

An analysis of Bitcoin OP_RETURN metadata

Massimo Bartoletti and Livio Pompianu

Università degli Studi di Cagliari, Cagliari, Italy
{bart,livio.pompianu}@unica.it

Abstract. The Bitcoin protocol allows to save arbitrary data on the blockchain through a special instruction of the scripting language, called OP_RETURN. A growing number of protocols exploit this feature to extend the range of applications of the Bitcoin blockchain beyond transfer of currency. A point of debate in the Bitcoin community is whether loading data through OP_RETURN can negatively affect the performance of the Bitcoin network with respect to its primary goal. This paper is an empirical study of the usage of OP_RETURN over the years. We identify several protocols based on OP_RETURN, which we classify by their application domain. We measure the evolution in time of the usage of each protocol, the distribution of OP_RETURN transactions by application domain, and their space consumption.

1 Introduction

Bitcoin was the first decentralized digital currency to be created, and now it is the most widely used, with a market capitalization of ~ 20 billions USD¹. Technically, the Bitcoin network is a peer to peer system, where users can securely transfer currency without the intermediation of a trusted authority. Transactions of currency are gathered in blocks, that are added to a public data structure called *blockchain*. The consensus algorithm of Bitcoin guarantees that, for an attacker to be able to alter an existing block, she must control the majority of the computational resources of the network [37]. Hence, attacks aiming at incrementing one's balance, e.g. by deleting transactions that certify payments to other users, are infeasible in practice. This security property is often rephrased by saying that the blockchain can be seen as an *immutable* data structure.

Although the main goal of Bitcoin is to transfer digital currency, the immutability and openness of its blockchain have inspired the development of new protocols, which “piggy-back” metadata on transactions in order to implement a variety of applications beyond cryptocurrency. For instance, some protocols allow to certify the existence of a document (e.g., [21, 29, 33]), while some others allow to track the ownership of a digital or a physical asset (e.g., [16, 24, 25]). Many of these protocols save metadata on the blockchain by using an instruction called OP_RETURN, which is part of the Bitcoin scripting language.

A debate about the scalability of Bitcoin has been taking place over the last few years [2, 30, 31]. In particular, users argue over whether the blockchain should

¹ Source: coinmarketcap.com, accessed on February 28th, 2017.

allow for storing spurious data, not inherent to currency transfers. Although many recent works analyse the Bitcoin blockchain [35, 38, 40, 41], as well as some services related to OP_RETURN [6, 22, 26, 32], many relevant questions are still open. What is the impact of the data attached to OP_RETURN on the size of the blockchain? Which kinds of blockchain-based applications are exploiting the OP_RETURN instruction, and how?

Contributions. We analyse the usage of OP_RETURN throughout the Bitcoin blockchain, collecting a total of 1,887,708 OP_RETURN transactions. We investigate to which protocols OP_RETURN transactions belong, identifying 22 distinct protocols (associated to 51% of these transactions). We find that 15% of this total are *empty* transactions, which attach no metadata to OP_RETURN. By studying the usage of OP_RETURN over time, we identify several transaction peaks related to empty transactions, and we show that they are mainly caused by stress tests and spam attacks happened in summer 2015. We classify protocols according to their application domain, and we study the numerical proportion of these applications. Finally, we measure the size of OP_RETURN metadata, and the proportion between the size of OP_RETURN transactions and the overall size of the transactions in the blockchain. To the best of our knowledge, ours is the widest investigation about the usage of OP_RETURN. All our analyses are supported by a tool we have developed. The sources of our tool, as well as the experimental data, are available at [5].

2 Background on Bitcoin

Bitcoin [39] is a decentralized infrastructure to exchange virtual currency — the *bitcoins*. The transfers of currency, called *transactions*, are the basic elements of the system. The transactions are recorded on a public, append-only data structure, called *blockchain*. To illustrate how Bitcoin works, we consider two transactions T_0 and T_1 of the following form:

T_0
in: \dots
in-script: \dots
out-script(T, σ): $ver_k(T, \sigma)$
value: v_0

T_1
in: T_0
in-script: $sig_k(\bullet)$
out-script(\dots): \dots
value: v_1

The transaction T_0 contains a value v_0 bitcoins. Anyone can *redeem* the amount of bitcoins in T_0 by putting on the blockchain a transaction (e.g., T_1), whose in field contains the identifier of T_0 (the hash of the whole transaction, displayed as T_0 in the figure) and whose in-script contains values making the out-script² of T_0 , a programmable boolean function, evaluate to true. When this happens, the value of T_0 is transferred to the new transaction T_1 , and T_0 becomes unredeemable. A subsequent transaction can then redeem T_1 likewise.

² in-script/out-script are called scriptPubKey/scriptSig in the Bitcoin wiki.

T
in[0]: $T_0[n_0]$
in-script[0]: ...
⋮
out-script[0](T_0, w_0): ...
value[0]: v_0
⋮
lockTime: s

T
in[0]: ...
in-script[0]: ...
⋮
out-script[0](...): OP_RETURN "EWHello!"
value[0]: 0
⋮

(a) General form of transactions. (b) An OP_RETURN transaction.

In the transaction T_0 above, the out-script just checks the digital signature σ on the redeeming transaction T w.r.t. a given key k . We denote with $ver_k(T, \sigma)$ the signature verification, and with $sig_k(\bullet)$ the signature of the enclosing transaction (T_1 in our example), including all the parts of the transaction but its in-script (obviously, because it contains the signature itself).

Now, assume that T_0 is redeemable on the blockchain when someone tries to append T_1 . The Bitcoin network accepts the redeem if (i) $v_1 \leq v_0$, and (ii) the out-script of T_0 , applied to T_1 and to the signature $sig_k(\bullet)$, evaluates true.

The previous example is a special case of a Bitcoin transaction: the general form is displayed in Figure 1a. First, there can be multiple inputs and outputs (denoted with array notation in the figure), and each output has its own out-script and value. Since each output can be redeemed independently, in fields must specify which one they are redeeming ($T_0[n_0]$ in the figure). A transaction with multiple inputs redeems *all* the (outputs of) transactions in its in fields, providing a suitable in-script for each of them. To be valid, the sum of the values of all the inputs must be greater or equal to the sum of the values of all outputs. The *Unspent Transaction Output* (in short, UTXO) is the set of redeemable outputs of all transactions included in the blockchain. To be valid, a transaction must only use elements of the UTXO as inputs.

In its general form, the out-script is a program in a non Turing-complete scripting language, which features a limited set of logic, arithmetic, and cryptographic operators. The lockTime field specifies the earliest moment in time when the transaction can appear on the blockchain.

Writing metadata in transactions. Bitcoin transactions do not provide a field where one can save arbitrary data. Nevertheless, users have devised various creative ways to encode data in transactions. A first method is to abuse the standard *Pay-to-PubkeyHash* script³, which implements the signature verification ver_k seen before (actually, the script does not contain the public key k , but its hash $h = H(k)$). To make the script evaluate to true, the redeeming transaction T has to provide the signature σ and a public key k such that $H(k) = h$ and $ver_k(T, \sigma)$. One can store an arbitrary message m (a few bytes long) within the out-script, by writing m in place of the hash h . Since computing a value k such

³ en.bitcoin.it/wiki/Transaction#Pay-to-PubkeyHash

that $H(k) = m$ (i.e., a preimage of m) and a signature σ such that $ver_k(T, \sigma)$ are computationally hard operations, outputs crafted in this way are unspendable in practice. However, these outputs are not easily distinguishable from the spendable ones, hence the nodes of the Bitcoin network must keep them in their UTXO set [3]. Since this set is usually stored in RAM for efficiency concerns [28], this practice negatively affects the memory consumption of nodes [35].

The OP_RETURN instruction allows to save metadata on the blockchain, as shown in Figure 1b⁴. However, unlike *Pay-to-PubkeyHash*, an out-script containing OP_RETURN always evaluates to false, hence the output is provably unspendable, and its transaction can be safely removed from the UTXO. In this way, OP_RETURN overcomes the UTXO consumption issue highlighted above. Although the OP_RETURN instruction has been part of the scripting language since the first releases of Bitcoin, originally it was considered *non-standard* by nodes, so transactions containing OP_RETURN were difficult to reliably get mined. In March 2014 [12], OP_RETURN became standard, meaning that all nodes started to relay unconfirmed OP_RETURN transactions⁵. The limit for storing data in an OP_RETURN was originally planned to be 80 bytes, but the first official client supporting the instruction, i.e. the release 0.9.0 [12], allowed only 40 bytes. This animated a long debate [7,8,17,18]. From the release 0.10.0 [9] nodes could choose whether to accept or not OP_RETURN transactions, and set a maximum for their size. The release 0.11.0 [10] extended the data limit to 80 bytes, and the release 0.12.0 [11] to a maximum of 83 bytes.

3 Methodology for classifying OP_RETURN transactions

We discuss our methodology for identifying protocols that use OP_RETURN.

We gather all the OP_RETURN transactions from the origin block up to the block number 453,200 (added on 2017/02/15). We end up with a set of 1,887,708 OP_RETURN transactions. For each of them, we save the following data in a database: (i) the hash of the transaction; (ii) the hash of the enclosing block; (iii) the timestamp of the block; (iv) the metadata attached to the OP_RETURN.

Next, we detect to which protocols the OP_RETURN transactions belong. Usually, a protocol is identified by the first few bytes of metadata attached to the OP_RETURN, but the exact number of bytes may vary from protocol to protocol. Hence, we associate OP_RETURN transactions to protocols as follows:

1. we search the web for known associations between identifiers and protocols;
2. we accordingly classify the OP_RETURN transactions that begin with one of the identifiers obtained at step 1;
3. on the remaining *unknown transactions*, we perform a frequency analysis of the first few bytes of metadata, to discover new protocol identifiers.

⁴ Hash: [d84f8cf06829c7202038731e5444411adc63a6d4cbf8d4361b86698abad3a68a](#)

⁵ Regarding the use of OP_RETURN, the release notes of Bitcoin Core version 0.9.0 state that: “*This change is not an endorsement of storing data in the blockchain.*” At the same time, some Bitcoin explorers, (e.g. [blockchain.info](#), [blockexplorer.com](#), [smartbit.com](#)) allow to inspect data encoded in OP_RETURN scripts.

Algorithm 1 Detect protocol identifiers

```

unknownTx  $\leftarrow$  set of all unknown transactions
Codes  $\leftarrow$   $\emptyset$ 
for  $i \leftarrow 1$  to  $D$  do
  H  $\leftarrow$  new hash table from protocol identifiers to number of occurrences
  for all tx  $\in$  unknownTx do
    code  $\leftarrow$  tx.substring( $i$ )  $\triangleright$  first  $i$  characters of tx
    if (H.contains(code)) then
      H.code  $\leftarrow$  H(code)+1 else H.code  $\leftarrow$  1
    end if
  end for
  expectedOccurrences  $\leftarrow$  unknownTx.size() / pow(16, $i$ )
  for all h  $\in$  H do
    if (h.occurrences > expectedOccurrences *  $\delta$  and h.occurrences >  $N$ ) then
      Codes  $\leftarrow$  Codes  $\cup$  {h.code}
    end if
  end for
end for
return Codes

```

In more details, in the first step we query Google to obtain public identifier/protocol bindings. For instance, the query “*Bitcoin OP_RETURN*”, returns $\sim 26,500$ results, and we manually inspect the first few pages of them. Note that a protocol can be associated with more than one identifier (e.g., Stampery, Blockstore [34], Remembr, CryptoCopyright), or even do not have any identifier. In this way we obtain 22 protocols associated to 33 identifiers; further, we find 3 protocols that do not use any identifier (Counterparty, Diploma [19], Chainpoint [14]).

The second step is performed by our tool: it associates 970,374 transactions to a protocol ($\sim 51\%$ of the total OP_RETURN transactions). The other transactions are classified either as *empty* or *unknown*. Empty transactions have no data attached to the OP_RETURN instruction (296,491 transactions, $\sim 15\%$ of the total); unknown transactions have no known identifier (620,843 transactions, $\sim 32\%$ of the total).

The final step analyses unknown transactions, attempting to discover new protocol identifiers. Since identifiers may have different lengths, we gather the first D bytes of unknown transactions, for D ranging from 1 to 12, and we perform a frequency analysis of these strings. This analysis does not reveal relevant statistical anomalies (roughly, the strings are uniformly distributed), hence this step does not yield any new identifier. Algorithm 1 details this search, which is executed with the following parameters: $D = 12$, $\delta = 2$, $N = 100$.

4 Qualitative analysis of OP_RETURN transactions

We now classify the protocols obtained in Section 3, associating each protocol to a *category* that describes its intended application domain. To this purpose, we manually inspect the web pages of each protocol.

Assets gathers protocols that exploit the immutability of the blockchain to certify ownership, exchange, and eventually the value of real-world assets. Metadata in transactions are used to specify e.g. the value of the asset, the amount of the asset transferred, the new owner, *etc.*

Document notary includes protocols for certifying the ownership and timestamp of a document. A user can publish the hash of a document in a transaction, and in this way he can prove its existence and integrity. Similarly, signatures can be used to certify ownership.

Digital arts includes protocols for declaring access right and copy rights on digital arts files, like e.g. photos or music.

Other includes protocols whose goals differ from the ones above. For instance, *Eternity Wall* [20] allows users to store short text messages on the blockchain; *Blockstore* [13] is a generic key-value store, on top of which more complex protocols can be implemented⁶.

Empty includes protocols that do not attach any data to OP_RETURN.

Unknown includes protocols for which we have not been able to detect an identifier (possibly, because they do not use any).

We report our classification of protocols in the first two columns of Table 1. Due to the OP_RETURN space limit, long pieces of metadata require to be split in many transactions, and higher fees. Hence, *assets* protocols usually feature complex rules, have space-efficient representations of data, and often propose off-chain solutions [15]. We distinguish document notary protocols from digital arts protocols for the following reason. Most document notary applications do not require users to provide their documents to the application, and the main purpose of the protocol (certifying ownership) can be fulfilled also when the application is no longer live. Instead, digital arts application usually need to gather user documents, and require interactions with users, e.g. they often play the role of broker between producers and consumers.

5 Quantitative analysis of OP_RETURN transactions

Table 1 shows some statistics about OP_RETURN transactions. The first column indicates the protocol categories, introduced in Section 4. The second and third columns show, respectively, the protocol names and the associated identifiers. The fourth column shows the date in which the protocol generated the first transaction. Since transactions do not have a “date” field, we infer dates from the timestamp of the block containing the transaction. The next two columns count the total number of transactions, and the total size (in bytes) of the OP_RETURN data contained therein. To compute the size we only consider the metadata, i.e. we do not count neither the OP_RETURN instruction nor the other fields of the transaction. The last column shows the average size of the transaction metadata.

Category	Protocol	Identifiers	First trans.	Tot. trans.	Tot. Size	Avg. Size
Assets	Colu	CC	2015/07/09	237,479	4,290,388	18.0
	CoinSpark	SPK	2014/07/02	28,026	956,904	34.1
	OpenAssets	OA	2014/05/03	133,570	1,728,350	12.9
	Omni	omni	2015/08/10	105,979	2,132,565	20.1
	Counterparty	N/A	N/A	N/A	N/A	N/A
	Total	-	-	505,054	9,108,207	18.0
Document Notary	Factom	Factom!!, FACTOM00, Fa, FA	2014/04/11	74,159	2,966,234	40.0
	Stampery	S1, S2, S3, S4, S5	2015/03/09	74,249	2,627,540	35.4
	Proof of Existence	DOCPROOF	2014/04/21	5,262	210,433	40.0
	Blocksign	BS	2014/08/04	1,460	55,192	37.8
	CryptoCopyright	CryptoTests-, CryptoProof-	2014/08/02	46	1,840	40
	Stampd	STAMPD##	2015/01/03	473	18,867	39.9
	BitProof	BITPROOF	2015/02/25	758	30,320	40
	ProveBit	ProveBit	2015/04/05	57	2,280	40
	Remembr	RMBd, RMBe	2015/08/25	28	1,128	40.3
	OriginalMy	ORIGMY	2015/07/12	126	4,788	38
	LaPreuve	LaPreuve	2014/12/07	67	2,623	39.1
	Nicosia	UNicDC	2014/09/12	20	684	34.2
	Chainpoint	N/A	N/A	N/A	N/A	N/A
	Diploma	N/A	N/A	N/A	N/A	N/A
	Total	-	-	156,705	5,921,929	37.8
Digital Arts	Monegraph	MG	2015/06/28	63,278	2,317,151	36.6
	Blockai	0x1f00	2015/01/09	527	34,225	64.9
	Ascribe	ASCRIIBE	2014/12/19	40,859	847,641	20.7
	Total	-	-	104,664	3,199,017	30.6
Other	Eternity Wall	EW	2015/06/24	3,715	160,191	43.1
	Blockstore	id, 0x5888, 0x5808	2014/12/10	191,907	5,494,174	28.6
	SmartBit	SB.D	2015/11/24	8,329	299,844	36
	Total	-	-	203,951	5,954,209	29.2
Empty	Total	-	2014/03/20	296,491	0	0
Unknown	Total	-	2014/03/12	620,843	20,023,345	32.3
TOTAL	-	-	2014/03/12	1,887,708	44,206,707	23.4

Table 1: Statistics about OP_RETURN protocols.

5.1 Overall statistics

We detect 1,887,708 OP_RETURN transactions, distributed into 98,233 blocks, by scanning the blockchain until block number 453,200. Overall, OP_RETURN transactions constitute $\sim 0.96\%$ of the total transactions in the blockchain, and $\sim 1.16\%$ of the portion of the blockchain from 2014/03/12 (when the first OP_RETURN transaction appeared). Although the former measurement considers 7 years of transactions while the latter only considers the last 3 years, we note that the values are very close. We explain this fact by observing that the daily number of transactions rapidly increased since July 2014.

5.2 Transaction peaks

Figures 2a and 2b display the number of OP_RETURN transactions per week, from 2014/03 (date of the first OP_RETURN transaction) to 2017/02 (end of our extraction). In the graph we note several peaks, that we explain as follows:

⁶ Hereafter we aggregate all the protocols built upon *Blockstore*, by identifying them with *Blockstore* itself.

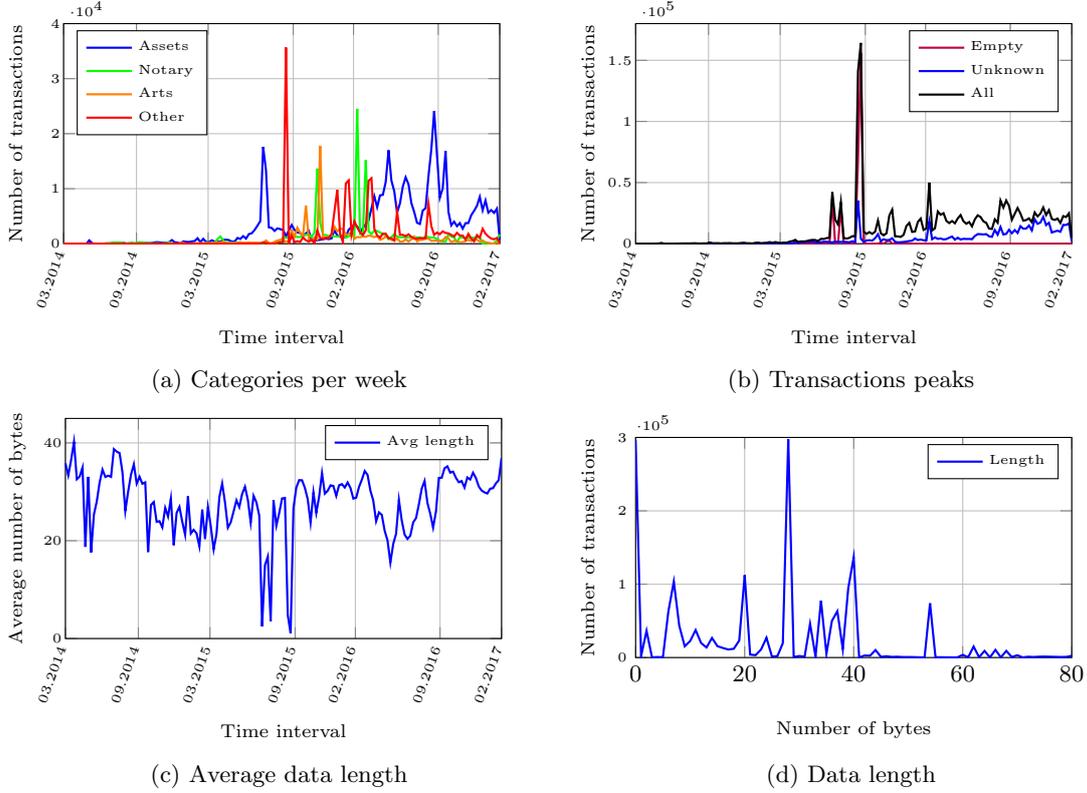


Fig. 2: Usage and size of OP_RETURN transactions.

1. $\sim 100,000$ transactions from 2015/07/08 to 2015/08/05. This peak is mainly composed of two different peaks of *empty* transactions: the july peak ($\sim 36,900$ transactions from 2015/07/08 to 2015/07/10) and the august peak ($\sim 29,200$ transactions from 2015-08-01 to 2015-08-03). Both peaks seem to be caused by a spam campaign that resulted in a DoS attack on Bitcoin which happened in the same period, as reported in [35].
2. $\sim 300,000$ transactions from 2015/09/09 to 2015/09/23. This second peak is the highest and longest-lasting one. As before, it is mainly caused by *empty* transactions ($\sim 223,000$), although here we also observe a component of *unknown* and *blockstore* transactions ($\sim 35,000$ each). The work [35] detects a spike also in this period, precisely around 2015/09/13, where an anonymous group performed a stress-test on the network with a *money drop*. This involves a public release of private keys, with the aim to cause a big race which would cause a large number of *double-spend* transactions.
3. $\sim 50,000$ transactions from 2016/03/02 to 2016/03/09. The last peak is due to the sum of two different peaks: *unknown* (about 18,000) and *stampery*

(about 23,000) transactions. We conjecture that this peak is caused by the testing and bootstrap of protocols.

We observe that the Bitcoin blockchain has also other peaks, not related to OP_RETURN transactions. For instance, starting from the 2015/05/22 and for a duration of 100 blocks, the Bitcoin network was targeted by a stress test [4], during which the network was flooded with a huge number of transactions. Actually, the usage of OP_RETURN transactions in the period of this peak does not seem to diverge from their normal usage.

5.3 Space consumption

A debated topic in the Bitcoin community is whether it is acceptable or not to save arbitrary data in the blockchain. The sixth column in Table 1 shows, for each protocol, the total size of metadata (i.e., not considering the bytes of the instructions OP_RETURN and PUSH_DATA). The last row of Table 1 shows that the total size of metadata is ~ 42 MB (in the same date, the size of the whole blockchain is ~ 102 GB). Figure 2c shows the average length of the data for each week.

Generally, the average length of metadata is less than 40 bytes, despite the extension to 80 bytes introduced on 2015/07/12. Peaks down on the same period are related to the *empty* transactions discussed in Section 5.2. Figure 2d represents the number of transactions with a given data length: also this chart confirms a small number of transactions that use more than the half of the available space. Note that the discussed peak appears also in this chart, in correspondence of the 0 value. From the last column of Table 1 we see that only the size of *Blockai* metadata is close to 80 bytes. Several *document notary* protocols take 40 bytes on average: this depends from their identifiers, composed of 16 bytes, and from the size of the hash they save. Generally, *document notary* protocols carry longer data than the other protocols.

We now evaluate the minimum space consumption of the OP_RETURN transactions on the whole blockchain. First, we observe that an *empty* transaction with one input and one output has a total size of 156 bytes. From Table 1 we see that OP_RETURN transactions carry ~ 23.4 bytes of metadata, on average. Hence, we approximate the average size of OP_RETURN transaction as ~ 179.4 bytes, and so an approximation of the space consumption of all the OP_RETURN transactions is ~ 323 MB.

Finally, we estimate the ratio between the total size of OP_RETURN transactions and the size of all the transactions on the blockchain. The block header has size 97 bytes at most. Hence, removing the size of the headers of our 453,200 extracted blocks (~ 42 MB) from the total size of the blockchain at 2017/02/15, we obtain ~ 102 GB of transactions. From this we conclude that OP_RETURN transactions consume $\sim 0.3\%$ of the total space on the blockchain.

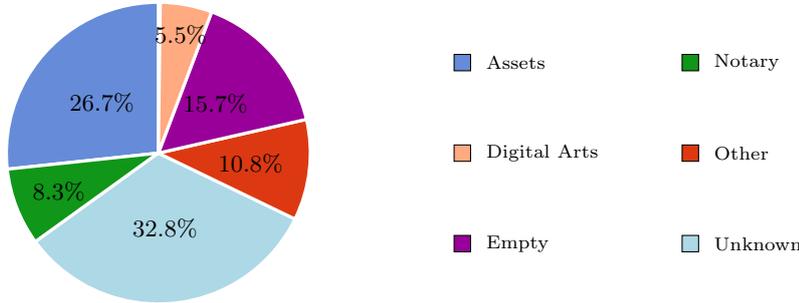


Fig. 3: Distribution of transactions by category.

5.4 Distribution of protocols by category

Figure 3 displays how the OP_RETURN transactions are distributed in the categories identified in Section 4. We note a relevant component of *empty* and *unknown* transactions. Although *assets* protocols produce the highest number of transactions, the most numerous category is *document notary*.

Figure 2a and the fourth column of Table 1 suggest that, originally, the protocols using OP_RETURN were in the categories *assets* and *notary*, while the other use cases were introduced subsequently (indeed, the *other* category was not inhabited before the end of 2014).

Empty transactions use OP_RETURN without any data attached, so they are not associated to any protocol. We evaluate that $\sim 96\%$ of these transactions are related to the transaction peaks discussed in Section 5.2. Since those peaks happened in the same period of the stress tests and spam campaign discussed in [35], we conjecture that *empty* transactions are related to those events⁷.

The *unknown* category contains $\sim 33\%$ of the OP_RETURN transactions. We identify 3 protocols [14,19,36] that write OP_RETURN data only as *unknown* transactions. We also identify one protocol [23] that besides using an identifier for saving document hashes, allows to save text messages without any identifier.

6 Conclusions

Our analysis shows an increasing interest in the OP_RETURN instruction. While in the first year of existence of OP_RETURN transactions (from March 2014) only a few hundreds of these transactions were appended per week, their usage has been steadily increasing since March 2015. In the last weeks of our experiments (February 2017) we counted $\sim 25,000$ new OP_RETURN transactions per

⁷ To verify this conjecture we would need to compare the transaction identifiers of our *empty* transactions with the identifiers of [35], which are not available online.

week, on average. Overall, we estimate that OP_RETURN transactions constitute $\sim 1\%$ of the transactions in the blockchain, and use $\sim 0.3\%$ of its space.

Besides using OP_RETURN and *Pay-to-PubkeyHash* as shown in Section 2, there are other techniques to save metadata on the Bitcoin blockchain. With a slightly different flavour, the “sign-to-contract” and “pay-to-contract” [1, 27] allow to prove that, if a certain transaction is redeemed, then a certain value was known at the time it was put on the blockchain. A benefit of these techniques is that they do not affect the size of transactions. Comparing different methods to store metadata on Bitcoin would be an interesting topic for future research.

Although the official Bitcoin documentation discourages the use of the blockchain to store arbitrary data⁸, the trend seems to be a growth in the number of blockchain-based applications that embed their metadata in OP_RETURN transactions. We think that the main motivation for not using cheaper and more efficient storage is the perceived sense of security and persistence of the Bitcoin blockchain. If this trend will be confirmed, the specific needs of these applications could affect the future evolution of the Bitcoin protocol.

Related work. Besides ours, other projects aim at analysing metadata in the Bitcoin blockchain. For instance, blockchainarchaeology.com collects files hidden in the blockchain. These files are usually split into several parts, stored e.g. on different output scripts in a transaction. Various techniques are used to detect how the files were encoded (e.g. by binary grep on the PNG pattern) and to reconstruct them. The Bitcoin wiki [6] provides a list of protocols using OP_RETURN, together with their identifiers. Excluding those protocol identifiers that, at time of writing, are not used yet in any OP_RETURN transaction, the collection in [6] is strictly included in ours. The website opreturn.org shows charts about OP_RETURN transactions over time, organised by protocol, and statistics about their usage on the last week and over the last two years. The website smartbit.com recognises some OP_RETURN identifiers and shows related statistics. Finally, the website kaiko.com sells data about Bitcoin, including data related to OP_RETURN transactions.

Acknowledgments. The authors thank the anonymous reviewers of [BITCOIN 2017](https://arxiv.org/abs/1702.07623) for their insightful comments on a preliminary version of this paper. This work is partially supported by Aut. Reg. of Sardinia P.I.A. 2013 “NOMAD”.

References

1. Alternatives to opreturn, <http://bitcoin.stackexchange.com/questions/37206/alternatives-to-op-return-to-store-data-in-bitcoin-blockchain>. Last accessed 2017/02/15

⁸ The release notes of Bitcoin Core version 0.9.0 state that: “*Storing arbitrary data in the blockchain is still a bad idea; it is less costly and far more efficient to store non-currency data elsewhere.*”

2. Bitcoin scalability, https://en.bitcoin.it/wiki/Scalability_FAQ. Last accessed 2016/12/15
3. Bitcoin core dev update 5 transaction fees embedded data, <http://www.coindesk.com/bitcoin-core-dev-update-5-transaction-fees-embedded-data/>. Last accessed 2016/12/15
4. Bitcoin network survives surprise stress test, <http://www.coindesk.com/bitcoin-network-survives-stress-test/>. Last accessed 2016/12/15
5. Bitcoin OPRETURN explorer, <https://github.com/BitcoinOpReturn/>. Last accessed 2016/12/15
6. Bitcoin OP_RETURN wiki page, https://en.bitcoin.it/wiki/OP_RETURN. Last accessed 2016/12/15
7. Bitcoin pull request 5075, <https://github.com/bitcoin/bitcoin/pull/5075>. Last accessed 2016/12/15
8. Bitcoin pull request 5286, <https://github.com/bitcoin/bitcoin/pull/5286>. Last accessed 2016/12/15
9. Bitcoin release 0.10.0, <https://bitcoin.org/en/release/v0.10.0>. Last accessed 2016/12/15
10. Bitcoin release 0.11.0, <https://bitcoin.org/en/release/v0.11.0>. Last accessed 2016/12/15
11. Bitcoin release 0.12.0, <https://bitcoin.org/en/release/v0.12.0>. Last accessed 2016/12/15
12. Bitcoin release 0.9.0, <https://bitcoin.org/en/release/v0.9.0>. Last accessed 2016/12/15
13. Blockstore website, <https://github.com/blockstack/blockchain-id/wiki/Blockstore>. Last accessed 2016/12/15
14. Chainpoint website, <http://www.chainpoint.org/>. Last accessed 2016/12/15
15. Colu protocol, torrents, <https://github.com/Colored-Coins/Colored-Coins-Protocol-Specification/wiki/Metadata#torrents>. Last accessed 2016/12/15
16. Colu website, <https://www.colu.com/>. Last accessed 2016/12/15
17. Counterparty open letter and plea to the Bitcoin core development team, <http://counterparty.io/news/an-open-letter-and-plea-to-the-bitcoin-core-development-team/>. Last accessed 2016/12/15
18. Developers battle over bitcoin block chain, <http://www.coindesk.com/developers-battle-bitcoin-block-chain/>. Last accessed 2016/12/15
19. Diploma website, <http://diploma.report/>. Last accessed 2016/12/15
20. Eternity wall website, <https://eternitywall.it/>. Last accessed 2016/12/15
21. Factom website, <https://www.factom.com/>. Last accessed 2016/12/15
22. Kaiko data store, <https://www.kaiko.com/>. Last accessed 2016/12/15
23. La preuve website, <http://lapreuve.eu/explication.html>. Last accessed 2016/12/15
24. Omni website, <http://www.omnilayer.org/>. Last accessed 2016/12/15
25. Open assets website, <https://github.com/OpenAssets/>. Last accessed 2016/12/15
26. opreturn.org, <http://opreturn.org/>. Last accessed 2016/12/15
27. Pay-to-contract and sign-to-contract, <https://bitcointalk.org/index.php?topic=915828.msg10056796#msg10056796>. Last accessed 2017/02/15
28. Peter Todd delayed txo commitments, <https://peter todd.org/2016/delayed-txo-commitments>. Last accessed 2016/12/15
29. Proof of existence website, <https://proofofexistence.com/>. Last accessed 2016/12/15

30. Scalability debate ever end, <https://www.cryptocoinsnews.com/will-bitcoin-scalability-debate-ever-end/>. Last accessed 2016/11/30
31. Scaling debate in Reddit, <http://www.coindesk.com/viabtc-ceo-sparks-bitcoin-scaling-debate-reddit-ama/>. Last accessed 2016/12/15
32. Smartbit OP_RETURN statistics, <https://www.smartbit.com.au/op-returns>. Last accessed 2016/12/15
33. Stampery website, <https://stampery.com/>. Last accessed 2016/12/15
34. Ali, M., Nelson, J., Shea, R., Freedman, M.J.: Blockstack: A global naming and storage system secured by blockchains. In: USENIX Annual Technical Conference (2016)
35. Baqer, K., Huang, D.Y., McCoy, D., Weaver, N.: Stressing out: Bitcoin “stress testing”. In: Bitcoin Workshop. pp. 3–18 (2016)
36. Dermody, R., Krellenstein, A., Slama, O., Wagner, E.: Counterparty: Protocol specification (2014), http://counterparty.io/docs/protocol_specification/. Last accessed 2016/12/15
37. Garay, J.A., Kiayias, A., Leonardos, N.: The Bitcoin backbone protocol: Analysis and applications. In: EUROCRYPT. pp. 281–310 (2015)
38. Möser, M., Böhme, R.: Trends, tips, tolls: A longitudinal study of bitcoin transaction fees. In: International Conference on Financial Cryptography and Data Security. pp. 19–33. Springer (2015)
39. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. <https://bitcoin.org/bitcoin.pdf> (2008)
40. Reid, F., Harrigan, M.: An analysis of anonymity in the Bitcoin system. In: Security and privacy in social networks, pp. 197–223. Springer (2013)
41. Ron, D., Shamir, A.: Quantitative analysis of the full Bitcoin transaction graph. In: International Conference on Financial Cryptography and Data Security. pp. 6–24. Springer (2013)

Appendix A Additional charts

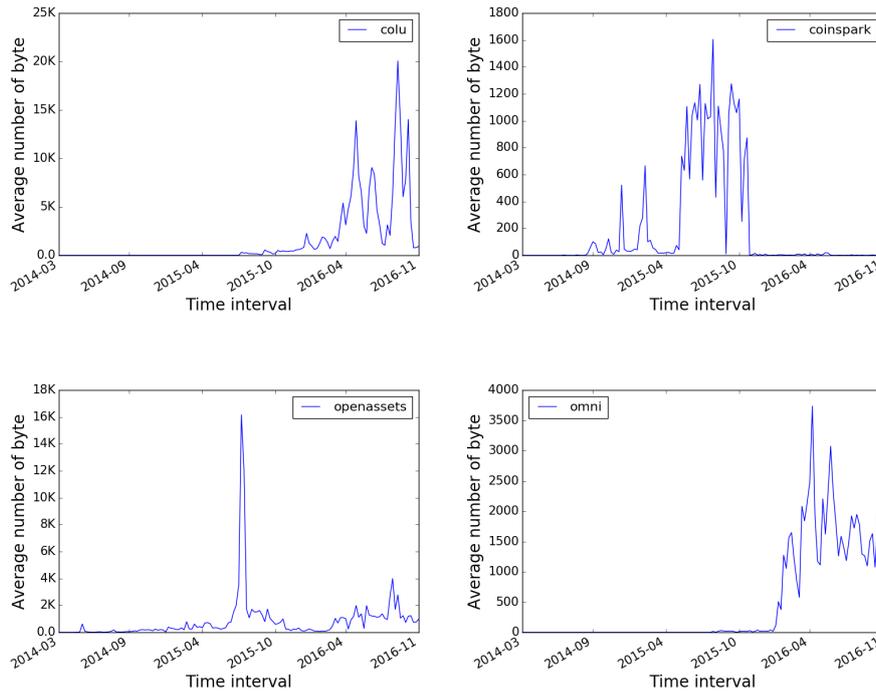


Fig. 4: Assets charts.

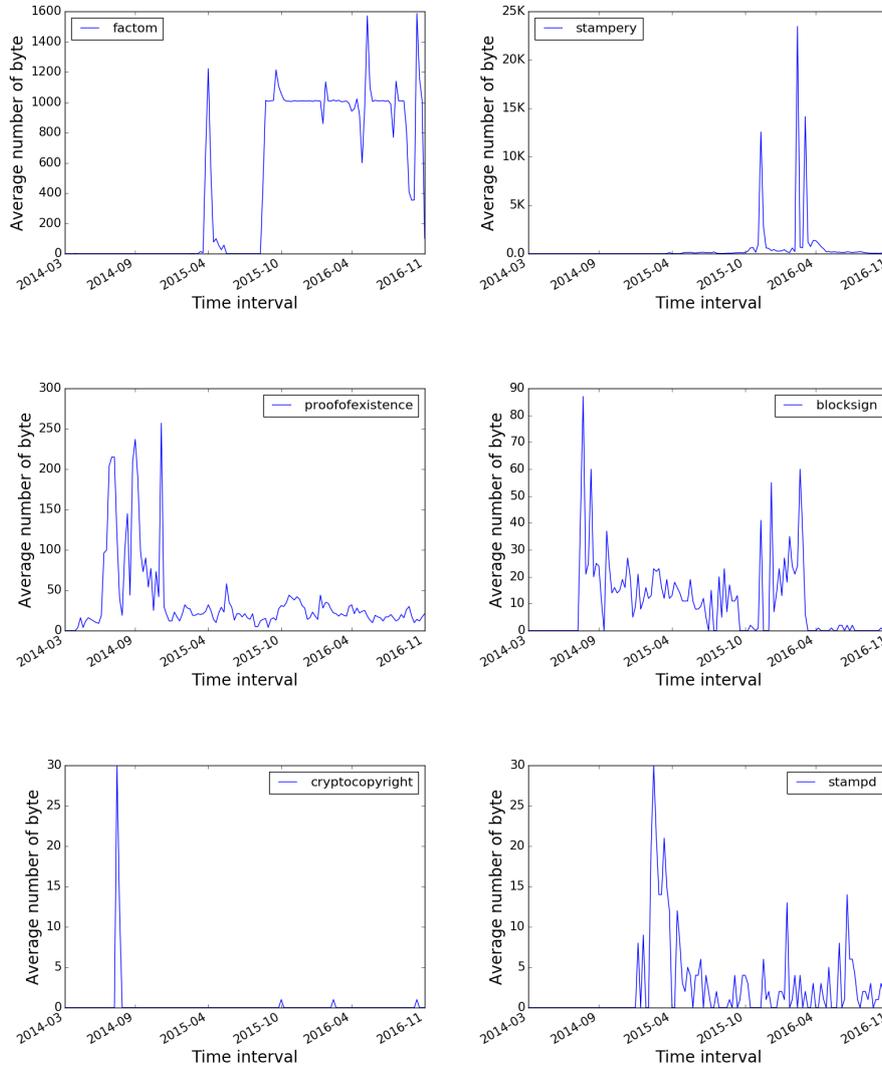


Fig. 5: Document Notary charts (1).

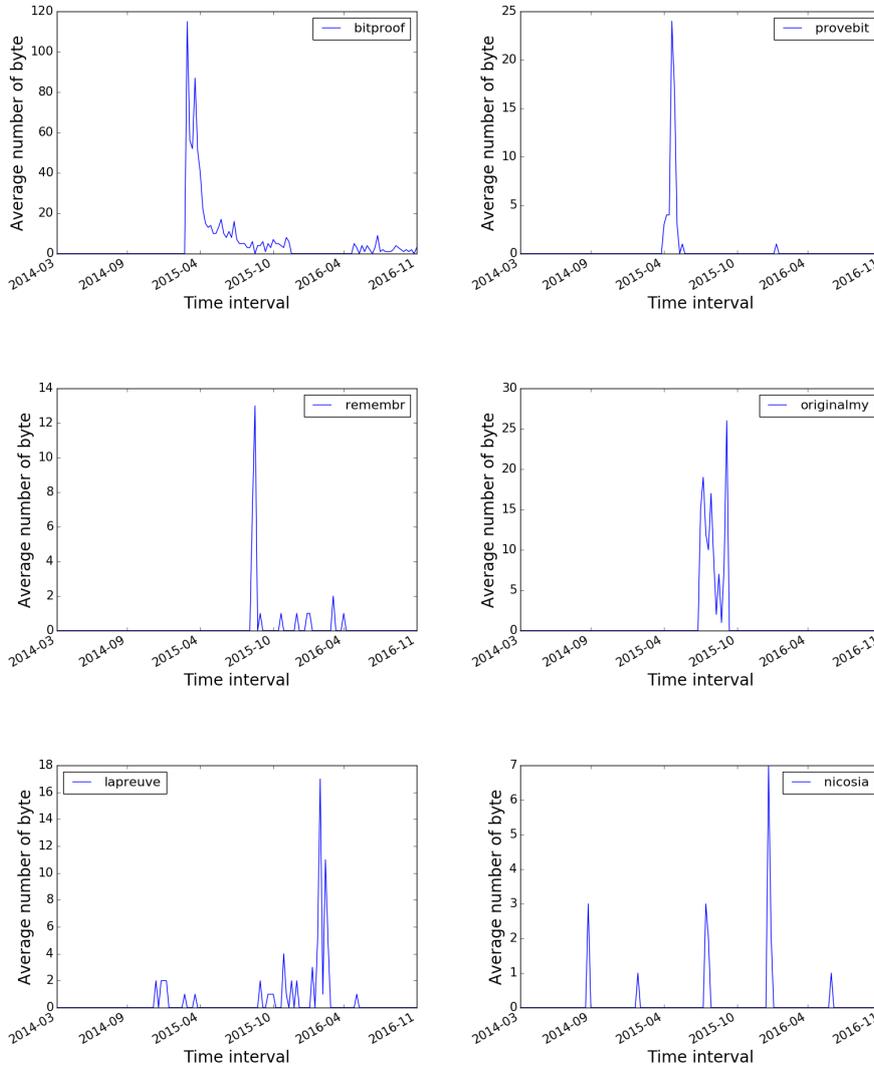


Fig. 6: Document Notary charts (2).

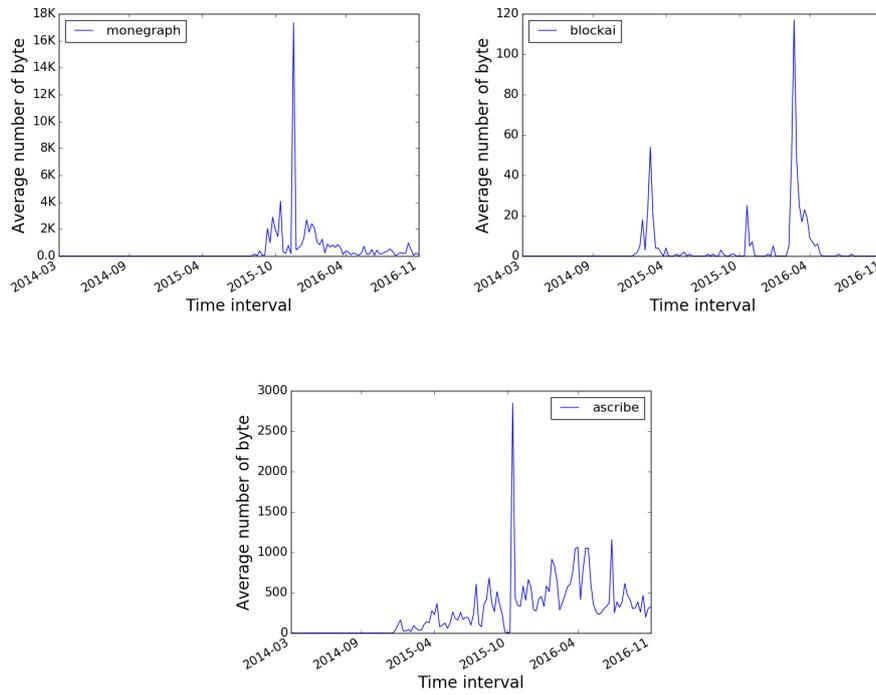


Fig. 7: Digital arts charts.

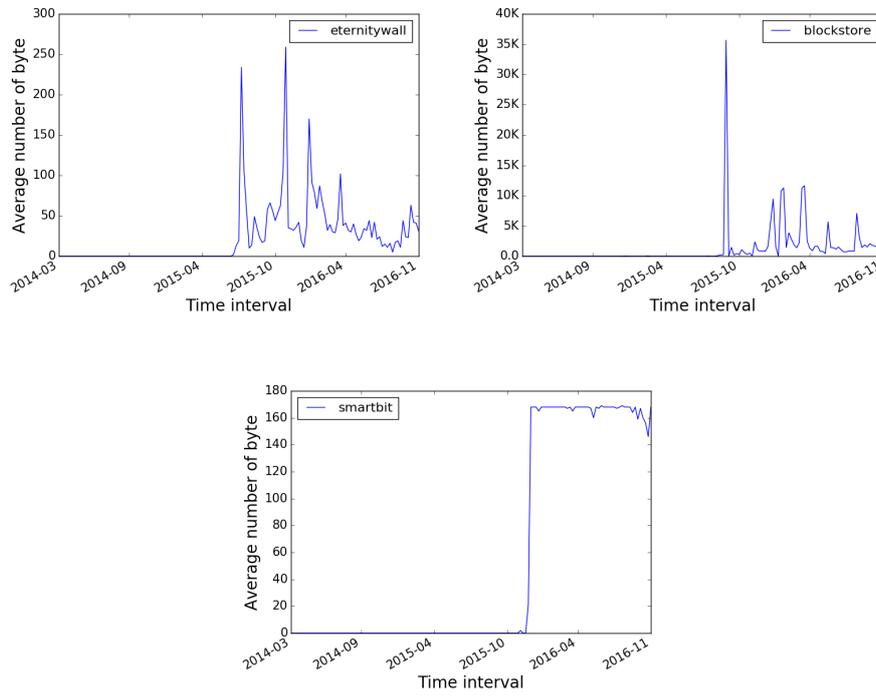


Fig. 8: Other Charts