Auction

拍賣

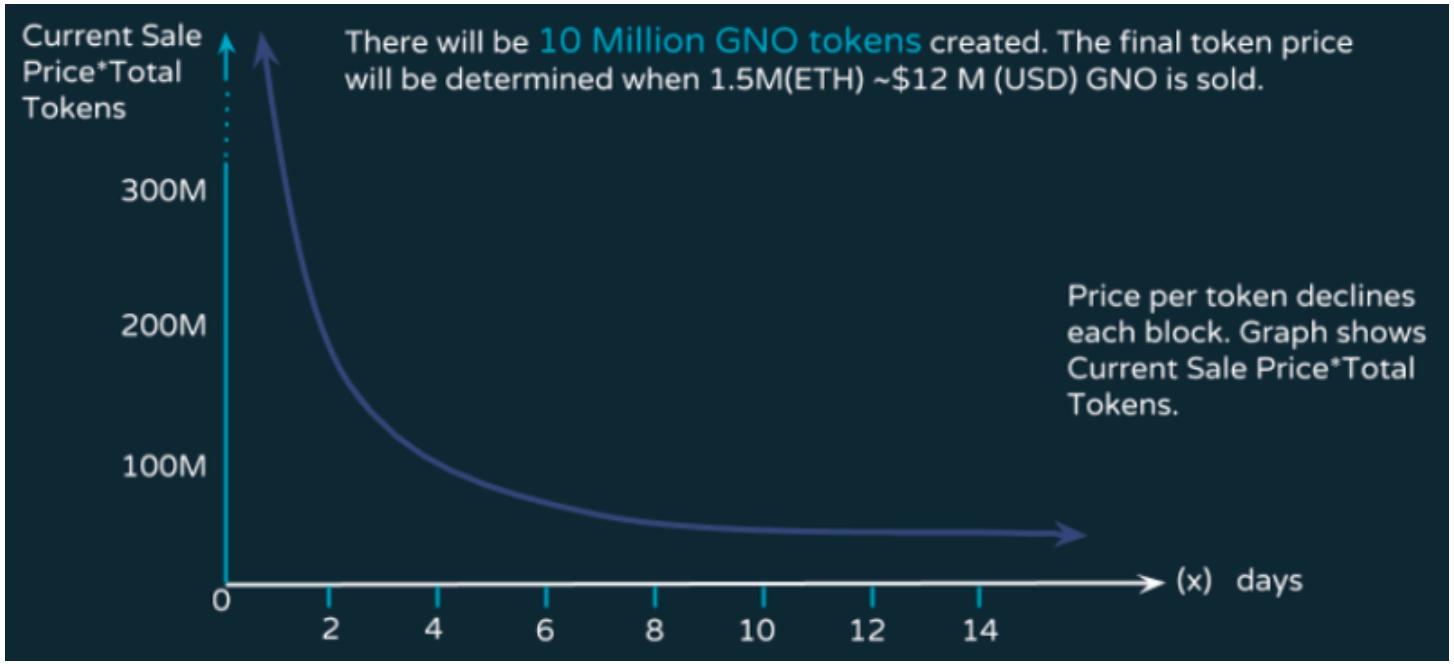
Image via https://bcbookprizes.ca/summer-2018-online-auction/

拍賣

- 頻譜拍賣
- 不動產拍賣
- 名畫古董拍賣

Gnosis (12 mins)

- Capped sale
- The reverse Dutch auction
- Fear of missing out (FOMO)
- The sale reached its cap of $12.5 million when it was only selling about 5% of all tokens



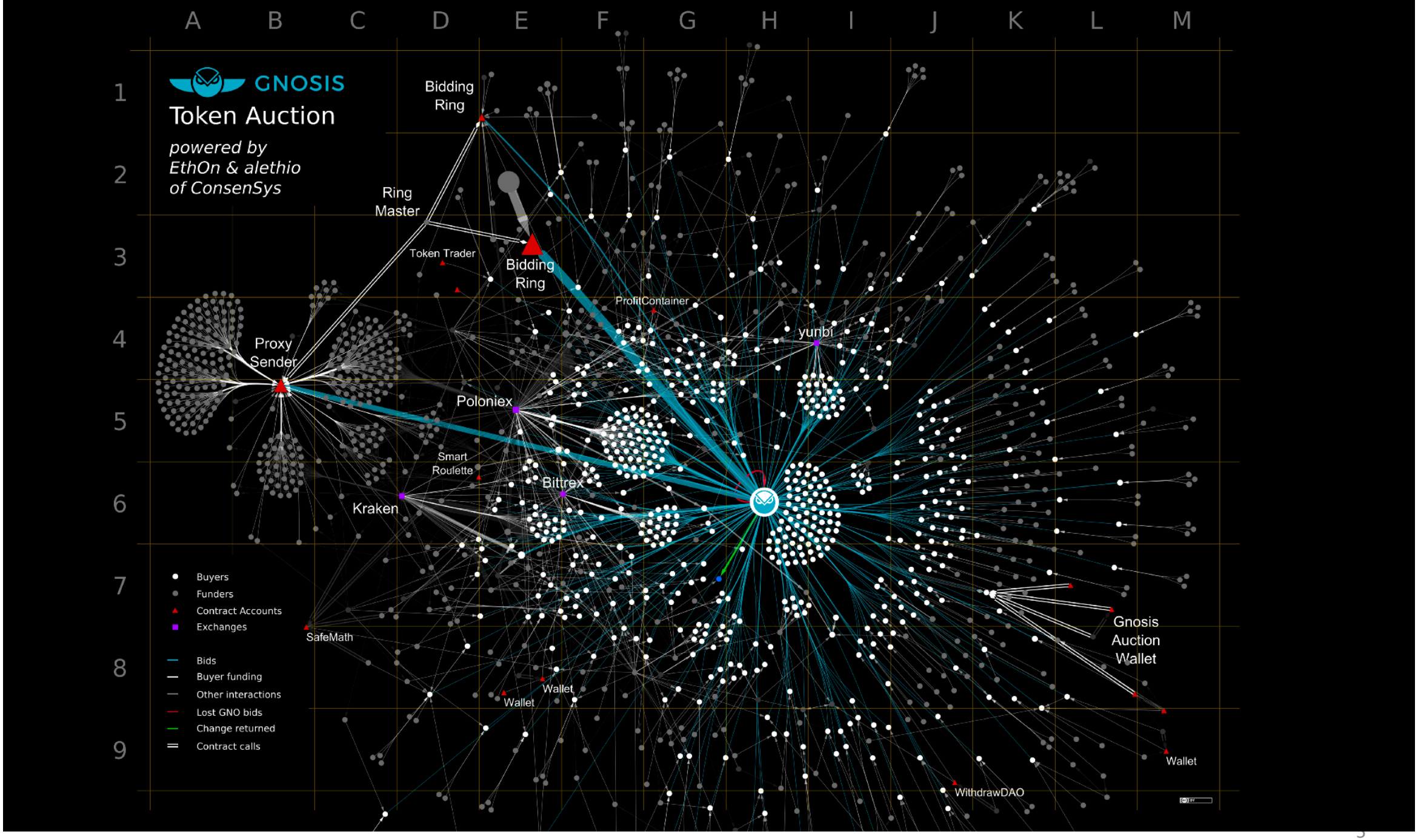
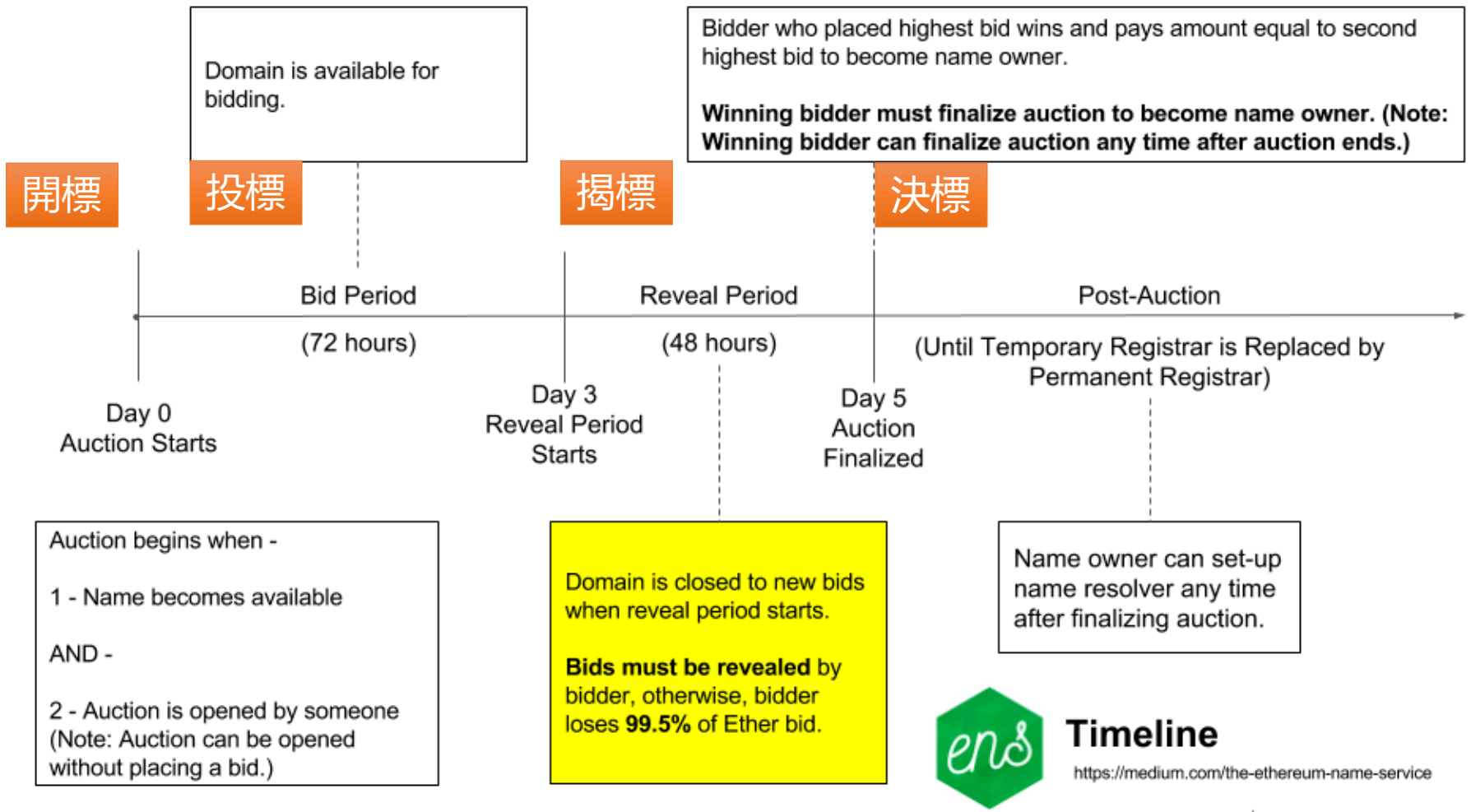


Image via https://media.consensys.net/the-gnosis-token-auction-9c2f59d2387

Image via [Newsfeed.org]

Timeline, Ethereum Name Service (ENS) Bid



https://medium.com/the-ethereum-name-service/a-visual-timeline-of-an-ethereum-name-service-bid-20c368865f56

ENS

- 官網
  - https://ens.domains
- 合約原始碼
  - https://github.com/ethereum/ens/
- 文件
  - http://docs.ens.domains/en/latest/userguide.html#registering-a-name-with-the-auction-registrar
- 市場
  - https://enslisting.com/

Simple Open Auction

- Everyone can send their bids during a bidding period.
- The bids already include sending money / ether in order to bind the bidders to their bid.
- If the highest bid is raised, the previously highest bidder gets her money back.

https://solidity.readthedocs.io/en/v0.4.25/solidity-by-example.html

```
pragma solidity ^0.4.22;

contract SimpleAuction {
    // Absolute unix timestamps or time periods in seconds.
    address public beneficiary;
    uint public auctionClose;

    // Current state of the auction.
    address public highestBidder;
    uint public highestBid;

    // Allowed withdrawals of previous bids
    mapping(address => uint) pendingReturns;

    // Set to true at the end, disallows any change
    bool ended;

    // Events that will be fired on changes.
    event HighestBidIncreased(address bidder, uint amount);
    event AuctionEnded(address winner, uint amount);

    /// Create a simple auction
    constructor(
        uint _biddingTime,
        address _beneficiary
    ) public {
        beneficiary = _beneficiary;
        auctionClose = now + _biddingTime;
    }
```

```
    /// Bid on the auction with the value
    function bid() public payable {
        // Revert the call if the bidding period is over.
        require(
            now <= auctionClose,
            "Auction already ended."
        );

        // If the bid is not higher, send the money back.
        require(
            msg.value > highestBid,
            "There already is a higher bid."
        );

        if (highestBid != 0) {
            pendingReturns[highestBidder] += highestBid;
        }

        highestBidder = msg.sender;
        highestBid = msg.value;
        emit HighestBidIncreased(msg.sender, msg.value);
    }
```

```
    /// Withdraw a bid that was overbid.
    function withdraw() public returns (bool) {
        uint amount = pendingReturns[msg.sender];
        if (amount > 0) {
            pendingReturns[msg.sender] = 0;

            if (!msg.sender.send(amount)) {
                pendingReturns[msg.sender] = amount;
                return false;
            }
        }
        return true;
    }

    /// End the auction and send the highest bid to the beneficiary.
    function auctionEnd() public {
        // 1. Conditions
        require(now >= auctionClose, "Auction not yet ended.");
        require(!ended, "auctionEnd has already been called.");

        // 2. Effects
        ended = true;
        emit AuctionEnded(highestBidder, highestBid);

        // 3. Interaction
        beneficiary.transfer(highestBid);
    }
}
```

DEMO

https://gist.github.com/changwu-tw/35049b508cad8cdf4d5c6fa98b38be45

問題

- 投標透明
- 時間壓力

Blind Auction

- 盲拍
- 標金隱藏
- 投標時間過才揭標

- Creating a blind auction on a transparent computing platform might sound like a contradiction, but cryptography comes to the rescue.

https://solidity.readthedocs.io/en/v0.4.25/solidity-by-example.html

```
pragma solidity >0.4.23 <0.5.0;

contract BlindAuction {
    struct Bid {
        bytes32 blindedBid;
        uint deposit;
    }

    address public beneficiary;
    uint public biddingEnd;
    uint public revealEnd;
    bool public ended;

    mapping(address => Bid[]) public bids;

    address public highestBidder;
    uint public highestBid;

    mapping(address => uint) pendingReturns;

    event AuctionEnded(address winner, uint highestBid);

    modifier onlyBefore(uint _time) { require(now < _time); _; }
    modifier onlyAfter(uint _time) { require(now > _time); _; }
```

```
    constructor(
        uint _biddingTime,
        uint _revealTime,
        address _beneficiary
    ) public {
        beneficiary = _beneficiary;
        biddingEnd = now + _biddingTime;
        revealEnd = biddingEnd + _revealTime;
    }

    /// Place a blinded bid with `_blindedBid` = keccak256(value,
    /// fake, secret).
    /// The sent ether is only refunded if the bid is correctly
    /// revealed in the revealing phase. The bid is valid if the
    /// ether sent together with the bid is at least "value" and
    /// "fake" is not true. Setting "fake" to true and sending
    /// not the exact amount are ways to hide the real bid but
    /// still make the required deposit. The same address can
    /// place multiple bids.
    function bid(bytes32 _blindedBid)
        public
        payable
        onlyBefore(biddingEnd)
    {
        bids[msg.sender].push(Bid({
        blindedBid: _blindedBid,
        deposit: msg.value
        }));
    }
```

```
    function reveal(/// Reveal your blinded bids.
        uint[] _values,
        bool[] _fake,
        bytes32[] _secret
    )
        public
        onlyAfter(biddingEnd)
        onlyBefore(revealEnd)
    {
        uint length = bids[msg.sender].length;
        require(_values.length == length);
        require(_fake.length == length);
        require(_secret.length == length);

        uint refund;
        for (uint i = 0; i < length; i++) {
            Bid storage bid = bids[msg.sender][i];
            (uint value, bool fake, bytes32 secret) = (_values[i], _fake[i], _secret[i]);
            if (bid.blindedBid != keccak256(value, fake, secret)) {
                // Bid was not actually revealed. Do not refund deposit.
                continue;
            }
            refund += bid.deposit;
            if (!fake && bid.deposit >= value) {
                if (placeBid(msg.sender, value))
                    refund -= value;
            }
            // Make it impossible for the sender to re-claim the same deposit.
            bid.blindedBid = bytes32(0);
        }
        msg.sender.transfer(refund);
    }
```

```
    // can only be called from the contract itself or derived contracts
    function placeBid(address bidder, uint value) internal
        returns (bool success)
    {
        if (value <= highestBid) {
            return false;
        }
        if (highestBidder != 0) {
            // Refund the previously highest bidder.
            pendingReturns[highestBidder] += highestBid;
        }
        highestBid = value;
        highestBidder = bidder;
       return true;
    }

    /// Withdraw a bid that was overbid.
    function withdraw() public {
        uint amount = pendingReturns[msg.sender];
        if (amount > 0) {
            // It is important to set this to zero because the recipient
            // can call this function again as part of the receiving call
            // before `transfer` returns (see the remark above about
            // conditions -> effects -> interaction).
            pendingReturns[msg.sender] = 0;

            msg.sender.transfer(amount);
        }
    }
```

```
    /// End the auction and send the highest bid
    /// to the beneficiary.
    function auctionEnd()
        public
        onlyAfter(revealEnd)
    {
        require(!ended);
        emit AuctionEnded(highestBidder, highestBid);
        ended = true;
        beneficiary.transfer(highestBid);
    }
}
```

混淆金

```
_blindedBid = keccak256(value, fake, secret)
```

- sha3(10 ETH, True, 1)
- value = '0000000000000000000000000000000000000000000x8ac7230489e80000'
- fake   = '0000000000000000000000000000000000000000000000000000000000000001'
- secret = '0000000000000000000000000000000000000000000000000000000000000001’

- 如果不加上混淆金的話, 雖然投標時沒有直接給入標金金額, 但是透過觀察合約地址的餘額, 可以利用觀察得出金額
- > eth.getBalance(blindauction.address)
- 3100000000000000000

- secret 的選擇也很重要, 如果不設定適當的 secret, 可能會受到 rainbow table 攻擊

Proof of Existence

存在性證明

Proof of Existence, version 1

一個證明

```
pragma solidity ^0.4.7;

// Proof of Existence contract, version 1
contract ProofOfExistence1 {
    bytes32 public proof;

    function ProofOfExistence1() {
    }

    // 公證
    function notarize(string document) {
        proof = calculateProof(document);
    }

    // SHA256
    function calculateProof(string document) constant returns(bytes32) {
        return sha256(document);
    }
}
```

Proof of Existence, version 2

```
pragma solidity ^0.4.7;

contract ProofOfExistence2 {
    bytes32[] private proofs;

    function storeProof(bytes32 proof) {
        proofs.push(proof);
    }

    function notarize(string document) {
        var proof = calculateProof(document);
        storeProof(proof);
    }

    function calculateProof(string document) constant returns (bytes32) {
        return sha256(document);
    }

    function checkDocument(string document) constant returns (bool) {
        var proof = calculateProof(document);
        return hasProof(proof);
    }

    function hasProof(bytes32 proof) constant returns (bool) {
        for (var i = 0; i < proofs.length; i++) {
            if (proofs[i] == proof) {
                return true;
            }
        }
        return false;
    }
}
```

Proof of Existence, version 3

```
pragma solidity ^0.4.7;

contract ProofOfExistence3 {
    mapping (bytes32 => bool) private proofs;

    function storeProof(bytes32 proof) {
        proofs[proof] = true;
    }

    function notarize(string document) {
        var proof = calculateProof(document);
        storeProof(proof);
    }

    function calculateProof(string document) constant returns (bytes32) {
        return sha256(document);
    }

    function checkDocument(string document) constant returns (bool) {
        var proof = calculateProof(document);
        return hasProof(proof);
    }

    function hasProof(bytes32 proof) constant returns (bool) {
        return proofs[proof];
    }
}
```

Improvement

- Blockcerts JSON Schema
- Merkle proof

Projects

- https://ethertify.com/
- https://www.blockcerts.org/ (MIT)
- https://chainy.info/
  - Source code: https://github.com/EverexIO/Chainy