Precursive Inscriptions: A Bitcoin-native Private Publishing Mechanism

someguy
someguy@precursive.tools
www.precursive.tools

4de67a207019fd4d855ef0a188b4519c7040281c9c121fc28574b0bd58bd3927
hash@precursive.tools
www.precursive.tools

Abstract. A specially authored recursive inscription process is employed to encrypt data during its upload, which spans multiple Bitcoin[1] transactions and blocks. This method not only optimizes publishing fees by spreading out the necessary blockspace demand over multiple blocks, but also enables the segmentation of files larger than a single block. Subsequently, decryption is facilitated through an eventual publishing transaction that discloses the private key, decrypting and recursively relating the seemingly unrelated preceding inscriptions. Additionally, this mechanism can be configured with a time-lock feature on the publishing transaction, thereby serving as a safeguard or "dead man's switch".

1. Introduction

With the advent of Bitcoin-native inscriptions[2], the possibility of immutable, durable and censorship resistant publishing has come to fruition. The current iteration of inscription technology allows for users to post their data via a permanent but public Bitcoin transaction. However, this reality has led to yet-to-be confirmed inscription transactions and their associated data being noticed while within the mempool itself. This issue can be mitigated by introducing encryption within the inscription process, leaving encrypted but otherwise innocuous data to be propagated by Bitcoin nodes and eventually published by Bitcoin miners, but with no ability to be censored due to content. This also removes the ability for inscriptions meant for speculation to be frontrun by malicious collectors who pull inscription data from the mempool and rebroadcast it at an increased fee rate in order to be confirmed sooner.

Precursive inscriptions aim to create the private, encrypted publishing of data spread out over multiple Bitcoin blocks that can be published at a whim via a recursive publishing transaction containing the private key to decrypt the previously inscribed data. For instance, a collective of whistleblowers could discreetly upload data to the Bitcoin blockchain, unbeknownst to miners or node runners, while deferring its publication until a preferred moment. This very mechanism could implement a time-locked bitcoin transaction for this publishing transaction, acting as a dead man's switch.

2. Bitcoin-native Inscriptions

Inscriptions represent a method of embedding arbitrary content within Bitcoin transactions, effectively creating bitcoin-native digital artifacts, often referred to as NFTs. Notably, the process of creating inscriptions does not necessitate the use of a sidechain or a separate token. These inscribed satoshis can subsequently be transacted using standard bitcoin transactions, sent to bitcoin addresses, and stored as bitcoin Unspent Transaction Outputs (UTXOs). It is essential to emphasize that these transactions, addresses, and UTXOs adhere to conventional bitcoin standards in all respects. The only divergence lies in the requirement for transactions to manage the order and value of inputs and outputs based on ordinal theory when sending individual satoshis.

The content model for inscriptions adheres to the structure commonly used on the web. Each inscription comprises a content type, often referred to as a MIME type, and the content itself, which is represented as a byte string. This approach allows for the retrieval of inscription content from a web server and the creation of HTML inscriptions that can utilize and remix content from other inscriptions.
Importantly, all inscription content is stored on-chain within taproot script-path spend scripts. Taproot scripts offer substantial flexibility in terms of content and benefit from the witness discount, resulting in cost-effective inscription content storage.

Due to the limitations of taproot script spends, inscriptions are created through a two-phase commit/reveal process. First, in the commit transaction, a taproot output that commits to a script containing the inscription content is generated. Subsequently, in the reveal transaction, the output created in the commit transaction is spent, thereby disclosing the inscription content on-chain.

The serialization of inscription content is achieved using data pushes within unexecuted conditionals, referred to as "envelopes". Envelopes are constructed using an “OP_FALSE OP_IF … OP_ENDIF” structure, enveloping any number of data pushes. Notably, envelopes function as effectively inert operations, preserving the underlying script's semantics, and can be seamlessly integrated with other locking scripts.

3. Recursive/Child-Parent Inscriptions

Each Bitcoin transaction containing an inscription has the capacity to store up to 4MB (4 million weight units) of data within Bitcoin's blockchain, according to the blocksize limit. The advent of permanently stored inscriptions on Bitcoin's blockchain prompted developers to introduce the concept of data retrieval from preexisting inscriptions for utilization in subsequent inscriptions, a concept termed as a “recursive inscription”.

Recursive inscription[3] entails the extraction of data from antecedent inscriptions. Through a cascading succession of data retrievals, it becomes possible for resilient software to function entirely on the blockchain, accommodating data volumes in the order of gigabytes, encompassing diverse applications such as video games, films, and intricate software programs. This efficiency gain results from the ability to leverage previously stored data, obviating the necessity to maintain redundant copies of files.

3.1. Inter-referential Mechanism
Recursive inscriptions, at their core, establish an inter-referential mechanism that allows individual entries within Bitcoin's blockchain to reference and utilize the content and code of other entries. This capability transforms seemingly isolated inscriptions into components that collectively contribute to the construction of a more comprehensive and detailed entity.

3.2. Reduced Data Redundancy
One of the primary benefits of recursive inscriptions is the marked reduction in data redundancy. By allowing inscriptions to share and leverage common code and content, the need for duplicative information is mitigated. As a result, the data requirements for each inscription decrease significantly, leading to a corresponding reduction in the cost associated with inscription.

3.3. Enhanced Detail, Quality, and Resolution
This composite inscription can encompass a multitude of detailed information, facilitating a higher degree of detail, quality, and resolution. The capacity to draw upon multiple inscriptions simultaneously elevates the overall quality of the Ordinal, making it a compelling asset within the blockchain ecosystem.

3.4. On-Chain Reveal Processes
An additional dimension of recursion within inscriptions pertains to on-chain reveal processes. Recursive inscriptions enable the creation of on-chain mechanisms that allow for the exposure and extraction of intricate information stored within the inscription. This feature has implications for various applications, including secure data retrieval and verification.

Several endpoints accessible to inscriptions include:
/blockheight: latest block height.
/blockhash: latest block hash.
/blockhash/<HEIGHT>: block hash at given block height.
/blocktime: UNIX time stamp of latest block.

Recursion additionally facilitates the implementation of on-chain disclosure procedures.

4. Precursive Inscriptions

Building off the concepts presented in recursive inscriptions, precursive inscriptions add encryption during the initial inscription process (child inscriptions), and decryption during the final recursive publishing transaction (parent inscription). In Precursive Inscriptions, the child/parent relationship is inverted. Rather than children pointing to or creating a relationship to a parent, the parent is the final transaction retroactively creating a relationship between it and all of its children.

4.1. Child - Encrypted Data Inscriptions
In order to both save on fees by spreading out the data over multiple inscriptions, as well as to allow for data storage above the blocksize limit, publishers using Bitcoin currently use recursive inscription tooling.

This technique is roughly the same within the proposed precursive inscription model, except it adds the process of encrypting the data before the child inscriptions are propagated.

4.2. Parent - Publishing Transaction

The parent, or publishing transaction, has two main purposes. The first is to recursively tie the child inscriptions together. The second is to decrypt the data previously innocuously inscribed by revealing a private key within its confirmation.

5. Dead Man's Switch Mechanism

A generally useful functionality when it comes to encrypted or publicly withheld datastores is a dead man's switch. The problem with distributing the key to a deadman's switch in the context of Precursive Inscriptions is that all of the data is inherently public on-chain despite being encrypted. If anyone possesses that key but the creator of the inscriptions, then there is the possibility of it leaking earlier than intended. With the final parent inscription being what relates all the ancestors together and reveals the encryption key, this requires the creator to retain the final publishing transaction and personally manage the operation of the deadman's switch, or risk the key leaking prematurely if they outsource this responsibility to an untrustworthy party. There needs to be a mechanism to provide the key material necessary to decrypt the encrypted inscription payloads in this final transaction while guaranteeing it cannot be decrypted immediately.

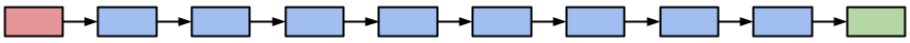

Figure 1. A hashchain, initialized with a random value salted with a recent Bitcoin blockheight (red). The initialization value is hashed a pre-determined number of times (blue), until the last hash value is used to generate a public/private keypair usable for encrypting data.

Timelock encryption[4] is a mechanism that can accomplish this goal. By starting with an initialization vector, in this case a random value concatenated and hashed with a recent Bitcoin blockheader, and

linearly hashing it a very large number of times, the resulting hash from the end of the chain can be used to generate a public/private keypair. This resulting keypair can be used to encrypt the Precursive Inscriptions. Generating such a key linearly would require as much time to generate as it would for any member of the public to regenerate it from the initialization vector. Gwern's timelock encryption proposal offers an intuitively simple solution to this problem.

By generating multiple hashchains with unique initialization vectors in parallel, and stringing them together using the key from the end of one hashchain to encrypt the initialization vector in the next one, a final encryption key can be generated in parallel by the creator much faster than it would take a member of the public to regenerate the key starting from the first initialization vector. The first initialization vector, and the subsequent encrypted ones in the next hashchains, can be included in the final publishing transaction to ensure the decryption key for the Precursive Inscription data is not made available when the transaction is first publicly broadcast. This can also be used as a safeguard when outsourcing the delayed broadcast of the publishing transaction to untrusted third parties. They will not be able to immediately access and prematurely leak the decryption key for the child inscriptions.

6. Implementation

The ideal implementation would be a simple standalone client capable of accepting a file, breaking up the file into chunks and encrypting it, then constructing the appropriate inscription transactions automatically once given funding UTXOs to begin constructing the transaction chains with. This need not be a fully functional Bitcoin wallet, simply a tool capable of constructing the required transactions when provided a PSBT with the funds necessary to do so.

The functionality for generating the required timelock encryption keys could be bundled into the transaction construction tooling, or implemented as a standalone tool.

A final software tool that would benefit the proposal would be a watchtower. This could be a very simple piece of software programmed to broadcast the publishing transaction once a countdown is completed, which could be remotely reset and extended periodically before it expired.

7. Conclusion

The specially authored precursive inscription process presented in this paper offers a novel approach to secure and censorship-resistant data publishing within the Bitcoin blockchain. By leveraging the inherent characteristics of the Bitcoin network, such as its decentralized and immutable nature, the method described here addresses several key challenges in the field of information goods, data inscription, and dissemination. The primary objective of this proposal is to enhance the security and privacy of data stored on the Bitcoin blockchain, while also mitigating the risk of premature disclosure. One of the most significant advantages of this approach is its ability to ensure that the content remains concealed until the user decides to reveal it. This process not only provides data security but also maintains data integrity and permanence within the Bitcoin blockchain.

Furthermore, the time-lock feature of the publishing transaction effectively creates a "dead man's switch". This feature is particularly valuable for scenarios where data must remain confidential until specific conditions are met or in situations where data must be automatically released in the absence of user intervention. The precursive inscription process offers a robust solution to the challenges associated with data publishing on the Bitcoin blockchain, enhancing data security, privacy, and censorship resistance, while also providing users with the flexibility to control when and how their data is revealed.

References

1. Satoshi Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," (October 31, 2008), https://bitcoin.org/bitcoin.pdf.
2. Casey Rodarmor, "Ordinals" (February 2, 2022), https://docs.ordinals.com/inscriptions.html.
3. Casey Rodarmor, "Recursion" (June 12, 2023) https://docs.ordinals.com/inscriptions/recursion.html
4. Gwern, "Time-lock encryption," (May 5, 2011, updated May 6, 2019) https://gwern.net/self-decrypting