Abstract— Delay and disruption tolerant networks are characterized by long periods of disconnection during which the nodes lie in small separated islands consisting of one or several nodes. While seen as an opportunity to bring communication to far flung areas where permanent connectivity is not possible or is infeasible, at the same time, DTN involves great risks and challenges. These risks and challenges have given rise to information security concerns in DTN. While several methods have been proposed to answer these concerns, these solutions still remain under-specified and far from being standardized.

This paper builds on an understanding of the security related challenges faced in DTN environments. It then provides a concise discussion and comparison of the different proposed solutions for implementing secure channels of communication in DTN environments and finally it concludes by suggesting some possible ideas that might play a note worthy role in designing future DTN security mechanisms.

Index Terms—Delay Tolerant Networks; Information Security

I. INTRODUCTION

Delay or disruption tolerant networks provide a useful way for efficient communication between far flung areas which do not have permanent connectivity with the communications infrastructure. Varying techniques are used in order to carry messages or bundles from such isolated locations. Most of these techniques employ long periods during which the messages or bundles are being transported by third party nodes. This presents a great loophole for potential adversaries to sniff these messages and use them against the communicating parties. This is a security concern where normal encryption and security mechanisms tend to falter. Due to the long delays which are characteristic of DTN environments, security handshakes between source and destination parties are rendered infeasible and impractical.

Consider a DTN scenario where two nodes that have no direct link in space-time and have had no prior contact have to communicate, as shown in Fig 1. In the simplest of cases, node A will use an intermediate node B as a messenger to carry it’s message to the destination D. While node B might not be connected to both A and D at the same time, but by enabling A’s message to stay in B’s memory for longer, it can deliver the message over to D at some later point in time when it does come in close proximity of node D.

Fig 1. Sample DTN network

II. PROBLEM STATEMENT

Continuing from the sample DTN network from Fig 1, consider the case when, the message that A wants to send to D contains confidential information and is not expected to land in the hands of an adversary. How is A going to make sure that the intermediate node B does not illegally read or misuse the contents of the message. The first idea that springs to mind is that of encryption. How about encrypting the message with a secret key before handing it over to the messenger node. While this scheme will work perfectly well in a non-DTN environment, where nodes A and D share some secret key (or know each other’s valid public keys) prior to communication, but its usage is hampered in our case since A and D have had no prior exchange of keys.

A keen reader would think at this point as to why not use some key exchange mechanism, say Diffe Helman to exchange the keys. The problem with DTNs is that every round trip of
message exchange between the source and destination has a huge cost both in terms of financial resources and time. Mechanisms like Diffe Helman require a key handshake between the communicating parties, and in DTN this might take days assuming that our messenger node only completes a trip a day. This would not only make the communication delayed but would render it practically use-less.

A second option would be to resort to public key encryption techniques like PKI. The problem with such mechanisms is that both or one of the communicating parties has to have a contact with the trusted third party or Certificate Authority (CA), either to fetch the other party’s public key certificate or to verify that it’s not revoked. Once again the limitations of DTN render this infeasible since a connection with CA might not be available at all times.

This opens ground for the discussion that follows. This paper attempts to address the question as to “how can two mutually disconnected parties exchange confidential information without having any previously established secure channel”.

But this is not all. Even if we do manage to exchange information over a secure channel, our communicating peers would also like to maintain their anonymity i.e. an intersecting party might not be able to judge to a certain degree of accuracy as to who is the source and intended destination of the message. Such attacks on anonymity are more conspicuous in social networks, and its importance in opportunistic networks can not be neglected since at some point, messages do get forwarded through links consisting of unknown peers. Anonymity guarantees are a must in such a scenario.

III. PROPOSED SOLUTIONS

Various possible secure communication models to work in delayed or no connectivity have been proposed. We will now briefly touch most of these.

A. Key boot strapping:

To start with, we see that [1] presents the idea of boot strapping DTN nodes with keys. This way the keys are pre-distributed among adjacently placed nodes so they can exchange data securely using these keys. However this method tends to show less flexibility when it comes to a network, where nodes join and leave dynamically.

B. Identity Based Encryption:

Majority of the proposed security mechanisms for delay tolerant networks implement identity based cryptography (IDE) [2] and claim it to be a reasonable solution for DTN environments. At first the system does look workable since the need to fetch and/or verify public key from the remote party has been dealt with, by using the well known identities such as names or email addresses as public keys. However it has been pointed out, that the identity based system does have its drawbacks [3]. To counter the revocation problem, IDE implements dynamic update of public keys. The idea is to append time stamps at the end of the well known identity. After a certain time lapse, a new public key is formulated. But this scheme has its limitations. For every newly generated public key, a new private key must be sought from the PKG. This makes it equivalent to replacing the CA with PKG and the problem of a constant connection to the CA effectively remains unaltered, since the connectivity with a PKG is also improbable just like a CA.

A further extension of the identity based encryption system for DTN has been proposed in [4] which employs Hierarchical Identity Based Cryptography.

A noteworthy comparison between application of traditional cryptographic techniques and identity based cryptography in DTN environments has been given in [5] which suggests that IBC has just about the same performance in DTN as tradition cryptography when it comes to authentication of communicating peers but IBC does hold promise when considered for use in end-to-end confidentiality of messages.

C. Use of Social Contacts:

Another solution as pointed out in [3], includes the opportunistic use of ones social contacts as intermediaries to exchange messages. The intermediaries are such that they have had previous communication with the source as well as the destination, that is to say, that they are well known to the source and destination, sometimes referred to as “friends” and thus have a security context already established (secret or public keys exchanged) with both the source and destination. The source sends an encrypted message to the intermediate node, which decrypts it using the already known secret key of the source node, and then later when it comes in close proximity of the destination node, it encrypts the message using the secret key of the destination node and forwards it. This method of course is not in any way limited to the use of just one intermediary step. The message can go through a number of hops, before reaching the destination where at each hop, the message gets decrypted using previous hop node’s key and then encrypted with the key of the next hop node.

### TABLE I
**TERMINOLOGIES USED IN THE PAPER**

<table>
<thead>
<tr>
<th>Term</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intermediary or</td>
<td>A node that carries the message from</td>
</tr>
<tr>
<td>Intermediate Node</td>
<td>source to destination.</td>
</tr>
<tr>
<td>Messenger Node</td>
<td>Same as intermediate node.</td>
</tr>
<tr>
<td>Xor-key</td>
<td>A key that is xor-ed with the contents of the</td>
</tr>
<tr>
<td></td>
<td>message to add confusion to the message thus</td>
</tr>
<tr>
<td></td>
<td>preventing any third party to read the</td>
</tr>
<tr>
<td></td>
<td>contents. A message might be xor-ed with n</td>
</tr>
<tr>
<td></td>
<td>number of keys one by one, thus forming an</td>
</tr>
<tr>
<td></td>
<td>onion skin structure.</td>
</tr>
<tr>
<td>Enc()</td>
<td>The encryption function between two</td>
</tr>
<tr>
<td></td>
<td>adjacent nodes that have prior</td>
</tr>
<tr>
<td></td>
<td>established secure context. It can either be</td>
</tr>
<tr>
<td></td>
<td>public key encryption or secret key encryption.</td>
</tr>
</tbody>
</table>

One might argue that if the message goes through n hops, this means that the original message goes through n+1 encryptions (once at the source and n times at the intermediate nodes) and n+1 decryptions (n times at the intermediate nodes and once at the destination). This not only adds processing-
load to the intermediate nodes but also might be time consuming to encrypt and decrypt a message so many times. However, time is measured on very large scales in DTN environments. Since disconnection is a feature that is innate in DTN, a packet might just have to wait for hours or days before being forwarded. Under such circumstances, the few seconds spent during encryption/decryption do not add much overhead.

A closer look however reveals a security hole in this scheme. The message itself is quite visible at every step to the node which decrypts it and then re-encrypts it. What if the intermediate node is an adversary masquerading at being someone “friendly”. To counter this possible security breach, an onion-skinning scheme is proposed in [3], where we can XOR the message with m different keys from $K_1$ to $K_m$ and then send the message through m different routes, and on each route, send one of the keys in the header. The XOR operation is in effect adding “confusion” to the message. Unless a party has all the m keys, it can not read the message. Or in other words, each of the intermediate nodes, that receives one of the messages can correctly decrypt the message contents with the probability of 1/m. It is also quite evident from this scheme that the security of the whole system is directly dependent upon the number m. Greater m will result in more robust security.

Fig 2. Simple message routing through m social contacts

The risk in this scheme is the probability that the m routes, pass through a single node at some point in their paths. That single node would then have all the keys and can effectively hamper the security. Hence we must identify some method by which we can guarantee that one intermediate node, gets one and only one of the m messages. See Section-IV for more discussion on this topic.

Further still our scheme is hampered by the so-called “collusion attack” where different intermediate nodes might collude with each other, secretly exchanging messages and thus are able to bring down the secrecy.

Consider two intermediate routes being used. $C_1$ and $C_2$ are the two encrypted messages that will flow over route one and route two respectively.

Where:

\[ c_1 = [\text{Enc}(K_1) \mid \text{Msg (XOR) } K_1 (\text{XOR) } K_2] \]

\[ c_2 = [\text{Enc}(K_2) \mid \text{Msg (XOR) } K_1 (\text{XOR) } K_2] \]

If the intermediate nodes exchange these packets after decrypting the xor-keys (K1 & K2 here) with their respective secret-keys, effectively both intermediate nodes can learn the contents of Msg.

Another problem associated with such schemes is that in order to deliver one message, the source sends out m copies of it using m routes. However, in a DTN scenario, the opportunities to transmit messages are few and far between. If for sending one bundle, we send out m copies, we are severely damaging the efficiency of our system. Worse still, if the message fails to reach the destination through any one of the paths, the rest of the m-1 messages that do reach will be useless. This problem is magnified when we see it in the context of DTN where the message delivery guarantees are very meager. And even if all the m messages get delivered, the whole system operates at the effective speed at which the slowest route manages to deliver its message. So, the chain is only as strong as its weakest link.

A very straightforward answer to this shortcoming is to add redundancy in the system. Forward the same packets (with same xor-keys in their headers) on two or three different paths. This way the reliability that the message does reach the destination multiplies, but the security guarantees fall by an equal amount. Since adding redundancy in effect increases the chances of all the keys falling in the hands of one single node. This inverse relationship between security and delivery reliability can be represented in the graph shown in Fig 3.

Fig 3. Relationship between security and reliability of communication

D. Ensuring Anonymity:

The solution proposed in [6] and as discussed in the preceding section works on using ones social contacts as intermediaries who can carry messages to other nodes. This
however brings to light the anonymity problem. At some point, along the route, the messages might get passed to a tier of friends that ideally should not be able to figure out the identity of the originator of the message. Thus protecting the privacy is a concern which needs to be looked into.

One of the most common attacks in such communication scenarios, which employ message routing through a network of friends and their friends, is the intersection attack [6]. Briefly described; if in a network of so called friend nodes, a minimum of two nodes decide to collude, that is, they have a secret agreement between themselves in which they share their received content together, then these two nodes can identify the originator of a packet marked ‘anonymous’ to a certain accuracy if both of them have received the same packet and both of them have one and only one common friend node in their connections. It is safe to assume that if both the colluding nodes receive exactly the same message, it must have come from the single common node to which they both are connected. This type of intersection attack can be planned and performed at a much larger level, where numerous nodes distributed strategically in the network collude together and act as distributed traffic collection points. These nodes can then pass information on the received traffic to a central node, where computer algorithms can be used to correlate the traffic contents from all the distributed collection points and identify the originators of various traffic streams to a certain extent.

A solution suggested to this problem is proposed in [ ] which is referred to as k-anonymity. This means that k nodes decide to build an imaginary clique around themselves. The clique is built by making sure that if some number of outside nodes have a node A as a common friend, then they also have B, C, … K nodes as their common friends. This way if these outside nodes decide to collude and find out that they have received the same message, by looking at their friends, they can at best only identify that the message must have come from one of the K nodes. The accuracy of identification does not go beyond the virtual clique’s boundary.

IV. PROPOSED IMPROVEMENTS

Now I delve into a detail of how we can improve the confidentiality guarantees in a DTN environment. My proposed improvements build on the scheme discussed in the preceding section of this paper which employs using social contacts for providing confidentiality in delay tolerant networks. One of the limitations that was pointed out in this scheme was that if all or majority of the messages get routed through a single intermediate node, this node would then have access to all the m xor-keys used to encrypt the message contents, and thus this node can effectively decrypt the message and confidentiality is breached.

Here we suggest a scheme to ensure that a received message has indeed been able to preserve its confidentiality and has not been read by some intermediate party on the route.

A. Record Routing:

To make sure that a number of encrypted packets amongst all the m packets C1, C2, … Cm don’t get routed through the same node we can use a methodology that is generally referred to as record-routing. Every packet records or stores some identifier (MAC address, email address, signature) of the current intermediate node inside its header before it leaves it and gets forwarded to the next node. In this way if a packet has gone through nodes say; L, O, R, and T, it will have these identifiers in its header when it reaches the destination. Upon receipt of all the m packets belonging to the same message from source, the destination node can quickly correlate the recorded routes of all the m packets and check whether a certain node has had more than n packets passing through it. This number n should ideally be not more than 1, but in a less ideal scenario, a value that is considerably less than m can be acceptable. When the value of n for a certain node reaches m, the destination can be certain that the confidentiality has been breached. And certain counter measures can then be taken to avoid further loss of confidential information.

Certain limitations are inherent in the above described methodology. For instance if a malicious intermediate node over-writes the record route field in the header, thus removing any trace that the packet ever passed through it. However this can be countered in the following way. A node on receipt of a packet, adds to the record route information, not its own identifier but the identifier of the node from which it received the message. Thus a malicious node R’s identifier will be added into the record route by its next hop node, making it impossible for R to remove its ID from the record route field.

B. Black Listing:

The idea is that once the destination D node discovers by looking at the record-route fields of m packets, that the number of packets that passed through the same intermediate node R has exceeded p such that p/m exceeds a certain threshold ρ, the node R can be black listed. This means that from now on, no node will pass messages to node R or simply put, the presence of node R will be neglected. Any routes that involve the node R are also black listed.

The generation and updating of the black list takes place at the destination node and for traffic that flows in the reverse direction, it is generated and updated by the source node. Then these both lists have to be propagated to all the rest of the nodes which are being used for communication. The object is to keep both the lists (the source black-list and the destination black-list) to be consistent with each other. While a list is being propagated through the intermediate nodes, every node that receives it, can analyze it, compare it with the previously maintained black-list and make the necessary updates in its own black-list.

To save the overhead of transmitting whole black-lists, only the delta between the new and last known black-list can be propagated. Also instead of sending special black-list messages through the network, these can be piggy-backed on any other message as a special header. The message format shown in Fig 4 can be used.

| Enc(Ki) | Enc(black-list delta) | Msg (XOR) K1….(XOR) Km |

Fig 4. Message including black-list information
Then it is upto the DTN routing protocol to use the black-list information and form routes which do not incorporate the black-listed nodes. Since in DTN scenarios, the network topology and routes change with time, it is advisable for the black-list data to time out or expire after certain time period.

C. Managing Social Linkages:
One possible further optimization from the security perspective can be, to be conservative as to which intermediate nodes can be trusted with routing of the message. A general idea is to create a list of nominated trusted nodes and only those will be used as messengers. An application layer implementation can be to use friends’ lists from online social networks such as facebook or twitter. A reasonable trust relationship can be defined by specifying which tier of friendship can be trusted. Direct friends are probably most trust worthy, while friends of friends and then their friends are less and less trustworthy. Hence, by using the friends’ information from social networks, one can project routes with minimum number of unknown intermediate nodes.

V. CONCLUSION
In this paper, I have compiled a summary of various security related problems that are faced in delay tolerant networks and the reasons why traditional encryption mechanism are rendered useless in a DTN scenario. A number of security solutions that have been proposed for DTN have been analyzed and are found to have a number of limitations which have also been pointed out in this paper. Where, at one end the importance of security can not be neglected, on the other end, the minimal availability and duration of network connectivity means that the security protocol should be strong yet efficient. Furthermore the low reliability of message delivery and lack of timely acknowledgements in DTN makes it challenging to establish a fool proof security context. Many of these proposed methods need to be put to practical use to find out for sure how capable they are at providing security guarantees that they claim.

A number of improvements have also been pointed out where future work could be carried out and hopefully concrete steps could be taken towards the development of a unified security protocol that can cater to the security requirements of delay tolerant networks.

REFERENCES
[2] Identity-Based Cryptography for Delay-Tolerant Networking. K. Fall, A. Chakrabarti