Bitmessage: A Peer-to-Peer Message Authentication and Delivery System

Jonathan Warren
jonathan@bitmessage.org
www.Bitmessage.org

November 27, 2012

Abstract. We propose a system that allows users to securely send and receive messages, and subscribe to broadcast messages, using a trustless decentralized peer-to-peer protocol. Users need not exchange any data beyond a relatively short (around 36 character) address to ensure security and they need not have any concept of public or private keys to use the system. It is also designed to mask non-content data, like the sender and receiver of messages, from those not involved in the communication.

1. Introduction

Email is ubiquitous but not secure. The ability to send encrypted messages is necessary but current solutions are too difficult for people to use: users must exchange both an email address and an encryption key through a trusted channel (like in person or by phone). Even users who do know how to use tools like PGP/GPG usually do not put forth the effort to do so unless they are particularly concerned about the message content. Novice users have a difficult time learning how to use the software because the relationship between public and private key pairs, and their uses, are foreign concepts. Even if users do manage to use PGP/GPG for communications, encryption alone does not mask the sender and receiver of messages. Government agencies in several countries are collecting call-detail records for all individuals and storing them in large databases for use in social-network-analysis [1][2][3]. There would be nothing stopping them from collecting the content of phone calls and messages also, and indeed, officials have told the New York Times that the United States' National Security Agency has engaged in "overcollection" [4].

Hiding one's identity is difficult. Even if throw-away email addresses are used, users must connect to an email server to send and retrieve messages, revealing their IP address. If they use Tor to connect to a web server, they depend on the X.509 system for security. This system enables HTTPS. X.509 certificate authorities have suffered hacks in recent years including one incident where Iran used a fraudulent but mathematically valid certificate that was signed with the key of a hacked certificate authority (DigiNotar) to conduct a man-in-the-middle attack against its citizens who were using Google services like Gmail [5][6].

As the security of HTTPS is only as strong as the least trustworthy or competent certificate authority, the fact that there are over 1000 CA certificates trusted by Windows or Firefox, which are owned by hundreds of different organizations [7], should give all users great pause. Also, if just one of those organizations is run by a government agency, and if they have certain network hardware in place between users and destination servers, then they would be able to perform a targeted man-in-the-middle attack of ostensibly secure communications at will.
What is needed is a communications protocol and accompanying software that encrypts messages, masks the sender and receiver of messages from others, and guarantees that the sender of a message cannot be spoofed, without relying on trust and without burdening the user with the details of key management. In this paper, we propose such a protocol.

2. Authentication

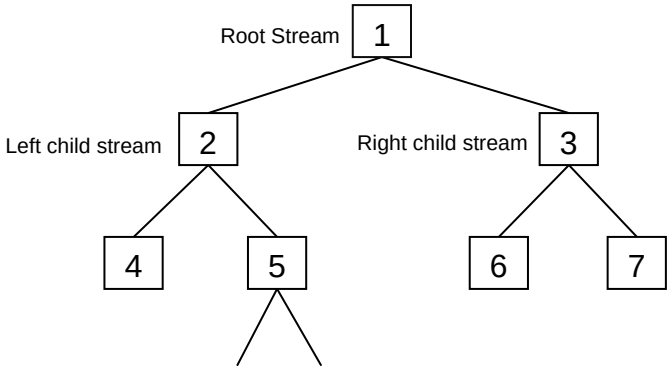
We propose a system where users exchange a hash of a public key that also functions as the user's address. If the public key can be obtained by the underlying protocol, then it can easily be hashed to verify that it belongs to the intended recipient. The data exchanged by the user can also include a version number for forwards capability, a stream number (the purpose of which will be discussed later), and a checksum. Encoded with base58 and prepended with recognizable characters (like BM for Bitmessage), an example address would be: BM-2nTX1KchxgnmHvy9ntCN9r7sgKTraxczzyE. While certainly more cumbersome than an email address, it is not too much to type manually or it can be made into a QR-code. Users have already demonstrated this to be acceptable as Bitcoin addresses are similar in format and length [8]. This address format is superior to email in that it guarantees that a message from a particular user or organization did, in fact, come from them. The sender of a message cannot be spoofed.

3. Message Transfer

We propose a message transfer mechanism similar to Bitcoin's transaction and block transfer system [8] but with a proof-of-work for each message. Users form a peer-to-peer network by each running a Bitmessage client and forward messages on a best-effort basis. In order to send a message through the network, a proof-of-work must be completed in the form of a partial hash collision. The difficulty of the proof-of-work should be proportional to the size of the message and should be set such that an average computer must expend an average of four minutes of work in order to send a typical message. With the release of new software, the difficulty of the proof-of-work can be adjusted. Each message must also include the time in order to prevent the network from being flooded by a malicious user rebroadcasting old messages. If the time in a message is too old, peers will not relay it. If the sender of a message did not receive an acknowledgement and wishes to rebroadcast his message, he must update the time and recompute the proof-of-work.
Just like Bitcoin transactions and blocks, all users would receive all messages. They would be responsible for attempting to decode each message with each of their private keys to see whether the message is bound for them.

4. Scalability

If all nodes receive all messages, it is natural to be concerned about the system's scalability. To address this, we propose that 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.* Users would start out using only stream 1. The stream number is encoded into each address. Streams are arranged in a hierarchy.



A Bitmessage client should use a negligible amount of hard drive space and processing power. Once it starts exceeding comfortable thresholds, new addresses should be created in child streams and the nodes creating those addresses should consider themselves to be members of that stream and behave as such. From then on, if the node has no active addresses in the parent stream, they need only maintain connections with peers which are also members of this child stream. With the exception of nodes in stream 1, the root stream, nodes should occasionally connect to peers in their parent stream in order to advertise their existence. 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 stream 1. In order to send a message, a node must first connect to the stream encoded in the Bitmessage address. If it is not aware of any peers in the destination stream, it connects to the closest parent stream for which it is aware of peers and then downloads lists of peers which are themselves in the two child streams. It can now connect to the child stream and continues this process until it arrives at the destination stream. After sending the message and listening for an acknowledgement, it can disconnect from the peers of that stream. If the user replies to the message, their Bitmessage client repeats the same process to connect to the original sender's stream. After this process has been carried out once, connecting to the destination stream a second time would be trivial as the sending node would now already have a list of nodes that are in the destination stream saved. The formulas to calculate a stream's parent and children are simple:

$$Parent\ of\ n = \begin{Bmatrix} \left\lfloor \frac{n}{2} \right\rfloor & if\ n > 1 \\ null & if\ n \leq 1 \end{Bmatrix}$$

$$Children\ of\ n = [n \cdot 2, (n \cdot 2) + 1]$$

5. Broadcasts

Because all users receive all messages, a natural extension of the system is to support broadcast messages. Users, through word of mouth, learn of a broadcaster who publishes content of interest to them. After entering the broadcaster's Bitmessage address into a 'Subscription' section of their Bitmessage client, messages from the broadcaster appear in the user's inbox or, if the Bitmessage protocol is implemented within another application, serve a different purpose. This would allow an individual or organization to anonymously publish content *using an authenticated identity* to everyone who wishes to listen.

6. Behavior when the receiver is offline

An object is a public key request, a public key, a person-to-person message, or a broadcast message. Objects are broadcast throughout a Bitmessage stream. We propose that nodes store all objects for two

days and then delete them. Nodes joining the network request a list of objects from their peer and download the objects that they do not have. Thus they will receive all messages bound for them that were broadcast during the last two days. If a node is offline for more than two days, the sending node will notice that it never received an acknowledgement and rebroadcasts the message after an additional two days. It will continue to rebroadcast the message, with exponential backoff, forever. In the worst case, if a user is offline for n days, he must go back online and stay connected for n days (or connect once every two days for n days) in order to receive all of his messages.

7. Passive operating mode

A particularly paranoid person who wishes to receive messages may operate in an entirely passive mode by specifying, in flags attached to his public key, that he will not send acknowledgements. It would, perhaps, be wiser for him to instead send acknowledgements but to recruit another possibly random node to send the acknowledgement for him. The other node need not even be aware that he has been recruited for this purpose.

Suppose that Alice sends Bradley a message but Bradley is too paranoid to send an acknowledgement because he fears that an attacker, Eve, is eavesdropping on his particular Internet connection in an attempt to locate him. Eve would be able to see that the acknowledgement from Bradley was broadcast from his machine earlier than from other machines, indicating that he is operating at that location. Bradley may instead choose to package the acknowledgement up in another message and send it to either a friend or a random Bitmessage public key (let us say it is owned by Charlie). If Charlie is online, he will broadcast the acknowledgement thus simultaneously acknowledging both messages at once. Bradley may also choose to distribute his public key, make broadcasts, or make a public key request in this manner. For example, in either the next message he sends to a friend or in a blank message to Charlie, he can include his public key as the acknowledgement data. This would serve to simultaneously acknowledge receipt of Bradley's message and also distribute his public key without having it originate from his Internet connection unencrypted. Even if Eve is also monitoring Charlie's Internet connection, she would not be able to tell whether Bradley is truly at that location (or if Bradley and Charlie are actually the same person). Indeed, Bradley probably is not at the location and Bradley and Charlie might very-well not even know each other. Even if most people do not use this operating mode, the fact that some people do would provide plausible deniability to those who do not.

8. Spam

The existing proof-of-work requirement may be sufficient to make spamming users uneconomic. If it is not then there are several courses of action that can be taken:

- Increase the difficulty of the proof-of-work.
- Have each client distribute x public keys for each public key that they actually use. Acknowledge messages bound for those public keys but never show the user the messages. Spammers would need x times as much computing power to spam the same number of users.
- Include extra bits in Bitmessage addresses and require that those bits be included in a message, thus proving that the sender has the Bitmessage address. Including an extra two bytes of data in a Bitmessage address would make the address 9% longer but would make spamming a user require 65536 times as much computing power. Bots who crawl the web looking for Bitmessage addresses would thwart this option.

9. Conclusion

We have presented a system that not only bridges the gap between the ease of use of email and the security of PGP/GPG, but also hides "non-content" data from prying eyes. The hassle of using a non-human-friendly address should be more than offset by the benefit of gaining privacy without having to trust fallible (or malicious) central authorities. The broadcast & subscription feature should prove especially useful to anyone wishing to anonymously publish content regularly. Paired with the BitTorrent protocol, individuals could distribute content of any size.

References

[1] "Now We Know What the Battle Was About," http://www.newsweek.com/id/174602 , 2008

[2] A. Harris, "Spy Agency Sought U.S. Call Records Before 9/11, Lawyers Say," www.bloomberg.com/apps/news?pid=newsarchive&sid=abIV0cO64zJE , 2006

[3] J. Bamford, "The NSA Is Building the Country's Biggest Spy Center (Watch What You Say)," http://www.wired.com/threatlevel/2012/03/ff_nsadatacenter/all/1 , 2012

[4] E. Lichtblau, J. Risen, "Officials Say U.S. Wiretaps Exceeded Law," http://www.nytimes.com/2009/04/16/us/16nsa.html , 2009

[5] E. Mills, "Fraudulent Google certificate points to Internet attack," http://news.cnet.com/8301-27080_3-20098894-245/fraudulent-google-certificate-points-to-internet-attack/ , 2011

[6] H. Adkins, "An update on attempted man-in-the-middle attacks," http://googleonlinesecurity.blogspot.com/2011/08/update-on-attempted-man-in-middle.html , 2011

[7] P. Eckersley, J. Burns, "An Observatory for the SSLiverse," https://www.eff.org/files/DefconSSLiverse.pdf , 2010

[8] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," http://bitcoin.org/bitcoin.pdf , 2008