A cryptographic investigation of Mimblewimble

Georg Fuchsbauer

joint work with

Michele Orrù 

and Yannick Seurin
What is it?

- Proposal for a cryptocurrency system
  - **Privacy** (all amounts hidden; forget spent tx’s)
  - **Scalability** (forget spent tx’s)

- proposed by “Tom Elvis Jedusor” in 2016
What is it?

- Proposal for a **cryptocurrency system**
  - **Privacy** (all amounts hidden; forget spent tx’s)
  - **Scalability** (forget spent tx’s)

- implemented by *Grin*

- uses ideas from Gregory Maxwell

- proposed by “Tom Elvis Jedusor” in 2016

- further developed by Andrew Poelstra
Bitcoin

- Transactions

2 BTC → 6 BTC
2 BTC → 1 BTC
3 BTC → 1 BTC
Bitcoin

- Block

```
2 BTC → 6 BTC
2 BTC → 1 BTC
3 BTC →
```

Transaction:
- In
- Out

Transaction:
- In
- Out
- In
Bitcoin

- Blockchain

```
2 BTC → 6 BTC
2 BTC → 1 BTC
3 BTC → 6 BTC

6 BTC → 2 BTC
6 BTC → 4 BTC
```
• Reference to previous output
• Coinbase transaction

Bitcoin transaction

- 12.5 BTC
- 2 BTC → 6 BTC
- 2 BTC → 1 BTC
- 3 BTC → 12 BTC
- 12.5 BTC → 12 BTC
- 12.5 BTC → 0.5 BTC
Bitcoin

- $\sigma$ is signature for $pk$

Diagram:

1. Transaction: $pk \rightarrow 6\ BTC$
2. Transaction: $pk' \rightarrow 1\ BTC$
3. Transaction: $\sigma, pk'' \rightarrow 2\ BTC$
4. Transaction: $pk \rightarrow 4\ BTC$
Security

- **signatures** ⇒ no theft
- **balancedness** of tx’s checkable ⇒ no inflation

- $\sigma$ is signature for $pk$
Bitcoin

Unspent transaction outputs (UTXO’s) = existing money in system
**Drawbacks**

- all tx's public
  -⇒ weak anonymity
- all data **must be kept** for verification
  -⇒ bad scalability
Scalability
Scalability

“cut-through”
Scalability

“cut-through”

not possible in Bitcoin:

$\sigma'$ is needed to verify validity

$\Rightarrow$ Mimblewimble
Anonymity

1 BTC → In → Out → 3 BTC
3 BTC → In → Out → 3 BTC
2 BTC → In
**CoinJoin**  [Maxwell’13]

- no *link* between inputs and outputs
- can we join many transactions together?
- *in Bitcoin:* only interactively, since all inputs must sign tx

**Anonymity**

- CoinJoin

```
1 BTC  →  3 BTC
3 BTC  →  3 BTC
2 BTC  →  3 BTC

Alice  Bob
```
Anonymity

- **Confidential Transactions** [Maxwell]
  - hide the input and output *amounts*
  - not compatible with Bitcoin system
  - balancedness verifiable?
Anonymity

How can we get

- Confidential transactions (check balancedness)
- Coin-join (non-interactively)
- Cut-through (thus scalability)

while maintaining verifiability?
Anonymity

- Confidential Transactions:
  - hide input and output amounts
  - not compatible with Bitcoin as is
  - balancedness verifiable?

Mimblewimble
Some maths . . . and crypto!
Elliptic curves

- defined over finite field
- curve points can be added “+” ⇒ group $G$
  - generator $G$
  - $xG := G + \ldots + G$

\[ y^2 = x^3 - x \quad \text{and} \quad y^2 = x^3 - x + 1 \]
Elliptic curves

- defined over finite field
- curve points can be added “+” ⇒ group $\mathbb{G}$
  - generator $G$
  - $xG := \underbrace{G + \ldots + G}_{x \text{ times}}$

- **Discrete logarithm** problem:
  - given $G, H \in \mathbb{G}$
  - find $x$ such that $H = xG$
Elliptic curves

- defined over finite field
- curve points can be added “+” ⇒ group $\mathbb{G}$
  - generator $G$
  - $xG := G + \ldots + G$ $x$ times

- **Discrete logarithm** problem:
  - given $G, H \in \mathbb{G}$
  - find $x$ such that $H = xG$

- used in **signature schemes** (e.g. ECDSA, Schnorr)
  - secret key: $x$
  - public key: $X = xG$
Pedersen commitment

Commitment

- “digital envelope”
Pedersen commitment

Commitment
- "digital envelope"

- **hiding**: commitment hides $v$
- **binding**: Alice can open commitment only to one value
Pedersen commitment

Commitment

• “digital envelope”

Commit $v$

Open

Pedersen

$G, H \in \mathbb{G}$

pick random $r$

$C := vH + rG$

reveal $v$ and $r$
Pedersen commitment

**Commitment**
- “digital envelope”

\[ C = vH + rG \]

**hiding:** for any \( v \) exists \( r \) so that \( C \) commits \( v \)
Pedersen commitment

**Commitment**
- “digital envelope”

\[ C := vH + rG \]
\[ \log_G C = v \cdot \log_G H + r \]

- **hiding:** for any \( v \) exists \( r \) so that \( C \) commits \( v \):
\[
(r = \log_G C - v \cdot \log_G H)
\]
Pedersen commitment

Commitment

- "digital envelope"

Commit 

Pick random $r$

$C := vH + rG$

Reveal $v$ and $r$

Binding: Assume Alice finds $v, r, v', r'$ with

$vH + rG = C = v'H + r'G$
Pedersen commitment

**Commitment**

- “digital envelope”

![Diagram showing commitment process]

- **binding:** assume Alice finds $v, r, v', r'$ with
  
  $$vH + rG = C = v'H + r'G,$$

  then
  
  $$\frac{r' - r}{v - v'} G = H$$

  $\Rightarrow$ Alice solved discrete log problem!
Pedersen commitment

Commitment

- “digital envelope”

Commit $v$

Open

- commitments are **homomorphic**:

$$\text{Com}(v_1; r_1) + \text{Com}(v_2; r_2) = (v_1 H + r_1 G) + (v_2 H + r_2 G)$$
$$= (v_1 + v_2)H + (r_1 + r_2)G$$
$$= \text{Com}(v_1 + v_2; r_1 + r_2)$$

\[\text{e.g.} : \text{Com}(1; 5) + \text{Com}(1; 10) - \text{Com}(2, 15) = 0\]
Confidential Transactions

- use commitments to amounts

\[
\begin{align*}
\text{In}_1 & \rightarrow \text{Transaction} & \text{Out}_1 \\
\text{In}_2 & \rightarrow \text{Transaction} & \text{Out}_2 \\
\text{In}_3 & \rightarrow \text{Transaction} & \text{Out}_2
\end{align*}
\]

\[
C = vH + rG
\]
Confidential Transactions

- use *commitments* to amounts
- ensure that transactions do not create money?

\[ C = vH + rG \]

\[ \text{Out}_1 + \ldots + \text{Out}_n - \text{In}_1 - \ldots - \text{In}_\ell = 0 \]
Confidential Transactions

- use commitments to amounts
- ensure that transactions do not create money?

\[
C = vH + rG
\]

\[
\sum \text{Out} - \sum \text{In} = 0
\]

\[
\sum C^{\text{out}}_i - \sum C^{\text{in}}_i = \sum (v^{\text{out}}_i H + r^{\text{out}}_i G) - \sum (v^{\text{in}}_i H + r^{\text{in}}_i G)
\]

\[
= (\sum v^{\text{out}}_i - \sum v^{\text{in}}_i) H + (\sum r^{\text{out}}_i - \sum r^{\text{in}}_i) G
\]

\[
\overset{\text{\( i \neq 0 \)}}{= 0}
\]

\[
\overset{\text{\( i \neq 0 \)}}{= 0}
\]
Confidential Transactions

- use *commitments* to amounts
- ensure that transactions do not create money?

\[ \sum \text{Out} - \sum \text{In} = 0 \]

\[ C = vH + rG \]

\[ \sigma_1 \quad \sigma_2 \quad \sigma_3 \]

\[ \text{Out}_1 \quad -5 \]

\[ \text{Out}_2 \quad 9 \]
Confidential Transactions

- use *commitments* to amounts
- ensure that transactions do not create money?

\[
\sum \text{Out} - \sum \text{In} = 0
\]

Range proofs
- add proofs that committed values are in \( \in [0, 2^{64}] \)

[Back, Maxwell '13–'15]
Confidential Transactions

Confidential transaction

\[ C = \nu H + rG, \quad \pi \]

\[ \sum \text{Out} - \sum \text{In} = 0 \]

Signatures ⇒
• no non-interactive CoinJoin
• no Cut-Through
Mimblewimble

\[ C = vH + rG, \quad \pi \]

\[ \sum \text{Out} - \sum \text{In} = 0 \]

secret key!

no more signatures!
Mimblewimble

In

Out

Transaction

In$_1$ → Out$_1$
In$_2$ → Out$_2$
In$_3$ →

C = vH + rG, \ \pi

secret key!

\[ \sum \text{Out} - \sum \text{In} = 0 \]

no more signatures!

But: sender knows sum of output r’s

[Jedusor ’16]
Mimblewimble

\[ C = vH + rG, \quad \pi \]

\[ \sum \text{Out} - \sum \text{In} = 0 \]

\[
\sum C_{i}^{\text{out}} - \sum C_{i}^{\text{in}} \\
= \sum (v_{i}^{\text{out}} H + r_{i}^{\text{out}} G) - \sum (v_{i}^{\text{in}} H + r_{i}^{\text{in}} G) \\
= \left( \sum v_{i}^{\text{out}} - \sum v_{i}^{\text{in}} \right) H + \left( \sum r_{i}^{\text{out}} - \sum r_{i}^{\text{in}} \right) G \\
\equiv x \]

[Jedusor '16]
Mimblewimble

In

Out

Transaction

C = vH + rG, \pi

\[ \sum \text{Out} - \sum \text{In} = 0H + xG \]

\[
\sum C_i^{\text{out}} - \sum C_i^{\text{in}} \\
= \sum (v_i^{\text{out}} H + r_i^{\text{out}} G) - \sum (v_i^{\text{in}} H + r_i^{\text{in}} G) \\
= (\sum v_i^{\text{out}} - \sum v_i^{\text{in}}) H + (\sum r_i^{\text{out}} - \sum r_i^{\text{in}}) G \\
\leq 0 \\
\Rightarrow : x
\]
Mimblewimble

In

Out

\[ C = vH + rG, \quad \pi \]

\[ \sum \text{Out} - \sum \text{In} = 0H + xG \]

one signature

“proves” that \( \sum \text{Out} - \sum \text{In} \) is commitment to 0
Mimblewimble

\[ \sum \text{Out}_1 - \sum \text{In}_1 = X_1 \]
\[ \sigma_1 \text{ valid for } X_1 \]

\[ \sum \text{Out}_2 - \sum \text{In}_2 = X_2 \]
\[ \sigma_2 \text{ valid for } X_2 \]
Non-interactive CoinJoin

\[ \sum \text{Out}_1 - \sum \text{In}_1 = X_1 \]
\[ \sigma_1 \text{ valid for } X_1 \]

\[ \sum \text{Out}_2 - \sum \text{In}_2 = X_2 \]
\[ \sigma_2 \text{ valid for } X_2 \]

\[ \sum \text{Out} - \sum \text{In} = X_1 + X_2 \]
\[ \sigma_1 \text{ valid for } X_1 \]
\[ \sigma_2 \text{ valid for } X_2 \]
Mimblewimble

Non-interactive CoinJoin

\[ \sum \text{Out} - \sum \text{In} = X_1 + X_2 \]

- \( \sigma_1 \) valid for \( X_1 \)
- \( \sigma_2 \) valid for \( X_2 \)
Mimblewimble

Cut-Through!

\[ \sum \text{Out} - \sum \text{In} = X_1 + X_2 \]

- \( \sigma_1 \) valid for \( X_1 \)
- \( \sigma_2 \) valid for \( X_2 \)
Mimblewimble

Cut-Through!

\[ \sum \text{Out} - \sum \text{In} = X_1 + X_2 \]

- \( \sigma_1 \) valid for \( X_1 \)
- \( \sigma_2 \) valid for \( X_2 \)
Mimblewimble

Cut-Through!

\[ \sum \text{Out} - \sum \text{In} = X_1 + X_2 \]

- \( \sigma_1 \) valid for \( X_1 \)
- \( \sigma_2 \) valid for \( X_2 \)
Scalability

“cut-through”

not possible in Bitcoin:

$\sigma'$ is needed to verify validity

$\Rightarrow$ Mimblewimble
Scalability

“cut-through”

not possible in Bitcoin:

\(\sigma'\) is needed to verify validity

\(\Rightarrow\) Mimblewimble
Scalability

“cut-through”

not possible in Bitcoin: 

$\sigma'$ is needed to verify validity

⇒ Mimblewimble
Mimblewimble

Cut through all transactions in blockchain

\[
\sum \text{Out} - \sum \text{In} = \sum X_i
\]

\[\forall i : \sigma_i \text{ valid for } X_i\]
Mimblewimble

Cut through all transactions in blockchain

Only coinbase transactions

\[ \sum \text{Out} - \sum \text{In} = \sum X_i \]

\[ \forall i: \sigma_i \text{ valid for } X_i \]
Mimblewimble

How to we actually make payments?

\[ \sum \text{Out} - \sum \text{In} = \sum X_i \]

\[ \forall i : \sigma_i \text{ valid for } X_i \]
Mimblewimble

How to we actually make payments?

Original proposal. To pay $p$:

- **Sender**
  - choose input coins worth $\sum v_i^{\text{in}} \geq p$
  - create change coins $C_i^{\text{chg}}$ worth $\sum v_i^{\text{chg}} = \sum v_i^{\text{in}} - p$
  - send $r = \sum r_i^{\text{chg}} - \sum r_i^{\text{in}}$

[Jedusor ’16]
Mimblewimble

How to we actually make payments?

Original proposal. To pay \( p \):

- **Sender**
  - choose input coins worth \( \sum v_{i}^{in} \geq p \)
  - create change coins \( C_{i}^{chg} \) worth \( \sum v_{i}^{chg} = \sum v_{i}^{in} - p \)
  - send \( r = \sum r_{i}^{chg} - \sum r_{i}^{in} \)

- **Receiver**
  - creates output coins \( C_{i}^{out} \) worth \( p \)
  - signs using \( x = r + \sum r_{i}^{out} \)
Mimblewimble

How to we actually make payments?

But:

- Receiver knows $x$  
  $\Rightarrow$ can revert transaction!
Mimblewimble

How to we actually make payments?

Grin: Sender & Receiver compute $\sigma$ interactively

But:
- Receiver knows $x$  
  $\Rightarrow$ can revert transaction!
Mimblewimble

Our proposal: non-interactive!

*Sender*, to pay \( p \), send:

\[
\begin{array}{c}
\text{In} \\
\text{In} \\
\text{In}
\end{array} \rightarrow \quad \begin{array}{c}
\quad \\
\quad \\
\quad
\end{array} \rightarrow \quad \begin{array}{c}
\text{Chg} \\
\text{Out}
\end{array} \text{ worth } p
\]

\[\sigma_1 \quad X_1 \quad \gamma_{\text{out}}\]
Mimblewimble

Our proposal: non-interactive!

Sender

Receiver
Mimblewimble

Our proposal: non-interactive!

Sender

Receiver

Merge:
Our contributions:

- **Formal security models:**
  - inflation-resistance
  - coin-theft-resistance
  - confidential amounts
Mimblewimble

Our contributions: to appear at EUROCRYPT’19

- **Formal security models:**
  - inflation-resistance
  - coin-theft-resistance
  - confidential amounts

- **Abstraction of Mimblewimble** from:
  - homomorphic commitments
  - compatible signatures
  - simulation-extractable NIZK range proofs
Mimblewimble

Our contributions: to appear at EUROCRYPT’19

• Formal security models:
  – inflation-resistance
  – coin-theft-resistance
  – confidential amounts

• Abstraction of Mimblewimble from:
  – homomorphic commitments
  – compatible signatures
  – simulation-extractable NIZK range proofs ... satisfying joint security
Mimblewimble

Our contributions: to appear at EUROCRYPT’19

- **Formal security models:**
  - inflation-resistance
  - coin-theft-resistance
  - confidential amounts

- **Abstraction of Mimblewimble** from:
  - homomorphic commitments
  - compatible signatures
  - simulation-extractable NIZK range proofs

- **Proof** that abstraction satisfies model
Mimblewimble

Our contributions: to appear at EUROCRYPT’19

- **Formal security models:**
  - inflation-resistance
  - coin-theft-resistance
  - confidential amounts

- **Abstraction of Mimblewimble** from:
  - homomorphic commitments
  - compatible signatures
  - simulation-extractable NIZK range proofs

- **Proof** that abstraction satisfies model

- **Instantiations:** proof that
  - Pedersen + Schnorr
  - Pedersen + (aggregate) BLS