batogb

Privacy in The Blockchain Era

A soft introduction to zero knowledge proofs

Bogdan Batog

batogb

Simple Marketing Real Story

Email Marketing

- Provider had signed agreements from a large demographic base
- Would promote our services to relevant profiles

Asks:

- For our own full list of user emails, to be used as “suppression”

Alarm:

- We have to fully trust them with our data, have no means of verifying the service was provided and we pay them

batogb

How We Protected Our Users Data

Email Marketing

- Provider had signed agreements from a large demographic base
- Would promote our services to relevant profiles

Solution: Send them a list of ***hashed*** email addresses

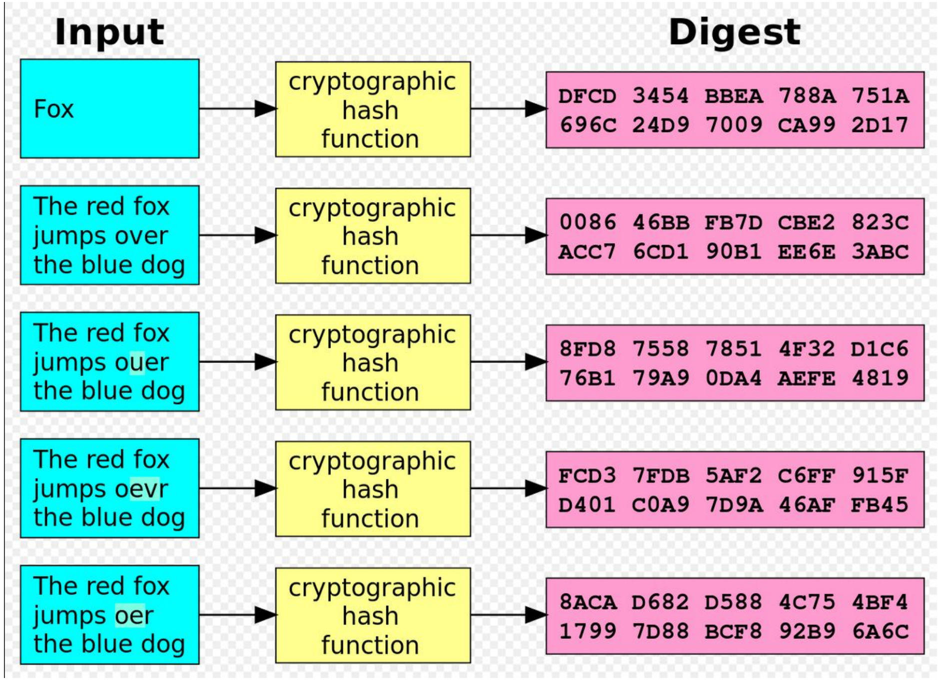
+ Our users emails are not disclosed
+ They can still “suppress” existing users
- Existing users are exposed

Hash Functions

Any function that can be used to map data of arbitrary size to fixed-size values.

Cryptographic hash function:

- Deterministic
- Quick to compute
- Preimage resistant
- Collision resistant



Source: Wikipedia

Enter Rock–paper–scissors

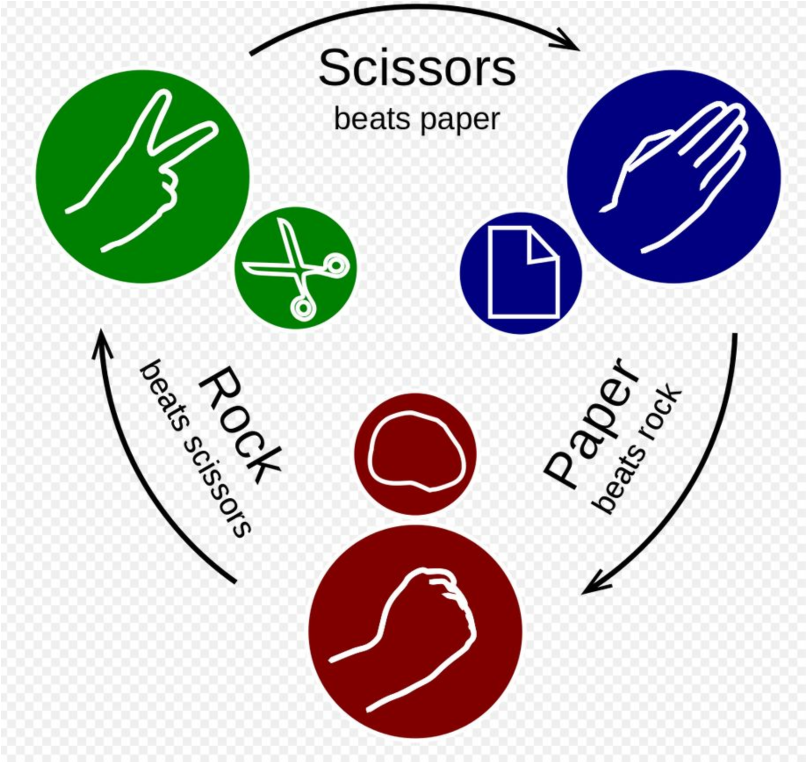
batogb

Game as a fair choosing method

Similar function as coin flipping

Not truly random

How do you do that online?



batogb

Commitment scheme

Cryptographic primitive that allows one to commit to a chosen value (or chosen statement) while keeping it hidden to others, with the ability to reveal the committed value later.

Say Alice and Bob play Rock-Paper-Scissors.

1. Alice:
   a. chooses `v` as her call
   b. chooses a random `R` and then computes `h=hash(v||R)`
   c. sends `h` to Bob
2. Bob makes his call and reports it
3. Alice reveals what she commited to by exposing both `v` and `R`
4. Bob verifies that `v` and `R` match the commitment

Public Key Cryptography

Symmetric encryption:

```
A: coded_message = encrypt(message, key)

B: message = decrypt(coded_message, key)    <<< same key
```

With public key cryptography, each entity has a pair of keys: one private and one public. The private key is generated offline and it doesn't need to be sent over the wire at all.

```
A: coded_message = encrypt(message, private_key_a)

B: message = decrypt(coded_message, public_key_a)
```

batogb

Zero Knowledge Proofs

Wikipedia: “a method by which

one party (**the prover**) can prove to

another party (**the verifier**)

that they know a value x,

without conveying any information

apart from

the fact that they know the value x.”

Zero Knowledge Proofs - Simple Examples

batogb

1. The color blind person can be convinced that objects have different colors even if they don't see it

2. Prove the color of a card you picked from a playing cards pack

3. Where's Wally?


29

Zero Knowledge Proofs

batogb

A sort of generalization of both hashes and public key cryptography.

Hashes → "proof of data"

Public key cryptography → "proof of private key ownership"

Zero Knowledge Proofs → "proof of computation"

Allows a Verifier to ascertain that the Prover executed a public/shared computation over private data that returned a public/shared result and *the risk of Prover cheating is negligible*.

Zero Knowledge Proofs

batogb

Key Properties

1. Completeness - if the statement is True, an honest Verifier will be convinced by this fact
2. Soundness - a malicious Prover cannot convince the Verifier of a false statement
3. Zero Knowledge - no other information except that the statement is True is revealed to the Verifier

Verification DOES NOT mean recomputation! (unlike blockchain)

zk SNARKS

SNARK referring to "Succinct Non-interactive ARgument of Knowledge"

Elements of a zkSNARK:

1. `(prover_key, verifier_key) := setup(circuit)`
2. `proof := generateProof(prover_key, inputs, circuit)`
3. `true/false := verifyProof(verifier_key, proof)`

zk SNARKS

batogb

- short and non interactive proofs
- zero knowledge
- verification cost independent of computational complexity
- proof: 3 EC Points = 127 bytes

zokrates.github.io

batogb

ZoKrates is a toolbox for zkSNARKs on Ethereum

```
def main(private field a, field b) -> (field):
  field result = if a * a == b then 1 else 0 fi
  return result
```

zokrates.github.io

batogb

```
# compile
./zokrates compile -i root.code

# perform the setup phase
./zokrates setup

# execute the program
./zokrates compute-witness -a 337 113569

# generate a proof of computation
./zokrates generate-proof

# export a solidity verifier
./zokrates export-verifier
```

Example: Confidential Transactions on Ethereum

batogb

Normally, an ERC20 token will hold:

```
mapping (address => uint256) balances;
```

And when a transfer is made, it must check:

```
balances[fromAddress] >= value
```

Source: https://media.consensys.net/introduction-to-zksnarks-with-examples-3283b554fc3b

Example: Confidential Transactions on Ethereum

batogb

In CT, we replace the balance with the hash of balance:

```
mapping (address => bytes32) balanceHashes;
```

What becomes private: balances and sent amounts.

And when a transfer is made, both sender and receiver must produce each a SNARK:

Example: Confidential Transactions on Ethereum

batogb

```
function senderFunction(x, w) {
 return (
   w.senderBalanceBefore > w.value &&

   sha256(w.value) == x.hashValue &&

   sha256(w.senderBalanceBefore) ==
          x.hashSenderBalanceBefore &&

   sha256(
     w.senderBalanceBefore - w.value
   ) ==  x.hashSenderBalanceAfter
 )
}
```

```
function receiverFunction(x, w) {

 return (

   sha256(w.value) == x.hashValue &&

   sha256(w.receiverBalanceBefore) ==

          x.hashReceiverBalanceBefore &&

   sha256(

    w.receiverBalanceBefore + w.value

   ) == x.hashReceiverBalanceAfter

 )

}
```

zkSNARK Contenders

	Prover Complexity	Verifier Complexity	Typical Proof Sizes	Security Assumption	Trusted Setup
zkSNARKS	O(n * log(n))	~O(1)	127 bytes [Groth16]	Knowledge of exponent	yes
zkSTARKs	O(n * poly-log(n))	O(poly-log(n))	Few kilobytes	Collision-resistant Hash-Function	no
Bulletproofs	O(n * log(n))	O(n)	Few hundred kilobytes	Discrete Log Hardness	no

Zero Knowledge Proofs - Applications

batogb

- No longer send plain text passwords over the wire, but proofs of password hashes
- Proof of owning a document, without revealing its content
- Authentication
- Scale blockchains: move computation off-chain, have smart contracts only do verification
- Confidential transactions
- Constant size blockchain: Coda Protocol - recursive composition of zk-SNARKs

batogb

Thank you

Bogdan Batog