Workshop on
Zero-Knowledge, Succinct Proofs and
Symmetric Cryptography

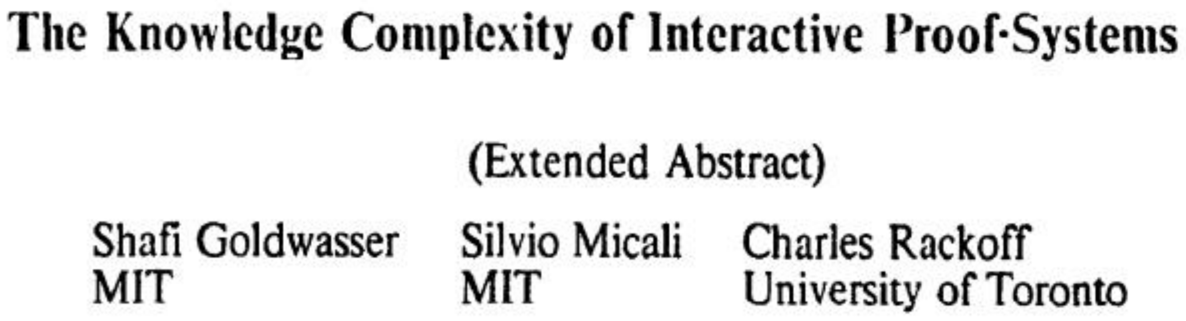
A Few Stops in the Zero-Knowledge Journey

Ivan Visconti
Sapienza University of Rome

visconti@diag.uniroma1.it

Vienna, February 9, 2026

GoldwMicRack STOC 85



The Knowledge Complexity of Interactive Proof-Systems

(Extended Abstract)

Shafi Goldwasser
MIT

Silvio Micali
MIT

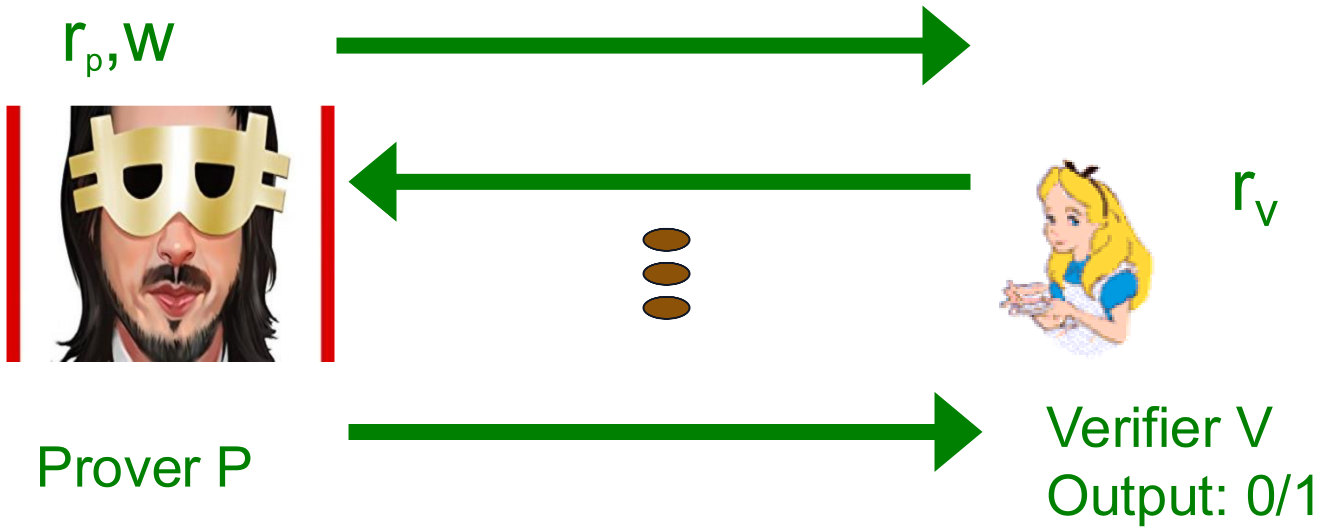
Charles Rackoff
University of Toronto

Fundamental concepts on:

Interactive Proof Systems, Knowledge Complexity, Simulation Paradigm, Zero Knowledge

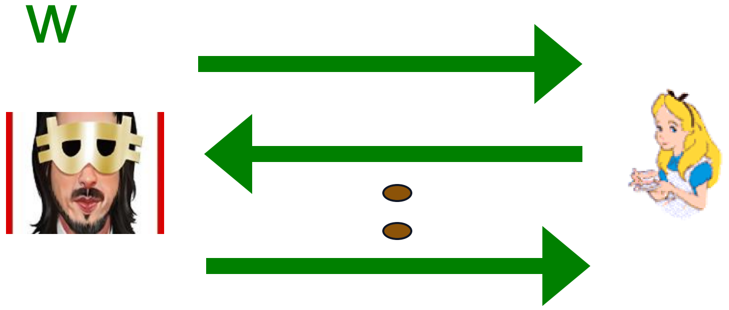
Interactive Proof System: typical setting

claim: x is in L



Interactive Proof/Argument System: basic informal requirements

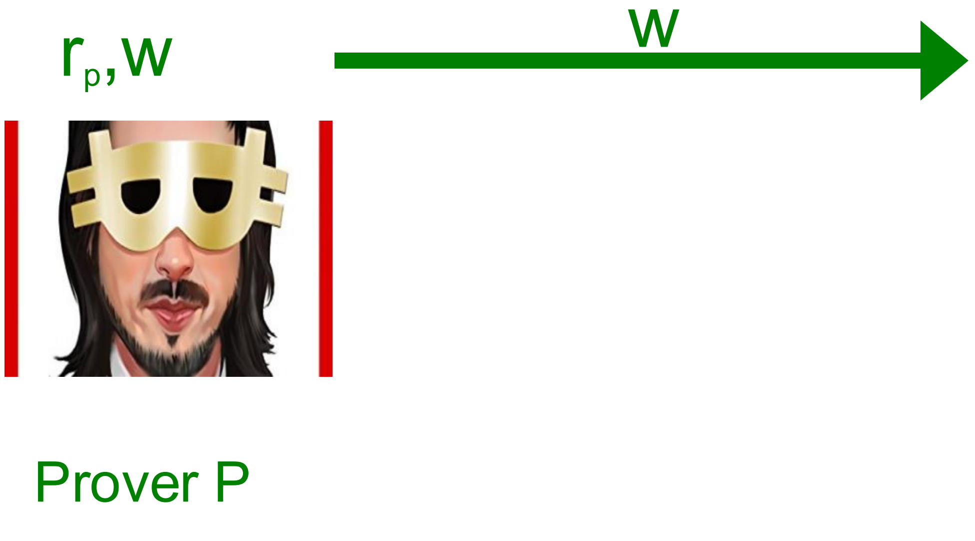
claim: x is in L



Completeness: if claim is true then V outputs 1
Unconditional Soundness (Proof): if claim is false then V outputs 0 (except with neg. prob.)
Computational Soundness (Argument): when P* is polytime, if claim is false then V outputs 0 (except with neg. prob.)

IPS: trivial protocol

claim: x is in L



IPS: beyond the trivial protocol

claim: x is in L

r_p, w $\xrightarrow{w}$ r_V

Three valid reasons justify complicated protocols:

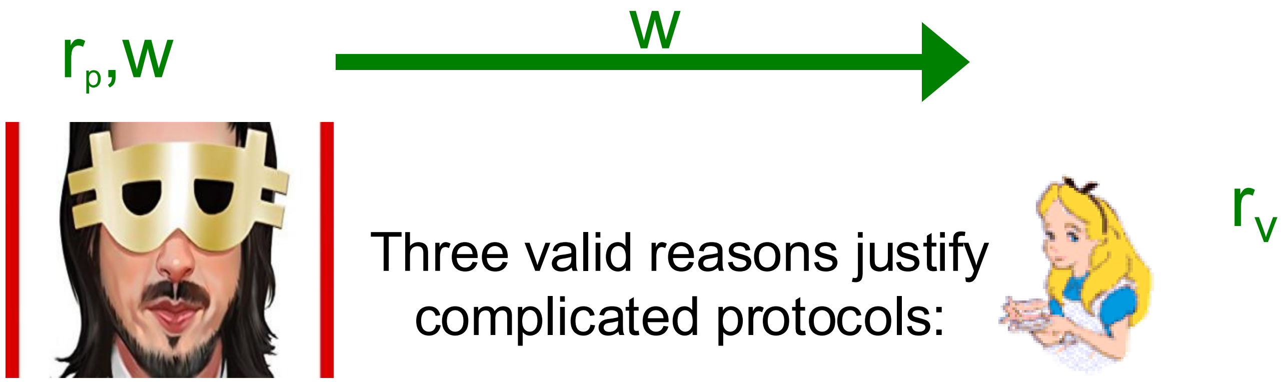
1) Protecting the confidentiality of **w**
2) Reducing the communication
3) Reducing the computations of **V**

Prover P

Verifier V
Output: 0/1

IPS: beyond the trivial protocol

claim: x is in L



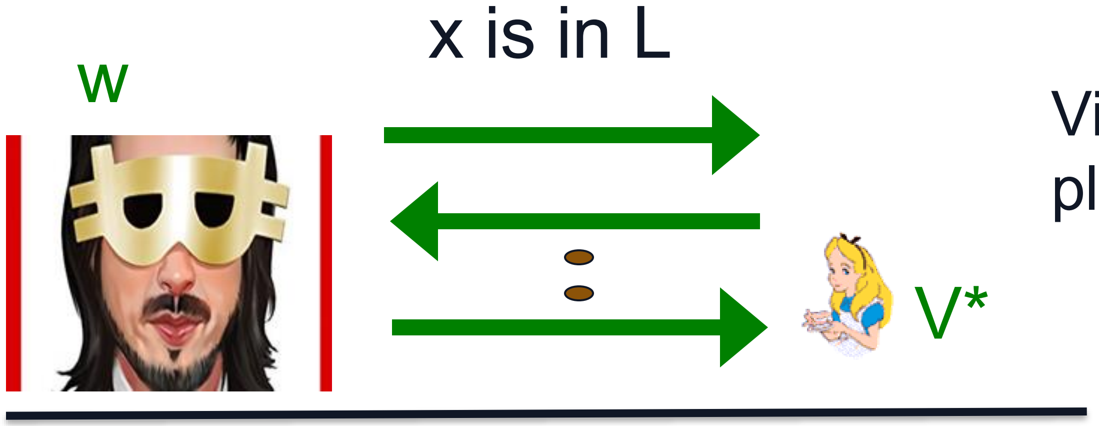
Three valid reasons justify complicated protocols:

1) Protecting the confidentiality of **w**
2) Reducing the communication
3) Reducing the computations of **V**

Prover P

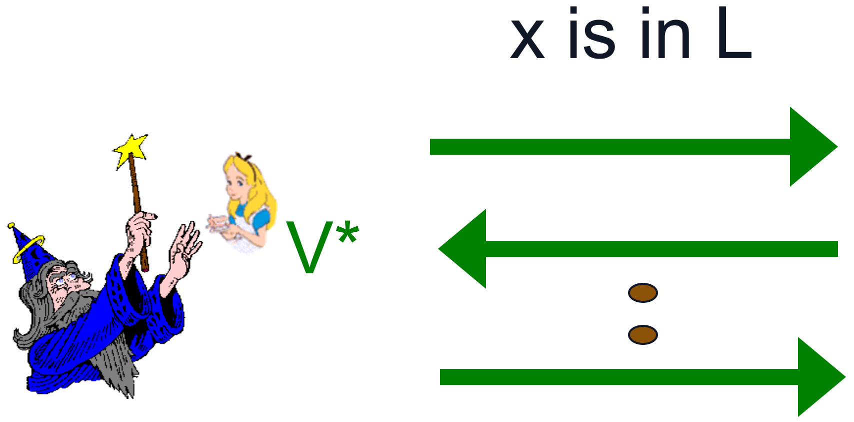
Verifier V
Output: 0/1

Black-Box Zero-Knowledge IPS



View of any polytime V* playing with P

must be (comp./stat./perf.) indistinguishable from



Output of an expected PPT Simulator on input the claim and **black-box** access to V*

(Non-)Black-Box: subtleties

BBZK: **one** Simulator works for **any** V*
NBBZK: for **any** V* there is (at least) **one** working simulator

Initial positive results were confined to BB simulation only.

Let's make a first stop on BB simulation.

(Non-)Black-Box: subtleties

BBZK: **one** Simulator works for **any** V*
NBBZK: for **any** V* there is (at least) **one** working simulator

Initial positive results were confined to BB simulation only.

Let's make a first stop on BB simulation.

There are **two** major and very different **approaches** to design a **simulator**:

(Non-)Black-Box: subtleties

BBZK: **one** Simulator works for **any** V*
NBBZK: for **any** V* there is (at least) **one** working simulator

Initial positive results were confined to BB simulation only.

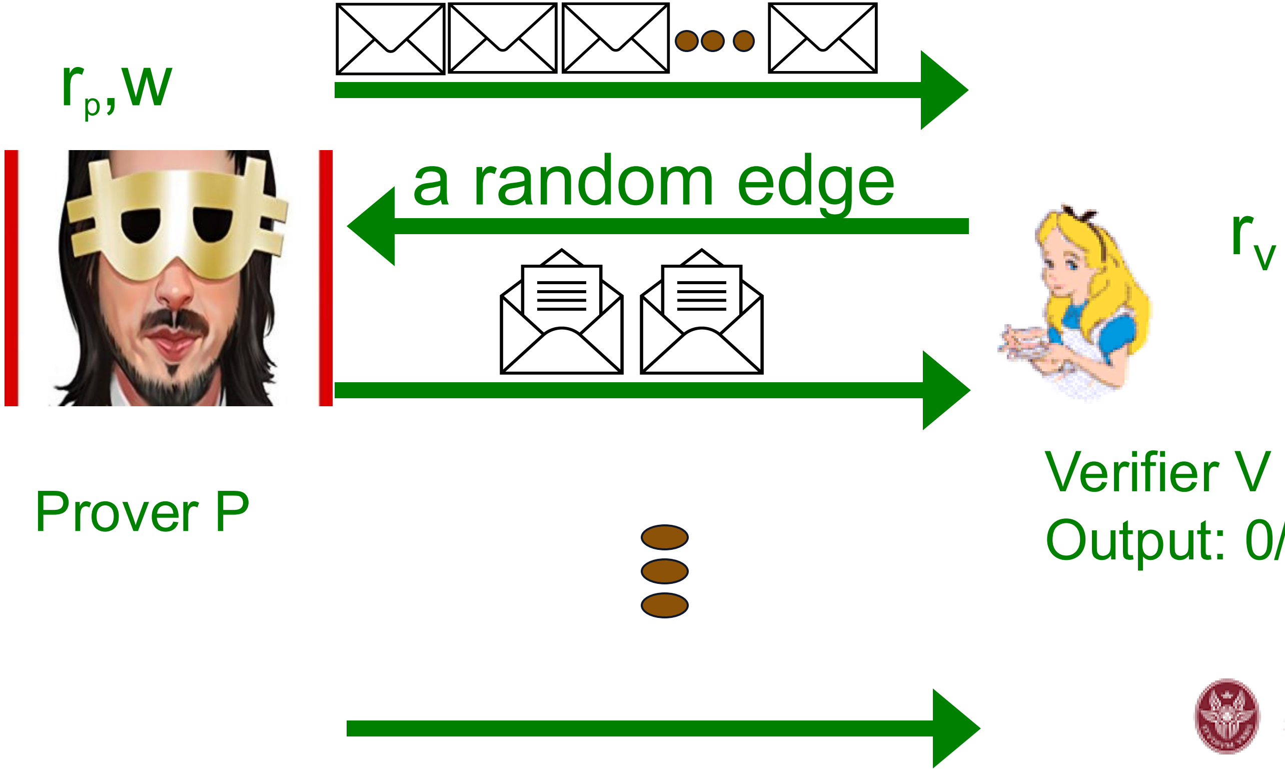
Let's make a first stop on BB simulation.

There are **two** major and very different **approaches** to design a **simulator**:

1) **guessing** (i.e., try and re-try to extend the current transcript until you predict correctly)
2) **extracting** (i.e., rewinds are performed to find secrets associated to the current transcript, the **main thread**)

Black-Box ZK: Approach 1 (guessing)

G is in 3COL [GMW, FOCS 86]

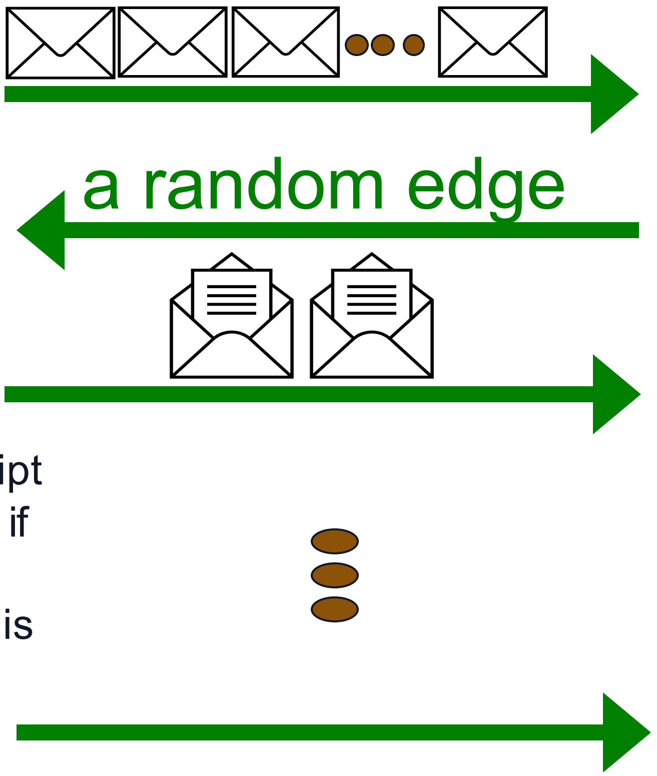


Black-Box ZK: Approach 1 (guessing)

G is in 3COL [GMW, FOCS 86]

r_p

Simulator



r_V

Adversarial Verifier V*

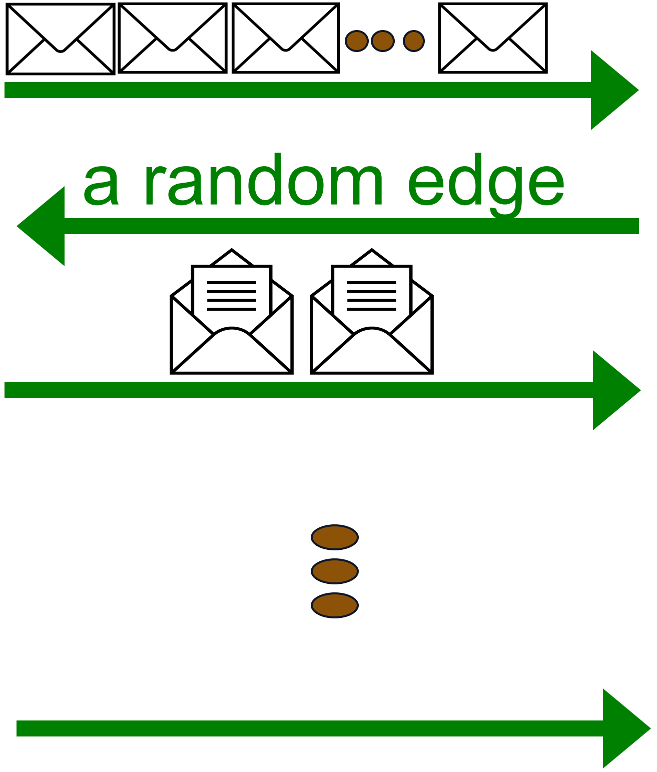
S, crossing fingers, starts an attempt to expand the transcript trying to guess the challenge; if it is unsuccessful, it tries again. Once a successful run is reached, the output is expanded with those messages.

Black-Box ZK: Approach 1 (guessing)

G is in 3COL [GMW, FOCS 86]

r_p

Simulator



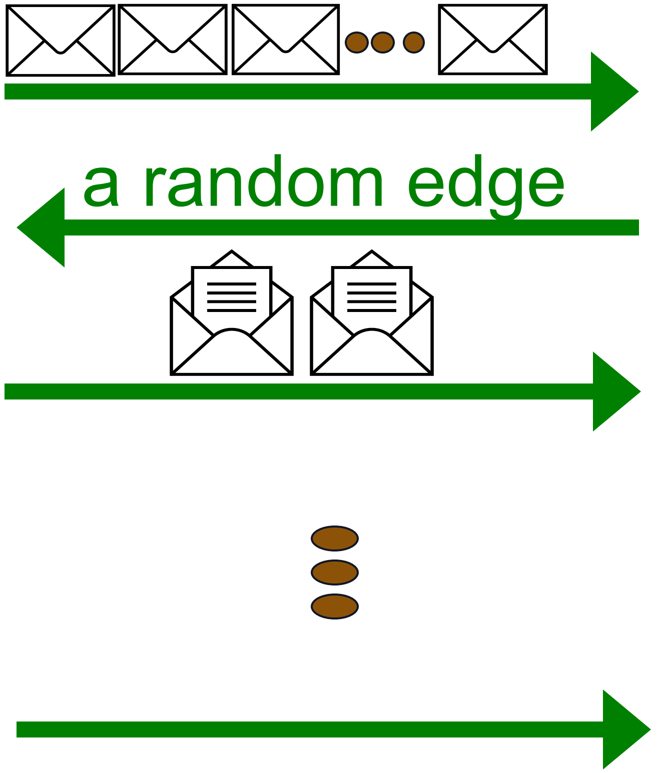
r_V

Adversarial Verifier V*

There is a caveat. V* can abort. If the first attempt of **S** is not aborting and unsuccessful, then the **S** keeps repeating (whatever it takes) until a successful non-aborting run is obtained.

Black-Box ZK: Approach 1 (guessing)

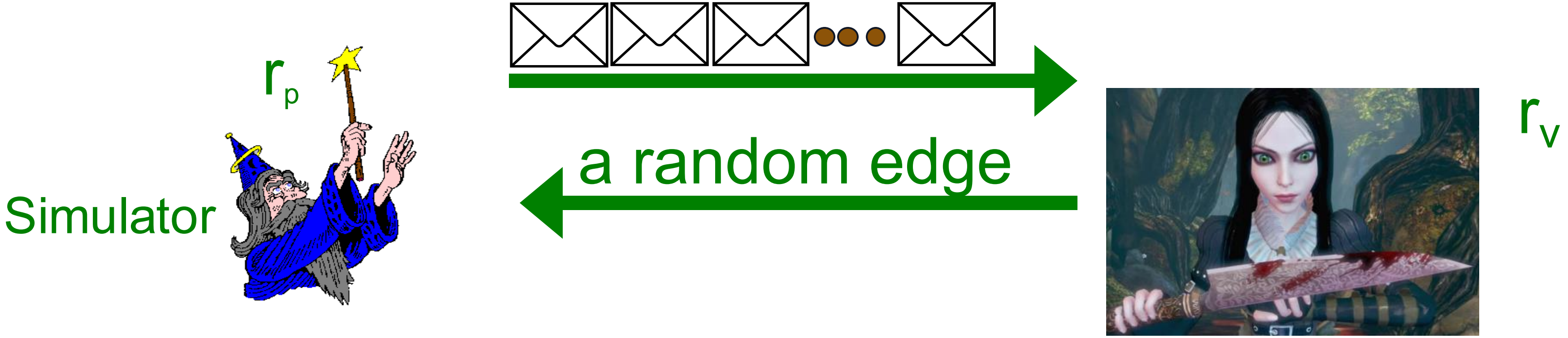
G is in 3COL [GMW, FOCS 86]



Repeating until there is again an answer from V* corresponds to waiting for the same configuration of the original run, except that, eventually, the simulator will have guessed correctly.
It works, but...

Black-Box ZK: Approach 1 (guessing)

G is in 3COL [GMW, FOCS 86]



In a scenario with many parallel (or concurrent) sessions (warning: I'm not talking about soundness amplification) repeating until the same configuration appears is hopeless.

As such, while Approach 1 is effective for public-coin ZK in the sequential setting, in the parallel/concurrent setting the approach is very problematic.

Several works using a guessing-like approach overlooked this issue as shown in [SV, EUROCRYPT 12].

SAPIENZA
UNIVERSITÀ DI ROMA

Black-Box vs Non-Black-Box Zero Knowledge

BBZK: **one** Simulator works for **any** V*
NBBZK: for **any** V* there is **one** working simulator

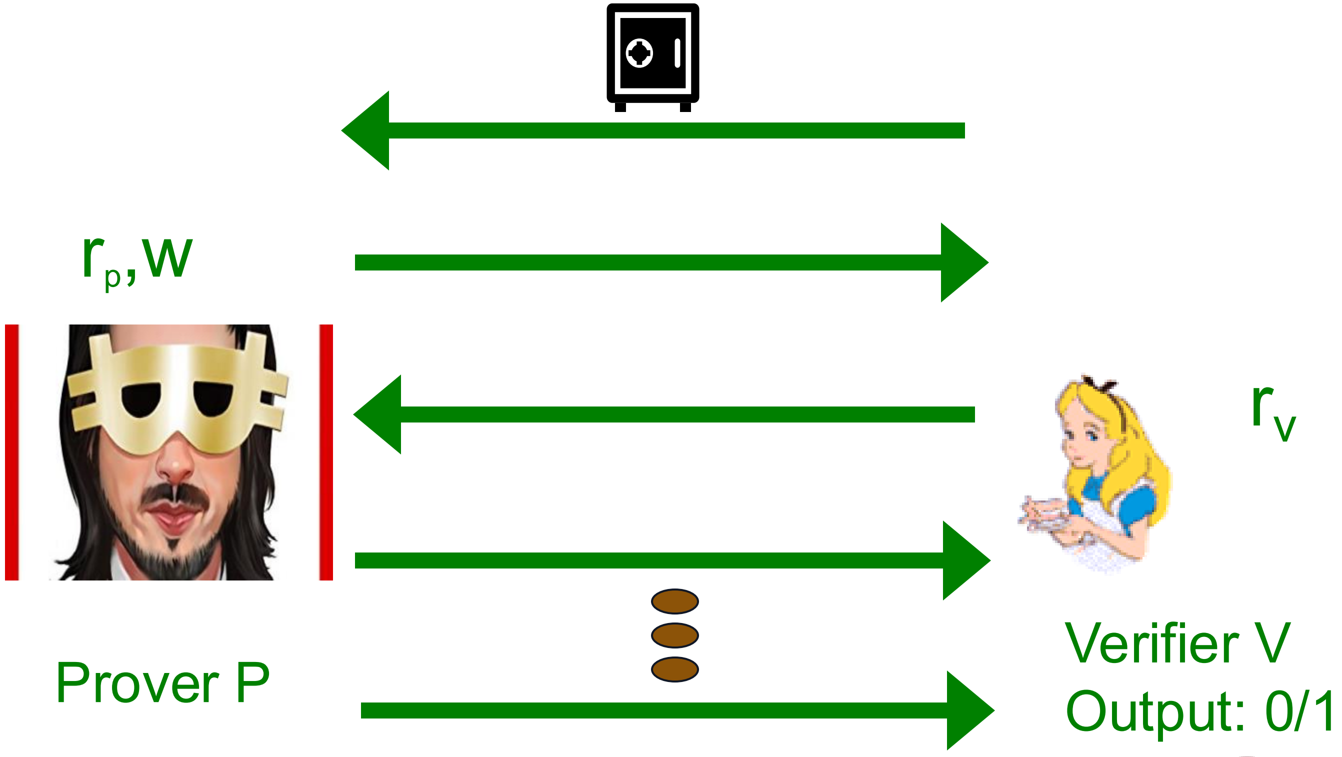
Initial positive results were confined to BB simulation only.

Let's make a first stop on BB simulation.

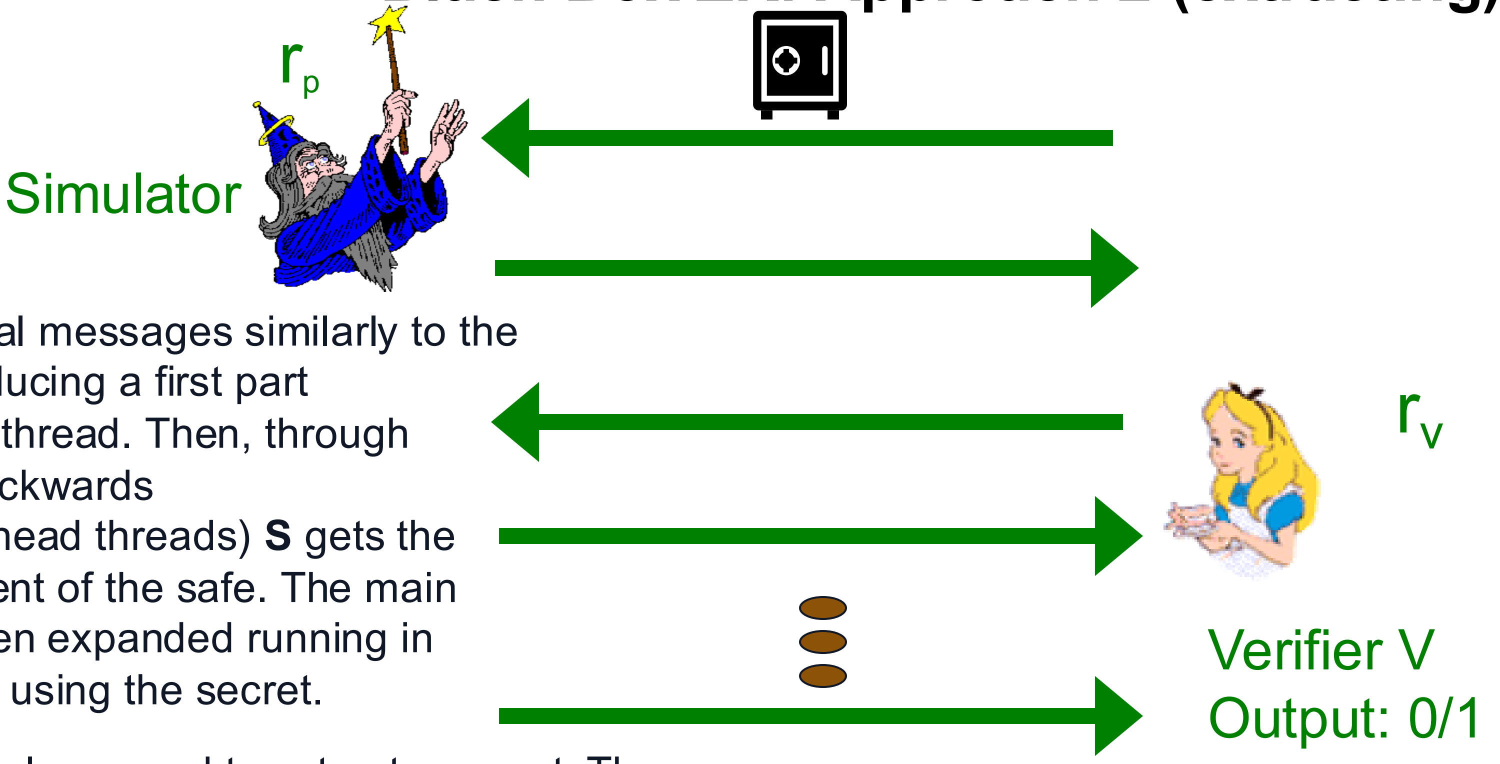
It allows one to design a simulator with **two** major and very different approaches:

1) **guessing** (i.e., try and re-try to extend the current transcript until you predict correctly)
2) **extracting** (i.e., rewinds are performed to find secrets associated to the current transcript, the **main thread**)

Black-Box ZK: Approach 2 (extracting)



Black-Box ZK: Approach 2 (extracting)



S plays initial messages similarly to the prover, producing a first part of the main thread. Then, through rewinds (backwards or in look-ahead threads) **S** gets the secret content of the safe. The main thread is then expanded running in straight-line using the secret.

Rewinds are leveraged to extract a secret. The configuration in those threads changes but it does not matter. Extracting is effective in the concurrent scenario, but do not end up with public coins.

Black-Box vs Non-Black-Box

BBZK: **one** Simulator works for **any** V*
NBBZK: for **any** V* there is (at least) **one** working simulator

Initial positive results were confined to BB simulation only.

Let's make a first stop on BB simulation.

It allows one to design a simulator with **two** major and very different approaches:

1) **guessing** (i.e., try and re-try to extend the current transcript until you predict correctly)
2) **extracting** (i.e., rewinds are performed to find secrets associated to the current transcript, the **main thread**)

NBB ZK. Let's make a second stop.

Non-Black-Box Assumptions and Simulation

NBBZK: for **any** V* there is (at least) **one** working simulator

This implies that, in order to prove that a protocol is not ZK, one should show that all possible simulators for all possible V* fail.

Non-Black-Box Assumptions and Simulation

NBBZK: for **any** V* there is (at least) **one** working simulator

This implies that, in order to prove that a protocol is not ZK, one should show that all possible simulators for all possible V* fail.
Unless the protocol is trivially insecure, proving that it is not ZK is tough.

Examples of very naïve insecure protocols that are not ZK:

- The prover openly leaks the witness. A simulator would allow to efficiently recover a witness from any instance of the language. Therefore, it is ZK only if deciding the language is trivial.

- The prover can internally run the simulator to obtain the messages for the verifier. In case of a non-trivial language this can be abused by P* to violate soundness.

Non-Black-Box Assumptions and Simulation

NBBZK: for **any** V* there is (at least) **one** working simulator

This implies that, in order to prove that a protocol is not ZK, one should show that all possible simulators for all possible V* fail

Unless the protocol is trivially insecure, proving that is not ZK is tough.

A famous class of (non-naïve) protocols that are **believed** not to be ZK consists of Sigma protocols* [DNRS, FOCS 99]:

*natural ones

Non-Black-Box Assumptions and Simulation

NBBZK: for **any** V* there is (at least) **one** working simulator

This implies that, in order to prove that a protocol is not ZK, one should show that all possible simulators for all possible V* fail

Unless the protocol is trivially insecure, proving that is not ZK is tough.

A famous class of (non-naïve) protocols that are **believed** not to be ZK consists of Sigma protocols* [DNRS, FOCS 99]:

- The Fiat-Shamir transform gives a NIZK in the PRO (or NIWI** in the NPRO) model. This is used not only for NIZK but also for signature schemes (e.g., Schnorr's scheme).
-

*natural ones

Non-Black-Box Assumptions and Simulation

NBBZK: for **any** V* there is (at least) **one** working simulator

This implies that, in order to prove that a protocol is not ZK, one
should show that all possible simulators for all possible V* fail
Unless the protocol is trivially insecure, proving that is not ZK is tough.

A famous class of (non-naïve) protocols that are **believed** not to be ZK consists of Sigma protocols* [DNRS, FOCS 99]:

- The Fiat-Shamir transform gives a NIZK in the PRO (or NIWI** in the NPRO) model. This is used not only for NIZK but also for signature schemes (e.g., Schnorr's scheme).

- A simulator for the Sigma-protocol against V* that just runs the cryptographic hash function would allow the prover of the NIZK/NIWI (or the adversary of the signature scheme) to obtain an accepting proof for any statement (or a signature of any message).

*natural ones
**if the Sigma protocol is WI

Non-Black-Box Assumptions and Simulation

NBBZK: for **any** V* there is (at least) **one** working simulator

This implies that, in order to prove that a protocol is not ZK, one should show that all possible simulators for all possible V* fail

Unless the protocol is trivially insecure, proving that is not ZK is usually tough/hopeless.

OK, so one can heuristically claim that a (non-naïve) protocol is (NBB) ZK and the claim is going to stand.

But how do we prove that a specific protocol is ZK when BB simulation fails?

Non-Black-Box Assumptions and Simulation

NBBZK: for **any** V* there is (at least) **one** working simulator

This implies that, in order to prove that a protocol is not ZK, one should show that all possible simulators for all possible V* fail

Unless the protocol is trivially insecure, proving that is not ZK is usually tough/hopeless.

OK, so one can heuristically claim that a (non-naïve) protocol is (NBB) ZK and the claim is going to stand.

But how do we prove that a specific protocol is ZK when BB simulation fails?

Method 1: using a NBB assumption (i.e., hiding dust under the rug).

Non-Black-Box Assumptions and Simulation

NBBZK: for **any** V* there is (at least) **one** working simulator

This implies that, in order to prove that a protocol is not ZK, one should show that all possible simulators for all possible V* fail

Unless the protocol is trivially insecure, proving that is not ZK is usually tough/hopeless.

OK, so one can heuristically claim that a (non-naïve) protocol is (NBB) ZK and the claim is going to stand.

But how do we prove that a specific protocol is ZK when BB simulation fails?

Method 1: using a NBB assumption (i.e., hiding dust under the rug).

Build an NBB Simulator [HT, CRYPTO 98] via NBB extraction assumption (e.g., KEA [Dam, CRYPTO 91]).

Non-Black-Box Assumptions and Simulation

NBBZK: for **any** V* there is (at least) **one** working simulator

This implies that, in order to prove that a protocol is not ZK, one should show that all possible simulators for all possible V* fail

Unless the protocol is trivially insecure, proving that is not ZK is usually tough/hopeless.

OK, so one can heuristically claim that a (non-naïve) protocol is (NBB) ZK and the claim is going to stand.

But how do we prove that a specific protocol is ZK when BB simulation fails?

Method 1: using a NBB assumption (i.e., hiding dust under the rug).

Build an NBB Simulator [HT, CRYPTO 98] via NBB extraction assumption (e.g., KEA [Dam CRYPTO 91]).
An NBB extraction assumption is unlikely to be disproved (non-falsifiable)
(e.g., on input (g,A), computing (B,C) such that (g,A,B,C) is a DH tuple requires
Knowledge of DLOG of B.

Non-Black-Box Assumptions and Simulation

NBBZK: for **any** V* there is (at least) **one** working simulator

This implies that, in order to prove that a protocol is not ZK, one
should show that all possible simulators for all possible V* fail

Unless the protocol is trivially insecure, proving that is not ZK is usually tough/hopeless.

OK, so one can heuristically claim that a (non-naïve) protocol is (NBB) ZK and the claim is going to stand.

But how do we prove that a specific protocol is ZK when BB simulation fails?

Method 2: (a.k.a. Barak's approach) proving that either x is in L or that the prover knows the code of the verifier (i.e., P commits to a machine that outputs the same messages that V actually sends in the protocol).

Non-Black-Box Assumptions and Simulation

NBBZK: for **any** V* there is (at least) **one** working simulator

This implies that, in order to prove that a protocol is not ZK, one should show that all possible simulators for all possible V* fail

Unless the protocol is trivially insecure, proving that is not ZK is usually tough/hopeless.

OK, so one can heuristically claim that a (non-naïve) protocol is (NBB) ZK and the claim is going to stand.

But how do we prove that a specific protocol is ZK when BB simulation fails?

Method 2: (a.k.a. Barak's approach) proving that either x is in L or that the prover knows the code of the verifier (i.e., P commits to a machine that outputs the same messages that V actually sends in the protocol).
The approach requires as subprotocol a proof about the output of a committed V* with a communication that does not depend on V* (i.e., the witness of the simulator).

IPS: beyond the trivial protocol

claim: x is in L

r_p,w

$$\xrightarrow{w}$$

r_V

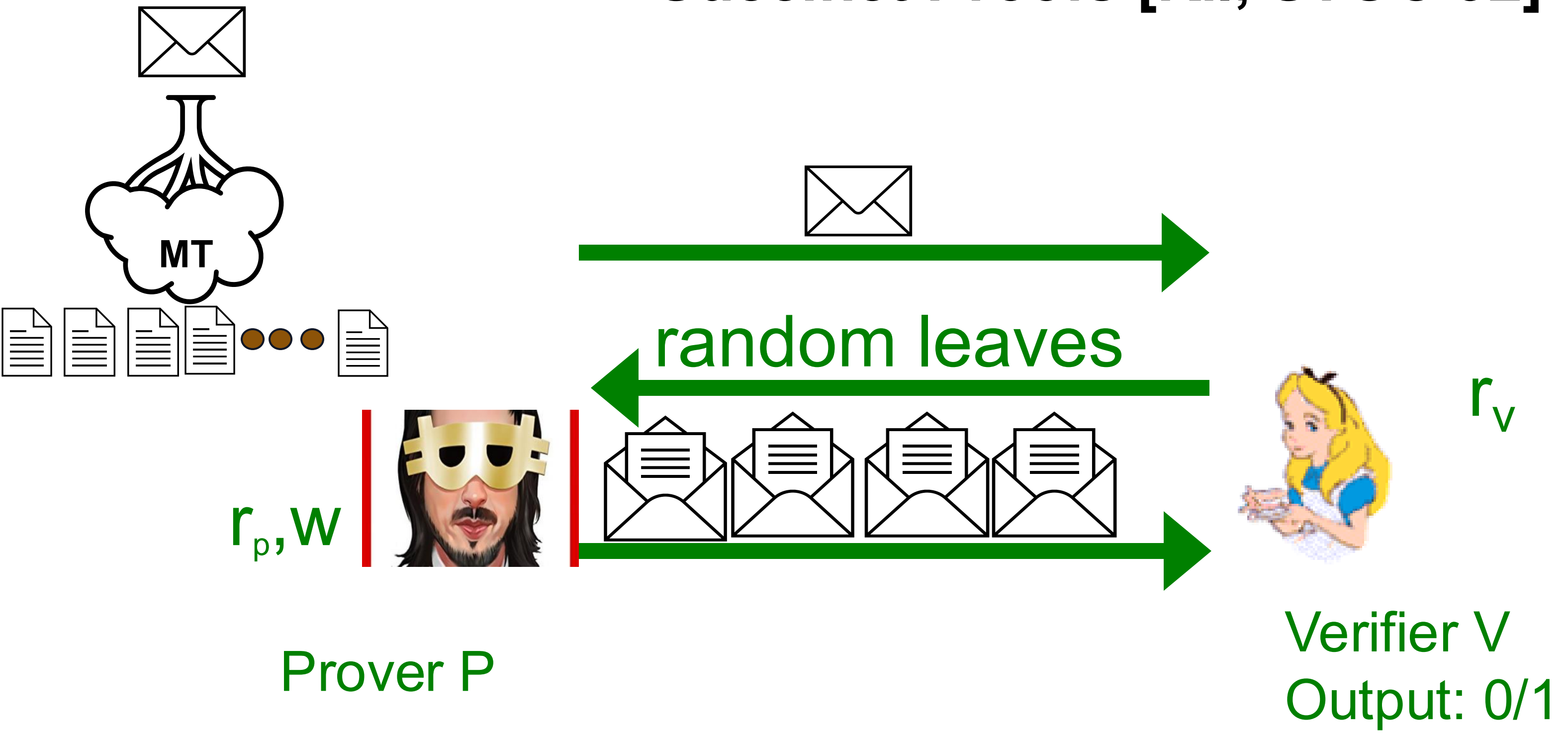
Three valid reasons justify complicated protocols:

1) Protecting the confidentiality of **w**
2) Reducing the communication
3) Reducing the computations of **V**

Prover P

Verifier V
Output: 0/1

Succinct Proofs [Kil, STOC 92]



Communication is polylogarithmic in the length of the witness and V is very efficient. Can be made WI in 3 rounds assuming a CRHF is available (4 rounds otherwise). It's public coin and with some adjustments it becomes a succinct NIZK in the random oracle model [Kil, STOC 92/Mic, SICOMP 00]

Non-Black-Box Assumptions and Simulation

NBBZK: for **any** V* there is (at least) **one** working simulator

This implies that in order to prove that a protocol is not ZK, one
should show that all possible simulators for all possible V* fail.

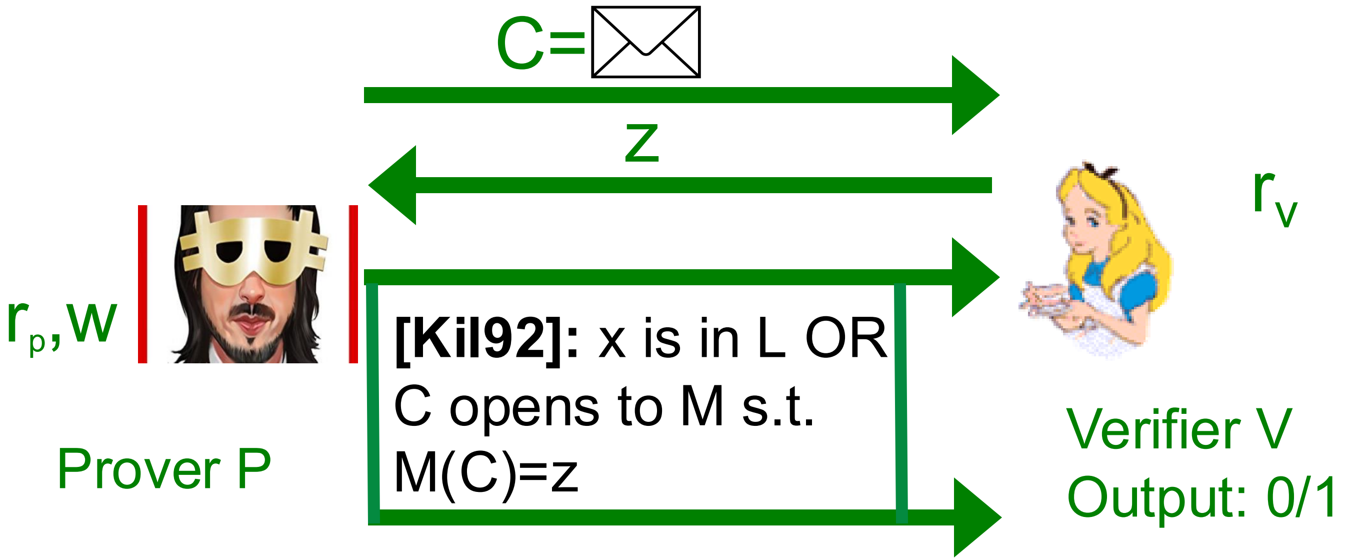
Unless the protocol is trivially insecure, proving that it is not ZK is tough.

Examples of very naïve insecure protocols that are not ZK:

- The prover openly leaks the witness. A simulator would allow to efficiently recover a witness from any instance of the language. Therefore, it is ZK only if deciding the language is trivial.

- The prover can internally run the simulator to obtain the messages for the verifier. In case of a non-trivial language this can be abused by P* to violate soundness.

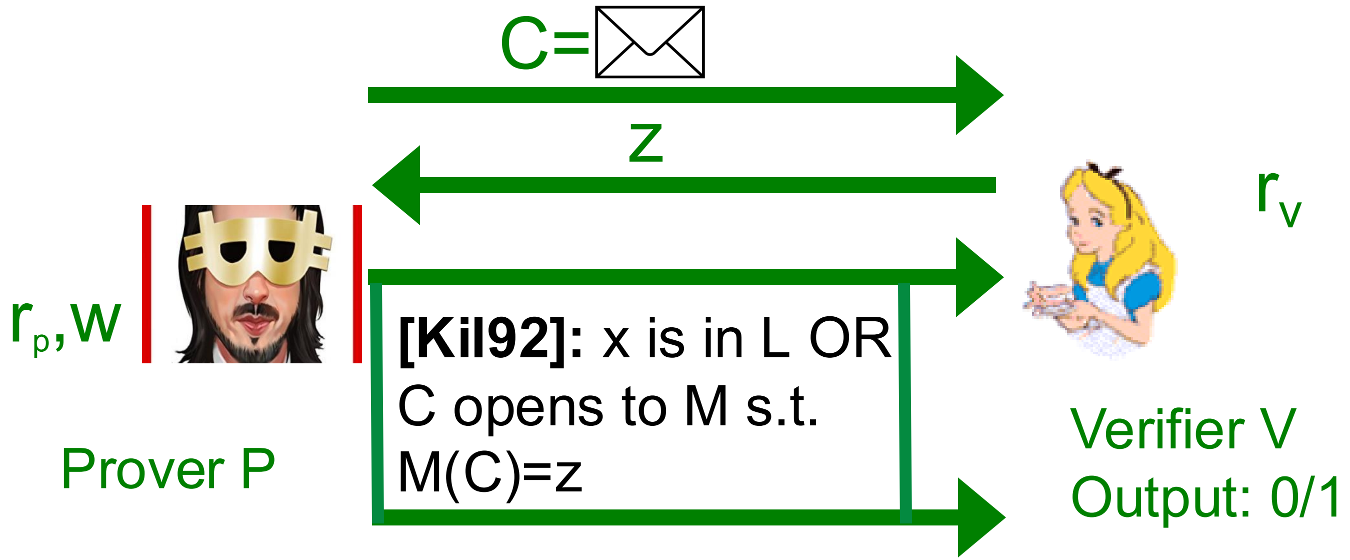
Barak's NBB ZK [Bar, FOCS 01]



[Kil92]: x is in L OR C opens to M s.t. M(C)=z

It is a public-coin constant-round ZK argument system based on CRHFs. Can we apply the FS transform?

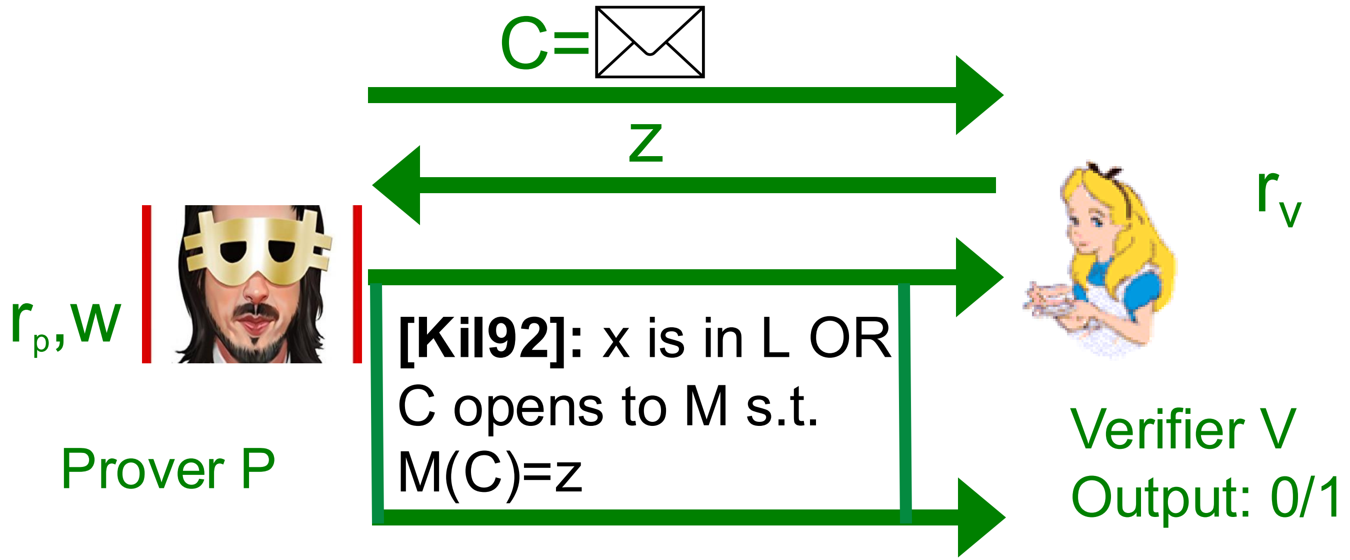
Barak's NBB ZK [Bar, FOCS 01]



It is a public-coin constant-round ZK argument system based on CRHFs. Can we apply the FS transform?

The resulting NIZK is not sound! P* can run the NBB interactive simulator internally!

Barak's NBB ZK [Bar, FOCS 01]



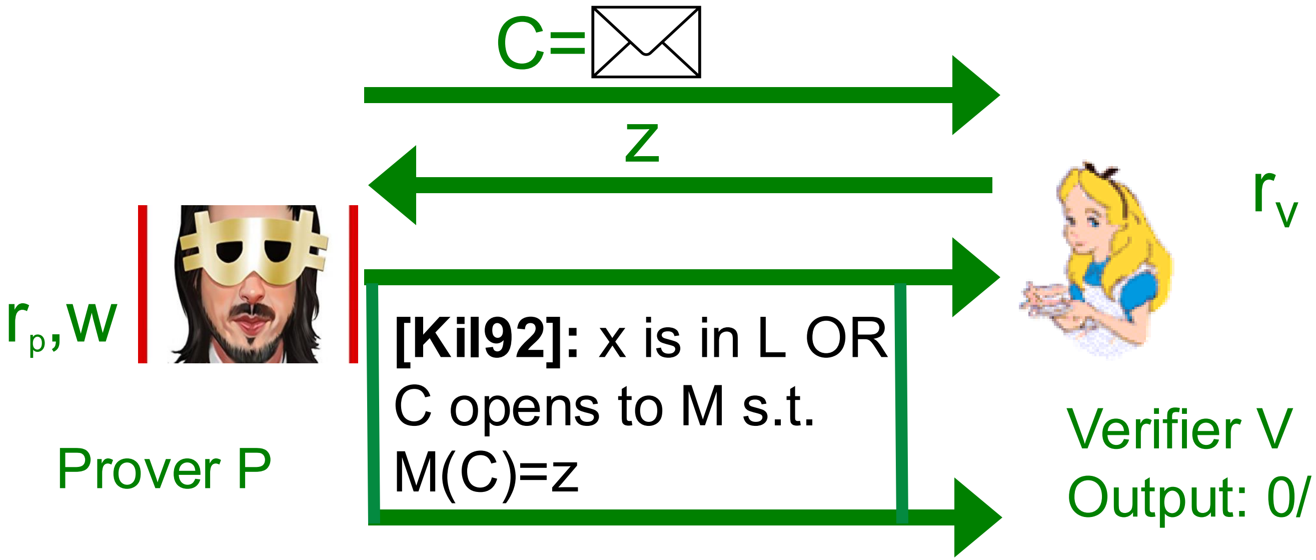
[Kil92]: x is in L OR C opens to M s.t. M(C)=z

It is a public-coin constant-round ZK argument system based on CRHFs. Can we apply the FS transform?

The resulting NIZK is not sound! P* can run the NBB interactive simulator internally!

[Bar, FOCS 01] is a counter-example to the FS transform based on the hardness of CRHFs.

Barak's NBB ZK [Bar, FOCS 01]



It is a public-coin constant-round ZK argument system based on CRHFs. Can we apply the FS transform?

The resulting NIZK is not sound! P* can run the NBB interactive simulator internally!

[Bar, FOCS 01] is a counter-example to the FS transform based on the hardness of CRHFs.

In [GOS**V**, STOC 14] we showed a construction that only needs NPRO hardness assumption). Still, the resulting NIZK is not sound!

NI Succinct Proofs

claim: x is in L

r_p,w

w →

r_V

Three valid reasons justify complicated protocols:

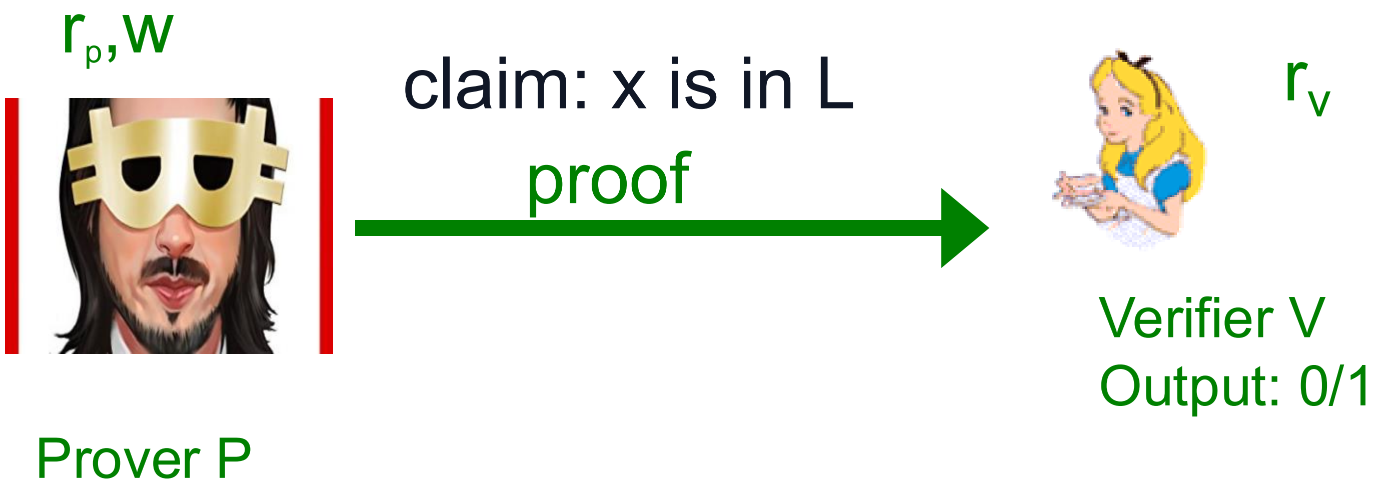
1) Protecting the confidentiality of **w**
2) Reducing the communication
3) Reducing the computations of **V**

Prover P

Verifier V
Output: 0/1

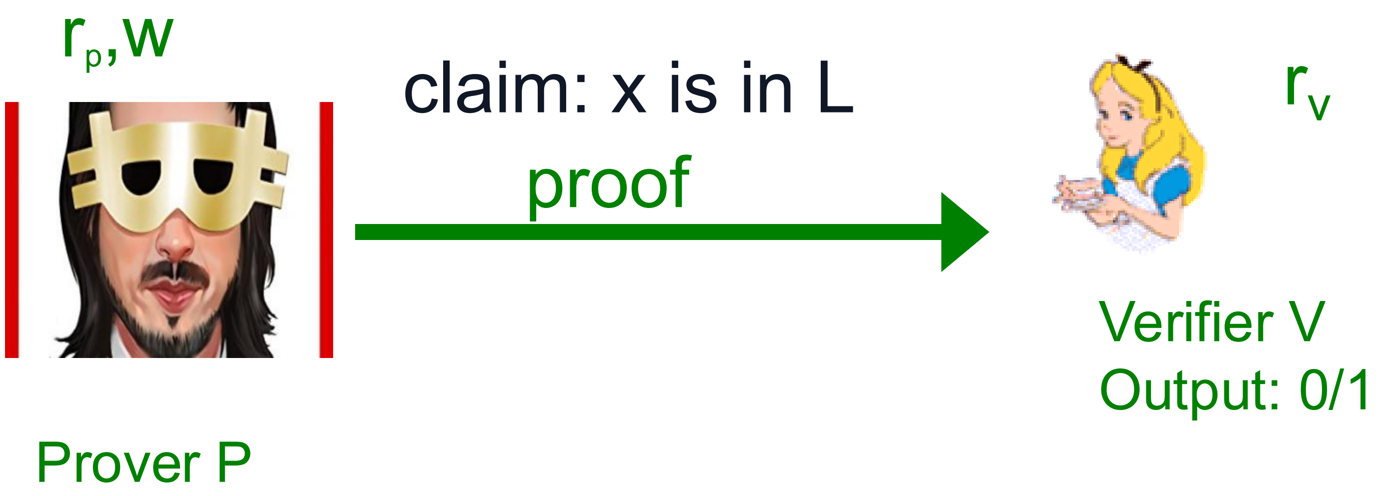
When 2) + 3) [succinctness] are both desired along with non-interactivity for any NP language you need idealized models (e.g., random oracles) or non-falsifiable assumptions [GW, ITCS 11]
(or a breakthrough about black-box separations).

SNARGs/SNARKs/STARKs...



SNARGs/SNARKs/STARKs, with their non-interactivity, succinctness, ZK (if desired) have been continuously improved and simplified for practical use

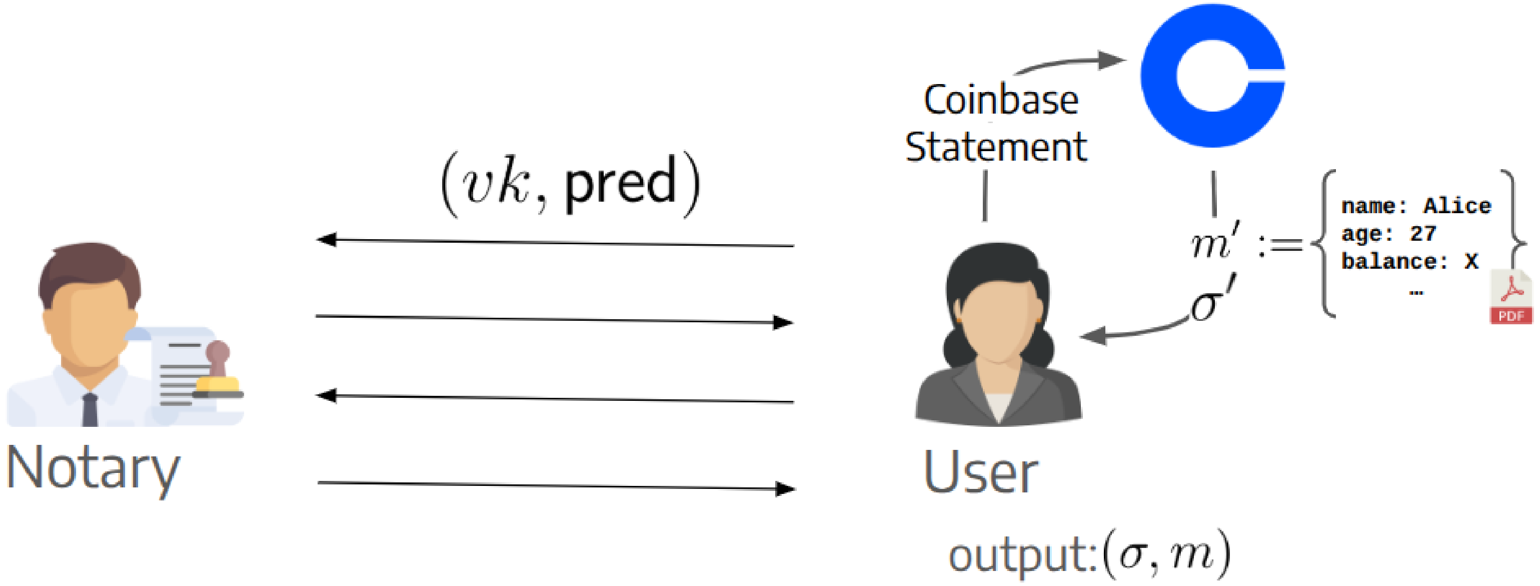
SNARGs/SNARKs/STARKs...



SNARGs/SNARKs/STARKs, with their non-interactivity, succinctness, ZK (if desired) have been continuously improved and simplified for practical use

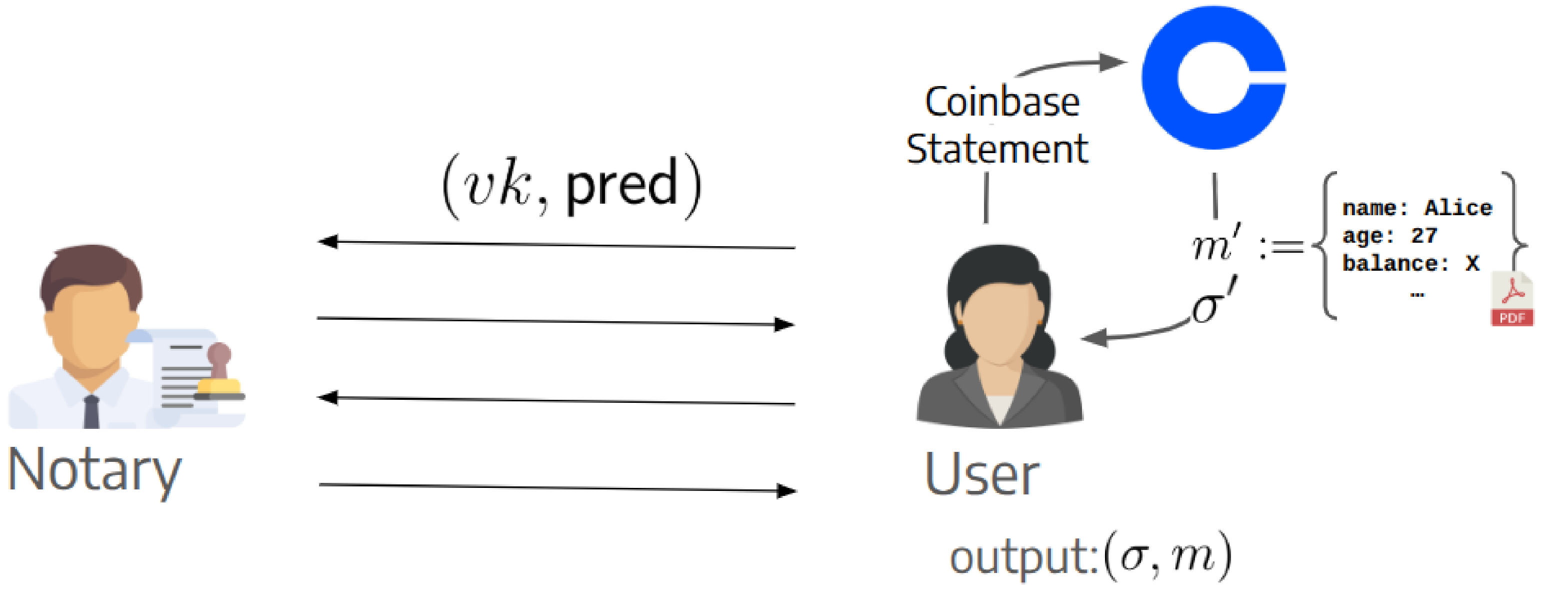
There is a price... heuristic security (**soundness also)**, complex schemes (ZCash...), lack of deniability

Deniability in Predicate Blind Signatures



In predicate blind signatures [FW, EUROCRYPT 24] the signer blindly signs a message only if it satisfies a predicate (naturally, different sessions can have different predicates)

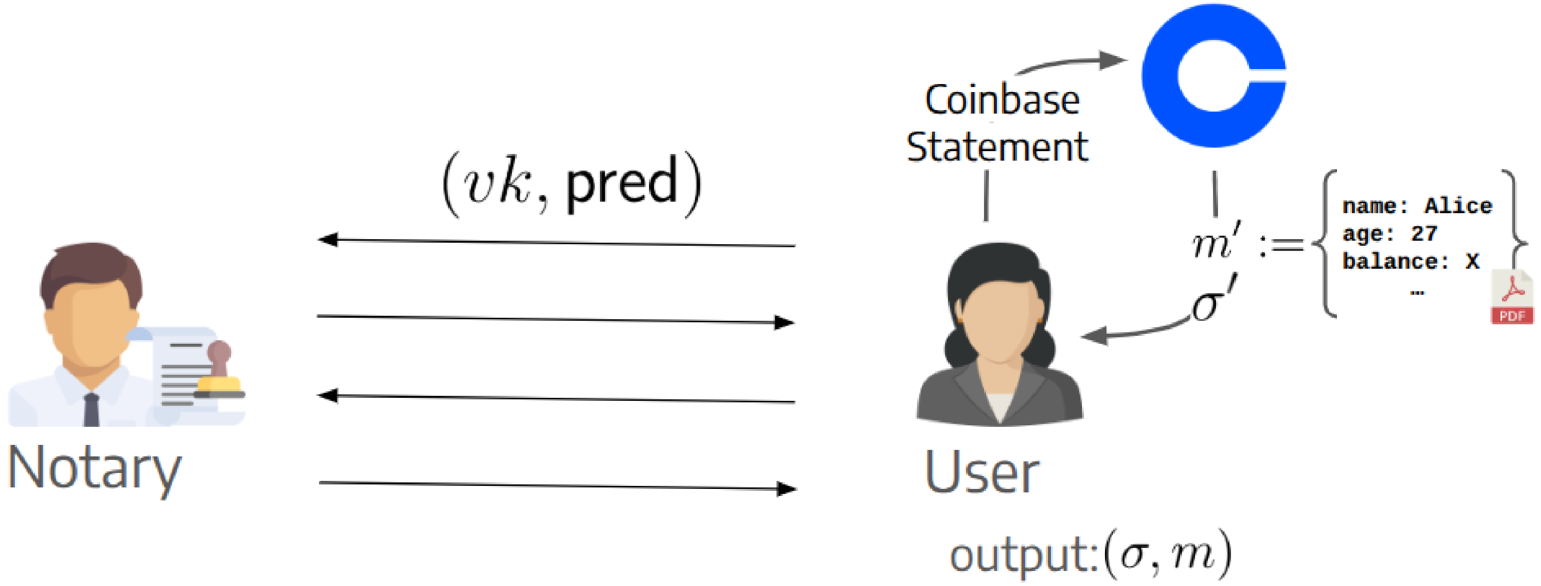
Deniability in Predicate Blind Signatures



In predicate blind signatures [FW, EUROCRYPT 24] the signer blindly signs a message only if it satisfies a predicate (naturally, different sessions can have different predicates). Crucially used for secure zkTLS [DVVZ, NDSS 26]

However, a NIZK about pred(m) gives too much information to the signer

Deniability in Predicate Blind Signatures

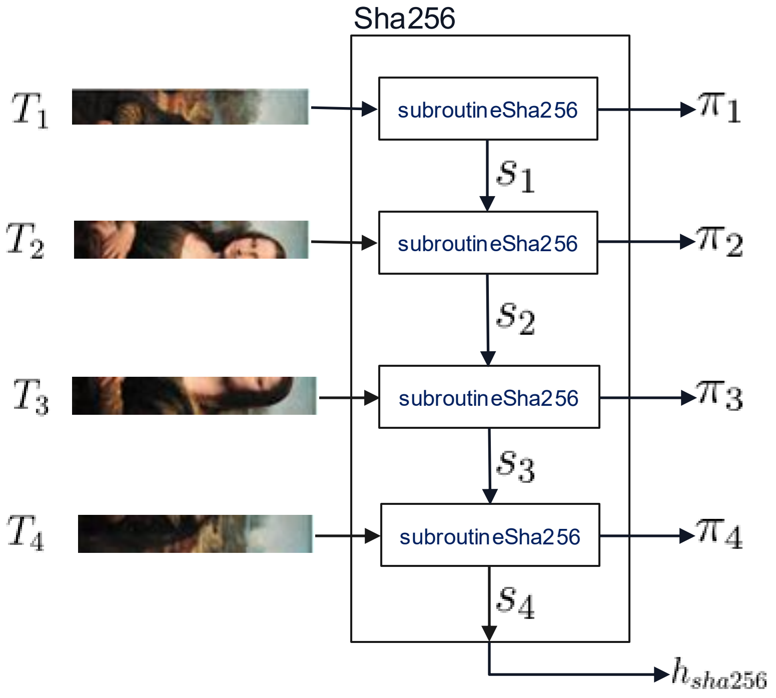


In predicate blind signatures [FW, EUROCRYPT 24] the signer blindly signs a message only if it satisfies a predicate (naturally, different sessions can have different predicates). Crucially used for secure zkTLS [D**V**VZ, NDSS 26]

Howeve, a NIZK about pred(m) gives too much information to the signer

Interactive ZK, provides deniability in predicate blind signatures [D**V**, eprint report 1992, 2025]

Are ZK Proofs for Efficient for Existing/Standardized Symmetric-Key Crypto?



$$\Big\{(\mathrm{Resize}(\cdot), \hat{T}_2, c_2, z_2, z_1), (T_2, s_2, s_1, r_1, r_2, r_3) : \\ s_2 = \mathrm{subroutineSha256}(T_2, s_1) \wedge \\ c_2 = \mathrm{Comm}(T_2, r_3) \wedge \\ z_1 = \mathrm{Comm}(s_1, r_1) \wedge \\ z_2 = \mathrm{Comm}(s_2, r_2) \wedge \\ \hat{T}_2 = \mathrm{Resize}(T_2)\Big\}$$

If you do not crucially require succinctness, then you can use commodity hardware to compute proofs [DVVZ, S&P 2025] remaining with existing standards.

Are 2) and 3) always together?

claim: x is in L

r_p,w

$\xrightarrow{w}$

r_V

Three valid reasons justify complicated protocols:

1) Protecting the confidentiality of **w**
2) Reducing the communication
3) Reducing the computations of V

Prover P

Verifier V
Output: 0/1

Are ZK Proofs for Efficient for Existing/Standardized Symmetric-Key Crypto?

Techniques like Stacking Sigmas [GGH-AK, EUROCRYPT 2022]
allow for compact NIZK proofs in the RO model with linear time computations for prover and verifier **using the sigma protocol in a black-box way**, under DLOG.

[AS**V** 2026, to appear] obtain a similar results without adding any hardness assumption.

Another evidence that, when putting succinctness on the side, a lot can be done to remain compatible with existing infrastructures.

Conclusion

Non-interactivity and succinctness are useful in some applications but an overkill in others, increasing risks of attacks to integrity and confidentiality.

It is nowadays unclear how to sanitize a claim to make sure that is not affected by the failures of the Fiat-Shamir transform.

Several heuristic assumptions/models are used for SNARKs/STARKs and similar heuristics could allow one to get efficient interactive ZK proofs to be used in many scenarios (out of the blockchain world).

Splitting succinctness in compactness and verifier efficiency can allow significant improvements or applicability to concrete scenarios.

THANKS!