Loporchio *et al. Applied Network Science* (2023) 8:34
https://doi.org/10.1007/s41109-023-00557-4

Applied Network Science

RESEARCH Open Access

Is Bitcoin gathering dust? An analysis of low-amount Bitcoin transactions

Matteo Loporchio[1*], Anna Bernasconi[1], Damiano Di Francesco Maesa[1] and Laura Ricci[1]

*Correspondence: matteo.loporchio@phd.unipi.it

[1] Department of Computer Science, University of Pisa, Largo Bruno Pontecorvo, 3, 56127 Pisa, Italy

Abstract

In the Bitcoin protocol, dust refers to small amounts of currency that are lower than the fee required to spend them in a transaction. Although "economically irrational", dust is commonly used for achieving unconventional side effects, rather than exchanging value. For instance, dust might be linked to on-chain services or to malicious activity, such as dust attacks attempting to break users' pseudonymity. To study this phenomenon, this paper presents an in depth analysis of Bitcoin transactions involving dust, showing how dust is created and consumed. We identify the top dust creators and consumers and discuss how consumption has evolved over time. Finally, we use the data to identify transactions suspected of being part of dust attacks and quantify their impact on address deanonymization. Our results show that dust is mainly related to on-chain betting services. Transactions likely to be part of dust attacks are a minority of dust creating transactions but, despite their relatively low number, they have a disproportionately high effect on helping attackers to break address pseudonymity.

Keywords: Blockchain, Bitcoin, Anonymity, Data analysis

Introduction

Blockchain technology has gained widespread popularity due to its ability of providing trust and transparency without the need for intermediaries. Initally made popular by cryptocurrencies, where they are used to record value and asset transfers, blockchains have eventually been adopted in several other fields of application, including identity management systems, access control systems and supply chain management (Di Francesco Maesa and Mori 2020).

Nakamoto (2008) is perhaps the most popular application of blockchain technology. Originally created by Satoshi Nakamoto in 2008[1], Bitcoin is a decentralized digital currency that enables peer-to-peer transactions without relying on intermediaries such as banks or financial institutions. Users transfer funds to other participants by means of transactions, which are permanently recorded on the Bitcoin blockchain, a public distributed ledger that guarantees the immutability of its contents through cryptographic mechanisms. The process of adding new transactions to the blockchain is called *mining*.

[1] Although the original Bitcoin whitepaper by Satoshi Nakamoto was published in 2008, the first transaction recorded on the blockchain was issued on January 3rd, 2009.

During this process, miners employ their computational power to solve a complex mathematical problem. The first miner solving the problem can add transactions to the blockchain and receives a reward in the form of newly created coins and transaction fees, paid by users to incentivize the inclusion of their transactions in the blockchain.

In this paper we study Bitcoin transactions involving *dust*, small amounts of value that are lower than the fee required to spend them in a transaction. Our analysis is motivated by the fact that, despite its "economically irrational" nature, dust is commonly used for achieving some external side effect, rather than exchanging value. As a result, by studying how dust is created and consumed, it is possible to discover which "unconventional" processes (i.e., different from the usual payments) are using the Bitcoin blockchain for their own purposes. We do note, however, that this phenomenon is not exclusive to Bitcoin, as dust amounts can also originate on other UTXO-based[2] currencies (e.g., Litecoin and Bitcoin Cash (Pérez-Solà et al. 2019)). Nonetheless, this paper will only discuss dust within the Bitcoin ecosystem, as it represents the most relevant currency in terms of users and market capitalization.

For instance, Bitcoin dust transactions can often be associated with *on-chain* services, i.e., applications that interact directly with the blockchain. As an example of such services we mention *Satoshi Dice*, a popular on-chain gambling game launched in 2012. To notify players of their loss, Satoshi Dice sends back a small fraction of the original wager, which is typically a dust amount. However, besides its role in such "honest" activities, Bitcoin dust can also be associated with malicious behavior. In fact, dust transactions can also constitute a source of spam for the blockchain network as they are used to conduct denial-of-service attacks (Bradbury 2013). A typical scenario is the one of attackers issuing many dust transactions to fill up the unconfirmed transaction pool, with the goal of slowing down the transaction confirmation process (Wang et al. 2018). Dust is also associated with *dust attacks*, where an adversary sends small amounts of bitcoin to a large number of addresses in an attempt to deanonymize the owners of those addresses. By leveraging the *multi-input heuristic* (Reid and Harrigan 2013)–a rule which states that all sending addresses of a transaction are likely to belong to the same user–the attacker hopes that users will eventually combine the received dust amounts with funds from other addresses owned by them. When this happens, all addresses involved can be linked to the same user. Moreover, if at least one Bitcoin address is linked to real-world information (e.g., an IP address), the user is deanonymized, meaning that their entire transaction history can be now traced back to a real identity. As a result, since transactions on the blockchain are public, this attack has the disruptive effect of compromising users' privacy by exposing sensitive information (i.e., the history of their payments).

In this work we analyze transaction inputs and outputs to understand how dust is created and consumed within the Bitcoin ecosystem. We then identify the major contributors to dust creation and consumption and trace the evolution of consumption patterns over time. Finally, we present an analysis of transactions that are possibly related to dust attacks, as they hold significant relevance in terms of network security.

[2] Unspent Transaction Output model as explained in the Transactions section.

Our paper extends the previous analysis of dust transactions presented in Loporchio et al. (2023). In our previous work, the analyzed transactions were collected using a new approach based on *authenticated data structures* (Tamassia 2003). The method we proposed guarantees the integrity of the received information and allows resource-constrained nodes to obtain data from the blockchain without downloading the entire transaction history. In addition to a general revision and improvement, this paper extends our previous contribution in the following directions.

1. We create a new Related work section by discussing existing papers that analyze the role of dust in Bitcoin and compare them to our contribution.
2. We expand the Background section by introducing a formal characterization of Bitcoin transactions, following the one proposed in Crowcroft et al. (2021).
3. We add a formal definition of dust, in accordance with the implementation of the *Bitcoin Core* client.
4. We extend our experimental analysis by studying the properties of transactions that are possibly related to dust attacks.

The above enhancements made to our previous work have yielded novel contributions to the existing literature, which can be summarized as follows.

1. We provide a formal definition of dust attack and establish when such attacks can be considered successful or unsuccessful.
2. We identify transactions created by possible attackers, namely those that are primarily aimed at the creation of dust.
3. We differentiate transactions issued by victims by taking into account all those that consume dust previously created by attackers.
4. Using heuristic clustering techniques, we assess the impact of attackers' transactions on users' anonymity.

Paper structure This paper is organized as follows. In the Related work section we present the most relevant works concerning Bitcoin dust and discuss how they relate to our contribution. Then, in the Background section we introduce some fundamental concepts concerning Bitcoin and blockchain technology. In the Bitcoin dust section we present the notion of dust in the Bitcoin blockchain and focus on the concept of dust attacks, whose goal is to deanonymize users. In the Experimental results section we illustrate our experimental analysis and finally, in the Conclusions and future work section, we present our conclusions and discuss possible directions for future work.

Related work

In this section we review the most relevant works concerning Bitcoin dust and discuss how they relate to our proposal.

Pérez-Solà et al. (2019) studied dust as part of their analysis of unspent transaction outputs in Bitcoin, Bitcoin Cash and Litecoin. They determined that a significant portion of unspent outputs for the three examined currencies can be considered "dust" or unprofitable (i.e., spending it would cost more in fees than the value of the output itself)

and that their impact in terms of size is significant. Baqer et al. (2016) conducted an empirical analysis of a spam campaign on the Bitcoin blockchain. They indicated dust outputs as a possible source of spam, as they are used to delay the confirmation of legitimate transactions. Similarly, Wang et al. (2018) and Saad et al. (2019) propose methods for identifying denial-of-service attacks based on dust transactions. In this scenario, malicious users issue a large number of dust transactions that will fill up the unconfirmed transaction pool, thus slowing down the confirmation process. A different perspective is the one presented by Bartoletti and Pompianu (2017), whose work focuses on small amounts of value and investigates their connection with the `OP_RETURN` scripting instruction as a way of writing arbitrary data on the Bitcoin blockchain. Within their analysis of cybercrime financial relationships, Gomez et al. (2022) have proposed an heuristic to identify and remove transactions related to dust attacks. To this aim, they defined transactions issued by attackers as the ones that send the same dust amount to at least 100 different output addresses.

If compared to the previously mentioned works, our paper focuses exclusively on dust amounts. Our intent is to analyze the properties of such amounts as well as their connections with on-chain services and malicious behavior, with particular attention to dust attacks. To the best of our knowledge, our work is the first one presenting a detailed characterization of transactions that are possibly related to dust attacks, both from the attackers' side and from the perspective of victims. Moreover, as far as the authors know, no prior work analyzing the impact on deanonymization of potential dust attack transactions currently exists.

Background

This section presents the key definitions and notation used throughout the paper, which formalize common concepts related to the Bitcoin blockchain.

Bitcoin blockchain

Users take part in the Bitcoin economy through addresses. An address is created by hashing a public key derived from a ECDSA key pair (Johnson et al. 2001). The address and public key are employed for sending and receiving payments, while private keys are used to provide proofs of ownership. Users typically create and use multiple addresses for sending and receiving payments through transactions. All transaction history is permanently recorded in a distributed ledger, called *blockchain*, which is publicly accessible and aims at preventing double spending. More precisely, the Bitcoin blockchain can be thought as an ordered list of blocks. Besides transactions, each block also comprises a header including all information needed to make the chain immutable, such as a cryptographic hash pointer to the previous block and the root of a Merkle Tree (Merkle 1980) which guarantees the integrity of the block content.

Transactions

In a Bitcoin transaction, the value gathered from the input addresses (or newly created, the special case of *coinbase* transactions) is distributed among output addresses, with a portion reserved as the transaction fee. Fees are used to reward miners for their work in validating the block. To represent Bitcoin transactions, in this paper we adopt

the formalization described in Crowcroft et al. (2021). In this regard, a transaction can be modeled as a pair $T = (\text{in}, \text{out})$ where $\text{in}, \text{out} \subseteq \mathcal{I} \times \mathcal{A} \times \mathbb{N}$ are two multisets, $\mathcal{I} \subseteq \mathbb{N}$ is a set of numeric identifiers and $\mathcal{A}$ is the set of all Bitcoin addresses. The in (resp. out) multiset contains a set of transaction inputs (resp. outputs) represented as tuples (i, a, v), where $i \in \mathcal{I}$ is a numerical value that uniquely identifies the transaction[3], $a \in \mathcal{A}$ is the address that is paying (resp. paid) and $v \in \mathbb{N}$ is the amount of value transferred, expressed in satoshi[4]. Coinbase transactions do not contain any inputs (i.e., it is $\text{in} = \emptyset$), as they constitute the source of new bitcoins. Given a non-coinbase transaction T, each of its inputs $(i, a, v) \in \text{in}$ represents a pointer to a previously created transaction output, namely there exists a unique older transaction $T' = (\text{in}', \text{out}')$ such that $(i, a, v) \in \text{out}'$. Moreover, the outputs of a transaction cannot be spent more than once: given a transaction $T' = (\text{in}', \text{out}')$, for each $(i', a', v') \in \text{out}'$ there exists at most one newer transaction $T = (\text{in}, \text{out})$ such that $(i', a', v') \in \text{in}$.

In this paper, we adopt the following notation conventions. Given a transaction T, we denote the transaction fee, namely the difference between the sum of the input and output values, by $\phi(T) = \sum_{(i,a,v)\in\text{in}} v - \sum_{(i,a,v)\in\text{out}} v$. Also, we indicate with $\text{inaddr}(T) = \{a \in \mathcal{A} \mid \exists\, i \in \mathcal{I},\ v \in \mathbb{N} :\ (i, a, v) \in \text{in}\}$ the set of all input addresses of T and by $\text{outaddr}(T) = \{a \in \mathcal{A} \mid \exists\, i \in \mathcal{I},\ v \in \mathbb{N} :\ (i, a, v) \in \text{out}\}$ the set of output addresses. Given a set of transactions $\mathcal{T}$, we represent the set of all inputs as $\text{txin}(\mathcal{T}) = \bigcup_{(\text{in},\text{out})\in\mathcal{T}} \text{in}$ and the set of all outputs as $\text{txout}(\mathcal{T}) = \bigcup_{(\text{in},\text{out})\in\mathcal{T}} \text{out}$. Then, with these definitions in mind, we can easily describe the set of all *unspent transaction outputs* (often abbreviated as UTXO) in $\mathcal{T}$ as $\text{utxo}(\mathcal{T}) = \text{txout}(\mathcal{T}) \setminus \text{txin}(\mathcal{T})$.

Scripts

Transaction outputs are associated with a *script*, i.e., a piece of code written in a stack-based and Turing-incomplete language specifying the conditions that must be met in order for someone to spend the corresponding value in a subsequent transaction. Scripts allow for specifying arbitrarily complex conditions, but many of them follow common patterns. For the purposes of this paper (i.e., the study of dust transactions) we distinguish between the most common script types, as discussed below.

1. Pay-to-Public-Key-Hash (`P2PKH`) scripts are the most common for standard Bitcoin transactions. They are used to pay a specific public key hash.
2. Pay-to-Public-Key (`P2PK`) scripts allow for a direct payment to a public key, rather than its hash.
3. Pay-to-Script-Hash (`P2SH`) scripts are used to pay a hash of another script and they are often used to implement multi-signature transactions.
4. Pay-to-Witness-Public-Key-Hash (`P2WPKH`) scripts have been introduced as part of the *Segregated Witness* (SegWit) protocol upgrade in August 2017 (Singh et al. 2020). SegWit addresses the issue of transaction *malleability*, a flaw in the original Bitcoin design that allowed malicious actors to modify transactions before they were con-

[3] In our dataset we have chosen to replace transaction hashes with unique progressive identifiers for transactions.

[4] A satoshi represents the smallest bitcoin denomination and is equivalent to 10^{-8} bitcoins.

firmed (Decker and Wattenhofer 2014). A `P2WPKH` script enables payments to public key hashes, similarly to the `P2PKH` counterpart.

5 Pay-to-Witness-Script-Hash (`P2WSH`) scripts have also been introduced with SegWit update. Similarly to `P2SH` scripts, these scripts allow users to pay a hash of another script.

6 `OP_RETURN` scripts allow for storing arbitrary data into the blockchain. The instruction is used to create *provably unspendable* outputs that cannot be consumed in a future transaction and do not appear in the UTXO set (Bartoletti and Pompianu 2017).

7 We use the type `OTHER` to label all other kinds of scripts, including standard scripts as Pay-to-Multisignature and all non-standard scripts, such as empty scripts that do not contain any instruction.

For the sake of notation, in the following we assume the existence of a function $script : \mathcal{I} \times \mathcal{A} \times \mathbb{N} \rightarrow \mathcal{S}$ that maps a given transaction output to the corresponding script type, where $\mathcal{S} = \{\texttt{P2PKH}, \texttt{P2PK}, \texttt{P2SH}, \texttt{P2WPKH}, \texttt{P2WSH}, \texttt{OP\_RETURN}, \texttt{OTHER}\}$ is the set of all script type identifiers listed above.

Clustering

Transactions recorded on the Bitcoin blockchain represent interactions between addresses. As stated previously in the Bitcoin blockchain section, users of the Bitcoin network typically generate and use multiple addresses, each associated with no real-world information about the controlling user. This weak protection of users' anonymity is commonly known as *pseudonimity* (Meiklejohn et al. 2013). The pseudonimity property leverages the fact that an address carries no information about its owner, and different addresses of the same owner share no common information between themselves. Attacks towards Bitcoin anonymity then attempt to break such property, mainly by heuristic rules (Harrigan and Fretter 2016). The most widely adopted of such rules is called *multi-input heuristic* (Reid and Harrigan 2013), stating that all input addresses of the same transaction belong to the same user. This rule can be described more formally as follows.

Property 1 (Multi-input heuristic) *Let $\mathcal{T}$ be a set of transactions. Then $\forall\, T \in \mathcal{T}$ and $\forall\, a_1, a_2 \in \text{inaddr}(T)$ it holds that a_1 and a_2 belong to the same user.*

We remark that other heuristic rules for grouping addresses together do exist as well. For instance, the *change address heuristic* (Meiklejohn et al. 2013) states that if a transaction includes change, the change address is likely to belong to the same user owning the input addresses. However, as observed in Zhang et al. (2020), this heuristic rule has some important limitations. For instance, due to the evolution of wallet management software over the years, it is often difficult to identify change addresses. In addition to this, if coins in the inputs are transferred without any change, the heuristic could erroneously identify one of the recipients as a change address. Motivated by these reasons, for the rest of this paper, and in particular for the experimental analysis conducted in the Experimental results section, we will only consider the

multi-input heuristic, as it is more conservative with respect to other heuristic rules. We informally consider conservative a rule that minimizes the grouping of addresses that should not be grouped together (i.e., false positives), at the expense of accepting an increase in not grouping together addresses that should be (i.e., false negatives).

In the literature (Maesa et al. 2017), the process of grouping addresses together in sets belonging to the same user is called *clustering*, and such sets are called *clusters*. Grouping addresses together in clusters allows to deanonymize all addresses belonging to the cluster if just one of the addresses ownership is disclosed. The clustering algorithm first starts with each address in a separate cluster. Then, the considered heuristic rules are applied to a transaction at a time, with the potential effect of merging together clusters previously separated. The end result is a partition of the addresses set in sets of addresses (i.e., clusters) believed to be owned by the same entity. As such, given the set of addresses $\mathcal{A}$ and transactions $\mathcal{T}$, the process of clustering $\mathcal{A}$ with respect to $\mathcal{T}$ can be described by a function $C : \mathcal{A} \times \mathcal{T} \rightarrow 2^{\mathcal{A}}$, where $2^{\mathcal{A}}$ denotes the power set of $\mathcal{A}$. The result of this process is indeed a set $\mathcal{P} \subseteq 2^{\mathcal{A}}$ that constitutes a partition of $\mathcal{A}$. Note that if $\mathcal{A}$ is not specified then it is assumed as the set of addresses implied by $\mathcal{T}$, defined as $\mathcal{A} = \bigcup_{T \in \mathcal{T}} \text{inaddr}(T) \cup \text{outaddr}(T)$, namely the set of all addresses appearing in any input or output of $\mathcal{T}$. Do note that the clustering of a set of Bitcoin addresses with respect to a set of transactions, if performed by only considering the previously described multi-input heuristic, can be computed in time linear in the number of addresses and number of inputs of all transactions, as explained in Di Francesco Maesa et al. (2018).

Bitcoin dust

The term *dust* typically refers to the amounts of value that are smaller than the minimum fee required to spend them in a transaction. We now introduce the formal definition of dust we adopt throughout this paper, namely the one detailed in Pérez-Solà et al. (2019) and based on the implementation of the *Bitcoin Core* client.

Definition 1 (Dust output) Let $T = (\text{in}, \text{out})$ be a Bitcoin transaction and $out = (i, a, v) \in \text{out}$ be one of its outputs. The output *out* is considered dust if and only if $v < f \cdot (41 + 107/\alpha + size(out))$, where:

1. $size(out)$ is the output size, expressed in bytes;
2. f represents the suggested fee-per-byte rate, namely the amount paid for the transaction to be included in the blockchain, measured in satoshi per byte;
3. α is the Segregated Witness discount factor. If T complies with the Segregated Witness soft fork, then it is $\alpha = 4$, otherwise $\alpha = 1$.

As will be discussed in the Experimental result section, our analysis will only consider transactions that took place before the Segregated Witness protocol update. Therefore, for the rest of this paper we will define dust outputs by considering a discount factor $\alpha = 1$. In this regard, as the size of a typical spendable non-Segregated Witness transaction output is equal to 34 bytes and the default rate is equal to 3000

satoshi per kilobyte, by applying the inequality from Definition 1 we can derive the minimum amount v for which an output is not considered dust, namely $(3000/1000) \cdot (41 + 107 + 34) = 546$ satoshi.

There are several ways to create dust on the Bitcoin blockchain. For instance, the popular on-chain betting game Satoshi Dice used to send back a tiny amount of bitcoin (i.e., a small fraction of the original wager) to players who lost their bets to notify their loss (Satoshi Dice: Bitcoin Casino Game). Another source of dust is constituted by transaction outputs with the special `OP_RETURN` redeeming instruction. As discussed in the Scripts section, `OP_RETURN` outputs are provably unspendable outputs often associated with amounts of value that are very small or possibly equal to zero. Provably unspendable outputs are typically created to store arbitrary data on the blockchain, as detailed in Bartoletti and Pompianu (2017).

We can also notice that, according to Definition 1, inputs and outputs with an associated amount equal to zero should also be considered dust. However, one the goals of our analysis is the study of dust attacks and no attack might occur using amounts of zero satoshi, because there would be no reason for someone to spend such amounts in a transaction. Therefore, for the rest of this paper we will label as dust all transaction outputs and inputs whose amount is included in the range between 1 and 545 satoshi, thus excluding zero amounts from our analysis. In addition to this, we introduce the following definition to characterize transactions that produce or consume dust.

Definition 2 (Dust-creating and dust-consuming transactions) Let $T = (\text{in}, \text{out})$ be a Bitcoin transaction. We say that:

- T is *dust-creating* if and only if there exists $(i, a, v) \in \text{out}$ such that $v \in [1545]$, namely T has at least one dust output.
- T is *dust-consuming* if and only if there exists $(i, a, v) \in \text{in}$ such that $v \in [1545]$, namely T has at least one dust input.

Notice that a transaction could be both dust-creating and dust-consuming at the same time. Furthermore, we will say that an address is a dust *receiver* (resp. a *sender*) if and only if it is associated with one of the dust outputs (resp. inputs) of a transaction. More formally, an address $a' \in \mathcal{A}$ is a dust receiver (resp. sender) if and only if for a given transaction $T = (\text{in}, \text{out})$ that there exists $(i, a, v) \in \text{out}$ (resp. $(i, a, v) \in \text{in}$) such that $a = a' \wedge v \in [1545]$.

Dust attacks

As previously discussed in the Clustering section, addresses in the Bitcoin blockchain are pseudonymous, namely they are not tied to any personally identifying information and thus do not reveal anything about their owner per se. This property, however, does not guarantee complete anonymity. The entire payment history is in fact recorded on the blockchain, which is a public ledger. If someone is able to link a real-world identity to a specific address, they will be able to track down all transactions associated with that identity. For this reason, *deanonymization* attacks, which attempt

to break users' anonymity by discovering their real-world identities, have become increasingly popular (Fanti and Viswanath 2017; Biryukov and Tikhomirov 2019).

In this paper we focus on *dust attacks*, which are also referred to as *forced address reuse attacks*, a particular kind of deanonymization attack involving small amounts of value. During a dust attack, an adversary sends tiny amounts of bitcoin to many different addresses. The attacker then hopes that the recipients (or their wallet software) will eventually aggregate these amounts as inputs to a larger transaction. If this is the case, the adversary can exploit the multi-input heuristic, discussed in the Background section, to link all input addresses to the same user. Then, if the attacker succeeds in associating one of these addresses to real-world information (e.g., an IP address obtained with network analysis techniques Fanti and Viswanath 2017), the user is deanonymized, with potentially disruptive consequences on their privacy. Indeed, as transactions on the blockchain are publicly visible, the entire payment history of the user can now be tied to their real identity and exposed to the public. To characterize dust attacks from a more formal standpoint, we present a definition based on their outcome, which may either be success or failure. This definition will be adopted throughout the rest of this paper.

Definition 3 (Dust attack) Let $T_A = (\text{in}, \text{out})$ be a dust-creating transaction from an attacker A and let $(i, a, v) \in \text{out}$ be a dust output of T_A sending value to an address a owned by a user U. We say that the dust attack of A towards U is:

1. *successful*, if there exists a later transaction $T_U = (\text{in}', \text{out}')$ created by U such that the following conditions hold:
   (a) $(i, a, v) \in \text{in}'$;
   (b) $|\text{inaddr}(T_U)| \geq 2$, namely if $\exists\, (i', a', v') \in \text{in}'$ such that $a' \neq a$;
2. *unsuccessful* otherwise.

In other words, the attack is successful if the user U eventually spends the received output (i, a, v) in combination with funds from at least another address $a' \neq a$ owned by them. According to the multi-input heuristic, this allows A to learn that both address a' and a are likely to be owned by the same user U. Conversely, the attack fails if the adversary is not able to learn any new information about the set of addresses owned by the victim. This is the case if the output (i, a, v) is never spent or if it is spent only with other funds from the same address a. As the deanonymization attack based on the multi-input heuristic is driven by multiple addresses appearing as inputs of the same transaction, the goal of a dust attack is to force such appearances by sending amounts that cannot be spent on their own.

Experimental results

In this section we present our experimental analysis of dust in the Bitcoin blockchain. Our contribution in this regard is twofold.

1 First, we present a general study of transactions that create and consume dust. In particular, we focus on dust inputs and outputs and examine how they are distributed across transactions. We then identify the top creators and consumers by computing the most frequent addresses in our data set. Finally, we provide an insight on how the expenditure of dust outputs has evolved over time and discuss how such outputs are combined with other outputs when consumed.
2 Secondly, in the Dust attack analysis section we extend our analysis by identifying transactions that are possibly related to dust attacks. To this aim, we differentiate possibly malicious transactions from other transactions involving dust. We use this classification to study the impact of transactions suspected to be part of dust attacks when performing clustering-based deanonymization attacks.

The code for the experiments has been written in Java and Python and is publicly available on GitHub at https://github.com/mloporchio/DustAnalysis.

Data set description

We considered the data set comprising all $N_{txs} = 245\,410\,083$ transactions in the first $N_b = 479\,970$ blocks of the Bitcoin blockchain, thus covering the time period between January 3rd, 2009 18:15 GMT and August 10th, 2017 18:03 GMT. Notice that all transactions included in this data set were issued before the introduction of the Segregated Witness soft-fork. For the sake of notation, we refer to this data set as the *full* data set and denote it by $\mathcal{F}$. In the following, we denote by

$$\mathcal{T} := \{(\text{in}, \text{out}) \in \mathcal{F} \mid \exists\, (i, a, v) \in \text{in} \cup \text{out} : \ v \in [1545]\}$$

the set of all transactions in $\mathcal{F}$ that create or consume dust. Finally, we indicate with $D_{out} = \{(i, a, v) \in \text{txout}(\mathcal{F}) \mid v \in [1545]\}$ the set of all dust outputs and with $D_{in} = \{(i, a, v) \in \text{txin}(\mathcal{F}) \mid v \in [1545]\}$ the set of all dust inputs of all transactions in $\mathcal{F}$. For ease of comprehension, we have gathered all symbols and abbreviations used throughout this section in Table 9 of the Appendix: symbols and abbreviations. Each symbol is accompanied by a brief explanation of its meaning.

Transaction analysis

In this section, we study the properties of the data set $\mathcal{T}$, containing all dust-creating and dust-consuming transactions. The data set is made up of 2,114,335 elements: more precisely, it comprises 1,705,560 dust-creating transactions (nearly 0.7% of the total number N_{txs}), 429,544 (nearly 0.175% of N_{txs}) that consume it and 20,769 transactions (approximately 0.008% N_{txs}) that simultaneously create and consume dust. First, we examined the average number of dust outputs (resp. inputs) for dust-creating (resp. dust-consuming) transactions and found out that each dust-creating transaction has 2.58 dust outputs on average, while dust-consuming ones have 5.98 dust inputs. Similarly, in order to understand how dust is distributed among transactions, we computed the average percentage of dust outputs (resp. inputs) in dust-creating (resp. dust-consuming) transactions. Our results show that, on average, the percentage of dust outputs in dust-creating is approximately 50.22%, while for dust-consuming ones the percentage of dust inputs is 35.83%.

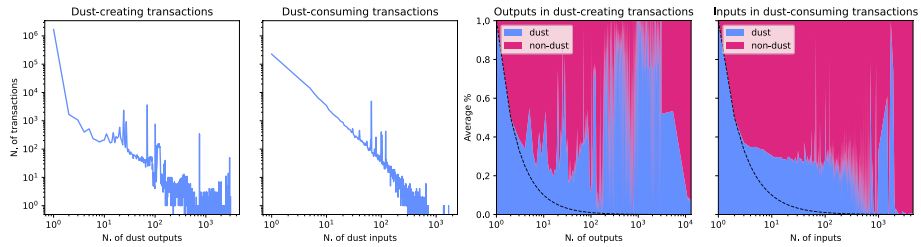


Fig. 1 Properties of dust-creating and dust-consuming transactions

We then evaluated the frequencies of dust outputs (resp. inputs) in dust-creating (resp. dust-consuming) transactions. Our findings are illustrated by the first two plots (from left to right) of Fig. 1. As the reader may notice, approximately 10^6 dust-creating transactions comprise exactly one dust output. Similarly, we can also notice that a consistent number (i.e., between 10^5 and 10^6) of dust-consuming transactions has only one input. More generally, it is possible to observe that transactions with a high number of dust inputs and outputs seem to be less common. From the second plot, it can also be inferred that the relation between the quantity of dust inputs n_i and the number of transactions including n_i dust inputs follows a trend similar to a power law. Finally, in the third (resp. fourth) plot of Fig. 1 we report the average percentages of dust and non-dust outputs (resp. inputs) in dust-creating (resp. dust-consuming) transactions. In both charts, the dashed lines correspond to the minimum percentages for dust outputs and inputs. For instance, since in the third (resp. fourth) plot we are only considering dust-creating (resp. dust-consuming) transactions, in a transaction with k outputs (resp. inputs) the percentage of dust outputs (resp. inputs) will always be at least $1/k$. If we exclude the obvious cases of transactions with one output or one input, we can notice that in both situations there seems not to be any consistent pattern in the partitioning between dust and non-dust, although non-dust amounts seem to be more common among the inputs and in transactions involving less than 10^2 outputs.

Address analysis

In order to determine the main senders and receivers of dust, we analyzed the most frequent addresses in both D_{in} and D_{out}. We then identified the top 5 addresses for each category and compiled our findings in Table 1. Regarding the top senders, we notice that all addresses are associated with Satoshi Dice, the on-chain betting game previously discussed in the Bitcoin dust section. Satoshi Dice addresses can be easily identified since they were generated with mnemonic `1dice` prefixes, and their winning odds (represented by W) and winning multipliers (represented by M) are predetermined and known to the players. As discussed in the Bitcoin dust section, this fact can be easily understood as Satoshi Dice addresses return a small fraction of the initial bet to the player in case of a loss. Additionally, it is worth noting that the activity of the highest receivers are also connected to this gambling game. In Table 1 we computed, for each reaceiver, the number of transactions having at least one Satoshi Dice input address. We discovered that, for 4 addresses out of 5, over 75% of the received transactions originated from a recognized Satoshi Dice address. This suggests that the owners of these addresses are likely to be participants in the gambling game, with the exception of

Table 1 The top 5 dust sender and receiver addresses

	Address	TXs	Notes
Senders	1dice8EMZmqKvrGE4Qc9bUFf9PX3xaYDp	374,464	Satoshi Dice ($W = 48.8281\%$, $M = 2.004\times$)
	1dice97ECuByXAvqXpaYzSaQuPVvrtmz6	177,201	Satoshi Dice ($W = 50.0000\%$, $M = 1.957\times$)
	1dice6YgEVBf88erBFra9BHf6ZMoyvG88	127, 790	Satoshi Dice ($W = 12.2070\%$, $M = 8.000\times$)
	1dice7fUkz5h4z2wPc1wLMPWgB5mDwKDx	123,005	Satoshi Dice ($W = 24.4141\%$, $M = 4.003\times$)
	1dice1e6pdhLzzWQq7yMidf6j8eAg7pkY	93,724	Satoshi Dice ($W = 0.0015\%$, $M = 64000\times$)
Receivers	1PEDJAibfNetJzM289oXsW1qLAgjYDjLgN	14,337	10,758 from SD (75%)-Satoshi Dice gambler
	15tcsumFPsrw2p9Egmk7wGszFJVxpw7UiD	11,131	11,126 from SD (100%)-Satoshi Dice gambler
	18d3HV2bm94UyY4a9DrPfoZ17sXuiDQq2B	8099	0 from SD (0%)-Eligius offline wallet
	14z1fVwxMG71WcijX9J9te8G1wyp7tVqdz	7628	7626 from SD (100%)-Satoshi Dice gambler
	1FE1CDgkqzMSFXFreXrET7hvhEfCP9QabY	3739	3739 from SD (100%)-Satoshi Dice gambler

18d3HV2bm94UyY4a9DrPfoZ17sXuiDQq2B. This specific address received dust in 8099 distinct transactions, of which 8098 are actually coinbase transactions used by miners to collect the block reward. This leads us to believe that the address may be part of a mining pool and, in fact, it represents the offline wallet of *Eligius*, a mining pool active between 2011 and 2017.

Output analysis

We classified the transaction outputs in D_{out} into three distinct groups, which can be described as follows.

1. *Unspent* outputs, namely those that are still left in the corresponding addresses. This set can be formally expressed as $U := D_{out} \setminus D_{in}$.
2. *Not Only Dust* (NOD), which includes dust outputs spent in combination with at least one non-dust output. These outputs can be characterized as

$$NOD := \{(i, a, v) \in D_{out} \mid \exists\,(\text{in}, \text{out}) \in \mathcal{T} : \ (i, a, v) \in \text{in} \wedge \\ (\exists\,(i', a', v') \in \text{in} : \ v' \notin [1545])\}.$$

3. *Only Dust* (OD), which comprises dust outputs spent exclusively in combination with other dust outputs. Formally, it is:

$$OD := \{(i, a, v) \in D_{out} \mid \exists\,(\text{in}, \text{out}) \in \mathcal{T} : \ (i, a, v) \in \text{in} \wedge \\ (\forall\,(i', a', v') \in \text{in} : \ v' \in [1545])\}.$$

Table 2 reveals that the majority of outputs, namely almost 55%, fall under the NOD category, while approximately 42% of them are still unspent. It is worth noting that our investigation only considers blocks up to August 10th, 2017, so these outputs may have been consumed at a later date. The remaining 3% consists of OD outputs, which are only

Table 2 Classification of dust outputs

	No. of outputs	Percentage (%)
Unspent	1,830,911	41.604
NOD	2,420,707	55.007
OD	149,139	3.389
Total	4,400,757	100.000

spent in conjunction with other dust outputs. The role of NOD and OD outputs has been further analyzed from a temporal perspective in the subsequent paragraph.

Temporal analysis We then conducted a temporal analysis on D_{out} with a dual purpose: (1) computing the average *duration*, i.e., the number of blocks between the creation and consumption of outputs; (2) discovering patterns in the consumption itself. For what concerns their consumption, we discovered that, on average, dust outputs get consumed after 25,165.33 blocks while non-dust outputs from all transactions in $\mathcal{F}$ only last for 3 207.36 blocks. We further analyzed the duration of outputs in the plots of Fig. 2. In the leftmost plot, we reported the distributions of duration for dust and non-dust outputs. More precisely, we counted the number of dust and non-dust outputs that have been spent after a given number of blocks. Note that the plot has a logarithmic scale on the y-axis, while each bin has a width of 10^4 blocks on the x-axis. On the other hand, the rightmost plot shows the percentages of dust and non-dust outputs for each duration. We can observe that dust and non-dust outputs are spent at different speeds, as the percentage of dust outputs depicted in the rightmost chart is not constant. In particular, we can observe how dust outputs take longer to be spent compared to non-dust ones, which is also confirmed by the previously computed mean durations. Indeed, if we consider an average inter-block time[5] of 10 minutes, we obtain a period of approximately 175 days for dust outputs and 22 days for non-dust amounts. Thus, we can observe that non-dust outputs are consumed nearly 8 times faster. This gap is probably due to the fact that spending dust outputs on their own is unprofitable and hence users typically wait until they have collected a sufficient number of such outputs before they can aggregate them into a single transaction. From both plots it is also possible to notice how there are no dust outputs spent after a certain date. This may be an artificial consequence of data collection, as only transaction outputs from very old transactions (i.e., before 2012) could have such duration. At that time, the Bitcoin protocol was not so popular so, although possible, the concept of dust was not yet considered due to the minimal value of the currency.

To discover meaningful patterns behind dust consumption, we first associated each output with the timestamp of the transaction where it has been spent and then count the number of consumed outputs on a yearly basis. The results of this temporal analysis are presented in the leftmost plot of Fig. 3, where each year is divided into quarters for clarity. We can observe that the consumption of NOD outputs began to rise during the second quarter of 2011 and peaked in 2013, with minor peaks in 2014 and 2015. To determine the primary factors contributing to NOD aggregations, we once again

[5] The inter-block time is the time that passes between the creation of two consecutive blocks.

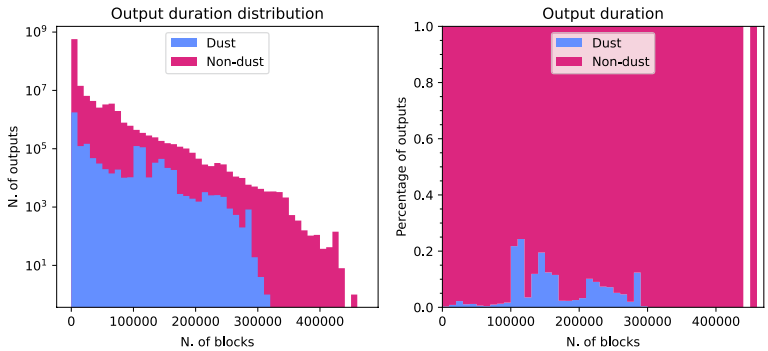


Fig. 2 Output duration

Table 3 Top 5 addresses for NOD output consumption

Address	NOD outputs	From SD	Perc. (%)	Description
1PEDJAibfNetJzM289oX-sW1qLAgjYDjLgN	12,502	10,758	86	Satoshi Dice gambler (see Table 1)
14z1fVwxMG71WcijX9J9te8G-1wyp7tVqdz	7628	7626	100	Satoshi Dice gambler (see Table 1)
18d3HV2bm94UyY4a9DrP-foZ17sXuiDQq2B	7288	0	0	Eligius offline wallet (see Table 1)
1GmREU2gwcvQHRQFgwHvbD-4dyL8iryCPMY	3614	3270	90	Satoshi Dice gambler
1dES7RLppoYc8mLQedwUo-JMZZ9RnuCP5f	3526	3526	100	Satoshi Dice gambler

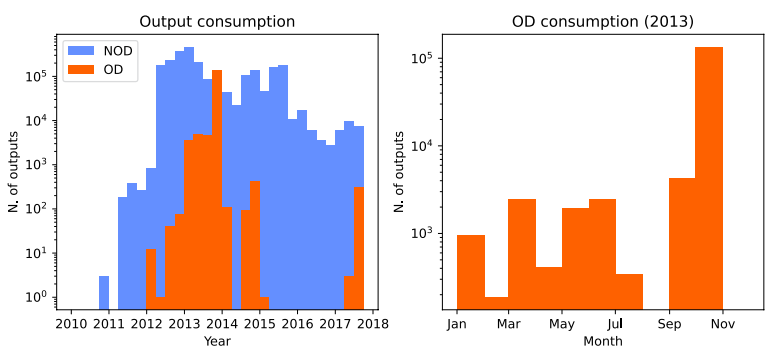


Fig. 3 Temporal analysis of dust output consumption

examined the most frequently occurring addresses. Table 3 lists the top 5 addresses in terms of NOD output usage. The reader may immediately notice that the first three of them have already been identified in Table 1. Moreover, such addresses have previously been associated with mining operations and Satoshi Dice. The remaining two addresses can also be associated with this popular gambling game. Indeed, Table 3 indicates that at least 90% of the NOD outputs associated with these addresses have been sent by a Satoshi Dice address, confirming our previous speculation. Furthermore, from the NOD histogram depicted in Fig. 3, we can observe a sharp spike in

aggregations during the second quarter of 2012. This sudden increase may coincide with the launch of the gambling game, which occurred on April 24th, 2012.

In terms of OD aggregations, the most prolific year was 2013, as evidenced by the histogram on the right-hand side of Fig. 3. More in detail, the histogram exhibits a peak with over 10^5 consumed outputs during the month of October. To discover the origin of such aggregations, we conducted further research on this phenomenon by examining the most frequent addresses and discovered that `1JwSSubhmg6iPtRjtyqhUYYH-7bZg3Lfy1T` consumed 134,693 outputs, which is about 90% of all 149,139 OD outputs, as indicated in Table 2. The interesting fact about this address is that its private key has been compromised[6], which allows anyone to redeem bitcoins as soon as they are sent to it. This leads us to believe that the main reason behind the sudden increase of aggregations from this address is the public disclosure of the address secret key.

Dust attack analysis

In this section we extend our analysis of dust-creating and dust-consuming transactions to identify candidate transactions that are part of a dust attack. However, given that dust inputs and outputs occur normally within the Bitcoin protocol, we would like to differentiate the rarer but impactful attack transactions from all those involving dust amounts.

Data filtering

To achieve our aim, we first filtered the data set $\mathcal{T}$, which contains all dust-creating and dust-consuming transactions, by keeping only those transactions T that satisfy the following two conditions.

(1) T does not contain the address of a *known* entity among its inputs. In our analysis, a known entity is one whose behavior or aim is universally recognized by other participants. Examples could include mining pools and on-chain services (e.g., Satoshi Dice).
(2) At least one of the following is true.
    (a) T has at least one dust output that is not associated with the `OP_RETURN` scripting instruction.
    (b) T has at least one dust input that was not created in a transaction from a known entity.

The result of this filtering step is a new data set $\mathcal{T}'$, formally described as

$$\begin{aligned}\mathcal{T}' := \{T = (\text{in}, \text{out}) \in \mathcal{T} \mid (\text{inaddr}(T) \cap \mathcal{K} = \emptyset) \wedge \\ ((\exists\,(i,a,v) \in \text{out}:\ v \in [1545] \wedge \\ \textit{script}(i,a,v) \neq \texttt{OP\_RETURN}) \vee \\ (\exists\,(i,a,v) \in \text{in}:\ v \in [1545] \wedge \\ (\forall\,(\text{in}', \text{out}') \in \mathcal{T}\ (i,a,v) \in \text{out}' \Rightarrow (\forall\,(i',a',v') \in \text{in}'\ a' \notin \mathcal{K})))\}\end{aligned}$$

[6] Source: https://privatekeys.pw/address/bitcoin/1JwSSubhmg6iPtRjtyqhUYYH7bZg3Lfy1T

Table 4 The top 10 services in terms of created transactions

Entity	Description	No. of transactions	Percentage (%)
SatoshiDice	On-chain betting service	1,464,813	69.280
ePay.info	Faucet service	43,259	2.046
BtcDice.com	On-chain betting service	8500	0.402
DiceOnCrack.com	On-chain betting service	7114	0.336
Betcoins.net	On-chain betting service	3218	0.152
SilkRoadMarketplace	Online black market	2877	0.136
Bitcoin-Roulette.com	On-chain betting service	1742	0.082
Instawallet.org	Online wallet service	1516	0.072
BitZino.com	On-chain betting service	1497	0.071
Cex.io	Exchange service	1337	0.063
Total		1,535,873	72.640

where $\mathcal{K} \subseteq \mathcal{A}$ is the set containing addresses of known entities. Notice that our choice of condition (1) is motivated by the multi-input heuristic. Indeed, if the inputs of a transaction include at least one address of a known entity, we assume that all other input addresses belong to the same entity. Additionally, since the entity is known, we assume that all transactions it creates do not deviate from the purpose for which the service was designed and thus we assume that it does not represent a potential threat for dust attacks. To identify known entities, we employed the *entity-address data set* (Jourdan et al. 2018, 2019), which provides a mapping between Bitcoin addresses and categorical labels describing the entity they belong to. The entity-address data set has been further expanded with the list of all known Satoshi Dice addresses, derived from the official website. With this strategy, we were able to collect 1,550,843 transactions created by 231 different labeled entities. In Table 4 we list the top 10 services according to the number of created transactions, which constitute 72.64% of the full data set. As the reader may notice, the most impactful service is Satoshi Dice, with nearly 1.5 millions of created transactions (nearly 70% of all items in the full data set). This is a further confirmation of our findings presented in the Address analysis section, where we computed the top dust senders and receivers. However, other popular on-chain betting services such as BtcDice, DiceOnCrack, and BetCoins can also be found among the top contributors. Interestingly enough, the list also comprises Silk Road, an online illegal marketplace on the dark web, which was active from 2011 to 2013 and was primarily used to buy and sell drugs. The marketplace heavily relied on Bitcoin payments as a way of hiding the identities of its users (Christin 2013).

On the other hand, the second condition captures all transactions potentially created by attackers and those which might have been generated by victims as a consequence of the attack. More precisely, condition (2a) refers to transactions issued by attackers. In order for their dust outputs to be spent, these should contain at least one which is not associated with the `OP_RETURN` scripting instruction. Instead, condition (2b) refers to transactions that might have been issued by victims: for a dust attack to be successful, we require that the victim spends at least one dust amount created in a transaction that has been issued by an unknown entity.

Table 5 A comparison of dust-creating and dust-consuming transactions in the "original" data set $\mathcal{T}$ and the "filtered" data set $\mathcal{T}'$. Values are computed as average over the corresponding data set

Data set	Dust outputs	Dust inputs	% of dust outputs	% of dust inputs
$\mathcal{T}$	2.58	5.98	50.22	35.83
$\mathcal{T}'$	14.71	4.10	51.57	29.30

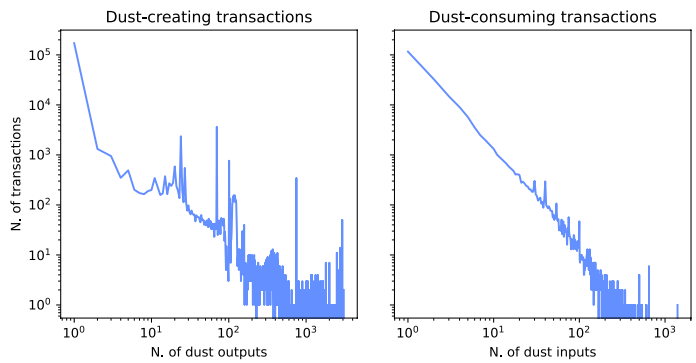


Fig. 4 Distributions of dust outputs in dust-creating transactions (left) and inputs in dust-consuming transactions (right) for data set $\mathcal{T}'$

Transaction analysis

The "filtered" data set $\mathcal{T}'$ is made up of 387,330 total transactions. The data set includes 194,628 dust-creating, 202,448 dust-consuming, and 9746 transactions that simultaneously create and consume dust. Similarly to what has been done in the Transaction analysis section, we examined the number of dust outputs (resp. inputs) in dust-creating (resp. dust-consuming) transactions in $\mathcal{T}'$. We found out that, on average, dust-creating transactions have 14.71 dust outputs, while dust-consuming ones include 4.1 dust inputs. We also computed the average percentage of dust outputs (resp. inputs) for dust-creating (resp. dust-consuming) transactions. In this regard, we discovered that the average percentage of dust outputs in dust-creating transactions is equal to 51.57%. In comparison, the average percentage of dust inputs in transactions that consume dust is 29.3%. In Table 5 we compare these mean values with the ones obtained in the Transaction analysis section for data set $\mathcal{T}$. We observe that the removal of transactions produced by known services has determined a significant increase of the average number of dust outputs, which is now approximately 5.7 times larger. On the contrary, for what concerns dust inputs, the average number of dust inputs in the new data set $\mathcal{T}'$ has decreased slightly.

For a more accurate insight on how dust is distributed among transactions, the leftmost (resp. rightmost) plot of Fig. 4 illustrates the distribution of the number of dust outputs (resp. inputs) in dust-creating (resp. dust-consuming) transactions, with a logarithmic scale on both axes. Despite the removal step, the shapes of both distributions look very similar to their counterparts computed for data set $\mathcal{T}$ and depicted in Fig. 1. Again, as observed for $\mathcal{T}$, the shape of the distribution of dust inputs resembles a power law.

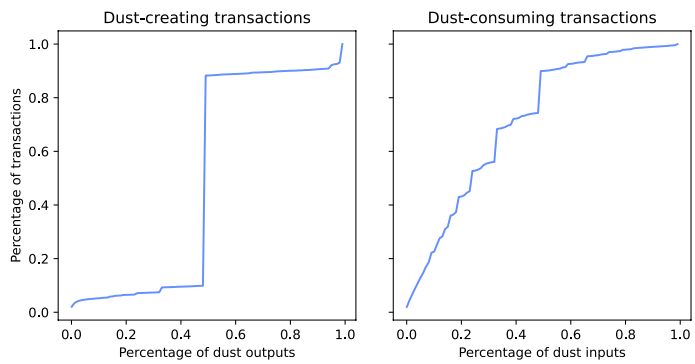


Fig. 5 Cumulative distributions of percentages of dust outputs in dust-creating transactions (left) and dust inputs in dust-consuming transactions (right) for the "filtered" data set

The leftmost (resp. rightmost) plot of Fig. 5 contains the cumulative distribution function for the percentage of dust outputs (resp. inputs) in dust-creating (resp. dust-consuming) transactions. From this plot, we can notice that approximately 90% of all dust-creating transactions have less than 50% dust outputs. The remaining 10%, on the other hand, have more than 50% dust outputs. Similarly, from the rightmost plot we can deduce that approximately 90% of all dust-consuming transactions have less than 50% dust inputs. As a result, transactions where dust outputs (resp. inputs) constitute the majority of all outputs (resp. inputs) seem to be quite uncommon. The percentages of dust outputs and inputs are further investigated in the Candidate identification section, where we identify transactions that might be related to dust attacks.

Candidate identification

The filtering step discussed in the Data filtering section allowed us to remove all transactions generated by well known services. In this section we further refine our data set with a dual objective. First, we would like to identify dust-creating transactions issued by potential attackers. Secondly, we would like to differentiate the ones, created by potential victims, that are spending these dust amounts as a consequence of the attack.

For what concerns potential attackers, we are interested in dust-creating transactions that exhibit "exceptional" behavior, i.e., transactions with dust creation as their main objective. To this aim, for each dust-creating transaction $T \in \mathcal{T}'$ we evaluated the *absolute* number $k(T)$ of dust outputs and the *relative* number $j(T)$, namely the percentage of dust outputs over the total number of outputs of T. Indeed, the higher j and k, the more the transaction's primary purpose is likely to be the creation of dust. Since it is more cost-effective in terms of size, and consequently transaction fees, to create many outputs in a single transaction, instead of splitting them into multiple transactions, a rational attacker would aim to create as many dust outputs as possible within a single transaction. In order to understand what the "exceptional" values for j and k are, in the plots of Fig. 6 we have computed the number of dust-creating transactions for different values of j and k. More precisely, in both grids, a cell with coordinates (x, y) contains the number of dust-creating transactions $T \in \mathcal{T}'$ such that $j(T) \geq x$ and $k(T) \geq y$. Do note that, for better readability, all values in the grids are rescaled by a factor of 10^3 and 10% increments are used on the x-axis.

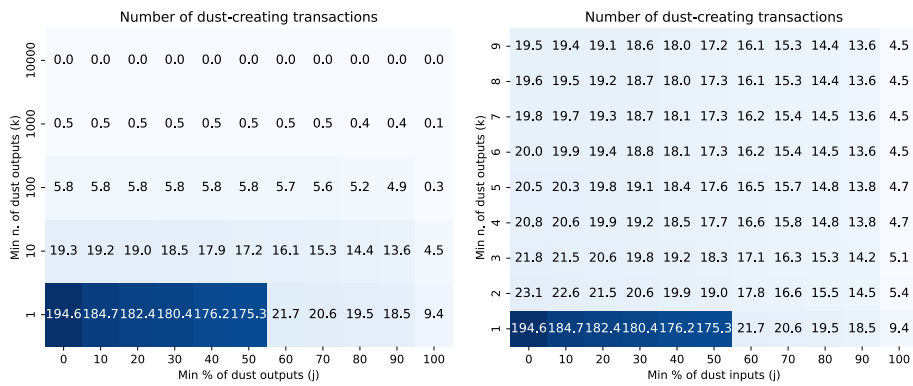


Fig. 6 Number of dust-creating transactions $T \in \mathcal{T}$ with $j(T) \geq x$ and $k(T) \geq y$ as x and y vary. All values have been rescaled by a factor of 10^3

From the leftmost plot of the figure we can see that the increase in both j and k mostly has a linear impact on the resulting number of transactions. An exception in this regard is constituted by all values in the darker areas of the plot, corresponding to $k \geq 1$ dust outputs and a percentage ranging from a minimum of 0% to a maximum of 50%. This fact becomes more evident if we look at the rightmost plot of Fig. 6, where we focused on the values of k in the range from 1 to 9. From this representation, the reader may notice that when the minimum number of dust outputs increases from 1 to 2 the total number of transactions decreases suddenly of an order of magnitude. A similar decrease can be found when the minimum percentage j increases from 50% to 60%. For other values, except the last column containing rarer only dust output transactions, the change is way less dramatic. As such we believe that $k \geq 2$ and $j > 50\%$ could be the most significant thresholds for discerning dust-focused transactions, primarily aimed at creating dust, from all transactions involving dust. The clear demarcation in the data considering these thresholds is a good indication of demarcation in transaction type. As a result, in the following we will call

$$\mathcal{T}_{sus} = \{T = (\text{in}, \text{out}) \in \mathcal{T}' \mid (\exists\, (i, a, v) \in \text{out} : \ v \in [1545]) \ \wedge \ k(T) \geq 2 \ \wedge \ j(T) > 50\%\}$$

the set of *suspect* dust-creating transactions, namely those T with $k(T) \geq 2$ and $j(T) > 50\%$. The set includes 18,840 transactions, which accounts for 4.86% of all transactions in $\mathcal{T}'$.

On the other hand, to identify all transactions issued by potential victims as a consequence of a dust attack, we partitioned the 256,906 dust-consuming transactions included in $\mathcal{T}'$ into three different sets, based on the nature of their inputs.

1 *Type 1*: the inputs of these transactions contain only one distinct address.

$$\mathcal{T}_1 := \{T = (\text{in}, \text{out}) \in \mathcal{T}' \mid (\exists\, (i, a, v) \in \text{in} : \ v \in [1545]) \ \wedge \ |\text{inaddr}(T)| = 1\}$$

2 *Type 2+*: the inputs of these transactions have at least two distinct addresses. More formally, it is:

$$\mathcal{T}_{2+} := \{T = (\text{in}, \text{out}) \in \mathcal{T}' \mid (\exists\, (i, a, v) \in \text{in} : \ v \in [1545]) \ \wedge \ |\text{inaddr}(T)| \geq 2\}.$$

Table 6 Classification of dust-consuming transactions on the basis of their inputs

Category	N. of TXs	Percentage (%)
1	86,284	42.620
2+	116,150	57.373
S	14	0.007
Total	202,448	100.000

3 *Type S*: this category comprises all dust-consuming transactions where the sum of all input amounts equals the fee. This set can be defined as

$$\mathcal{T}_S := \left\{ T = (\text{in}, \text{out}) \in \mathcal{T}' \;\middle|\; (\exists\, (i, a, v) \in \text{in} : \; v \in [1545]) \;\wedge \sum_{(i,a,v)\in\text{in}} v = \phi(T) \right\}.$$

In this classification, type 2+ transactions may represent the consequence of a successful dust attack. Indeed, as discussed in the Dust attacks section, such transactions may reveal other addresses owned by the spending entity, according to the multi-input heuristic. On the other hand, transactions in the type 1 category might be related to an unsuccessful dust attack. In this case, in fact, users are aggregating dust from the same source, namely from the same address the dust amount was sent to by the potential attacker. Concerning type S transactions, their connection with dust attacks seems to be unlikely, since no amount of value is actually transferred to a recipient and everything in the inputs is paid as fee. Such transactions may be linked to dust-collecting services such as *Dust-B-Gone*, which allow users to clean up their wallets from dust. Indeed, a dust-collecting service typically aggregates dust from multiple users in a single transaction which spends all funds into mining fees. With this strategy, the multi-input heuristic no longer holds and thus potential attackers are not able to learn any meaningful information about the identity of the transaction senders.

The results of our classification are reported in Table 6. As the reader may notice, the majority of dust-consuming transactions (approximately 57%) belongs to the type 2+ category, i.e., it has at least two distinct input addresses. Since they are aggregating different addresses in their inputs, we can observe that transactions in this category constitute good candidates for being exploited by the multi-input heuristic. Conversely, we can observe that about 43% of the classified transactions have only one unique input address, while only 14 transactions (which correspond to 0.007% of all dust-consuming transactions) belong to the S category, whose inputs funds are entirely spent in fees.

Taking into account this classification of dust-consuming transactions and our previous definition of suspect dust-creating transactions, we can now characterize the set of all dust-consuming transactions that may correspond to successful dust attacks. As stated in Definition 3, in a successful dust attack victims aggregate previously received dust outputs with other funds from other addresses owned by them. This intuitive description leads us to define the set of *candidate* dust-consuming transactions

$$\mathcal{T}_C := \{T = (\text{in}, \text{out}) \in \mathcal{T}_{2+} \mid \exists\, (\text{in}', \text{out}') \in \mathcal{T}_{sus} : \; \text{in} \cap \text{out}' \neq \emptyset\}$$

including all type 2+ transactions that are spending at least one output from a suspect dust-creating transaction. According to our characterization of dust attacks, we were able to identify 89,102 candidate transactions, which account for approximately 44% of all dust-consuming transactions.

Clustering analysis

In this section we study the impact of candidate transactions on clustering-based deanonymization attempts. As discussed in the Dust attacks section, we recall that a successful dust attack aims at strengthening the effectiveness of the multi-input heuristic by forcing dust receiving addresses to be grouped with other addresses in a single transaction. However, the attack could be considered really successful if such addresses were not already grouped together in the same cluster without the transaction caused by the attack. In other words, the attack goal is to force the common use of addresses that were previously not known to belong to the same user, and so to the same cluster.

To evaluate this effective successfulness of dust attacks on Bitcoin, we have performed a partial deanonymization attack on our data set by executing the multi-input heuristic clustering algorithm on three sets of transactions, and corresponding sets of addresses.

1. the full data set $\mathcal{F}$, comprising all Bitcoin transactions in our data set;
2. the data set $\mathcal{N}_C := \mathcal{F} \setminus \mathcal{T}_C$, obtained by removing all candidate dust-consuming transactions from $\mathcal{F}$;
3. the data set $\mathcal{N}_{2+} := \mathcal{F} \setminus \mathcal{T}_{2+}$, i.e., the one obtained by removing all type 2+ transactions from $\mathcal{F}$.

Informally, $\mathcal{N}_{2+}$ and $\mathcal{N}_C$ represent the set of transactions without all transactions caused by dust or dust attack candidate transactions respectively, while $\mathcal{F}$ contains all transactions, and so has the full effect of dust attack transactions. Do note that $\mathcal{T}_C \subseteq \mathcal{T}_{2+} \subseteq \mathcal{F}$, and so $\mathcal{N}_{2+} \subseteq \mathcal{N}_C \subseteq \mathcal{F}$ by construction. If we execute the clustering algorithm on this three sets of transactions we would obtain the set of clusters:

1. with the full effect of dust and dust attack transactions for $\mathcal{F}$;
2. without the effect of transactions suspected of being dust attacks for $\mathcal{N}_C$;
3. without the effect of all dust transactions for $\mathcal{N}_{2+}$.

As such, if the dust attack truly increases the effectiveness of the clustering deanonymization we would expect the clustering performed on $\mathcal{N}_C$, i.e., without dust attack candidates to be less precise then the one on $\mathcal{F}$, that includes dust transactions. Similarly, comparing the effectiveness of the clustering on $\mathcal{N}_C$ and $\mathcal{N}_{2+}$ should show how impactful the dust attack candidate transactions alone are compared to all dust transactions.

For each of the three sets of transactions we applied the efficient version of the clustering algorithm based only on the multi-input heuristic rules presented in Di Francesco Maesa et al. (2018). As discussed in the Clustering section, the inputs of this algorithm are the set of addresses $\mathcal{A} = \bigcup_{T \in \mathcal{F}} \text{inaddr}(T) \cup \text{outaddr}(T)$, namely the set of addresses induced by $\mathcal{F}$, and a set of transactions $X \subseteq \mathcal{F}$. The output is the partition $C(\mathcal{A}, X)$, namely the partition of $\mathcal{A}$ induced by the transactions in X. Note that each element of

Table 7 Number of transactions and clustering results statistics for the three examined sets of transactions $\mathcal{F}$, $\mathcal{N}_C = \mathcal{F} \setminus \mathcal{T}_C$, and $\mathcal{N}_{2+} = \mathcal{F} \setminus \mathcal{T}_{2+}$

	N. of TXs	N. of clusters	Cluster size			
			Min	Max	Avg	Std
$\mathcal{F}$	245,410,083	139,962,731	1	12,431,337	2.099	1,193.883
$\mathcal{N}_C$	245,320,981	140,391,544	1	12,431,337	2.093	1,187.956
$\mathcal{N}_{2+}$	245,293,933	140,623,307	1	12,431,337	2.089	1,186.860

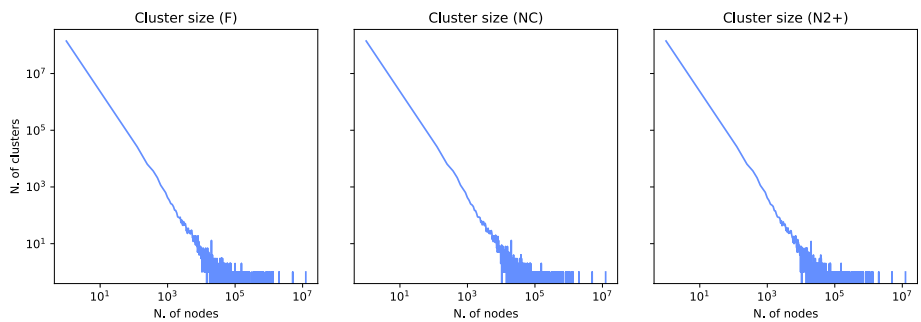


Fig. 7 Clusters size distributions for $P(\mathcal{F})$ (left), $P(\mathcal{N}_C)$ (middle), and $P(\mathcal{N}_{2+})$ (right)

$C(\mathcal{A}, X)$ is a cluster, namely a subset of addresses of $\mathcal{A}$. The algorithm iterates through the transactions in X and assigns all addresses appearing as inputs of the same transaction to the same cluster. Also, two clusters can be merged if they share at least one common address. We also recall that the complexity of this procedure is linear in the number of addresses and number of inputs of all transactions. Given a set of transactions $X \subseteq \mathcal{F}$, we define $P(X) := C(\mathcal{A}, X)$ as the partition obtained by applying the clustering algorithm to X while considering the global set of addresses $\mathcal{A}$ induced by all transactions in $\mathcal{F}$, not just those in X.

Statistics on the three considered sets and on their clustering results $P(\mathcal{F})$, $P(\mathcal{N}_C)$, and $P(\mathcal{N}_{2+})$ are reported in Table 7. Unsurprisingly, the less transactions we consider in each set, the more clusters there will be, and the smaller each cluster would be on average. This is to be expected, as the more transactions there are, the more the multi-input heuristic is likely to group clusters together. However, we can argue that the effect of dust transactions is more than ordinary, by having a disproportionate effect on the clustering. For example, if we compare the clustering results for $\mathcal{F}$ and $\mathcal{N}_C$, while the number of transactions only decreases by 0.036%, the number of clusters and their average size increases by 0.306% and 0.285% respectively. The minimum and maximum cluster sizes do not change however, meaning that some addresses remain isolated in a single cluster in both cases, and that the biggest cluster is formed independently from dust transactions. Moreover, from the table we can see how the extremely high standard deviation, compared to the mean value, as well as the huge difference between minimum and maximum size of the clusters, suggests a wide span of cluster sizes for all three sets of transactions considered. To better understand this phenomenon we plot the cluster sizes distribution in Fig. 7. However, beside a few outliers, the differences in the three plots are hardly noticeable. This is likely because the three sets considered have at least 99.95%

Table 8 Number of clusters and clusters sizes statistics for the four considered sets $diff(\mathcal{F}, \mathcal{N}_C)$, $diff(\mathcal{N}_C, \mathcal{F})$, $diff(\mathcal{F}, \mathcal{N}_{2+})$, and $diff(\mathcal{N}_{2+}, \mathcal{F})$, i.e., the sets obtained by removing the common clusters between the two partitions considered for each one

	No. of clusters	Cluster size			
		Min	Max	Avg	Std
$diff(\mathcal{F}, \mathcal{N}_C)$	35,094	2	5,089,846	362.539	27,672.847
$diff(\mathcal{N}_C, \mathcal{F})$	463,907	1	4,956,574	27.426	7,415.206
$diff(\mathcal{F}, \mathcal{N}_{2+})$	45,137	2	5,089,846	490.035	26,836.537
$diff(\mathcal{N}_{2+}, \mathcal{F})$	705,713	1	4,954,411	31.342	6,639.323

transactions in common. As such, it is not surprising that the clustering results show many similarities between the three sets.

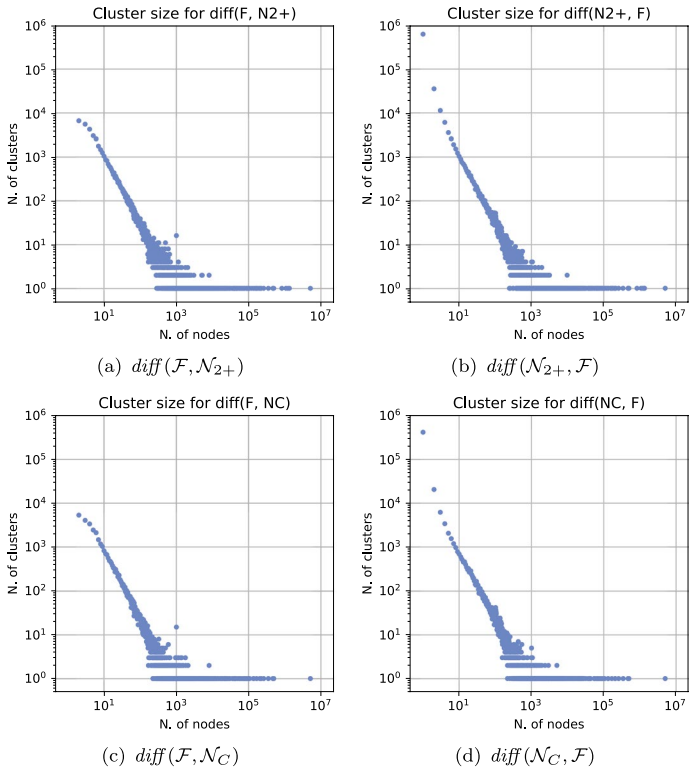
To better evaluate the impact of dust and dust attack candidate transactions, we have chosen to ignore the clusters that are equal between different transaction sets. The goal is to remove from the analysis the bulk of clusters that are not affected at all by the transactions we are interested in. This way we can fully concentrate on the sole contribution of those transactions.

To this aim we define the function $diff(X, Y) := P(X) \setminus (P(X) \cap P(Y))$, i.e., a function that given two sets returns the set containing all and only the elements of the first set that do not also belong to the second set. Do note that this is equivalent to the set difference, i.e., $diff(X, Y) = P(X) \setminus P(Y)$. We remind that the clustering results are partitions, and so the intersection of two partitions will give us the set of sets of addresses that are grouped identically in the two partitions. Given the sets $\mathcal{F}$, $\mathcal{N}_C$, and $\mathcal{N}_{2+}$ then:

- $diff(\mathcal{F}, \mathcal{N}_C)$ will be the set of all clusters of $P(\mathcal{F})$ that are not also contained in $P(\mathcal{N}_C)$;
- $diff(\mathcal{N}_C, \mathcal{F})$ will be the set of all clusters of $P(\mathcal{N}_C)$ that are not also contained in $P(\mathcal{F})$;
- $diff(\mathcal{F}, \mathcal{N}_{2+})$ will be the set of all clusters of $P(\mathcal{F})$ that are not also contained in $P(\mathcal{N}_{2+})$;
- $diff(\mathcal{N}_{2+}, \mathcal{F})$ will be the set of all clusters of $P(\mathcal{N}_{2+})$ that are not also contained in $P(\mathcal{F})$.

We report in Table 8 the size of the four sets, i.e., the number of clusters they contain, and some statistics about their cluster sizes. We remark how each of these sets is a set of clusters, i.e., a set whose elements (named clusters) are sets of addresses.

The first two lines of the table can be used to compare the effect of dust attack candidate transactions on the clustering. We see that removing the candidate transactions induces a clustering approximately 13 times "worse", as the number of clusters is 16 times higher in $diff(\mathcal{N}_C, \mathcal{F})$ compared to $diff(\mathcal{F}, \mathcal{N}_C)$ (that does include the candidate transactions). By looking at the cluster sizes, we see how the biggest cluster increases in size, but not much, while the average size of clusters does increase considerably, again, by a factor of approximately 13, while its standard deviation decreases. This means that not only the clusters are less, but also bigger. The same trend can be observed by looking



(a) $diff(\mathcal{F}, \mathcal{N}_{2+})$ (b) $diff(\mathcal{N}_{2+}, \mathcal{F})$

(c) $diff(\mathcal{F}, \mathcal{N}_{C})$ (d) $diff(\mathcal{N}_{C}, \mathcal{F})$

Fig. 8 Cluster sizes distributions for the four sets considered

at the results for $diff(\mathcal{F}, \mathcal{N}_{2+})$ and $diff(\mathcal{N}_{2+}, \mathcal{F})$, however the number of clusters and average size differs for a lower factor of approximately 16. We note how in both cases (considering $\mathcal{N}_C$ or $\mathcal{N}_{2+}$) the minimum size of the clusters is 2 instead of 1, which is unsurprising as we are considering transactions of type 2+ category in both cases and so the transactions will always have at least two inputs with different addresses that will then be placed in the same cluster by the multi-input heuristic.

To better inspect this difference in cluster sizes we show in Fig. 8 the cluster size distribution for each set. If we compare Fig. 8a with Fig. 8b and Fig. 8c with Fig. 8d we notice the same trend. For both cases, the plots show how the main effect caused by dust transactions on cluster sizes is a stark decrease in the number of clusters of smaller sizes (not just the clusters of size one that disappear completely). The overall number of addresses is the same in each of the two scenarios as well as the sum of all cluster sizes. This necessarily means that, even if less evident in the plot, there are more clusters containing more addresses, as attested by the difference in the mean size values shown in Table 8. If we consider only the clusters of size one that disappear entirely when considering dust transactions of both kinds, they are 414,236 for $\mathcal{N}_C$ and 623,614 for $\mathcal{N}_{2+}$ respectively. These are especially important as they represent addresses that would not have been grouped with any other if it were not for the dust transactions.

Overall this study highlights the relevant impact of dust attacks on the clustering effectiveness. If it where not for the transactions of category 2+, there would exists 623,614 more clusters with a single address, i.e., addresses which the clustering fails

to deanonymize. Similarly, if we consider the dust attack candidate transactions alone (which are a subset of all transactions of type 2+), then 414,236 addresses would have not been clustered at all if the attacks had not taken place. We observe that 414,236 is lower than 623,614, but we have to remember that dust attack candidate transactions are caused by the 18,840 suspect transactions of the set $\mathcal{T}_{sus}$, which is much smaller than the set of all dust creating transactions. This means that the dust attack transactions, despite being only 4.86% of all dust creating transactions, allow to cluster 66.43% of all dust induced clustered addresses. Considering the whole data set, the transactions suspected of being part of dust attacks are only 0.008% of all transactions but allow to cluster 0.14% of all addresses that would have otherwise remained isolated.

Conclusions and future work

In this paper we have presented an analysis of dust transactions in the Bitcoin blockchain, focusing on unconventional behavior of users and deanonymization attacks based on dust. After formally defining the concepts of dust and dust attack, we examined the creation and consumption of dust outputs. We then highlighted the major creators and consumers of dust and traced the evolution of consumption patterns over time. Moreover, in the second part of our study we identified transactions likely to be associated with dust attacks, from the sides of both attackers and victims. Finally, we conducted an analysis of such transactions and used heuristic clustering techniques to assess the impact of suspected dust attack transactions on the deanonymization of Bitcoin addresses.

Our results show that the majority of dust outputs can be traced back to Satoshi Dice, a popular on-chain betting service launched in 2012. Among the remaining top creators we can find other gambling services (e.g., BtcDice, DiceOnCrack, and BetCoins) and Silk Road, an online illegal marketplace on the dark web. On the other hand, for what concerns the top consumers, we were able to identify a compromised address, i.e., one whose private key has been leaked to the public. Moreover, the temporal analysis of dust consumption allowed us to establish that non-dust outputs typically get spent 8 times faster than dust outputs, probably due to the fact that dust outputs cannot be spent on their own and thus need to be combined with other amounts of value.

The experimental results have also shown the impact of dust transactions, especially the ones suspected of being part of dust attacks, on Bitcoin addresses pseudonymity. For example, we have measured how the relatively few transactions likely related to dust attacks disproportionately increase the effectiveness of addresses deanonymization. These transactions are just 4.86% of all dust creating transactions and 0.008% of all transactions overall, but allow to cluster 66.43% of all addresses affected by dust transactions and 0.14% of all addresses. These results prove how dust negatively impacts on users' pseudonymity in Bitcoin in general, and especially so if intentionally used during dust attacks.

Regarding future work, we plan to extend our analysis by examining a more recent version of the Bitcoin transaction data set. Specifically, our intention is to include also transactions issued after the introduction of the Segregated Witness update, which took place in August 2017. Indeed, with the introduction of Segregated Witness, the threshold to

consider amounts as dust has changed, thus potentially altering the quantity of dust outputs that are created and consumed. In addition to this, we plan to enrich our clustering analysis and further evaluate the impact of candidate and type 2+ transactions by comparing their effect on address clustering with respect to randomly chosen subsets of transactions.

Appendix: symbols and abbreviations

This appendix contains a general scheme of all symbols and abbreviations results used in the Experimental results section to indicate data sets of transactions. All used symbols are listed in Table 9, each accompanied by a brief description of its meaning.

Table 9 List of symbols used throughout the Experimental results section

Symbol	Description
$\mathcal{F}$	Set of all Bitcoin transactions from January 3rd, 2009 to August 10th, 2017
$\mathcal{T}$	Transactions in $\mathcal{F}$ that create or consume dust
D_{out}	Set of all dust outputs in $\mathcal{T}$
D_{in}	Set of all dust inputs in $\mathcal{T}$
U	Unspent outputs of D_{out}
NOD	Outputs of D_{out} spent in combination with at least one non-dust output
OD	Outputs of D_{out} spent exclusively with other dust outputs
$\mathcal{T}'$	Transactions in $\mathcal{T}$ possibly related to dust attacks
$k(T)$	Absolute number of dust outputs of transaction T
$j(T)$	Relative number of dust outputs of transaction T
$\mathcal{T}_{sus}$	Suspect dust-creating transactions in $\mathcal{T}'$, issued by potential attackers
$\mathcal{T}_1$	Dust-consuming transactions in $\mathcal{T}'$ with one unique input address
$\mathcal{T}_{2+}$	Dust-consuming transactions in $\mathcal{T}'$ with at least two unique input addresses
$\mathcal{T}_S$	Dust-consuming transactions in $\mathcal{T}'$ where all input value is spent in fees
$\mathcal{T}_C$	Dust-consuming transactions in $\mathcal{T}'$ issued by potential victims of dust attacks
$\mathcal{N}_C$	$\mathcal{F} \setminus \mathcal{T}_C$, i.e., set obtained by removing all candidate dust-consuming transactions from $\mathcal{F}$
$\mathcal{N}_{2+}$	$\mathcal{F} \setminus \mathcal{T}_{2+}$, i.e., set obtained by removing all type 2+ transactions from $\mathcal{F}$
$\mathcal{A}$	Set of Bitcoin addresses induced by $\mathcal{F}$
$P(X)$	Partition of $\mathcal{A}$ obtained by applying the clustering algorithm to X
$diff(X, Y)$	$P(X) \setminus P(Y)$

Acknowledgements
Not applicable.

Author Contributions
All authors have contributed equally to the paper. All authors have read and approved the final manuscript.

Funding
Not applicable.

Availability of data and materials
All code used for the experiments is publicly available at the following GitHub repository: https://github.com/mloporchio/DustAnalysis.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 28 February 2023 Accepted: 28 May 2023
Published online: 15 June 2023

References

Baqer K, Huang DY, McCoy D, Weaver N (2016) Stressing out: Bitcoin stress testing. In: International conference on financial cryptography and data security, pp 3–18. Springer
Bartoletti M, Pompianu L (2017) An analysis of Bitcoin OP_RETURN metadata. In: Financial cryptography and data security, pp 218–230. Springer International Publishing
Biryukov A, Tikhomirov S (2019) Deanonymization and linkability of cryptocurrency transactions based on network analysis. In: IEEE European symposium on security and privacy, EuroS &P 2019, Stockholm, Sweden, June 17–19, 2019, pp 172–184. IEEE
Bitcoin Wiki: Privacy–Forced address reuse. https://en.bitcoin.it/wiki/Privacy#Forced_address_reuse Accessed 15 Feb 2023
Bradbury D (2013) The problem with Bitcoin. Comput Fraud Secur 2013(11), 5–8
Christin N (2013) Traveling the silk road: a measurement analysis of a large anonymous online marketplace. In: Proceedings of the 22nd international conference on World Wide Web, pp 213–224
Crowcroft J, Di Francesco Maesa D, Magrini A, Marino A, Ricci L (2021) Leveraging the users graph and trustful transactions for the analysis of bitcoin price. IEEE Trans Netw Sci Eng 8(2):1338–1352
Decker C, Wattenhofer R (2014) Bitcoin transaction malleability and mtgox. In: Computer Security-ESORICS 2014: 19th European symposium on research in computer security, Wroclaw, Poland, September 7–11, 2014. Proceedings, Part II 19, pp 313–326. Springer
Di Francesco Maesa D, Marino A, Ricci L (2018) Data-driven analysis of bitcoin properties: exploiting the users graph. Int J Data Sci Anal 6(1):63–80
Fanti G, Viswanath P (2017) Deanonymization in the bitcoin P2P network. In: Advances in neural information processing systems 30: annual conference on neural information processing systems 2017, December 4–9, 2017, Long Beach, CA, USA, pp 1364–1373
Di Francesco Maesa D, Mori P (2020) Blockchain 3.0 applications survey. J Parallel Distributed Comput 138:99–114
Gomez G, Moreno-Sanchez P, Caballero J (2022) Watch your back: identifying cybercrime financial relationships in bitcoin through back-and-forth exploration. In: Proceedings of the 2022 ACM SIGSAC conference on computer and communications security, pp 1291–1305
Harrigan M, Fretter C (2016) The unreasonable effectiveness of address clustering. In: 2016 Intl IEEE conferences on ubiquitous intelligence and computing, advanced and trusted computing, scalable computing and communications, cloud and big data computing, internet of people, and smart world congress (uic/atc/scalcom/cbdcom/iop/smartworld), pp 368–373 . IEEE
Hughes J Wizstats-Bitcoin pool Web statistics. https://github.com/wizkid057/wizstats/blob/fb200de98c2889a558cbf109fd3da4e916f6dc5a/config.php.example Accessed 15 Feb 2023
Johnson D, Menezes A, Vanstone SA (2001) The elliptic curve digital signature algorithm (ECDSA). Int J Inf Sec 1(1):36–63
Jourdan M, Blandin S, Wynter L, Deshpande P (2018) Characterizing entities in the bitcoin blockchain. In: 2018 IEEE international conference on data mining workshop (ICDMW), IEEE
Jourdan M, Blandin S, Wynter L, Deshpande P (2019) A probabilistic model of the bitcoin blockchain. In: Computer vision and pattern recognition workshop (CVPRW), 2019, IEEE
Loporchio M, Bernasconi A, Di Francesco Maesa D, Ricci L (2023) An analysis of bitcoin dust through authenticated queries. In: Complex networks and their applications XI, Cham, pp. 495–508. Springer International Publishing
Maesa DDF, Marino A, Ricci L (2017) Detecting artificial behaviours in the bitcoin users graph. Online Soc Netw Media 3, 63–74
Meiklejohn S, Pomarole M, Jordan G, Levchenko K, McCoy D, Voelker GM, Savage S (2013) A fistful of bitcoins: characterizing payments among men with no names. In: Proceedings of the 2013 conference on internet measurement conference, pp 127–140
Merkle RC (1980) Protocols for public key cryptosystems. In: Proceedings of the 1980 IEEE symposium on security and privacy, Oakland, California, USA, April 14–16, 1980, pp 122–134. IEEE Computer Society
Nakamoto S (2008) Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf Accessed 15 Feb 2023
Pérez-Solà C, Delgado-Segura S, Navarro-Arribas G, Herrera-Joancomartí J (2019) Another coin bites the dust: an analysis of dust in utxo-based cryptocurrencies. R Soc Open Sci 6(1), 180817
Reid F, Harrigan M (2013) An analysis of anonymity in the bitcoin system. Springer: New York, pp 197–223
Saad M, Njilla L, Kamhoua C, Kim J, Nyang D, Mohaisen A (2019) Mempool optimization for defending against ddos attacks in pow-based blockchain systems. In: 2019 IEEE international conference on blockchain and cryptocurrency (ICBC), pp 285–292. IEEE
Satoshi Dice: Bitcoin Casino Game. https://web.archive.org/web/20120501094159/http://satoshidice.com Accessed 15 Feb 2023
Singh A, Parizi RM, Han M, Dehghantanha A, Karimipour H, Choo K-KR (2020) Public blockchains scalability: an examination of sharding and segregated witness. In: Blockchain cybersecurity, trust and privacy, pp 203–232. Springer
Tamassia R (2003) Authenticated data structures. In: Algorithms-ESA 2003: 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003. Proceedings 11, pp. 2–5. Springer
Todd P Dust-B-Gone. https://github.com/petertodd/dust-b-gone Accessed 15 Feb 2023
Twitter: Samourai Wallet on Twitter. https://twitter.com/SamouraiWallet/status/1055345822076936192 Accessed 2023-02-15

The Bitcoin Core developers: Bitcoin dust limit. https://github.com/bitcoin/bitcoin/blob/c536dfbcb00fb15963bf5d507b7017c241718bf6/src/policy/policy.cpp Accessed 15 Feb 2023
Wang Y, Yang J, Li T, Zhu F, Zhou X (2018) Anti-dust: a method for identifying and preventing blockchain's dust attacks. In: 2018 international conference on information systems and computer aided education (ICISCAE), pp 274–280. IEEE
Zhang Y, Wang J, Luo J (2020) Heuristic-based address clustering in bitcoin. IEEE Access 8, 210582–210591

Publisher's Note