

Anonymous Communication

Tor is a network for making people anonymous

- Thousands of volunteer servers (non-profits, univs, individuals)
- Millions of users (hard to count!)
- Terabytes of traffic

Aiming for anonymity

NOT - don't write your name on it
- IP addresses are anonymous
street addresses " " "

Technical sense: Adversary can't tell which participant in the anonymity set did some action.
any better than guessing

Related: Unlinkability

Two actions: can't tell better than guessing whether they were done by the same party or not.

NOT unobservability

Unobservability means you can't tell who is participating in the system & who isn't
— lot harder to achieve, usually

Tor Threat Model

A little "fuzzy" — words of the designers!

MAXIMIZE SECURITY UNDER:

- Req: - usable for web browsing & anything you want to do over TCP.
- Usable with the Internet as it existed.

Why?

Usability matters for privacy/anonymity

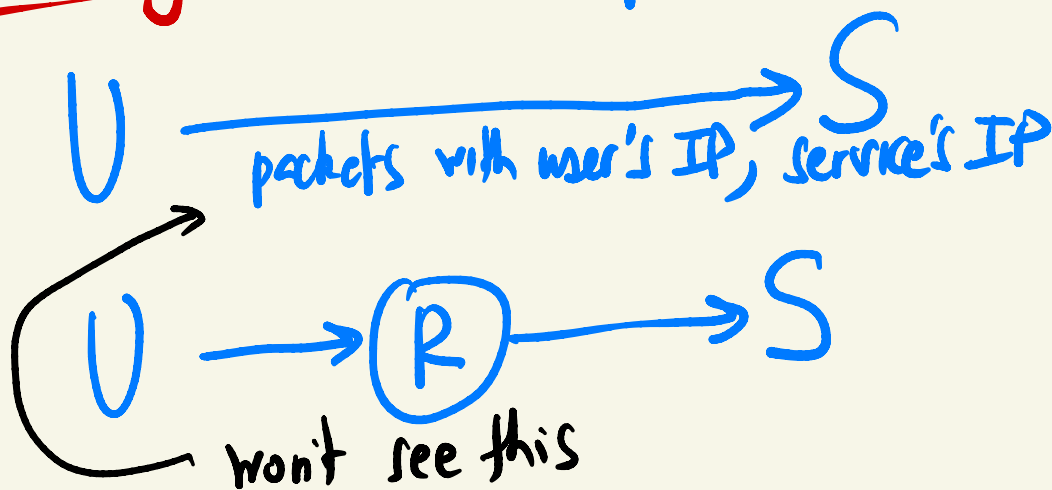
need lots of users, i.e., large
anonymity sets

Perfect anonymity $\Rightarrow$ 3 days for
messages to arrive $\Rightarrow$ 10 people use
the system $\Rightarrow$ low anonymity

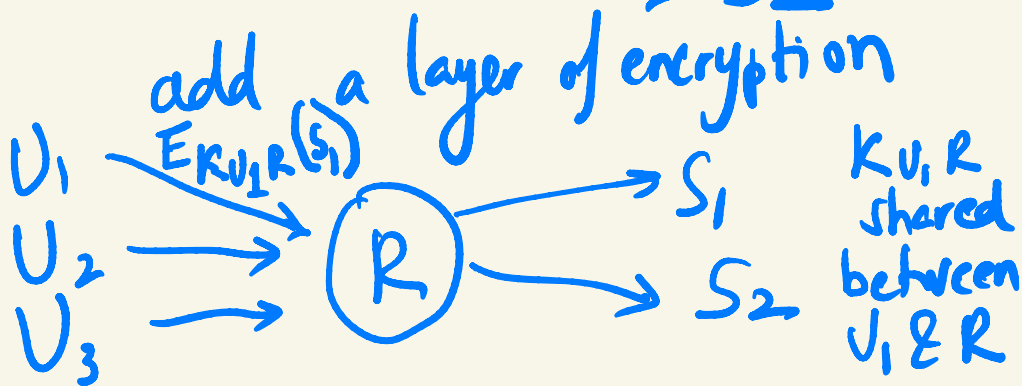
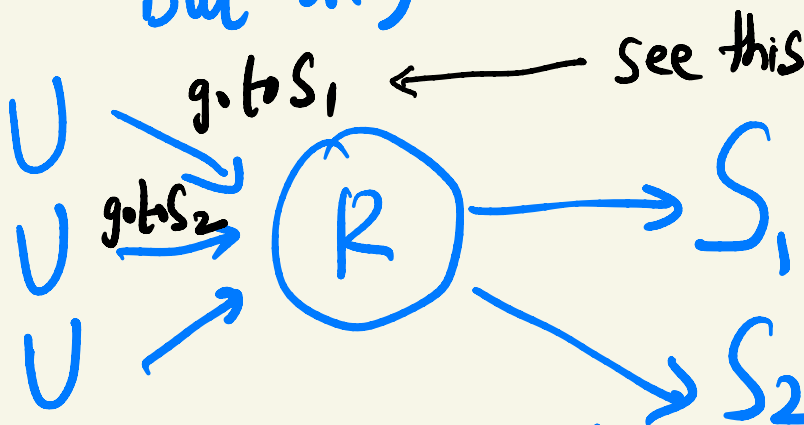
Tor defends against less powerful
adversaries (e.g., limited view
of network) but enables large
anonymity sets.

Design

Eavesdropper

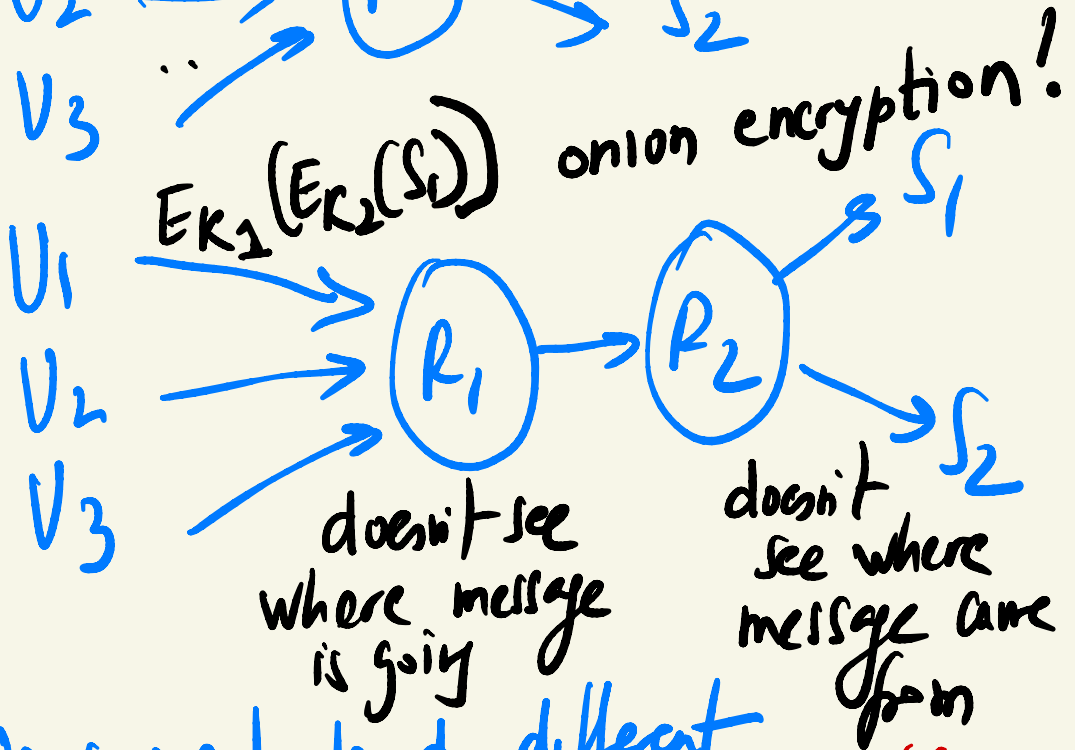
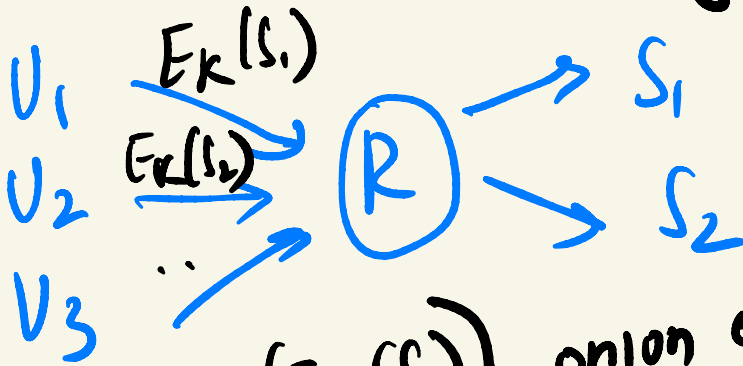


But only one user



Design (contd.)

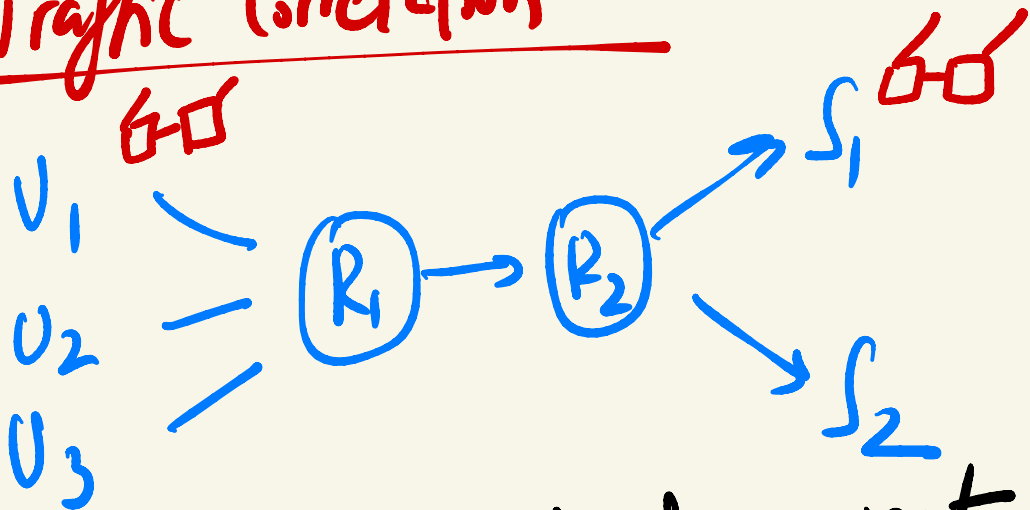
Adversary becomes relay!



Users want to do different things at different times $\Rightarrow$ audio, video, web

Traffic Analysis

Traffic Correlation



observing both sender & recipient
shows correlations in traffic
volume, traffic timing, and
causality.

com
puter
failures?

- send same volume
- at the same time
- introduce random delays?
- OR introduce dummy messages?
- OR

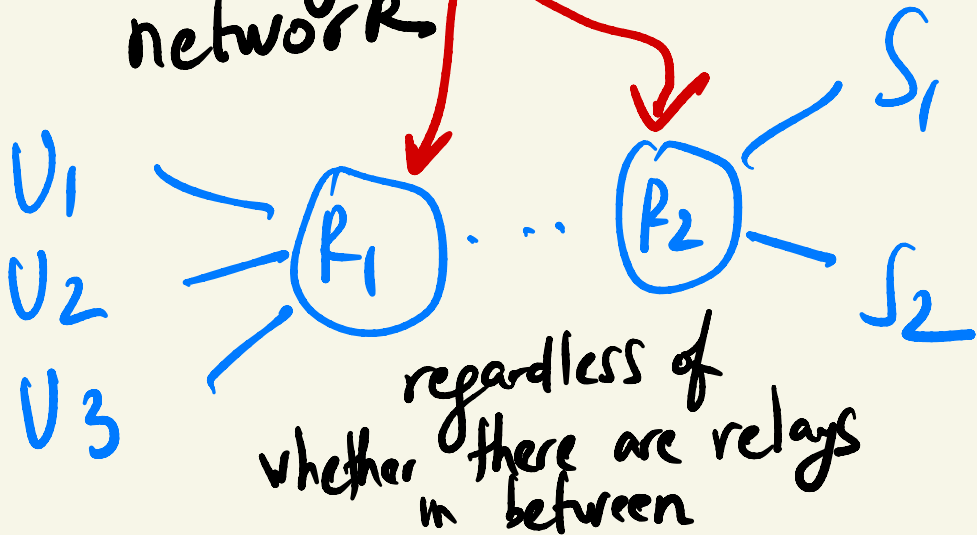
lot!
Large!

Attacker in Tor Assumption

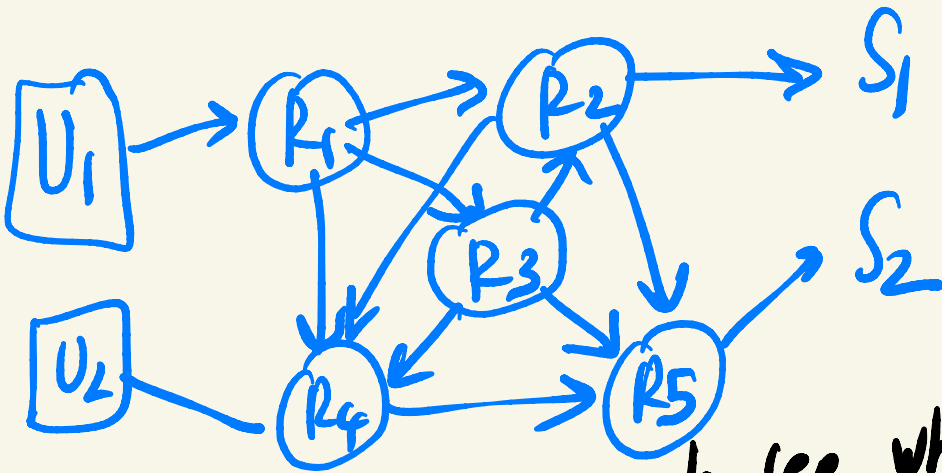
Not going to defend against
an attacker who sees both
ends!

— make it harder for anyone
to be that attacker

— Attacker who compromises
2 relays sees the whole
network



Network of Relays



- no easy way to see whole net.
- more relays you compromise the more you see.

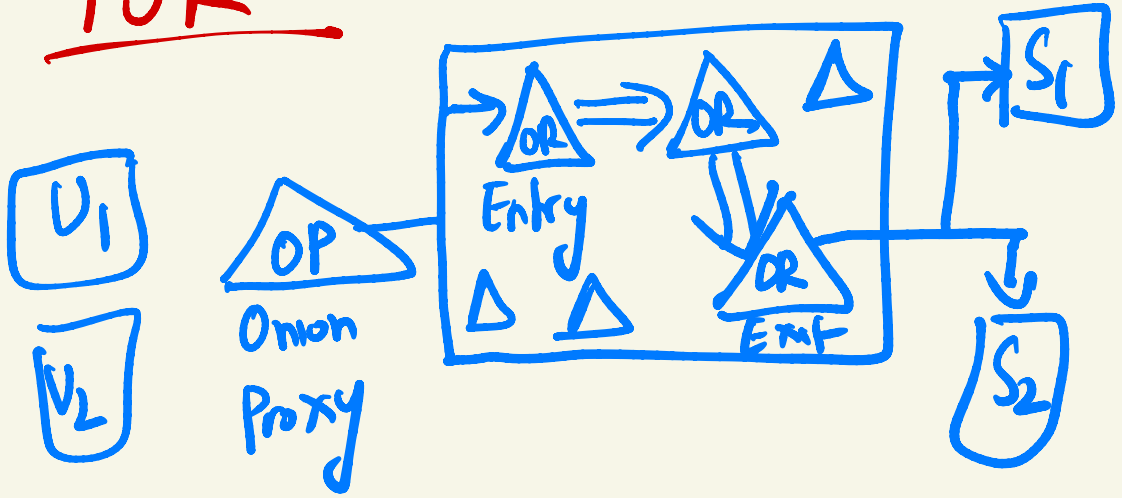
Efficiency

Minimize public key cryptography

Users negotiate a stream thru network

- persistent object for period of time; users share ephemeral keys with relays on paths.

TOR

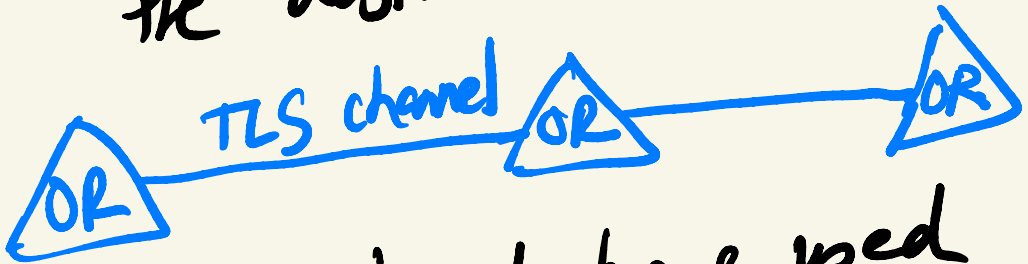


OR : Onion routers

1. Client U_i connects to proxy OP .
2. OP establishes a path to the destination address called a **circuit**, consisting of 3 OR s.
3. To create a circuit, OP contacts the **Authority Directory** and chooses an OR to be its entry point: entry guard.

TOR (contd.)

4. The entry guard extends the circuit to the next hop, reaching the exit guard.
5. Exit guard connects to the destination.

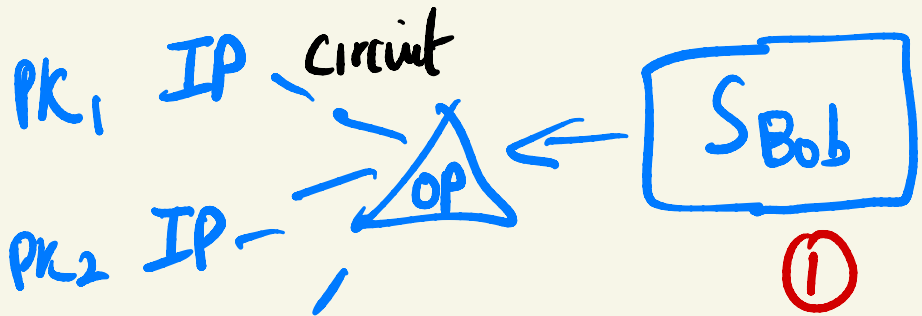


OP has shared keys used for onion encryption with every hop in the circuit (using key exchange similar to TLS).

$$E_{K_1}(E_{K_2}(E_{K_3}(V_i \rightarrow S_i))))$$

Hidden Services

Receiver anonymity



Bob randomly selects Introduction Points (IPs) to his service

Service Descriptor : $Sign(SK_{Bob}, PK_i)$

②

distributed to clients

③ Client selects random relays as rendezvous points (RPs), builds circuits to them.

④

Client chooses an IP .
Sends introduce message to IP ,
containing address of an RP
encrypted with PK_{Bob} .

⑤

Service receives message,
decrypts using SK_{Bob} , reads
 IP address,

Creates circuit to RP ,

Sends message to RP .

(message contains a one-time secret.)

⑥

RP informs client that
connection has been established
to Service. $\longleftrightarrow$

Tor Basic Assumption

Adversary trying to figure out who is communicating with whom. Tor assumes adversary can only observe a fraction of the communication.

- controlling ORs

Adversary compromising first and last nodes of a route with prob $\frac{c^2}{n^2}$ where c is # compromised nodes, n # total nodes.

Tor assumes c^2/n^2 is low!

Tor Defenses

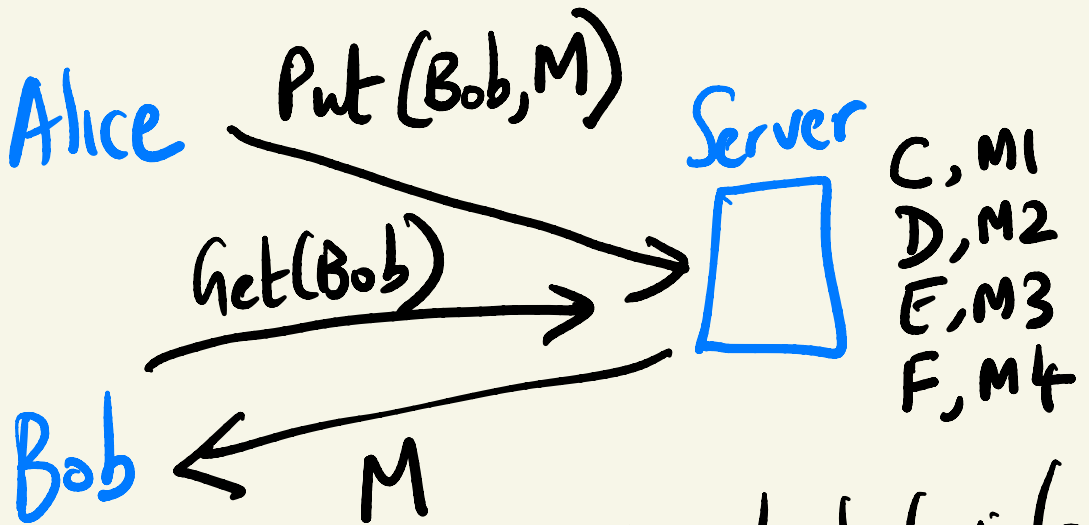
- + Onion encryption, forward secrecy, rotation of keys, circuit lifetime limit
- No real defense against traffic correlation using end-to-end timer & packet size. $\frac{c^2}{n^2}$ failure prob.

Website Fingerprinting

- 1) Adversary's clients visit website and create fingerprint(s), network traces of clients: **Size, timing**
- 2) Train supervised classifier with traces
- 3) Use model to classify network traces of live users.

Pung

Cryptographic approach



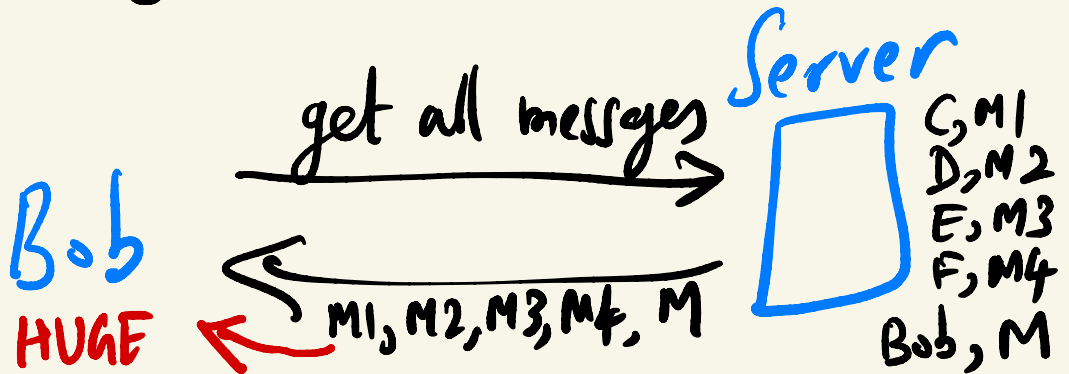
Put & Get should not leak info

(1) Data and sizes

Use encryption to hide the content & padding to hide sizes

Pung (contd.)

- (2) Destination
Use random ids to hide identities
- (3) Timing information
Periodically put and get messages, add cover traffic
- (4) Alice and Bob use the same id and message



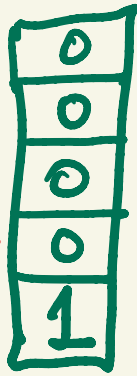
Private Information Retrieval

Assume Alice & Bob know Bob is in position 5 in the Server DB

Alice & Bob share a key beforehand and agree on 5

encrypted by A.H. scheme

Server



x
x
x
x
x

C, M1
D, M2
E, M3
F, M4
B, M

Bob
all look different

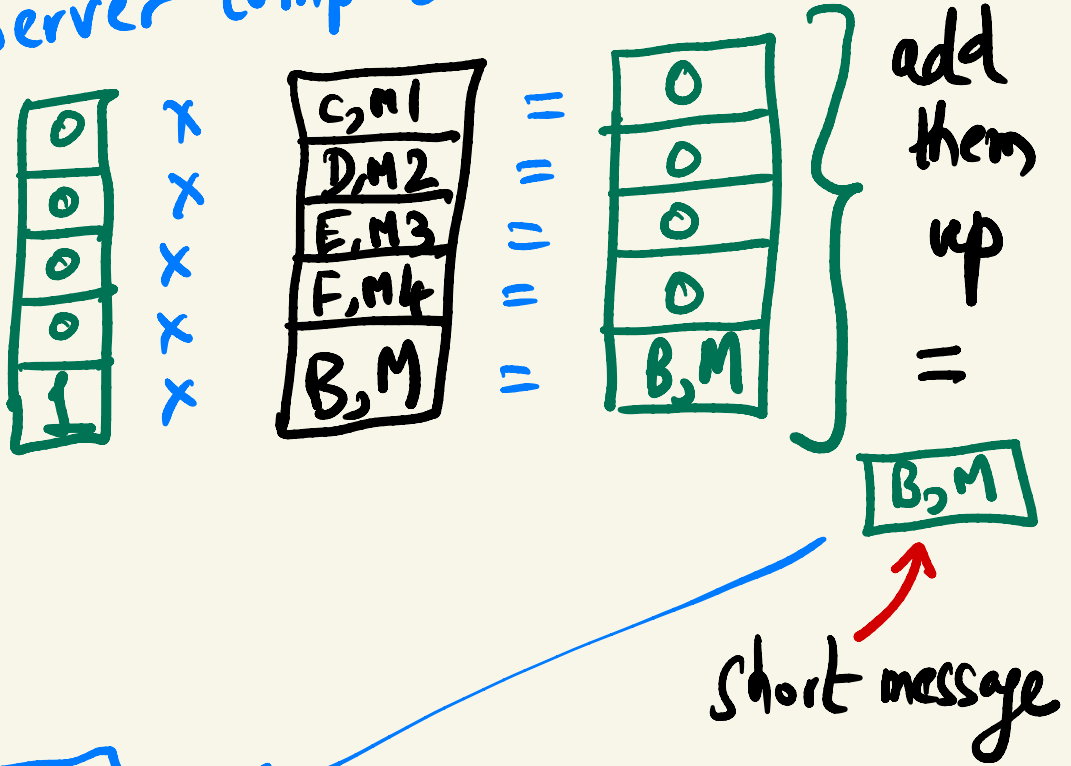
Bob's encryption is additively homomorphic

$$\text{Enc}(2) + \text{Enc}(5) = \text{Enc}(7)$$
$$3 \times \text{Enc}(2) = \text{Enc}(6)$$

effectively plaintext

PIR (contd.)

Server computes:



Bob

who decrypts B, M to
get B, M using A.H. key
and decrypts M using SK_{bob} .