

Privacy-preserving “electronic civic infrastructure”

Andrew J. Blumberg (blumberg@math.utexas.edu)

Objective: design potentially invasive “electronic civic infrastructure” services to preserve privacy.

Examples to keep in mind:

- ① Automated tolling/ “pay as you drive” insurance.
- ② Electronic payment systems (debit cards).
- ③ Access cards (e.g., bike room at train station).
- ④ “Find a friend” location services.
- ⑤ Aggregate statistics computation (e.g., average speed on various roadways).
- ⑥ Cell phones.

Problem: Default implementation is invasive.

- 1 Track you everywhere, figure out what tolls/ insurance premiums you owe.
- 2 Record all of your purchases, debit your account.
- 3 Record your identity, decide if you're entitled to access.
- 4 Track you everywhere, decide if your friends are nearby.

Flexible and cheap, but susceptible to serious privacy violations — permits arbitrary offline analysis and searches.

This is something to be very worried about:

- ① Inconsistent with civic life as we currently understand it: rule of law violations, for instance.
- ② Dangerous: corrupt employees can misuse the data, even if you trust institutions.
- ③ Coercive: “electronic civic infrastructure” will be hard to avoid and still live a normal life.
- ④ Legal: because many of these are about activity in public space or “voluntary” activities, few legal protections.

Location privacy

- We will focus on discussion of “locational privacy” (or “location privacy”): privacy while moving in public space.
- Locational privacy is hard: qualitative change occurring because of quantitative change. Always legal to track you, but constant, pervasive, silent tracking is something different.
- Violation of reasonable expectation of privacy “most of the time”.

- Legal/ legislative solutions mandating proper management of such information are essential.
- However, the best way to avoid misuse of location information is *not to collect it* in the first place.
- Modern cryptography makes it possible to design systems which provide location-based services while collecting/ revealing only the minimum amount of information necessary.

Instead, goal is to design systems which reveal/ capture only the minimum amount of information needed. For example:

- Electronic tolling protocols should reveal only the total amount I owe during a billing period.
- Access card protocols should reveal only that I am authorized to enter a secured area.
- Aggregate statistics computation should reveal only the statistic in question.

Slogan: Don't need to track me everywhere to provide location-based services.

Key cryptographic primitives:

- Blind signatures. “Sign across sealed envelope.”
- Zero knowledge proofs.
- Secure multi-party protocols.

Blind signatures:

Two parties, Alice and Bob. Alice wants a signature from Bob on her secret data m *without revealing* m .

- RSA ordinary signing: compute $m^d \pmod N$, where d is Bob's secret key.
- Blind version: have Bob sign $mr^e \pmod N$, where Alice chooses r randomly (and relatively prime to N).

Can be used to implement electronic cash.

Zero-knowledge proofs:

- 1 Interactive protocol: randomized tests.
- 2 Example of “counting gumballs in a jar”: I prove I can do this, as follows:
 - 1 You ask how many gumballs are in the jar.
 - 2 I reply with a number.
 - 3 You randomly remove 1 or 2 gumballs, while my eyes are closed.
 - 4 Repeat.
- 3 Works for a very broad range of computations.

Can be used to implement credential systems.

For some applications, goal can be achieved with essentially “off-the-shelf” cryptographic protocols:

- Anonymous electronic cash provides a perfect solution for “point tolling” (i.e., tollbooths) and subway fares.
- Anonymous unlinkable credentials provide a perfect solution for access cards.

(Camenisch, Lysyanskaya, and co-authors have applicable recent work on this subject.)

However, for many applications, such as

- Sophisticated congestion pricing,
- Aggregate statistics computation, and
- “Friend nearby” notification,

specialized protocols must be designed.

Main observation: It is possible to design *efficient* protocols for a wide range of applications which achieve *provable privacy guarantees*.

At a high level, we know this sort of thing is generically possible using a big hammer from modern cryptography: *secure multi-party computation*.

This allows multiple individuals who don't trust each other to collaboratively compute a function of private information such that:

- No one learns anyone else's secret information, but
- Everyone is convinced that if the protocol succeeds, the function was computed correctly, and
- If anyone tries to cheat, the protocol will fail.

Examples of such problems include:

- (The classic) Millionaire's problem: you and I can learn which of us has more money without revealing our actual personal wealth. (Introduced in Yao.)
- Tolling: you and the tolling agency can learn how much you owe without revealing your path.
- (Real world example) Secret ballot voting — result is obtained without revealing which way any individual voted.

Vision:

- To support location-based services, I store my location information in an encrypted format.
- As the need arises, the server and I engage in a secure protocol to compute various functions of the data, *without revealing its decrypted value to the server.*

Key property:

- Raw location data is never revealed: server learns only results of *mutually agreed upon* computation — data mining is prevented.

Unfortunately, general-purpose compilers (e.g., Fairplay), which take protocol specifications expressed in a subset of C to implementation as secure multi-party computation are *extremely inefficient*.

Technical problem: design *efficient* protocols which achieve similar privacy guarantees.

- User protocol should run on a smartphone or tolling transponder.
- Server protocol should require a manageable number of commodity hardware servers.

In work with Hari Balakrishnan and Raluca Popa (MIT Cartel group), we've built:

- System for congestion pricing/ sophisticated tolling protocols.
- System for aggregate statistics computation.

Cryptographic tools:

- *commitment schemes* (I give you a promise I'm holding a certain value without revealing that value).
- *homomorphic encryption* (operation on ciphertext corresponds to operation on unencrypted data).
- *zero-knowledge proofs of knowledge* (proof that you know a property of a hidden value which you've committed to).

Despite sophisticated protocols, user experience is straightforward. For example, tolling is very similar to current setup:

- At the beginning, user registers device, performs initialization/handshaking with server (to get cryptographic tokens, for instance).
- While driving, periodically the user's device interacts with server to upload encrypted location information.
- At the end of some period (e.g., a month), the user reconciles with the server and is charged for tolls accrued. This could happen automatically, or via a manual web interface.

In slightly more detail:

- During initialization, user's device commits to a long sequence of random "license plates" — these are kept hidden from server, but server holds evidence of commitment.
- While driving, user's device periodically uploads anonymized location data along with a license plate.
- During reconciliation, user and server jointly compute tolls owed *without user revealing which license plates she holds*; commitments ensure honesty.

Aggregate statistics computation forces different user assumptions:

- Can't assume users will comply with or complete protocol (no legal compulsion).
- Spurious uploads (to bias statistics) a significant concern.

Necessitates different protocol design:

- During registration, user must obtain anonymous authorization tokens permitting bounded number of uploads in a particular location; tokens are unforgeable, compact, unlinkable, and do not reveal location to which they are attached.

Our systems provide an *existence proof*: it is feasible to build location-based services which are efficient and protect location privacy.

- Varying levels of formal privacy guarantees; more privacy requires more computation, more infrastructure.
- Protection against various attacks/ efforts to misbehave by both users and participants.
- Efficient implementations (will run on smartphones, in-car RFID transponders, stock hardware servers).

Some caveats:

- Anonymizing databases is hard: location databases with (location, time) pairs leak a surprising amount of information!
- “Electronic” protocol design only part of the story; various other system design questions, some exacerbated by anonymity (e.g., ensuring compliance, preventing spoofing).
- Loss of flexibility; some modifications require re-design of system.
- Although protocols do run on smartphones/ stock hardware, use of cryptography does impose a computational/ programming burden (e.g., increased implementation complexity).

- Nonetheless, since it is *possible* and *practical* to design systems for which the default is to preserve location privacy, this seems like the right thing to do.
- Consequence of *existence proof*: It is reasonable for governmental procurement documents to stipulate protection of location privacy as a requirement.

Links to find references:

- <http://nms.csail.mit.edu/projects/privacy/>
- <http://cartel.csail.mit.edu/>
- http://www.math.utexas.edu/~blumberg/loc_writing.html