A Data Structure for Integrity Protection with Erasure Capability

D. Richard Kuhn
Computer Security Division
Information Technology Laboratory

May 31, 2018

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

Abstract

This note describes a data structure, which can be referred to as a block matrix, that supports the ongoing addition of hash-linked records while also allowing the deletion of arbitrary records, 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

cryptographic hash; data structure; distributed ledger; integrity protection

Disclaimer

Any mention of commercial products or reference to commercial organizations is for information only; it does not imply recommendation or endorsement by NIST, nor does it imply that the products mentioned are necessarily the best available for the purpose.

Additional Information

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

Public Comment Period: *May 31, 2018 through August 3, 2018*

National Institute of Standards and Technology
Attn: Computer Security Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930
Email: block-matrix@nist.gov

All comments are subject to release under the Freedom of Information Act (FOIA).

1 Background

This note describes a data structure, which can be referred to as a block matrix, that supports the ongoing addition of hash-linked records while also allowing the deletion of arbitrary records, preserving hash-based integrity assurance that other blocks are unchanged.

2 Data Structure

Consider a matrix as shown in Fig. 1, with rows and columns numbered indexing data blocks, where a block may contain unspecified data (e.g., single record or multiple transactions). Each row and column is terminated with a hash of that row or column, e.g., $H_{0,-}$ is the hash of row 0. Alternatively, the hash value can be stored in the last block of the row or column. A second alternative could be to concatenate hashes of each block in a row or column and use the hash of this concatenation 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. Block matrix

Suppose that it is desired to delete the block labeled "X", by writing all zeroes to that block, or otherwise change it. This change disrupts 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.

3 Algorithm

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.)

```
while (new blocks) {// i, j = row, column indices
    if      (i == j) {add null block; i = 0;  j++;}
    else if (i  <  j) {add block(i,j); swap(i,j);}
    else if (i  >  j) {add block(i,j); j++; swap(i,j);}
}
```

where 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 Fig. 2.

	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

4 Properties

We can show that certain desirable properties are maintained with this data structure.

Balance: 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. Note that the following invariant is maintained for each iteration of the loop:

$$(i = j \ \lor \ i < j) \land \ u = l \ \lor \ \ i > j \ \land \ u = l + 1$$

where u = number of cells above diagonal, and l = number of cells below diagonal.

Hash chain length: The number of blocks in a row or column hash chain is proportional to $\sqrt{N}$ for a matrix with N blocks, by the balance property.

Block dispersal: No consecutive blocks appear in the same row or column, i.e., for any two blocks numbered a, b, where $b = a+1$, in rows ia and ib, and columns ja and jb respectively, $ia \neq ib$ and $ja \neq jb$. This can be shown by considering cases below.

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

2. If $i > j$, then block a will be written to cell (ia,ja), j incremented, and then i and j swapped. Then either the relationship is unchanged, with $i > j$, or $i = j$.
    - 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 ib = 0 and $jb = ja+1$, ensuring that $ia \neq ib$ and $ja \neq jb$.
    - if $i > j$, then on the next iteration, block b will be written with $ib = ja$ and $jb = ia$, and $i \neq j$, so that $ia \neq ib$ and $ja \neq jb$.

Because no two consecutive blocks appear in the same row or column, it is possible to delete two consecutive blocks simultaneously without disturbing integrity protection for others. This is possible because the diagonal is null. Without this property, for example, the following could occur:

	0	1	2	3
0	1	2	5	10
1	3	4	7	12
2	6	8	9	14
3	11	13	15	16

Figure 3. Block matrix with diagonal used

In Fig. 3, if blocks 7 and 8 are deleted, then integrity protection for 4 and 9 would be lost as well, because hashes would be invalidated for row 1, column 2, and row 2, column 1. Then cells (1,1) and (2,2) have neither row nor column hashes.

Number of blocks: 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 $N^2 - N$. With rows and columns numbered from 0, $i = N$-1 and the last data block in the lower half (below diagonal) is $(i+1)^2 - (i+1) = i^2 + i$, and for any row i, the last data block in the lower half is numbered $i^2 + i$. Thus the last data block in row i -1 in the lower half is $i^2 - i$ and the first in row i is $i^2 - i + 2$. Similarly, the last upper half data block in column j is $j^2 + j - 1$.

5 Block location

With the relations above, we can derive expressions to locate a given block within the matrix. For a block B in the lower half (B is even):

$$s = \lfloor \sqrt{B} \rfloor$$
$$i = B \leq s^2 + s\ ?\ s : s + 1$$
$$j = (B - (i^2 - i + 2))/2$$

and for block B in the upper half (B is odd):

$$s = \lfloor \sqrt{B+1} \rfloor$$
$$j = B < s^2 + s\ ?\ s : s + 1$$
$$i = (B - (j^2 - j + 1))/2$$

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 private (permissioned) blockchains [1],[2],[3], 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 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 straightforward extensions to the algorithms above.

References

[1] M. Berberich and M. Steiner, “Blockchain Technology and the GDPR,” *2 Eur. Data Protection Law Review*. 422, 2016.

[2] H.Chang, “Blockchain: Disrupting Data Protection?”, *Privacy Law and Business Intl. Rpt.*, Nov.2017.

[3] O. Kharif, “Is Your Blockchain Doomed?”, *Bloomberg Business Week*, Mar. 22, 2018.

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

Disclaimer: Products may be identified in this document, but identification does not imply recommendation or endorsement by NIST, nor that the products identified are necessarily the best available for the purpose.