Using Block Matrices to Provide Erasure Capabilities to Blockchains Without Losing Integrity

Arsen Klyuev

Computer Security

Security Components & Mechanisms

August 1st, 2018

Hashing and Hash Functions

- Hashing: "A method of calculating a relatively unique output for an input of any size."
- Any change to the input, no matter how small, will result in a completely different output
- Preimage resistant
- Second preimage resistant
- Collision resistant
- SHA-256

Input Text	SHA-256 Digest Value
1	0x6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b
2	0xd4735e3a265e16eee03f59718b9b5d03019c07d8b6c51f90da3a666eec13ab35
Hello, World!	0xdffd6021bb2bd5b0af676290809ec3a53191dd81c7f70a4b28688a362182986f

What is a Block Matrix?

- A data structure which supports addition of hash-linked records while also allowing the deletion of records yet still maintaining assurance other blocks are unchanged

- Stores hashes of each row and 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

How do Block Matrices Maintain Security?

- Hashes provide us assurance information in every other block is unchanged if one block is modified

	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

Block Matrix Population Algorithm

- Algorithm:

```
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);}
}
```

- Special ordering creates certain desirable properties
  - Balance
  - Capability of deleting consecutive blocks

	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

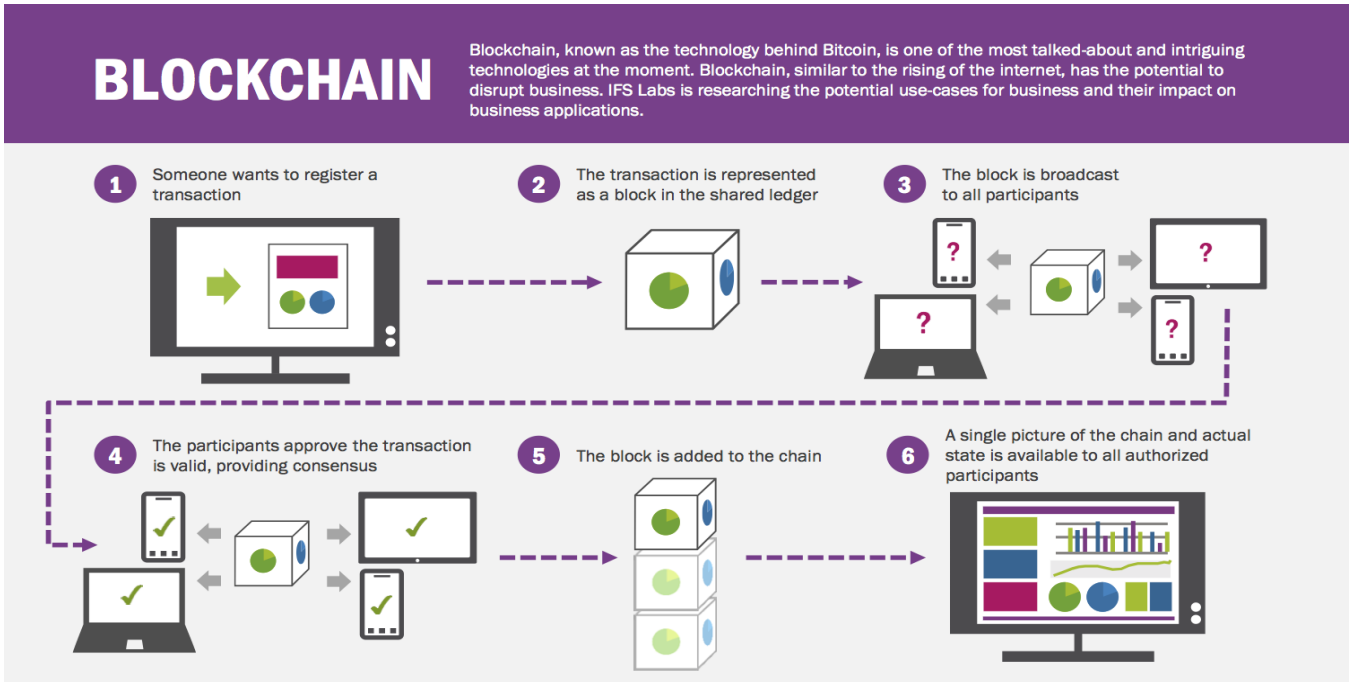
Vs.

	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

What are blockchain networks?

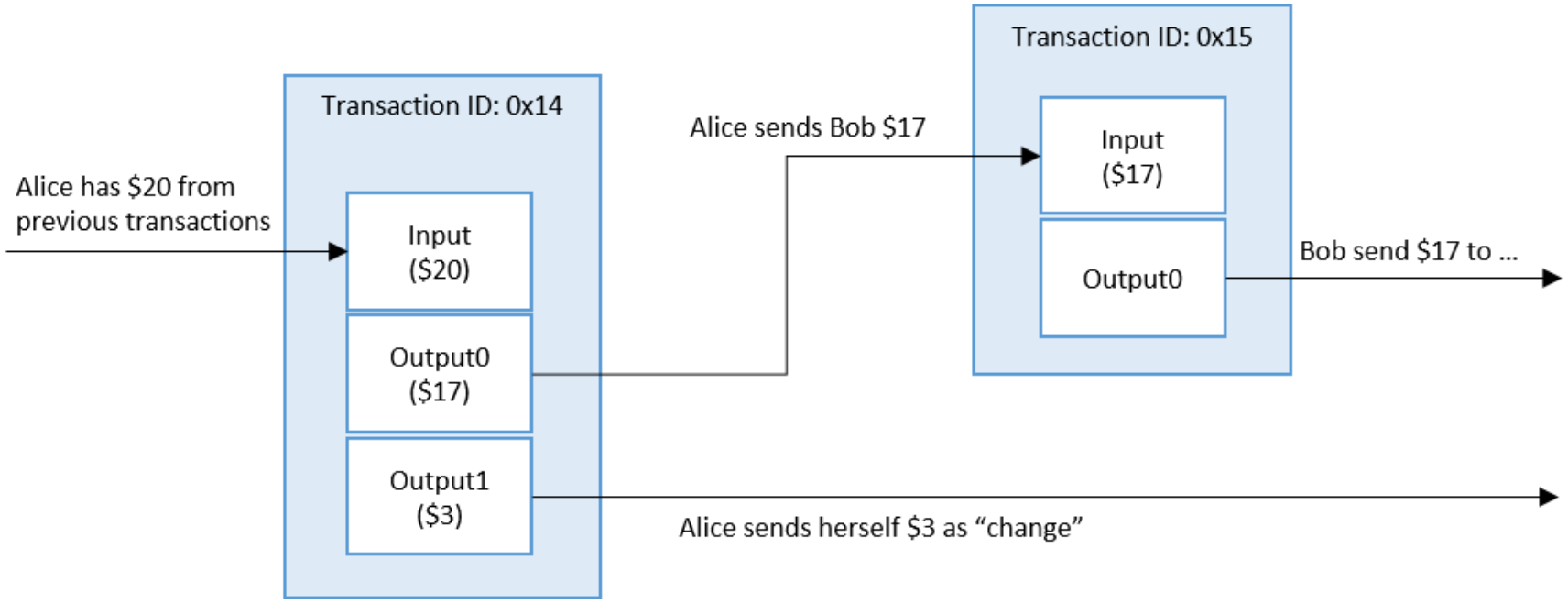
- Distributed digital ledger systems
  - Tamper evident
- Transactions are public
- Each member of the network has a full record of transactions



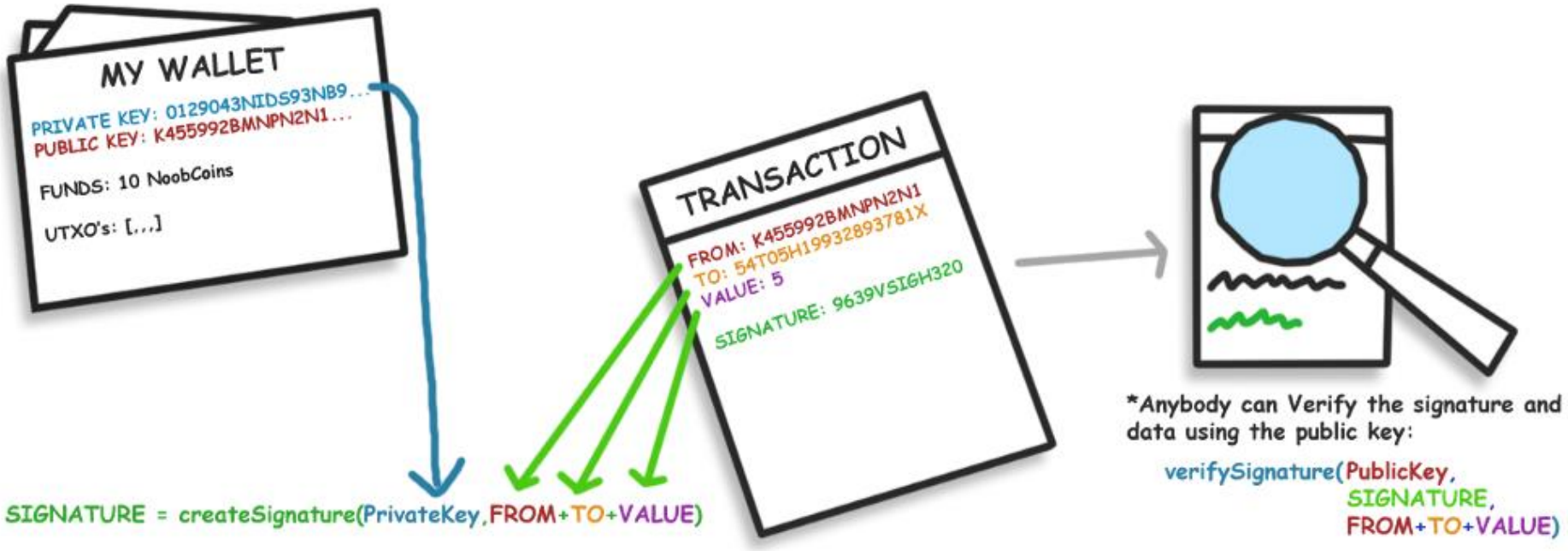
Proof of Work Consensus Model

- Blocks will contain a nonce, an arbitrary number that is only used once
  - hash(data + nonce)=digest
- Need to meet a target criteria:
  - E.g. SHA256("blockchain" + Nonce) = Hash Value starting with "000000"
  - SHA256("blockchain0") = 0xbd4824d8ee63fc82392a6441444166d22ed84eaa6dab11d4923075975acab938 (not solved)
  - SHA256("blockchain1") = 0xdb0b9c1cb5e9c680dfff7482f1a8efad0e786f41b6b89a758fb26d9e223e0a10 (not solved)
  - SHA256("blockchain10730895") = 0x000000ca1415e0bec568f6f605fcc83d18cac7a4e6c219a957c10c6879d67587 (solved!)
- Once this is solved, the block can be added to the blockchain and people get to work on mining the next "block"
- Generally there is an incentive for mining a block (e.g. some part of the asset a.k.a some "coins"

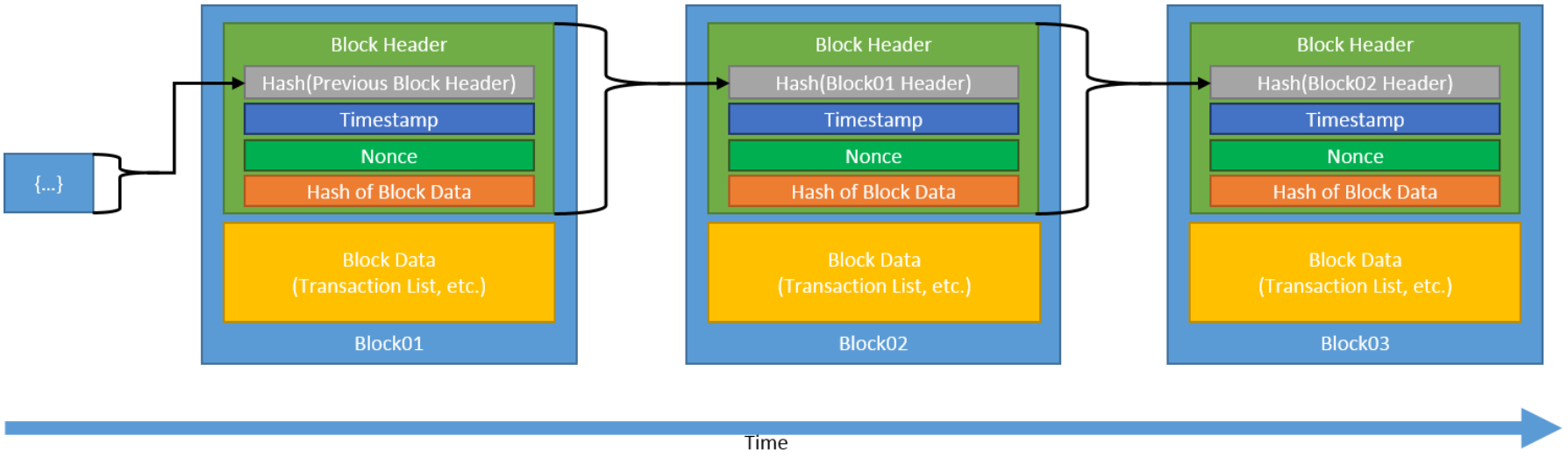
Transactions



Public and Private Keys



Structure of a Traditional Blockchain

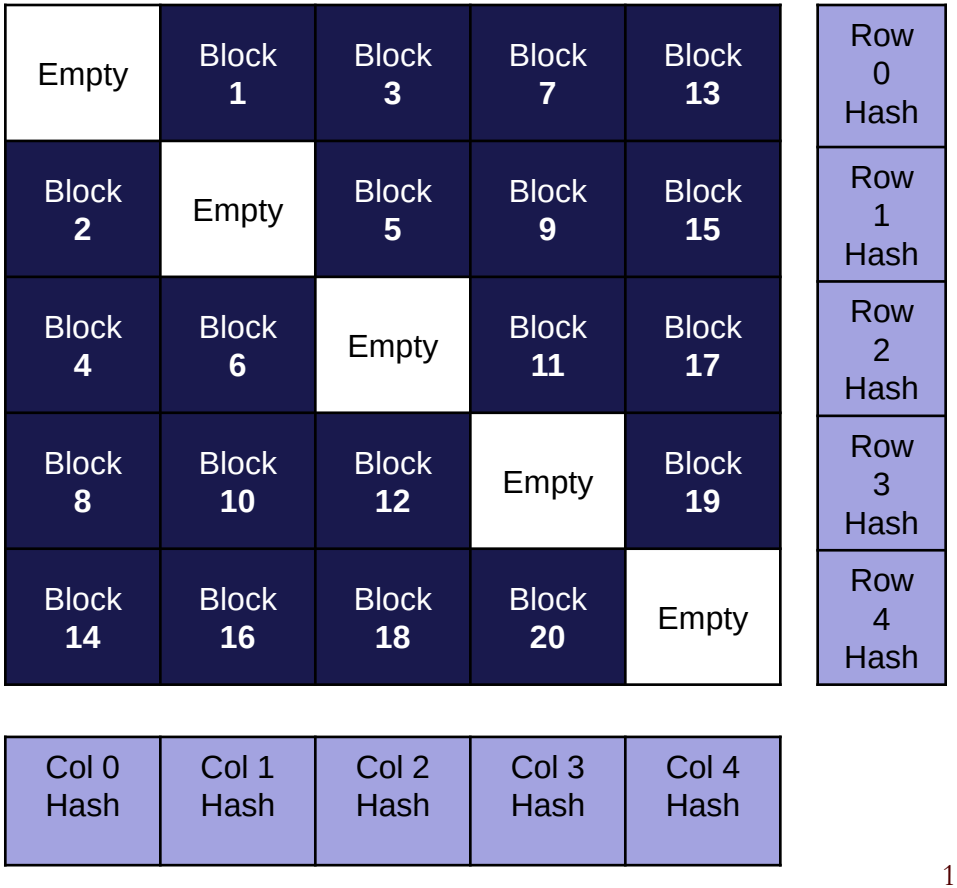


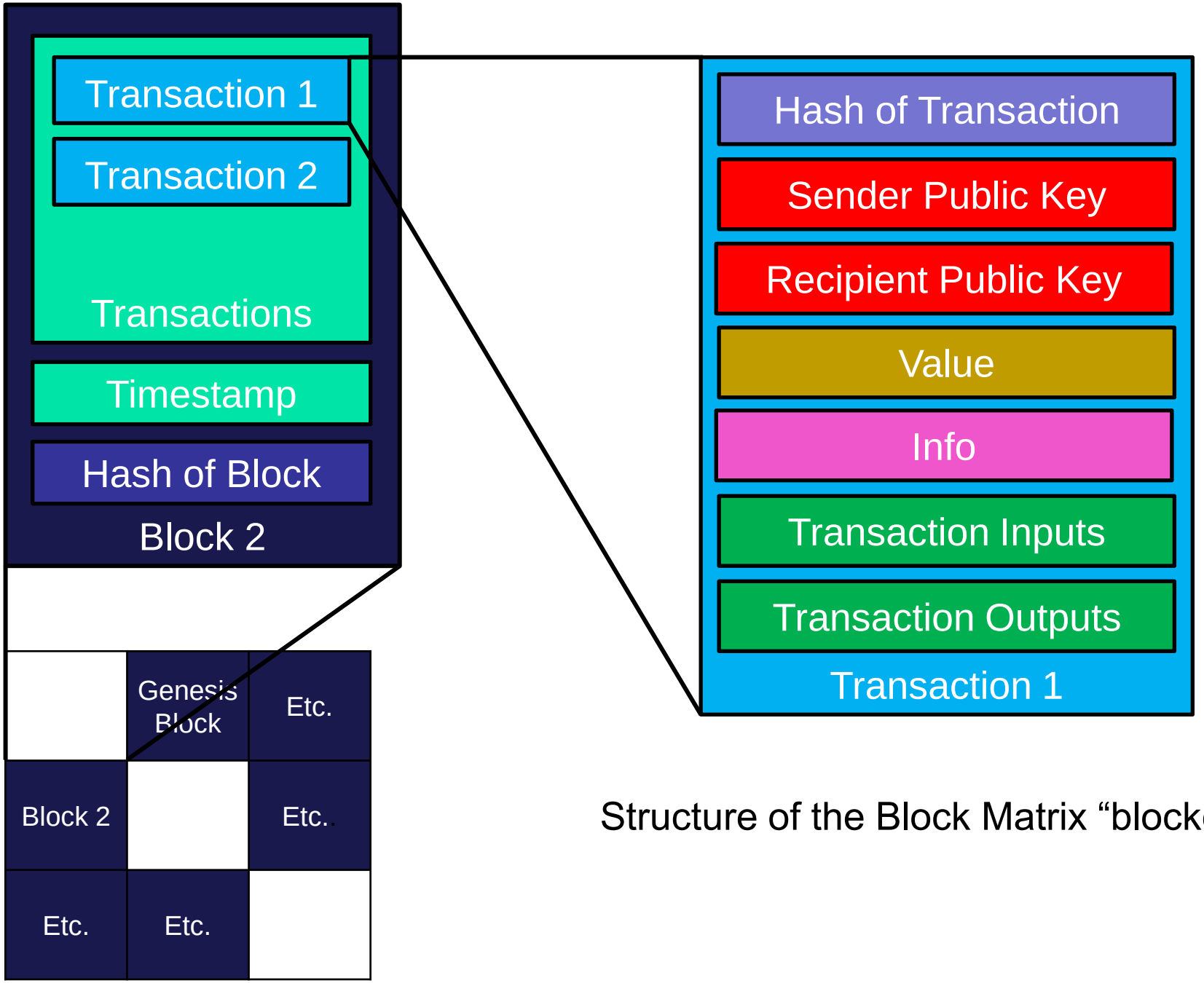
The Problem

- Immutability means any information on a blockchain can never be removed
- Increasing privacy regulations make this a problem
- EU General Data Protection Regulation (GDPR)

Applying Block Matrices to Blockchains

- Similar structure and security as a blockchain
- capability of deleting or modifying certain parts of a transaction or block
- Same transaction model, same cryptographic key/address model





Structure of the Block Matrix “blockchain”

Java BlockMatrix Package

A mutable blockchain you can use yourself!

```
import blockmatrix.*;

public class Main {

    public static void main(String[] args) {
        BlockMatrix bm = new BlockMatrix(5);
        bm.setUpSecurity();

        //Create wallets:
        Wallet walletA = new Wallet();
        bm.generate(walletA, 200f);
```

Making our block matrix with our desired dimensions

Creating our initial wallet that starts with all of the “coins”

Making our genesis transaction with the desired number of “coins”

- SHA-256 hashing
- Elliptic-Curve Keypairs

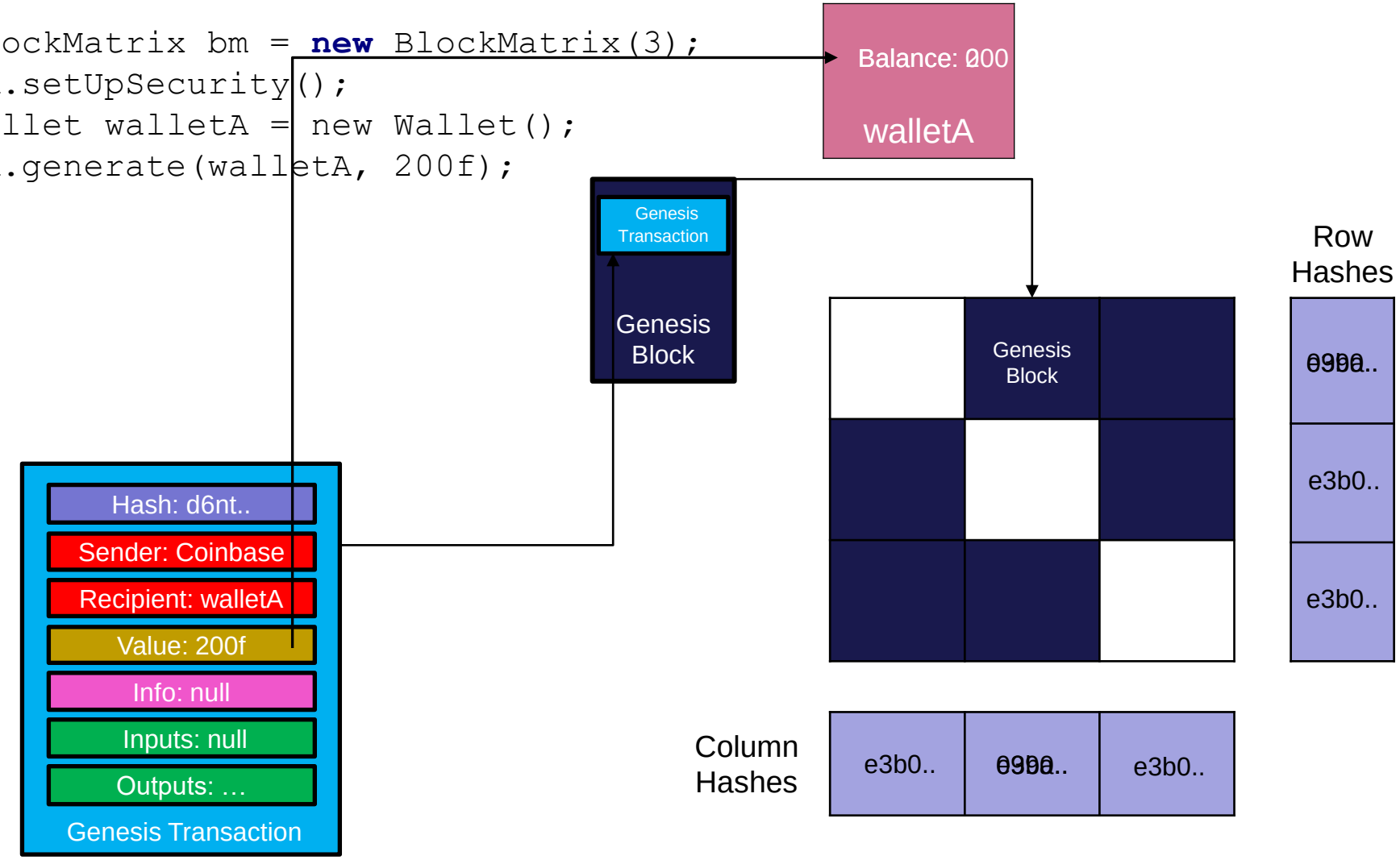
How to use it

- Create wallets: `Wallet walletB = new Wallet();`
- Create Blocks: `Block block2 = new Block();`
- Create transactions
  - `Transaction tr = walletA.sendFunds(walletB.getPublicKey(), 40f, "This is for the bananas!");`
- Add the transactions to blocks: `block2.addTransaction(tr);`
- Add the blocks to the block matrix! `bm.addBlock(block2);`

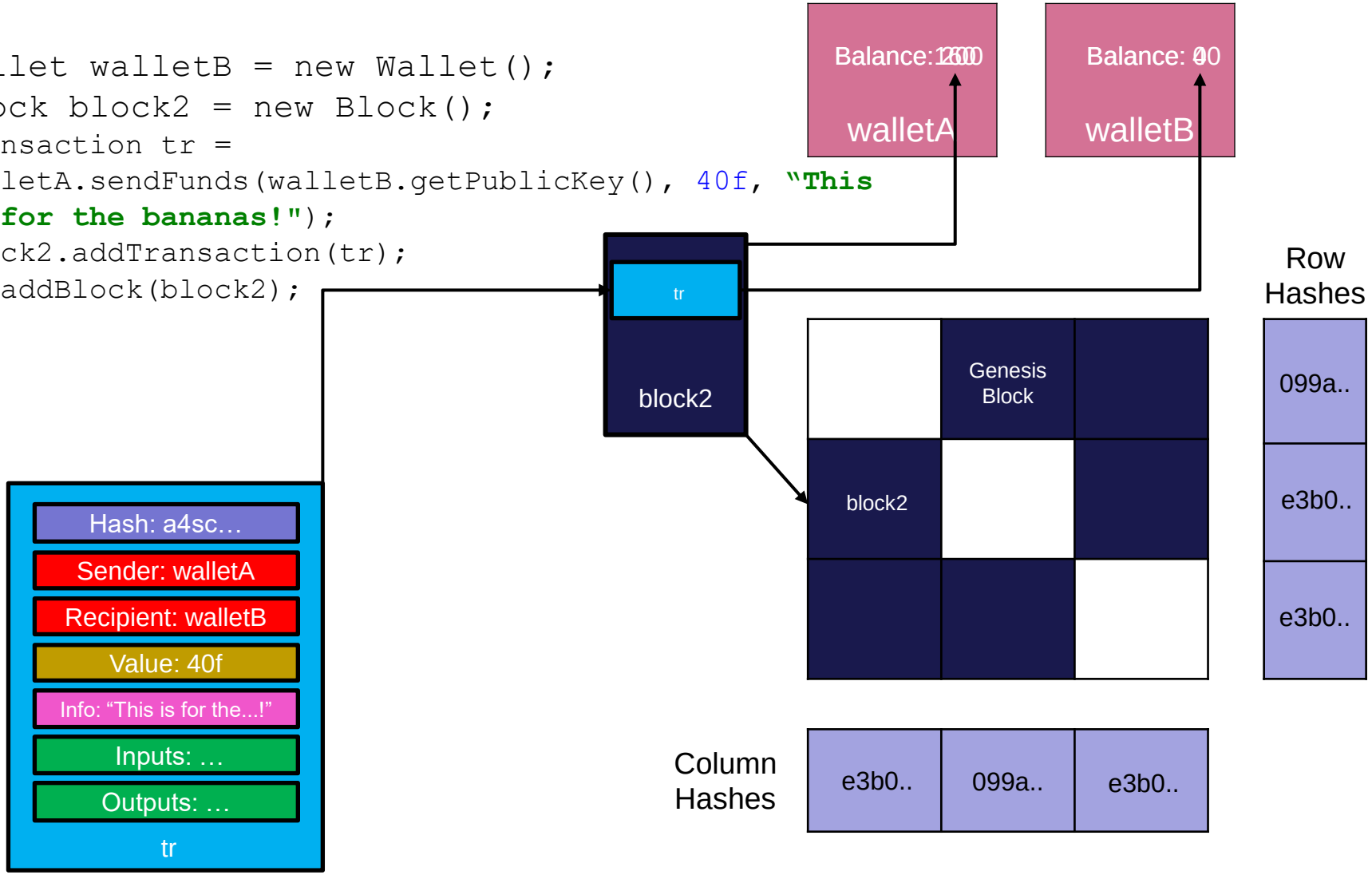
```
//testing
Wallet walletB = new Wallet();
Block block2 = new Block();
System.out.println("\nWalletA's balance is: " + walletA.getBalance());
System.out.println("\nWalletA is sending 40 coins to WalletB...");
block2.addTransaction(walletA.sendFunds(walletB.getPublicKey(), 40f, "This is for the
bananas!"));
bm.addBlock(block2);
System.out.println("\nWalletA's balance is: " + walletA.getBalance());
System.out.println("WalletB's balance is: " + walletB.getBalance());
```

- Clearing info in blocks: `bm.clearInfoInTransaction(2, 0);`

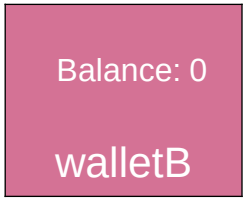
```
BlockMatrix bm = new BlockMatrix(3);
bm.setUpSecurity();
Wallet walletA = new Wallet();
bm.generate(walletA, 200f);
```

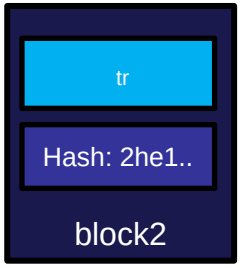


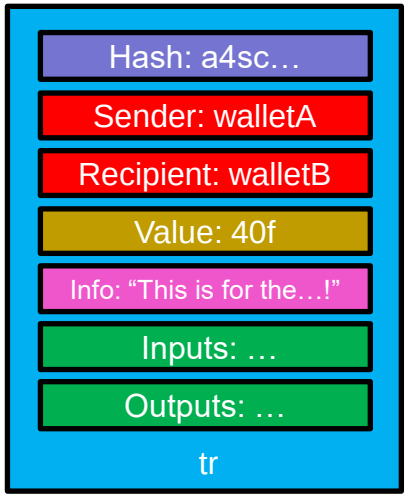
```
Wallet walletB = new Wallet();
Block block2 = new Block();
Transaction tr =
walletA.sendFunds(walletB.getPublicKey(), 40f, "This
is for the bananas!");
block2.addTransaction(tr);
Bm.addBlock(block2);
```

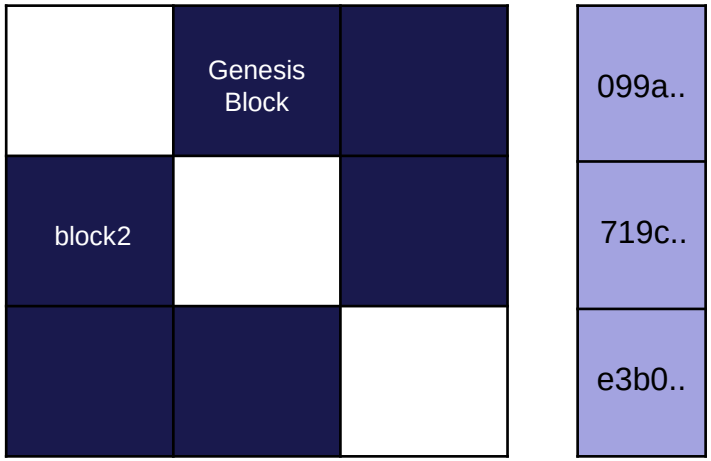


```
Wallet walletB = new Wallet();
Block block2 = new Block();
Transaction tr =
walletA.sendFunds(walletB.getPublicKey(), 40f, "This
is for the bananas!");
block2.addTransaction(tr);
bm.addBlock(block2);
bm.clearInfoInTransaction(2, 0);
```

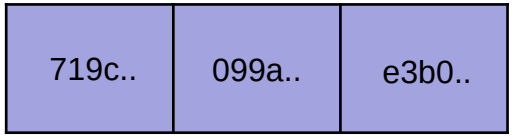




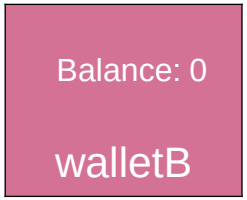


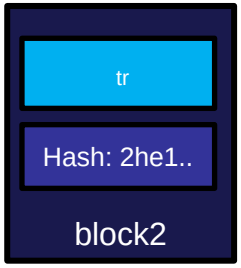


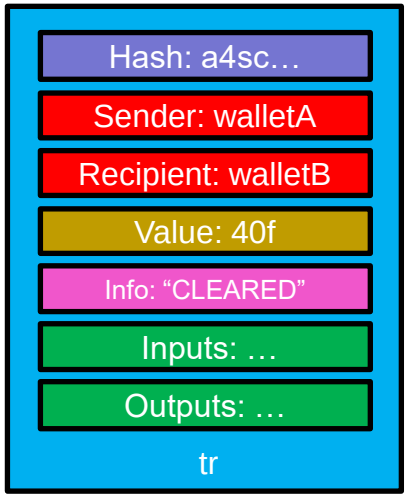
Column Hashes

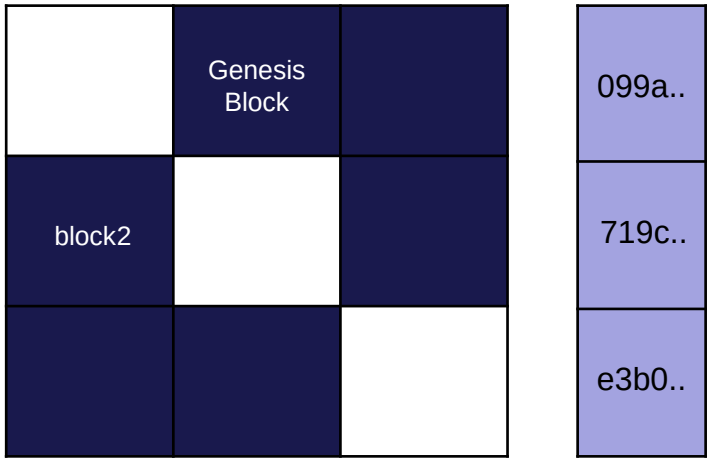


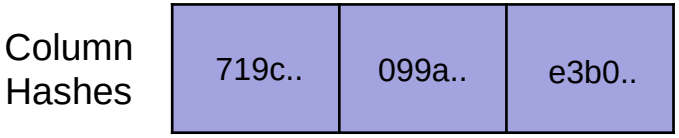
```
Wallet walletB = new Wallet();
Block block2 = new Block();
Transaction tr =
walletA.sendFunds(walletB.getPublicKey(), 40f, "This
is for the bananas!");
block2.addTransaction(tr);
bm.addBlock(block2);
bm.clearInfoInTransaction(2, 0);
```



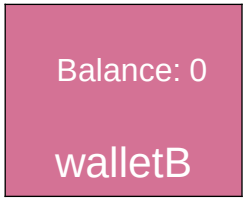


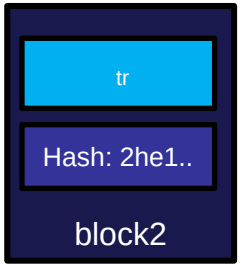


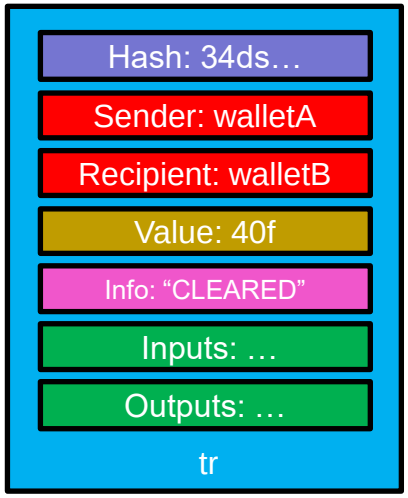


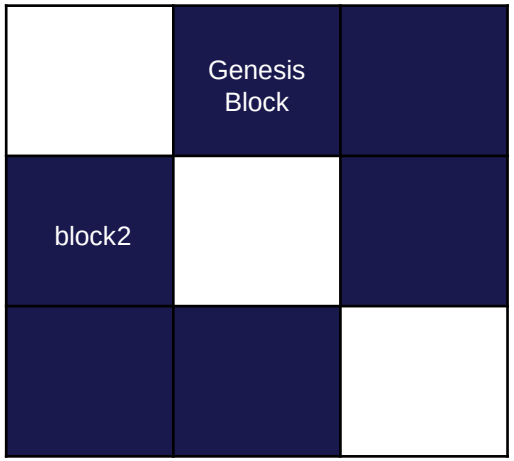


```
Wallet walletB = new Wallet();
Block block2 = new Block();
Transaction tr =
walletA.sendFunds(walletB.getPublicKey(), 40f, "This
is for the bananas!");
block2.addTransaction(tr);
bm.addBlock(block2);
bm.clearInfoInTransaction(2, 0);
```

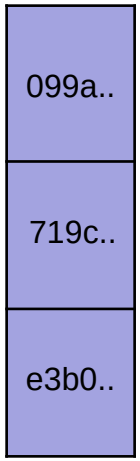




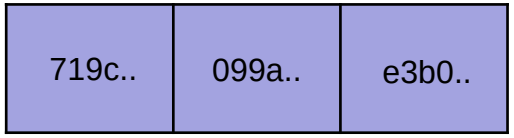




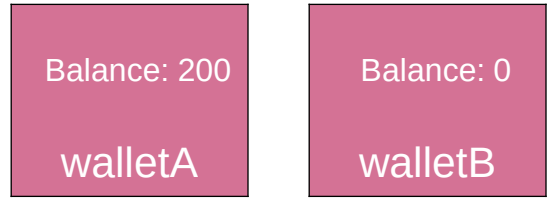
Row Hashes

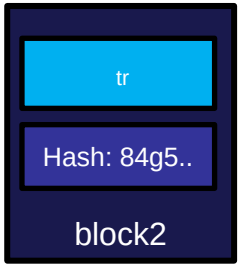


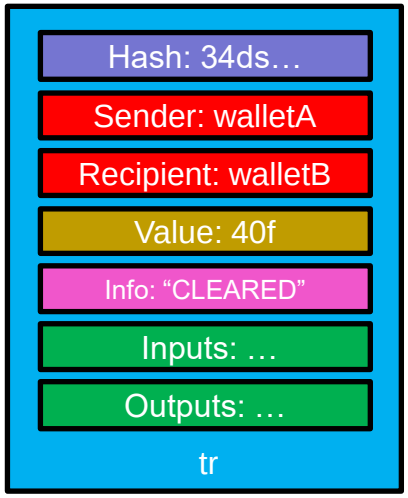
Column Hashes

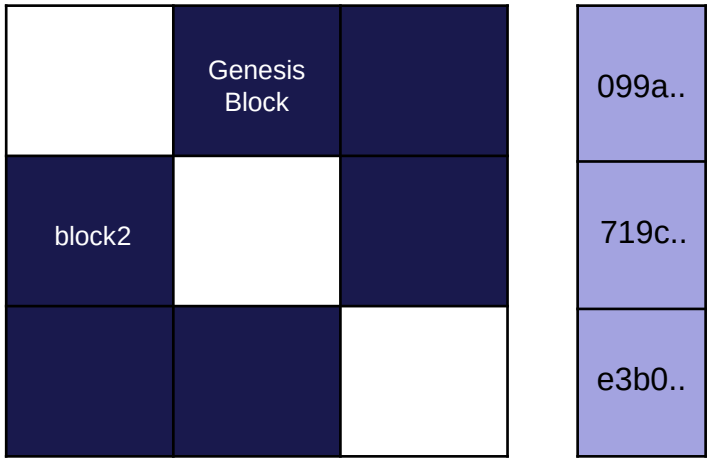


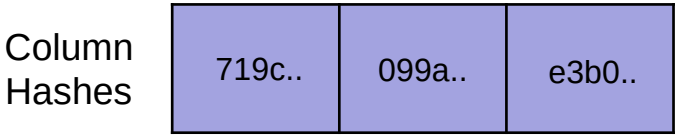
```
Wallet walletB = new Wallet();
Block block2 = new Block();
Transaction tr =
walletA.sendFunds(walletB.getPublicKey(), 40f, "This
is for the bananas!");
block2.addTransaction(tr);
bm.addBlock(block2);
bm.clearInfoInTransaction(2, 0);
```



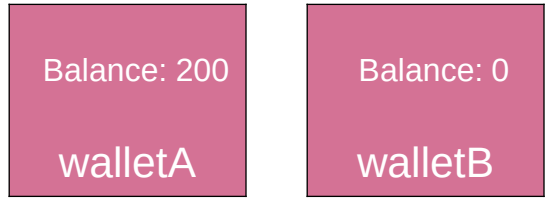


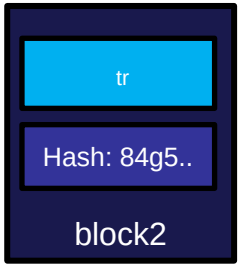


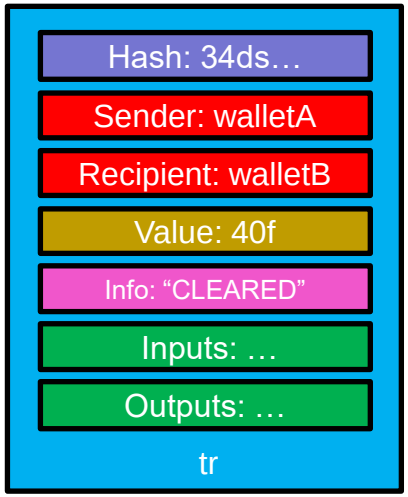


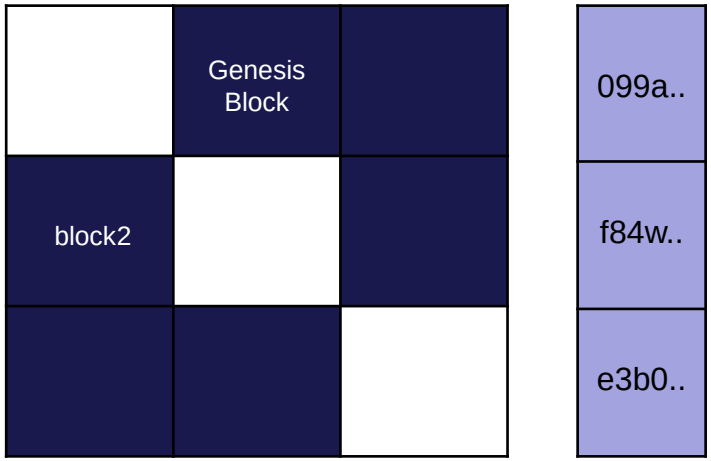


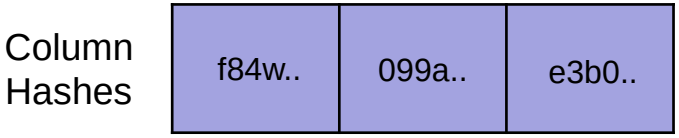
```
Wallet walletB = new Wallet();
Block block2 = new Block();
Transaction tr =
walletA.sendFunds(walletB.getPublicKey(), 40f, "This
is for the bananas!");
block2.addTransaction(tr);
bm.addBlock(block2);
bm.clearInfoInTransaction(2, 0);
```











Ensuring Matrix Validity

- isMatrixValid() method
  - Encompassing function which checks if blockmatrix is secure

- Features:
  - Checks every block and ensures its hash is what it should be
  - Checks every row and column and ensures its hash is what it should be
  - Checks every transaction in each block and makes sure that
    - The transactions signature is valid
    - Inputs are equal to outputs in the transaction
    - Etc.
  - Checks that all deletions/modifications of data changed only one row and column hash, and the rest are unchanged

Future of BlockMatrix

- For the package:
  - Option to make BlockMatrix use proof of work or alternate consensus schemes
  - Web tool to easily see structure of your BlockMatrix
- General code:
  - Extension to peer-to-peer system
  - Creation of generic BlockMatrix data structure which can be used for any purpose
- For the concept:
  - Implementation in existing blockchains
    - Multichain
    - Hyperledger Fabric

Acknowledgements

- Rick Kuhn, National Institute of Standards & Technology
- Dylan Yaga, National Institute of Standards & Technology
- SURF Undergraduate Research Program, National Institute of Standards & Technology

References

- Kuhn, D. Richard, *A Data Structure for Integrity Protection with Erasure Capability*, May 2018.
- Yaga, Dylan, et al. *Blockchain Technology Overview Draft NISTIR 8202 v2*, January 2018