

# A Ethereum-based Product Identification System for anti-counterfeits

Yuhang Jiang, Shashank Gupta

# MOTIVATION

- The global counterfeit products market is valued at mega-billion dollars and is growing at an unprecedented rate.
- A lot of people who buy fake branded products are not aware that the products are duplicates as they make a purchase online through respectable websites, outlets, or replicas of the manufacturer's website.
- This has an impact on both the producer and the consumer. This affects the company name, sales, revenue and increased customer churn rate.



# Goal

- To Design an anti-counterfeit System using Blockchain.
- To secure product details using a QR code.
- Provide security to the clients by offering data.
- To stop the making of fake products by conducting transparency about the products to the notice of the consumers.
- To become aware of fake products from the primary product available in the marketplace and to enhance this performance.

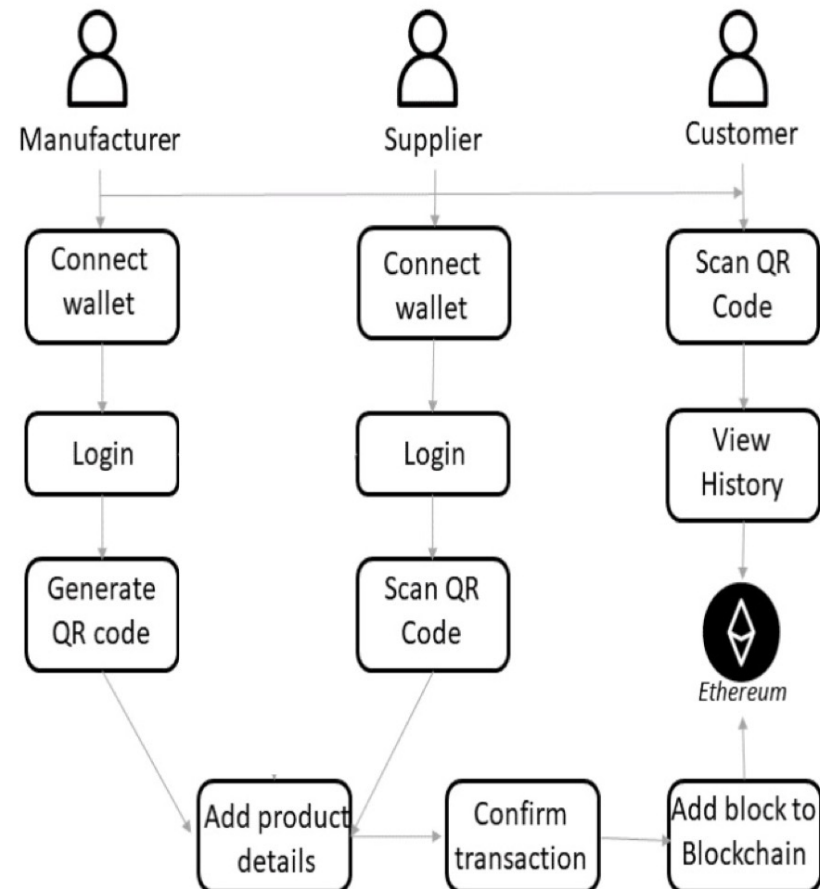
# Problem Formulation

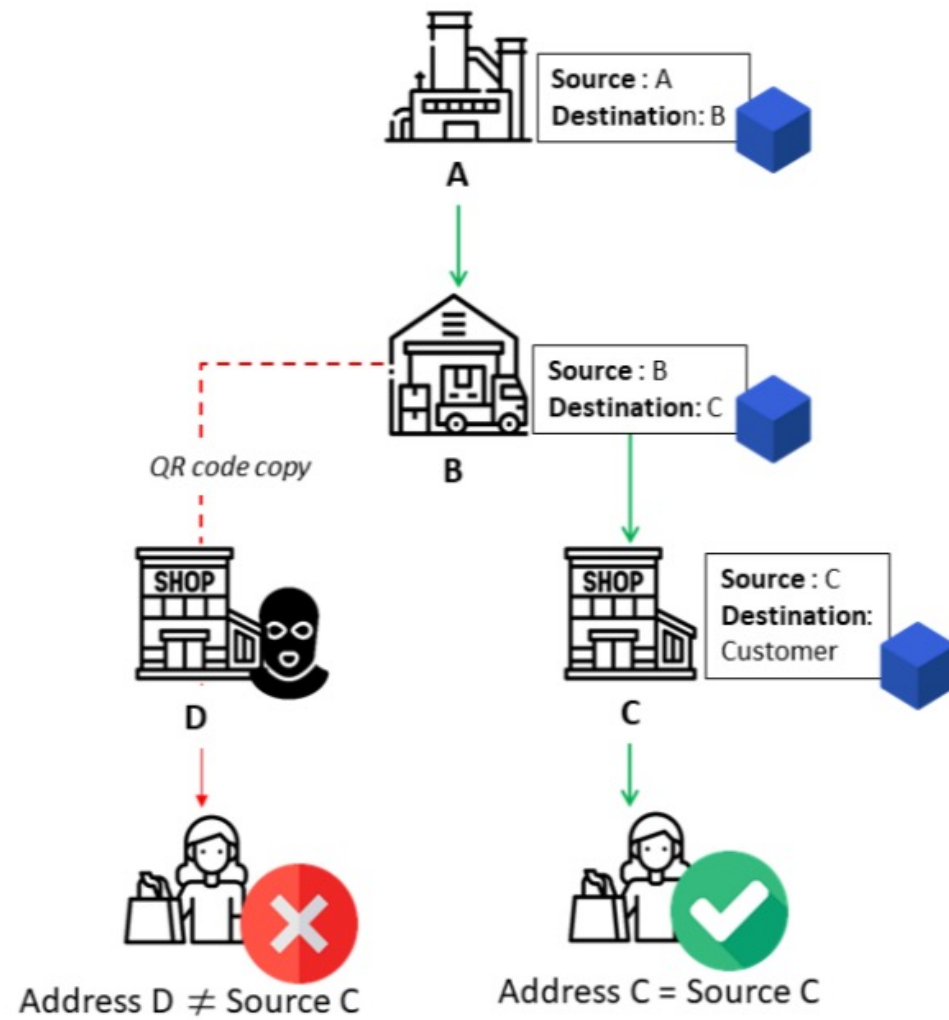
- We assume all the producers, distributors and retailers are trusted nodes.
- System is maintaining Status of product i.e., Manufacturer of product, current owner of product, and history of owners, time stamp i.e., at what time product was updated.
- Each product will be assigned a unique product ID (i.e., a string). The product ID will be used to generate a QR code.
- The product ID will be maintained on blockchain as the key to a product for tracking all history of this product.

# Problem Formulation continued

- At each status change of a product, manufacturer, distributor or retailer will update new information to the block chain.
- When the product is created, it will be labeled as “available” on the blockchain. When the product is sold to the distributor or retailer, it will be labeled as “available” and when it is being sold to the end customer, only then it will be labeled as “unavailable”.
- Only the products with “available” can be sold. Customers can check the availability and track history of the product before purchase.

# FLOW DIAGRAM





# Methodology

1. Use Meta Mask to create a wallet.
2. Adding Goerli Test network/Ganache to the wallet, a cross-client proof-of-authority testing network for Ethereum.
3. Add some dummy Ethers to our wallet using Goerli Faucet.
4. Use editor remix to write the Ethereum smart contract in Solidity.
5. Connect meta mask to your remix IDE and deploy smart contract.
6. Generate QR codes based on product id and store all the product information (color, prod type, etc.)
7. Store the same product ID on block chain.



# QR code generator

## Generator interface

URL

VCARD

TEXT

E-MAIL

SMS

WIFI

BITCOIN

TWITTER

FACEBOOK

PDF

MP3

APP STORES

IMAGES

0xE3Dc3776e032728658a919BCd4329e1D1E8080C0


[Upload any file](#) (.jpg, .pdf, .mp3, .docx, .pptx)

☐ OFF Scan tracking





</>





SCAN ME

With Logo?




FRAME NEW!






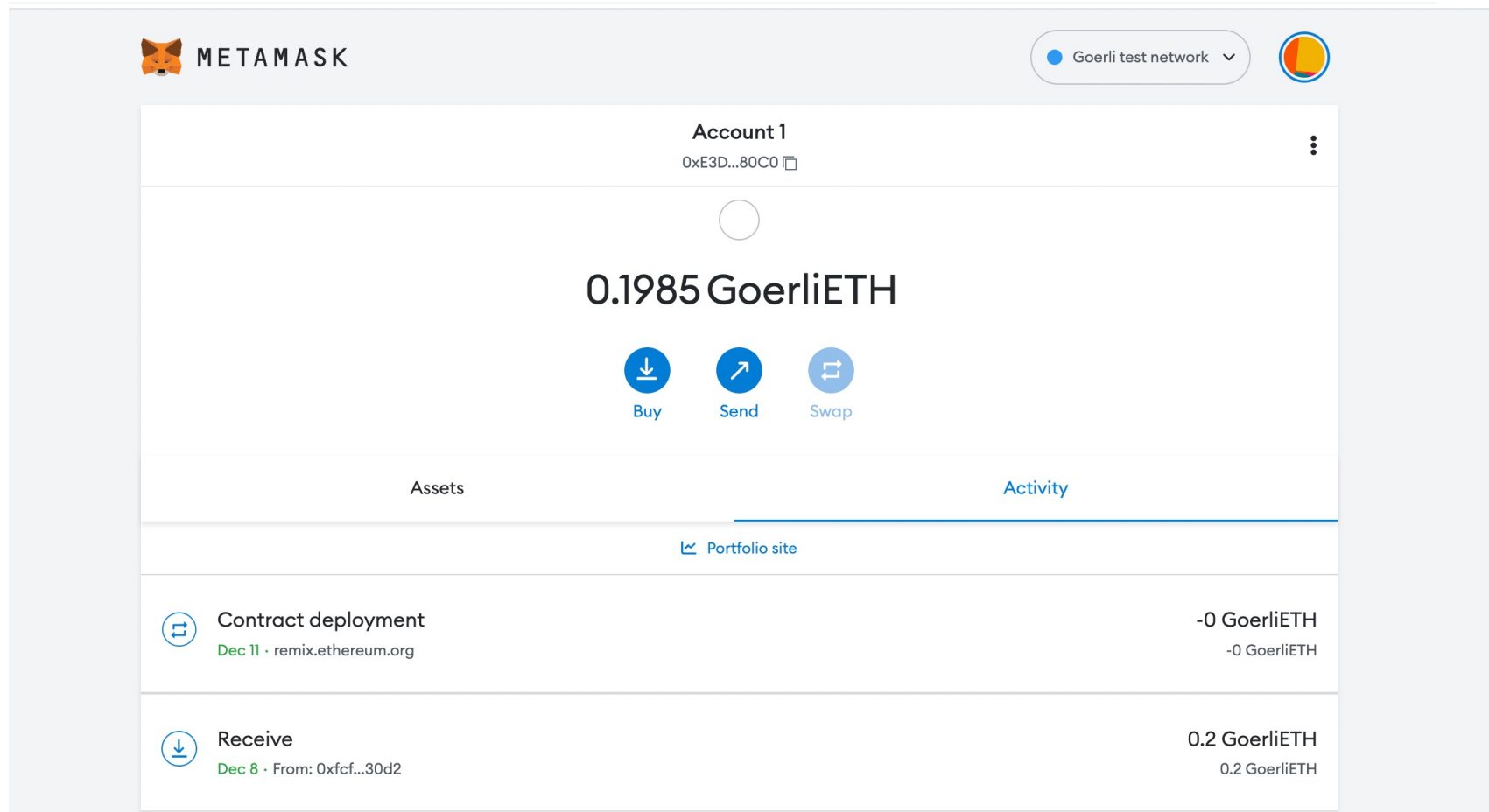
SHAPE & COLOR

LOGO

 DOWNLOAD  
JPG

 VECTOR  
SVG/EPS

# Meta Mask account



# Smart contract

- Set product

```
function setProduct(bytes32 productId, bytes32 pOwner) public{  
  
    require(!vProducts[productId]);  
    vProducts[productId] = true;  
  
    products.push(productId);  
    owners.push(pOwner);  
    pStatus.push("Available");  
  
}
```

# Smart contract

- Sell product

```
function sellProduct (bytes32 sProductId, bytes32 sDestination) public {
    bytes32 status;
    uint i;
    uint j=0;

    if(products.length>0) {
        for(i=0;i<products.length;i++) {
            if(products[i]==sProductId) {
                j=i;
            }
        }
    }

    status=pStatus[j];
    if(status=="Available" && sDestination == "Customer") {
        pStatus[j]="NA";
    }
}
```

# Smart contract

- Verify fakeness

```
function verifyFakeness(bytes32 vProductId) public view returns(bytes32,bytes32,bytes32) {

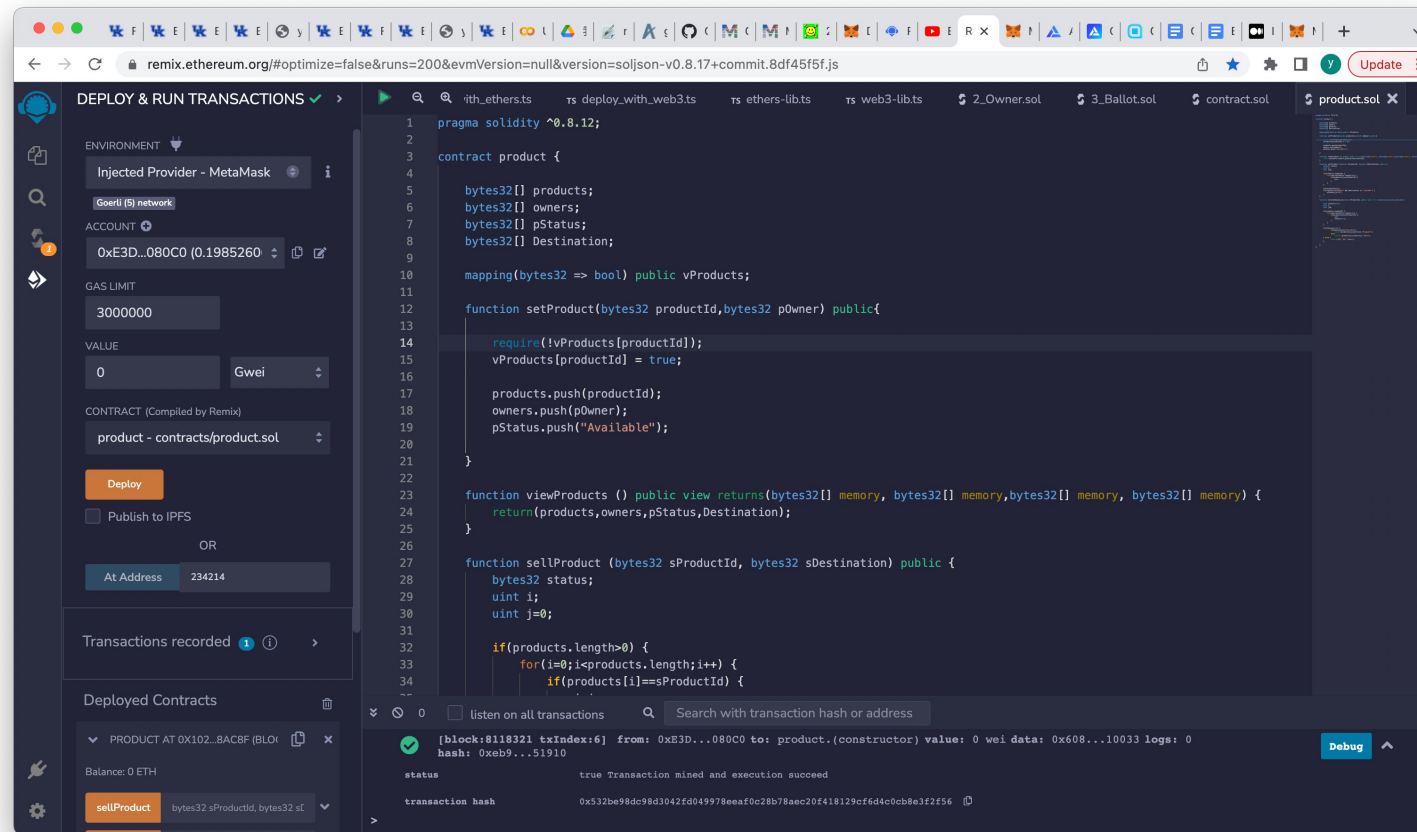
    bool status=false;
    uint i;
    uint j=0;

    if(products.length>0) {
        for(i=0;i<products.length;i++) {
            if(products[i]==vProductId) {
                j=i;
                status=true;
            }
        }
    }

    if(status==true) {
        if(pStatus[j]=="Available")
            return(products[j],owners[j],"Original");
        else
            return (products[j],owners[j],"Fake");
    } else {
        return("NA","NA","Fake");
    }

}
```

# Deploy on Goerli test net



# Evaluation for deploying smart contract

## ETH cost

Contract deployment


Status


Confirmed

[View on block explorer](#)

[Copy transaction ID](#)

From

 0xE3D...80C0



To

New contract

Transaction

Nonce	0
Amount	-0 GoerliETH
Gas Limit (Units)	589525
Gas Used (Units)	589525
Base fee (GWEI)	0.000205799
Priority fee (GWEI)	2.5
Total gas fee	0.001474 GoerliETH
Max fee per gas	0.000000003 GoerliETH
Total	0.00147393 GoerliETH

+ Activity log

+ Transaction data

# Evaluation for deploying smart contract

## Gas and transaction cost

✓ [block:8118321 txIndex:6] from: 0xE3D...080C0 to: product.(constructor) value: 0 wei data: 0x608...10033 logs: 0  
hash: 0xeb9...51910

status true Transaction mined and execution succeed

transaction hash 0x532be98dc98d3042fd049978eeaf0c28b78aec20f418129cf6d4c0cb8e3f2f56 [🔗](#)

from 0xE3Dc3776e032728658a919BCd4329e1D1E8080C0 [🔗](#)

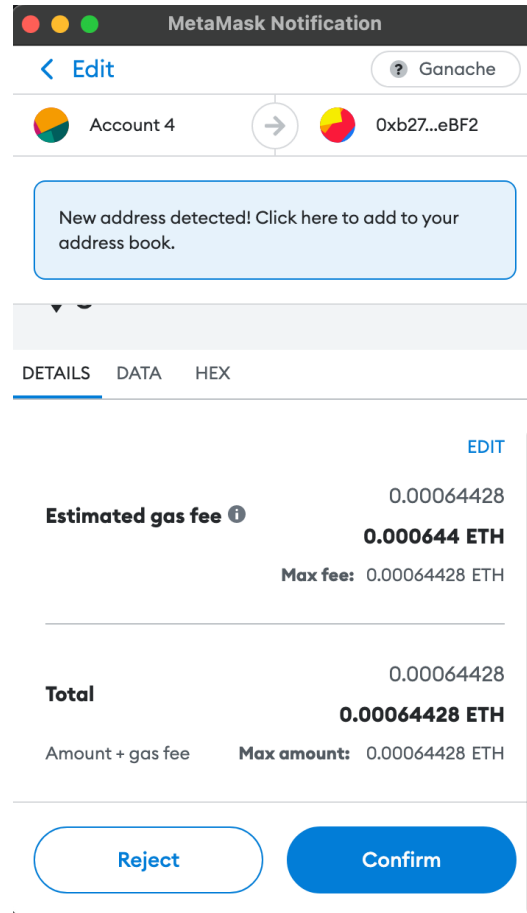
to product.(constructor) [🔗](#)

gas 589525 gas [🔗](#)

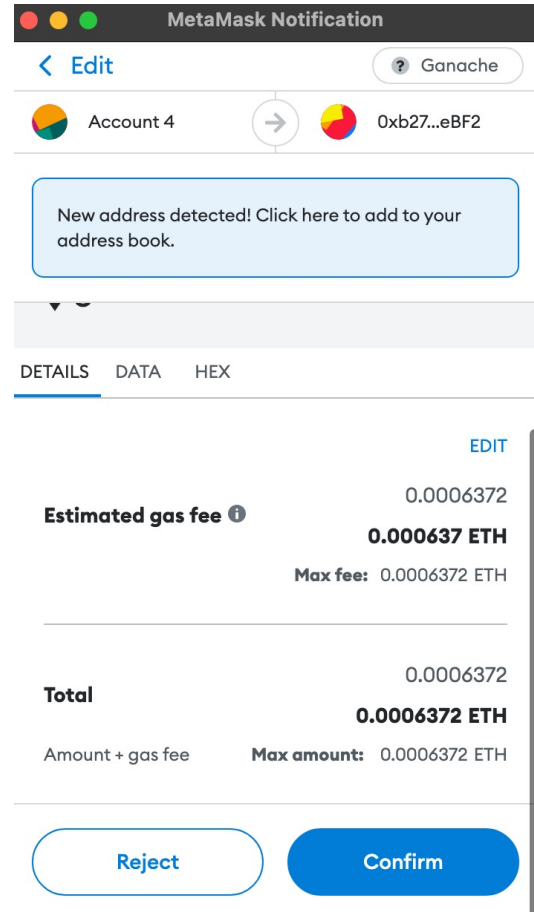
transaction cost 589525 gas [🔗](#)



# Product Registration Cost



# Product Selling Cost



# Conclusion

- We built our own smart contract and deployed on Goerli Testnet using Remix IDE.
- We show that blockchain can be used as an anti-counterfeit system for maintaining product status and history.
- We evaluated our blockchain by ETH cost, gas price and transaction cost for deploying our smart contract.

Thank You 😊