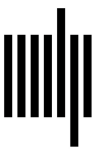
Cryptographic City

Decoding the Smart Metropolis

By: Richard Coyne

Citation:

Cryptographic City: Decoding the Smart Metropolis
By: Richard Coyne
DOI: 10.7551/mitpress/14712.001.0001
ISBN (electronic): 9780262374811
Publisher: The MIT Press
Published: 2023



The MIT Press

10 Hashing the City

A signature is an identifying mark. The cryptographic city, modern and ancient, is populated with signatures. I have mentioned my stone apartment built in the 1830s with the v-shaped mark chiseled into one of the time-worn stone steps leading to the front door. Around the corner is a garage door that the owner repaints periodically to obscure the graffiti that it seems to attract. Half a mile away there are rows of workers' homes that bear crests featuring the tools used by the tradespeople who once lived there. In Scottish towns you occasionally see markings in the stone lintels above doorways that indicate the date of the wedding of the original occupants. The shopping streets nearby are embellished with brand logos. In each case it's as if someone or some organization has left a mark in these places to assert ownership.

We sign off on a home delivery by leaving a mark on a piece of paper or touch pad or tap a button to show we have taken possession of some goods. Signatures authenticate our role in transactions that circulate by digital means and in digital formats in the cryptographic city. Non-fungible tokens (NFTs) as certificates of authenticity (COAs) expand on the concept of a signature to substantiate transactions. Signatures are a means of integrity checking and fall within the purview of cryptography.

A signature is a broad concept. It is common in architecture to speak of signature buildings, which serve either as markers of the company that owns or occupies them, or of the architect, developer, or benefactor responsible for their existence. I want to demonstrate in this chapter that in so far as our cities are permeated by digital transactions, they are imbued with signatures operationalized as hashes, passwords, and encryption keys.

Urban Profiles

In 1997, the Hubble Space Telescope's Space Telescope Imaging Spectrograph (STIS) picked up a signal of a black hole.[1] Instead of the usual vertical straight scan line, the STIS showed an S shape. It is common in astronomy and signal processing to call such a blip in a signal shape a "signature." Such signatures point to the presence of something. But you see the thing only indirectly, that is, you get to know that the referent exists, even though you can't see it simply by looking for it, or at it—even if you could. In this case the invisible black hole was such a referent. Viruses and other microscopic entities produce a genomic signature, or perhaps several, depending on the means of detection and measurement.

A signature can be physical or behavioral. Signatures have a broad range of uses and contexts of use. Some scholars have applied the term *signature* to large-scale human-made entities, such as cities, societies, or political movements. Many businesses, buildings and cities claim the attribution of signature synonymous with *landmark*, *showcase*, and *flagship*. That is to assert a signature object, like a building, as *significant*, as if it stands in for the larger whole, the enterprise, the collection, the style, or a nation. Some scholars even propose that places can have *affective* signatures. For example, an article on "emotional signatures" asserts: "Political ideology thus has a discrete emotional signature, one favoring anxiety among conservatives and anger among liberals."[2] Do cities have signatures? By some readings, "melancholy" is the emotional signature of cities like Porto in Portugal[3] and other cities bear affective signatures delivered in marketing, song, consumer comments on TripAdvisor, and Instagram tags, such as "delirious" New York, "friendly" Dublin, and "chilled" Amsterdam.

Digital technologies have expanded the opportunities to identify urban signatures. For example, scholars have detected the flow of people moving around a city derived from traces left across a map of the city by home delivery or taxi rides, or concentrations of images uploaded to photo-sharing websites.[4] Researchers have used these diagrams as indices of the city's cultural and social structure. Degen and Rose identify swirling linear graphics as a feature in marketing material for the smart city: "Digital data is made visible as geometric patterns of blue glowing light traveling effortlessly between various digital devices in the urban environment."[5]

Terms such as *footprint*, *fingerprint*, and *profile* also come to mind as synonyms for *signature* in an urban context. By this reading there are many

signatures inscribed across the cryptographic city. In what follows I will focus on digital signatures and their role.

Scale and Compress

As indicated in chapter 9, drawings, photographs, image manipulation, and the application of graphic encryption bring us closer to the spatial aspects of the cryptographic city. City planners and developers identify densities and concentrations of populations and built structures, as well as the proportions of open to built space. In these and other respects, I think that aspects of urban space management are analogous to image manipulation. That helps me justify this foray into digital graphics as a facet of the cryptographic city.

If a signature is a surrogate for something larger, then a thumbnail image of a picture provides some of the functionality of a signature. Thumbnail images are simply reduced versions of a much larger file.[6] Such image reduction shows the functionality of displaying something large, like a picture that takes up most of the screen and reducing it to something small and more portable, such as a thumbnail image. Size reduction is among a family of techniques for turning an image into a signature.

Image compression, also known as *down sampling*, provides a similar function while retaining something close to the size and appearance of the original image. In that case compression algorithms down sample a full-color, high-definition image to a smaller file that is close in appearance to the original in terms of size and color range. The usual role of compression is to create a version of an image file that is as visually close to the original as possible but of a smaller file size.

There are several methods of image compression.[7] Lossless data compression produces a smaller file, but no information is lost, and display programs can reconstruct and display the image as if uncompressed. "Lossy" compression, such as JPEG image compression, adopts more cunning methods of file size reduction, such as reducing the palette of pixel color values where there are a lot of adjacent pixels with high contrast. The eye is less sensitive to color variation across a busy part of the picture compared with large areas of subtle color variation. We are more likely to notice dramatic contrasts than we are the subtle differences within regions of high contrast. A row of pixel values in a grayscale photograph will typically show a range of intensities. If you imagine these data points joined together by a smooth

curve then that looks a bit like a fragment of an oscillating frequency curve, similar to a sound wave. There is an algorithm for breaking down complex curves into several regular cosine curves of different frequencies. The curve of the original row of pixels can be approximated as a series of numbers indicating the intensity of the contribution from several such regular cosine curves as sets of coefficients. Saving only the coefficients for the lower-frequency cosine curves reduces the file size. This cosine reduction method breaks the image into 8×8 pixel squares and applies the algorithm to each square in turn.

What I want to show here is that this compression method strips away a substantial part of the image information, thereby reducing the file size. Once it is lost, there is no way to bring that information back from the compressed file. The loss is more noticeable if the person viewing the image wants to zoom into the picture, enlarge it, or apply various filters as in a picture-editing program, or recover LSB steganographic image content as described in chapter 9, which will mostly be obliterated by JPEG compression.

Scaling down the size of an image involves a simpler method than JPEG compression. It simply averages the color value of every square group of pixels (e.g., 8×8) to a single pixel. The averaging process loses information. It is not possible to reconstitute the original information from a set of averages, though it is usually recognizable by a human being as a surrogate for the original image. A thumbnail is such a reduced-size version of an image.

These techniques introduce the idea that you can reduce the size of an image by scaling, image compression, and other manipulations. The process leads eventually to something small and unrecognizable by the human eye as derived from the original referent. But this much-reduced image serves as a digital signature, a pointer to the original, a means of authentication and a file that is more portable than its referent. Importantly, the creation, storage, sorting, manipulation, and matching of this signature can be automated.

Picture Hashing

The kind of signature I have just described serves some of the functions of a *hash*. An image hash is a reduced version of an image file and serves many of the functions of a signature with additional benefits, such as matching,

indexing, lookup, and search. The hash image provides an almost unique identifier for an image in a lookup table or index, and the hash may preserve some of the properties of the source image, such as its average color, and the general distribution of colors across the image. Image indexes can deploy searchable lists of hashed images. Matching hashed images narrows the search process. The use of the hash from the point of view of graphical images is relevant to visual images, maps, remote sensing scans, photographs, screen displays, and other visual representations on screens on smartphones, laptops, and urban displays.

I introduced the hash idea in chapter 4 as it accords with concepts of urban collage, and in chapter 7 as I was explaining the authentication of blocks of transactions on the blockchain.[8] There I was referring to a hash as a sequence of characters of fixed length that is derived from the contents of a file, whether text, image, or other medium. Though blockchains can reference images, they deal ostensibly in strings of alphanumeric characters, of text documents. Another important function of a signature is that of authentication. I referred to the challenges of authenticating crypto art in chapter 9 via certificates.[9] In the rest of this chapter I will elaborate on the hash idea, already introduced in relation to the blockchain in chapter 7.

Hashing on the Blockchain

To hash a file does not encrypt it, as there is no way to reverse the process to recover a file from its hash. Hashing assists in validating data. However, the hash function is one of a range of tools that cryptographic security systems use, not least to protect passwords, verify the integrity of data transfer, and to construct the blockchain.

There are different hash standards, and well-known algorithms that generate hashes from text or other data.[10] As an example, the SHA256 standard algorithm will always convert "abc" to a certain bit string (1s and 0s) 256 binary bits long generally printed as a hexadecimal number of 64 characters:

ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad

The phrase "cryptographic city" converts to the equally long and arbitrary looking

ab4588fce73f79a6ef7bf717872ae010203b8f26e7ba9df71dd14028a09be5f0

To illustrate the generality of the method, here is a section from a non-confidential government document outlining a graffiti strategy for Edinburgh.[11] Though the content is interesting from the point of view of signatures in the city, I am only interested here in this passage as a character string:

> Current procedures and guidelines have been reviewed and best practice identified to ensure that a balanced approach is taken. Robust policy(s) and procedure(s) on Graffiti Management are key components of the future strategy, aiming to reduce instances of "tagging," while still providing space for the more creative elements and potential benefits of graffiti, street art, and murals for local communities. This will also ensure that the city's residents and stakeholders are clear on the approach being taken by the local authority.

Under the SHA256 hashing algorithm that passage translates to the random-looking string

1d6d8d911d3fa0b03af4791c99ea3f3c56f6ecf0d4cc7f201d9cf3783948cce2

The 64-character string output serves as a fixed-size unique condensed representation of the original text about graffiti strategy. Take out the first full stop in the passage and the hash algorithm produced a different string output:

8a5d4771399383e4535bec29432aab2e8d5ac63dafad83844b3fb0b01eaf5136

Neither human nor algorithm can detect any pattern in those strings that indicates the proximity to their original source strings. The conversion is not a method for compressing or encrypting data. There is no algorithm to unpack the 64-character string to recover the original text. The output string is for machine consumption rather than for human beings to read, check, and ponder. The output string is the *hash*, and the SHA256 algorithm is one of the most sophisticated and the least likely to generate "a collision," meaning, where two or more sets of different data have the same hash. The SHA256 algorithm is no secret, and there are many explanations of it in cryptography textbooks and online,[12] and knowing how the routine works to convert a string of text to a hash string provides no advantage to a would-be hacker.

What is the use of this (near) unique hash representation of a block of data? It is used for authentication. As well as transmitting the document,

you transmit the hash—perhaps in a different channel. The receiver software then runs the data through the SHA256 algorithm, and if the receiver-generated hash does not match the hash sent with the data, then that alerts the receiver that the data has been tampered with en route. It won't tell you what was changed, or the magnitude, but it may send an alert, or notify the sender software to transmit the data again.

Here is a graphical example of the use of the hash algorithm. The CryptoPunks auction site sells CryptoPunk NFT files. As introduced in chapter 9 these are stylized 8-bit portrait images each sized 24×24 pixels. The auction site describes how they use hashing so customers can verify that once they have made a purchase they are in possession of an actual CryptoPunk file: "The actual images of the punks are too large to store on the blockchain, so we took a hash of the composite image of all the punks and embedded it into the contract. You can verify that the punks being managed by the Ethereum contract are the True Official Genuine CryptoPunks by calculating an SHA256 hash on the cryptopunks image and comparing it to the hash stored in the contract."[13]

You don't need to purchase a CryptoPunk NFT file to test that the file you have downloaded is exactly the same as in the Ethereum contract. The Ethereum contract cites the hash of the composite graphic file displaying the composite of ten thousand CryptoPunk images as

ac39af4793119ee46bbff351d8cb6b5f23da60222126add4268e261199a2921b

The composite CryptoPunk file is a picture, but it is just data as far as the SHA256 algorithm is concerned, a file of 0s and 1s. Sure enough, the CryptoPunk graphic file from the LarvaLabs website returns the identical hash string when I run it through an independent SHA256 calculator.[14]

A hash can also be used to transmit a password over the Internet. My online banking service has my password in a database saved as a hash. So, if I had a password such as "appLeTr33" and wanted to log on to my digital bank website then the transmission software would convert that password to the hash

80dfcee4ead6ef8cd1ba4edfc782d5c4b4505affc223d1345bcce51c9818fc61

and send the hash through the Internet. The server software receives the hash and looks that up to access my account. If the hash falls into the wrong hands, then there is no way to reverse engineer the hash string to

the original password. So that's one of several measures required to keep passwords safe in transit.

Hashing is also used for indexing whole files via so-called "content addressing," the unique hash as a signature derived from running an entire file through the hashing algorithm. IPFS, the so-called "Interplanetary File System," is a peer-to-peer file storage and sharing platform. The system breaks computer files into segments (256 kB "chunks") and distributes them to different servers on the network. The segments are indexed via their hash so they can be reassembled and accessed via their hash strings, which can in turn be secured on a blockchain.[15]

So, I have introduced the role of the hash in the blockchain and alluded to other processes pertinent to the blockchain. The rest of this chapter provides a deeper dive into blockchain processes for readers who share my enthusiasm for cryptographic methods.

Hash Chains

Table 10.1 shows a page from a spreadsheet of transactions. It is technically a Merkle chain.[16] I have added a column to show the hash of the data in each row, added to the hash of the previous row. You can create a hash of a hash, a process that can be continued indefinitely. If someone changes the content of one of the rows in the ledger then that changes the hash values for the transactions that follow, as shown in the second part of table 10.1 (doctored ledger). With access to the hash algorithm, someone who wanted to change a transaction in the ledger could change the hash value of that transaction and those that follow. But if the ledger has already been shared with others on a network and verified, then the discrepancy will be obvious, and rejected as invalid by auditing algorithms in the network.

In fact, this method of verification via hashing works on whole pages of transactions as well as for individual transactions. One of the best descriptions of this method of verification that I have found comes from a post by blogger Antony Lewis.[17] He explains that a traditional paper ledger consists of pages of transactions. Instead of page numbers, at the top of each page (header) print a hash of the previous page. At the bottom of the page print a hash of that page (including its header). Lewis calls the hash a "unique fingerprint" of the page: the page is a block, the whole set of pages linked in this way is a chain.

Table 10.1
Securing data on a Merkel chain

Original ledger of transactions

Date	Description	Value	Previous hash	Combined record	New hash
12-Jun-22	Accommodation	–$210.00	0000000	12-Jun-22 Accommodation –$210.00 0000000	63896d1c
14-Jun-22	Bike hire	–$67.00	63896d1c	14-Jun-22 Bike hire –$67.00 63896d1c	a36aca28
17-Jun-22	Refund	$50.00	a36aca28	17-Jun-22 Refund $50.00 a36aca28	fe52f149
19-Jun-22	Home delivery	–$35.00	fe52f149	19-Jun-22 Home delivery –$35.00 fe52f149	561dc761
21-Jun-22	Expenses	$250.00	561dc761	21-Jun-22 Expenses $250.00 561dc761	07dc2d3d

Doctored ledger

Date	Description	Value	Previous hash	Combined record	New hash
12-Jun-22	Accommodation	–$210.00	0000000	12-Jun-22 Accommodation –$210.00 0000000	63896d1c
14-Jun-22	Bike hire	–$67.00	63896d1c	14-Jun-22 Bike hire –$67.00 63896d1c	a36aca28
17-Jun-22	Refund	**$100.00**	a36aca28	17-Jun-22 Refund $100.00 a36aca28	**2a619b4a**
19-Jun-22	Home delivery	–$35.00	2a619b4a	19-Jun-22 Home delivery –$35.00 2a619b4a	**2a2e40eb**
21-Jun-22	Expenses	$250.00	2a2e40eb	21-Jun-22 Expenses $250.00 2a2e40eb	**60fe8744**

The first section shows a page from a spreadsheet of transactions with chained hash codes. The doctored section shows a spreadsheet with the "Refund" transaction altered. That change alters the subsequent hash codes in the chain. In a shared ledger system, the change would be rejected. Note that for simple illustration the hash code used here (CRC-32) is much shorter than the SHA250 standard.

Internal consistency is assured if the hash strings or "unique fingerprints" are consistent with the data and they chain together effectively. In order for someone to adjust the ledger they would have to adjust all of the chained hash strings that follow the adjusted transaction. The algorithms of the blockchain would soon detect the inconsistency in the final state of the blockchain and reject the rogue transaction.

The term *immutable* is used commonly to describe this state of a data set.[18] Each page of transactions is embedded in the transactions that follow. So, you end up with a very large unalterable database. As discussed in the context of the blockchain in chapter 7 it is a bit like sedimentary rock formations where the newest strata depend for their integrity on older formations below, which in turn require more effort to penetrate, or alter. As we have seen, this "immutability" is a key feature of distributed digital ledgers as used in cryptocurrencies, where the ledger is duplicated and shared across a large number of users, and there is no centralized keeper of the data or auditing authority such as a bank. Hashing secures the integrity of the blockchain.

Hash Puzzles

To reverse the process of bedding transactions into a blockchain, and to unpack the original string from a hash would be a mammoth computational task, requiring software to generate every combination of characters, of different lengths, and calculate the SHA256 hash of each combination to see if they match the target hash string. Even if this brute force search yields results for hackers, there may be several strings that produce the same hash. So, they cannot be sure they have found the right one. The SHA256 algorithm is designed to make this reverse engineering virtually impossible. Unlike various computational puzzle-solving tasks, the algorithm would be unable to detect that it is getting closer to a solution. It might as well iterate every combination of characters for the input at random.

The proof of work (PoW) verification procedure recruits a simpler hash puzzle: find an input string that generates a hash with certain characteristics. For example, generate a hash string that starts with a zero. I experimented with an arbitrary string of characters "301253" by changing, adding, and deleting characters, fed each string through the SHA256

algorithm, and eventually and randomly came up with "301252" that generates this hash:

0135ce9c1094a7b56219acca34ba9e3bb68bd883cf842577c1e8c107346af173

That's a hash string starting with "0." I solved the puzzle manually by pasting strings into the online SHA256 calculator.[19] But the process could be automated. Imagine this is a contest in an online multiuser game. The first person to "solve" the puzzle of producing a hash string that begins with 0 gets a game reward—some points. Once you guessed an input string that produces a hash string starting with zero, you would send around your input string (301252), and the rest of the players could easily verify that you came up with an answer. They would run the string you distributed through their own copy of the SHA256 algorithm to see if it did indeed generate a hash starting with zero.

In the preceding case the challenge was too easy, computationally. You could set a harder challenge: what input string will generate a hash string that starts "01010," "ababab," "0000," or some other sequence? Aiming for a hash that starts with a row of zeros is as good as any target, as it's easy to verify by eye if needed in a demonstration, and the degree of difficulty will be obvious from the number of zeros required. One leading zero presents a trivial challenge; four zeros automatically increase the degree of difficulty and the number of iterations required considerably. If the challenge appears to be too easy, solved too quickly, then an algorithm shared across the network will increase the degree of difficulty of the challenge by increasing the number of zeros required. It seems that cryptocurrency systems that use this method expect solving the challenge to take about ten minutes of dedicated CPU time, which is a lot of brute force computation. The competing computers are the validating nodes, the miners on the network that supports the blockchain. As faster CPUs enter the network, an algorithm will increase the degree of difficulty to match the capability of the competing validating CPUs. The degree of difficulty for the Bitcoin system is at the time of writing set at a hash with nineteen leading zeros![20]

As someone invested in architecture, design, and gaming, I and many others find it difficult to warm to this extravagant use of processor time in an invisible contest that yields no nuance, color, spectacle, or entertainment. In fact, during lockdown I tried to purchase a PS5 gaming console

without success. Though it has no need for graphic functionality, cryptocurrency mining favors the use of fast graphics processing units (GPU processors). GPUs were in high demand for cryptocurrency mining, resulting in a serious undersupply of gaming consoles.[21]

Proof of Work

What I have described involves securing a page of transactions via a trivial but time-consuming computational puzzle. I have demonstrated how a page of transactions can also be hashed. A page of data in the blockchain includes in its header a hash of the previous page followed by a list of transactions.

To summarize, this puzzle-solving is used to verify and embed a set of transactions. Cryptocurrencies, such as Bitcoin store-linked pages (blocks) of transactions and distribute these to all the validating nodes connected together on a vast peer-to-peer network. Any node on the network, usually a self-appointed subset with adequate computing power, can use the solution to this puzzle to bed down a set of transactions. The node must collect all the transactions coursing through a network since the last page was formed, check that they are consistent and legal, and put them into a virtual ledger page (a *block*) in a standard format. Many of the independent nodes will be doing this at the same time, incentivized by the potential financial reward. So, it is a high stakes contest.

As soon as one of these nodes generates a solution it broadcasts the result to all the other nodes, which quickly verify that the answer is correct. It is easier to test that the solution is correct than to discover it. The block of the winning node then gets added to the set of all approved ledger pages, which is in turn distributed around the network as the approved set of transactions making up the correct and current state of the ledger. That is the blockchain.

As already indicated, the work required to bed down the pages in the ledger (i.e., verify, approve, and seal the blocks in the blockchain) is the proof of work. Were any potential hacker to try and overwrite any transactions or otherwise doctor the ledger, from the most recent page going back many other pages, then they would have to adjust the whole sequence of hashes in all the blocks. According to the seminal article on the technique by Nakomoto, “To modify a past block, an attacker would have to redo

the proof-of-work of the block and all blocks after it and then catch up with and surpass the work of the honest nodes. . . . The probability of a slower attacker catching up diminishes exponentially as subsequent blocks are added."[22]

There are other sophistications to the process, and I've simplified some steps and concepts, but I think the explanation provides at least an insight into the purpose of setting arbitrary puzzles to keep transactions safe from tampering, a necessity where ledgers are open, duplicated, and distributed across networks, breaking the need for a single "trusted" custodian of the database, such as a bank. The deliberate and extravagant "waste" of CPU time is put to a purpose. It is part of the cost of security in a distributed peer-to-peer data system. I mentioned an alternative to PoW in chapter 7, namely proof of stake (PoS).[23] Both systems make use of hash algorithms to embed blocks of transactions in the blockchain, though the PoS draws on other computational methods that require less CPU time in validating transactions.

In this chapter I have demonstrated that the idea of the signature is at home in the smart city, especially in its cryptographic instantiations as hash strings. It also permeates urban digital infrastructures and security and validation processes. To the extent that cryptocurrency is prominent in the cryptographic city, the processes I have described assume significance as city infrastructural elements. They also parallel certain urban processes that run deep in the economy of the city: competition, networks, ledgers, hidden processes, opacity, in the communicative functions of the cryptographic city.

The MIT Press would like to thank the anonymous peer reviewers who provided comments on drafts of this book. The generous work of academic experts is essential for establishing the authority and quality of our publications. We acknowledge with gratitude the contributions of these otherwise uncredited readers.

This book was set in ITC Stone Serif Std and ITC Stone Sans Std by New Best-set Typesetters Ltd.

Library of Congress Cataloging-in-Publication Data

Names: Coyne, Richard, author.
Title: Cryptographic city : decoding the smart metropolis / Richard Coyne.
Description: Cambridge, Massachusetts ; London, England : The MIT Press, [2023] | Includes bibliographical references and index.
Identifiers: LCCN 2022021507 (print) | LCCN 2022021508 (ebook) | ISBN 9780262545679 (paperback) | ISBN 9780262374811 (pdf) | ISBN 9780262374828 (epub)
Subjects: LCSH: Smart cities. | Internet of things. | Urban development—Data processing. | Public administration—Security measures. | Data encryption (Computer science)
Classification: LCC TD159.4 .C69 2023 (print) | LCC TD159.4 (ebook) | DDC 004.67/8—dc23/eng/20221011
LC record available at https://lccn.loc.gov/2022021507
LC ebook record available at https://lccn.loc.gov/2022021508

10 9 8 7 6 5 4 3 2 1