

Steganography-Free Zero-Knowledge

Behzad Abdolmaleki ¹, Nils Fleischhacker ², Vipul Goyal ³, Abhishek Jain ⁴,
and Giulio Malavolta ¹

¹ Max Planck Institute for Security and Privacy, Bochum, Germany

² Ruhr University Bochum, Germany

³ NTT Research and Carnegie Mellon University, USA

⁴ Johns Hopkins University, Baltimore, USA

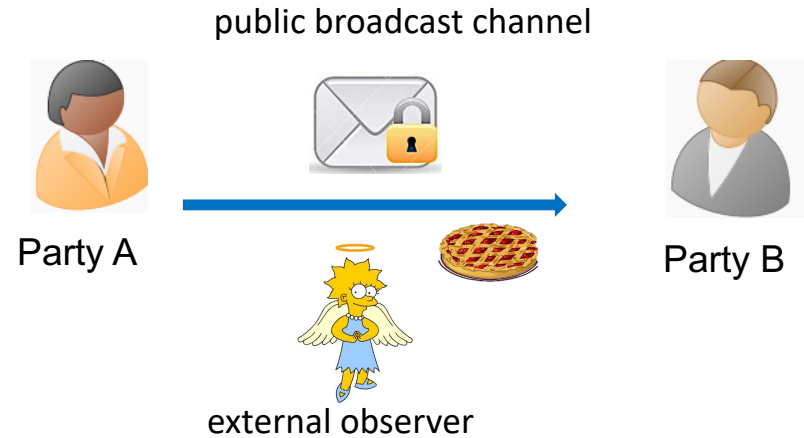
Outline

- Motivations (definition and challenges in steganography)
- Results of the paper
- A new model for preventing steganography
 - Defining steganography freeness
- Construction:
 - Preliminaries and terminologies
 - Steganography-Free Zero-Knowledge argument system

Motivations and Challenges

- Definition: Steganographic Communication:

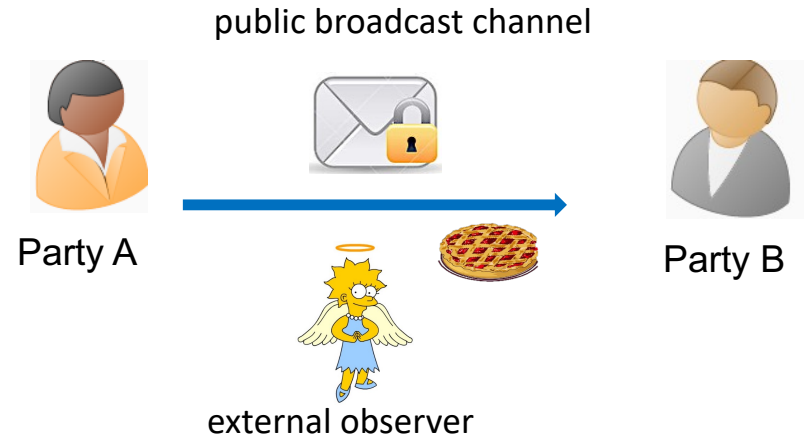
Party A wants to transmit a secret message to another party B by communicating over a public broadcast channel without being detected by an external observer who is listening on the channel.



Motivations and Challenges

- Definition: Steganographic Communication:

Party A wants to transmit a secret message to another party B by communicating over a public broadcast channel without being detected by an external observer who is listening on the channel.



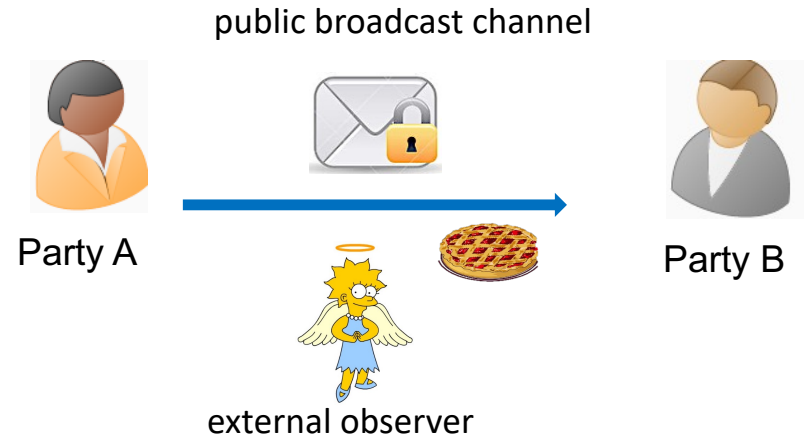
➤ An encrypted channel can be easy to detect.

A may instead try to embed its message in an innocuous-looking conversation. (i.e., it may send a photograph of a person to securely transmit bit 0 if the 30th hair from the left is white, and 1 otherwise.) [LMS05].

Motivations and Challenges

- Definition: Steganographic Communication:

Party A wants to transmit a secret message to another party B by communicating over a public broadcast channel without being detected by an external observer who is listening on the channel.



➤ An encrypted channel can be easy to detect.

A may instead try to embed its message in an innocuous-looking conversation. (i.e., it may send a photograph of a person to securely transmit bit 0 if the 30th hair from the left is white, and 1 otherwise.) [LMS05].

➤ It is known that such steganographic communication is always possible in any system with some entropy, and is provably impossible to detect [KatPet02,HLA02,BacCas05,Cashin20].

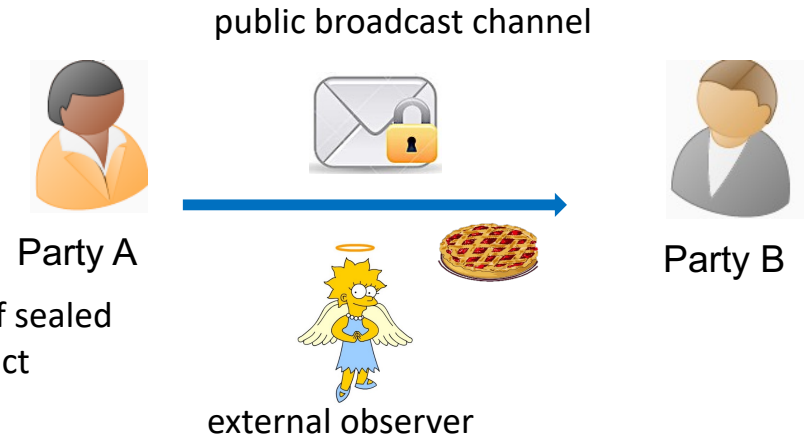
Motivations and Challenges

- Preventing Steganographic Communication:

- ✓ Collusion-Free Protocols.

a collusion-free multiparty protocol prevents a group of adversarial parties from colluding with each other to gain an unfair advantage over honest participants [LMS05].

Requirement: Physical assumptions: simultaneous exchange of sealed envelopes, and an interactive pre-processing model to construct collusion-free protocols.



Motivations and Challenges

- Preventing Steganographic Communication:

- ✓ Collusion-Free Protocols.

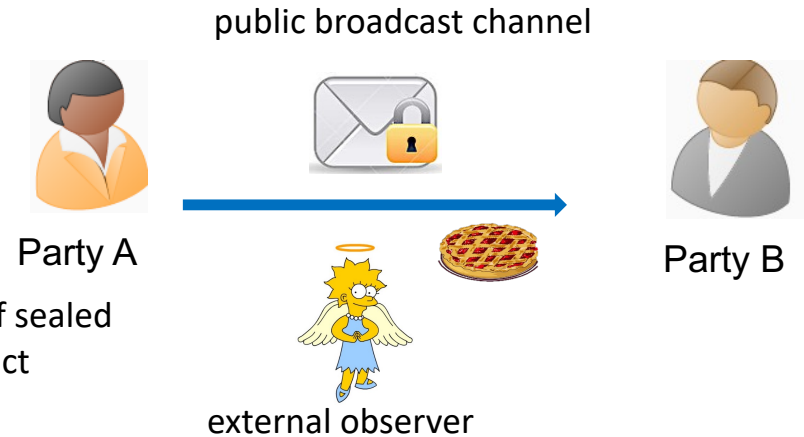
a collusion-free multiparty protocol prevents a group of adversarial parties from colluding with each other to gain an unfair advantage over honest participants [LMS05].

Requirement: Physical assumptions: simultaneous exchange of sealed envelopes, and an interactive pre-processing model to construct collusion-free protocols.

- ✓ via Sanitization

It considers a mediator model for collusion-free protocols to avoid the use of pre-processing and physical channels. This active mediator has the ability to modify the messages of the protocol participants [ASV08, MirDav15, ..., CDN20, CGPS21].

Requirement: There is an entity (namely, the reverse firewall) that sits on the network of each participant and has the ability to re-randomize the messages sent by the parties.



Motivations and Challenges

- Preventing Steganographic Communication:

- ✓ Collusion-Free Protocols.

a collusion-free multiparty protocol prevents a group of adversarial parties from colluding with each other to gain an unfair advantage over honest participants [LMS05].

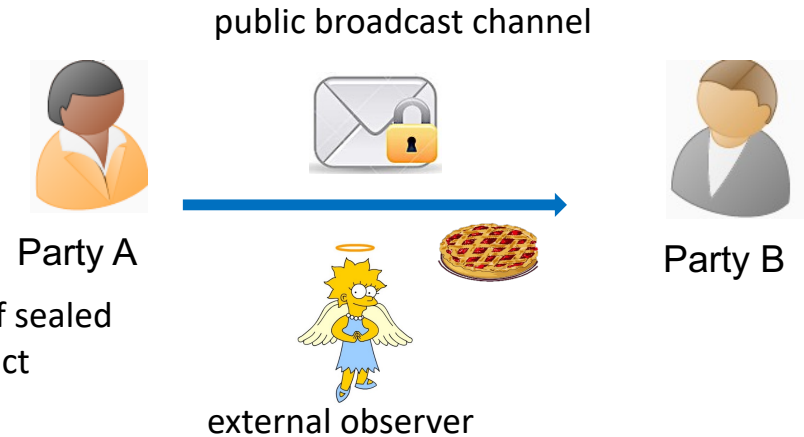
Requirement: Physical assumptions: simultaneous exchange of sealed envelopes, and an interactive pre-processing model to construct collusion-free protocols.

- ✓ via Sanitization

It considers a mediator model for collusion-free protocols to avoid the use of pre-processing and physical channels. This active mediator has the ability to modify the messages of the protocol participants [ASV08, MirDav15, ..., CDN20, CGPS21].

Requirement: There is an entity (namely, the reverse firewall) that sits on the network of each participant and has the ability to re-randomize the messages sent by the parties.

- ✓ Trusted Initialization Phase

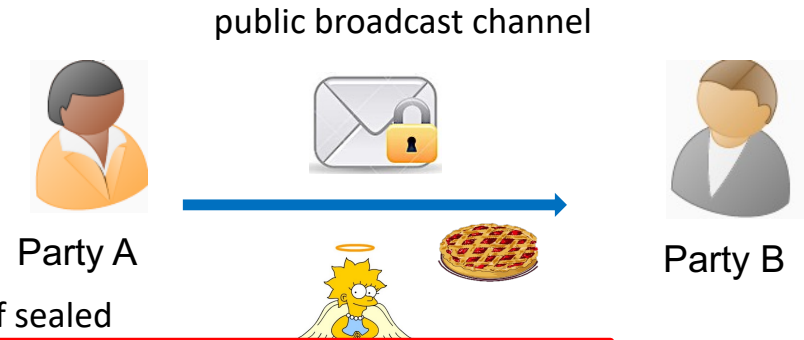


Motivations and Challenges

- Preventing Steganographic Communication:

- ✓ Collusion-Free Protocols.

a collusion-free multiparty protocol prevents a group of adversarial parties from colluding with each other to gain an unfair advantage over honest participants [LMS05].



Requirement: Physical assumptions: simultaneous exchange of sealed

en
co

Main question:

Is there any (more realistic model) in the steganography communications to ensure that any attempts at steganographic communication during the execution phase will be detected by the external observer?

It c
processing and physical channels. This active mediator has the ability to modify the messages of the protocol participants [ASV08,MirDav15,...,CDN20,CGPS21].

Requirement: There is an entity (namely, the reverse firewall) that sits on the network of each participant and has the ability to re-randomize the messages sent by the parties.

- ✓ Trusted Initialization Phase

Our Results

Our Results

- Defining an new model for steganography freeness
- New Primitive: Steganography-Free Zero-Knowledge (SF-ZK)
- Construct SF-ZK argument systems (with black-box simulation for all languages in NP):
 - In the single-execution setting
 - In the he multi-execution setting. (where the pre-processing can be refreshed to allow for an unbounded number of execution phases)
- Optimality of our Model: showing that our adversarial model is “tight”. Specifically, we show that when both the prover and the verifier are malicious during the pre-processing, SF-ZK is impossible, except for languages in BPP.

Our Results

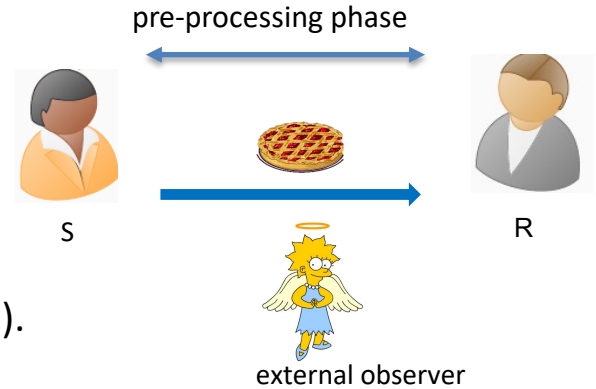
- Defining an new model for steganography freeness
- New Primitive: Steganography-Free Zero-Knowledge (SF-ZK)
- Construct SF-ZK argument systems (with black-box simulation for all languages in NP):
 - In the single-execution setting
 - In the multi-execution setting. (where the pre-processing can be refreshed to allow for an unbounded number of execution phases)
- Optimality of our Model: showing that our adversarial model is “tight”. Specifically, we show that when both the prover and the verifier are malicious during the pre-processing, SF-ZK is impossible, except for languages in BPP.

Steganography-Free Zero-Knowledge

➤ A New Model for Preventing Steganography

Any communication (via an interactive protocol) proceeds in two phases:

- (i) A non-interactive pre-processing phase
(Each party publishes a single message)
- (ii) An execution phase (corresponds to the actual protocol execution).



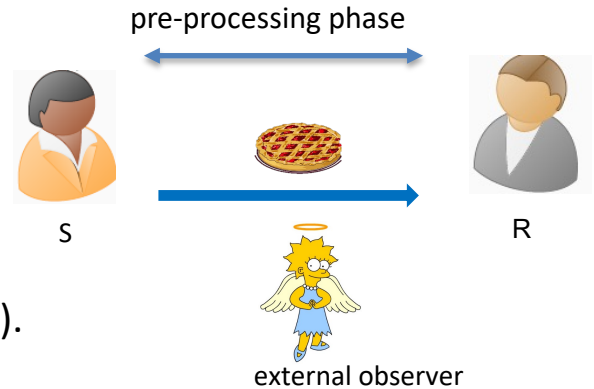
Steganography-Free Zero-Knowledge

➤ A New Model for Preventing Steganography

Any communication (via an interactive protocol) proceeds in two phases:

- (i) A non-interactive pre-processing phase
(Each party publishes a single message)
- (ii) An execution phase (corresponds to the actual protocol execution).

We assume that only one of the parties is required to be honest during the pre-processing phase, but may be completely malicious during the execution phase.

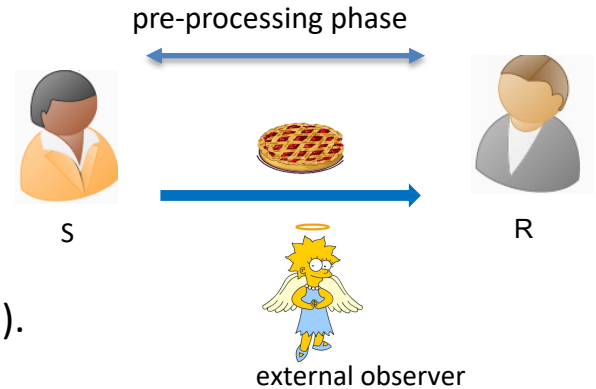


Steganography-Free Zero-Knowledge

➤ A New Model for Preventing Steganography

Any communication (via an interactive protocol) proceeds in two phases:

- (i) A non-interactive pre-processing phase (Each party publishes a single message)
- (ii) An execution phase (corresponds to the actual protocol execution).



We assume that only one of the parties is required to be honest during the pre-processing phase, but may be completely malicious during the execution phase.

➤ Defining Steganography Freeness

It is for generic interactive protocols (S,R) in the non-interactive pre-processing model:

- At the start of the execution phase, the adversarial S is given a randomly chosen bit b .
- At the end of the execution phase:

$\Pr[R \text{ correctly guesses } b \wedge \text{ the execution transcript is accepted by the observer}]$ is $\text{neg}(\cdot)$

The Key-Primitives:

Zero-Knowledge with Super-Polynomial Simulation

NIWI with Honest Pre-Processing

The Key-Primitives

□ Zero-Knowledge with Super-Polynomial Simulation [Pass03]


We assume the existence of a three-round, public-coin zero-knowledge protocol $(\text{SPS-P}, \text{SPS-V})$ for NP with the following properties:



The Key-Primitives

□ Zero-Knowledge with Super-Polynomial Simulation [Pass03]


We assume the existence of a three-round, public-coin zero-knowledge protocol (SPS-P,SPS-V) for NP with the following properties:

- 
- 1) The protocol is simulatable in super-polynomial time.
 - 2) For any first and second round message pair (α, β) , there exists only one accepting third message γ
 - 3) Fix an accepting transcript (α, β, γ) , then either of the following conditions must hold:
 - (a) The witness is extractable (possibly in super-polynomial time) from (α, β, γ) .
 - (b) There is at most one β such that (α, β, γ) is accepting.

The Key-Primitives

❑ Zero-Knowledge with Super-Polynomial Simulation [Pass03]

We assume the existence of a three-round, public-coin zero-knowledge protocol (SPS-P,SPS-V) for NP with the following properties:

- 
- 1) The protocol is simulatable in super-polynomial time.
 - 2) For any first and second round message pair (α, β) , there exists only one accepting third message γ
 - 3) Fix an accepting transcript (α, β, γ) , then either of the following conditions must hold:
 - (a) The witness is extractable (possibly in super-polynomial time) from (α, β, γ) .
 - (b) There is at most one β such that (α, β, γ) is accepting.

❑ NIWI with Honest Pre-Processing [BGT20]:

HPP-NIWI is a special kind of NIWI arguments where computation of the argument is split into **two steps**.

The Key-Primitives

❑ Zero-Knowledge with Super-Polynomial Simulation [Pass03]

We assume the existence of a three-round, public-coin zero-knowledge protocol (SPS-P,SPS-V) for NP with the following properties:

- 1) The protocol is simulatable in super-polynomial time.
- 2) For any first and second round message pair (α, β) , there exists only one accepting third message γ
- 3) Fix an accepting transcript (α, β, γ) , then either of the following conditions must hold:
 - (a) The witness is extractable (possibly in super-polynomial time) from (α, β, γ) .
 - (b) There is at most one β such that (α, β, γ) is accepting.

❑ NIWI with Honest Pre-Processing [BGT20]:

HPP-NIWI is a special kind of NIWI arguments where computation of the argument is split into **two steps**.

In an initial statement-independent step : The prover commits to a preliminary value.

In the second step, which now depends on the statement x to be proven, the actual argument π is computed. The soundness of an HPP-NIWI is only guaranteed as long as the pre-processing step is performed honestly.

Our Results II: Construction

New Primitive: Steganography-Free Zero-Knowledge (SF-ZK)

Construction of SF-ZK

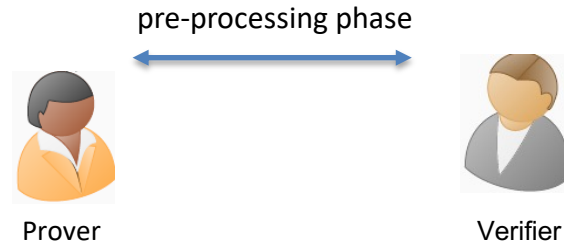
New Primitive: Steganography-Free Zero-Knowledge

➤ Definition: Steganography-Free Zero-Knowledge (SF-ZK)

An SF-ZK argument proceeds in two phases:

1) **Non-interactive pre-processing:** both parties send a single message to each other.

(This is before the prover receives the **statement x** and the **witness w**).

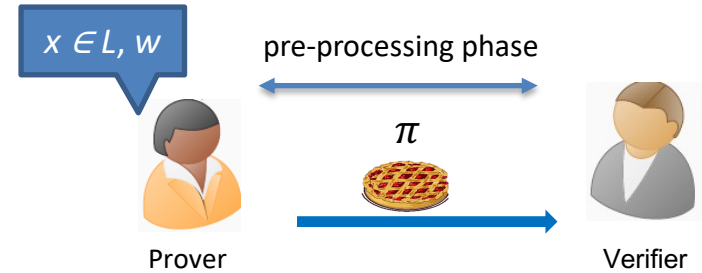


New Primitive: Steganography-Free Zero-Knowledge

➤ Definition: Steganography-Free Zero-Knowledge (SF-ZK)

An SF-ZK argument proceeds in two phases:

- 1) **Non-interactive pre-processing:** both parties send a single message to each other.
(This is before the prover receives the **statement x** and the **witness w**).
- 2) **Execution phase:** the prover proves the validity of the **statement x** .

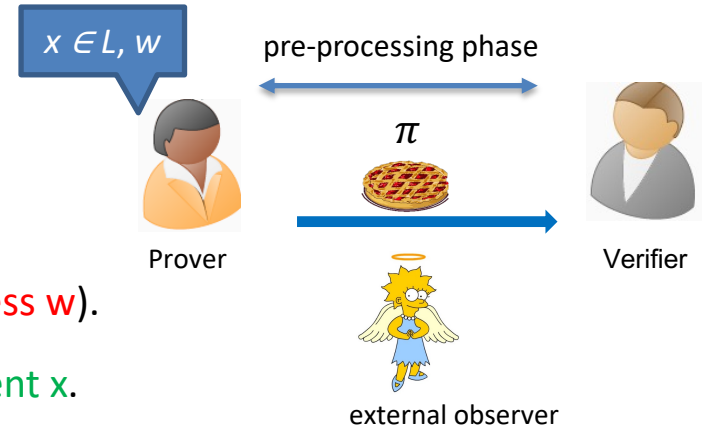


New Primitive: Steganography-Free Zero-Knowledge

➤ Definition: Steganography-Free Zero-Knowledge (SF-ZK)

An SF-ZK argument proceeds in two phases:

- 1) **Non-interactive pre-processing:** both parties send a single message to each other.
(This is before the prover receives the **statement x** and the **witness w**).
- 2) **Execution phase:** the prover proves the validity of the **statement x** .



Security requirements:

Completeness: $x \in L \Rightarrow V \text{ accepts } \pi$

Soundness: $x \notin L \Rightarrow V \text{ rejects } \pi$

Zero-Knowledge: $\pi \text{ leaks nothing beyond } x \in L$



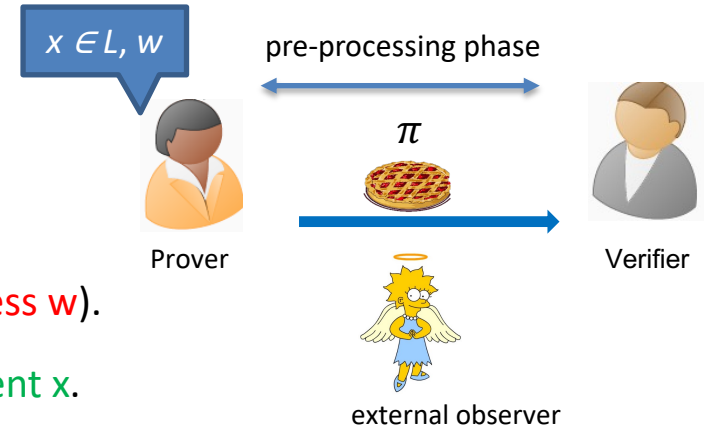
standard ZK scheme's properties

New Primitive: Steganography-Free Zero-Knowledge

➤ Definition: Steganography-Free Zero-Knowledge (SF-ZK)

An SF-ZK argument proceeds in two phases:

- 1) **Non-interactive pre-processing:** both parties send a single message to each other.
(This is before the prover receives the **statement x** and the **witness w**).
- 2) **Execution phase:** the prover proves the validity of the **statement x** .



Security requirements:

Completeness: $x \in L \Rightarrow V$ accepts π

Soundness: $x \notin L \Rightarrow V$ rejects π

Zero-Knowledge: π leaks nothing beyond $x \in L$

} standard ZK scheme's properties

Observer Soundness: $x \notin L \Rightarrow$ no coalition of P and V can produce a transcript that will be accepted by the external observer

Computationally Unique Transcripts (CUT):

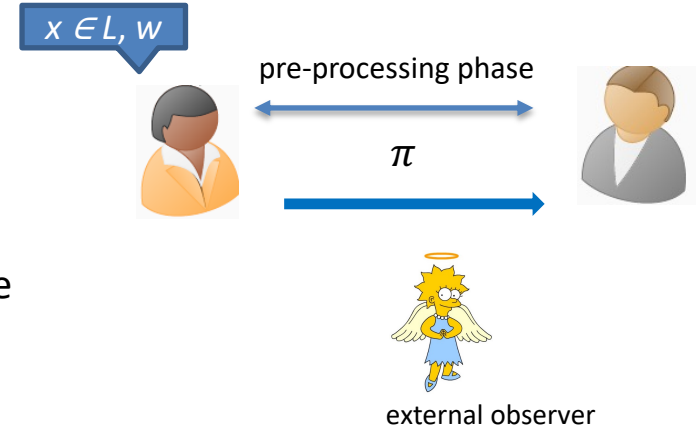
For any $x \in L \Rightarrow$ two different sets of efficient prover and verifier strategies cannot produce two different transcripts of the execution phase that will both be accepted by the observer.

Construction: Steganography-Free Zero-Knowledge

➤ SF-ZK argument system with black-box simulation in the single-execution setting:

- **Key conceptual challenge in constructing SF-ZK:**

- A black-box simulator works by rewinding the adversarial verifier potentially multiple times.
- This involves creating multiple protocol transcripts which are necessarily different (for the rewinding to be “successful”)



Construction: Steganography-Free Zero-Knowledge

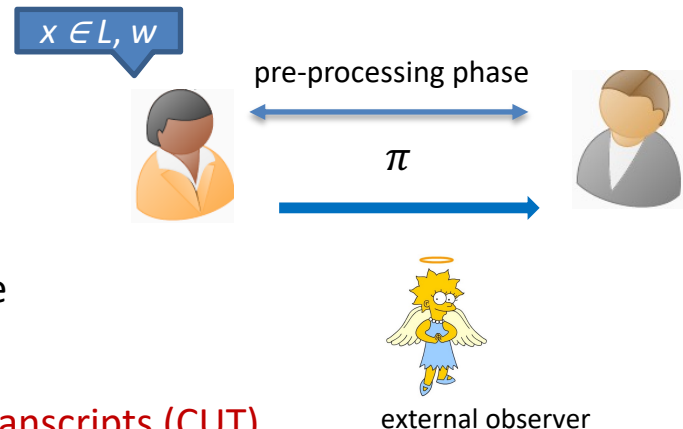
➤ SF-ZK argument system with black-box simulation in the single-execution setting:

- Key conceptual challenge in constructing SF-ZK:

- A black-box simulator works by rewinding the adversarial verifier potentially multiple times.

- This involves creating multiple protocol transcripts which are necessarily different (for the rewinding to be “successful”)

- This seems to be at odds with the **computationally unique transcripts (CUT)** property of SF-ZK; indeed, since the simulator is also an efficient algorithm, intuitively, it should also not be able to produce multiple transcripts of the execution phase.



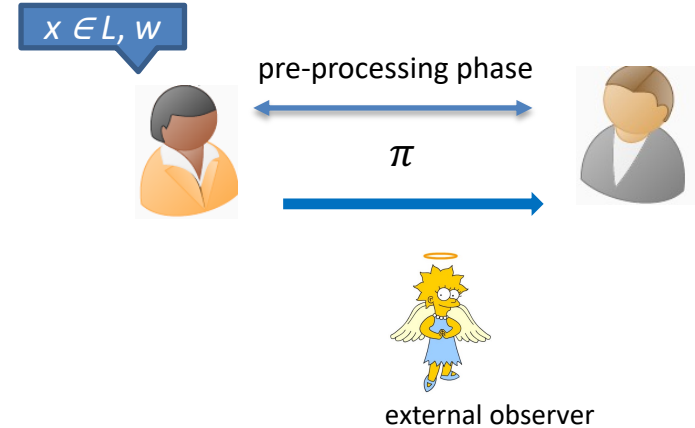
How can we achieve ZK property without violating the CUT property (or vice-versa)?

Construction: Steganography-Free Zero-Knowledge

- SF-ZK argument system with black-box simulation in the single-execution setting:

Assuming sub-exponentially hard injective one-way functions f .

$$L_{NIWI} = \left\{ \begin{array}{l} (x, w) \in R \wedge c'' = \text{Com}(0; r') \wedge c' = \text{Com}(r') \\ \forall c = \text{Com}(1; r) \wedge c' = \text{Com}(w'; r') \\ \forall (x, w) \in R \wedge c'' = \text{Com}(1; r) \wedge f(z) = y \end{array} \right.$$

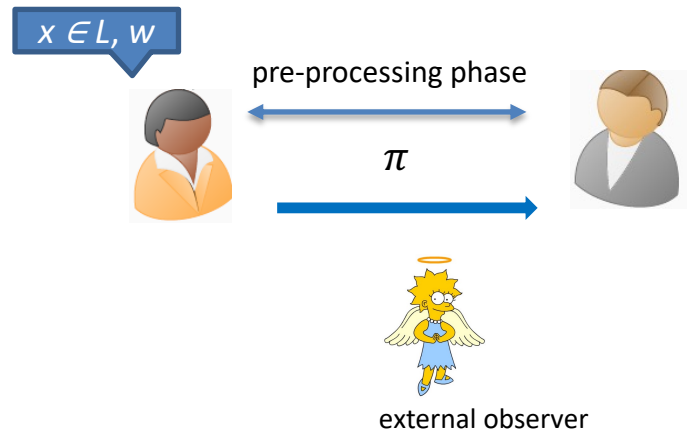


Construction: Steganography-Free Zero-Knowledge

- SF-ZK argument system with black-box simulation in the single-execution setting:

Assuming sub-exponentially hard injective one-way functions f .

$$L_{NIWI} = \left\{ \begin{array}{l} (x, w) \in R \wedge c'' = \text{Com}(0; r') \wedge c' = \text{Com}(r') \\ \vee \quad c = \text{Com}(1; r) \wedge c' = \text{Com}(w'; r') \\ \vee \quad (x, w) \in R \wedge c'' = \text{Com}(1; r) \wedge f(z) = y \end{array} \right.$$



Pre-Processing:

- Computes (i) c as a **commitment to 0** and (ii) to **some random coin r'** .
 - (i) Guarantees if P's pre-processing is honest, then it is hard to cheat in the execution phase.
 - (ii) the latter fixes the random coins used later in the execution phase.
- Initializes the pre-processing τ of an **HPP-NIWI** proof.
- Computes the **first message α** of an SPS-ZK proof: τ is well-formed.

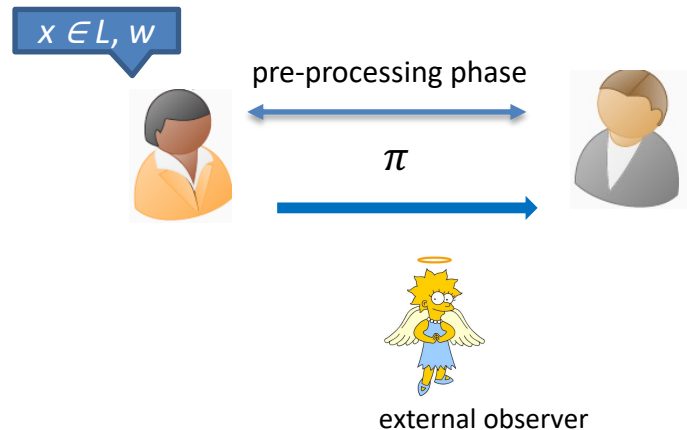


Construction: Steganography-Free Zero-Knowledge

➤ SF-ZK argument system with black-box simulation in the single-execution setting:

Assuming sub-exponentially hard injective one-way functions f .

$$L_{NIWI} = \left\{ \begin{array}{l} (x, w) \in R \wedge c'' = \text{Com}(0; r') \wedge c' = \text{Com}(r') \\ \vee \quad c = \text{Com}(1; r) \wedge c' = \text{Com}(w'; r') \\ \vee \quad (x, w) \in R \wedge c'' = \text{Com}(1; r) \wedge f(z) = y \end{array} \right.$$



Pre-Processing:

- Computes (i) c as a **commitment to 0** and (ii) to **some random coin r'** .



- (i) Guarantees if P's pre-processing is honest, then it is hard to cheat in the execution phase.
- (ii) the latter fixes the random coins used later in the execution phase.

- Initializes the pre-processing τ of an **HPP-NIWI** proof.

- Computes the **first message α** of an SPS-ZK proof: τ is well-formed.



- Samples a **random image y** from the domain of the one-way function f .

- Samples a **random instance x'** of an average-case hard language R' with unique witnesses w'

- Computes a **commitment cv** to a randomly sampled second message β of the SPS-ZK proof.

Construction: Steganography-Free Zero-Knowledge

- SF-ZK argument system with black-box simulation in the single-execution setting:

$$L_{NIWI} = \left\{ \begin{array}{l} (x, w) \in R \wedge c'' = \text{Com}(0; r') \wedge c' = \text{Com}(r') \\ \forall c = \text{Com}(1; r) \wedge c' = \text{Com}(w'; r') \\ \forall (x, w) \in R \wedge c'' = \text{Com}(1; r) \wedge f(z) = y \end{array} \right.$$

Execution phase:



commitment c'' to 0
(using the random coins r' fixed in the pre-processing)



Construction: Steganography-Free Zero-Knowledge

➤ SF-ZK argument system with black-box simulation in the single-execution setting:

$$L_{NIWI} = \begin{cases} (x, w) \in R \wedge c'' = \text{Com}(0; r') \wedge c' = \text{Com}(r') \\ \forall c = \text{Com}(1; r) \wedge c' = \text{Com}(w'; r') \\ \forall (x, w) \in R \wedge c'' = \text{Com}(1; r) \wedge f(z) = y \end{cases}$$

Execution phase:



commitment c'' to 0
(using the random coins r' fixed in the pre-processing)



decommitment β to c_v and reveals the unique witness w'



Checks: - β is a valid decommitment for c_v
- (x', w') is in R'

SPS-P2 (certifies that τ is well-formed)
WI-P2 (x accepting instance of L_{NIWI})



Construction: Steganography-Free Zero-Knowledge

➤ SF-ZK argument system with black-box simulation in the single-execution setting:

$$L_{NIWI} = \begin{cases} (x, w) \in R \wedge c'' = \text{Com}(0; r') \wedge c' = \text{Com}(r') \\ \forall c = \text{Com}(1; r) \wedge c' = \text{Com}(w'; r') \\ \forall (x, w) \in R \wedge c'' = \text{Com}(1; r) \wedge f(z) = y \end{cases}$$

A ZK's Sim:

- produce a transcript of the execution phase by committing to 0 in c'' and then learn the w' for the trapdoor statement.
- The Sim can rewind the verifier to the start of the execution phase and generate a new transcript where it commits to the trapdoor witness w' .

Execution phase:

commitment c'' to 0
(using the random coins r' fixed in the pre-processing)



decommitment β to cv and reveals the unique witness w'



Checks: - β is a valid decommitment for cv
- (x', w') is in R'

SPS-P2 (certifies that τ is well-formed)
WI-P2 (x accepting instance of L_{NIWI})



Summary

- Defining a new model for steganography freeness
- New Primitive: Steganography-Free Zero-Knowledge (SF-ZK)
- Construct SF-ZK argument systems (with black-box simulation for all languages in NP):
 - In the single-execution setting
 - In the multi-execution setting. (where the pre-processing can be refreshed to allow for an unbounded number of execution phases)
- Optimality of our Model: showing that our adversarial model is “tight”. Specifically, we show that when both the prover and the verifier are malicious during the pre-processing, SF-ZK is impossible, except for languages in BPP.
- Open Problem: Investigating SF in the multi-party computation setting.

Summary

- Defining a new model for steganography freeness
- New Primitive: Steganography-Free Zero-Knowledge (SF-ZK)
- Construct SF-ZK argument systems (with black-box simulation for all languages in NP):
 - In the single-execution setting
 - In the multi-execution setting. (where the pre-processing can be refreshed to allow for an unbounded number of execution phases)
- Optimality of our Model: showing that our adversarial model is “tight”. Specifically, we show that when both the prover and the verifier are malicious during the pre-processing, SF-ZK is impossible, except for languages in BPP.
- Open Problem: Investigating SF in the multi-party computation setting.

Thanks!