Identifying Bitcoin users by transaction behavior

John V. Monaco
Pace University, 861 Bedford Rd, Pleasantville, NY, U.S.A

ABSTRACT

Digital currencies, such as Bitcoin, offer convenience and security to criminals operating in the black marketplace. Some Bitcoin marketplaces, such as Silk Road, even claim anonymity.[1] This claim contradicts the findings in this work, where long term transactional behavior is used to identify and verify account holders. Transaction features, such as timestamp, coin-flow, and connectivity, contribute to revealing the account-holder's identity. The time between successive transactions is the result of low-frequency effects, such as the desire purchase an item and daily schedule, as well as higher frequency effects, such as hardware and network latency. In addition to transaction time-intervals, dynamic network features of each transaction, such as coin flow and number of edge outputs and inputs, can also be used to identify account-holders. In this paper, we propose novel methodology for identifying and verifying Bitcoin users based on the observation of Bitcoin transactions over time. The behavior we attempt to quantify occurs in the social band of Newell's time scale and is low-frequency compared to other behavioral biometrics. A subset of Blockchain 230686 is analyzed, selecting users that initiated between 100 and 1000 unique transactions per month for at least 6 different months. This dataset shows evidence of being nonrandom and nonlinear, thus a dynamical systems approach is taken. Identification and verification accuracies are obtained using monthly Bitcoin samples. Outgoing transactions, as well as both outgoing and incoming transactions, are considered. Results show an inherent lack of anonymity by exploiting patterns in long-term transactional behavior.

Keywords: Bitcoin, behavioral biometrics, random time-interval, Wald-Wolfowitz test, nonlinear dynamics

1. INTRODUCTION

> You tunnel your traffic through a chain of anonymous proxy servers and begin doing business on the black market for the day. Illegal information is bought, sold, and traded with other black hats within your network. You've been using the digital currency *Bitcoin* to do business on the black market for several months now, utilizing a dedicated set of Bitcoin addresses that are in no way linked to your identity. But, each transaction you make contains a time stamp. It must, and there is no way of avoiding this. Over time, could these timestamps be used to reveal your identity? They might. The time the transaction took place is in some way a result of you, your schedule, daily life and activities, computer performance, network latency, and so on.
>
> You also make legitimate Bitcoin transactions. You use a mutually exclusive set of Bitcoin addresses to make legitimate purchases, possibly initiating up to several dozen transactions per day. There's no need to hide your identity for these. In fact, you even advertise some of your Bitcoin addresses to receive payments for private consulting jobs. It just so happens that the time at which these transactions occur are also a natural byproduct of you and may be used to reveal your alternate identity.

The preceding scenario is realized in this work, as timing information alone is shown to discriminate between users in the Bitcoin network. When other transaction information is utilized, the ability to discriminate between users increases further.

Traditionally, behavioral biometrics rely on a rich stream of information to identify or authenticate an individual. Well-defined methodology exists for keystroke, mouse, gait, mobile, and others.[2] Performance generally decreases as one moves up in Newell's Time Scale of Human Action.[3,4] This work considers events that occur in

For author information:
Email: jmonaco@pace.edu, Website: `http://vmonaco.com`

the social band by comparing month-long samples that contain several hundred Bitcoin transactions. The random time-interval (RTI) biometric, along with dynamic network features, are utilized. The RTI is the sequence of time-intervals between successive event timestamps. Given a series of timestamps t_i, the RTI can be obtained by taking

$$y_i = [t_1 - t_0, t_2 - t_1, \ldots, t_{n+1} - t_n]$$

The timestamps could arise from any timed recurring event. Sources of RTI could be key-blows[5] (where a single key is pressed), web history,[4] or mouse motion.[4] A nonlinear dynamical systems approach has successfully been applied to the RTI biometric, and a similar approach is taken here.

The purpose of this work is to demonstrate the following:

1. Bitcoin transaction histories from the same user are correlated and can be used to reveal that user's identity.
2. The RTI biometric relies on the observation of timestamps from a recurring event. Even if all other information is hidden or encrypted, the timestamps alone may be enough to identify a user. This is significant because the RTI biometric is truly ubiquitous.
3. Some proposed Bitcoin transaction features are more useful than others in discriminating between users.

Several properties of Bitcoin transactions are also discovered through statistical hypothesis testing. The data are shown to be largely nonrandom and nonlinear, justifying the nonlinear dynamical systems approach taken in this work.

This paper is organized as follows. Sections 2 and 3 provide background information pertaining to the Bitcoin network and related work, respectively. In Section 4, the methodology for preprocessing, phase space reconstruction, and classification is described. Experimental results are presented in Section 5. Statistical hypothesis testing for nonrandom and nonlinear behavior is performed in Section 6, as well as a preliminary evaluation of the stationarity of the data. Conclusions are drawn in Section 7, and Appendices A and B contain background information for some of the mathematical concepts.

2. THE BITCOIN NETWORK

Bitcoin* was first proposed in 2008[6] and implemented in 2009. It was originally designed to solve the double-spending problem and eliminate the central authority responsible for verifying the legitimacy of online transactions, such as in traditional credit and banking systems. What makes the Bitcoin network especially unique is that its entire transaction history is publicly available, giving one the opportunity to analyze economic and spending behavior.[7] The network operates in a peer-to-peer (P2P) fashion, where clients collectively verify the legitimacy of each transaction.

A Bitcoin address is similar to a bank account number. Anyone can generate an unlimited number of Bitcoin addresses; thus it is common for users to control multiple addresses. A Bitcoin transaction is a many-to-many function, able to have multiple inputs and outputs. Transactions are specified by the input and output addresses, the bitcoin value of each output address, and the time the transaction takes place. The cumulative bitcoin value from the inputs must be distributed to the outputs. Therefore, it is common for a sender to use an output address to receive change from a transaction. These addresses are normally referred to as *change addresses*[8]. For a transaction to be verified by the network, the sender must prove ownership of all the bitcoins associated with the input addresses. After the transaction is verified, the bitcoins are distributed to the output address as specified. Figure 1 shows two transactions with multiple outputs and multiple inputs, respectively.

A Bitcoin transaction can be interpreted as a behavioral event initiated on behalf of the sender†. The sender could be a single person, organization, or a business. Additionally, a transaction could be made directly or be the

*The proper noun *Bitcoin* refers to the peer-to-peer network, while lowercase *bitcoin* refers to the individual unit of currency.

†The terms *user, sender,* and *identity* are used interchangeably. They all refer to the owner of a set of Bitcoin addresses, whether a person or organization.

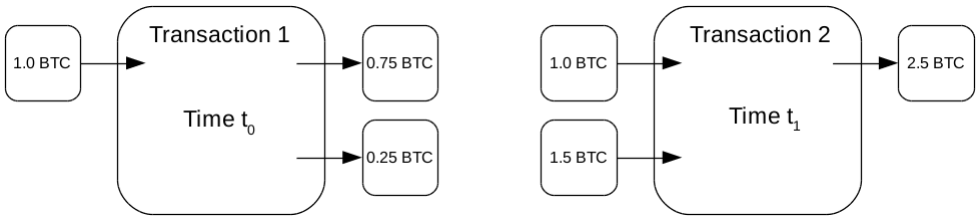


Figure 1: Bitcoin transactions with multiple outputs and multiple inputs. The total value of the inputs must be distributed to the outputs.

consequence of a computer program. In either scenario, the event occurs at some definite point in time and may be observed publicly. This is in contrast to many other behavioral biometrics that remain private by default. It is also an event-driven biometric, in contrast to those that are observed through a sensor with a fixed sampling rate, such as gait and eye movement.[9,10] When dealing with data from a sensor with a fixed sampling rate, user actions must be inferred by segmenting the data, such as identifying footsteps in gait or fixations and saccades in eye movement.

The anonymity that could potentially be achieved using Bitcoin is perhaps only a side effect of the ability to create an unlimited number of addresses that act similarly to bank account numbers.[8] It is recommended that users create a new address for every transaction, although many fail to do so. Additionally, multi-input transactions can be used to link addresses to the same owner since ownership of each address must be proven through public key cryptography. This was shown in a work where the *user network* was reconstructed[11] from the publicly available transaction network.

If a user follows certain protocols it may be possible to maintain some level of anonymity despite all transactions being made public. As the example in the introduction demonstrates, users could maintain mutually exclusive pools of addresses for different types of business dealings, and this is the scenario considered in this work. It is the dynamic transactional behavior of the user that we are interesting in quantifying. Over time, this transactional behavior becomes a leakage of information and may be used to deprive a user of anonymity.

3. RELATED WORK

The RTI biometric was proposed in Ref. 5, where users were asked to repeatedly hit a single key on a keyboard. The RTI was taken as the time-interval between successive key-blows. Without the rich set of features that is normally used in keystroke biometrics, the author took a dynamical systems approach, where the RTI samples were compared to each other in reconstructed phase space (RPS). With 40 users supplying 10 samples each, and 127 time-intervals per sample, an EER as low as 5.4% was obtained with a minimum class variance SVM (MCVSVM) classifier.

A similar approach was used to obtain results on several publicly available datasets, including mouse movement and web-browsing history.[4] Results were obtained on equally-sized datasets of 60 users, with 7 samples per user and 130 events (or 129 RTIs) per sample. Users could be identified by their web-browsing history timestamps with 16.2% accuracy. The web-browsing history samples spanned 22 days on average (a mean event frequency of $6.8 \times 10^{-5} Hz$ with 130 events per sample). These modest results show the potential to discriminate between users based on low-frequency social band behavior. It is interesting to note that timestamps alone provide this type of discrimination, which is clearly above chance prediction accuracy. The same algorithm also achieved the highest classification accuracy in the Second Eye Movements Identification and Verification Competition.[12] An approximation of the Wald-Wolfowitz (WW) test was used to compare large samples, where the WW test statistics were used as similarities in a kNN classifier. Because of its computational efficiency, the approximate multivariate Wald-Wolfowitz (AMWW) test[4] is also used in this work (see Appendix B).

A dynamical systems approach has been met with some success in other behavioral modalities, such as eye movement,[13] gait,[9] and voice.[14] In each, a suitable time-delay embedding is first determined, and samples are compared in RPS. The primary difference between Refs. 9, 13, 14 and this work is that Bitcoin transaction behavior is event-driven while eye movement, gait, and voice are all sampled from a sensor with fixed sampling rate.

Table 1: Bitcoin transaction attributes. A transaction can be described by its timestamp, connectivity (number of inputs and outputs to other users), and coin flow.

Attribute	Description
t_i	UNIX timestamp (in seconds) of the i^{th} transaction
u_i^{out}	number of unique outputs to other users
u_i^{in}	number of unique inputs from other users
b_i	bitcoin value with respect to the user (negative for loss, positive for gain)

There has also been some research from a non-biometrics perspective as to the level of anonymity that can be obtained in the Bitcoin network. Using network structure alone, it is possible to cluster addresses belonging to the same user.[11] This creates a *user network*, where vertices represent ownership of a set of addresses belonging to the same user, and edges represent transactions between users. This is only an approximation of the true user network, as it relies on the proof-of-ownership in multiple-input transactions. It fails for the scenario presented in the Introduction, in which users maintain mutually exclusive sets of addresses. Other heuristics for attributing multiple addresses to a single user have been proposed. In Ref. 8, the use of change addresses is exploited to help resolve address ownership. Creating a user network is a necessary preprocessing step to the methods in this work. The methods proposed in this work may also be used as a heuristic to merge nodes in the user network.

4. METHODOLOGY

The raw dataset used in this work is a subset of the user network compiled by Ivan Brugere[‡], using tools modified from Martin Harrigan.[11] Blockchain 230686 includes Bitcoin transactions made up to April 7, 2013, a network of 6.3M vertices and 37.4M edges. From the raw dataset, only users with between 100 and 1000 unique outgoing transactions per month, for at least separate 6 months (not necessarily consecutive), are kept. This yields 61 users, and the most recent 6 month-long samples for each user are kept, for a total of 366 samples. Restricting sample sizes to be within the same order of magnitude avoids discrimination by transaction frequency alone.

4.1 Transaction features

As a first preprocessing step, the transaction histories are transformed into the compact representation shown in Table 1. The attributes in Table 1 capture both timing information (t_i) and network information (u_i^{out}, u_i^{in}, and b_i) of a single transaction. That is, each transaction can be described by the timestamp, number of inputs/outputs to other users, and cumulative bitcoin value with respect to the user. For transactions with 0 outputs and a positive number of inputs, the bitcoin value would be strictly positive. This represents a payment to the user. Similarly, for transactions with 0 inputs and a positive number of outputs, the bitcoin value would be strictly negative, representative of a payment sent from the user. A transaction with 0 outputs and 0 inputs would have a 0 coin value, and represents a self-loop, i.e a transaction between addresses belonging to the same user. This may occur if a user chooses to consolidate bitcoin to a single address, or distribute bitcoin to multiple addresses.

From the representation in Table 1, we can extract several features that may characterize some aspect of transaction behavior over time. Additionally, we may consider only outgoing transactions (where the number of outputs is nonnegative, i.e. $u_i^{out} > 0$) or both outgoing and incoming transactions from a user. Outgoing transactions capture the behavior of the user, since it is the user who initiates the transaction, while outgoing and incoming transactions together capture both the user's behavior and behavior of the user's clientèle. The following 6 features are extracted from outgoing and both outgoing and incoming transactions to obtain 12 time series for each monthly sample:

Random time-interval (RTI) The time interval, in seconds, between successive transaction timestamps. Values are always nonnegative and represent the velocity of transaction behavior. The RTI is calculated as follows.

[‡]`http://compbio.cs.uic.edu/data/bitcoin/`

$$RTI_i = t_i - t_{i-1}$$

Hour of day (HOD) The hour of the day {1...24} that the transaction took place. This may capture scheduling behavior without the finer granularity of second-precision. Values increase in a step-wise fashion (non-decreasing from the start to end of each day). The HOD can be determined by performing integer division (// operator) on the timestamp followed by a modulus to get the time of day.

$$HOD_i = t_i//3600 \bmod 24$$

Time of hour (TOH) The TOH is the number of seconds elapsed since the start of the hour, calculated by taking a modulus of the transaction timestamp.

$$TOH_i = t_i \bmod 3600$$

Time of day (TOD) The TOD is the number of seconds elapsed since the start of the day. This is similar to the HOD, with finer granularity. The TOD is calculated by taking a modulus of the timestamp.

$$TOD_i = t_i \bmod 86400$$

Coin flow (CF) The user's bitcoin value throughput may be characterized by the coin flow. The coin flow is the cumulative bitcoin value of the inputs (excluding the user) minus the cumulative bitcoin value of the outputs (again, excluding the user). For loss, it is negative, and for gain, it is positive. In a self loop, where a transaction is made between addresses belonging to the same user, the CF would be 0. The CF is negative for strictly outgoing (no self loops) transactions and positive for strictly incoming transactions

$$CF_i = b_i$$

Input/output balance (IOB) The number of inputs from other users minus the number of outputs to other users. Self-loops would have an IOB of 0, as would an outgoing transaction with one output to another user and change returned to the sending user. Note that this considers edges to other users and not other addresses.

$$IOB_i = u_i^{in} - u_i^{out}$$

With each feature taken on the month-long transaction samples, we end up with 12 time series for each sample (6 features for outgoing transactions only and 6 features for outgoing and incoming transactions). In a sample with n outgoing transactions, the HOD, TOH, TOD, CF, and IOB time series would be of length n, and the RTI would be of length $n-1$. In the rest of the paper, the preceding acronyms will refer to features taken on outgoing transactions only. Features taken on both outgoing and incoming transactions are denoted by $RTI^{(out/in)}$, $HOD^{(out/in)}$, etc, or $\{RTI, HOD\}^{(out/in)}$ to refer to several features. For example, $RTI, \{CF, HOD\}^{(out/in)}$ refers to the RTI feature time series for outgoing transactions and the CF and HOD feature time series for outgoing and incoming transactions.

4.2 Phase space reconstruction

This work stands on the celebrated Takens' Theorem,[15] which states that a proper time-delay embedding of a one-dimensional time series may reveal the structure of the underlying dynamical system. Time-delay embedding requires the estimation of an embedding dimension, d_e, and time lag, τ, with well-researched heuristics for determining each.[16,17] For a given d_e and τ, the embedding procedure transforms the original time series, y_i, into a series of embedded vectors, $\mathbf{y}_i$:

$$\mathbf{y}_i = [y_i, y_{i-\tau}, y_{i-2\tau}, \ldots, y_{i-(d_e-1)\tau}]$$

Embedding parameters are determined for each feature time series using the mutual information (MI) to estimate τ and the method of false nearest neighbors (FNN) to estimate d_e. The procedure is as follows. For each feature and each sample, τ and d_e are determined using the heuristics described below. The global embedding parameters for each feature are then taken to be the parameters with the highest frequency, i.e. the mode, over all samples.

The MI, given in Equation (1), serves as a good heuristic for selecting τ.

$$I(T) = \sum_{n=1}^{N} P(y_n, y_{n+T}) \log_2 \frac{P(y_n, y_{n+T})}{P(y_n)P(y_{n+T})} \tag{1}$$

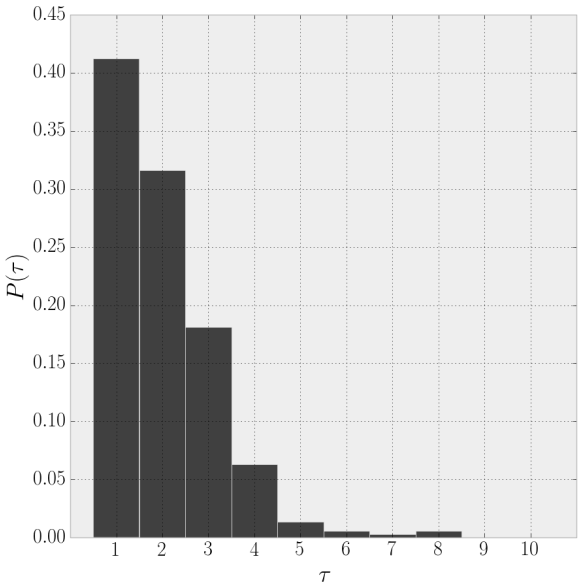
Generally, the lag at which the first local minimum occurs will yield a good embedding. This ensures that the lagged data are correlated, but not too correlated much.[17] As suggested,[16] in cases where there is no local minimum up to some T_{max}, τ is taken to be T where $I(T)/I(0) \approx \frac{1}{5}$. We use $T_{max} = 10$ to avoid unnecessarily large lags.

Next, the method of FNN is used to determine embedding dimension. Put simply, two embedded vectors, $\mathbf{y}_i$ and $\mathbf{y}_j$ are FNN if they are neighboring in dimension d_e and become distant in dimension $d_e + 1$. The correct dimension can be determined by increasing d_e until the proportion of FNN falls below some threshold. This ensures that the underlying dynamical system is unfolded and not projected into a dimension which is too small causing an unnecessarily large number of FNN. To classify two vectors as FNN, consider the normalized increase in distance[17] when going from d_e to $d_e + 1$, where $\mathbf{y}_i^{NN}$ is the nearest neighbor to $\mathbf{y}_i$, and i^{NN} is the index of $\mathbf{y}_i^{NN}$:

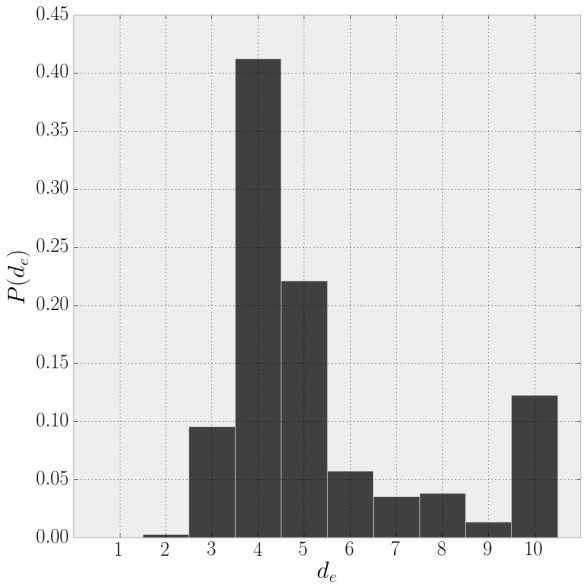
$$R = \frac{\left| y_{i-(d_e+1)\tau} - y_{i^{NN}-(d_e+1)\tau} \right|}{\| \mathbf{y}_i - \mathbf{y}_i^{NN} \|} \tag{2}$$

When R is above some threshold, R_{thresh}, then $\mathbf{y}_i$ and $\mathbf{y}_i^{NN}$ are considered to be FNN. We use $R_{thresh} = 15$, a value suggested to give reasonable results in many situations.[17] The embedding dimension is chosen to be the smallest d_e that results in less than 5% FNN. If the proportion of FNN does not fall below 5% up to $d_{e\,max} = 10$, then d_e is take to be $d_{e\,max}$. An upper bound on the embedding dimension is necessary since noisy samples are prone to high embedding dimensions.[17]

The densities of d_e and τ for the 366 *RTI* feature time series are shown in Figure 2. The high density at $d_e = 10$ indicates there are many samples that desire a much higher embedding dimension, due to either noise or an inherently high-dimensional underlying system. The embedding parameters determined for each feature are given in Table 2. These embedding parameters are used in the rest of this paper anywhere the dataset requires time-delay embedding. An example of the *RTI* time series from a randomly selected sample and the embedding of the *RTI* can be seen in Figure 4. Note that the determination of embedding parameters does not require knowing the class labels, and this can be performed as a preprocessing step before classifying an unlabeled sample.



(a) Distribution of τ for all RTI samples. For each sample, τ is selected where the first minimum of mutual information occurs.



(b) Distribution of d_e for all RTI samples. For each sample, d_e is selected where the proportion of FNN drops below 5%.

Figure 2: Embedding parameter densities for RTI, using MI and FNN as criteria for parameter selection.

Table 2: Embedding parameters for each feature. Parameters are determined by choosing d_e and τ with the highest frequency, using mutual information and the method of false nearest neighbors as selection criteria.

	Outgoing						Outgoing/Incoming					
	RTI	HOD	TOH	TOD	CF	IOB	RTI	HOD	TOH	TOD	CF	IOB
d_e	4	4	3	4	5	10	5	4	4	4	10	10
τ	1	4	2	4	1	2	1	9	2	9	1	2

4.3 Classification

The multivariate Wald-Wolfowitz test[18] is an extension of the original Wald-Wolfowitz runs test,[19] a nonparametric test to determine whether two sets of observations originated from the same distribution. To compute the WW statistic requires $O(N^3)$ time, although an approximation can give comparable results[4] and runs in $O(nk\log(Nk))$, where k is the number of neighbors considered when constructing the minimum spanning tree (MST). For a description of the AMWW, see Appendix B.

The classification procedure works as follows. For a single feature time series from a query sample, the AMWW test statistic is computed between the unlabeled query sample and the labeled reference samples. The reference samples are ranked by their similarity to the query sample. A score for each identity is obtained by taking the normalized rank sum of the identity samples, up to some maximum rank k. This is similar to a traditional kNN classifier, which simply uses the proportion of k neighboring samples for each class.

To illustrate how the score for each identity in the database is obtained, consider a small example. The RTI feature time series is obtained for all samples, including an unlabeled query sample q. The AMWW test statistic is calculated between q and reference samples r_1, r_2, r_3, r_4, r_5 that have identities a, a, b, b, c. The AMWW measures the similarity between two samples. Suppose the test statistics between the query and reference samples are $r_1 = 0.1, r_2 = 0.02, r_3 = 1.2, r_4 = 0.09, r_5 = 0.04$. The reference samples are ranked by the test statistics in order of decreasing similarity to the query sample, and we get b, a, b, c, a as the rank labels. If we let $k = 4$, then the scores for each identity will be $b = \frac{4+2}{4+3} = \frac{6}{7}$, $a = \frac{3}{4+3} = \frac{3}{7}$, $c = \frac{1}{4}$. This is because identities b and a each have 2 samples in the reference set. Therefore the maximum obtainable rank sum they can get is one sample with rank 4 and another sample with rank 3. Identity c only has one sample in the reference set and has a maximum rank

sum of 4. Anything beyond rank 4 is ignored because $k = 4$.

To combine multiple features, each feature is used in a separate linear-weighted classifier as described above. The scores from each classifier are summed to produce a fused score for the unknown sample. A summation of classifier output scores is thought to be reasonable robust, especially if independence between classifiers is assumed.[20] For identification, the identity with the highest score is chosen. For verification, the score of the claimed identity is compared to a threshold.

The identification and verification procedure can be summarized as follows. Let s_i be the classifier output score for identity i when classifying an unknown sample.

1. Given a feature time series for an unlabeled query sample, and reference samples with corresponding features, determine the embedding parameters using all the samples, as described in Section 4.2. Use the embedding parameters in to embed everything.
2. Calculate the AMWW test statistic between the query sample and the reference samples. Using the test statistics as similarity measures, determine the normalized rank sum for each identity, up to rank k. This is the classifier output score for each identity.
3. To combine features, perform Steps 1 and 2 for each feature. Take the sum of classifier output scores for each identity.
4. For classification: take the identity with the maximum score as the label for the query sample.
5. For authentication: given claimed identity c, use a threshold parameter $s_{thresh} \in [0 \dots 1]$ to obtain a verification decision, where $s_c \geq s_{thresh}$ indicates a positive decision.

Identification and verification errors rates are calculated in the usual way. The identification accuracy (ACC1) is the proportion of correctly classified samples. Verification decisions are tallied to obtain the false acceptance rate (FAR) and false rejection rate (FRR) for each $s_{thresh} \in [0 \dots 1]$. The verification accuracy is reported as the equal error rate (EER), the point on the receiving operator characteristic (ROC) curve where FAR = FRR.

5. EXPERIMENTAL RESULTS

Leave-one-out cross fold validation (LOOCV) is used to obtain experimental results. Thus, for each query sample left out, 365 samples remain in the reference set. In obtaining the ACC1, 366 samples are identified. To get the EER, $61 \times 61 \times 6$ verifications are performed, with 61×6 mated pairs and $61 \times 60 \times 6$ non-mated pairs.

For the first set of experiments, k is varied from 1 to 365 for each feature of outgoing transactions only. Results are shown in Figure 3.

Most features provide optimal identification with $k \approx \sqrt{365}$. This is consistent with the empirical rule-of-thumb for a traditional kNN classifier: selecting to k to be the square root of the number of samples in the reference set.[21] Verification error rates experience a similar trend, usually finding minimum error rates with slightly larger k. Following the rule-of-thumb, k is chosen to be 20 in all of the following results. Table 3 contains classification and authentication results for each feature on outgoing and both outgoing and incoming transactions, while Table 4 shows results for multiple features.

As seen in Table 3, *RTI* performs well for outgoing and both outgoing and incoming transactions. These results are consistent with results obtained on similarly-sized RTI datasets[4] with ACC1 ranging from about 30% to 40%. The *TOH* is only slightly above chance, leading one to believe that the resolution at which interesting behavior occurs is not the hour-scale, but perhaps the day-scale. The performance of all the timing features (*HOD*, *TOH*, *TOD*) degrades significantly when both outgoing and incoming transactions are considered together, yet the performance of the time interval feature, *RTI*, increases when outgoing and incoming transactions are considered.

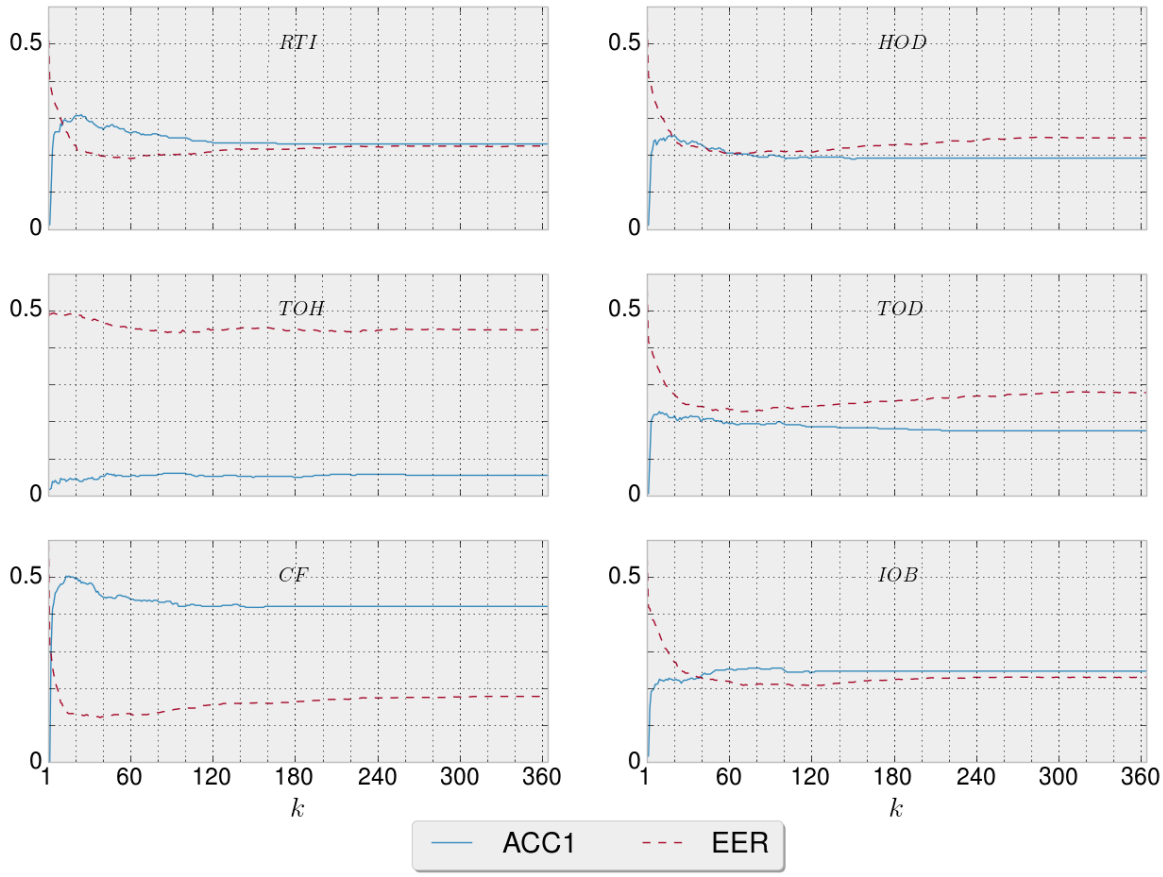


Figure 3: Experimental results obtained for each feature extracted from outgoing transactions only, varying k from 1 to 365. The ACC1 and EER experience optima close to $\sqrt{365} \approx 20$, consistent with the empirical rule-of-thumb[21] in choosing k.

Table 3: Experimental results for each feature, with $k = 20$, obtained by a linear-weighted kNN classifier. The performance of RTI increases when both outgoing and incoming transactions are considered, an indication that the user's clientèle offers valuable information about the identity of the user.

	Outgoing		Outgoing/Incoming	
	ACC1(%)	EER(%)	ACC1(%)	EER(%)
RTI	30.3	22.6	31.1	17.8
HOD	25.1	24.8	3.3	49.0
TOH	4.4	48.8	3.9	47.2
TOD	21.0	27.5	12.8	38.0
CF	49.7	13.1	56.3	8.4
IOB	22.1	27.5	43.2	13.1

Table 4: Experimental results obtained for combined features in a fusion of linear-weighted kNN classifiers with $k = 20$. Items inside $\{\cdot\}^{(out/in)}$ indicate features taken on both outgoing and incoming transactions; elsewhere features are for outgoing transactions only.

	ACC1(%)	EER(%)
RTI, HOD	32.0	18.3
HOD, TOH	21.3	27.9
TOD, CF	53.6	11.6
RTI, CF	60.7	12.6
CF, IOB	54.1	11.0
RTI, HOD, IOB	43.2	15.4
RTI, HOD, CF	58.5	10.7

	ACC1(%)	EER(%)
RTI, HOD, CF, IOB	62.0	10.1
$\{RTI, HOD, CF, IOB\}^{(out/in)}$	65.8	9.0
$\{CF, IOB\}^{(out/in)}$	64.2	7.7
$RTI, IOB^{(out/in)}$	51.6	11.0
$RTI, HOD, IOB^{(out/in)}$	55.7	11.6
$HOD, \{RTI, CF, IOB\}^{(out/in)}$	73.5	6.7
$RTI, HOD, \{CF, IOB\}^{(out/in)}$	76.0	6.8

With 12 different features (6 features for outgoing and 6 for outgoing and incoming transactions), there are $12^2 - 1$ possible ways to combine them. When several features are used in a fusion of linear weighted kNN classifiers, the performance generally increases. Table 4 shows some selected results from the $12^2 - 1$ feature combinations. The best performance for timing-only information is seen in $RTI, HOD, IOB^{(out/in)}$ with 55.7% ACC1 and 11.6% EER. The coin flow seems to be a significant contributor to discriminating behavior, as performance always increases when coin flow is considered. The best identification rate is obtained by $RTI, HOD, \{CF, IOB\}^{(out/in)}$ with 76% accuracy, while the best verification accuracy is seen in $HOD, \{RTI, CF, IOB\}^{(out/in)}$, at 6.7% EER. Generally, timing information is more valuable if only outgoing transactions are considered, while network information (coin flow and input/output balance) are better indicators when both outgoing and incoming transactions are examined.

6. TESTS FOR RANDOMNESS, NONLINEARITY, AND STATIONARITY

It is important to understand the nature of the Bitcoin dataset before attempting to build a classifier for identifying users. The following analysis will serve as justification for the methods proposed and perhaps offer some insight to properties of the transaction histories. The method of surrogate data testing[22] can be used to determine whether a time series is nonrandom or nonlinear. The process is as follows. A discriminative test statistic is first calculated for the original time series. The statistic is then calculated for several surrogate time series that exhibit behavior consistent with a null hypothesis (e.g. the data is random). The original statistic is then compared to the surrogate test statistics to determine whether the null hypothesis should be rejected. The tests serve as a statistical proof by contradiction and, therefore, only give evidence for the alternate hypothesis.

For each sample, three different tests are performed under each null hypothesis using three different test statistics: the MI, given in Equation (1), the proportion of FNN, as described in Section 4.2, and the nonlinear prediction error (PE). The PE can be computed using the *drop one out* method.[17,23] In this work, a variation of the usual PE is used. See Appendix A for a description of PE calculation.

A confidence value, ρ, is determined by a rank statistic[22§]. Let q_0 be the statistic obtained from the original time series, and q_i be the values of the test statistic obtained from n surrogates. For a left-sided test, $\rho = \frac{I\{q_0 < q_i\}}{n}$, where I is the indicator function and $I\{q_0 < q_i\}$ is the number of surrogates for which q_i is greater than q_0 (right-sided tests are calculated similarly). For a two-sided test, $\rho = 2 \times \frac{\min(I\{q_0 < q_i\}, I\{q_0 > q_i\})}{n}$. In all tests, H_0 is rejected with at least 90% confidence (i.e. H_0 is rejected if $\rho \leq 0.10$), and 100 surrogates are generated for each test. Each feature time series is tested for each sample on outgoing transactions only. Thus, a total of 6588 tests are performed: 3 discriminative statistics $\times$ 6 features $\times$ 366 samples.

6.1 Test for randomness

First, consider the null hypothesis H_0^r, that a sample shows no temporal correlation at all (i.e. y_i can be completely described by an independent and identically distributed random variable). The results of this test

[§]If we assumed q_i followed a Gaussian distribution, then we could compute the confidence intervals in the usual way.[24] However we do not know, nor will attempt to discover the distribution of q_i in each test.

Table 5: Proportion of samples that reject the null hypothesis H_0^r, that the samples are random noise with the same empirical distribution as the original data, at 90% confidence. Surrogates are generated by shuffling the values in the original time series, and ρ is found using a rank-statistic. PE and MI are both one-sided tests and FNN is two-sided.

f	Prediction error	Mutual information	False nearest neighbors	At least 1	All 3
RTI	0.34	0.45	0.37	0.69	0.11
HOD	0.96	1.00	0.97	1.00	0.93
TOH	0.47	0.71	0.52	0.85	0.30
TOD	0.96	1.00	0.96	1.00	0.92
CF	0.28	0.52	0.28	0.68	0.08
IOB	0.45	0.75	0.46	0.86	0.24
Avg	0.58	0.74	0.59	0.85	0.43

Table 6: Proportion of samples that reject the null hypothesis H_0^{nl}, that the samples are the result of linear stochastic process, at 90% confidence in a two-sided test. Surrogates are generated using the AATFT algorithm with a cutoff frequency of $0Hz$, and ρ is determined by a rank-statistic.

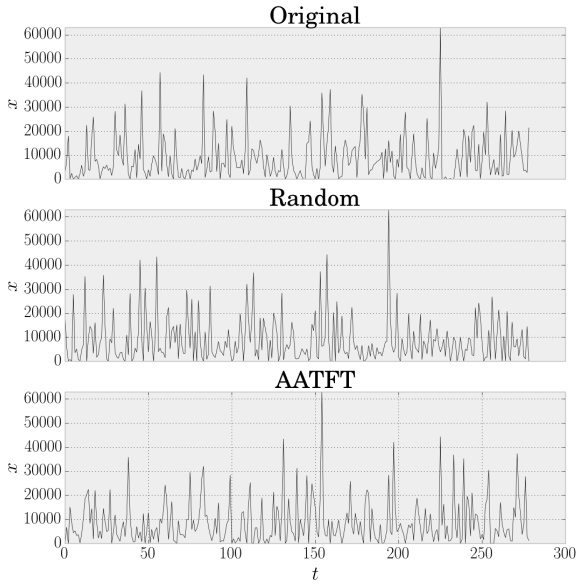
f	Prediction error	Mutual information	False nearest neighbors	At least 1	All 3
RTI	0.23	0.31	0.36	0.61	0.06
HOD	0.78	1.00	0.96	1.00	0.74
TOH	0.25	0.66	0.53	0.82	0.16
TOD	0.79	1.00	0.95	1.00	0.74
CF	0.19	0.44	0.26	0.61	0.05
IOB	0.28	0.48	0.36	0.66	0.13
Avg	0.42	0.65	0.57	0.78	0.31

will indicate whether the data has any structure at all, or we are just dealing with uncorrelated noise. Surrogates can be generated by random permutations of the original time series. This preserves the empirical distribution of the time series while destroying any temporal correlation, which is consistent with the null hypothesis. An example H_0^r surrogate is shown in Figure 4a. Tests with PE and MI are one-sided, since the surrogates can always be expected to have higher prediction errors and lower mutual information than the original time series. The test with FNN is two-sided, since it is not clear whether the proportion of FNN in the surrogates would always be less or greater than that of the original time series.

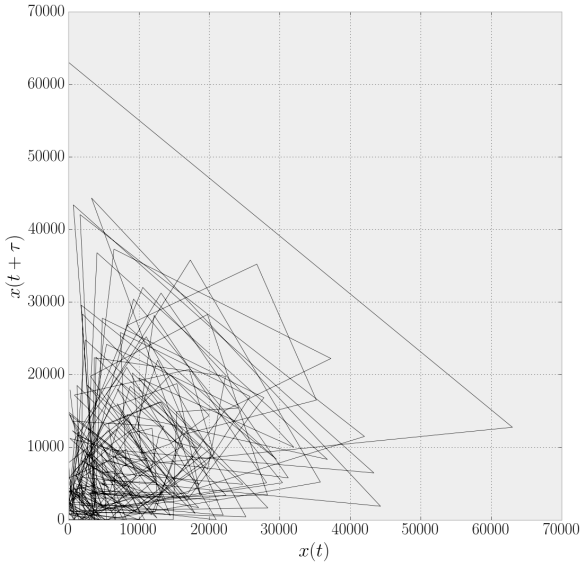
The proportion of samples for which H_0^r is rejected for each feature is shown in Table 5. The HOD is clearly not random, since the values must increase monotonically during each day. Other features indicate a modest proportion of the dataset is nonrandom and probably worth exploring further. It is interesting to not that very few CF time series pass all three tests.

6.2 Test for nonlinearity

To test for non-linearity, we let H_0^{nl} be the null hypothesis that data came from a linear stochastic process (i.e. the data is completely described its autocorrelation or power spectrum). Surrogate samples are generated using the method of *Amplitude Adjusted Truncated Fourier Transform* (AATFT).[25] This method preserves both the power spectrum and amplitude distribution of the original time series, a drawback sometimes encountered when generating surrogates with the *Amplitude Adjusted Fourier Transform* (AAFT). A cutoff frequency of $0Hz$ is used in the algorithm. An example surrogate for H_0^{nl} and the original time series are shown in Figure 4a. Again, 100 surrogates are generated in each test, and each feature is tested for each sample in the database on outgoing transactions only. The ρ-value is calculated using a rank statistic similar as above. All tests for H_0^{nl} are two-sided, since it is not obvious whether any of the statistics calculated on the surrogate time series would be strictly less than or greater than that of the original time series. Again, the null hypothesis is rejected in a modest proportion of samples for each feature.



(a) Example of an original RTI time series (top) and surrogates generated for H_0^r (middle) and H_0^{nl} (bottom).



(b) Example of an embedded RTI sample, with $d_e = 4$ and $\tau = 1$. Only the first two dimensions are shown.

Figure 4: Examples of an original RTI time series, H_0^r and H_0^{nl} surrogates, and reconstructed phase space using the embedding parameters found in Section 4.2. The H_0^r surrogate is a random permutation, and the H_0^{nl} surrogate is generated using the AATFT.[25] The amplitude distribution and spectrum of original time series are both preserved in the AATFT surrogate, while all time correlation is destroyed in the random permutation surrogate.

6.3 Test for stationarity

Using PE, it is of interest to determine whether the transaction data are stationary. Out-of-sample predictions can be made by using different samples for "training" and "testing" (see Appendix A). If the data are stationary, the prediction error should remain relatively constant throughout time. If the prediction error increases or decreases with time between samples, then the data may show signs of non-stationarity.[23]

For each feature on outgoing transactions, a PE matrix is calculated for each user. The PE is calculated for every combination of sample pairs within each user ($6 \times 6 = 36$ PE values, where 6 of these are in-sample and 30 are out-sample predictions). The error matrix is then normalized using the minimum and maximum PE, so that all prediction errors are between 0 and 1. This magnifies the difference in errors, if any, between samples and places every user PE on the same scale. The element-wise mean for all users is calculated, to get the PE matrix for each feature. Results are shown in Figure 5. In-sample prediction errors lie along the diagonal, while out-sample prediction errors are everywhere else.

The data seems mostly stationary, as the relative prediction error does not increase when moving away from the diagonal. It is interesting to note that the timing features (HOD, TOH, TOD) generally experience a higher PE for in-sample predictions (along the diagonal). The coin flow is somewhat non-stationary, as the darker regions appear to be in areas where the training and prediction samples are far apart (upper left and lower right corners). This may be attributed to the increased global coin flow that comes with increased adoption. Despite this, the methods presented in this work assume stationarity of the data. A full treatment of the Bitcoin dataset, in which non-stationarity is accounted for, is left for future work.

7. CONCLUSION

Bitcoin transaction behavior is largely nonrandom and somewhat nonlinear. This finding is the basis for the methods described in this work, which compared Bitcoin transaction features in reconstructed phase space. Experimental results indicate that behavioral patterns observed over time can be used to deprive a user of

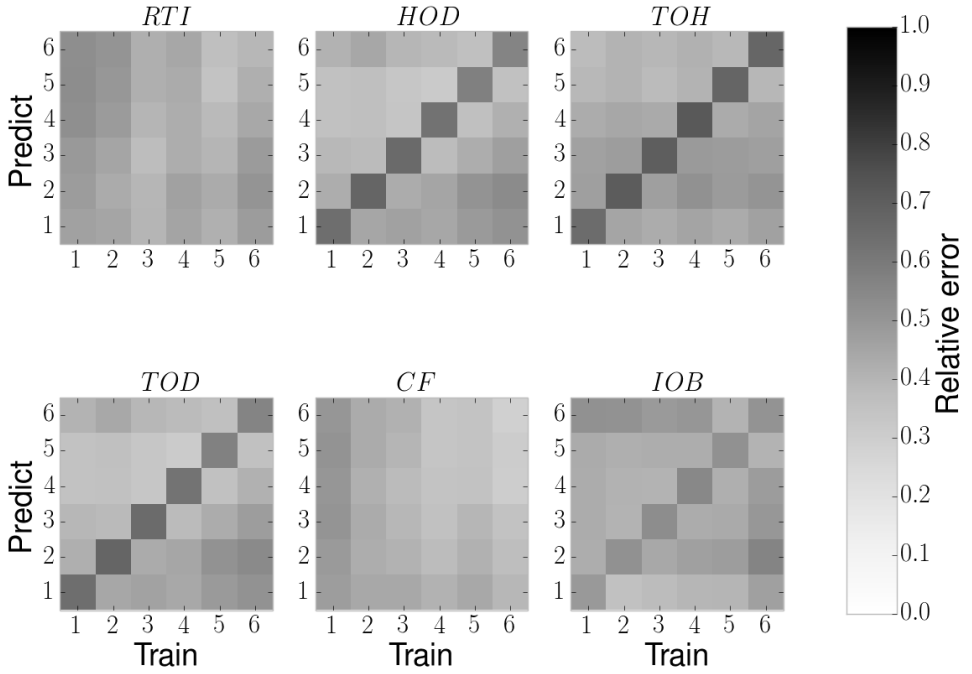


Figure 5: Mean normalized prediction error obtained for every pair of samples within each user. In-sample predictions lie along the diagonals, and out-sample prediction are everywhere else. Non-stationarity is seen when the relative error increases when moving away from the main diagonal.

anonymity. This is both good and bad news. It may allow users to be tracked, despite measures taken to remain anonymous. It also paves the way for applications that may increase the security of digital currencies, such as fraud detection and the identification of cyber criminals.

This work proposed several discriminative features for Bitcoin transactions. Both timing and complex network information play a critical role in capturing the dynamics of a user's behavior. The methods described in this work may be used as an additional heuristic in constructing the Bitcoin user network from the transaction network. At the same time, the classification procedure described is naive from a pattern recognition viewpoint; higher accuracies may be obtained with more sophisticated classification algorithms, such as boosting to combine feature similarities as opposed to summation.

Encryption and anonymity alone may no longer be sufficient for concealing one's identity. Human behavior is complex, but in some cases, predictable. Over time, behavior is a leak of information. Obfuscation against RTI biometrics may help conceal one's identity, and in the future we may see techniques like this being employed as more RTI sources are exploited as behavioral biometrics. We have already seen patches released to SSH in an attempt to mitigate text-reconstruction attacks on the timing information from typing behavior,[26] though timestamp obfuscation in a protocol such as Bitcoin may not be possible. Further work is needed to evaluate the capacity of this very immature behavioral biometric modality. The use of time-intervals as a behavioral biometric is truly ubiquitous, as billions of people are connected to the Internet and in some way or another, generate events with observable timestamps; from general actions such as web browsing and email messages to application specific events, such as Bitcoin transactions and repository commits. The methods presented here are a preliminary analysis of just one public source of information, moving one step closer to quantifying human behavior.

APPENDIX A. NONLINEAR PREDICTION ERROR

Given a time series y_i and predictions $\hat{y}_i$, the root mean squared (rms) prediction error is:

$$e = \sqrt{\langle (\hat{y}_n - y_n)^2 \rangle}$$

where $\langle(\hat{y}_n - y_n)^2\rangle$ is the expected value of the squared prediction error. In a deterministic nonlinear dynamical system, the system history history can be used to predict future states. The concept was originally proposed by Lorenz.[27]

To predict y_{i+1}, consider an the embedded vector $\mathbf{y}_i$ in $\mathbf{R}^{d_e}$. Let the k nearest neighbors to $\mathbf{y}_i$ be $\mathbf{y}_i^{NN}$, and let $\mathbf{y}_{i+1}^{NN}$ be the successors to each of the neighbors $\mathbf{y}_i^{NN}$. We can predict y_{i+1} using each of the successors $\mathbf{y}_{i+1}^{NN}$ by taking a weighted average of y_{i+1}^{NN}. In this work, predictions are made by assigning the y_{i+1}^{NN} quadratically decreasing weights when $\mathbf{y}_i^{NN}$ are ordered by their distance from $\mathbf{y}_i$. The first neighbor is assigned a weight of $\frac{k^2}{\frac{k^3}{3}+\frac{k^2}{2}+\frac{k}{6}}$, the second a weight of $\frac{(k-1)^2}{\frac{k^3}{3}+\frac{k^2}{2}+\frac{k}{6}}$, and so on. This places more emphasis on close neighbors, while retaining some influence from distance neighbors. The quadratically decreasing weights were chosen to be robust with sample size ranging from 100 to 1000. For small samples, there are relatively few neighbors $\mathbf{y}_i^{NN}$ that may give good predictions, and distant $\mathbf{x}_i^{NN}$ generally give worse predictions. Exponentially decreasing weights may be appropriate in this case. On the other hand, with large samples, there are more good $\mathbf{y}_i^{NN}$, and it would be desirable to assign these linearly decreasing weights. The quadratically decreasing weights are a compromise between linearly decreasing and exponentially decreasing weights.

The out-sample PE can also be obtained by determining $\mathbf{y}_i^{NN}$ from another sample. To make out-sample predictions, take $\dot{\mathbf{y}}_i^{NN}$ (and consequently $\dot{\mathbf{y}}_{i+1}^{NN}$) from a different sample $\dot{y}_i$, making predictions similarly as above. In-sample predictions are made by taking $\mathbf{y}_i^{NN}$ and $\mathbf{y}_{i+1}^{NN}$ from the same sample.

APPENDIX B. APPROXIMATE MULTIVARIATE WALD-WOLFOWITZ TEST

The multivariate Wald-Wolfowitz (WW) test[18] is an extension of the nonparametric Wald-Wolfowitz runs test.[19] Computing the WW test statistic relies on the construction of the minimum spanning tree (MST), requiring $O(N^3)$ time, where $N = m + n$, and m and n are the sizes of the two samples x_i and y_i respectively. For large data, this is computationally expensive; thus an approximation that yields similar results is used.

The approximate multivariate Wald-Wolfowitz[4] (AMWW) test works as follows: first select a proper embedding for x_i and y_i as described in Section 4.2 to get embedded vectors $\mathbf{x}_i$ and $\mathbf{y}_i$ in dimension d_e. Use a k-d tree to efficiently find the k nearest neighbors to each vector, and construct the approximate MST in $\mathbf{R}^{d_e}$. As long as k is large enough and the data is well-behaved, the approximate MST will be similar to the true MST, while time cost in reduced to $O(Nk\log(Nk))$. The AMWW statistic is then computed similar to the WW: count the number of runs, R, in the MST, where a run is a segment of the tree that connects vectors from the same sample. The expected number of runs is given in Equation (3) and W can be computed from Equation (4).

$$E(R) = \frac{2mn}{N} + 1 \tag{3}$$

$$W = \frac{R - \frac{2mn}{N} - 1}{\left(\frac{2mn(2mn-N)}{N^2(N-1)}\right)^{\frac{1}{2}}} \tag{4}$$

ACKNOWLEDGMENTS

The author would like to thank the US Department of Defense Information Assurance Scholarship Program (IASP) for support provided in conducting this research. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the US government.

REFERENCES

[1] Christin, N., "Traveling the silk road: A measurement analysis of a large anonymous online marketplace," in [*Proceedings of the 22nd international conference on World Wide Web*], 213–224, International World Wide Web Conferences Steering Committee (2013).

[2] Yampolskiy, R. V. and Govindaraju, V., "Behavioural biometrics: a survey and classification," *International Journal of Biometrics* **1**(1), 81–113 (2008).

[3] Newell, A., [*Unified theories of cognition*], Harvard University Press (1994).
[4] Monaco, J. V., "Classification and authentication of one-dimensional behavioral biometrics," in [*Proceedings of the 2014 International Joint Conference on Biometrics (IJCB)*], IEEE,IAPR (2014).
[5] Laskaris, N. A., Zafeiriou, S. P., and Garefa, L., "Use of random time-intervals (rtis) generation for biometric verification," *Pattern Recognition* **42**(11), 2787–2796 (2009).
[6] Nakamoto, S., "Bitcoin: A peer-to-peer electronic cash system," *Consulted* **1**(2012), 28 (2008).
[7] Kondor, D., Pósfai, M., Csabai, I., and Vattay, G., "Do the rich get richer? an empirical analysis of the bitcoin transaction network," *PloS one* **9**(2), e86197 (2014).
[8] Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G. M., and Savage, S., "A fistful of bitcoins: characterizing payments among men with no names," in [*Proceedings of the 2013 conference on Internet measurement conference*], 127–140, ACM (2013).
[9] Frank, J., Mannor, S., and Precup, D., "Activity and gait recognition with time-delay embeddings.," in [*AAAI*], (2010).
[10] Kasprowski, P. and Ober, J., "Eye movements in biometrics," in [*Biometric Authentication*], 248–258, Springer (2004).
[11] Reid, F. and Harrigan, M., [*An analysis of anonymity in the bitcoin system*], Springer (2013).
[12] Kasprowski, P. and Harezlak, K., "The second eye movement verification and identification competition," in [*Proceedings of the 2014 International Joint Conference on Biometrics (IJCB)*], IEEE,IAPR (2014).
[13] Rigas, I., Economou, G., and Fotopoulos, S., "Human eye movements as a trait for biometrical identification," in [*Biometrics: Theory, Applications and Systems (BTAS), 2012 IEEE Fifth International Conference on*], 217–222, IEEE (2012).
[14] Petry, A. and Barone, D. A. C., "Speaker identification using nonlinear dynamical features," *Chaos, Solitons & Fractals* **13**(2), 221–231 (2002).
[15] Takens, F., "Detecting strange attractors in turbulence," in [*Dynamical systems and turbulence, Warwick 1980*], 366–381, Springer (1981).
[16] Abarbanel, H., "Analysis of observed chaotic data," (1996).
[17] Small, M., [*Applied nonlinear time series analysis: applications in physics, physiology and finance*], vol. 52, World Scientific (2005).
[18] Friedman, J. H. and Rafsky, L. C., "Multivariate generalizations of the wald-wolfowitz and smirnov two-sample tests," *The Annals of Statistics* , 697–717 (1979).
[19] Wald, A. and Wolfowitz, J., "On a test whether two samples are from the same population," *The Annals of Mathematical Statistics* **11**(2), 147–162 (1940).
[20] Kittler, J., Hatef, M., Duin, R. P., and Matas, J., "On combining classifiers," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **20**(3), 226–239 (1998).
[21] Duda, R. O., Hart, P. E., and Stork, D. G., [*Pattern classification*], John Wiley & Sons (2012).
[22] Theiler, J., Eubank, S., Longtin, A., Galdrikian, B., and Doyne Farmer, J., "Testing for nonlinearity in time series: the method of surrogate data," *Physica D: Nonlinear Phenomena* **58**(1), 77–94 (1992).
[23] Kantz, H. and Schreiber, T., [*Nonlinear time series analysis*], vol. 7, Cambridge university press (2004).
[24] Kugiumtzis, D., "On the reliability of the surrogate data test for nonlinearity in the analysis of noisy time series," *International Journal of Bifurcation and Chaos* **11**(07), 1881–1896 (2001).
[25] Guarin, D., Orozco, A., and Delgado, E., "A new surrogate data method for nonstationary time series," *arXiv preprint arXiv:1008.1804* (2010).
[26] Song, D. X., Wagner, D., and Tian, X., "Timing analysis of keystrokes and timing attacks on ssh.," in [*USENIX Security Symposium*], **2001** (2001).
[27] Lorenz, E. N., "Atmospheric predictability as revealed by naturally occurring analogues," *Journal of the Atmospheric sciences* **26**(4), 636–646 (1969).