A Data Structure for Integrity Protection with Erasure Capability

D. Richard Kuhn
Computer Security Division
Information Technology Laboratory

May 20, 2022

This publication is available free of charge from:
https://doi.org/10.6028/NIST.CSWP.25

This publication is a final version of a draft cybersecurity white paper originally published on May 31, 2018: https://csrc.nist.gov/publications/detail/white-paper/2018/05/31/data-structure-for-integrity-protection-with-erasurecapability/draft.

National Institute of Standards and Technology
U.S. Department of Commerce

Abstract

This document describes a data structure, referred to as a data block matrix, that supports the ongoing addition of hash-linked records while also allowing for the deletion of arbitrary records, thereby preserving hash-based integrity assurance that other blocks are unchanged. The block matrix data structure may have utility for incorporation into applications requiring integrity protection that currently use permissioned blockchains. This capability could for example be useful in meeting privacy requirements such as the European Union General Data Protection Regulation (GDPR), which requires that organizations make it possible to delete all information related to a particular individual, at that person's request.

Keywords

blockchain; computer security; data structure; distributed ledger; hash; integrity protection.

Patent and Licensing

The data block matrix technology (US Patent #11,175,826) is available to be licensed. For information on patent and licensing see NIST Technology Partnerships Office (TPO). Parties interested in licensing a NIST-owned invention may contact TPO, or may complete a research license application or commercial use license application. Instructions on how to submit the license application are provided at the end of each application. When a license application is received, TPO will contact you regarding the licensing process and to obtain additional information that may be necessary for the application process. A license application should include the prospective licensee's intentions for research and development of the invention(s) and a description of the resources and technical capabilities of the prospective licensee to bring the invention to practical application.

Disclaimer

Additional Information

For additional information on NIST's Cybersecurity programs, projects and publications, visit the Computer Security Resource Center. Information on other efforts at NIST and in the Information Technology Laboratory (ITL) is also available.

Submit comments on this publication to: block-matrix@nist.gov

National Institute of Standards and Technology
Attn: Computer Security Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930

I. BACKGROUND

This document describes a data structure, referred to as a data block matrix, that supports the ongoing addition of hash-linked records while also allowing for the deletion of arbitrary records, thereby preserving hash-based integrity assurance that other blocks are unchanged. (This publication is a final version of [1].)

II. DATA STRUCTURE

Fig. 1 shows a matrix with numbered data blocks, where a block may contain a single record or multiple transactions. Every row or column is terminated with a block containing a hash of that row or column (e.g., $H_{0,-}$ is the hash of row 0). Various forms of the hash structure are also possible. For example, the hash value can be stored in the last block of the row or column instead of a separate hash block. Another alternative is to concatenate hashes of each block in a row or column, similar to the blockchain process. The hash of this concatenation would then serve as the hash value for that row or column.

	0	1	2	3	4	
0						$H_{0,-}$
1						$H_{1,-}$
2						$H_{2,-}$
3			X			$H_{3,-}$
4						$H_{4,-}$
	$H_{-,0}$	$H_{-,1}$	$H_{-,2}$	$H_{-,3}$	$H_{-,4}$	

Figure 1. Basic block matrix

As an example of the process, the block labeled "X" may be deleted by writing all zeroes to that block, or revised with different values. Either of these changes will affect the hash values of $H_{3,-}$ and $H_{-,2}$ for row 3 and column 2. However, the integrity of all blocks except the one containing "X" is still ensured by the other hash values. That is, other blocks of row 3 are included in the hashes for columns 0, 1, 3, and 4. Similarly, other blocks of column 2 are included in the hashes for rows 0, 1, 2, and 4. Thus, the integrity of blocks that have not been deleted is assured. An algorithm to maintain this structure is given below and its properties described.

Within the data structure, blocks are numbered $1..k$, and are added to the data structure starting with cell $0,1$. (It is desirable to keep cells on the diagonal null, for reasons explained later.) Variables i, j are column indices, and swap(i, j) exchanges the values of i and j, i.e., $i' = j$ and $j' = i$. With this algorithm, cells are filled as shown in Algorithm 1.

Algorithm 1 Data block matrix construction
loop invariant: $i < j \wedge odd(B_{i,j}) \vee i > j \wedge even(B_{i,j})$

```
i ← 0
j ← 1
B ← 1
while new blocks do
    if i = j then
        add null block B_{i,j}        // diagonal
        i ← 0
        j ← j + 1
    else if i < j then
        add block B_{i,j}             // upper half
        B ← B + 1
        swap(i, j)
    else if i > j then
        add block B_{i,j}             // lower half
        B ← B + 1
        j ← j + 1
        swap(i, j)
    end if
end while
```

	0	1	2	3	4	
0	•	1	3	7	13	$H_{0,-}$
1	2	•	5	9	15	$H_{1,-}$
2	4	6	•	11	17	$H_{2,-}$
3	8	10	12	•	19	$H_{3,-}$
4	14	16	18	20	•	$H_{4,-}$
	$H_{-,0}$	$H_{-,1}$	$H_{-,2}$	$H_{-,3}$	$H_{-,4}$	etc.

Figure 2. Block matrix with numbered cells

III. PROPERTIES

Certain desirable properties are maintained with this data structure. These features allow for the efficient storage and retrieval of data blocks (results originally introduced in [1]).

Theorem 1 (Balance property)**.** *Cells are filled in a balanced manner so that the upper half (above diagonal) contains at most one additional cell more than the lower half.*

Proof. The following invariant implies the property, and for each iteration of the loop, the invariant is maintained,

$$(i = j \vee i < j) \wedge u = l \vee i > j \wedge u = l + 1$$

where u = number of cells above diagonal, and l = number of cells below diagonal. Initially, $i = j = 0$ and $u = l = 0$. If $i = j$, then u, l are unchanged, so the invariant remains true. If $i < j$, then $u = l$. A block is then added to the upper half and $u' = l + 1$

and $i' = j, j' = i$ so that $i' > j'$ and the invariant is maintained. If $i > j$, then $u = l+1$ and a block is added to the lower half such that $u' = l'$ and $i' = j+1, j' = i$, so that $i' < j'$, maintaining the invariant. □

Theorem 2 (Hash chain length)**.** *The number of blocks in a row or column hash chain is proportional to N for a matrix with N blocks.*

Proof. The balance property ensures filling in a square form. □

Theorem 3 (Block numbering)**.** *All even-numbered blocks are placed below the diagonal and all odd-numbered blocks are placed above the diagonal.*

Proof. The following invariant implies the property.

$i < j \wedge odd(B_{i,j}) \vee i > j \wedge even(B_{i,j})$

Initially, $i < j$ and $B = 1$, so the invariant holds, and for each iteration of the loop, the invariant is maintained. □

Theorem 4 (Block dispersal)**.** *No consecutive blocks appear in the same row or column. That is, for any two blocks numbered a, b, where $b = a+1$, in rows i_a and i_b, and columns j_a and j_b respectively, $i_a \neq i_b$ and $j_a \neq j_b$.*

Proof. This can be shown by considering cases below.

1. If $i < j$, then block a will be written to cell (i_a, j_a) and then i and j swapped, so that in the next iteration, $i > j$, and block b written to cell (i_b, j_b). Since $i_b = j_a$ and $j_b = i_a$, and $i \neq j$, $i_a \neq i_b$ and $j_a \neq j_b$.

2. If $i > j$, then block a will be written to cell (i_a, j_a), j incremented, and then i and j swapped. Then either the relationship is unchanged, with $i > j$, or $i = j$.

2(a). If $i = j$, then no data block will be written in the next iteration, but i will be set to 0 and j will be incremented such that $i < j$, and the next data block written with $i_b = 0$ and $j_b = j_a+1$, ensuring that $i_a \neq i_b$ and $j_a \neq j_b$.

2(b). if $i > j$, then on the next iteration, block b will be written with $i_b = j_a$ and $j_b = i_a$, and $i \neq j$, so that $i_a \neq i_b$ and $j_a \neq j_b$.

□

IV. Block Location

With the relations above, one can derive expressions to locate a given block within the matrix. The following relations are clear (see Fig. 2 for example) where N = number of columns = number of rows; row and column indexes range from 0 to $N-1$.

- The total number of data blocks in the matrix is $N^2 - N$ since the diagonal is null. Thus, the last numbered block in a filled matrix of N rows and columns is number $N^2 - N$.
- Lower half: Numbered from 0, with no data blocks on the diagonal, the total of blocks for i rows in the lower half (below diagonal) is $(i+1)^2 - (i+1) = i^2 + i$, with the last data block in the lower half numbered as $i^2 + i$, and because the diagonal is empty, $j = i - 1$. The first data block in row i is $i^2 - i + 2$, with $j = 0$.
- Upper half: the last upper half data block in column j is $j^2 + j - 1$, with $i = j - 1$.

For a block B in the lower half (B is even), i, j indices can be computed as:

$$i = \lfloor\sqrt{B}\rfloor + [B > \lfloor\sqrt{B}\rfloor(\lfloor\sqrt{B}\rfloor + 1)]$$
$$j = (B - (i^2 - i + 2))/2$$

and for block B in the upper half (B is odd), i, j indices can be computed as: :

$$j = \lfloor\sqrt{(B+1)}\rfloor + [B \geq \lfloor\sqrt{(B+1)}\rfloor(\lfloor\sqrt{(B+1)}\rfloor + 1)]$$
$$i = (B - (j^2 - j + 1))/2$$

Note: $[\epsilon]$ is the Iverson bracket where $[\epsilon] = 1$ if expression ϵ evaluates to $true$, and 0 otherwise.

Blocks can now be deleted by overwriting with zeroes, with one row and one column hash recalculated. Specifically, after deleting block i, j, row i and column j hash values are recalculated. The block matrix data structure may have utility for incorporation into applications requiring integrity protection that currently use permissioned blockchains. This capability could for example be useful in meeting privacy requirements such as the European Union General Data Protection Regulation (GDPR), which requires that organizations make it possible to delete all information related to a particular individual, at that person's request. This requirement may be incompatible with current blockchain data structures, including permissioned blockchains [2] [3] [4], because blockchains are designed to ensure that block contents are immutable. Any change in a blockchain will invalidate subsequent hashes in following blocks, losing integrity protection. The data block matrix structure retains integrity protection of non-deleted blocks. Note that this data structure could also be extended beyond two dimensions to an arbitrary number of dimensions, with extensions to the algorithms above.

Acknowledgments

Many thanks to Lee Badger, Jeff Voas, Sandy Ressler, Dylan Yaga, and Peter Mell for review and discussion of the original paper on this work. Thanks to Peter for suggesting the alternative of hashing a concatenation of block hashes.

References

[1] D Richard Kuhn. A data structure for integrity protection with erasure capability. *NIST Cybersecurity Whitepaper*, 2018.

[2] Matthias Berberich and Malgorzata Steiner. Blockchain technology and the gdpr-how to reconcile privacy and distributed ledgers. *Eur. Data Prot. L. Rev.*, 2:422, 2016.

[3] Henry Chang. Blockchain: Disrupting data protection? *Privacy Law and Business International Report, November*, 2017.

[4] O. Kharif. Is your blockchain doomed? *Bloomberg Business Week, Mar. 22*, 2018.