Monero

Sebastian Kung

November 27, 2019

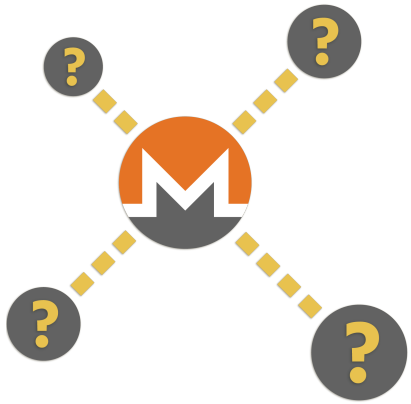
Overview

- What is Monero?
- Elliptic curve crypto primer
- Ring signatures
- Confidential transactions
- One-Time adresses
- Problems when working with anonymity networks
- Scaling Monero

What is Monero?

Secure

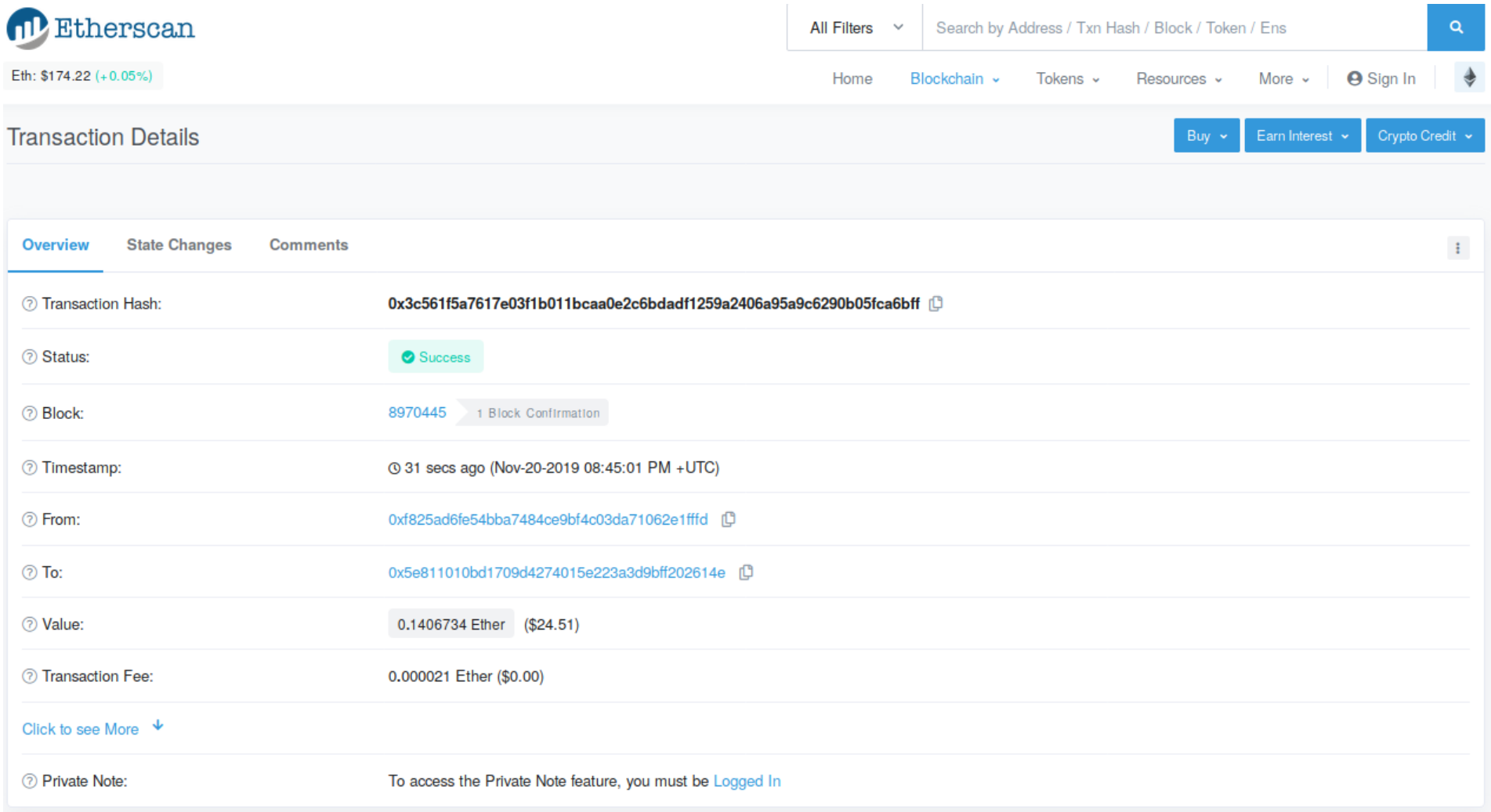
Untraceable



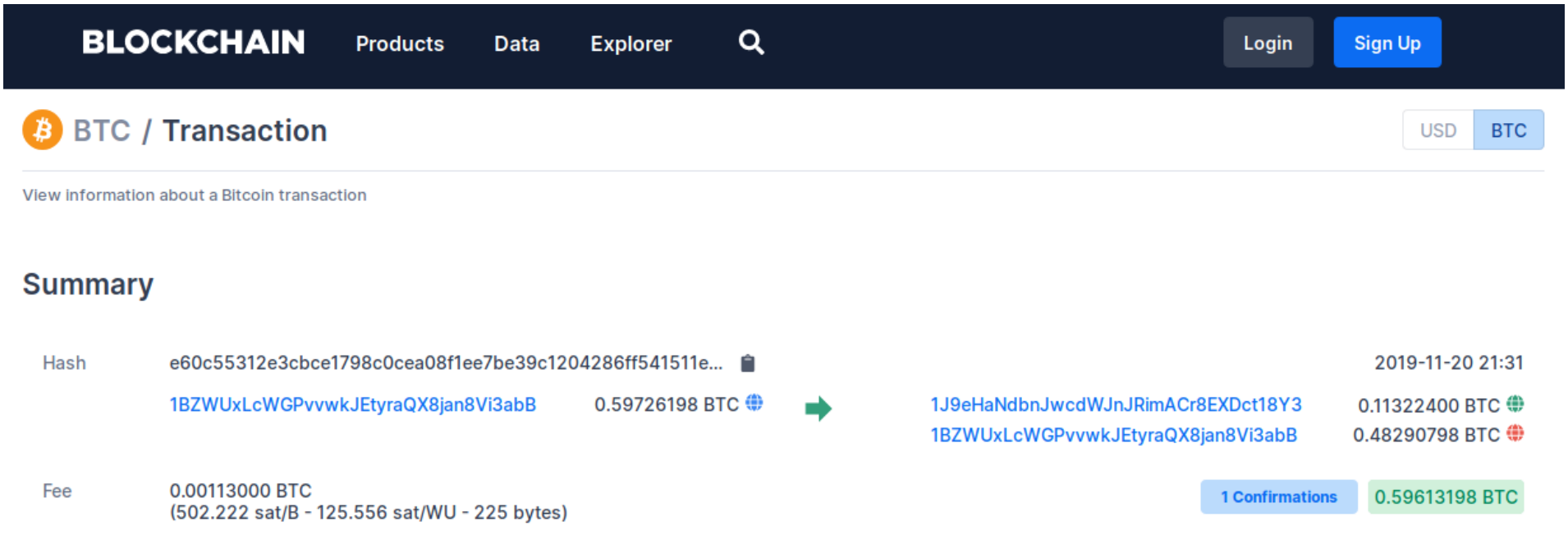
Private

Fungible

Ethereum transaction information



Bitcoin transaction information



Monero transaction information

Transaction 61bb2a2ed0a1e940ae3dcb3a584bcd7db3dd83cfeeb34a5e30a5f107995f9033

From Block	*1971297*
Output total	*confidential*
Fee	*0.000063720000 XMR*
Size	*1779 bytes*
Mixin	*10*
Unlock	*0*

Confidential Transaction – amounts are not disclosed.

Inputs (1)

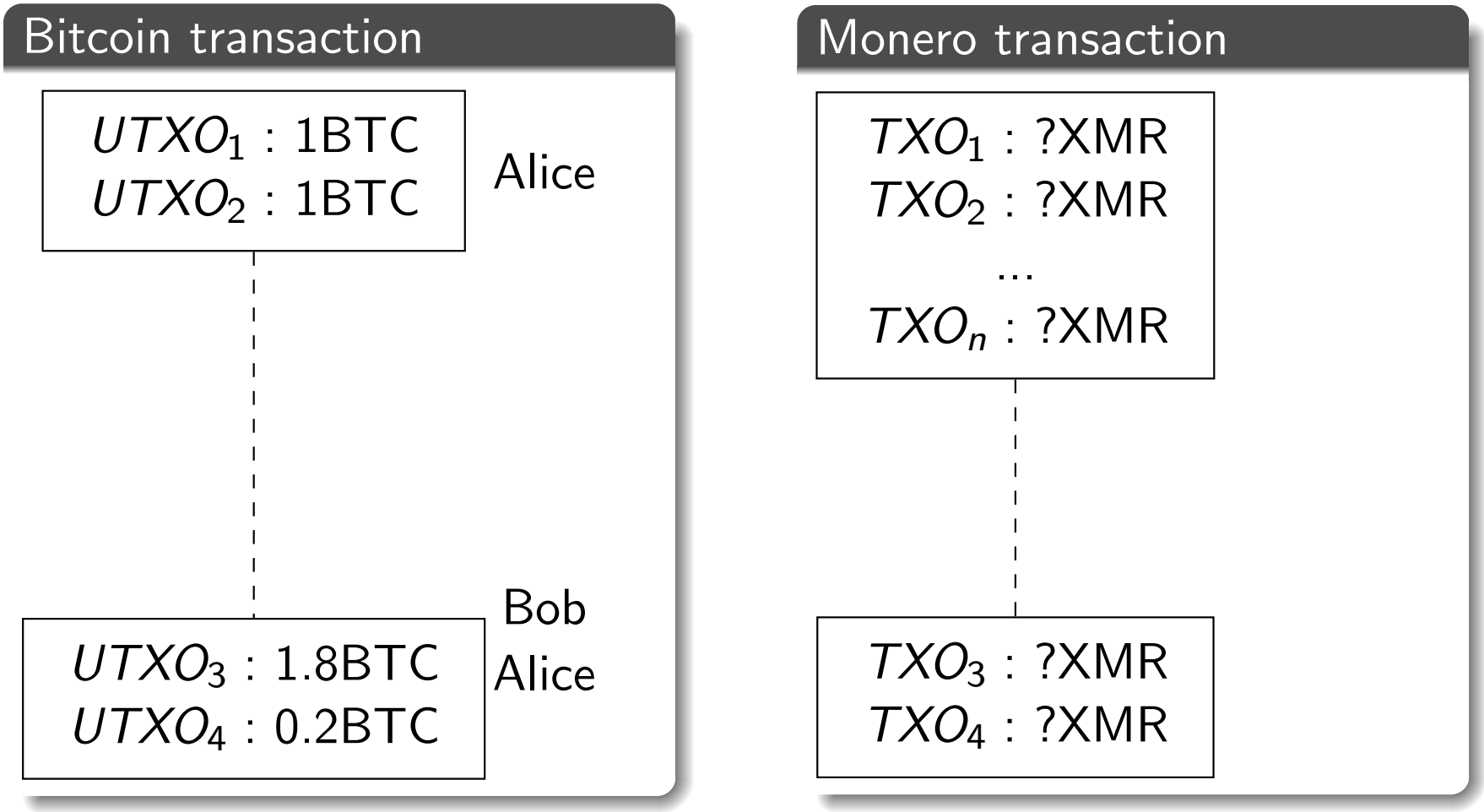
	Amount	Key Image
−	0.000000000000	4cebb4ef7dbb678d1c4a110e5a7c7ab4616aae306a3f61cdd4e4329a5073215e

From Block	Public Key
1419313	e21b35824205954f81ad511faef16ebd2f78e17026269712c28d27258fe53481
1453218	308d262c0ae71ca4dacb577e5aeaf84594aacdc9589ca4d8a2f113c4c6957cdf
1658091	cd8b166d476c828f4d5253f52db73cde86181d2d678d6136f665c1c03adc0c32
1729818	3ff122af46f46bcf311f2e28805f3ad673d5847ecaad80a4023f2fedd1db7c9a
1828197	befb281a17c623c6b1afb5d6888397751650e726e2693b3b316c961eb93ae64b
1860208	9e5fd64f79a8049649f4554688f34db86928c3dccaac258b8cac51ec8a086510
1872092	d7e3c98152949401ab52e3b708af9ff57254137f7d99c7070a1c008f38ed99a2
1884946	5162ac27b0393b585a5d59dd819e03725e191e0118d2fec311a7519099b92b92
1911583	bc2b5bf1b4b94f255d593e08552fcd5e7b9284ade5edc69751887cecf76e6154
1920646	29eff9e006f654e4eaca66e97654f3ddde182d1c383062aefb1a5256e4e2049e
1970882	6dcf0c6b7211c25c84da33b44c3365366d3d0676ae302b7175678a5c31589154

Outputs (2)

Amount	Public Key
0.000000000000	782b888df5e6a1ee81682746af0bab100231eb1105bea1bf4d92df76b39bd553
0.000000000000	661cf702f07936639effd8ef029c54332d0805bea1ec254c32f7355fa87cdea8

Unlinkable transactions



Other differences to Bitcoin and Ethereum

- No scripting
- RandomX proof of work algorithm
- LMDB
- Fluffy blocks

RANDOM X

- 'ASIC-hard' proof of work algorithm
- Usage of the full CPU instruction set and high memory requirement makes it difficult to develop an optimized IC
- Tweaks some parameters according to previously mined blocks.

LMDB

- Database to store blockchain
- Very stable, corruption is seldom
- Small (40KB)
- Stores Key/Value pairs as byte strings
- Purely transactional. No log required as with Berkeley DB

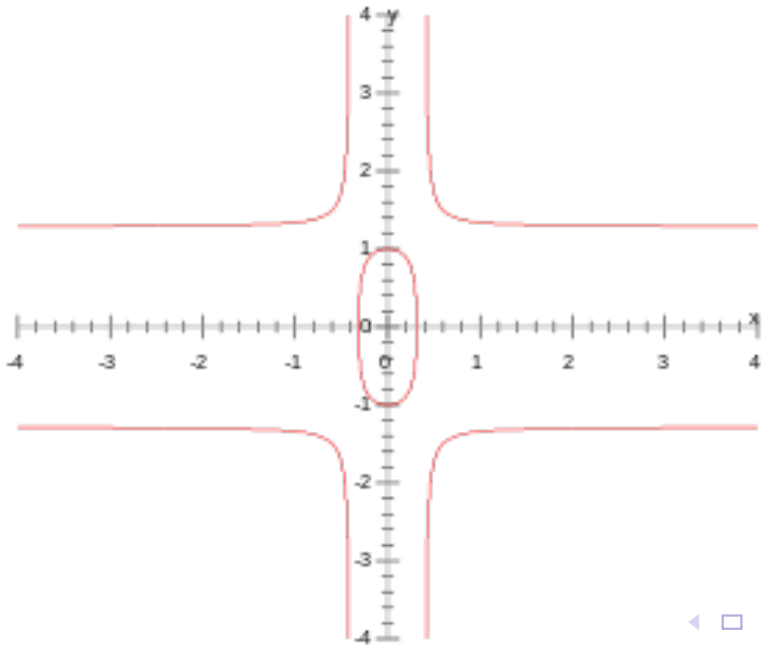
Fluffy blocks

- Nodes only flood Block Headers and transaction IDs first
- Full block content on demand

Elliptic curve crypto primer

- Monero uses the elliptic curve ED25519 with the KECCAK hashing algorithm
- 6 fundamental crypto 'building blocks' needed to define all operations, 1 assumption on hardness

⚠These are broad simplifications and do not take any underlying nuances into account



Elliptic curve crypto primer

- A public key K is a point on the elliptic curve
- A private key k is a scalar number
- Their relationship is $kG = K$, where G is the unique, publicly known generator point
- Operations on the curve are distributive: $(k + k')G = kG + k'G$
- Operations on the curve are commutative: $(kk')G = (k'k)G$
- We can define two different methods for hashing a value, $\mathcal{H}_n(m)$ returns a scalar number value and $\mathcal{H}_p(m)$ returns a point on the curve for arbitrary m
- It is *hard* to calculate k given G and K

Quick warmup: Diffie-Hellman

- Produce shared secret S with key pairs (k_A, K_A) and (k_B, K_B)
- Exchange public keys, then
- $S = k_A K_B = k_A k_B G = k_B k_A G = k_B K_A = S$

Schnorr signatures - signing

- A and B have key pairs (k_A, K_A) and (k_B, K_B)
- A wants to sign a message m
- A generates random scalar number α, computes αG
- Generate a challenge $c = \mathcal{H}_n(m, \alpha G)$
- Define a response $r := \alpha - ck_A$
- Publish (c, r, K_A, m)

Schnorr signatures - verification

- B now has to verify if r is the correct response to the challenge c
- $c' = \mathcal{H}_n(m, rG + cK_A)$
- If $c = c'$, the Sig is correct

Schnorr signatures - explanation

- Goal: Show why $c' = c = \mathcal{H}_n(m, \alpha G)$
- Remember: $r = \alpha - ck_A$; $c' = \mathcal{H}_n(m, rG + cK_A)$

$$\begin{aligned} c' &= \mathcal{H}_n(m, rG + cK_A) \\ &= \mathcal{H}_n(m, (\alpha - ck_A)G + cK_A) \\ &= \mathcal{H}_n(m, \alpha G - ck_A G + cK_A) \\ &= \mathcal{H}_n(m, \alpha G - cK_A + cK_A) \\ &= \mathcal{H}_n(m, \alpha G) = c \end{aligned} \tag{1}$$

Ring signatures

- Goal: Hide transaction origin
- LSAG, Linkable Spontaneous Anonymous Group Signatures
- Linkable: Detect signature re-use (double spend detection)
- Spontaneous: A signature can be produced at any time, no collaboration needed
- Anonymous: Unclear which ring member actually signed

Ring Signatures - signature

- Let $\mathcal{R}$ be the set of distinct public keys: $\mathcal{R} = \{K_1, K_2, K_3, ..., K_n\}$
- Let u be the index number of the user owned key pair in the ring
- Calculate key image: $\tilde{K} = k_u \mathcal{H}_p(K_u)$
- Generate scalar α and random scalars r_i for $i \in 1, 2, ..., u-1, u+1, ..., n$
- Calculate $c_{u+1} = \mathcal{H}_n(m, \alpha G, \alpha \mathcal{H}_p(K_u))$
- Then calculate iteratively up to $u-1$: $c_{i+1} = \mathcal{H}_n(m, r_i G + c_i K_i, r_i \mathcal{H}_p(K_i) + c_i \tilde{K})$
- Define $r_u = \alpha - c_u k_u$
- Complete sig: $\sigma(m) = \{c_1, r_1, r_2, ..., r_n\}$, extra: $\tilde{K}$, $\mathcal{R}$, m

Ring Signatures - verification

- For $i = 1, 2, ..., n$ iteratively compute:

$$c'_{i+1} = \mathcal{H}_n(\mathcal{R}, \tilde{K}, m, r_i G + c_i K_i, r_i \mathcal{H}_p(K_i) + c_i \tilde{K})$$

- If $c'_1 = c_1$ then the signature is valid
- In principle like a Schnorr signature, but with multiple keys

Ring signatures - example with 3 ring members

$R = \{K_1, K_2, K_3\}$

Signer: (k_2, K_2)

Generate α, r_1, r_3

$\tilde{K} = k_2 \mathcal{H}_p(K_2)$

$r_2 = \alpha - c_2 k_2$

$c_3 = \mathcal{H}_n(m, \alpha G, \alpha \mathcal{H}_p(K_2))$

Ring signatures - example with 3 ring members

$R = \{K_1, K_2, K_3\}$

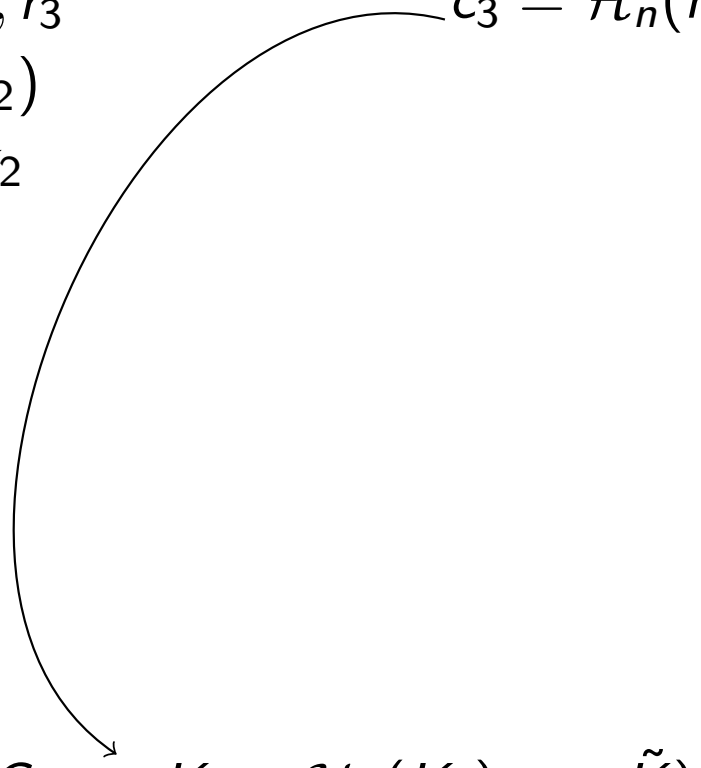
Signer: (k_2, K_2)

Generate α, r_1, r_3

$\tilde{K} = k_2 \mathcal{H}_p(K_2)$

$r_2 = \alpha - c_2 k_2$

$c_3 = \mathcal{H}_n(m, \alpha G, \alpha \mathcal{H}_p(K_2))$



$c_1 = \mathcal{H}_n(m, r_3 G + c_3 K_3, r_3 \mathcal{H}_p(K_3) + c_3 \tilde{K})$

Ring signatures - example with 3 ring members

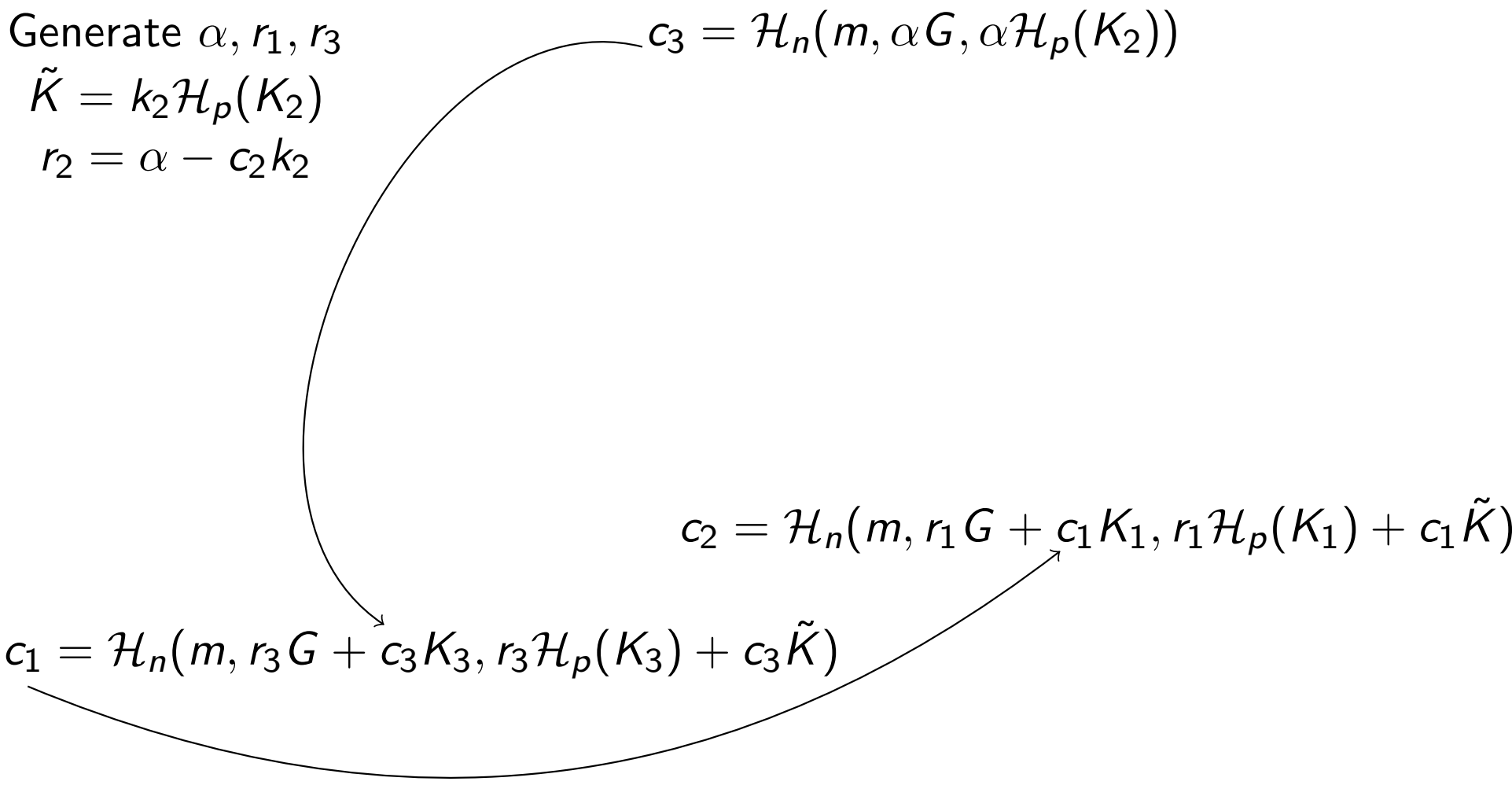
$R = \{K_1, K_2, K_3\}$

Signer: (k_2, K_2)

Generate α, r_1, r_3

$\tilde{K} = k_2 \mathcal{H}_p(K_2)$

$r_2 = \alpha - c_2 k_2$



Ring signatures - example with 3 ring members

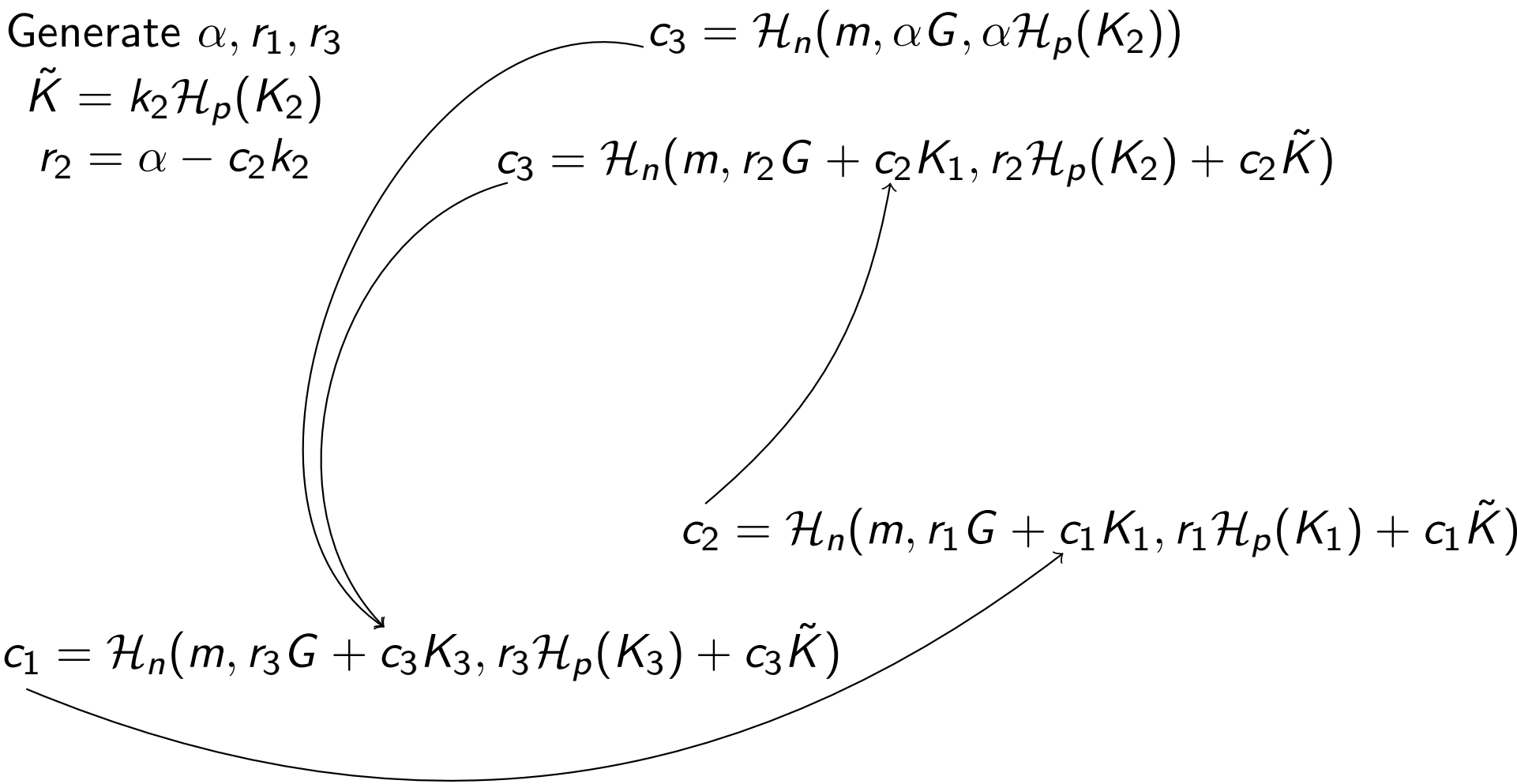
$R = \{K_1, K_2, K_3\}$

Signer: (k_2, K_2)

Generate α, r_1, r_3

$\tilde{K} = k_2 \mathcal{H}_p(K_2)$

$r_2 = \alpha - c_2 k_2$



Confidential transactions - Pedersen commitments

- Goal: 'Encrypt' transaction amounts
- Cryptographic commitments allow you to commit to a specific value without revealing the value
- Pedersen commitments: Preserve values under addition
- Assuming $a = b$:

$$aG - bG = (a - b)G = 0$$

$$C_a - C_b = C_{a-b} = 0$$

Confidential transactions

- If we use plain amounts, we can construct lookup tables
- Solution: Add 'blinding factors' x and y: $x - y = 0$
- Use a new generator point $H = \gamma G$ (with unknown scalar γ) for the amounts
- In Monero's case $H = \mathcal{H}_p(G)$

$$(aH + xG) - (bH + yG) = xG - yG = 0$$

$$C_a^x - C_b^y = C_{a-b}^{x-y} = 0$$

Confidential transactions - Monero

- If we sign a transaction with multiple ring members and have commitments that subtract to zero we can detect which commitment in the signing ring is constructed by us
- Solution, do not use $x{=}y$:

$$(aH + xG) - (bH + yG) = zG$$

$$C_a^x - C_b^y = C^{x-y} = C^z = zG$$

- Sender only needs to prove knowledge of z with a signature

Confidential transactions - usage

- Raw amount and blinding factors are encrypted with Diffie-Hellman
- The 'zero commitment' zG is treated as an extra member in the Ring of transaction public keys
- Network only needs to check if the sender knows z by checking its signature
- If the sender forges the amounts, he wont be able to produce a valid signature for zG. Observe for forged amounts $a - b = c$:

$$aH + xG - bH + yG = zG + cH$$

$$C_a^x - C_b^y = C^z + C_c$$

- Which can't be checked, since the network only checks signatures with G

Monero addresses

- Users of Monero have two key pairs
- View key K^v and spend key K^s
- A Monero address is the checksummed concatenation of the public 'view key' and public 'spend key'

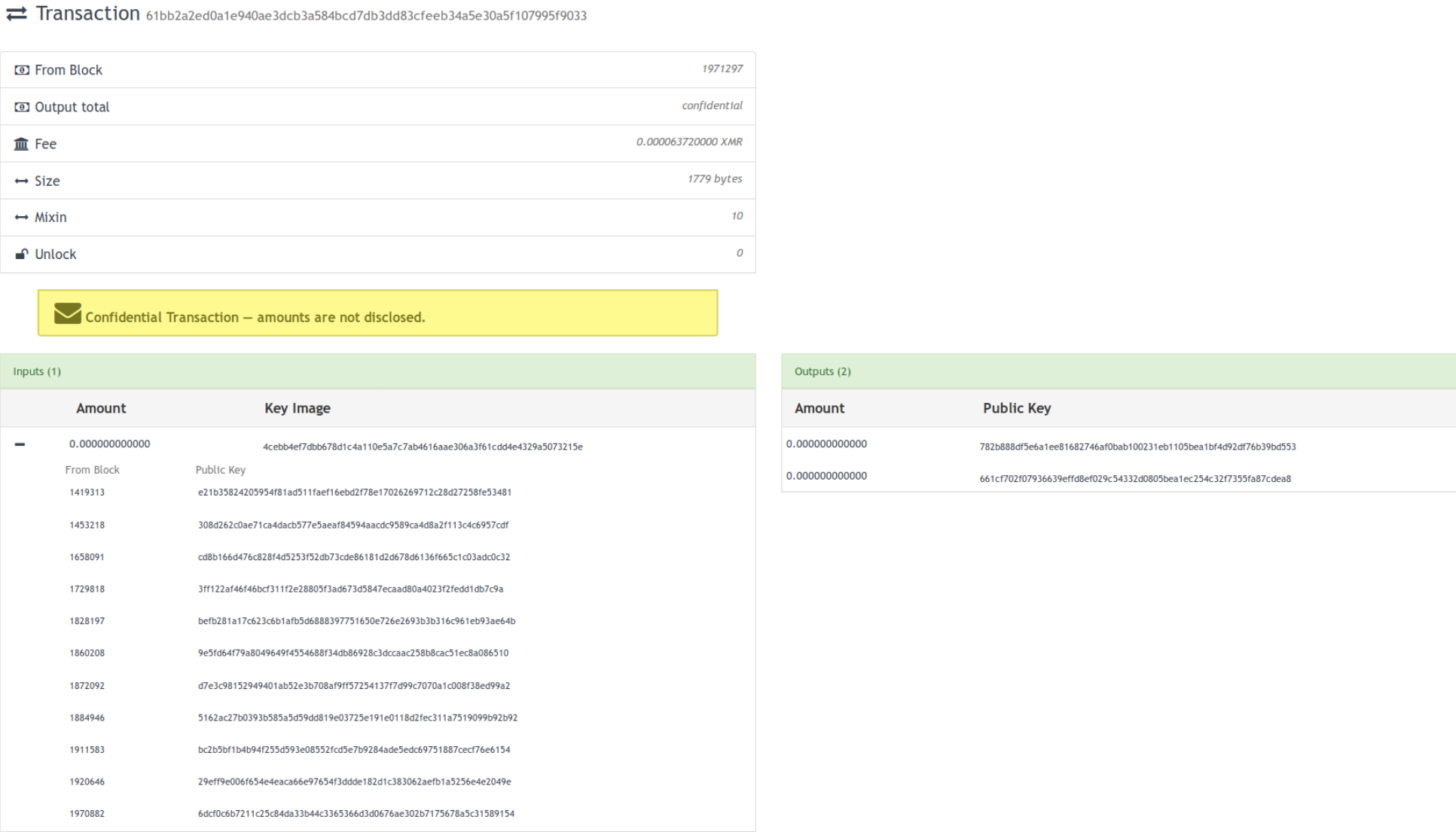
One-Time addresses

- Recipient with pairs (k^v, K^v) and (k^s, K^s)
- Sender generates random scalar number r
- Then calculates one-time key:

$$K^\circ = \mathcal{H}_n(rK^v)G + K^s$$

- Sender adds K° and rG to the transaction data
- Recipient receives the transaction and uses his unique knowledge of $k^v rG = rK^v$
- When he sees that $K^\circ - \mathcal{H}_n(k^v rG)G = K^s$, he knows the transaction is addressed to him

Monero transaction information



Transaction 61bb2a2ed0a1e940ae3dcb3a584bcd7db3dd83cfeeb34a5e30a5f107995f9033

From Block	1971297
Output total	confidential
Fee	0.000063720000 XMR
Size	1779 bytes
Mixin	10
Unlock	0

Confidential Transaction – amounts are not disclosed.

Inputs (1)

	Amount	Key Image
−	0.000000000000	4cebb4ef7dbb678d1c4a110e5a7c7ab4616aae306a3f61cdd4e4329a5073215e

From Block	Public Key
1419313	e21b35824205954f81ad511faef16ebd2f78e17026269712c28d27258fe53481
1453218	308d262c0ae71ca4dacb577e5aeaf84594aacdc9589ca4d8a2f113c4c6957cdf
1658091	cd8b166d476c828f4d5253f52db73cde86181d2d678d6136f665c1c03adc0c32
1729818	3ff122af46f46bcf311f2e28805f3ad673d5847ecaad80a4023f2fedd1db7c9a
1828197	befb281a17c623c6b1afb5d6888397751650e726e2693b3b316c961eb93ae64b
1860208	9e5fd64f79a8049649f4554688f34db86928c3dccaac258b8cac51ec8a086510
1872092	d7e3c98152949401ab52e3b708af9ff57254137f7d99c7070a1c008f38ed99a2
1884946	5162ac27b0393b585a5d59dd819e03725e191e0118d2fec311a7519099b92b92
1911583	bc2b5bf1b4b94f255d593e08552fcd5e7b9284ade5edc69751887cecf76e6154
1920646	29eff9e006f654e4eaca66e97654f3ddde182d1c383062aefb1a5256e4e2049e
1970882	6dcf0c6b7211c25c84da33b44c3365366d3d0676ae302b7175678a5c31589154

Outputs (2)

Amount	Public Key
0.000000000000	782b888df5e6a1ee81682746af0bab100231eb1105bea1bf4d92df76b39bd553
0.000000000000	661cf702f07936639effd8ef029c54332d0805bea1ec254c32f7355fa87cdea8

Can we do better?

- There are still open attacks against Monero's privacy
- Perfect Privacy in a system is probably impossible
- It is a constant cat and mouse game and this is what makes it fun!

EABE attack

- EABE transactions are a useful model for transaction privacy
- Eve-Corp, an exchange collecting kyc information on its users, allows Alice to withdraw 3.2415 XMR
- Alice then makes a purchase at Bob's shop for 3.2415 XMR
- Bob uses Eve-Corp's exchange for custody of his coins and sends the 3.2415 XMR back to Eve
- Eve can correlate to effectively de-anonymize transactions between Alice and Bob
- Problem: Ringsize 11 is not enough

EABE attack

- Solution 1: Bigger ring sizes!
- Remember: Current ring signatures scale linearly, since responses (random scalars) are added individually to the signature; $\sigma(m) = \{c_1, r_1, r_2, ..., r_n\}$
- Ringsize $\approx 10^2$ could be a future, but is still work in progress
- Active Research, three slightly different proposals (Omniring, Lelantus, RCT3), each with different trade-offs
- Next optimization (CLSAG) probably in the ballpark of ring size 13

EABE attack

- Solution 2: DLSAG $\rightarrow$ add a second key to the key image with a Diffie-Helman: $\tilde{K} = k_A k_B G\mathcal{H}_p(K_1)$
- The second key can be encumbered by a spending condition t, which corresponds to a block height.
- Add an extra commitment to zero for the locktime
- This allows for the construction of a "Lightning Network" on Monero
- Comes with slight increase in transaction size

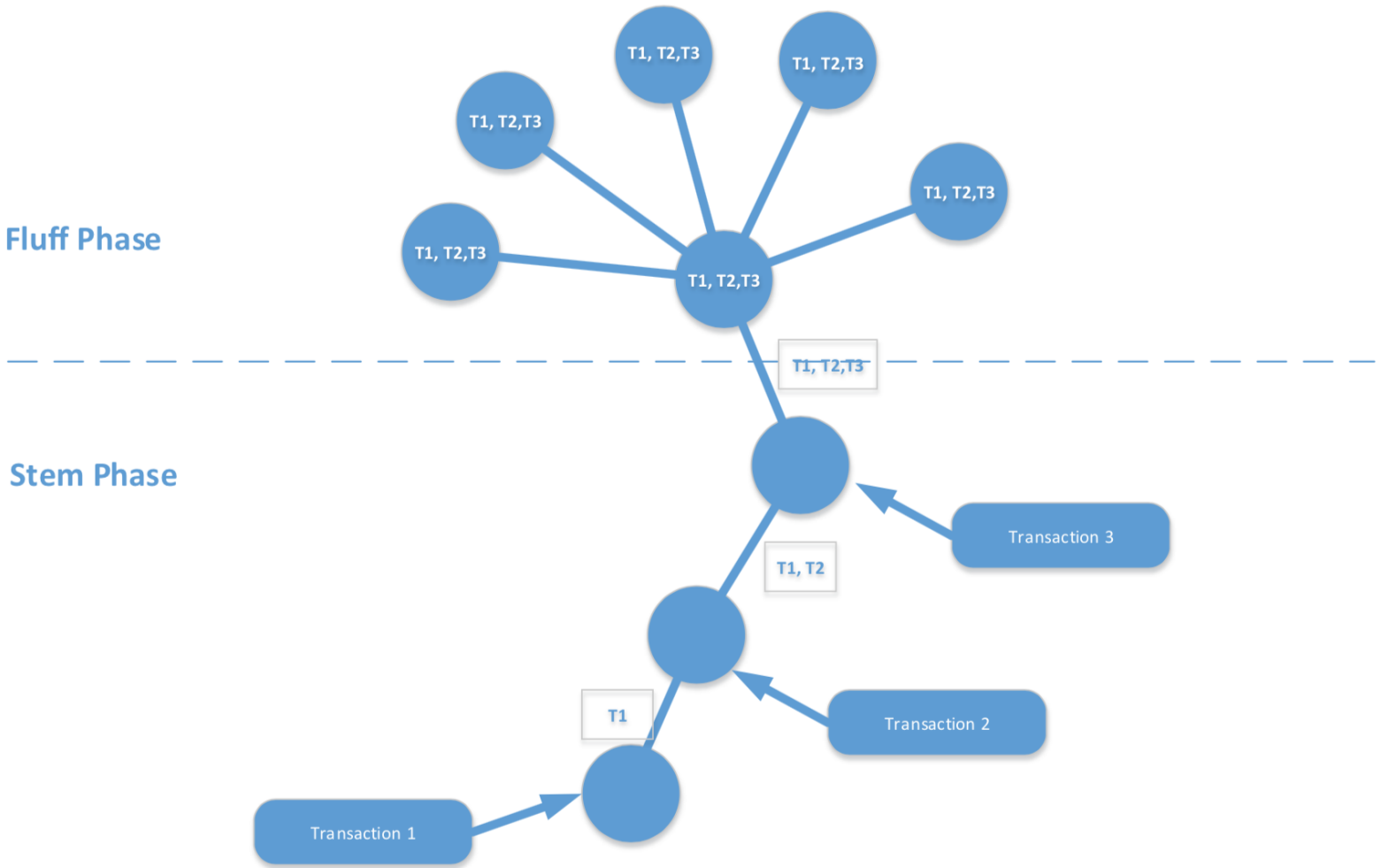
P2P layer attacks

- Problem: P2P call response times give information on the P2P node's transaction state
- Solution: Clear domain separation between wallet and network daemon

P2P layer attacks

- Problem: Enough Sybils inside the network effectively de-anonymize broadcasting
- Solution: "Dandelion" for transaction relay

Dandelion


Fluff Phase
Stem Phase
T1, T2,T3
T1, T2,T3
T1, T2,T3
T1, T2,T3
T1, T2,T3
T1, T2,T3
T1, T2,T3
T1, T2
T1
Transaction 3
Transaction 2
Transaction 1

P2P layer attacks

- Problem: Enough Sybils inside the network effectively de-anonymize broadcasting
- Solution: "Dandelion" for transaction relay
- Add "white noise" to the P2P chatter
- mixnets, I2P and Tor
- But beware: Different packet padding between the protocols gives away packet contents

Anonymity puddles

- Problem: Some transactions stick out from others
- locktime, fees, mining time, decoy selection, spending maturity
- Monero used to have a variable ring size, but later made constant to increase uniformity
- Some puddles, like consolidation transactions, are hard to mitigate

Scaling issues

- All key images need to be in memory
- Restoring a wallet from a seed takes a long time, since every transaction needs to be scanned
- Current transaction format is quite space intensive and pales in compactness compared to Bitcoin and Ethereum

Scaling

- As mentioned, DLSAG and payment networks might be a future
- Variable block size that increases and decreases with demand
- Tail inflation, meaning there will always be a mining subsidy

Developing

- Written in c++
- RPC interface, cli tool, gui wallet
- Functionality split between network daemon and wallet
- All functionality exposed in separate libraries
- Bindings available in Java, Python, JavaScript, Rust...

Contributing

- Github
- Community Crowdfunding System
- Monero Research Lab

References I

- *Compact linkable ring signatures andapplications*, https://eprint.iacr.org/2019/654.pdf, Last Accessed: 2019-11-24.
- *Dandelion onions: Protecting transaction privacy in monero*, https://www.youtube.com/watch?v=bxQ0ic1Fccs, Last Accessed: 2019-11-24.
- *Dlsag: Non-interactive refund transactions for interoperable payment channels in monero*, https://eprint.iacr.org/2019/595.pdf, Last Accessed: 2019-11-24.
- *Lelantus: Towards confidentiality and anonymity o fblockchain transactions from standard assumptions*, https://eprint.iacr.org/2019/595.pdf, Last Accessed: 2019-11-24.

References II

- *Omniring: Scaling private paymentswithout trusted setup*, https://eprint.iacr.org/2019/580.pdf, Last Accessed: 2019-11-24.
- *Research on anonymity puddles by noncesense research lab*, https://github.com/noncesense-research-lab, Last Accessed: 2019-11-24.
- *Randomx proof of work function*, https://github.com/tevador/RandomX/blob/master/doc/specs.md, Last Accessed: 2019-11-24.
- *Ringct 3.0 for blockchain confidential transaction:shorter size and stronger security*, https://eprint.iacr.org/2019/508.pdf, Last Accessed: 2019-11-24.

References III

- *Ring confidential transactions*, https://lab.getmonero.org/pubs/MRL-0005.pdf, Last Accessed: 2018-11-17.
- *An efficient implementation of monero subaddresses*, https://lab.getmonero.org/pubs/MRL-0006.pdf, Last Accessed: 2018-11-17.
- *Thring signatures and their applications to spender-ambiguous digital curencies*, https://eprint.iacr.org/2018/774.pdf, Last Accessed: 2018-11-17.
- *Zero to monero*, https://www.getmonero.org/library/Zero-to-Monero-1-0-0.pdf, Last Accessed: 2018-11-17.

[Zer] [Thr] [Rin] [Sub] [Ran] [DLS] [Omn] [RCT] [Lel] [Pud] [?] [CLS] [Dan]