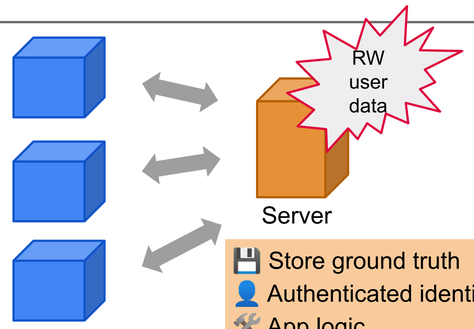
Gaggle: a Private and Consistent Communication Model

Shai Caspin, Natalie Popescu, and Amit Levy; Princeton University

Traditional application architecture is unfit for protecting user privacy



- 💾 Store ground truth
- 👤 Authenticated identity
- 🛠️ App logic
- 🚪 Access control
- 🌎 Consistency logic
- 🤼 Conflict resolution

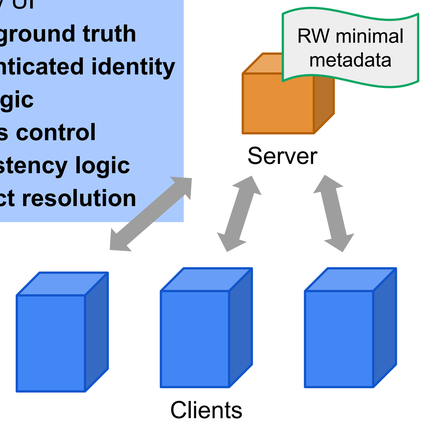
- 💡 Interpret user-intraxns
- 🎨 Display UI

How can we achieve the same application functionality without requiring server access to user data and most metadata?

Gaggle: an alternative, private architecture

- 💡 Interpret user-intraxns
- 🎨 Display UI
- 💾 **Store ground truth**
- 👤 **Authenticated identity**
- 🛠️ **App logic**
- 🚪 **Access control**
- 🌎 **Consistency logic**
- 🤼 **Conflict resolution**

- 📶 **Message ordering**



The Gaggle architecture enables higher levels of privacy for a wide range of applications

User Devices

Local Database

- Committed state 💾
- History of uncommitted changes
- Groups 👥

Stores all data and metadata used by app and Gaggle library

The Gaggle Library

- Authenticated identity 👤
- Access control 🚪
- Consistency Logic 🌎
- Conflict Resolution 🤼

Lets app developers easily use Gaggle's protocols for simple cross-user sharing

App

- Intraxns 👇
- UI 🎨
- App logic 🛠️
- Data type invariants

API

```
new_user()
new_device()
get()
set()
remove()
...
```

App devs write local-only app that plugs into Gaggle and *automatically* get:
- Cross-device sync
- E2E privacy

Server

- Accepts incoming messages from clients
- Assigns sequence IDs to incoming messages 📶
- Dispatches messages to all mailboxes 📫 as a *transaction*
- Tracks pending transactions

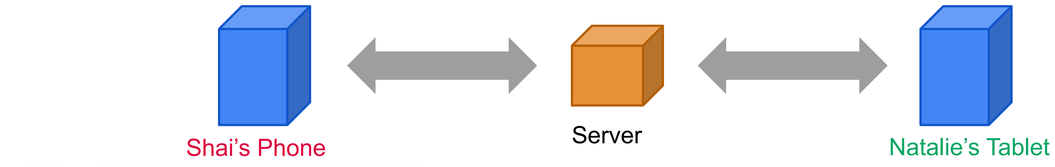
Sequencer

Public Key Directory ↓ Encrypted Mailboxes

834073	23: y92dkf…nd2801
958101	28: xi3v6k…99f3g5, 39: lsih8g…eijr8h7
⋮	⋮
870054	21: n4h30j…903bdj

🌎 Consistency protocol

How does a data change on one client reach another?



TIME

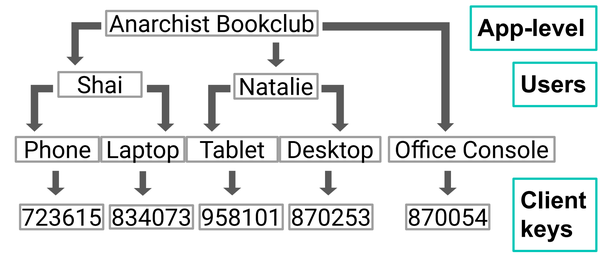
1. Shai makes an **update** on the Gaggle-based application
2. Gaggle updates local state
3. Gaggle adds update to **uncommitted local history**
4. Gaggle determines that update should also go to Natalie's Tablet 👥
5. Gaggle packages update as a **"message"** and sends it to the server
6. Server generates a **sequence ID** for the message 📶
7. Server forwards message to Natalie's Tablet's **mailbox** 📫
8. Server sends **ACK** to Shai's Phone with the assigned sequence ID
9. Receives ACK, links sequence ID with update, **commits history up to and including update**, and clears uncommitted history
10. Receives message from server
11. Gaggle **compares message sequence ID** with the highest sequence ID in uncommitted local history: if larger, **appends update to uncommitted history**; else, **rollback** uncommitted history and **re-apply** updates in order of increasing sequence ID 🤼
12. Gaggle updates local state
13. Natalie's application view now reflects Shai's update

📱 Suitable Applications

Ideal application properties:

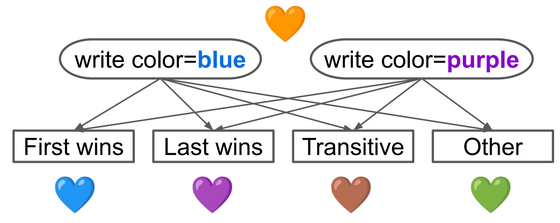
1. The amount of data stored fits on a single device
   - **Yes:** text messaging, note taking, games
   - **No:** search engine
2. Users generate data
   - **Yes:** period tracker, fitness tracker
   - **No:** weather app, maps, streaming
3. Data is shared in "small" circles
   - **Yes:** book club app, medical communication, financial tracking, neighborhood restaurant recommendation, money transfer
   - **No:** social media

👥 Groups



- ➔ Gaggle enforces sharing policies via groups 🚪
- ➔ Composable
- ➔ Kept consistent just like any other data in Gaggle

🤼 Conflict Resolution



- ➔ Message ordering from Gaggle server provides strong consistency
- ➔ Sequence IDs enable deterministic conflict resolution via developer-chosen techniques