# Unmasking Criminal Enterprises: An Analysis of Bitcoin Transactions

Jonathan Oakley, Carl Worley, Lu Yu, and Richard Brooks
Dept. of Electrical and Computer Engineering
Clemson University
Clemson, SC., USA
Email: {joakley, cworley, lyu, rrb}@clemson.edu

Anthony Skjellum
Cyber Security Research Center
Auburn University
Auburn, AL., USA
Email: tony-skjellum@utc.edu

## Abstract

*With the rise of cryptographic ransomware, Bitcoin has found a niche as the standard currency for ransoms. While Bitcoin is pseudonymous, it provides no guarantee of untraceability. As a result, another niche has arisen–Bitcoin money laundering. Hidden Markov Models (HMMs) have previously been used in a number of applications where traditional pattern recognition falls short. In this paper, HMMs are inferred from transactions in the public blockchain in an attempt to link users, events, and enterprises. We introduce a proof-of-concept algorithm to infer HMMs from the Bitcoin blockchain.*

## 1 Introduction

In May of 2017, a new cryptographic ransomware, WannaCry [11], began infecting computers and encrypting files. The premise was simple–encrypt all of the victim's files, then demand a ransom in exchange for decrypting them. To pay the ransom, the victim sent a payment to a bitcoin address. Since bitcoin uses a public ledger, it's trivial to see where the coins are sent. The question is, how can the bitcoins be *tracked*? There is no guarantee the owner of the coins after a transaction was a part of their illegal acquisition. The challenge is then to group pseudonymous accounts and link them to physical users.

Prior to the rise of cryptocurrencies, these types of attacks were far less prevalent. In 1989, the AIDS trojan was distributed on floppy disks, and after 90 power cycles, it demanded a random of $189 be sent to a Post Office box in Panama [9].

Like a Post Office box, a Bitcoin address is public and can be watched for activity. In fact, it's so easy to watch a particular address, the WannaCry epidemic became a spectator sport–news outlets reported daily on the profits generated from the attack. Since WannaCry used hard coded Bitcoin addresses for the ransom payments, it is trivial to watch the Bitcoin blockchain for transactions to those addresses. In fact, the website blockchain.info allows you to enter a Bitcoin address and view the current balance associated with the address. However, even this level of effort is unnecessary since there are Twitter bots that tweet the total profit of the attack (@actual_ransom). Interestingly, as of July, 29$^{th}$ 2017, none of ransom has been withdrawn. Due to the public nature of Bitcoin, it will be possible to track any future addresses associated with those funds.

In Section 2, work relating to Bitcoin anonymity and pattern inference are introduced. Section 3 provides the essential framework for understanding Bitcoin transactions. Section 4 introduces a theoretical approach for unraveling these transactions, and Section 5 presents conclusions and directions of future research.

## 2 Related Work

Barber et al. [5] looked at the key weaknesses in Bitcoin's architecture and found several substantial anonymity exploits. The most popular solution is mixing. Bitcoin mixers are trusted entities that seek to anonymize transactions by combining funds before dispersing them to newly created Bitcoin addresses [3].

Ober et al. [14] investigate simplifying Bitcoin transaction graphs with "merging events"–an assumption that all the inputs to a transaction are owned by the same user. Androulaki et al. [4] use the same process of "merging events" to simplify transactions and attack user anonymity by comparing transactions and identify geographic clusters of users. Narayanan and Shmatikov [13] used the overlap between graphs to de-anonymize users. They found that user's habits outside of an anonymous social network compromised their anonymity. Ruffing et al. [16] propose CoinShuffle as an alternative to traditional bitcoin mixing. Coin-

978-1-7281-0155-2/18/$31.00 ©2018 IEEE

Shuffle omits the traditional trusted third party. Watkins et al. [18] lists cluster analysis, neural networks, fuzzy logic, and genetic algorithms as possible ways to detect money laundering.

Fu et al. used Hidden Markov Models (HMMs) to detect botnets using Domain Generation Algorithms [8] , as well as to communicate covertly through probabilistic domain names[7]. Zhong et al. [20] used HMMs to disguise botnet traffic as smart grid traffic. Ruiwen et al. [17] used Markov models to characterize power systems.

## 3 Background

### 3.1 Hashes

A hash is a function that uniformly maps data from a domain to a codomain–they are easy to compute in one direction and hard to compute in the other direction. Bitcoin uses two hashes: SHA-256 and RIPEMD-160 with 256 and 160 bit outputs respectively.

### 3.2 Bitcoin

Bitcoin is a peer-to-peer decentralized currency conceptualized by Nakamoto [12] in 2008. Bitcoin's claim as a pseudonymous crytocurrency provides users with no delusions of anonymity [1]. Users are dissociated from their transactions, which are stored on a distributed ledger, by a veil of uncertainty. Since the ledger is public, it is possible to trace transactions between addresses.

Below, Figure 1 shows how Bitcoin addresses are constructed. The steps are as follows [6]:

1. Acquire an Ecliptic Curve Digital Signature Algorithm (ECDSA) keypair (public and private keys) and prepend the '0x00' version byte to the public key.
2. Hash this extended key using the SHA-256 hashing algorithm.
3. Hash the data again using the RIPEMD-160 hashing algorithm.
4. Prepended a description byte ('0x00' for mainnet, '0x6F' for testnet) to the RIPEMD-160 hash.

5-6. Calculate the checksum by taking two sequential SHA-256 hashes of the extended RIPEMD-160 hash.

7. Append the first four bytes of this checksum to the extended RIPEMD-160 hash from Step 4.
8. Encode these 25 bytes using Base58 encoding to produce a valid Bitcoin address.

When a Bitcoin user makes a transaction, the miners group that transactions with all the other unconfirmed transactions in the memory pool. Transactions are chosen from the memory pool to be included in blocks based on their fee

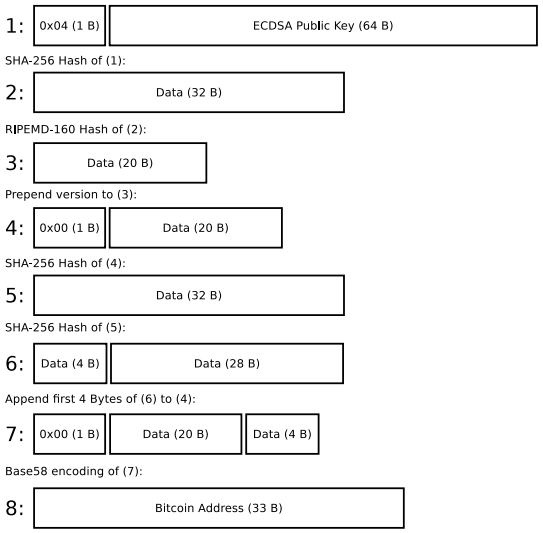


Figure 1: Creating a Bitcoin address from a public key.

and their age. When a miner solves the computational puzzle, its block (and the solution to the puzzle) is broadcast to the other miners, and the process begins anew. Below, Figure 2 shows the structure of these transactions in relation to the blockchain.

An input consists of an existing Transaction ID and the index of the desired output in the old transaction. No value is specified in an input since the entire value of the output is consumed in the transaction. In addition to the previous output, a ScriptSig (ScriptSignature) is provide in order to prove ownership of the output. The ScriptSignature consists of a signature concatenated with the corresponding public key. The signature is a cryptographic proof-of-ownership of the public key, since it requires the private key associated with the public key.

The outputs of a transaction include the value (in bitcoins), the current index of the output in the output array, and the ScriptPubKey. The ScriptPubKey is the usual script for sending bitcoins to a specific address. The script consists of the steps required to prove ownership of the output. Instead of providing the public key, the hash in Step 3 is provided.

Since spending an output requires the entire output to be consumed, *change addresses* are created to receive any leftover funds. While these change addresses are unique, there is some evidence that they can be associated with the sender [14, 4]. Below, Figure 3 presents a simplified version of such a transaction with the *change address* indicated by an asterisk.

In a transaction, the difference between the input amount and the output amount is the fee the miner collects by including the transaction in the block. In Figure 3, there are 1.5 BTC consumed by the transaction and 1.49 BTC spent. This leaves a transaction fee of 0.01 BTC. This transaction

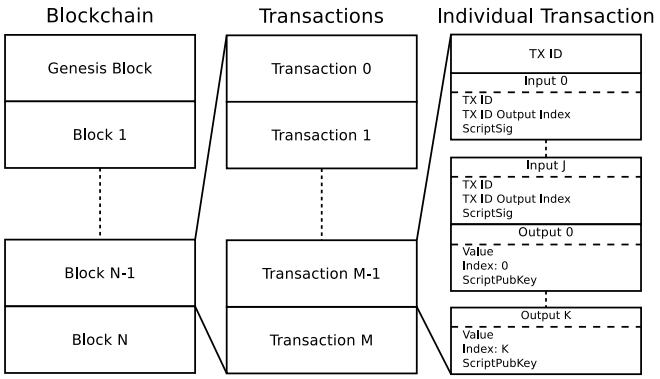


Figure 2: Hierarchy of a Bitcoin transaction.

can be simplified by forcing the change address to be the owner's address.

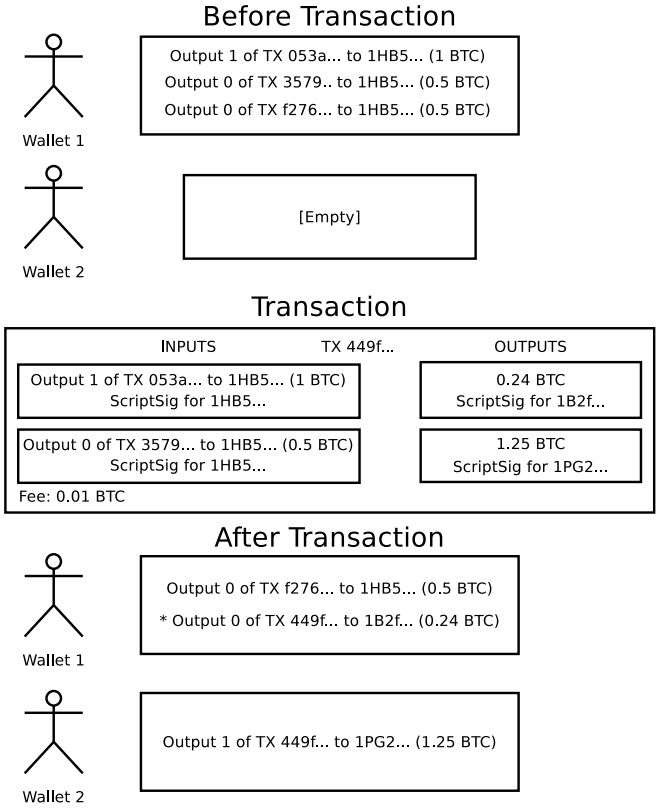


Figure 3: Bitcoin transaction with *change addresses*.

### 3.3 HMMs

A Markov model represents a stochastic random process in the form of a directed graph in which state transition probabilities depend only on the previous state. A Hidden Markov model (HMM) refers to a Markov model where the states are either unobservable or non-obvious [15]. An example of a Markov model is shown in Figure 4. When considering Bitcoin transactions, the observations are a time series of transactions between nodes. This model is not evident for any non-trivial examples. As such, the model in Figure 4 can be thought of as hidden in the time series observations. Specifically, a Deterministic HMM is assumed for this model [10]. Section 4.2 details the novel method for inferring this HMM from the Bitcoin blockchain.

## 4 Theoretical Approach

Since testing this inference method on the actual blockchain would be pointless without having target addresses, it was sufficient to simulate the model using Bitcoin's regression testing environment [2]. This mode generates a new blockchain locally and provides the user with the power to generate blocks at will. In addition to ensuring that tests are run quickly, this environment ensures that the desired transactions are always included in the block, irregardless of the transaction fee.

### 4.1 A Simple Criminal Enterprise

Consider the criminal enterprise depicted below in Figure 4.

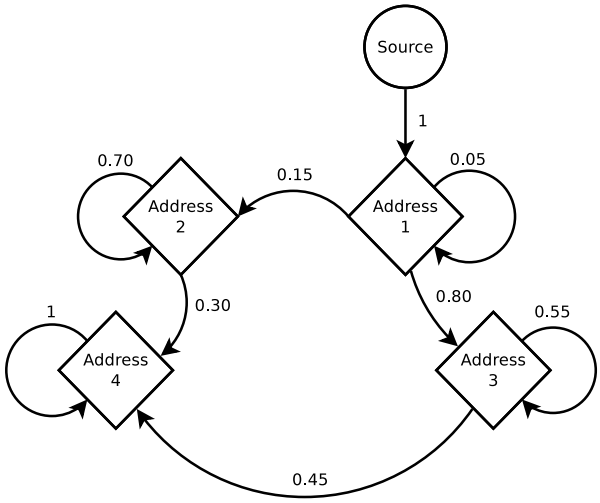


Figure 4: A theoretical criminal enterprise.

In this scenario, there are four actors. One actor is consistently receiving an income. This first actor saves 5%, give 15% to a second actor, and gives the remaining 85% to a third actor. This second actor sends 30% to a fourth actor and saves the remaining 70% of the funds. The third actor send 45% of their received funds to the same fourth actor and saves the remaining 55% of the funds. While this may seem trivial, it could represent a number of completely

realistic scenarios. In order to generate the model shown in Figure 4, the following algorithm was used:

**Input :** head node
`SendFunds(`*coinbase,head node*`)`
**Function** `SendFunds(`*source node, dest node*`)`
  **Input :** Node to send transaction too.
  **Result:** Transaction is created between the source node and the destination node.
**Event** `ReceivedTransaction(`*source node*`)`
  **Input :** Node that received the transaction
  Determine 'dest node' according to probabilities
  **if** *source node != dest node* **then**
    `SendFunds(`*source node, dest node*`)`
  **else**
    Keep Funds
  **end**

**Algorithm 1:** Model generation algorithm.

Raw transactions were used to disperse the funds. This ensured control over the transaction fee, as well as which previous transactions were used to fund the current transaction. The raw transactions also allowed the *change addresses* to be the same as the sender's address, which is not usually not the case.

Algorithm 1 takes the head node to a model, such as the one in Figure 4. The head node deterministically disperses one BTC to an account according to the model. When the first node receives the funds (*event*), that node determines the where the funds should be sent according to the probabilities in the model (Equation 1). Note that the event condition in Algorithm 1 continues to occur until a node keeps the one BTC. This algorithm can be thought of as the 'plinko' game, where a one BTC is dropped into the head of the model and 'falls' according to the probabilities.

## 4.2 Inferring the Model

Below, Equation 1 details how the probability was calculated for dispersing funds between nodes *i* and *j*, where $G$ is the set of all nodes, and $S_{i,j}$ is the amount of money sent from node $i$ to node $j$. In the case when $i = j, S_{i,i}$ is the amount of funds currently available to the node.

$$P_{i,j} = \frac{S_{i,j}}{\sum_{k \in G} S_{i,k}} \tag{1}$$

As expected, the sum of all outgoing probabilities is 1:

$$\sum_{k \in G} P_{i,k} = \sum_{k \in G} \frac{S_{i,k}}{\sum_{m \in G} S_{i,m}} = \frac{\sum_{k \in G} S_{i,k}}{\sum_{k \in G} S_{i,k}} = 1 \tag{2}$$

Algorithm 2 provides the pseudocode for scanning the blockchain.

**Input:** blockchain, target addresses
**Result:** Ledgers associated with the target addresses
**foreach** *block in blockchain* **do**
  **foreach** *transaction in block* **do**
    identify owner
    **foreach** *ouput in transaction* **do**
      handle new funds
    **end**
    **foreach** *ouput in transaction* **do**
      handle spending old funds
    **end**
  **end**
**end**

**Algorithm 2:** Model inference algorithm.

Identifying the owner was simple with the assumption that transactions are constructed with inputs owned by the same address. The sole address in the inputs was identified as the owner of the transaction. With the owner identified, it was trivial to identify the *change address*, since the model assumes the owner sends the change back to the originating address. Handling the outputs involves registering the new funds to the associated destination address. Each address has an associated ledger that tracks all the available funds. When an input is spent, the corresponding ledger entry is removed and re-associated with the destination. This makes it possible to track the amount of money sent to each address.

## 4.3 Results

## 4.4 Generation Results

Applying Algorithm 1 to the model shown in Figure 4 yielded the model shown in Figure 5.

Algorithm 1 was repeated 1,000 times in order to ensure the output probabilities were sufficiently accurate. This can be thought of dropping 1,000 coins into the 'plinko' game in order to ensure the probabilities are accurately represented. This produced a local blockchain that could exercise the inference capabilities of the model.

## 4.5 Inference Results

After scanning the blockchain as shown in Algorithm 2 and applying Equation 1 to the ledgers associated with each node, the model shown below in Figure 6 was generated.

Table 1 compares the probabilities in the theoretical model to the generated model and the inferred model, shown in Figure 4, Figure 5, and Figure 6 respectively. While the model in Figure 6 is almost identical to Figure 5, the main difference is that the inferred model was not initialized with a starting node, so the source of the funds was not identified.

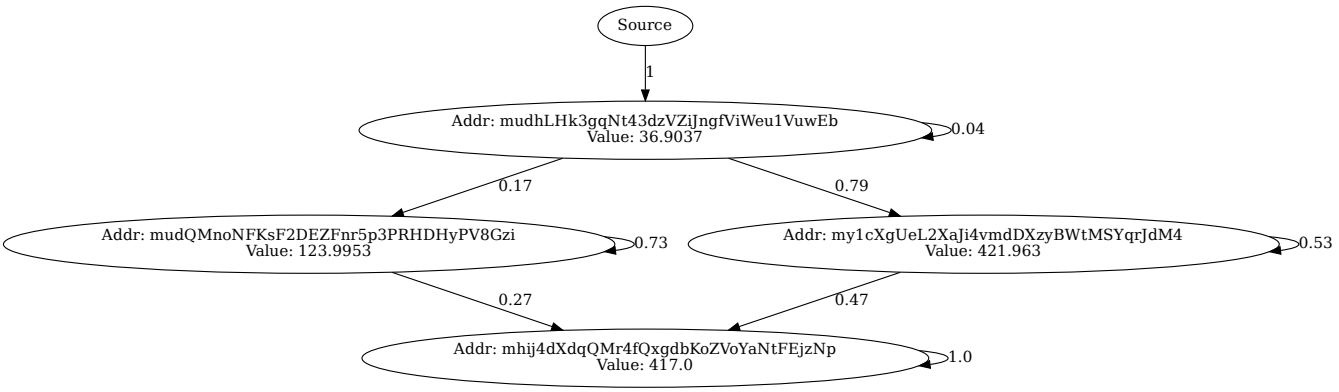


Figure 5: The simulated criminal enterprise.

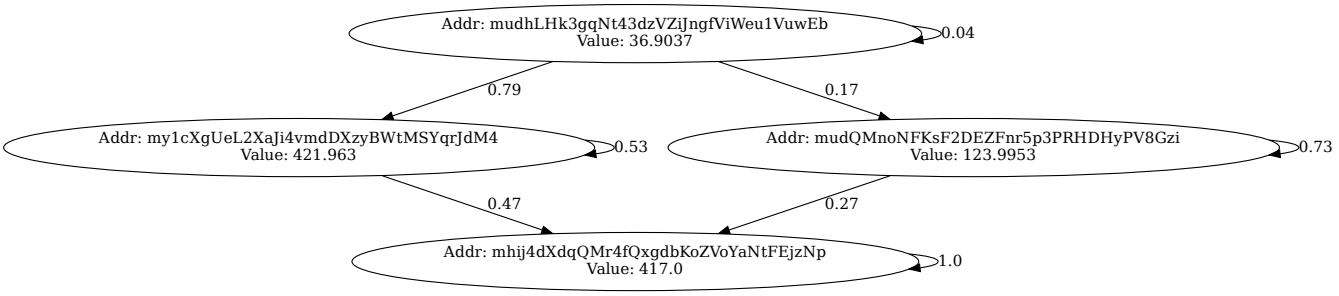


Figure 6: The inferred criminal enterprise.

Table 1: Simulated transition probabilities.

| Transition | Model | Generated | Inferred |
|---|---|---|---|
| $1 \to 1$ | 0.05 | 0.04 | 0.04 |
| $1 \to 2$ | 0.15 | 0.17 | 0.17 |
| $1 \to 4$ | 0.8 | 0.79 | 0.79 |
| $2 \to 2$ | 0.7 | 0.73 | 0.73 |
| $2 \to 3$ | 0.3 | 0.27 | 0.27 |
| $3 \to 3$ | 1 | 1 | 1 |
| $4 \to 3$ | 0.45 | 0.47 | 0.47 |
| $4 \to 4$ | 0.55 | 0.53 | 0.53 |

## 5 Conclusion and Future Work

A method for determining transition probabilities between Bitcoin addresses was presented. While simplifying assumptions were made in the model, they are supported by existing research [14, 4]. It was possible to generate a model similar to the theoretical model on the local Regression Testing blockchain. It was also possible to infer the generated model from the blockchain using the inference techniques presented herein.

Future work will address (1) more complex transaction, including CoinJoins and transactions with unique *change addresses*, (2) grouping multiple addresses used by the same user, and (3) applying these techniques to expose real criminal enterprises.

## References

[1] Anonymity, Dec 2015.
[2] Bitcoin developer examples, Sep 2015.
[3] Mixing service, Sep 2016.

[4] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun. Evaluating user privacy in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 34–51. Springer, 2013.

[5] S. Barber, X. Boyen, E. Shi, and E. Uzun. Bitter to better—how to make bitcoin a better currency. In *International Conference on Financial Cryptography and Data Security*, pages 399–414. Springer, 2012.

[6] Address format for pay-to-script-hash. Standard, Bitcoin, Oct. 2011.

[7] Y. Fu, Z. Jiay, L. Yu, X. Zhong, and R. Brooks. A covert data transport protocol. In *Malicious and Unwanted Software (MALWARE), 2016 11th International Conference on*, pages 1–8. IEEE, 2016.

[8] Y. Fu, L. Yu, O. Hambolu, I. Ozcelik, B. Husain, J. Sun, K. Sapra, D. Du, C. T. Beasley, and R. R. Brooks. Stealthy domain generation algorithms. *IEEE Transactions on Information Forensics and Security*, 12(6):1430–1443, 2017.

[9] KnowBe4. Aids trojan or pc cybor ransomware.

[10] C. Lu, J. M. Schwier, R. M. Craven, L. Yu, R. R. Brooks, and C. Griffin. A normalized statistical metric space for hidden markov models. *IEEE transactions on cybernetics*, 43(3):806–819, 2013.

[11] P. Misner. Customer guidance for wannacrypt attacks, May 2017.

[12] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[13] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *Security and Privacy, 2009 30th IEEE Symposium on*, pages 173–187. IEEE, 2009.

[14] M. Ober, S. Katzenbeisser, and K. Hamacher. Structure and anonymity of the bitcoin transaction graph. *Future internet*, 5(2):237–250, 2013.

[15] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[16] T. Ruffing, P. Moreno-Sanchez, and A. Kate. Coinshuffle: Practical decentralized coin mixing for bitcoin. In *European Symposium on Research in Computer Security*, pages 345–364. Springer, 2014.

[17] H. Ruiwen, D. Jianhua, and L. L. Lai. Reliability evaluation of communication-constrained protection systems using stochastic-flow network models. *IEEE Transactions on Smart Grid*, PP(99):1–1, 2017.

[18] R. C. Watkins, K. M. Reynolds, R. Demara, M. Georgiopoulos, A. Gonzalez, and R. Eaglin. Tracking dirty proceeds: exploring data mining technologies as tools to investigate money laundering. *Police Practice and Research*, 4(2):163–178, 2003.

[19] L. Yu, J. M. Schwier, R. M. Craven, R. R. Brooks, and C. Griffin. Inferring statistically significant hidden markov models. *IEEE Transactions on Knowledge and Data Engineering*, 25(7):1548–1558, 2013.

[20] X. Zhong, Y. Fu, L. Yu, R. Brooks, and G. K. Venayagamoorthy. Stealthy malware traffic-not as innocent as it looks. In *Malicious and Unwanted Software (MALWARE), 2015 10th International Conference on*, pages 110–116. IEEE, 2015.