De Cifris Trends in *Cryptographic Protocols*

University of Trento and *De Componendis Cifris*

October 2023

Lecture 1

Alessandra Scafuro North Carolina State University

Security and Composition of Cryptographic Protocols

Alessandra Scafuro

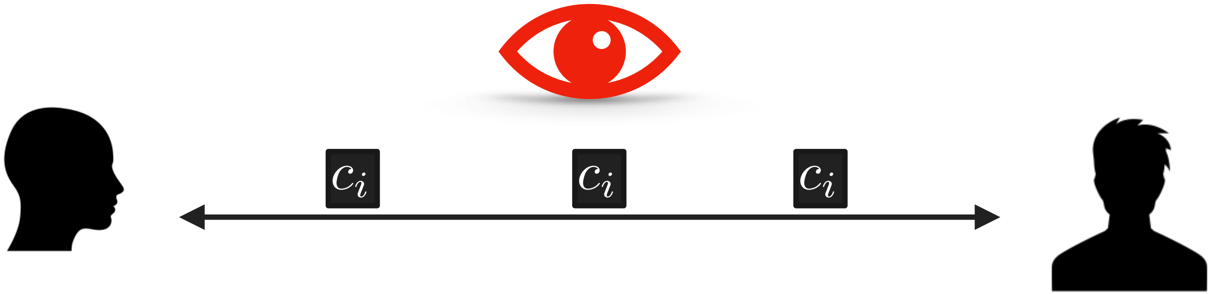
North Carolina State University

NC STATE UNIVERSITY

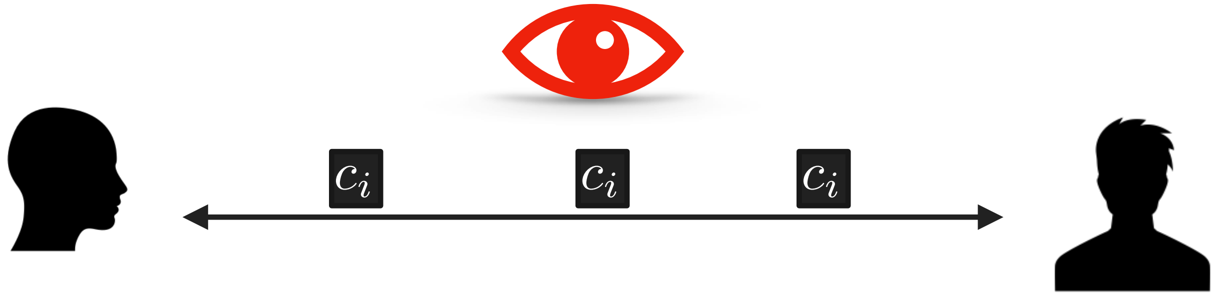
Why formal definitions?

- Formal definitions are necessary to write a formal proof that a scheme achieves the security property we expect.
- Formal proofs are necessary to have rigorous guarantees that a scheme is secure.

Example: security of an encryption scheme



Example: security of an encryption scheme



Classical approach to defining security:

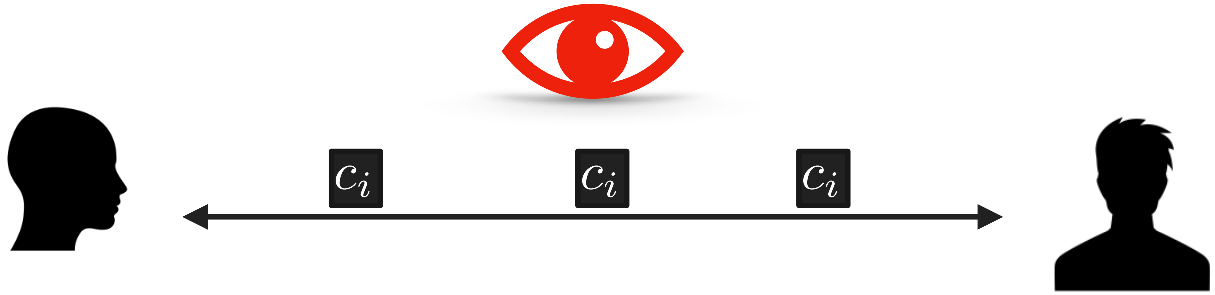
- Intuitively, from the ciphertext, an adversary should learn no information about the plaintext.
    - What does it mean "no information"?

Classical approach to proving security:

- Prove that the scheme withstands all previously known attacks.
    - What if there are other attacks we have not thought of?
      Example: enigma, known plaintext attack.

Alessandra Scafuro **North Carolina State University**

Example: security of an encryption scheme



Classical approach to defining security:

- Intuitively, from the ciphertext, an adversary should learn no information about the plaintext.
    - What does it mean “no information”?

Classical approach to proving security:

- Prove that the scheme withstands all previously known attacks.
    - What if there are other attacks we have not thought of?
      Example: enigma, known plaintext attack.

Modern approach: Formal Definition:

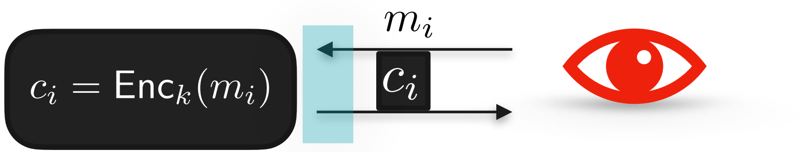
- Formally define what it means **“no information”**

Modern approach: Formal Proof:

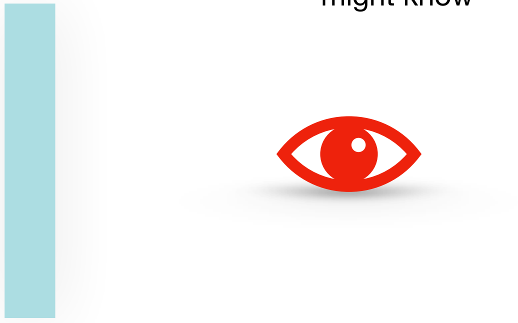
- Prove the scheme stands any attack that could **ever** occur — assuming computational restrictions.

Alessandra Scafuro North Carolina State University

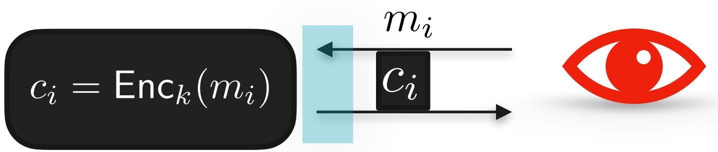
Formal Definition of a Semantically Secure Encryption Scheme



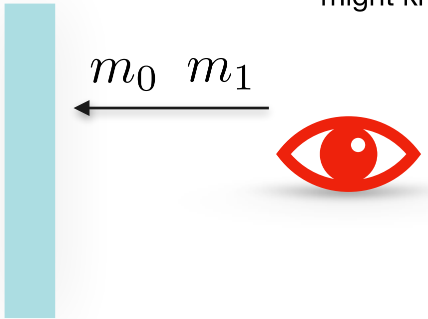
- Capture the adversarial power of **observing traffic** of content they might know



Formal Definition of a Semantically Secure Encryption Scheme

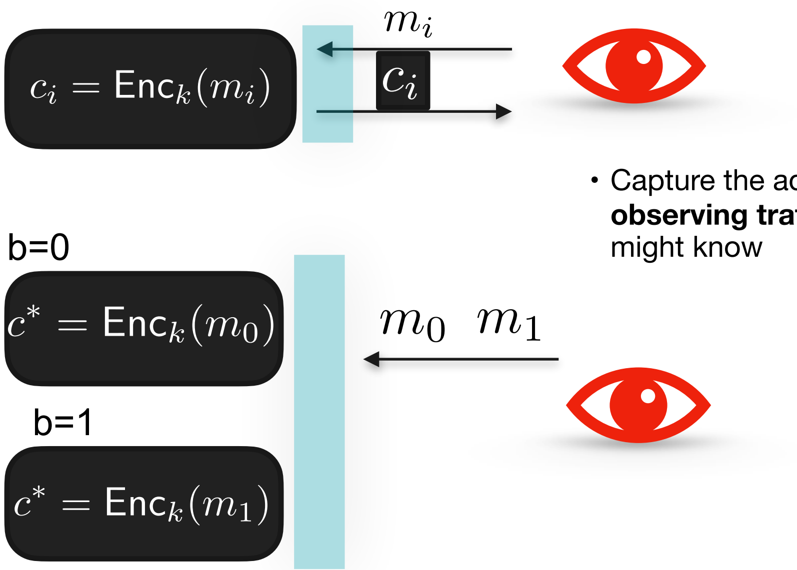


- Capture the adversarial power of **observing traffic** of content they might know



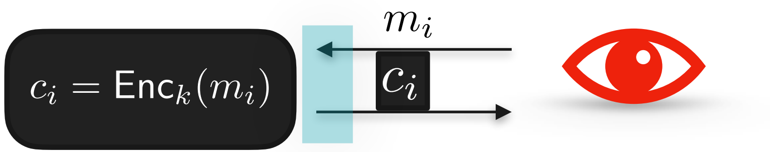
Alessandra Scafuro **North Carolina State University**

Formal Definition of a Semantically Secure Encryption Scheme

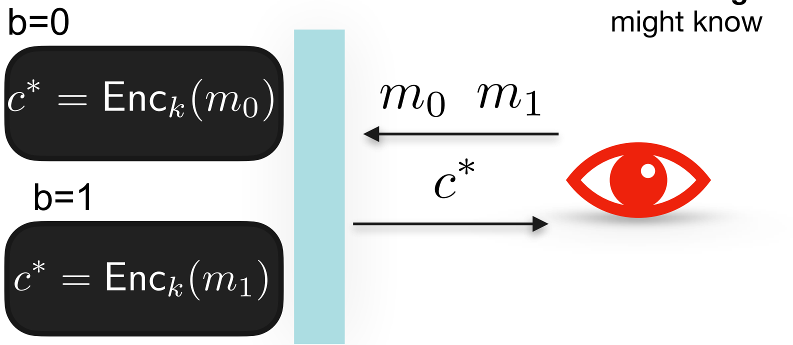


- Capture the adversarial power of **observing traffic** of content they might know

Formal Definition of a Semantically Secure Encryption Scheme

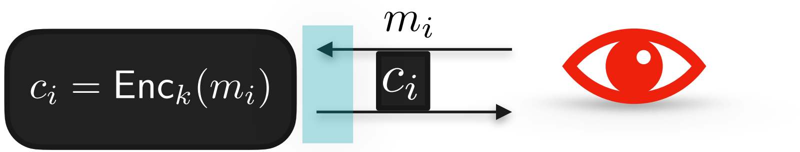


- Capture the adversarial power of **observing traffic** of content they might know

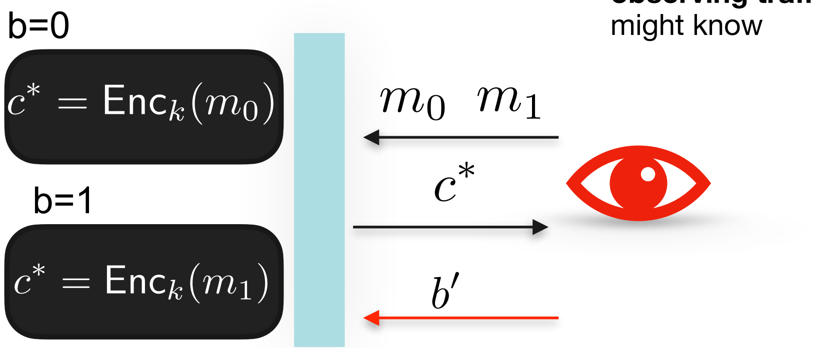


Alessandra Scafuro **North Carolina State University**

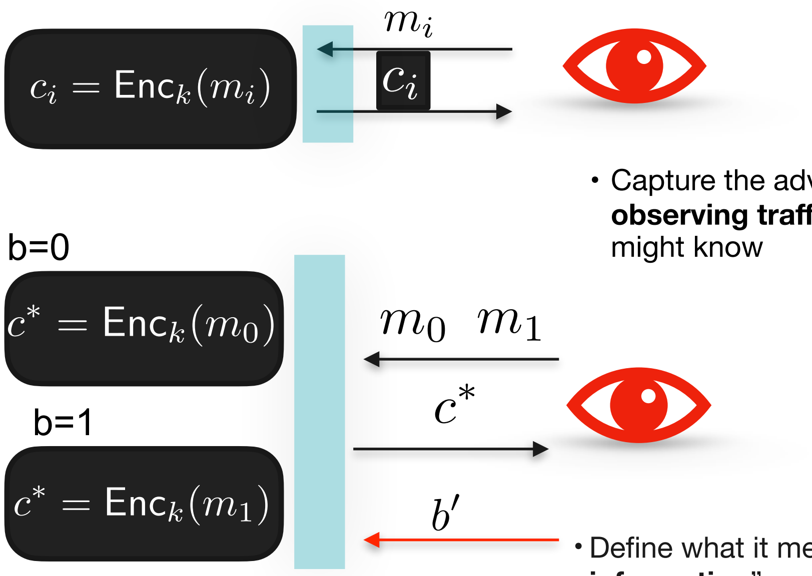
Formal Definition of a Semantically Secure Encryption Scheme



- Capture the adversarial power of **observing traffic** of content they might know



Formal Definition of a Semantically Secure Encryption Scheme

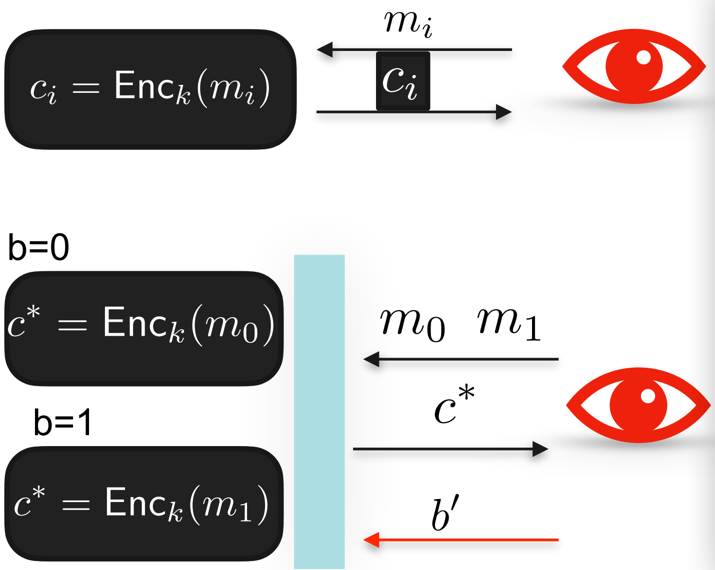


- Capture the adversarial power of **observing traffic** of content they might know

- Define what it means to learn "**no information**": an adversary cannot tell which message is encrypted.

Alessandra Scafuro North Carolina State University

Formal Definition of a Semantically Secure Encryption Scheme



The CPA indistinguishability experiment $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$:

1. *A key k is generated by running* $\mathsf{Gen}(1^n)$.
2. *The adversary $\mathcal{A}$ is given input 1^n and oracle access to* $\mathsf{Enc}_k(\cdot)$, *and outputs a pair of messages m_0, m_1 of the same length.*
3. *A uniform bit $b \in \{0,1\}$ is chosen, and then a ciphertext $c \leftarrow \mathsf{Enc}_k(m_b)$ is computed and given to $\mathcal{A}$.*
4. *The adversary $\mathcal{A}$ continues to have oracle access to* $\mathsf{Enc}_k(\cdot)$, *and outputs a bit b'.*
5. *The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise. In the former case, we say that $\mathcal{A}$* succeeds.

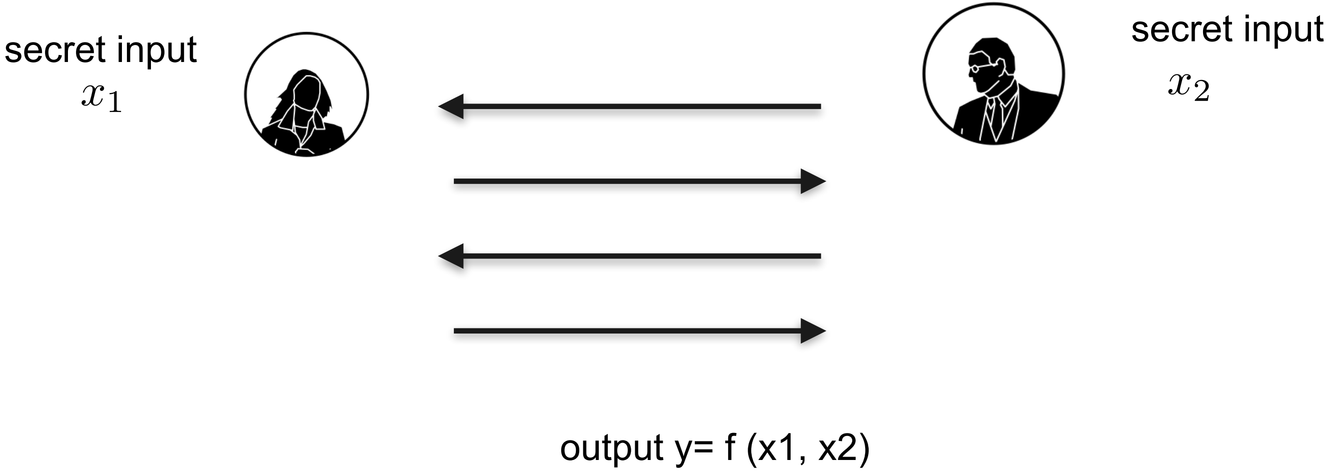
Why formal indistinguishability game facilitates writing rigorous proof of security

DEFINITION 3.22 *A private-key encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *has* indistinguishable encryptions under a chosen-plaintext attack, *or is* CPA-secure, *if for all probabilistic polynomial-time adversaries* $\mathcal{A}$ *there is a negligible function* negl *such that*

$$\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(n),$$

- In the proof, we can make a mathematical connection from an adversary A distinguishing the ciphertexts, to an adversary B breaking a mathematical problem that is believed hard to solve
  - Example: a PPT adversary that distinguishes El-Gamal ciphertexts can be used to build a PPT algorithm that invalidates the hardness of some problems based on Discrete Log of certain groups.

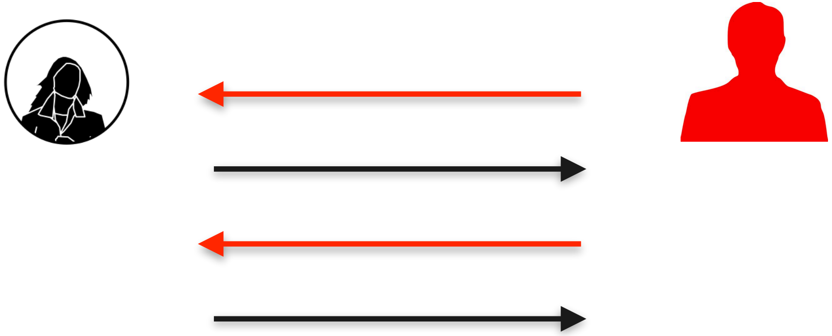
Defining Security of Cryptographic Protocols



Alessandra Scafuro **North Carolina State University**

Defining Security of Cryptographic Protocols

secret input
x_1



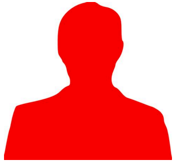
output y

Alessandra Scafuro **North Carolina State University**

Protocols for Proving Knowledge of a Secret

secret input
(witness)
sk: 7DC941A2:

pk 5EC948A1:
“Satoshi Nakamoto <satoshin@gmx.com>"

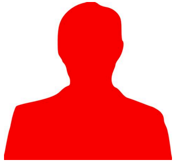


Alessandra Scafuro **North Carolina State University**

Protocols for Proving Knowledge of a Secret

secret input
(witness)
sk: 7DC941A2:

pk 5EC948A1:
“Satoshi Nakamoto <satoshin@gmx.com>"



Security we might want, informally:

- Satoshi should be able to **convince the verifier** of his identity, **without revealing his signing key.**

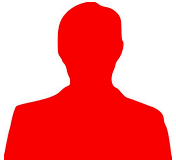
what if the protocol requires the prover to just sign a message?

Alessandra Scafuro **North Carolina State University**

Zero-Knowledge Proofs

secret input
(witness)
sk: 7DC941A2:

pk 5EC948A1:
“Satoshi Nakamoto <satoshin@gmx.com>"

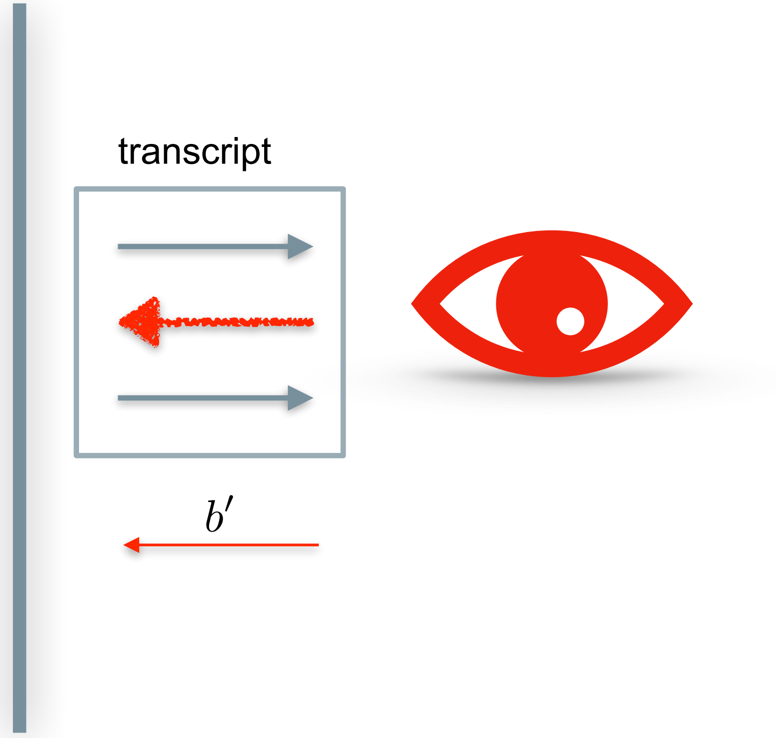


Security **we want**:
no matter what she does, the verifier should learn **nothing besides yes/no.**

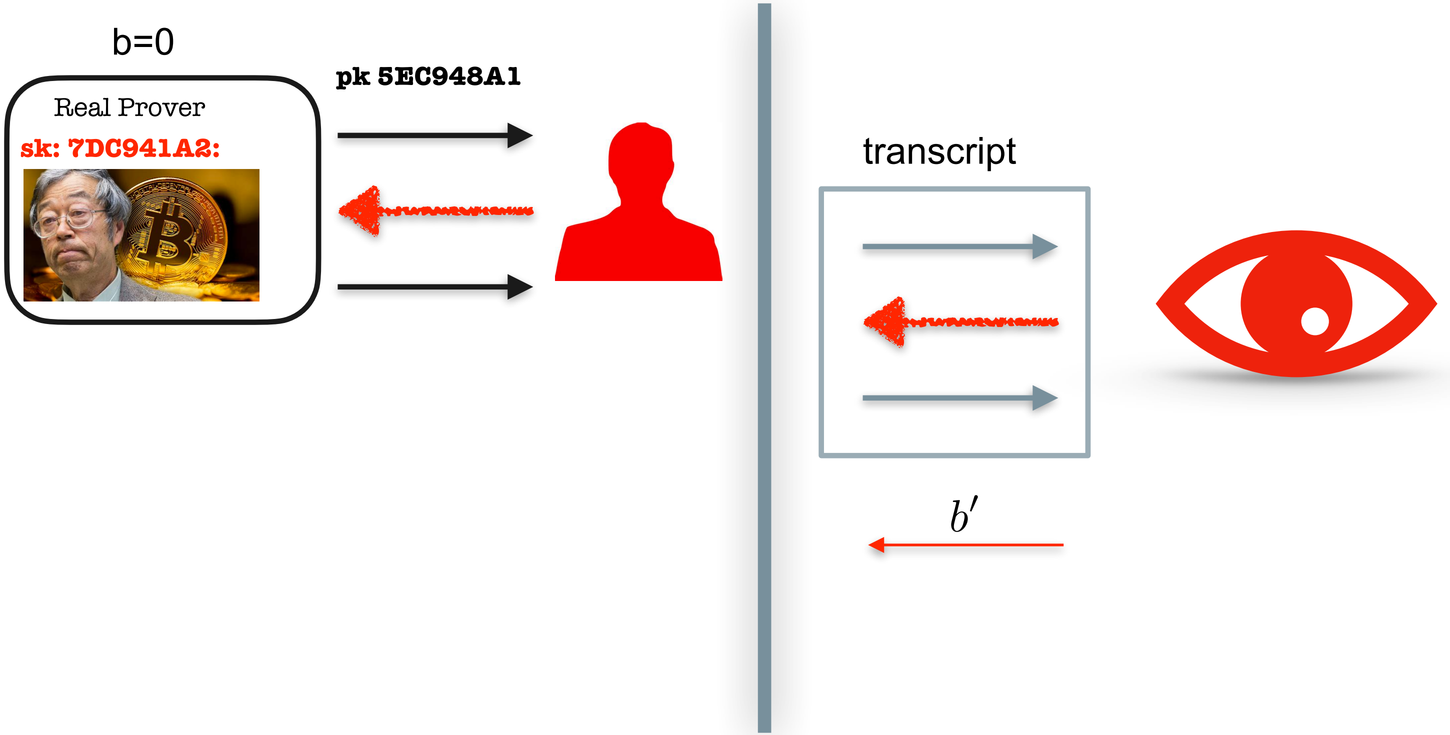
How do we **formally** define, that a protocol leaks **nothing?**

Alessandra Scafuro **North Carolina State University**

Formal Definition of Zero-knowledge Proof System

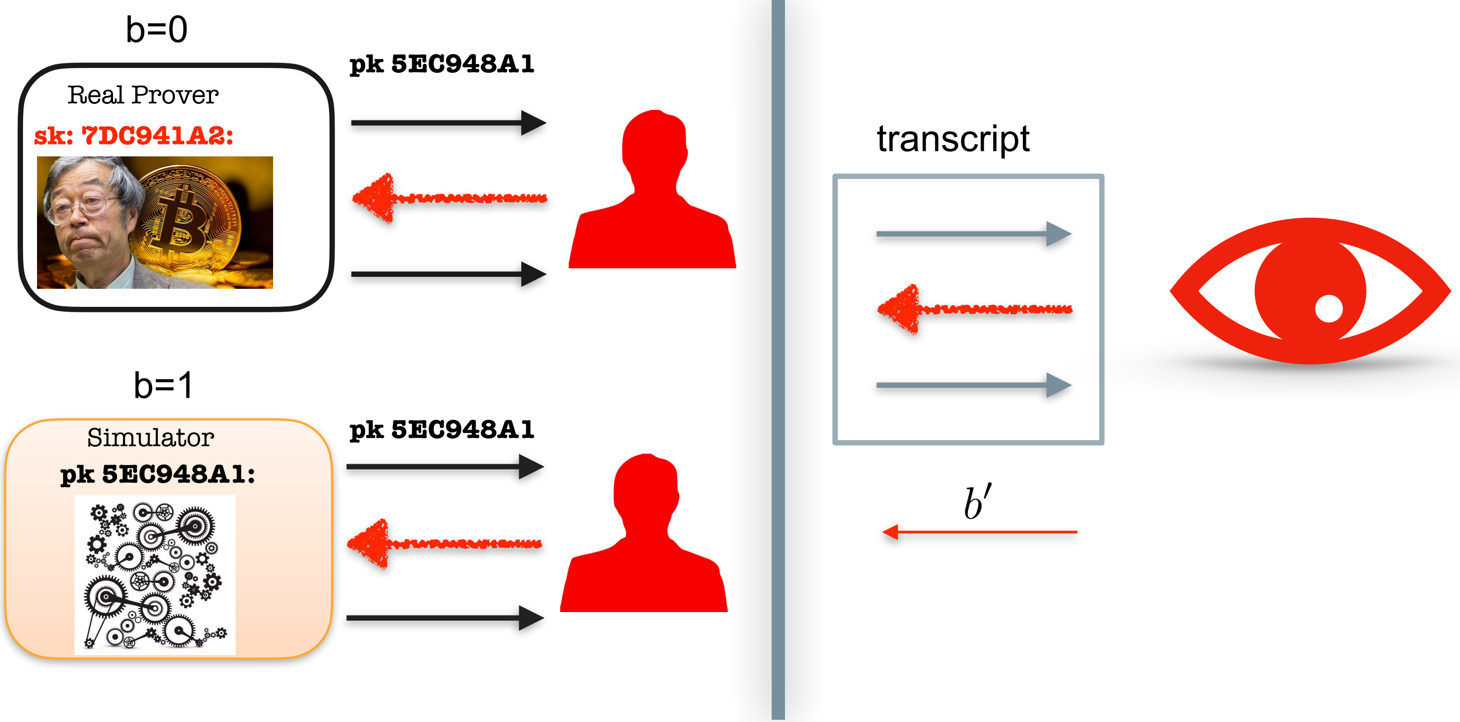


Formal Definition of Zero-knowledge Proof System



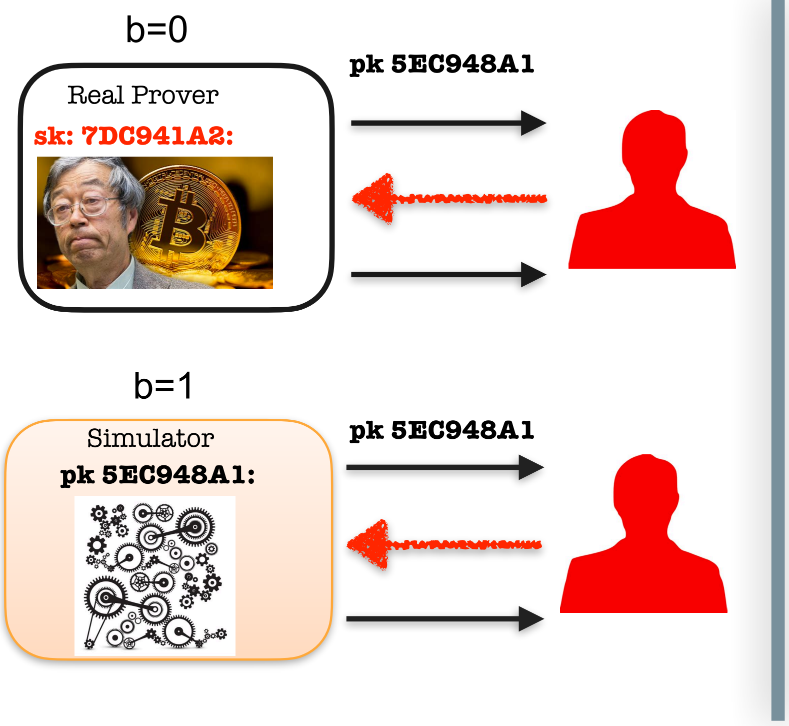
Alessandra Scafuro North Carolina State University

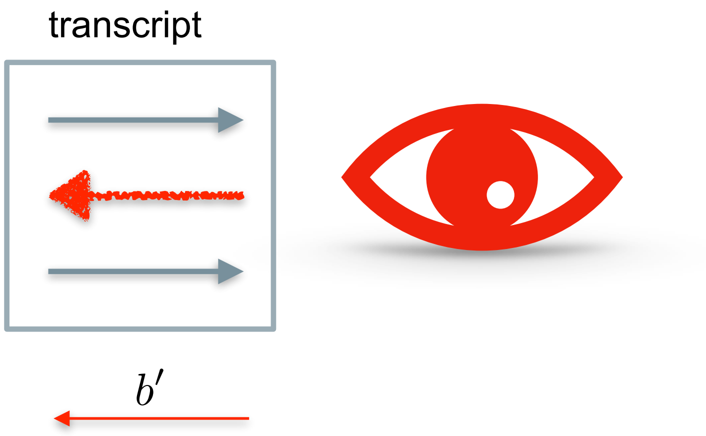
Formal Definition of Zero-knowledge Proof System



Alessandra Scafuro North Carolina State University

Formal Definition of Zero-knowledge Proof System





Why does this definition correctly capture the concept of “zero-knowledge”?

Alessandra Scafuro North Carolina State University

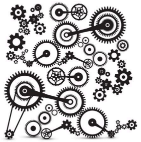
How does this looks like formally….

Definition 10 (Zero Knowledge). *An interactive protocol (P, V) for a language L is* zero knowledge *if for every PPT adversary V^*, there exists a PPT simulator S such that the probability ensembles* $\{\langle P, V^*(z)\rangle(x)\}_{x\in L, z\in\{0,1\}^*}$ *and* $\{S(x,z)\}_{x\in L, z\in\{0,1\}^*}$ *are computationally indistinguishable, where $\langle P, V^*(z)\rangle(x)$ denotes the output of V^* when interacting with P on common input x and auxiliary input z.*

How to formally proof that a protocol is zero-knowledge

Must provide a **simulator** that creates a **“good”** transcript, **without any secret**

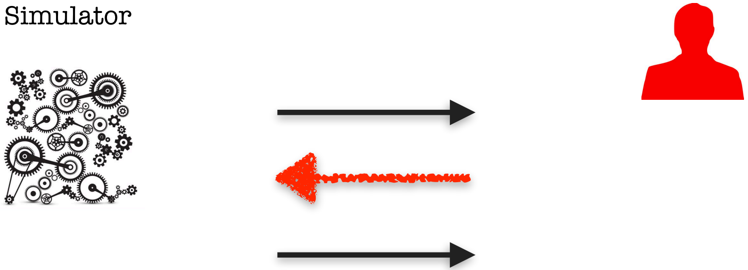
Simulator



How to formally proof that a protocol is zero-knowledge

Must provide a **simulator** that creates a **“good”** transcript, **without any secret**

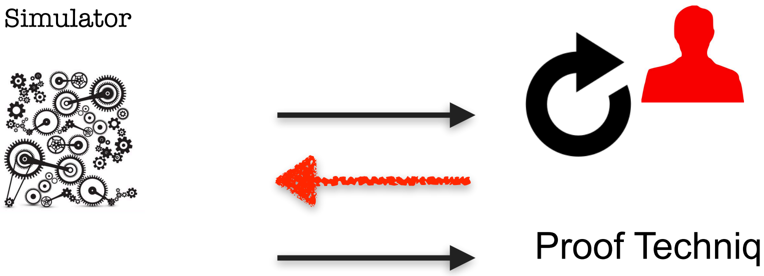
Simulator



Alessandra Scafuro **North Carolina State University**

How to formally proof that a protocol is zero-knowledge

Must provide a **simulator** that creates a **“good”** transcript, **without any secret**

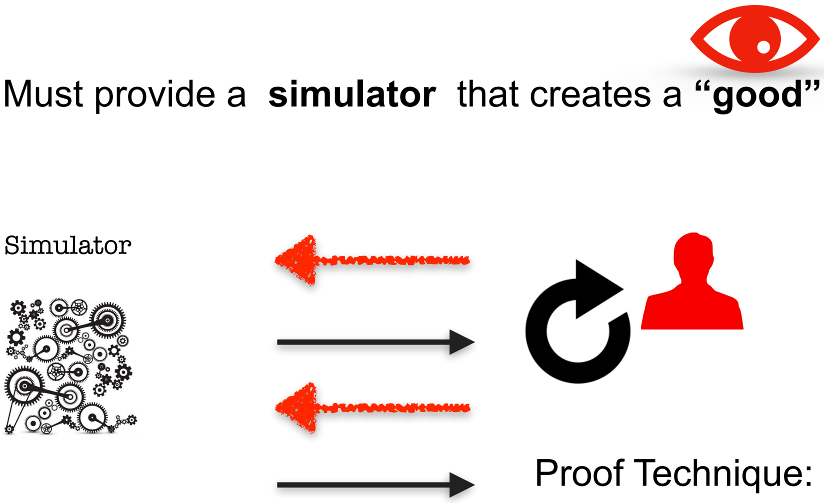


Proof Technique: **Rewinding**

Alessandra Scafuro **North Carolina State University**

How to formally proof that a protocol is zero-knowledge

Must provide a **simulator** that creates a **"good"** transcript, **without any secret**

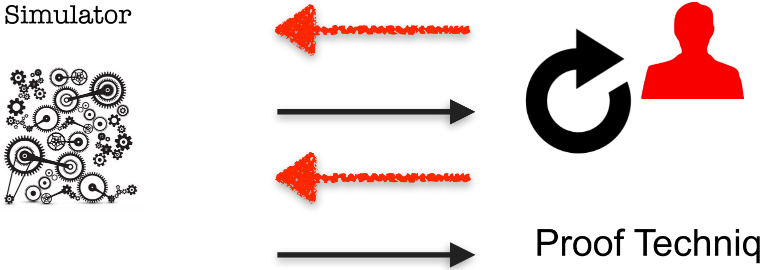


Proof Technique: **Rewinding**

Alessandra Scafuro North Carolina State University

How to formally proof that a protocol is zero-knowledge

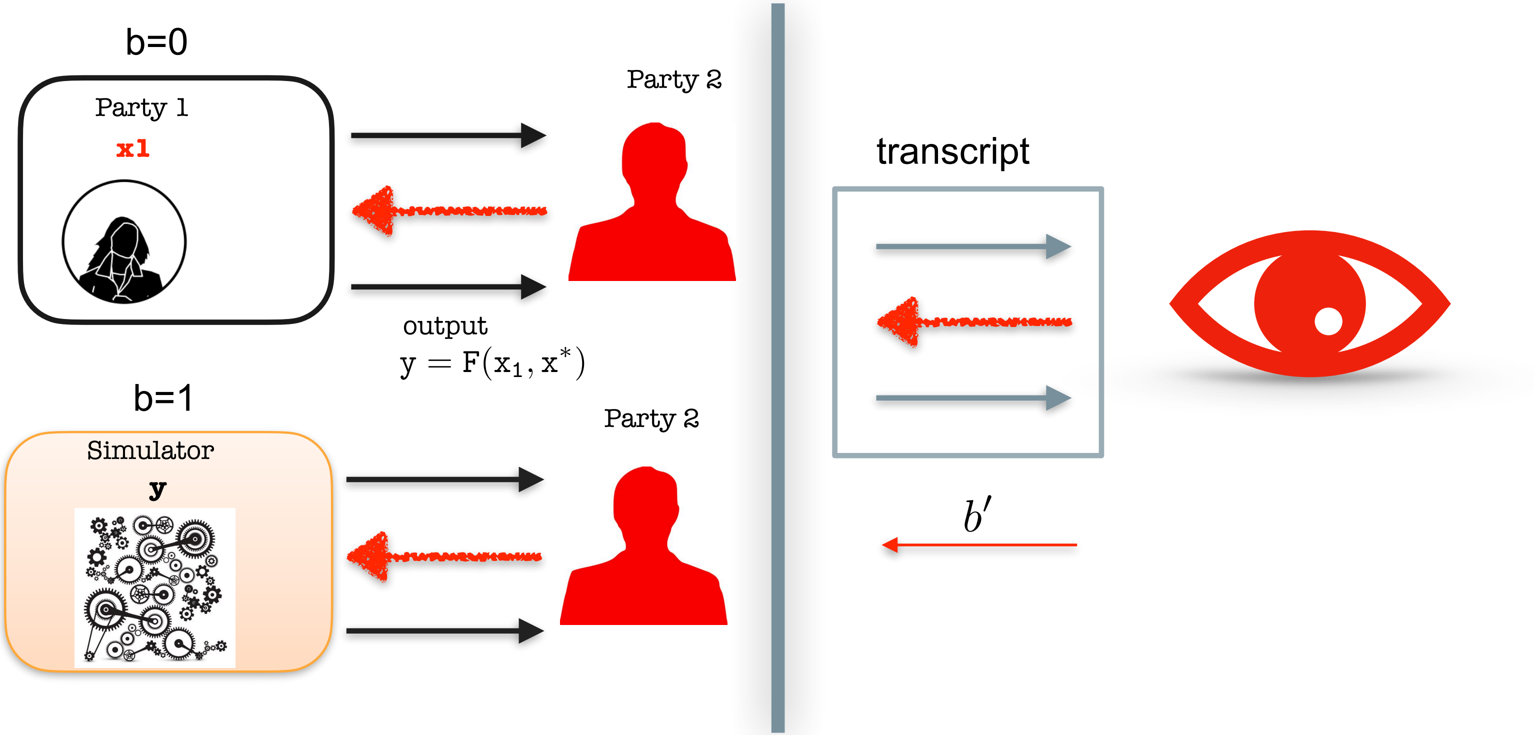
Must provide a **simulator** that creates a **“good”** transcript, **without any secret**



Protocol complexity increases: one more round is required

Proof Technique: **Rewinding**

Formal Definition of Protocols for General Functions (not just proving)



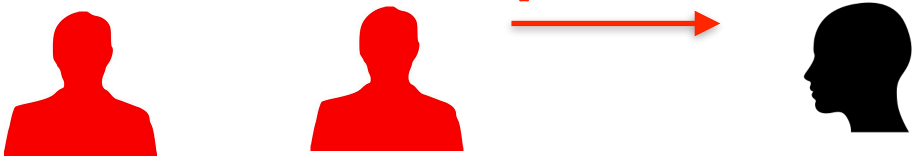
Alessandra Scafuro North Carolina State University

What happens when two secure protocols are executed in parallel?

Nakamoto

sk: 7DC941A2:

pk 5EC948A1

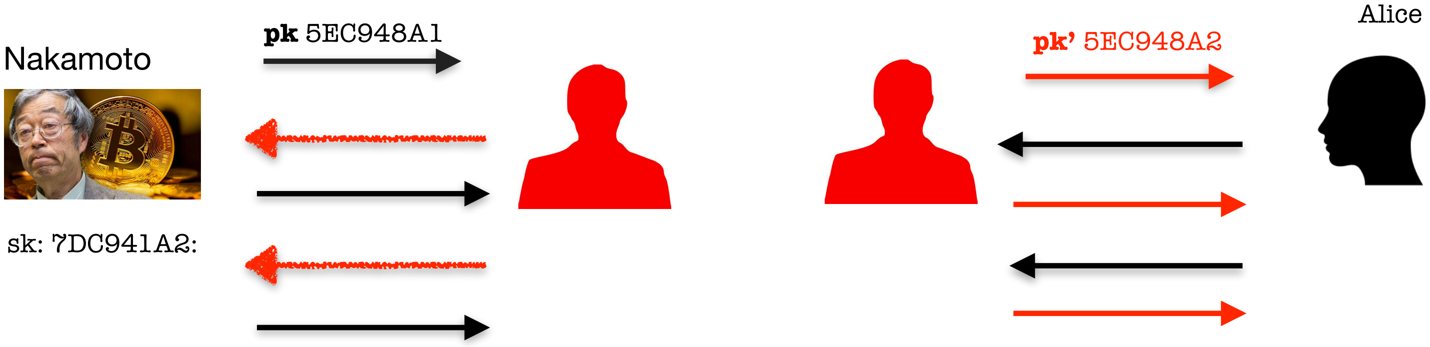


pk' 5EC948A2

Alice

Alessandra Scafuro North Carolina State University

What happens when two secure protocols are executed in parallel?



Alessandra Scafuro North Carolina State University

What happens when two secure protocols are executed in parallel?

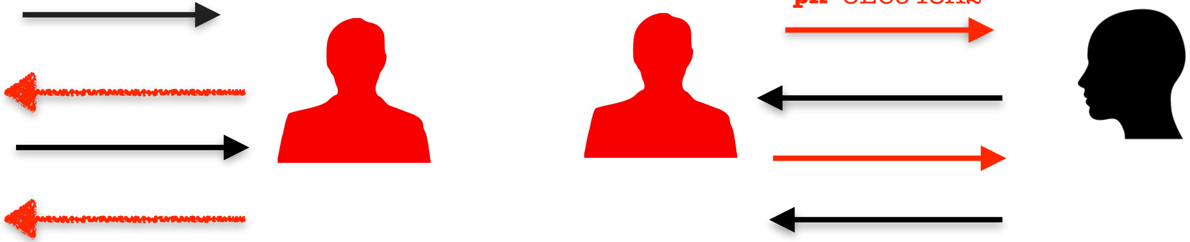
Nakamoto

sk: 7DC941A2:

pk 5EC948A1

Alice

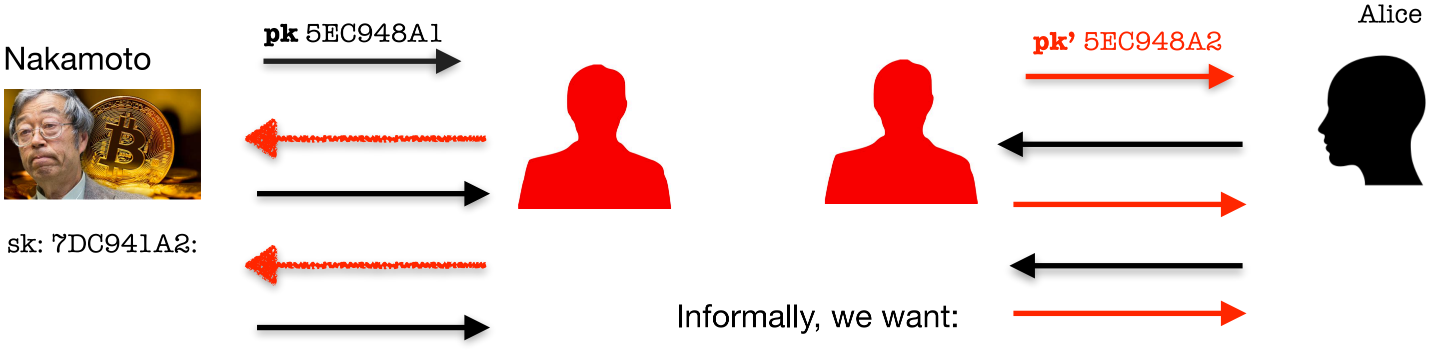
pk’ 5EC948A2



Informally, we want:

by talking to Nakamoto who is proving something about pk, an adversary **should not gain any advantage** in proving something about a related theorem about pk’, to another person.

What happens when two secure protocols are executed in parallel?



Informally, we want:

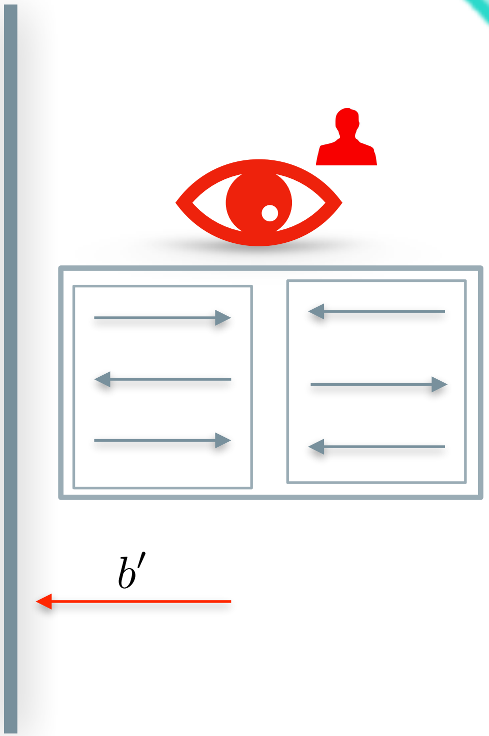
by talking to Nakamoto who is proving something about pk, an adversary **should not gain any advantage** in proving something about a related theorem about pk', to another person.

Is this already covered by our ZK definition?

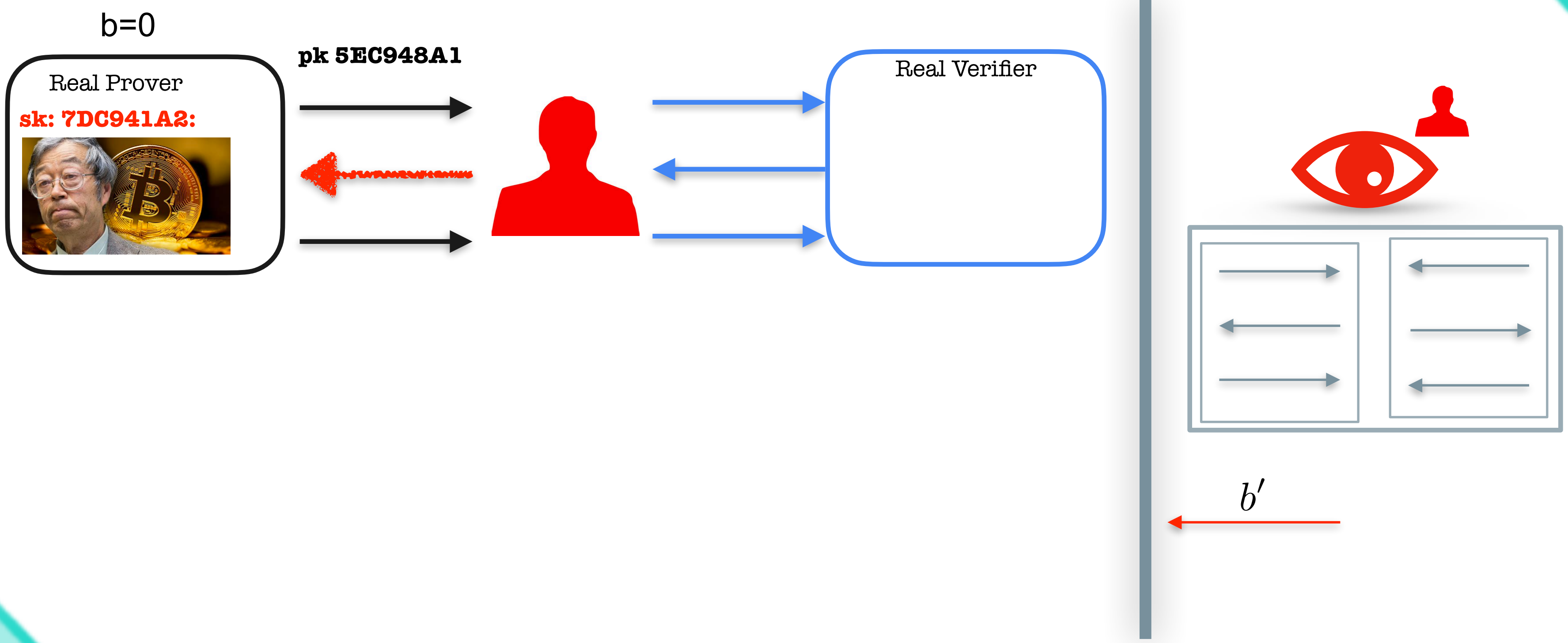
Not really: if the adversary convinces another verifier does not mean that it learns something…

Alessandra Scafuro **North Carolina State University**

Non-malleable zero-knowledge

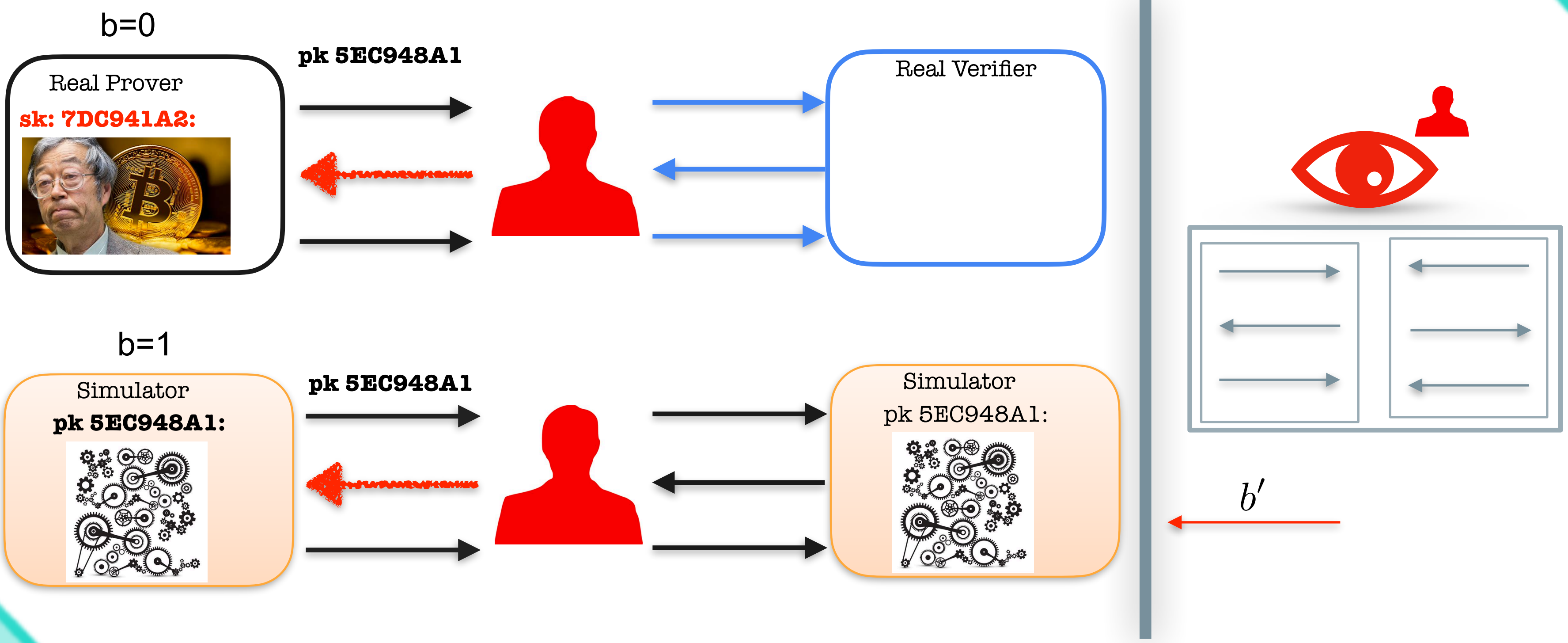


Non-malleable zero-knowledge



Alessandra Scafuro **North Carolina State University**

Non-malleable zero-knowledge



Alessandra Scafuro North Carolina State University

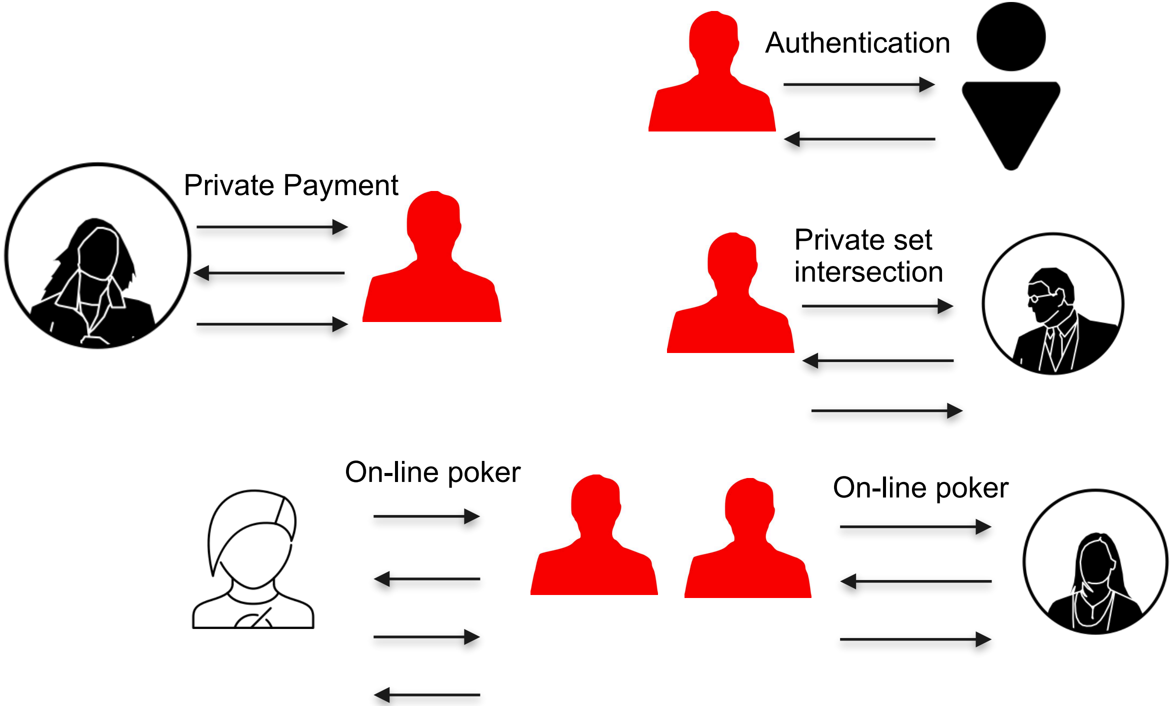
Observations

- Only two executions of the same protocol, made the definition more complex.

- Proving that a protocol achieves this definition is often a complex task. And we are only talking about two parallel executions of the **SAME** PROTOCOL.

- What if we had **many** executions of **arbitrary protocols** in arbitrary order??

Arbitrary Execution of Arbitrary Protocols: General Concurrent Composition



Alessandra Scafuro **North Carolina State University**

Challenges of the concurrent setting

Formally defining such a setting must consider that:

- Inputs of honest parties could be chosen adaptively on the transcripts of previous/ concurrent protocol (e.g., bidding and payment protocol)

- There are many functions computed among many parties, which the definition should be aware of.

In the proof of security:

- The simulator would need to be aware of all the parties and simulate them accordingly.

The Universal Composability Model [C,PW]

A framework to prove security that guarantees that, if your protocol is proved secure in this framework, then the protocol can safely run in an arbitrary environment.

[C] Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS 2001). pp. 136–145. IEEE Computer Society (2001)

[PW] Pfitzmann, B., Waidner, M.: A Model for Asynchronous Reactive Systems and its Application to Secure Message Transmission. In: IEEE Symposium on Security and Privacy, 2001

The Universal Composability Model [C,PW]

A framework to prove security that guarantees that, if your protocol is proved secure in this framework, then the protocol can safely run in an arbitrary environment.

Founding Principle:
Proving security for a protocol computing a function F, should be independent of any other protocol and party existing in the world.

[C] Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS 2001). pp. 136–145. IEEE Computer Society (2001)
[PW] Pfitzmann, B., Waidner, M.: A Model for Asynchronous Reactive Systems and its Application to Secure Message Transmission. In: IEEE Symposium on Security and Privacy, 2001

The Universal Composability Framework

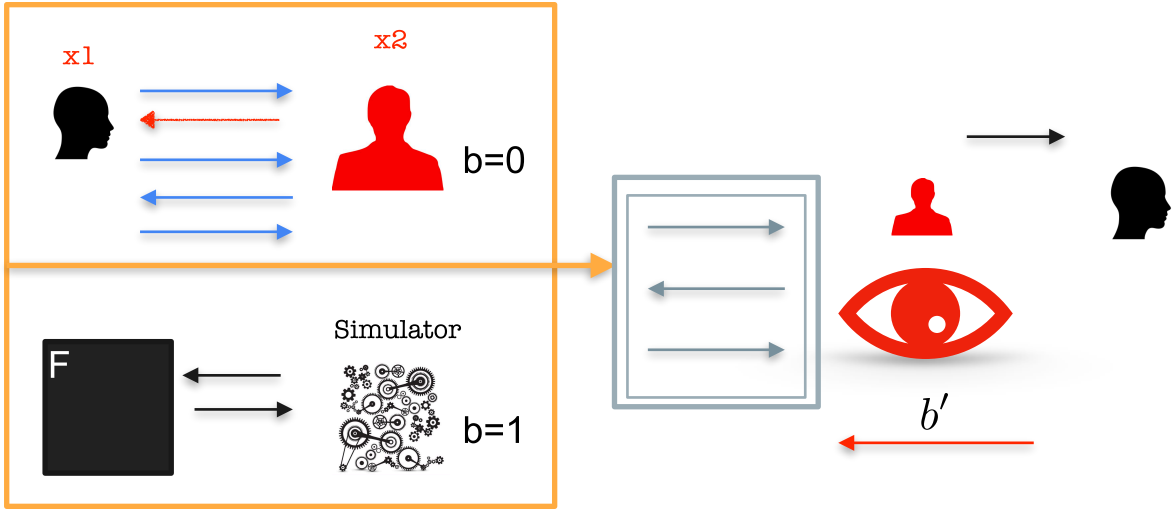
The environment: The concurrent protocols are captured by a the concept of an ``environment''. The environment decides the inputs of all the honest parties, and the order of the execution

The ideal functionality: The security requirements for a certain function are captured by the concept of an *ideal* functionality.

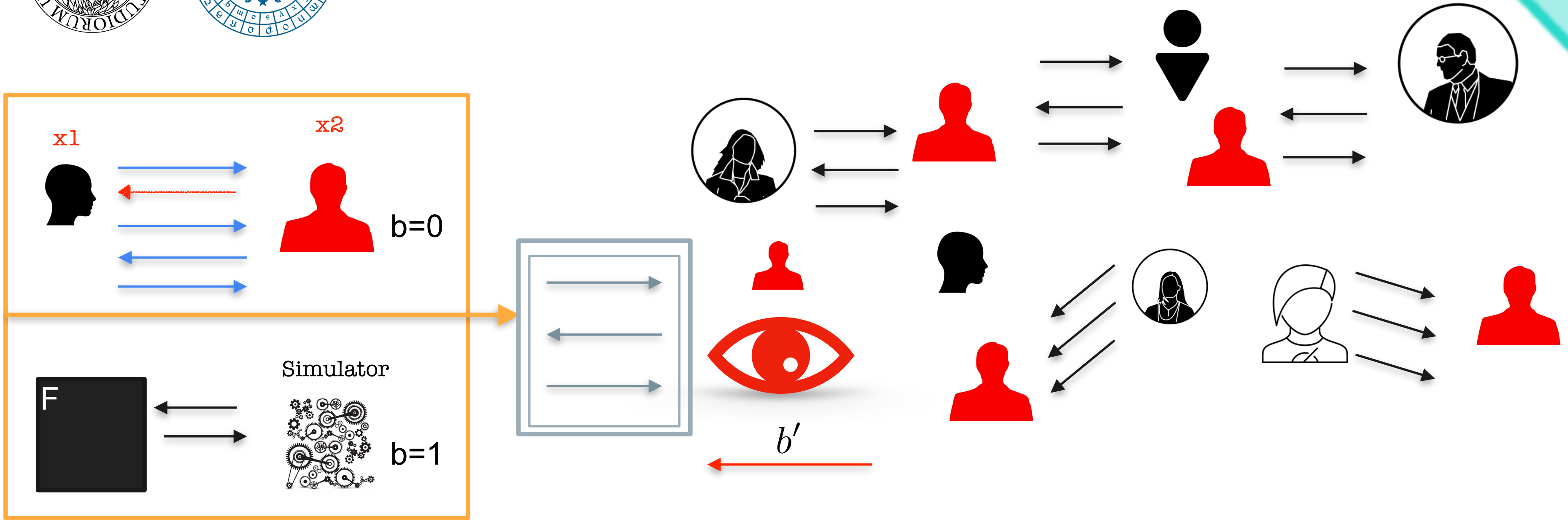
The simulator: it only exists in this ideal functionality, and is **not aware** of any other execution.

Security proof: to prove that a protocol securely realizes an ideal functionality, it means to show such an ``agnostic'' simulator that is able to compute a distinguishable transcript

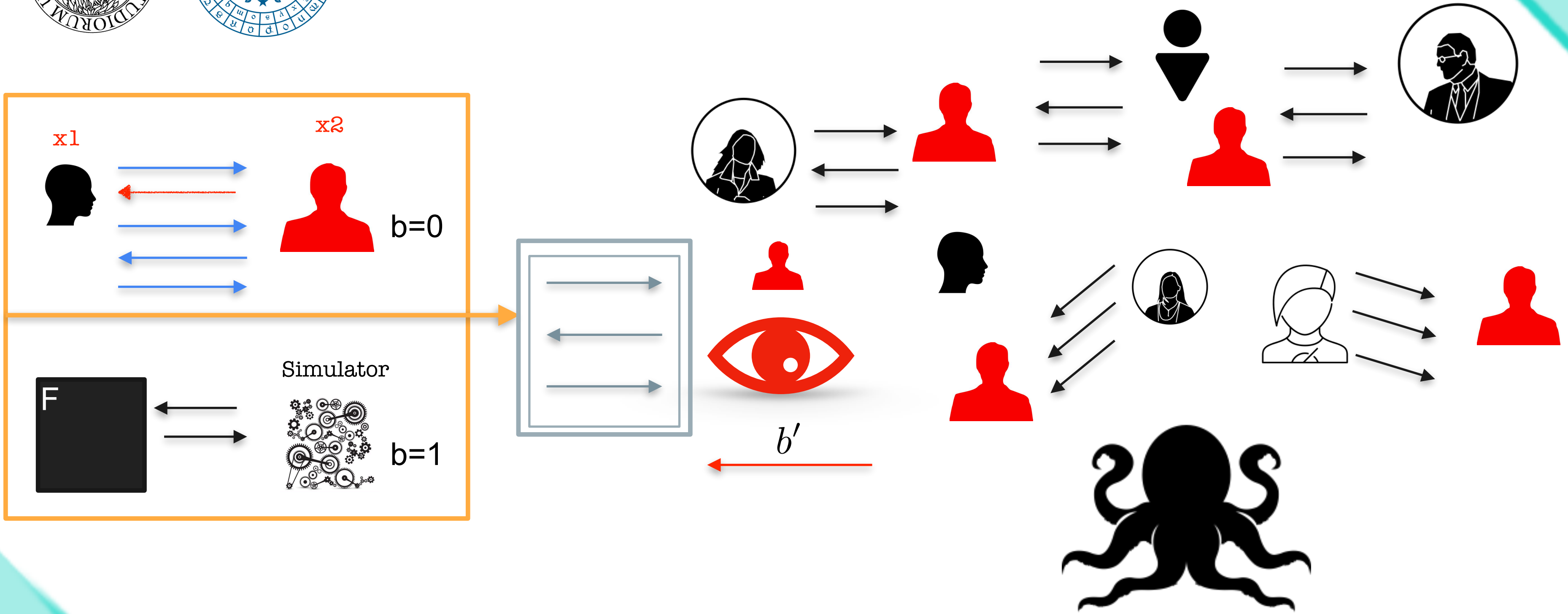
A possible visualization



A possible visualization



A possible visualization



Example: ideal functionality for non-interactive zero-knowledge

1. The ideal functionality must capture the correctness and security properties that we want from a real protocol.

Functionality $\mathcal{F}_{\mathrm{NIZK}}^{R}$

$\mathcal{F}_{\mathrm{NIZK}}$ is parametrized by a relation R for which we can efficiently check membership. It keeps an initially empty list L of proven statements.

1. On input $(\mathtt{prove}, y, w)$ from a party P, such that $(y, w) \in R$,[a] send $(\mathtt{prove}, y)$ to $\mathcal{A}$.
2. Upon receiving a message $(\mathtt{done}, \psi)$ from $\mathcal{A}$, with $\psi \in \{0,1\}^*$, record (y, ψ) in L and send $(\mathtt{done}, \psi)$ to P.
3. Upon receiving $(\mathtt{verify}, y, \psi)$ from some party P', check whether $(y, \psi) \in L$. If not, output $(\mathtt{verify}, y, \psi)$ to $\mathcal{A}$ and upon receiving answer $\mathtt{witness}, w$. Check $(y, w) \in R$ and if so, store (y, ψ) in L. If (y, ψ) has been stored, then output 1 to P', else output 0.

[a] Inputs that do not satisfy the respective relation are ignored.

Observations for the UC-model

Pros

The security proof only focuses on one protocol executed in isolation

Ideal functionality helps capturing the security property of complex tasks

Cons

Additional, strong setup assumptions are required. Example, trusted CRS.

Conclusion

- Formal definitions are necessary for providing provable security guarantees.

- Formally defining security for complex tasks in complex environment is challenging.

- The Universally Composable Model provide a framework to express (and prove) such security requirements.

De Componendis Cifris

https:/www.decifris.it

Alessandra Scafuro **North Carolina State University**