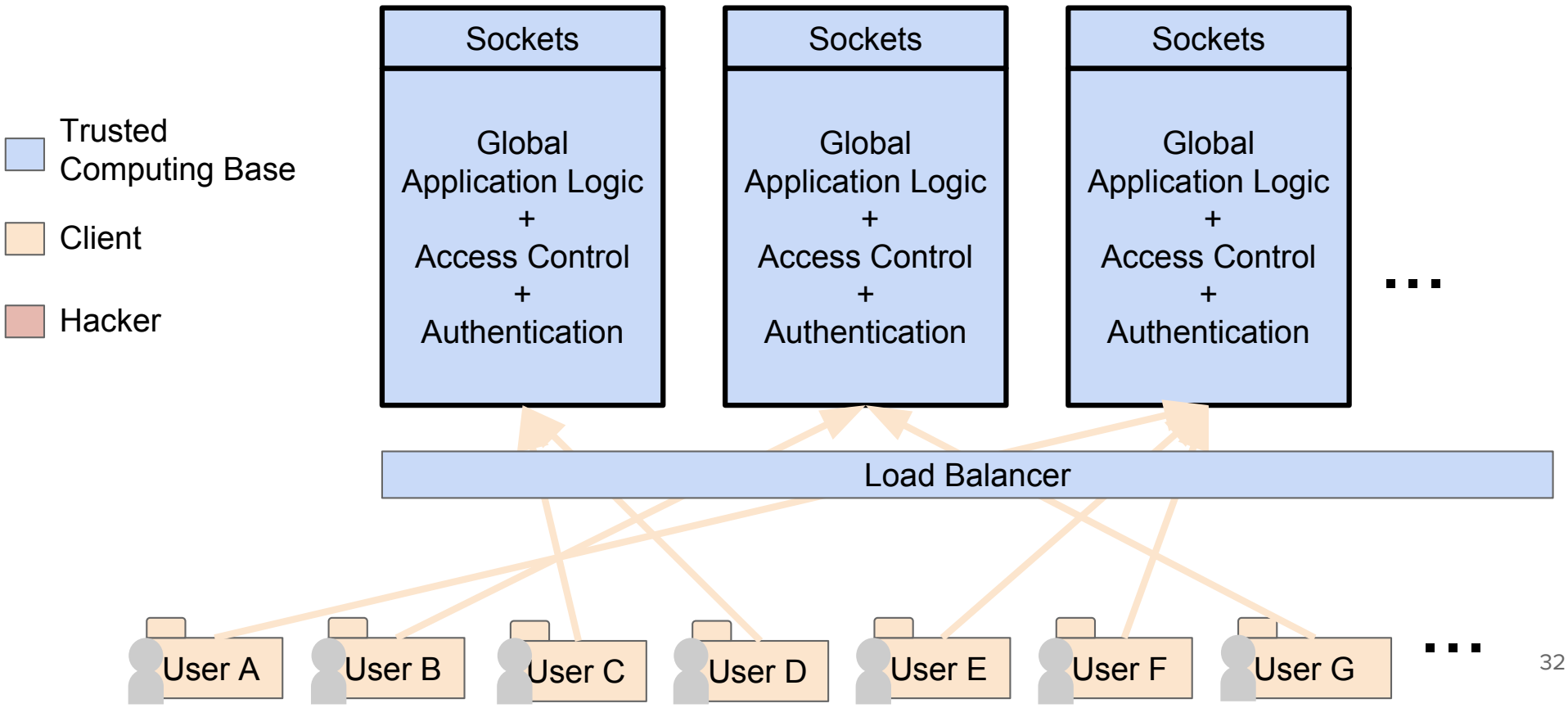
- Craft arbitrary network packets



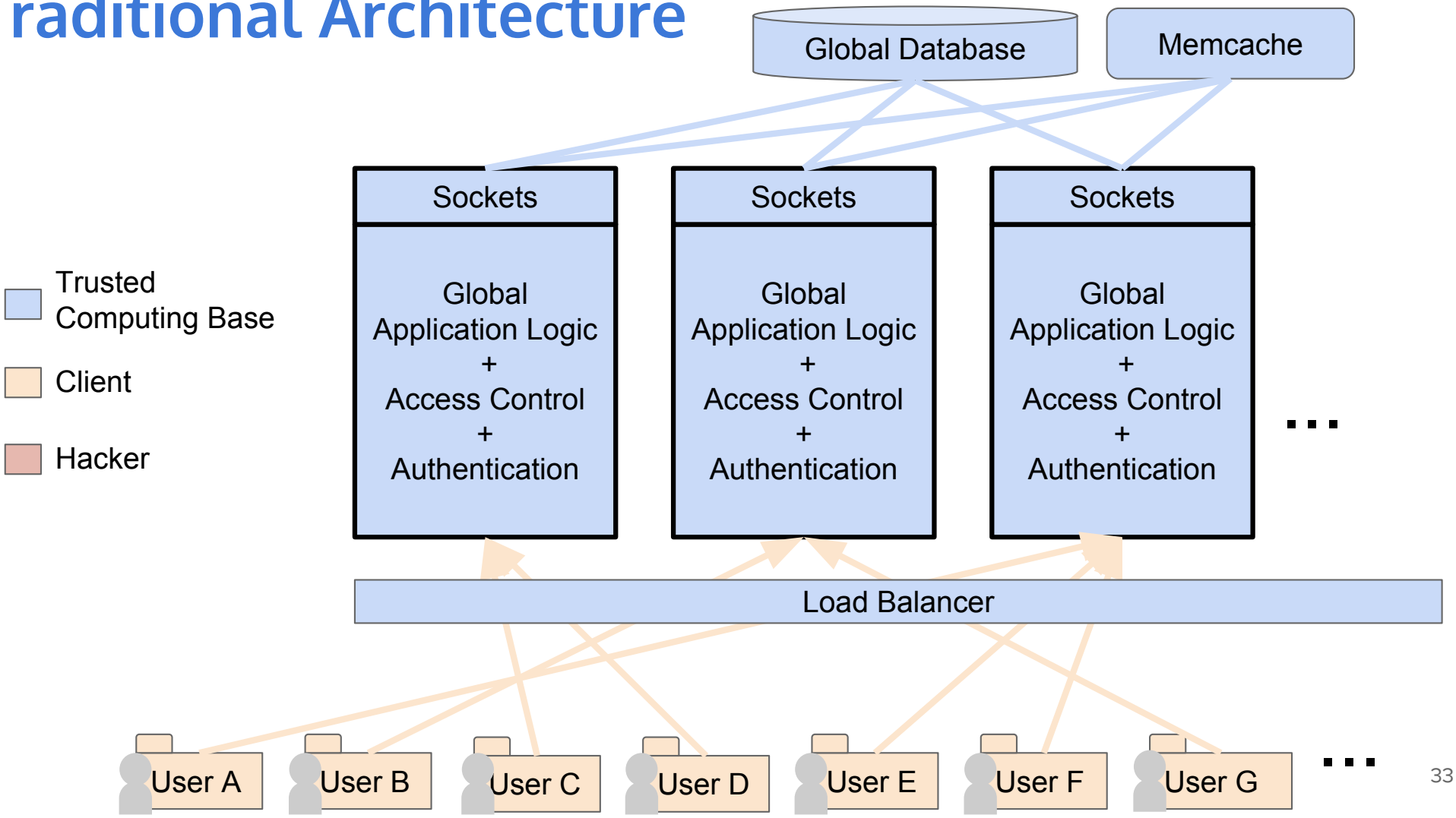
# Traditional Architecture



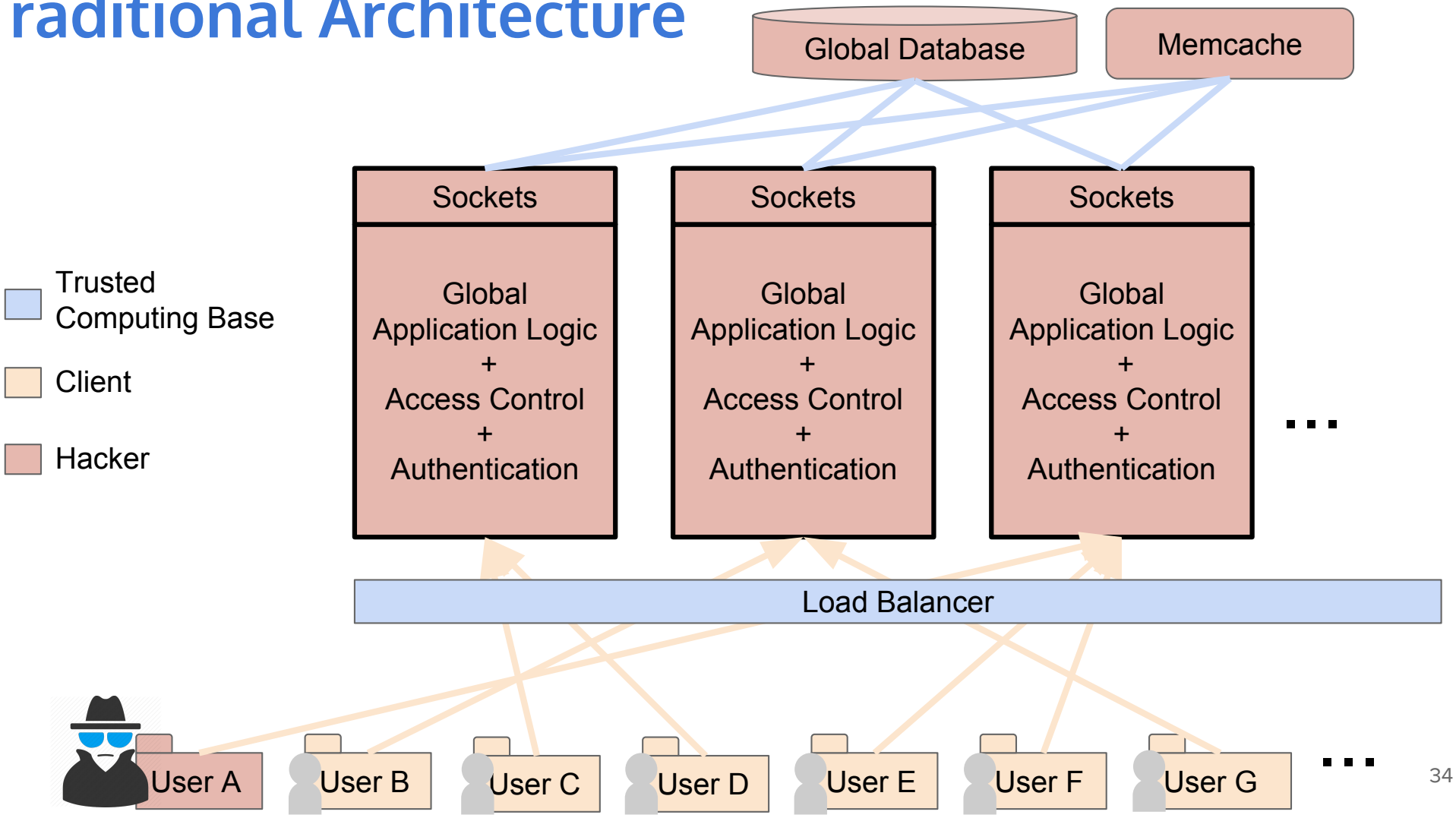
# Traditional Architecture



# Traditional Architecture



# Traditional Architecture

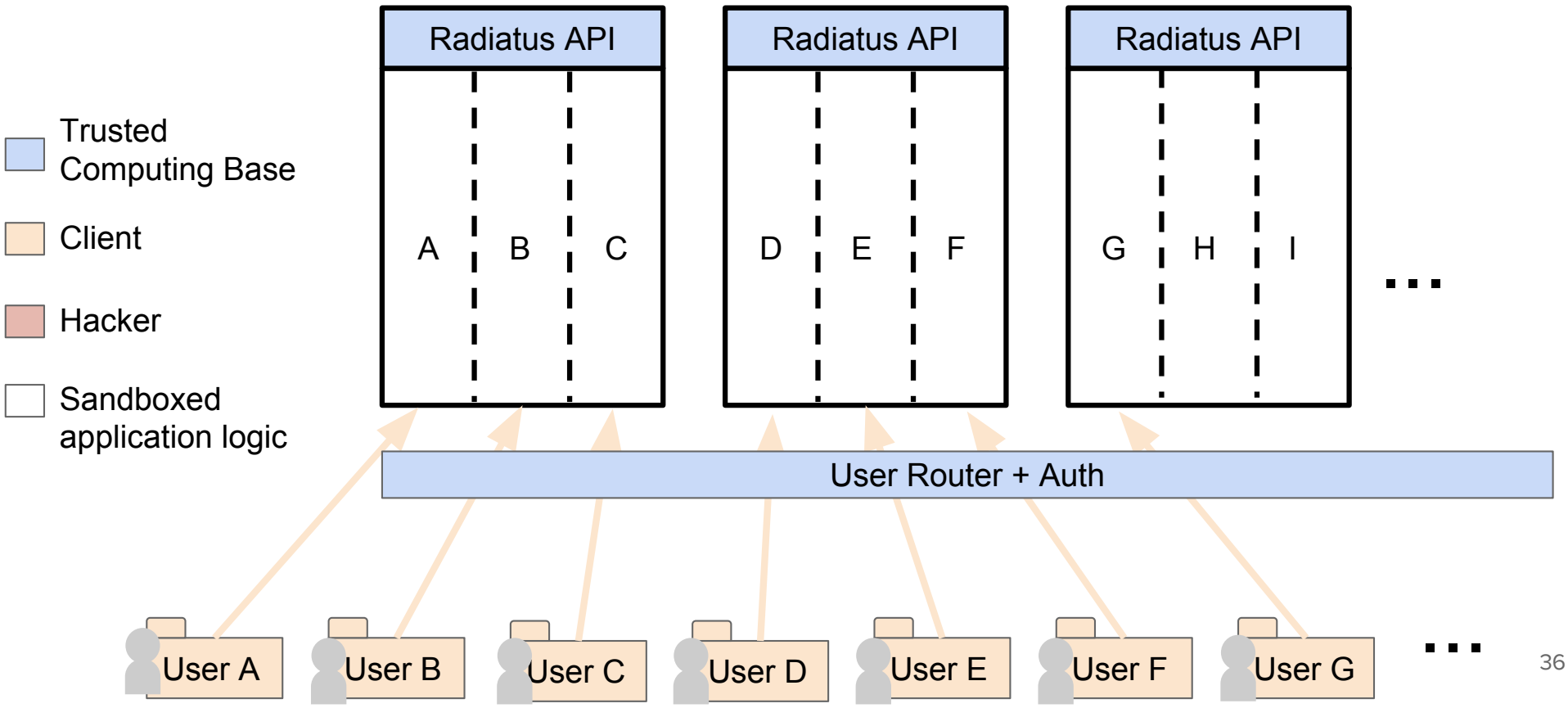


# Radiatus

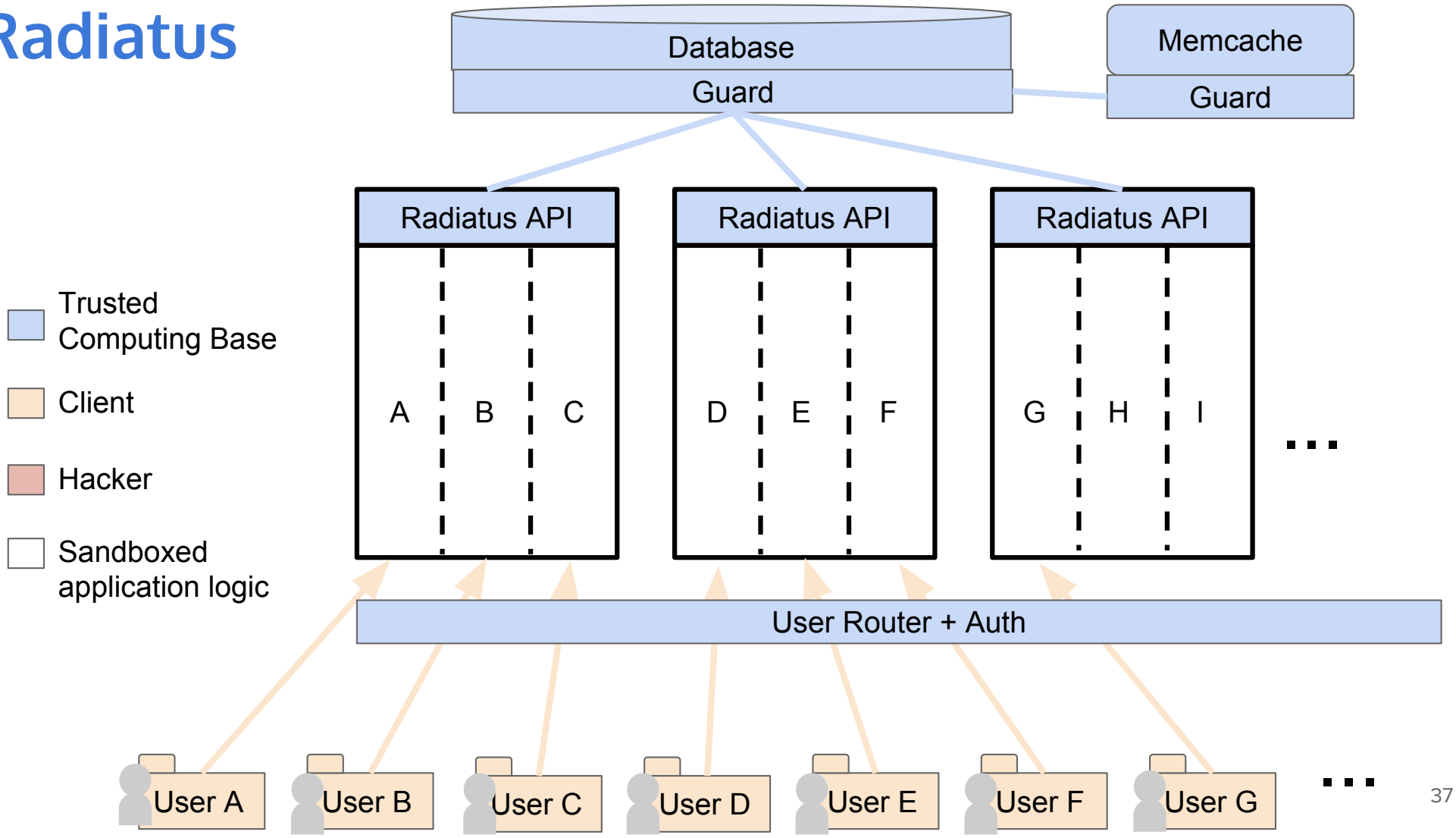
Shared-nothing server-side architecture for strongly isolating users in web applications

- Sandboxed user containers for code and data
- Limit impact of unknown vulnerabilities

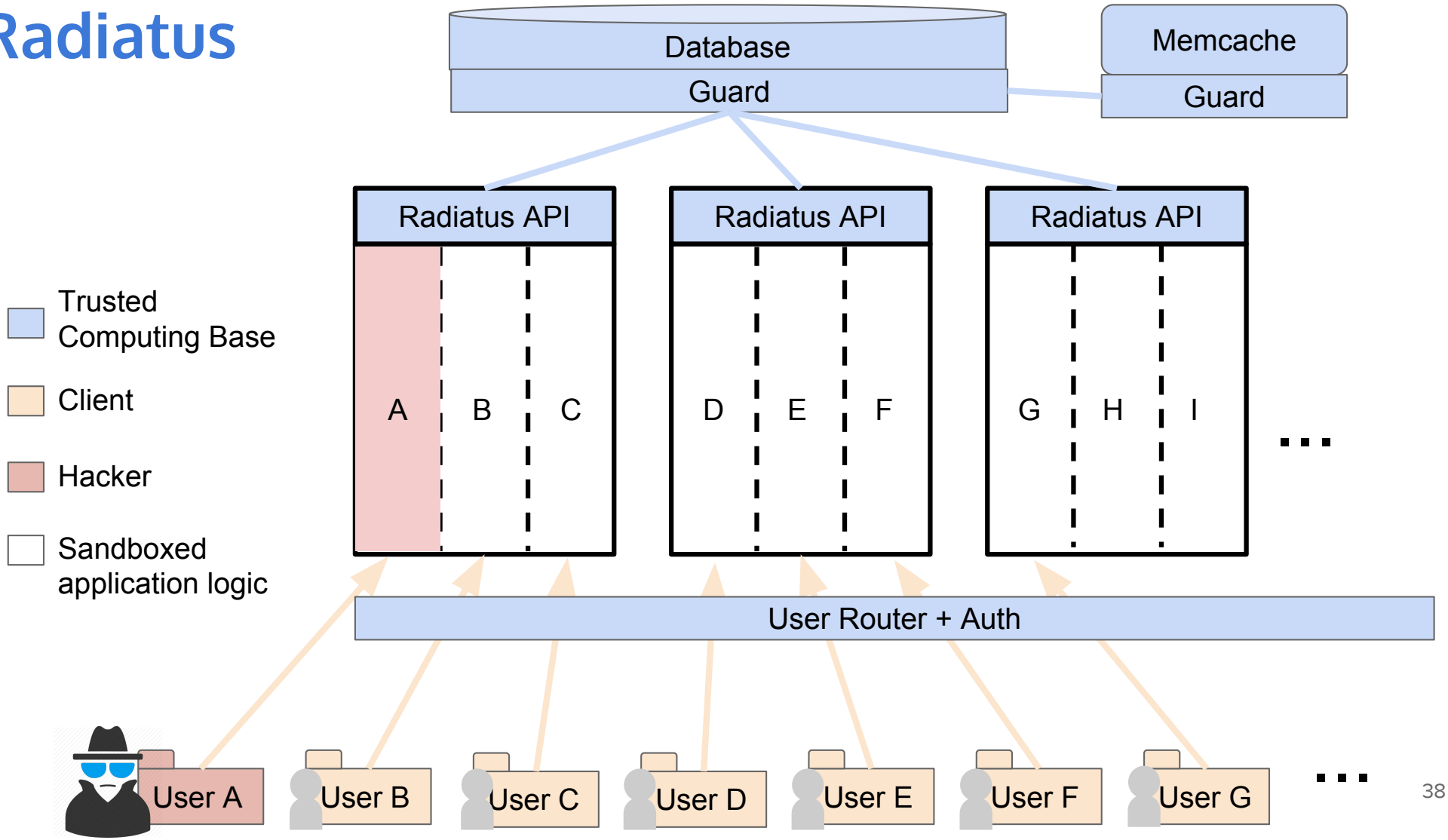
# Radiatus


Trusted Computing Base
Client
Hacker
Sandboxed application logic
Radiatus API
A
B
C
Radiatus API
D
E
F
Radiatus API
G
H
I
...
User Router + Auth
User A
User B
User C
User D
User E
User F
User G
...

# Radiatus


Database
Guard
Memcache
Guard
Radiatus API
Radiatus API
Radiatus API
A
B
C
D
E
F
G
H
I
...
Trusted Computing Base
Client
Hacker
Sandboxed application logic
User Router + Auth
User A
User B
User C
User D
User E
User F
User G
...

# Radiatus


Database
Guard
Memcache
Guard
Radiatus API
Radiatus API
Radiatus API
A
B
C
D
E
F
G
H
I
...
Trusted Computing Base
Client
Hacker
Sandboxed application logic
User Router + Auth
User A
User B
User C
User D
User E
User F
User G
...

# Radiatus Results

Benefits:

- Scales linearly
- Prevents most severe web-related vulnerabilities

Trade-offs:

- Additional cost: ~$0.008 / user-year
- Programmability of explicit message passing

https://github.com/freedomjs/radiatus

# Overview

**Malicious Cloud**

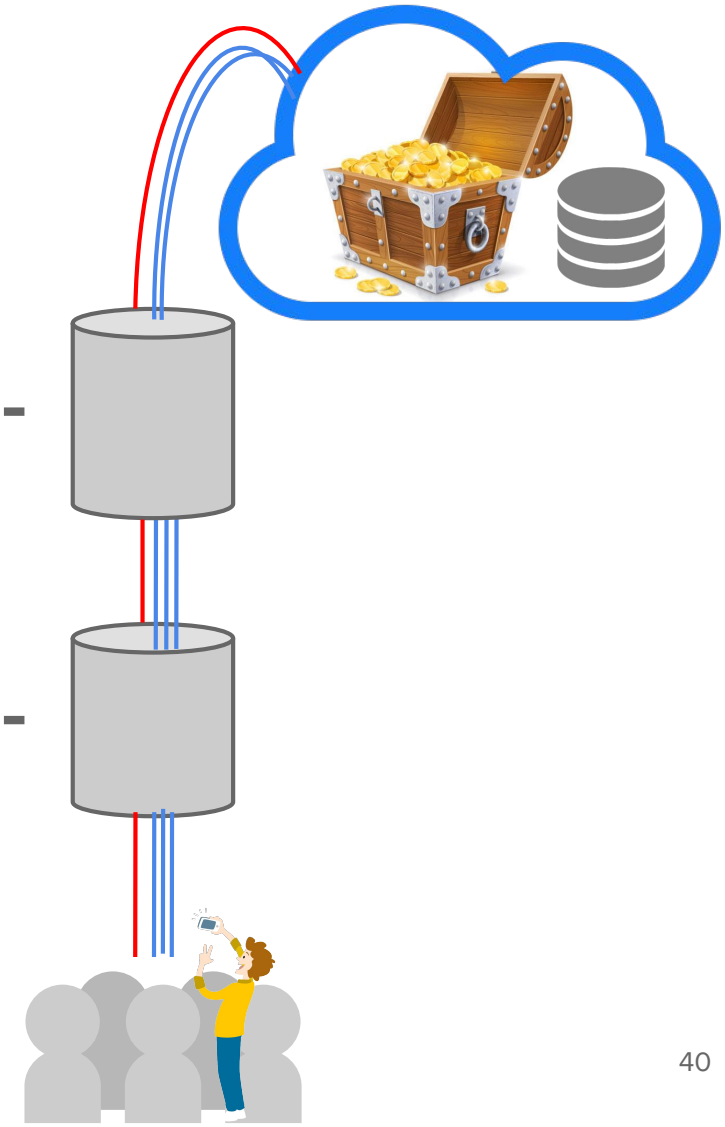
3. **Oblivious Cloud Services**
**Talek** - private publish-subscribe

**Malicious Network**

1. **uProxy** - censorship circumvention

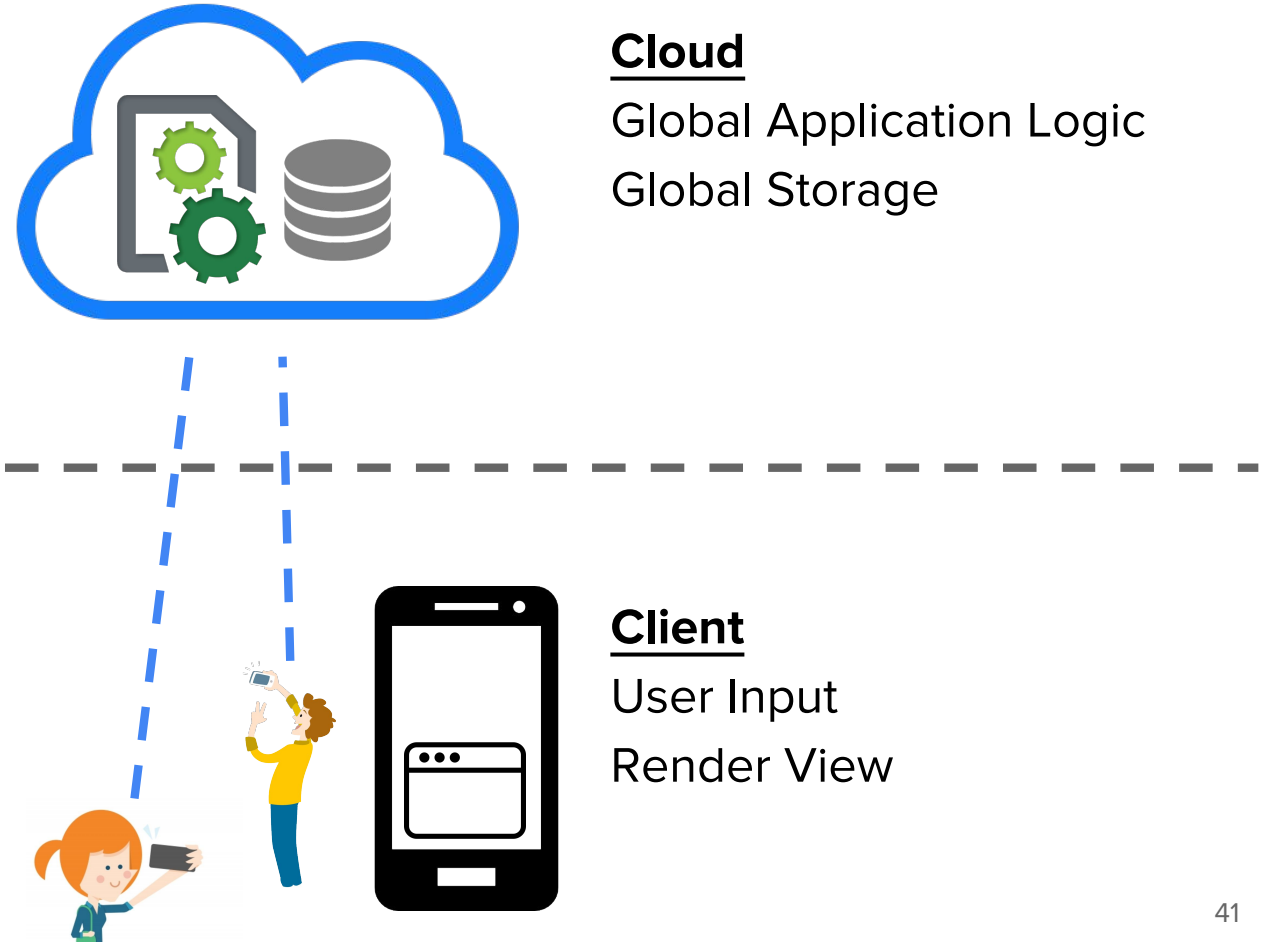
**Malicious Clients**

2. **Radiatus** - harden web applications from external intrusion



# Trusted Cloud

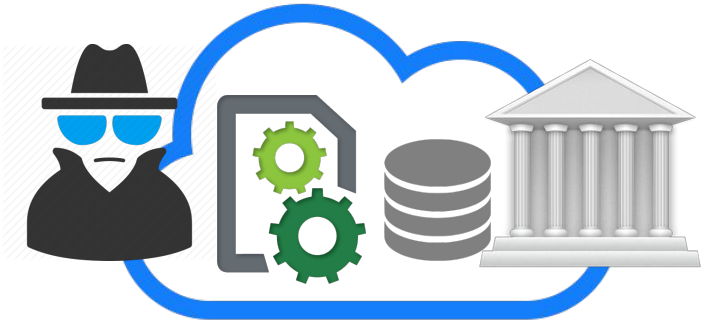
Safeguarding security



**Cloud**
Global Application Logic
Global Storage

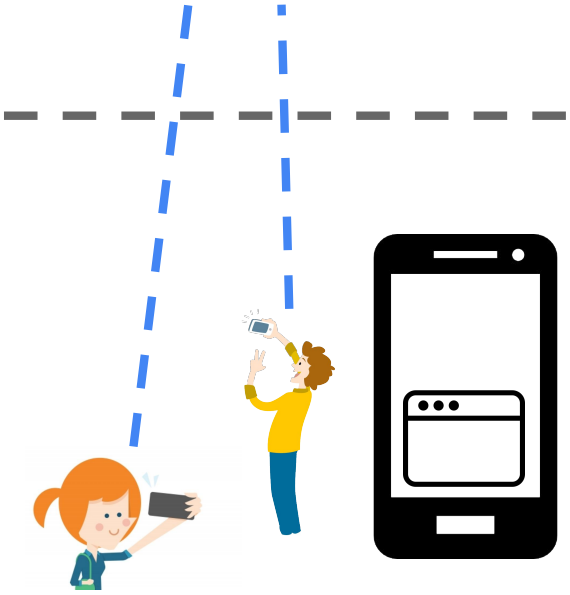
**Client**
User Input
Render View

# Untrusted Cloud



**Cloud**
Global Application Logic
Global Storage

What if we don't trust the cloud?

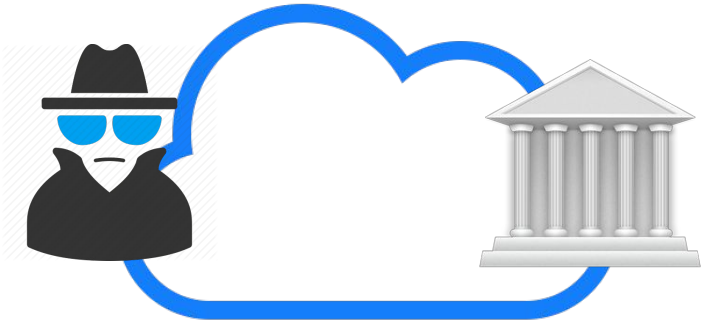




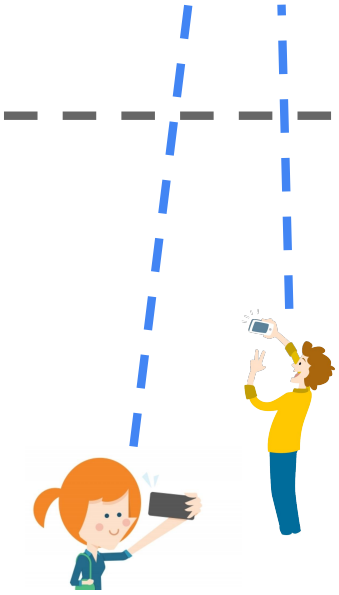
**Client**
User Input
Render View

# Untrusted Cloud

What do we need the cloud to do?



**Cloud**



**Client**

Per-user application logic

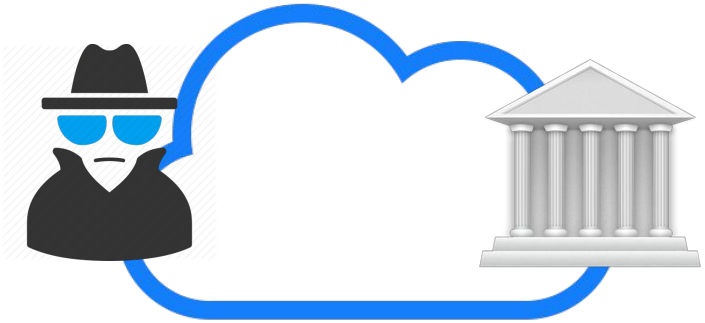
Per-user storage

User Input

Render View

# Untrusted Cloud

What do we need the cloud to do?

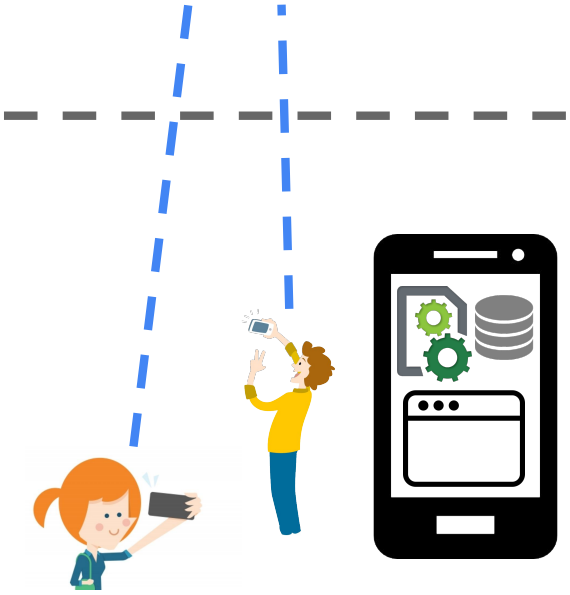


**Cloud**

Send data between users

Backup/sync storage

Analytics



**Client**

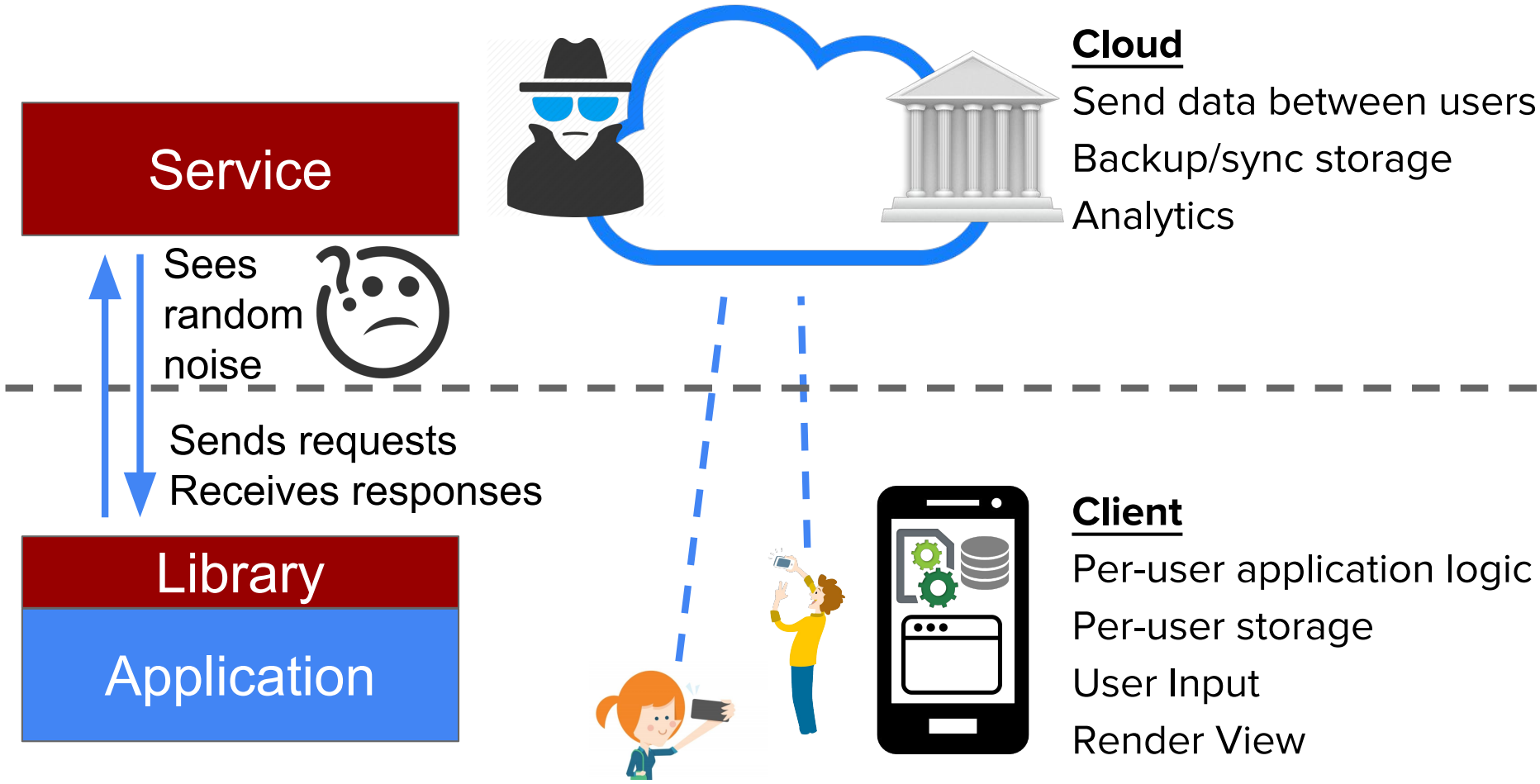
Per-user application logic

Per-user storage

User Input

Render View

# the Vision of *Oblivious Cloud Services*



# the Vision of *Oblivious Cloud Services*

**Cloud**

Send data between users

Backup/sync storage

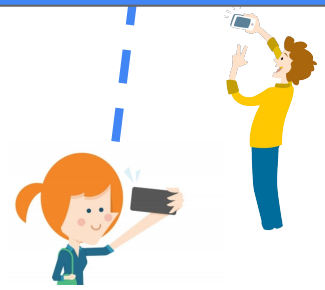
Analytics

Service

Library

Application

**Cloud services that are secure by design**



Per-user application logic

Per-user storage

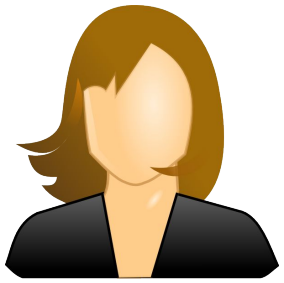
User Input

Render View

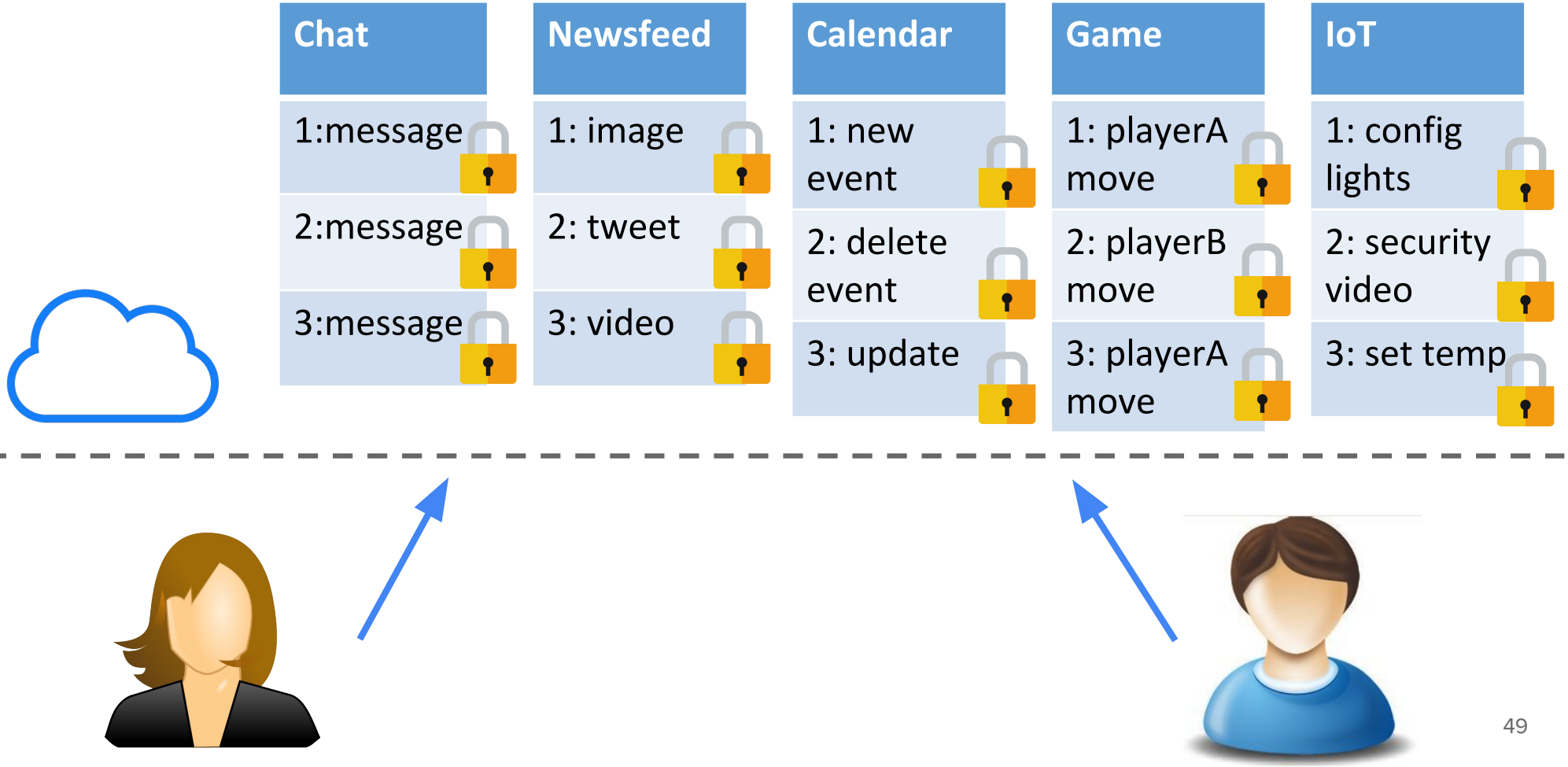
# Talek: a Private Publish-Subscribe Protocol

# Publish-Subscribe

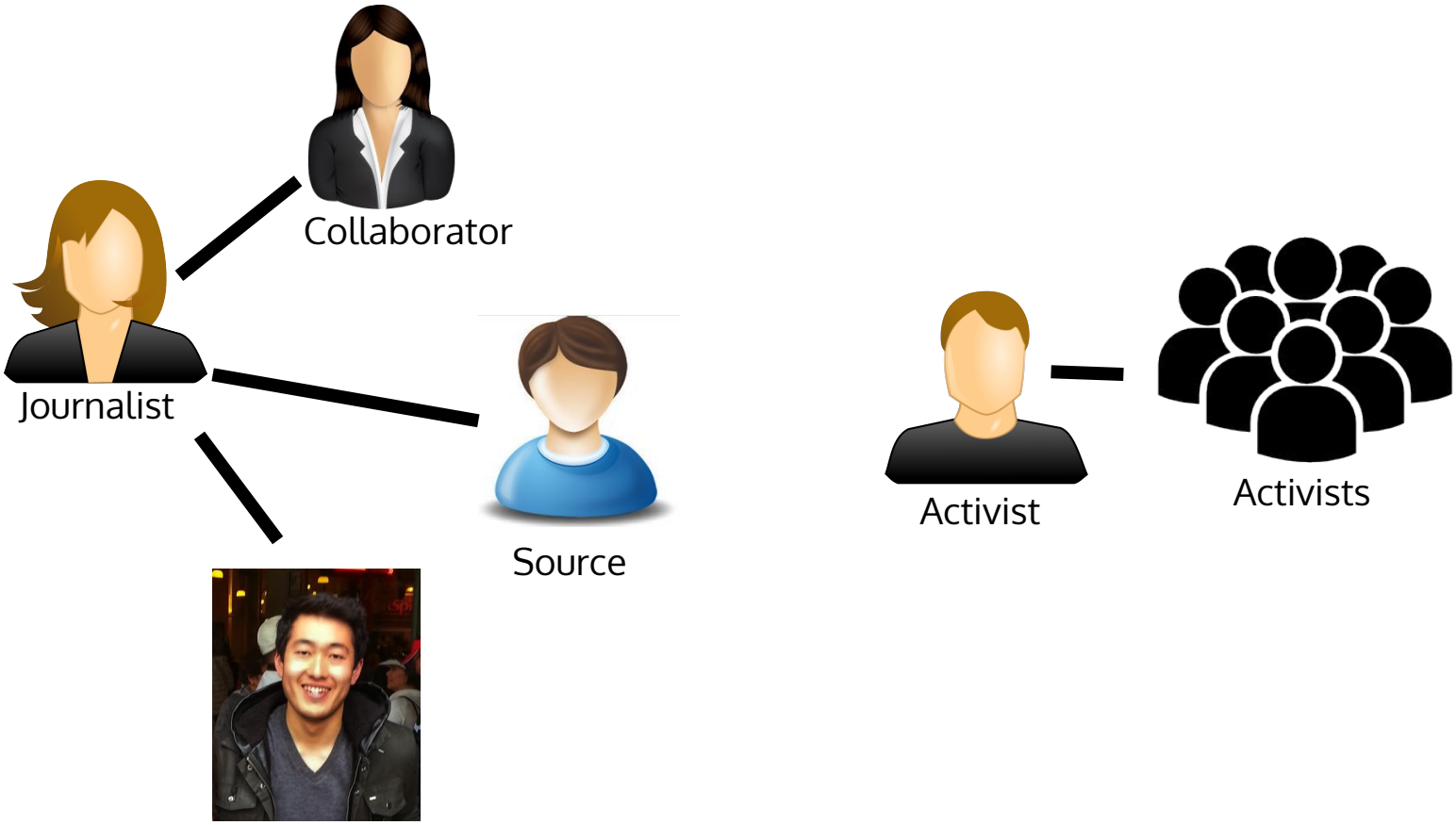
| Chat | Newsfeed | Calendar | Game | IoT |
|---|---|---|---|---|
| 1:message | 1: image | 1: new event | 1: playerA move | 1: config lights |
| 2:message | 2: tweet | 2: delete event | 2: playerB move | 2: security video |
| 3:message | 3: video | 3: update | 3: playerA move | 3: set temp |



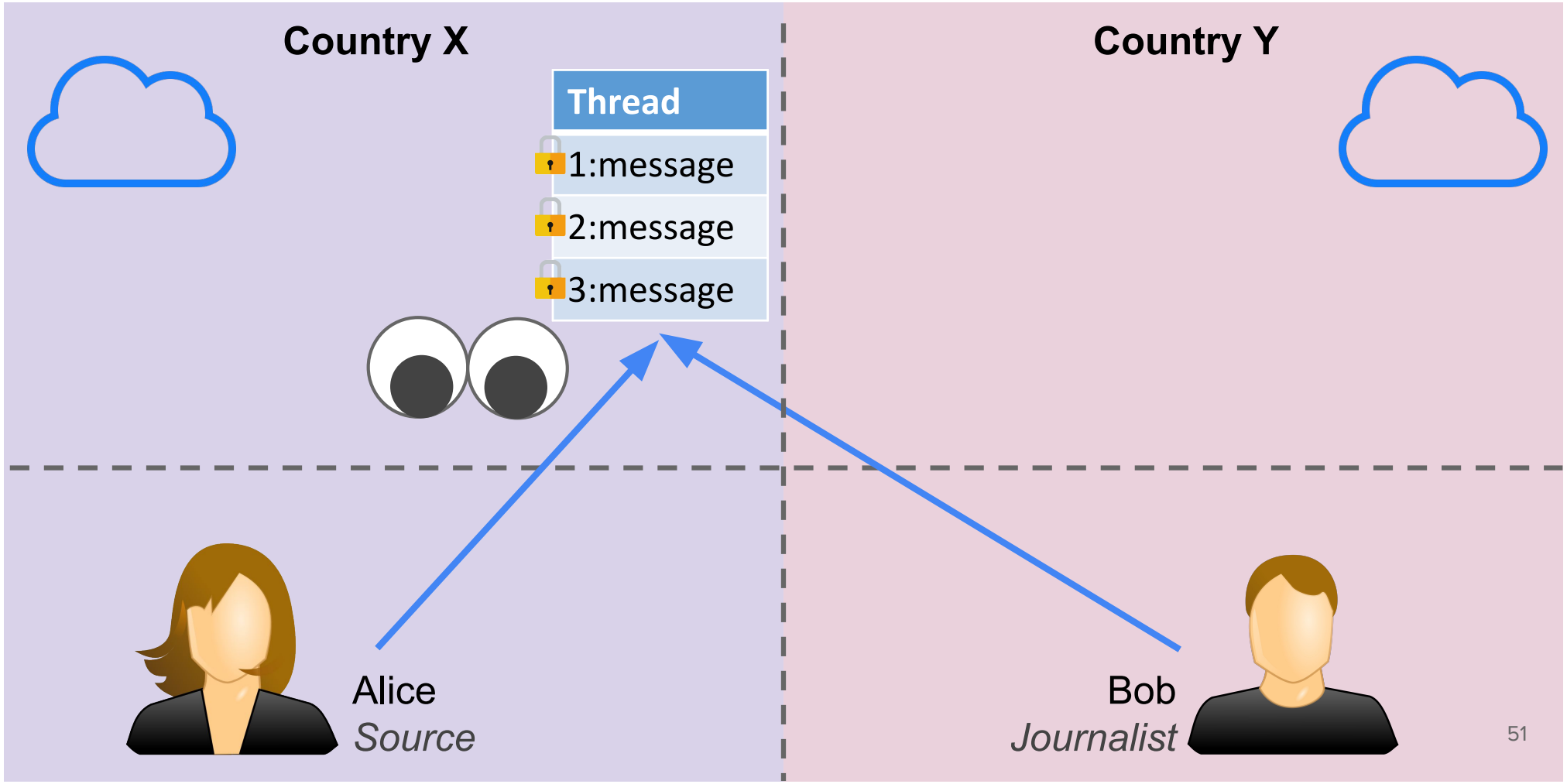
# Encryption protects the content…



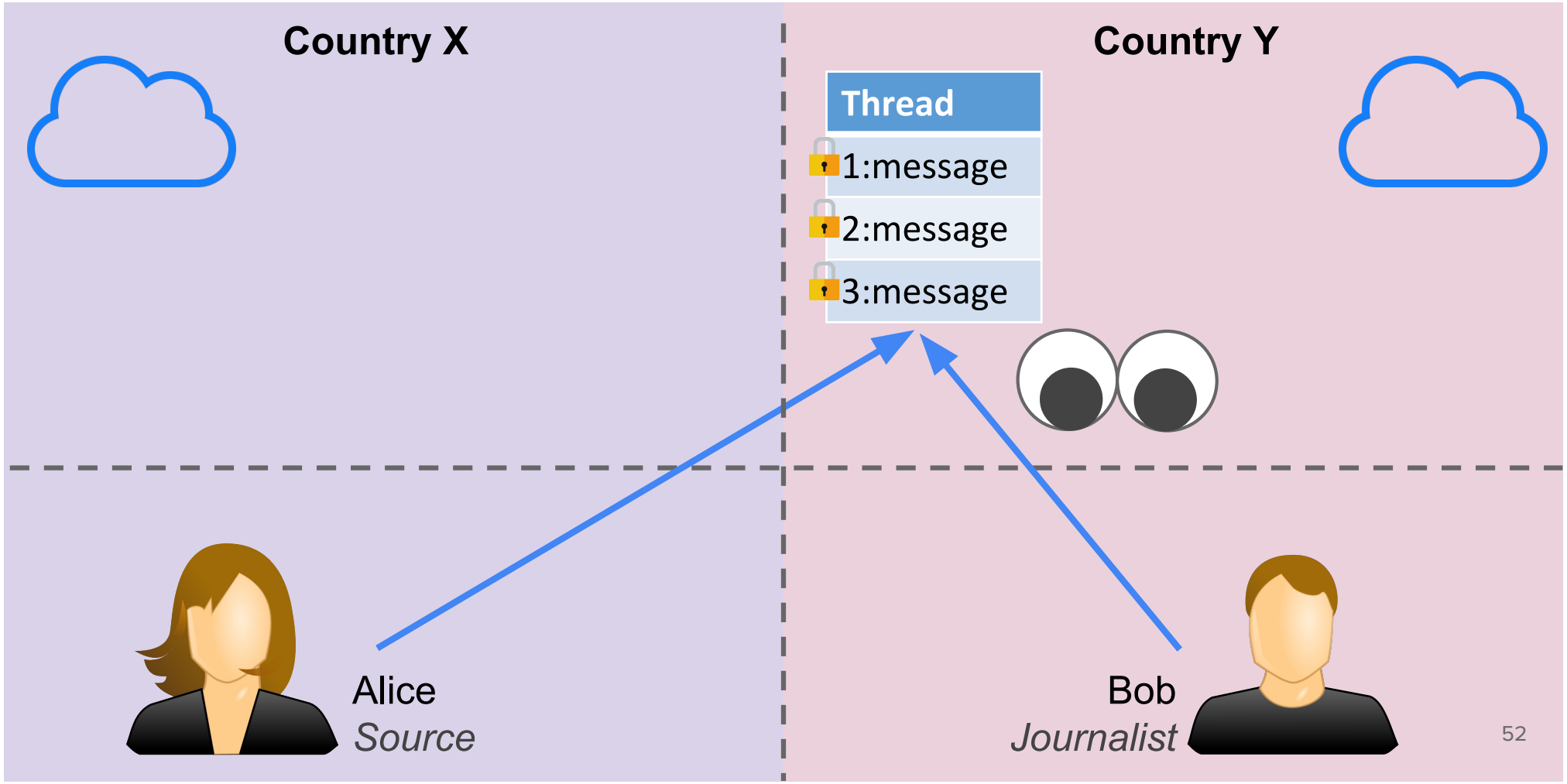
# … but communication patterns are exposed



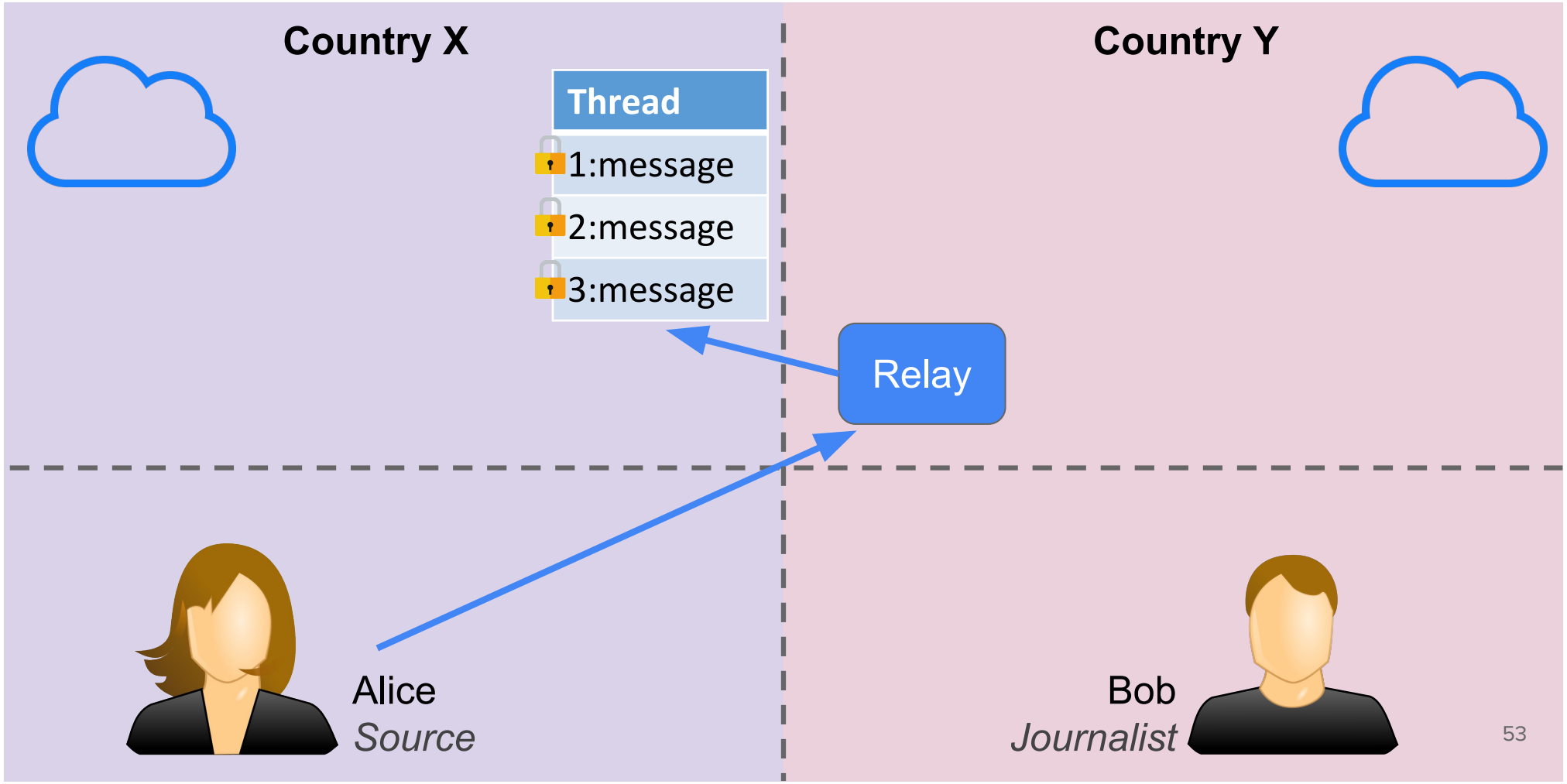
# New York Times Source



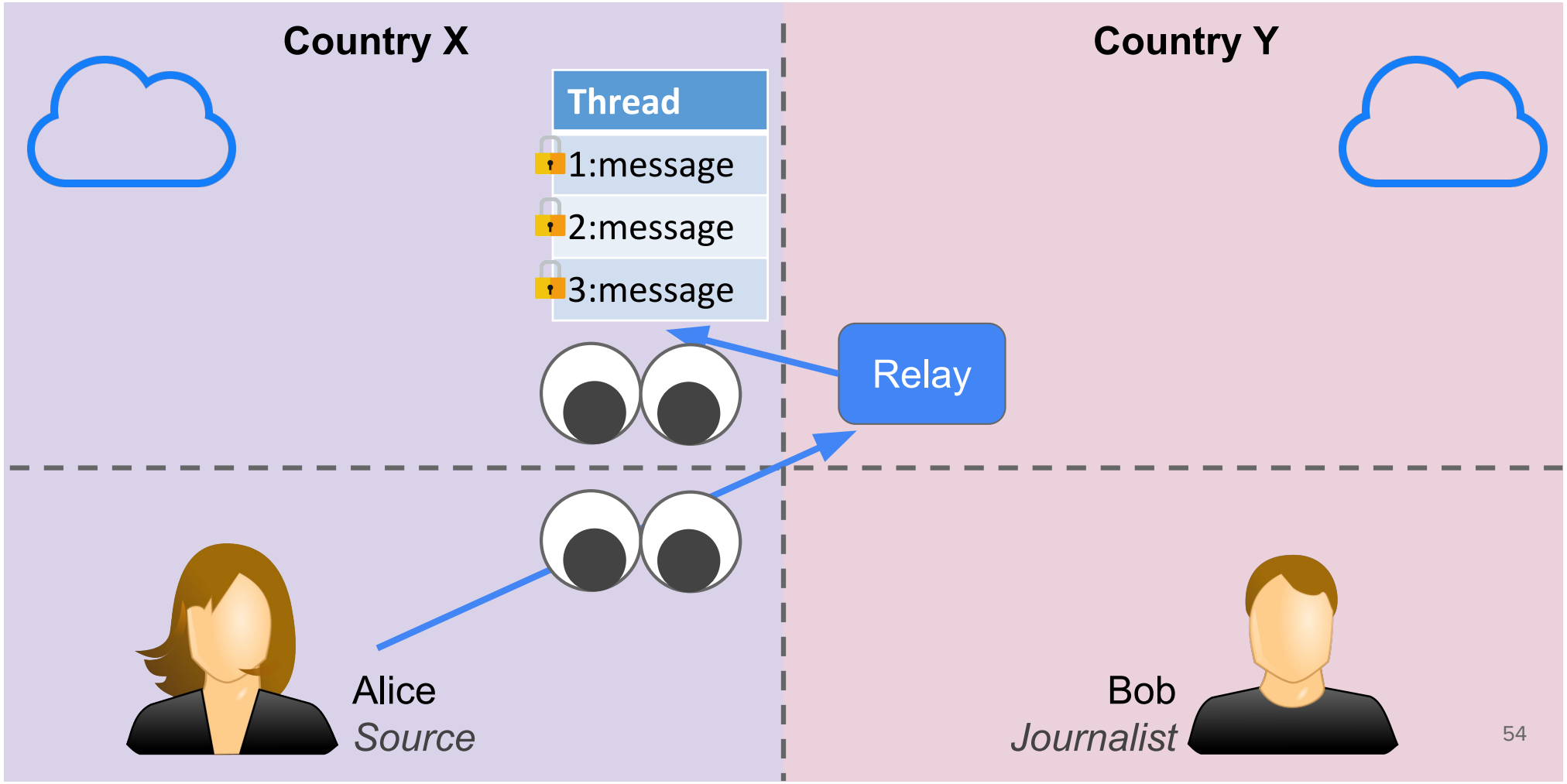
# New York Times Source



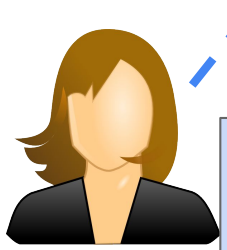
# New York Times Source



# New York Times Source



# Talek



Private publish-subscribe (pub/sub) system for sharing data through untrusted clouds

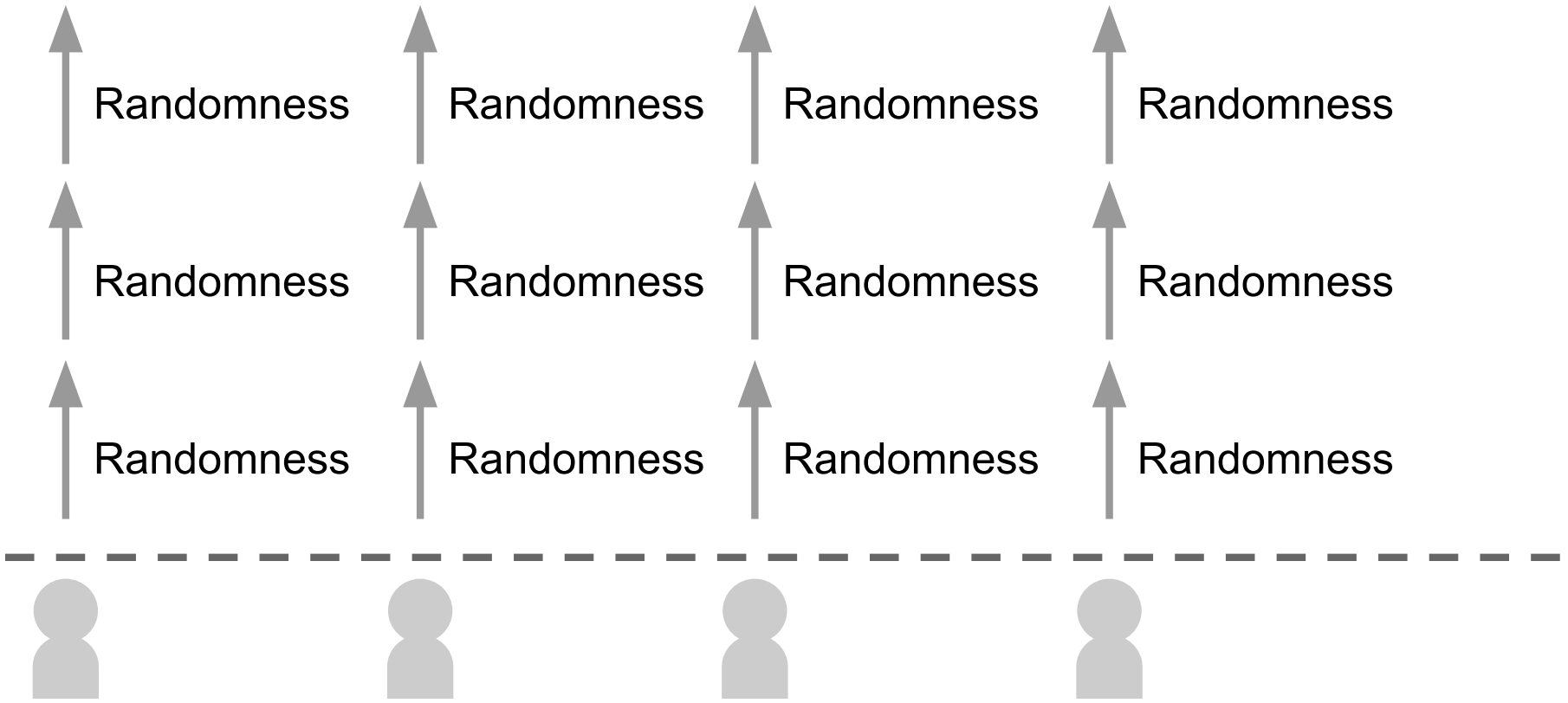
- Hide both contents and communication patterns
- Made practical using ***oblivious logging*** and ***private notifications***
- System with 3-4 orders of magnitude better performance than closest related work

# Security Goal: Indistinguishability

*Any two access sequences from a client look indistinguishable to the adversary*

# Security Goal: Indistinguishability

*Any two access sequences from a client look indistinguishable to the adversary*



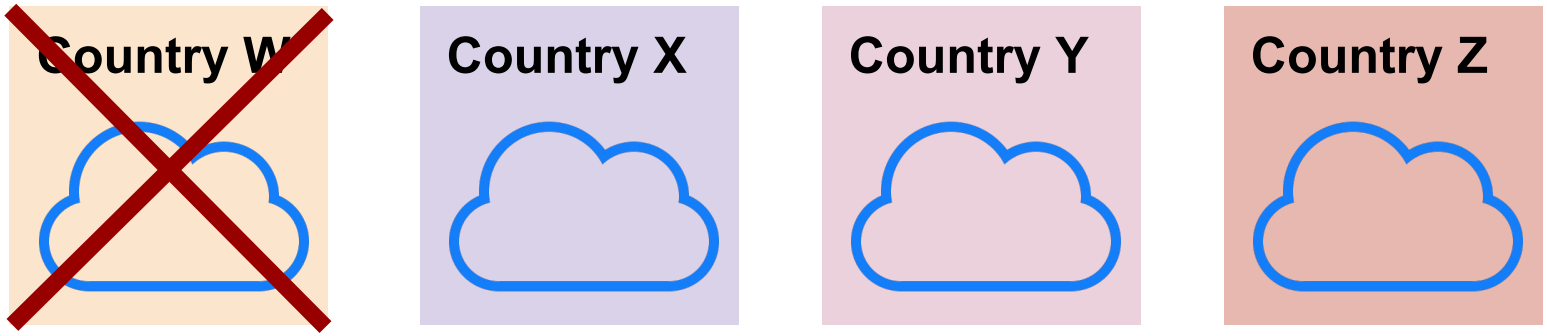
# Talek Goals

**Security Goal: Indistinguishability**
*Any two access sequences from a client look indistinguishable to the adversary*

**Systems Goals:**

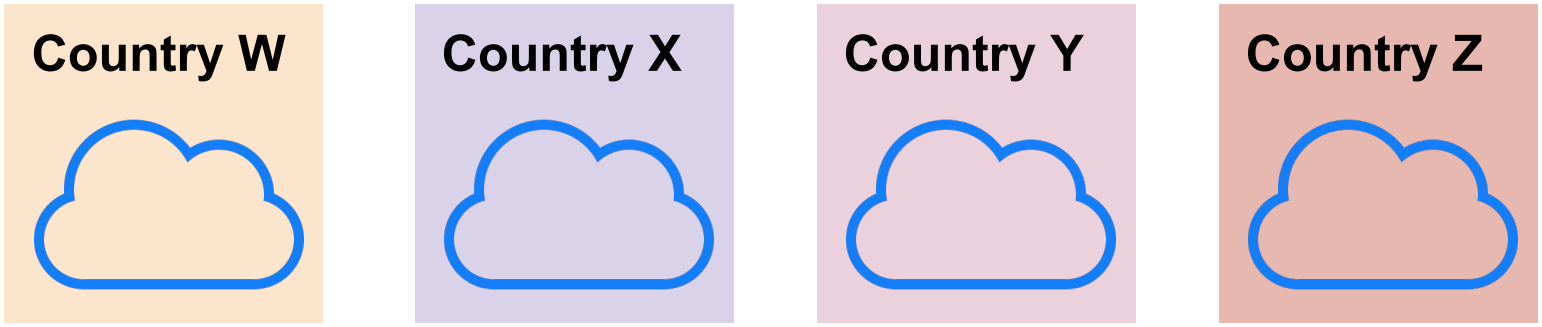
- Mobile-friendly: 1 message per request/response
- Efficient: Thousands of online users sending a message every 5 seconds
- General Purpose: messaging and newsfeeds
- Low latency: ~5-10s

# Limitations




- Any unavailable cloud will prevent access
- Host in widely used cloud providers

# Anytrust Threat Model

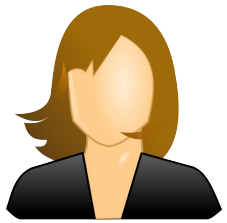


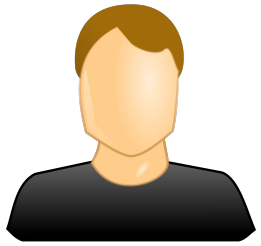

- Application configured with >1 independent clouds
- Clouds logging everything about users

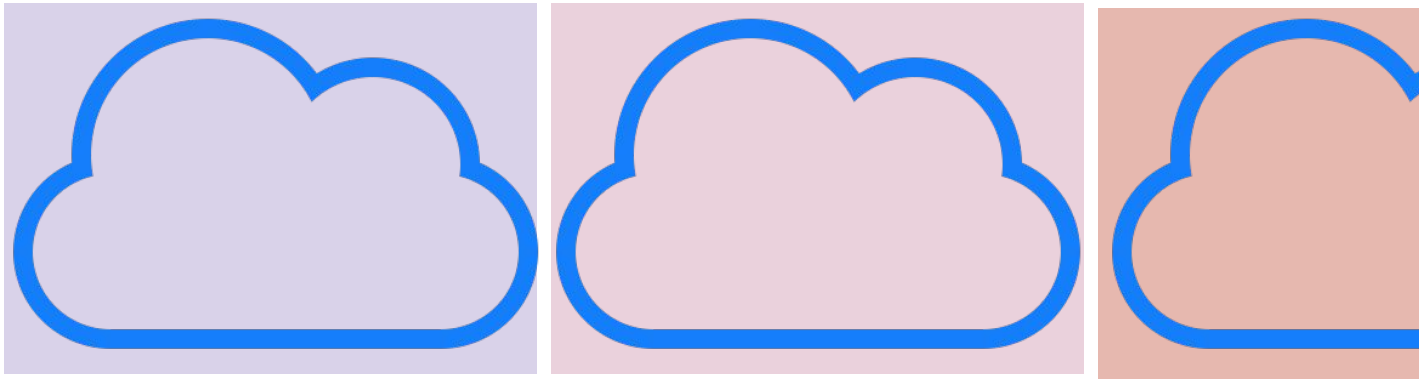
**At least 1 non-colluding**

# Talek Threat Model

Trusted groups



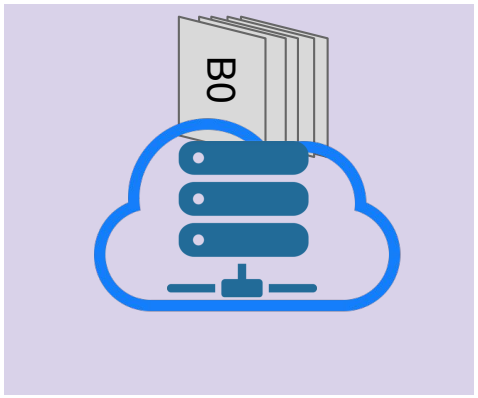


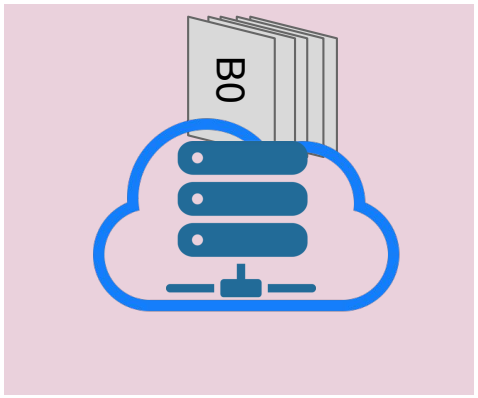


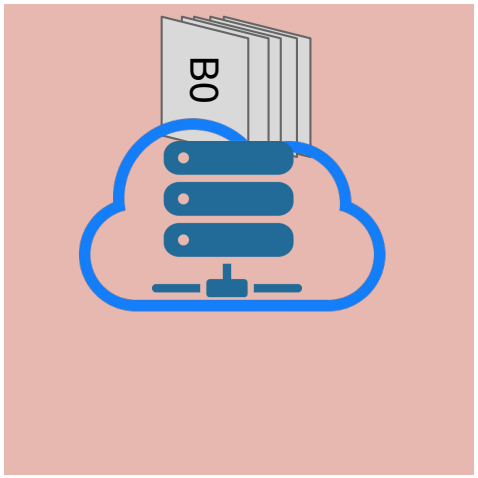
Anytrust: At least 1 non-colluding

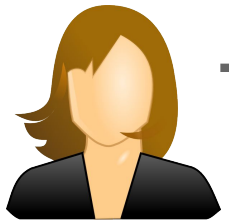
Mutually distrusting users

# Private Information Retrieval (PIR) (Chor,1998)



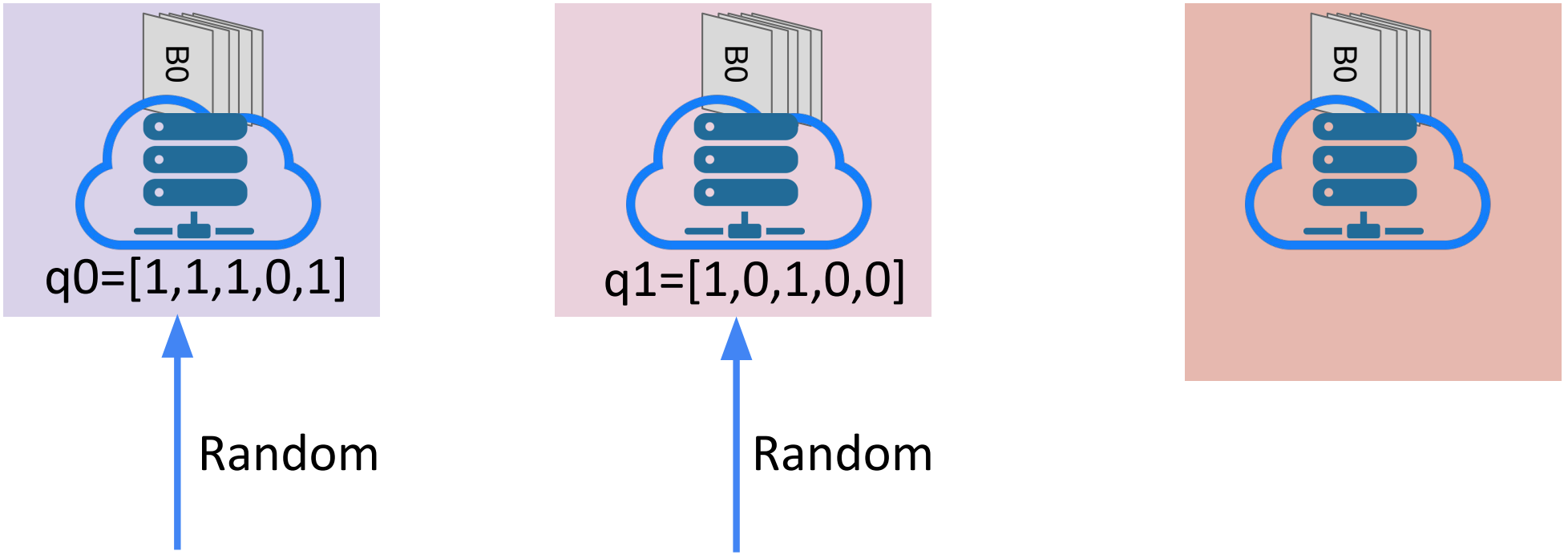



Client

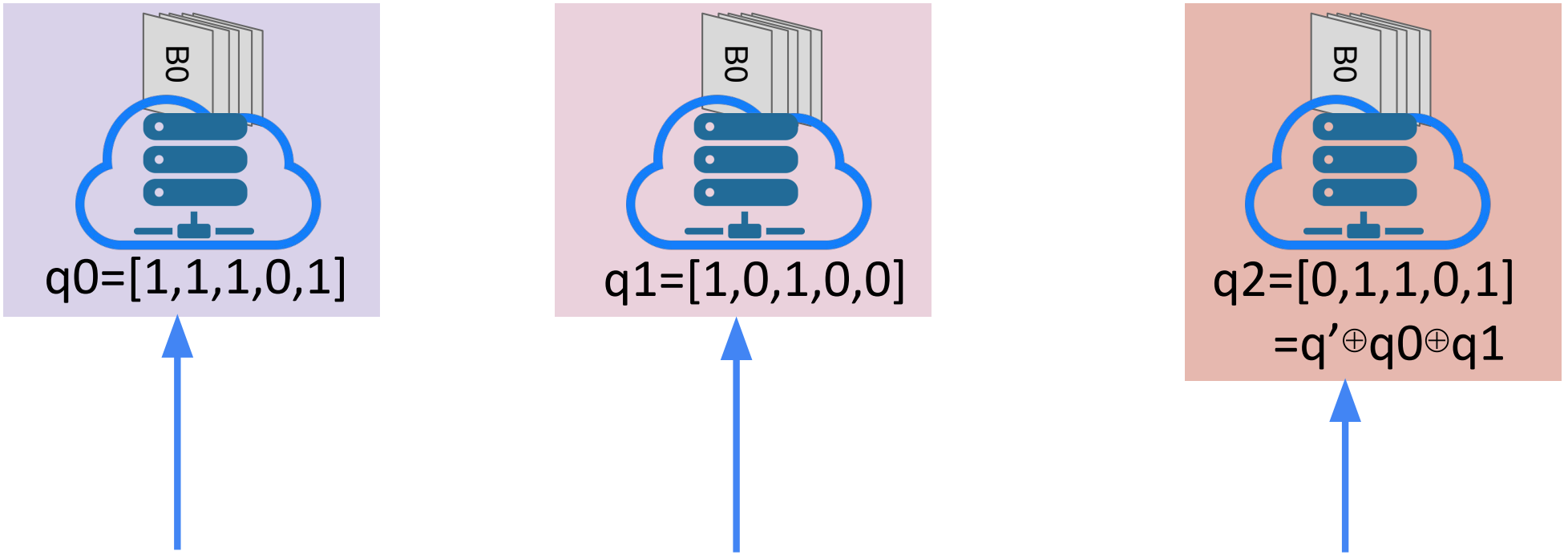
Read bucket 2

q’=[0,0,1,0,0]

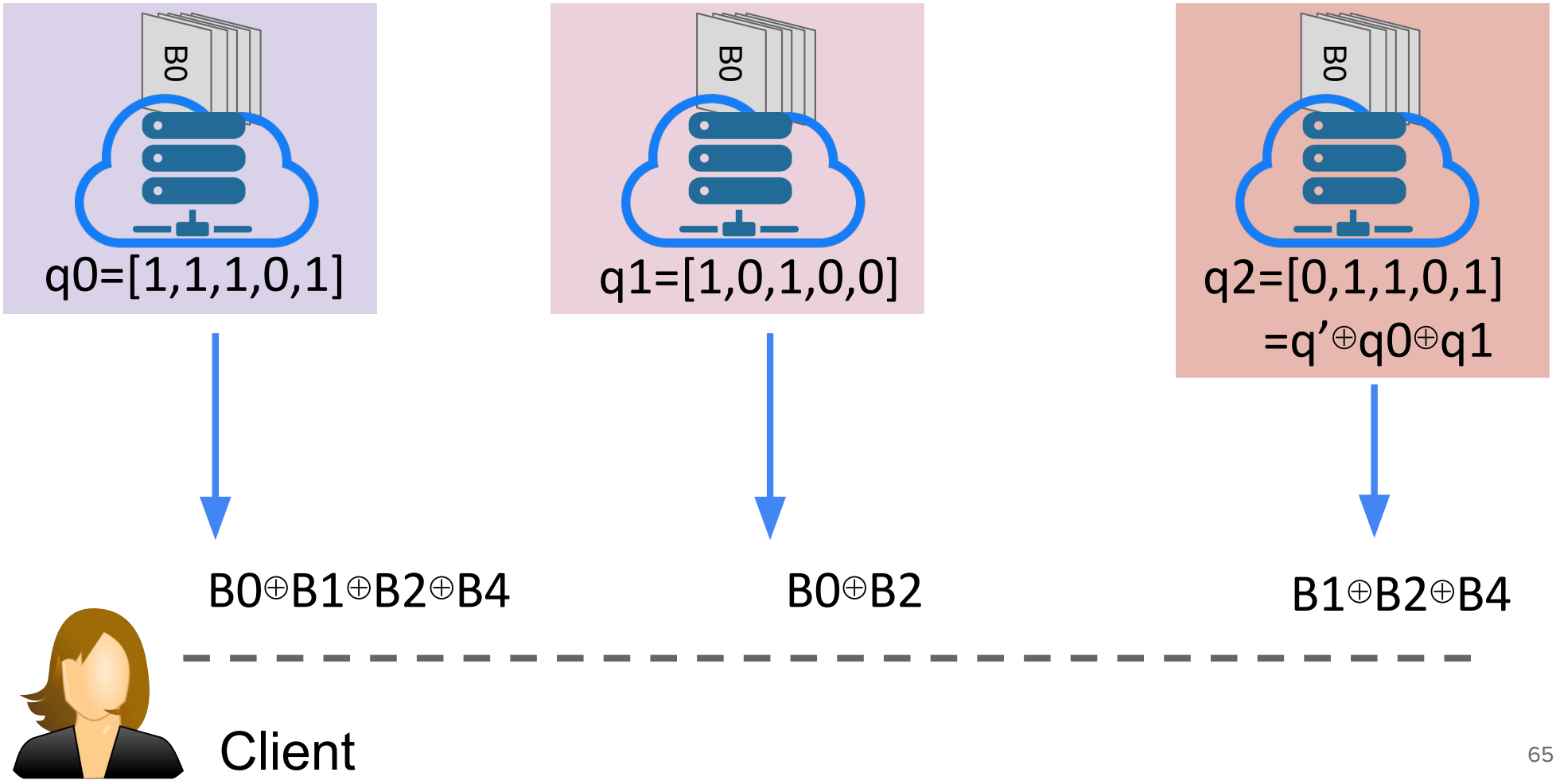
# Private Information Retrieval (PIR)



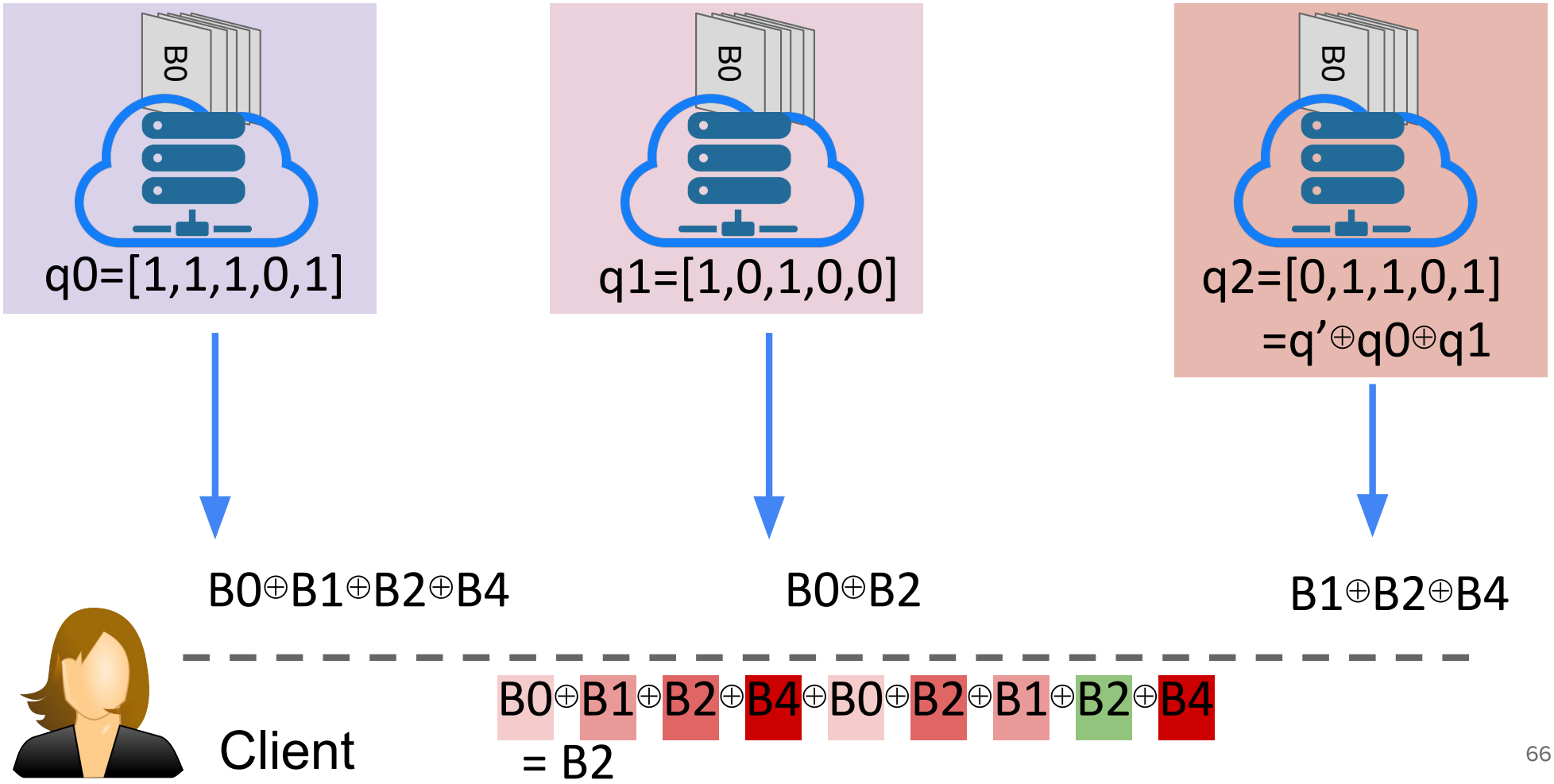
# Private Information Retrieval (PIR)



# Private Information Retrieval (PIR)



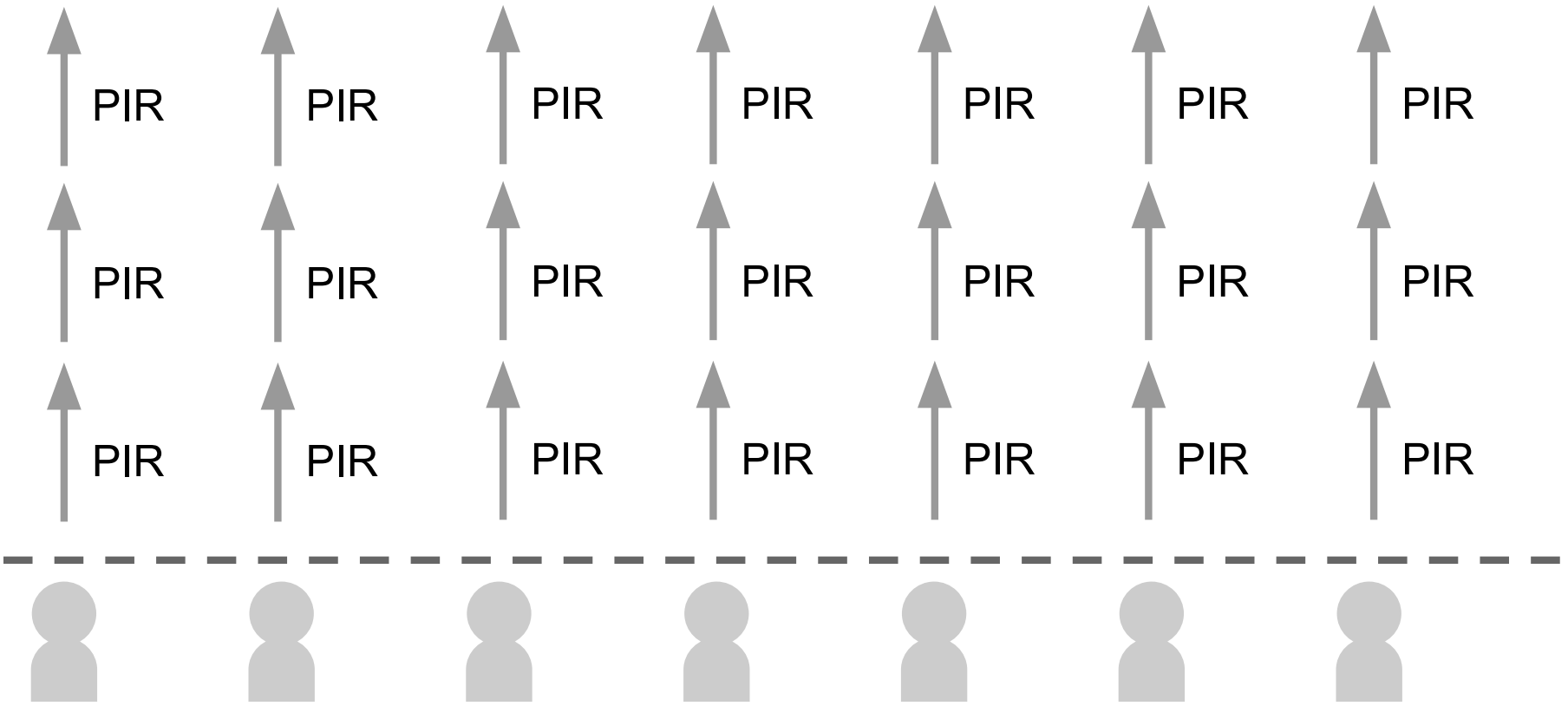
# Private Information Retrieval (PIR)



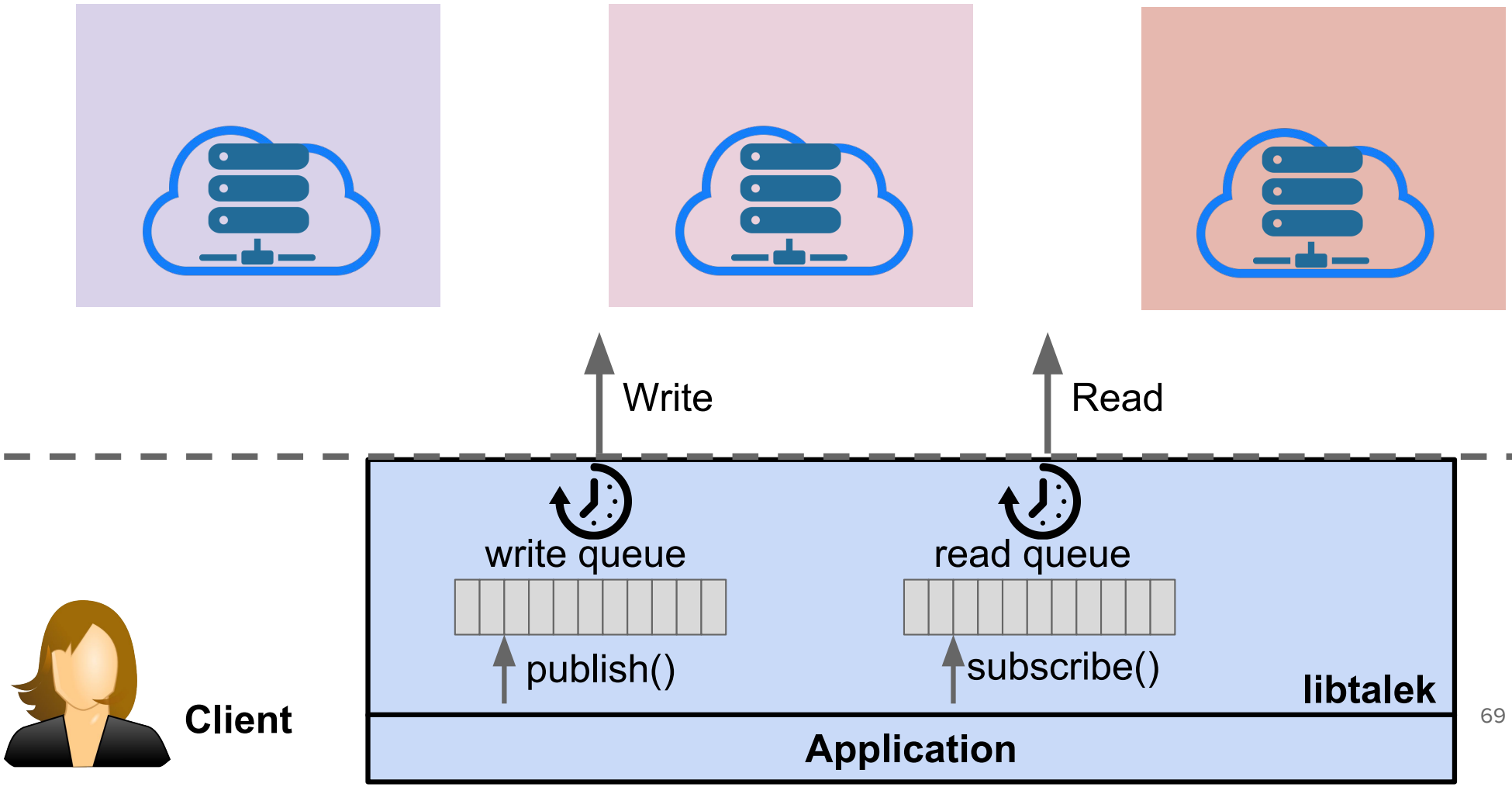
# PIR Limitations

- Expensive: Read requires scan of database
- Equal-sized buckets
- Consistent snapshots across all servers
- Read-only

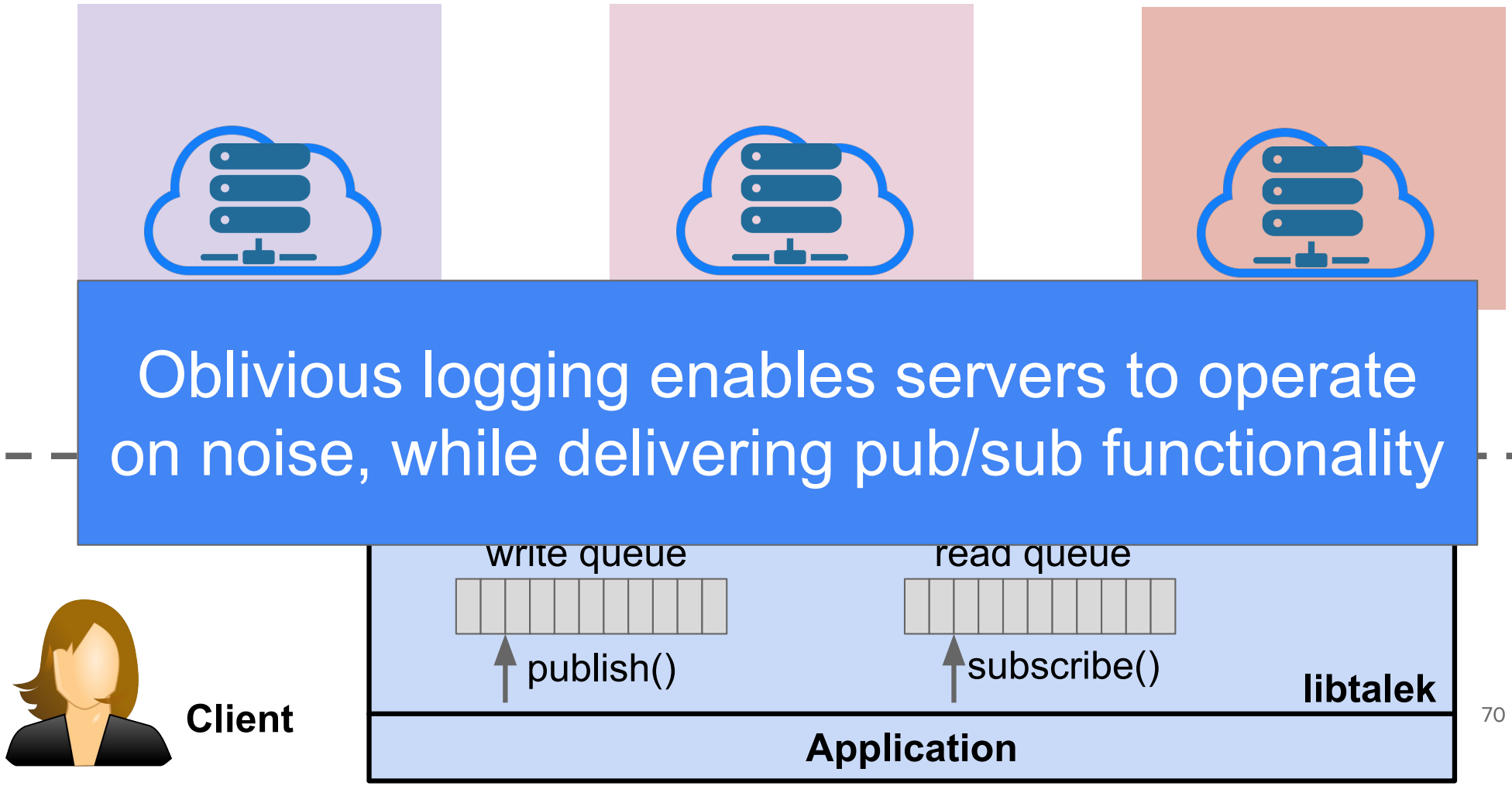
# Client Indistinguishability



# Talek Overview



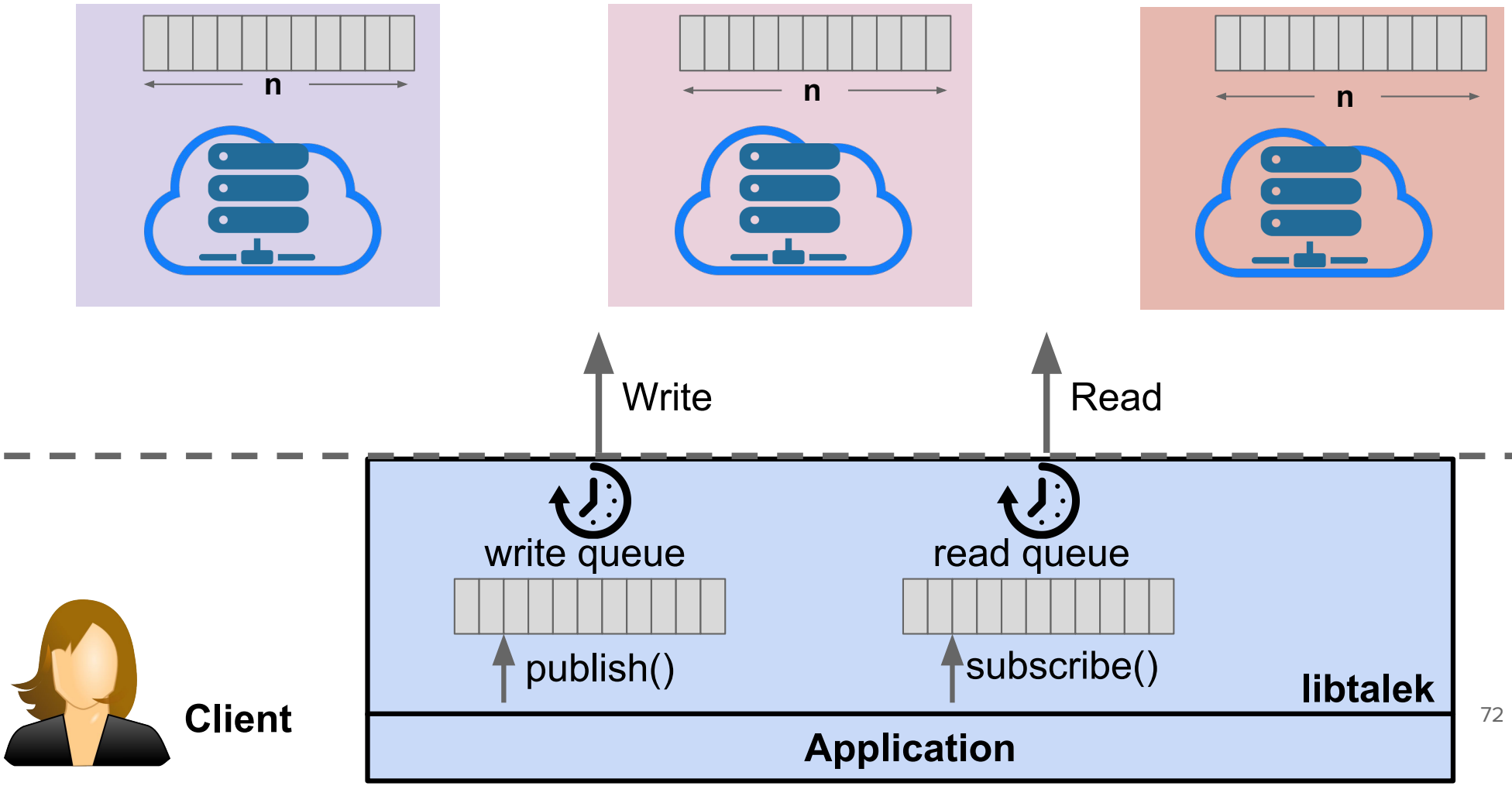
# Talek Overview



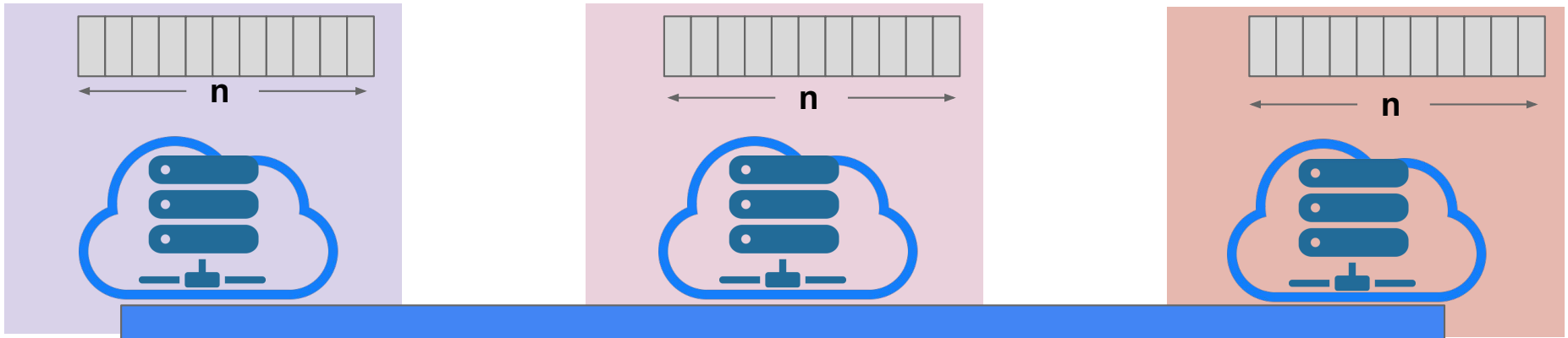
# Oblivious Logging

1. How do we bound the cost of a PIR operation?
2. How do publishers write in a way that looks random?
3. How do subscribers find messages on the server?
4. How do we deal with write conflicts?
5. How do we keep all servers consistent?

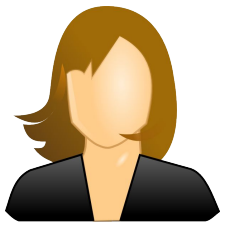
# Fixed Size Server-side State



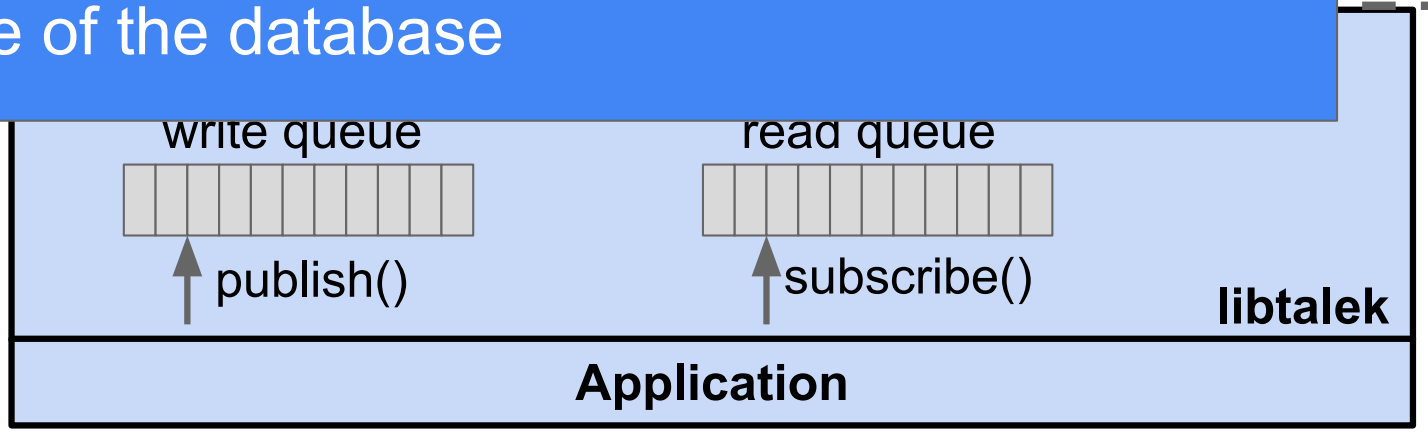
# Fixed Size Server-side State




**1. PIR Cost**
Bound the cost of a PIR by configuring the size of the database

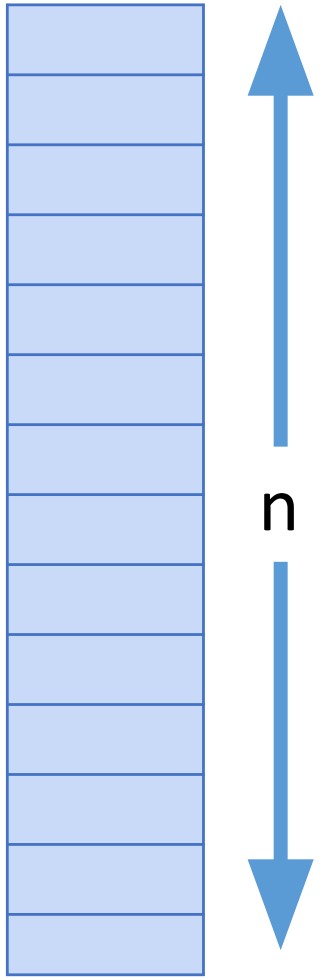


**Client**



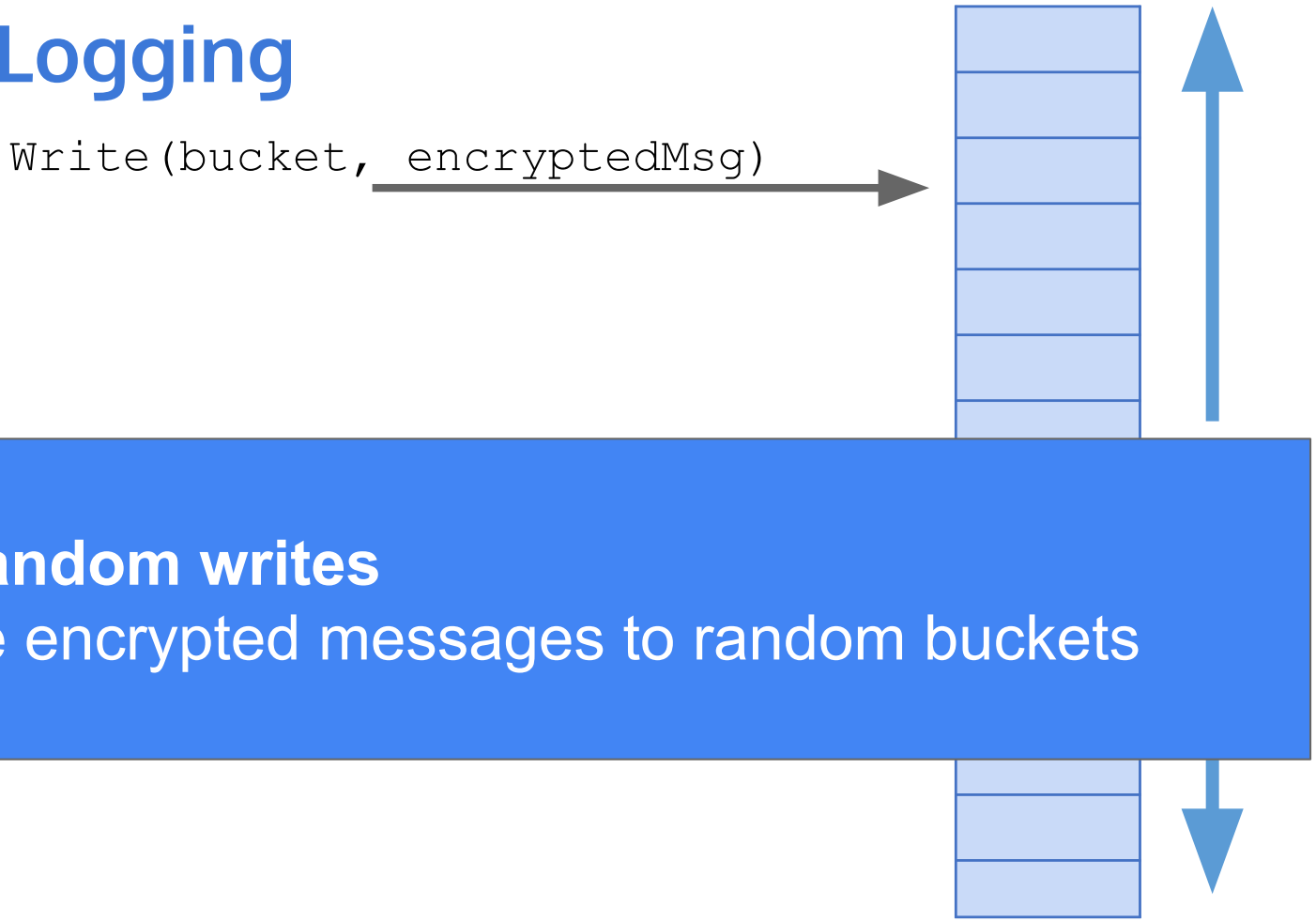
# Oblivious Logging

`Write(bucket, encryptedMsg)`




1. Remove oldest message
2. Insert message at specified bucket

# Oblivious Logging

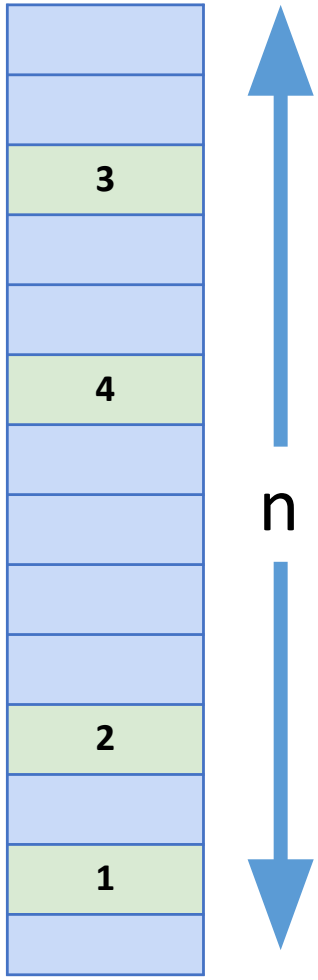



1. Rer
2. Inse

**2. Random writes**
Write encrypted messages to random buckets

# Topics and Log Trails

```
Write(bucket, encryptedMsg)
```




**Topic Handle:**

```
{
  topicId: uint128,
  encKey: byte[]
  seed: uint128
}
```

**Log Trail:**

*PRF(seed, seqNo) mod n*

# Topics and Log Trails

```
Write(bucket, encryptedMsg)
```


**Topic Handle:**

```
{
   topi
   encK
   seed
}
```

**3. Zero Coordination**
Publishers and subscribers use secret topic handles to coordinate

**Log Trail:**

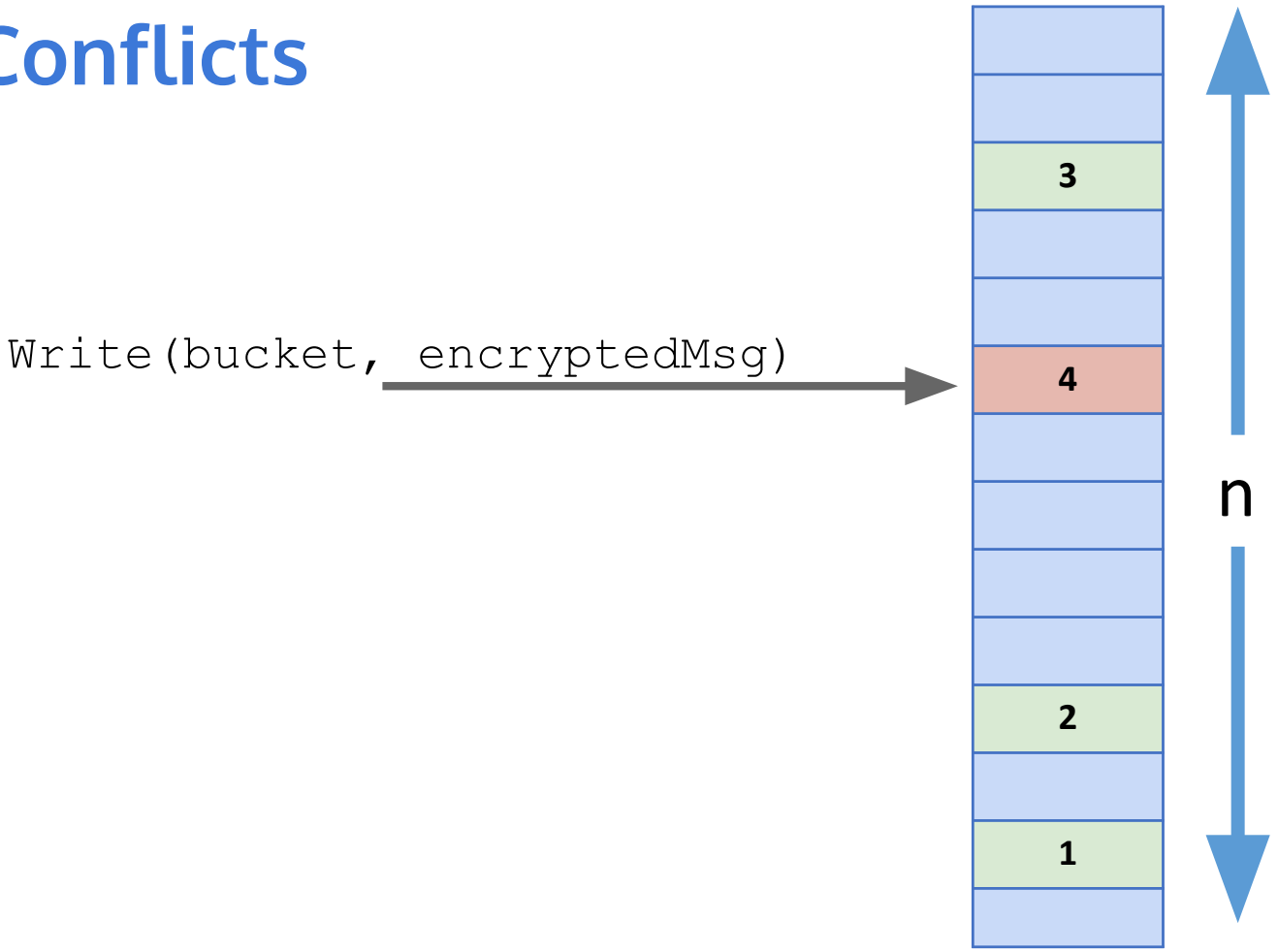
*PRF(seed, seqNo) mod n*

# Indistinguishable Writes

```
{
  topicId: uint128,
  encKey: byte[],
  seed: uint128
}
```

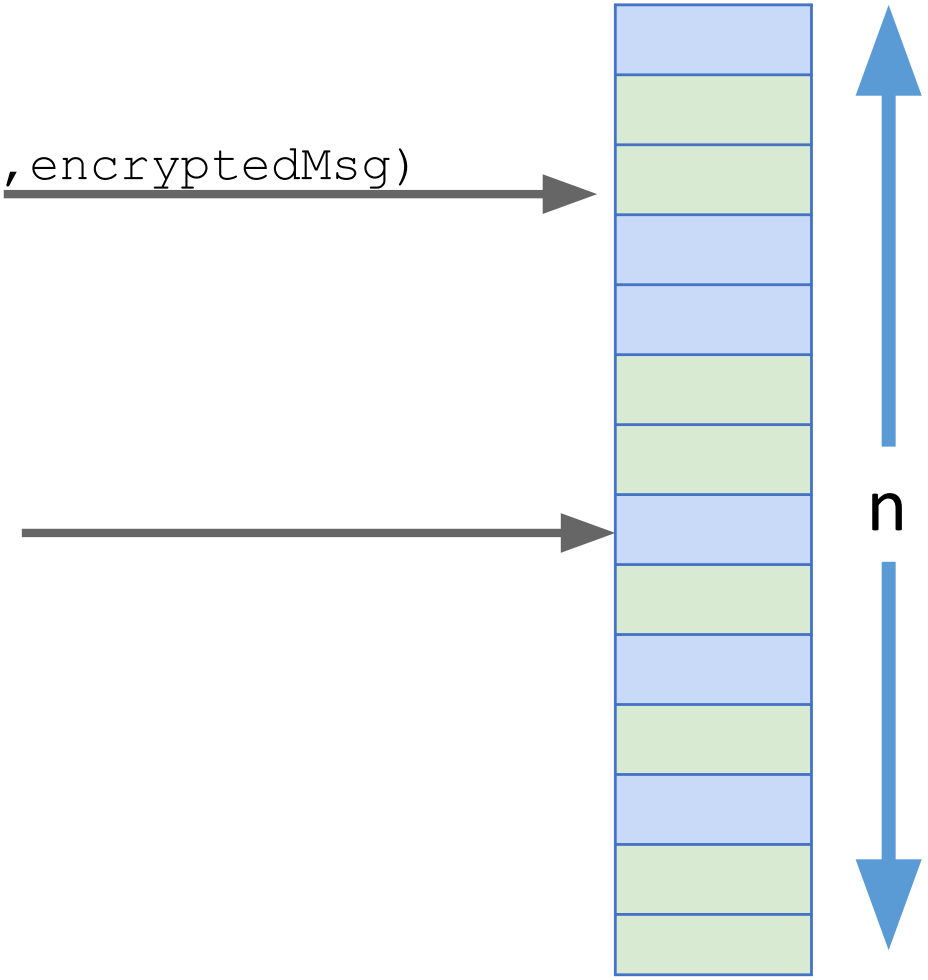
| Write | bucket | payload |
|---|---|---|
| Dummy | *PRF(idleSeed, i \| 1) mod b* | *Enc(idleKey, PRF(idle, i \| 2))* |
| Legitimate | *PRF(seed, seqNo) mod b* | *Enc(encKey, message)* |

# Handling Conflicts



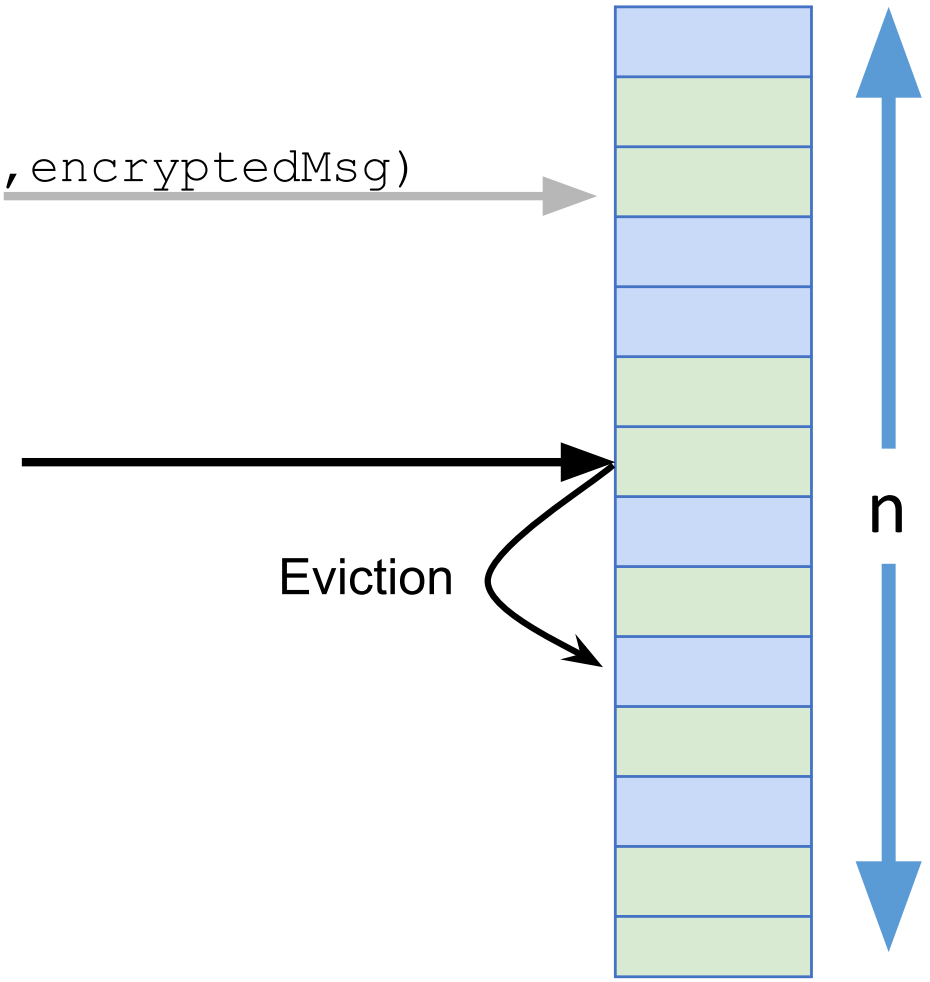
# Cuckoo Hashing

```
Write(bucket1,bucket2,encryptedMsg)
```



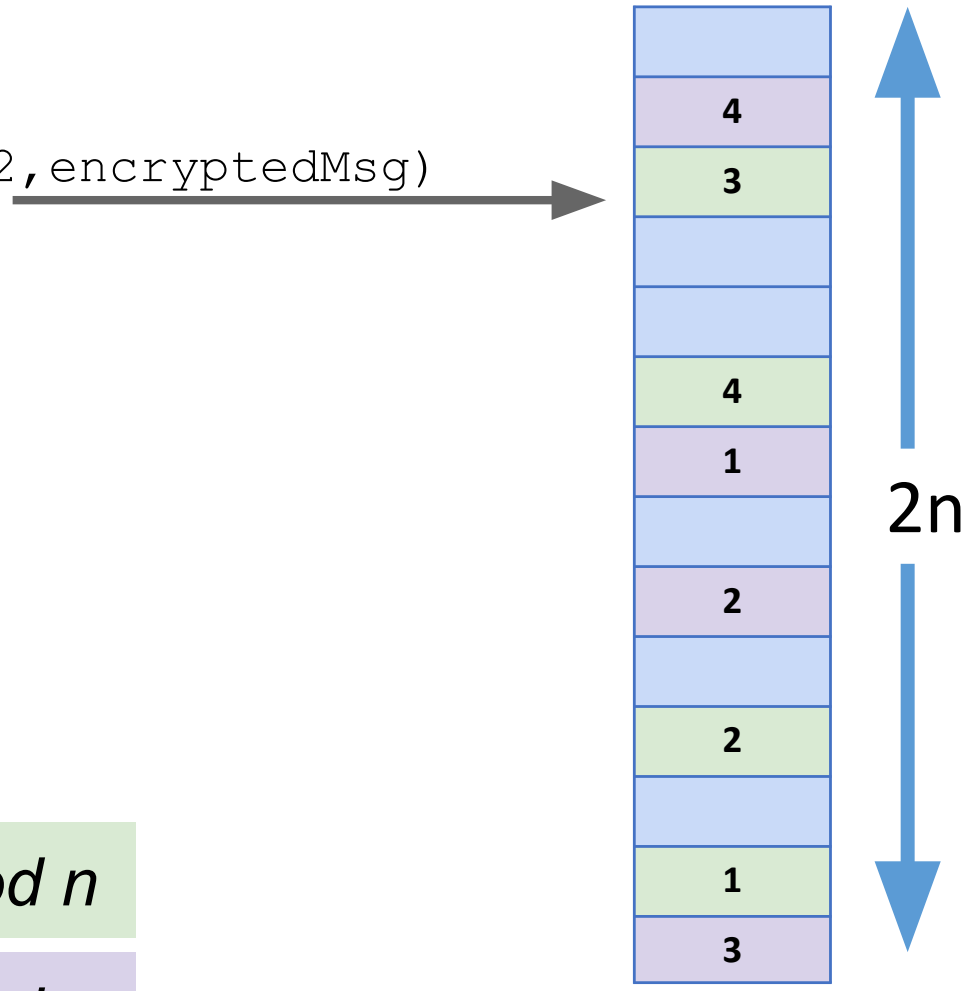
# Cuckoo Evictions

```
Write(bucket1,bucket2,encryptedMsg)
```



# Cuckoo Hashing

```
Write(bucket1,bucket2,encryptedMsg)
```




**Topic Handle:**

```
{
  topicId: uint128,
  encKey: byte[]
  seed1: uint128
  seed2: uint128
}
```

**Log Trail:**

*PRF(seed1, seqNo) mod n*

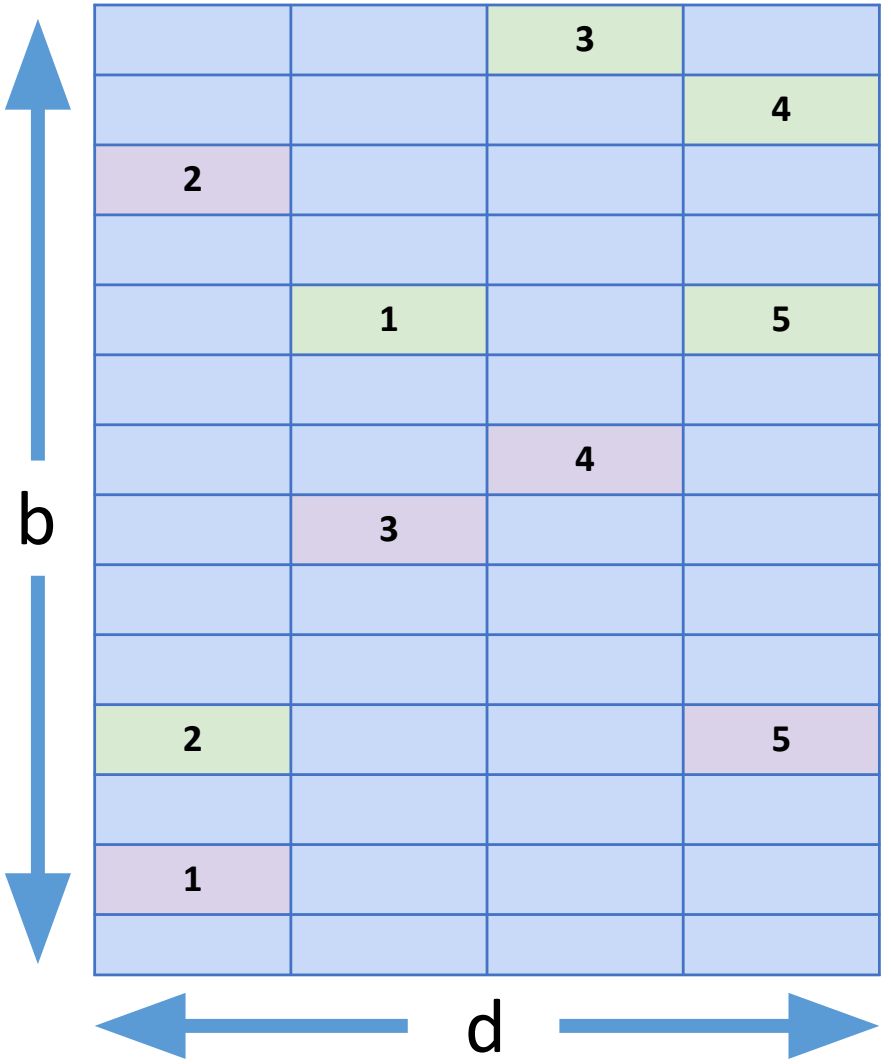
*PRF(seed2, seqNo) mod n*

# Blocked Cuckoo Table

```
{
  topicId: uint128,
  encKey: byte[]
  seed1: uint128,
  seed2: uint128
}
```

*PRF(seed1, seqNo) mod b*

*PRF(seed2, seqNo) mod b*



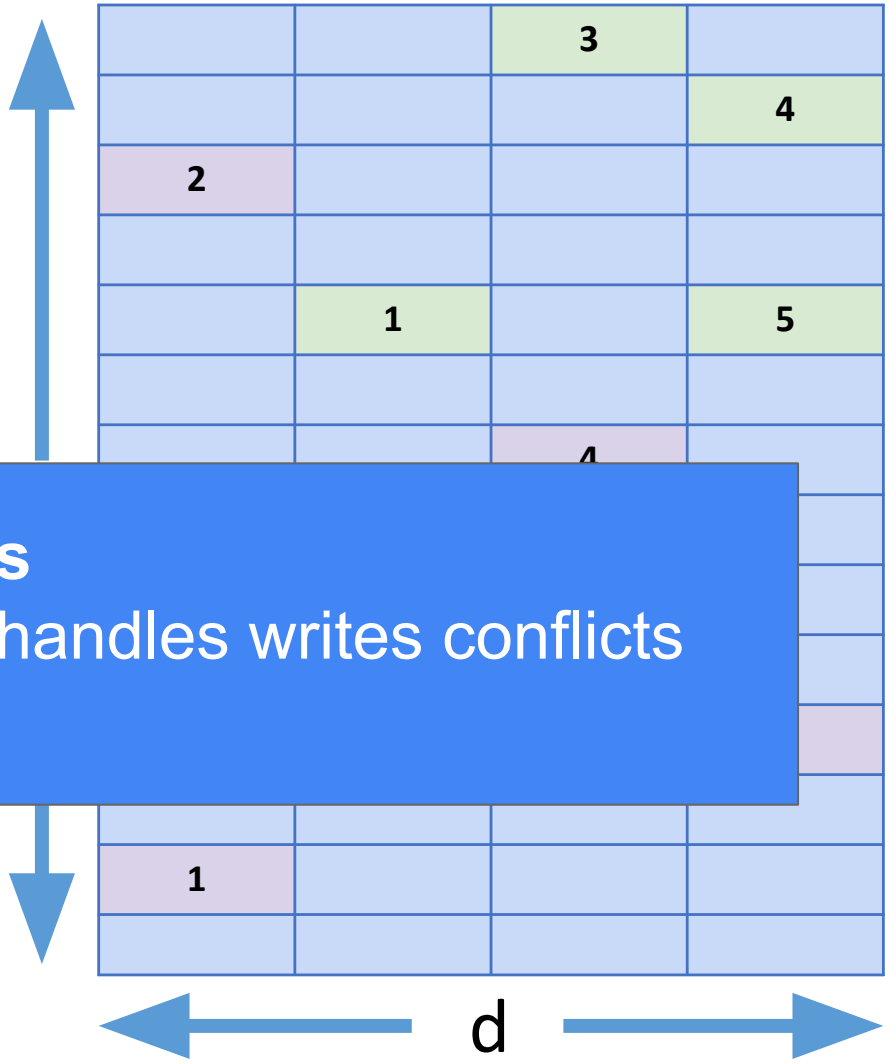
# Blocked Cuckoo Table

```
{
  topicId: uint128,
  encKey: byte[]
  seed1: uint128,
  seed2: uint128
}
```

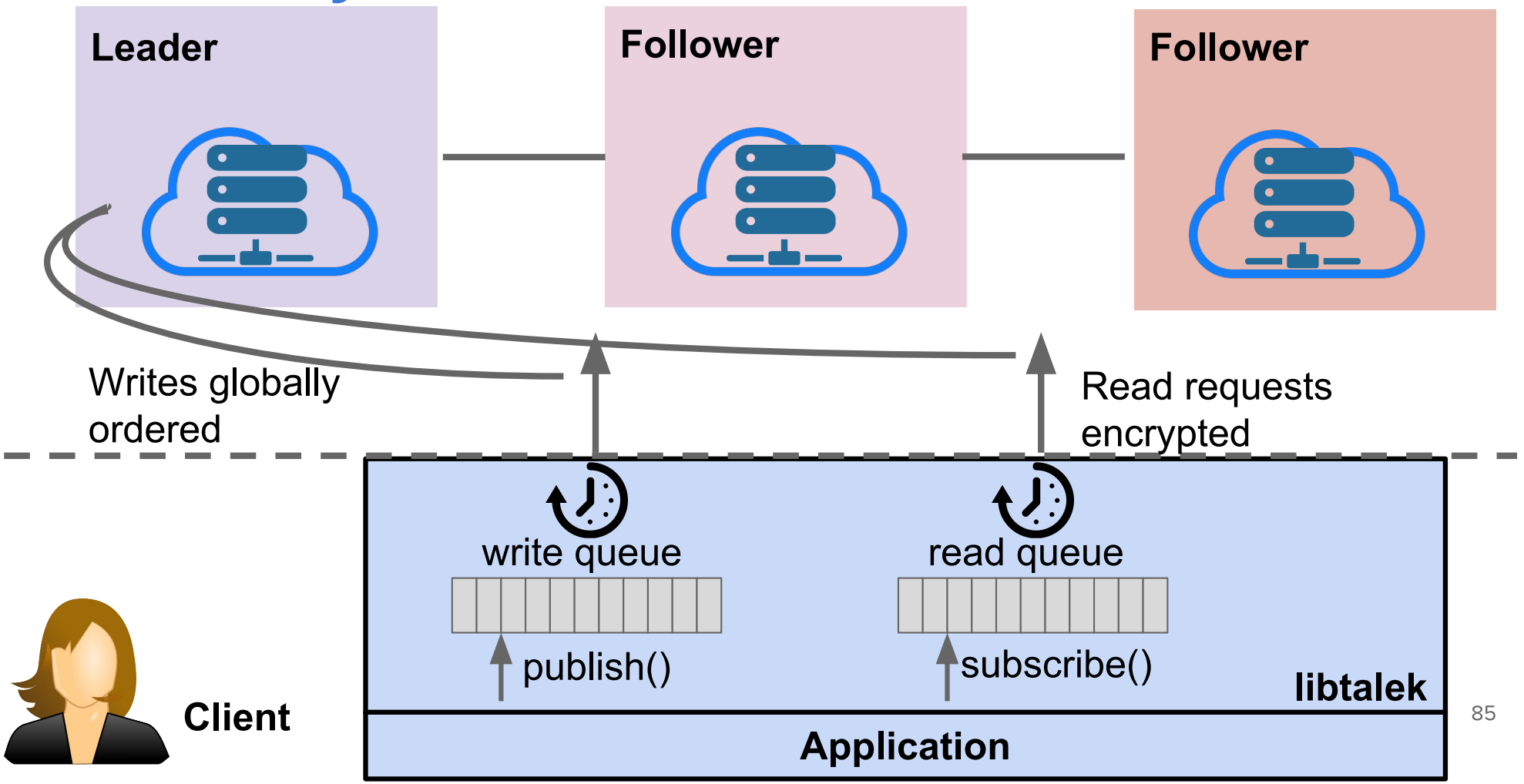
*PRF(s*

*PRF(seed2, seqNo) mod b*

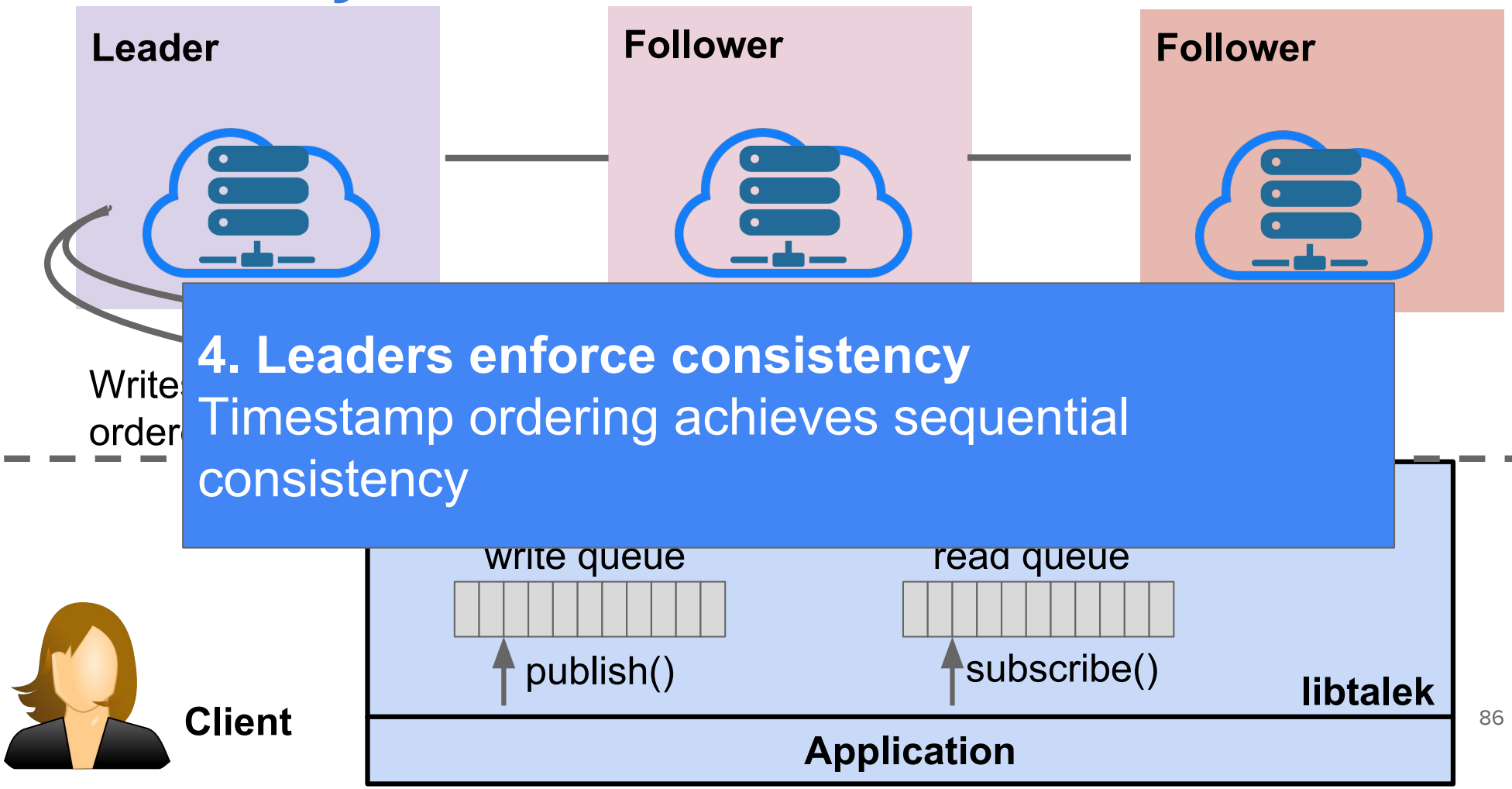
**4. Dense data structures**
Blocked cuckoo hashing handles writes conflicts with high density



# Consistency



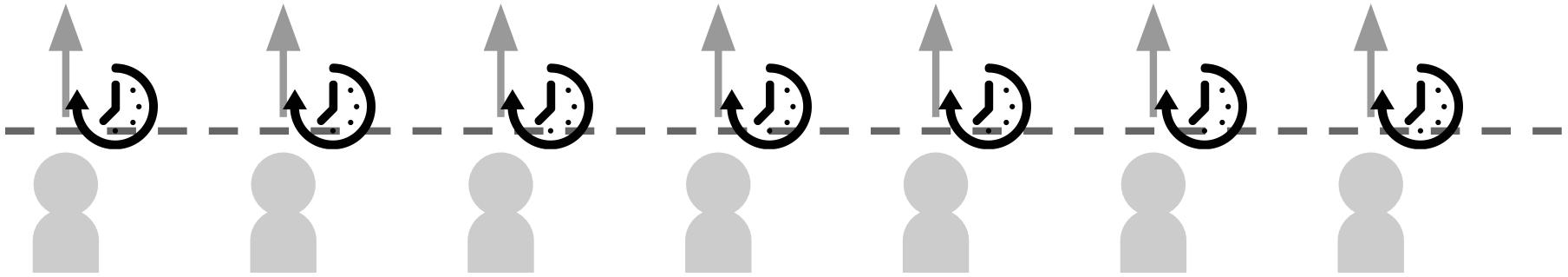
# Consistency



# Indistinguishable Writes

```
{
  topicId: uint128,
  seed1: uint128,
  seed2: uint128,
  encKey: byte[]
}
```

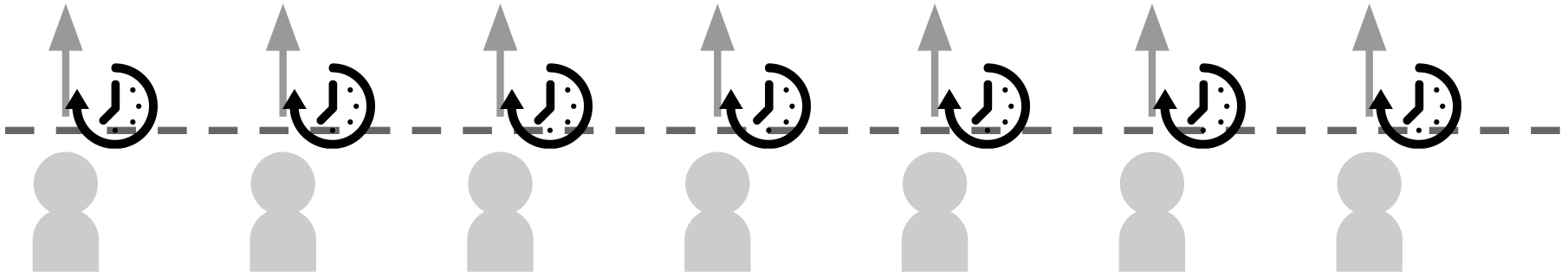
| Write | bucket1 | bucket2 | payload |
|---|---|---|---|
| Dummy | *PRF(idle, i \| 1) mod b* | *PRF(idle, i \| 2) mod b* | *Enc(idle, PRF(idle, i \| 3))* |
| Legitimate | *PRF(seed1, seqNo) mod b* | *PRF(seed2, seqNo) mod b* | *Enc(encKey, message)* |



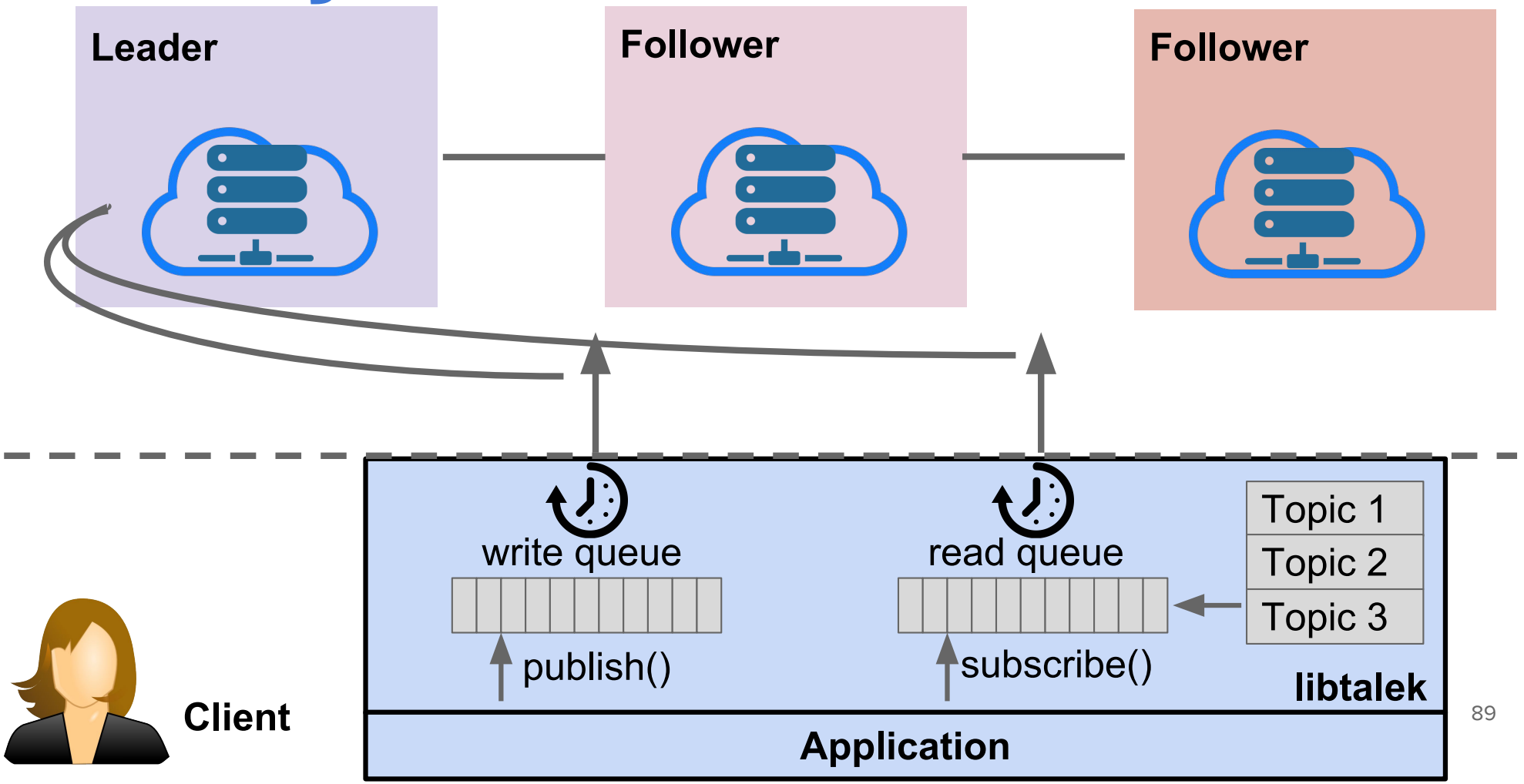
# Indistinguishable Reads

```
{
  topicId: uint128,
  seed1: uint128,
  seed2: uint128,
  encKey: byte[]
}
```

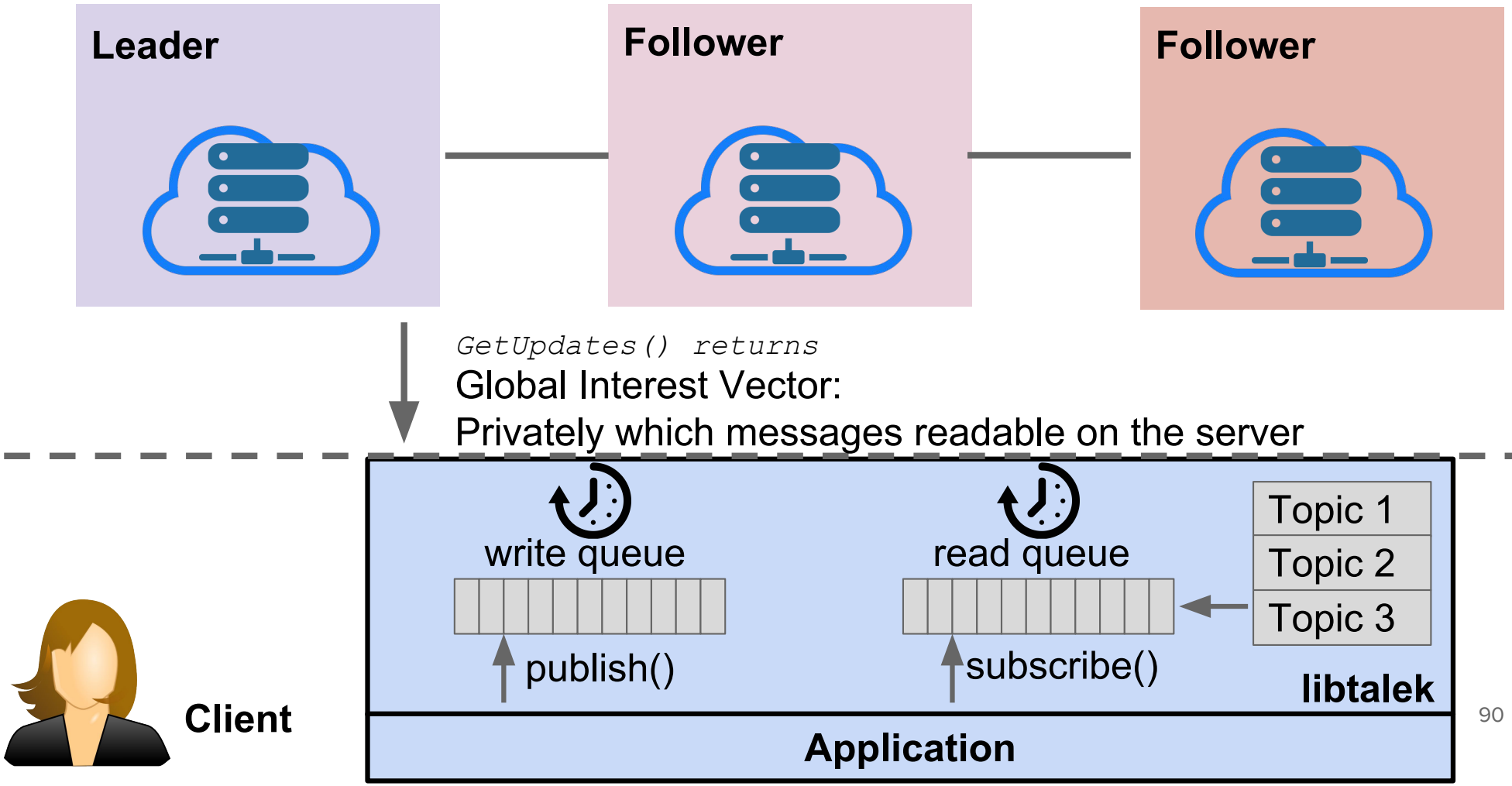
| Read | server0 | server1 | server2 |
|---|---|---|---|
| Dummy | *Enc(serverKey0, pirVector)* | *Enc(serverKey1, pirVector)* | *Enc(serverKey2, pirVector)* |
| Legitimate | *Enc(serverKey0, pirVector)* | *Enc(serverKey1, pirVector)* | *Enc(serverKey2, pirVector)* |



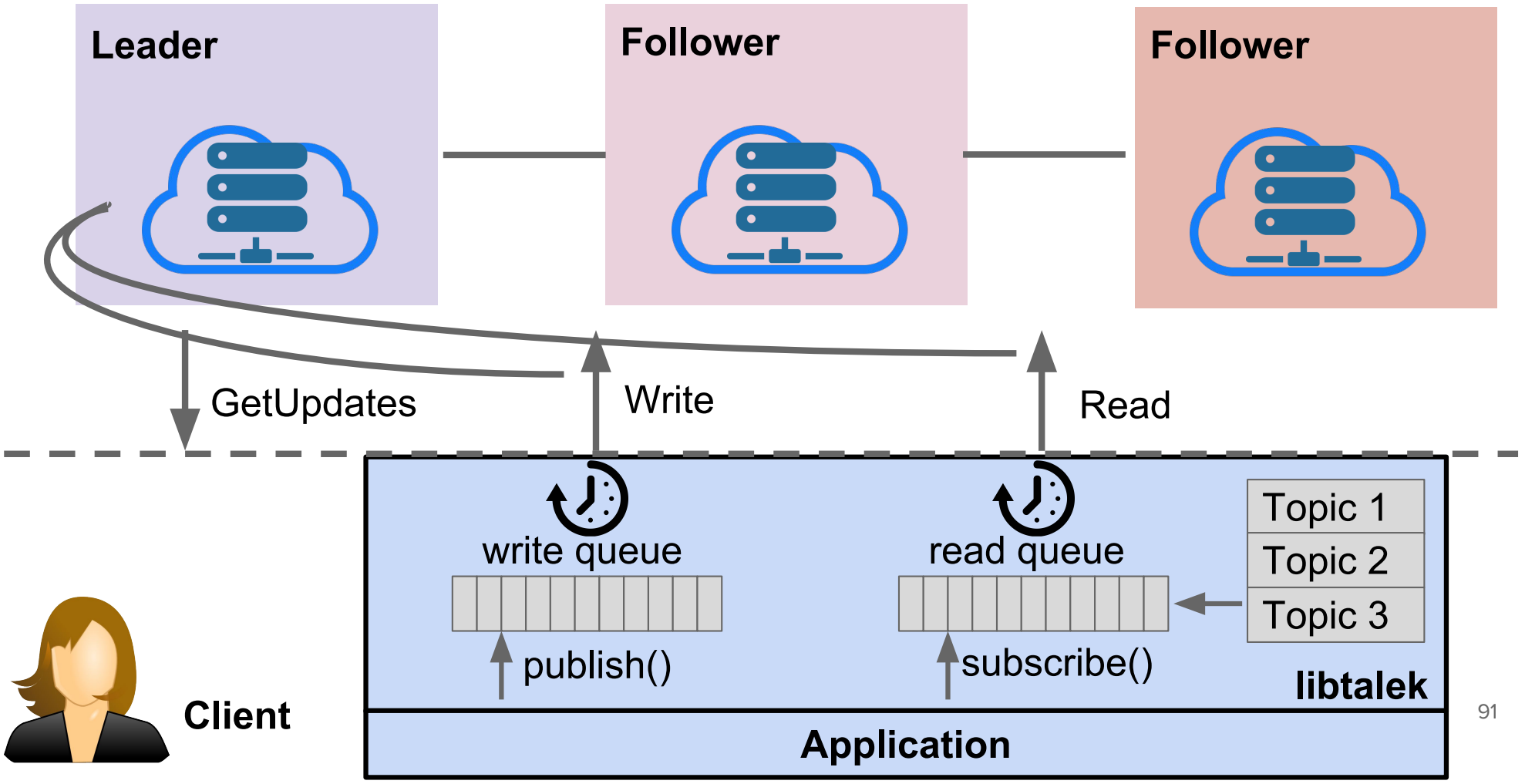
# Scheduling Reads



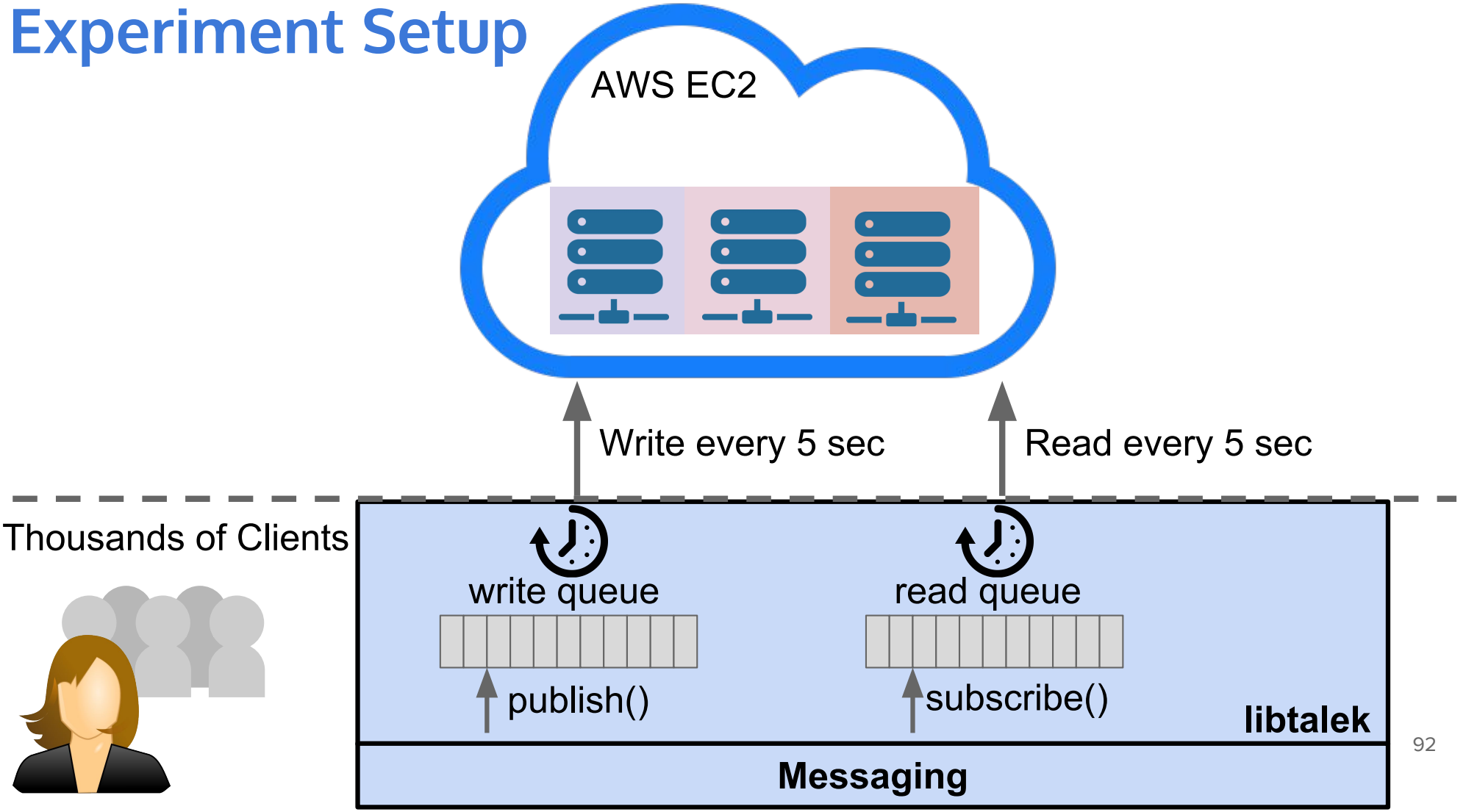
# Private Notifications



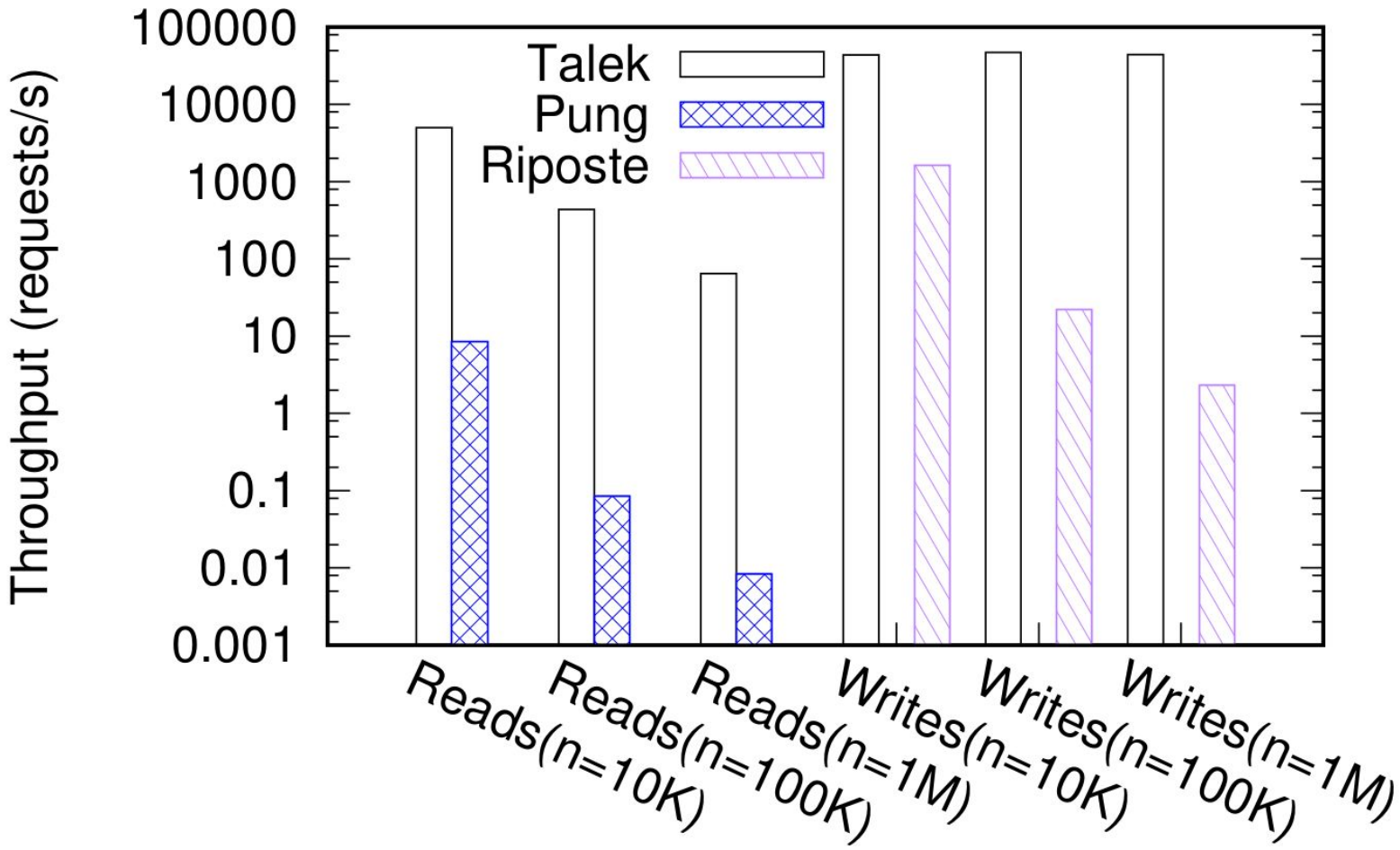
# Talek Overview



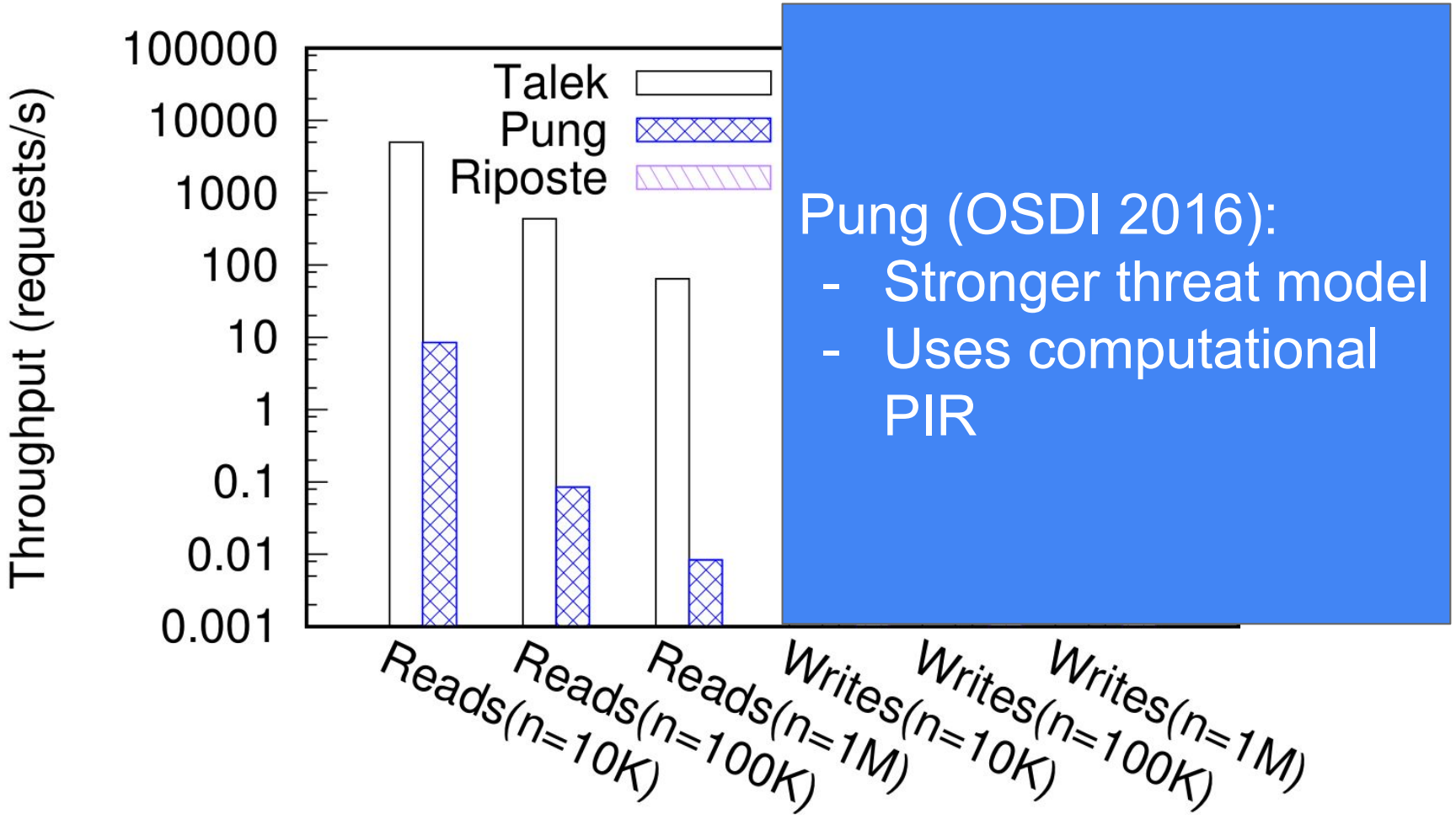
# Experiment Setup



# Comparison to Previous Work



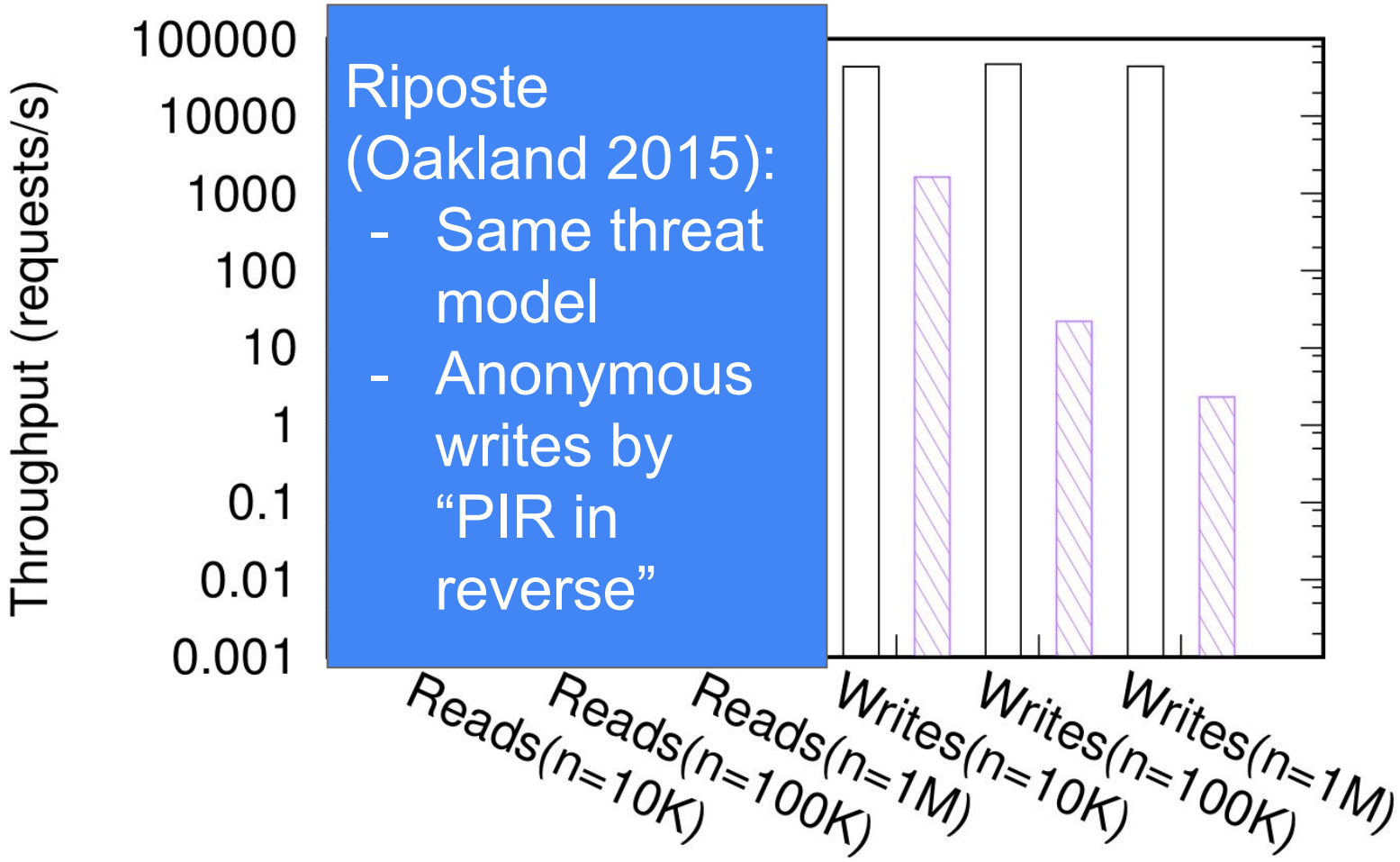
# Comparison to Previous Work




Pung (OSDI 2016):

- Stronger threat model
- Uses computational PIR

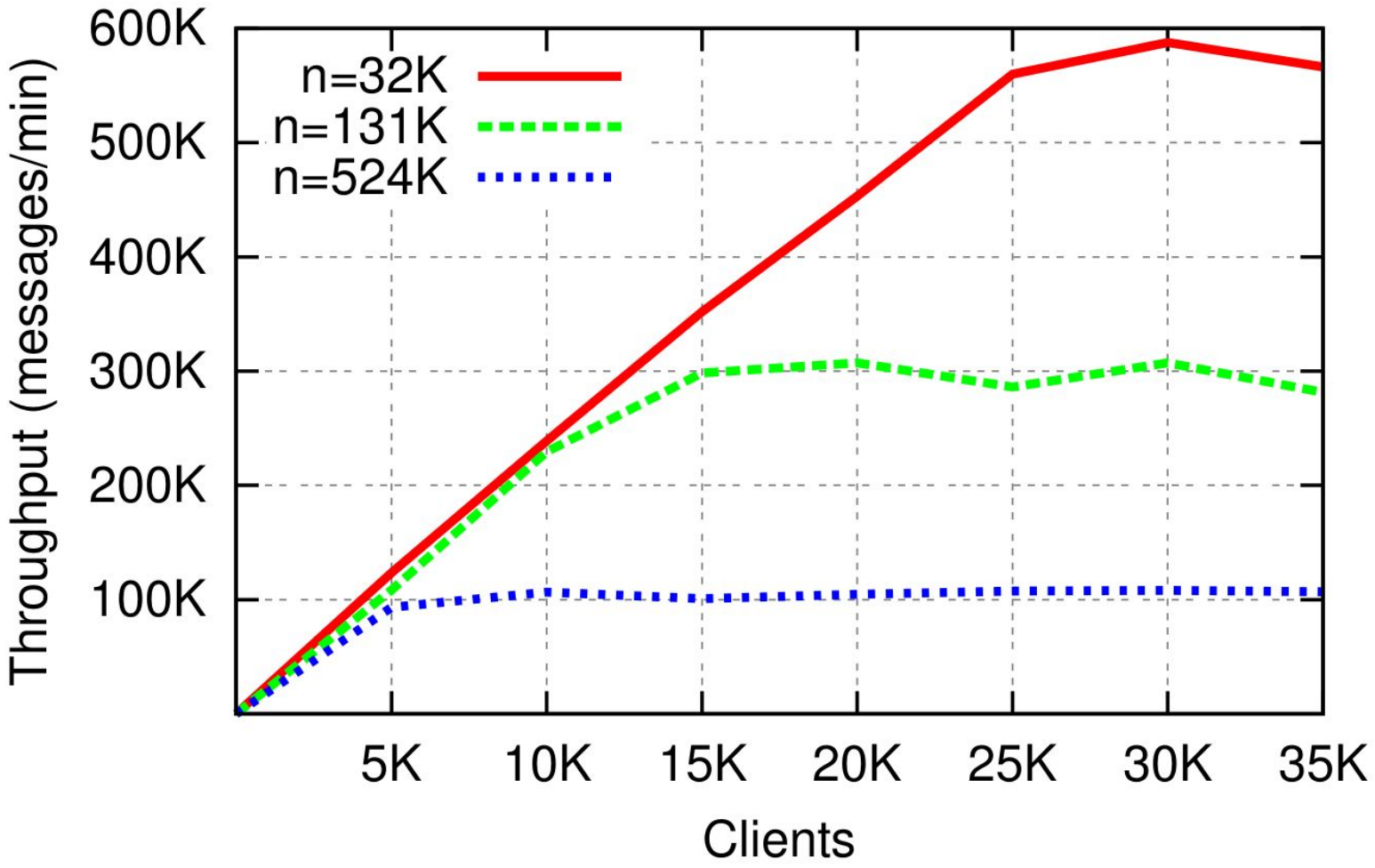
# Comparison to Previous Work

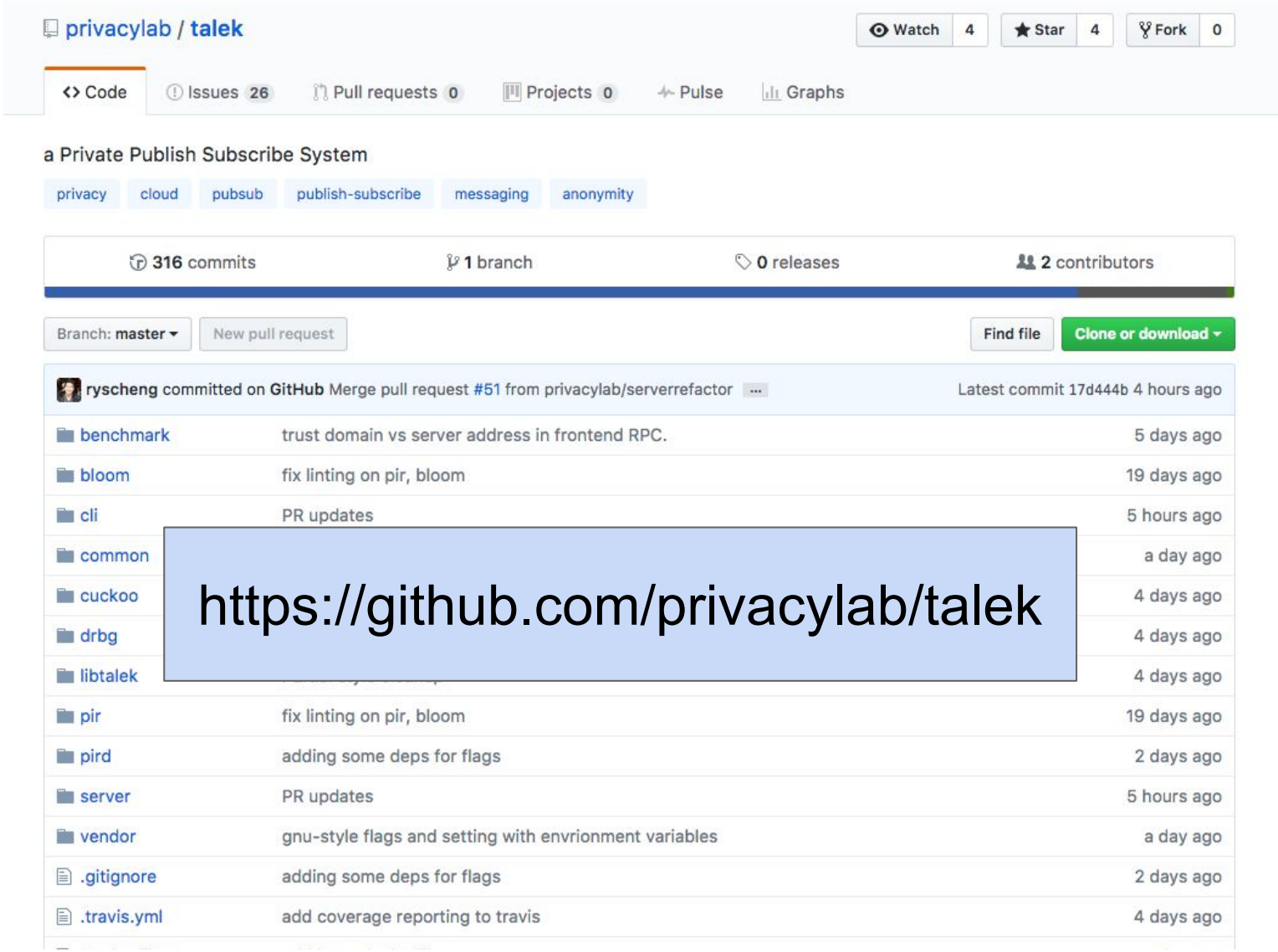



Riposte (Oakland 2015):

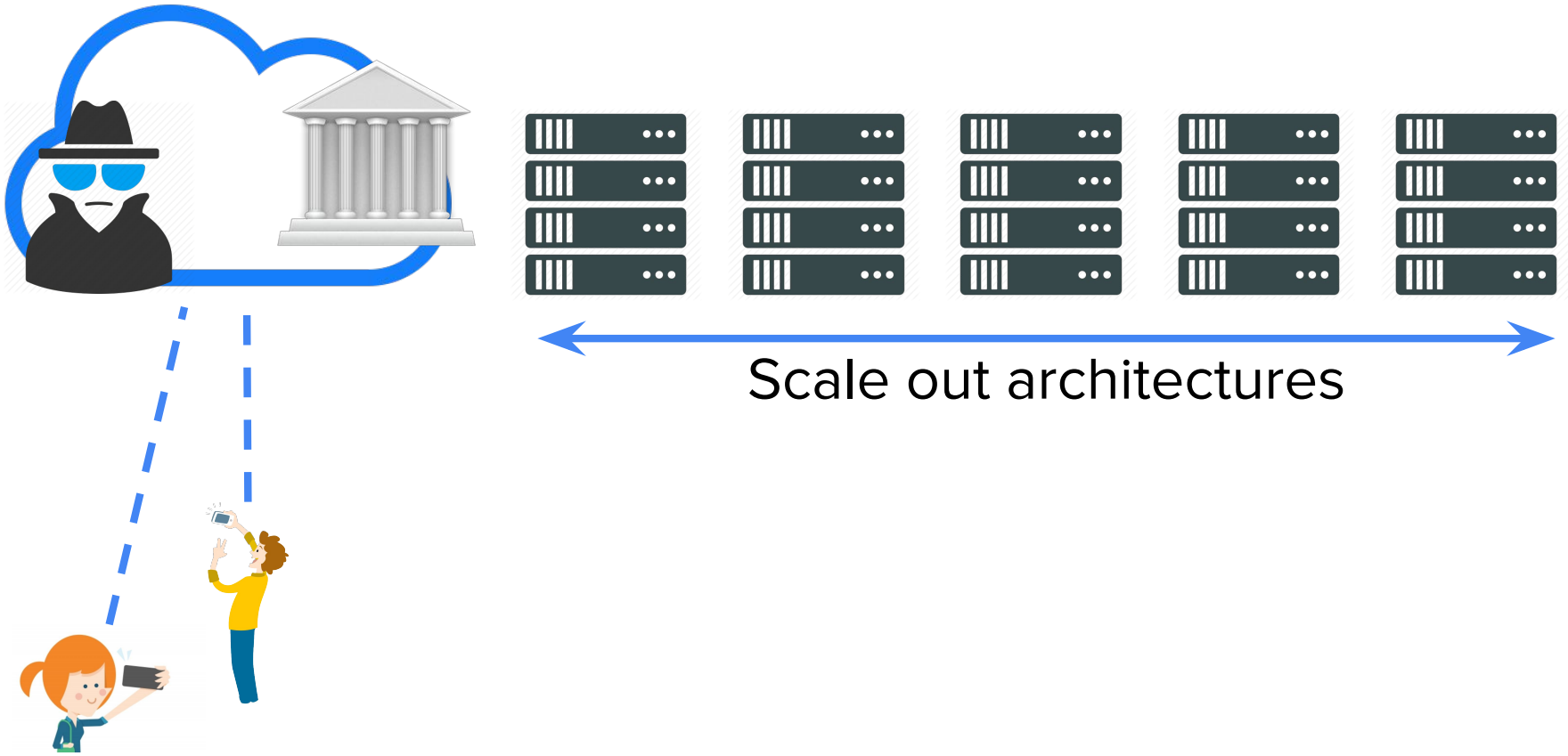
- Same threat model
- Anonymous writes by "PIR in reverse"

# Scaling Clients

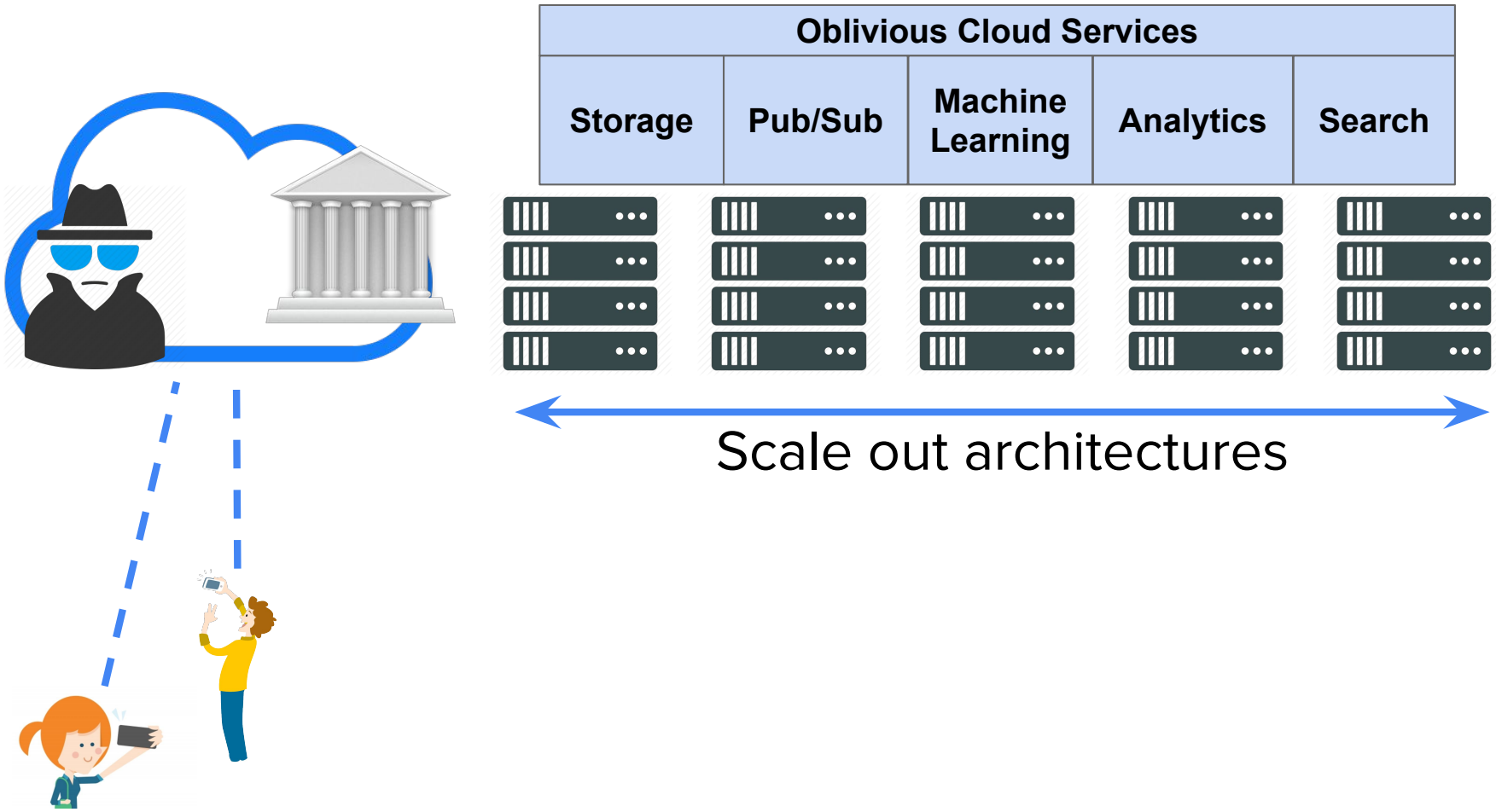



privacylab / talek
Watch 4
Star 4
Fork 0
Code
Issues 26
Pull requests 0
Projects 0
Pulse
Graphs
a Private Publish Subscribe System
privacy
cloud
pubsub
publish-subscribe
messaging
anonymity
316 commits
1 branch
0 releases
2 contributors
Branch: master
New pull request
Find file
Clone or download
ryscheng committed on GitHub Merge pull request #51 from privacylab/serverrefactor
Latest commit 17d444b 4 hours ago
benchmark trust domain vs server address in frontend RPC. 5 days ago
bloom fix linting on pir, bloom 19 days ago
cli PR updates 5 hours ago
common a day ago
cuckoo 4 days ago
drbg 4 days ago
libtalek 4 days ago
pir fix linting on pir, bloom 19 days ago
pird adding some deps for flags 2 days ago
server PR updates 5 hours ago
vendor gnu-style flags and setting with envrionment variables a day ago
.gitignore adding some deps for flags 2 days ago
.travis.yml add coverage reporting to travis 4 days ago
https://github.com/privacylab/talek

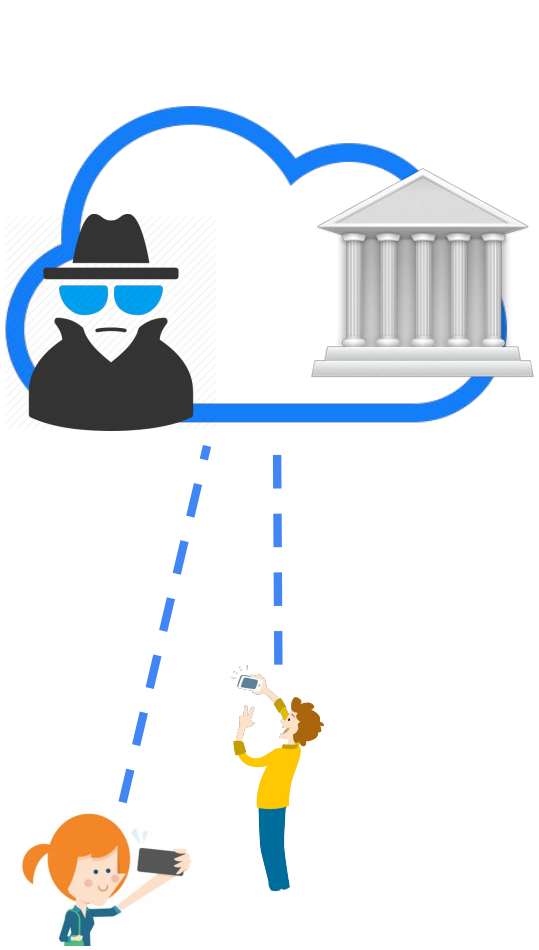
# Future Work: Scale Private Cloud Services

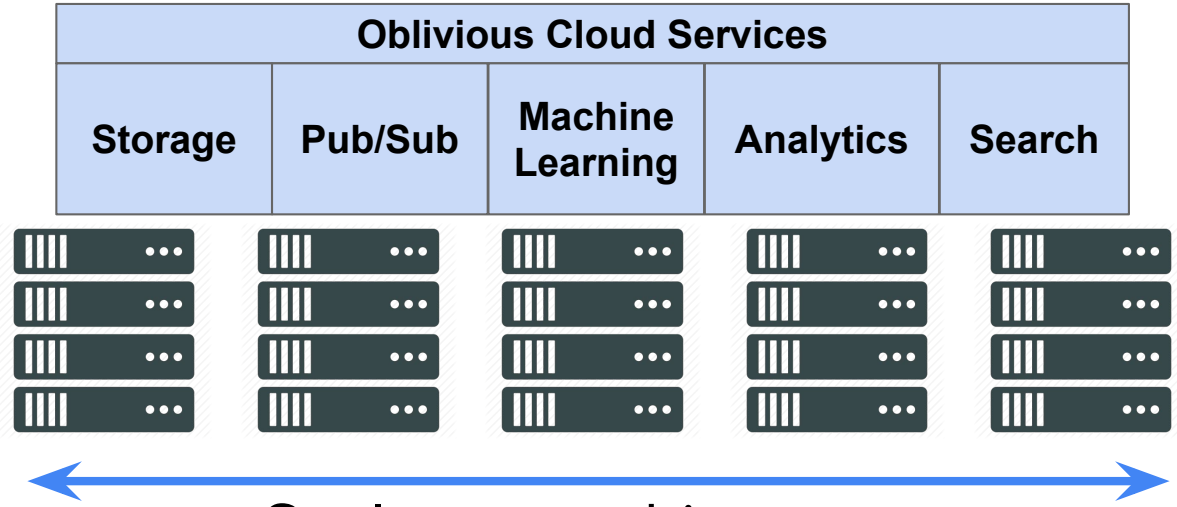


# Future Work: Support Diverse Functionality



# Future Work: Application Integration



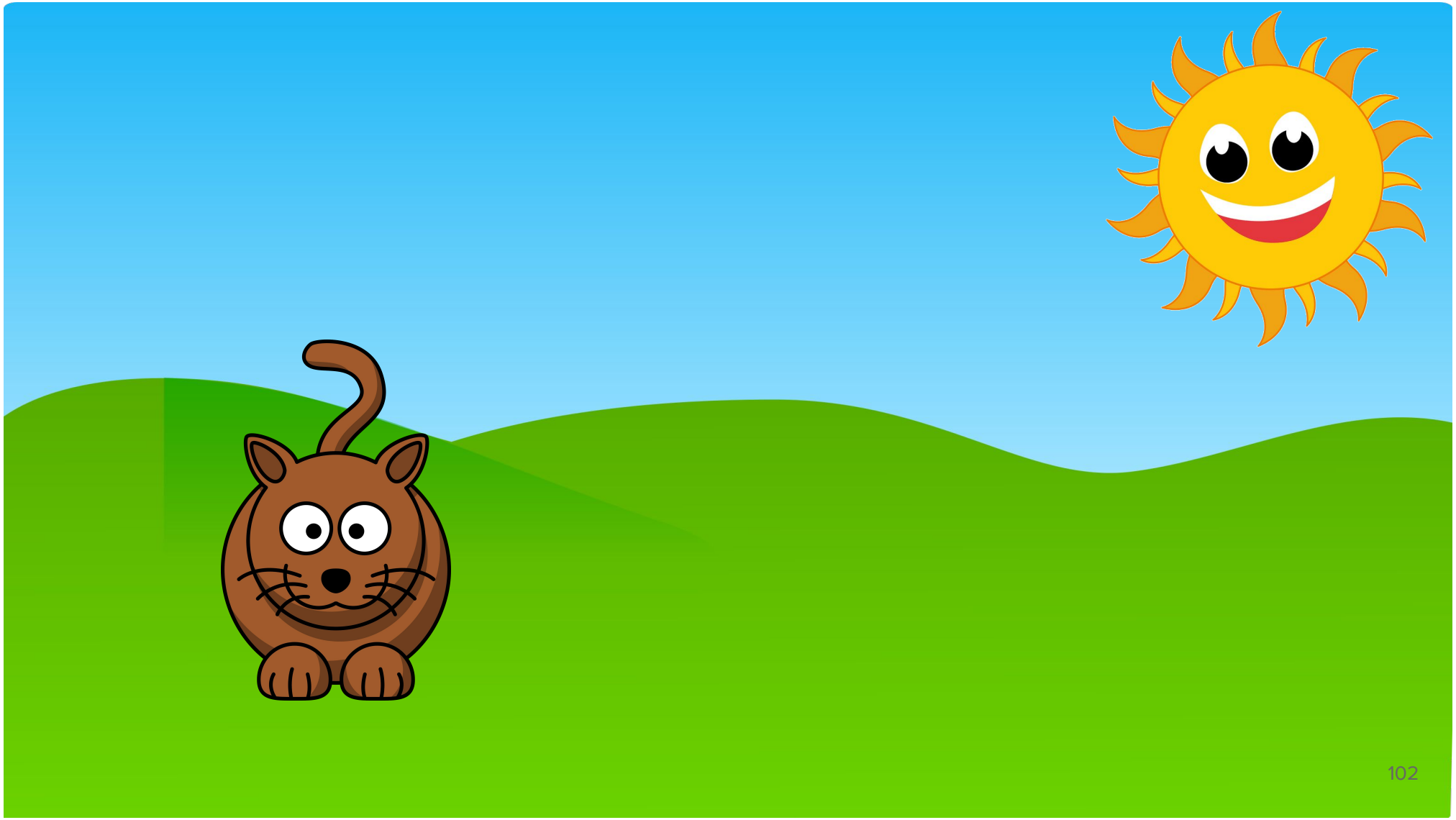



Scale out architectures


Application Integration

# Future Work


**Build practical cloud services that protect user privacy from powerful threats**



# References

[1] Cheng, R., Scott, W., Parno, B., Zhang, I., Krishnamurthy, A., Anderson, T. Talek: a Private Publish-Subscribe Protocol.
[2] Cheng, R., Scott, W., Ellenbogen, P., Howell, J., Roesner, F., Krishnamurthy, A., and Anderson, T. Radiatus: a Shared-Nothing Server-Side Web Architecture. ACM Symposium on Cloud Computing (SOCC). 2016
[3] Zhang, I., Lebeck, N., Fonseca, P., Holt, B., Cheng, R., Norberg, A., Krishnamurthy, A., Levy, H. Diamond: Automating Data Management and Storage for Wide-area, Reactive Applications. 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI). 2016.
[4] Bhoraskar, R., Langenegger, D., He, P., Cheng, R., Scott, W., and Ernst, M. User scripting on Android using BladeDroid. The 5th ACM SIGOPS Asia-Pacific Workshop on Systems (APSYS). 2014.
[5] Cheng, R., Scott, W., Krishnamurthy, A., and Anderson, T. FreeDOM: a New Baseline for the Web. The 11th ACM Workshop on Hot Topics in Networks (HotNets XI). 2012.
[6] Cheng, R., Hong, Ji., Kyrola, A., Miao, Y., Weng, X., Wu, M., Yang, F., Zhou, L., Zhao, F., and Chen, E. Kineograph: Taking the Pulse of a Fast-Changing and Connected World. Proceedings of the 7th ACM European Conference on Computer Systems (Eurosys). 2012.
[7] Scott, W., Cheng, R., Li, J., Krishnamurthy, A., and Anderson, T. Blocking Resistant Network Services using Unblock. UW Technical Report UW-CSE-14- 06-01. 2014.
[8] Cheng, R., Schueppert, M., Becker, H., and Thakur, M. SolocoRank: Social Signals for Local Search Quality. UW Technical Report UW-CSE-13-11-05. 2013.
[9] Scott, W., Cheng, R., Krishnamurthy, A., and Anderson, T. freedom.js: an Architecture for Serverless Web Applications UW Technical Report. UW-CSE-13-05- 03. 2013.
[10] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private Information Retrieval. Journal of the ACM (JACM), 45(6):965–981, 1998

# Talek Related Work

| System | Security Goal | Threat Model | Technique | Application |
|---|---|---|---|---|
| Talek | indistinguishability | ≥1 | IT-PIR | pub/sub |
| Pynchon Gate | k-anonymity | ≥1 | mixnet/IT-PIR | email |
| Riffle | k-anonymity | ≥1 | mixnet/IT-PIR | file-sharing |
| Riposte | k-anonymity | ≥1 | IT-PIR | broadcast |
| Dissent | k-anonymity | ≥1 | DC-nets | broadcast |
| Vuvuzela | differential privacy | ≥1 | mixnet | 1-1 messaging |
| DP5 | indistinguishability | ≥1 | IT-PIR | chat presence |
| Popcorn | indistinguishability | ≥1 | C-PIR/IT-PIR | video streaming |
| Pung | indistinguishability | 0 | C-PIR | key-value store |
| ORAM | indistinguishability | 0 | ORAM | storage |