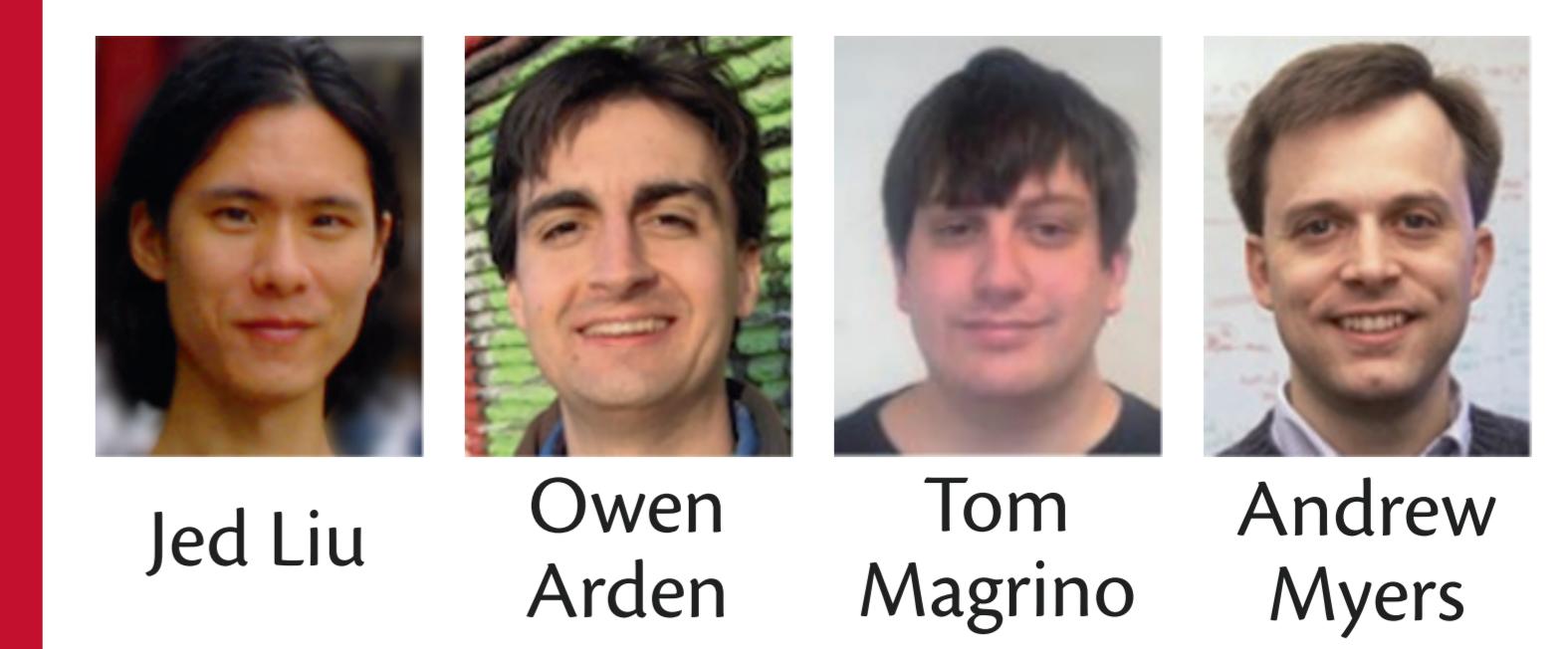


A high-level abstraction for Securely Programming the Internet

www.cs.cornell.edu/projects/fabric



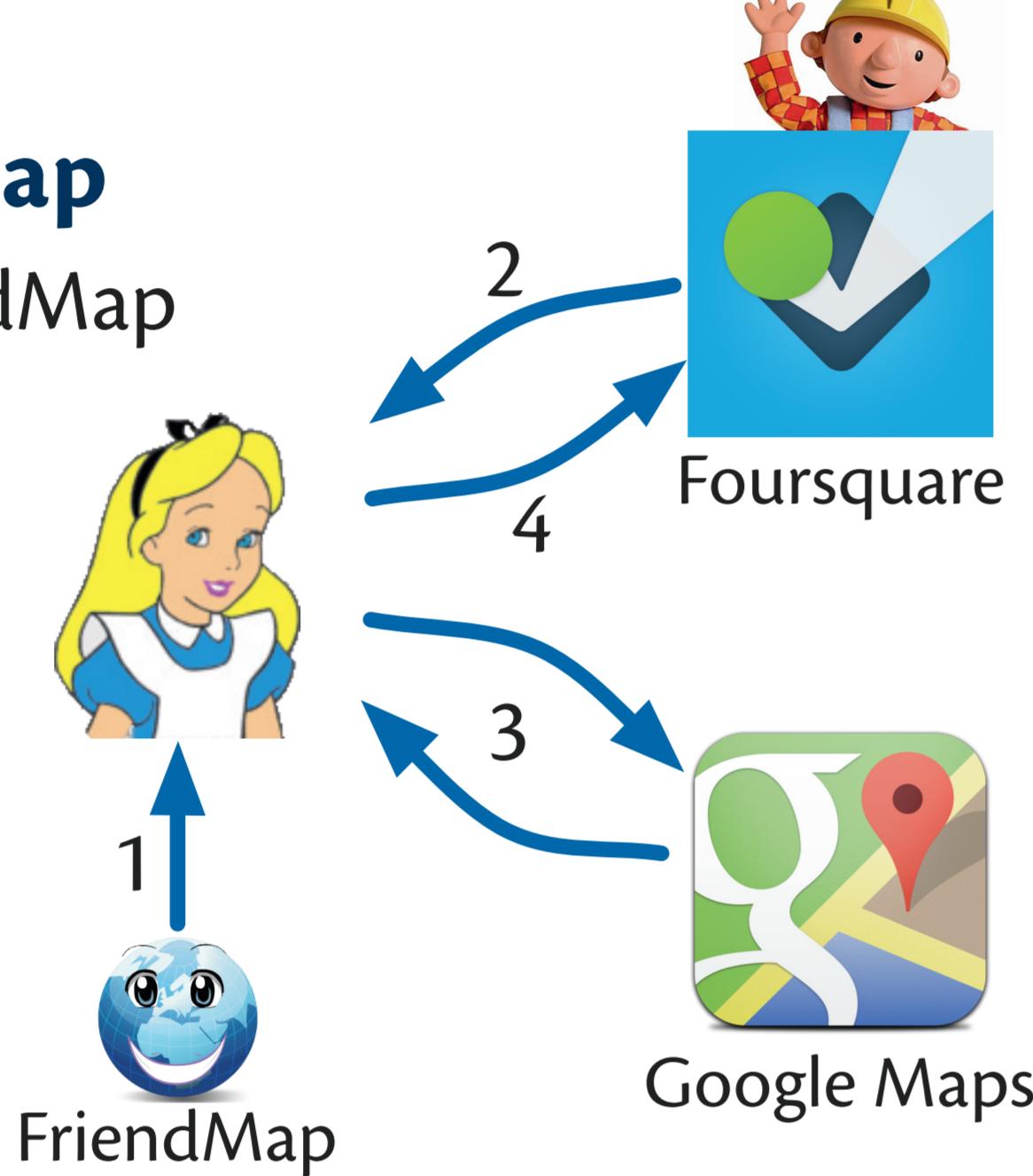
Decentralized sharing

Goal

An open, decentralized platform that makes it easy to build secure software that integrates code and data across organizational and trust boundaries

Example application: FriendMap

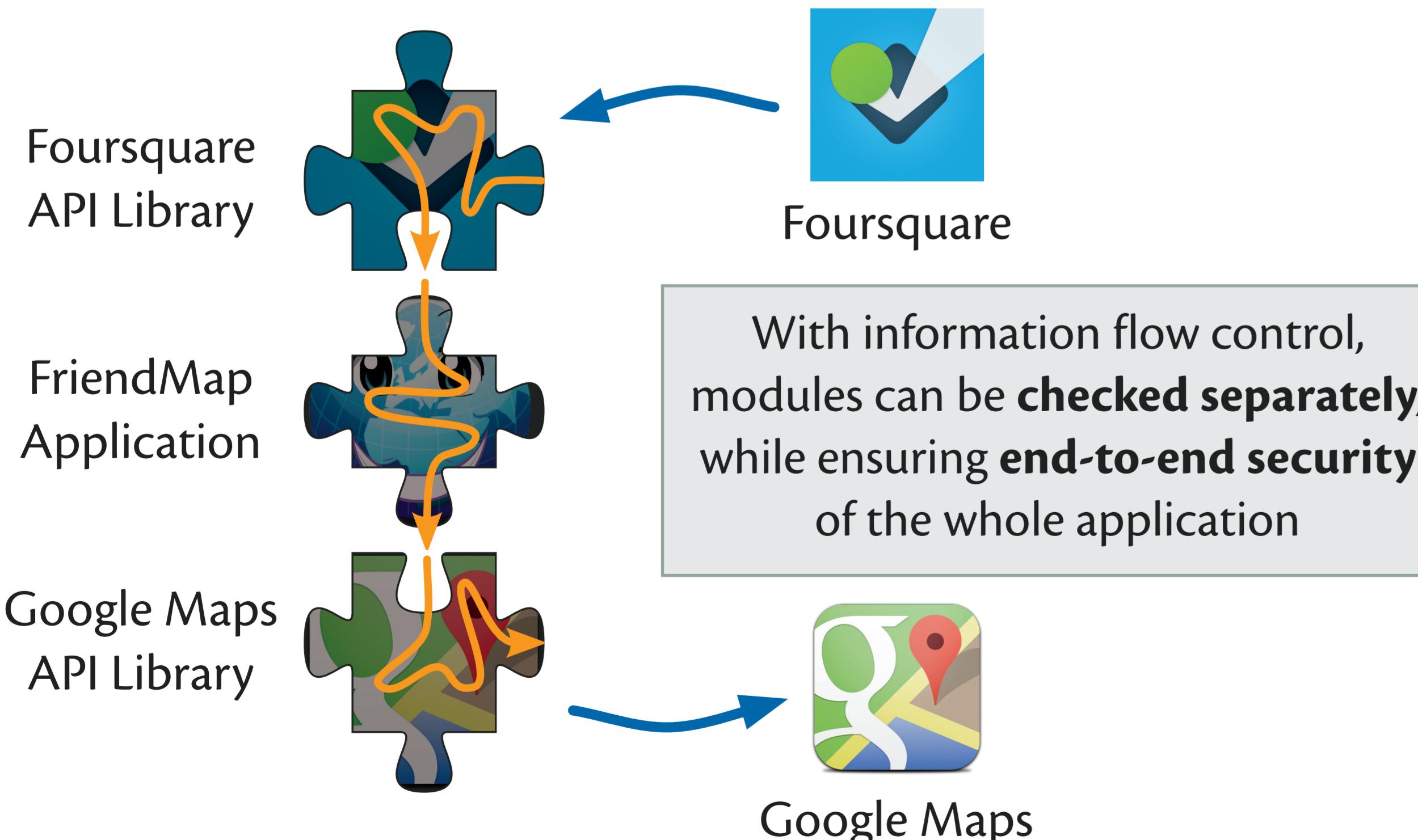
1. Alice downloads and runs FriendMap
2. Fetch friends' locations
3. Get a map and place pins for nearby friends
4. Post to network



Must ensure code running on Alice's client doesn't leak Bob's information appropriately.

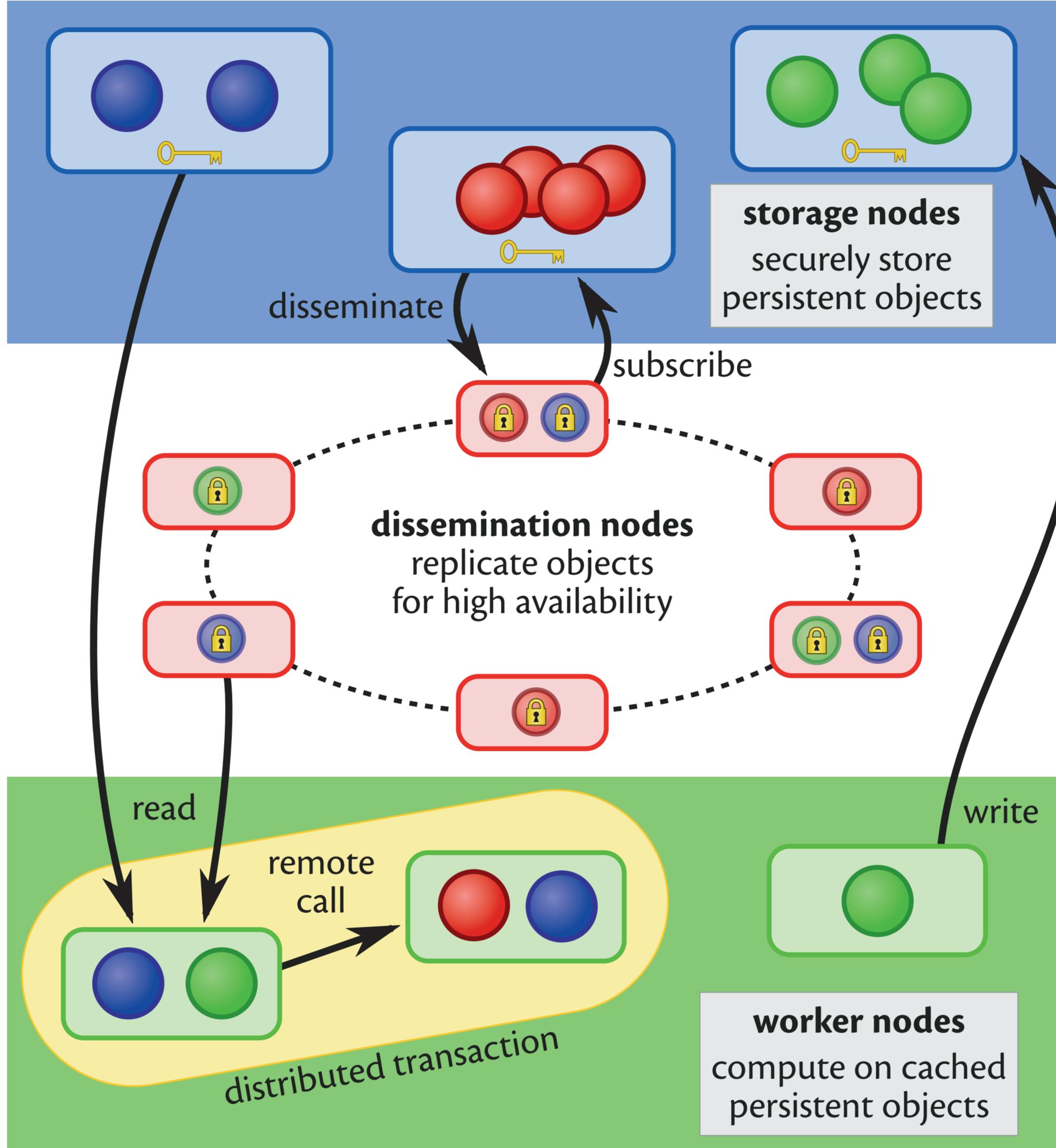
Composable decentralized security

- Access control is inadequate
 - Only controls **who** is allowed to see information
 - Doesn't constrain **what** is done with it
- **Information flow control** is inherently compositional
 - Rejects **insecure data flows** instead of users
 - Decentralized: enforces confidentiality & integrity for all principals in the system

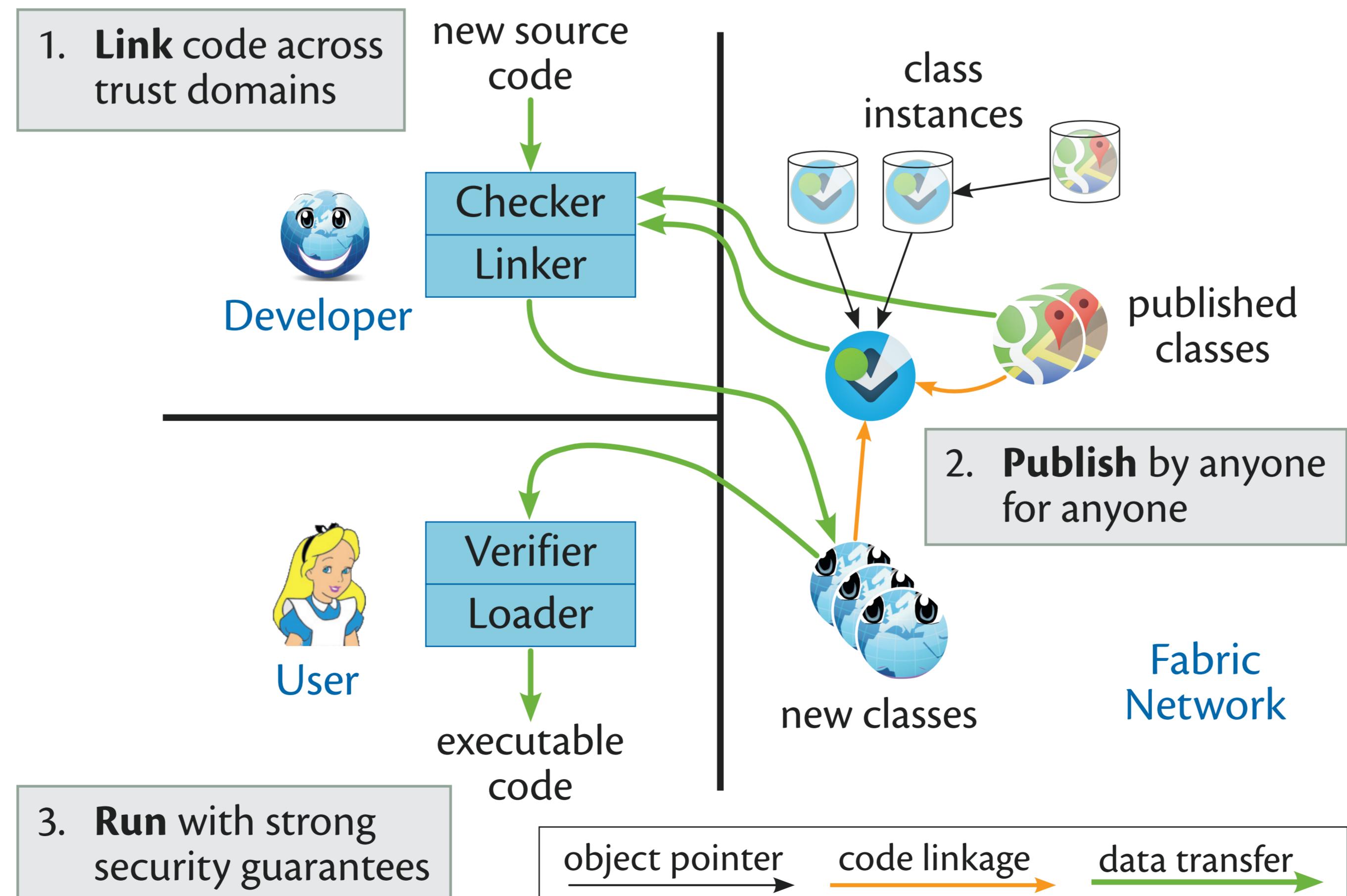


System model

System architecture

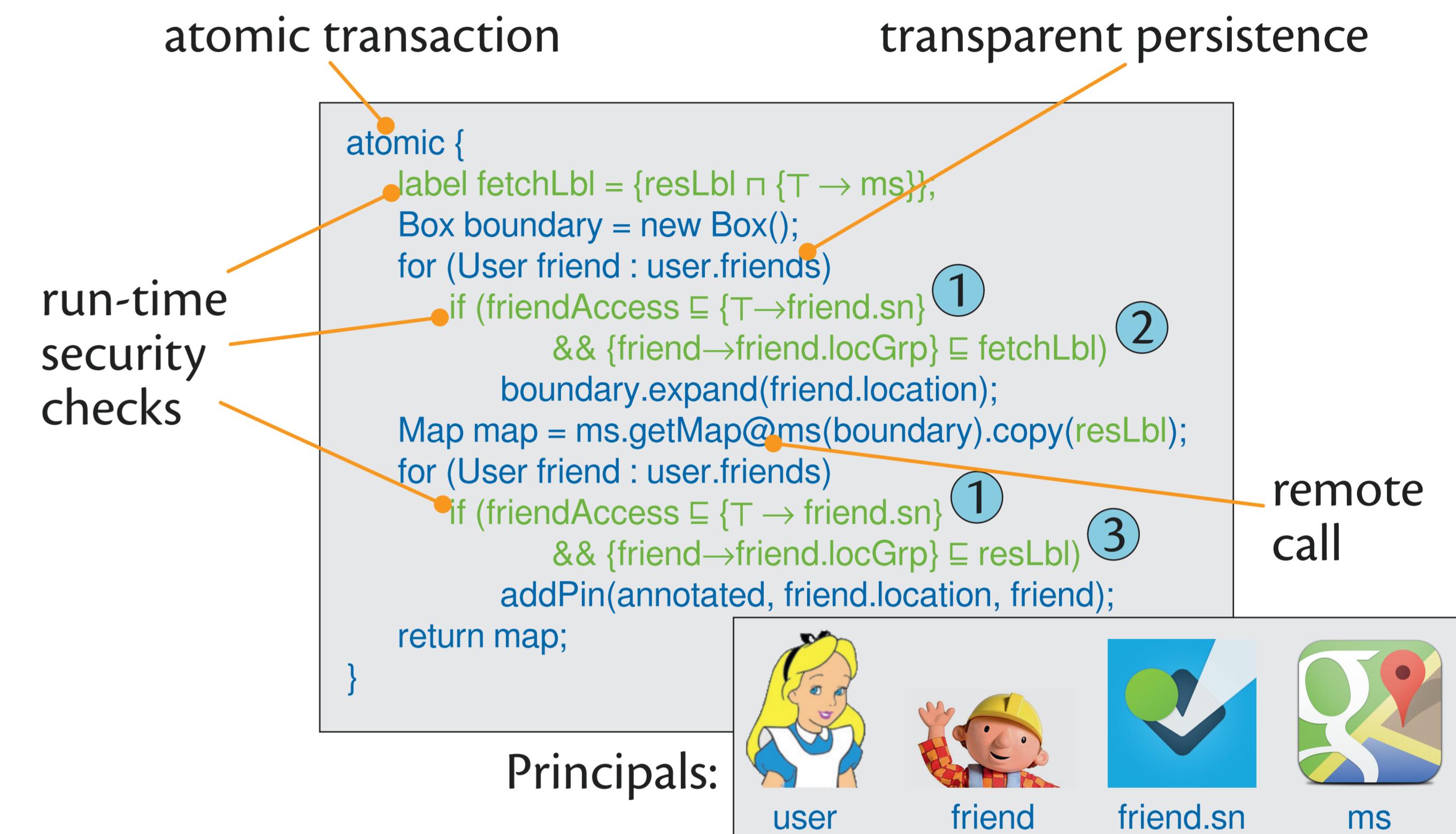


Software lifecycle



Programmability & Security

Code example



Security checks forced by compiler:

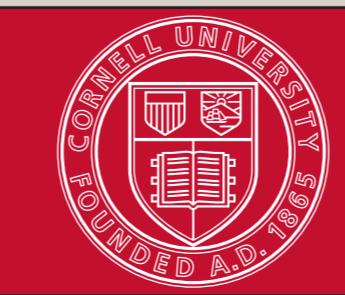
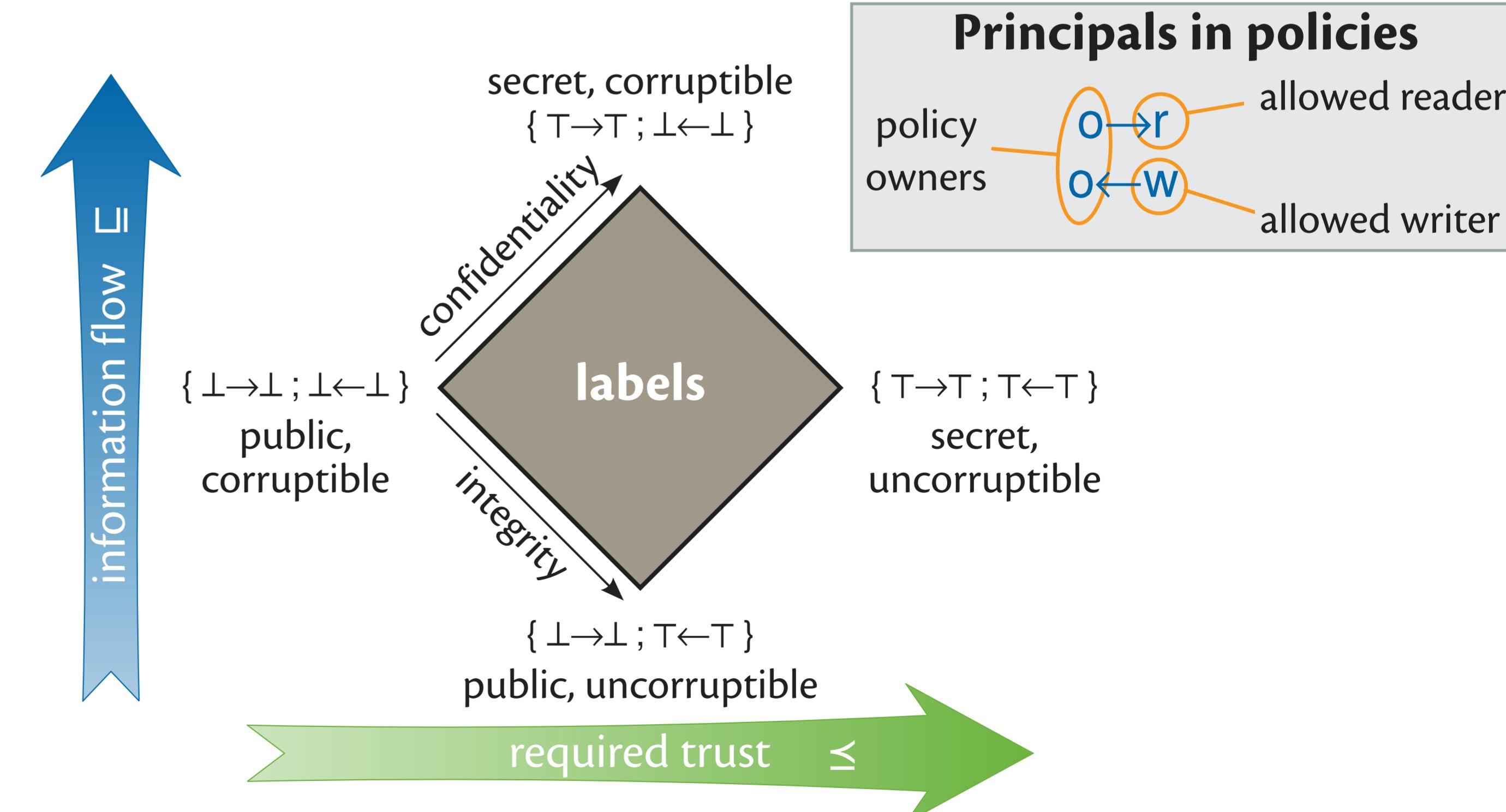
- ① Querying social network doesn't leak user's or friend's information.
- ② Friend allows map service to learn location.
- ③ Friend allows location to be in posted result.

Decentralized security principle

You can't be harmed by what you don't trust

Decentralized security labels

- Base policies: confidentiality ($o \rightarrow r$) and integrity ($o \leftarrow w$)
- Label ordering \sqsubseteq specifies allowed information flows
- Information flows can be checked statically:
 - for statement $y=x$, compiler checks $L_x \sqsubseteq L_y$



Cornell University

Past contributors:
Aslan Askarov
Michael D. George
Xin Qi
K. Vikram
Lucas Waye