Zero-Knowledge Proofs

Rami Khalil
rami.khalil@imperial.ac.uk

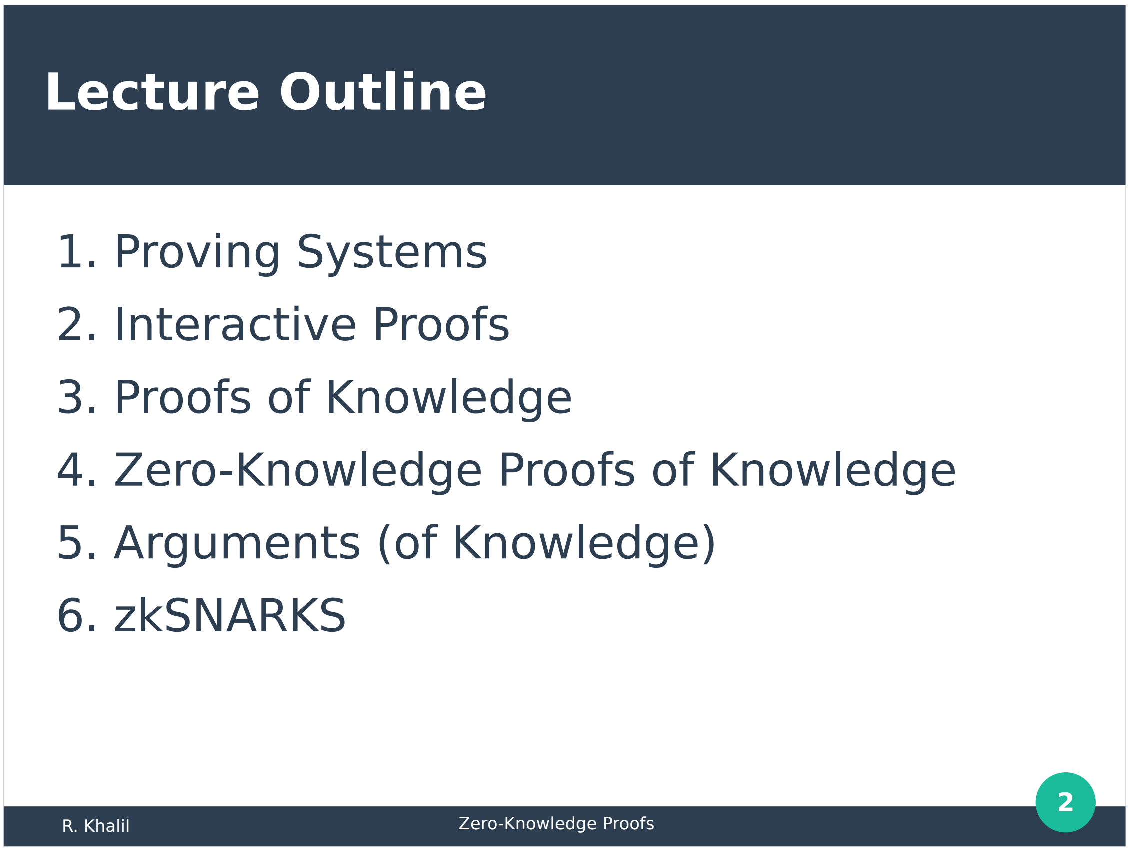
https://www.doc.ic.ac.uk/~nd/peng

I am a Computing PhD Student supervised by Dr. Naranker Dulay at Imperial College London. My research interests lie primarily in Blockchain incentive mechanism design and Off-chain protocol design.

I received my Master's degree in Computer Science from ETH Zurich in 2018, where I focused my courses on Information Security.

Before that, I obtained a Bachelor's degree in Computer Science and Engineering from the German University in Cairo.

This lecture is about ZKPs, an important tool for protocol designers and system engineers concerned with privacy aspects.



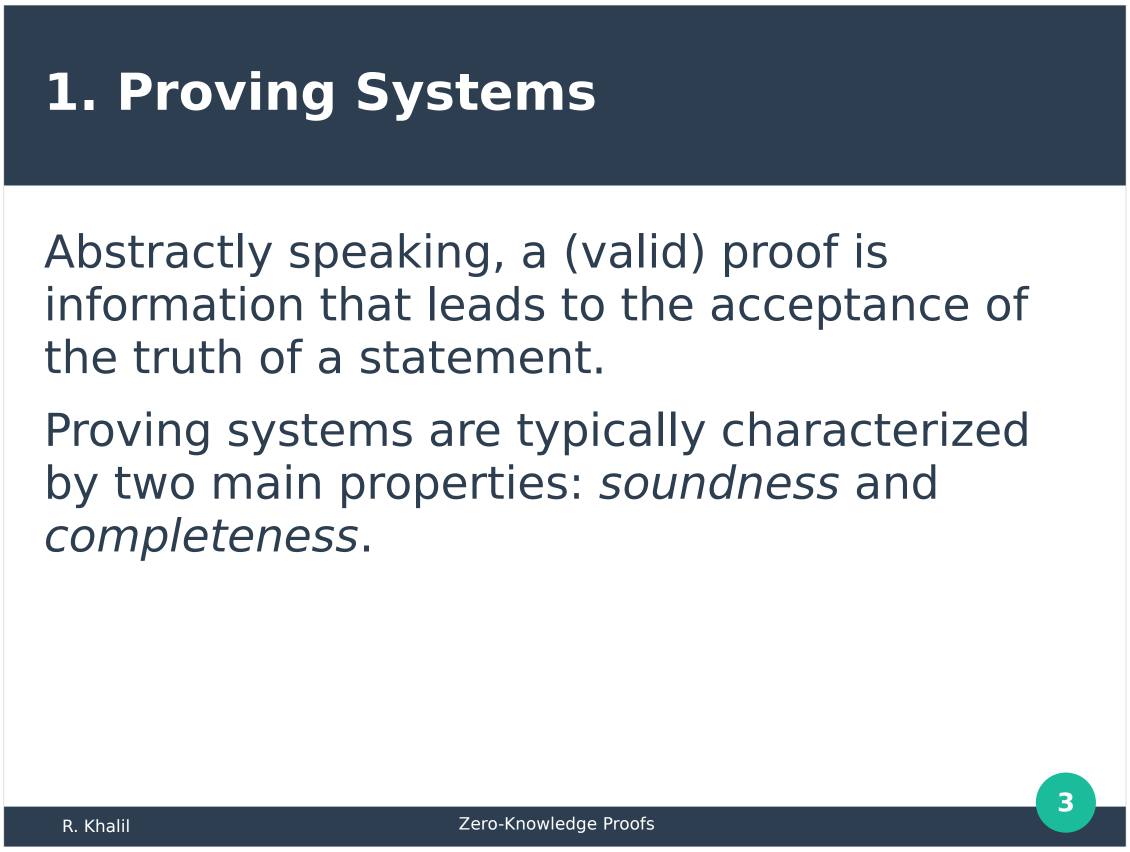
Lecture Outline

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

This lecture is only an introductory guide to the topics mentioned within, and contains many informal simplifications.

Its purpose is to give you basic intuitions that allow you to unpack the many concepts contained within a “fully succinct zero-knowledge argument of knowledge”, a concept we will heavily use to explain the Zcash protocol.

The level of intuition conveyed should be sufficient for you to understand how Zcash provides privacy guarantees, and also gain a contextual understanding that allows you to further explore the covered concepts on your own without being overwhelmed by the depth and breadth of existing systems.

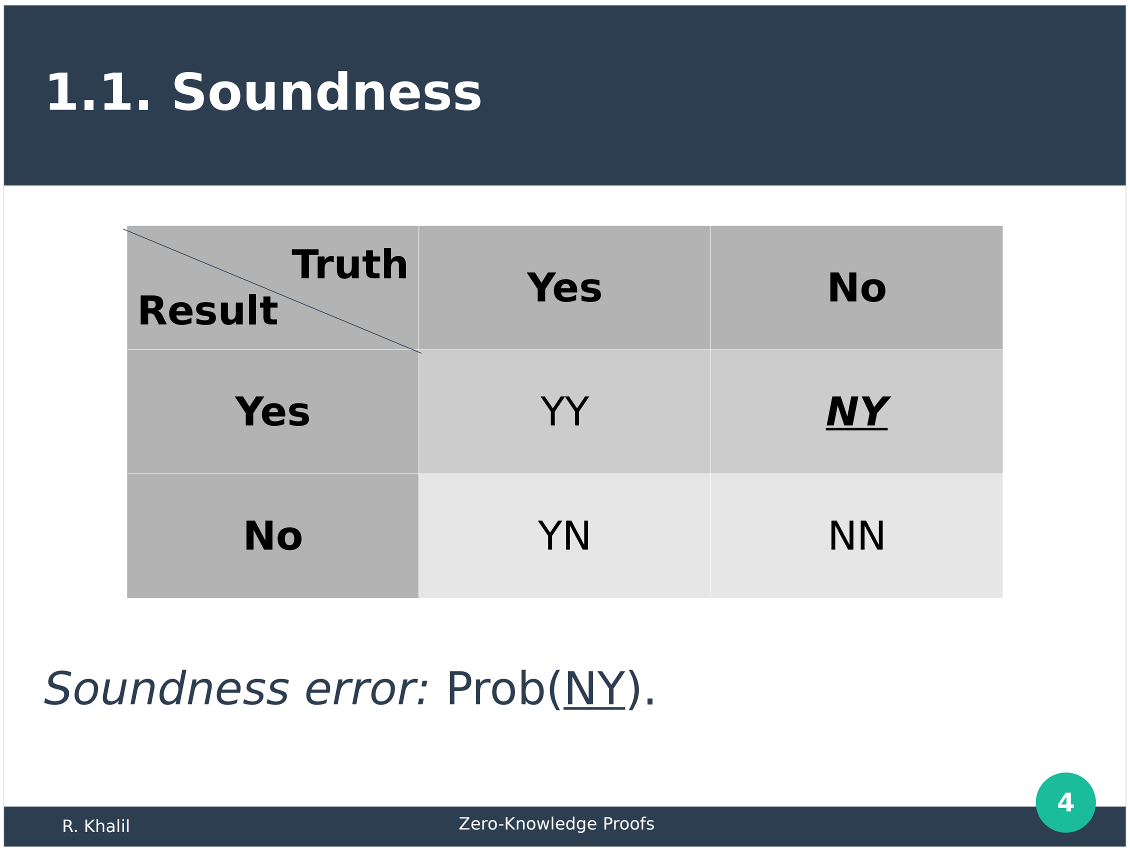

1. Proving Systems

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*.

More concretely, we will discuss proving systems and protocols whereby a verifier **V** validates whether a statement is true.

Peggy & **V**ictor.



1.1. Soundness

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

Soundness error: Prob(NY).

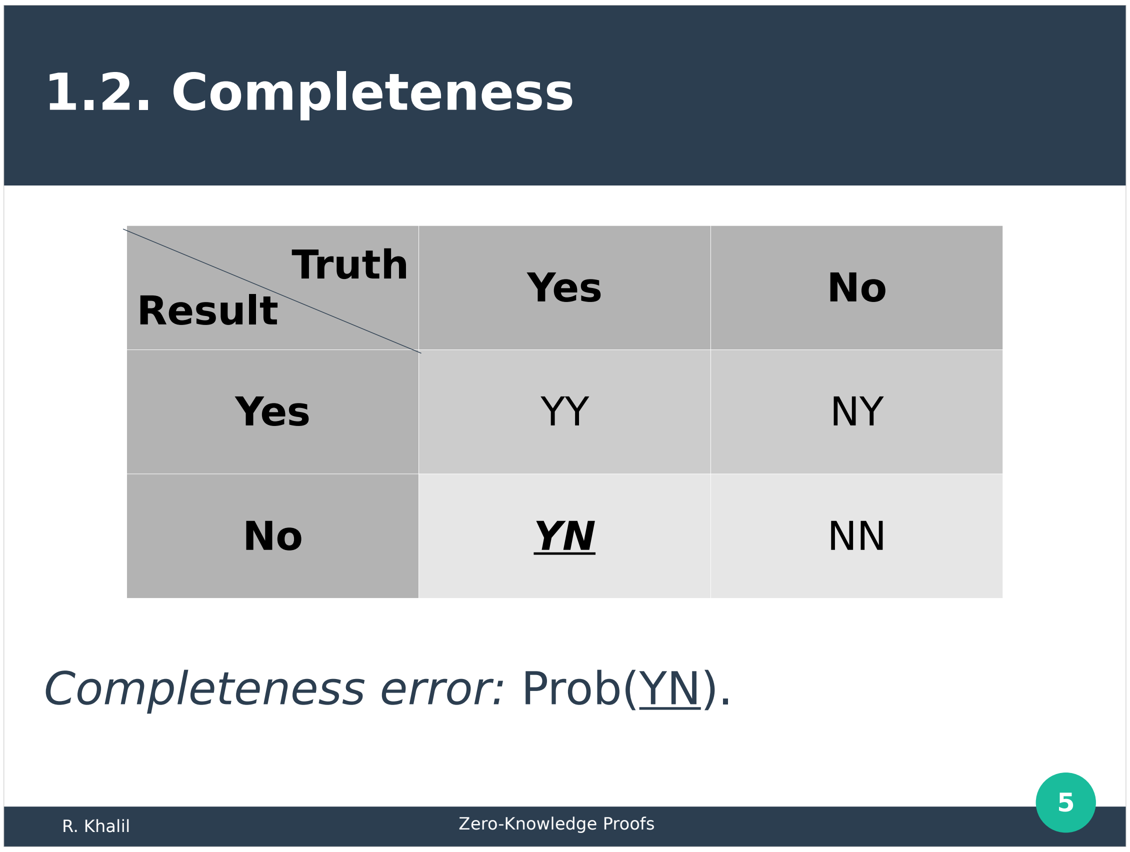
R. Khalil

Zero-Knowledge Proofs

4

The *soundness* of a proving system is quantified by its effectiveness at *preventing* an honest verifier **V** from being convinced of the truthfulness of false statements.

The *soundness error* quantifies the probability of **V** accepting false statements under a given proof system.



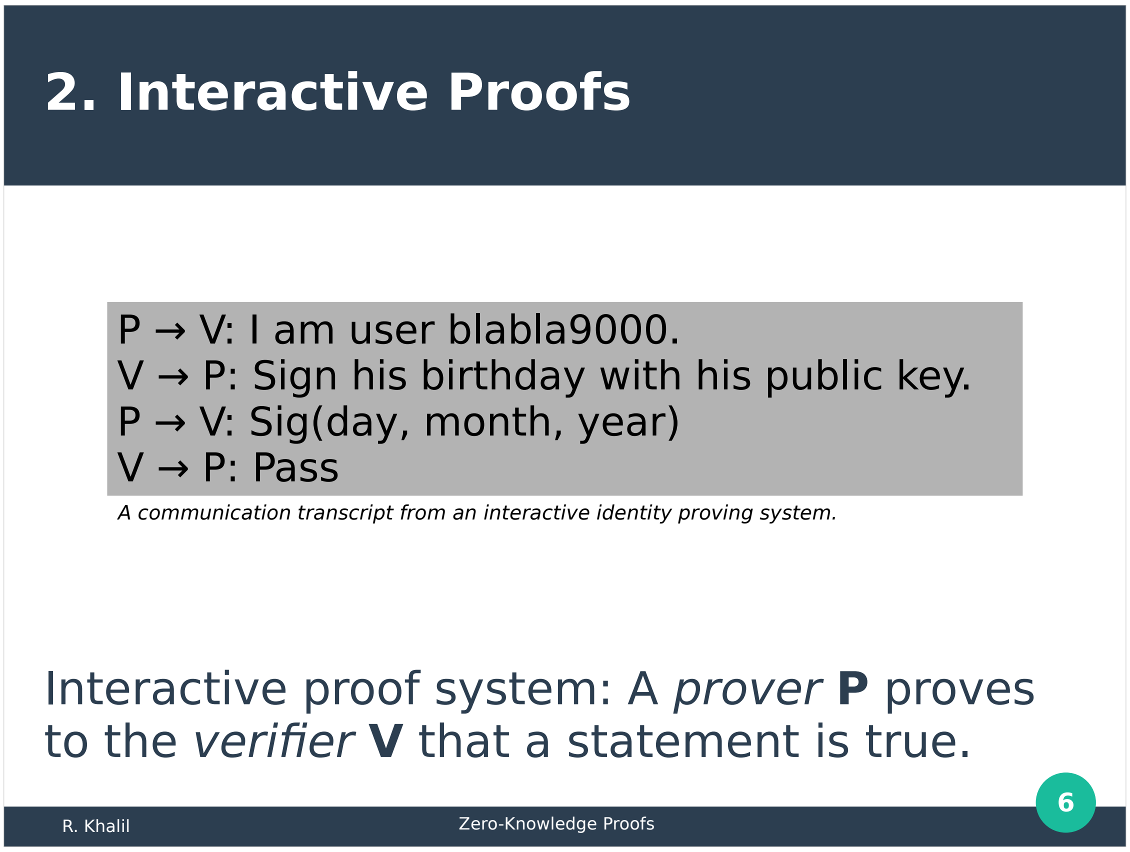
1.2. Completeness

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

Completeness error: Prob(YN).

The *completeness* of a proving system is quantified by its effectiveness at *enabling* an honest verifier **V** of being convinced of the truthfulness of true statements.

The *completeness error* quantifies the probability of **V** rejecting true statements under a given proof system.



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

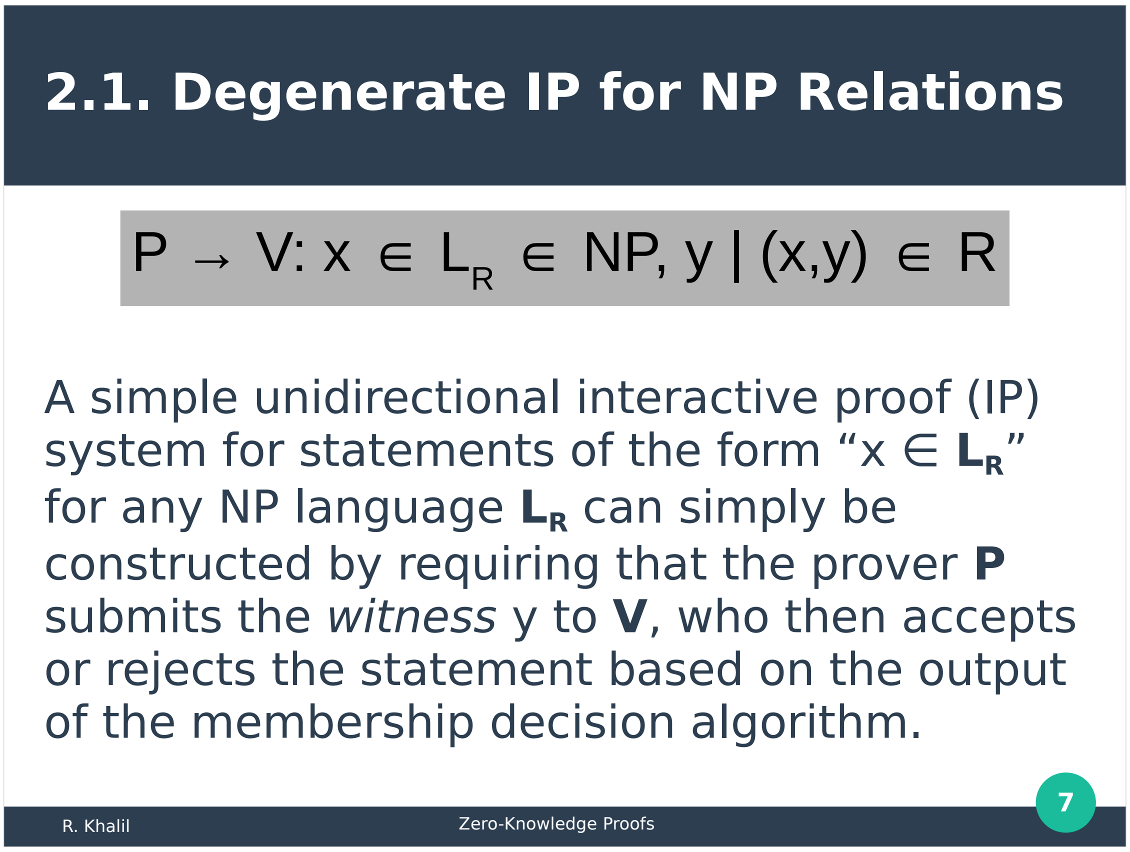
A communication transcript from an interactive identity proving system.

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

Interactive proof systems describe a two-party protocol where one party, the prover **P**, “produces” a proof that a statement is true while interacting with the other party, the verifier **V**, which validates the proof of this statement’s truth.

The duration and complexity of this interaction is not set in stone, but usually takes the form of an “interrogation” of the prover by the verifier.

Is the above loosely defined protocol secure?



2.1. Degenerate IP for NP Relations

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

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.

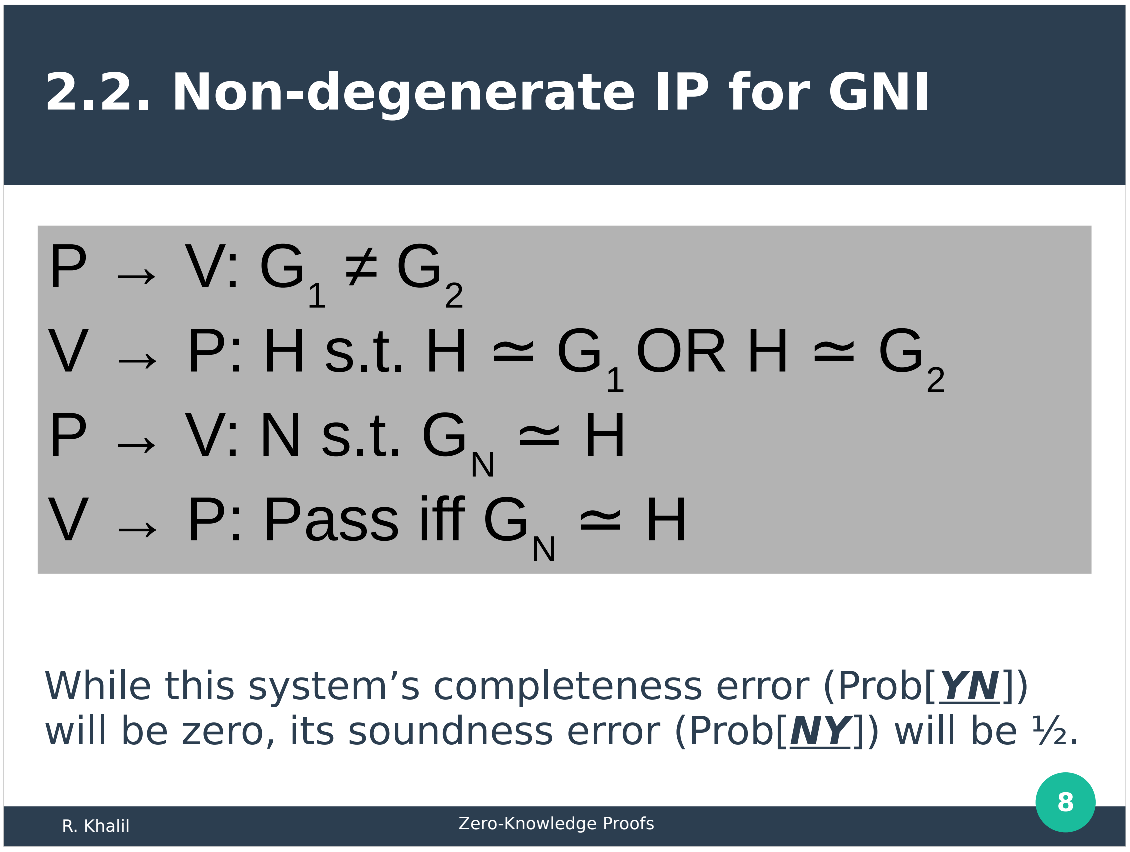
R. Khalil Zero-Knowledge Proofs 7

This is a straightforward IP system that enjoys perfect soundness and perfect completeness, i.e. soundness and completeness errors both equal zero.

Let $\mathbf{R} \subseteq \{0,1\}^* \times \{0,1\}^*$ be a binary relation and let $\mathbf{L_R} = \{x\text{: } \exists y \text{ such that } (x,y) \in \mathbf{R}\}$ be the language defined by **R**.

We say that **R** is an **NP** relation (and $\mathbf{L_R} \in \mathbf{NP}$) iff there exists a polynomial p such that for all $(x,y) \in \mathbf{R}$ it holds that $|y| < p(|x|)$, and there exists a polynomial time algorithm for deciding if $(x,y) \in \mathbf{R}$.

y is called a witness for x iff $(x,y) \in \mathbf{R}$.



2.2. Non-degenerate IP for GNI

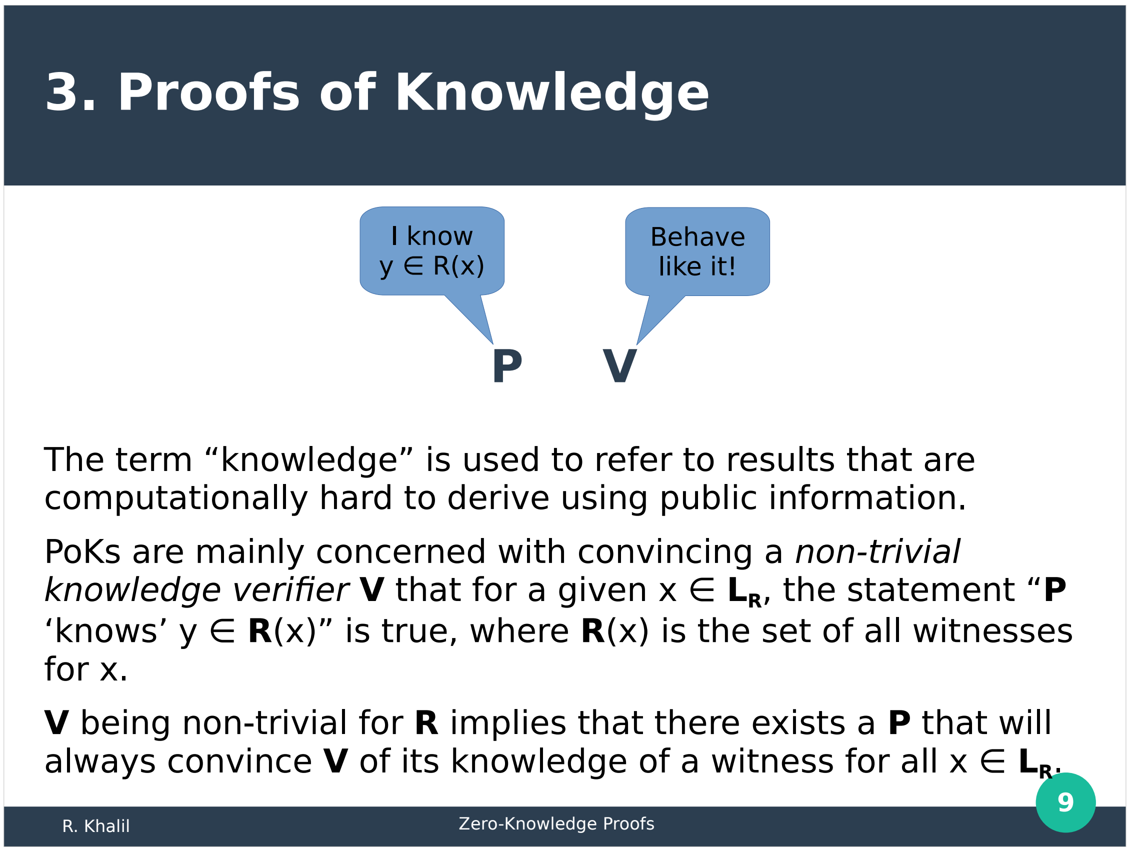
$P \rightarrow V$: $G_1 \neq G_2$
$V \rightarrow P$: H s.t. $H \simeq G_1$ OR $H \simeq G_2$
$P \rightarrow V$: N s.t. $G_N \simeq H$
$V \rightarrow P$: Pass iff $G_N \simeq H$

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

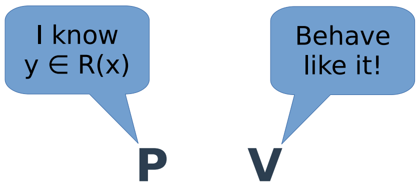
Recall the definition of *graph isomorphism* (GI), where two graphs (G_1, G_2) with the same number of vertices and edges are isomorphic if there exists a relabelling φ of the vertices of one graph that produces the other such that $\varphi(G_1) = G_2$.

We leverage bidirectional interaction to sketch out an interactive proof system for *graph non-isomorphism* (GNI) that allows a prover **P** to convince a verifier **V** that two mutually known graphs are not isomorphic:

V first randomly selects one of the two graphs, and sends a random isomorphic copy H of that graph to **P**. **P** determines (not necessarily efficiently) which of the two mutually known graphs the isomorphic copy corresponds to, and notifies **V** of the result. **V** checks whether **P**'s answer is the mutually known graph that **V** initially randomly selected (i.e. whether **P** guessed **V**'s initial random choice correctly).



3. Proofs of Knowledge

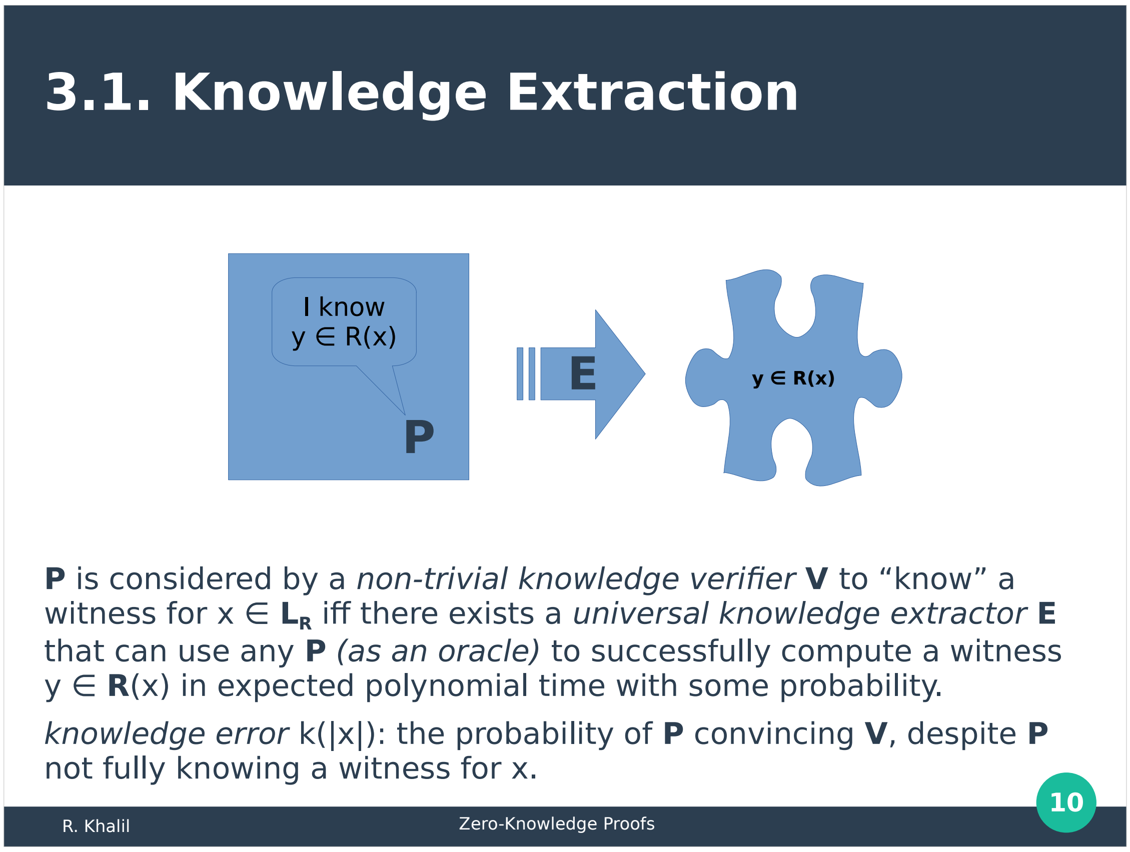


The term “knowledge” is used to refer to results that are computationally hard to derive using public information.

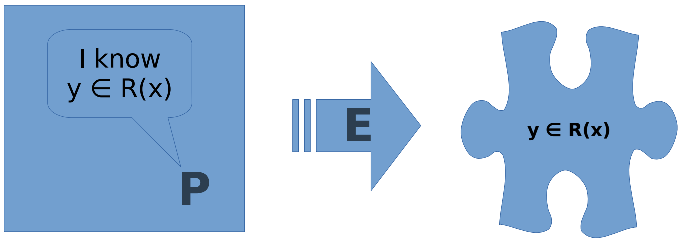
PoKs are mainly concerned with convincing a *non-trivial knowledge verifier* **V** that for a given $x \in \mathbf{L_R}$, the statement “**P** ‘knows’ $y \in \mathbf{R}(x)$” is true, where **R**(x) is the set of all witnesses for x.

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 $x \in \mathbf{L_R}$.

“**P** ‘knows’ ...” does not necessarily imply that the knowledge (such as a private key) is directly available to **P** in a simple form (readable/accessible), but is instead defined in behavioural terms, i.e. **P** behaves as if it has a $y \in \mathbf{R}(x)$.



3.1. Knowledge Extraction

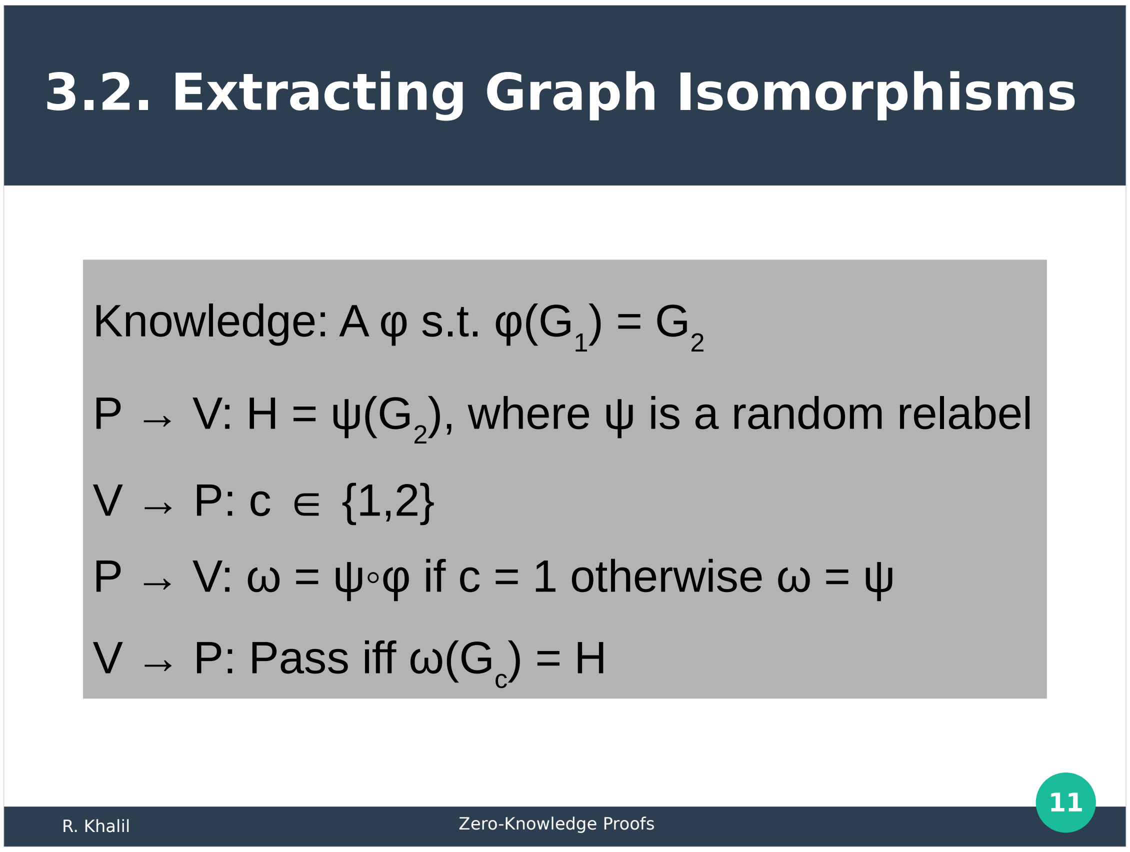


P is considered by a *non-trivial knowledge verifier* **V** to "know" a witness for x ∈ $\mathbf{L}_\mathbf{R}$ iff there exists a *universal knowledge extractor* **E** that can use any **P** *(as an oracle)* to successfully compute a witness y ∈ **R**(x) in expected polynomial time with some probability.

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

As a simplified definition, **P** is considered to "know" a witness for x ∈ $\mathbf{L}_\mathbf{R}$ by a *non-trivial knowledge verifier* **V** 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 probability at least p(x)/q(|x|), where p(x) is the probability of **P** convincing **V** of its knowledge of a witness for x, and q(x) is a positive polynomial.

A broader definition accounts for the possible *knowledge error* k(|x|), which quantifies the probability of **P** convincing **V**, despite **P** not fully knowing a witness for x. When k(x) > 0, the probability of **E** successfully extracting a witness becomes at least (p(x) – k(|x|))/q(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 $\omega(G_c) = H$

R. Khalil

Zero-Knowledge Proofs

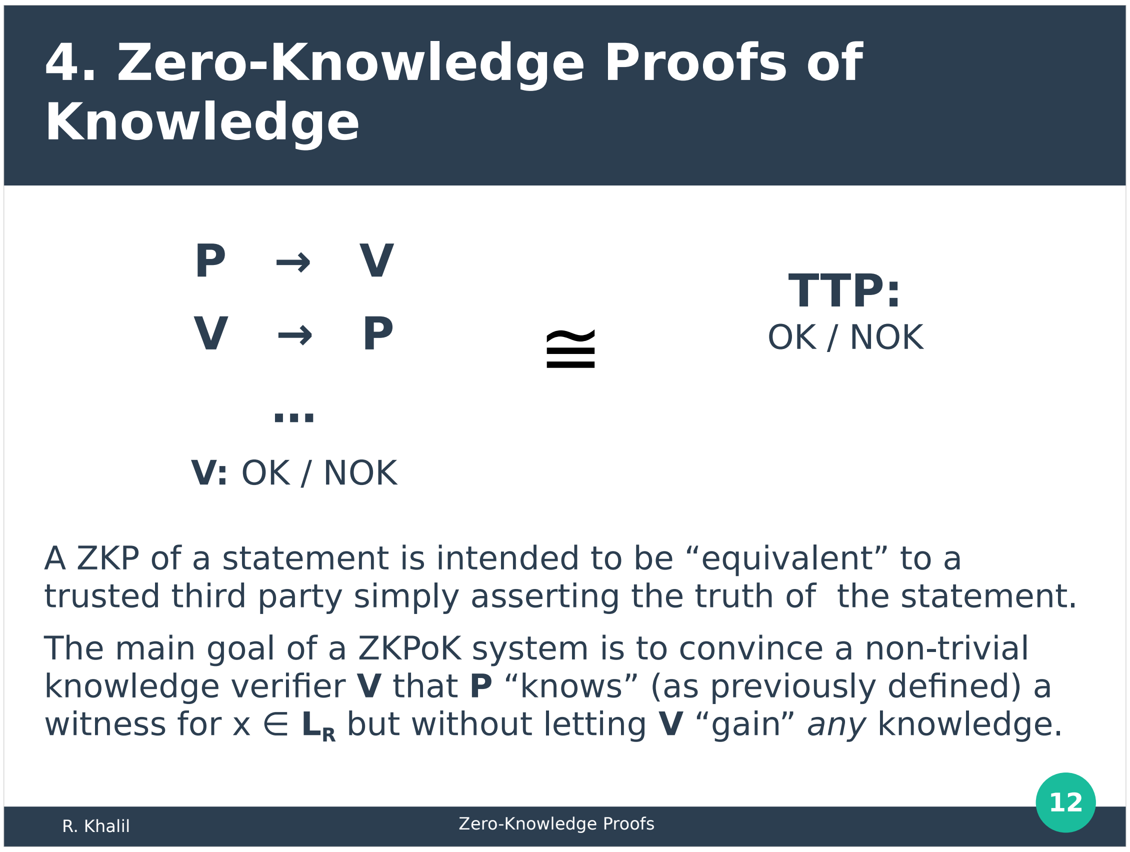
11

Said system enjoys perfect completeness, and the above definition of **P** qualifies **V** as non-trivial. However, the soundness error here is equal to ½.

Consider the following interactive proof of knowledge system of an isomorphism φ on (G_1, G_2): **P** creates a random isomorphism ψ on G_2, and computes the graph $H = \psi(G_2) = \psi \circ \varphi(G_1)$. **P** sends H to **V**, and **V** responds with a random challenge $c \in \{1,2\}$. If $c = 1$, **P** reveals $\psi \circ \varphi$, otherwise **P** reveals ψ to **V**. **V** verifies if the received isomorphism transforms its selected graph to H.

Given said **V**, we can define a knowledge extractor **E** that executes **P** twice using the same randomness. In the first invocation, **E** provides $c = 2$ to **P** (receiving ψ), and in the second invocation, **E** receives $\psi \circ \varphi$ through providing $c = 1$. **E** (very easily) derives ψ^{-1} and outputs $\varphi = \psi^{-1} \circ \psi \circ \varphi$ if $\varphi(G_1) = G_2$. Otherwise, **E** reports failure to derive a witness.

Without φ, **P** may fool **V** with probability ½ on any valid input by correctly predicting **V**'s challenge in advance.



4. Zero-Knowledge Proofs of Knowledge

$$\begin{array}{ccc} \mathbf{P} & \rightarrow & \mathbf{V} \\ \mathbf{V} & \rightarrow & \mathbf{P} \\ & \ldots & \end{array} \quad \cong \quad \textbf{TTP:}\ \text{OK / NOK}$$

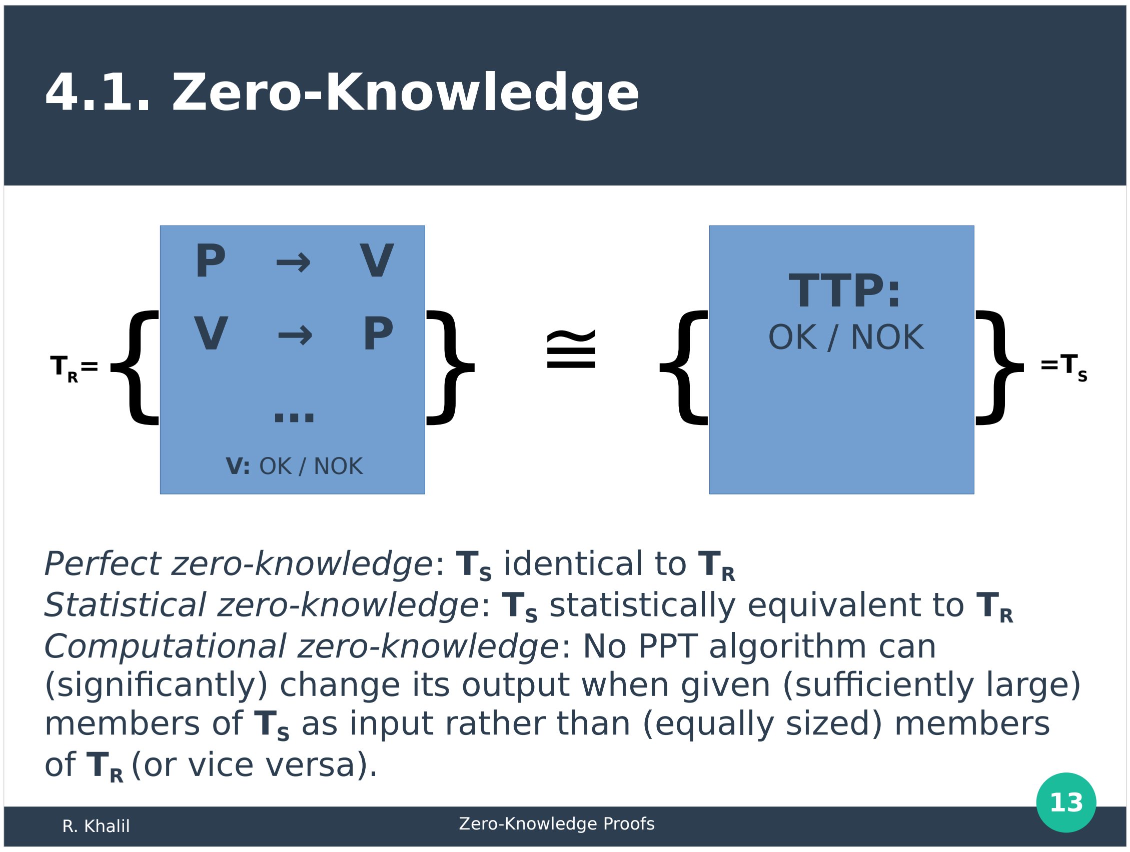
V: OK / NOK

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

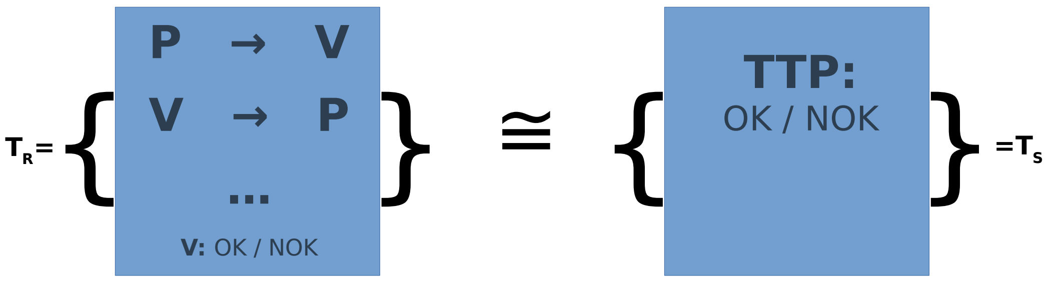
The main goal of a ZKPoK system is to convince a non-trivial knowledge verifier **V** that **P** "knows" (as previously defined) a witness for $x \in \mathbf{L_R}$ but without letting **V** "gain" *any* knowledge.

A party is said to have gained knowledge when it learns the results of a computation that was infeasible for the party to perform.

Note: we do not assess the knowledge gain of **V** only in terms of what was leaked about the witness, but also in terms of how much knowledge was gained about *anything* from interacting with **P**, i.e. we quantify the robustness of **P** against attempts to gain knowledge from interacting with it.



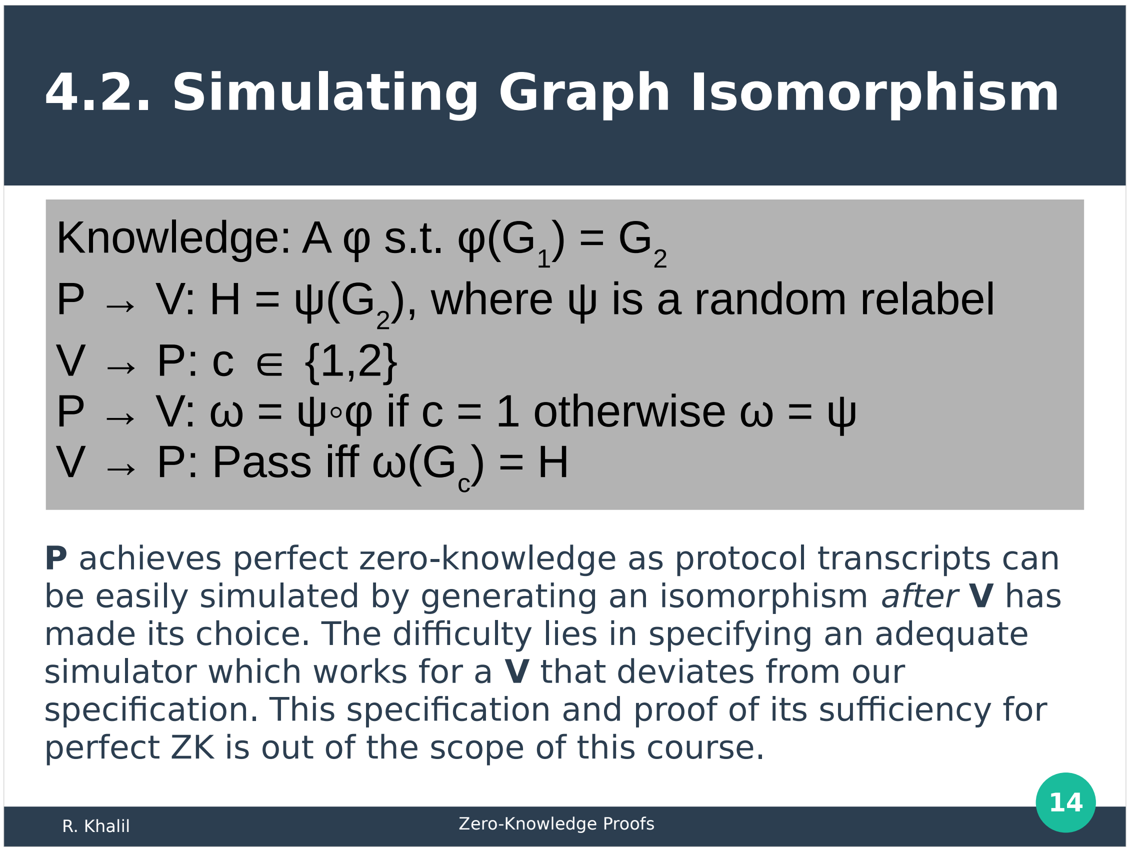
4.1. Zero-Knowledge



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).

The extent to which *any* **V** can gain knowledge through interacting with **P** can be demonstrated through showing that all computational results feasibly reachable by V after interacting with **P** on input x ∈ **L** are feasibly reachable by **V** on input x without interaction with **P**.

More concretely, this “knowledge” equivalence is shown through demonstrating that for *any* **V** there exists a probabilistic polynomial-time *simulator* **S** which can create a set of communication transcripts $\mathbf{T_S} = \{\mathbf{S}_x\}x \in \mathbf{L}$ that is “indistinguishable” from the set of real communication transcripts $\mathbf{T_R} = \{\mathbf{R}_x\}x \in \mathbf{L}$ between **V** and **P**.



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

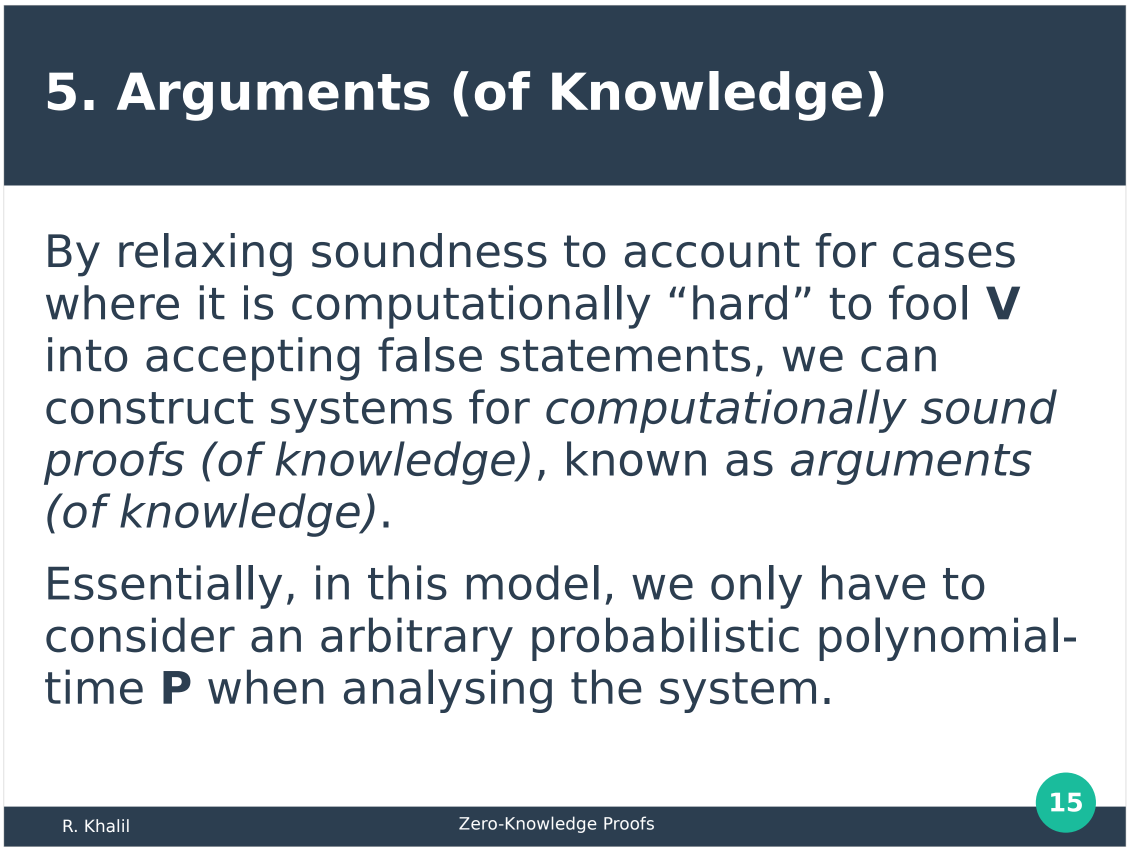
$V \rightarrow P$: $c \in \{1,2\}$

$P \rightarrow V$: $\omega = \psi \circ \varphi$ 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.

Recall the previously described interactive proof of knowledge system of an isomorphism φ on two equally sized graphs (G_1, G_2).

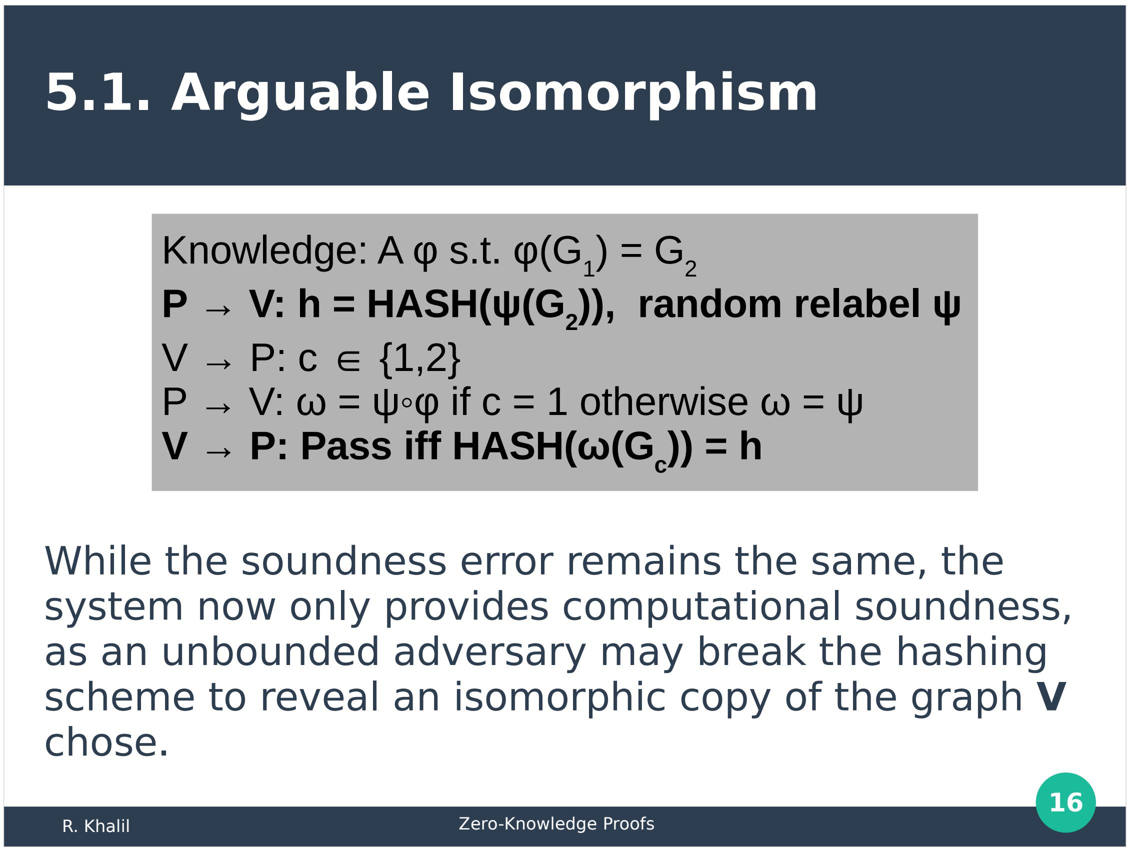


5. Arguments (of Knowledge)

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.

When we know that any **P** under consideration must be “efficient”, we can grant the knowledge extractor **E** access to **P**’s internal state and code without worrying that **E** will become inefficient. This is in contrast to our previous definition of **E**, which permitted only black-box oracle access to **P**.



5.1. Arguable Isomorphism

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

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

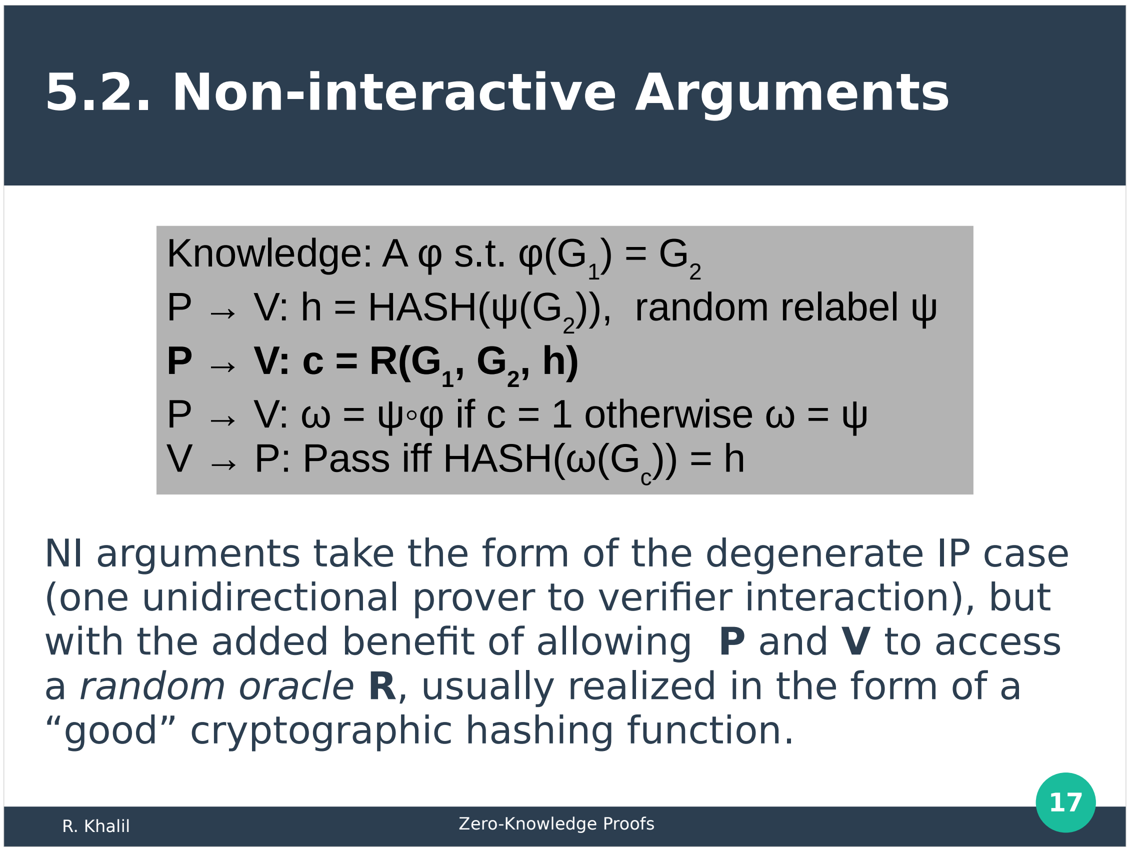
V $\rightarrow$ P: $c \in \{1,2\}$

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

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

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.

Recall that in our graph isomorphism proving system, **P** simply provides the isomorphic copy **H** to **V** in a perfectly binding manner (in plain). We can instead replace this plaintext message with only a hash of **H**, and require **V** to check whether hashing the graph (supposed **H**) resulting from applying the isomorphism supplied by **P** in response to the challenge corresponds with the hash supplied by **P**.



5.2. Non-interactive Arguments

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

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

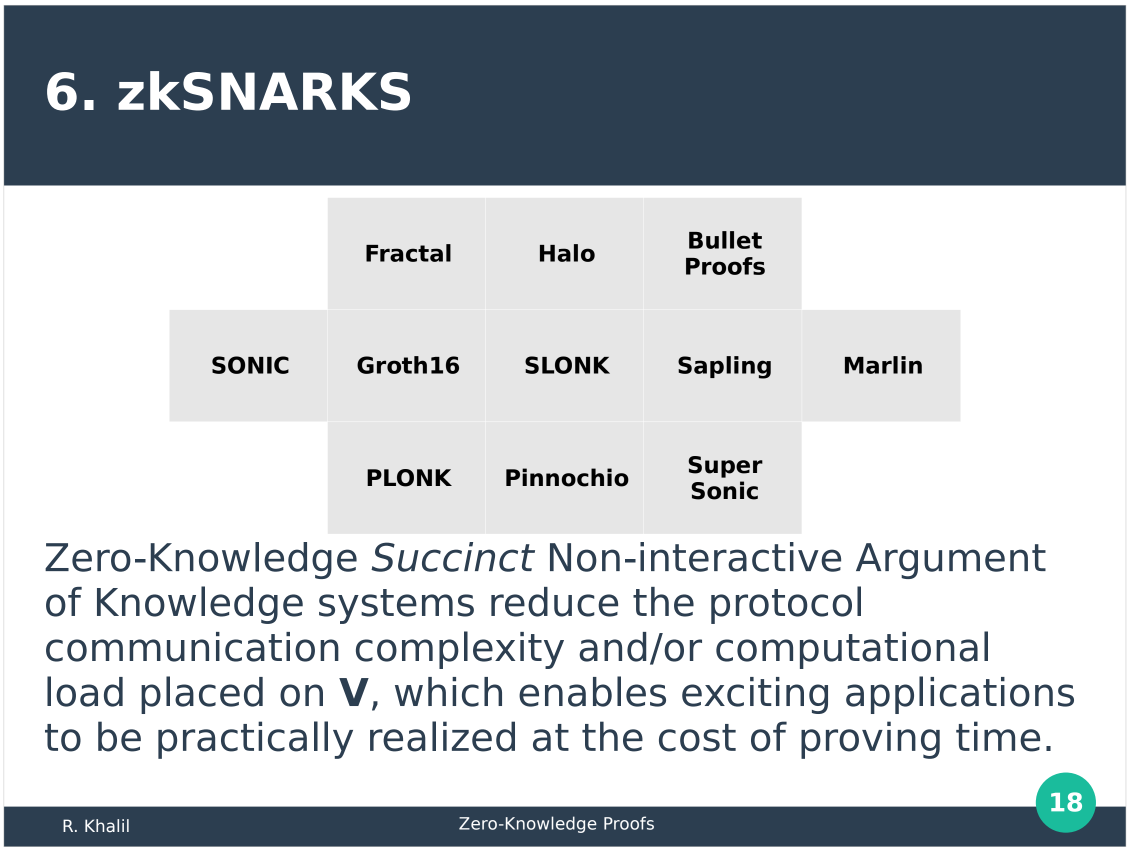
$P \rightarrow V$: $c = R(G_1, G_2, h)$

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

$V \rightarrow P$: 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.

R allows **P** to independently simulate the equivalent of a "random" challenge in response to **P**'s commitment, and allows **V** to validate whether the challenge responded to by **P** is "random" (derived from **R** using the statement and **P**'s commitment).m



6. zkSNARKS

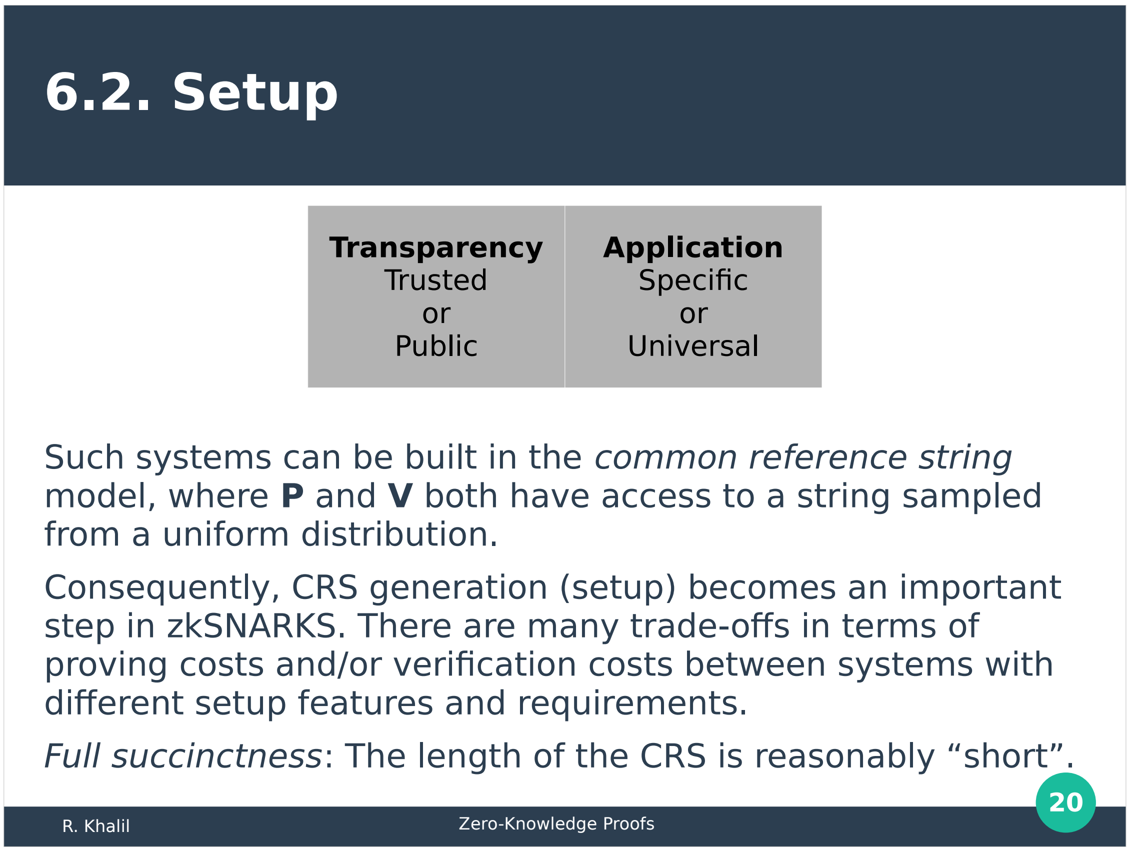
	Fractal	Halo	Bullet Proofs	
SONIC	Groth16	SLONK	Sapling	Marlin
	PLONK	Pinnochio	Super Sonic	

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.

However, the savings in communication and/or verification costs do not come for free, but are paid for by the prover, who has to perform potentially more complex computations to yield a valid argument.

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".

Some systems require that the procedure for generating this string be performed by a trusted third party (or as an MPC), while others do not.

Some systems can only generate a CRS that may handle one relation, while others generate a CRS that is only bound by the number of verifiable "steps".

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.

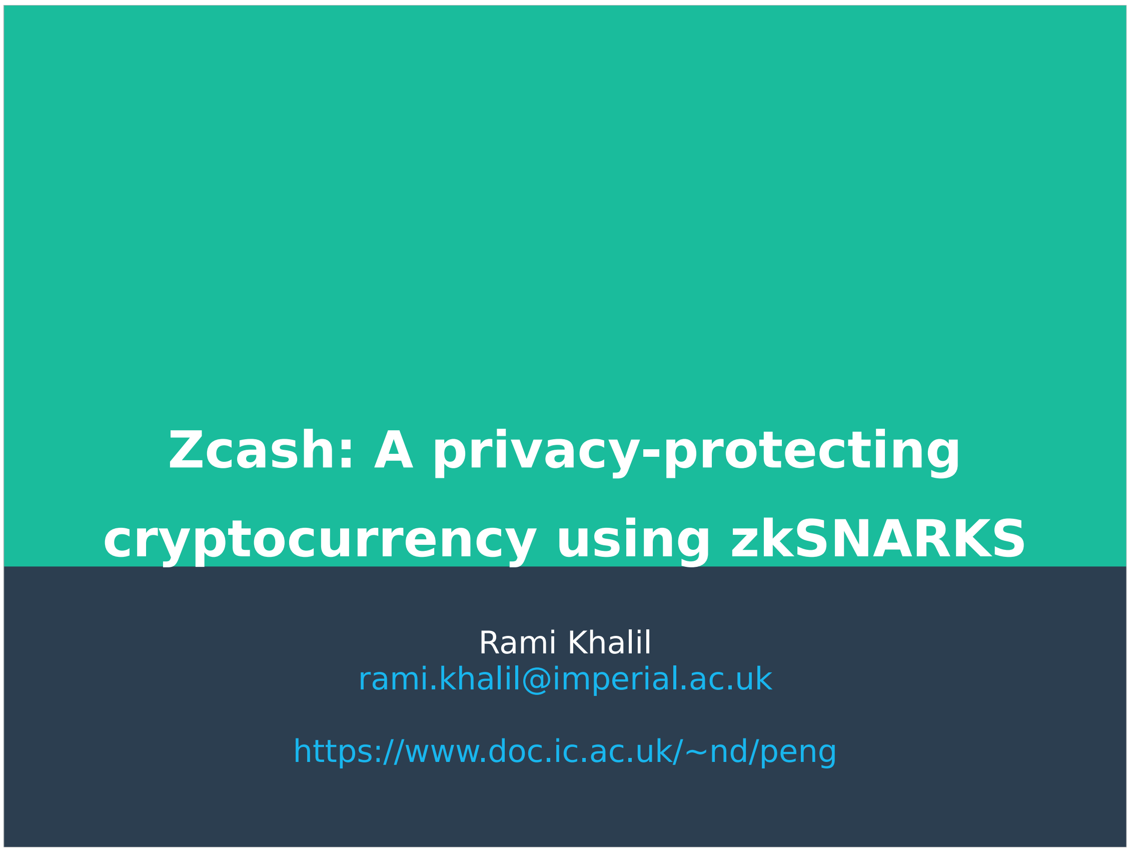
Groth, Jens. "On the size of pairing-based non-interactive arguments." Annual international conference on the theory and applications of cryptographic techniques. Springer, Berlin, Heidelberg, 2016.

End-of-Text

Further Reading:

Smart, Nigel P. Cryptography made simple. Springer, 2016. *(Chapter 21 p. 425-438)*

Goldreich, Oded. Foundations of cryptography: volume 1, basic tools. Cambridge university press, 2007. *(Chapter 4 p. 184-330)*

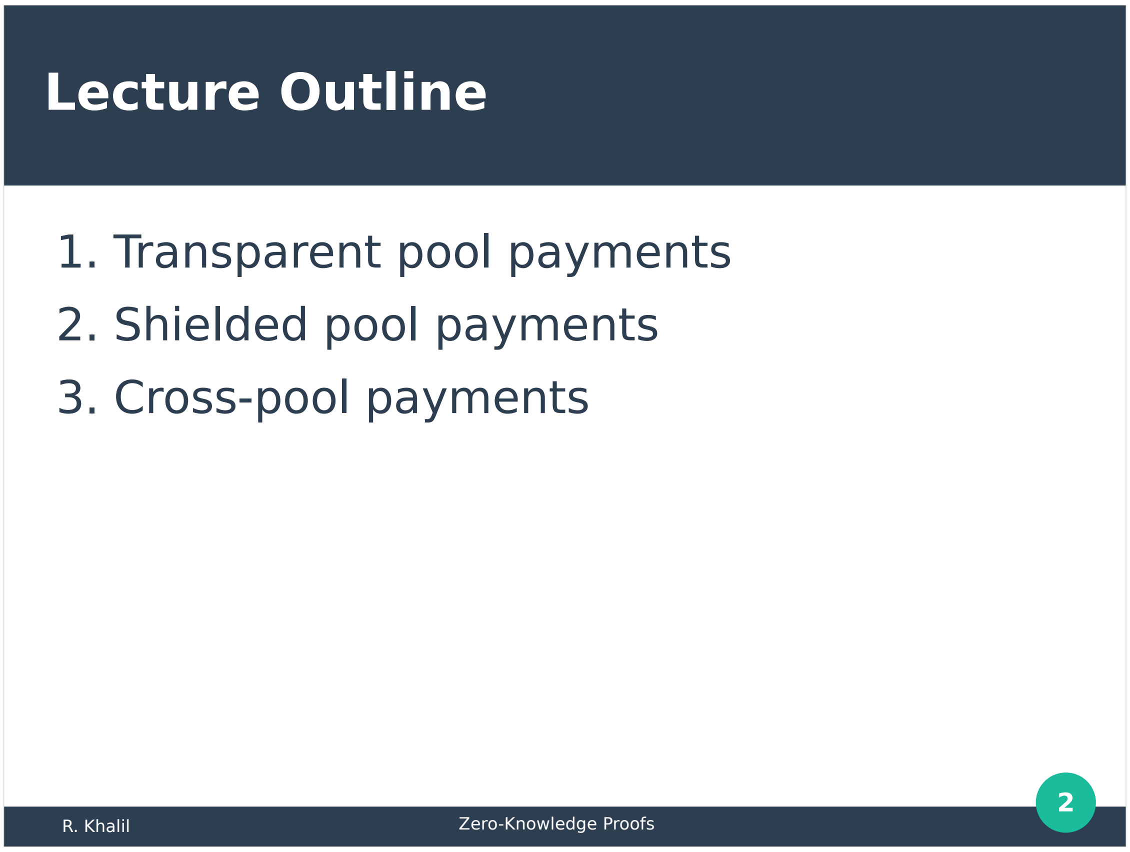


Zcash: A privacy-protecting cryptocurrency using zkSNARKS

Rami Khalil
rami.khalil@imperial.ac.uk

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

This is an introduction to Bitcoin/Crypto/Zcash and transaction linkability.

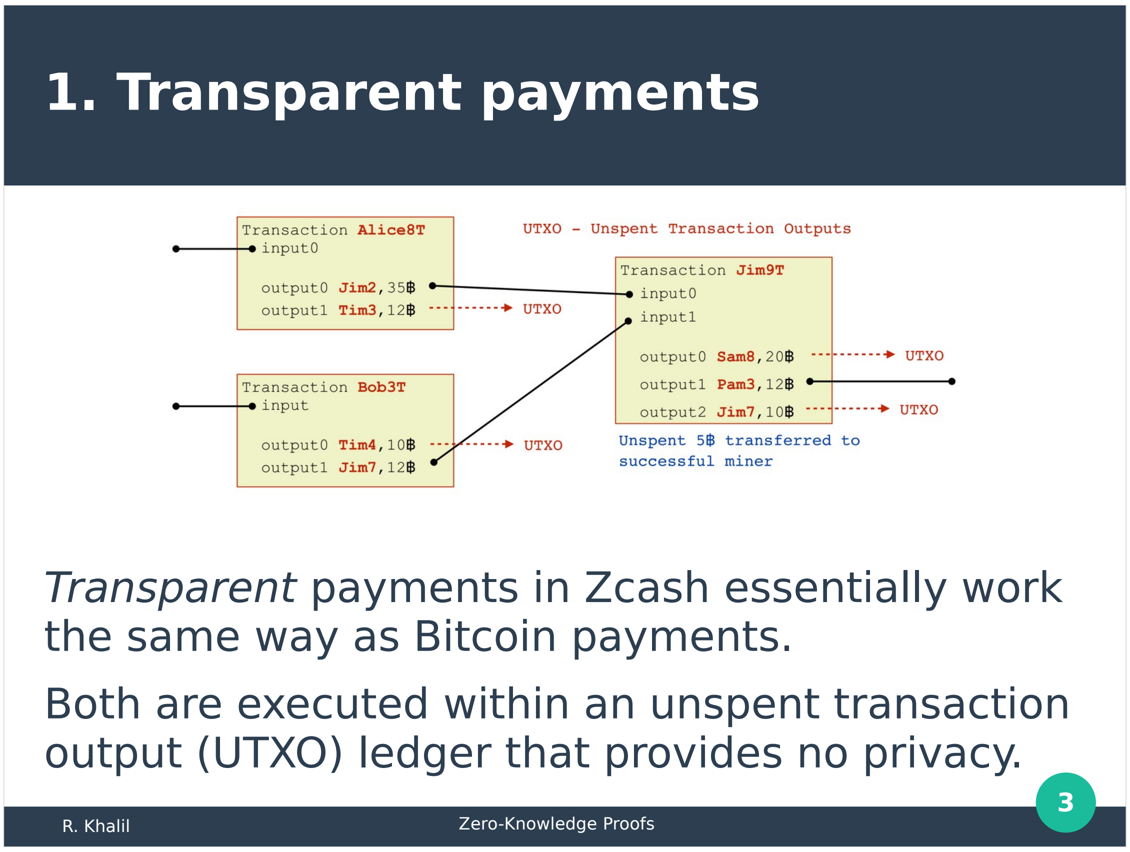

Lecture Outline

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

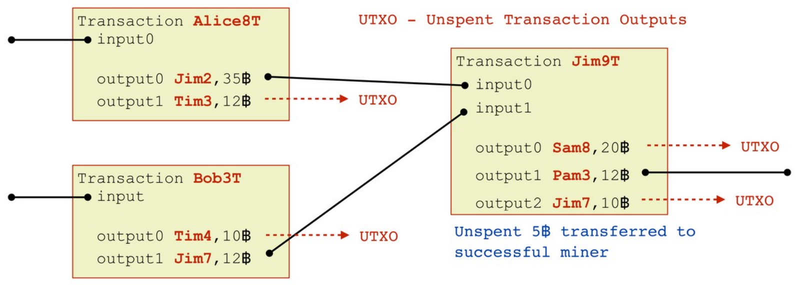
This lecture is only an introductory guide to the topics mentioned within, contains many informal simplifications, and is not comprehensive.

Its purpose is to give you basic intuitions that allow you to unpack the plethora of concepts used in the Zcash protocol.

The level of intuition conveyed should be sufficient for you to understand how Zcash provides privacy guarantees, and also gain a contextual understanding that allows you to further explore the covered concepts on your own without being overwhelmed (too much) by the depth and breadth of existing work.



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.

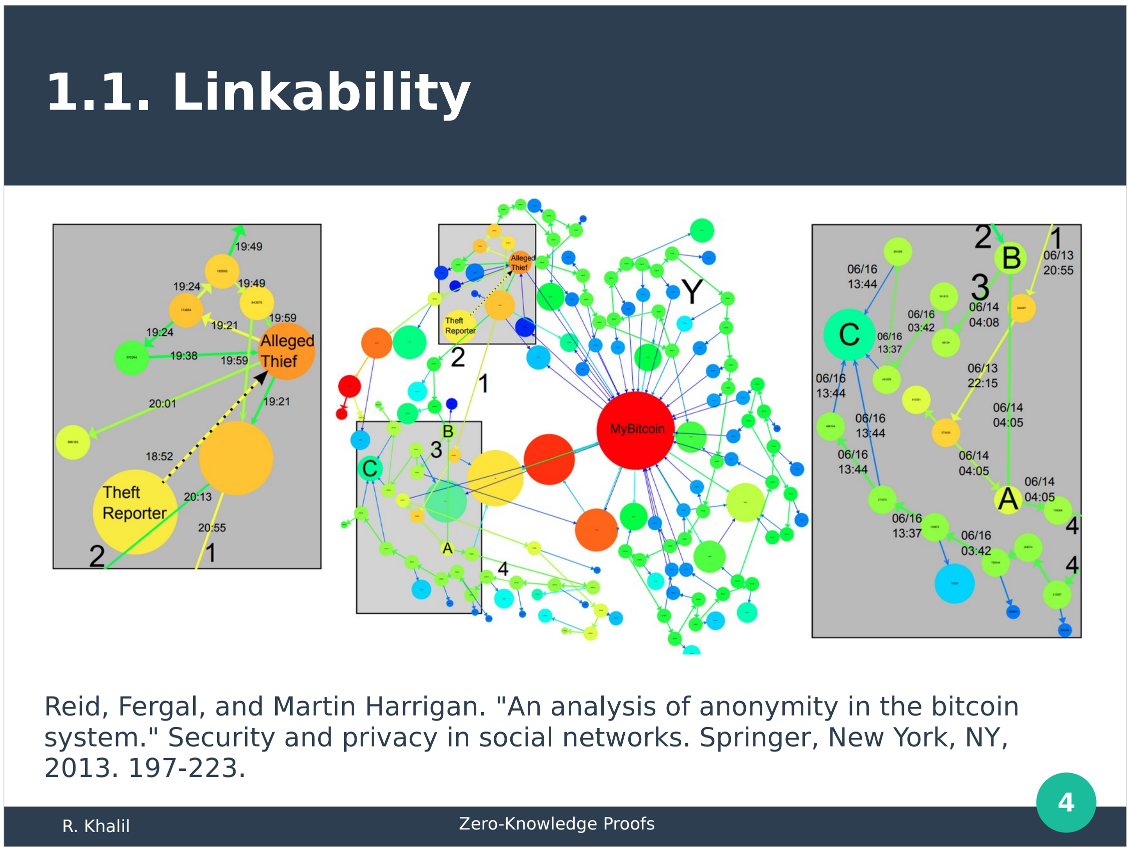
R. Khalil Zero-Knowledge Proofs 3

A UTXO ledger is a record of transactions, each of which spends the total value contained in all inputs it references to create new outputs with specific values.

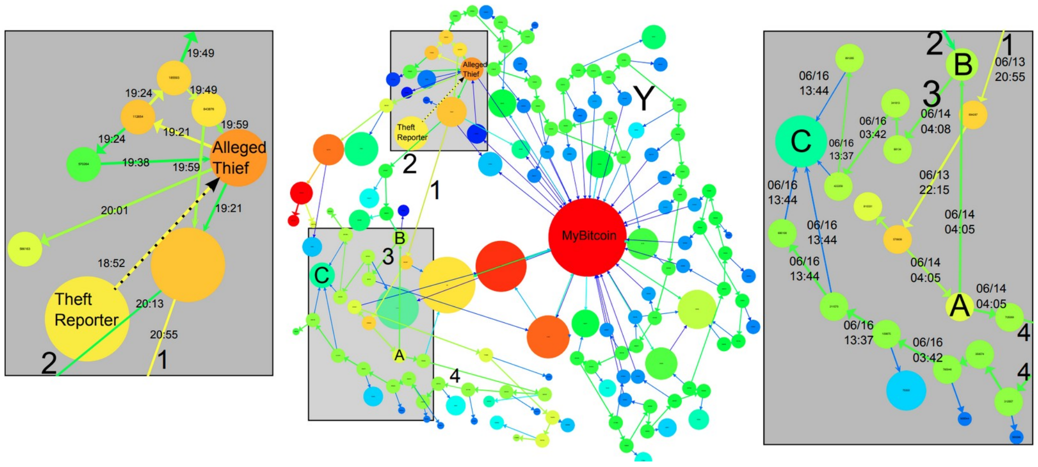
A transaction output specifies under what conditions the coins allocated to it can be spent (when referenced as an input), i.e. who can spend it.

An output specifies under what conditions the coins allocated to it can be spent (when referenced as an input).

If the sum of coins in the outputs of a transaction is less than the sum of coins in its inputs, the remainder is considered to be the transaction fee.



1.1. Linkability

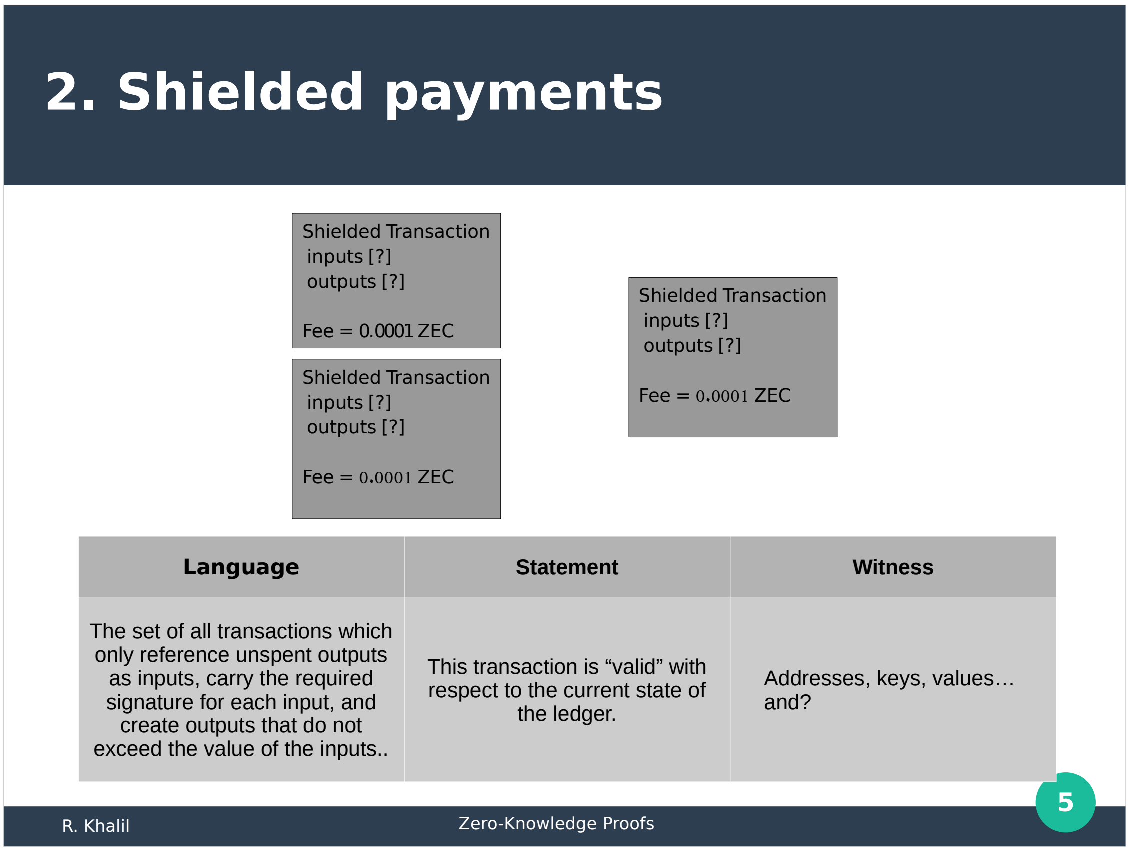


Reid, Fergal, and Martin Harrigan. "An analysis of anonymity in the bitcoin system." Security and privacy in social networks. Springer, New York, NY, 2013. 197-223.

Inputs, outputs, and amounts are specified in plaintext. This enables the construction of a directed acyclic graph that represents the flow of currency between addresses.

While in the early days, several media outlets claimed that “Bitcoin” is an “anonymous” payment system, several works later demonstrated that a lot of information about the users behind the pseudo-anonymous payments can be inferred.

This inference is mainly enabled by the linkability of transparent transactions, which permits analysts to piece together how senders and recipients are connected in real life given only partial information about address owners in real-life.



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?

R. Khalil Zero-Knowledge Proofs 5

Shielded transaction in Zcash leverage zkSNARKs to allow the validation of transactions without revealing addresses or values.

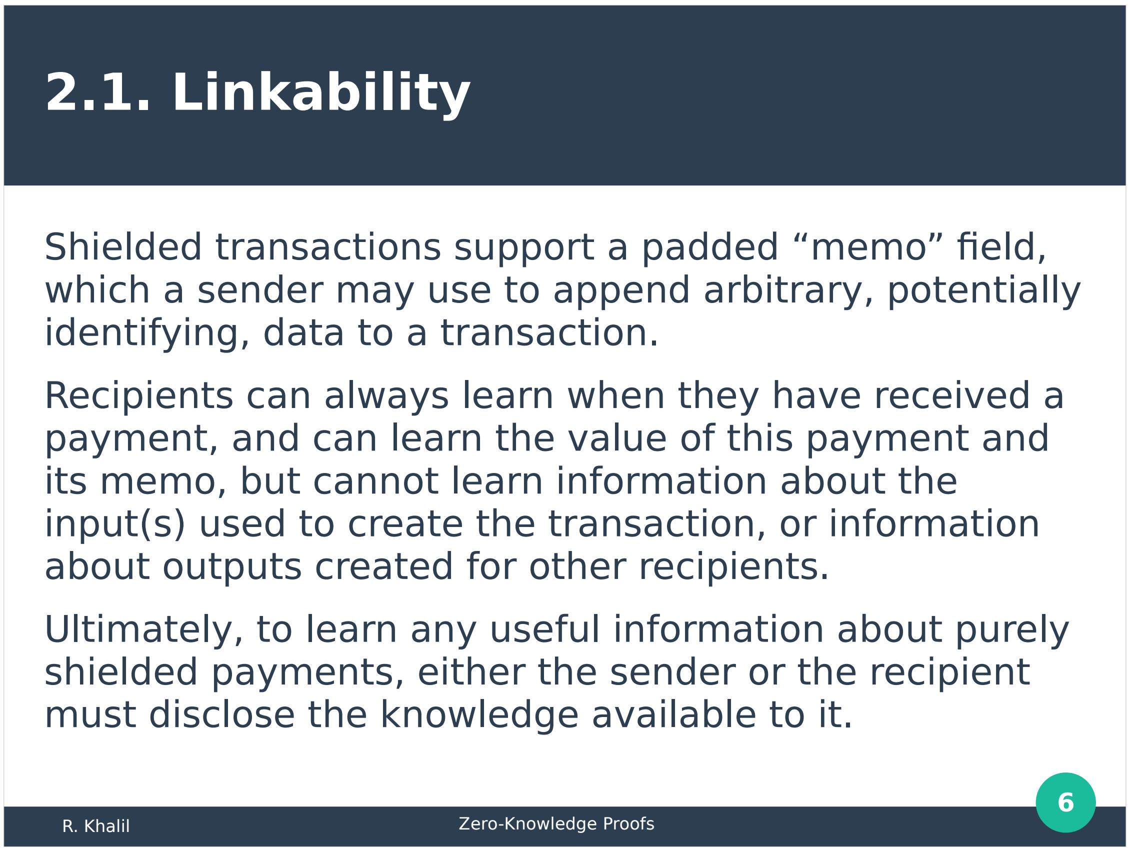
Each transaction bears a zkSNARK (in lieu of a signature) that argues in zero-knowledge that:

- All inputs are unspent.
- The prover controls all inputs (knows their private key).
- The total value of created outputs does not exceed that of inputs.

What must the statement reveal?

What must the witness hide?

V must receive enough information about the transaction it is currently validating to update the ledger such that V is able validate future transactions.



2.1. Linkability

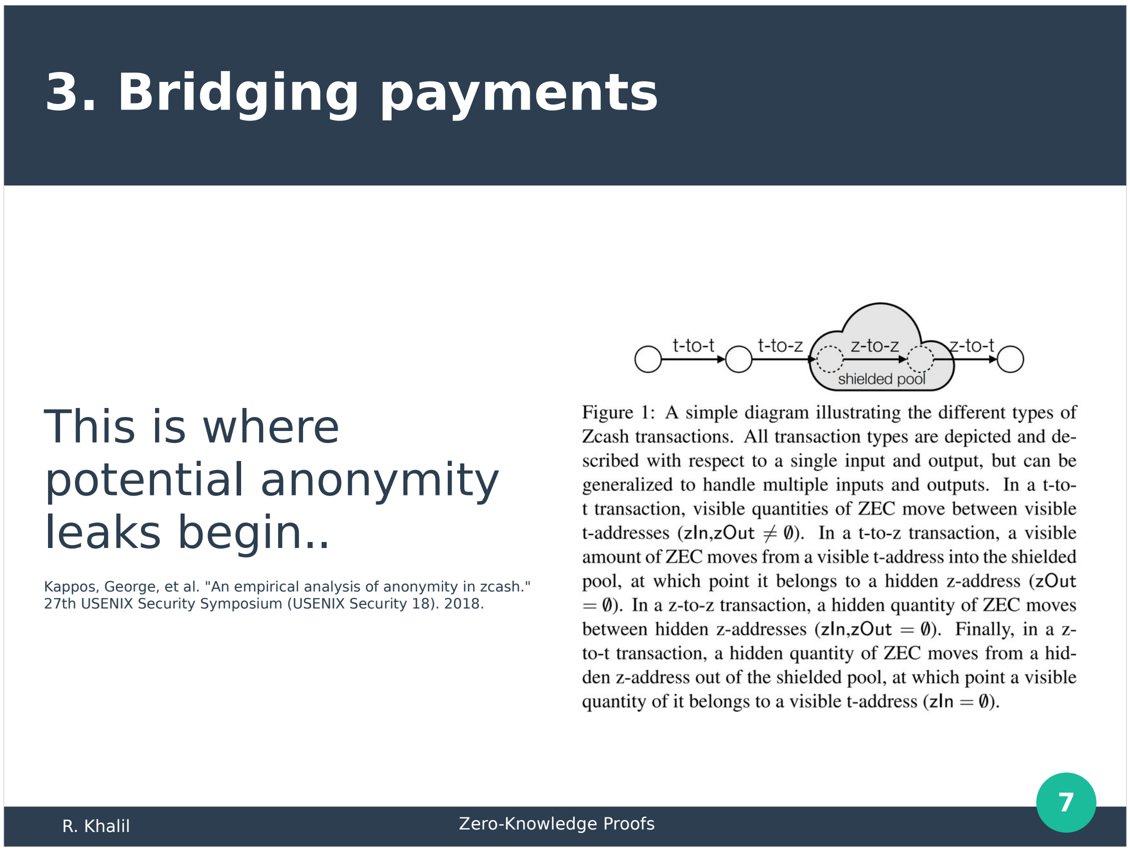
Shielded transactions support a padded “memo” field, which a sender may use to append arbitrary, potentially identifying, data to a transaction.

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.

R. Khalil Zero-Knowledge Proofs 6

Senders and recipients can voluntarily disclose some information about a transaction to third parties using “viewing keys”.



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.

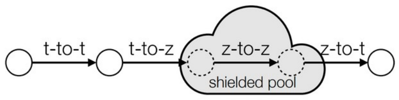
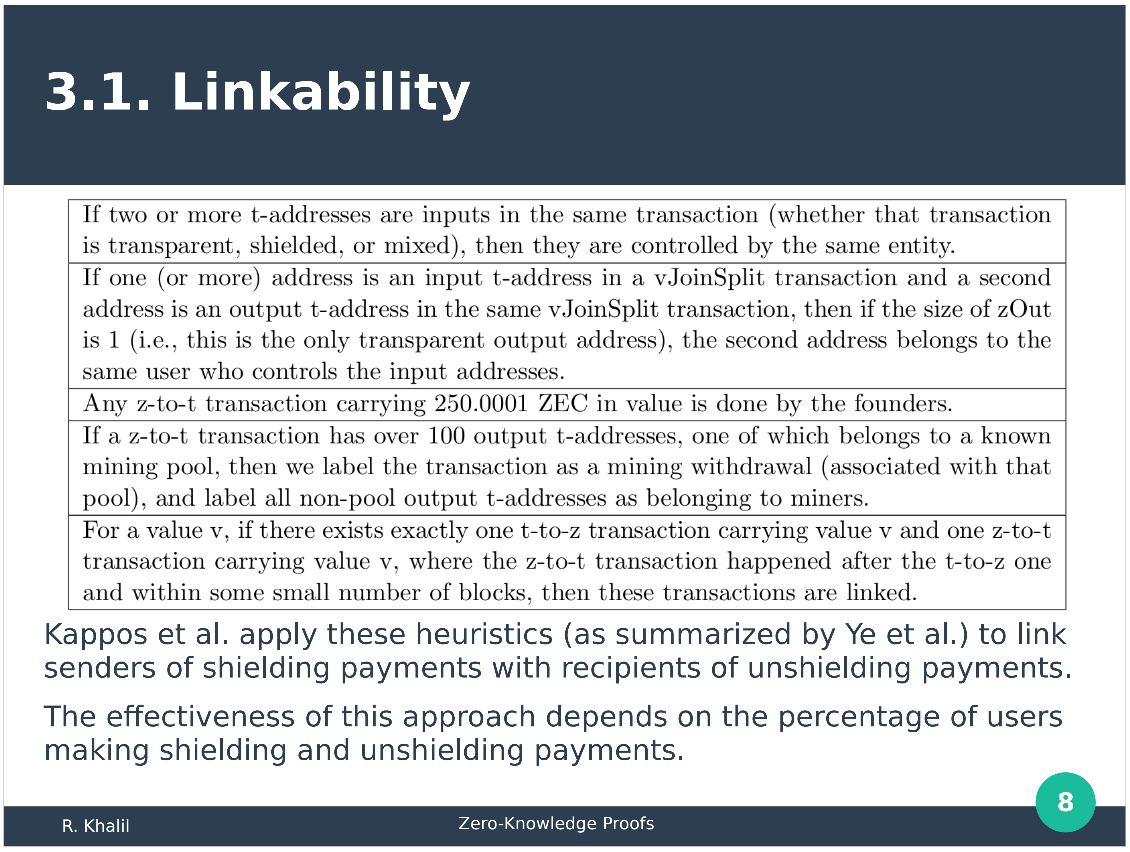


Figure 1: A simple diagram illustrating the different types of Zcash transactions. All transaction types are depicted and described with respect to a single input and output, but can be generalized to handle multiple inputs and outputs. In a t-to-t transaction, visible quantities of ZEC move between visible t-addresses (zIn,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 (zOut $= \emptyset$). In a z-to-z transaction, a hidden quantity of ZEC moves between hidden z-addresses (zIn,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 (zIn $= \emptyset$).

Shielding transactions (t-to-z) transfer coins from the transparent pool to the shielded pool by spending transparent inputs and creating shielded outputs.

Unshielding transactions (z-to-t) do the opposite.

Evidently, one can attempt to connect shielding transactions with unshielding transactions, but while losing the information related to any purely shielded payments that happen in between.



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.

Kappos et al. perform a very interesting case study in their paper on analysing the activity of a hacker group.

End-of-Text

Further Reading:

Hopwood, Daira, et al. "Zcash protocol specification." GitHub: San Francisco, CA, USA (2016).
https://github.com/zcash/zips/blob/master/protocol/protocol.pdf