Model validated by measuring probe rate on port 80. Experimental data curve looks very similar to the one from the above equation. The authors were able to choose appropriate values of $K$ and $T$ to fit the experimental data curve. Hourly probe rate at a server would increase as $a$ grows with time – more hosts are infected and they probe more. After some time the growth of $a$ starts saturating as fewer new hosts are infected.

[Reference: http://www.icir.org/vern/papers/cdc-usenix-sec02/]
Anonymity

References:
Basic introduction from “Computer Security” by Matt Bishop
Onion Routing Literature (www.torproject.org)

Hiding who is talking to whom (not just the contents of the message)

An anonymizer – site that hides the origin of connections – acts as a proxy server
- User connects to the anonymizer and tells its destination
- Anonymizer makes the connection to the destination
- Primarily for email and http traffic
- E.g., Finnish anonymizer – email from jdoe@cs.utah.edu to discussion@a.b.com - change source address to anon374@anon.penet.fi and keep a mapping for email address translations
- Problem – keeps the mapping of the anonymous identities and the associated origins
- Copyrighted material got circulated through this site – Finnish court directed the owner of the anonymizer to identify the source for this material – leading to the shutdown of this site.

Anonymity – good or bad?
Anonymity allows whistleblowers considerable protection
- Anonymous sources (one named “Deep Throat”) helped uncover the Watergate scandal.
Anonymity protects privacy
Access to different websites
Other people eavesdropper should not be allowed to find out who is talking to whom

Cons:
Irresponsible behavior -> spam, DoS attacks

Next, we see how anonymity can be provided in the Internet.
- with one intermediate node -> the intermediate node knows both the source and the destination.

To provide better anonymity
- Have multiple intermediate nodes (no one node directly interacts with the source and the destination)
- Do not keep record of the originating address and the next intermediate hop address (so that someone cannot comeback to trace a message to the source)

A ----> Remailer1----> Remailer2----> Bob

{Encrypt with Remailer1’s public key[Encrypt with Remailer2’s public key[Encrypt with \(K_{AB}(\text{Message}), \text{Bob} \}, \text{Remailer1}, \text{Remailer1}]}

[Note that in the above not everything is encrypted using the public keys as shown. Actually, only secret keys are encrypted which in turn encrypt the rest of the message.]

Here, Remailer2 does not know about Alice’s identity, Remailer1 does not know about Bob’s identity.

If the message were not encrypted, even if Remailer1 changes the IP address (src, destn) eavesdroppers sitting at different locations in the network and possibly colluding can match the input email content with the output email content and the time.

Remailers use a FIFO message processing scheduling

This will allow an attacker eavesdropping the input and the output links of the remailer to correlate the order and determine which input message went to which destination address (the outer header cannot be encrypted).

Solution: Randomize the order of processing

What if very few messages arrive?

- Wait for at least n messages, but that would increase delay

Now, message size decreases as the message moves through the intermediaries towards the destination => knowing the exact decrease one could associate inbound and outbound messages. Therefore, append padding or junk.

Another attack

An adversary can eavesdrop and replay large number of copies of a packet and look for a large number of similar looking packets along the output links to associate input and output messages.

Solution: Use sequence numbers to prevent replay attacks.
**Chaum’s Mix:**
- Encrypt everything (outer headers cannot be encrypted)
- Input, output message sizes are same
- Unique numbers to prevent replay attacks
- Wait until enough messages have been received and forward messages to the next destination in an order that is not predictable (tradeoff between waiting and mixing).

**Tor Project**
- An application independent infrastructure for traffic-analysis-resistant and anonymous Internet connections.
- Strong private communications in real time over Internet

**Basic Architecture:**
Routing nodes are roughly real-time Chaum mixes.

Each end point connects to an onion proxy (which is the most trusted node – knows the true source and destination addresses of the connections that it builds and manages) -> converts data streams into application independent fixed size cell formats.

Onion proxy knows
Topology and link state of the network
Public certificates of nodes in the network
Exit policies of nodes in the network

Anonymous connections exist in three phases

Phase 1: Connection setup - begins when the initiator onion proxy creates an onion. An onion is a (recursively) layered data structure that specifies properties of the connection at each point along the route.

crypto control information specific properties
- different symmetric algorithms, keys to be used in the data movement phase
- make encryption simpler, decryption harder (decryption at different nodes)

Phase 2: Data movement
- Data cells -> onions – use symmetric keys – as data cells move, remove layers of onion, finally the receiver onion proxy receives the data, decrypts it and sends to the Bob.
- Onion routers have TCP connections between them.
- Instead of waiting for a long time, synthetic traffic could be added for high bandwidth links to randomize output (instead of FIFO as described above).
- Routes are permitted to have loops of length greater than one hop.
- Routers might follow policies - certain routers might not allow .com traffic or allow only .com or .gov traffic etc. (similar to BGP policies)

Phase 3: Connection Termination