ShadowChat Secure Messaging: A P2P Encrypted Instant Messaging System

Rynomster and Tecnovert
http://www.shadowproject.io
Email: devs@shadowproject.io

July 29, 2014

I. Preamble

Communication is an essential component of doing business.

Modern technology gives us cheap, reliable and effortless methods to communicate with others regardless of physical distance. However this technology does little to safeguard the content of our messages from the scrutiny of interested observers.

We live in an age of constant and ubiquitous surveillance, where it becomes more difficult by the day to retain our privacy.

Privacy is paramount when conducting business, the consequences of invasions of privacy can be devastating to both businesses and individuals, whether the attacker is a rival firm, a malicious individual or an overbearing government.

At Shadow we believe privacy is a human right - as enshrined in article 12 of the Universal Declaration of Human Rights of the United Nations. As such we strive to provide you with tools to communicate in confidence.

II. Introducing ShadowChat P2P Encrypted Instant Messaging system.

Shadow has implemented a peer-to-peer Encrypted Instant Messaging system utilising state-of-the-art technology to keep your communications private.

All messages are encrypted by the proven AES-256-CBC algorithm, and distributed between nodes in such a way as to prevent the recipients of messages from being inferred by assailants utilising sophisticated traffic analysis, even if the assailants can view the entire network and/or run nodes of the network.

To eliminate the risk and hassle of sharing passwords, we utilise the proven and trusted method of Elliptic Curve Diffie-Hellman (ECDH) key exchange.

The Elliptic Curve Digital Signature Algorithm (ECDSA) is used to give you confidence that the messages you receive come from where they claim to.

Messages are distributed over the existing Shadow peer-to-peer network, and a copy of each encrypted message is stored on each node for a period of 48 hours.

III. Technical Details

A. Key Sharing

The Elliptic Curve Diffie-Hellman (ECDH) key exchange method allows a secret key for encryption to be shared between the sender and recipient using the data embedded in the message along with the private keys of Shadow addresses held by the sender and recipient.

In order to send a ShadowChat encrypted message, you must possess the public key to a private key of the intended recipient. The public keys embedded in the Shadow transaction blockchain when an amount is spent. If you are sending to an address that has not spent a transaction in the blockchain the public key to that address must be provided manually.

Shadow uses curve secp256k1 for all elliptic curve functions. This is the same curve used by bitcoin along with the vast majority of altcoins. With such widespread use underpinning systems of immense value it is extremely unlikely that curve secp256k1 is not secure.

Messages are signed by the keys they were sent with, this allows you to be confident of the origin of the messages you receive and also allows the public key of the sender to be extracted from the message, providing you all the information needed to send a reply.

B. Encryption

Detailed Procedure:

- Get public key K from destination address.
  - Find in database created from scanning for public keys in the blockchain and user additions.
- Generate 16 random bytes using a secure random number generator. Call them IV.
  Generate a new random EC key pair with private key called r and public key called R.
- Generate shared secret key P using public key K and private key r.
  - Elliptic Curve Diffie-Hellman
- Use the shared secret key P and calculate the SHA512 hash H.
  - ECDH_compute_key of OpenSSL
  - Call the first 32 bytes of H key_e and the last 32 bytes key_m.

- Calculate a 32 byte MAC with HMACSHA256, using key_m as salt and (timestamp + destination + cipher text).
    - Message authentication code used
    - By also checking time-stamp and destination, recipients can be certain that these fields have not been tampered with.
- Generate a compact signature from the message data and sender's address.
    - Only if not sending anonymously
    - Recipient can verify that the message came from the sender
    - Also allows the public key to be reconstructed (useful to reply)
- Include address and compact signature in the payload to be encrypted.
- Compress the plain-text message with lz4 if the message is larger than 128 bytes.
- Encrypt the payload data with AES-256-CBC, using IV as initialization vector, key_e as encryption key.

C. Message Propagation

Encrypted Messages are duplicated on every participating node in the Shadow network - this prevents adversaries from uncovering the recipient of an encrypted message by using network traffic analysis.

The messages are stored on each node for a maximum period of 48 hours, after which the message is deleted. If the recipient is absent from the network for 48 hours or more the possibility exists that they may not receive messages sent to them. It is recommended to connect to the network each day in order to prevent such an occurrence.

Stored messages are grouped by time in divisions of 1 hour. The system operates on the grouped buckets of messages to save bandwidth.

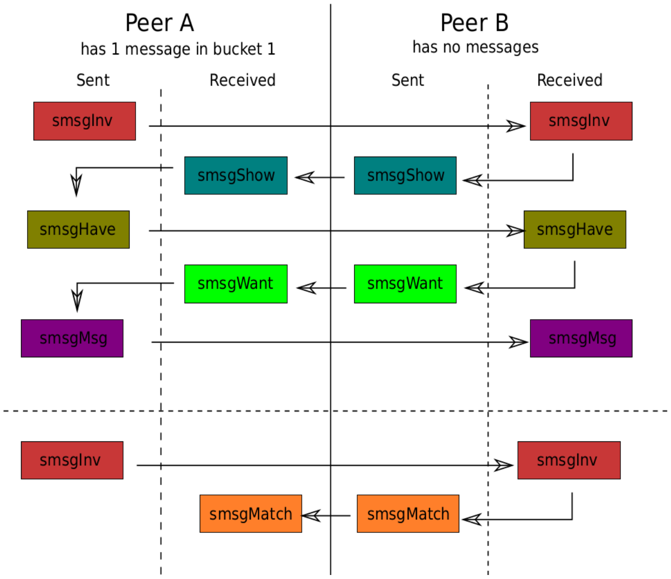

Fig. 1. Synchronisation of buckets between peer nodes

D. Decryption

For each incoming message a node will attempt to decode the message with every owned address contained in the nodes white-list of addresses to receive messages on.

To speed up the process and allow for any payload format the Message authentication code (MAC) is calculated for the generated shared secret key, if it does not match the MAC provided in the message, decryption will fail and the function ends.

Detailed Procedure:

- Get IV and R from the message block
- Get the private key k of the recipient used to decrypt.
- Generate shared secret key P using with private key k and public key R.
    - Elliptic Curve Diffie-Hellman
- Use the shared secret key P to generate the SHA512 hash H.
    - Call the first 32 bytes of H key_e and the last 32 bytes key_m.
- Calculate MAC' with HMACSHA256, using key_m as salt and hash of (time-stamp + destination + cipher text).
- Compare MAC with MAC'.
    - Return if not equal, decryption will fail.
- Decrypt the encrypted payload with AES-256-CBC, using IV as initialization vector, key_e as decryption key.
- Decompress message portion with lz4 if message is larger than 128 bytes.
- If address and compact signature were included then verify the message Address and compact signature are not included when message is sent anonymously Strip the sender's public key and add it to the public key database.

References

[1] *The Universal Declaration of Human Rights*, 1948, [Online]. Available at http://www.un.org/en/documents/udhr/index.shtml.

[2] S. Nakamoto, *Bitcoin: A PeertoPeer Electronic Cash System*, 2008, [Online]. Available at http://bitcoin.org/bitcoin.pdf.

[3] J. Warren, *Bitmessage: A PeertoPeer Message Authentication and Delivery System*, 2012, [Online]. Available at http://bitmessage.org/bitmessage.pdf.

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

[5] E. Mills, *Fraudulent Google certificate points to Internet attack*, 2011, [Online]. Available at http://news.cnet.com/8301 27080_320098894245/fraudulentgooglecertificatepointstointernetattack/.

[6] J. Bamford, *The NSA Is Building the Countrys Biggest Spy Center (Watch What You Say)*, 2012, [Online]. Available at http://www.wired.com/threatlevel/2012/03/ff_nsadatacenter/all/1.

[7] *Now We Know What the Battle Was About*, 2008, [Online]. Available at http://www.newsweek.com/id/174602.

[8] E. Lichtblau and J. Risen, *Officials Say U.S. Wiretaps Exceeded Law*, 2009, [Online]. Available at http://www.nytimes.com/2009/04/16/us/16nsa.html.

[9] H. Adkins, *An update on attempted maninthemiddle attacks*, 2011, [Online]. Available at http://googleonlinesecurity.blogspot.com /2011/08/updateonattemptedmaninmiddle.html.

[10] P. Eckersley and J. Burns, *An Observatory for the SSLiverse*, 2010, [Online]. Available at https://www.eff.org/files/DefconSSLiverse.pdf.