曾彬輝

B05902099

1. Screenshots

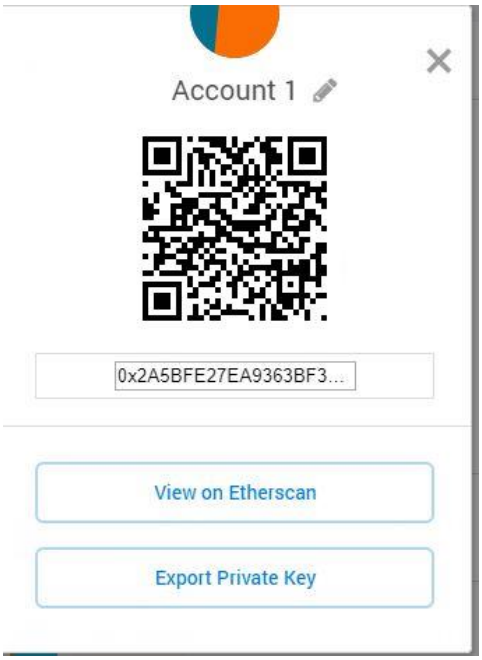

Figure 1.1

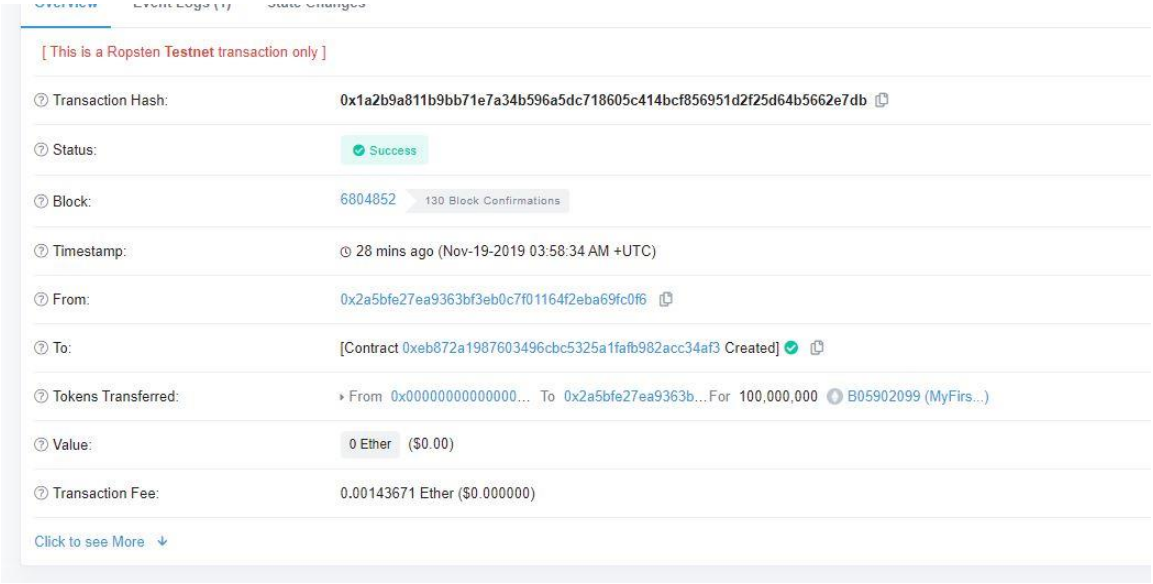

Figure 1.2

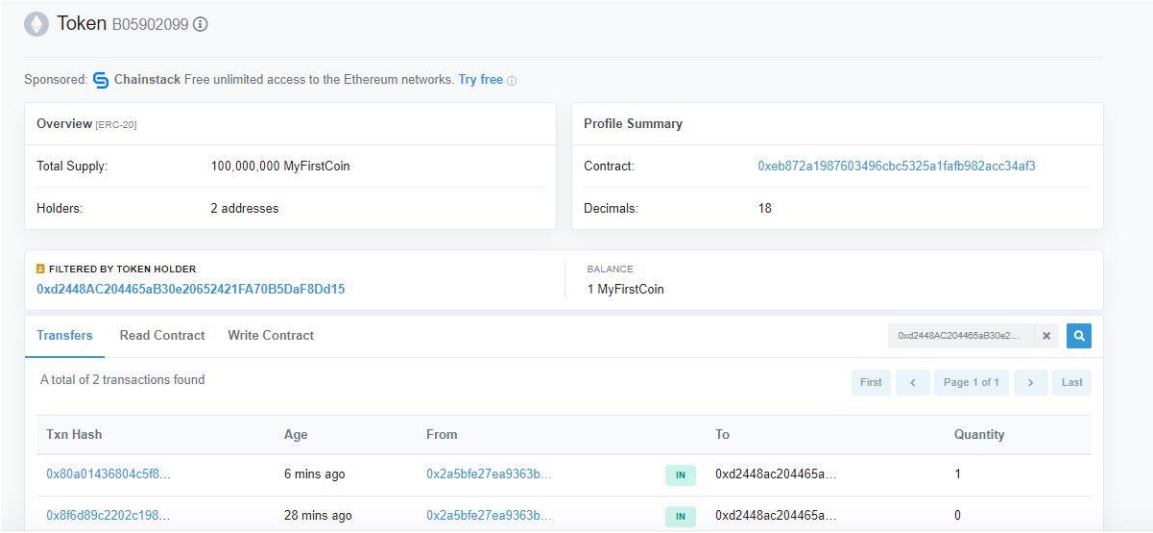

Figure 1.3

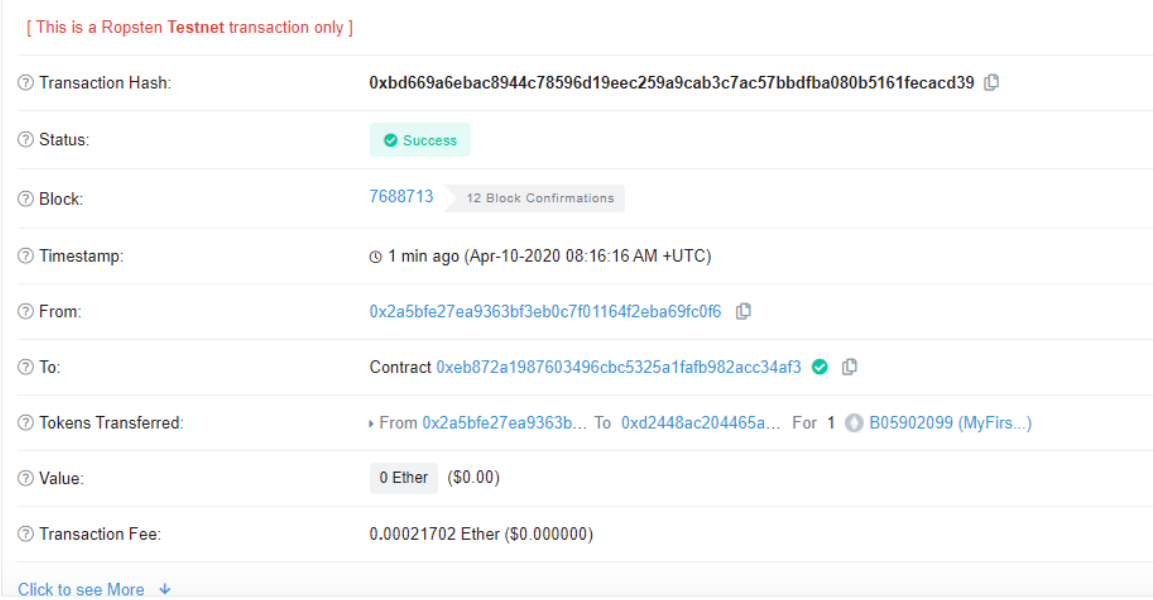

Figure 1.4

I used the Token created last year, and send another token to the given address.

Figure 1.1 shows my address (Metamask). **Figure 1.2** shows token creation. **Figure 1.3** shows my token details (transaction & contract). **Figure 1.4** shows transaction, sending **1 Token B05902099** to the given address.

2. Contents of the code (ERC20 Protocol)

a) //TotalSupply

```
function totalSupply() public constant returns (uint) {
    return _totalSupply  - balances[address(0)];
  }
```

b) //BalanceOf

```
function balanceOf(address tokenOwner) public constant returns (uint balance) {
    return balances[tokenOwner];
  }
```

c) //Allowance

```
function allowance(address tokenOwner, address spender) public constant
returns (uint remaining) {
    return allowed[tokenOwner][spender];
  }
```

d) //transfer

```
function transfer(address to, uint tokens) public returns (bool success) {
    balances[msg.sender] = safeSub(balances[msg.sender], tokens);
    balances[to] = safeAdd(balances[to], tokens);
    emit Transfer(msg.sender, to, tokens);
    return true;
}
```

e) //Approve

```
function approve(address spender, uint tokens) public returns (bool success) {
    allowed[msg.sender][spender] = tokens;
    emit Approval(msg.sender, spender, tokens);
    return true;
  }
```

f) //transferFrom

```
function transferFrom(address from, address to, uint tokens) public returns (bool success) {
        balances[from] = safeSub(balances[from], tokens);
        allowed[from][msg.sender] = safeSub(allowed[from][msg.sender], tokens);
        balances[to] = safeAdd(balances[to], tokens);
        emit Transfer(from, to, tokens);
        return true;
    }
```

g) //Approve and call: for spender to transferFrom(...) tokens from the token owner's account, and then receiveApproval is executed.

```
function approveAndCall(address spender, uint tokens, bytes data) public returns (bool success) {
        allowed[msg.sender][spender] = tokens;
        emit Approval(msg.sender, spender, tokens);
        ApproveAndCallFallBack(spender).receiveApproval(msg.sender, tokens, this, data);
        return true;
    }
```

h) //ReceiveApproval: Function to receive approval and execute function in one call

```
contract ApproveAndCallFallBack {
    function receiveApproval(address from, uint256 tokens, address token, bytes data) public;
}
```

i) //ReceiveApproval: Function to receive approval and execute function in one call

```
contract ApproveAndCallFallBack {
    function receiveApproval(address from, uint256 tokens, address token, bytes data) public;
}
```

```
//MY CONTRACT CODE
contract B05902099 is ERC20Interface, Owned, SafeMath {
    string public symbol;
    string public  name;
    uint8 public decimals;
    uint public _totalSupply;

    mapping(address => uint) balances;
    mapping(address => mapping(address => uint)) allowed;

    // Constructor
    constructor() public {
        symbol = "B05902099";
        name = "B05902099";
        decimals = 18;
        _totalSupply = 100000000000000000000000000;
        balances[0x2A5BFE27EA9363BF3EB0c7F01164F2eBa69FC0F6] =
_totalSupply;
        emit Transfer(address(0),
0x2A5BFE27EA9363BF3EB0c7F01164F2eBa69FC0F6, _totalSupply);
    }
    // Total supply
    function totalSupply() public constant returns (uint) {
        return _totalSupply  - balances[address(0)];
    }
    // Get the token balance for account tokenOwner
    function balanceOf(address tokenOwner) public constant returns (uint
balance) {
        return balances[tokenOwner];
```

```
    }

    // Transfer the balance from token owner's account to to account
    // - Owner's account must have sufficient balance to transfer
    // - 0 value transfers are allowed
    function transfer(address to, uint tokens) public returns (bool success) {
        balances[msg.sender] = safeSub(balances[msg.sender], tokens);
        balances[to] = safeAdd(balances[to], tokens);
        emit Transfer(msg.sender, to, tokens);
        return true;
    }

    // Token owner can approve for spender to transferFrom(...) tokens
    // from the token owner's account
    // https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-
standard.md
    // recommends that there are no checks for the approval double-spend attack
    // as this should be implemented in user interfaces
    function approve(address spender, uint tokens) public returns (bool success) {
        allowed[msg.sender][spender] = tokens;
        emit Approval(msg.sender, spender, tokens);
        return true;
    }
```

```
    // Transfer tokens from the from account to the to account
    // The calling account must already have sufficient tokens approve(...)-d
    // for spending from the from account and
    // - From account must have sufficient balance to transfer
    // - Spender must have sufficient allowance to transfer
    // - 0 value transfers are allowed
    function transferFrom(address from, address to, uint tokens) public returns (bool success) {
        balances[from] = safeSub(balances[from], tokens);
        allowed[from][msg.sender] = safeSub(allowed[from][msg.sender], tokens);
        balances[to] = safeAdd(balances[to], tokens);
        emit Transfer(from, to, tokens);
        return true;
    }


    // Returns the amount of tokens approved by the owner that can be
    // transferred to the spender's account
    function allowance(address tokenOwner, address spender) public constant returns (uint remaining) {
        return allowed[tokenOwner][spender];
    }
```

```
    // Token owner can approve for spender to transferFrom(...) tokens
    // from the token owner's account. The spender contract function
    // receiveApproval(...) is then executed
    function approveAndCall(address spender, uint tokens, bytes data) public
returns (bool success) {
        allowed[msg.sender][spender] = tokens;
        emit Approval(msg.sender, spender, tokens);
        ApproveAndCallFallBack(spender).receiveApproval(msg.sender, tokens,
this, data);
        return true;
    }


    // Don't accept ETH
    function () public payable {
        revert();
    }


    // Owner can transfer out any accidentally sent ERC20 tokens
    function transferAnyERC20Token(address tokenAddress, uint tokens) public
onlyOwner returns (bool success) {
        return ERC20Interface(tokenAddress).transfer(owner, tokens);
    }
```

```
//Safe math
contract SafeMath {
    function safeAdd(uint a, uint b) public pure returns (uint c) {
        c = a + b;
        require(c >= a);
    }
    function safeSub(uint a, uint b) public pure returns (uint c) {
        require(b <= a);
        c = a - b;
    }
    function safeMul(uint a, uint b) public pure returns (uint c) {
        c = a * b;
        require(a == 0 || c / a == b);
    }
    function safeDiv(uint a, uint b) public pure returns (uint c) {
        require(b > 0);
        c = a / b;
    }
}
```

```
// ERC Token Standard #20 Interface

contract ERC20Interface {

    function totalSupply() public constant returns (uint);

    function balanceOf(address tokenOwner) public constant returns (uint
balance);

    function allowance(address tokenOwner, address spender) public constant
returns (uint remaining);

    function transfer(address to, uint tokens) public returns (bool success);

    function approve(address spender, uint tokens) public returns (bool success);

    function transferFrom(address from, address to, uint tokens) public returns
(bool success);


    event Transfer(address indexed from, address indexed to, uint tokens);

    event Approval(address indexed tokenOwner, address indexed spender, uint
tokens);

}
```