Illicit Blockchain Content – Its Different Shapes, Consequences, and Remedies

Roman Matzutt, Martin Henze, Dirk Müllmann, and Klaus Wehrle

Abstract Augmenting public blockchains with arbitrary, non-financial content fuels novel applications that facilitate the interactions between mutually distrusting parties. However, new risks emerge at the same time when illegal content is added. This chapter thus provides a holistic overview of the risks of content insertion as well as proposed countermeasures. We first establish a simple framework for how content is added to the blockchain and subsequently distributed across the blockchain's underlying peer-to-peer network. We then discuss technical as well as legal implications of this form of content distribution and give a systematic overview of basic methods and high-level services for inserting arbitrary blockchain content. Afterward, we assess to which extent these methods and services have been used in the past on the blockchains of Bitcoin Core, Bitcoin Cash, and Bitcoin SV, respectively. Based on this assessment of the current state of (unwanted) blockchain content, we discuss (a) countermeasures to mitigate its insertion, (b) how pruning blockchains relates to this issue, and (c) how strategically weakening the otherwise desired immutability of a blockchain allows for redacting objectionable content. We conclude this chapter by identifying future research directions in the domain of blockchain content insertion.

Key words: blockchain content insertion, illicit content, pruning, redaction

Roman Matzutt
RWTH Aachen University, Ahornstraße 55 in 52074 Aachen (Germany) e-mail: matzutt@comsys.rwth-aachen.de

Martin Henze
RWTH Aachen University, Ahornstraße 55 in 52074 Aachen (Germany) and Fraunhofer FKIE, Fraunhoferstraße 20 in 53343 Wachtberg (Germany) e-mail: henze@cs.rwth-aachen.de

Dirk Müllmann
Goethe-Universität Frankfurt am Main, Theodor-W.-Adorno-Platz 4 in 60629 Frankfurt/Main (Germany) e-mail: muellmann@jur.uni-frankfurt.de

Klaus Wehrle
RWTH Aachen University, Ahornstraße 55 in 52074 Aachen (Germany) e-mail: wehrle@comsys.rwth-aachen.de

1 Introduction

Bitcoin introduced the blockchain as a digital ledger for financial transactions of its digital currency. The blockchain allows Bitcoin nodes to reach consensus about what transactions took place in the past and thereby prevents the double spending of coins by irrevocably recording all accepted transactions. The immutable and distributed ledger that is the blockchain proved highly beneficial for this application, and the technology is now being explored for various applications other than digital currencies, most notably digital notary services [8] and conflict resolution [69].

In addition to the trustworthy digital payments offered by cryptocurrencies, these applications store *non-financial* on the blockchain, i.e., they make use of smart contracts or augment Bitcoin transactions with arbitrary data. However, allowing any user to irrevocably store arbitrary data on the blockchain and distribute that data across the Bitcoin network also has notable downsides, especially if objectionable or even illegal content is added [34, 41, 43, 48]. Consequently, non-financial blockchain content has both great potential and a great inherent risk of misuse. Blockchain users, developers, and researchers should consider such negative consequences when using, developing, or improving blockchain systems.

This chapter aims to provide a holistic overview of the different shapes of non-financial blockchain content, the negative consequences objectionable or even illegal blockchain content can have, and to encourage researching further remedies. We first outline how blockchain content gets distributed across the blockchain's underlying peer-to-peer network and subsequently assess the technical and legal implications thereof. There, we focus on German law to show that illegal content could cause the possession of a full blockchain copy to become illegal in the future in one of the most prominently represented jurisdictions within the Bitcoin network [72]. We further provide a deep dive into the low-level methods for content insertion in Bitcoin and outline the protocols used by services and applications that build upon these methods, e.g., for vastly simplified file uploads to the blockchain.

Based on this foundation of blockchain content insertion, we discuss different lines of research for countermeasures aiming to mitigate negative impacts of blockchain content, i.e., (a) preventing unwanted content from ever entering the blockchain, (b) locally or globally pruning data from the blockchain without changing the global consensus of what is part of the blockchain, and (c) retrospectively redacting data from the blockchain. Based on this overview, we identify future research directions to strengthen the good uses of non-financial blockchain content while simultaneously mitigating the negative consequences of objectionable blockchain content.

Organization. In Section 2, we reiterate the network distribution of blockchain content and its potential technical and legal consequences. Section 3 then illustrates the basic methods for content insertion and how content insertion services use these building blocks. We then provide an overview of the *status quo* of content insertion in Bitcoin in Section 4. Afterward, we present proposed mitigation strategies: We discuss pre-insertion mitigation in Section 5, pruning both objectionable content and also obsolete data in Section 6, and redactable blockchains in Section 7. We outline future research directions in Section 8, and Section 9 concludes this chapter.

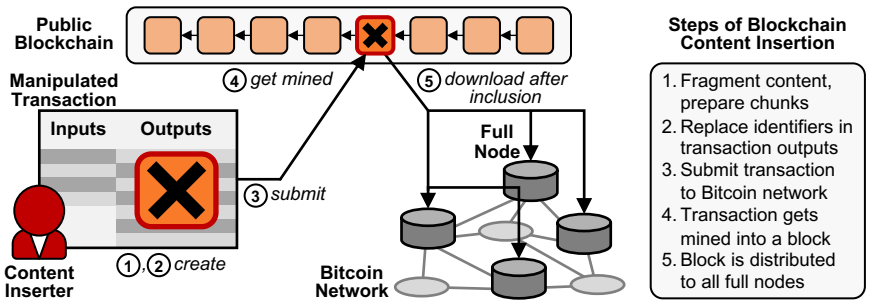

Fig. 1 Manipulated transactions get included on the blockchain and are hence distributed to the full nodes of the Bitcoin network. On the right, we detail the steps for inserting content.

2 Distribution of Blockchain Content and Its Consequences

We start off by reiterating the path a transaction takes from its proposal to the network to its acceptance into the blockchain. Based on this illustration, we highlight technical consequences of manipulating financial transactions to hold content as well as potential legal consequences if that content is privacy-sensitive or even illegal.

2.1 The Path of Blockchain Content and Technical Consequences

Blockchain systems establish consensus by distributing all valid data within the network so that each node (a) is aware of all information required to execute the next state transition and (b) can independently validate which data to accept. In this section, we show how objectionable content gets distributed accordingly. We further highlight the technical consequences of both this content distribution and naïve strategies nodes can apply locally to avoid storing such content.

Fig. 1 gives a high-level overview of the flow of information within the Bitcoin network and how objectionable content gets spread accordingly. The figure further details the basic steps of blockchain content insertion, from augmenting a transaction with content through manipulation to the content's distribution to the Bitcoin nodes. Any Bitcoin user can alter their transactions to contain arbitrary content. Most commonly, users will add the content by splitting it into small chunks and inserting them into the outputs of large transactions, e.g., by manipulating standard Bitcoin transactions. As we elaborate in Sect. 3.1, the user can, for instance, replace the mutable fields of standard Bitcoin transactions with the data chunks. In this case, extracting the data again becomes easy and only requires concatenating the mutable values again. The user then submits the transaction for insertion in a future block. While the transaction is effectively distributed across the network at this stage, we focus on the case that a miner irrevocably adds the content to the blockchain in the remainder of this chapter. As Bitcoin was not designed with transaction

manipulability in mind, the full nodes responsible for validating each transaction's correctness do not reject content-holding transactions or blocks. In fact, full nodes have no means to reliably identify manipulated transactions as we further discuss in Sect. 5.1. Hence, the objectionable content can enter the blockchain and is irrevocably stored once subsequent blocks have been mined on top of the content-holding block.

At this point, full nodes indefinitely store a verbatim copy of this block and the content within. Theoretically, they can choose to prune this raw block data after processing it completely. However, in this case, pruning nodes lose the ability to reconstruct their local block and transaction indexes if they experience technical issues. Instead, they rather have to download the full blockchain again. Moreover, newly joining nodes have to obtain and process a full copy of the blockchain as part of their bootstrapping process [44]. Thus, Bitcoin requires that a sufficient number of full nodes maintain a full blockchain copy and refrain from local pruning for the system to remain operable.

The inclusion of non-financial blockchain content also has multiple technical consequences. Beyond forcing full nodes to store unwanted data indefinitely, the practice of manipulating standard transactions unnecessarily bloats the set of unspent transaction outputs (UTXO set), which records all transaction outputs that are not provably unspendable, including manipulated outputs. Full nodes use the UTXO set as a lookup table to improve the performance of validating pending transactions [47]. Hence, bloating the UTXO set has negative consequences for all full nodes. Finally, this form of manipulating transaction outputs costs the content inserter money, which will become inaccessible and thus *burned* due to the manipulation (cf. Sect. 3.1). In the following, we further consider potential legal consequences node operators may face because of non-financial blockchain content.

2.2 Considerations of Potential Legal Consequences

As we discussed in the previous section, blockchain content gets distributed to all full nodes, which constitute the network's backbone. We now consider potential legal consequences resulting from this distribution pattern regarding both *illegal blockchain content* and potential *privacy violations*.

Illegal Content. We first consider the extreme case of child abuse imagery, which was already stored on Bitcoin SV's blockchain [11, 49], to discuss potential legal consequences for node operators. Courts need to rule whether such blockchain content constitutes an *accessible document* and whether a user *knowingly possesses* the content to determine whether that user becomes prosecutable. For instance, the USA [54], England [2], and Ireland [35] deem all data illegal that can be converted into a visual representation of illegal content. Here, the term *conversion* only covers creating a visual presentation, e.g., decoding an image file for displaying it. However, we expect that the interpretation of "convertibility" could be extended to also recognize blockchain content, given that its data chunks can be easily reassembled from a Bitcoin transaction (cf. Sect. 3.1).

This easy convertibility is already acknowledged, e.g., under German law. In Germany, criminal prosecution would require a statutory offense which, in our case, penalizes the users' mere possession of digital content. The German Criminal Code (StGB) sanctions the possession and the undertaking to obtain child and youth pornographic documents via §§ 184 b sec. 3 and 184 c sec. 3. Crucially, German law considers even storage devices, e.g., hard disks, to constitute "documents" (§ 11 sec. 3 StGB). Hence, possessing the storage device holding the blockchain suffices to possess its contents. Notably, these strict rulings even extend to illegal content that is only present in cache memory or RAM [19]. As a consequence, every user who maintains a local blockchain copy holding illegal content possesses prosecutable content within the meaning of the law; this still holds even though the content is typically fragmented due to its easy reassembly.

Criminal prosecution nevertheless requires that an alleged offender is also *knowingly* maintaining control over illegal content. In our scenario, where an unknown user stores illegal content on the blockchain, honest users verify incoming transactions obliviously and unaware of any illegal content. However, when such content is brought to a user's attention (e.g., as previously attempted by Interpol [34] or via media coverage [10, 11]), and the user does not immediately remove that content, the law assumes that the user *intends* to keep control over the content [67]. If full nodes were viewed as service providers, their blockchain copy could fall under punishment liberty [1, 32] as service providers have certain privileges when storing external information (§ 10 sec. 1 TMG[1]). However, punishment liberty again only applies as long as the provider has no knowledge of the illegal data or makes it inaccessible immediately, i.e., it may not apply anymore because of corresponding public reports [61, 67]. In consequence, any user who is aware of illegal content is required to delete it and thus cannot keep a full blockchain copy anymore.

Privacy Considerations. In addition to illegal content, also legal content that conflicts with the European General Data Protection Regulation (GDPR) can become problematic, e.g., when considering the right to delete or correct data and the right to be forgotten (Art. 16, 17 sec. 1, 2 GDPR). The GDPR here is concerned with the processing (Art. 4 No. 2 GDPR) of personal (Art. 4 No. 1 GDPR) data. Even though public blockchains operate pseudonymously, persons can remain identifiable, which then allows linking some publicly accessible blockchain information back to them [40]. At this point, we have to ask who is the *controller* when it comes to processing blockchain data within the meaning of Art. 4 No. 7 GDPR, as the controller has to ensure compliance with the privacy rights provided by the GDPR. However, who assumes this role in our scenario is currently subject to controversy and could include, e.g., the developers, miners, or network nodes [40]. While the developers define the processing rules for validating transactions, the miners and full nodes are responsible for the actual data processing. Further controversies debate whether all network nodes can be considered *joint* controllers (Art. 26 sec. 1 GDPR) [12, 40, 62]. In any case, the controller needs to ensure that the blockchain remains changeable to

[1] Concerning the applicability of TMG and Blockchain as a Telemedium cf. Peters [58].

Insertion Method	Capacity	Intended	Burns Coins	Constraints	Applications / Services
Identifier Manipulation	●	no	yes	—	Satoshi Uploader, CryptoGraffiti, Apertus (discussed in Sect. 3.2)
P2SH Stuffing	●	no	no	Previous TX & Colluding miner	P2SH Injectors (Sect. 3.2)
Non-St. Outputs	●	no	depends	Colluding miner	—
Non-St. Inputs	●	no	no	Previous TX	—
Value Encoding	◐	no	no	—	Bitcoin Message [65]
OP_RETURN	○ / ●*	yes	no	—	Various [8], Bitcoin Data Protocol (Sect. 3.2)
Nonce Field	○	no	no	Miner only	—
Coinbase	○	yes	no	Miner only	Miner advertisements [9], voting on BIPs [9], CoinPrune (Sect. 6.2)

Table 1 Overview of content insertion methods. The last column exemplifies known applications or insertion services. Bitcoin SV has an increased `OP_RETURN` capacity.

comply with the GDPR's defined privacy rights. Only few blockchain systems offer this property today, as we will further discuss in Sect. 7.

Summary. Despite the lack of court rulings, we anticipate that the outlined scenario and the current legal perspective in jurisdictions such as the USA or England are more worrisome than reassuring regarding the prosecution of blockchain users. In conclusion, we believe that blockchain developers must keep the potential that illegal content can jeopardize the whole system in mind.

3 Content Insertion Methods

In this section, we illustrate the intended and unintended ways for Bitcoin users to augment their transactions with additional content. First, discuss where content can be stored within a Bitcoin transaction or block. After presenting these methods, we present higher-level protocols and services used to further facilitate content insertion.

3.1 Low-Level Insertion of Non-Financial Blockchain Data

We first review the technical options users have to augment Bitcoin transactions with additional content. Table 1 provides an overview of the available methods.

Identifier Manipulation. Bitcoin transactions consist of inputs, which unlock funds from previous transactions, and outputs, which specify new spending conditions. Both inputs and outputs make use of Bitcoin's scripting language [16] to express and satisfy the spending conditions, respectively. While this scripting language, in theory, allows for flexible spending conditions, only a few standard script patterns based on *ownership identifiers* are used for transferring bitcoins [55]. The *pay to public key (P2PK)* and *pay to public-key hash (P2PKH)* patterns specify a public key or hash value thereof, and spending these outputs requires knowledge of

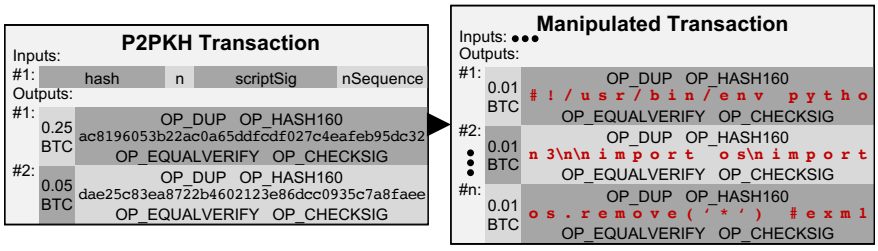

Fig. 2 The mutable identifiers defining spendability conditions of transaction outputs can be replaced with chunks of arbitrary data.

the corresponding private key. Similarly, the *m-out-of-n pay to multisig (m-n-P2MS)* pattern specifies n public keys and m corresponding private keys are required for spending the associated coins, i.e., the key holders have to agree on spending the funds. Finally, the *pay to script hash (P2SH)* pattern commits to a spending condition by only publishing the hash value of the script specifying the full condition. Spending the corresponding funds then requires disclosing the script matching the hash value alongside the other values required for unlocking funds.

While purely intended for financial transactions, all standard script patterns can be misused to store arbitrary blockchain content instead. Fig. 2 illustrates this *manipulability* using P2PKH as an example; the method, however, generalizes to other patterns with varying per-byte costs [43]. Most Bitcoin transactions have two P2PKH outputs [42], which have the pattern `OP_DUP OP_HASH160 [public key hash] OP_EQUALVERIFY OP_CHECKSIG` [55]. The user can then spend coins secured by this kind of transaction output by providing a `[signature][public key]` pair. Hence, the P2PKH script pattern first matches the submitted public key against the required hash value before verifying the supplied signature, proving that the user possesses the corresponding private key and can spend the coins.

However, even though Bitcoin nodes execute these checks for each attempt to spend the coins of a P2PKH output, the nodes cannot verify that the hash value within the output script indeed corresponds to a known key pair. As illustrated in Fig. 2, a user can thus replace the 20 B-long hash value with arbitrary data. Furthermore, the user can use additional P2PKH outputs to encode more data chunks within the same transaction. By packing a single transaction with as many manipulated outputs as possible, the user can maximize the transaction's capacity for holding content. This insertion method is simple but has the drawback of *burning* bitcoins, i.e., making them inaccessible in the future: Each transaction output (except for `OP_RETURN`, see below) needs to spend a minimum amount of Bitcoins, called *dust* [55]. Since the user arbitrarily manipulated the output's identifier, it is highly unlikely that a matching private key exists, and the associated bitcoins are likely lost forever [43].

Notably, this content insertion method satisfies the easy extractability of the content we discussed in Sect. 2.2. Namely, the transaction creator can entirely determine the ordering of all outputs, and concatenating all mutable fields of the transaction's output scripts is sufficient to obtain the content.

P2SH Stuffing. Pay to script hash (P2SH) transaction outputs only contain a commitment to a *redeem script*, which specifies the actual spending conditions of transferred funds. The redeem script should also follow a standard pattern as described above [5] and require that only corresponding input values are present. However, while full nodes discard pending transactions that violate these rules, they accept the transaction if a miner included them in a block. Bypassing these restrictions allows for more sophisticated content insertion methods involving the redeem script. Sward et al. [66] investigated how to utilize P2SH input scripts for content insertion. A key factor here is that individual stack elements, i.e., data that is being pushed onto the stack when evaluating a script, are restricted to a size of 520 B [66]. The redeem script itself is a stack element as well and therefore also affected by these restrictions. However, content can be encoded as additional input values and then processed by the redeem script, e.g., by dropping the inputs [66], which allows for utilizing P2SH input scripts more efficiently. Using this approach further has the advantage of not burning any coins, as identifier manipulation would.

Non-Standard Transaction Scripts. Like non-standard P2SH redeem scripts, Bitcoin nodes should not relay transactions whose outputs diverge from standard patterns. However, since the same checks are in place here as well, such transactions can still occasionally enter the blockchain. This form of content insertion promises the highest capacity since the inserting user does not have to cater to any other rule, e.g., the script could only push and drop new content chunks. However, the high likelihood that the transaction will never be mined into a block makes the method cumbersome to use. Input scripts, contrarily, are not prone to the same restriction, e.g., a similar pattern to P2SH stuffing could be applied.

Encoding via Values. Transactions can further be augmented with arbitrary content by encoding it using the values transferred via multiple outputs. Sleiman et al. [65] proposed using arithmetic coding to encode single letters per output, which would yield a maximum capacity of around 3 KiB for a standard transaction. Furthermore, the user needs more coins than the required dust amounts to encode the different letters in their outputs. Hence, the value fields can be used to insert additional blockchain content, and different methods than the one proposed by Sleiman et al. [65] are conceivable. Yet, this way of inserting blockchain data is more restricted than, e.g., identifier manipulation, even though value encoding does not burn coins.

OP_RETURN. Since 2014 [17], Bitcoin has provided a special script type for explicitly augmenting regular transactions with small chunks of arbitrary data. Starting the script of a transaction output with the `OP_RETURN` operation causes the output to be provably unspendable [17], i.e., adding data after the `OP_RETURN` operation will not bloat the UTXO set (cf. Sect. 2.1). However, since content insertion is still discouraged and `OP_RETURN` was meant as a form of compromise [17], additional rules are in place to prevent mass content insertion. Namely, a transaction may only have one `OP_RETURN` output, and the payload of this output is restricted to 83 B (formerly 80 B) [8]. The fork Bitcoin SV, for instance, allows for much larger payloads of `OP_RETURN` transaction outputs (up to 100 kB [50]), which was previously used for inserting child abuse imagery into that blockchain [11, 49].

Miner-Exclusive Insertion. Miners have the most control over shaping the blockchain, but they need to follow the consensus rules to prevent their blocks from being rejected by the network's full nodes. Hence, block headers are comparably resilient to content insertion. The version field is fixed by the consensus rules, and the values of both the timestamp and target fields must lie within reasonable ranges in order to be accepted by full nodes. Contrarily, the miner is expected to randomly fill the nonce field for its proof of work (PoW). In theory, the miner could thus fix this field and solve the PoW by varying other aspects of its block, e.g., altering the block's transactions, to find a valid block nevertheless. However, this approach is tedious, yields only four bytes of space for arbitrary content, and a higher-capacity alternative exists: The first transaction of each block is a special *coinbase* transaction, which mints new bitcoins that the miner is free to transfer to itself. The special structure of coinbase transactions allows for inserting arbitrary data as well. The input script of the coinbase transaction contains a special coinbase field with a length of up to 100 B. The field is expected to contain the block's position within the blockchain encoded as a 4 B-integer [4]. However, the gap created by specifying a larger field length can be filled with arbitrary data. In fact, Bitcoin's genesis block already uses this method [64], and thus we consider using the coinbase field as an intended means of blockchain content insertion. In the past, the coinbase field has been used, e.g., to advertise mining pools [9] or to vote for improvements to Bitcoin's consensus rules [5, 9]. We further discuss in Sect. 6.2 how nodes can use the coinbase field to coordinate extended features such as globally pruning obsolete blockchain data.

We have illustrated the variety of available methods to insert non-financial data into Bitcoin's blockchain in this section. The most important among those methods are (a) identifier manipulation, as this approach allows for inserting larger content, (b) `OP_RETURN` transactions, which provide an intended method for augmenting transactions with small chunks of data, and (c) storing a small data chunk in the coinbase field, which facilitates extended coordination processes among miners. We now give an introduction to content insertion services, which build upon these methods to further facilitate blockchain content insertion.

3.2 Content Insertion Services

In this section, we present auxiliary tools and online services that facilitate the insertion of blockchain content. In particular, we discuss the technical details of how these services utilize the building blocks (cf. Sect. 3.1) and their unique features.

Satoshi Uploader. This content insertion service consists of two Python scripts that were published on the Bitcoin blockchain, one for encoding a file as a Bitcoin transaction and one for extracting it later on [64]. We refer to these scripts as the *Satoshi Uploader*, as their copyright label reads Satoshi Nakamoto, although this claim is questionable [64]. The Satoshi Uploader uses 1-3-P2MS transaction outputs and uncompressed public keys, which have a length of 65 B per key, to embed the input file and its CRC32 checksum into a Bitcoin transaction. The script pads the

last chunk with zero-bytes to fill up the last public key and, if possible, reduces the number of public keys in the last P2MS output.

CryptoGraffiti. *CryptoGraffiti* [27] is a web service launched in 2014, which originally was intended for parsing blockchain content and subsequently started to also offer a form to write text and files to the blockchain [22]. Initially launched on the Bitcoin (Core) blockchain, the service shifted to Bitcoin Cash in October 2017 and, as of October 2020, only operates on Bitcoin SV. Users of the service are instructed to send the funds required to engrave the content as well as a 10 % service fee to a Bitcoin address, which is then used to create the transaction [27]. Similar to the Satoshi Uploader, CryptoGraffiti relies on identifier manipulation, but the service uses P2PKH scripts instead of P2MS scripts. Each output spends the dust amount of 546 sat/B, and the profits are sent to a fixed Bitcoin address, which facilitates the detectability of CryptoGraffiti transactions. The service allows uploading files of up to 50 KiB together with optional comments.

P2SH Injectors. *P2SH Injectors* describe a class of services that insert content using the P2SH stuffing method we described in Sect. 3.1. As with the Satoshi Uploader, the origin of this service can be traced back to a Python script [39], but slight variations of corresponding transactions can be found on Bitcoin's blockchain.

Apertus. The service Apertus [33] provides a sophisticated protocol for encoding large files across multiple transactions. Pending transactions are usually independent of each other, and a user submitting multiple transactions has no direct control over which blocks the transactions will be added to and in which order. To circumvent this limitation, Apertus uses a second layer of chunking. The file to be uploaded is stored in P2PKH outputs of multiple transactions, and a final *archive transaction* provides a lookup table of the transaction identifiers required for reassembling the original file. Notably, archive transactions can be stored hierarchically, i.e., one archive transaction can point to multiple fragments of another, larger archive. Apertus transactions can contain additional meta information, such as a digital signature, a textual comment, and an optional filename.

Bitcoin Data Protocol. The *Bitcoin Data Protocol (BDP)* [68] specifies a protocol for `OP_RETURN`-based data insertion. This protocol is heavily utilized on Bitcoin SV (cf. Sect. 4.3), where an `OP_RETURN` output can hold up to 100 kB [50]. BDP uses a fixed protocol identifier and stores binary data and meta information as individual push operations in the `OP_RETURN` output script: `[ID][payload][mime type][encoding (optional)][filename (optional)]...` [68]. Other services, such as Money Button (`https://www.moneybutton.com`) or the blockchain-backed social network Twetch (`https://twetch.app`), rely on BDP and encapsulate a BDP message in their `OP_RETURN` outputs. Money Button was used to store child abuse imagery on Bitcoin SV's blockchain in the same month the full capacity of Bitcoin SV's `OP_RETURN` outputs could be used [11, 49].

In conclusion, content insertion services are as heterogeneous as the building blocks they rely on. The spectrum ranges from simple scripts for merely embedding content into a transaction, over easy-to-use web services, to sophisticated protocols to overcome space limitations imposed by consensus rules. In the following, we investigate how these different content insertion methods have been used over time.

4 Quantitative and Qualitative Analysis

We now provide present to which extent the respective content insertion methods and services we presented in Sect. 3 are used in the wild. To this end, we monitored the blockchain of the original Bitcoin network, i.e., Bitcoin Core, as well as its direct active forks Bitcoin Cash and Bitcoin SV. After outlining our methodology, we first analyze the overall utilization of low-level insertion methods across blockchains and then focus on the number and types of files inserted via content insertion services.

4.1 Methodology

For our measurements, we extend the measurement framework we implemented for our previous study [43] and rely on the same methodology if not stated otherwise. With this extension, we can now analyze transactions from the blockchains of Bitcoin Core, Bitcoin Cash (forked off Bitcoin Core at block height 478 558), and Bitcoin SV (forked off Bitcoin Cash at block height 556 766) up to a block height of 640 000. While Bitcoin Core reached this block height on Jul 20, 2020, Bitcoin Cash reached it on June 17, 2020, and Bitcoin SV reached it on June 19, 2020, respectively. In all measurements, we only consider blocks exclusive to the respective blockchain. For instance, we ignore the shared prefix of blocks between Bitcoin Core and Bitcoin Cash when discussing Bitcoin Cash. Furthermore, we aggregate all data month-wise. We apply automated checks to identify and classify transactions based on the distinct patterns used by the content insertion services we identified in Sect. 3.2. Specifically, the MIME type field used in the Bitcoin Data Protocol (BDP) and assumptions about which data push operation of a BDP `OP_RETURN` output should contain the payload data allow us to investigate the (claimed) type of data inserted via this protocol without manual investigation.

4.2 Low-Level Content Insertion

In this section, we provide an overview of the extent of content insertion in general by first discussing its evolution on the blockchain of Bitcoin Core since our first study [43] and by then identifying similarities and differences between content insertion in Bitcoin Core, Bitcoin Cash, and Bitcoin SV, respectively.

We first compare the overall utilization of content insertion methods across the three considered blockchains. Fig. 3 shows how many transactions using the different insertion methods we discussed in Sect. 3.1 were cumulatively included on the blockchains of Bitcoin Core (Fig. 3a), Bitcoin Cash (Fig. 3b), and Bitcoin SV (Fig. 3c), respectively. This analysis continues our previous study [43], where we only considered the blockchain of Bitcoin Core up to a block height of 482 870 as of August 31, 2017. Compared to that study, the number of transactions with

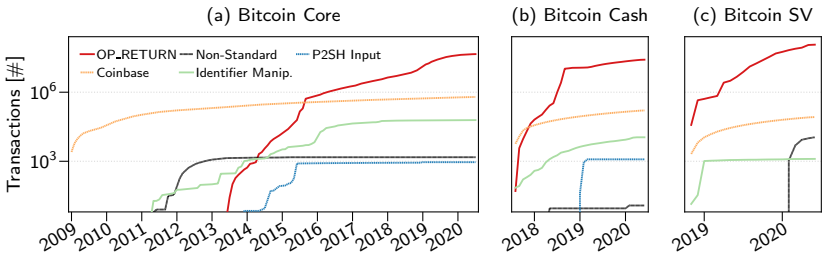

Fig. 3 Number of content-carrying transactions exclusive to the blockchains of (a) Bitcoin Core, (b) Bitcoin Cash, and (c) Bitcoin SV based on our content detectors.

manipulated transaction outputs on Bitcoin Core's blockchain increased by 24.5 % from 49 342 to 61 433 instances. Contrarily, we observed only 62 additional transactions that insert content through P2SH input scripts (using P2SH Injectors) and only 3 additional non-standard transactions on this blockchain since our last measurement [43]. The blockchains of Bitcoin Cash and Bitcoin SV have a similar yet accelerated general development. However, we make four additional observations. First, identifier manipulation is actively being used on all considered blockchains. We observe 11 009 instances of identifier manipulation exclusive to Bitcoin Cash, i.e., this method was used on both Bitcoin Cash and Bitcoin Core to a similar extent in parallel. Contrarily, we only observed 1269 instances of identifier manipulation on the blockchain of Bitcoin SV since its inception, and utilization steeply declined since only 242 instances occurred after January 2019. Second, both Bitcoin Cash and Bitcoin SV make increased utilization of `OP_RETURN` outputs compared to Bitcoin Core. Notably, Bitcoin SV has already accumulated 2.57× the number of transactions making use of an `OP_RETURN` output compared to Bitcoin Core despite its short lifetime. Fig. 4 further highlights this observation by showing the share of all transactions in each considered blockchain that have an `OP_RETURN` output. While the share of Bitcoin Core transactions holding an `OP_RETURN` output increased to 8.2 % of all transactions, both Bitcoin Cash (55.5 %) and Bitcoin SV (68.3 %) experience a vastly higher `OP_RETURN` utilization. Third, Bitcoin SV accumulated a comparably large amount of non-standard transactions (11 059) in only one and a half years. For comparison, we observed 1500 non-standard transactions in Bitcoin Core and only 12 non-standard transactions in Bitcoin Cash. Fourth, we did not observe any content insertion via P2SH input scripts for Bitcoin SV.

In summary, content is actively being inserted via both intended (`OP_RETURN`) and unintended means (identifier manipulation) on the considered blockchains. Notably, non-standard transactions are still being mined into blocks as of 2020.

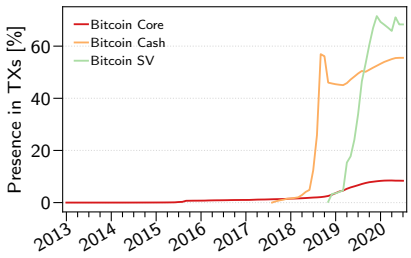

Fig. 4 The utilization of `OP_RETURN` outputs differs across Bitcoin variants.

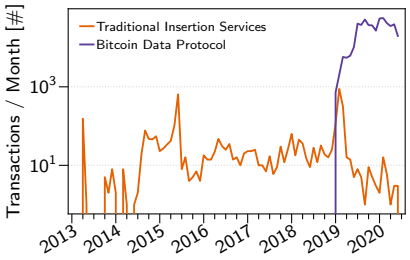

Fig. 5 BDP is now used frequently compared to traditional insertion services.

4.3 Service-based Content Insertion

In this section, we assess to which extent content insertion services are used to insert non-financial data into the blockchains of Bitcoin Core, Bitcoin Cash, and Bitcoin SV. We first give an overview of the general utilization of content insertion services across the blockchains. Then, we reconsider those content insertion services we identified in our previous study [43], i.e., the Satoshi Uploader, CryptoGraffiti, Apertus, and P2SH Injectors (cf. Sect. 3.2), which we collectively refer to as *traditional* content insertion services in the following. Finally, we focus on the utilization of the Bitcoin Data Protocol (BDP) on the blockchain of Bitcoin SV.

Overall Utilization. Fig. 5 provides an overview of the utilization of traditional content insertion services and BDP across the considered blockchains. We observe that service-backed content insertion occurs continually starting from June 2014, with the earliest utilization in April 2013. However, even though this kind of content insertion is present each month, traditional content insertion services were only rarely used. With five exceptions, all traditional content insertion services combined have only been used at most 80 times a month across all considered blockchains. We observed 25.5 insertions per month across all considered blockchains using traditional content insertion methods and a total of 3540 insertions. In stark contrast to traditional content insertion services, the `OP_RETURN`-based BDP is being heavily used on Bitcoin SV. Since its inception in January 2019, we recognized an average of 25 461.7 BDP insertions per month, accumulating to a grand total of 509 234 instances within the 83 234 blocks we considered.

Traditional Content Insertion Services. We further investigate each service's contribution to the overall extent of content insertion, starting with the traditional services. Fig. 6 shows the month-wise utilization of the traditional content insertion services accumulated over all considered blockchains. The overall largest contribution here comes from P2SH Injectors, which were used to insert data in 2011 instances. However, their utilization is mostly concentrated in shorter bursts. The first burst occurred on Bitcoin Core during May and June 2015 (679 instances). During the second burst, we first observed 59 instances of P2SH Injector utilization on Bitcoin Core in December 2018 and January 2019 and subsequently observed

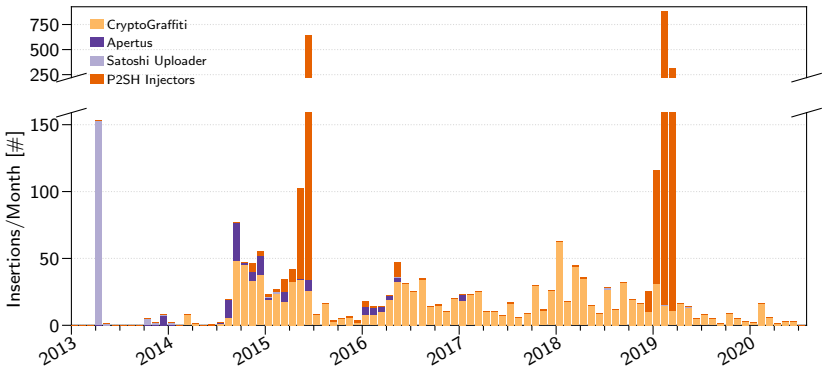

Fig. 6 This per-month overview shows the ongoing, but declining (in favor of BDP) utilization of traditional insertion services.

1215 instances between January and March 2019 on Bitcoin Cash. Next, we observed 1245 instances where CryptoGraffiti was used to insert data. In contrast to P2SH Injectors, CryptoGraffiti is being used continually since July 2014, with 9 initial instances during March and April. In correspondence to the service operator's announcements (cf. Sect. 3.2), we observe a transition of CryptoGraffiti utilization from Bitcoin Core to Bitcoin Cash during October 2017 and another shift from Bitcoin Cash to Bitcoin SV during November and December 2018. Both remaining services, the Satoshi Uploader and Apertus, are less frequently used with 167 and 117 observed instances, respectively. However, we only counted archive transactions for Apertus, and we provide more details on the occurring data fragmentation using this service in our previous study [43]. Specifically, we did not observe any new utilization of Apertus since that study. Contrarily, we observed two new instances of Satoshi Uploader transactions in July 2018 on Bitcoin Cash and in May 2019 on Bitcoin Core, respectively. In conclusion, the overall utilization of traditional content insertion decreased noticeably after March 2019, but at least CryptoGraffiti is actively being used to insert new content via identifier manipulation.

Bitcoin Data Protocol. As Fig. 5 indicates, the decline in utilizing traditional content insertion roughly correlates to the increasing popularity of the Bitcoin Data Protocol (BDP) on Bitcoin SV. BDP has the advantage for content inserters of allowing them to use 100 kB-large `OP_RETURN` outputs to insert their data and meta information, such as the inserted data's MIME type, in a structured manner. Figs. 7 and 8 show the absolute numbers of transactions with BDP payload per month on Bitcoin SV and highlight the relative shares of data types, respectively. We observe that the utilization of BDP is dominated by text-based formats such as plain text and HTML or Markdown documents. Collectively, text-based payloads constitute 64.38 % of all BDP instances. Next, 20.58 % of BDP instances relate to image data, with a total of 104 820 instances we observed within our considered time frame. Third, structured data, such as JSON, XML, or CSV data, makes up 12.62 % of all BDP instances. This data was inserted in a larger bulk between August and December

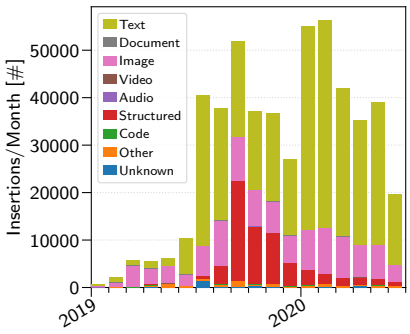

Fig. 7 Absolute numbers of BDP insertions per month and payload type.

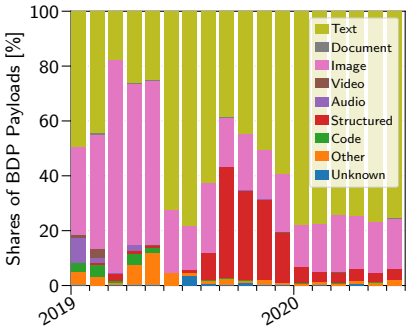

Fig. 8 Relative shares of payload types among BDP insertions per month.

2019 but continues to constitute 4.25 % of BDP insertions on average. Noticeably, BDP is also being used to insert (alleged) audio (363 instances) and video data (170 instances) as well as code (1044 instances), e.g., Python, JavaScript, and Shell scripts, and other documents (325 instances) such as PDFs or Microsoft Office documents. In summary, BDP lifted blockchain data insertion to an unprecedented level. While most data stored on the blockchain of Bitcoin SV this way is text or structured data, we observe over 100k alleged images and other media and documents that were inserted via the protocol.

Our analysis highlights that content is actively being inserted into the blockchains of Bitcoin Core, Bitcoin Cash, and Bitcoin SV. The Bitcoin Data Protocol has elevated the phenomenon to a new level in Bitcoin SV, but also identifier manipulation is actively being used in parallel to insert data, e.g., via the CryptoGraffiti service.

5 Mitigation of Unwanted Blockchain Content Insertion

After having illustrated and quantified the methods for blockchain content insertion and the consequences of inserting illicit content, we now shift our focus to mitigation strategies. In this section, we start by exploring the available design space for hindering content-holding transactions from entering the blockchain in the first place. Namely, we discuss *filtering* content-holding transactions out, imposing *mandatory minimum fees*, *commitments* to a transaction's benignity, and alternative *balance-based blockchain models*. In the subsequent sections, we then discuss approaches that provide countermeasures against objectionable content after the fact.

5.1 On the Capability to Filter Objectionable Content

The intuitive countermeasure against blockchain content insertion would be a "content firewall," i.e., nodes actively analyze transactions for content and filter out any content-holding transactions. In the following, we highlight the fundamental complications of this approach, namely that *full detection of manipulation is impossible* and that *frequent rule updates are not feasible for public blockchain systems* [42].

Impossible Prevention of Content Insertion. Bitcoin allows for easy identifier manipulation as full nodes have to accept transactions in good faith without knowing whether the transferred coins will, or can, be spent in the future. However, content insertion remains possible even if the full nodes had the means to reject any transaction that sends funds to unknown private keys [42]. Intuitively, a content inserter can still brute-force private keys such that the resulting on-chain identifier holds a chunk of content. By only using parts of the identifier for storing content, e.g., the first five bytes, and by again packing transactions with many outputs of this kind, the user can still encode content (albeit at a reduced capacity) [42]. This practice is already used to find so-called vanity addresses [9], and the resulting transactions remain fully spendable. Hence, content can still be added to entirely valid Bitcoin transactions in a computationally feasible manner, even in the strictest possible scenario.

Challenges for Filtering Content. Because of the impossibility of reliably preventing all content insertion, full nodes would have to inspect transactions further to identify encoded content. However, naïvely applying dedicated *content detectors* leads to the following three main problems.

First, content inserters may slightly adapt their methods to evade known content detectors, e.g., by using the last bytes of each identifier to encode content instead of the first ones. As a consequence, each node likely has to apply a large number of content detectors to vet a transaction.

Second, this flexibility is challenging regarding the coordination required between nodes. Full nodes must assume that individual content-holding transactions are mined into a block just like non-standard transactions (cf. Sect. 4.2). In this case, full nodes must universally agree on a set of *mandatory* content detectors to prevent blockchain forks. Hence, frequent variations of insertion methods require equally frequent updates of full nodes, which can result in uncontrolled forking [42].

Finally, false positives can negatively impact the filtering process and render benign transactions invalid [42]. False positives are unlikely to result from larger transactions and could easily be resolved, e.g., by randomly reordering the outputs. However, individual Bitcoin addresses may become unusable, e.g., because their corresponding public key hash consists of printable ASCII characters and is mistaken for a text-holding transaction output [42]. Nodes can mitigate the influence of false positives by applying content detectors only to transactions with more than, e.g., five outputs [42]. If the content only requires fewer outputs, the content inserter could use a single `OP_RETURN` output instead, and he would face lower costs with that approach.

In conclusion, a firewall against objectionable content would be desirable to have. However, the additional required network coordination and the high flexibility for content inserters prevent this firewall from being easily deployable.

5.2 Financially Disincentivizing Content Insertion

A different approach to detecting and discarding content-holding transactions is to deter users from inserting content. In the following, we explore the possibility to *financially disincentivize* the insertion of non-financial blockchain content.

The main idea here is to extend the transaction fees users pay to miners as an incentive to favor their transactions to *mandatory minimum fees* [42]. Instead of letting miners prioritize transactions paying higher fees, all full nodes would then reject transactions that underpay the mandatory fees. This proposal is motivated by the observation that, typically, many transaction outputs are required to encode objectionable content. For instance, inserting 1 kB of data using P2PKH identifier manipulation requires 50 outputs [42], while the vast majority of Bitcoin transactions have fewer outputs. Many transactions even have only up to five outputs [42]. Hence, the goal is to penalize creating very large and potentially content-holding transactions while not increasing the fees required for the vast majority of benign transactions.

Currently, the Bitcoin Core client monitors the recent per-byte fees paid by other transactions and the delay after which they were added to the blockchain to estimate the fees required for its own transactions to be included in a timely manner. Instead of this probabilistic and locally obtained per-byte fee [26], mandatory minimum fees rely on a fixed per-byte component and additionally have an increasingly penalizing per-output component [42]. The per-output component could be chosen to grow disproportionally for transactions with more outputs since most transactions only have few ($\leq$5) outputs [42]. Hence, penalizing extraordinarily large transactions suitable to contain considerable amounts of content (e.g., $\geq$50 outputs or 1 kB of data) can create an artificial barrier to inserting such content. Treating larger transactions in this special way further helps to mitigate any negative financial impact on the vast majority of benign (typically small) transactions [42]. However, a refinement of this approach that takes the fluctuation of fees into account while ensuring network-wide consensus about the mandatory minimum for transaction fees would be required for practical deployability and currently remains an open research question.

5.3 Cryptographic Commitments to Transaction Benignity

In Sect. 3.1, we have established that blockchain content insertion is currently as easy as replacing mutable identifiers within transactions with chunks of the content. We now summarize an approach [42] to prevent this simple manipulability by updating these identifiers such that alterations become detectable by any node. Hence, content inserters lose control over what data is persisted on and derivable from the blockchain beyond brute-forcing the updated identifiers (cf. Sect. 5.1).

Namely, the approach replaces the mutable identifiers with irreversible *identifier commitments* that are *self-verifying* [42]. This means that (a) transactions only hold one-way commitments to the mutable identifiers, i.e., obtaining the original identifier

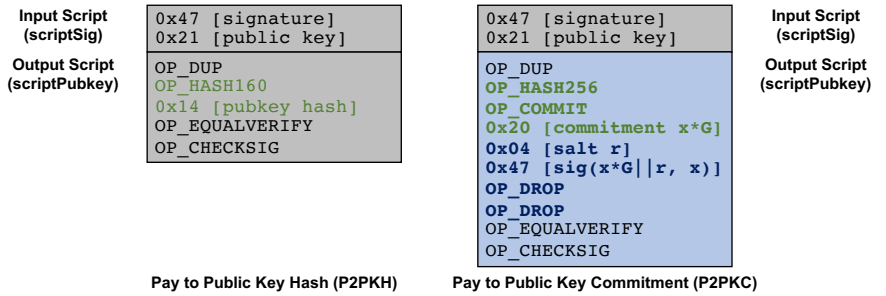

Fig. 9 P2PKC, a slight adaption of P2PKH output scripts, enables the utilization of identifier commitments on the blockchain.

from the identifier commitment is infeasible, and (b) identifier commitments are created in a way such that any tampering with them is easily detectable.

Identifier commitments $C(x)$ for a mutable identifier x are constructed as follows [42]. To commit to x, the transaction creator first computes $x \cdot G$, where G is the generator of Bitcoin's elliptic curve secp256k1. This way, other nodes cannot obtain x from $x \cdot G$ due to the hardness of the discrete logarithm problem for elliptic curves [31]. The user then makes the commitment self-verifying by interpreting x as a private key and $x \cdot G$ as the corresponding public key, which allows her to create an ECDSA signature of $x \cdot G$ that can be verified by itself. Publishing both $x \cdot G$ and the signature thus allows other nodes to verify the correctness of the commitment's signature with the commitment itself. Thereby, nodes can detect any attempts to manipulate either the commitment or the signature for inserting content there. The final identifier commitment released on the blockchain thus looks as follows:

$$C(x) \quad := \quad (x \cdot G,\ r,\ \mathrm{sig}(x \cdot G \| r, x))$$

Besides the commitment and signature, this value includes an additional salt r to mitigate rainbow attacks. The salt is derived from the previous transactions referenced by the new transaction's inputs. The salt thus satisfies a freshness property that prevents content inserters from precomputing and reusing brute-forced identifier commitments that hold content.

Fig. 9 illustrates how identifier commitments can be deployed to Bitcoin with only slight adaptions. In this example, we consider *pay to public key commitment (P2PKC)* output scripts as a hardened replacement for P2PKH output scripts. Instead of the manipulable public key hash x, P2PKC output scripts contain the commitment $x \cdot G$, the salt r, and the signature $\mathrm{sig}(x \cdot G || r, x)$. Notably, the salt and signature are only required for validating the integrity of $x \cdot G$, but spendability only depends on $x \cdot G$. Hence, r and $\mathrm{sig}(x \cdot G || r, x)$ are omitted during the script validation via the `OP_DROP` operation. However, P2PKC output scripts must still only be unlocked by showing possession of the right private key. The P2PKC output script uses a new `OP_COMMIT` operation to recalculate the commitment $x \cdot G$ from the submitted input script. After this adaption, the validation process operates like with P2PKH output scripts. The

construction additionally replaces the OP_HASH160 operation of P2PKH output scripts with OP_HASH256 to utilize the order of secp256k1 better. As a consequence, $C(x)$ is robust against simple manipulation, as other nodes can easily detect alterations using the commitment's signature. Furthermore, the manipulable identifier x is only disclosed on-chain if an input script successfully spends the P2PKC output, i.e., x corresponds to a valid private key known by a user.

This construction illustrates that there are additional tools to mitigate blockchain content insertion. However, identifier commitments come at the cost of larger transaction outputs [42]. This overhead further burdens Bitcoin's blockchain, which already suffers from a size of 284 GiB as of October 14, 2020 [18]. In Sect. 6.2, we will further investigate how block pruning can vastly reduce blockchain sizes to ultimately regain capacities for deploying countermeasures against the insertion of objectionable content. First, however, we discuss how alternative blockchain designs can inherently mitigate content insertion through pruning-by-design and through forfeiting the script-based transactions of Bitcoin.

5.4 Alternative Blockchain Models

The main entry point for unwanted blockchain content in Bitcoin is its reliance on easily manipulable script-based transactions. The problem is aggravated by the fact that other nodes have to accept transaction outputs without any assertion that the output can be spent in the future. In this section, we consider the mini blockchain scheme [20] to discuss how alternatives to this script-based design could mitigate unwanted content insertion. However, this section only constitutes a high-level discussion as this design has not yet been thoroughly investigated in this regard.

The main goal of the mini blockchain scheme [20] is to create a blockchain that succinctly represents its current state without keeping track of obsoleted information. The mini blockchain scheme achieves this goal by forfeiting Bitcoin's UTXO set and instead maintains an *account tree*, i.e., the state consists only of the users' balances. Instead of storing a list of accounts, the account tree is realized as a radix trie [52], where the nodes are an account's balance, and the account's address determines the path within the radix trie required to look up the balance. Furthermore, the mini blockchain scheme removes Bitcoin's scripts and only allows for basic operations to manipulate the account tree within a transaction instead. Hence, this alternate model allows for a succinct and account-based state representation but has only limited means to define spendability conditions for a coin.

The scheme's *mini blockchain* is only a short "tail" of full blocks, which is kept to reach consensus about the longest, i.e., accepted, chain of blocks. The mini blockchain thus holds transactions only for a short time before they are pruned entirely. The scheme bootstraps new nodes from only the genesis block via an additionally maintained *proof chain*, which consists only of the chain of all, even pruned, block headers. Compared to Bitcoin's block headers, the block headers here also contain the account tree's root. This data enables joining nodes to (a) get the

longest proof chain, (b) obtain and apply the account tree corresponding to this chain, and (c) obtain the remaining full blocks to gain confidence in the chain's validity and finish bootstrapping based on the recent account tree.

Revisiting the basic content insertion methods for Bitcoin (cf. Sect. 3.1), two methods are not entirely prevented by the pruning performed by the mini blockchain scheme. First, users can still manipulate identifiers, i.e., send coins to non-existent addresses, and these fake addresses would be persisted as paths in the account tree. However, due to the account tree's structure as a radix trie, a content inserter cannot ensure that content chunks spread over different fake addresses remain linkable. The content inserter can hence insert individual paths into the account tree, but this data structure shows a larger resilience to content insertion than, for instance, Bitcoin's UTXO set. Second, using output values to encode data [65] remains possible at first glance if the content inserter sends the funds to adjacent receiver addresses, e.g., `00..00`, `00..01`, ..., `00..FF`. In that case, there is a simple extraction rule for the content. However, other users can undermine these efforts by sending coins to addresses that are suspected to be used for this kind of content insertion. As the account tree only holds the addresses' total balance, initially inserted content can, later on, be destroyed again.

This discussion illustrates that alternative designs can further improve the resilience of a blockchain against content insertion. However, further investigation of other real-world cryptocurrencies diverging significantly from Bitcoin's initial design, such as Monero [60] or Mimblewimble [59], is required.

6 Removing Blockchain Data Without Consensus Changes

We have seen in Sect. 5.4 that pruning-by-design can increase a blockchain's resilience against unwanted content insertion. We now investigate to which extent these benefits can be carried over to already established blockchain systems. First, we present a way for node operators to *locally remove objectionable content* [28] and discuss consequences for the overall network when full nodes exercise selective erasure. Second, we outline a recently proposed *block-pruning scheme for Bitcoin* [44] and how it can deal especially with blockchain content [47].

6.1 Local Erasure of Unwanted Content

Florian et al. [28] proposed enabling full nodes to locally remove content they deem objectionable using *functionality-preserving local erasure (FPLE)*. The main goal of FPLE is to minimize the impact on the erasing node's capability to validate new transactions, which might reference older transactions the node locally erased. FPLE focuses on cases where personal preferences or differences between jurisdictions cause only subsets of users to erase any content perceived as objectionable [28].

Using FPLE, full nodes keep only references to objectionable blockchain data in a local *erasure database*. Via the erasure database, nodes flag individual transaction outputs for erasure, e.g., when transactions contain both content-holding and spendable outputs. Flagging a subset of outputs X of a transaction T for erasure involves four steps [28]. First, the node creates a partially erased transaction $T' = T \setminus X$ and stores T' in the erasure database. Second, the node executes the erasure either by fully deleting all occurrences of T on its hard disk or by replacing those occurrences with T'. Third, the node only relies on T' when validating new transactions and stops relaying any unconfirmed transactions that depend on X. Fourth, the node accepts any transaction depending on X in good faith if another miner includes it in a block, assuming that other nodes did not erase X and validated the transaction properly.

This last step, however, requires further consideration since nodes that locally erase transaction outputs might lose their ability to independently validate new transactions. In the following, consider a node receiving a transaction T_2 that attempts to spend a locally erased output from a previous transaction T_1, i.e., the node cannot validate T_2 based on T_1'. If T_2 is still pending, the node discards it. However, if a (malicious) miner includes T_2 in a block and sends that block, the node would accept the block in good faith even if T_2 invalidly spends the output of T_1 [28]. Depending on the fraction of nodes that erased the relevant output of T_1, this scenario could either lead to a temporary blockchain fork or a successful theft of coins [28]. If only a minority accepts the block without validating T_2, the majority, which fully validates T_2, will overrule their decision. In this case, the erasing nodes will revert their wrong decision since the honest majority will not confirm the invalid block. However, if a majority of nodes validate the block based on T_1' instead of T_1, then the block will ultimately be accepted, and T_2 can steal the funds.

The authors argue that this risk of coin theft is generally acceptable since it affects (and thus potentially penalizes) the creator of a transaction holding content that gets erased by a majority of the network [28]. Namely, this scenario is most likely if (a) the content is generally condemned or illegal or (b) the nodes follow a request to remove privacy-sensitive data [28]. The authors anticipate that nodes can nevertheless regain *trustless validation capabilities* and mitigate the potential for coin theft by hashing the manipulable identifiers of content-holding transaction outputs via a cryptographic hash function instead of fully erasing them [28]. This additional hash layer can then be locally applied again whenever the node validates a pending transaction that depends on erased outputs, but recovering the content from the rehashed identifiers becomes infeasible. This approach is potentially applicable to the 99 % of Bitcoin's UTXO set that relies on hash-based identifiers [28].

FPLE shows that nodes can locally erase content they find objectionable in a targeted manner. However, FPLE forfeits the node's capability to independently validate all transactions and relies on a majority of nodes *not* to erase the same content. In the next section, we discuss a block-pruning scheme for Bitcoin that allows full nodes to coordinate global pruning phases to reduce the storage dedicated to obsolete information. While the approach mainly focuses on scalability, we highlight how this additional coordination can also benefit the global erasure of blockchain content based on the initial considerations by Florian et al. [28].

6.2 Bitcoin-compatible Block Pruning

In this section, we discuss CoinPrune [44, 47], a block-pruning scheme that is designed to retroactively enable Bitcoin nodes to forget obsolete blockchain data. While CoinPrune's focus is the scalability of established blockchains with frequent and long-term utilization, pruning can also affect a blockchain's non-financial content. First, reducing a blockchain's storage footprint creates capacities for deploying countermeasures that come with an overhead, e.g., the self-verifying identifier commitments we presented in Sect. 5.3. Second, when nodes routinely prune obsolete data, this process may be extended by making further careful decisions about potentially unwanted content to be pruned additionally. Finally, block-pruning schemes must remain aware of higher-level application data. For instance, notary services may rely on `OP_RETURN` transaction outputs [8], which are inherently prunable.

The main goal of CoinPrune is to reduce the blockchain information that is kept by network nodes and distributed to bootstrap joining nodes to a minimum. In theory, joining nodes only need to obtain a *snapshot*, i.e., a deterministic serialization of the current UTXO set (roughly 4 GiB as of October 14, 2020 [18]) instead of a full blockchain copy (roughly 284 GiB [18]), to bootstrap. However, simply relying on a recent snapshot creates trust issues. The joining node has to trust that the snapshot comes from an honest source to prevent inconsistencies with the other nodes' states.

The mini blockchain scheme (cf. Sect. 5.4) faces a similar problem with the initial distribution of the account tree. The scheme solves this problem by referencing the account tree's root in all full blocks and has other nodes reject blocks that reference an invalid account tree root [20]. Bitcoin has no comparable feature. Furthermore, changes to Bitcoin's consensus rules can provoke heated discussions that ultimately can lead to permanent forks [51]. Hence, Bitcoin's block headers cannot easily be extended retrospectively with a commitment to a recent snapshot, similarly to the mini blockchain scheme. However, the coinbase field (cf. Sect. 3.1) already provides a dedicated space in the block where miners can include additional values such as a snapshot commitment. Unfortunately, the *interpretation* of these commitments requires further attention as CoinPrune nodes must not reject blocks based on these commitments when legacy nodes ignore them at the same time.

Recently, *velvet forks* have been proposed to address these kinds of deployability issues and to allow for a *gradual deployment* of new features [37, 73]. Velvet forks introduce new features in a way that legacy nodes are not affected by the rule changes, i.e., (a) updated blocks remain valid to legacy nodes, and (b) upgraded nodes do not discard legacy blocks [73]. CoinPrune uses a velvet fork to gradually deploy its block-pruning scheme by storing snapshot commitments in the coinbase field, i.e., without changing the block structure for legacy nodes. Furthermore, CoinPrune nodes do *not* reject blocks with invalid or missing commitments, but they look for redundant *mutual reaffirmations* of the snapshot commitment. Namely, CoinPrune nodes coordinate the periodic creation of new snapshots in *pulses* (e.g., every 10 000 blocks). During a short *reaffirmation window* following the pulse, CoinPrune miners commit to their most recent snapshot by adding a corresponding cryptographic identifier to the coinbase fields of their blocks. CoinPrune nodes explicitly *tolerate*

invalid information, i.e., invalid snapshot commitments, to avoid diverging from legacy nodes. Joining nodes rather look for the snapshot with the most reaffirmations. Assuming that enough honest Bitcoin nodes support CoinPrune, the pulse's true snapshot will accumulate reaffirmations the fastest [47]. This way, CoinPrune enables Bitcoin users to seize the efficient bootstrapping of the mini blockchain scheme. Namely, joining nodes now only have to obtain the chain of all block headers, a recent snapshot, and the full blocks following the snapshot to validate the snapshot's reaffirmations and to finish synchronizing beyond the snapshot's state. Thereby, all CoinPrune nodes can completely prune the historic blocks preceding the snapshot without hurting the network, and they can reduce their storage footprint by 303 GiB for Bitcoin's blockchain up until a block height of 640 000 (June 17, 2020), which reduces initial synchronization times from seven hours to below one hour [47].

Finally, CoinPrune also directly deals with non-financial blockchain content in two regards. First, the scheme's pulses provide an additional synchronization point where CoinPrune nodes derive a snapshot from the current UTXO set. Hence, additional considerations beyond the mere serialization become possible. To this end, CoinPrune combines the double-hashing approach proposed by Florian et al. [28] (cf. Sect. 6.1) to *obfuscate* objectionable transaction outputs while remaining capable of validating new transactions spending obfuscated transaction outputs. The nodes only store obfuscated UTXOs and locally rewrite the UTXOs' output scripts during the validation, similarly to the P2PKC output scripts we discussed in Sect. 5.3. Specifically, CoinPrune provides the option to locally obfuscate almost all entries of the UTXO set regardless of whether or not the entries contain any content and allows distributing snapshots of obfuscated UTXO sets while retaining full transaction verifiability for all joining nodes [47]. Second, the scheme acknowledges the benefits of storing small data chunks via `OP_RETURN` transaction outputs for providing additional services on top of Bitcoin's blockchain. In a purely UTXO-oriented pruning scheme, all `OP_RETURN` transaction outputs would be pruned and, thereby, break such services. Hence, CoinPrune additionally maintains an *application data storage* that contains all `OP_RETURN` payloads in conjunction with meta-information about which blocks or transactions they were part of, which also allows pruning nodes to up look this data.

In conclusion, pruning blockchains promises vast savings regarding storage requirements and improved synchronization processes. Block-pruning schemes should further remain aware of blockchain content insertion to enable extended mitigation strategies and to respect higher-level applications. Finally, the storage savings can enable further changes that are currently prohibited by the blockchain's growth.

7 Retrospective Removal of Unwanted Blockchain Content

Up until now, we discussed smaller tweaks to existing blockchains that aim to mitigate the impact of inserting objectionable content. However, none of the presented approaches globally challenge the immutability property of a blockchain, i.e., once objectionable content enters the blockchain it resides there indefinitely. In this sec-

tion, we present alternative blockchain designs that center around the retrospective removal of such content. These *redactable blockchains* extend traditional blockchains with a moderation capability to alter or remove past blockchain transactions. After presenting the initial proposal to enable blockchain redactions by a single moderator or a fixed moderator group, we discuss implications for the permissionless setting, and we briefly outline GDPR-centered blockchain redactions.

7.1 A Cryptographically Redactable Blockchain

In 2017, Ateniese et al. introduced the concept of *redactable blockchains* [7], i.e., an alternative blockchain design that allows redacting transactions retrospectively. In traditional blockchain systems, altering the blockchain after the fact is made impractical through interlinking the blocks and the difficulty to (re)establish these links. For instance, an adversary had to create a proof of work for the altered block and all its successors in Bitcoin. Realizing a redactable blockchain requires weakening this property to carry out the redactions.

To this end, Ateniese et al. [7] replace the traditional cryptographic hash function used to chain blocks together with a *chameleon hash function (CHF)* [6, 7, 38]. CHFs behave as traditional cryptographic hash functions but are parametrized with an asymmetric cryptographic key pair. Everyone can compute a valid hash value using the public key, but knowledge of the private key enables a party to *efficiently compute a collision* for this hash value later on. In the context of blockchains, this means that a party in possession of the CHF's private key can replace a block anywhere in the blockchain without having to recompute all subsequent blocks. This approach allows establishing a blockchain *moderator*. By additionally relying on secure multiparty computation [13, 63], the private key can further be distributed among a fixed *moderator group*. Whenever objectionable content is brought to the attention of this moderator or the moderator group, the block can be updated to delete the objectionable content, and it can then be distributed to other network nodes.

Before we give a design overview of the redactable blockchain proposed by Ateniese et al. [7], we first provide more background on how CHFs are defined in the context of redactable blockchains. Given a group $\mathbb{G}$ for which the discrete logarithm problem is hard, and generator g, the blockchain moderator creates a key pair (k, g^k) and shares g^k with the miners. Miners use a *hashing algorithm* $\mathcal{H}(\cdot)$ to compute the redactable hash value $\mathcal{H}(M, g^k) = (h, \xi)$ for a given message (or block) M. Here, h is the message's hash value and ξ is a random *check value*. Anybody can now use a *verification algorithm* $\mathcal{V}(M, (h, \xi), g^k)$ to ascertain the correctness of h. Knowing the secret trapdoor key k, the blockchain moderator can furthermore use a *collision-finding algorithm* $\mathcal{C}(M', (M, h, \xi), k) = \xi'$ to obtain a new check value ξ' such that $\mathcal{V}(M', (h, \xi'), g^k)$ still yields that h is correct.

We show how CHFs integrate with the blockchain based on the following notation [29]. A block within a blockchain BC is a tuple $B = (s, x, ctr)$, where $s \in \{0,1\}^\kappa$ is the reference to the block's predecessor, $x \in \{0,1\}^*$ is the payload, i.e., the set of

transactions in the block, and $ctr \in \mathbb{N}$ is the nonce that is gradually increased during the mining process and, for simplicity reasons, is assumed to always start at $ctr = 1$. Further, let the tip of BC, i.e., the most recently mined block, be $\text{Head}(BC)$ with identifier s'. The authors further consider the current difficulty level $D \in \mathbb{N}$ and a maximum number $q \in \mathbb{N}$ of allowed tries per block. The model then distinguishes an *inner hash function* $G(\cdot)$ and an *outer hash function* $H(\cdot)$ and defines a newly mined block to be valid if the following conditions hold:

$$H(ctr, G(s,x)) = s' \wedge H(ctr, G(s,x)) < D) \wedge (ctr < q) \stackrel{!}{=} 1$$

Note that the outer hash function only covers ctr and the inner hash value $G(s,x)$. Hence, by replacing $G(\cdot)$ with a CHF, the blockchain moderator can exchange x with a redacted set of transactions x' and compute a collision for the inner hash function without altering the outer hash value. This way, the blockchain remains intact despite the redaction. The block model is extended to account for the CHF, i.e., a block is now a tuple $B = (s, x, ctr, (h, \xi))$, where $\mathcal{H}((s,x), g^k) = (h, \xi)$ is the initial inner hash value. Additionally, the block validity check for a redactable blockchain has the following adapted conditions:

$$(H(ctr, h) = s') \wedge (H(ctr, h) < D) \wedge \mathcal{V}((s,x), (h,\xi), g^k) \wedge (ctr < q) \stackrel{!}{=} 1$$

In summary, now the inner hash value (h, ξ) must be a valid chameleon hash value and h must yield a valid block in the non-redactable model. If the blockchain moderator now computes a collision $C((s,x'), ((s,x), h, \xi), k) = \xi'$ based on a redaction x', then only the CHF verification step for other nodes changes to $\mathcal{V}((s,x'), (h,\xi'), g^k)$, but all other conditions, especially the block interlinking, remain the same.

The redactable blockchains due to Ateniese et al. [7] showcase how novel blockchain systems can be extended to allow for moderation and the removal of objectionable content. However, single blockchain moderators conflict with the decentralized approach of permissionless blockchains, such as Bitcoin. The authors address this problem by presenting a decentralized redaction protocol based on secure multiparty computation, in which multiple blockchain moderators have to cooperate to redact a transaction. While providing a step in the right direction, the scalability of this distributed approach remains questionable due to the performance limitations of secure multiparty computation [74]. Furthermore, blockchain moderators can arbitrarily alter blocks, which means that they are additionally responsible for manually keeping the transaction graph intact, i.e., never redact spent transaction outputs. Finally, CHF-based redactable blockchains do not allow for joining nodes to learn whether a block has been redacted in the past since they remain oblivious to whether they are served the initial check value ξ or a check value ξ' resulting from computing a collision. Next, we thus consider an alternative approach to redactable blockchains that is tailored to the permissionless setting.

7.2 Voting-based Redactions in the Permissionless Setting

Deuber et al. propose a *voting-based* redaction scheme to prepare redactable blockchains for the permissionless setting where all nodes are considered equal [25]. In the following, we give an overview of how Deuber et al. realize their voting-based redaction scheme based on transparently reaching consensus about accepted redactions instead of relying on dedicated blockchain moderators.

In contrast to the moderator-driven approach by Ateniese et al. [7] (cf. Sect. 7.1), blockchain users *report* transactions they deem to contain objectionable content to the miners, who then engage in a voting period to determine whether to accept or reject the user's report. Users trigger the voting by proposing a *candidate block*. The candidate block is a valid replacement block that redacts content from an already mined block according to a predefined *blockchain policy*. As proposed by the authors, the voting is performed as a simple-majority vote over a period of 1024 blocks, i.e., roughly one week of real-world time in the case of Bitcoin [25]. Hence, miners can vote to accept proposed transactions during the voting period. The corresponding candidate block gets accepted by all nodes if at least 50 % of the miners' votes will have approved it by the end of the voting period.

The scheme does not enact the new block by relying on cryptographically replacing the block within the blockchain as the moderated redactable blockchain would (cf. Sect. 7.1). Instead, the checks performed when validating the blockchain are extended to account for potential redactions. To this end, the scheme again extends the block structure, i.e., a block now consists of $B = (s, x, ctr, y)$, where $y = G(s, x) \in \{0, 1\}^\kappa$ is the inner hash value of the block in its initial state. The backlink s' of a block's successor is then computed analogously to traditional blockchain systems in conjunction with the initial inner hash value, i.e., $s' = H(ctr, G(s, x), y)$. If the block is now redacted, i.e., replaced with the candidate block $B^* = (s, x^*, ctr, y)$, the backlink $s' \stackrel{!}{=} H(ctr, G(s, x), y)$ breaks with high probability as $G(s, x^*) \neq G(s, x)$. However, as a fallback, a validating node can substitute $G(s, x^*)$ with the initial inner hash value and check $s' \stackrel{?}{=} H(ctr, y, y)$ instead. If B^* is a valid candidate block and this fallback check succeeds, then the validating node knows that B^* is a redacted version of a formerly valid block since the block was part of the blockchain in its initial state. By extending the initial inner hash value y to a vector y^* of former inner hash values, this scheme also allows for tracking accepted redactions over time in case multiple transactions are redacted as a result of different voting processes. Furthermore, and in contrast to the moderator-based approach, Deuber et al. [25] define explicit redaction policies to prevent constructing inconsistent states via an accepted redaction. These policies enforce that (a) candidate blocks can only remove data from a transaction, (b) the removed outputs must be provably unspendable (e.g., `OP_RETURN` transaction outputs), and (c) no protocol-specific messages, such as other miners' votes, are redacted [25].

This consensus-based approach to redactable blockchains increases the overall transparency and does not rely on especially trusted parties, which makes it a promising approach for the targeted permissionless setting. However, the required

additional considerations trade off the desired trustlessness with increased delays and a reduced scope for possible redactions. On the one hand, the voting period introduces an inherent delay. The proposed voting period of 1024 blocks would, for instance, incur a delay of one week when applied to Bitcoin. During this period, identified illegal content would still be distributed within the blockchain network. On the other hand, the policy for valid redactions is rather conservative to avoid any inconsistencies or ripple effects caused by accepted redactions. As a consequence of this design decision, only provably unspendable transaction outputs can be redacted. However, as we discussed in Sect. 4.2, content is actively being inserted using identifier manipulation. We further established in Sect. 5.1 that nodes cannot reliably determine that a transaction output is indeed manipulated and becomes unspendable as a result. Hence, further improving the reaction times and efficiency of redactions in the permissionless setting is required.

7.3 Redactions for Enhanced Privacy

Up until now, we considered blockchain redactability from a global perspective: the blockchain was rewritten because either a blockchain moderator deemed the edit necessary to protect the network from objectionable content or the network nodes reached consensus among themselves to execute such edits. However, in cases where personal data is also involved, the individual user may request their data to be removed after it was included on the blockchain, e.g., in the context of the GDPR's "right to be forgotten" (cf. Sect. 2.2). In both previously presented approaches, a user would have to file an editing request with the blockchain moderators or the network nodes to exercise this right. We now present another approach [24] that gives users more direct control over their personal data on the blockchain.

Derler et al. [24] achieve this goal by revisiting chameleon hash functions (CHFs, cf. Sect. 7.1) and extending them with *ciphertext-policy attribute-based encryption (CP-ABE)* [15, 30]. Using CP-ABE, users can decrypt an encrypted message based on policies specified during the encryption step instead of having to get access to a single decryption key [24]. Each user is assigned a set of *attributes* where each attribute corresponds to a secret key. The message creator specifies the *access policy* as a Boolean formula over the available attributes obliviously of the users' assigned attributes [24]. In the context of CHF-based redactable blockchains, the gain promised by integrating CP-ABE is that *users can efficiently redact individual transactions* based on an access policy specified by the transaction creator.

However, this technical integration bears further challenges. For instance, traditional CHFs are fixed, i.e., the same key pair is used for all hash values and their collisions. Each policy requires a different secret key for computing collisions to enforce a fine-granular per-transaction access control [24]. The authors realize this property by integrating CP-ABE not with CHFs directly but with special *chameleon-hashes with ephemeral trapdoors (CHETs)* [21], which require both a long-term secret key and a per-hash-value secret key for computing a collision [24]. Based on CP-ABE

and CHETs, the authors then construct *policy-based chameleon-hashes* to be able to implement the possibility of rewriting the blockchain similarly to Ateniese et al. [7] (cf. Sect. 7.1), but now on a per-transaction level and according to fine-granular access policies. For efficiency reasons, the scheme expresses *monotone access policies* using monotone span programs [36], which allow positively specifying groups of users that are eligible to redact a transaction based on their attributes and the `and` and `or` operators [24].

This construction allows for fine-grained control over who can rewrite which transaction in a blockchain system. A common use case for such access policies is a mutual agreement between cooperating partners, e.g., when a customer uses a blockchain-backed service. If the customer, later on, revokes the right to process personal data, e.g., by exercising the GDPR's "right to be forgotten," and that data was previously stored on-chain, then the customer can now enact this desire and rewrite the transaction holding the personal data. However, this scenario is orthogonal to the setting we established in Sect. 2, namely that individual users may, and have [11, 49], deliberately added objectionable or even illegal content to a blockchain with the intent of jeopardizing the system. For such cases, the malicious user must not be allowed to implement a policy that prevents any moderation by other parties. Thus, proper fallback mechanisms are required to hinder objectionable content insertion in blockchains that rely on policy-based redactability.

8 Discussion and Future Research Directions

In this chapter, we discussed blockchain content insertion and presented recent countermeasures to mitigate its negative consequences. However, while the presented approaches tackle blockchain content insertion from different angles, our overview also identifies research questions that remain open. In this section, we thus summarize our findings and highlight further issues that future research should consider to sensibly counteract blockchain content insertion. Corresponding approaches should prevent the insertion of unwanted, objectionable, or even illegal content to the best extent possible while keeping other factors, such as the approaches' efficiency, the trustlessness of permissionless blockchain systems, and the value added by benign blockchain-backed services in mind.

Deeper Understanding of Implications. In Sect. 2, we provided a general framework for blockchain content distribution and potential legal consequences. Thereafter, in Sects. 3 and 4, we gave an overview of how content can be and has been inserted into blockchains. We focused on permissionless and Bitcoin-like blockchains used to create ledgers for cryptocurrencies. Further research into understanding the implications and the extent of blockchain content insertion is required to establish a more solid foundation for required remedies, ensure that they are effective, and determine legal obligations. A first research open research question is thus: *To which extent are conceptually different blockchain systems prone to unwanted blockchain insertion?* Examples of relevant blockchain systems are privacy-focused cryptocurren-

cies [59, 60], scaling blockchains [20], or smart contract-based blockchain systems, such as Ethereum [70], that allow for storing arbitrary data by design. Furthermore, the methods applied to insert blockchain content require a deeper understanding. For instance, steganographic approaches started to utilize Bitcoin as an oblivious medium for exchanging messages that cannot be tampered with [71]. Hence, another question for future research is: *How can we improve the detection of blockchain content?* The more content researchers can unveil, the better we can understand (all) applications, benign and malicious, of blockchain content insertion. Finally, up until now, we only project past court decisions and views on how general law could be applied in the context of blockchain content insertion and distribution (cf. Sect. 2.2). To conceptualize more sustainable blockchain systems despite the possibility of inserting unwanted and objectionable content, researchers need a dependable legal framework to identify both actionable challenges and potential non-challenges that allow for steering future research efforts. An important question thus remains: *What are the true consequences of illegal and privacy-sensitive blockchain content for honest node operators in permissionless blockchain systems?*

Improve Prevention Mechanisms. The mitigation schemes we presented in Sect. 5 are capable of reducing the amount of inserted unwanted blockchain content. Yet, rejecting transactions early on promises to remain the most efficient means of avoiding negative consequences as the immutability of a blockchain does not have to be weakened by the respective countermeasures. Future endeavors into this area could hence address the following research question: *How can full nodes detect unwanted content more reliably before it enters the blockchain?* Improving the detection rate of unwanted content before it enters a blockchain would reduce the number of cases in which special needs for content removal are required, such as the redactable blockchains we presented in Sect. 7. For instance, tweaking and thoroughly analyzing the deterring effect of mandatory minimum fees (cf. Sect. 5.2) or the costs having to brute-force non-manipulable identifiers (cf. Sect. 5.3) remain open questions as of now. Other works have further questioned the efficiency of these mitigation schemes and advocate for revisiting the applicability of *Notice-and-Take-Down* approaches for blockchains instead [61]. In pursuing an increased resilience of blockchain systems against unwanted blockchain content, there remain further open research questions regarding the incorporation of external services. For instance, designs for non-cryptocurrency blockchain systems now regularly rely on external storage systems to store application-relevant data and only insert pointers to such data into the blockchain [46, 56, 57, 75]. In this regard, the inter-planetary file system [14] was proposed as a promising candidate to facilitate the storage of files that are to be referenced on a blockchain, and first applications started to rely on it [3, 53]. In anticipation of an increasing need for such applications, another research question is thus: *How can different applications be integrated and coexist, e.g., via standardized message formats?* External services can further aid the detection of objectionable content. For instance, Cremona et al. [23] proposed establishing a decentralized lookup table of hash values of illegal content such as child abuse imagery[2], which

[2] A centralized related service is, e.g., Microsoft's PhotoDNA service, URL: `https://www.microsoft.com/en-us/photodna`.

could help to reject such transactions before they enter the blockchain. A general research question thus remains: *How can (decentralized) third-party services improve a blockchain's data quality without violating the trustlessness property?*

Swift and Trustless Redactions. We provided an overview of different approaches to redactable blockchains in Sect. 7, which all had a different objective. Ateniese et al. [7] initially established cryptographic means to redact objectionable content anywhere in the blockchain. Deuber et al. [25] focused on increasing the trustworthiness of redactions in the permissionless setting. Finally, Derler et al. [24] realized fine-grained per-transaction redactability by cryptographically enforcing access policies. All presented approaches have their individual advantages and disadvantages. Per-transaction access policies help to keep the original data owner in control even when using a blockchain-backed service. However, the approach does not deter malicious transaction creators, who would insert objectionable blockchain content to jeopardize the whole system, even though we have experienced this behavior in the past [11, 49]. The initial redactable blockchain, as proposed by Ateniese et al. [7], explicitly enables a dedicated blockchain moderator to remove any objectionable blockchain content to counteract this malicious behavior. The idea of having only one or few permanent blockchain moderators may be sensible for permissioned blockchains controlled by a known set of stakeholders, but it is detrimental to the idea of "trustless" permissionless blockchains where no node has elevated privileges by design. Deuber et al. [25] transfer the idea of redactable blockchains to the permissionless setting, but they ensure the trustworthiness of redactions through a lengthy voting phase. Hence, even if illegal blockchain content has been identified, which will result in an obvious redaction, this content will reside on the blockchain and will be distributed to joining nodes for another week. Hence, an open research question remains: *How to design a redactable blockchain that allows for a swift yet trustworthy response to identified illegal blockchain content?* Another area for future research is the definition of policies for redacting blockchain content. The approach of Ateniese et al. [7] enables the moderators to operate without any oversight. Theoretically, this approach can even create inconsistent states when redactions are not planned carefully (cf. Sect. 7.1). Contrarily, Deuber et al. [25] presented explicit policies for valid redactions. However, these policies are restrictive to avoid inconsistencies (cf. Sect. 7.2), to a point where identifier manipulation cannot be redacted (cf. Sect. 7.2) even though this method is an actively exploited entry point for blockchain files (cf. Sects. 3, 4). Another open problem thus is: *What are adequate redaction policies to counteract all objectionable content without risking the blockchain's integrity?*

Foster Benign Applications. Throughout this chapter, we have concentrated on objectionable blockchain content and its negative consequences. While we argue that protecting blockchain-based systems against such consequences is imperative, novel applications such as digital notary services [8], conflict resolution services [69], bootstrapping services [45], or even higher-level coordination between blockchain nodes [44], all rely on inserting application-specific data into a permissionless blockchain to leverage its trustlessness property. When investigating further remedies that prevent unwanted blockchain content from being inserted or that reduce the blockchain's size, researchers should keep these benign blockchain-backed services

in mind. These services oftentimes rely on the intended insertion methods provided by the coinbase field and `OP_RETURN` transaction outputs (cf. Sect. 3.1). However, there are also arguments for utilizing the unintended insertion methods, such as identifier manipulation. For instance, while this medium can be used to spread illegal content, whistleblowers and politically prosecuted journalists could use it as a means for censorship-resistant publication [43]. Hence, future research efforts could ask: *How can we provide a (moderated) medium for benign blockchain-backed applications services while deterring the spreading of objectionable or even illegal data over the same medium?* Future designs should refrain from using unintended content insertion methods for that purpose but strive to create an additional channel that can properly be moderated by design, in a trustworthy manner, and that integrates well with the underlying blockchain's design principles. While the decision of Money Button to use large `OP_RETURN` fields for content insertion instead of identifier manipulation (cf. Sect. 3.2) already follows the latter advice, the near-immediate abuse of this feature underpins the necessity for enabling a proper moderation or oversight of such channels.

9 Conclusion

In this chapter, we have systematically analyzed how non-financial content enters the blockchains of Bitcoin-based cryptocurrencies, how such content is subsequently distributed through the peer-to-peer network maintaining the blockchain, and what consequences such content distribution can have for blockchain users. Based on this analysis, we discussed recently proposed countermeasures to (a) mitigate the insertion of new blockchain content, (b) deal with excessive blockchain data and unwanted blockchain content after the fact with minimal adaptions to existing systems, and (c) extend blockchain designs to implement moderation processes that allow for retroactively redacting objectionable blockchain content. While these approaches provide promising ideas, we identified further future research directions in this field. These research directions engulf generating a deeper understanding of the implications of blockchain content insertion, improving mechanisms that prevent objectionable content from ever entering the blockchain, researching means to respond swiftly and in a trustworthy manner if unwanted content still enters the blockchain, and fostering benign applications that rely on storing data on permissionless blockchains.

References

[1] (2014) KG Berlin, Beschl. v. 25.08.2014 - 4 Ws 71/14
[2] (2015) Protection of Children Act, Chapter 37, Section 7. URL `http://www.legislation.gov.uk/ukpga/1978/37`

[3] Ali MS, Dolui K, Antonelli F (2017) IoT Data Privacy via Blockchains and IPFS. In: IoT, ACM
[4] Andresen G (2012) Block v2, Height in Coinbase. URL https://github.com/bitcoin/bips/blob/master/bip-0034.mediawiki, accessed 2021-09-11
[5] Andresen G (2012) Pay to Script Hash. URL https://github.com/bitcoin/bips/blob/master/bip-0016.mediawiki, accessed 2021-09-11
[6] Ateniese G, de Medeiros B (2005) On the Key Exposure Problem in Chameleon Hashes. In: SCN
[7] Ateniese G, Magri B, Venturi D, Andrade E (2017) Redactable Blockchain – or – Rewriting History in Bitcoin and Friends. In: IEEE EuroS&P
[8] Bartoletti M, Pompianu L (2017) An analysis of Bitcoin OP_RETURN metadata. In: IFCA BITCOIN Workshop
[9] Bartoletti M, Bellomy B, Pompianu L (2019) A Journey into Bitcoin Metadata. J Grid Computer 17:3–22
[10] BBC News (2018) 'Child porn links could make Bitcoin blockchain illegal'. URL https://www.bbc.com/news/technology-43485572, accessed 2021-09-11
[11] BBC News (2019) Child abuse images hidden in crypto-currency blockchain. URL https://www.bbc.com/news/technology-47130268, accessed 2021-09-11
[12] Bechtolf H, Vogel N (2018) Datenschutz in der Blockchain – eine Frage der Technik: technologische Hürden und konzeptionelle Chancen. ZD p 66 (69)
[13] Ben-Or M, Goldwasser S, Wigderson A (1988) Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In: ACM STOC
[14] Benet J (2014) IPFS - Content Addressed, Versioned, P2P File System (DRAFT 3). White paper
[15] Bethencourt J, Sahai A, Waters B (2007) Ciphertext-Policy Attribute-Based Encryption. In: S&P, IEEE
[16] Bitcoin Project (2010) Script. URL https://en.bitcoin.it/wiki/Script, accessed 2021-09-11
[17] Bitcoin Project (2014) Bitcoin Core version 0.9.0 released – OP_RETURN and data in the block chain. URL https://bitcoin.org/en/release/v0.9.0#opreturn-and-data-in-the-block-chain, accessed 2021-09-11
[18] Blockchaincom (2011) Blockchain Charts. URL https://www.blockchain.com/charts, accessed 2021-09-11
[19] Bron CM (2010) Germany–Judicial and Legislative Developments on Internet Child Pornography. Tech. rep., URL https://merlin.obs.coe.int/newsletter/download/154/pdf/en, accessed 2021-09-11
[20] Bruce JD (2014) The Mini-Blockchain Scheme. White paper
[21] Camenisch J, Derler D, Krenn S, Pöhls HC, Samelin K, Slamanig D (2017) Chameleon-Hashes with Ephemeral Trapdoors. In: Fehr S (ed) PKC, Springer

[22] Chesnokow M (2016) CryptoGraffiti: Permanently Preserve Images on the Blockchain. URL https://news.bitcoin.com/cryptographfiti-images-blockchain, accessed 2021-09-11
[23] Cremona K, Tabone D, De Raffaele C (2019) Cybersecurity and the Blockchain: Preventing the Insertion of Child Pornography Images. In: CyberC, IEEE
[24] Derler D, Samelin K, Slamanig D, Striecks C (2019) Fine-Grained and Controlled Rewriting in Blockchains: Chameleon-Hashing Gone Attribute-Based. In: NDSS
[25] Deuber D, Magri B, Thyagarajan SAK (2019) Redactable Blockchain in the Permissionless Setting. In: IEEE S&P
[26] Donet Donet JA, Pérez-Solà C, Herrera-Joancomartí J (2014) The Bitcoin P2P Network. In: IFCA FC
[27] Erstu E (2014) Cryptograffiti.info. URL http://cryptograffiti.info, accessed 2021-09-11
[28] Florian M, Henningsen S, Beaucamp S, Scheuermann B (2019) Erasing Data from Blockchain Nodes. In: IEEE EuroS&PW
[29] Garay J, Kiayias A, Leonardos N (2015) The Bitcoin Backbone Protocol: Analysis and Applications. In: EUROCRYPT
[30] Goyal V, Pandey O, Sahai A, Waters B (2006) Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data. In: CCS, ACM
[31] Hankerson D, Menezes AJ, Vanstone S (2004) Guide to Elliptic Curve Cryptography. Springer
[32] Hilgendorf E, Valerius B (2012) Computer- und Internetstrafrecht, 2nd edn, Springer. pp. 60 et seq.
[33] HugPuddle (2013) Apertus – Archive data on your favorite blockchains. URL http://apertus.io, accessed 2021-09-11
[34] INTERPOL (2015) INTERPOL cyber research identifies malware threat to virtual currencies. URL https://www.interpol.int/News-and-Events/News/2015/INTERPOL-cyber-research-identifies-malware-threat-to-virtual-currencies, accessed 2021-09-11
[35] Irish Office of the Attorney General (1998) Child Trafficking and Pornography Act, Section 2. Irish Statue Book pp pp. 44–61, URL http://www.irishstatutebook.ie/eli/1998/act/22/enacted/en/pdf
[36] Karchmer M, Wigderson A (1993) On span programs. In: Proceedings of the Eigth Annual Structure in Complexity Theory Conference
[37] Kiayias A, Miller A, Zindros D (2017) Non-interactive proofs of proof-of-work. IACR Cryptology ePrint Archive 2017/963
[38] Krawczyk H, Rabin T (2000) Chameleon Signatures. In: Internet Society NDSS
[39] Le Calvez A (2015) Non-standard P2SH scripts. URL https://medium.com/@alcio/non-standard-p2sh-scripts-508fa6292df5, accessed 2021-09-11
[40] Martini M, Weinzierl Q (2017) Die Blockchain-Technologie und das Recht auf Vergessenwerden – Zum Dilemma zwischen Nicht-Vergessen-Können und Vergessen-Müssen. NVwZ p 1251 (1253)

[41] Matzutt R, Hohlfeld O, Henze M, Rawiel R, Ziegeldorf JH, Wehrle K (2016) POSTER: I Don't Want That Content! On the Risks of Exploiting Bitcoin's Blockchain as a Content Store. In: ACM CCS
[42] Matzutt R, Henze M, Ziegeldorf JH, Hiller J, Wehrle K (2018) Thwarting Unwanted Blockchain Content Insertion. In: IEEE BTA Workshop
[43] Matzutt R, Hiller J, Henze M, Ziegeldorf JH, Müllmann D, Hohlfeld O, Wehrle K (2018) A Quantitative Analysis of the Impact of Arbitrary Blockchain Content on Bitcoin. In: IFCA FC
[44] Matzutt R, Kalde B, Pennekamp J, Drichel A, Henze M, Wehrle K (2020) How to Securely Prune Bitcoin's Blockchain. In: IFIP NETWORKING, IFIP
[45] Matzutt R, Pennekamp J, Buchholz E, Wehrle K (2020) Utilizing Public Blockchains for the Sybil-Resistant Bootstrapping of Distributed Anonymity Services. In: ACM ASIACCS
[46] Matzutt R, Pennekamp J, Wehrle K (2020) A Secure and Practical Decentralized Ecosystem for Shareable Education Material. In: ICOIN, IEEE
[47] Matzutt R, Kalde B, Pennekamp J, Drichel A, Henze M, Wehrle K (2021) CoinPrune: Shrinking Bitcoin's Blockchain Retrospectively. IEEE TNSM 18(3):3064–3078
[48] McReynolds E, Lerner A, Scott W, Roesner F, Kohno T (2015) Cryptographic Currencies from a Tech-Policy Perspective: Policy Issues and Technical Directions. In: IFCA FC
[49] Money Button (2019) Against Illegal Content on the Blockchain. URL https://blog.moneybutton.com/2019/01/31/against-illegal-content-on-the-blockchain, accessed 2021-09-11
[50] Money Button (2019) How We Added Support for Giant OP_RETURN Data in Money Button. URL https://blog.moneybutton.com/2019/01/26/how-we-added-support-for-giant-op_return-data-in-money-button, accessed 2021-09-11
[51] Morgan D (2017) The Great Bitcoin Scaling Debate – A Timeline. URL https://hackernoon.com/the-great-bitcoin-scaling-debate-a-timeline-6108081dbada, accessed 2021-09-11
[52] Morrison DR (1968) Patricia–practical algorithm to retrieve information coded in alphanumeric. J ACM 15(4):514–534
[53] Nizamuddin N, Salah K, Ajmal Azad M, Arshad J, Rehman M (2019) Decentralized document version control using ethereum blockchain and IPFS. Computers & Electrical Engineering 76:183–197
[54] Office of the Law Revision Counsel of the United States House of Representatives (1996) U.S. Code, Title 18, Chapter 110, § 2256
[55] Okupski K (2014) Bitcoin Developer Reference. White paper
[56] Pennekamp J, Alder F, Matzutt R, Mühlberg JT, Piessens F, Wehrle K (2020) Secure End-to-End Sensing in Supply Chains. In: CPS-Sec, IEEE
[57] Pennekamp J, Bader L, Matzutt R, Niemietz P, Trauth D, Henze M, Bergs T, Wehrle K (2020) Private Multi-Hop Accountability for Supply Chains. In: BIoTCPS, IEEE

[58] Peters A (2018) Sabotage von Blockchains durch Einschleusung strafrechtsrelevanter Inhalte? In: Taeger J (ed) Rechtsfragen digitaler Transformationen, 1st edn, OlWIR, pp. 392 et seq.
[59] Poelstra A (2016) Mimblewimble. White paper
[60] van Saberhagen N (2013) CryptoNote v 2.0. White paper
[61] Schellekens M (2019) Does regulation of illegal content need reconsideration in light of blockchains? International Journal of Law and Information Technology 27(3):292–305
[62] Schrey J, Thalhofer T (2017) Rechtliche Aspekte der Blockchain. NJW p 1431 (1434)
[63] Shamir A (1979) How to Share a Secret. CACM 22(11)
[64] Shirriff K (2014) Hidden surprises in the Bitcoin blockchain and how they are stored: Nelson Mandela, Wikileaks, photos, and Python software. URL http://www.righto.com/2014/02/ascii-bernanke-wikileaks-photographs.html, accessed 2021-09-11
[65] Sleiman MD, Lauf AP, Yampolskiy R (2015) Bitcoin Message: Data Insertion on a Proof-of-Work Cryptocurrency System. In: 2015 International Conference on Cyberworlds (CW)
[66] Sward A, Vecna I, Stonedahl F (2018) Data Insertion in Bitcoin's Blockchain. Ledger 3
[67] Taylor G (2004) Concepts of Intention in German Criminal Law. Oxford Journal of Legal Studies 24(1):pp. 99–127
[68] User "unwriter" (2019) B:// – Bitcoin Data Protocol. URL https://github.com/unwriter/B, accessed 2021-09-11
[69] Wagner E, Völker A, Fuhrmann F, Matzutt R, Wehrle K (2019) Dispute Resolution for Smart Contract-based Two Party Protocols. In: ICBC, IEEE
[70] Wood G (2014) Ethereum: A Secure Decentralised Generalised Transaction Ledger. Yellow paper
[71] Xu M, Wu H, Feng G, Zhang X, Ding F (2020) Broadcasting Steganography in the Blockchain. In: Digital Forensics and Watermarking
[72] Yeow A (2018) Bitnodes: Global Bitcoin Nodes Distribution. URL https://bitnodes.earn.com/dashboard/?days=730, accessed 2021-09-11
[73] Zamyatin A, Stifter N, Judmayer A, Schindler P, Weippl E, Knottenbelt WJ (2018) A Wild Velvet Fork Appears! Inclusive Blockchain Protocol Changes in Practice. In: IFCA BITCOIN Workshop
[74] Ziegeldorf JH, Matzutt R, Henze M, Grossmann F, Wehrle K (2018) Secure and Anonymous Decentralized Bitcoin Mixing. FGCS 80
[75] Zyskind G, Nathan O, et al. (2015) Decentralizing Privacy: Using Blockchain to Protect Personal Data. In: S&P Workshops, IEEE