# Blockchain for Secure IoT Firmware Updates

Ahmed Refai and Eugene Wu

# Introduction


Smart homes growing more popular; ballooning ecosystem of sensors and actuators

Need a way to efficiently update devices and patch newly discovered vulnerabilities; quickly respond to attacks like Mirai malware

Over-the-air (OTA) updates allow convenience of wirelessly patching devices

How do manufacturers efficiently, reliably, and securely distribute updates to a ballooning number of devices deployed around the globe?

# The Ethereum Blockchain


Released in 2015 - improves on Bitcoin by adding “smart contracts”

Smart contracts exist at addresses on the blockchain and implement arbitrary (Turing complete) logic programmed by developers

Consensus Algorithm: Proof of work (POW) - network fees paid using ether

Trustless, immutable, resilient

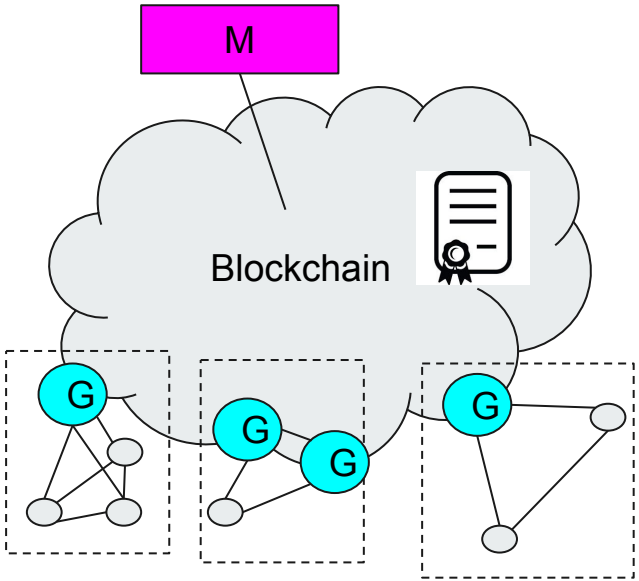
Burgeoning developer community - many tools for testing and development, implemented in JS, Python, and C++

# Goal

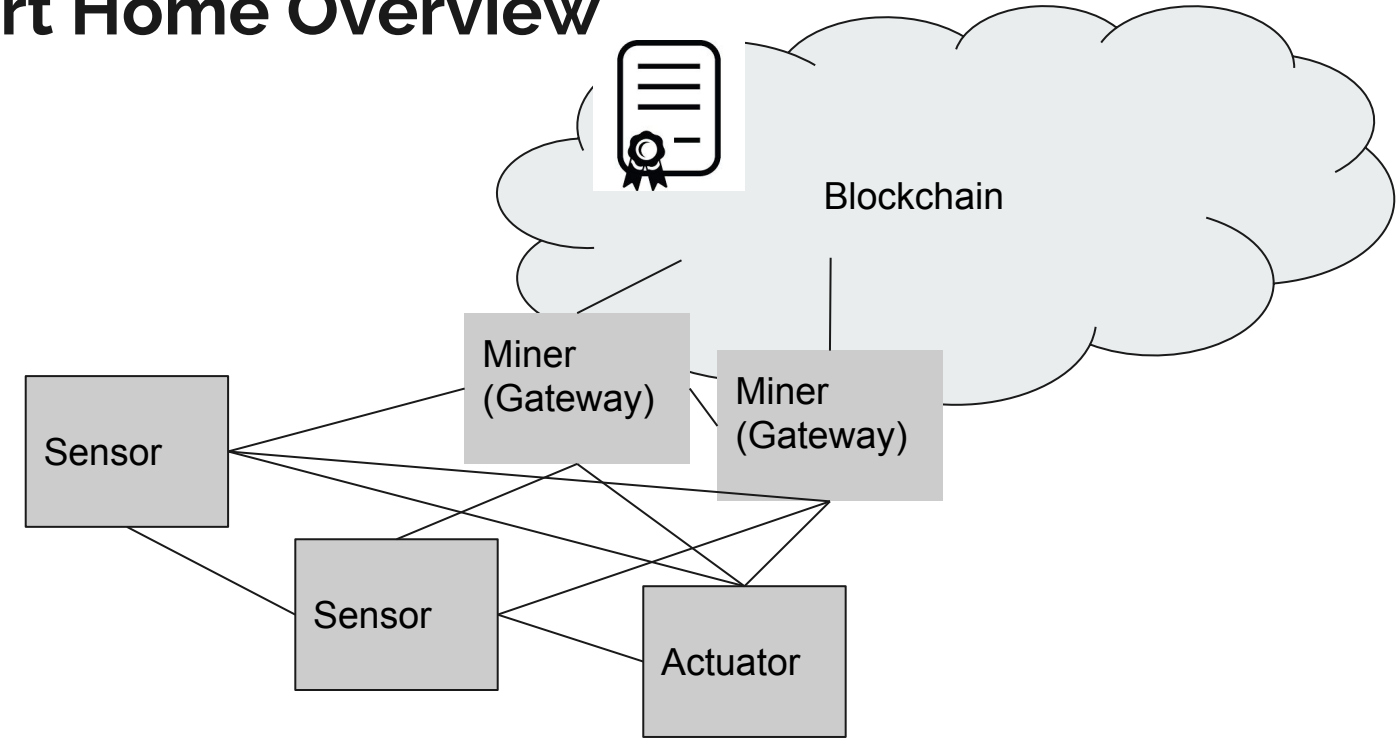
**Implement Ethereum-based identity management and firmware updates system**

- Integrate smart home “clusters” into a private blockchain managed by manufacturer
- Program smart contract to provide interface to both manufacturer and smart home nodes, executing the necessary logic
- Design smart home that interacts with information pushed by the manufacturer onto the blockchain and acts accordingly

# System Overview



# Smart Home Overview



# Development Tools

Hardware:

- Raspberry Pi Zero W
- Unix/Windows machines

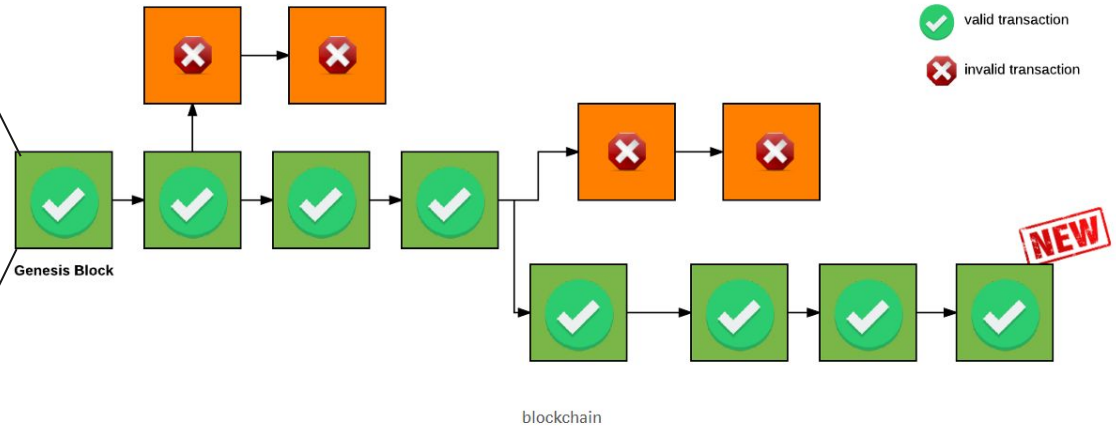
Software:

- Remix Solidity IDE
- Metamask
- Geth (Go-Ethereum)
- Node.js
- Python

# Private Blockchains

## Genesis Block (Block 0)

```
1  {
2     "config": {
3        "chainId": 1997,
4        "homesteadBlock": 0,
5        "eip155Block": 0,
6        "eip158Block": 0,
7        "byzantiumBlock": 0
8     },
9     "difficulty": "400",
10    "gasLimit": "0x8000000",
11    "alloc": {
12       "f3fad6797e82f0b2a3213b664dc4ccaaa2314866": {
13           "balance": "100000000000000000000000"
14       }
15    }
16 }
```




```
geth --rpc --networkid 9001 --rpcaddr "127.0.0.1" --rpcport 30305 --rpccorsdomain "*"
--rpcapi="db,eth,net,web3,personal,web3,miner" --nodiscover --port 30304 --datadir final_9001 2>>
final.log
```

# Smart Contracts

```
contract System {
    bytes public firmware;                          // Newest Manufacturer Firmware Binary
    address public owner;                           // Address of contract owner

    struct Node {                                   // Instantiation of IOT sensor
        string name;                                // Device Name
    }

    mapping(address => Node) public nodes;                  // Map addresses to devices
```

# Smart Contracts

```
function pushUpdate(bytes new_firmware) public {
    require(msg.sender == owner);
    firmware = new_firmware;
}

function checkForUpdate(bytes myFirmware) public view returns(bool updateRequired) {
    if (myFirmware.length != firmware.length) {
        return true;
    }
    for (uint i = 0; i < myFirmware.length && i < firmware.length; ++i) {
        if (firmware[i] != myFirmware[i]) {
            return true;
        }
    }
    return false;
}
```

## Remix Solidity IDE

browser/test.sol

```
pragma solidity ^0.4.13;

contract System {
    bytes public firmware;                          // Newest Manufacturer Firmware Binary
    address public owner;                           // Address of contract owner

    struct Node {                                   // Instantiation of IOT sensor
        string name;                                // Device Name
    }

    mapping(address => Node) public nodes;                  // Map addresses to devices

    // Alert events logged to blockchain
    // event updateSucceed(address from, string message, uint temp);
    // event updateFail(address from, string message, uint humid);

    // Functions to set up network and devices
    function System() public {                              // Constructor
        owner = msg.sender;                                 // Caller of contract is owner
    }

    function pushUpdate(bytes new_firmware) public {
        require(msg.sender == owner);
        firmware = new_firmware;
    }

    function checkForUpdate(bytes myFirmware) public view returns(bool updateRequired) {
        if (myFirmware.length != firmware.length) {
            return true;
        }
```

Compile Run Settings Analysis Debugger Support

Start to compile | Auto compile

System | Details | Publish on Swarm

Static Analysis raised 9 warning(s) that requires

System

[2] only remix transactions, scr... | Search transactions | Listen on network

## Metamask Extension

Ropsten Test Net

Account 1

0xC0615...

31.834 ETH
17073.74 USD

BUY | SEND

SENT | TOKENS

| | | |
|---|---|---|
| 49 | March 18 2018 16:50<br>0xb434BA82...a596 | 0 ETH |
| 48 | March 18 2018 16:34<br>0xb434BA82...a596 | 0 ETH |
| 47 | March 18 2018 16:32<br>Contract Deployment | 0 ETH |
| 46 | March 18 2018 16:06<br>0x0c328032...Ef76 | 0 ETH |

# Firmware Update Process

1. Manufacturer signs firmware binary file with private key, pushes to blockchain
2. Miner regularly queries contract to receive most recent version of firmware
3. Sensors and actuators regularly send firmware binary to miner
4. When the miner notices a discrepancy, it initiates an update of the device firmware

# Manufacturer

```
// Grab the latest binary
const message = fs.readFileSync('/home/refai/Desktop/hexfiles/firmware.hex', 'utf-8')

//Define signer object
const signer = crypto.createSign('sha256');

var messagepp = "";
for (var i = 0; i< message.length; i++){
    //console.log(package[i])
    if ((message[i] != '\n') && (message[i] != ":")) {
        //console.log(package[i]);
        //package[i] = '';
        //i-=1;
        messagepp += message[i];
    }
}

// load signer buffer
signer.update(messagepp);
signer.end();

const signature = signer.sign(privateKey);
const signature_hex = signature.toString('hex')
```

Signs latest firmware version

```
// Get coinbase
web3.eth.getCoinbase()
.then(function(coinbase){
    // Push new update. Device must be mining!
    web3.miner.start();
    console.log("Waiting for transaction to be mined...")
    contract.methods.pushUpdate(packagepp).send({from: coinbase, gas: 4700000})
    .then(function(receipt){
        // Check new firmware
        web3.miner.stop();
        console.log("Transaction mined!")
        contract.methods.firmware().call()
        .then(function(response){
            //console.log("New Firmware Hash is ", response);
            manufacturer_signature = response.substring(0,514);
            manufacturer_firmware = response.substring(514, response.length)

            console.log("signature:", manufacturer_signature, "\n")
            console.log("firmware:", manufacturer_firmware)
        });
    });
});
```

Pushes to blockchain

# Miner

```javascript
var cronJob = cron.schedule('*/'+interval+' * * * * *', function(){
    contract.methods.firmware().call()
    .then(function(response){
        //console.log("Current Firmware is ", response);
        // Define a verifier object
        const verifier = crypto.createVerify('sha256');
        manufacturer_signature = response.substring(2,514).trim();
        manufacturer_firmware = response.substring(514, response.length).trim()

        var signature = Buffer.from(manufacturer_signature, 'hex');
        verifier.update(manufacturer_firmware);
        verifier.end();


        // If firmware signature checks out, save firmware to miner node
        var verified = verifier.verify(publicKey, signature);
        if (verified){
            console.log("Manufactuere signature verified")
            var writepath = path.join(__dirname, "FIRMWARE.hex")
            fs.writeFile(writepath, manufacturer_firmware, function(err) {
                if(err) {
                    return console.log(err);
                }
                console.log("Latest firmware saved");
```

Verifies manufacturer signature and saves locally.

```javascript
                // Iterate through devices and check their firmware.
                // if out of date, update. If not, move on.
                for (var i = 0; i < numDevices; ++i) {
                    var filepath = path.join(__dirname, 'sensorRequests' + i.toString() + '.txt');
                    var buffer = fs.readFileSync(filepath);
                    console.log('Device', i.toString(), 'firmware:', buffer.toString());

                    if (verified){
                        if (buffer != manufacturer_firmware) {
                            console.log('Update Required for Device ' + i.toString());
                            client.scp("/home/refai/Desktop/EE209AS/Blockchain\ Project/FIRMWARE.hex", {
                                host: device_array[i],
                                username: 'pi',
                                password: '2018ee209as',
                                path: '/home/pi/Desktop/Version/FIRMWARE.hex'
                            }, function(response) {
                                //console.log(response)
                                console.log("Device successfully updated");
                            })
                        }else{
                            console.log("Device firmware up to date!")
                        }
                    }else{
                        //console.log("Manufacturer Signature not verified. Package will be discarded.")
                    }
```

Iterates through devices and updates devices that hold dated frimware

```
refai@refai-XPS-15-9560:~/Desktop/EE209AS/Blockchain Project$ node miner.js
Contract Location: 0xc21b34efb431c0a8494be88c8b76fd3018a61241
----------------------------------
----------------------------------
Manufactuere signature verified
Latest firmware saved
Device 0 firmware: deadbeef010102
Device firmware up to date!


Manufactuere signature verified
Latest firmware saved
Device 0 firmware: deadbeef010102
Device firmware up to date!


Manufactuere signature verified
Latest firmware saved
Device 0 firmware: deadbeef010102
Device firmware up to date!


Manufactuere signature verified
Latest firmware saved
Device 0 firmware: deadbeef010102
Device firmware up to date!


Manufactuere signature verified
Latest firmware saved
Device 0 firmware: deadbeef010102
Update Required for Device 0


Device successfully updated
Manufactuere signature verified
Latest firmware saved
Device 0 firmware: deadbeef01010203
Device firmware up to date!
```

```
firmware: deadbeef010102
refai@refai-XPS-15-9560:~/Desktop/EE209AS/Blockchain Project$ node manufacturer.js
Contract Location: 0xc21b34efb431c0a8494be88c8b76fd3018a61241
Waiting for transaction to be mined...
Transaction mined!
signature: 0x679d4e50a899444eabf5cb4311b1c81b51738a394362b9108557a5b0ac69ca966e09ab62bda253594a19e1e949ec
2a3eddd17237c2944315f3b229d6fc0f43bc85fbbb25101c06503ed57e482d3964ee4f0151f79c4856ba65e852433fd8967a64f3b
bf3f67185d923e881e59d2aa243ca9aedec34ad351fe4298771e9349be19e26a1627f42f80b906d1b853881e66c331fa8d9414aed
442d6f07da3a60e7a1f72eba0fb56652fa5e13ebf6b10c757fd6f891e9bfebab782f2add7f79286fe66c293597088e9ed39de3cdb
96c915b19a74a95d2113068ae9ccc26c73fcb33629ebd12e0252515ce2f6b3e7814eda13ec51362d8c5f6f5f36bd603fa2f632922


firmware: deadbeef01010203
```

```
refai@refai-XPS-15-9560:~/Desktop/EE209AS/Blockchain Project$ py3 minerInput.py
deadbeef010102
deadbeef010102
deadbeef010102
deadbeef010102
deadbeef010102
deadbeef01010203
deadbeef01010203
deadbeef01010203
```

```
> eth.getTransaction("0x373cca7cd3fd9fa1b75165a2986c24e2f7f1128d616c85b0c2932b07cd6f48ed")
{
  blockHash: "0xde4b45454eabc6286d0194ea09b1b764b5080b7395a1f699c30b57ed6c83005c",
  blockNumber: 1384,
  from: "0x15fc2f78d606900cb827025ac04617f13ab02e14",
  gas: 4700000,
  gasPrice: 18000000000,
  hash: "0x373cca7cd3fd9fa1b75165a2986c24e2f7f1128d616c85b0c2932b07cd6f48ed",
  input: "0x38382355000000000000000000000000000000000000000000000000000000000000000200000000000000000000000
000000000000000000000000000000000000000000108679d4e50a899444eabf5cb4311b1c81b51738a394362b9108557a5b0ac69ca
966e09ab62bda253594a19e1e949ec2a3eddd17237c2944315f3b229d6fc0f43bc85fbbb25101c06503ed57e482d3964ee4f0151f
79c4856ba65e852433fd8967a64f3bbf3f67185d923e881e59d2aa243ca9aedec34ad351fe4298771e9349be19e26a1627f42f80b
906d1b853881e66c331fa8d9414aed442d6f07da3a60e7a1f72eba0fb56652fa5e13ebf6b10c757fd6f891e9bfebab782f2add7f7
9286fe66c293597088e9ed39de3cdb96c915b19a74a95d2113068ae9ccc26c73fcb33629ebd12e0252515ce2f6b3e7814eda13ec5
1362d8c5f6f5f36bd603fa2f632922deadbeef0101020300000000000000000000000000000000000000000000000000",
  nonce: 27,
  r: "0xf1fbbe238447df87ea1afc8f54787529a62d89add0fe6be962d851c7da27298a",
  s: "0x66c1e2254836427c39b0c75f5fe1b83da682f3c4cdcf4e006391c99c33e527fa",
  to: "0xc21b34efb431c0a8494be88c8b76fd3018a61241",
  transactionIndex: 0,
  v: "0xfbe",
  value: 0
}
>
```

pi@raspberrypi: ~/D... | Version | pi@raspberrypi: ~/D... | 0 % | 00:03

pi@raspberrypi: ~/Desktop/Version

File Edit Tabs Help

```
pi@raspberrypi:~ $ ls
Desktop    node-v8.9.4-linux-armv6l          python_games  Videos
Documents  node-v8.9.4-linux-armv6l.tar.gz   server.py
Downloads  Pictures                          server.pyc
Music      Public                            Templates
pi@raspberrypi:~ $ cd Desktop/Version/
pi@raspberrypi:~/Desktop/Version $ ls
FIRMWARE.hex  server.py  server.pyc
pi@raspberrypi:~/Desktop/Version $ cat FIRMWARE.hex
deadbeef0101pi@raspberrypi:~/Desktop/Version $ cat FIRMWARE.hex
deadbeef0101pi@raspberrypi:~/Desktop/Version $ ls
FIRMWARE.hex  myfile.txt  server.py  server.pyc
pi@raspberrypi:~/Desktop/Version $ cat FIRMWARE.hex
deadbeef010102pi@raspberrypi:~/Desktop/Version $
```

pi@raspberrypi: ~/Desktop/Version

File Edit Tabs Help

```
192.168.0.20:54922 - - [20/Mar/2018 23:28:57] "HTTP/1.1 GET /" - 200 OK
192.168.0.20:54928 - - [20/Mar/2018 23:29:02] "HTTP/1.1 GET /" - 200 OK
192.168.0.20:54934 - - [20/Mar/2018 23:29:08] "HTTP/1.1 GET /" - 200 OK
192.168.0.20:54940 - - [20/Mar/2018 23:29:13] "HTTP/1.1 GET /" - 200 OK
192.168.0.20:54946 - - [20/Mar/2018 23:29:18] "HTTP/1.1 GET /" - 200 OK
192.168.0.20:54952 - - [20/Mar/2018 23:29:23] "HTTP/1.1 GET /" - 200 OK
192.168.0.20:54960 - - [20/Mar/2018 23:29:28] "HTTP/1.1 GET /" - 200 OK
192.168.0.20:54976 - - [20/Mar/2018 23:29:33] "HTTP/1.1 GET /" - 200 OK
192.168.0.20:54988 - - [20/Mar/2018 23:29:38] "HTTP/1.1 GET /" - 200 OK
192.168.0.20:54994 - - [20/Mar/2018 23:29:43] "HTTP/1.1 GET /" - 200 OK
192.168.0.20:55000 - - [20/Mar/2018 23:29:48] "HTTP/1.1 GET /" - 200 OK
192.168.0.20:55006 - - [20/Mar/2018 23:29:53] "HTTP/1.1 GET /" - 200 OK
192.168.0.20:55012 - - [20/Mar/2018 23:29:58] "HTTP/1.1 GET /" - 200 OK
192.168.0.20:55018 - - [20/Mar/2018 23:30:04] "HTTP/1.1 GET /" - 200 OK
192.168.0.20:55028 - - [20/Mar/2018 23:30:09] "HTTP/1.1 GET /" - 200 OK
192.168.0.20:55036 - - [20/Mar/2018 23:30:14] "HTTP/1.1 GET /" - 200 OK
192.168.0.20:55070 - - [20/Mar/2018 23:36:26] "HTTP/1.1 GET /" - 200 OK
192.168.0.20:55074 - - [20/Mar/2018 23:36:31] "HTTP/1.1 GET /" - 200 OK
192.168.0.20:55078 - - [20/Mar/2018 23:36:36] "HTTP/1.1 GET /" - 200 OK
192.168.0.20:55082 - - [20/Mar/2018 23:36:41] "HTTP/1.1 GET /" - 200 OK
192.168.0.20:55090 - - [20/Mar/2018 23:36:46] "HTTP/1.1 GET /" - 200 OK
192.168.0.20:55112 - - [20/Mar/2018 23:36:51] "HTTP/1.1 GET /" - 200 OK
192.168.0.20:55116 - - [20/Mar/2018 23:36:56] "HTTP/1.1 GET /" - 200 OK
192.168.0.20:55122 - - [20/Mar/2018 23:37:01] "HTTP/1.1 GET /" - 200 OK
192.168.0.20:55126 - - [20/Mar/2018 23:37:06] "HTTP/1.1 GET /" - 200 OK
```

4 items

Free space: 8.5 GiB (Total: 13.2 GiB)

# Weaknesses

Firmware is not encrypted, currently not suitable for protecting intellectual property

Our private blockchain untested at large scale; needs large number of miner nodes to achieve resilience

The process of replacing a smart contract is very difficult (must go through some medium besides BC)

If a miner is compromised, devices lose access to BC. Can be solved with miner redundancy

# Related/Future Work

- Distribute shared key between devices and their miner nodes to allow for secure communication
- Peer-to-peer file distribution
- Implement encryption schemes to keep firmware contents confidential
- Move to smart contract capable blockchain without currency or mining
- System by which network can come to consensus on innocuousness of new updates
- Implement Proof-of-Authority for consensus