CHETNA SINGHAL
DYLAN BRAY
SAM BELL
AADITYA KULKARNI
MADI GWYNN
RAVITEJA JASTI

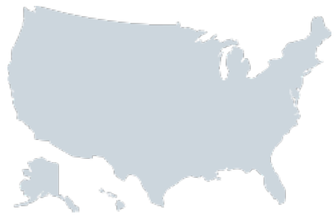
RED GOLD CRYPTO

A BLOCKCHAIN SOLUTION

BUSINESS PROBLEM

DEMAND =/ SUPPLY

Only about **3% of age-eligible** people donate blood yearly**.**



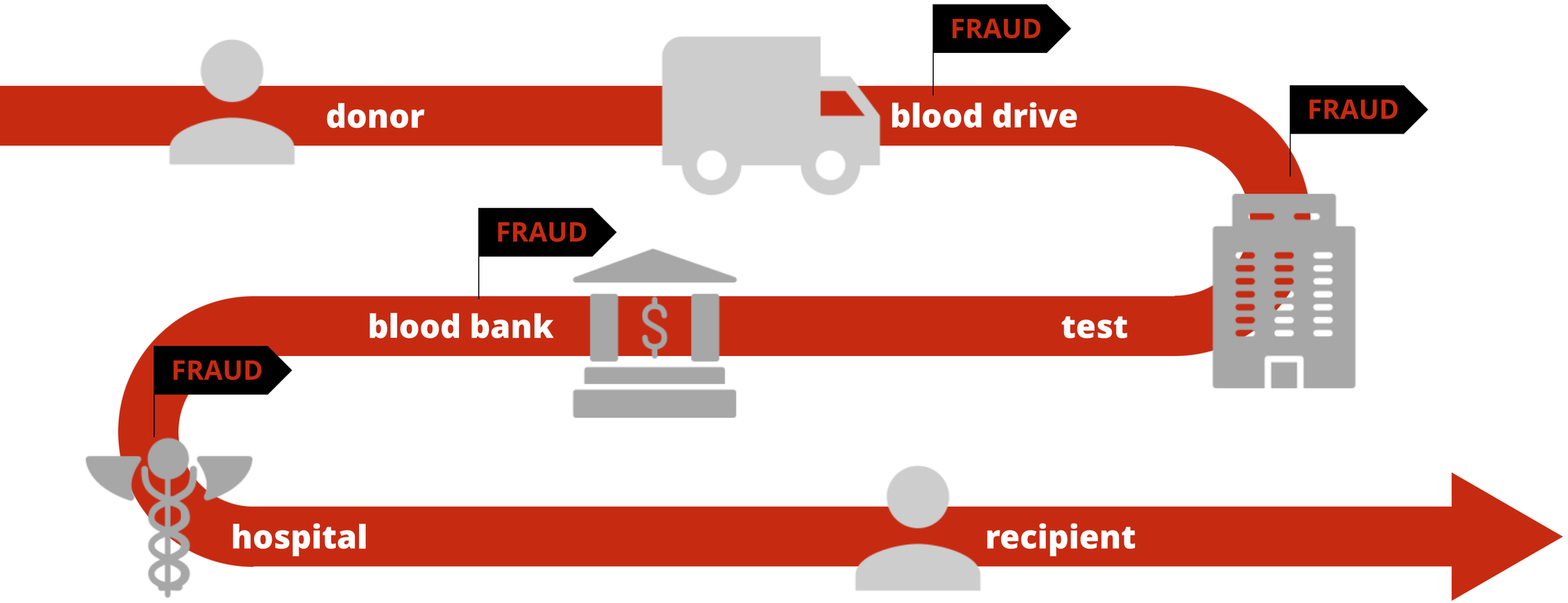
Less than **38% of the population** is eligible to give blood.

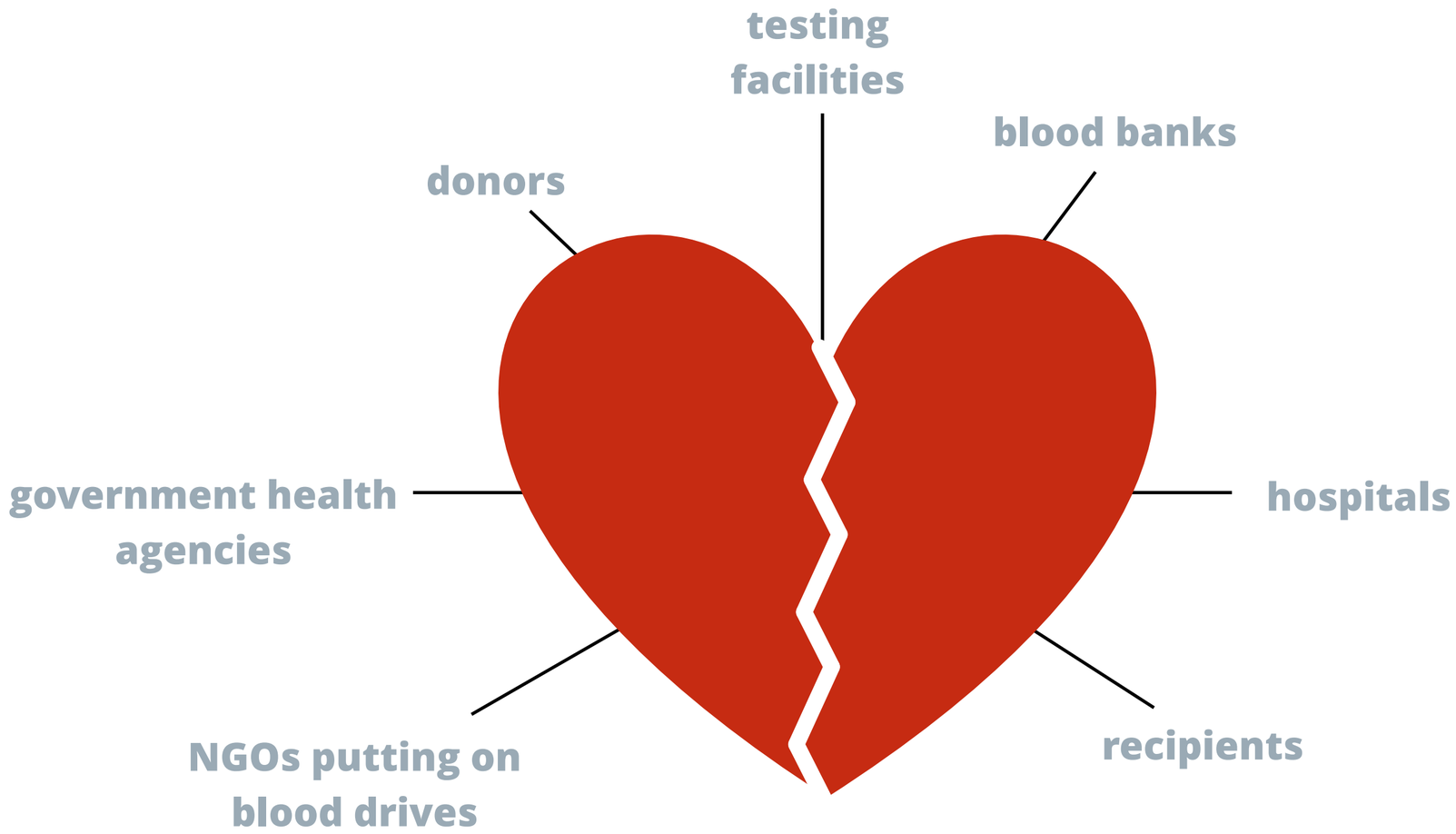
Blood cannot be manufactured; **only donated**.

Red blood cells must be used **within 42 days** or less.

SOURCE: **Red Cross Blood**
https://www.redcrossblood.org/donate-blood/how-to-donate/how-blood-donations-help/blood-needs-blood-supply.html

BLOOD DONATION PROCESS





STAKEHOLDERS AND SPONSORS

BUSINESS OUTCOMES

TIME FRAME:

1

YEAR

Triple the percentage of donors from **3% to**	**9%**	**American Red Cross**
Reduce discards due to blood expiration from **33% to**	**25%**	**American Association of Blood Banks**
Reduce acquisition cost for one unit of blood from **$210 to**	**$190**	**American Hospital Association**

S O U R C E S :

Red Cross Blood
https://www.redcrossblood.org/

WHO. (2016). Global Status Report on Blood Safety and Availability. World Health Organisation.

Jefferson School of Population Health, Thomas Jefferson University
https://www.ncbi.nlm.nih.gov/pubmed/21174480

BLOCKCHAIN SUITABILITY

Do we need a shared, consistent data store?

Does more than one entity need to contribute data?

Data records are never updated or deleted?

Sensitive identifiers WILL NOT be written to the data store?

Do writing entities have a hard time deciding who should be in control of the data store?

Do we want a tamperproof log of all writes to the data store?

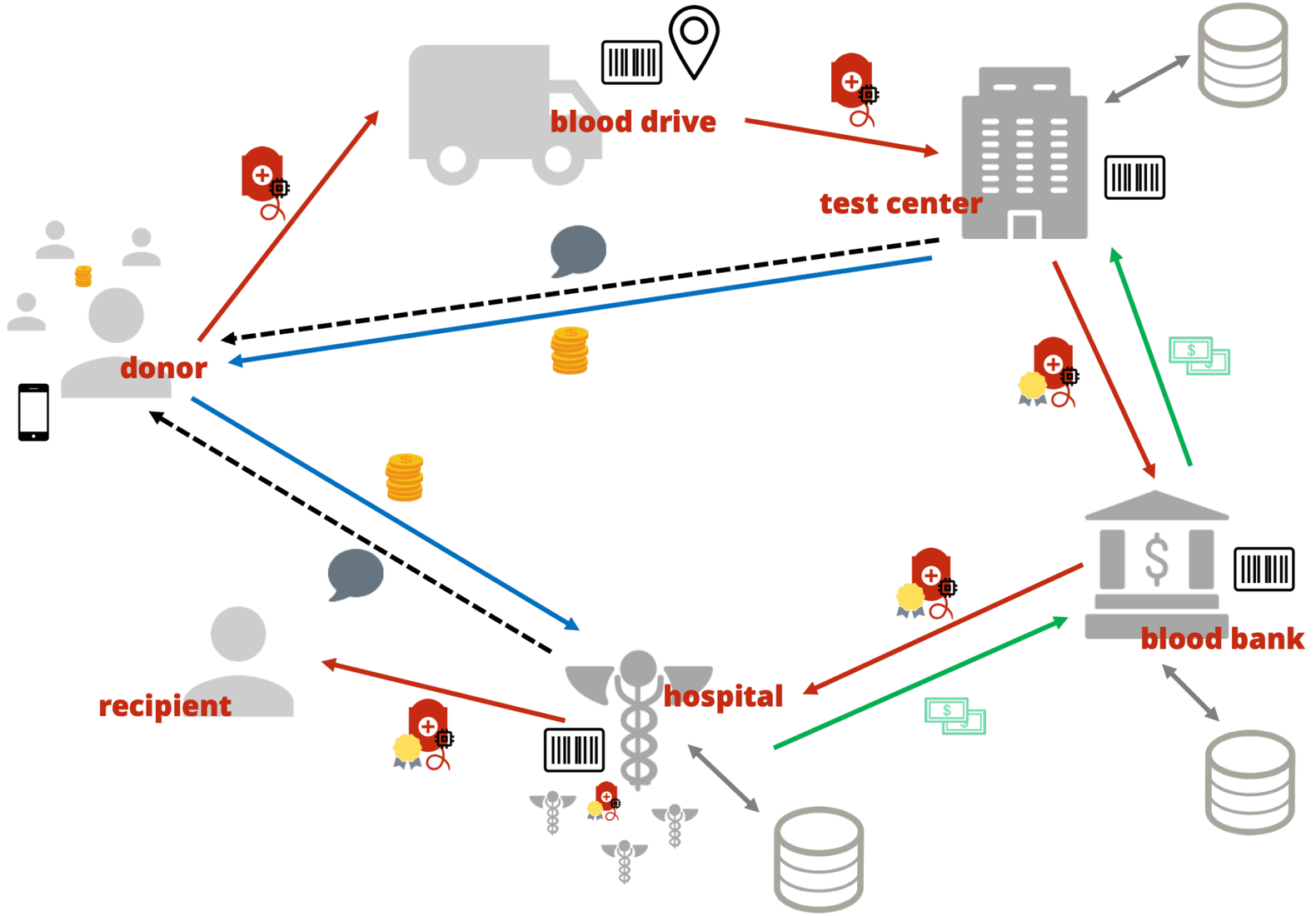
BLOCKCHAIN IS

suitable

FOR THIS BUSINESS PROBLEM

BLOCKCHAIN DESIGN

BLOCKCHAIN ARCHITECTURE



BLOOD TRANSFERED

PAYMENT TRANSFERED

TOKENS TRANSFERD

MESSAGE SENT

RFID READERS

LOCATION SENSOR

MOBILE APPLICATION

DATABASE SYSTEMS

BLOCKCHAIN CHARACTERISTICS

Permissioned

Ethereum

Proof of Work

INTEGRATION WITH OTHER SYSTEMS

Hospital Systems

RFIDs and Readers

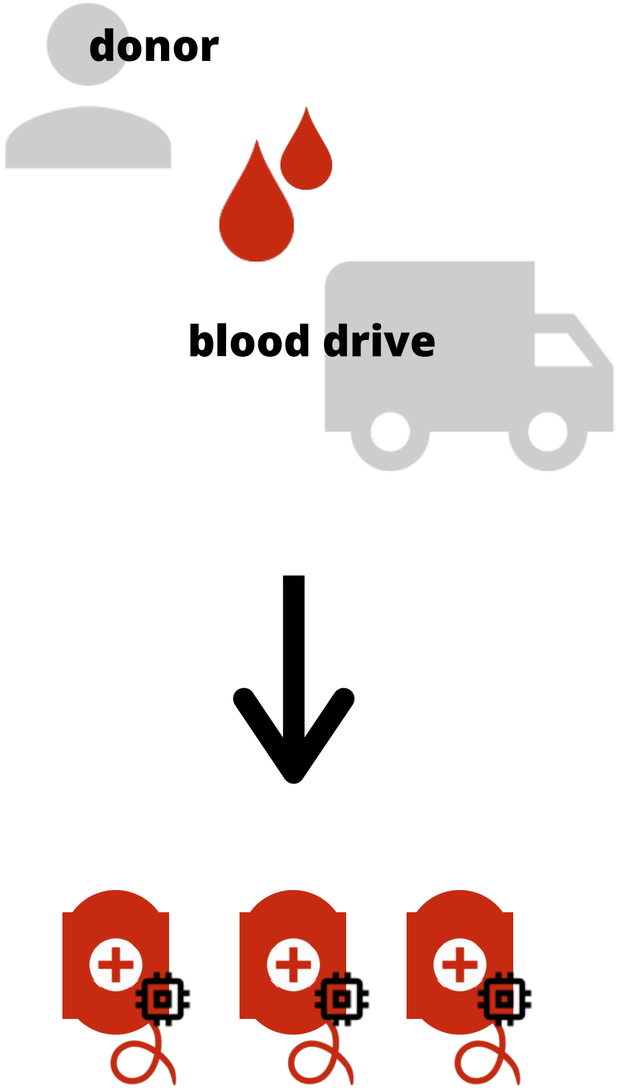
Donor App

Testing Centers Systems

Blood Storage Systems

Location Sensors

DONATION TRANSACTION



TRANSACTION 000001

To Address: Donation Drive ID

From Address: Donor ID

Input-

Blood Information

Location Sensor Details

Date

Signature of medical personnel

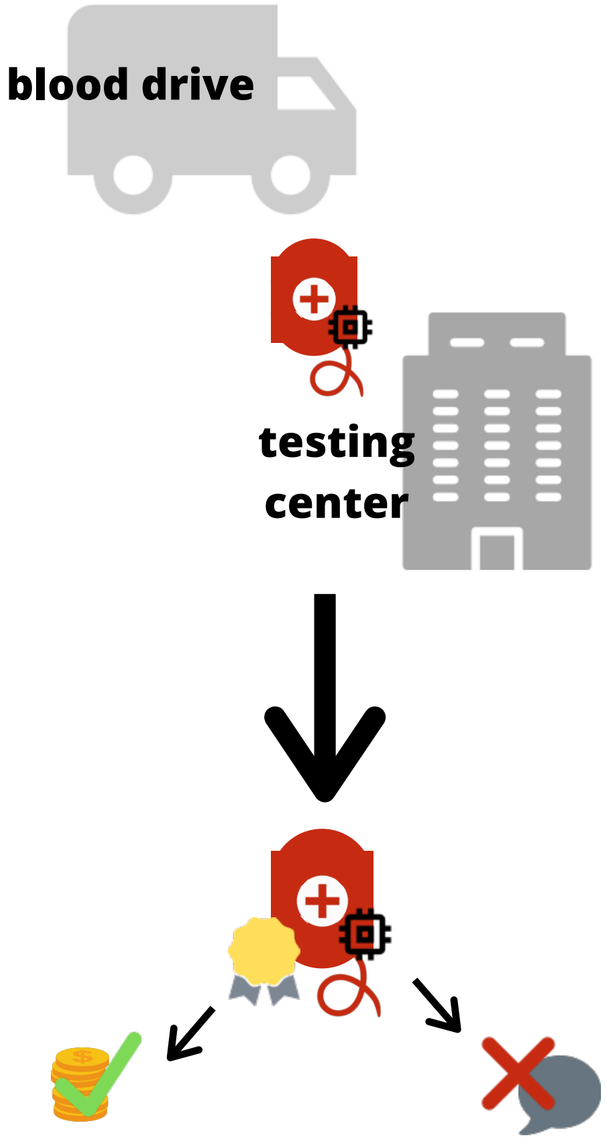
Output -

RFIDs of the blood bag

LEDGER

TRANSACTION 000001

TESTING CENTER
SMART CONTRACT



SMART CONTRACT

TRANSACTION 000002

To Address: Testing Center ID

From Address: Donation Drive ID

Input-

RFIDs of the blood bag

Output -

RFIDs of the blood bag

test certificate hashes

good or bad

If testing certificate states blood is **good**, donor receives **Donor Tokens**

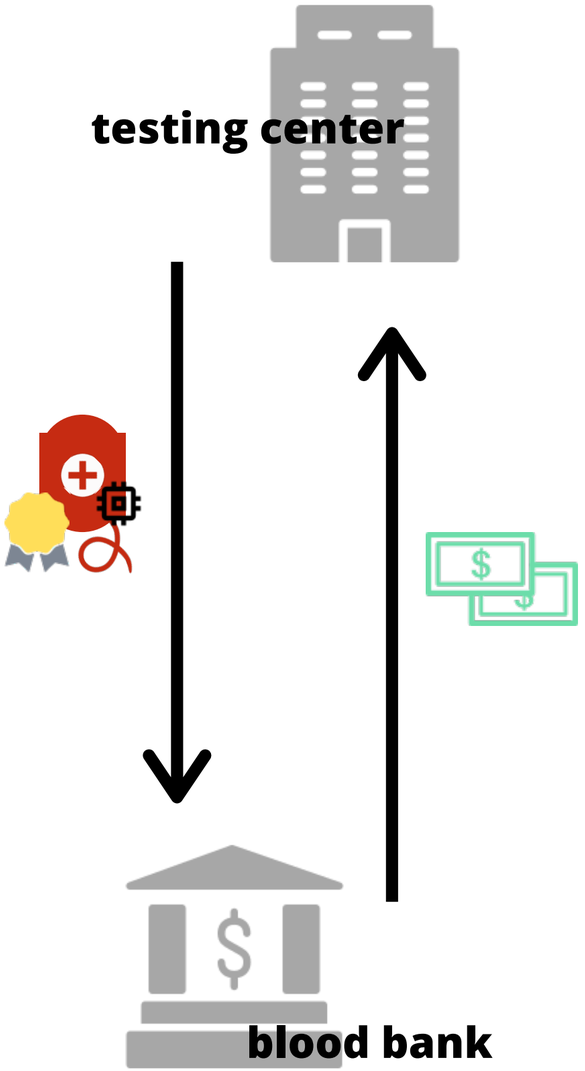
If testing certificate states blood is **bad**, donor receives **message stating why**

LEDGER

TRANSACTION 000001

TRANSACTION 000002

BLOOD BANK SMART CONTRACT



SMART CONTRACT

TRANSACTION 000003

To Address: Blood Bank ID

From Address: Testing Center ID

Input-

RFIDs of the blood bag

Output -

RFIDs of the blood bag

storage conditions [temp]

TRANSACTION 000004

Payment from blood bank to testing center

LEDGER

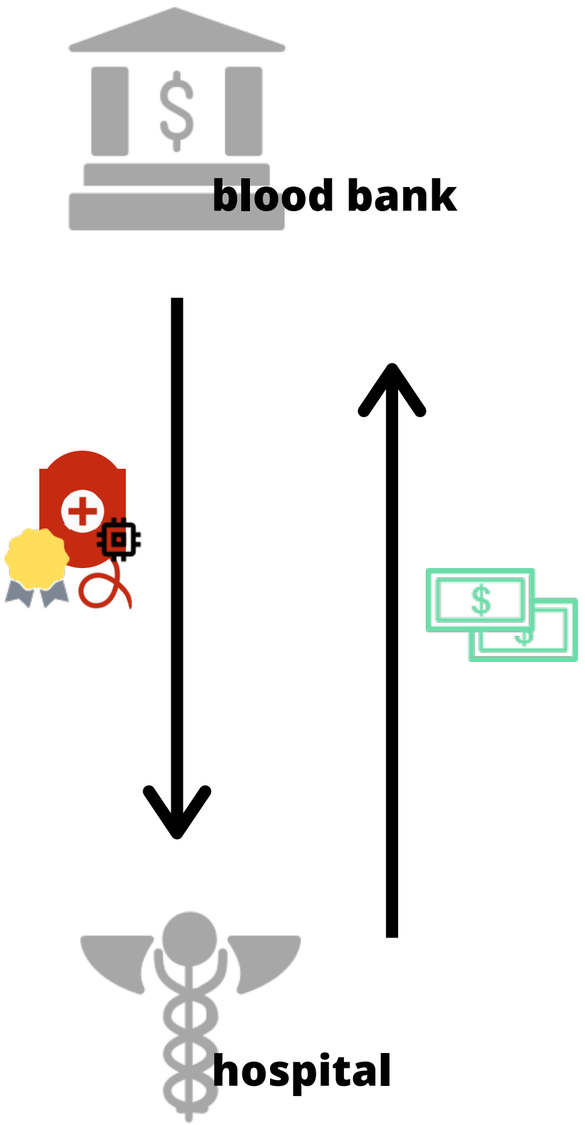
TRANSACTION 000001

TRANSACTION 000002

TRANSACTION 000003

TRANSACTION 000004

BLOOD CHECKOUT SMART CONTRACT



SMART CONTRACT

TRANSACTION 000005

To Address: Hospital ID

From Address: Blood Bank ID

Input-

RFIDs of the blood bag

Output -

RFIDs of the blood bag

TRANSACTION 000006

Payment from hospital to blood bank

LEDGER

TRANSACTION 000001

TRANSACTION 000002

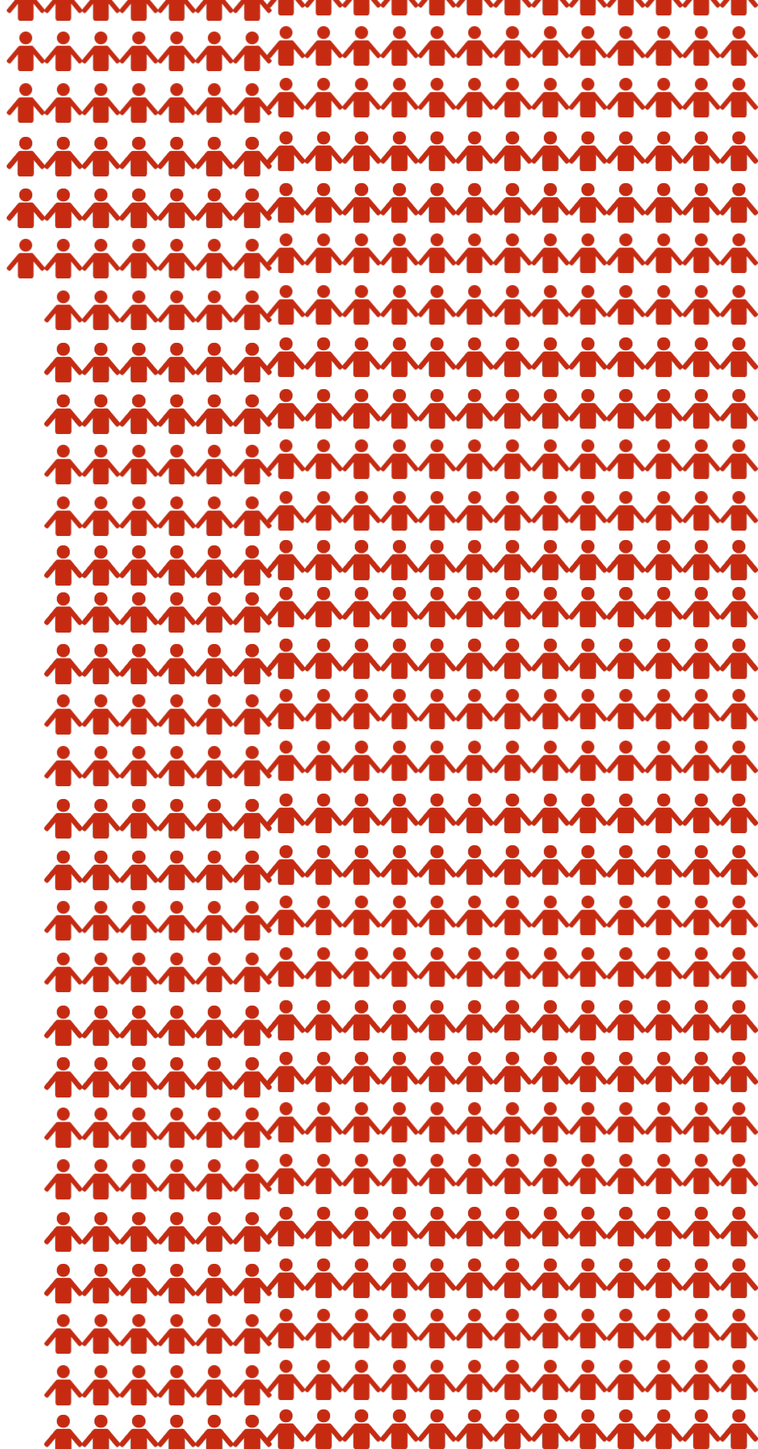
TRANSACTION 000003

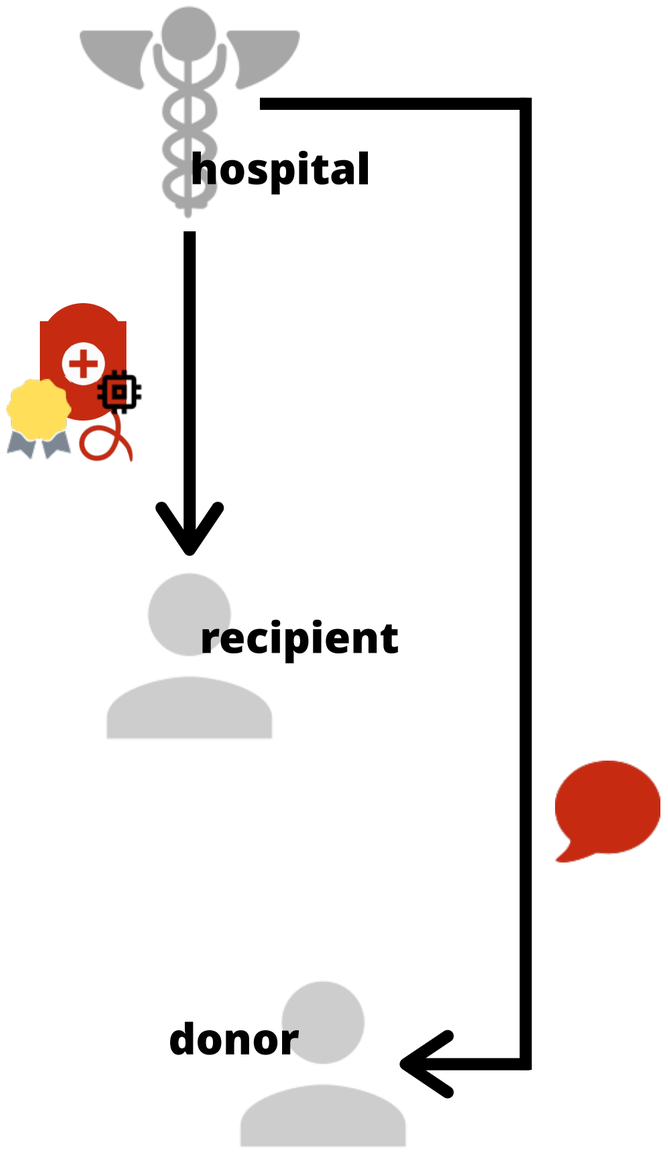
TRANSACTION 000004

TRANSACTION 000005

TRANSACTION 000006

BLOOD IS USED TRANSACTION





TRANSACTION 000008

To Address: Recipient ID

From Address: Hospital ID

Input-

RFIDs of the blood bag

Output -

Used Date

Donor is notified the **blood was used**

LEDGER

- TRANSACTION 000001
- TRANSACTION 000002
- TRANSACTION 000003
- TRANSACTION 000004
- TRANSACTION 000005
- TRANSACTION 000006
- TRANSACTION 000007
- TRANSACTION 000008

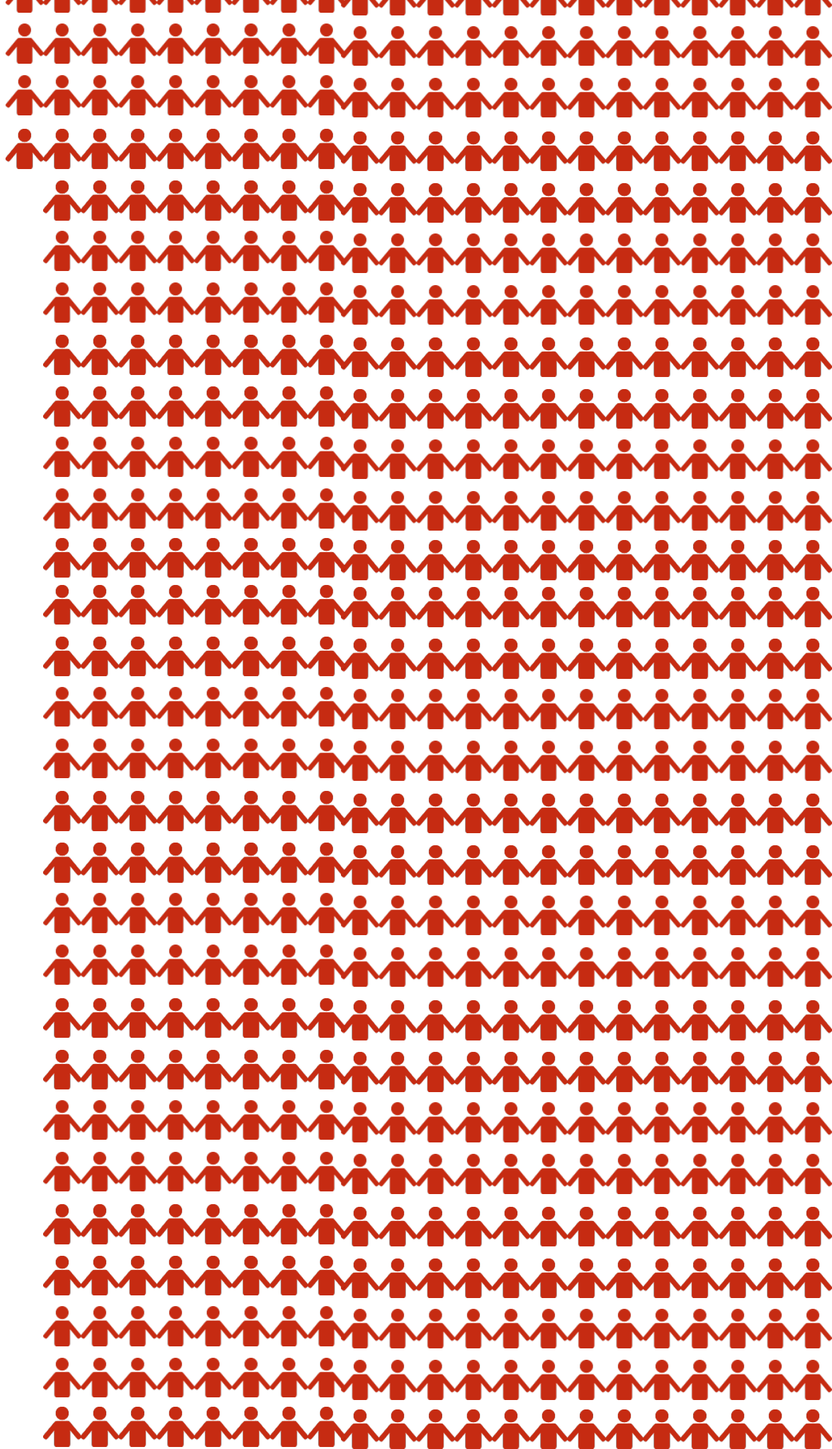
DATA

OFF CHAIN

Personal Information
Test Certificates
External Databases

ON CHAIN

Stakeholder IDs
RFIDs
Blood Information (ex. Type)
Test Certificate Hashes
Date
Storage Conditions



HOW ARE THEY AWARDED?

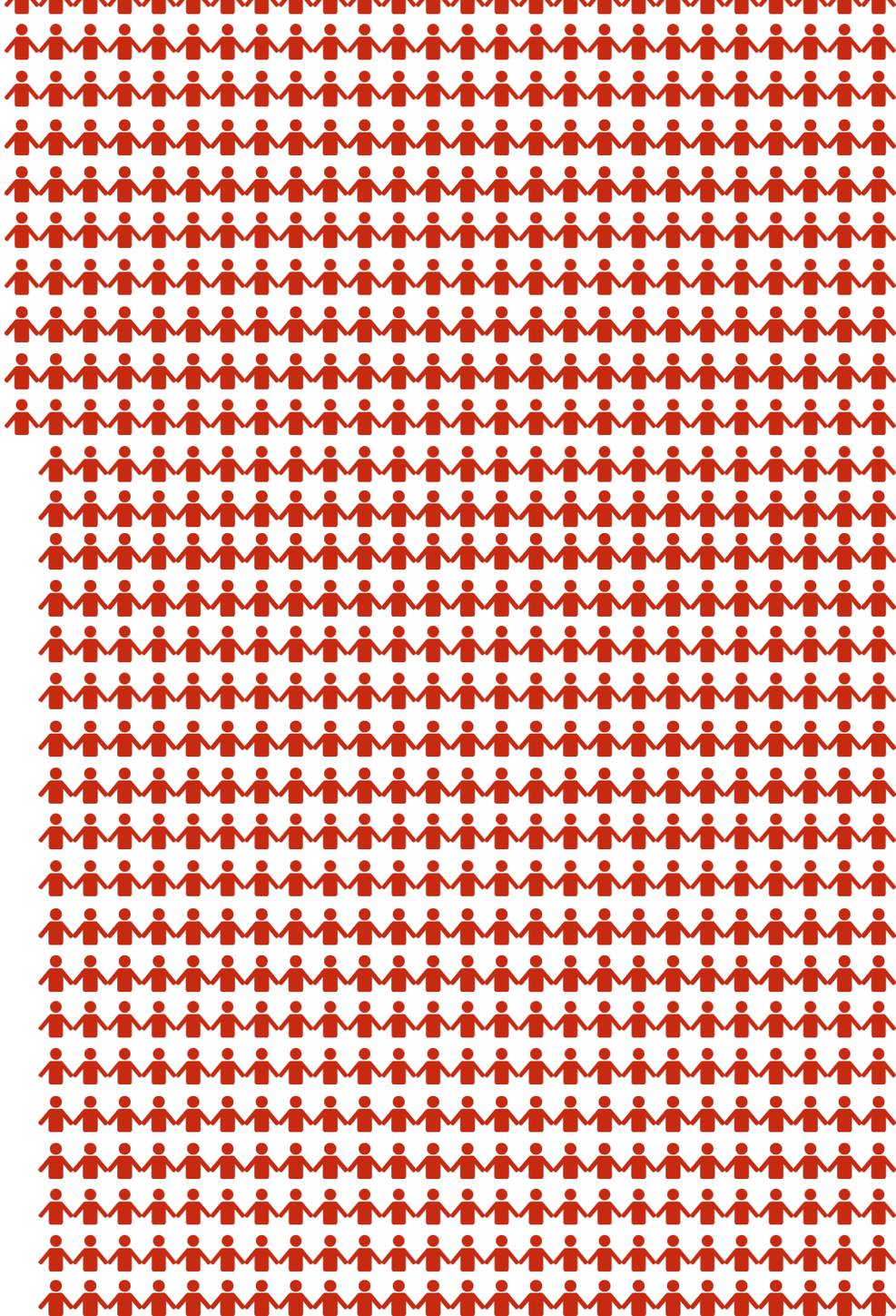
When a testing center verifies good blood, the donor receives donor tokens.

HOW CAN THEY BE SPENT?

Tokens can be exchanged for discounts on health services such as doctor visits, health insurance, and medication.

DONOR TOKENS

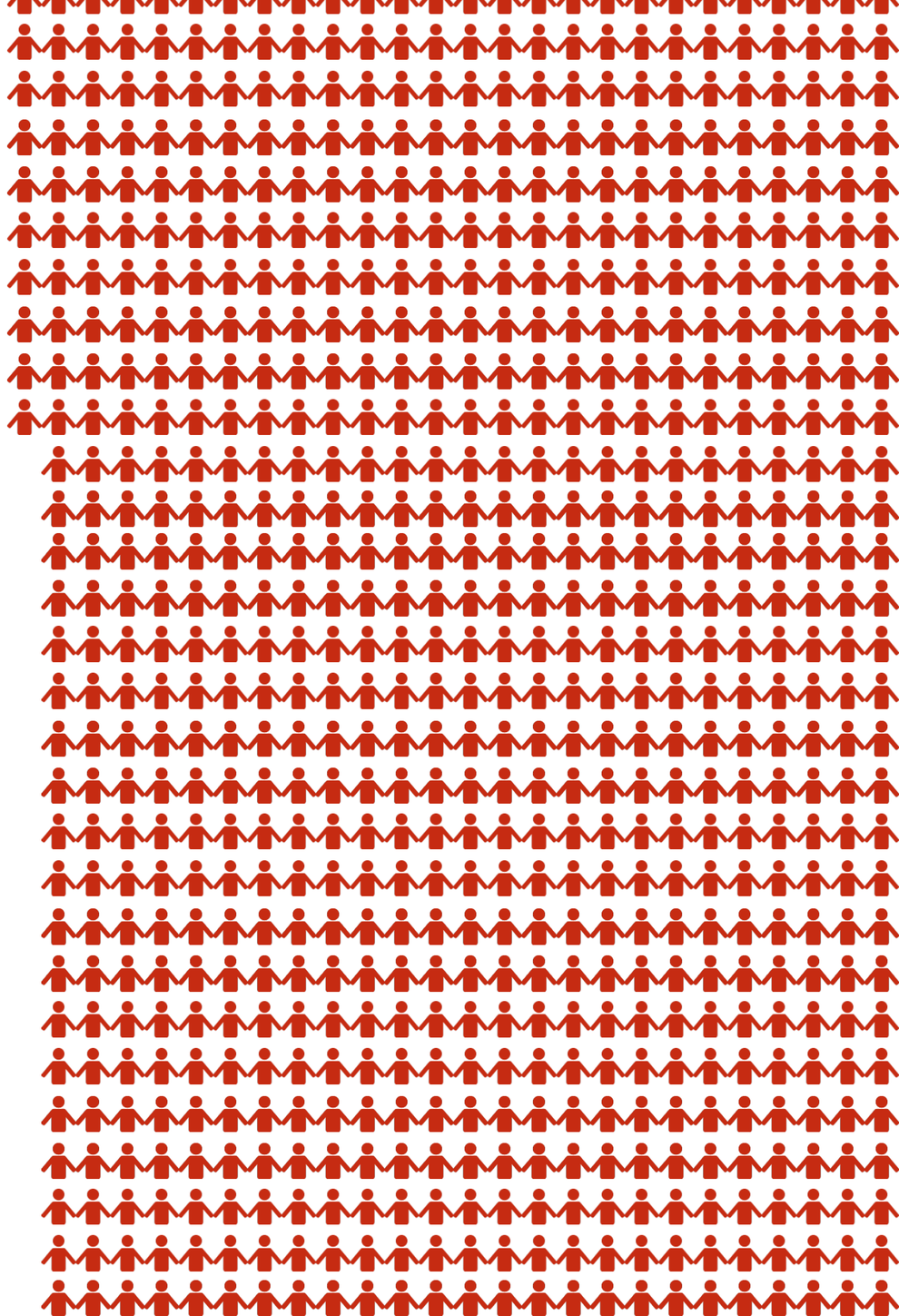
Every **2 seconds** someone in the U.S. needs blood.



SOURCE: **Red Cross Blood**
https://www.redcrossblood.org/donate-blood/how-to-donate/how-blood-donations-help/blood-needs-blood-supply.html

300

MORE PEOPLE NOW NEED BLOOD



SOURCE: **Red Cross Blood**
https://www.redcrossblood.org/donate-blood/how-to-donate/how-blood-donations-help/blood-needs-blood-supply.html

+

QUESTIONS

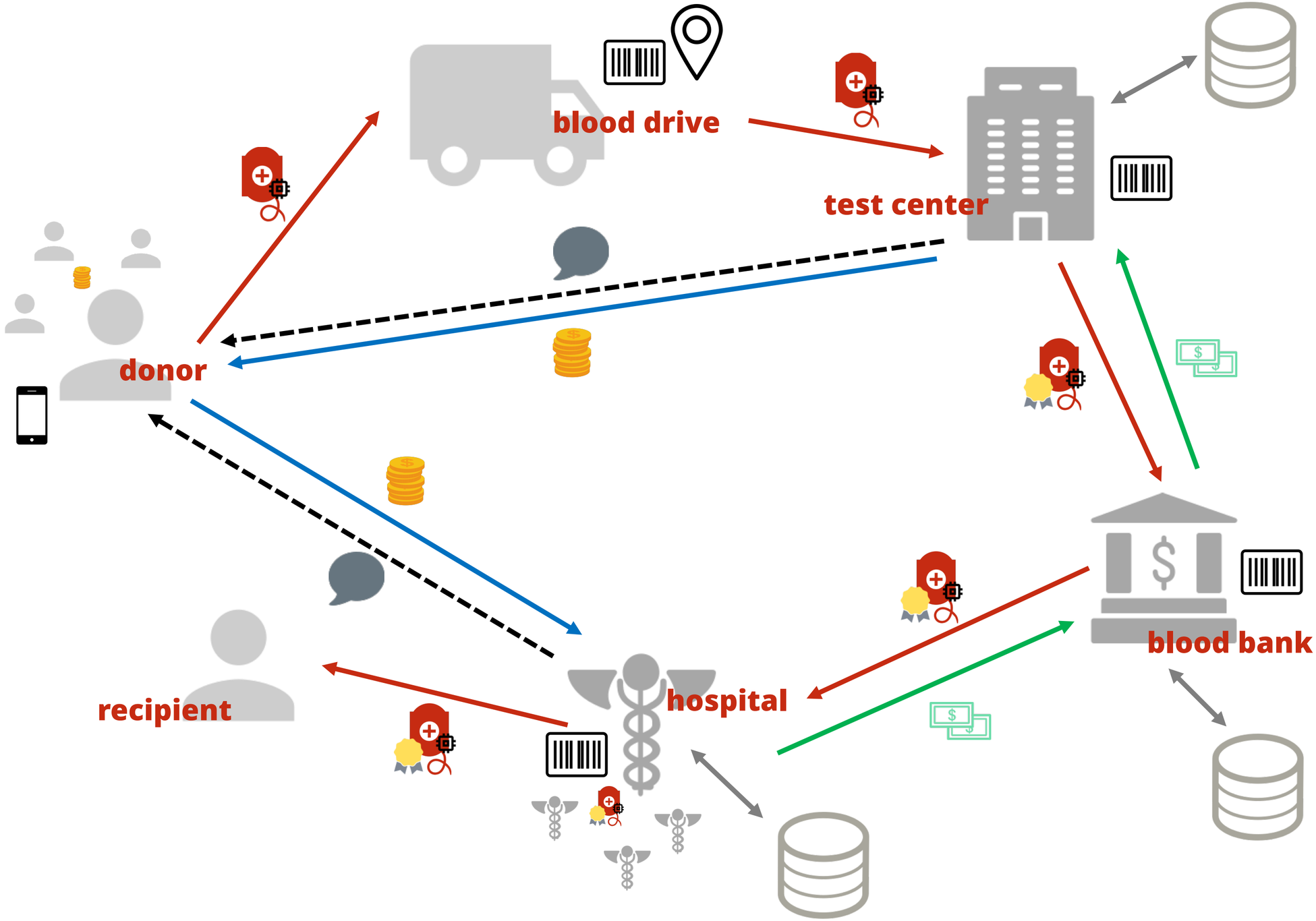
APPENDIX

```
pragma solidity ^0.4.4;

//data collection code

contract RGC {

   struct BloodDonation {
    address origin;
    uint amount;
    string bloodType;
    uint date;
    string location;
    address userid;
    mapping (uint => address) history;
    mapping (uint => uint256) timestamps ;
  }

  mapping (string => BloodDonation) donations;

  function RGC() {
    // constructor
  }

  function newDonation(uint _amount, string _bloodType, uint _date, string _uniqueID) {
    donations[_uniqueID] = BloodDonation(msg.sender, _amount, _bloodType, _date, now, 0);
    donations[_uniqueID].history[donations[_uniqueID].historyLength] = msg.sender;
    donations[_uniqueID].timestamps[donations[_uniqueID].historyLength] = now;
    donations[_uniqueID].historyLength++;

  }

  function getOrigin(string _uniqueID) constant returns (address) {
    return donations[_uniqueID].origin;
  }

  function getAmount(string _uniqueID) constant returns (uint) {
    return donations[_uniqueID].amount;
  }

  function getBloodType(string _uniqueID) constant returns (string) {
    return donations[_uniqueID].bloodType;
  }

  function getAgeRange(string _date) constant returns (uint) {
    return donations[_uniqueID].date;
  }

  function getDate(string _uniqueID) constant returns (uint) {
    return donations[_uniqueID].userid;
  }
```

```
  }

  function getHistory(string _uniqueID) constant returns (address[],uint256[]) {
    uint size = donations[_uniqueID].historyLength;
    address[] memory _history = new address[](size);
    uint256[] memory _timestamps = new uint256[](size);

    for (uint i = 0; i < donations[_uniqueID].historyLength; i++) {
      _history[i] = donations[_uniqueID].history[i];
      _timestamps[i] = donations[_uniqueID].timestamps[i];
    }
    return (_history,_timestamps);
  }
}

//oracle example code for location detection

 {
    address public oracleAddress;

    constructor (address _oracleAddress) public {
    oracleAddress = _oracleAddress;
  }

  event Locationdetection (
    string Eddystone EID,
    string time,
    string date,
    string Location,
  );

  function locationinfo (
    string Eddystone EID,
    string time,
    string date,
    string Location,
  )
  public
  {
    require(msg.sender == oracleAddress);

    emit Locationdetection (
      Eddystone EID,
      time,
      date,
      Location,
    );
  }
}
```

APPENDIX

```
//oracle example code for testing labs

    function Bloodtestingresults(bool _res) private {
        require(now >= endTime);

        require(msg.sender ==  0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB);

            if(res==true) {
                Sendtoken = true;
                Payee = donor;
            } else {
                Sendtoken = false;
            }
        }

    }

    //reward example code

    function withdraw() public {
        require(Payee != address(0x0));
        Payee.transfer(address(this).balance);
    }
```


blood drive
test center
donor
recipient
hospital
blood bank