Zero-Knowledge Proofs

Rami Khalil
rami.khalil@imperial.ac.uk

https://www.doc.ic.ac.uk/~nd/peng

Lecture Outline

1. Proving Systems
2. Interactive Proofs
3. Proofs of Knowledge
4. Zero-Knowledge Proofs of Knowledge
5. Arguments (of Knowledge)
6. zkSNARKS

1. Proving Systems

Abstractly speaking, a (valid) proof is information that leads to the acceptance of the truth of a statement.

Proving systems are typically characterized by two main properties: *soundness* and *completeness*.

1.1. Soundness

Result \ Truth	**Yes**	**No**
Yes	YY	***NY***
No	YN	NN

Soundness error: Prob(NY).

1.2. Completeness

Result \ Truth	Yes	No
Yes	YY	NY
No	***YN***	NN

Completeness error: Prob(YN).

2. Interactive Proofs

P → V: I am user blabla9000.
V → P: Sign his birthday with his public key.
P → V: Sig(day, month, year)
V → P: Pass

A communication transcript from an interactive identity proving system.

Interactive proof system: A *prover* **P** proves to the *verifier* **V** that a statement is true.

2.1. Degenerate IP for NP Relations

$$P \rightarrow V: x \in L_R \in NP, y \mid (x,y) \in R$$

A simple unidirectional interactive proof (IP) system for statements of the form "$x \in \mathbf{L_R}$" for any NP language $\mathbf{L_R}$ can simply be constructed by requiring that the prover **P** submits the *witness* y to **V**, who then accepts or rejects the statement based on the output of the membership decision algorithm.

2.2. Non-degenerate IP for GNI

$P \rightarrow V: G_1 \neq G_2$

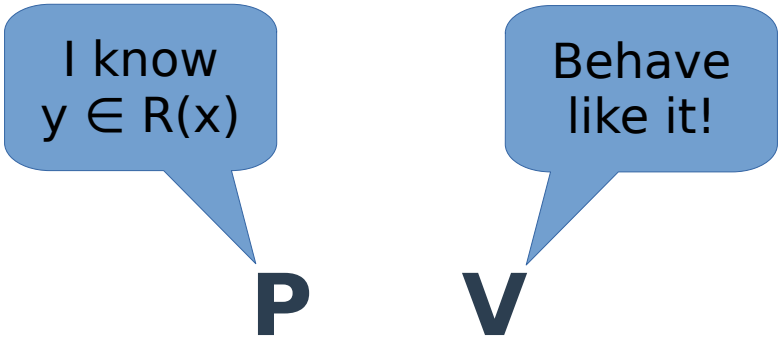
$V \rightarrow P: H \text{ s.t. } H \simeq G_1 \text{ OR } H \simeq G_2$

$P \rightarrow V: N \text{ s.t. } G_N \simeq H$

$V \rightarrow P: \text{Pass iff } G_N \simeq H$

While this system's completeness error (Prob[***YN***]) will be zero, its soundness error (Prob[***NY***]) will be ½.

3. Proofs of Knowledge

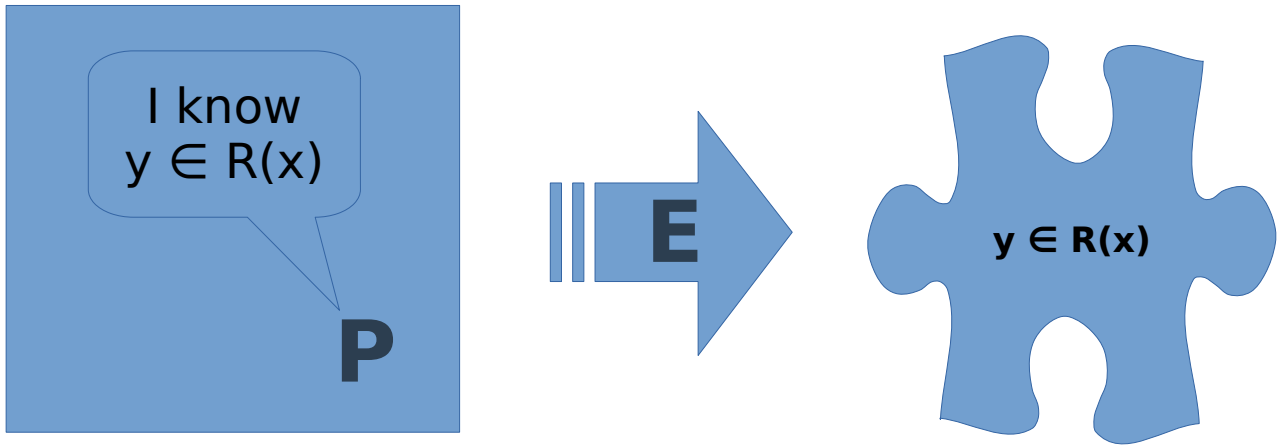


The term “knowledge” is used to refer to results that are computationally hard to derive using public information.

PoKs are mainly concerned with convincing a *non-trivial knowledge verifier* **V** that for a given x ∈ $\mathbf{L_R}$, the statement “**P** ‘knows’ y ∈ **R**(x)” is true, where **R**(x) is the set of all witnesses for x.

V being non-trivial for **R** implies that there exists a **P** that will always convince **V** of its knowledge of a witness for all x ∈ $\mathbf{L_R}$.

3.1. Knowledge Extraction



P is considered by a *non-trivial knowledge verifier* **V** to "know" a witness for x ∈ $\mathbf{L_R}$ iff there exists a *universal knowledge extractor* **E** that can use any **P** *(as an oracle)* to successfully compute a witness y ∈ **R**(x) in expected polynomial time with some probability.

knowledge error k(|x|): the probability of **P** convincing **V**, despite **P** not fully knowing a witness for x.

3.2. Extracting Graph Isomorphisms

Knowledge: A φ s.t. $\varphi(G_1) = G_2$

P $\rightarrow$ V: $H = \psi(G_2)$, where ψ is a random relabel

V $\rightarrow$ P: $c \in \{1,2\}$

P $\rightarrow$ V: $\omega = \psi \circ \varphi$ if $c = 1$ otherwise $\omega = \psi$

V $\rightarrow$ P: Pass iff $\omega(G_c) = H$

4. Zero-Knowledge Proofs of Knowledge

P → **V**

V → **P**

...

V: OK / NOK

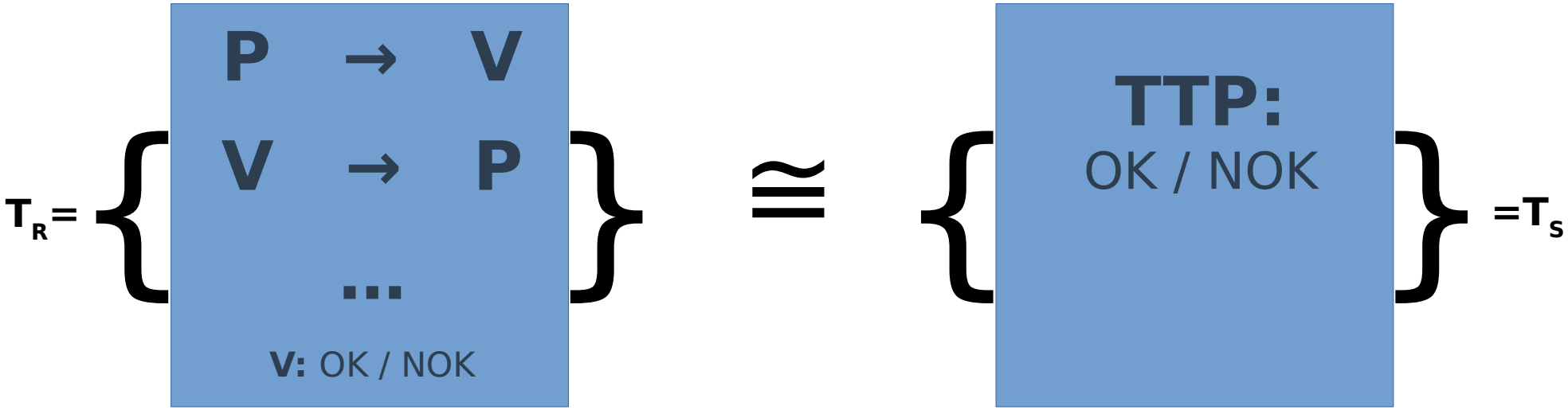
≅

TTP:
OK / NOK

A ZKP of a statement is intended to be “equivalent” to a trusted third party simply asserting the truth of the statement.

The main goal of a ZKPoK system is to convince a non-trivial knowledge verifier **V** that **P** “knows” (as previously defined) a witness for $x \in \mathbf{L_R}$ but without letting **V** “gain” *any* knowledge.

4.1. Zero-Knowledge



Perfect zero-knowledge: $\mathbf{T_S}$ identical to $\mathbf{T_R}$
Statistical zero-knowledge: $\mathbf{T_S}$ statistically equivalent to $\mathbf{T_R}$
Computational zero-knowledge: No PPT algorithm can (significantly) change its output when given (sufficiently large) members of $\mathbf{T_S}$ as input rather than (equally sized) members of $\mathbf{T_R}$ (or vice versa).

4.2. Simulating Graph Isomorphism

Knowledge: A φ s.t. $\varphi(G_1) = G_2$

P $\rightarrow$ V: $H = \psi(G_2)$, where ψ is a random relabel

V $\rightarrow$ P: $c \in \{1,2\}$

P $\rightarrow$ V: $\omega = \psi \circ \varphi$ if $c = 1$ otherwise $\omega = \psi$

V $\rightarrow$ P: Pass iff $\omega(G_c) = H$

P achieves perfect zero-knowledge as protocol transcripts can be easily simulated by generating an isomorphism *after* **V** has made its choice. The difficulty lies in specifying an adequate simulator which works for a **V** that deviates from our specification. This specification and proof of its sufficiency for perfect ZK is out of the scope of this course.

5. Arguments (of Knowledge)

By relaxing soundness to account for cases where it is computationally “hard” to fool **V** into accepting false statements, we can construct systems for *computationally sound proofs (of knowledge)*, known as *arguments (of knowledge)*.

Essentially, in this model, we only have to consider an arbitrary probabilistic polynomial-time **P** when analysing the system.

5.1. Arguable Isomorphism

Knowledge: A φ s.t. $\varphi(G_1) = G_2$

P $\rightarrow$ V: $h = HASH(\psi(G_2))$, random relabel ψ

V $\rightarrow$ P: $c \in \{1,2\}$

P $\rightarrow$ V: $\omega = \psi \circ \varphi$ if $c = 1$ otherwise $\omega = \psi$

V $\rightarrow$ P: Pass iff $HASH(\omega(G_c)) = h$

While the soundness error remains the same, the system now only provides computational soundness, as an unbounded adversary may break the hashing scheme to reveal an isomorphic copy of the graph **V** chose.

5.2. Non-interactive Arguments

Knowledge: A φ s.t. $\varphi(G_1) = G_2$

P → V: $h = HASH(\psi(G_2))$, random relabel ψ

P → V: $c = R(G_1, G_2, h)$

P → V: $\omega = \psi \circ \varphi$ if $c = 1$ otherwise $\omega = \psi$

V → P: Pass iff $HASH(\omega(G_c)) = h$

NI arguments take the form of the degenerate IP case (one unidirectional prover to verifier interaction), but with the added benefit of allowing **P** and **V** to access a *random oracle* **R**, usually realized in the form of a "good" cryptographic hashing function.

6. zkSNARKS

	Fractal	**Halo**	**Bullet Proofs**	
SONIC	**Groth16**	**SLONK**	**Sapling**	**Marlin**
	PLONK	**Pinnochio**	**Super Sonic**	

Zero-Knowledge *Succinct* Non-interactive Argument of Knowledge systems reduce the protocol communication complexity and/or computational load placed on **V**, which enables exciting applications to be practically realized at the cost of proving time.

6.1. Succinctness

A computationally sound non-interactive knowledge proving system is *succinct* if it enables verifying NP statements with complexity independent from deciding membership in the target NP language.

6.2. Setup

Transparency	**Application**
Trusted or Public	Specific or Universal

Such systems can be built in the *common reference string* model, where **P** and **V** both have access to a string sampled from a uniform distribution.

Consequently, CRS generation (setup) becomes an important step in zkSNARKS. There are many trade-offs in terms of proving costs and/or verification costs between systems with different setup features and requirements.

Full succinctness: The length of the CRS is reasonably “short”.

6.3. Groth16

A *very* popular system for zkSNARK for circuit satisfiability (NP-Complete) based on elliptic curve pairings.

Perfect zero-knowledge.

Constant-sized proofs.

Trusted, application-specific setup.

Used in Zcash.

Groth, Jens. "On the size of pairing-based non-interactive arguments." Annual international conference on the theory and applications of cryptographic techniques. Springer, Berlin, Heidelberg, 2016.


End-of-Text

Zcash: A privacy-protecting cryptocurrency using zkSNARKS

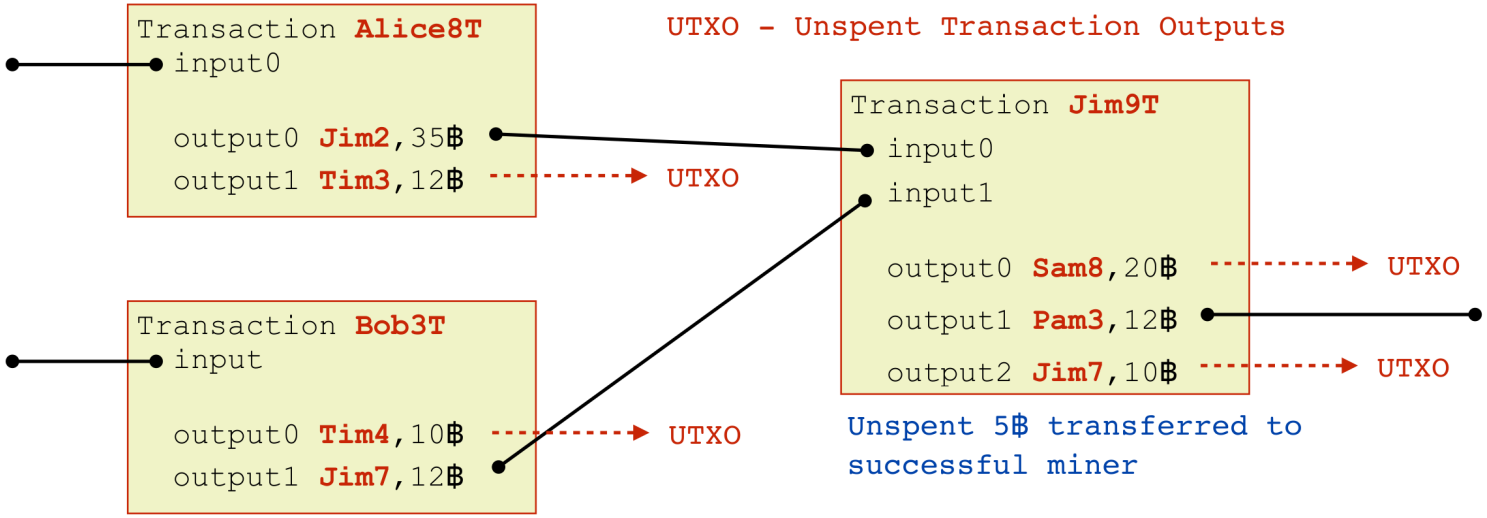
Rami Khalil
rami.khalil@imperial.ac.uk

https://www.doc.ic.ac.uk/~nd/peng

Lecture Outline

1. Transparent pool payments
2. Shielded pool payments
3. Cross-pool payments

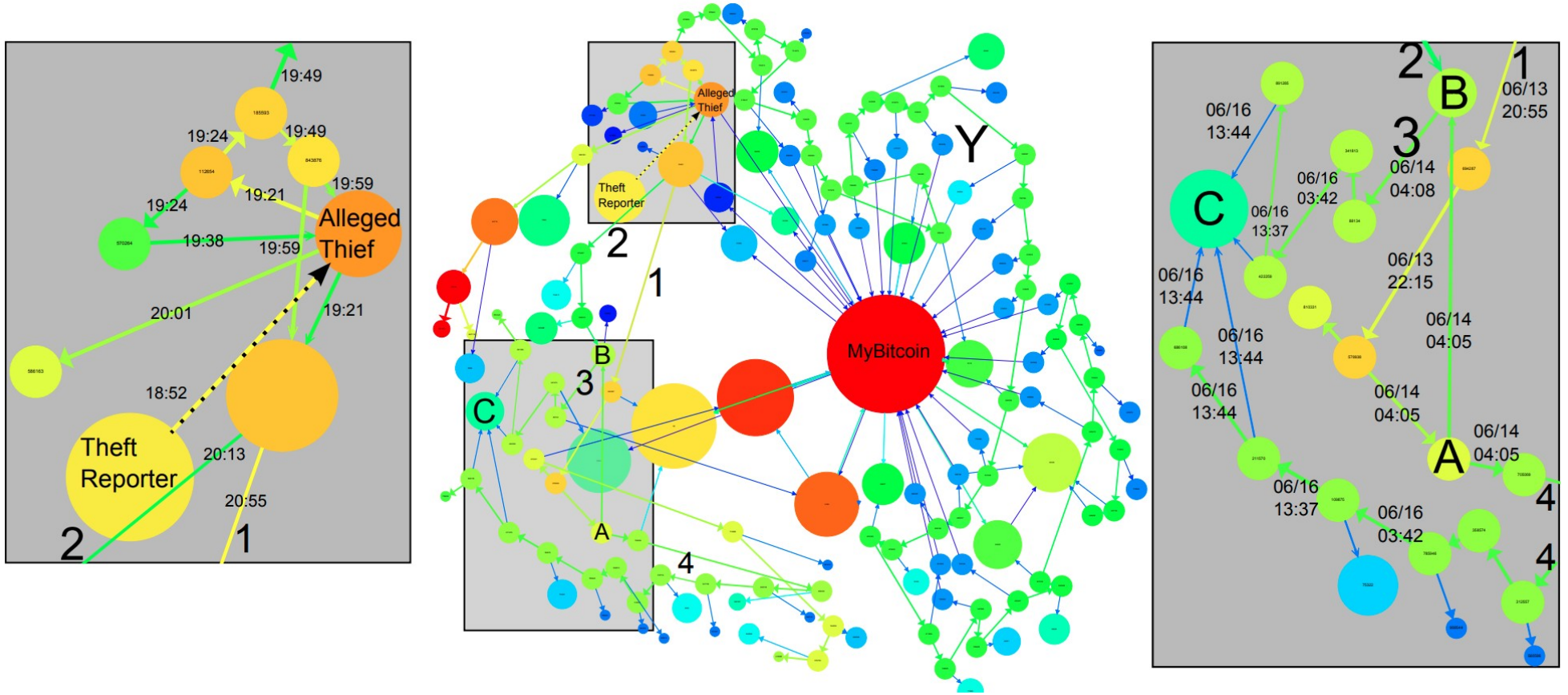
1. Transparent payments



Transparent payments in Zcash essentially work the same way as Bitcoin payments.

Both are executed within an unspent transaction output (UTXO) ledger that provides no privacy.

1.1. Linkability



Reid, Fergal, and Martin Harrigan. "An analysis of anonymity in the bitcoin system." Security and privacy in social networks. Springer, New York, NY, 2013. 197-223.

2. Shielded payments

Shielded Transaction
inputs [?]
outputs [?]

Fee = 0.0001 ZEC

Shielded Transaction
inputs [?]
outputs [?]

Fee = 0.0001 ZEC

Shielded Transaction
inputs [?]
outputs [?]

Fee = 0.0001 ZEC

Language	Statement	Witness
The set of all transactions which only reference unspent outputs as inputs, carry the required signature for each input, and create outputs that do not exceed the value of the inputs..	This transaction is “valid” with respect to the current state of the ledger.	Addresses, keys, values… and?

2.1. Linkability

Shielded transactions support a padded “memo” field, which a sender may use to append arbitrary, potentially identifying, data to a transaction.

Recipients can always learn when they have received a payment, and can learn the value of this payment and its memo, but cannot learn information about the input(s) used to create the transaction, or information about outputs created for other recipients.

Ultimately, to learn any useful information about purely shielded payments, either the sender or the recipient must disclose the knowledge available to it.

3. Bridging payments

This is where potential anonymity leaks begin..

Kappos, George, et al. "An empirical analysis of anonymity in zcash." 27th USENIX Security Symposium (USENIX Security 18). 2018.

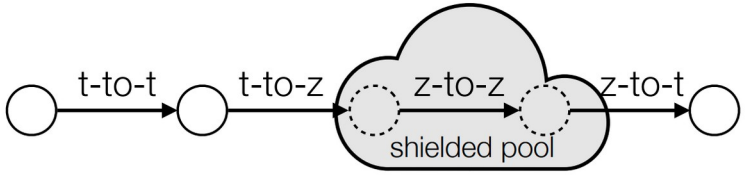


Figure 1: A simple diagram illustrating the different types of Zcash transactions. All transaction types are depicted and described with respect to a single input and output, but can be generalized to handle multiple inputs and outputs. In a t-to-t transaction, visible quantities of ZEC move between visible t-addresses (zIn,$\mathsf{zOut} \neq \emptyset$). In a t-to-z transaction, a visible amount of ZEC moves from a visible t-address into the shielded pool, at which point it belongs to a hidden z-address ($\mathsf{zOut} = \emptyset$). In a z-to-z transaction, a hidden quantity of ZEC moves between hidden z-addresses (zIn,$\mathsf{zOut} = \emptyset$). Finally, in a z-to-t transaction, a hidden quantity of ZEC moves from a hidden z-address out of the shielded pool, at which point a visible quantity of it belongs to a visible t-address ($\mathsf{zIn} = \emptyset$).

3.1. Linkability

If two or more t-addresses are inputs in the same transaction (whether that transaction is transparent, shielded, or mixed), then they are controlled by the same entity.
If one (or more) address is an input t-address in a vJoinSplit transaction and a second address is an output t-address in the same vJoinSplit transaction, then if the size of zOut is 1 (i.e., this is the only transparent output address), the second address belongs to the same user who controls the input addresses.
Any z-to-t transaction carrying 250.0001 ZEC in value is done by the founders.
If a z-to-t transaction has over 100 output t-addresses, one of which belongs to a known mining pool, then we label the transaction as a mining withdrawal (associated with that pool), and label all non-pool output t-addresses as belonging to miners.
For a value v, if there exists exactly one t-to-z transaction carrying value v and one z-to-t transaction carrying value v, where the z-to-t transaction happened after the t-to-z one and within some small number of blocks, then these transactions are linked.

Kappos et al. apply these heuristics (as summarized by Ye et al.) to link senders of shielding payments with recipients of unshielding payments.

The effectiveness of this approach depends on the percentage of users making shielding and unshielding payments.

End-of-Text