Privacy on the Blockchain

Dan Boneh
The need for privacy in the financial system

Supply chain privacy:
• A manufacturer does not want to reveal how much it pays its supplier for parts.

Payment privacy:
• A company that pays its employees in crypto wants to keep list of employees and salaries private.
• Endusers need privacy for rent, donations, purchases

Business logic privacy: Can the code of a smart contract be private?
The need for privacy in the financial system

The bottom line:

Blockchains cannot reach their full potential without some form of private transactions
Types of Privacy

Pseudonymity: (weak privacy)
- Every user has a long-term consistent pseudonym (e.g. reddit)
  - **Pros:** reputation
  - **Cons:** link to real-world identity can leak over time

Full anonymity: User’s transactions are unlinkable
- No one can tell if two transactions are from the same address
A difficult question: privacy from who?

No privacy: Everyone can see all transactions

Privacy from the public: Only a trusted operator can see transactions

Semi-full privacy: only “local” law enforcement can see transactions

full privacy: no one can see transactions
Negative aspects of complete privacy

How to prevent criminal activity?

The challenge:

• How to support positive applications of private payments, but prevent the negative ones?

• Can we ensure legal compliance while preserving privacy?

• Yes! The key technology: zero knowledge proofs
Are Bitcoin and Ethereum Private?

The base systems are definitely not ...
Privacy in Ethereum?

• Every account balance is public
• For Dapps: code and internal state are public
• All account transactions are linked to account

etherscan.io:

Address 0x1654b0c3f62902d7A86237...

<table>
<thead>
<tr>
<th></th>
<th>Txn Hash</th>
<th>Method</th>
<th>Block</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0x0269eff8b4196558c07...</td>
<td>Set Approval For...</td>
<td>13426561</td>
</tr>
<tr>
<td></td>
<td>0xa3dabc0e7c579a99cd...</td>
<td>Cancel Order...</td>
<td>13397993</td>
</tr>
<tr>
<td></td>
<td>0x73785abcc7ccf030d6a...</td>
<td>Set Approval For...</td>
<td>13387834</td>
</tr>
<tr>
<td></td>
<td>0x1463293c495069d61c...</td>
<td>Atomic Match...</td>
<td>13387703</td>
</tr>
</tbody>
</table>

Balance: 1.114479450024297906 Ether
Ether Value: $4,286.34 (@ $3,846.05/ETH)
Alice can have many addresses (creating address is free)

Inputs: A1:4,  A2: 5  

out: B:6,  A3:3  

Change address

Alice’s addresses

Bob’s address

Transaction data can be used to link an address to a physical identity

(Chainalysis)
Alice buys a book from a merchant:

- Alice learns one of merchant’s address (B)
- Merchant links three addresses to Alice (A1, A2, A3)

Alice uses an exchange (ETH ↔ USD)

- BSA: a US exchange must do KYC (know your customer)
  ... collect and verify Alice’s ID
- Exchange links Alice to her addresses (A1, A2, A3)
De-anonymization strategy: Idioms of use

A general strategy for de-anonymizing Bitcoin addresses

**Heuristic 1:**

Two addresses are input to a TX

⇒ both addresses are controlled by same entity
De-anonymization strategy: Idioms of use

Heuristic 2:

Change address is controlled by the same user as input address

Which is the change address?

• Heuristic: a new address that receives less than every input address

---

<table>
<thead>
<tr>
<th>Address</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>16k4365RzdeCPKGwJDNNBEkXj696MbChwx</td>
<td>0.53333328 BTC</td>
</tr>
<tr>
<td>1Bsh4KD9J7T4dJcoo7S5u51jvtm1tVmREb7</td>
<td>1.47877788 BTC</td>
</tr>
<tr>
<td>1JgVBpwsTDMTRoZx9XpPDQRRHtNb5csP</td>
<td>0.01031593 BTC (U)</td>
</tr>
<tr>
<td>1AFLhD4e21ZfXmfdXCyGUNqCqD5887u</td>
<td>2 BTC (S)</td>
</tr>
</tbody>
</table>

FEE: 0.00179523 BTC
step 1: Heuristic 1 and 2 ⇒ 3.3M clusters

step 2: 1070 addresses identified by interacting with merchants
- Coinbase, Bitpay, ...

step 3: now 15% of all addresses identified
- Learn total assets for all clusters

Commercial efforts: Chainalysis, Elliptic, ...
Private coins on a Public Blockchain
Attempt 1: simple mixing

Observer knows Y belongs to one of \{Alice, Bob, Carol\} but does not know which one $\implies$ anonymity set of size 3.

Problems: (i) mixer M knows shuffle, (ii) mixer can abscond with 3 ETH !!
Increasing the anonymity set

M1: mix $n$ inputs from $n$ users $\Rightarrow$ $X'$ has anonymity set size = $n$

M2: mix output from $m$ mixers $\Rightarrow$ $X''$ has anonymity set size = $n \times m$

Privacy: as long as one of M1 or M2 are honest
Secure mixing without a mixer?

**Problem:** Mixer can abscond with funds or reveal the shuffle.

Can we securely mix without a trusted mixer?  
**Answer:** yes!

- on Bitcoin: **CoinJoin**  (used by, e.g., Wasabi wallet)
- on Ethereum: **Tornado cash**
  
  ... a large scale single mixer using ZK proofs – next lecture
The setup: Alice, Bob, and Carol want to mix together.

Alice owns UTXO $A_1: 5$, Bob owns UTXO $B_1: 3$, Carol owns $C_1: 2$

- $A_1$: 5, $A_3$ (change addr)
- $A_2$ (post mix address over Tor)
- $B_1$: 3, $B_3$ (change addr)
- $B_2$ (post mix address over Tor)
- (same as Alice and Bob)

Public forum mix addresses:

- $A_1$: 5, $A_3$
- $B_1$: 3, $B_3$
- $C_1$: 2, $C_3$
- $B_2$, A2, C2

Public forum
**CoinJoin TX**: all three prepare and sign the following Tx:

<table>
<thead>
<tr>
<th>inputs (not private)</th>
<th>A1: 5, B1: 3, C1: 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>outputs (private)</td>
<td>B2: 2, A2: 2, C2: 2</td>
</tr>
<tr>
<td>outputs (not private)</td>
<td>A3: 3, B3: 1</td>
</tr>
</tbody>
</table>

Mixed UTXOs all have same value = min of inputs  (2 in this case)

All three post sigs on Pastebin ⇒ one of them posts Tx on chain.
Coinjoin drawbacks

In practice: each CoinJoin Tx mixes about 40 inputs
- Large Tx: 40 inputs, 80 outputs

All participants must sign CoinJoin Tx for it to be valid
⇒ ensures all of them approve the CoinJoin Tx
... but any one of them can disrupt the process
Beyond simple mixing

Private Tx on a public blockchain
Can we have private transactions on a public blockchain?

Naïve reasoning:

universal verifiability $\Rightarrow$ transaction data must be public
otherwise, how we can verify Tx??

crypto magic $\Rightarrow$ private Tx on a publicly verifiable blockchain

Crypto tools: commitments and zero knowledge proofs
A paradigm for Private Tx

public blockchain

committed state  \( \pi \)  updated committed state

anyone can verify \( \pi \)
(reveals nothing about Tx data or state)

Committed data: short (hiding) commitment on chain

Proof \( \pi \): succinct zero-knowledge proof that
(1) committed Tx data is consistent with committed current state, and
(2) committed updated state is correct
Cryptographic commitment: emulates an envelope

Many applications: e.g., a DAPP for a sealed bid auction

- Every participant commits to its bid,
- Once all bids are in, everyone opens their commitment
Cryptographic Commitments

Syntax: a commitment scheme is two algorithms

- `commit(msg, r) → com`
  - secret randomness
  - commitment string

- `verify(msg, com, r) → accept or reject`
  - anyone can verify that commitment was opened correctly
Commitments: security properties

- **binding**: Bob cannot produce two valid openings for \( \text{com} \)
  
  More precisely: no efficient adversary can produce
  
  \[
  \text{com}, (m_1, r_1), (m_2, r_2)
  \]
  
  such that \( \text{verify}(m_1, \text{com}, r_1) = \text{verify}(m_2, \text{com}, r_2) = \text{accept} \)
  
  and \( m_1 \neq m_2 \).

- **hiding**: \( \text{com} \) reveals nothing about committed data
  
  \( \text{commit}(m, r) \rightarrow \text{com}, \) and \( r \) is sampled uniformly in a set \( R \),
  
  then \( \text{com} \) is statistically independent of \( m \)
Example: hash-based commitment

Fix a hash function $H: M \times R \rightarrow C$ (e.g., SHA256) where $H$ is collision resistant, and $|R| \gg |C|$

- commit($m \in M$, $r \leftarrow R$): $com = H(m, r)$
- verify($m$, $com$, $r$): accept if $com = H(m, r)$

binding: follows from collision resistance of $H$

hiding: follows from a mild assumption on $H$
What is a zk-SNARK?

Succinct zero knowledge proofs: an important tool for privacy on the blockchain
What is a zk-SNARK? (intuition)

**SNARK:** a succinct proof that a certain statement is true

Example statement: “I know an $m$ such that $\text{SHA256}(m) = 0$”

- **SNARK:** the proof is “short” and “fast” to verify

  [if $m$ is 1GB then the trivial proof (the message $m$) is neither]

- **zk-SNARK:** the proof “reveals nothing” about $m$
<table>
<thead>
<tr>
<th>zk-SNARK: Blockchain Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Private Tx on a public blockchain:</strong></td>
</tr>
<tr>
<td>• Tornado cash, Zcash, IronFish</td>
</tr>
<tr>
<td>• Private Dapps: Aleo</td>
</tr>
<tr>
<td><strong>Compliance:</strong></td>
</tr>
<tr>
<td>• Proving that private Tx are in compliance with banking laws</td>
</tr>
<tr>
<td>• Proving solvency in zero-knowledge</td>
</tr>
<tr>
<td><strong>Scalability:</strong> privacy in a zk-SNARK Rollup (next week)</td>
</tr>
<tr>
<td><strong>Bridging between blockchains:</strong> zkBridge</td>
</tr>
</tbody>
</table>
Arithmetic circuits

- Fix a finite field $\mathbb{F} = \{0, \ldots, p - 1\}$ for some prime $p > 2$.
- **Arithmetic circuit**: $C : \mathbb{F}^n \rightarrow \mathbb{F}$
  - directed acyclic graph (DAG) where internal nodes are labeled $+,$ $–,$ or $\times$
  - inputs are labeled $1, x_1, \ldots, x_n$
  - defines an $n$-variate polynomial with an evaluation recipe
- $|C| = \#$ gates in $C$
Interesting arithmetic circuits

Examples:

• $C_{\text{hash}}(h, m)$: outputs 0 if $\text{SHA256}(m) = h$, and $\neq 0$ otherwise

\[
C_{\text{hash}}(h, m) = (h - \text{SHA256}(m)), \quad |C_{\text{hash}}| \approx 20K \text{ gates}
\]

• $C_{\text{sig}}(pk, m, \sigma)$: outputs 0 if $\sigma$ is a valid ECDSA signature on $m$ with respect to $pk$
(preprocessing) **NARK: Non-interactive ARgument of Knowledge**

Public arithmetic circuit: \( C(x, w) \rightarrow \mathbb{F} \)

- public statement in \( \mathbb{F}^n \)
- secret witness in \( \mathbb{F}^m \)

Preprocessing (setup): \( S(C) \rightarrow \) public parameters \((pp, vp)\)

![Diagram showing the process of NARK](image-url)
A preprocessing NARK is a triple \((S, P, V)\):

- \(S(C) \rightarrow\) public parameters \((\text{pp}, \text{vp})\) for prover and verifier
- \(P(\text{pp}, x, w) \rightarrow\) proof \(\pi\)
- \(V(\text{vp}, x, \pi) \rightarrow\) accept or reject
NARK: requirements (informal)

Prover $P(pp, x, w)$

Verifier $V (vp, x, \pi)$

proof $\pi$

accept or reject

Complete: $\forall x, w: C(x, w) = 0 \Rightarrow Pr[ V(vp, x, P(pp, x, w)) = \text{accept} ] = 1$

Adaptively knowledge sound: $V$ accepts $\Rightarrow P$ “knows” $w$ s.t. $C(x, w) = 0$
(an extractor $E$ can extract a valid $w$ from $P$)

Optional: Zero knowledge: $(C, pp, vp, x, \pi)$ “reveal nothing” about $w$
A **succinct preprocessing NARK** is a triple \((S, \ P, \ V)\):

- \(S(C) \rightarrow \) public parameters \((pp, vp)\) for prover and verifier
- \(P(pp, x, w) \rightarrow \) **short** proof \(\pi\) ; \(|\pi| = O_\lambda(\log(|C|))\)
- \(V(vp, x, \pi) \rightarrow \) **fast to verify** ; \(\text{time}(V) = O_\lambda(|x|, \log(|C|))\)

Why preprocess \(C\)??

short “summary” of circuit
SNARK: a **Succinct ARgument of Knowledge**

A **succinct preprocessing NARK** is a triple \((S, P, V)\):

- **S\((C)\)** → public parameters \((pp, vp)\) for prover and verifier

- **P\((pp, x, w)\)** → **short** proof \(\pi\) ; \(|\pi| = O_\lambda(\log(|C|))\)

- **V\((vp, x, \pi)\)** | **fast to verify** ; \(\text{time}(V) = O_\lambda(|x|, \log(|C|))\)

**SNARK:** \((S, P, V)\) is **complete**, **knowledge sound**, and **succinct**

**zk-SNARK:** \((S, P, V)\) is a SNARK and is **zero knowledge**
The trivial SNARK is not a SNARK

(a) Prover sends $w$ to verifier,
(b) Verifier checks if $C(x, w) = 0$ and accepts if so.

Problems with this:
(1) $w$ might be secret: prover does not want to reveal $w$ to verifier
(2) $w$ might be long: we want a “short” proof
(3) computing $C(x, w)$ may be hard: we want a “fast” verifier
Next lecture:
more on zk-SNARKs and their applications