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 (any better than guessing) which participant in the anonymity set did some action.

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

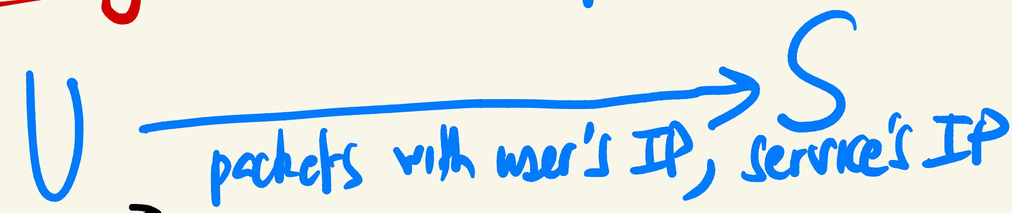
need lots of users, i.e., large anonymity sets

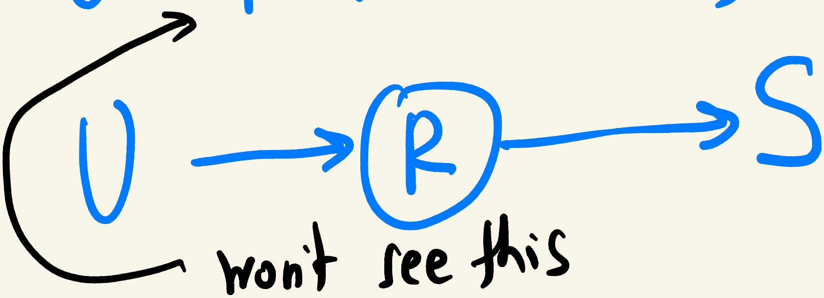
Perfect anonymity ⇒ 3 days for messages to arrive ⇒ 10 people use the system ⇒ low anonymity

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

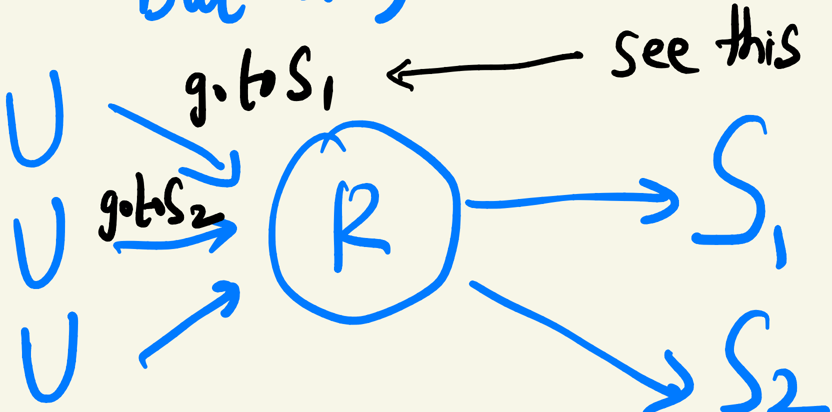
Design

Eavesdropper

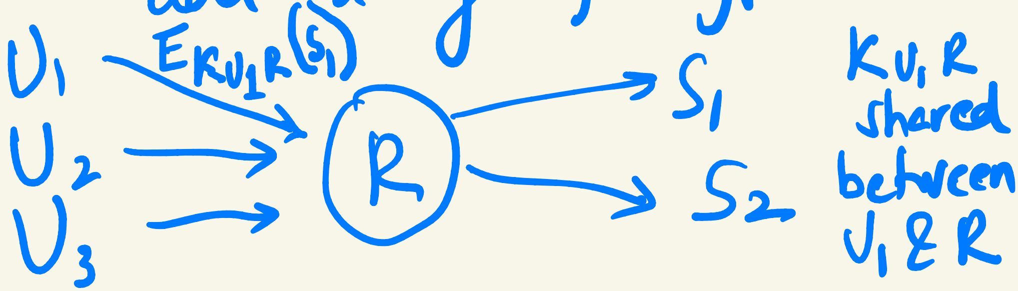




But only one user

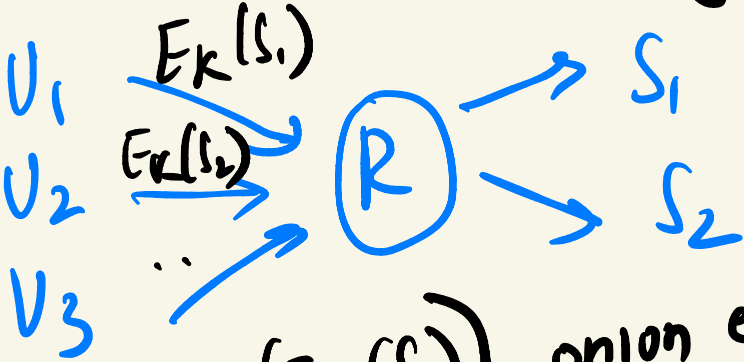


add a layer of encryption

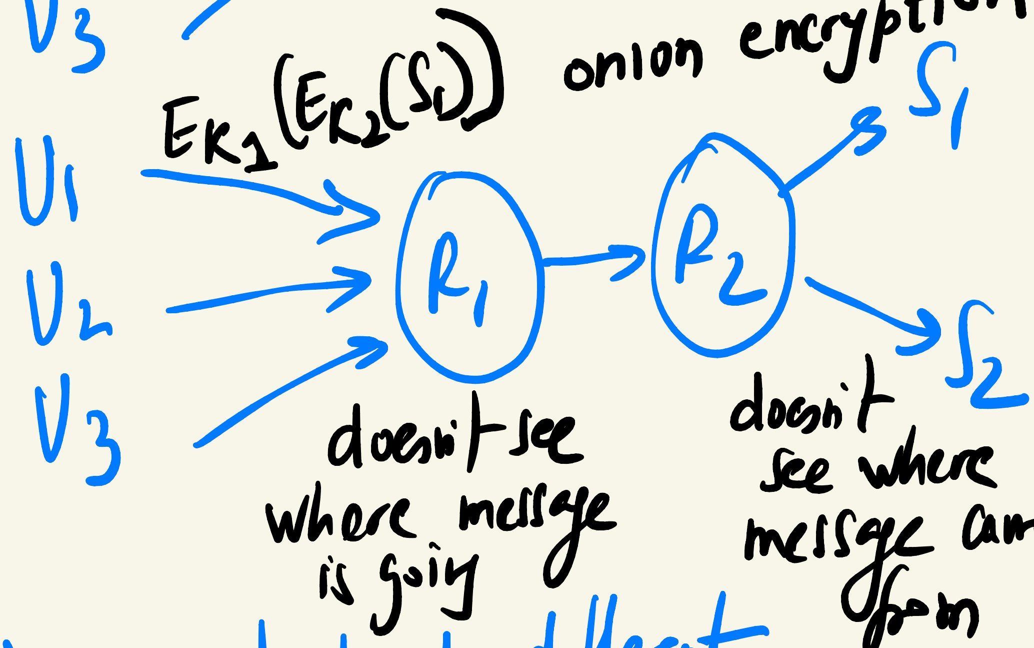


Design (contd.)

Adversary becomes relay!

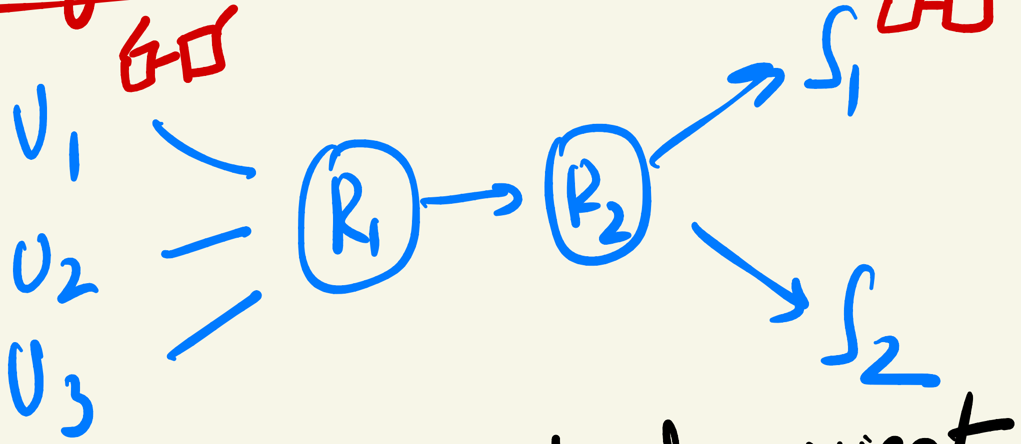


onion encryption!



Users want to do different things at different times (audio, video, web) ⇒ Traffic Analysis

Traffic Correlation



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

- Send same volume
- at the same time

computer failures?

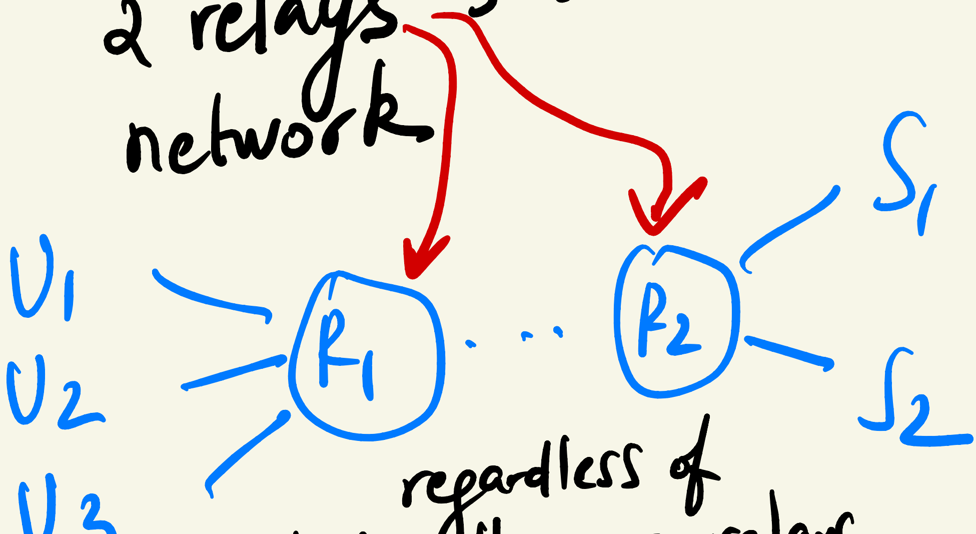
OR introduce random delays? — Large!

OR dummy messages? — Lot!

Attacker in Tor Assumption

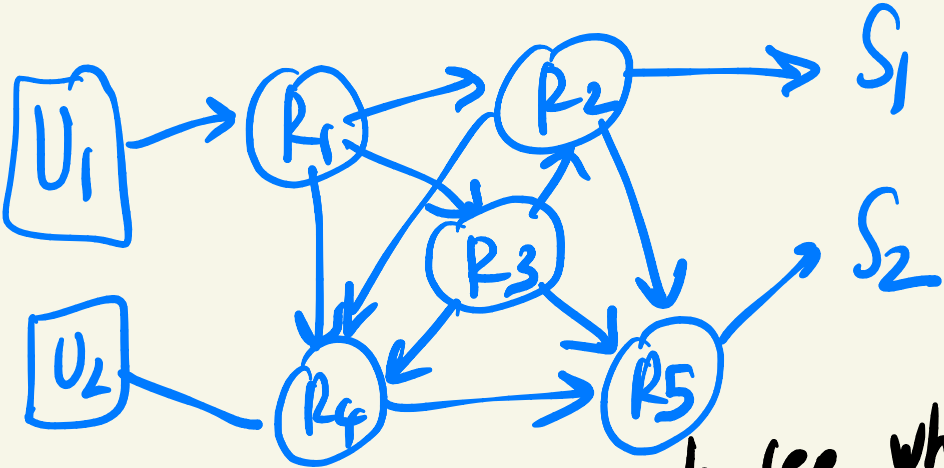
Not going to defend against an attacker who sees both ends!
- make it harder for anyone to be that attacker

- Attacker who compromises 2 relays sees the whole network



regardless of whether there are relays in between

Network of Relays



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

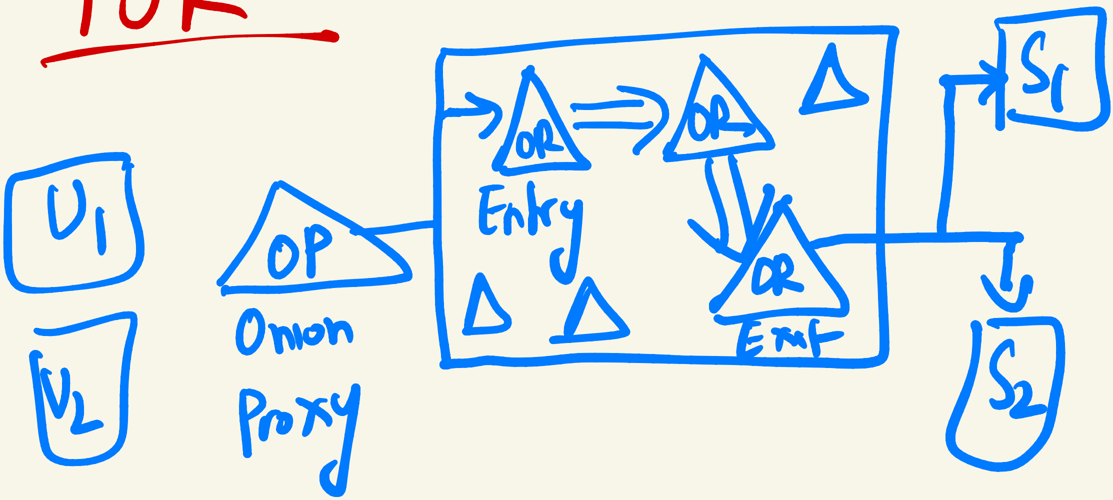
Efficiency

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

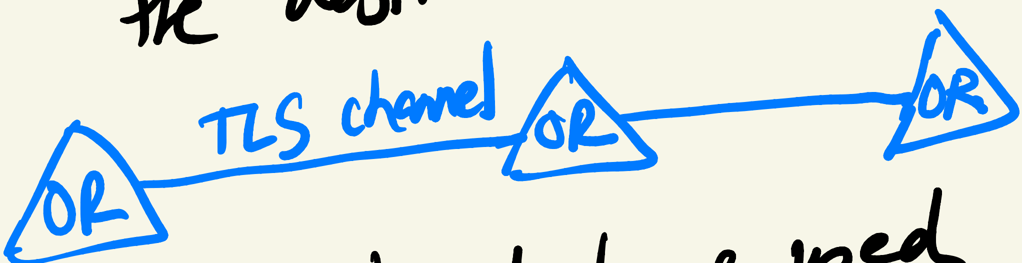


OR: Onion routers

1. Client U_i connects to proxy OP.
2. OP establishes a path to the destination address called a circuit, consisting of 3 ORs.
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.

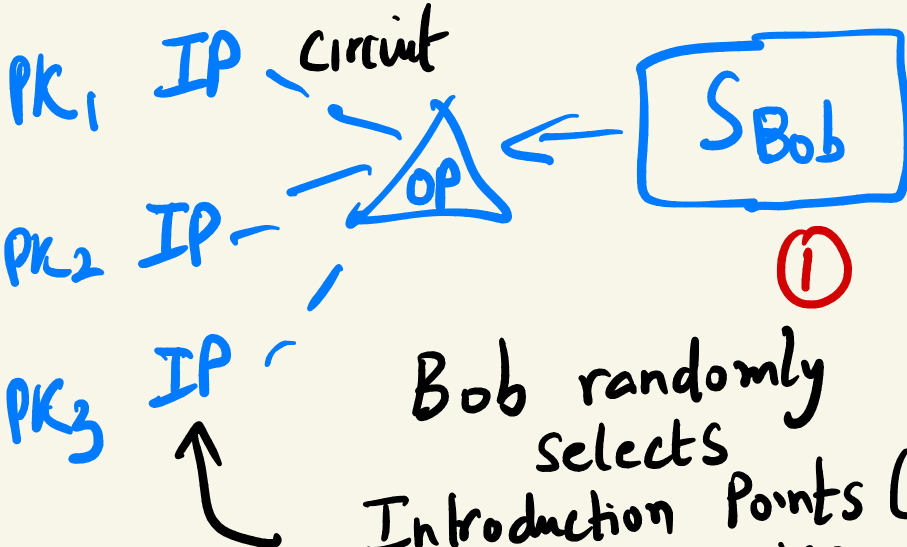


OP has shared keys used for onion encryption with every hop in the circuit (using key exchange similar to TLS).

$$E_{K_1}(E_{K_2}(E_{K_3}(U_1 \rightarrow S_1)))$$

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. ↔

Tor Basic Assumption

Adversary trying to figure out who is communicating with whom. Tor assumes adversary can only observe a <u>fraction</u> 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

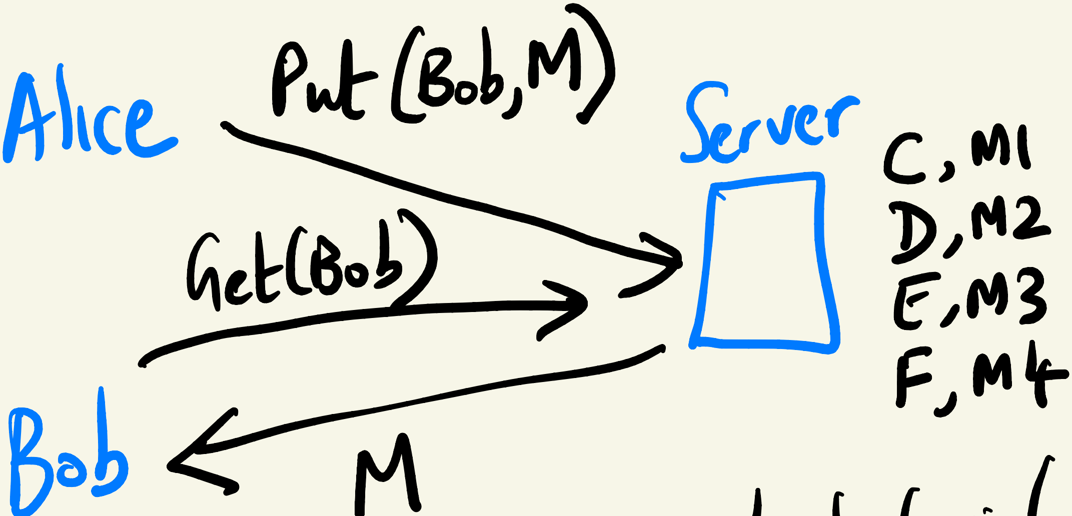
+ Onion encryption, forward secrecy, rotation of keys, circuit lifetime limit
- No real defense against traffic correlation using end-to-end timing & packet size. c^2/n^2 failure prob.

Website Fingerprinting

1) Adversary's clients visit website(s), network and create fingerprint 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 leek info

(1) Data and sizes

Use encryption to hide the content & padding to hide sizes

Pung (Contd.)

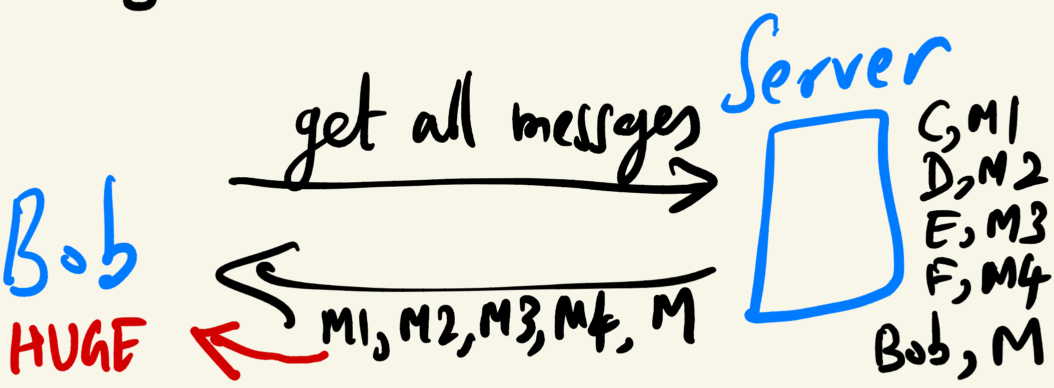
(2) Destination

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



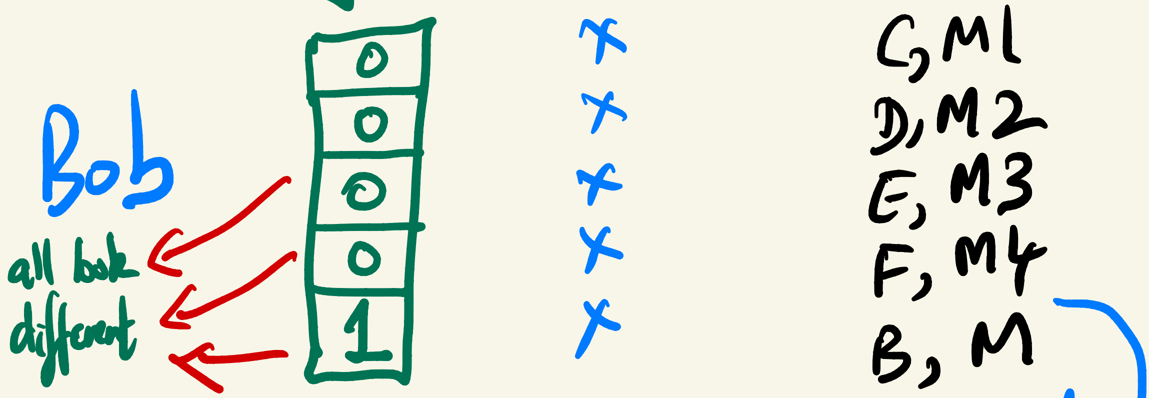
Private Information Retrieval

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

Alice & Bob share a key beforehand and agree on 5

Server

encrypted by A.H. scheme



Bob's encryption is additively homomorphic

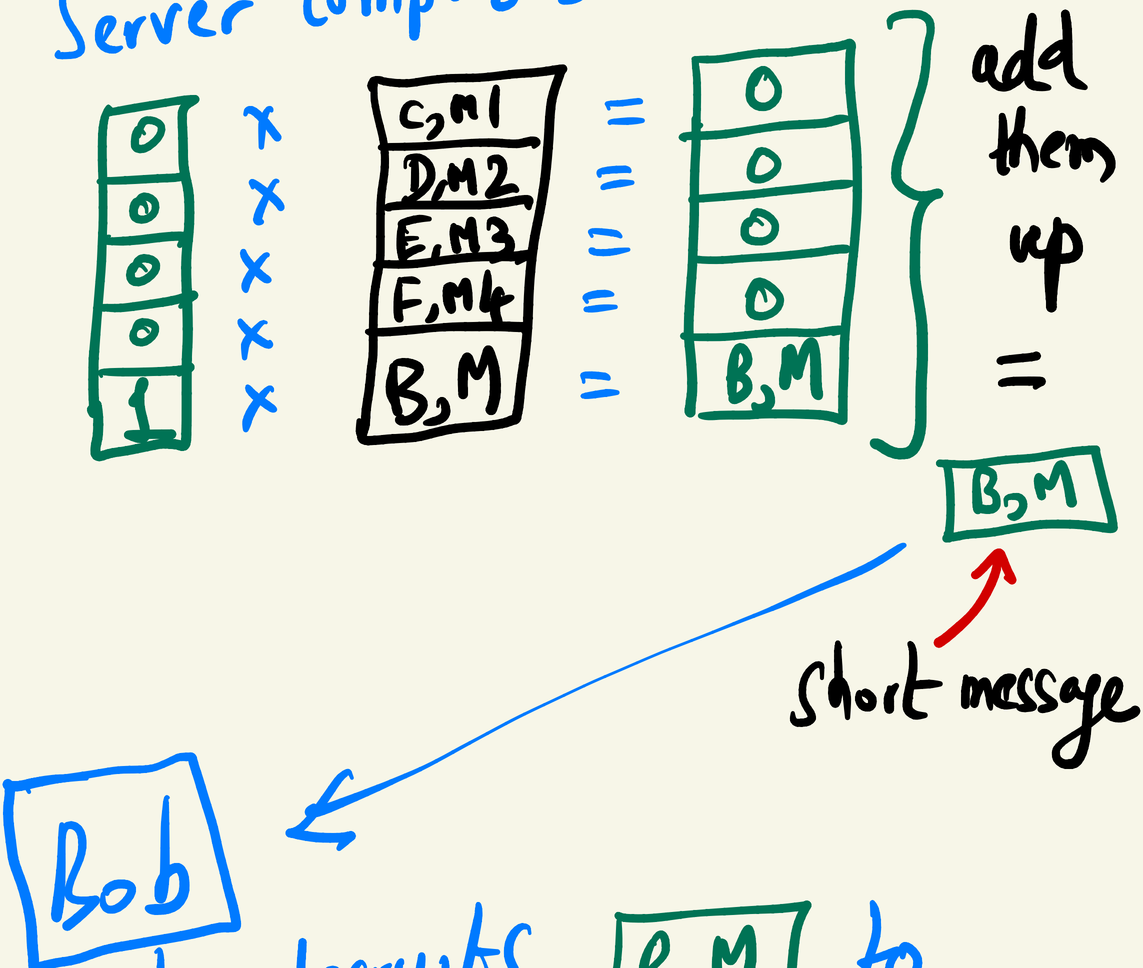
effectively plaintext

$$Enc(2) + Enc(5) = Enc(7)$$

$$3 \times Enc(2) = Enc(6)$$

PIR (contd.)

Server computes:



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