

Crowds: Anonymity for Web Transactions

The main Idea:

Crowds is a system for protecting users' anonymity on the world-wide-web. Crowds, named for the notion of "blending into a crowd", operates by grouping users into a large and geographically diverse group (crowd) that collectively issues requests on behalf of its members.

The main idea behind Crowds is to allow the user to join a crowd and blend in (passing its message to another member of the crowd-possibly itself again) randomly, which decides if will submit the request to the web server or forward this message to another crowd member that can also submit the request or forward it etc. The request is submitted by a random member so the end server is prevented from identifying message initiator. Also, the initiator of the message cannot be identified and its anonymity is kept against a possible attacker in the network.

Types of Anonymous Communication properties:

- Sender anonymity means that the identity of the party who sent a message is hidden, while its receiver (and the message itself) might not be.
- Receiver anonymity similarly means that the identity of the receiver is hidden.
- Unlinkability of sender and receiver means that though the sender and receiver can each be identified as participating in some communication, they cannot be identified as communicating with each other.

Mechanism:

The membership maintenance procedures of a crowd are those procedures that determine who can join the crowd and when they can join, and that inform members of the crowd membership. Membership in a crowd is controlled and reported to crowd members by a server called the blender. To make use of the blender (and thus the crowd), the user must establish an account with the blender, i.e., an account name and password that the blender stores. When the user starts a jondo, the jondo and the blender use this shared password to authenticate each

other's communication. As a result of that communication (and if the blender accepts the jondo into the crowd), the blender adds the new jondo (i.e., its IP address, port number, and account name) to its list of members, and reports this list back to the jondo. In addition, the blender generates and reports back a list of shared keys, each of which can be used to authenticate another member of the crowd. The blender then sends each key to the other jondo that is intended to share it (encrypted under the account password for that jondo) and informs the other jondo of the new member.

Each member maintains its own list of the crowd membership. This list is initialized to that received from the blender when the jondo joins the crowd, and is updated when the jondo receives notices of new or deleted members from the blender. The jondo can remove jondos from its list of crowd members, if it detects that those jondos have failed.

Does this method provides anonymity?

- If the attacker is a local eavesdropper, then Crowds offers no sender anonymity. The local eavesdropper can view the messages from the sender's computer. The goal here is to provide anonymity for the receiver.
- If the message is forwarded to another jondo, then the local eavesdropper will only see an encrypted message and therefore, it cannot reveal the receiver.
- If the message is submitted directly from the sender to the web server, then the receiver is also revealed, since the attacker will observe the submitted request.

The more members we have in the crowd the higher is the probability of forwarding to another jondo and the therefore the probability to blend into the crowd. If the attacker is an end server, there is no receiver anonymity since, when request is submitted, the receiver (web server) is revealed.

Aims at sender anonymity: Every member of the crowd can be the initiator since the attacker cannot find the path that leads to the initiator due to encryption.

If the attacker is a member of the crowd (collaborating jondo), there is no receiver anonymity because collaborating jondos have access to the message. For a collaborating jondo, the goal is to find the initiator. All

non-collaborating members are equally likely to be the initiator. So increasing the number of jondos in a crowd leads to absolute privacy for sender's anonymity and receiver's anonymity, since the participation of collaborating jondos in the path decreases.

Timing Attacks: if html request contains a URL, user's browser issues another request and the sender is exposed. In order to understand this case, we assume that sender A initiates an html request and uses path $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ where 2, 3, 4 are jondos and 3 is a collaborating jondo. The jondo 4 submits the request and sends the reply back to the path and then requests the URLs and also sends them back along the same path. Sender A receives the reply, loads the page and receives requests of the URLs from its browser. Then immediately requests the URLs using path $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$. The collaborating jondo measures the time from the first request until the URL request. If this time is small, then it can reveal the proximity of sender and therefore expose the sender.

Crowds can deal with timing attacks. If A implements crowds it does not submit requests for the URLs but waits to load the URLs from the replies that the 4 jondo sent.

Static Paths: tends to decrease the anonymity properties provided by the system against collaborating jondos. Collaborators link distinct paths that seem to be initiated from the same jondo.

Static paths used to prevent attacker to have multiple paths to link to the same jondo.

A jondo sets up one path for all its users' communications, and this path is altered only under two circumstances:

- If a jondo in a path fails (or appears to fail), the path remains the same up until the predecessor of that faulty jondo, who reroutes the remainder of the path randomly
- The paths are altered, when new jondos join the crowd

Advantages and Disadvantages

- No receiver anonymity since when request submitted the receiver (web server) is revealed.
- No receiver anonymity since collaborating jondos have access to the message.
- Crowds aims at sender anonymity.
 - Every member of the crowd can be the initiator since the attacker cannot find the path that leads to the initiator due to encryption so the actual initiator's sender anonymity is beyond suspicion.
- Crowds does not provide anonymity against global eavesdroppers. An eavesdropper on the local area network of the user, such as an administrator monitoring web usage at a local firewall. However, if the same LAN also serves the end server, then the eavesdropper is effectively global, and we provide no protections against it. Another approach to achieving anonymous web transactions is to use a **mix**. A mix is actually an enhanced proxy that, in addition to hiding the sender from the receiver, also takes measures to provide sender and receiver unlinkability against a global eavesdropper.
 - (A mix is actually an enhanced proxy that, in addition to hiding the sender from the receiver, also takes measures to provide sender and receiver unlinkability against a global eavesdropper.)
- More difficult to detect are active attacks where crowd members substitute wrong information in response to web requests that they receive from other crowd members.
- If a local eavesdropper and the end server to which the user's request is destined collaborate in an attack, then our techniques achieve neither sender anonymity nor receiver anonymity.
- Firewalls present a problem for Crowds. Like all network servers, jondos are identified by their IP address and port number. Most corporate firewalls do not allow incoming connections on ports other than a few well-known ones.
- Another technology that presents some difficulties is SSL, a protocol by which web pages can be encrypted during transport.
- When using Crowds, it is recommended that Java and ActiveX be disabled in the browser, which can typically be done via a simple preferences menu in the browser.
- Crowds is vulnerable to denial-of-service attacks.