Anonymity - Lecture Notes

Tzachy Reinman  
I.D. 032072886  
Email: reinman@cs.huji.ac.il  

Computer Science and Engineering School  
The Hebrew University, Jerusalem

---

1 These lecture notes are based on the lectures given by Dahlia Malkhi on May 9, 23, 30, 2002 in the “Advanced Course in Computer and Network Security”, at the Hebrew University, Jerusalem. Extensions and additions are based on Web sites and papers that are listed in the bibliography.
<table>
<thead>
<tr>
<th>Content</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>2</td>
</tr>
<tr>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>What is Anonymity?</td>
<td>3</td>
</tr>
<tr>
<td>Motivation</td>
<td>3</td>
</tr>
<tr>
<td>Dining Cryptographers (DC)</td>
<td>7</td>
</tr>
<tr>
<td>Definition</td>
<td>7</td>
</tr>
<tr>
<td>Explanation</td>
<td>7</td>
</tr>
<tr>
<td>Extensions</td>
<td>8</td>
</tr>
<tr>
<td>Practical Considerations</td>
<td>10</td>
</tr>
<tr>
<td>Drawbacks</td>
<td>11</td>
</tr>
<tr>
<td>DC Networks</td>
<td>11</td>
</tr>
<tr>
<td>Anonymizer</td>
<td>13</td>
</tr>
<tr>
<td>Chaum Mixes</td>
<td>14</td>
</tr>
<tr>
<td>Definition</td>
<td>14</td>
</tr>
<tr>
<td>How it works</td>
<td>14</td>
</tr>
<tr>
<td>“Cascade of Mixes” or Chain of Mixes</td>
<td>15</td>
</tr>
<tr>
<td>Reply (“Return Address”)</td>
<td>16</td>
</tr>
<tr>
<td>Drawbacks</td>
<td>17</td>
</tr>
<tr>
<td>Comparison between Mix-net and DC-net</td>
<td>18</td>
</tr>
<tr>
<td>Onion Routing</td>
<td>19</td>
</tr>
<tr>
<td>Definition</td>
<td>19</td>
</tr>
<tr>
<td>How it works</td>
<td>19</td>
</tr>
<tr>
<td>Practical considerations and information</td>
<td>20</td>
</tr>
<tr>
<td>Attacks that Onion Routing is vulnerable to</td>
<td>20</td>
</tr>
<tr>
<td>Comparison between Onion Routing and Mixes</td>
<td>20</td>
</tr>
<tr>
<td>Crowds</td>
<td>21</td>
</tr>
<tr>
<td>Definition</td>
<td>21</td>
</tr>
<tr>
<td>How it works</td>
<td>21</td>
</tr>
<tr>
<td>Security analysis</td>
<td>22</td>
</tr>
<tr>
<td>Encryption</td>
<td>25</td>
</tr>
<tr>
<td>Static vs. dynamic paths</td>
<td>25</td>
</tr>
<tr>
<td>Timing attacks</td>
<td>26</td>
</tr>
<tr>
<td>Scale</td>
<td>27</td>
</tr>
<tr>
<td>Drawbacks</td>
<td>27</td>
</tr>
<tr>
<td>Vision</td>
<td>28</td>
</tr>
<tr>
<td>Comparison between Crowds and Other Protocols</td>
<td>28</td>
</tr>
<tr>
<td>Degradation of Anonymity Methods – The Predecessor Attack</td>
<td>29</td>
</tr>
<tr>
<td>Questions and Answers</td>
<td>32</td>
</tr>
<tr>
<td>References</td>
<td>35</td>
</tr>
</tbody>
</table>
Anonymity

Introduction

What is Anonymity?
Anonymity is a property of network security. An entity in a system has anonymity if no other entity can identify the first entity, nor is there any link back to the first entity that can be used, nor any way to verify that any two anonymous acts are performed by the same entity.

Related term - Pseudonymity: A weaker, related property is pseudonymity. Pseudonymity means that one cannot identify an entity, but it may be possible to prove that two pseudonymous acts were performed by the same entity.

"For example, imagine that you have received a letter in the mail, with no signature, no return address, and no method for you to identify the sender or respond. This letter is anonymous. If the letter contains a secret key, and you then get later letters containing the same secret key, you can be pretty sure they came from the same entity. These latter letters are pseudonymous. If the letter contains instructions for responding, other than by some public channel, and you respond and the writer then responds to you, the writer is now pseudonymous rather than anonymous. This is because you have two (or more) acts (mailing letters) that were performed by the same person” [5].

Motivation

Naively, there is no privacy\(^2\) on the Web. Browsers advertise IP address, domain name, organization, referring page, platform (OS, browser) and which information is requested. The information is available to end servers, local system administrator, and other third parties (see the example doubleclick.com below). Cookies are another violation of privacy.

Example: A typical HTTP request looks like:

GET http://www.amazon.com/ HTTP/1.0
User-Agent: Mozilla/3.01 (X11; I; SunOS 4.1.4 sun4m)
Host: www.amazon.com
Referrer: http://www.alcoholists-anonymous.org/
Accept: image/gif, image/x-xbitmap, image/gpeg, image/pjpeg, */*
Cookie: session-id-time=868867200; session-id=6828-2461327-649945; group_discount_cookie=F

---

\(^2\) Privacy is more general term than anonymity. Privacy can be achieved by disconnectivity, i.e. not being connected to a network.
Anonymity

It can be easily seen that the site that accepts that request knows (without doing anything) the referrer (i.e., the previous site). This is neither privacy nor anonymity! Suppose numerous sites link to doubleclick.com. Due to *Referrer* field, doubleclick may capture your whole click-stream!

![Diagram showing referrer field]

The adversary
There are many kinds of adversaries:

- The receiver or the target-site
- End servers, other users
- Eavesdroppers
  - Global – e.g., ISP, backbone administrator
  - Partial – e.g., in a cable Internet system, all the users use the same channel and can get everyone’s messages (encrypted), so an eavesdropper can perform a traffic analysis of another user.
  - Local – e.g., system administrator
- Active attackers – an individual or a group, local or global, that can cause worse damage than just listening.

Anonymity in the network is relevant to:

- *Electronic voting*

- *E-commerce* – The efficiencies of the public Internet are strong motivation for companies to use it instead of private intranets. However, these companies may want to protect their interests. The existence of inter-company collaboration may be confidential. Private
people are also interested in anonymous e-commerce. A person shopping on the Web may not want his visits tracked.

- **Sending anonymous messages or distributing anonymous content**
- **Other data communications (E-mail, Web browsing, Chatting) – Avoiding traffic analysis**

- **Hiding the existence of a VPN** (Virtual Private Network) between two or more participants

- **Remote Login**

- **Interest group** Examples are: Private health concerns - a person who is an AIDS carrier (and therefore accesses relevant data bases) is interested that this will remain unknown; Support groups of victims of crimes (rape, violence, etc.).

- **PIR – private information retrieval** For example, a researcher using the World Wide Web to access a patents database may expect his particular focus to remain private.

- **Privacy of the communication patterns** (defected by cookies)

---

**Types of anonymity protection**

- **Sender anonymity** – the receiver (and others) cannot know who sends the message.

- **Receiver anonymity** – servers in the message path cannot know to whom the message is designated.

- **Unlinkability of sender and receiver.** Linkability is the possibility to link between different actions in the Internet. For example, if a specific IP address appears in several transactions, then it can be concluded that there is a connection between those transactions.

- **Publisher anonymity** (broadcast).

---

3 *Traffic analysis* can be used to infer who is talking to whom over a public network. For example, in a packet switched network, packets have a header used for routing, and a payload that carries the data. The header, which must be visible to the network (and to observers of the network), reveals the source and destination of the packet. Even if the header were obscured in some way, the packet could still be tracked as it moves through the network. Encrypting the payload is similarly ineffective, because the goal of traffic analysis is to identify who is talking to whom and not (to identify directly) the content of that conversation.
Anonymity

- Information anonymity - For example, a few years ago, a convicted child rapist working as a technician in a Boston hospital rifled through 1,000 computerized records looking for potential victims (and was caught when the father of a nine-year-old girl used caller ID to trace the call back to the hospital).

- Client anonymity (in client-server systems).
Anonymity

**Dining Cryptographers (DC)**

**Definition**
The *Dining Cryptographers* problem and its solution were first introduced by Chaum [2]: “Three cryptographers are sitting down to dinner at their favorite three-star restaurant. Their waiter informs them that arrangements have been made with the maitre d’hôtel for the bill to be paid anonymously. One of the cryptographers might be paying for the dinner, or it may be the NSA (U.S. National Security Agency). The three cryptographers respect each other’s right to make an anonymous payment, but they wonder if NSA is paying.”

We phrase the above in the following way: Given three cryptographers A, B, and C, we want to enable one of them to transmit a bit ‘1’, such that all of them get it, but no one knows who sends it.

To resolve the problem we carry out the following protocol:

We arrange the cryptographers on a circle. Each cryptographer has a two-side coin that only he and the cryptographer in his right can see. They flip the coins. Each cryptographer calculates the XOR of his coin and the coin of his left. The cryptographer who wishes to transmit the bit, actually broadcasts to all (1 minus the XOR), the others broadcast their XOR. If an odd number of 1s were distributed, than a bit ‘1’ was transmitted. For example: A’s coin is 1, B’s coin is 1, C’s coin is 0 (so A sees his coin and B’s coin, etc.). A wants to transmit ‘1’. So A broadcasts 1 - XOR(1,1)=1 – 0 = 1, B broadcasts XOR(1,0) = 1, C broadcasts XOR(0,1) = 1. Three 1s were distributed, i.e. a bit ‘1’ was transmitted.

**Explanation**
1. Correctness – why an odd number of 1s indicates that a bit ‘1’ was transmitted: The total sum of the distributed bits if there is no transmission is: XOR (A, B) + XOR (B, C) + XOR (C, A).
Anonymity

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>XOR (A, B)</th>
<th>XOR (B, C)</th>
<th>XOR(C, A)</th>
<th>Sum of XORs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

It can be easily seen that the sum is always even. If one of the XORs is replaced with its converse (i.e. 1 - XOR), the sum is odd.

2. Anonymity - A knows he is the transmitter (he distributed 1 - XOR), B and C don’t know who is the transmitter: B knows that A distributed 1 and C distributed 1, he also knows that his coin is 1, and that C’s coin is 0, but he doesn’t know what is A’s coin, so he can’t know whether A’s coin is 1 and A is the transmitter (because the XOR of A’s and B’s coins is 0 and A distributed 1) or A’s coin is 0 and C is the transmitter (because the XOR of C’s and A’s coins is 0 and C distributed 1). The same argument goes for C.

Extensions

1. For sending messages longer than one bit, say n bits, we perform the protocol in rounds (say one round per milli-second), i.e. each round performs the protocol for one bit. In other words, each participant should have n coins or n coin-flips.

2. When more than one participant may try to transmit at the same time there is a collision. Collisions are detected by the transmitters themselves, when two participants try to transmit together – the result is an even number of 1s. If all three try to transmit together – it wouldn’t be detected. In case of a collision, a simple solution is to wait randomly and retry.

3. We can extend the number of participants from 3 to k. Each pair shares a secret bit. To transmit ‘0’ a cryptographer broadcasts the sum of his
Anonymity

bits modulo 2; to transmit ‘1’ – the converse is broadcast (i.e. 1 - this sum). In total, each bit is counted twice, so the sum is expected to be even, if there is no transmission; if someone transmits ‘1’, the sum is odd. To transmit a longer message than one bit, each pair of cryptographers shares a chain of secret bits, one bit for each round. (Collision detection is as before: even number of transmissions can be detected, by the transmitters, odd number – cannot be detected).

4. We want to represent the Dining Cryptographers by a graph: Participants are vertices, shared bits are edges. If the graph is not a full graph, than a connected component is anonymized. Collusion is the cooperation of participants who pool their keys in efforts to trace the messages of others. Collusion of k-1 participants (or m-1 in a connected component of m participant) breaks the anonymity. Collusion of fewer participants yields no gain that would help distinguish between different participants (only the parity of the group). Now consider noncomplete graphs. If a full collusion (i.e. each member of the collusion exposes all of his keys to the other collusion members) includes a cut-set of vertices (i.e., one whose removal partitions the graph), the collusion can learn something about the origin of messages originating outside the collusion; the noncolluding vertices are partitioned into disjoint subgraphs [2].

Examples are:

A collusion of 2,3,4 breaks the anonymity. The collusion isolates both 1 and 5, so it can know who transmits.

4 A proof is included in [2].
Anonymity

A collusion of 4,5 partitions the graph into two subgraphs: \{1,2,3,6,7,8\} and \{a,b,c,d,e,f\}. The collusion can know whether the message origined in \{1,2,3,6,7,8\} or in \{a,b,c,d,e,f\} (4,5 know that they are not the origin of the message). If 1 is also in the the collusion, the subgraph \{1,2,3,6,7,8\} is partitioned to \{2,3\} and \{7,8\}. That is because \{4,5\} is a cut-set of the whole graph, and \{1\} is a cut-set of \{1,2,3,6,7,8\}.

Practical Considerations

1. Establishing Keys: For long messages there is a need to provide long keys (i.e. chain of bits). One way to do this is for one member of each pair to toss many coins in advance, or – in other words - to generate a sufficient number of bits by a RNG (random number generator) and burn it on a CD. A second way is to establish a short key and expand it as needed. A third way is to use a Diffie-Helman key exchange [2].

2. Underlying Communication Techniques: The communication topology is independent of the key-sharing graph.

3. For efficiency, communication can be made in blocks (i.e. participants have to group bits into blocks, instead of transmitting bit-by-bit). “In high-capacity broadcast systems, such as those based on coaxial cable, surface radio, or satellites, more efficient use of channel capacity is obtained by grouping a participant's contribution into a block about the size of a single message” [2]. If the block size is one message, contentions protocols may be used: One frame is used to transmit a block. In case of collision, the participant waits a random number of frames and tries to retransmit. If the block size is more than one message, a first block may be used to request a “slot reservation” in a second block. “A simple scheme would be for each anonymous sender to invert one randomly selected bit in the first block for each slot they wish to reserve in the second block. After the result of the first block becomes known, the participant who caused the \textit{i}^{th} 1 bit in the first block sends in the \textit{i}^{th} slot of the second block” [2].

4. Example Key-Sharing Graphs: In a complete graph \(m(m - 1)/2\) keys are required. Sometimes this number is too large. In a circle-topology graph, each entity alone cannot learn anything, but two collaborators can partition the graph, perhaps
Anonymity

compromising an entity between them (e.g. if A’s right neighbor is X and A’s left neighbor is Y, so if X and Y collaborate, they can identify A). Even so, in cases where nearby entities are not likely to maliciously collaborate, circle-topology may be used. “If many people wish to participate in an untraceable communication system, hierarchical arrangements may offer further economy of keys. Consider an example in which a representative from each local fully connected subgraph is also a member of the fully connected central subgraph. The nonrepresentative members of a local subgraph provide the sum of their outputs to their representative. Representatives would then add their own contributions before providing the sum to the central subgraph. Only a local subgraph’s representative, or a collusion of representatives from all other local subgraphs, can recognize messages as coming from the local subgraph. A collusion comprising the representative and all but one nonrepresentative member of a local subgraph is needed for messages to be recognized as coming from the remaining member” [2].

Drawbacks

1. Although malicious participants cannot break the anonymity (unless there is a big-enough collusion), they can block others from transmitting by DoS (denial of service) attacks. In addition, all participants are required to form a transmission.
2. For every bit of information sent, three bits of messages have to be passed all the way around the circle.
3. The last participant to transmit can “read” a message and pass a different message on.

DC Networks

The DC problem is the basis of a kind of network that gives absolute sender anonymity for messages, which is called a DC-net or “Dining Cryptographer’s network”.

In DC-nets, each participant can send messages to all others and none can tell from whom this message is. If a participant wishes to send a message to a specific recipient (only), he can encrypt it in a way that only the intended recipient can decrypt. A DC network allows both the sender and the recipient to remain anonymous [5].
Anonymity
Anonymity

**Anonymizer**

Anonymizer is an entity that intermediates between senders and receivers, and provides anonymous Web browsing. It removes identifying headers and fields. For example, the Anonymizer will “strip out all references to your e-mail address, computer type, and previous page visited before forwarding your request” [1]

Optionally it also encrypts messages.

Anonymizer can be implemented by a proxy or a router. If a group of users trust each other, they can choose one of them to be the anonimizer in turns.

Anonymizer’s main disadvantage is that it has access to the sensitive information. If an adversary takes control over the Anonymizer, he breaks the anonymity, and furthermore, he can control the data transmission.

In addition, the traffic between a browser and the Anonymizer is sent in the clear, so it is vulnerable to eavesdropping, and even if it was encrypted, traffic analysis could be used to match incoming data with outgoing data.

In addition, it anonymizes only the data stream, not the connection itself.

Another disadvantage is that it can add advertisements to pages.

To overcome these disadvantages, we will try to distribute trust in multiple authorities.

We will see that in contrast to Anonymizer, which anonymizes only the data stream, there are solutions that hide also the connection.
Anonymity

Chaum Mixes

Definition
A Mix is a computer that mediates between senders and recipients. It enables anonymous communication by means of cryptography, scrambling the messages, and unifying them (padding to constant size, fixing a constant sending rate by sending dummy messages, etc.). Chaum Mixes support sender anonymity, and protect from traffic analysis.

How it works
To assist reading fluency and convenience, this section is ordered as follows: First we introduce a partial solution, which is simpler than the complete solution. Then we point out the weakness of that solution, and improve it to the complete solution.
Suppose there is a Mix $M$ with private-public keys. Consider participants $A$ and $B$.
$M$ fogs all the communication, to protect from traffic analysis, i.e., sends all messages in the same format, length, and sends also dummy messages, such that every permutation of the incoming/outgoing messages looks approximately the same.
We use the notation $K(X, Y)$ to denote: attach $Y$ to $X$ and encrypt/decrypt it with the key $K$.
Participant $A$ prepares a message for delivery to participant $B$ by appending a random value to the message, sealing it with the addressee's public key $K_B$, appending $B$’s address, and then sealing the result with the Mix’s public key $K_M$.
$A$ sends $K_M(K_B(\text{message, R}^5), \text{B’s address})$ to $M$.
$M$ opens it with his private key, now he knows $B$’s address, and he sends $K_B(\text{message, R})$ to $B$.

---

$^5$ R is a random value.
Anonymity

If an adversary performs a traffic analysis, he sees a packet transmitted from A to M, and then he sees a packet transmitted from M to B. He can conclude that A sent something to B. The middleware M only provides anonymity to A against B (B cannot know who sent him the message).

An eavesdropper, who sees the packet that M sends to B, can concatenate B’s address to it, then encrypt it with M’s public key, and compare it to packets that arrived to M. When he finds an identical packet, he finds the sender (which is A).

The eavesdropper performs:

\[
K_B(message, R) \quad \text{add B’s address} \quad (K_B(message, R), B’s address) \quad \text{encrypt with M’s public key} \quad K_M(K_B(message, R), B’s address).
\]

To overcome the ability of a global eavesdropper to know who sends the message to B, a random number \(R_1\) is added to the message that A sends to M, so it is \(K_M(R_1, K_B(message, R), B)\). \(R_1\), and any other random values that are being added during the protocol, would be removed by the *Mix or Mixes* that the message passes.

**“Cascade of Mixes” or Chain of Mixes**

We do not want to count on a single *Mix*, so we would like to have a chain of *Mixes*. A sender chooses a path of *Mixes*, and encrypts the message in “onion format”: \(K_n \ldots (K_2(K_1(K_B(message, R), R_1), R_2), \ldots R_n)\) that suits to the chain: A ? \(M_n\) ? \(M_{n-1}\) ? \(\ldots M_1\) ? B. The random values are for avoiding a traffic analysis attack in each point in the chain. If the path is not pre-defined (i.e. dynamically changed), the next *Mix* in the chain should be defined (i.e. a *Mix* \(M_x\) (for some \(x\)) must know who is his follower in the chain).
Reply ("Return Address")

So far we saw how to send an anonymous message. But if the recipient doesn’t know who is the sender, how can he reply?

To get a reply, we have to send the address. We send it encrypted, in a way that only the Mix knows to decrypt.

Single Mix:

The sender A sends a return-address: $K_1 (A, S_1)$, $K_X$ where $K_X$ is a public one-time key chosen for the current occasion only, and $S_1$ is a key that will also act as a random string for purposes of sealing. A can send this return address to B as part of a message sent by the techniques already described (see the chart below).

The following indicates how B uses this untraceable return address to form a response to A: B sends $K_1 (A, S_1)$, $K_X (S_0, \text{response})$ to M, and M transforms it to $A, S_1 (K_X (S_0, \text{response})$.

This Mix uses the string of bits $S_1$ that it finds after decrypting the address part $K_1 (S_1, A)$ as a key to re-encrypt the message part $K_X (S_0, \text{response})$. Only the addressee A can decrypt the resulting output because A created both $S_1$ and $K_X$.

---

6 $S_1$ is a random element that will be used as a key in a later phase.
Anonymity

The additional key $K_X$ assures that the Mix cannot see the content of the reply-message.

1. $K_f(R_0, K_B(R, \text{message}, K_f(S_1, A), K_X), B)$
2. $K_B(R, \text{message}, K_f(S_1, A), K_X)$
3. $K_f(S_1, A), K_X(S_0, \text{response})$
4. $S_1(K_X(S_0, \text{response}))$

$K_B = B$’s public key, $K_1 = \text{the mix’s public key}$

Chain of Mixes:
With a cascade of $n$ mixes, the message part is prepared the same as for a single mix, and the address part is as shown in the following:

B sends to the first Mix (‘first’ - from B’s point of view):

$K_1(S_1, K_2(S_2, ... , K_{n-1}(S_{n-1}, K_n(S_n, A))...), K_X(S_0, \text{response}))$.

The first Mix “peels off” the “external layer” from the address part, adds it as an encryption layer to the response part, and sends to the next Mix:

$K_2(S_2, ... , K_{n-1}(S_{n-1}, K_n(S_n, A))...), S_1(K_X(S_0, \text{response}))$.

and the final result of the remaining $n-1$ Mixes is:

$A, S_0(S_{n-1} ... S_2(S_1(K_X(S_0, \text{response})))...)$. 

Note:
1. At each link in the chain the message is being encrypted with some random value $S_i$.
2. We assume that there are no more than $n-1$ malicious Mixes in the chain.

Drawbacks
1. Chaum Mixes requires public-key cryptography, which is computationally expensive and slow for synchronous communication.
2. It is vulnerable to collaboration by first and last Mixes in the chain (timing attacks).
3. There is no sender anonymity from the first Mix or system administrator [4].
Comparison between Mix-net and DC-net

Chaum[2] compares between DC-net and Mix-net, and says that under some trust assumptions and channel limitations, mix-nets can operate where dc-nets cannot. “Suppose that a subset of participants is trusted by every other participant not to collude and that the bandwidth of at least some participants' channels to the trusted subset is incapable of handling the total message traffic. Then Mix-nets may operate quite satisfactorily but DC-nets will be unable to protect fully each participant's untraceability. Mix-nets can also provide recipient untraceability in this communication environment, even though there is insufficient bandwidth for use of the broadcast approach.”

He summarizes: “If optimal protection against collusion is to be provided and the crypto-security of Mix-nets is acceptable, a choice between Mix-nets and DC-nets may depend on the nature of the traffic. With a mail-like system that requires only periodic deliveries, and where the average number of messages per interval is relatively large, Mix-nets may be suitable. When messages must be delivered continually and there is no time for batching large numbers of them, DC-nets appear preferable.”
Anonymity

Onion Routing

Definition

Onion Routing is a practical real-time bi-directional application-independent Mix-based infrastructure for private communication over a public network. It strongly resists traffic analysis, eavesdropping, and other attacks both by outsiders (e.g. Internet routers) and insiders (Onion Routers themselves). It hides the identities of the communication’s parties and its content. The purpose is not to hide the sender and the receiver identities from each other, but to protect their communication from others. In addition, Onion Routing can remove cookies, but it cannot protect from JavaScript, Java applets, and ActiveX if they are enabled by the browser.\(^7\)\(^6\)\(^7\)\(^8\).

How it works

Onion Routing works in the following way: The sender makes a connection to an onion router. That onion router builds a chain of onion routers to the receiver. Each onion router in that chain can only identify his immediate onion router neighbors. To send data, the first onion router creates encryption layers for all onion routers in the chain and adds these layers to the data. This is done by means of an onion. “That is, for an onion route of size \(n\), the key for the last router, \(K_n\), is encrypted using the key for the previous router, \(K_{n-1}\), and so on up to first onion router. This onion is then sent to the first router \(R_1\). Each successive router \(R_i\) decrypts the onion, extracts the key seed information \(K_i\), used for decrypting the information to and from the next hop, as well as the address next hop (or in the case of router \(R_n\), 0 to identify the end of the tunnel)”\(^10\). After adding the encryption layers to the data, it is sent through the onion routers. Each onion router, when receiving the encrypted packet, removes one encryption layer, so the data arrives decrypted to the receiver. As a consequence of the layering, data appears different at each onion router, so it cannot be tracked and onion routers cannot maliciously collaborate. If the connection is broken, each onion router erases the information about the connection that it had\(^6\)\(^7\)\(^10\).

---

\(^7\) Java applets and ActiveX control embedded in a retrieved web page could connect back to its server directly and reveal the users IP address to that server.\(^9\).
Anonymity

Practical considerations and information

Onion Routing uses a public-cryptography for establishing the communication channels and symmetric-cryptography for the communication itself and thus saves costs.

Currently, Onion Routing can be used with: HTTP, FTP, SMTP, rlogin, telnet, NNTP, finger, whois, and raw sockets [8].

A prototype of Onion Routing Network already exists, and a lot of statistic reports can be found at [6].

Attacks that Onion Routing is vulnerable to

Onion Routing does not prevent traffic confirmation. “If an attacker wants to confirm that two end-points often communicate, and he observes that they each connect to an anonymous connection at roughly the same time”, he can infer that they are indeed communicating [7].

Passive internal attacks are possible if at least two onion routers are compromised [7]. Another attack is internal volume attack. Compromised onion routers can keep track of the amount of communication that has passed over any given anonymous connection. They can then broadcast totals to other compromised onion routers. Similarities between totals at the same time in different routers are likely to imply that they belong to the same anonymous connection [7].

Comparison between Onion Routing and Mixes

Onion Routing differs from Mixes in using an indeterminate number of mixes in an indeterminate order, and in at least two other ways [7] [8]:

1. Since Onion Routing is expected to be a real-time system, it is more limited in the extent to which they delay traffic at each node (router).
2. Onion Routers are typically configured to be entry points to the network, “and traffic entering or exiting them may not be visible. This makes it hard to track packets, because they may drop out of the network at any node, and new packets may be introduced at each node” [7].
Anonymity

Crowds

Definition
Crowds is a system for protecting Web browsing privacy. “For example, Crowds prevents a web server that you visit from learning information that could identify you. Crowds operates by grouping users into a large and geographically diverse group (crowd) that collectively issues requests on behalf of its members” [4]. Web servers cannot identify the real origin of a request, because it may equally be any one of the crowd members. Collaborating crowd members cannot distinguish the initiator of request from a crowd member who only forwards the request. As Onion Routing, Crowds cannot protect from JavaScript, Java applets, and ActiveX if they are enabled by the browser [9].

Crowds can be seen as a distributed and chained Anonymizer\(^8\), with encrypted links between crowd members [7] [8].

The Crowds system is implemented, and its code is now available in the US and Canada.

How it works
The basic idea of Crowds is “blending into a crowd”. That means, hiding an individual’s actions within actions of many others. To perform web transactions in the Crowds system, a user should first join a crowd of other users. “Each crowd member is called a jondo, patterned after the ubiquitous generic name, John Doe” [10]. Upon receiving a Web traffic for the first time on a path, a jondo flips a weighted coin\(^9\), and depending on the outcome, continues the path to another randomly chosen jondo or terminates the path and forwards this (and any future traffic in the path) to its ultimate destination. The jondos remember the path the request takes and use it to forward subsequent requests from the same jondo, as well as responses from the destination back to the originating jondo. An advantage of the Crowds system is that as the crowd grows, each jondo is less implicated. Any jondo could have been the initiator, so even

\(^8\) Like Anonymizer, each crowd member strips out identifying headers (From, Referer, User-Agent, Pragma), and cookies (option to include).

\(^9\) This is true for all jondos except the initiator. The first jondo always forward the traffic to a random chosen jondo (possibly itself).
if there is a collaboration of malicious jondos, the existence of at least two “good” ones is enough to cause doubt about the identity of the initiator [8] [9] [10].
This makes communication resistant to local observers [7].
“This technique is rather flexible and light, and is suitable for real time traffic” [10].
The Crowds development team proposes everyone with high-speed connection to the Internet to join the crowd and serve as a crowd member. They even promise prizes to users who keep their Crowds servers alive all the time [4].

Example

It can be seen how a message originated at jondo x is being forward to other jondos (“blended into a crowd”) until it is transferred to its destination. For example, the green path represents a message originated at jondo 3, moved through jondos 1 and 6, and reaches destination no. 3.

**Security analysis** [9]
As we said above, when a jondo gets a web traffic, he flips a weighted coin, and forward the traffic to another jondo or to the destination. We will denote the probability of forwarding to another jondo by $p > \frac{1}{2}$. 
Anonymity

We define \( n \) as the number of jondos (we treat this as a static), \( c \) as the number of collaborating jondos, and the following degrees of anonymity:

- **Absolute privacy**: adversary cannot observe communication.
- **Beyond suspicion**: no user is more suspicious than any other.
- **Probable innocence**: each user is more likely innocent than not.
- **Possible innocence**: nontrivial probability that user is innocent.
- **Exposed** (default on web): adversary learns responsible user.
- **Provably exposed**: adversary can prove your actions to others.

**Crowds** achieves:

<table>
<thead>
<tr>
<th>Attacker</th>
<th>sender anonymity</th>
<th>receiver anonymity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local eavesdropper(^{10})</td>
<td>Exposed</td>
<td>P (beyond suspicion) ? 1</td>
</tr>
<tr>
<td>c collaborating jondos, where</td>
<td>probable innocence</td>
<td>P (absolute privacy) ? 1</td>
</tr>
<tr>
<td>( n \geq p_f^*(c+1) / (p_f^{-1/2}) )</td>
<td></td>
<td>as ( n \leq 8 )</td>
</tr>
<tr>
<td>end server</td>
<td>beyond suspicion</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Local eavesdropper**

Since nothing is done to hide from the eavesdropper correlations between input to and output from the initiator, there is no sender anonymity against local eavesdropper.

Due to encryption, a local eavesdropper is able to identify the receiver only if he is the last jondo. Since the probability that this is the case is \( 1/n \), the probability that he identify the receiver decreases as a function of \( n \).

**End servers**

The end server is the receiver, so receiver anonymity is obviously not possible. Sender anonymity is strong: Since the initiator always first forwards to a jondo, the end server is equally likely to receive the request from any jondo. So, the sender is beyond suspicion, and it does not depend on the path length or on \( p_f \).

---

\(^{10}\)“A local eavesdropper is an attacker who can observe all (and only) communication to and from the user’s (initiator’s) computer” [9]. It may be the LAN administrator monitoring web usage at a local firewall.
Collaborating jondos

To prove that Crowds provides “probable innocence” anonymity we define:

- $H_k$ = first collaborator is at position $k = 1$, where the initiator itself occupies the $0^{th}$ position (and possibly others).
- $H_{k+}$ = first collaborator is at position $k$.
- $I$ = first collaborator is immediately preceded by the initiator.

Note that $H_1 ? I$, but the converse is not true, because the initiator might appear on the path multiple times.

We say that path initiator has **probable innocence** if $P(I | H_{1+}) = \frac{1}{2}$.

We now prove that Crowds provides “probable innocence” anonymity:

$$P(H_i) = \left(p_f \frac{(n-c)}{n}\right)^{i-1} \left(\frac{c}{n}\right)$$

This is (prob. forwarding * prob. of noncollaborator $)^{i-1} \times$ prob of collab.

So, we get

$$P(H_{1+}) = \left(\frac{c}{n}\right) \sum_{k=0}^{8} \left(p_f \frac{(n-c)}{n}\right)^k = \left(\frac{c}{n}\right) \frac{1}{1 - p_f(n-c)/n} = \frac{c}{n - p_f(n-c)}$$

$$P(H_{2+}) = \left(\frac{c}{n}\right) \sum_{k=1}^{8} \left(p_f \frac{(n-c)}{n}\right)^k = \left(\frac{c}{n}\right) \frac{p_f(n-c)}{1 - p_f(n-c)/n} = p_f(n-c) \frac{c}{n - p_f(n-c)n}$$

$$P(H_1) = \frac{c}{n}$$

$$P(I | H_1) = 1$$

$$P(I | H_{2+}) = (n-c)^{-1}$$

($I | H_{2+}$ means collaborators get from initiator on hop = 2)

Now, $P(I)$ can be captured as

$$P(I) = P(H_1) P(I | H_1) + P(H_{2+}) P(I | H_{2+}) = c(n - np_f - cp_f) / (n^2 - p_f(n-c)n)$$

Given that a collaborator is on the path ($H_{1+}$), what is the probability that the initiator is the first collaborator’s immediate predecessor ($P(I | H_{1+})$)?

Since $I ? H_{1+}$, we get

$$P(I | H_{1+}) = P(I n H_1) / P(H_{1+}) = P(I) / P(H_{1+}) = (n - p_f(n-c-1))/n$$

So, if $n = p_f (c+1) / (p_f - \frac{1}{2})$, then $P(I | H_{1+}) = \frac{1}{2}$, i.e. the initiator has probable innocence against $c$ collaborators.

E.g. take $p_f = 3/4$, then $n = 3(c+1)$ ensures probable innocence.
Anonymity

So, we can tolerate having almost 1/3 of the jondos in the crowd collaborating for this value of \( p \). More generally, there is a tradeoff between the length of paths (i.e. performance) and ability to tolerate collaborators.

A consequence from the above is that if \( c \) and \( p \) are constants, then as \( n \) grows \( P(H_{1+}) \) get smaller, or: \( P(H_{1+}) \rightarrow 0 \) as \( n \rightarrow \infty \). Assuming that collaborators cannot observe a path on which they occupy no position (\( P(H_{1+}) = 0 \)), then \( P(\text{absolute privacy}) \rightarrow 1 \) as \( n \rightarrow \infty \) for sender and receiver anonymity.

If \( c = n-1 \) (i.e. all jondos except a single one are collaborating), then \( P(I \mid H_{1+}) = (n-pf(n-c-1))/n = n/n = 1 \), i.e. the probability that the initiator is exposed is 1 (100%), unless the path is the single non-collaborator jondo (it can appear many times in this single-node-path, by choosing itself as the next jondo).

Encryption

Crowds uses encryption to defend against local eavesdropper in the following way:

Path key, which is shared among all jondos on a path (it is created by initiator and forwarded along the path), used to encrypt request and replies. It is re-encrypted with pairwise keys on each hop. Crowds uses fast stream cipher (build by requester or submitter while waiting for reply) to encrypt replies [4].

Static vs. dynamic paths

The Crowds developers wanted to use dynamic paths because of performance considerations. But, they found out that it tends to decrease anonymity properties\(^{11}\). Collaborating jondos can link different paths (initiated by the same unknown jondo) based on content or timing of communication on path. Such multiple linked paths could compromise anonymity. The jondo from which paths are most often received is, with high probability, the initiator. Recalling the “probable innocence” proof, we saw that given that a collaborator is on the path, the probability that the initiator is the first collaborator’s immediate predecessor is \( P(I \mid H_{1+}) = (n-pf(n-c-1))/n < 1/2 \). But if we identify \( k \) paths from the same (unknown) initiator, than the expected number of paths with this property is \( k^*(n-pf(n-c-1))/n \). In all other paths, the first collaborator’s

---

\(^{11}\) Later in these notes we will introduce a similar, more general attack, called “the predecessor attack”.

25
Anonymity

Immediate predecessor is not the initiator, but one of the other n-1 jondos. So the initiator appears as the first collaborator’s immediate predecessor more than any other jondo. To prevent this, we make the paths static. With static paths, the first collaborator’s immediate predecessor is always the same jondo (because the path has not been changed). “Static paths” means that paths are rerouted only under two circumstances: First, when a jondo fails (path is rerouted as to subvert the failure), and second, when a new jondo joins (to protect the anonymity of the joining jondo). To avoid multiple linked paths, we do not perform reroute for every new jondo. Instead, we batch new jondos joins together so they occur in one scheduled event, called “join commit”. At join commit all static paths are rerouted once. All users are notified about the join commit and the path rerouting, and alerted from continuing to browse content related to what they were browsing prior to the commit.

Timing attacks

Crowds is mainly used for web browsing. “The possibility of timing attacks in Crowds results from the structure of HTML. An HTML page can include a URL (e.g. an address of an image) that, when the page is retrieved, causes the user’s browser to automatically issue another request. The first collaborating jondo on a path can time the duration until it receives the request for that URL. If the duration is sufficiently short, then this could reveal that the collaborator’s immediate predecessor is the initiator” [9].

To eliminate timing attacks:
1. Submitting jondo (last on the path) parses the HTML page to identify all URLs that the browser will automatically request and sends them along with page.
2. Requesting jondo, upon receiving requests for these URLs from the browser, does not forward them, but simply waits for the URL’s content to arrive, and then feeds it to the browser. This way, other jondos never see these requests [9].

---

12 These URL are contained in, for example, the src attributes of embed, frame, iframe, img, input type=image, and script tags, the background attributes of body, table, tr, and td tags, the content attributes of meta tags, and others [9].
13 Note that this is not possible with SSL.
Anonymity

Scale
Expected number of appearances of a jondo on all paths at any point in time is:
O \{ (1- p_f)^2 (1+n^{-1}) \}. This means that Crowds scales well as crowd size grows.

Expected path length is (1- p_f)^{-1}

Drawbacks
1. If an initiator was revealed once, it can be recognized at each time due to the use of static paths.
2. The Crowds technique “requires a centralized jondo server (called a “blender”) to advertise the location of jondos to jondos wishing to form jondo paths” [10]. This “blender” is a trusted party for the purpose of key distribution and membership reporting [9].
3. Crowds does not protect against internal denial-of-service attacks. A jondo could, e.g. refuse to pass messages [9].
4. Crowds cannot work well with firewalls. A firewall prevents a jondo outside the firewall from connecting to another behind the firewall, insiders can still participate, with weaker anonymity [9].
5. Each jondo might be the one who actually sends the request to its destination. Thus, a jondo might be accused of sending a request that he didn’t initiate. This jondo would have to explain that he is participating in a crowd and thus he could not control what requests were submitted from his machine [4] [9].
6. “By routing your requests through others' machines, Crowds may increase the risk of disclosure of the data in the request and corresponding reply. That is, while other jondos are not able to determine who originated a given request, the contents of the request and reply may be exposed to them. This is primarily a concern when, e.g., passwords for accessing web pages are included in this content”. In such cases, the Crowds mechanism is not recommended [4].
7. Crowds protects only from internal attacks (malicious jondos and destinations), not from global eavesdropping.
8. Any argument against anonymity, e.g. target advertising.

---

14 A proof can be found at [9] in section 7.
Anonymity

Vision
The vision is a wide-scale adoption of Crowds on the Internet – If indeed, some of the drawbacks will vanish, e.g. firewalls will probably allow jondos to communicate. This will cause that no more correlation will be between a web request and the machine that made the request, and will reduce the amount of targeted advertising. Generally, more user privacy is expected\(^6\).

Comparison between Crowds and Other Protocols

Chaum Mixes \([9]\)
1. Mixes ensures sender and receiver unlinkability, but does not provide sender anonymity (e.g. if a group of Mixes collaborate).
2. Mixes protects from a global eavesdropper, Crowds does not. However, Crowds’ intention is for a crowd to span multiple administrative domains, where the existence of a global eavesdropper is unlikely.
3. Mixes typically rely on public-key cryptography.
4. In Mixes, the length of the message grows as the path length grows, and more keys are needed, so long paths cause a large overhead. In Crowds, the length of a path does not influence the length of a message.

Onion Routing \([8]\)
Crowds is less general than Onion Routing, both in its applications (it is designed only for Web traffic) and its anonymity goals (there is no attempt to hide the ultimate destination of traffic from any node on the path).

---

\(^{15}\) This is due to the fact that most firewalls do not allow incoming connections in ports other than a few well-known ones.

\(^{16}\) An extensive comprehensive review of privacy today including advises can be found at the Time magazine - \[http://www.time.com/time/reports/privacy\] (choose “Cover Story).
Degradation of Anonymity Methods – The Predecessor Attack

Wright et al. [11] proved that when a particular initiator continues communication with a particular responder across path reformations, there are attacks that degrade the anonymity over time that protocols like Crowds, Onion Routing, DC-Net and more, are subject to. They showed that fully connected DC-Net is the most resilient to these attacks, though it is the most expensive protocol and subject to simple and anonymous denial-of-service attacks.

We will discuss such an attack from Crowds perspective, and later we will refer shortly to other protocols.

As written earlier in this document, Crowds keeps the path from the initiator (static paths), for a limited period of time, after which paths are reformed.

The predecessor attack works by counting the number of rounds (i.e. path reformation) in which each node sends a message that is part of an identifiable stream of communications through an attacker. The attack exploits the process of path initialization.

We define:

\( I \) = initiator of a connection.
\( R \) = responder to a connection (the intended receiver). (\( R \) is not a participant in the protocol).

Session = continuing communications between an initiator and a responder.
Sender = a node that sends a packet directly to another node or to the responder
Receiver = a node accepting packets from other nodes

The receiver of any packet can determine the identity of the sender and the sender of a packet knows the identity of the receiver.

\( A \) = the active set of participants used by \( I \) to forward \( I \)'s messages. \( I \) is always in \( A \).
\( ? \) = total ordering of the nodes in \( A \).
Anonymity

We assume that $I$ is selected uniformly random over all nodes in $A$, except $I$. $I$ may participate as a regular node (as addition to its part as the initiator), in such case $I$ may be a part of the uniformly random selected $I$.

**Round** = a period in a session, on which $\pi$ is constant, i.e. the period between path reformations.

$A_{min}$ = the minimum over all active sets $A$ and total orders $\pi$, of the number of attackers required to identify a partial group that includes $I$. In *Crowds*, $A_{min} = 1$, since it is sufficient for an attacker to be the first participant on the path. In DC-Net, $A_{min}$ = subgroup that diverse the graph\(^1\), so in DC-Ring, $A_{min} = 2$.

Assumptions:
1. For simplicity, only one initiator $I$ communicates with $R$.
2. A round has a short, bounded duration. (This is reasonable since attackers can leave and join the crowd, and cause resets of $A$ and $\pi$)
3. There are many rounds during a session.
4. In each round there is a participant whose job is to send the message to $R$, and he is an attacker.

Let $G(n,c,T)$ be the probability of correctly guessing the initiator after $T$ rounds with $n$ participants and $c = A_{min}$ attackers working cooperatively.

We argue that for any $1 > \epsilon > 0$ exists $T$ such that $G(n,c,T) > \epsilon$. This means that there is no constant threshold that the probability of exposure is lower from for a long time.

Proof:

We run the system for $T$ rounds. In each round, a configuration on which $I$ exposed happens with probability $d > 0$ (due to the fact that $c = A_{min}$).

In $dT$ rounds, $I$ is suspected. In the rest $(1-d)T$ rounds the suspicions are divided between the rest of the participants, i.e. in $(1-d)T / (n-c-1)$ rounds each of the other participants is suspected.

In *Crowds*, we partition the cases on which there is an attacker on the path in the following way:

\(^1\) Obviously, in a fully connected DC-Net $A_{min} = n-1$, so it is not subject to the attack.
### Anonymity

<table>
<thead>
<tr>
<th></th>
<th>$P(H_1)$</th>
<th>$P(H_{2+})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>The probability of I to appear</td>
<td>$c/n$</td>
<td>$(n-c)^{-1}P(H_{2+})$</td>
</tr>
<tr>
<td>The probability of any other suspect to appear</td>
<td>0</td>
<td>$(n-c)^{-1}P(H_{2+})$</td>
</tr>
</tbody>
</table>

In total, the probability of I to appear is larger than the probability of any other suspect ($c/n > (n-c)^{-1}P(H_{2+})$), so in the end, I will be exposed, and actually a single attacker is enough for this.

In DC-ring:
At each round the ring is divided to two subgroups.
The probability of I to be part of the suspected subgroup = 1.
The probability of any other participant = $\frac{1}{2}$.
Again, after a time (in average $T(n \log n)$), I is exposed.
Generally, this is true for any topology that is not a full graph.

Here is an illustration of the predecessor attack on DC-ring:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Diagram" /></td>
<td><img src="image2.png" alt="Diagram" /></td>
<td><img src="image3.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

A and B are collaborating attackers. A is constant. B leaves and joins, and decreases the suspected subgroup, until there is only one node between B and A. This node is the initiator.
The same goes for Mixes, assuming that the last Mix is an attacker.
Anonymity

Questions and Answers
From last year (2001) exam:

Dining Cryptographers

1. In Chaum's "Dining Cryptographers" algorithm for communication, when two cryptographers wish to transmit simultaneously they "collide", and:
   
   (a) all cryptographers detect the collision
   
   (b) one of the transmitting cryptographers detects the collision
   
   (c) the two transmitting cryptographers detect the collision

   **A: the two transmitting cryptographers detect the collision**

   They both detect that their transmissions were "cancelled", but to everyone else, this is indistinguishable from the case that no one transmits anything. When two cryptographers try to transmit, they both broadcast the inverse of what each of them would broadcast if she wouldn't try to transmit. So, the number of 1s is even, as it would be if no one were trying to transmit. All the rest of participants interpret this as no transmission, but the two cryptographers who tried to transmit, know that this is not the situation, and conclude that there is a collision (See the paragraph “Extensions”).

Mixes and Crowds

2. Chaum "mixes" and Crowds provide anonymity. Which of the following holds:

   a. Every attack that mixes protects from, Crowds also protects from, but not vice versa.

   b. Every attack that Crowds protects from, mixes also protects from, but not vice versa.
Anonymity

c. Every attack that Crowds protects from, mixes also protects from and vice versa.

d. Some attacks are protected only by Crowds, and some only by mixes.

A: **Some attacks are protected only by Crowds, and some only by mixes.**

Mixes protect against global network eavesdropping, Crowds does not. Crowds provides sender anonymity of some degree, mixes does not. (See the paragraph “Comparison between Crowds and Other Protocols”). Indeed, both Crowds and Chaum Mixes are vulnerable to the predecessor attack.

3. In Crowds, the paths from a sender to different destinations (web servers) are:

(a) different (dynamic), to prevent linking them all to the same sender

(b) same (static), to prevent linking multiple paths from the same sender

A: **static, to prevent linking multiple paths from the same sender**

Otherwise, multiple paths of the same sender will have the original sender immediately precede collaborating members more frequently than other members. (See the paragraph “Static vs. dynamic paths”). For better understanding of the multiple linked path problem, we quote the original paper of Crowds [9]: “Note that if the collaborators identify paths $P_1... P_k$ from the same (unknown) initiator, then the expected number of paths on which the first collaborator is directly preceded by the path initiator is $\mu = k(n-p(n-c-1))/n$. By Chernoff bounds, the probability that the first collaborator is immediately preceded by the initiator on substantially fewer of these paths is small: the first collaborator is immediately preceded by the path initiator on fewer than $(1-d)\mu$ paths with probability only $e^{-\mu(\delta^2)/2}$. Thus, the initiator would be identified with high probability”.

4. In Crowds, the path from a sender to the a particular destination (web server) is:
Anonymity

(a) dynamic, to prevent linking multiple transmissions to the same sender

(b) static, to prevent linking multiple paths from the same sender

A: static, to prevent linking multiple paths from the same sender

See the previous answer. The fact that in this question the destination is single and in the previous question there were different destinations does not matter.
References


