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

Gaggle: a Private and Consistent Communication Model
Shai Caspin, Natalie Popescu, and Amit Levy; Princeton University

Gaggle updates local state
Gaggle adds update to uncommitted local history
Gaggle determines that update should also go to Natalie’s Tablet
Gaggle packages update as a “message” and sends it to the server

Consistency protocol

How does a data change on one client reach another?

Shai’s Phone
Server
Natalie’s Tablet

Server

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

User Devices

Local Database
• Committed state
• History of uncommitted changes
• Groups

The Gaggle Library
• Authenticated identity
• Access control
• Consistency Logic
• Conflict Resolution

App
• Intraxns
• UI
• App logic
• Data type invariants

API
devs write local-only app that plugs into Gaggle and automatically gets:
- Cross-device sync
- E2E privacy

Server

Public Key Directory
Encrypted Mailboxes

834073: 23: y922df...nd2801
958101: 28: xju3zk...9559p
39: tlahg...eij867
870054: 21: n4h30j...808bd

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

Anarchist Bookclub
App-level
Users

Shai
Natalie
Office Console
Phone
Laptop
Tablet
Desktop

Client keys

→ Gaggle enforces sharing policies via groups
→ Composable
→ Kept consistent just like any other data in Gaggle

Suitable Applications

First wins
Last wins
Transitive
Other

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

Conflict Resolution

Shai makes an update on the Gaggle-based application
Gaggle updates local state
Gaggle adds update to uncommitted local history
Gaggle determines that update should also go to Natalie’s Tablet
Gaggle packages update as a “message” and sends it to the server

Server generates a sequence ID for the message
Server forwards message to Natalie’s Tablet’s mailbox
Gaggle compares sequence IDs, applies update if sequence ID is higher
Natalie receives message from server
Gaggle updates local state
Natalie’s application view now reflects Shai’s update

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

Shai’s Phone
Server
Natalie’s Tablet

Server

Public Key Directory
Encrypted Mailboxes

834073: 23: y922df...nd2801
958101: 28: xju3zk...9559p
39: tlahg...eij867
870054: 21: n4h30j...808bd

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

Anarchist Bookclub
App-level
Users

Shai
Natalie
Office Console
Phone
Laptop
Tablet
Desktop

Client keys

→ Gaggle enforces sharing policies via groups
→ Composable
→ Kept consistent just like any other data in Gaggle

Conflict Resolution

Shai makes an update on the Gaggle-based application
Gaggle updates local state
Gaggle adds update to uncommitted local history
Gaggle determines that update should also go to Natalie’s Tablet
Gaggle packages update as a “message” and sends it to the server

Server generates a sequence ID for the message
Server forwards message to Natalie’s Tablet’s mailbox
Gaggle compares sequence IDs, applies update if sequence ID is higher
Natalie receives message from server
Gaggle updates local state
Natalie’s application view now reflects Shai’s update

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

Shai’s Phone
Server
Natalie’s Tablet

Server

Public Key Directory
Encrypted Mailboxes

834073: 23: y922df...nd2801
958101: 28: xju3zk...9559p
39: tlahg...eij867
870054: 21: n4h30j...808bd

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

Anarchist Bookclub
App-level
Users

Shai
Natalie
Office Console
Phone
Laptop
Tablet
Desktop

Client keys

→ Gaggle enforces sharing policies via groups
→ Composable
→ Kept consistent just like any other data in Gaggle

Conflict Resolution

Shai makes an update on the Gaggle-based application
Gaggle updates local state
Gaggle adds update to uncommitted local history
Gaggle determines that update should also go to Natalie’s Tablet
Gaggle packages update as a “message” and sends it to the server

Server generates a sequence ID for the message
Server forwards message to Natalie’s Tablet’s mailbox
Gaggle compares sequence IDs, applies update if sequence ID is higher
Natalie receives message from server
Gaggle updates local state
Natalie’s application view now reflects Shai’s update

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