Detecting Cybercriminal Bitcoin Relationships through Backwards Exploration

Gibran Gomez
gibran.gomez@imdea.org
IMDEA Software Institute
Madrid, Spain

Pedro Moreno-Sanchez
pedro.moreno@imdea.org
IMDEA Software Institute
Madrid, Spain

Juan Caballero
juan.caballero@imdea.org
IMDEA Software Institute
Madrid, Spain

ABSTRACT
Cybercriminals often leverage Bitcoin for their illicit activities. In this work, we propose back-and-forth exploration, a novel automated Bitcoin transaction tracing technique to identify cybercrime financial relationships. Given seed addresses belonging to a cybercrime campaign, it outputs a transaction graph, and identifies paths corresponding to relationships between the campaign under study and external services and other cybercrime campaigns. Back-and-forth exploration provides two key contributions. First, it explores both forward and backwards, instead of only forward as done by prior work, enabling the discovery of relationships that cannot be found by only exploring forward (e.g., deposits from clients of a mixer). Second, it prevents graph explosion by combining a tagging database with a machine learning classifier for identifying addresses belonging to exchanges.

We evaluate back-and-forth exploration on 30 malware families. We build oracles for 4 families using Bitcoin for C&C and use them to demonstrate that back-and-forth exploration identifies 13 C&C signaling addresses missed by prior work, 8 of which are fundamentally missed by forward-only explorations. Our approach uncovers a wealth of services used by the malware including 44 exchanges, 11 gambling sites, 5 payment service providers, 4 underground markets, 4 mining pools, and 2 mixers. In 4 families, the relations include new attribution points missed by forward-only explorations. It also identifies relationships between the malware families and other cybercrime campaigns, highlighting how some malware operators participate in a variety of cybercriminal activities.

1 INTRODUCTION
Cryptocurrencies such as Bitcoin are attractive for cybercriminals due to their decentralized nature, (pseudo-)anonymity, irreversible transactions, and the ease to buy and sell them. However, some cryptocurrencies such as Bitcoin have a public transaction ledger, which has enabled the analysis of diverse cybercrime activities such as ransomware [28, 38, 46, 55, 57, 63], thefts [51], scams [22, 51, 56], human trafficking [58], hidden marketplaces [25, 45, 60], money laundering [52], and cryptojacking [65].

A key component of a Bitcoin cybercrime investigation is the analysis of how money flows, e.g., how cybercriminals finance their activities and where the profits of those activities go. Identifying the financial relationships (relations for short) between a cybercrime campaign and external services (e.g., exchanges, mixers, underground markets), or with other cybercrime campaigns, is fundamental in attribution. For example, if a coin flow is identified between a malicious campaign of interest and a service (e.g., an exchange) with know-your-customer (KYC) requirements [32], the service can be leveraged as an attribution point by law enforcement to identify the perpetrators. Moreover, identifying relations between cybercrime campaigns, which were initially considered independent, sheds light into services supporting cybercriminal activities [26], and their operators, which become then targets for takedown [29, 31, 67]. Yet another application is identifying the malicious clients of a specific service, e.g., a mixer suspected of laundering money from illicit activities. This information has been fundamental in past takedown actions, by justifying to a judge that the service is a hotspot of malicious activity. For example, the arrest warrant for the Bitcoin Fog mixer operator was justified on its abuse for money laundering by multiple underground markets [23].

A common technique for analyzing Bitcoin abuse is forward transaction tracing, i.e., tracing the flow of BTCs starting from a given set of seed addresses known to belong to a cybercrime campaign [38, 45, 55, 56]. Examples of seed addresses are those in ransom notes (e.g., [38, 55]) and those in scam emails (e.g., [56]). But, forward transaction tracing has two fundamental limitations. First, it only considers forward propagation of the funds from the seeds, i.e., it does not explore backwards. For example, if the seed is used as input to a transaction with two inputs and one output, it traces the output, but it does not trace (backwards) the other input, which we can also attribute to the campaign since multiple inputs of the same transaction are associated to the same actor (i.e., multiple-input heuristic) [51]. Second, if the output address receives another deposit transaction (where the seed is not an input) that deposit transaction is ignored. Thus, forward-only exploration may often miss fundamental relations. For instance, malware that uses the Bitcoin blockchain for signaling C&C servers (e.g., [57, 64]) needs to seed the signaling addresses with funds. Failure to explore backwards from the signaling addresses may miss the exchange used to seed the C&C signaling that can be leveraged as an attribution point. Furthermore, since the funds are consumed to pay the fees of the signaling transactions, there may not be a cash out that forward-only exploration can identify. Another situation is when examining the clients of a potentially malicious service, e.g., a mixer. Forward-only exploration from the mixer addresses is useless as the mixer’s goal is precisely to prevent forward tracing. However, backwards exploration from the mixer’s addresses allows identifying the clients that leverage the mixer, which can be used to justify an arrest warrant [23]. Last but not least, existing approaches perform shallow explorations to prevent the transaction graph from exploding in size. We posit that the key challenge to prevent such explosion, and also for identifying relations, is to identify change of ownership, i.e., when BTCs change hands from the operators of the campaign under study to other entities. Prior works detect change of ownership using commercial databases of tagged addresses [15, 16]. However, such databases have limited coverage, which can make the graph explode.
For example, failing to identify that the cybercriminals cashed out at an exchange would include a massive amount of transactions from other innocent clients of the exchange into the transaction graph.

To address these limitations, we contribute the following:

• We propose back-and-forth exploration, a novel automated transaction tracing technique to identify cybercrime financial relationships. Given a set of seed addresses belonging to a cybercrime campaign, it outputs a transaction graph and it extracts paths in the transaction graph that capture relations to other entities and campaigns. The transaction graph can be used as evidence of the relations identified, and can be shared with law enforcement and other analysts for independent validation, or with a judge to justify a warrant. Moreover, the relations could be publicly shared, preventing services (e.g., exchanges) that interact with the cybercrime campaigns from claiming that they were unaware of the abuse. Back-and-forth exploration provides two key contributions over current forward transaction tracing techniques. First, it explores both forward and backwards, enabling the discovery of important relations missed otherwise. Second, it combines the use of a tagging database with a machine learning classifier to identify addresses belonging to exchanges. The classifier can detect previously unknown (i.e., untagged) exchange addresses, preventing graph explosion.

• We propose operation oracles, binary classifiers that can identify specific types of malicious addresses using sequences of distinctive transactions. We build operation oracles for four malware families that use the Bitcoin blockchain for signaling C&C servers. We use them to identify signaling addresses in the transaction graphs, demonstrating that back-and-forth exploration can discover 13 signaling addresses missed by previous works [39, 57, 64] and that backwards exploration is key for their discovery.

• We build a database of 230K tagged addresses from open sources and expand it into 116M tagged addresses through multi-input clustering. We show that tag databases need to handle double ownership, i.e., Bitcoin addresses with more than one owner. Double ownership is common in services offering online wallets, which create an address for a specific user. In that scenario the service owns the address (i.e., has the private key), but the address is handled (and thus indirectly owned) by the user for which it is created.

• We apply back-and-forth exploration on 30 malware families: 19 clippers that replace addresses in the OS clipboard of the infected host with an address controlled by the malware operators, 7 ransomware, and 4 families using Bitcoin for C&C. Back-and-forth exploration finds relations in 93% of the explorations. For 13% of the families, it identifies new attribution points, missed by forward-only explorations. It uncovers that 23 replacement addresses for 9 clippers are online wallets in exchanges (double ownership). This shows that clippers, largely overlooked, but highly effective, malware class, often directly cash out the stolen funds without intermediate steps. It is surprising that the abusive nature of those addresses goes unnoticed for the exchanges. Overall, the most common relations are with 44 exchanges. Exchanges used by multiple families (e.g., localbitcoins, coinpayments, coinbase, poloniex, cryptonator, btc-e) are often not the largest ones, which could indicate a more lax approach to abuse detection. In fact, one of those exchanges has already been taken down [67]. Other identified relations are 2 mixers (wasabi, bitcoinfog), the latter being recently taken down for money laundering [29]; 11 gambling sites likely also used for money laundering; 5 payment services used for acquiring resources; 4 mining pools, and 4 underground markets. In 7 explorations, relations are found between a malware family and other cybercrime entities. We observe that some malware operators participate in a variety of cybercriminal activities. For example, the MrPr0gr4mmer operator sells spam tools, sends sextortion scam campaigns, and handles a clipper. We also identify clippers run by the same operators and observe malware operators using profits to upgrade their software arsenal (e.g., buying licenses for malware kits and spam tools) and for seeding new C&C signaling approaches. Finally, we perform a backwards-only exploration on the bitcoinfog mixer identifying 15 underground markets, 130 other services, and 15 malicious operations that leveraged it as clients.

2 STATE OF THE ART

2.1 Bitcoin Address Ownership

The notion of ownership of a Bitcoin address is associated to the user who possesses the corresponding signing key and can thus authorize a transfer of coins from that address. A user may own multiple addresses. For example, a (privacy-aware) user may create a new address for every payment he receives.

Multi-input clustering. We leverage a popular technique to identify Bitcoin addresses belonging to the same owner called multi-input (MI) clustering [21, 51, 54, 59]. It assumes that input addresses to the same transaction have the same owner because their private keys are used together to sign the transaction. Clusters can be created transitively. If a transaction has addresses A and B as inputs, and another transaction has B and C as inputs, then A, B and C are all clustered together and have the same owner. One exception are CoinJoin [49] transactions where a group of users create a single transaction that simultaneously spends all their inputs into a (shuffled) list of outputs. For this reason, before computing MI clustering, we apply proposed heuristics to identify CoinJoin transactions [33, 42]. There exist other address clustering heuristics such as shadow address [21], change address [51], and its variants (peeling chain, power of ten, address reuse) [42]. However, those techniques rely on usage patterns that change over time and can lead to high false positives.

Tagging. While MI clustering identifies addresses belonging to the same owner, it does not determine who the owner is. For this, commercial services (e.g., [15, 16]) provide paid access to databases of addresses tagged with a label stating their owner or usage. Instead, we leverage public resources to build our own tag database. An intrinsic limitation of tag databases is that they only include a small fraction of Bitcoin addresses, having thus a low coverage. In this work, we show that when tagging addresses, it is fundamental to handle double ownership, i.e., that a Bitcoin address may have more than one owner. In the double ownership scheme, a first user (i.e., the owner) possesses the signing key corresponding to the address and accepts transaction requests (possibly authenticated with e.g., username and password) from a second user (i.e., the beneficiary) who is thereby the one governing where the coins are transferred and when. Note that the owner does not share the signing key to the beneficiary, so the beneficiary cannot produce transactions on its own. This double ownership scheme is used for instance by some exchanges and online wallets. These services, on user’s request to deposit its coins, may create a fresh address (and the corresponding signing key) where afterwards the user’s coins are deposited.
Handling double ownership is fundamental when collecting tagged addresses from multiple public sources. For example, the same address may be tagged in one source as belonging to an exchange and in another source as being the replacement address for a clipper malware. Without taking double ownership into account that would create a conflict in assigning ownership to the address. Instead, the address should be tagged as belonging to an exchange (i.e., the owner) that has assigned that address to an exchange user (i.e., the beneficiary) who happens to be the clipper’s operator. Our tag database construction is detailed in Section 4.

**Exchange address classifier.** We propose a machine learning classifier to identify addresses belonging to exchanges, even if the specific exchange it belongs to was never tagged and is therefore not part of the training dataset. Our exchange classifier prevents graph explosion due to limited tagging coverage, but can only determine a given address belongs to an exchange, not to which specific exchange. Our exchange classifier is detailed in Section 5.

**Operation oracles.** Previous works build functions that given a ransomware collection address distinguish victim payments by examining if deposit amounts are within payment ranges used by the family [28, 38, 46, 55]. We expand this concept by proposing operation oracles, binary classifiers that identify specific types of malicious addresses (performing distinctive transactions) by applying rules to the sequence of transactions involving the address. We build operation oracles, detailed in Section 6, for three malware families that leverage the Bitcoin blockchain to signal C&C endpoints. Our operation oracles consider transaction sequences, e.g., that address A sends a payment to address B and soon after address B returns the same amount, minus the transaction fee, back to A. Considering multiple transactions and dependencies between them makes the classifier more distinctive than considering only transaction values, reducing false positives (FPs).

### 2.2 Transaction Tracing

Forward transaction tracing focuses on the outgoing transactions performed by the seeds and transitively by their destination addresses [38, 45, 55, 56]. A main limitation of forward transaction tracing is that it only explores forward, but it does not explore backwards. In particular, in forward transaction tracing, if an address is being traced, then the outputs of all transactions where that address is used as input will also be traced. But, other inputs in those transactions will not be traced. In addition, deposit transactions into a traced address, where the traced address appears as output, but no input is currently traced, are not traced either. Due to this limitation, forward transaction tracing cannot discover relations only reachable by exploring backwards. For example, forward transaction tracing starting from addresses belonging to a mixer makes the exploration explode due to the mixing behavior, while exploring backwards can reveal the clients using the mixer. In another example, exploring backwards from a C&C signaling address can reveal the exchange used to fund the signaling. But, forward transaction tracing may not reveal such attribution point as the funds are consumed in the signaling.

To address this limitation we propose a novel back-and-forth exploration, which traces forward and backward. In a bit more detail, beginning from a seed, our back-and-forth exploration retrieves deposit addresses of incoming transactions, withdrawal addresses of incoming transactions, deposit addresses of outgoing transactions, and withdrawal addresses of outgoing transactions.

We illustrate the differences between forward transaction tracing and back-and-forth exploration using the example transaction graph in Figure 1. In the graph, circle nodes correspond to addresses and square nodes to transactions. Nodes filled in black are those belonging to a cluster tagged as exchange. Edges capture coins flowing from an input address to a transaction, or from a transaction to an output address. The exploration starts from the two seeds in gray. The graph is split in two. The right side corresponds to the graph that forward transaction tracing would produce and the left side is the additional subgraph that back-and-forth exploration would additionally discover, including a new malware address. There are two points where both explorations differ. First, back-and-forth exploration traces deposit addresses to the leftmost seed, including addresses f, e, and d into the graph. Second, once address a has been discovered through a forward step from the leftmost seed, the transaction where a is an input and h is an output will be traced. At that point, back-and-forth exploration would include address d in the tracing while forward transaction tracing would not.

One risk of backwards exploration is that any entity can deposit BTCs into any address. If a benign entity sends funds to a malicious address, the backwards exploration may consider the sender address part of the malicious campaign. Since addresses in ransom notes and scam emails receive victim’s payments and are typically seeds for our back-and-forth exploration, our platform has an option to prevent exploration backwards from the seeds. Note that even if this option is used, back-and-forth exploration still explores backwards at intermediate nodes. For example, in Figure 1, even if backwards exploration on the seeds is disabled, the final graph would be the same because the transaction between a and h would bring d into the graph, and successive backwards steps would then include e, i, and f. There are two other cases where backwards exploration can be problematic. One case is forced address reuse (or dust) attacks where an entity performs micro-payments to addresses it does not own, hoping to incentivize the receivers to move those small amounts in later transactions, so that ML clustering may group some of those addresses [38]. In this work, we use some filtering heuristics to identify and remove forced address reuse transactions. The other case is researchers trying to hijack the C&C infrastructure of a malware by sending payments to the signaling addresses [64]. In such cases, researchers are likely to specify their transactions to avoid being incorrectly linked to the malicious campaign, allowing their filtering.
when BTCs flow into exchanges, e.g., cybercriminals cashing out
check for change of ownership using the input tag database (detailed
to parse the blockchain data up to the desired block height and to pre-
framework [42]. Before launching any exploration, BlockSci is used
transactions. The blockchain data is processed using the BlockSci
It performs an iterative process. At each iteration, it selects a Bitcoin
slot number, and the BTC amount transferred in that slot.
Edges have as attributes the transaction ID, the
amount transferred. Edges have as attributes the transaction ID, the
address from the worklist and performs three steps. First, it obtains
they may connect an output slot in a transaction to the address
between unrelated cybercriminal campaigns that simply use the
same exchange. For example, in Figure 1 address b is identified as an
exchange by our ML classifier, while c is identified as belonging
to the poloniex exchange through a previously available tag. Failure to
detect these addresses would have made the graph explode.

3 BACK-AND-FORTH EXPLORATION
Given seed Bitcoin addresses belonging to a cybercrime campaign,
back-and-forth exploration outputs a transaction graph and identi-
fies paths in the graph that capture relations to other entities and
campaigns. As illustrated in Figure 1, nodes in the transaction graph
are of two types: addresses and transactions. Edges are directed;
they may connect an output slot in a transaction to the address
where funds are received, or connect an address to the input slot in
a transaction where funds are sent to. If the same address appears
several times as input or output of the same transaction, a single
edge connects address and transaction. Address nodes are reused if
the same address appears in multiple transactions. Address nodes
have attributes capturing their alphanumeric address string, the list
of tags associated to the address (if any), the identifier of the MI
cluster the address belongs to, and the address type (e.g., if detected
by the exchange classifier or by the operation oracle). Transaction
nodes have as attributes the transaction identifier and the total BTC
amount transferred. Edges have as attributes the transaction ID, the
slot number, and the BTC amount transferred in that slot.

The back-and-forth exploration architecture is shown in Figure 2.
It performs an iterative process. At each iteration, it selects a Bitcoin
address from the worklist and performs three steps. First, it obtains
the address context by identifying the MI-cluster the address belongs
to and extracting from the blockchain its deposit and withdrawal
transactions. The blockchain data is processed using the BlockSci
framework [42]. Before launching any exploration, BlockSci is used
to parse the blockchain data up to the desired block height and to pre-
compute the MI clustering results. Second, it classifies the address to
check for change of ownership using the input tag database (detailed
in Section 4) and the exchange classifier (detailed in Section 5).
Finally, it updates the transaction graph with the obtained transac-
tions and addresses; and adds to the worklist any new addresses that
need to be explored. The process starts by placing the seeds in the
worklist and finishes when the worklist is empty, or a configurable
exploration limit is reached. At the end of this iterative process the
transaction graph is output.

The graph analysis module takes as input the transaction graph
and an optional operation oracle. It outputs relations between the
campaign under study and other entities in the graph. First, it tags
addresses satisfying the oracle, e.g., as a C&C signaling address.
Then it identifies relations, which are paths between a seed and a
tagged address that satisfy the following constraints. First, the found
tag should differ from the seeds’ tag. Second, the path should be
directed from the seed to the tagged address, or from the tagged
address to the seed, i.e., it should not mix edges with opposite
directions. This constraint makes sure that the relation captures
money flow from the campaign under analysis to an external entity
or vice-versa. Third, addresses identified by the exchange classifier,
for which no tags were available, are excluded since it is not known
to what specific exchanges they belong to.

Address context. At each iteration, the address being explored is
used to query BlockSci to obtain the ID of the MI cluster to which
the address belongs. If the cluster has already been examined, e.g.,
if detected as an exchange when examining a previous address in
the cluster, its classification results are recovered. Then, BlockSci
is queried again to obtain the list of all transactions involving the
address. CoinJoin and forced address reuse transactions are filtered
from the exploration. CoinJoin transactions are identified using a set
of previously proposed heuristics [33, 42] implemented by BlockSci.
To identify force address reuse transactions we use our own heuristic,
which looks for transactions that send the same amount to a large
number of (at least 100) output addresses. Next, it obtains from
BlockSci the list of addresses involved in the remaining transactions,
both in input and output slots. Transactions and addresses are added
to the transaction graph and passed to the address classification that
determines which addresses should be explored in later iterations.

Address classification. Each address received from context extrac-
tion is examined for change of ownership. Addresses identified as
belonging to other entities, either through tags or by the exchange
classifier, will be labeled as such and will not be traced. The rest will
be added to the worklist to be traced. First, the address is queried
to the tag database. If tagged, and if its tag category identifies it
as a service (e.g., exchange, onlinewallet, mixer, tormarket, gam-
bling), the address is not added to the worklist. There is no need
to keep exploring the address since that would explore the service,
introducing many extraneous nodes in the transaction graph. Second,
the ML exchange classifier, detailed in Section 5, is applied to the
address. If detected, the address is not added to the worklist. As an
optimization, the ID of the address’ MI-cluster is cached, so that
any other address in the same cluster can be directly marked as an
exchange without going again through the classification. Addresses
added to the worklist are assigned a priority inversely proportional
to the total number of input and output slots in their transactions.
The intuition is that addresses with many transactions, or with very
large transactions, could belong to services not in the tag database.

Figure 2: Back-and-forth exploration architecture.
Appendix details the top 20 sources by number of tagged addresses. (e.g., the SatoshiDice gambling service [18]), forums where users provide their own tagging database, which can be automatically imported into our platform. For others like BitcoinTalk, tags are propagated to the cluster. If multiple tags of the mining category are assigned to the same address or cluster, we explore a variety of publicly available sources to tag Bitcoin addresses including databases of already tagged addresses, (e.g., WalletExplorer tags [19] available in GraphSense [36]), web pages where services disclose their address policy (e.g., the SatoshiDice gambling service [18]), forums where users rate each other (e.g., BitcoinTalk [14], Bitcoin-OTC [13]), crowdsourced databases where users report abusive addresses (e.g., Bitcoin Abuse [11]), and over 10K security blog articles where we apply a regular expression to identify those discussing Bitcoin addresses. Some of these sources such as WalletExplorer and Bitcoin-OTC provide their own optional component that details the usage of the address, e.g., if it is the hot wallet of an exchange or a malware C&C signaling address. The final component is the list of source URLs that discuss the address. For example, address INDxJt has a category exchange, label binance (the specific exchange), hot wallet as subtype, and a URL to the WalletExplorer webpage that describes it. Throughout the paper, when we refer to a tag we mean its category and label, e.g., exchange:binance. We identify addresses by their first 6 characters, but provide full addresses in Table 14 in the Appendix.

We initially assign a tag to an address for each source that mentions it. Then, we select a unique address tag by applying the following criteria for resolving potential ambiguity in addresses with multiple tags. If an address has multiple tags with the same category and label, we favor the tag that provides more information, i.e., the one with a subtype, or the longest one otherwise. If an address has multiple tags with different category and label pairs, we favor tags not coming from Bitcoin Abuse because categories in that service are assigned by users with varying expertise and it is not rare that they are wrong. If an address has multiple tags of the mining category, we merge its labels to capture all the mining pools where the miner has participated. In any other cases, we manually resolve the inconsistency. If we cannot confidently assign a tag, we remove the address from the database. When applying this procedure to the 230K addresses, only 66 (0.03%) required manual inspection and we could disambiguate those, showing that this strategy is effective.

Next, we obtain the MI clusters for all the tagged addresses. If a single tagged address appears in the cluster, we assign that tag to all other addresses in the cluster. If multiple tagged addresses appear in the same cluster, we apply the following disambiguation process. If a cluster has only one service tag, we assume this is a case of double ownership where the owner of the address is the service, and the beneficiary is the owner of the other tags. If multiple service tags are present, we try to manually disambiguate them, e.g., checking if the two services are related to each other. For cases where no service tags are present, we compute the edit distance between their labels. If they are close, e.g., user:Metabank.ru and user:Metabank, we consider them aliases and merge them. Otherwise, we manually try to disambiguate them. If manual disambiguation fails, we do not expand the tags to the cluster.

Applying MI clustering to the 230K tagged addresses produces 61K clusters comprising of 116M addresses. Of the 61K clusters, 817 (1.15%) have more than one tag, from which 368 have only malware tags from Bitcoin Abuse and 286 can be automatically solved since they contain multiple tagged addresses of the same owner. Of the remaining 163 clusters, we could manually resolve conflicts in 91 corresponding to 54 services with double-ownership (e.g., exchanges) and 37 equivalent tags (e.g., usernames with two aliases). We could not resolve 72 clusters, for which we do not propagate tags. Of those, 66 were clusters without service tags (63 of them with multiple Bitcoin-OTC user tags) and 6 clusters with more than one service tag (4 of them with multiple mining tags). An interesting case is one of the clusters with multiple service tags. It is due to the special address 3J98t1, which corresponds to the empty string private key. Surprisingly, that address occasionally receives...
We evaluate the models to make predictions on unseen data by using the 20% of the dataset reserved for testing. The results, on the right side of Table 4, show that the best model is again Random Forest with a F1 of 0.958, so we use that model throughout the explorations with the following hyper-parameters: 600 trees, maximum depth of 40, and prediction threshold of 0.5. Figure 7 in the Appendix shows the classifier’s ROC curve.

6 BACK-AND-FORTH EXPLORATION EVALUATION

We evaluate our platform on 30 malware families: 4 families that use Bitcoin for C&C, 19 clipplers that replace addresses in the OS clipboard of the infected host with an address controlled by the malware operators, and 7 ransomware. For each family, we run two explorations: back-and-forth and forward-only. Table 5 summarizes the back-and-forth explorations and the delta with the forward-only explorations (in parenthesis). Full forward-only results are in Table 12 in the Appendix.

| Class | Source | StAddrs. | Entities | MI Clust. | Addr Clusters |
|-------|--------|----------|----------|-----------|--------------|
| positive | WalletExplorer | 199 | 117 | 151 | 14,365 |
| negative | Bitcoin-OTC | 923 | 923 | 788 | 38,702 |
| negative | SatoshiDice | 49 | 49 | 15 | 1,154 |
| negative | Ransomware | 7,223 | 65 | 78 | 14,409 |

Table 2: Dataset of 108,530 addresses used for training and testing exchange classifier.

| Feature | New | Type | Class | Description |
|---------|-----|------|------|-------------|
| type | ✓ | Cat. | AD | Address type |
| equiv_addrs | ✓ | Int. | AD | Number of equivalent addresses |
| lifetime | ✓ | Int. | TS | Total seconds from first TX to last |
| timepan_d | ✓ | Int. | TS | Seconds from first deposit TX to last |
| timepan_w | ✓ | Int. | TS | Seconds from first withdrawal TX to last |
| activity | ✓ | Int. | TS | Number of days with at least one TX |
| activity_d | ✓ | Int. | TS | Number of days with at least one deposit TX |
| activity_w | ✓ | Int. | TS | Number of days with at least one withdrawal TX |
| idle_time | ✓ | Int. | TS | Number of days without TXES |
| daily_d_rate | ✓ | Float | TS | Rate of deposits per day in its lifetime (tx/day) |
| daily_w_rate | ✓ | Float | TS | Rate of withdrawals per day in its lifetime (tx/day) |
| yearly_d_txes | ✓ | Float | TS | Average number of deposit TXES per year |
| yearly_w_txes | ✓ | Float | TS | Average number of withdrawal TXES per year |
| balance | ✓ | Int. | TX | Total balance of this address |
| withdrawn | ✓ | Int. | TX | Total satoshis withdrawn from address |
| txes_out | ✓ | Int. | TX | Total number of transactions |
| txes_in | ✓ | Int. | TX | Total number of TXES with this address as output |
| addd_as_change | ✓ | Float | TX | Ratio of TXES address used as input and output |
| inputs | ✓ | Int. | TX | Total input slots in which the address appear |
| outputs | ✓ | Int. | TX | Total output slots in which the address appear |
| utxos | ✓ | Int. | TX | Number of unspent transaction outputs to date |
| tx_size_mean | ✓ | Float | TX | Average total size (in bytes) of all in TXES |
| tx_weight_mean | ✓ | Float | TX | Average weight of TXES |
| tx_fee_mean | ✓ | Float | TX | Average fee paid in TXES |
| ins_age_mean | ✓ | Float | TX | Average blocks between input and output |
| coinbase | ✓ | Int. | TX | Number of TXES that are coinbase |
| coinjoin | ✓ | Int. | TX | Number of TXES that are coinjoin |
| coinjoin_in | ✓ | Int. | TX | Coinjoin TXES with this address as input |
| coinjoin_out | ✓ | Int. | TX | Coinjoin TXES with this address as output |
| outs_per_tx | ✓ | Float | TX | Average number of output slots per TX |
| ins_per_tx | ✓ | Float | TX | Average number of input slots per TX |
| outs_per_out | ✓ | Float | TX | Average number of output slots per output TX |
| ins_per_out | ✓ | Float | TX | Average number of input slots per output TX |
| ins_per_in | ✓ | Float | TX | Average number of input slots per input TX |
| ins_per_out | ✓ | Float | TX | Average number of input slots per output TX |
| outs_per_in | ✓ | Float | TX | Average number of output slots per input TX |
| profit_rate | ✓ | Float | TX | Total satoshis deposited over lifetime |
| expense_rate | ✓ | Float | TX | Total satoshis withdrawn over lifetime |
| d_per_ts | ✓ | Float | TX | Total deposited amount per TX (satoshi/tx) |
| w_per_tx | ✓ | Float | TX | Total withdrawn amount per TX (satoshi/tx) |

Table 3: Classifier features extracted per address.

5 EXCHANGE DETECTOR

We build a binary ML classifier to identify if an address belongs to an exchange in three steps. First, we build a dataset of 108K labeled addresses. Then, we propose 42 features and use them to train different models. Finally, we evaluate the accuracy of the models.

Dataset. To train and test the classifier we build a dataset of 108,530 labeled addresses, summarized in Table 2. Half of them (positive class) are addresses tagged as exchanges by WalletExplorer [19]. The other half (negative class) are non-exchange addresses belonging to individuals, ransomware, and gambling sites. Individuals correspond to Bitcoin-OTC [13] users with a registered Bitcoin address and GPG key, and having received ratings. To avoid abusive users we require that they have received less than 100 negative ratings and have an overall positive rating. Ransomware addresses come from [55] and gambling addresses belong to SatoshiDice [18]. We obtain the MI clusters of those starting addresses and select additional addresses from the clusters until the dataset is balanced. We use 80% of the dataset for training and hold the remaining 20% for testing.

Model training. We extract 42 features for an address, summarized in Table 3. Features can be grouped into three classes. The 29 transaction (TX) features capture properties of the transactions where the address appears such as number of (all/deposit/withdrawal/CoinJoin) transactions, number of transactions where the address appears as input or output, and statistics on the transferred amounts and paid fees. The 11 block timestamp (TS) features capture time properties such as address lifetime, time differences between noteworthy events (e.g., first and last deposits), and statistics on the daily rate of transactions. Finally, two address (AD) features capture the address type (e.g., pubkeyhash, multisig) and the number of other addresses (e.g., multisig) built from the address. Of the 42 features, 74% (31) are novel and the rest have been used in previous works [22, 34, 41, 48, 68]. Features are normalized using z-score to have zero mean and unit standard deviation.

We train three models using the 42 features and 80% of the labeled dataset: Decision Tree, Random Forest, and Gradient Boosting. We focus on tree-based models because they are easy to interpret. The left side of Table 4 shows the precision, recall, and F1 score for each model, using a stratified 5-fold cross validation training. The results show that Random Forest achieves the best F1 score of 0.956.

Testing. We evaluate the models to make predictions on unseen data by using the 20% of the dataset reserved for testing. The results,
### Table 5: Back-and-forth explorations for the 30 malware families. In parenthesis, forward-only exploration results for comparison:

| Seeds | Graph | Classification |
|-------|-------|----------------|
| Class | Operation | OW Exp. | Addrs | Txs | Unexp | Tagged | Exch. | Relat. | Time |
| C&C   | ✓      | 4 | 0 | 4 | 608 (71.7%) | 1,050 (80.0%) | 12 | 209 | 1 (0) | 26 |
|       | ✓      | 4 | 0 | 2 | 177 (13.6%) | 190 (16.0%) | 5 | 12 | 76 | 0 | 0 | 12 |
|       | ✓      | 4 | 0 | 2 | 1,154 (23.8%) | 4,252 (36.4%) | 78 | 379 | 188 | 1 | 0 | 57 |
|       | ✓      | 1 | 1 | 1 | 132 (0.8%) | 107 (11.2%) | 5 | 37 | 7 | 5 | 1 | 40 |
|       | ✓      | 1 | 1 | 0 | 54 (13.0%) | 44 (18.2%) | 6 | 7 | 19 | 1 | 0 | 4 |
|       | ✓      | 1 | 1 | 1 | 1 (1.0%) | 0 (-) | 0 | 1 | 0 | 1 | 1 |
|       | ✓      | 1 | 1 | 1 | 1 (1.0%) | 0 (-) | 0 | 1 | 0 | 1 | 1 |
|       | ✓      | 1 | 0 | 1 | 504 (47.2%) | 162 (19.3%) | 5 | 48 | 55 | 4 | 0 | 51 |
|       | ✓      | 1 | 1 | 1 | 125 (47.2%) | 162 (19.3%) | 5 | 48 | 55 | 4 | 0 | 51 |
|       | ✓      | 1 | 1 | 1 | 1 (1.0%) | 0 (-) | 0 | 1 | 0 | 1 | 1 |
|       | ✓      | 1 | 1 | 1 | 1 (1.0%) | 0 (-) | 0 | 1 | 0 | 1 | 1 |
|       | ✓      | 1 | 1 | 1 | 125 (47.2%) | 162 (19.3%) | 5 | 48 | 55 | 4 | 0 | 51 |
|       | ✓      | 1 | 1 | 1 | 1 (1.0%) | 0 (-) | 0 | 1 | 0 | 1 | 1 |
|       | ✓      | 1 | 1 | 1 | 125 (47.2%) | 162 (19.3%) | 5 | 48 | 55 | 4 | 0 | 51 |
|       | ✓      | 1 | 1 | 1 | 1 (1.0%) | 0 (-) | 0 | 1 | 0 | 1 | 1 |
|       | ✓      | 1 | 1 | 1 | 125 (47.2%) | 162 (19.3%) | 5 | 48 | 55 | 4 | 0 | 51 |

For clipper and ransomware families back-and-forth exploration is configured to avoid exploring backwards on the seeds since addresses depositing into the seeds belong to victims (SDD column).

Note however, that back-and-forth exploration will still explore backwards after the first forward step from the seeds. The seed information is broken into three columns. All is the total addresses in our tag database for the family. OW captures how many of those seeds are identified as belonging to online wallets. Those seeds are not explored forward as that would explore the transactions of the seed holding the online wallet, but they can still be explored backwards if they do not belong to a clipper or ransomware (i.e., skidmap). Exp. captures the number of addresses effectively explored, after removing addresses in online wallets and addresses without transactions. Some families like clipps and locky have thousands of seeds. In the case of clipps, when replacing the address in the clipboard, this clipper selects a visually-similar address from an address database, so that the replacement has a higher chance of not being noticed by the victim. Since the address database ships with the malware, researchers have published its contents [61]. However, only 375 (4%) of clipps addresses have transactions and thus are effectively used in the exploration. Those 375 addresses collect nearly 18 BTC ($277K). Other families like n40 collect even larger BTC amounts (27.4 BTC, $187K) using a single replacement address. Thus, replacement addresses hardcoded in clipper samples are often hosted in exchanges. It is surprising that the abusive nature of those addresses goes unnoticed for the exchanges, as they are often mentioned in public abuse reports by the victims, as well as in analyses by security vendors. Note that this does not happen for addresses in ransom notes, possibly because of ransomware’s higher visibility. Based on this observation, we discuss in Section 8 a defense that would examine public reports for malicious addresses, identify online wallets, and (publicly) report them to exchanges, so that it cannot be claimed that their abusive nature was not known.

The middle section of the table details the output graphs. For 7 clipper families all seeds are in online wallets. Thus, no exploration is performed for them and the graph contains only the seeds. For the remaining 23 families, 69% contain a single component. Each component contains at least one seed. Thus, multiple components indicate that those seeds were not connected to the rest of the seeds. The graphs for those 23 families contain an average of 2,274 addresses, with a maximum of 13.8K for globeimposter. Unexplored addresses are those not processed before the exploration limit was reached. The numbers in parenthesis show the fraction of addresses and transactions in the forward-only graph compared to the back-and-forth graph. On average, the back-and-forth graph is three times larger (3.6x addresses, 2.6x transactions) offering a more complete view of an operation.

In Sections 6.1 and 7 we show that the back-and-forth graphs contain critical information not available in the forward-only graphs such as previously unknown C&C signaling addresses for Cerber and Pony, unreported relations between malware families (e.g., DMalokcer and Slave), and new attribution points (Pony, Skidmap, MrPr0gr4mmer, Phorpiextldr).
The right section of the table shows the number of addresses tagged and those without a tag identified as exchanges by the ML classifier. However, for the four families that use the Bitcoin blockchain for C&C we are able to build operation oracles that can identify previously unknown C&C signaling addresses in the output graph, demonstrating the benefits of back-and-forth exploration.

The use of the Bitcoin blockchain for C&C follows a similar pattern for the four families. Samples have a hardcoded signaling address. Upon infection, the sample queries a external Bitcoin monitoring service (e.g., [12]) to obtain the most recent transactions involving the signaling address. Pony and Skidmap use the same signaling method that encodes C&C IP addresses in the value of two consecutive deposits into the signaling address [64]. Additionally, transactions involving the address should satisfy the following constraints: deposit transactions should have three or less inputs and two or less outputs, and withdrawal transactions should have only one output and all inputs should belong to the given address. An exploration is performed starting from four seeds obtained from two sources [4, 64]. The operation oracle identifies 8 previously unknown signaling addresses. The 12 signaling addresses are detailed in Table 7. Discovering 6 of the 8 new addresses requires at least one backwards exploration step, which again highlights the importance of back-and-forth exploration. Five of the new addresses are SegWit (starting prefix bc1). They seem to be used for testing before switching to the production non-SegWit addresses (starting prefix bc1). For example, bc1qg0 generates two IP addresses on December 9th, 2019. The next day, 19hi8B starts signaling IP addresses and the last IP signaled by bc1qg0 is the first IP signaled by 19hi8B.
other SegWit addresses behave similarly. Using SegWit addresses for testing could be motivated by smaller transaction fees.

In addition to the new signaling addresses, backwards exploration also finds that the funds used by two signaling transactions come from the MINE exchange [17]. This provides a previously unknown attribution point, illustrating how backwards exploration can find important relations that forward transaction tracing cannot.

Encoding the C&C servers on deposit transactions has the downside that analysts can deposit funds into a signaling address to hijack the C&C communication. On August 14th, 2020, Taniguchi et al. [64] performed a takeover on Pony by signaling a server under their control to 1Bke4Gq. However, they did not sustain the takeover, so the attackers regained control two days later.

**Skidmap.** Skidmap is a crypto mining malware targeting Linux hosts [40]. On February 2021, Akamai researchers identified that Skidmap had started using Pony’s signaling as a backup C&C mechanism, and mentioned one signaling address [62]. That seed is an online wallet. Thus, forward tracing from the seed cannot be performed. But, the exploration can proceed backwards from the seed. In this case the exploration does not find any new signaling address. However, in Section 7, we show how the backwards exploration identifies that the funds used in the signaling come from a previous Skidmap crypto-jacking campaign and that operators cashed-out part of the funds in three exchanges that can serve as attribution points, again illustrating how back-and-forth exploration identifies important relations forward transaction tracing could not.

**Glupteba.**

Glupteba is a botnet that has been operating since at least 2011 [35]. In September 2019, it was reported to have added Bitcoin as a mechanism to distribute backup C&C domains [37]. The C&C signaling leverages Bitcoin transactions with an OP_RETURN output, which stores in the blockchain an AES-256 encrypted backup C&C domain. Glupteba binaries have a hardcoded script hash used to query external services for obtaining the transactions encoding the backup C&C domain. We build an oracle that checks for transactions with an OP_RETURN output, validates the format of the OP_RETURN string, and tries to decrypt the content using the AES keys obtained from the malware binaries [53]. An exploration is performed starting from three seeds obtained from prior reports [39, 53]. The operation oracle identifies 2 previously unknown signaling addresses in the graph. The 6 signaling addresses are detailed in Table 8. The two discovered addresses seem to have been used for testing. They signal the same domain in consecutive days and one day later that domain is signaled by address 15y7ds. In December 2021, Google performed a take-down on the Glupteba botnet, although their note mentions that thanks to the Bitcoin C&C signaling the malware authors are likely to regain control of the botnet [39].

In summary, the results show back-and-forth exploration finds new C&C signaling addresses and important relations that forward transaction tracing could not. In particular, of the 13 new signaling addresses discovered, 8 are only discovered through backwards exploration, and the 4 attribution points found in Skidmap and Pony are also only discovered through backwards exploration.

### 6.2 Exchange Classifier Accuracy Impact

A false positive by the exchange classifier (C-FP) might make the exploration discard valid paths and thus miss true relations, introducing end-to-end false negatives (R-FN). On the other hand, an exchange classifier false negative (C-FN) leads to larger graphs that may contain false relations introducing end-to-end false positives (R-FP). To measure the impact of exchange classification errors on the identification of end-to-end relations, we run the back-and-forth exploration multiple times on each of the 30 families. Each exploration introduces errors to the exchange classifier results with an $\epsilon$ probability, and compares the relations found to the family’s baseline in Table 5. An R-FN is the absence of a relation in the baseline, while an R-FP is a new relation not present in the baseline.

For ease of interpretation, we perform two experiments: introducing only C-FPs and introducing only C-FNs. For example, when introducing C-FPs, if the classifier says an address is not an exchange, with $\epsilon$ probability the decision is flipped to mark it as an exchange. We increase $\epsilon$ between 5% and 40% in steps of 5%. Figure 4 shows, for each experiment, the F1 score of the relation detection as a function of $\epsilon$. The results show that the introduction of C-FPs creates a more pronounced decrease in the relation detection results. Misclassifying a non-exchange address as an exchange (C-FP) stops the exploration too early, which causes relations to be missed (R-FNs). On the other hand, C-FNs introduce additional addresses in the graph, but since many of those new addresses are untagged, the number of R-FPs introduced is lower in comparison, and thus the F1 score reduces more slowly. These results indicate that favoring a high precision of the exchange classifier is beneficial to the overall relation detection. This is already true for our classifier as Table 4 shows that its precision is higher than its recall.

### 7 RELATIONSHIPS

Tagged addresses in a transaction graph may capture relations between the campaign under analysis and external services (e.g, exchanges, mixers, gambling, markets) and other malicious campaigns. Those relations are identified by the graph analysis module as paths that detail the specific chain of transactions capturing money flow from the campaign under analysis to an external entity, or vice-versa. Such paths constitute evidence that can be shared with third parties to allow independent validation of the relations.

---

Table 8: Glupteba signaling addresses. Exploring from 3 seeds, 2 previously unknown signaling addresses were discovered.

| Address | In | Step | Found | FQDNs | Signaling Period |
|---------|----|------|-------|-------|-----------------|
| 15y7ds  | ![Image](359x614 to 515x708) | ![Image](362x328) | ![Image](362x328) | ![Image](362x328) | ![Image](362x328) |
| ![Image](359x614 to 515x708) | ![Image](362x328) | ![Image](362x328) | ![Image](362x328) | ![Image](362x328) | ![Image](362x328) |
Table 9: Summary of relationships found. Tags with an asterisk at the end were found by multi-input clustering on the seeds, the rest were found by the exploration.

| Class           | Operation       | Relations Found                                                                 |
|-----------------|-----------------|-------------------------------------------------------------------------------|
| C&C             |                 |                                                                                |
| rabber          | exchange        | bitstamp, exchange:bittrex, exchange:btc-e.com, exchange:cex.io, exchange:localbitcoins, exchange:poloniex, exchange:tradeogre* |
|                 |                 |                                                                                |
| aggh            | exchange        | bitstamp, exchange:bittrex, exchange:coincase, exchange:coinbase, exchange:coincasa*, exchange:huobi |
|                 |                 |                                                                                |
| androidclipper  |                 |                                                                                |
| asordt          |                 |                                                                                |
| bitcoinclipper  |                 |                                                                                |
| casheiro        |                 |                                                                                |
| clipbanker      |                 |                                                                                |
| clpsx           |                 |                                                                                |
| cryptoipuffer   |                 |                                                                                |
| masad           |                 |                                                                                |
| mkefotto40      |                 |                                                                                |
| misupau         |                 |                                                                                |
| phorex-dtdr     |                 |                                                                                |
| phorex-trik     |                 |                                                                                |
| protatortheif   |                 |                                                                                |
| protobit        |                 |                                                                                |
| cryptotlocker2015 |                 |                                                                                |
| cryptxxx        |                 |                                                                                |
| dmalocker       |                 |                                                                                |
| globemipositer  |                 |                                                                                |
| locky           |                 |                                                                                |
| samsam          |                 |                                                                                |
| saucery         |                 |                                                                                |

Other common relations are gambling sites and mixers, likely used for money laundering. Six families are observed transacting with 11 gambling sites (e.g., satoshidice) with the cryptotlocker2015 ransomware using a record 5 gambling sites. Gambling sites are also possible attribution points since they may be subjected to legislation mandating KYC practices such as the Licence Conditions and Codes of Practice (LCCP) in the UK [27]. Two mixers appear across 4 families: wasabi (3) and bitcoinfog (1). Wasabi [20] is an open-source Bitcoin wallet implementing CoinJoin shuffling over Tor for anonymity. The operator of bitcoinfog was arrested on April 2021 on counts of unlicensed money transmission and money laundering [29]. Other relations found include three malware families using profits for buying products from four Tor hidden markets (alphabaymarket, evolutiononmarket, nuclearmarket, silkroad2market) and two families transacting with four mining pools (btc-pool, cunterra, deepbit, esbot). Finally, several relations between malicious campaigns are found, as detailed next.

Phorpiex-tldr and SaveYourself. The exploration from phorpiex-tldr reveals multiple connections with the saveyourself sextortion campaign. Figure 5 shows the paths capturing these relations, other
paths are excluded for readability. There are six saveyourself tagged addresses in the graph, corresponding to addresses reported by Bitcoin Abuse users as appearing in sextortion emails with a common SaveYourself pattern in the subject and content. The figure shows how the clipper’s profits flow towards multiple saveyourself addresses through different paths, e.g., to tagged address C from the seed (in gray) through addresses A and B, to tagged address J from the seed, and through collector node G to tagged address O. Addresses O and J correspond to online wallets in the duckdice.io gambling service used by SaveYourself operators for betting. Some connections are found through forward transaction tracing, but others such as the path going from the seed to address D through A, B, and C or the path going from the seed through G and H to reach node I require exploring backwards. This illustrates the benefits of back-and-forth exploration. An external report mentions that the Phorpiex malware sends sextortion emails [9]. Checking the Bitcoin addresses in that report on Bitcoin Abuse confirms the link between Phorpiex and the SaveYourself campaign. Furthermore, the multiple connections between both campaigns and how the clipper profits flow into the sextortion addresses makes us conclude that the sextortion campaign is run by the malware operators, as opposed to the Phorpiex operators renting their botnet for sending third-party spam.

**MrPr0gr4mmer clipper and scam.**

The MrPr0gr4mmer exploration, summarized in Figure 6, starts from two seeds (A, B) in the same MI cluster, each with multiple clipper reports in Bitcoin Abuse. Address F is reported in sextortion scam emails [9]. The sender of the scam emails is “Mr Robot”. Manual searches uncover that seed A appears in the @MrPr0gr4mmer Telegram channel, where software to send spam through the telegram exchange and transactions to the mercadobitcoin.br mention a connection. In addition, both graphs show cash outs to thewasabi mixer in the non-shared components, further reinforcing that both campaigns have the same operators. Another relation in the mekotio graph is a payment from a mekotio seed to the seed of mrpr0gr4mmer used for selling the Telegram spam tool, illustrated in Figure 6. Thus, mekotio operators are using the obtained profits to pay for a spam tool possibly used as part of their malware distribution.

**PredatorTheThief and Masad.** The predatorothief clipper seed makes a transaction to an address that a report identifies as belonging to the Masad Clipper and Stealer malware [43]. Masad is offered as a malware kit in forums. It has a free version and a full version that costs $85. The transaction on June 2020 is for 0.00910768 BTC, roughly $85.31 USD using the conversion at that time. We believe this relation captures how predatorothief operators bought a license of masad, possibly to upgrade their own malware.

**DMALocker and Slave.** The dmalocker ransomware graph contains a transaction to an address tagged as belonging to the Slave banking trojan [24]. Slave receives 2.3 BTC from DMALocker, more than 1.3K USD at that time. Unfortunately, in this case we are not able to conclude what type of relation the transaction captures.

**Skidmap and Sysupdate.** The skidmap cryptominer backwards exploration (forward transaction tracing is not possible as the seed is an online wallet) identifies a cryptojacking:sysupdate address that sends funds to the seed six times. This address belongs to a mining pool used by a Linux cryptomining malware to receive the shares produced by the botnet’s mining activity [7]. We believe this malware is an older skidmap version (prior to using Bitcoin for C&C) and that the relation shows that funds from Skidmap’s cryptomining were used to seed the new Bitcoin C&C signaling. The Sysupdate address cashes out in three exchanges (coinbase, huobi, bitmex) that can be used as attribution points and are only found exploring backwards.

**CryptoTorLocker2015 and Scam Campaigns.** The MI cluster of the cryptotorlocker2015 ransomware seeds contains 10 addresses beginning with short English words involved in multiple scam campaigns in 2014-2015. Four of those imitate satoshidice gambling addresses beginning with “1DiCE”, another four (e.g., 1BonUS, 1sYSTEM) were used in scams that promise to double the BTCs sent [2], and two addresses beginning with “1bet” and “1shop” were used in campaigns claiming to have hacked the bitcointalk forum [1]. Those scams are likely run by the ransomware operators.

### 7.1 Limited Ground Truth Validation

Ground truth is not available for cybercrime operations and can only rarely be obtained, e.g., through leaks [44, 50]. Still, we tried hard to find some ground truth, even if limited, encountering two major challenges. First, prior analysis focuses on forward-only explorations and thus does not mention relations only reachable by exploring backwards. Second, analysis reports rarely exhaustively list relations, mentioning only those of interest to the report. We found only one report that partially addresses those challenges, which we use as a limited ground truth. It corresponds to the affidavit filed in April 2021 by the US Internal Revenue Service to justify the takedown of the bitcoinfog mixer [23]. The affidavit mentions bitcoinfog received deposits from 35 darknet markets, but only names the five largest: agora, alphabay, evolution, silkroad, and silkroad2.

Our database has one address tagged as bitcoinfog belonging to a MI cluster with 244K addresses. Forward exploration from the mixer addresses does not work as the mixer’s goal is to hamper forward
transaction tracing. Thus, we perform a backwards-only exploration using as seed all addresses in the bitcoinfog MI cluster. Due to the large number of seeds, we limit the exploration to one backwards step. The exploration finds a wealth of relations including all 15 darknet markets in our database. Those 15 markets include the five named in the affidavit and another 10: abraxas, babylon, blackbank, bluesky, cannabissroad, doctor, middleearth, nucleus, pandora, and sheep. In addition, it identifies relations with 130 services and 15 malicious operations not mentioned in the affidavit: 4 ransomware (cryptolocker, loccy, xlocker, xplocker), 3 scams (mintpal, mcxnow, bitoomba), 3 thefts (binance-hack, bips-hack, flexcoin-hack). 3 mixers (bitlauder, wasnabi, and an unknown mixer), a malware (slave), and a ponzi scheme (nanoindustry).

This experiment shows backwards exploration finds correct relations, although it may miss relations for which no tags are available. Unfortunately, we cannot evaluate false positives as the affidavit does not list all markets, and other services and malicious operations.

8 DISCUSSION

Backwards exploration. Our results show that back-and-forth exploration can discover important relations that forward transaction tracing cannot. One limitation is that backwards exploration is not suitable for every transaction. However, we show how it is possible to identify the problematic cases. For example, we provide a simple heuristic to identify dust attacks that may deposit small amounts to many addresses. We also provide an option to disable backwards exploration on seeds, which requires an easy decision by the analyst: use it when seeds may get deposits from other addresses belonging to the same campaign (e.g., C&C signaling) whereas avoid it when deposits come from victims (e.g., clippers, ransomware). Furthermore, even if a transaction was included that it should not, the transaction and its dependencies can later be pruned from the graph.

Exchange defenses. 90% of malware families analyzed had at least one relation with an exchange. This includes 10 malware families where addresses hardcoded in the malware correspond to online wallets in exchanges. These exchanges are either unaware about the abusive nature of those addresses, or they simply do not care. This creates an opportunity for intervention where public malware analysis reports could be automatically analyzed to extract tagged Bitcoin addresses (e.g., using indicator extraction approaches [47, 69]). The identified addresses could then be automatically explored, and those identified as online wallets or receiving cash-outs from the malware activities could be reported to the exchanges. This would prevent exchanges from claiming they were unaware of the abuse, and allow evaluating the reaction of exchanges to abuse reports.

Address classification. Tagging Bitcoin addresses is fundamental to identify relations. A well-known tagging limitation is low coverage, as only a very small percentage of addresses is tagged. Furthermore, tagging is often delegated to commercial services (e.g., [15, 16]), which use proprietary approaches to build their tag databases and do not disclose how they handle challenges such as double-ownership and potentially conflicting tags. We believe open research on building large, accurate, tag databases is fundamental to allow tagging to sustain challenges likely to appear in court cases. Furthermore, we have shown how to build classifiers to identify certain types of addresses such as those belonging to exchanges and those used by malware for C&C signaling. A possible avenue of future work is to build classifiers for many other address types.

Evasion. An attacker could evade our approach by making its addresses look uninteresting, e.g., like exchanges. However, feature analysis shows that the high volume of transactions is a key feature of the exchange classifier, so impersonating an exchange would be expensive. Alternatively, an attacker could pollute our tag database so that some of its addresses are considered benign services. Evasion is also possible by using mixers since they prevent forward exploration. However, we have shown that backwards exploration reveals their clients. Yet another evasion would pollute the graphs with unrelated transactions, e.g., by using sophisticated dusting techniques that our approach may fail to filter. This would make the graphs less readable, but would not prevent automated extraction of relations.

Other blockchains. Our approach generalizes to other cryptocurrencies following Bitcoin’s UTXO model and scripting language. BlockSci already supports variants of Bitcoin: Bitcoin Cash, Dash, Litecoin, Namecoin, and ZCash. Supporting them would require extending our tag database with addresses for those blockchains and to retrain the exchange classifier.

9 CONCLUSION

Our study of the financial relationships of 30 malware families sheds light on cybercrime’s financial inner workings. Our novel back-and-forth exploration identifies important relations missed by forward transaction tracing such as previously unknown C&C signaling addresses, a wealth of relations to external services and other cybercrime operations, and new attribution points.

For this, we have proposed a novel back-and-forth exploration that given a set of seed addresses belonging to a cybercrime campaign outputs a transaction graph and identifies paths in the graph corresponding to relations with other entities. To enable back-and-forth exploration, we have built a database of 230K tagged addresses from open sources and expanded it into 116M tagged addresses through multi-input clustering, and also a ML classifier to identify exchange addresses. We have also proposed operation oracles that can identify certain types of malicious addresses based on sequences of distinctive transactions. We have evaluated back-and-forth exploration using 30 malware families identifying a wealth of relations to external services and other cybercrime.

Finally, the detailed graphs and relations our approach produces could be used as evidence to warrant interventions.

ACKNOWLEDGMENTS

This work has been partially supported by the Madrid Regional Government as part of the program S2018/TCS-4339 (BLOQUES-CM) co-funded by EIE Funds of the European Union. This work has been partially supported by the Spanish Government through the SCUM Project (RTI2018-102043-B-I00) MCIN/AEI/10.13039/501100011033/ERDF and through grant PRE2019-088472/MCIN/AEI/10.13039/501100011033 co-funded by ESF. It has been partially supported by the grants HACRYPT and IJIC2019-041599-I/MCIN/AEI/10.13039/501100011033 and European Union NextGenerationEU/PRTR. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors or originators, and do not necessarily reflect the views of the sponsors.
[63] Michele Spagnuolo, Federico Maggi, and Stefano Zanero. 2014. BitIodine: Extracting Intelligence from the Bitcoin Network. In Financial Cryptography and Data Security.

[64] Tsuyoshi Taniguchi, Harm Griffioen, and Christian Doerr. 2021. Analysis and Takeover of the Bitcoin-Coordinated Pony Malware. In ACM ASIA Conference on Computer and Communications Security.

[65] Ege Tekiner, Abbas Acar, A. Selcuk Uluagac, Engin Kirda, and Ali Aydin Selçuk. 2021. SoK: Cryptojacking Malware. In IEEE European Symposium on Security and Privacy.

[66] Taylor Tepper and John Schmidt. 2021. Best Crypto Exchanges For 2021. https://www.forbes.com/advisor/investing/best-crypto-exchanges/.

[67] US Department of Justice. 2017. Russian National And Bitcoin Exchange Charged In 21-Count Indictment For Operating Alleged International Money Laundering Scheme And Allegedly Laundering Funds From Hack Of Mt. Gox. https://www.justice.gov/usao-ndca/pr/russian-national-and-bitcoin-exchange-charged-21-count-indictment-operating-alleged.

[68] Haohua Sun Yin and Ravi Vatrapu. 2017. A First Estimation of the Proportion of Cybercriminal Entities in the Bitcoin Ecosystem using Supervised Machine Learning. In IEEE International Conference on Big Data.

[69] Ziyun Zhu and Tudor Dumitras. 2018. ChainSmith: Automatically Learning the Semantics of Malicious Campaigns by Mining Threat Intelligence Reports. In IEEE European Symposium on Security and Privacy.

Table 10: Clipper (top) and ransomware (bottom) profits measured on seeds.

| Family                  | Addresses | Funds Received | Dates              |
|-------------------------|-----------|----------------|--------------------|
|                         | All  | wTX | BTC  | USD | Start | End  |
| clipa                   | 9,412| 375 | 17,974  | 217,514  | 2018-08-21 | 2021-02-23 |
| cliptomaner             | 1    | 1   | 16,1839  | 216,834  | 2019-11-17 | 2021-02-23 |
| ad0                     | 15   | 15  | 19,2527  | 175,473   | 2015-03-31 | 2021-02-21 |
| phorpiextldr            | 8    | 7   | 11,4699  | 66,482    | 2016-08-14 | 2021-02-06 |
| masad                   | 2    | 2   | 5,4072   | 46,850    | 2019-01-24 | 2021-02-20 |
| clipbanker              | 1    | 1   | 9,2829   | 36,425    | 2016-04-22 | 2021-02-20 |
| cryptoshuffler          | 1    | 1   | 23,5951  | 26,083    | 2016-09-27 | 2021-02-11 |
| predatorsthief          | 1    | 1   | 2,3695   | 22,539    | 2018-09-09 | 2020-11-15 |
| avicorl                 | 1    | 1   | 5,9413   | 21,872    | 2018-07-19 | 2019-03-24 |
| mekotio40               | 1    | 1   | 4,3212   | 14,614    | 2018-01-29 | 2021-02-08 |
| casbancito              | 1    | 1   | 1,3246   | 8,922     | 2019-03-07 | 2021-01-07 |
| protonbot               | 2    | 2   | 9,4248   | 7,339     | 2018-04-17 | 2020-11-25 |
| mispadu                 | 1    | 1   | 0,4752   | 5,235     | 2019-09-28 | 2021-01-12 |
| mekotio                 | 1    | 1   | 0,3088   | 3,725     | 2020-06-25 | 2021-02-11 |
| androidclipper          | 1    | 1   | 0,1287   | 942       | 2018-02-23 | 2018-12-06 |
| aggah                   | 1    | 1   | 0,0130   | 141       | 2020-05-18 | 2020-08-26 |
| bitcoingruber           | 1    | 1   | 0,0082   | 71        | 2019-06-12 | 2019-10-28 |
| locky                   | 7,076| 7,074| 16,353    | 8,240,542 | 2016-01-14 | 2017-05-04 |
| dmalocker               | 10   | 10  | 1,131,762 | 1,072,132 | 2015-12-28 | 2017-09-21 |
| samsam                  | 45   | 20  | 435,678  | 795,434   | 2016-01-13 | 2018-01-19 |
| globeimposter           | 3    | 1   | 594,4195  | 260,862    | 2014-11-23 | 2017-12-27 |
| wannacry                | 6    | 5   | 59,5967  | 118,460   | 2017-03-31 | 2021-02-05 |
| cryptexx                | 1    | 1   | 94,8582  | 60,604    | 2016-06-04 | 2016-08-23 |
| cryptotorlocker2015     | 1    | 1   | 5,6998   | 1,268     | 2015-01-18 | 2015-02-14 |
| Total                   | 7,142 | 7,112 | 18,675,7790 | 10,589,322 | 2014-11-23 | 2021-02-05 |

Figure 7: Exchange classifier ROC curve.
Detecting Cybercriminal Bitcoin Relationships through Backwards Exploration

| Table 11: Top-20 Tags source |

| Seeds | Graph | Classification |
|-------|-------|----------------|
| Class | Operation | All OW Exp. | Comp. Addr | Txs | Unexp. | Tagged | Exchanges | Relations | Runtime |
| C&C   | cerber  | 4  0  4 | 1  448  904  0 | 5  320 | 0  1'00" |
|       | glupteba | 2  0  2 | 2  24  24  0 | 2  6 | 0  0'23" |
|       | joey    | 4  0  4 | 2  27  1,548  1 | 17  2 | 0  0'33" |
|       | skidmap | 1  1  0 | 1  1  0  0 | 1  0 | 1  0'00" |
| Clipper | agah  | 1  0  1 | 1  7  8  1 | 3  2 | 1  0'21" |
|        | androidclipper | 1  1  0 | 1  1  0  0 | 1  0 | 1  0'09" |
|        | azorult | 1  1  0 | 1  1  0  0 | 1  0 | 1  0'09" |
|        | bitcoingrabber | 1  0  1 | 1  94  305  7 | 20  9 | 5  0'47" |
|        | cashaneiro | 1  0  1 | 1  59  96  2 | 32  11 | 4  1'03" |
|        | chlbanker | 1  1  0 | 1  0  0  0 | 1  0 | 1  0'11" |
|        | clippa | 1  1  0 | 1  7  8  1 | 3  2 | 1  0'21" |
|        | cliptomaner | 1  1  0 | 1  0  0  0 | 1  0 | 1  0'09" |
|        | cryptorshiffer | 1  0  1 | 1  507  2,351  18 | 134  180 | 12  4'10" |
|        | masad | 2  2  0 | 1  471  1,478  22 | 99  170 | 11  3'37" |
|        | mekoto | 3  0  2 | 2  146  385  8 | 35  53 | 17  1'12" |
|        | mekotion40 | 3  0  3 | 3  145  311  1 | 25  46 | 8  7'29" |
|        | mispda | 1  1  0 | 1  0  0  0 | 1  0 | 1  0'00" |
|        | mrpr0gr4mmer | 1  0  1 | 1  0  0  0 | 1  0 | 1  0'00" |
|        | n40 | 1  1  0 | 1  577  1,570  7 | 274  145 | 2  1'21" |
|        | phorpiex-tldr | 15  14  1 | 1  11  55  0 | 6  2 | 3  0'21" |
|        | phorpiex-trik | 8  1  6 | 4  103  189  1 | 16  37 | 8  0'58" |
|        | protonbot | 2  2  0 | 1  2  0  0 | 2  0 | 2  0'00" |
| Ransomware | cryptotorlocker2015 | 1  0  1 | 1  147  2,137  17 | 60  23 | 8  115'22" |
|        | cryptox | 1  0  1 | 1  621  1,836  9 | 115  233 | 7  2'38" |
|        | dinalocker | 1  1  1 | 1  482  1,372  1 | 91  145 | 4  1'22" |
|        | globepimposer | 3  0  1 | 1  601  5,240  15 | 72  249 | 4  3'01" |
|        | locky | 7076  0  0  7074 | 1  7,987  11,686  29 | 7,207  200 | 12  12'08" |
|        | sansam | 45  0  20 | 15  303  533  5 | 55  77 | 5  0'47" |
|        | wannacry | 6  0  5 | 2  30  520  0 | 13  10 | 0  0'24" |

| Table 12: Summary of forward-only explorations for the 30 malware families. |
| Class | Operation          | Relations Found                                                                 |
|-------|--------------------|-------------------------------------------------------------------------------|
| C&C   | cerber             |                                                                              |
|       | eraser             | 0 -                                                                            |
|       | ponny              | 0 -                                                                            |
|       | skidmap            | 1 exchange:coinsdc*                                                           |
|       | agghab             | 1 exchange:binance                                                            |
|       | androidclipper     | 1 exchange:huo.com*                                                            |
|       | bitcoingrabber     | 5 exchange:bitmex, exchange:coinbase, exchange:okex, exchange:yobit.net, payment:payeer |
|       | cashanero          | 4 exchange:bittrex.com, exchange:exmo, exchange:localbitcoins, payment:coinpayments |
|       | clipbanker         | 1 exchange:coinbase*                                                           |
|       | clipmaper          | 1 exchange:tradeorange*                                                        |
|       | cryptosshuffler    | 12 exchange:bittrex, exchange:btc-e.com, exchange:bttarin.com, exchange:cripto pay.me, exchange:cubits.com, exchange:exmo, exchange:poloniex, exchange:spectrocoin, mining:btc-pool, payment:binance, payment:coinpayments, payment:webmoney |
|       | masad              | 11 exchange:any-cash, exchange:binance, exchange:bitlato, exchange:coinbase, exchange:criptonator, exchange:localbitcoins, exchange:whitebit, exchange:z-exchange:ru, gambling:1xbit, payment:payeer, payment:coinpayments |
|       | mekotio            | 17 clipper: mketion40*, exchange:binance, exchange:bitco, exchange:bittrex, exchange:btcuk, exchange:coinbase, exchange:haftbc, exchange:localbitcoins, exchange:mercadobitcoin, exchange:paykop, gambling:1xbit, misascl:mrpr0gr4mmer, mixer:wasabi-fee, payment:coinpayments, service:gate.io, service:muchbetter |
|       | mekotio40          | 8 clipper: mekotio*, exchange:criptonator, exchange:dsx-exchange, exchange:mercadobitcoin, gambling:foxbit, gambling:yolo-group, mixer:wasabi-fee, payment:coinpayments |
|       | mispadu            | 1 exchange:mercadobitcoin*                                                      |
|       | mrpr0gr4mmer       | 2 exchange:binance, payment:coinpayments                                        |
|       | n40                | 1 exchange:mercadobitcoin*                                                      |
|       | phorpiex-tdr       | 3 exchange:criptonator*, gambling:duckdice.io, misascl:saveyourself             |
|       | phorpiex-trik      | 8 exchange:bitcoin.de, exchange:coindn, exchange:criptonator*, exchange:localbitcoins, exchange:matrea.com, exchange:poiloniex, gambling:rollin.io, payment:webmoney |
|       | predatorthethief   | 9 clipper: masad, exchange:binance, exchange:bitlato, exchange:btcbank.com.ua, exchange:criptonator, exchange:localbitcoins, exchange:totalcoin, exchange:whitebit, payment:coinpayments |
|       | protonbot          | 2 exchange:criptonator*, exchange:haftbc*                                      |
| Ransomware | crypotolocker2015 | 8 exchange:bitcoin.de, gambling:bitcoin-dice, gambling:fortunejack.com, gambling:luckybit, gambling:satoshinotes, gambling:satoshidice, mining:deepbit, tor:market:silkroad2market |
|       | crypppx            | 11 exchange:bitcoin.de, exchange:bttlixir, exchange:bttrex, exchange:btc-e.com, exchange:coindb, exchange:criptonator, exchange:coins.co.th, exchange:huobi, exchange:localbitcoins, exchange:okcoin.com, exchange:xapo |
|       | dmlocker            | 6 exchange:bitcoin.de, exchange:localbitcoins, exchange:okex, exchange:poloniex, gambling:cloudbet.com, payment:coinpayments |
|       | globemimposter     | 1 exchange:tradeorange, exchange:shapeshift                                  |
|       | locky              | 15 exchange:bittrex, exchange:bitstamp, exchange:btc-e.com, exchange:coinbase, exchange:coins.co.th, exchange:estio, exchange:localbitcoins, exchange:matrea.com, exchange:poloniex, exchange:xxz.biz, gambling:cloudbet.com, mixer:bitcoinlog, payment:binance, payment:coinpayments, payment:webmoney, payment:tor:market:alphahubmarket |
|       | samsam             | 6 exchange:bitcoin.de, exchange:coins.com, exchange:localbitcoins, exchange:poloniex, exchange:yobit.net, tor:market:alphahubmarket |
|       | wannacry           | 2 exchange:tradesheep, exchange:shapeshift                                    |

Table 13: Summary of relationships found with forward-only explorations. Tags with an asterisk at the end were found by multi-input clustering on the seeds, the rest were found by the exploration.
Detecting Cybercriminal Bitcoin Relationships through Backwards Exploration

Table 14: Mapping between short name and full address.

| Operation | Short Name | Address |
|-----------|------------|---------|
| emptystring | 3J98t1 | emptystring 3J98t1WpEZ73CNmQviecrnyiWrnqRhWNLy |
| binance | 1NDyJt | 1NDyJtjmw5k5PNaHgJmuMH4Djgo5bVz |
| cerber | 1CpTCV | 1CpTCVcjkgNF3D7PsAqPv3AcnuAv4DMxnt |
| cerber | 1DRvu | 1DRvuCl/L1Yuau9NL4LH3534u9UECSS |
| cerber | 1CncLk | 1CncLkL7C3iuur9n6133ww5585QNe6n5Y6e |
| cerber | 1HTDy9 | 1HTDy99kfw69NCFXA9e8bVFCNY3Gpnmn94b |
| cerber | 1ML54w | 1ML54w15G3atlHFFHw6Yv5n0w4Gw1Y58 |
| glupteba | 15y7ds | 15y7dsT5NY5x6ck33gs9yYVY4K5HgZC |
| glupteba | 1CgPCp | 1CgPCp3E9399ZFodMnTSSvaf5TpGiym2N1 |
| glupteba | 1CUhaT | 1CUhaT3AIPF486fse6WeYv3j9nL9aD57 |
| glupteba | 34Rqyw | 34Rqyw9s4H4VPN6mV6wGawFw9w5WBW |
| pony | 19hi8B | 19hi8BbKc8k54CMv55s56v1Fw55h5YCC |
| pony | 1BkeGq | 1BkeGqpo8M5KNVYXW3obmQt1R58zXAqLBQ |
| pony | 1CeLgF | 1CeLgF979179aureBZ8AYR5955bov9Y9 |
| pony | 1GLi6N | 1GLi6Ng3vDcvc1C63Wknm5c557KGB5v9c |
| pony | 1N09ALZ | 1N09ALZg6yq6FQG4DxYMY317PGM5QvY590d |
| pony | 1NQ8T | 1NQ8TvEs1K6c8pnl4U9Y744btP155PZvQToq |
| pony | bc1q6m | bc1q6m2q9m6iok64as33maaq9z9n4wetc5type49qeps |
| pony | bc1q6g | bc1q6g9x7yr9v9rd9v9m4n0t9ltm99f9vlgf9n8n5y57 |
| pony | bc1q6h | bc1q6h956e6w72bq2339a230z0e9wwg5k9v9g |
| pony | bc1q6d | bc1q6dM958tck58ew9qk3w4q65n35y327y7y947 |
| pony | bc1q6c | bc1q6c19v44x4x32v23kmc3n759u8vd3c |
| pony | 3S8s | 3S8sVw2QPs217dPGDX3S9Q79T7k5a7m |
| phorpiextldr | phorpiextldr | phorpiextldr1849YVdKvT78dbWYFS5ZBbwKjy9YvN3R |
| phorpiextldr | phorpiextldr | phorpiextldr1GWChunYF7w67z5C6Cj6rRw99glU9A89 |
| phorpiextldr | phorpiextldr | phorpiextldr1BH5hC72CPCQ8xyV3r9yvst9zqMDmMu |
| phorpiextldr | phorpiextldr | phorpiextldr1Y8Vzvzvzv1NOCYxAQ688y727L2PL9n8 |
| phorpiextldr | phorpiextldr | phorpiextldr1EFA3V5vS866sB34ZRYGxvYN1X7pt9QZ3F |
| phorpiextldr | phorpiextldr | phorpiextldr1GlALvE6P75886f9847u792y7eUEZ6P |
| phorpiextldr | phorpiextldr | phorpiextldr1IK4R05F7h9h9n6vCL8HSVdCy61QUUN |
| phorpiextldr | phorpiextldr | phorpiextldr1CSBcPs6yH3b6T6bZB8H38YL6G5F85tV |
| phorpiextldr | phorpiextldr | phorpiextldr1I1910Ny9q9q9y9ys8Z7H8R5m53m51 |
| phorpiextldr | phorpiextldr | phorpiextldr3GvBvLdV3gDy8X5k3G3796bXwpdV8y |
| phorpiextldr | phorpiextldr | phorpiextldr1Q5O5e66Qg3n2e5L6b7t7779y9W63nMn8u |
| phorpiextldr | phorpiextldr | phorpiextldr1K666hG19797w7797VYK599Pav99Km |
| phorpiextldr | phorpiextldr | phorpiextldr1N1s3H35zklkz3La2A963d9ZNNYu5V26e |
| phorpiextldr | phorpiextldr | phorpiextldr13E12V6f54576576V57S9iH8E6VQ4T3Y |
| phorpiextldr | phorpiextldr | phorpiextldr3EG88TmGmmcmP3m39v3Qkm8v65M8y |
| phorpiextldr | phorpiextldr | phorpiextldr3G5R6H99F88n35QQ55R6Bb9P6vYeX |
| phorpiextldr | phorpiextldr | phorpiextldr3Pnks1f6s6w6U6A9DFD1B6vXoEndZ5gq |
| phorpiextldr | phorpiextldr | phorpiextldr3MT3g3j3h95u3A3583v35NT25Z |
| phorpiextldr | phorpiextldr | phorpiextldr1k459W9GZX173TuA7un9w8Y3W3L8Vgdj |
| phorpiextldr | phorpiextldr | phorpiextldr1MNWk6hF9b3u8w85r9b9559xux55WGD |
| mrpr0gr4mmer | mrpr0gr4mmer | mrpr0gr4mmer1Q85W7n86n8Jm5Qn99PHRQ4WQ8S3L1566s4tV8 |
| mrpr0gr4mmer | mrpr0gr4mmer | mrpr0gr4mmer1A55p9kV6xW1Nq9x6yN6WLMo5Mr60n1x69 |
| mrpr0gr4mmer | mrpr0gr4mmer | mrpr0gr4mmer15W92JF49z8yG8G2dGx87GwW799 |
| mrpr0gr4mmer | mrpr0gr4mmer | mrpr0gr4mmer18792Pp648x575x62zC5D9V7479 |
| mrpr0gr4mmer | mrpr0gr4mmer | mrpr0gr4mmer15652z5G5b19W44T5bu5b656Y55B9 |
| mrpr0gr4mmer | mrpr0gr4mmer | mrpr0gr4mmer1MT3xhuPn6851g656W7hjd9VLEDsM9h |
| mrpr0gr4mmer | mrpr0gr4mmer | mrpr0gr4mmer1BRLLb9Br9yVt619V6y6BQG9Mmk5H5 |
| mrpr0gr4mmer | mrpr0gr4mmer | mrpr0gr4mmer1HT36395x62y6F37pm35L9wpA5y |
| mrpr0gr4mmer | mrpr0gr4mmer | mrpr0gr4mmer118699CDlxL15R5yP4Fagb6M4WGM1 |
| mrpr0gr4mmer | mrpr0gr4mmer | mrpr0gr4mmer135b6tHnr3zg57z7nsl5n96U9d |
| mrpr0gr4mmer | mrpr0gr4mmer | mrpr0gr4mmer13B26296567c6y83544964p6v6f9 |
| mrpr0gr4mmer | mrpr0gr4mmer | mrpr0gr4mmer3L1v6u5Gh5D5CVp5Fla5F49t5n6Pht64f9L |
| mrpr0gr4mmer | mrpr0gr4mmer | mrpr0gr4mmer1G4N6hbasL3Y3uP6G0d64DQgj3Xz7F4b |
| mrpr0gr4mmer | mrpr0gr4mmer | mrpr0gr4mmer1K526Bu9A3q54c8an5U5CF5E5P437Q7 |

Table 14: Mapping between short name and full address.