1 Overview of System Requirement

1. The system shall encrypt messages. (Government agencies in several countries are collecting call-detail records for all individuals and storing them in large database for use in social-network-analysis.)

2. The system shall mask the sender and receiver of messages from others. (Government agencies can use fraud but mathematically valid certificate to conduct man-in-the-middle attack.)

3. The system shall guarantee that the sender of a message cannot be spoofed. (This feature can attract more use in business field because authentication is then the most important factor.)

4. The system shall not rely on trust. (Certificate authorities are around the world, even if only one of them is hacked, man-in-the-middle attack can be performed. And there are authorities run by governments.)

5. The system shall not burden the user with detailed key management. (This feature can attract novice users and give them a smooth learning curve of how-to-use.)

2 Details of System Requirement

2.1 Authentication

Function	Generate address: hash of public key.
Description	Obtain the public key by underlying protocol, then hash to verify that it belongs to the intended recipient.
Inputs	Public key, and may also include a version number for forward capability, a stream number, a checksun.
Source	Obtained from underlying protocol.
Outputs	Hash of public key.
Action	Encode the public key with base58 and prepend with recognizable characters.

2.2 Message Transfer

Function	Forward messages: a best-effort basis and proof-of-work (POW) prerequisite.
Description	Complete a POW before sending a message.
Inputs	Initial hash, and target POW difficult if specified.
Source	Random initial hash generated from Authentication.
Outputs	POW and message time.
Action	POW is in the form of a partial hash collision. The difficulty of the POW should be proportional to the size of the message. The POW running time should be set so that an average computer must expend an average of four minutes of work in order to send a typical message. Each message must also include the time in order to prevent the network from being flooded by a malicious user rebroadcasting old messages.

2.3 Scalability

Function	Self-Segregation: use *streams* and distinct *stream number* to manage large clusters.
Description	After the number of messages being sent through the Bitmessage network reaches a certain threshold, nodes begin to self-segregate into large clusters or *streams*.
Inputs	Stream number.
Source	Stream number is encoded into each address and in the second field. So just decode and read it.
Outputs	*Parent of* n: $\lfloor \frac{n}{2} \rfloor$ *if* $n > 1$; *null* *if* $n \leq 1$. *Children of* n: $[n \cdot 2, (n \cdot 2) + 1]$.
Action	When a bitmessage client starts exceeding certain threshold (in hard drive space and processing power), new address should be created in child streams and the nodes creating those address should consider themselves to be members of that stream and behave as such. Each node should maintain a list of peers in their stream and in the two child streams. Additionally, nodes should each maintain a short list of peers in root stream (the first stream that splits). The message sending algorithm is backwards-traverse.

2.4 Broadcasts

Function	Broadcast: learn through word and mouth.
Description	In bitmessage, all users receive all messages (decoding to find specific messages to them). Then broadcast messages is a natural extension.
Inputs	Broadcasters' Bitmessage address.
Source	Bitmessage client's 'Subscription' section (may vary if implemented in other applications.)
Outputs	Messages from broadcasters.
Action	In Bitmessage, messages are "broadcast". A subscription to a broadcaster will then enable you to listen to content from an authenticated identity (guaranteed by POW).

2.5 Receiver Offline Behavior

Function	Rebroadcast message: exponential backoff.
Description	In bitmessage, an object is a public key request, a public key, a person-to-person message, or a broadcast message. It is proposed that nodes store all objects for two days and then delete them.
Inputs	Bitmessage objects.
Source	Bitmessage stream.
Outputs	All messages bound for the offline node that were broadcast during the last two days.
Action	Nodes joining the network request a list of objects from their peer and download the objects that they do not have. If a node is offline for more than two days, the sending node will continue to rebroadcast the message, with exponential backoff forever.

2.6 Passive Operating Mode

Function	Passive mode: acknowledgement by random node.
Description	Specify in flags to public key, and "recruit"
Inputs	Certain flags, attached to a node's public key.
Source	Certain flags.
Outputs	Enter the passive operating mode.
Action	Package the acknowledgment up in another message and send it to either a friend or a random Bitmessage public key. The distribution of one node's public key, broadcasts or request for a public key may also execute in this manner.

2.7 Spam

Function	POW: spam uneconomic.
Description	The existing proof-of-work requirement may be sufficient to make spamming users uneconomic.
Inputs	Spam.
Source	Spamming users.
Outputs	POW workload, and if necessary, even more computing power requirement for sending messages.
Action	Existing POW requirement. And several more proposed courses of action: more difficult POW; more public key distribution for each public key in use more bits in Bitmessage address. To be short, avoiding spam in Bitmessage is just to increase spammers' workload, for every message.