White Paper on Recursive Ethscriptions Protocol [REP]

0xanth Author[a] and |3/\53|) _64 Author[b]

[a,b]Affiliation, ViaMaris Labs

ABSTRACT

Ethscriptions have paved the way for a new paradigm on Ethereum, by providing an answer to the cost burden, centralization, and other issues surrounding smart contracts. The Ethscriptions Protocol bypasses smart contract storage and execution through a novel and innovative use of Ethereum transaction call data. The resultingethscriptions are stored directly on the Ethereum blockchain, permissionlessly, and are censorship resistant. As an extension to the Ethscriptions Protocol, Recursive Ethscriptions Protocol [REP], provides additional functionality by utilizing an innovative storage method to store, retrieve, and compose ethscription data.

Keywords: Ethereum, Ethscriptions, Blockchain Storage, On-Chain Assets

1. INTRODUCTION

This white paper will introduce the methodology of using transaction hashes for SVG retrieval and composition, its potential applications, and the underlying technology stack called the Recursive Ethscriptions Protocol [REP].

2. PURPOSE AND FUNCTION

The immutable and transparent properties of Ethereum makes it an attractive canvas for a wide range of applications beyond cryptocurrencies, such as digital collectables, among others.

There is a strong demand for Ethereum's blockspace which is only expected to continue to rise. The network effects and security properties of Ethereum make it a natural choice as a data availability and settlement layer. A rollup centric future and the greater shift, across the blockchain space, towards modular architecture coupled with increasing global adoption are among a few of the many catalysts driving this demand.

The primary purpose of REP is to provide a truly on-chain immutable non-fungible digital asset classification which conserves Ethereum's blockspace, empowers additional creative innovation, and reduces, by several orders of magnitude, the costs of minting, creating, deploying, and transferring high definition or composited digital assets of arbitrary size on the Ethereum blockchain.

By referencing SVG content with a unique transaction hash, users can reliably and securely fetch and compose vector graphics content, ensuring the integrity of the data.

For further information send correspondence to 0xanth and |3/\53|) _64
Web: https://www.viamar.is
GitHub: https://www.github.com/viamarislabs
X: https://x.com/viamarislabs
X: https://x.com/0xanth
X: https://x.com/I3453D_64

3. METHODOLOGY

3.1 Transaction Hash Reference

Each Ethscription transaction generates a unique hash, which can be used as an identifier. By associating SVG* content with a specific transaction hash, we can ensure the content's immutability and traceability.

3.2 SVG Retrieval

Given a transaction hash: The system will parse the API content_uri request of a recursive SVG transaction hash designated by

data:application/vnd.esc.recursion.image/svg+xml+json)

An API call is made to an endpoint to fetch the associated content. Looping through the transaction hashes presented in the recursive SVG.

The returned content_uri content, a base64 encoded format, is decoded to reveal the SVG. The SVG content can be parsed to identify any sub-references (other transaction hashes) pointing to additional content.

The API will then compile this SVG and output it with all the necessary changes as

data:image/svg+xml;

3.3 SVG Composition

Transaction hashes within the SVG content, usually in the form of image references, are extracted. Each extracted hash undergoes a similar retrieval process, replacing the reference with the actual content associated with that hash. The final SVG is a composition of the main content and all the content fetched from the indexer.

3.4 Technical Analysis

The system requires an indexer endpoint, to which the system will initiate a request through the its API, using a transaction hash.

If the response is successful and the content is in JSON format, it parses the JSON. However, If the content_uri data for a transaction hash matches data:application/vnd.esc.recursion, the system will initiate the recursion process.

if content_uri&.start_with?('data:application/vnd.esc.recursion')

The system must also match the mime type.

mime_type_match = content_uri.match(/data:application\/vnd\.esc\.recursion\.(\w+\/[\w+-]+)\+json/)

*SVG is only one implementation among many. While SVG content is used for working examples throughout this paper, it should remain noted that the protocol is extensible to all MIME types.

A valid recursion header with mime type is as follows:

data:application/vnd.esc.recursion.text/html+json

replace_placeholders is defined to replace placeholders in the content with the actual data. There are two valid placeholders.

esc:// and **escr://**

Both fetch corresponding data from the API and replace the placeholder with the content_uri data from the API response.

3.4.1 Usage Examples 'esc://' and 'escr://'

"esc://0x5397a0a88fd97fb257e4e6c6193fa9fad24352866d8d730ccdb8bb122d9ce05d"
returns **""**
in place of **"esc://(transactionhash)"**

While **"escr://0x5397a0a88fd97fb257e4e6c6193fa9fad24352866d8d730ccdb8bb122d9ce05d"**
returns the data in raw format, functional for media in binary format, executable, or audio/video. If the content_uri is present, it also decodes the base64 encoded data for 'escr'placeholders.

3.4.2 Initial Data Handling

If the hash is valid, it fetches the initial data.

Otherwise, if the content_uri starts with a specific pattern indicating recursion data (data:application/vnd.esc.recursion), it processes this data by decoding it from base64, replacing any placeholders with data from the API, and then re-encodes the updated content back to base64 with the correct MIME type.

The system outputs the updated content and its base64 representation with the MIME type. If the content_url does not contain the recursion data, it simply outputs the data.

4. APPLICATIONS

Digital Art Platforms: Digital Art Platforms: Artists can ethscribe layers, as individual ethscriptions, and reference them to generate new recursive ethscriptions. Recursive ethscriptions are themselves an ethscription, which contain all of the composite layers referenced and are committed to Ethereum's blockchain remaining forever immutable and forever on-chain.

REP conserves on the use of Ethereum's blockspace while also reducing the cost for both creators and consumers of ethscriptions. Committing high definition works of art to the Ethereum blockchain, directly by ethscribing, has been prohibitively expensive and impractical at scale, previously.

Table 1. *Total Cost of Ethscription Collection Mint ($) , Collection Size = 10,000 Units , Eth = $1798.00*

Image Size / Gwei	1 KB	10 KB	25 KB	50 KB	100 KB
5	$3,271	$16,527	$38,621	$75,444	$149,090
10	$6,542	$33,054	$77,242	$150,888	$298,180
20	$13,084	$66,109	$154,484	$301,776	$596,361
50	$32,709	$165,272	$386,210	$754,441	$1,490,902
100	$65,418	$330,544	$772,421	$1,508,882	$2,981,803
250	$163,546	$826,361	$1,931,052	$3,772,204	$7,454,508

Table 2. *Total Cost of Recursive Collection Mint ($) , Collection Size = 10,000 Units , Eth = $1798.00*

Image Size / Gwei	1 KB	10 KB	25 KB	50 KB	100 KB
5	$746	$786	$852	$963	$1,184
10	$1,493	$1,572	$1,705	$1,926	$2,367
20	$2,985	$3,144	$3,409	$3,851	$4,735
50	$7,463	$7,860	$8,523	$9,628	$11,837
100	$14,925	$15,721	$17,046	$19,256	$23,675
250	$37,314	$39,302	$42,616	$48,140	$59,187

At an average Gwei rate of 20 and Eth price of $1798, a 10,000 unit collection of 25 KB sized ethscriptions has a total cost burden of 85.91 Eth ($154,484) and requires 250 MB of on-chain storage. If minted with REP the same collection would only require 5.8 MB of total blockspace and the cost burden drops to 1.9 Eth ($3,409).

Table 3. *Total Data Usage (MB) Ethscription Vs Recursive Collection , Collection Size = 10,000 Units*

Image Size / Aggregate	1 KB	10 KB	25 KB	50 KB	100 KB
Ethscription (MB)	10	100	250	500	1,000
Recursive (MB)	5.0	5.3	5.8	6.5	8.0
Storage Reduction (%)	-49.70%	-94.70%	-97.70%	-98.70%	99.20%
Compression Ratio	0.5030	0.0530	0.0230	0.0130	0.0080

Table 4. *Total Cost Ethscription Vs Recursive Collection ($) , Size = 10,000 Units , Gwei = 20 Base Case, Eth = $1798.00*

Image Size / Aggregate	1 KB	10 KB	25 KB	50 KB	100 KB
Ethscription ($)	$13,084	$66,109	$154,484	$301,776	$596,361
Recursive ($)	$2,985	$3,144	$3,409	$3,851	$4,735
Reduction Recursive (X)	5 X	22 X	46 X	79 X	126 X
Reduction Recursive (%)	438%	2103%	4531%	7836%	12595%

Overall, as illustrated in Tables 3-4, utilization of REP to produce recursive collections reduces total costs between 2 times - 126 times cheaper than an ordinary ethscribed collection. REP also compresses on-chain data storage between -49.70% - -99.20%. As collection sizes increase, the compression ratio of total storage and cost reduction percentage both scale exponentially in favor of conservation.

Recursive Ethscriptions Verification: SVGs with recursive content can be verified for authenticity by tracing back to the original transaction hash of the recursively Ethscribed traits or art which make up the final composition.

5. FIRST IMPLEMENTATION [†]

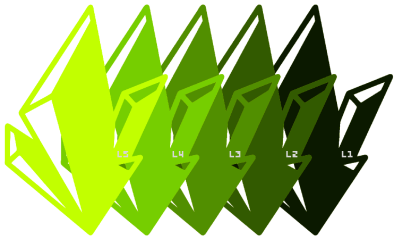
ViaMaris Labs has created and minted the first ever recursive ethscription on Nov-06-2023 09:49:47 AM +UTC. The five 1800X1800 px layers, which compose the recursive ethscription, used a total of 153 KB of block space.

Figure 1 *Individual Recrusive Layers 1-5*

Layer 1: 0x5397a0a88fd97fb257e4e6c6193fa9fad24352866d8d730ccdb8bb122d9ce05d
Layer 2: 0xdaeb10f82914528cad9fa4fe0d09f025b09a971d2474def4de2ba7f4bc9f183a
Layer 3: 0x44c02ab9204038b97717704db6f2e711fc8aaf1aa08c2959cde8ccb5841dd2cb
Layer 4: 0xbe380613c4dff47fb69ab7882a0c98733fd8e1312ff113bb4f2758ca0f0096cf
Layer 5: 0x2199ae0354a36d61820cb42e3b905a098d99d4e0dc9ea0e4292c992b42af9615

The final composed recursive ethscription, depicted below, required 1.1 KB of block space. Five layers could generate 120 unique permutations. Storing such a collection of high definition 1800x1800 px ethscriptions would require 18.38 MB of block space and a cost burden of 8.26 Eth ($14,853); through REP the same collection could be stored with only 0.29 MB of total block space used costing 0.1683 Eth ($302).

Figure 2 *Individual Recrusive Layers 1-5*



Recursive Ethscription: 0x30fdacca03c158a74d75cc8ae24b675828d14e53e07deac1a559f4a1cb828e51

6. CONCLUSION

The Recursive Ethscriptions Protocol extends the frontier of possibilities for entirely on-chain digital assets of arbitrary size, through an innovative storage and retrieval mechanism. Rep enables new classifications of digital assets to be birthed, while upholding the properties of security, traceability, provenance, decentralization, permissionlessness and immutability. Recursive SVG ethscriptions is only one application used as a working example, however, REP's generalization to all MIME types and nesting property creates a far wider scope, than exists today, for creators to explore.

Technological innovation and cost burden has constrained such pursuits currently, REP's purpose is to remove those constraints. Network fee cost burden, required of creators to implement these new sophisticated assets, is drastically reduced by as much as 99.2%. Ethereum's block space is conserved through REP, as data utilization is effectively compressed by 2X, at a minimum, and at over 126X, depending on asset and collection size.

[†]The first on-chain recursive ethscription and sample explorer web implementation are viewable on https://www.viamar.is/recursive future repositories will be stored on https://github.com/viamarislabs