Privacy on Public Networks or Anonymity

- Internet is designed as a public network
  - Machines on your LAN may see your traffic, network routers see all traffic that passes through them
- Routing information is public
  - IP packet headers identify source and destination
  - Even a passive observer can easily figure out who is talking to whom
- Encryption does not hide identities
  - Encryption hides payload, but not routing information
Applications of Anonymity

• Privacy
  • Hide online transactions, Web browsing, etc. from intrusive governments, marketers and archivists
• Anonymous electronic voting
• Censorship-resistant publishing
Chaum’s Mix

• Early proposal for anonymous email

• Public key crypto + trusted re-mailer (Mix)
• Modern anonymity systems use Mix as the basic building block
Basic Mix Design

What if the mixer is compromised?
Idea: Randomized Routing

- Hide message source by routing it randomly
  - Popular technique: Crowds, Freenet, Onion routing
- Routers don’t know for sure if the apparent source of a message is the true sender or not
Onion Routing

Sender chooses a random sequence of routers

- Some routers are honest, some controlled by attacker
- Sender controls the length of the path

[Reed, Syverson, Goldschlag '97]
Route Establishment

- Routing info for each link encrypted with router’s public key
- Each router learns only the identity of the next router
Disadvantages of Basic Mixnets/Onion Routing

• Public-key encryption and decryption at each mix/router are computationally expensive

• Basic mixnets have high latency
  • Ok for email, not Ok for anonymous Web browsing

• Challenge: low-latency anonymity network
Tor

- Second-generation onion routing network
  - [https://www.torproject.org/](https://www.torproject.org/)
  - Developed by Roger Dingledine, Nick Mathewson and Paul Syverson
  - Specifically designed for low-latency anonymous Internet communications

- Running since October 2003

- Thousands of users

- “Easy-to-use” client proxy
  - Freely available, can use it for anonymous browsing
Tor Circuit Setup

• Client proxy establishes symmetric session keys with onion routers
Tor Circuit Setup (details)

- Routing info for each link encrypted with router's public key
- Each router learns only the identity of the next router and symmetric key with source
Using a Tor Circuit

• Client applications connect and communicate over the established Tor circuit
  • Note onion now uses only symmetric keys for routers
Using a Tor Circuit (details)

Note onion now uses only symmetric keys for routers
Tor Management Issues

• Many applications can share one circuit
  • Multiple TCP streams over one anonymous connection

• Tor router doesn’t need special privileges
  • Encourages people to set up their own routers
  • More participants = better anonymity for everyone

• Directory servers
  • Maintain lists of active onion routers, their locations, current public keys, etc.
  • Control how new routers join the network
    • “Sybil attack”: attacker creates a large number of routers
  • Directory servers’ keys ship with Tor code
Deployed Anonymity Systems

• Tor (http://tor.eff.org)
  • Overlay circuit-based anonymity network
  • Best for low-latency applications such as anonymous Web browsing

• Mixminion (http://www.mixminion.net)
  • Network of mixes
  • Best for high-latency applications such as anonymous email
Private Information Retrieval

A new primitive:

Private Information Retrieval (PIR)
How to protect privacy of queries?

User $U$ wants to retrieve some data from database $D$ without $D$ learning what $U$ retrieved.
Let's make things simple!

The user should learn $B_i$, for index $i = 1, \ldots, w$. Each $B_i \in \{0, 1\}$

Database $B$: $B_1, B_2, \ldots, B_i, \ldots, B_w$
Trivial solution

The database simply sends everything to the user!
Non-triviality

The previous solution has a drawback: the communication complexity is huge!

Therefore we introduce the following requirement:

“Non-triviality”:

the number of bits communicated between U and D has to be smaller than w.
**Private Information Retrieval (PIR)**

- **input:** index $i = 1, \ldots, w$
- **input:** $B_1, B_2, \ldots, B_w$

- At the end the user learns $B_i$
- The database does not learn $i$
- The total communication is $< w$

*Note:* secrecy of the database is not required.

- **correctness**
- **secrecy (of the user)**
- **non-triviality**

This property needs to be defined more formally.

- Polynomial time randomized interactive algorithms
Quadratic Residuosity Assumption (QRA):

For a random $a \in Z_N^+$ it is computationally hard to determine if $a \in \text{QR}(N)$.

Formally: for every polynomial-time probabilistic algorithm $G$ the value:
$$|P(G(a) = Q(a)) - 0.5|$$
(where $a$ is random) is negligible.
We are ready to construct PIR!

Our PIR will work in the group $\mathbb{Z}_N^+$, where $N=pq$.

What’s so good about this group?:

- testing membership in $\mathbb{Z}_N^+$ is easy,
- testing membership in $\text{QR}(N)$ is hard for random elements on $\mathbb{Z}_N^+$, unless one knows $p$ and $q$.

Example:
- $\mathbb{Z}^*_{15} = \{1,2,4,7,8,11,13,14\}$
- From these 1/4 are QRs (1,4) and the others are not
- If I give you an element at random, it is computationally infeasible to figure out if it is a quadratic residue or not (except for flipping coins)
First (wrong) idea

For every $j = 1, \ldots, w$ the database sets

$$Y_j = \begin{cases} X_j^2 & \text{if } B_j = 0 \\ X_j & \text{otherwise} \end{cases}$$

$Y_i$ is a QR iff $B_i = 0$

$M$ is a QR iff $B_i = 0$

The user checks if $M$ is a QR

Set $M = Y_1 \cdot Y_2 \cdot \ldots \cdot Y_w$
Problems!

PIR from the previous slide:

- **correctness** \( \checkmark \)
- **security?**

To learn \( i \) the database would need to distinguish **NQR** from **QR**. \( \checkmark \)

```
<table>
<thead>
<tr>
<th>QR</th>
<th>QR</th>
<th>...</th>
<th>QR</th>
<th>NQR</th>
<th>QR</th>
<th>...</th>
<th>QR</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_1 )</td>
<td>( X_2 )</td>
<td>...</td>
<td>( X_{i-1} )</td>
<td>( X_i )</td>
<td>( X_{i+1} )</td>
<td>...</td>
<td>( X_{w-1} )</td>
</tr>
</tbody>
</table>
```

- **non-triviality?** doesn’t hold!

**communication:**

- user \( \rightarrow \) database: \(|B| \cdot |Z^*_n|\)
- database \( \rightarrow \) user: \(|Z^*_n|\)

**Call it:** \((|B|, 1) - PIR\)
How to fix it?

**Idea:**
Given: \(|B| = v^2\)

construct \((|B|, 1)\)-PIR.

\[
\begin{bmatrix}
B_1 & B_2 & B_3 & B_4 \\
B_5 & B_6 & B_7 & B_8 \\
B_9 & B_{10} & B_{11} & B_{12} \\
B_{13} & B_{14} & B_{15} & B_{16}
\end{bmatrix}
\]

**Suppose** that \(|B| = v^2\) and present \(B\) as a \(v \times v\)-matrix:

consider each row as a separate database
An improved idea

Let $j$ be the column where $B_i$ is.

In every "row" the user asks for the $j$th element

So, instead of sending $v$ queries the user can send one!

Observe: in this way the user learns all the elements in the $j$th column!
Putting things together

<table>
<thead>
<tr>
<th>QR</th>
<th>X₁</th>
<th>...</th>
<th>QR</th>
<th>NQR</th>
<th>QR</th>
<th>Xᵢ</th>
<th>...</th>
<th>QR</th>
<th>Xᵥ</th>
</tr>
</thead>
</table>

Multiply elements in each row

\[ B_j = 0 \text{ iff } M_k \text{ is QR} \]

For every \( j = 1, \ldots, v \) set

\[ Y_j = \begin{cases} X_j^2 & \text{if } B_j = 0 \\ X_j & \text{otherwise} \end{cases} \]

Multiply elements in each row

Here the same row is copied \( v \) times:

\[
\begin{array}{cccccccc}
B_1 & \ldots & B_{j-1} & B_j & B_{j+1} & \ldots & B_v \\
\vdots & & \vdots & \boxed{B_j} & \vdots & \ddots & \vdots \\
X_1 & \ldots & X_{j-1} & X_j & X_{j+1} & \ldots & X_v \\
\vdots & & \vdots & \boxed{X_j} & \vdots & \ddots & \vdots \\
Y_1 & \ldots & Y_{j-1} & Y_j & Y_{j+1} & \ldots & Y_v \\
\vdots & & \vdots & \boxed{Y_j} & \vdots & \ddots & \vdots \\
\end{array}
\]
So we are done!

PIR from the previous slide:
- **correctness** √
- **non-triviality**: communication complexity = \(2\sqrt{|B| \cdot |Z_n|}\) √
- **security**?
  
The to learn i the database would need to distinguish NQR from QR.

**Formally:**

from any adversary that **breaks our scheme**
we can construct an algorithm that **breaks QRA**