Bitcoin Message: Data Insertion on a Proof-of-Work Cryptocurrency System

Matthew D. Sleiman, Adrian P. Lauf, Roman Yampolskiy
Dpt. of Computer Engineering and Computer Science
University of Louisville
Louisville, Kentucky, USA
adrian.lauf@louisville.edu

***Abstract*— The Bitcoin currency is steadily growing in popularity as an alternative to physical currencies. This paper presents an approach for utilizing the Bitcoin system for creating permanent messages that are located on computers worldwide. By manipulating the amount field of Bitcoin transactions, messages can be embedded into the Bitcoin block chain. This approach was implemented by extending the Bitcoin-Qt v0.7.2 application and the source code is freely available.**

***Keywords*— arithmetic coding, Bitcoin, C++, open source, message**

I. Introduction

Bitcoin is a rapidly-growing, fully-digital crypto-currency that eschews the need for a central financial authority. Unlike traditional Internet transactions, Bitcoin transac-tions have no need for a financial institution as a trusted third party, or the need to trust any other user. This allows the Bitcoin currency to avoid many of the problems inherent to a trust-based transaction model. In the trust-based model, when either a merchant or a customer disputes a transaction, the financial institution is forced to mediate and, if need be, reverse the transaction. Transaction reversal creates the possibility for fraud and engenders an inherent mistrust between merchants and customers. These problems can be avoided with the use of physical currency but physical currency cannot be used for Internet transactions. Thus, Bitcoin was introduced in 2009 by Satoshi Nakamoto to alleviate these problems for electronic payment [1, 2].

The cryptocurrency (Bitcoin) removes the need for a trusted third party by utilizing a global transaction log known as the block chain [3]. The block chain consists of an ever-growing num-ber of blocks with each block containing information for transactions that have not already been included in a previous block, as well as a hash of the previous block. In this manner, the blocks form a chain that is used as proof of the sequence of transac-tions. Blocks are generated through a process known as mining. Mining is a proof-of-work process that involves searching for a block's SHA-256 hash value that begins with a certain number of zero bits. A nonce value is incremented within the block until hashing the block produces the desired value. Given that this process is compu-tationally intensive and that each block includes the hash of the previous block, modifying a block becomes increasingly difficult the deeper the block is embedded in the block chain. Therefore, the block chain can be trusted as a record of all transactions provided that honest miners control the majority of the Bitcoin network, and that the blocks in consideration are deeper in the block chain. In this manner, a third party is not needed to verify transactions [1].

Because the block chain is permanent and maintained by every node on the Bitcoin network, it represents an opportunity to be used for storing information be-yond just a record of transactions. This fact, along with the partially-anonymous nature of Bitcoin addresses [1, 2, 4], makes Bitcoin an appealing system for storing and sending secret messages. This paper proposes an approach for storing and retrieving arbitrary messages within and from the Bitcoin block chain, and demonstrates this as an effective vector for possible data insertion attacks, as well as an opportunity to send and store permanent messages that exploit Bitcoin's organization.

II. Prior Work

A messaging system based on the Bitcoin protocol has been proposed by Warren in [5]. This system uses a network that processes messages similar to how the Bitcoin network processes transactions. Messages are transmitted using a "Bitmessage" net-work, which operates on peer-to-peer principles similar to BitTorrent. In this method-ology, a Bitcoin address is used as an identifier, and the Bitmessage network serves to send the message through distributed communications. Although the service uses proof of work and authentication schemes that are inherent to the Bitcoin network, it still requires a specialized network that works independently of the Bitcoin network. Each message has its own proof-of-work (in contrast, only blocks have a proof-of-work in Bitcoin). There is no equivalent to the block chain [5]. Unlike this system, the approach we propose does not require a separate network for messaging; it instead utilizes the Bitcoin network and block chain for sending and storing messages.

A service for embedding messages into the Bitcoin block chain called "Bitcoin Message Service" was developed for transmitting information via Bitcoin, and re-mains the closest approximation to the proposed work [6]. The service uses a central-ized server to which users can submit requests for embedding a message in the block chain. Users are required to send Bitcoin payments to cover the cost of embedding the

978-1-4673-9403-1/15 $31.00 © 2015 IEEE
DOI 10.1109/CW.2015.56

message. The process by which messages were embedded into the block chain is not described. This project is no longer in active development, and is not in use. [6].

The work proposed here is in contrast to messaging systems that intentionally hide the source of a transmission through a semi-trusted anonymous network. In light of the recent surges in popularity of anonymous internet services, such as the Onion Router (TOR) project, which permits users to post to message groups, access re-sources, and browse content anonymously [7-9]. By utilizing a set of three connection nodes, one of which is an exit node, and a set of concentric encryption layers, TOR provides a degree of anonymity to message posting, web browsing, and other activity. However, with sufficient statistical knowledge, or the use of compromised intermediary nodes, identity can be implied.

Other networks, such as darknets, further obfuscate both source and destination through proof-of-work addressing concepts, hidden services, and anonymous network connections between peers. Messaging in this domain is exclusive to machines that have been trusted with long addresses, and authenticated on the network itself [10]. Our proposed protocol and architecture provides methods of anonymity that do not require secondary anonymous internet mechanisms.

III. Approach

In order to store an arbitrary message within the block chain, some aspect of the block chain must be manipulated. The easiest and most obvious aspect under user control is the amount of a transaction. Therefore, this paper's approach utilizes an algorithm to encode messages into the amount field of a Bitcoin transaction.

We have elected to use arithmetic coding. In this form of coding, a message is represented by a real number greater than or equal to zero, and less than one. The range from zero to one is subdivided based on the relative probabilities of the alphabet used by the message [11-13]. For example, if both the symbols "A" and "B" occur at a frequency of 10% within messages, then "A" could be represented by the range 0 up to, but not including, 0.1 and "B" could be represented by 0.1 up to, but not including, 0.2. The entire range from zero to one is mapped to the symbols of the alphabet in this manner.

TABLE I
Relative Frequencies of English Characters

Letter	Frequency	Letter	Frequency
Space	12.17	N	5.44
A	6.09	O	6
B	1.05	P	1.95
C	2.84	Q	0.24
D	2.92	R	4.95
E	11.36	S	5.68
F	1.79	T	8.03
G	1.38	U	2.43
H	3.41	V	0.97
I	5.44	W	1.38
J	0.24	X	0.24
K	0.41	Y	1.3
L	2.92	Z	0.03
M	2.76	Other	6.57

With a message consisting of only a single symbol, any real number within that symbol's range could be used to represent the message. For example, the message "A" would be represented by any real number between 0 up to, but not including, 0.1. To add a second symbol to the message, the range of the first symbol is subdivided based on the probabilities of the alphabet in the same manner the original range was subdivided. So the message "AB" would be represented by any real number between 0.01 up to, but not including, 0.02. This process repeats, each time producing an ever smaller range, until the entire message is encoded. Lastly, the final number used to represent the message is typically selected by finding the number within the resulting range that requires the fewest bits to represent [11, 14]. Table I shows the relative frequencies of English characters [15].

Arithmetic coding was chosen mainly because it produces code words that are between zero and one. This allows messages to be sent without the need for a large amount of bitcoins. Bitcoins are able to be divided down to eight decimal places which can be used for encoding a message. However, eight decimal places are not enough to encode an arbitrarily long message. To solve this problem, the entirety of a message is denoted by address to which the Bitcoin is sent. Therefore, multiple transactions can be used to construct a longer message. In order to reconstruct the message in the correct order, the implementation ensures the previous transaction is included in a block before the next transaction is broadcast to the network. Additionally, sequence information can be encoded in each transaction, though this further reduces the amount of information that can be encoded in each transaction. It is possible for a user to send Bitcoins from an address he owns to another address he owns even if the from and to addresses are the same. This allows Bitcoins to be reused in the same message, making it possible to send infinitely long messages with only a single Bitcoin.

The largest disadvantage to this approach is the time required to send messages. Each transaction must be included in a block by a miner who successfully completes the proof-of-work process before any other miner. Transaction fees can be used to incentivize miners to include a user's transaction in a block [1, 3]. However, including transaction fees means that messages can no longer be sent free of charge. Without a transaction fee, a single transaction can take anywhere from 10 minutes to several days before being included in a block. Likewise, transactions that use Bitcoins that were recently used in another transaction are given a low priority, further

```
QList<qint64> CMessageCoder::encode(const
          QString &message) const
          {QList<qint64> result;
         for (int i = 0, chunkLength =
          maxSymbolsPerCodeWord;
               i < message.length();
         i += chunkLength, chunkLength =
         maxSymbolsPerCodeWord){
          QString chunk = message.mid(i,
               chunkLength);
           qint64 encodedChunk;
            while (chunkLength > 0 &&
!tryEncode(chunk + terminator, &encodedChunk)){
                    chunkLength--;
                chunk = message.mid(i,
              chunkLength);}
           result.append(encodedChunk);}
               return result;}
```

Fig. 1. The encode function of the implementation.

```
bool CMessageCoder::tryEncode(const QString
        &text, qint64 *result) const
     {long double low = 0.0, high = 1.0;
    for (int i = 0; i < text.length(); i++){
          long double range = high - low;
          long double previousLow = low;
            low = previousLow + range *
      characterRanges[text[i]].first;
           high = previousLow + range *
     characterRanges[text[i]].second;}
             QString lowString =

QString::fromStdString(boost::lexical_cast<std::
        string, long double>(low));
             QString highString =

QString::fromStdString(boost::lexical_cast<std::
        string, long double>(high));
            QString output = "0.";
   for (int i = 2; i < lowString.length() && i
        < highString.length(); i++){
        if (lowString[i] == highString[i]){
             output.append(lowString[i]);}
               else if ((output +
    highString[i]).toDouble() == high){
             output.append(lowString[i]);
          for (i++; i < lowString.length();
                 i++){
                    int digit =
      QString(lowString[i]).toInt();
                  if (digit == 9){

       output.append(lowString[i]);}
                          else{

  output.append(QString::number(digit + 1));
                            break;
                             }
                           }
                         break;}
                        else
           {output.append(highString[i]);
                         break;}
                        }
                      return
BitcoinUnits::parse(BitcoinUnits::BTC, output,
                    result);
                      }
```

Fig. 2. The tryEncode function used by the encode function shown in Fig. .

decreasing the likelihood the transaction will quickly be included in a block [16]. This becomes especially apparent when Bitcoins are reused within the same message.

IV. Implementation

We implemented our proposed method by extending the open-source application Bitcoin-Qt v0.7.2. The source code for this application can be found at [17]. A new tab was added to the application's main window that gives users access to the encod-ing and decoding functionality.

The encoding algorithm is based on the arithmetic coding algorithm in [14]. Only lower-case English characters, spaces, and periods can be encoded. However, the period is reserved as a termination symbol to stop the decoding algorithm. The relative probabilities used for each symbol were taken from [15]. The implementation extends the arithmetic coding algorithm by first attempting to encode seven characters plus the termination symbol. If the resulting code word is a valid Bitcoin amount, it is accepted. However, if it is not valid, the algorithm will try to encode six characters plus the termination symbol. This repeats, each time with one less character, until a valid Bitcoin amount is produced. Since the maximum amount of characters that can be encoded in a single code word is dependent on the characters being encoded, this approach helps ensure maximum information density for each code word. The source code for this algorithm is shown in Fig. 1 and Fig. 2. The decoding algorithm, also based on the algorithm in [14],

```
QString CMessageCoder::decode(const QList<qint64>
             &encodedMessage) const
               {QString message;
                   qint64 chunk;
          foreach (chunk, encodedMessage)

{message.append(decodeChunk(BitcoinUnits::format(
    BitcoinUnits::BTC, chunk).toDouble()));}
                  return message;}
 QString CMessageCoder::decodeChunk(long double
                value) const
            {QString decodedMessage;
                    QChar symbol;
                   while (true){
          long double low = 0.0, high = 0.0;
        foreach (symbol, characterRanges.keys())
         {low = characterRanges[symbol].first;
                        high =
       characterRanges[symbol].second;
           if (value >= low && value < high){
                         break;}
                       }
             if (symbol == terminator)
                      {break;}
          decodedMessage.append(symbol);
           if (decodedMessage.length() >
          maxSymbolsPerCodeWord)
             {decodedMessage.clear();
                      break;}
       value = (value - low) / (high - low);
                     }
            return decodedMessage;
                    }
```

Fig. 3. The decode algorithm used by the implementation.

simply decodes each piece or "chunk" of the message and concatenates the results. This algorithm is shown in Fig. 3. The entire implementation is available as an open-source application at the URL in [18].

A. Applications

The applications of this method are clear; permanent embedding of information into a globally-accepted and verifiable financial currency can be done easily, cost-effectively, and – with sufficient obfuscation – privately. Only parties with a vested interest in an arithmetic coding algorithm need know what additional contents are part of a transaction.

By utilizing the approach presented in this paper, arbitrarily long messages can be permanently embedded within the Bitcoin block chain. These messages are located on computers worldwide because every node on the Bitcoin network maintains a copy of the block chain.

The first, and most obvious, use of this ability is secret communication. As long as addresses remain anonymous, a user can send messages to anyone worldwide without exposing his identity or the identity of the messages' recipients. Utilizing a coding algorithm known only to the sender and intended recipients can obfuscate the contents of the message to others.

A second use is simply backing up important information. The block chain is permanent by nature and exists on thousands of systems. As long as Bitcoin remains in use, any information embedded within the block chain will remain always accessible; and, if the recent surge in popularity is any indication, Bitcoin will remain in use indefinitely.

The permanency and global nature of the block chain also allows for posting of information that cannot be censored or deleted. The proof-of-work required to add blocks to the block chain makes transactions increasingly difficult to alter further down in the block chain. Anyone attempting to alter or delete a transaction would require significantly more hashing power than the aggregate hashing power of legitimate miners to win the race condition. This makes it highly unlikely that any message encoded in the block chain will ever be altered or removed, which allows users to post information without the concern of being censored.

The ability to embed information in the block chain also allows for commitment schemes to be developed. A user could encrypt his commitment with a certain key and send the result in a message. Eventually, the message would be included in the block chain which would serve as proof that the commitment was made before a certain time. At a later time, when the user wants to reveal the commitment, he can send the key he used to encrypt the commitment in a message. This will make the key publicly available and allow anyone to verify his commitment. A commitment scheme utilizing Bitcoin has been developed in [19].

Of course, nothing restricts our proposed approach to storing only text data. Any information such as images, video, software, and sound could be embedded in the block chain. As long as the information can be converted to a digital format, it can be stored in the block chain. The storing of different types of information coupled with the permanency of the block chain can also be used to encode personal posteriority.

Lastly, the capability to store information within the block chain could be used in a negative manner. Because the information sent using this approach cannot be censored, is broadcast worldwide, and senders are mostly anonymous, this approach could very easily be used to transmit and store illicit information, such as child pornography or copyright material. It would be very difficult to keep illegal information out of the block chain while still maintaining the integrity of the Bitcoin currency.

V. Test Results

Several tests on the actual Bitcoin network were performed. The first message successfully encoded in the block chain simply reads "test one". It was sent to the Bitcoin network on March 15th, 2013. The message is broken into two transactions with amounts of 0.68502336 and 0.24. Both transactions were sent to the Bitcoin address 1KQjQJDESW676eis1sxvs3gE1CMcosqPtf.

A third transaction was sent on March 16th, 2013 to the same address in order to append "end" to the first message. This transaction had an amount of 0.834187.

This message can be decoded by the implementation by using the above address and searching blocks created on March 16th and 17th, 2013.

A second successful test was performed on March 27th, 2013. The message was "hello world" and was sent to the address 12jVDm2YAKUE49vAXjtR6KHKRMQEj39Q7q. The message was broken into three transactions with amounts of 0.28017303, 0.9086597, and 0.37306. The first two transactions were included in blocks on March 28th, 2013 and the third transaction was included in a block on March 29th, 2013. Similar to above, these can be decoded by using the above address and the dates the transactions were included in a block.

VI. Conclusions

This paper demonstrates a way of embedding information, such as text or other digital data, into Bitcoin transactions using an application of arithmetic coding. By embedding data into the amount field of the transaction, the authenticity of the data is verified and preserved once it becomes part of a block in the Bitcoin block chain. We have demonstrated that this can be done with minimal computational effort, small amounts of required currency, and without the need for a centralized or decentralized second network. Essentially, Bitcoin message provides a simple yet effective way of using existing properties of the network to transmit data in a distributed, secure and verifiable fashion without needing to change anything about Bitcoin itself.

VII. References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Consulted,* vol. 1, p. 28, 2008.

[2] F. Reid and M. Harrigan, *An analysis of anonymity in the bitcoin system*: Springer, 2013.

[3] D. Ron and A. Shamir, "Quantitative analysis of the full bitcoin transaction graph," in *Financial Cryptography and Data Security*, ed: Springer, 2013, pp. 6-24.

[4] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating user privacy in bitcoin," in *Financial Cryptography and Data Security*, ed: Springer, 2013, pp. 34-51.

[5] J. Warren, "Bitmessage: A peer-to-peer message authentication and delivery system," *white paper (27 November 2012), https://bitmessage. org/bitmessage. pdf,* 2012.

[6] "Bitcoin Message service," ed: Wordpress.com, 2011.

[7] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," DTIC Document2004.

[8] D. Goldschlag, M. Reed, and P. Syverson, "Onion routing," *Communications of the ACM,* vol. 42, pp. 39-41, 1999.

[9] M. G. Reed, P. F. Syverson, and D. M. Goldschlag, "Anonymous connections and onion routing," *Selected Areas in Communications, IEEE Journal on,* vol. 16, pp. 482-494, 1998.

[10] P. Biddle, P. England, M. Peinado, and B. Willman, "The darknet and the future of content protection," in *Digital Rights Management*, ed: Springer, 2003, pp. 155-176.

[11] G. G. Langdon Jr, "An introduction to arithmetic coding," *IBM Journal of Research and Development,* vol. 28, pp. 135-149, 1984.

[12] J. Rissanen and G. G. Langdon Jr, "Arithmetic coding," *IBM Journal of research and development,* vol. 23, pp. 149-162, 1979.

[13] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM,* vol. 30, pp. 520-540, 1987.

[14] Z.-N. Li, M. S. Drew, and J. Liu, *Fundamentals of multimedia*: Springer, 2004.

[15] E. S. Lee, "Essays about computer security," *Centre for Communications Systems Research Cambridge, c Cambridge,* 1999.

[16] (2013, March 31 2013). Available: https://en.bitcoin.it/wiki/FAQ

[17] "bitcoin/bitcoin at 0.7.2 - GitHub," 0.7.2 ed.

[18] M. D. Sleiman, "mdslei01/bitcoin-message • GitHub," ed, 2013.

[19] J. Clark and A. Essex, "Commitcoin: Carbon dating commitments with bitcoin," in *Financial Cryptography and Data Security*, ed: Springer, 2012, pp. 390-398.