Alin Tomescu, CSE408
Thursday, May 12th, Lecture #27

Anonymity

Anonymity systems

Goal is to enable two parties to communicate such that an adversary can't tell who is talking to whom.

- Hide the fact that you talked at all
- Hide conversational partners

Threat models

You may simply want to hide your identity from your conversant, for example if you wanted to visit *amazon.com* and look at a product but you would not want them to know that you looked at it (weak threat model form).

Non-global adversary: the bad guy might be someone like the government. He can monitor a few people's internet connection, but not that many. He cannot see both your connection and Amazon's connection. If he could see packets in both our networks, he can do a timing attack and guess that you talked to Amazon.

Global adversary: where the adversary can monitor all the nodes on the network. So the protocol has to deal with timing attacks. He can also control some nodes on the network, but not all of them.

Attacks on anonymity

Client talks to Amazon. Packets go through the network. Eve is watching. She can record the times at which packets go out from the Client and the times at which Amazon.com receives them. This would give her a good assurance that the client is talking to Amazon.

If Eve controls a node along the path she can introduce **jitter** and look for it on Amazon's side.

Anonymity – hiding your identity. Privacy – hiding facts about yourself.

Anonymity applications: internet browsing, electronic voting protocols (learn the results of the elections without learning what people voted)

Mix nets

Principle: you can be anonymous if you have a crowd of people to be anonymous in.

You have a bunch of people and they send messages to a Mix, which scrambles (permutes) their messages in a random way and then outputs the messages. Scrambling is not enough, since the input and output can be easily related to each other, so encryption is added under the public key of the Mix. The initial message could also be encrypted for the recipient under his key. The mix decrypts and scrambles the messages.

You can have multiple mixes tied together. People would send $E_{mix_1}\left(E_{mix_2}\left(E_{mix_3}(m)\right)\right)$. What if mix1 and mix3 is bad but mix2 is good?

Re-randomizing El Gamal

Good old El Gamal encryption:

$$ctext = g^r, g^{xr}m$$

Alin Tomescu, CSE408
Thursday, May 12th, Lecture #27
You can re-radomize El Gamal by doing:

$$ctext' = \left(g^r g^{r'}, g^{xr'} g^{xr} m\right) = \left(g^{r+r'}, g^{x(r+r')} m\right)$$

The mixes can shuffle and re-randomize if they use El-Gamal PKE.

$mix_i\ has\ g^{x_i}, x_i\ as\ his\ PK$. Together they can compute $PK_{group} = g^{x_1} g^{x_2} g^{x_3} = g^{x_1+x_2+x_3}$

First mix gets $g^r, g^{r(x_1+x_2+x_3)} m$ and he can divide $g^r, \frac{g^{r(x_1+x_2+x_3)} m}{g^{r x_1}} = g^r, g^{r(x_2+x_3)}$ and finally re-randomize. Mix2 and 3 repeat, getting the message m.

To verify that a mix only permuted and re-randomized and decrypted messages you can use ZKPs.

Low-latency anonymity systems

Tor is a system for browsing the web anonymously. It has found its application in helping people behind firewalls and proxies to get access to website. People in Iran used tor to get to Twitter and the Chinese used it for Facebook.

Tor stands for **The Onion Router**. There's a whole bunch of **Tor nodes**, all the clients have a list of them. About 2000 right now. When a client c wants to connect to some website w he picks 3 of these nodes. He constructs a tunnel to the first one using SSL, telling it to connect to the second one using SSL, and this repeats. The client sends:

$$E_{node1}\left(E_{node2}\left(E_{node3}\left(E_{dest}(m)\right)\right)\right)$$

There are entry nodes, middle nodes, exit nodes. Many people don't want to be exit nodes, since you open requests to any website a Tor client wants. So from the outside it could look like you're doing bad stuff.