PIR with Nearly Optimal Online Time and Bandwidth

Elaine Shi (CMU)

Joint work with Aqeel, Chandrasekaran, and Maggs

To appear in CRYPTO'21

Oblivious DNS Deployed by Cloudflare and Apple

Nick Feamster Follow

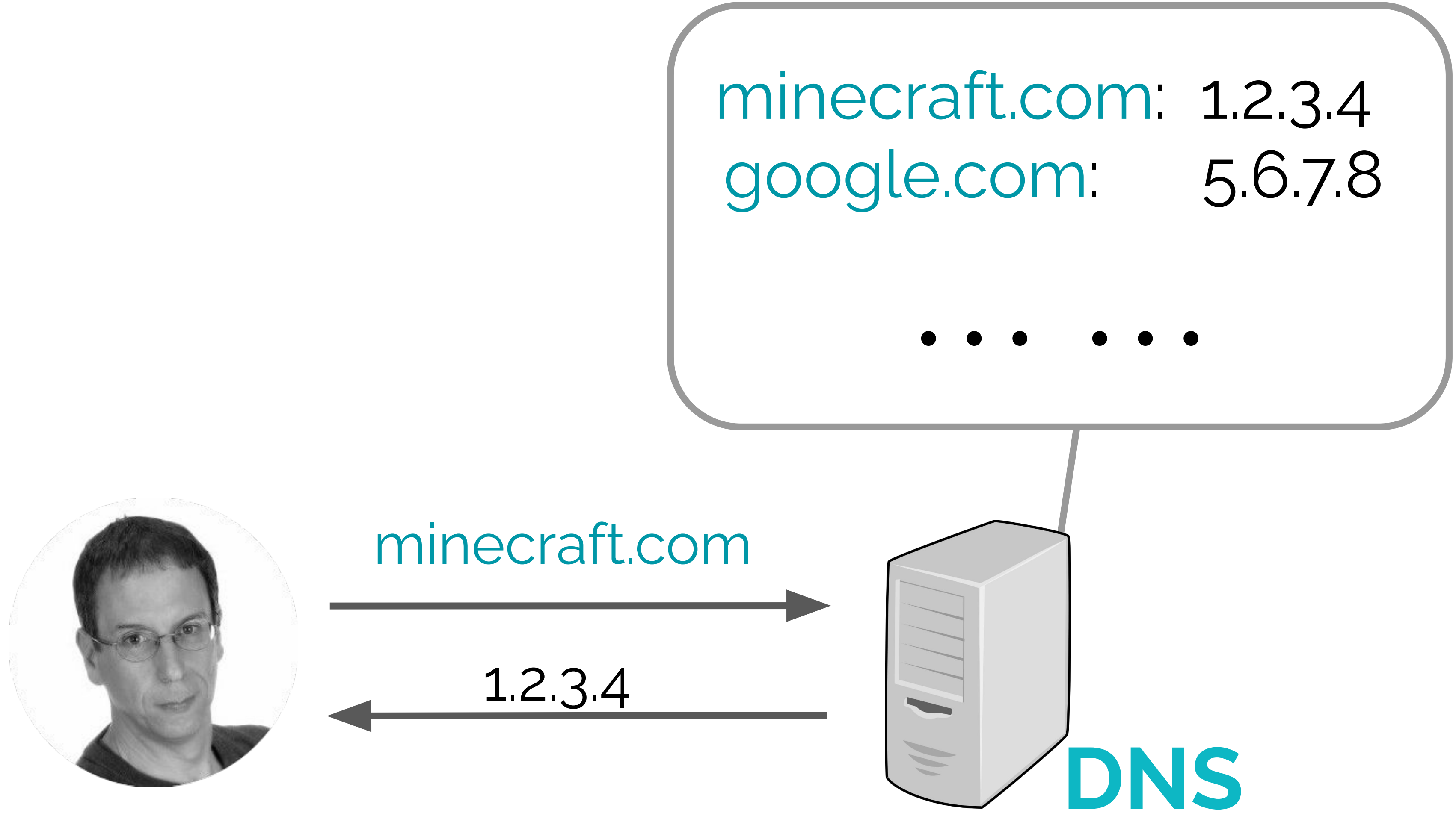
Dec 8, 2020 · 5 min read ★

Enable Private DNS with 1.1.1.1 on Android 9 Pie

08/16/2018

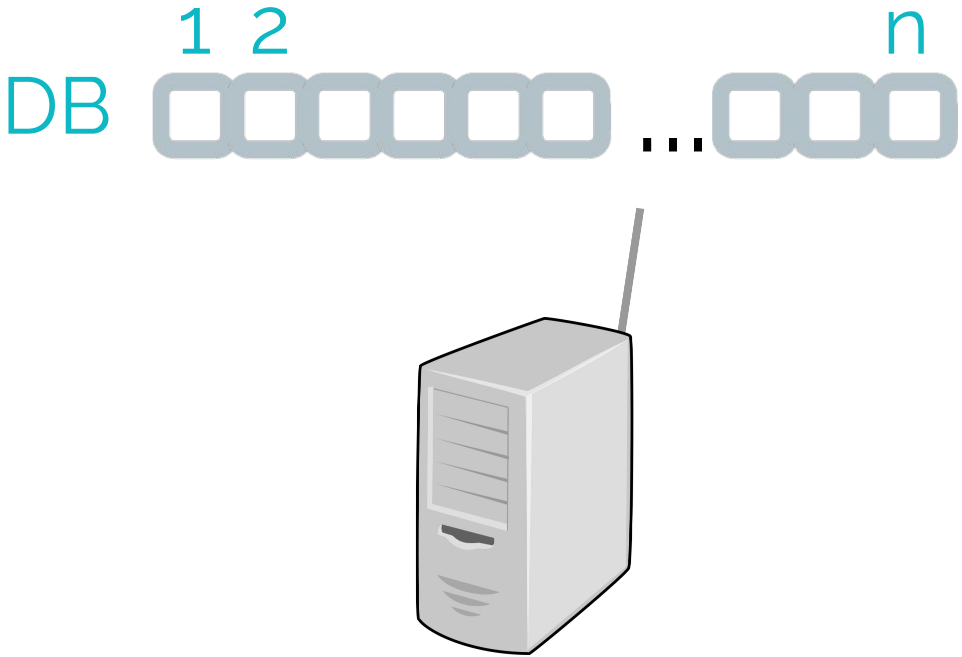
Stephen Pinkerton

1.1.1.1

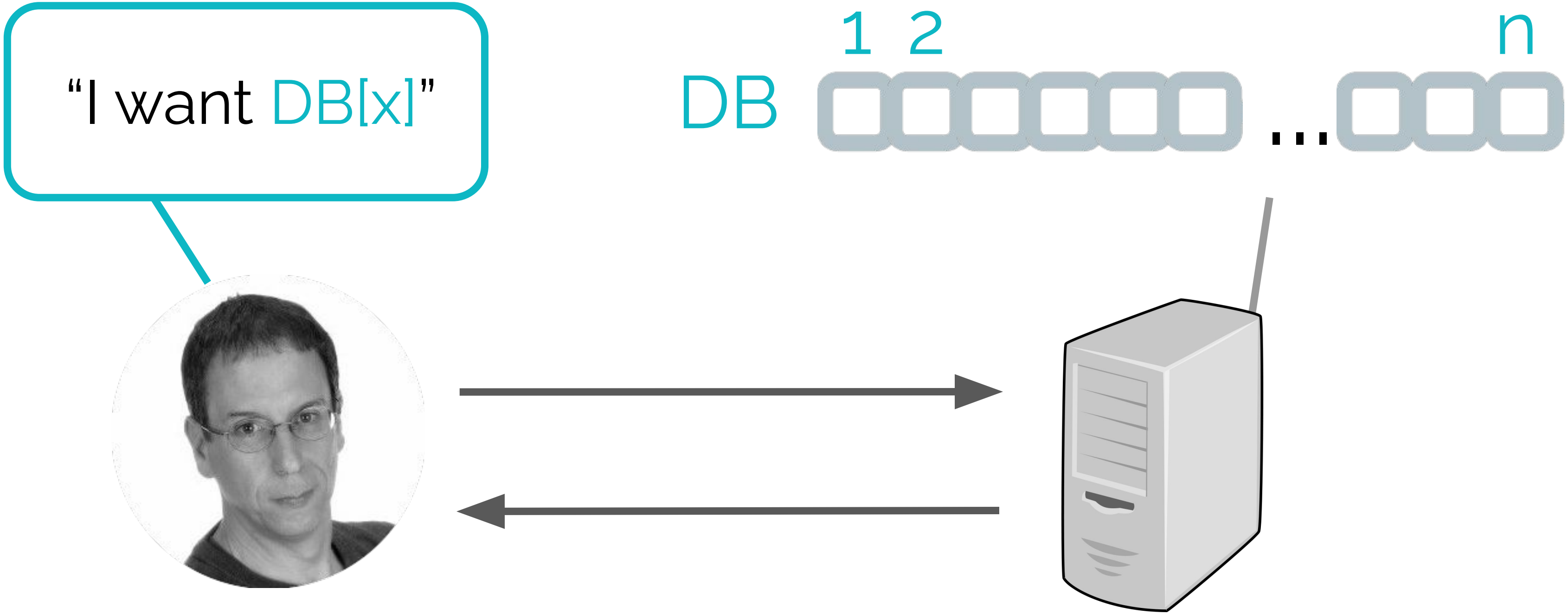

minecraft.com: 1.2.3.4
google.com: 5.6.7.8
... ...
minecraft.com
1.2.3.4
DNS

Problem Definition

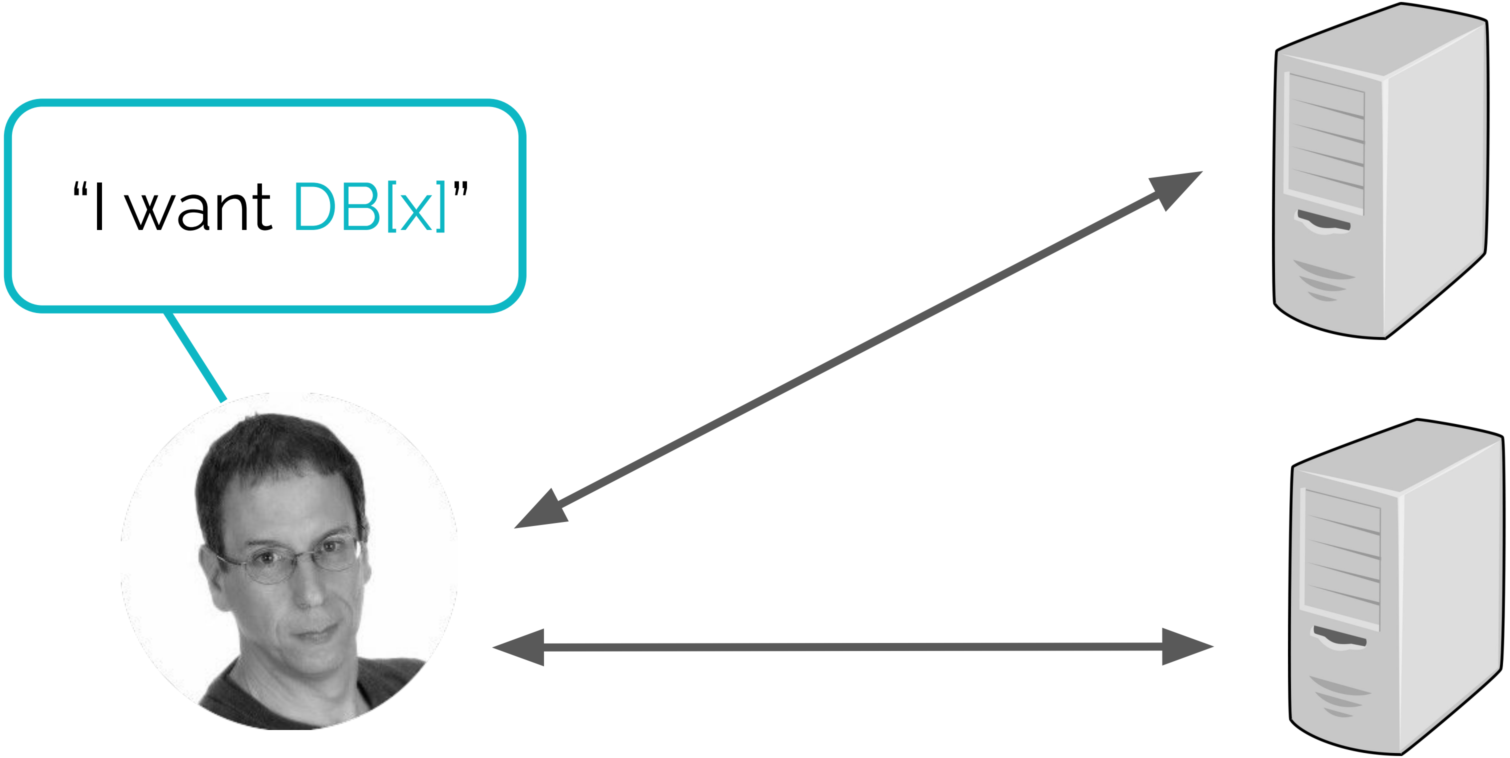
"I want DB[x]"



Problem Definition



PIR with **2 non-colluding servers**



Classical PIR
(no preprocessing)

- Linear-time
- $\widetilde{O}(1)$ -BW

Classical PIR

(no preprocessing)

- Linear-time
- $\widetilde{O}(1)$ -BW

Preprocessing PIR

(one-time preprocessing, unbounded queries)

Classical PIR

(no preprocessing)

Linear-time

$\widetilde{O}(1)$ -BW

Preprocessing PIR

(one-time preprocessing, unbounded queries)

$O(\sqrt{n})$ -time

$O(\sqrt{n})$ -BW

[CK, Eurocrypt'19 best student paper]

Assume: $O(\sqrt{n})$ client storage, OWF

The best of both worlds?

Linear-time

$\tilde{O}(1)$ -BW

$O(\sqrt{n})$ -time

$O(\sqrt{n})$ -BW

[CK, Eurocrypt'19 best student paper]

Assume: $O(\sqrt{n})$ client storage, OWF

Our result: 2-server preprocessing PIR

$O(\sqrt{n})$ -time

$\tilde{O}(1)$ -BW

Assume: **hardness of LWE**

$O(\sqrt{n})$ client storage

Open question:

A truly practical PIR scheme ?

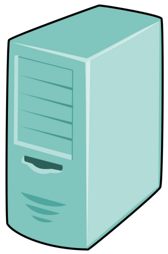
Our scheme

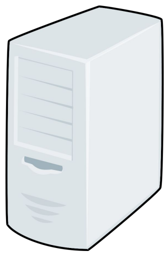
Privately Puncturable Pseudorandom Sets

Inefficient strawman

Inspired by [CK19]

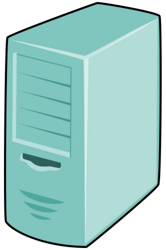
Preprocessing phase

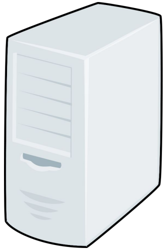




Samples a set:

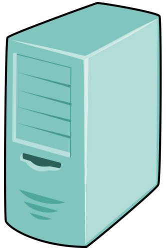
include each index w.p. $\frac{1}{\sqrt{n}}$



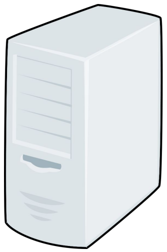


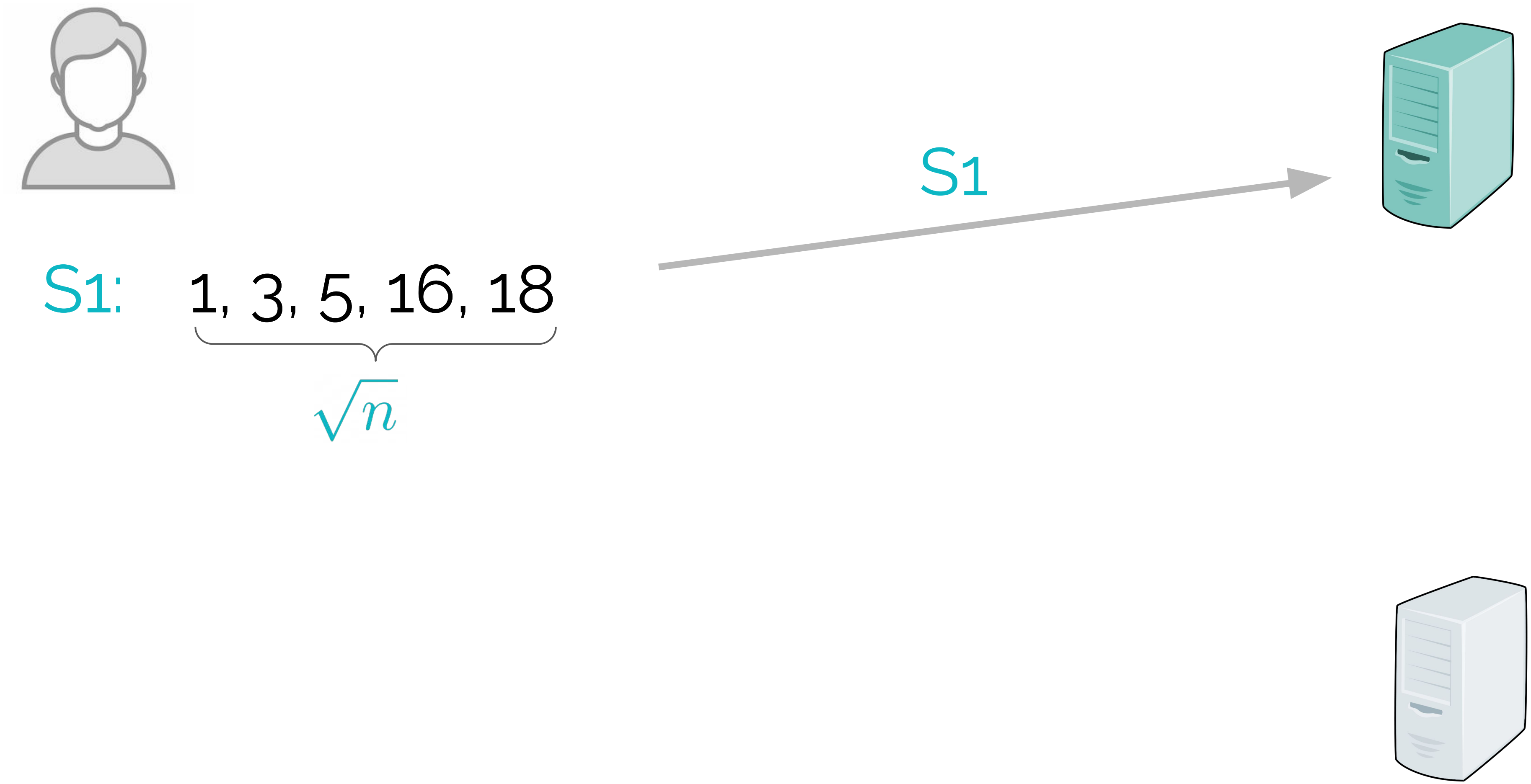
Samples a set:

include each index w.p. $\frac{1}{\sqrt{n}}$

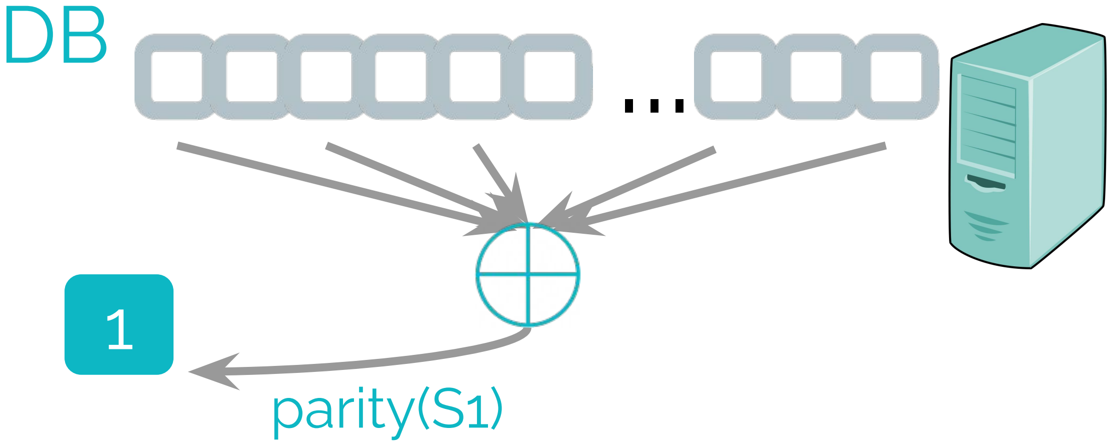


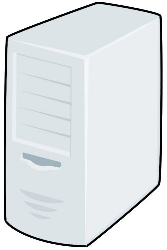
S1: 1, 3, 5, 16, 18 $\underbrace{}_{\sqrt{n}}$

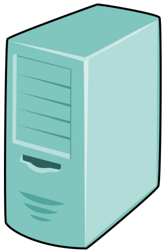



S1: 1, 3, 5, 16, 18
$\sqrt{n}$
S1

S1: 1, 3, 5, 16, 18





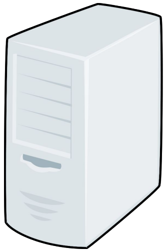


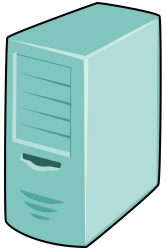
S1: 1, 3, 5, 16, 18 | 1

S2: 3, 6, 13, 19, 33 | 0

...

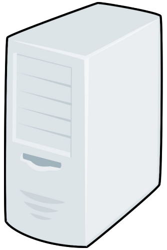
$S_{\sqrt{n}\log^2 n}$: 2, 5, 6, 7, 8, 10 | 0

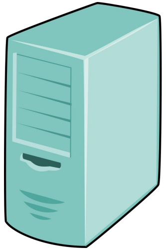




S1:	1, 3, 5, 16, 18	1
S2:	3, 6, 13, 19, 33	0
	...	
$S_{\sqrt{n}\log^2 n}$:	2, 5, 6, 7, 8, 10	0

Hint



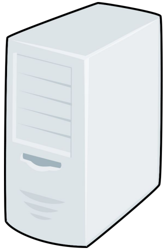


S1: 1, 3, 5, 16, 18 | 1

S2: 3, 6, 13, 19, 33 | 0

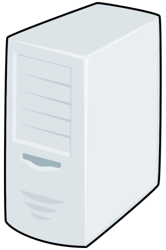
...

$S_{\sqrt{n}\log^2 n}$: 2, 5, 6, 7, 8, 10 | 0



This requires $\widetilde{O}(n)$ client space!

Online phase: want 6

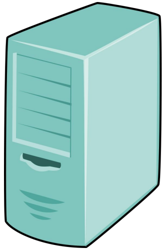


S1: 1, 3, 5, 16, 18 — 1

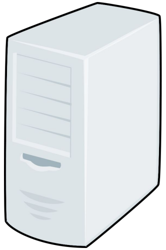
S2: 3, 6, 13, 19, 33 — 0

...

$S_{\sqrt{n}\log^2 n}$: 2, 5, 6, 7, 8, 10 — 0



Online phase: want 6

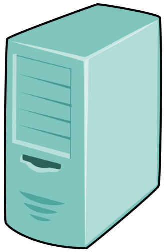


S1: 1, 3, 5, 16, 18 | 1

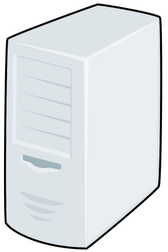
S2: 3, 6, 13, 19, 33 | 0

...

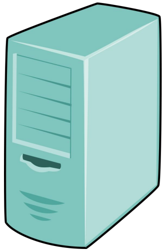
$S_{\sqrt{n}\log^2 n}$: 2, 5, 6, 7, 8, 10 | 0



Online phase: want 6



S2: 3, 6, 13, 19, 33 0



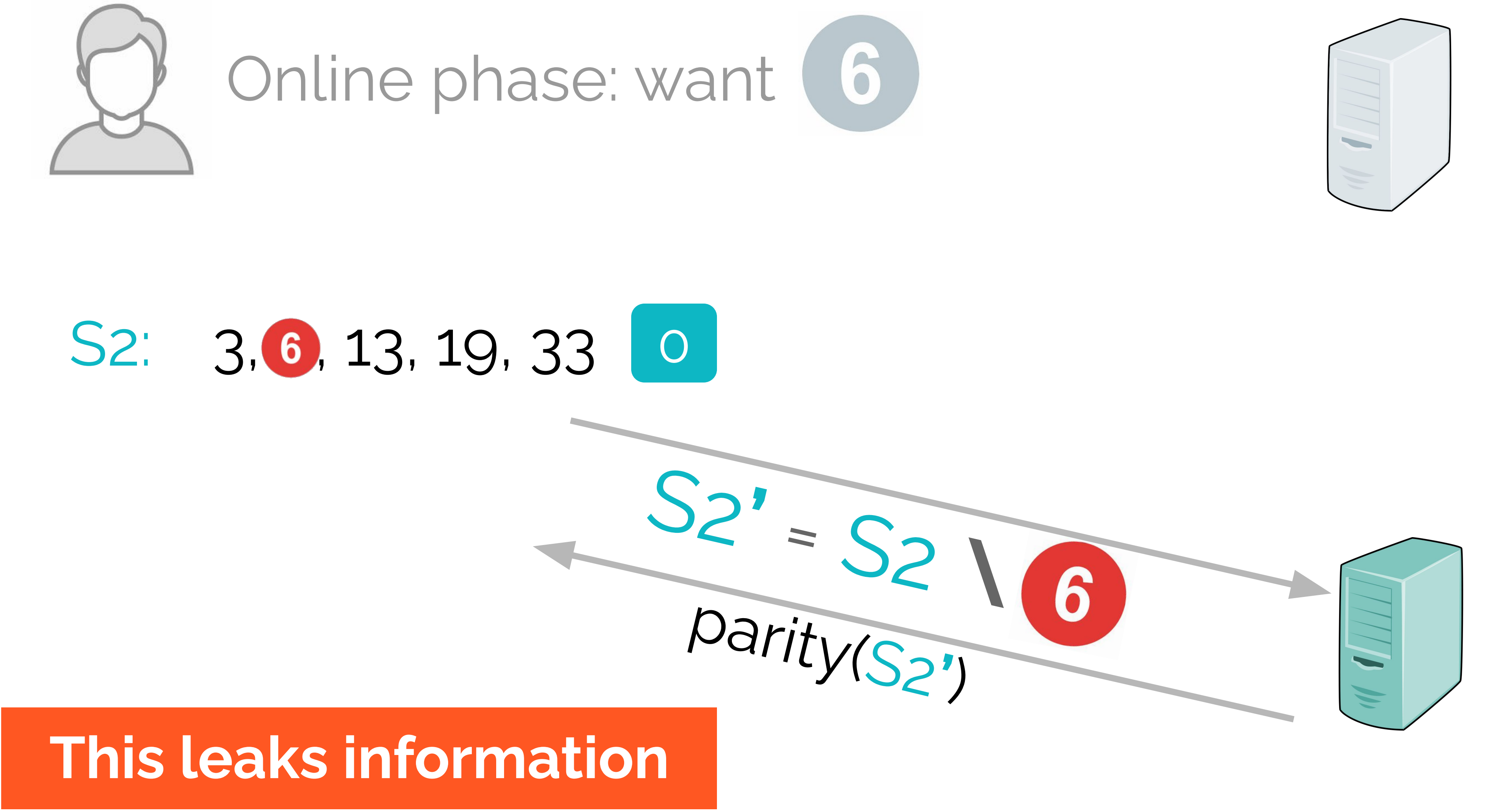
Online phase: want 6

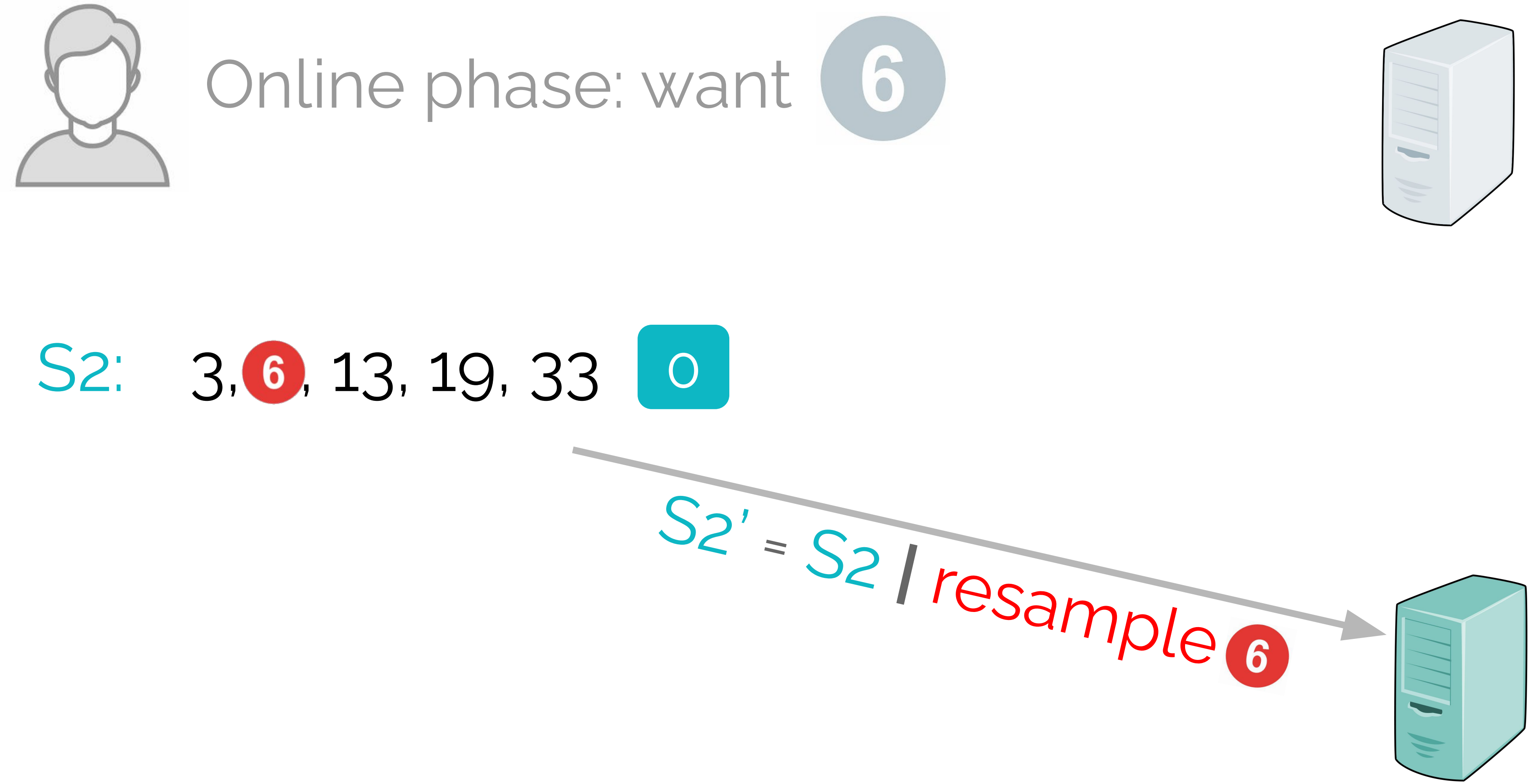
S2: 3, 6, 13, 19, 33 0

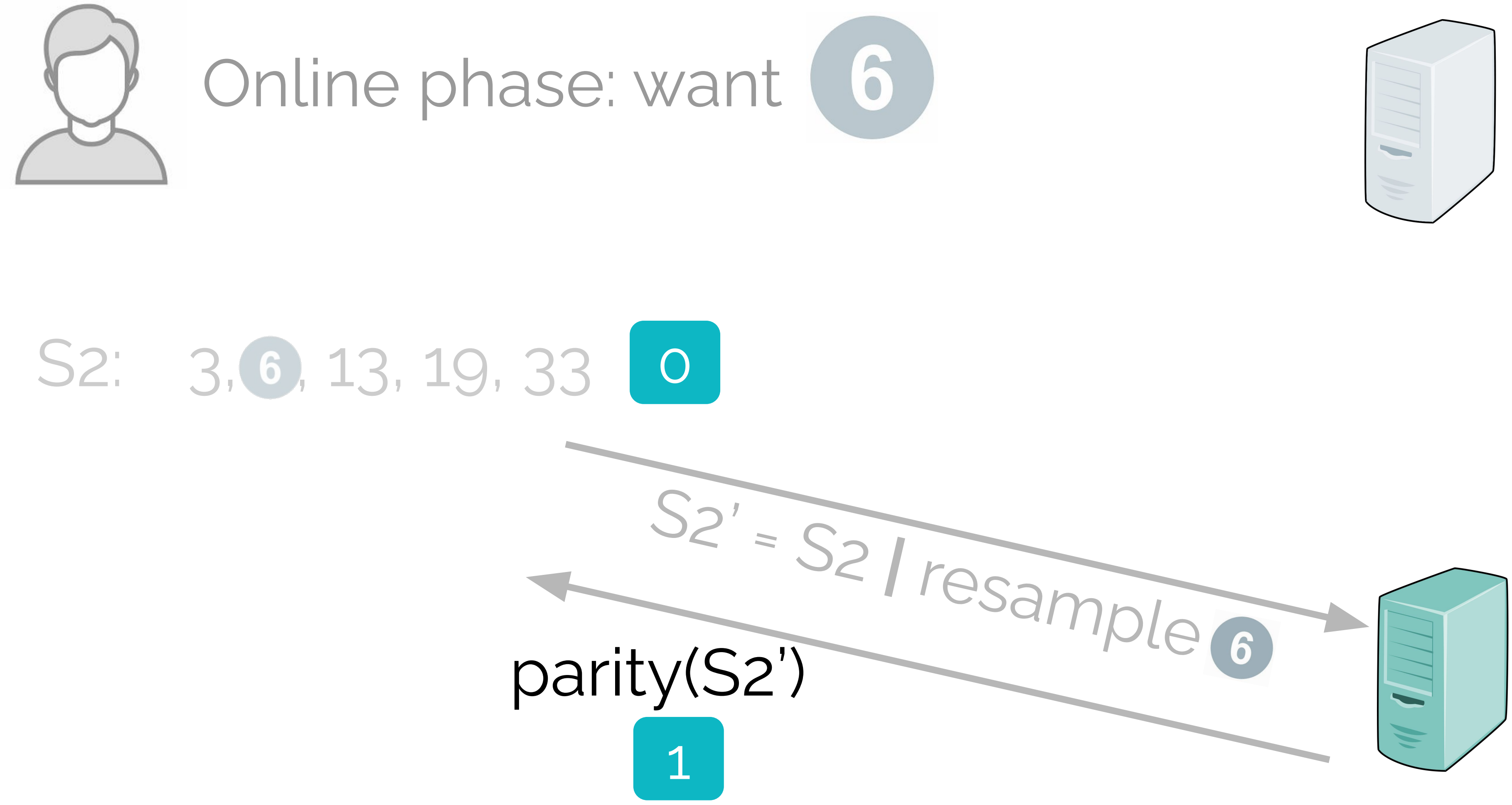
S2' = S2 \ 6

parity(S2')

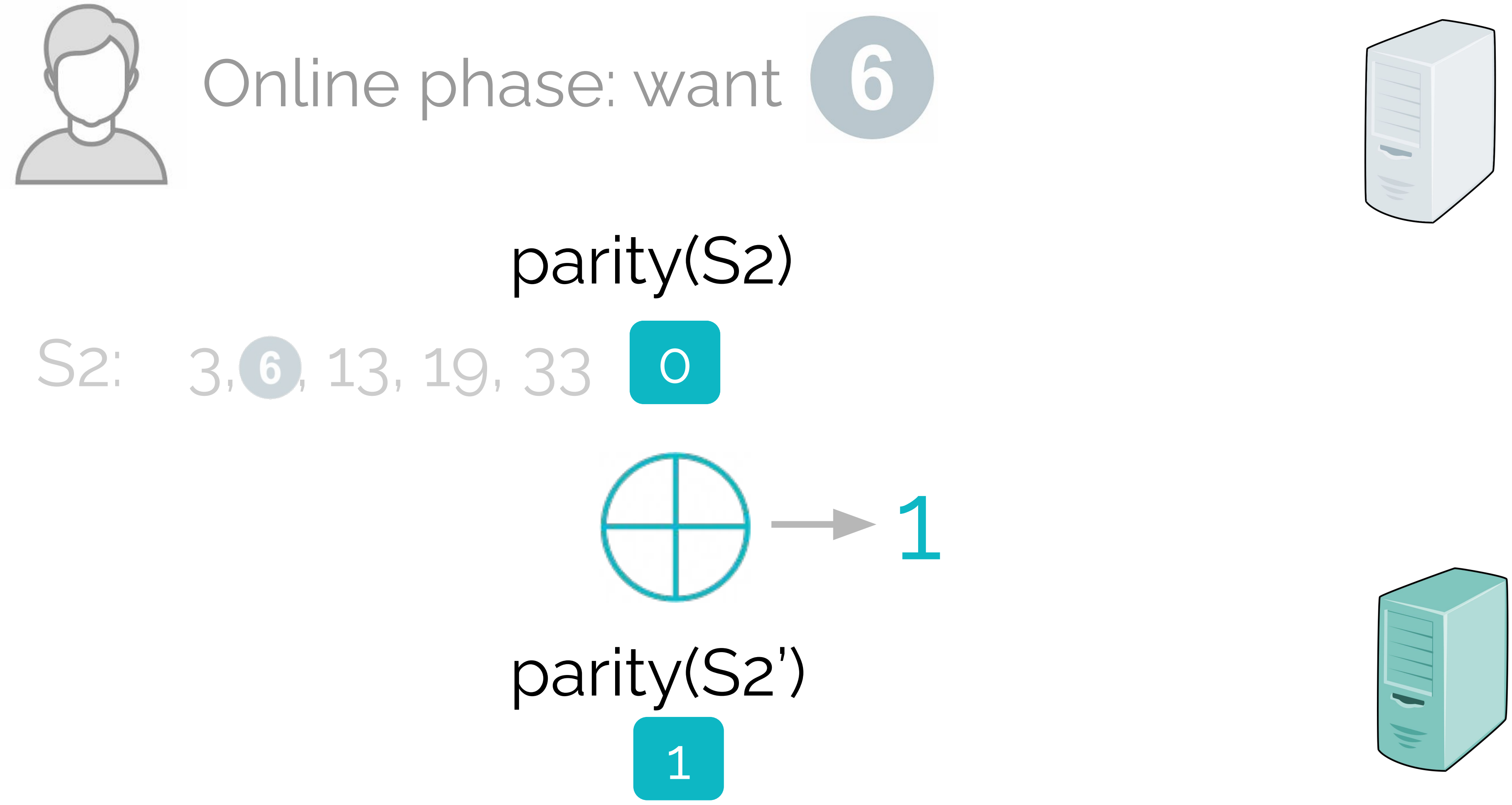
Naive approach


Online phase: want 6
S2: 3, 6, 13, 19, 33 0
S2' = S2 \ 6
parity(S2')
This leaks information


Online phase: want 6
S2: 3, 6, 13, 19, 33 0
S2' = S2 | resample 6



Online phase: want 6
S2: 3, 6, 13, 19, 33
0
S2' = S2 | resample 6
parity(S2')
1


Online phase: want 6
parity(S2)
S2: 3, 6, 13, 19, 33
0
1
parity(S2')
1

Correct if (S2' = S2 | resample 6) removes 6

This happens w.h.p.

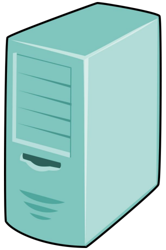
parity(S2)

S2: 3, 6, 13, 19, 33 | 0

$\oplus$ → 1

parity(S2')

1



k-fold repetition amplifies correctness

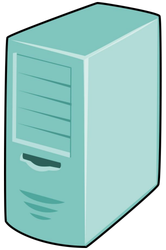
parity(S2)

S2: 3, 6, 13, 19, 33 0

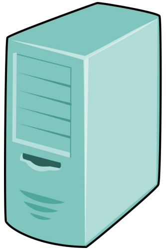
$\oplus \rightarrow$ 1

parity(S2')

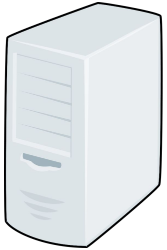
1



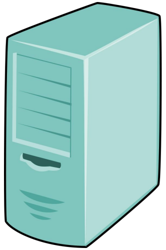
Online: refresh



~~S2: 3, C, 13, 19, 33 C~~



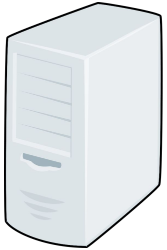
Online: refresh



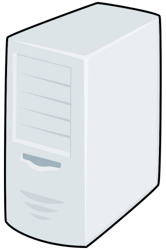
S' = S | resample 6

S: 5, 6, 11, 16, 32

~~S2: 3, 6, 13, 19, 33~~



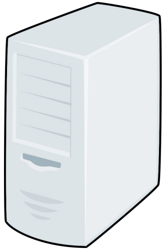
Recap



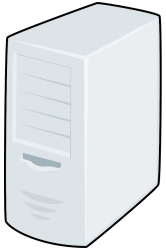
client space $\tilde{O}(n)$

online BW $\tilde{O}(\sqrt{n})$

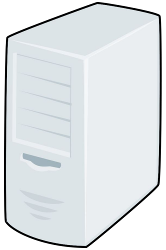
online time $\tilde{O}(\sqrt{n})$



What we want



client space	$\tilde{O}(n)$	→	$\tilde{O}(\sqrt{n})$
online BW	$\tilde{O}(\sqrt{n})$	→	$\tilde{O}(1)$
online time	$\tilde{O}(\sqrt{n})$	→	$\tilde{O}(\sqrt{n})$

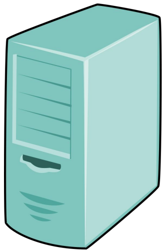


Our scheme

Privately Puncturable Pseudorandom Sets

Inefficient strawman

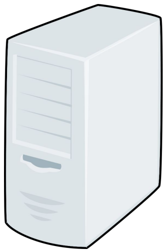
Preprocessing

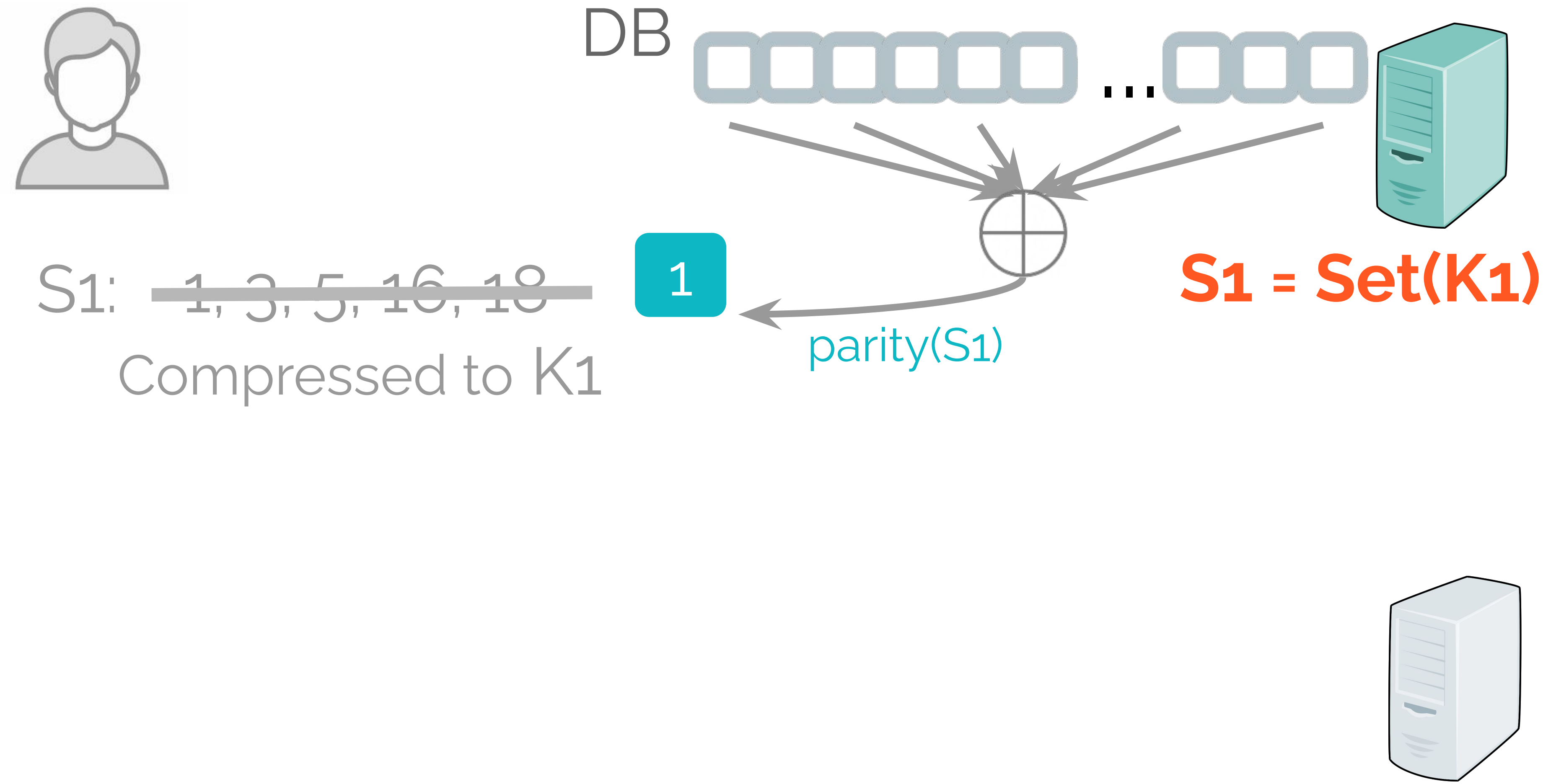


K1

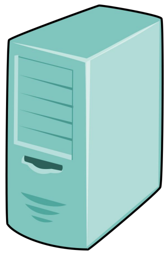
S1: ~~1, 3, 5, 16, 18~~

Compressed to K1





DB
...
S1: ~~1, 3, 5, 16, 18~~
Compressed to K1
1
parity(S1)
S1 = Set(K1)

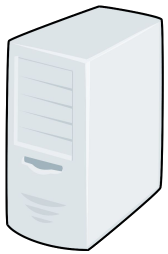


K1: 1

K2: 0

...

$K_{\sqrt{n}\log^2 n}$: 0



Online phase: want 6

K1: 1

K2: 0

...

$K_{\sqrt{n}\log^2 n}$: 0

Online phase: want 6

K1: 1

K2: 0

...

$K_{\sqrt{n}\log^2 n}$: 0

Find i s.t. 6 $\in \mathrm{Set}(\mathrm{K}_i)$

Online phase: want 6

K1: 1

K2: 0

...

$K_{\sqrt{n}\log^2 n}$: 0

- Find i s.t. 6 $\in \mathrm{Set}(K_i)$
- K_i' = Puncture (K_i , 6)

Online phase: want 6

K1: 1

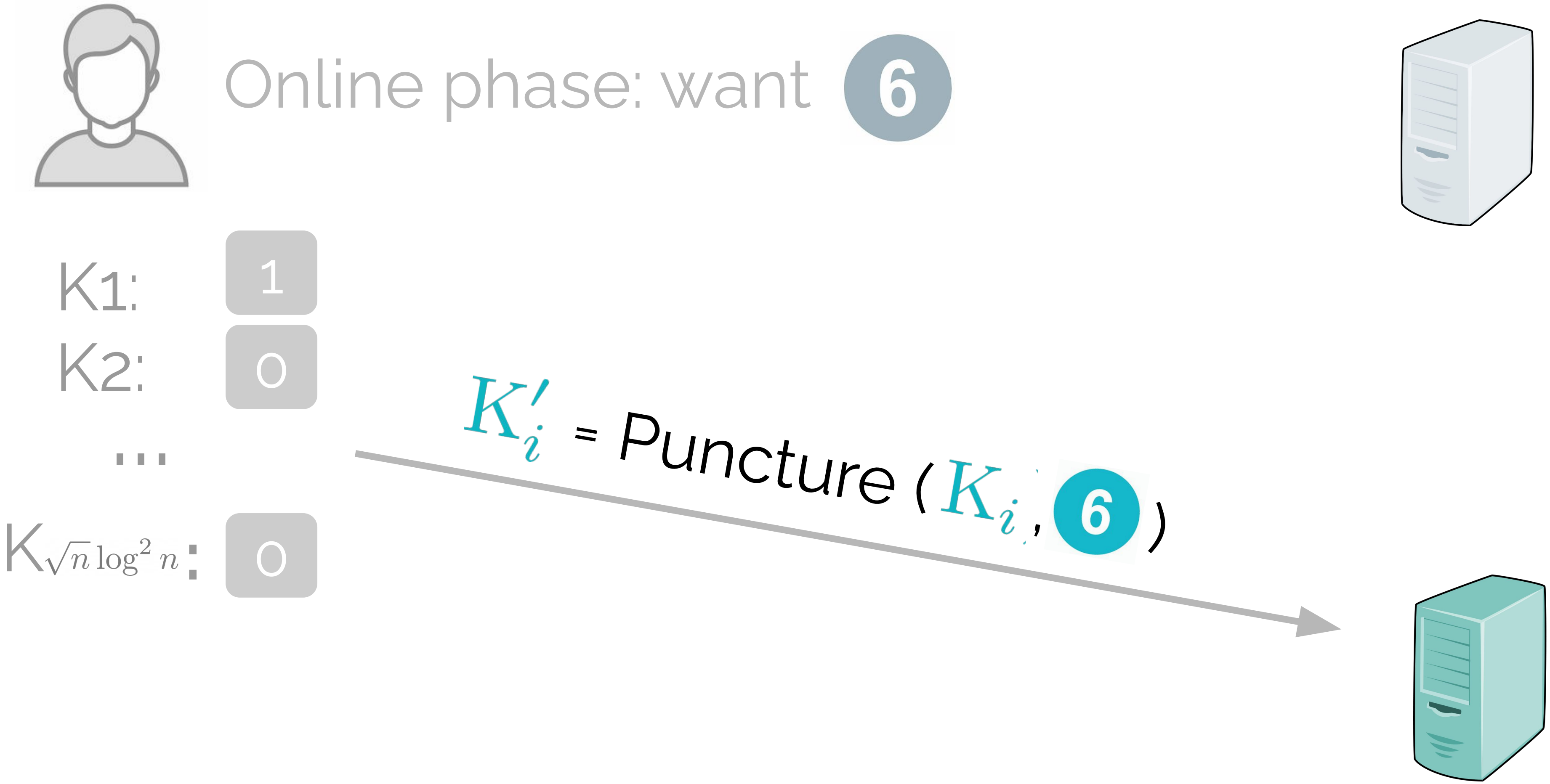
K2: 0

...

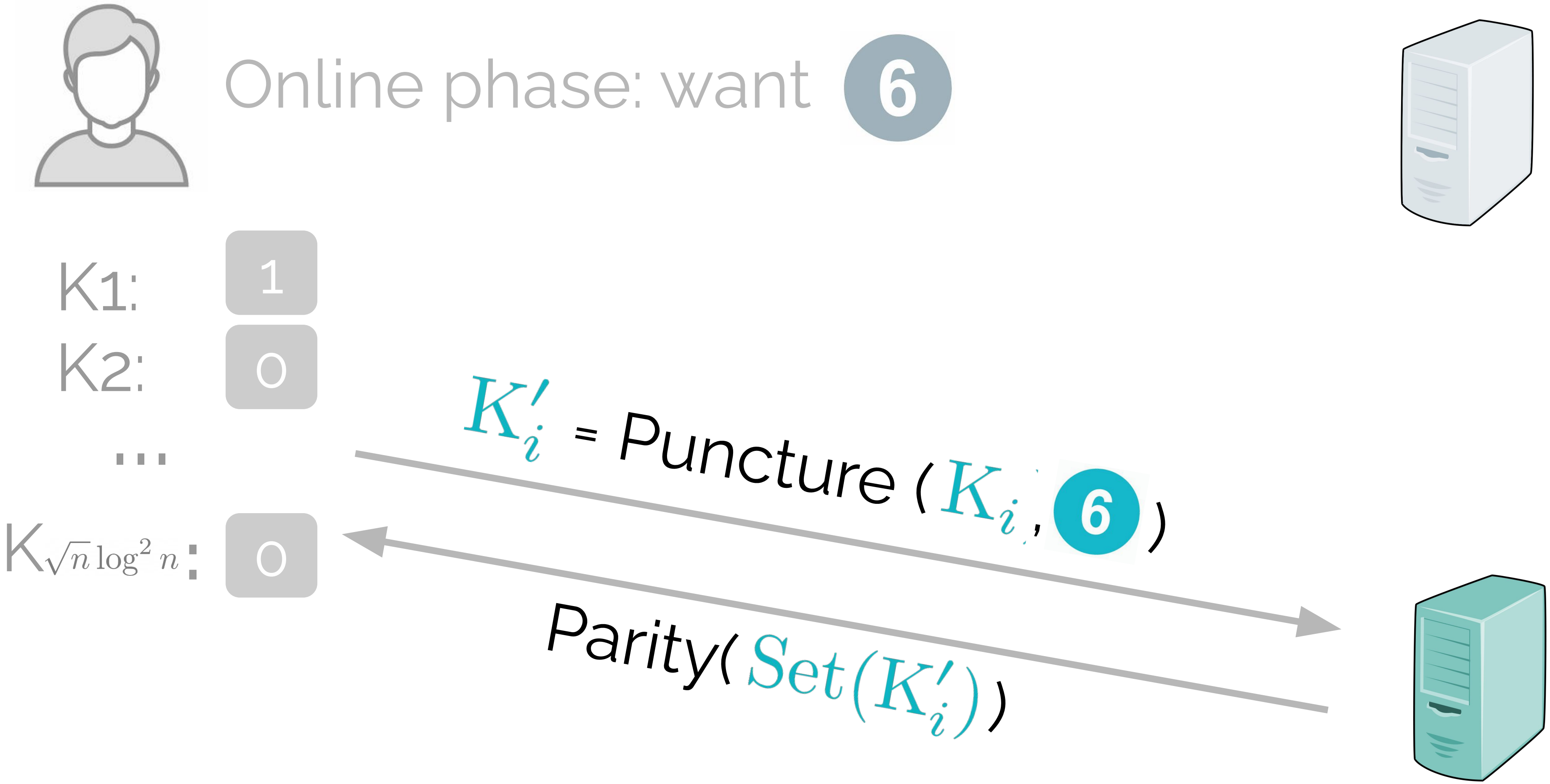
$K_{\sqrt{n}\log^2 n}$: 0

- Find i s.t. $6 \in \mathrm{Set}(\mathrm{K}_i)$
- K'_i = Puncture (K_i , 6)

$$\mathrm{Set}(\mathrm{K}'_i) = \mathrm{Set}(\mathrm{K}_i) | \mathrm{Resample}(6)$$



Online phase: want 6
K1: 1
K2: 0
...
$K_{\sqrt{n}\log^2 n}$: 0
K_i' = Puncture (K_i , 6)



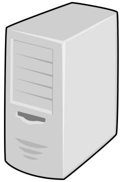
Online phase: want 6
K1: 1
K2: 0
...
$K_{\sqrt{n}\log^2 n}$: 0
K_i' = Puncture (K_i, 6)
Parity($\mathrm{Set}(K_i')$)

Puncturable Pseudorandom Set

- Sample a key K
- **Set(K)** enumerates the set
- **Puncture(K, x)** gives a key that resamples whether x is in the set

Desiderata: Puncturable Pseudorandom Set

Punctured key hide punctured point



sees punctured key

Desiderata: Puncturable Pseudorandom Set

Punctured key hide punctured point

Fast membership test : $\widetilde{O}(1)$

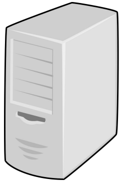
Find i s.t. 6 $\in \mathrm{Set}(\mathrm{K}_i)$

Desiderata: Puncturable Pseudorandom Set

Punctured key hide punctured point

Fast membership test : $\tilde{O}(1)$

Fast set enumeration : $\tilde{O}(\sqrt{n})$



enumerates set with punctured key

Desiderata: Puncturable Pseudorandom Set

Punctured key hide punctured point

Fast membership test : $\widetilde{O}(1)$

Fast set enumeration : $\widetilde{O}(\sqrt{n})$

Strawman using **Privately Puncturable PRF**

Ordinary PRF

$$\begin{aligned} K &\leftarrow \mathrm{Gen}(1^\lambda) \\ y &\leftarrow \mathrm{Eval}(K, x) \end{aligned}$$

Privately Puncturable PRF

$$K \leftarrow \mathrm{Gen}(1^\lambda) \qquad K_x \leftarrow \mathrm{Puncture}(K, x)$$

$$y \leftarrow \mathrm{Eval}(K, x) \qquad y \leftarrow \mathrm{PEval}(K_x, x')$$

[BKM17,CC17,BTVW17]

Privately Puncturable PRF

$$K \leftarrow \text{Gen}(1^\lambda) \qquad K_x \leftarrow \text{Puncture}(K, x)$$
$$y \leftarrow \text{Eval}(K, x) \qquad y \leftarrow \text{PEval}(K_x, x')$$

- Punctured key K_x hides punctured point x
- $\text{PEval}(K_x, x) \Longrightarrow$ pseudo-random

[BKM17,CC17,BTVW17]

Privately Puncturable PRF: known from LWE

$$K \leftarrow \mathrm{Gen}(1^\lambda) \qquad K_x \leftarrow \mathrm{Puncture}(K, x)$$
$$y \leftarrow \mathrm{Eval}(K, x) \qquad y \leftarrow \mathrm{PEval}(K_x, x')$$

- Punctured key K_x hides punctured point x
- $\mathrm{PEval}(K_x, x) \Longrightarrow$ pseudo-random

[BKM17,CC17,BTVW17]

Strawman Puncturable Pseudorandom Set

6 is included iff

$\mathrm{PRF.Eval}(K, 6)$ has $\frac{1}{2}\log n$ trailing 0s

Strawman Puncturable Pseudorandom Set

6 is included iff

$\mathrm{PRF.Eval}(K, 6)$ has $\frac{1}{2}\log n$ trailing 0s

$\mathrm{PRF.Puncture}(K, 6)$ punctures 6

Would this work?

6 is included iff

$\mathrm{PRF.Eval}(K, 6)$ has $\frac{1}{2}\log n$ trailing 0s

$\mathrm{PRF.Puncture}(K, 6)$ punctures 6

Would this work?

6 is included iff

$\mathrm{PRF.Eval}(K, 6)$ has $\frac{1}{2}\log n$ trailing 0s

$\mathrm{PRF.Puncture}(K, 6)$ punctures 6

Set enumeration takes O(n) time!

Other strawman attempts

Set

$$\mathrm{PRF.Eval}(K, 1) \quad \mathrm{PRF.Eval}(K, 2) \quad \ldots \quad \mathrm{PRF.Eval}(K, \sqrt{n}) \Rightarrow \{x_1, x_2, \ldots, x_{\sqrt{n}}\}$$

Slow membership test!

Our scheme

Privately Puncturable Pseudorandom Sets

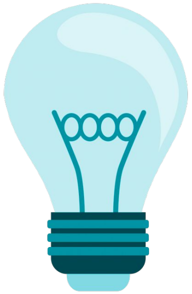
Inefficient strawman

Key Insight

Sample the set with a **carefully crafted distribution**

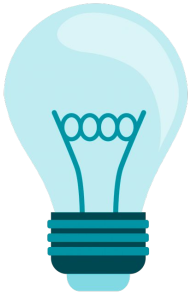
- Fast membership test
- Fast set enumeration
- ⚠ "**Breaks**" puncturing "**just a little**"

$x = 38$



X =

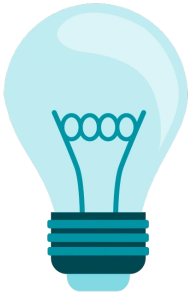
001010



$X =$

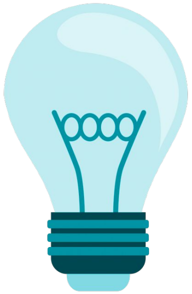
00001010

2 loglog n



X =

$$00\underbrace{001010}_{\frac{1}{2}\log n} \quad H(010) \stackrel{?}{=} 1$$



X =

00001010

$H(010) \overset{?}{=} 1$

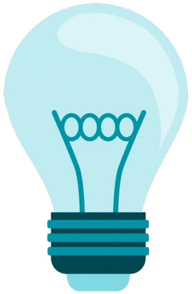
$H(1010) \overset{?}{=} 1$

$H(01010) \overset{?}{=} 1$

$H(001010) \overset{?}{=} 1$

$H(0001010) \overset{?}{=} 1$

$H(00001010) \overset{?}{=} 1$



X =

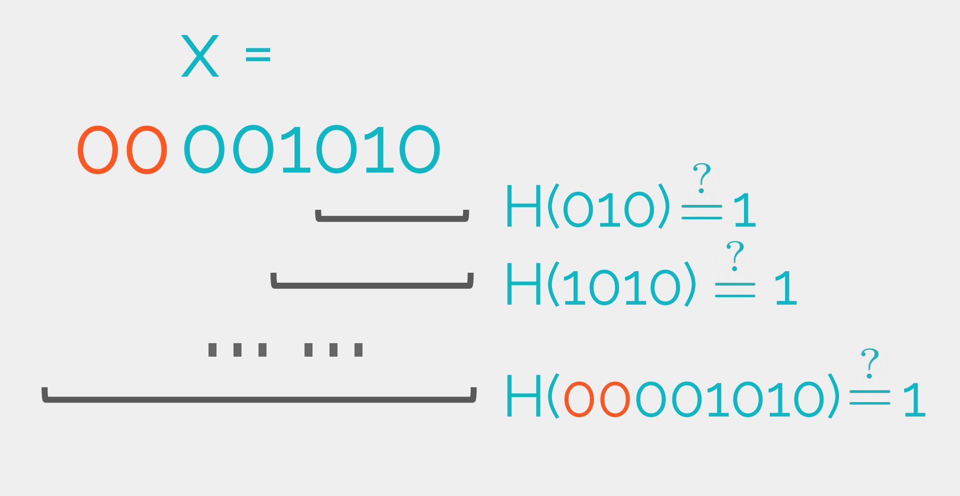
00 001010

$H(010) \stackrel{?}{=} 1$

$H(1010) \stackrel{?}{=} 1$

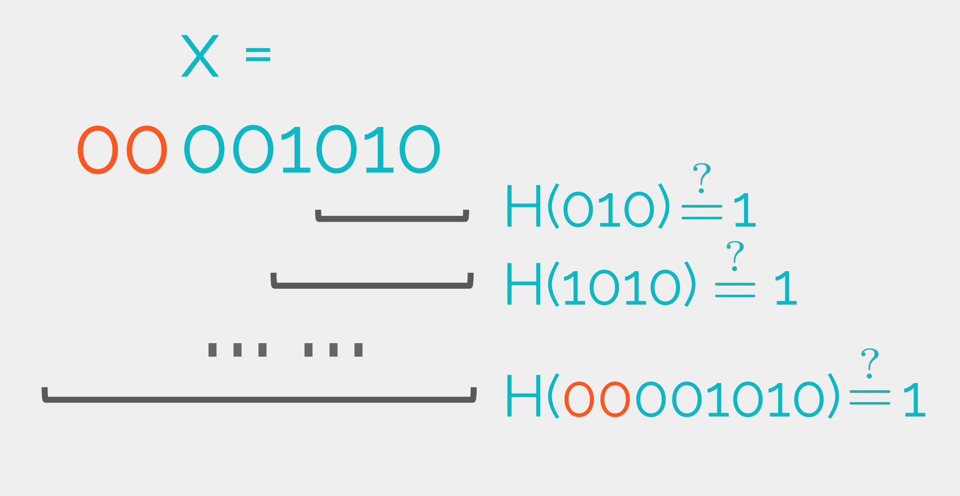
...

$H(00001010) \stackrel{?}{=} 1$



To puncture a point x = **00001010:**

Puncture all relevant suffixes from the PRF key



- Set size $\widetilde{O}(\sqrt{n})$
- Membership test $\widetilde{O}(1)$
- Set enumeration $\widetilde{O}(\sqrt{n})$

Set Enumeration

010

H(010) = 1

110

H(110) = 1

Set Enumeration

$$H(0010) \stackrel{?}{=} 1$$

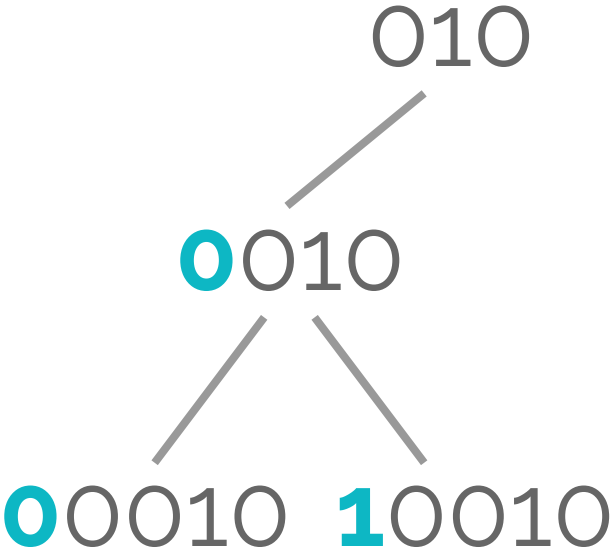
$$H(1010) \stackrel{?}{=} 1$$

$$H(0110) \stackrel{?}{=} 1$$

$$H(1110) \stackrel{?}{=} 1$$

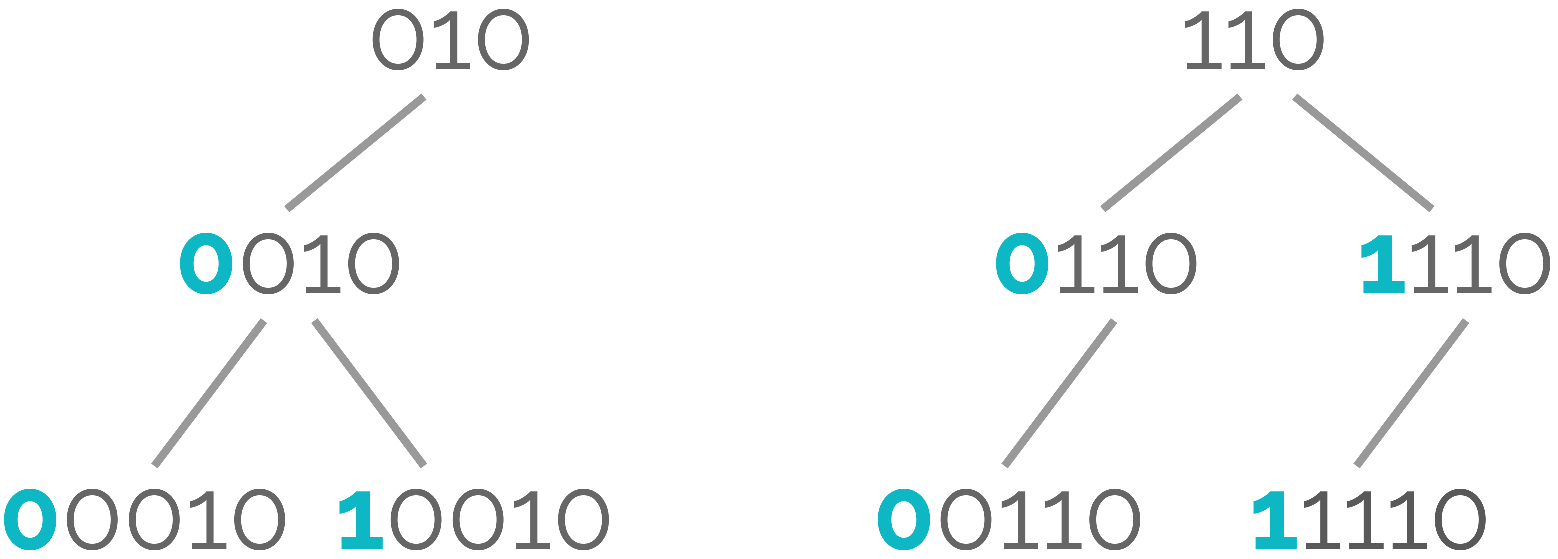
Set Enumeration

Set Enumeration



- Each level has $\widetilde{O}(\sqrt{n})$ size in this tree

Set enumeration time: $\widetilde{O}(\sqrt{n})$



x = 00001010

y = 00111010

x included ⇒ **y** more likely included x = 00001010

y = 00111010

x included ⇒ **y** more likely included

! Puncturing **x** removes **y** with small prob!

Summary: Our PIR scheme

- Key idea: a new puncturable PR Set
- Conceptually very simple construction
- Proofs are non-trivial
- Towards practicality: need a concretely efficient Privately puncturable PRF

See our paper for:

Detailed proofs

Correctness proof is actually tricky!

Trade off client space and online time

https://eprint.iacr.org/2020/1592

Open question:

A truly practical PIR scheme ?

Thank you !

runting@cs.cmu.edu