Private Blocklist Lookups with Checklist

Dima Kogan
Stanford

Henry Corrigan-Gibbs
MIT

USENIX Security Symposium 2021

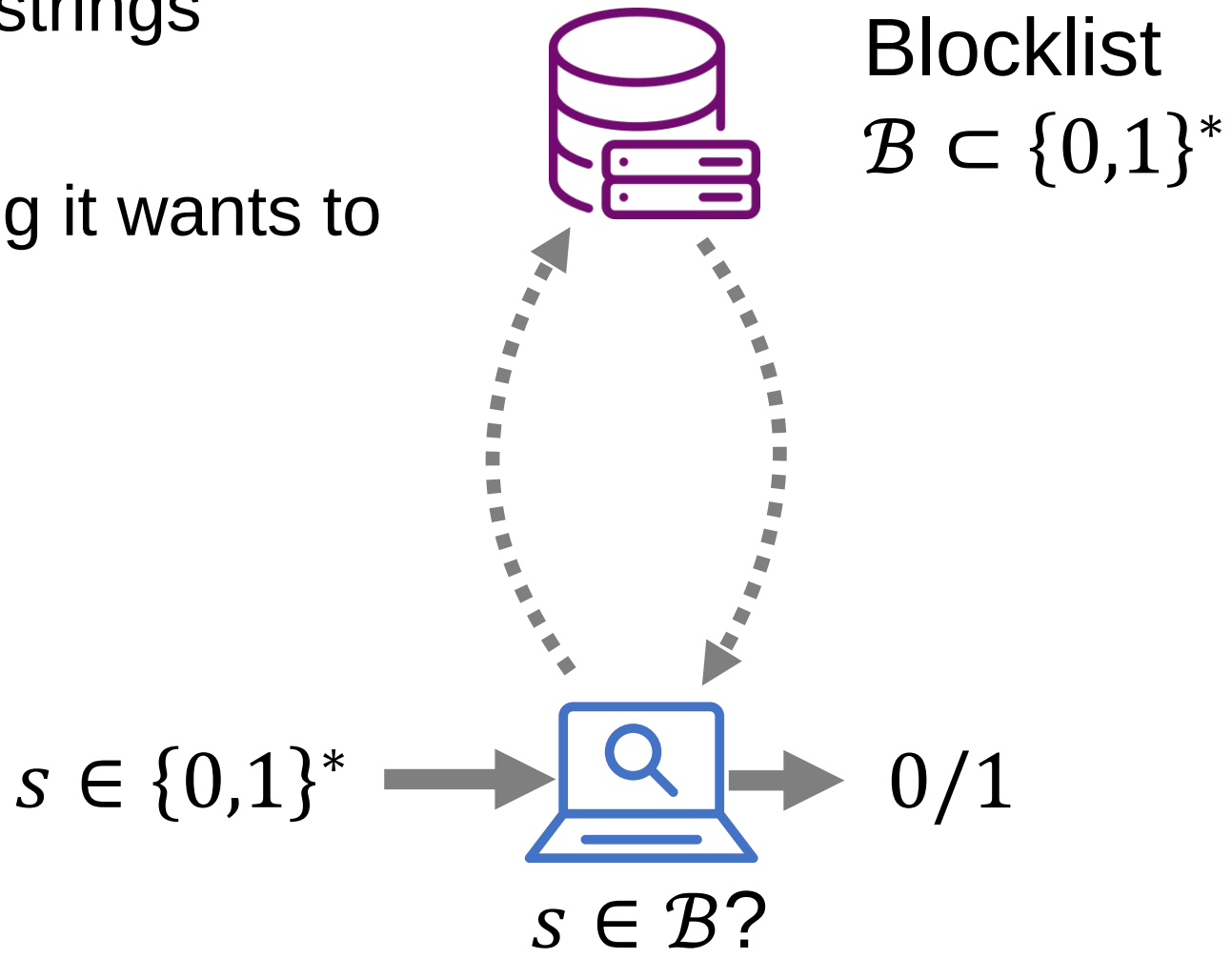
Blocklist lookup

Server holds a blocklist of strings

Client holds a **private** string it wants to check against the blocklist

Examples:

- Certificate revocation
- Password checkup
- Safe Browsing



Safe Browsing

Leaks information about user's browsing history
[GKL16,Per19,BK20]

Firefox browser

uyghuracademy.com/

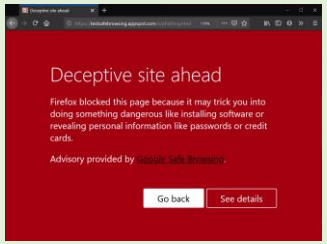
Partial hashes

0x104
0x130
0x1F3
0x1FF
0x24C
0x2B2
...

Lookup
0x24C

Full hash
0x24C1A8…

Warn



Google Safe Browsing

Hashes of ~3M dangerous URLs

0x104A158413B9…
0x130CA8F45BE3…
0x1F346FFD10DA…
0x1FF60910E4BF1…
0x24CC8335003A…
0x2B26ED8D140B…
…

This work

Checklist – a system for private blocklist lookups

Builds on offline/online private information retrieval [CK20]

- Allows for sublinear online server time

Contributions:

- New offline/online PIR (reduces server computation by >100x)
- A technique to efficiently support database updates
- Implementation & evaluation of a private Safe Browsing system

Requirements

Correctness

Client learns whether $s \in \mathcal{B}$ (with overwhelming prob.)

Privacy for the client

Non-goal: privacy for the server

(see paper for discussion of this extension)

(Malicious) server “learns nothing” about client’s string

Efficiency

Minimize latency, communication, computation, storage

Two-server private information retrieval [CGKS95,…]

Server computation is large
Servers needs to do $\Omega(n)$ work [BIM04]

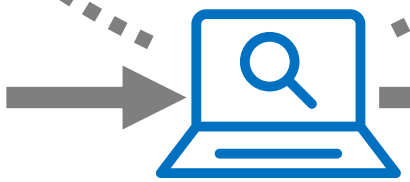
$x \in \{0,1\}^n$

$x \in \{0,1\}^n$

Communication is small
$O(\log n)$ from DPFs [GI14, BGI15]
(Distributed Point Functions)

Security
Each server **on its own** learns nothing about i

Index
$i \in [n]$



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

$[n] = \{1, \dots, n\}$

Offline/Online PIR [CK20]

Step 1: Offline phase

$x \in \{0,1\}^n$

$x \in \{0,1\}^n$

$O(n)$ time

Hint $\approx \lambda\sqrt{n}$ bits

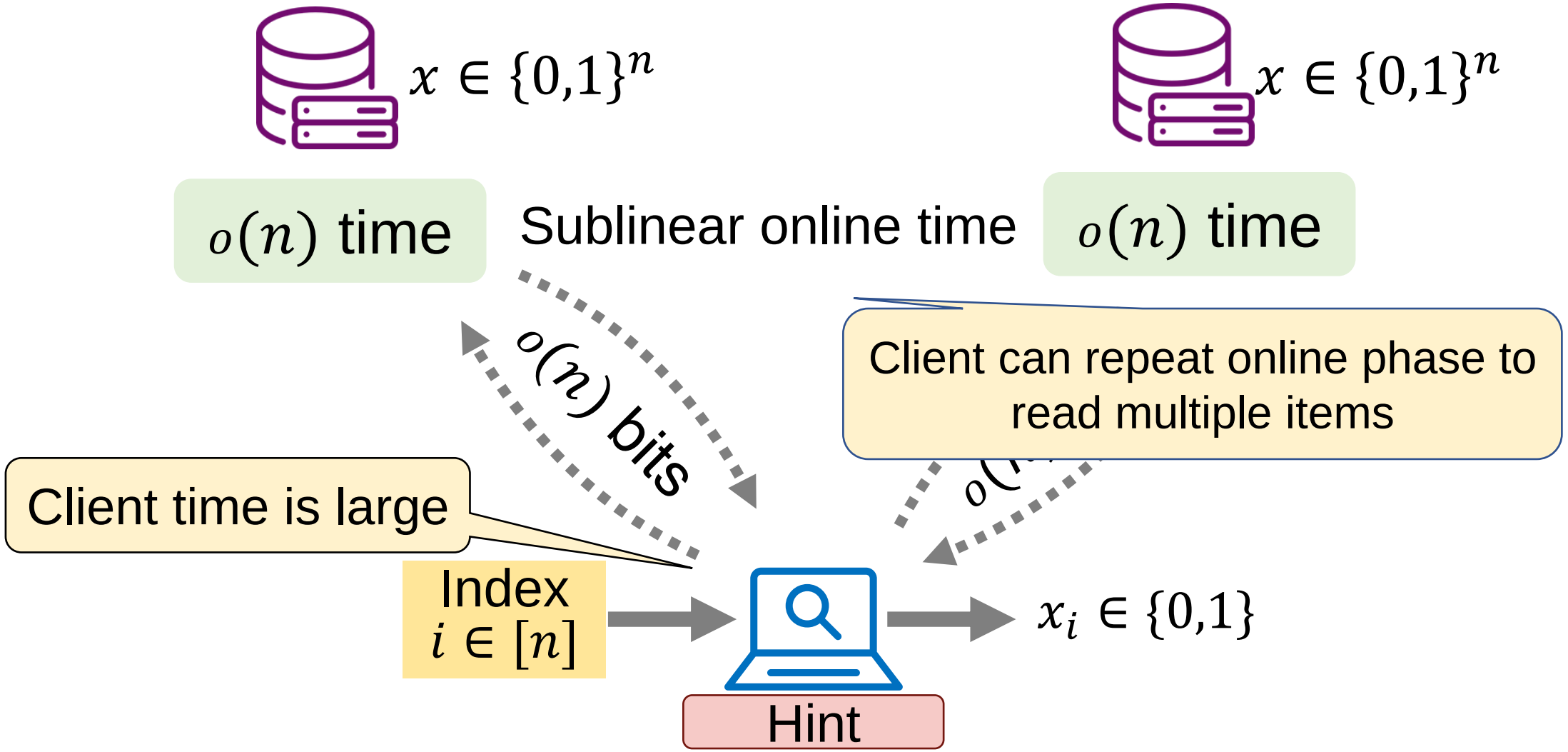
$\lambda\sqrt{n}$ bits

- The left server runs in linear time.
- But only once per client.

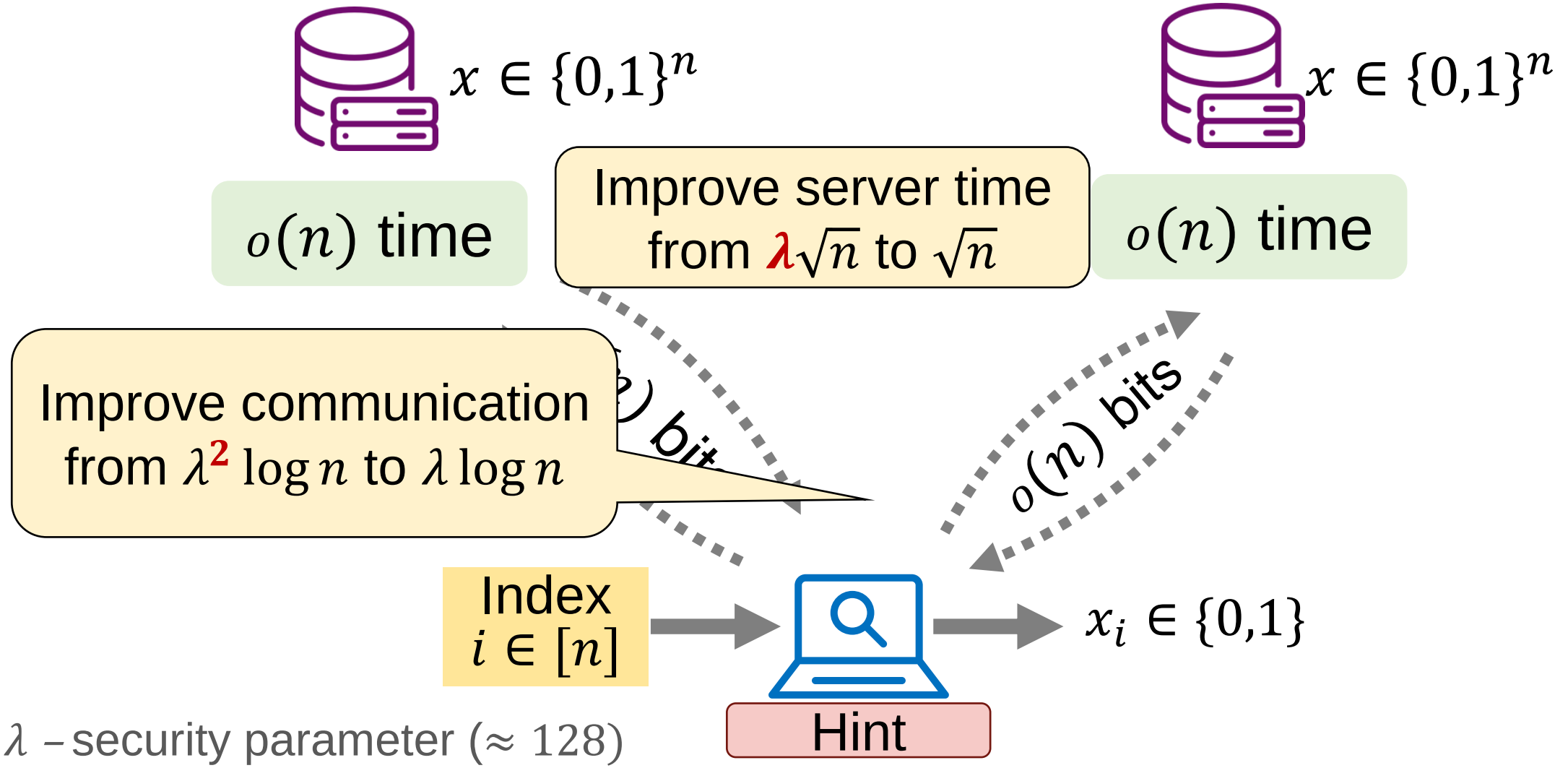
λ – security parameter (≈ 128)

Offline/Online PIR [CK20]

Step 2: Online phase – reading x_i



Our first contribution:
$\lambda\times$ faster offline/online PIR



λ – security parameter (≈ 128)

Our second contribution:
Offline-online PIR with DB updates

When the database changes, client needs a new hint

Naïve approach: rerun the offline phase after each change

- Linear amount of server work on each change

Refined approach: incremental preprocessing

- Use “buckets” of exponentially increasing size
- Logarithmic amount of server work on each change

Static-to-dynamic data structures [BS80,GO96,CKO90,SSP13,SPS14,PT16]

Our third contribution:
Private Safe Browsing

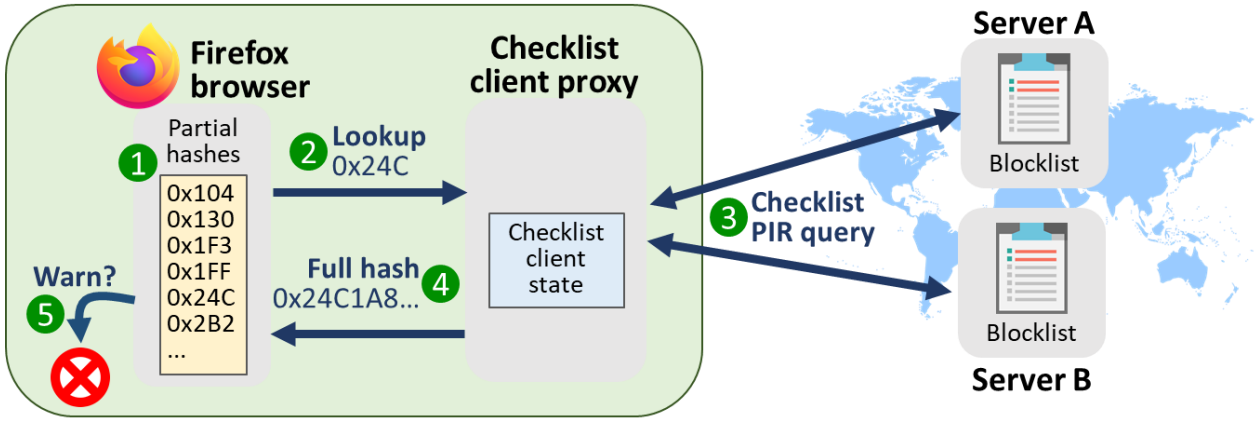
Checklist: a system for private Safe Browsing queries, integrated and evaluated with Firefox browser

Implementation

Roughly 2500 lines of Go + 500 lines of C

Support offline/online and DPF-based PIR

Browser integration: local proxy + browser config change



Estimating the Safe Browsing parameters

Monitor a week of Safe Browsing requests and responses

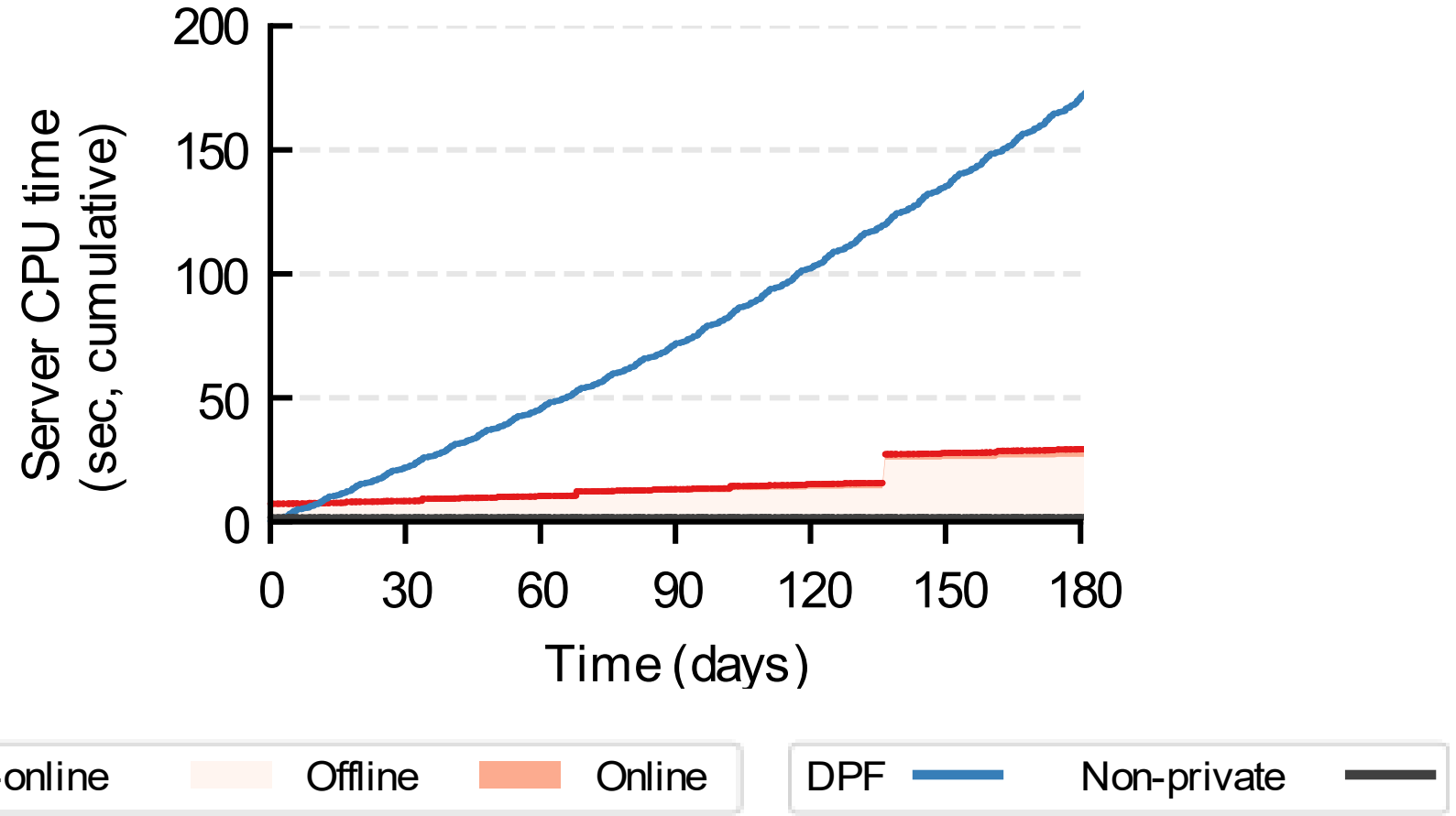
Local proxy forwards requests to the real Safe Browsing service

Deduce:

- Frequency of lookups
- Frequency of updates
- Database growth

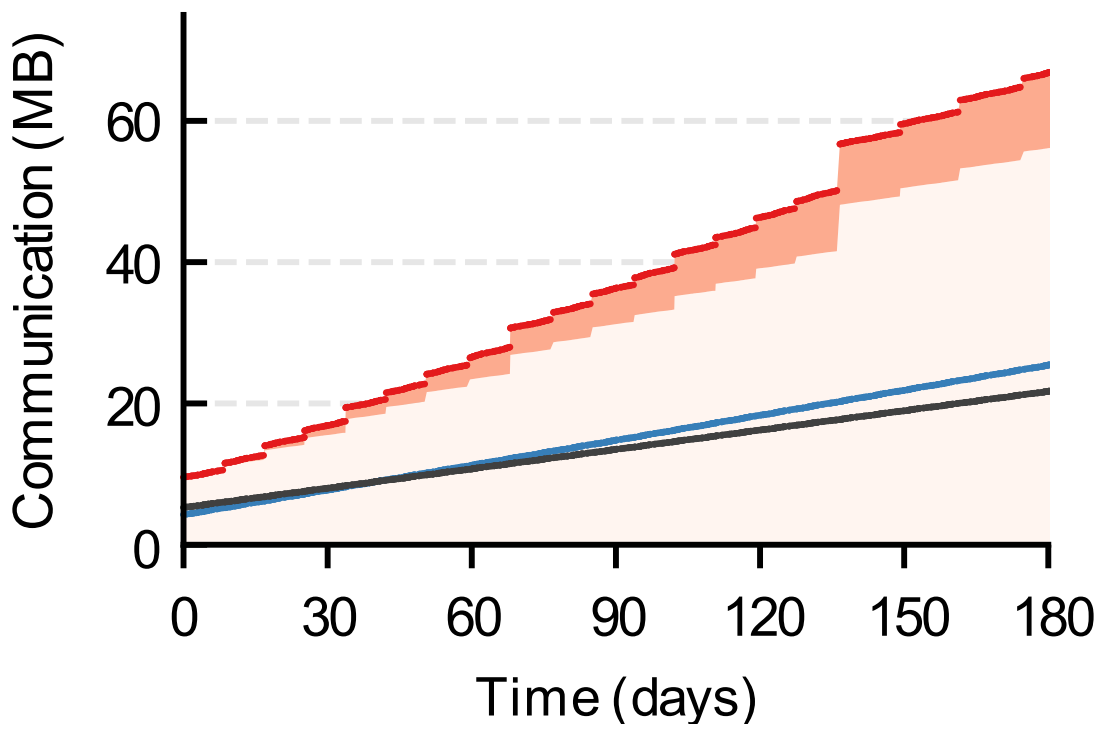
Checklist with offline/online PIR is more efficient for the server

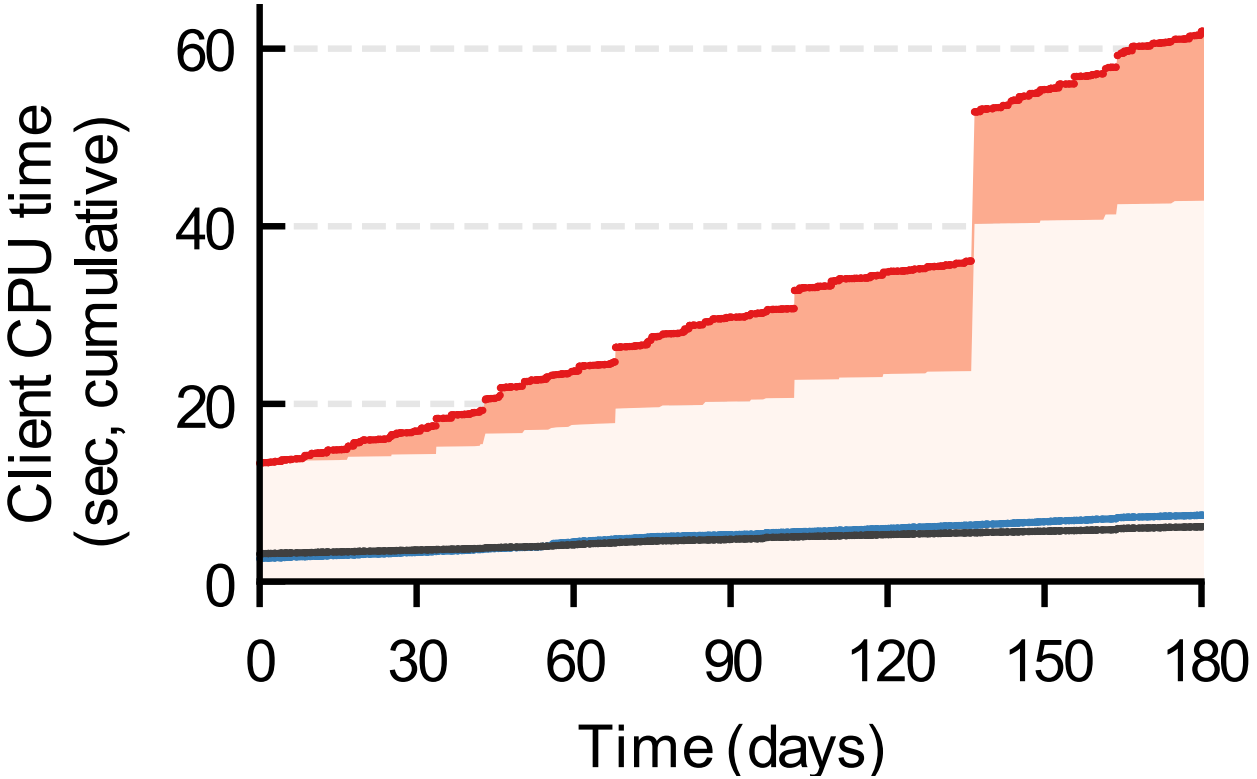
Replay recorded trace of a single user

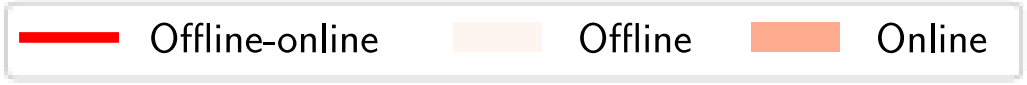


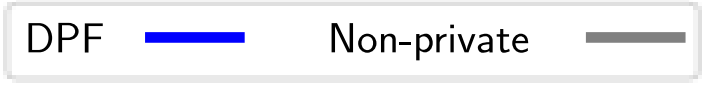
Server running on e2-standard-4 Google Compute Engine machine (4 vCPUs, 16 GB RAM)

Checklist with DPF-based PIR is more efficient for the client









Client running on a Pixel 5 mobile phone

Overview of results

Approach	Server Costs	Latency	Client computation		Communication		Client Storage	
	(servers per 1B users)	(ms)	Initial (sec)	Running (sec/month)	Initial (MB)	Running (MB/month)	Initial (MB)	Running (MB/month)

Overview of results

Approach	Server Costs	Latency	Client computation		Communication		Client Storage	
	(servers per 1B users)	(ms)	Initial (sec)	Running (sec/month)	Initial (MB)	Running (MB/month)	Initial (MB)	Running (MB/month)
Non-private	143	91	3.1	0.5	5.0	3.0	4.3	0.2

Overview of results

Approach	Server Costs	Latency	Client computation		Communication		Client Storage	
	(servers per 1B users)	(ms)	Initial (sec)	Running (sec/month)	Initial (MB)	Running (MB/month)	Initial (MB)	Running (MB/month)
Non-private	143	91	3.1	0.5	5.0	3.0	4.3	0.2
Full list	Very small – not measured				91.8	13.2	91.8	4.5

Overview of results

Approach	Server Costs	Latency	Client computation		Communication		Client Storage	
	(servers per 1B users)	(ms)	Initial (sec)	Running (sec/month)	Initial (MB)	Running (MB/month)	Initial (MB)	Running (MB/month)
Non-private	143	91	3.1	0.5	5.0	3.0	4.3	0.2
Full list	Very small – not measured				91.8	13.2	91.8	4.5
DPF PIR	9047	122	2.6	0.8	5.0	3.6	4.3	0.2

Overview of results

Approach	Server Costs	Latency	Client computation		Communication		Client Storage	
	(servers per 1B users)	(ms)	Initial (sec)	Running (sec/month)	Initial (MB)	Running (MB/month)	Initial (MB)	Running (MB/month)
Non-private	143	91	3.1	0.5	5.0	3.0	4.3	0.2
Full list	Very small – not measured				91.8	13.2	91.8	4.5
DPF PIR	9047	122	2.6	0.8	5.0	3.6	4.3	0.2
Offline/online PIR	1348	90	13.3	8.0	10.3	9.8	24.5	1.6

Conclusion & future work

Two-server PIR is a practical tool for privacy preserving systems

- Several alternatives that allow for different trade-offs

Future direction: better single-server offline/online PIR

Conclusion & future work

Two-server PIR is a practical tool for privacy preserving systems

- Several alternatives that allow for different trade-offs

Future direction: better single-server offline/online PIR

dkogan@cs.stanford.edu henrycg@csail.mit.edu

https://eprint.iacr.org/2021/345

https://github.com/dimakogan/checklist