The evolution of Challenge Response protocols.

Alessandro Buldini

alessandro.buldini@unibo.it

TODO:

Today we will explore two arguments:

- **WiFi**:
  - Introduction.
  - WiFi security.
  - WPA Personal vs Enterprise.
  - WPA2-PSK.
  - Good passwords.
- **Zero-Knowledge Proofs**:
  - Hard problems.
  - Ali Baba's Cave.
  - Zero-Knowledge Proof Properties.
  - Statistical vs Computational.
  - A concrete example.

SHA - Secure Hashing Algorithm Recap

Under the ROM assumption, the hashing function SHA256 acts as a **random oracle**, meaning that it outputs **256 random bits**.

Moreover, SHA256 is a **unidirectional function:**

- Having an input, it's extremely easy to compute the hash.
- Having the hash, it's extremely difficult to compute the input.

As of today, the best way to retrieve the input from a digest is to perform **brute force attacks** or, in some cases, **dictionary attacks**.

WiFi

- Introduced in September 1997, Wi-Fi is a trademark of the Wi-Fi alliance, a non-profit organization that permits the usage of the Wi-Fi **certification** to those devices that successfully allow device **interoperability**.

- It is based on a family of protocols, namely the **IEEE 802.11 protocols**, and provides network connectivity to devices without the need of wires.

- Devices compliant with IEEE 802.11 protocols but not interoperable enough are simply referred as **WLAN**, Wireless Local Area Network.

802.11 protocols

- Each component of the family (802.11a, 802.11b, 802.11g, 802.11n, 802.11ac, …) differs from each other either in transfer speed or radio frequency. Each variation has a high degree of retro-compatiblity.

- For example, the most widespread version as of today is still **802.11ac**, namely **Wi-Fi 5**, has a maximum speed of ~6900 Mbit/s and leverages the 2.4, 5, and 6 Ghz radio frequencies.

- More recent evolutions, like **Wi-Fi 8**, use the same radio frequencies, but offer higher transmission rates up to the 100'000 MBit/s mark.

Typical Interaction

In a typical scenario, the Wi-Fi network has two architectural components:

- One or multiple **Stations (STA)**: a wireless endpoint device. Typical examples are laptops, phones, and other electronical devices compatible with the IEEE 802.11 standard.
- The **Access Point (AP)**: the component that connects stations to the wired network.

Wi-Fi lower layers

- Wi-Fi covers both the **Data Link** and the Physical layers of the ISO Open Systems Interconnection stack.

- Just like for general devices in every other protocol in the Data Link layer, each **station** in the Wi-Fi network is identified by a unique **MAC** (medium access control) address.

- On the other the **Access Point** is identified uniquely by a **Basic Service Set ID, BSSID** for short. The BSSID is strictly derived from the Access Point's MAC address and keeps the same format.

MAC address and BSSID are expressed as follows: `01:23:45:67:89:ab`

Wi-Fi Protected Access

Wi-Fi Protected Access (WPA) is the interoperable implementation of the IEEE 802.11 security standard as certified by the Wi-Fi alliance.

Over the years three main versions have arisen, each providing stronger guarantees than the previous one: **WPA**, **WPA2**, and **WPA3**.

Each protocol was split in several phases: *discovery*, ***authentication***, ***key generation and distribution***, ***protected data transfer***, *connection termination*.

WPA – Protected Data Transfer

To encrypt traffic and authenticate traffic, **WPA** used to leverage **RC4** algorithm, an extremely weak encryption algorithm if compared to modern solutions.

WPA2 is the first Wi-Fi security standard that leverages a properly standardized Authenticated Encryption algorithm. Specifically, WPA2 leverages the **Advanced Ecnryption Standard** (**AES**) in Counter with CMAC mode (**CCM**) using 128-bit long blocks.

Finally, **WPA3** offers a better security than **AES-CCM** by growing the block size to **256**-bits and using the Galois-Counter Mode. **AES-GCM-256** is nowadays possibly one of the strongest and most robust symmetric block ciphers available.

Wi-Fi Protected Access (2)

Each of the three versions of the WPA protocol is dual:

- **WPA*X*-Enterprise**: variant of the protocol for large organizations, companies, or universities where **control** and **accountability** need to be enforced on a singolar user basis.

- **WPA*X*-Personal**: variant of the protocol for personal usage like home networks, small offices, or environments where enforcing strong security guarantees is not required.

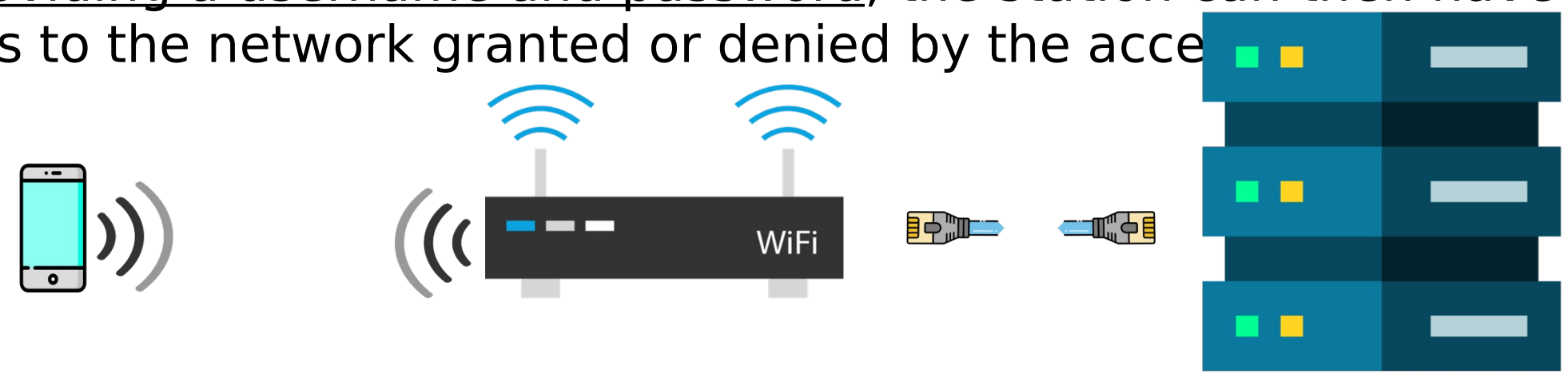
From now on, after a brief presentation of the authentication in the Enterprise variant, we will focus on the authentication of WPA2-Personal.

WPA*X*-Enterprise

In this solution, during the authentication phase, the Access Point acts as an intermediary between the Station and an **Authentication Server** (AS).

Specifically, a secure connection is created between the Authentication Server and the Station, by leveraging a previously exchanged **certificate** that proves AS's identity.

By providing a username and password, the station can then have its access to the network granted or denied by the acce



WPA*X*-Personal

WPA-Personal, on the other hand, cannot rely on an Authentication Server.

All the required steps for authenticating a Station must be performed between the Station itself, and the Access Point.

WPA1 and WPA2 leverage **PSK**, a protocol based on **Pre-Shared Keys**. Given the weaknesses of this protocol with respect to dictionary attacks, **WPA3** now uses **SAE**, **Simultaneous Authentication of Equals**.

WPA2-PSK

We are now interested in understanding why **Pre-Shared Keys**, the WPA2 authentication protocol, is flawed.

PSK is a challenge-response protocol that requires the usage of a **password**.

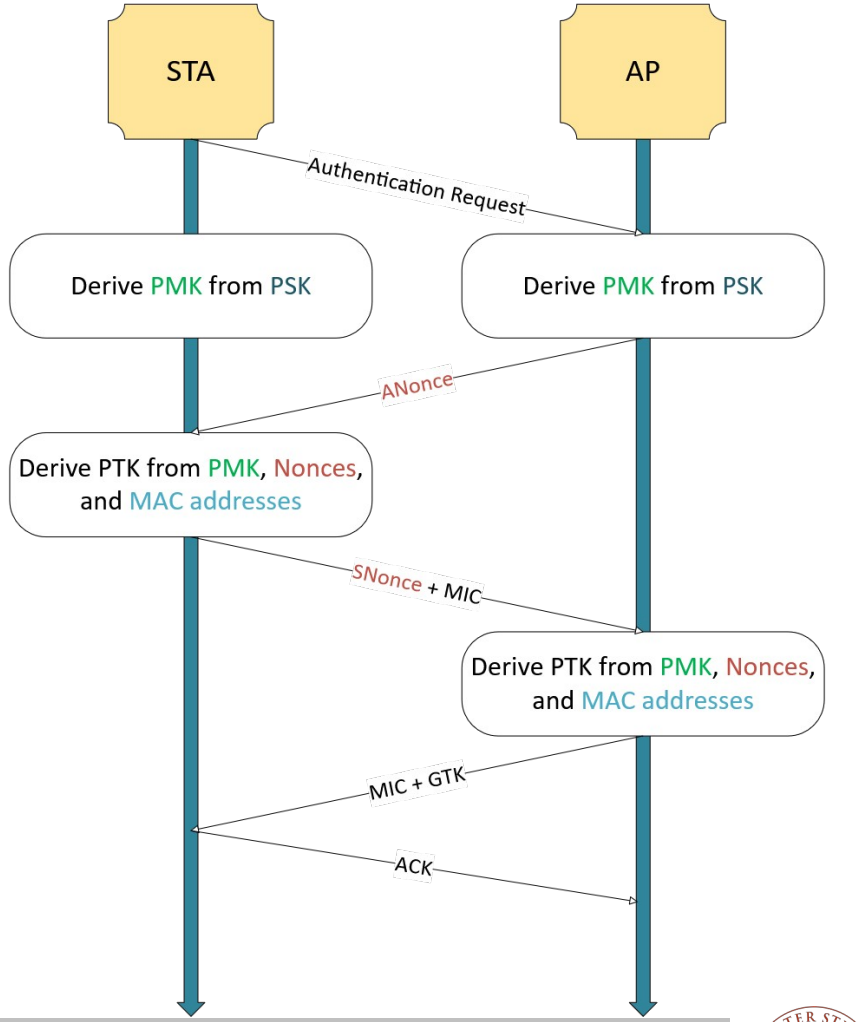
By leveraging a **Key Derivation Function**, both the STA and the AP are able to authenticate each others, and to derive a key to encrypt the traffic with.

The keys to encrypt and authenticate traffic are computed via a 4-way handshake.

4-Way handshake: preparation

Both AP and STA use the **Password Based Key Derivation Function 2** (PBKDF2) algorithm to produce a **Pairwise Master Key** (PMK) from the secret password (PSK). This key is never transmitted over the air.

Internally, PBKDF2 uses an HMAC algorithm to convert a string and a salt into a variable-length key.
In the specific case of WPA2, PBKDF2 uses 4096 rounds of HMAC-SHA1 to produce a 256-bit PMK.



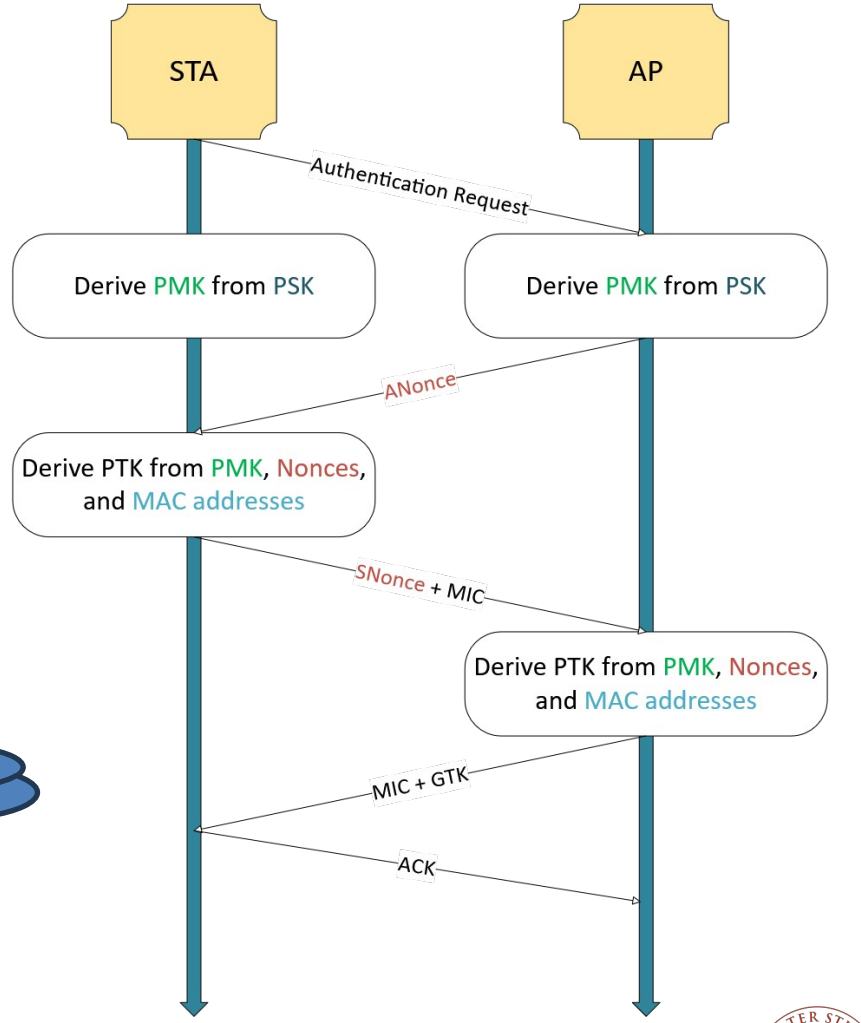

```
PBKDF2_HMAC_SHA1(string, salt, iterations, output_length)
PMK=PBKDF2_HMAC_SHA1(PSK, network_name, 4096, 256)
```

4-Way handshake: nonces exchange and PTK calculation

1. To avoid replay attacks, the AP sends a nonce ANonce to the station.

2. STA now generates another nonce, SNonce.
   Internally, it uses the previously computed PMK, SNonce, the ANonce, its MAC address, and the AP's MAC address to compute a **512-bit long Pairwise Tr** (**PTK**) using a pseudo rando (similar to PBKDF2).

   Finally, STA sends the SNonce to the AP, together with a **Message Integrity Check**



Why is it *"safe"* to send the MIC unencrypted?

Computing the PTK

802.11i-PRF(*K*, *A*, *B*, *Len*)

$R \leftarrow$ ""

for $i \leftarrow 0$ **to** $((Len+159)/160) - 1)$ **do**

$R \leftarrow R \,||\, \text{HMAC-SHA1}(K, A \,||\, B \,||\, i)$

return Truncate-to-len(R, Len)

Example for AES-CCMP:

PTK ← 802.11i-PRF(PMK, "Pairwise key expansion", min(AP-Addr, STA-Addr) ||
max(AP-Addr, STA-Addr) || min(ANonce, SNonce) || max(ANonce, SNonce),
384)

Initially 384 bits, now 512 for enhanced security.

Slide 47 of the IEEE 802.11i Overview. All the slides are available here.

PTK

The **PTK** is a **512-bit long** key that is used to create other keys. Specifically:

- Bits [0;127] constitute the **KCK, Key Confirmation Key**: a key whose role is uniquely to be used in the MICs exchanged in the 4-way handshake.

- Bits [128;255] constitute the **KEK, Key Encryption Key**: a key solely used for encrypting another key, the **Group Temporal Key**, used for multicast and broadcast for devices in the network.

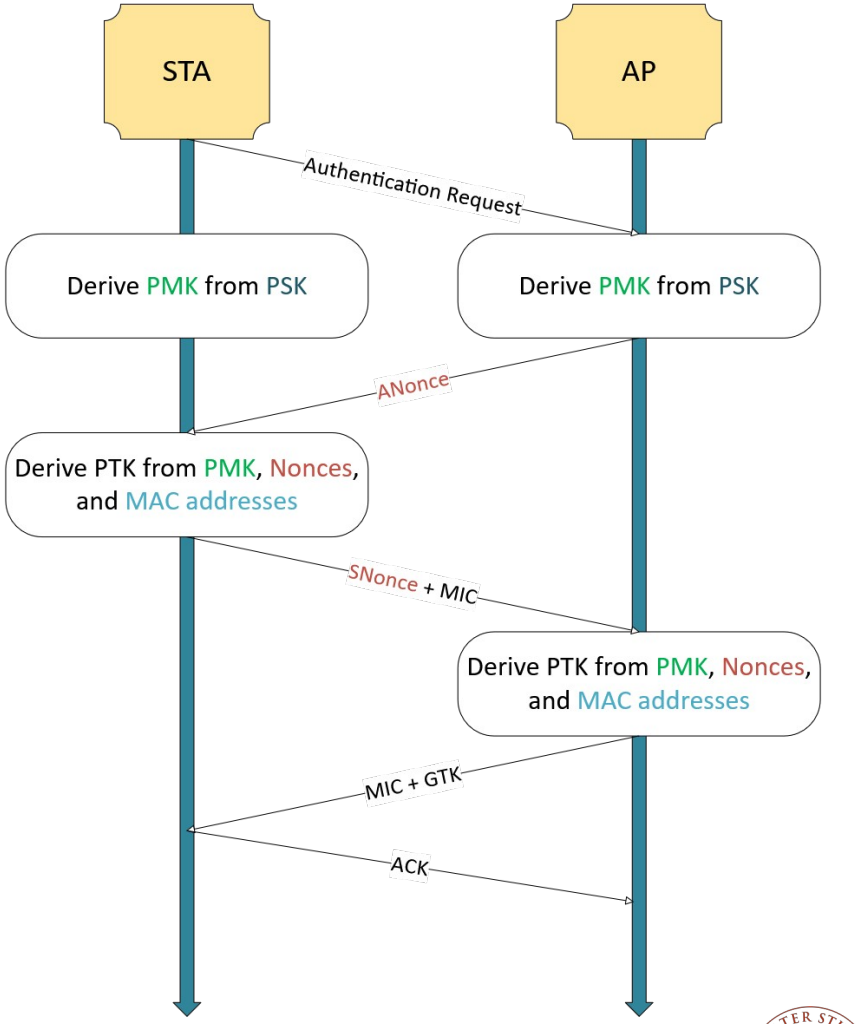
- Bits [256;511] constitute the **TK, Temporal Key**: the symmetric key used to encrypt data in transit between AP and

4-Way handshake: AP PTK calculation

3. Having obtained the SNonce, the AP is now able to compute the PTK itself, using the same exact argument as STA did. It also internally computes its own version of the MIC, that it leverages to know whether the STA knows the password or not:

   If the password used by STA is incorrect:

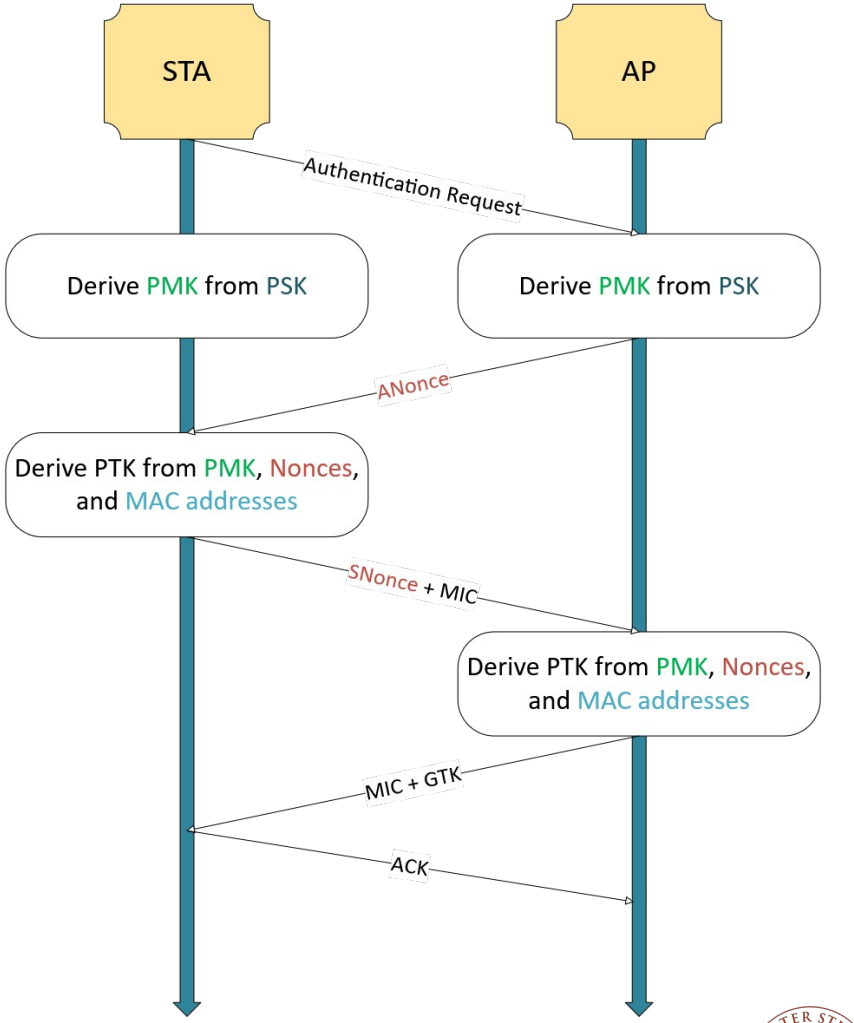
   - The **PMK** computed by STA won't be correct.
   - The **PTK** computed by STA won't be correct.
   - The **MIC** provided by STA will be



4-Way handshake: GTK dispatch and termination

3. (still the same step)
   Once AP establishes that STA knows the password, it will provide its own **MIC** and dispatch the GTK (encrypted using the KEK) to STA.

4. Finally, STA replies with an acknowledgement message to terminate the handshake. It has now every element needed to participate in the Wireless Local Area Network.

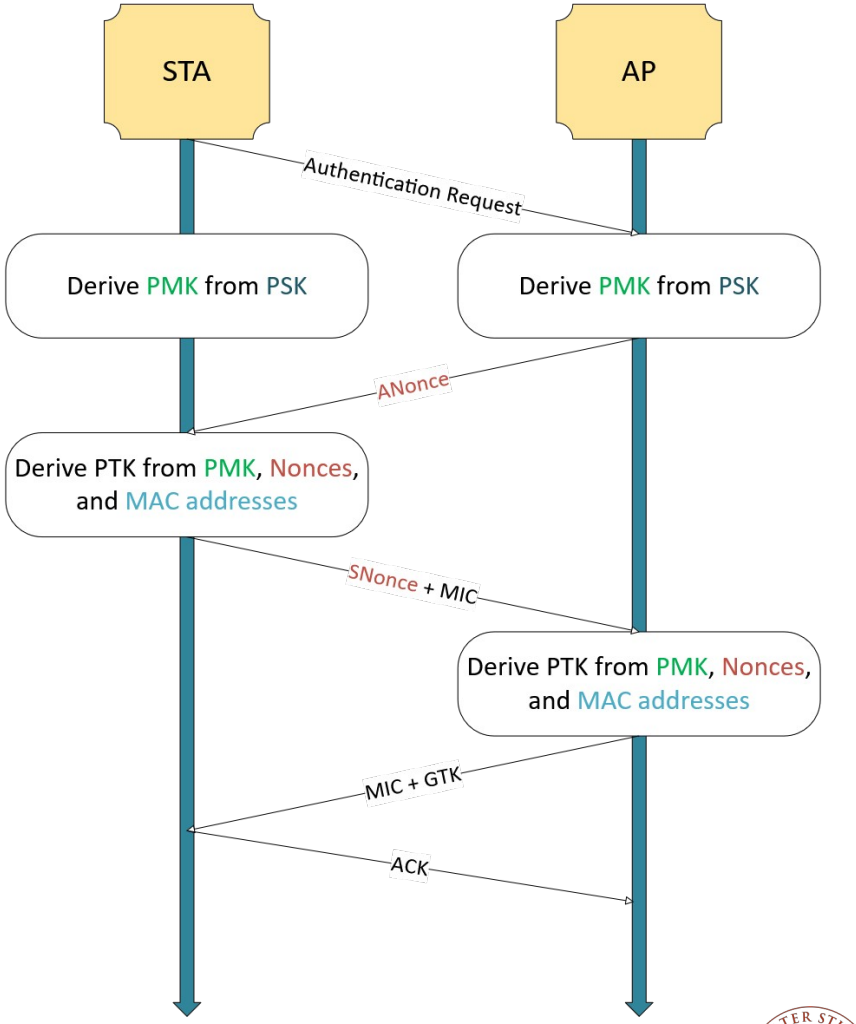


The problem with WPA2 – PSK

When using weak passwords for authentication in a WPA2-secured network, an attacker could easily gain access to the network.

To do so it suffices to **eavesdrop** the authentication process, capture the nonces, capture the MICs, retrieve the MAC addresses of STA and AP, and perform a **brute-force attack** or a **dictionary attack**.

Both these attacks can be performed offline. Using an RTX 4090, a Wi-Fi password simply



How to choose a good password?

Let’s say we have two *characters sets*.

1. The first one is the set of all **26** English letters, from **a** to **z**.
2. The second one is the set of out **10** digits, from **0** to **9**.

What’s better? To have a password that is:

- 10 characters long including only elements from the first *characters set* (26 letters)

or

- 26 characters long including only elements from the second *characters set* (10 digits)

Entropy

To check the strength of a password, we use the concept of **entropy**. Entropy is the number of combinations that the password can assume. It is measured as follows:

Where:

- is the number of elements in the character set of the password.
- is the length of the password itself.

Since it’s measured in bits, the actual formula is:


ALMA MATER STUDIORUM
A.D. 1088

Hard problems

In computational complexity theory, problems can be divided mainly in two main kinds:

- **Decision** problems can be solved with a yes/no answer.
  "*Is 77 prime?" – No.*

- **Search** problems cannot be solved with a yes/no answer but require an actual solution.
  "*What are the prime factors of 77?" –*

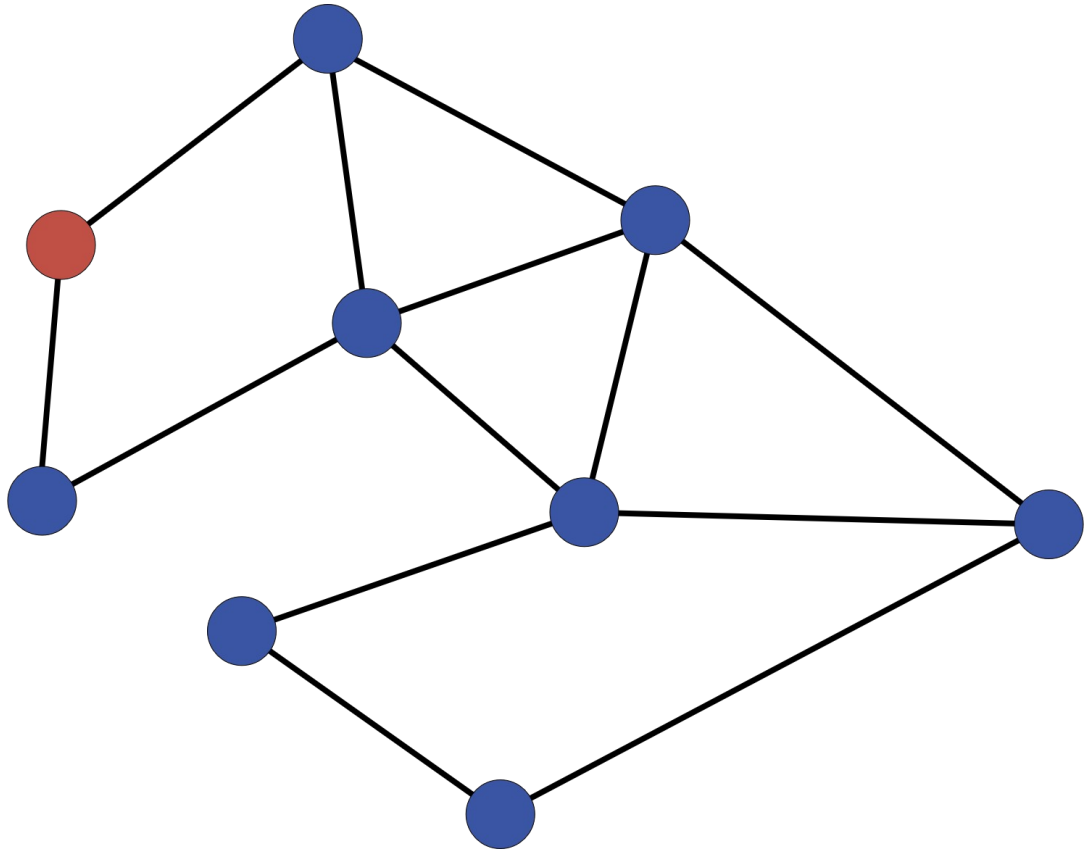
Solving decision problems is **at most** as hard as solving search problems. The first ones are considered *easier* in some cases.

Hamiltonian Circuits

A **Hamiltonian tour** is a path along a graph that visits each vertex exactly once.

In this *NP-complete* problem, answering the following questions is ***equally hard***:

- **Decision**: *does a Hamiltonian circuit exist*?
- **Search**: *can you find a Hamiltonian circuit*?

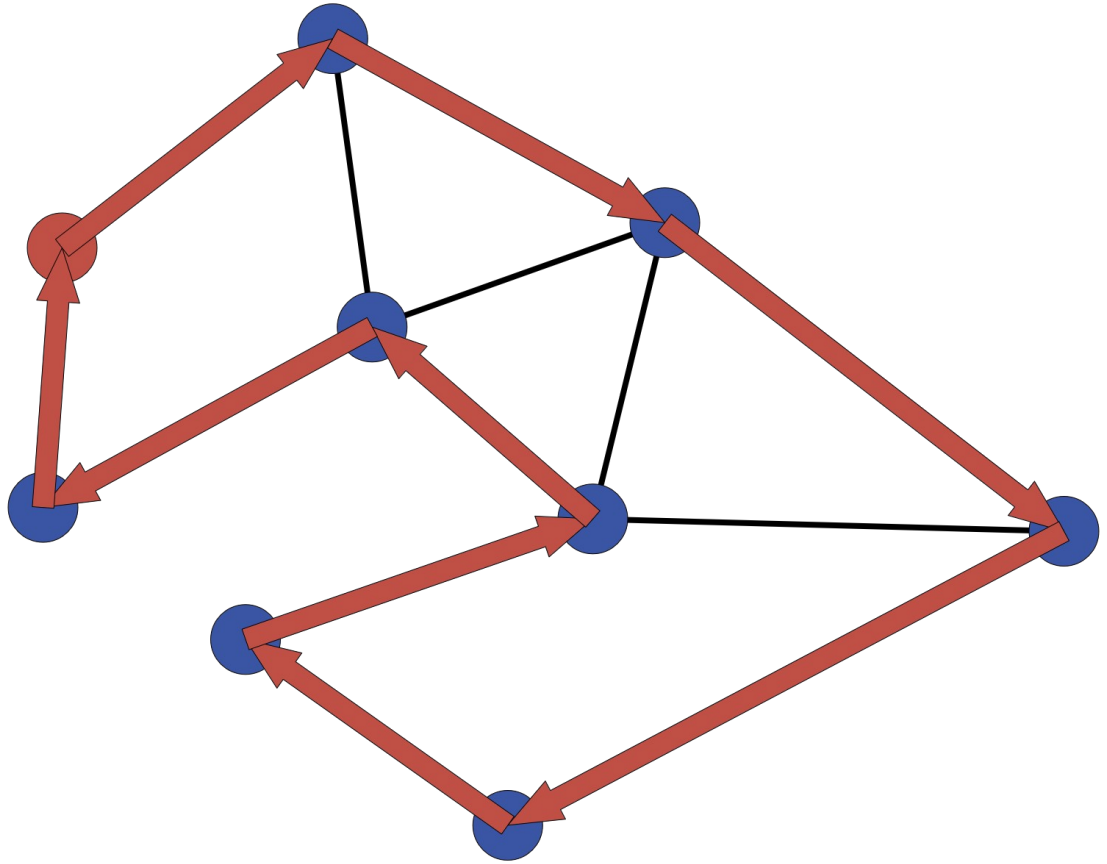


Hamiltonian Circuits

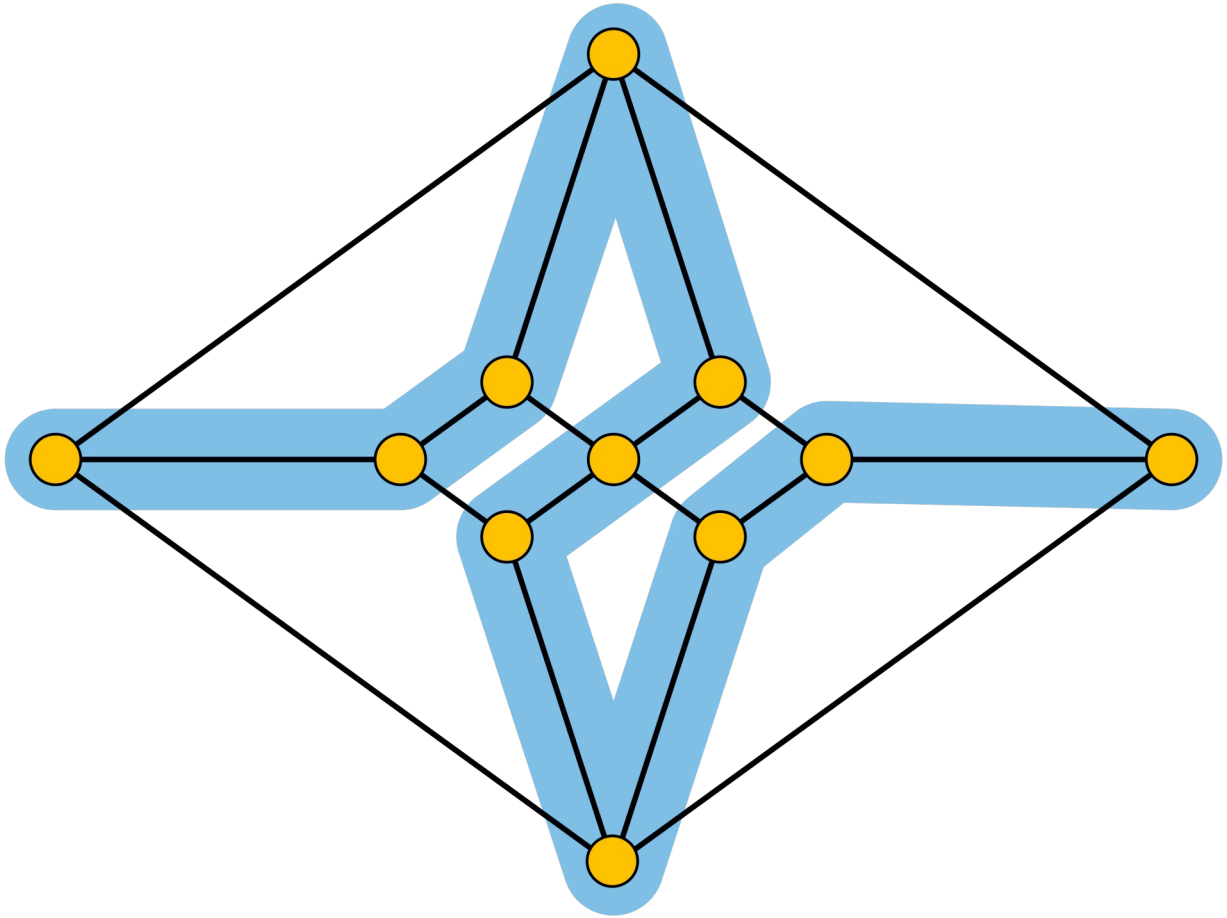
A **Hamiltonian tour** is a path along a graph that visits each vertex exactly once.

In this *NP-complete* problem, answering the following questions is ***equally hard***:

- **Decision**: *does a Hamiltonian circuit exist*?
- **Search**: *can you find a Hamiltonian circuit*?



Hamiltonian Circuits



From
Wikipedia

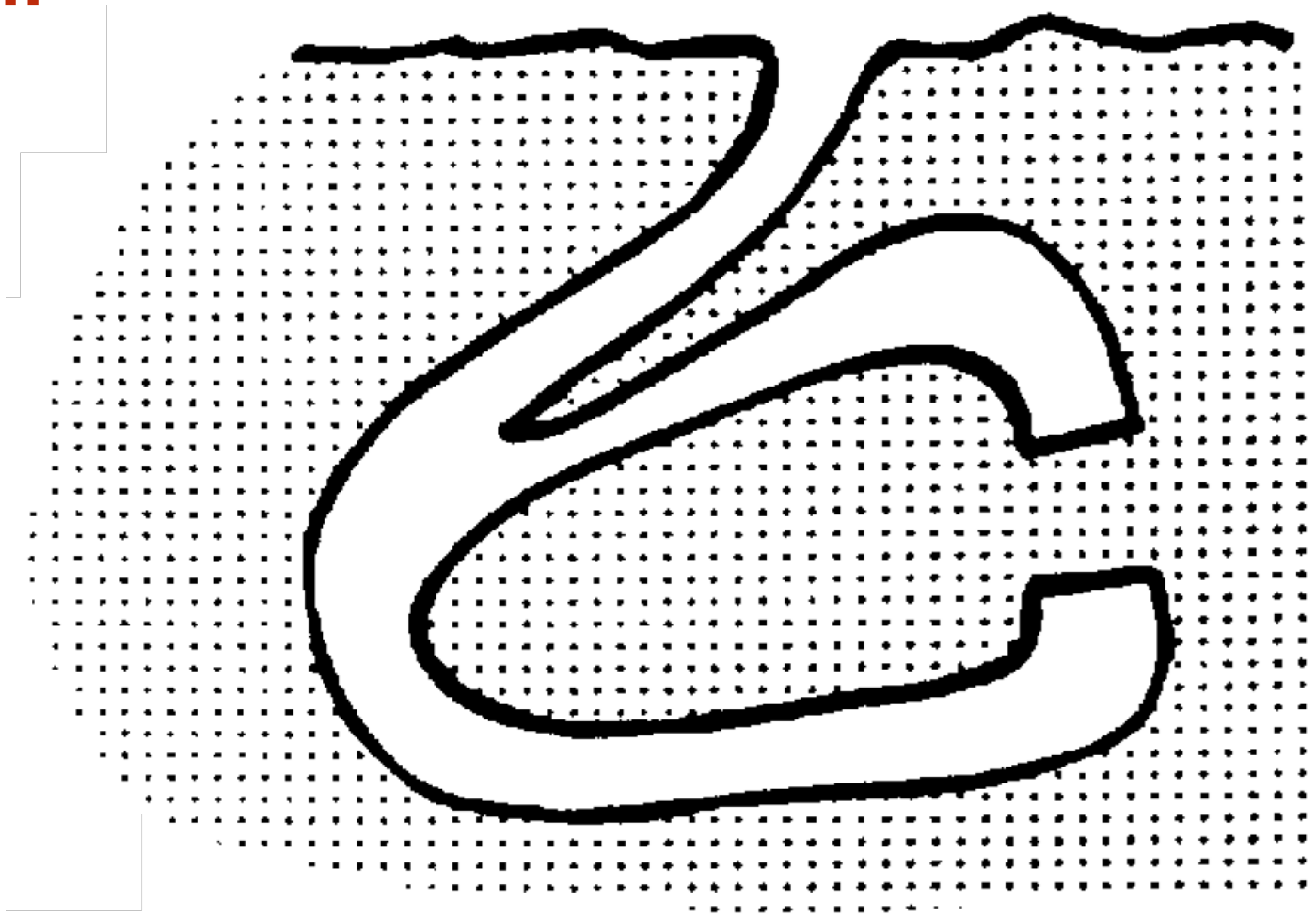
Zero-Knowledge Proofs: a brief introduction

Zero-Knowledge Proofs are first conceived in the paper “**The Knowledge Complexity of Interactive Proof-Systems**” by **Shafi Goldwasser** and **Silvio Micali** in 1985. In this paper, the authors begin by asking the question “How much knowledge should be communicated for proving a theorem T? ”.

To prove that a graph is Hamiltonian, it suffices to exhibit a Hamiltonian tour.

But is it possible to prove the knowledge of a solution to a given theorem without disclosing the actual solution?

How to explain Zero-Knowledge Protocols to your children



Full text available
here.

Takeaways

Simplifying a bit, there are two key takeaways from the Ali Baba cave story:

- The password “Open Sesame” is never revealed.

- The journalist can easily fake the proof: he could agree with an actor to **simulate** the proof by preemptively agreeing on which side of the cave to come out without knowing the password “Open Sesame”.

The authenticity of a zero-knowledge proof resides in the **interaction** between Prover and Verifier.

Statistical Zero-Knowledge

The Ali-Baba cave is a perfect example of **Statistical Zero-Knowledge Protocol** in which a decisional problem is repeated times to achieve a desired amount of probability.

For a person not knowing the solution, picking a random answer would yield the correct guess with a probability of . After querying times, the probability decreases to .

Naturally, if I repeat the experiment times, the probability for a random guesser to succeed is .

Computational Zero-Knowledge

Let’s now make a purely educational example by hashing the “Open Sesame” password with SHA256. We obtain the following bitstring:

- SHA256(“Open Sesame”) =
  01111100101110001001101111100010011000110010010100111110001000011001011001100111000011111001100000011100100100110110011111011000110111100011001100010100100110000001110010001000000100100001101010011010010010011001101101111111101101110011010111010111100000010

Wrongly assuming that no collision exists in SHA256, if I asked you to find a string that hashed produces the above bit string, you would have a chance of succeeding equal to .

Statistical vs. Computational Zero Knowledge

In both the **statistical** and the **computational** cases, the probability of an attacker to successfully prove its statement without knowing the secret is , the two methods are fundamentally different:

- **Computational** Zero-Knowledge is based on the assumption that it's infeasible to solve a hard problem. A machine with unlimited power could break this assumption and retrieve the secret.
- **Statistical** Zero-Knowledge protocols are built in such a way that the secret is never transmitted in any form. Even a machine with unlimited power could not retrieve the secret.

While S-ZKP offer much better properties, they're also time consuming and require a lot of space for the proof. In practice, C-ZKP are the best tradeoff for modern mechanisms.

Zero-Knowledge Properties

For a protocol to be a Zero-Knowledge Protocol, it must satisfy three properties:

- **Completeness**: an honest prover that does know the secret, will always be able to prove a true statement to an honest verifier.
- **Soundness**: a dishonest prover that does **not** know the secret, won't be able to prove a true statement to an honest verifier.
- **Zero-Knowledge**: if the statement is true, the verifier learns nothing from the interaction except for the statement being true. A dishonest verifier does not learn anything about the secret.

Zero-Knowledge Properties in practice

Given a protocol based on mathematical premises, the protocol is zero knowledge if:

- **Completeness**: it is correct. The construction of the protocol via mathematical equations always yields correct results if performed correctly.
- **Soundness**: to perform correctly the protocol, one must know the secret. This is formally proved by formulating an **Extractor** algorithm that, by running the ZKP protocol differently, can extract the secret from it.
- **Zero-Knowledge**: the data sent to the verifier could easily be constructed in another way without knowing the secret. This is formally proved by creating a **Simulator** algorithm that, by running the protocol differently, can simulate a conversation between the

Schnorr’s Identity Protocol (1)

Let be a cyclic group of prime order with generator .

- Prover P has a **secret** .
- The corresponding **public** verification key is . This data can be public due to the assumed infeasibility of computing the **Discrete Logarithm** of .

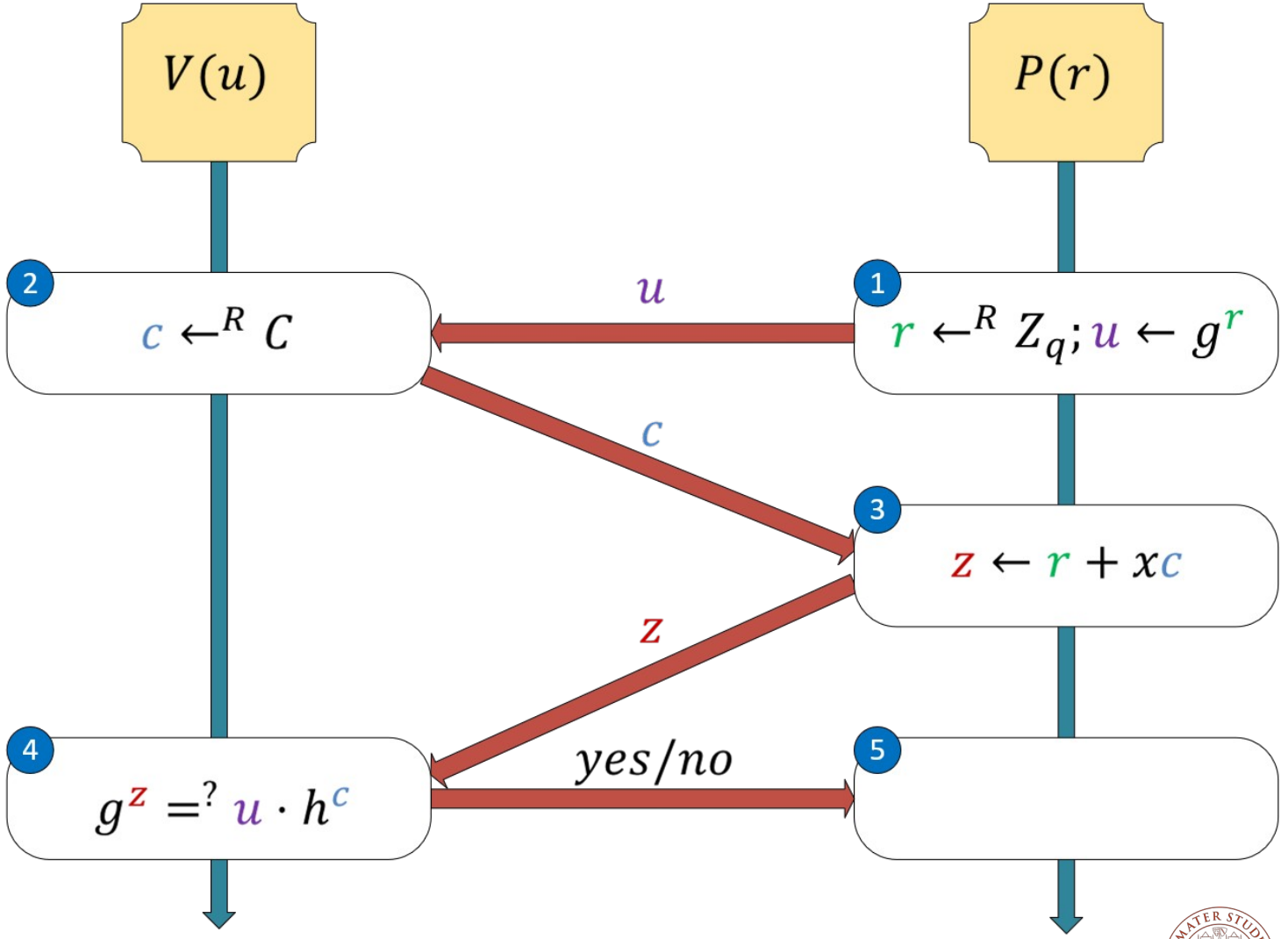
To prove its identity to a Verifier V, P could send and let V compute . This is just a password protocol and, while being resilient against **active attacks**, it is not resilient towards **eavesdropping attacks**.

Schnorr's Identity Protocol (2)

Instead, according to the Schnorr's Identity Protocol, the Prover runs the algorithm on the side.

This is a Zero-Knowledge protocol of the family of **Sigma Protocols**.

Let's see how it satisfies the three properties.

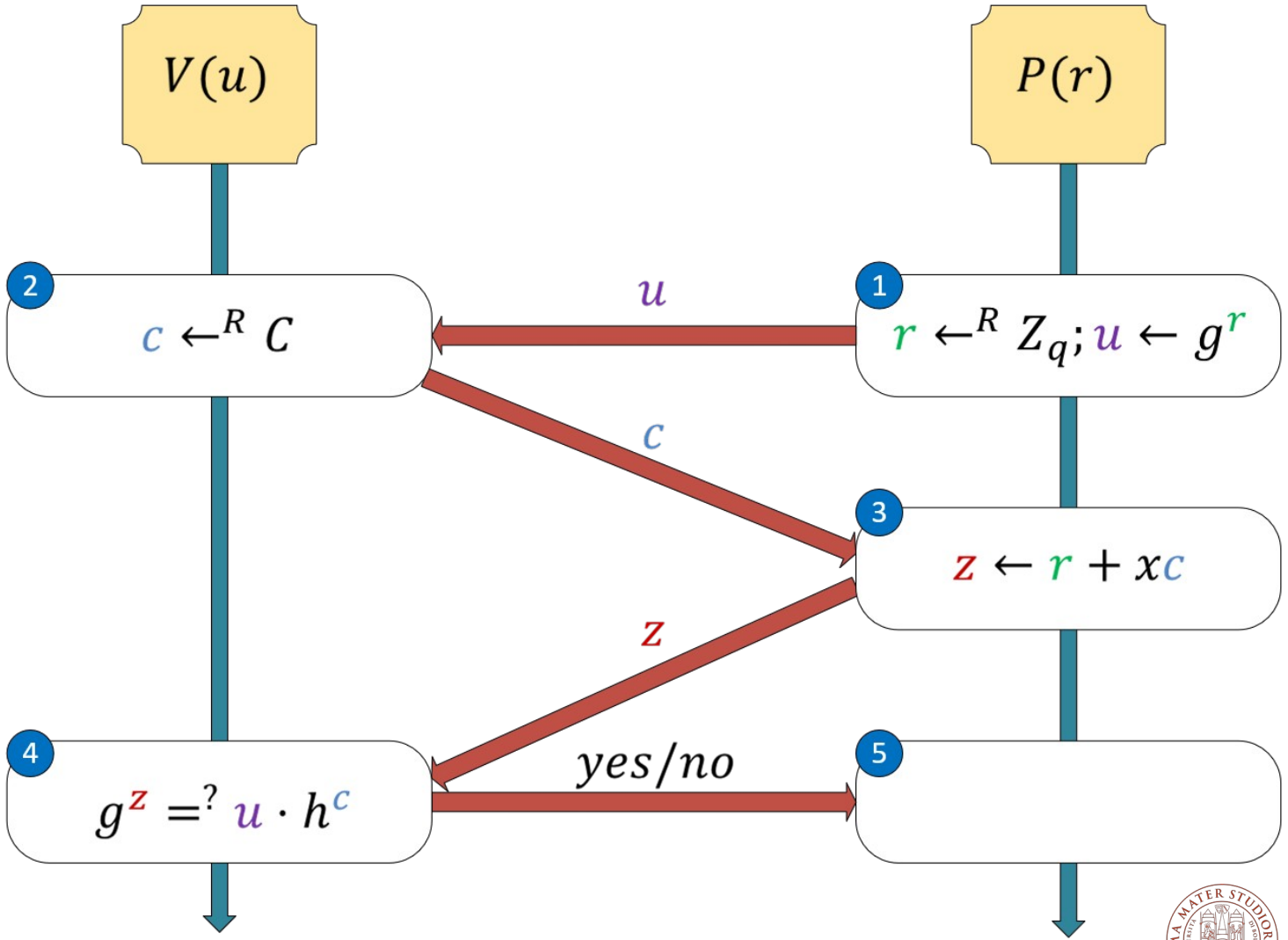


Schnorr's Identity Protocol: Completeness

Preliminary data:

By substituting in the last equation, we have:

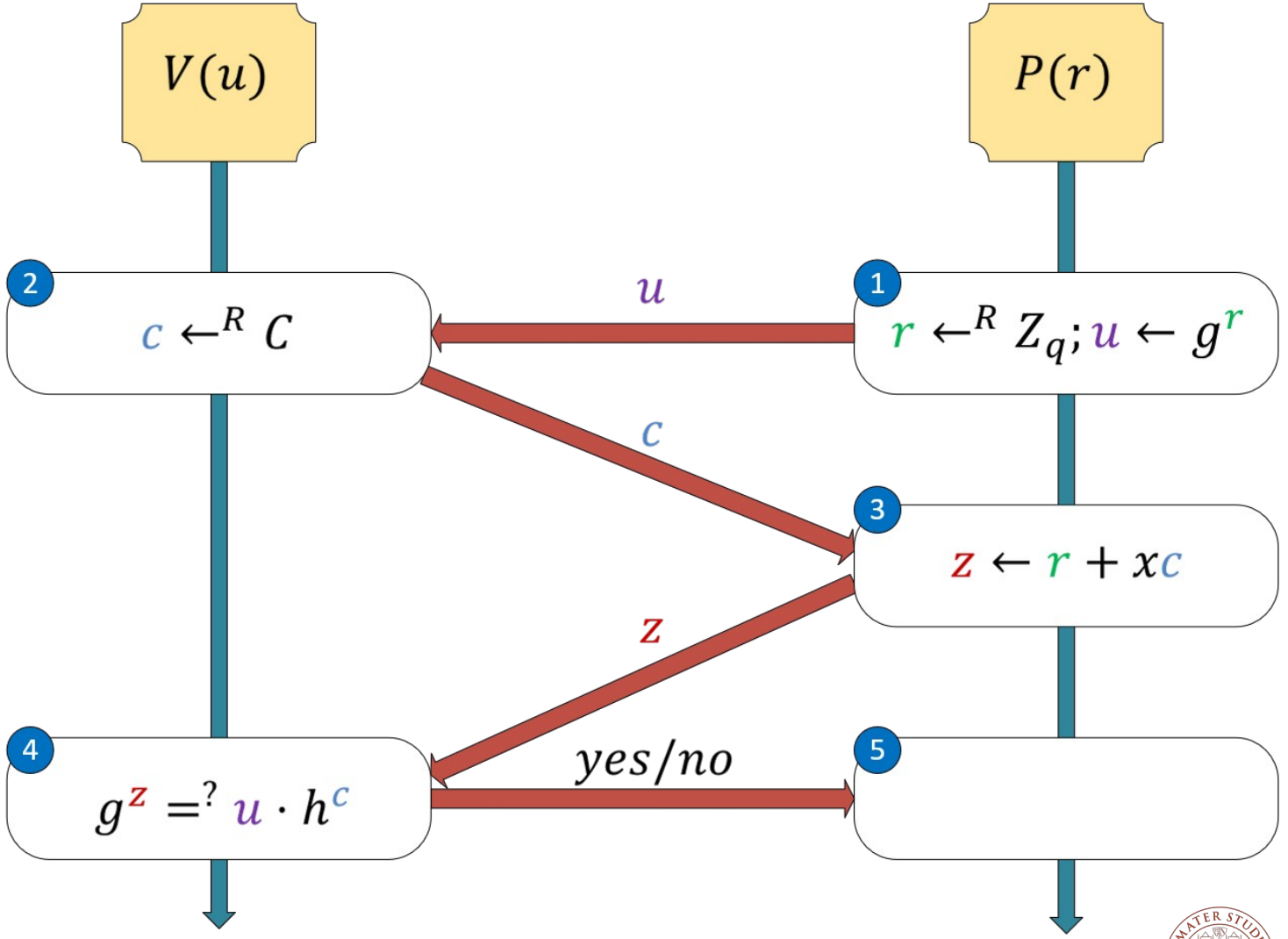
The protocol is, in fact, correct.



Schnorr's Identity Protocol: Soundness (1)

Let's now create an **Extractor** algorithm in which a malicious verifier wants to extract the secret .

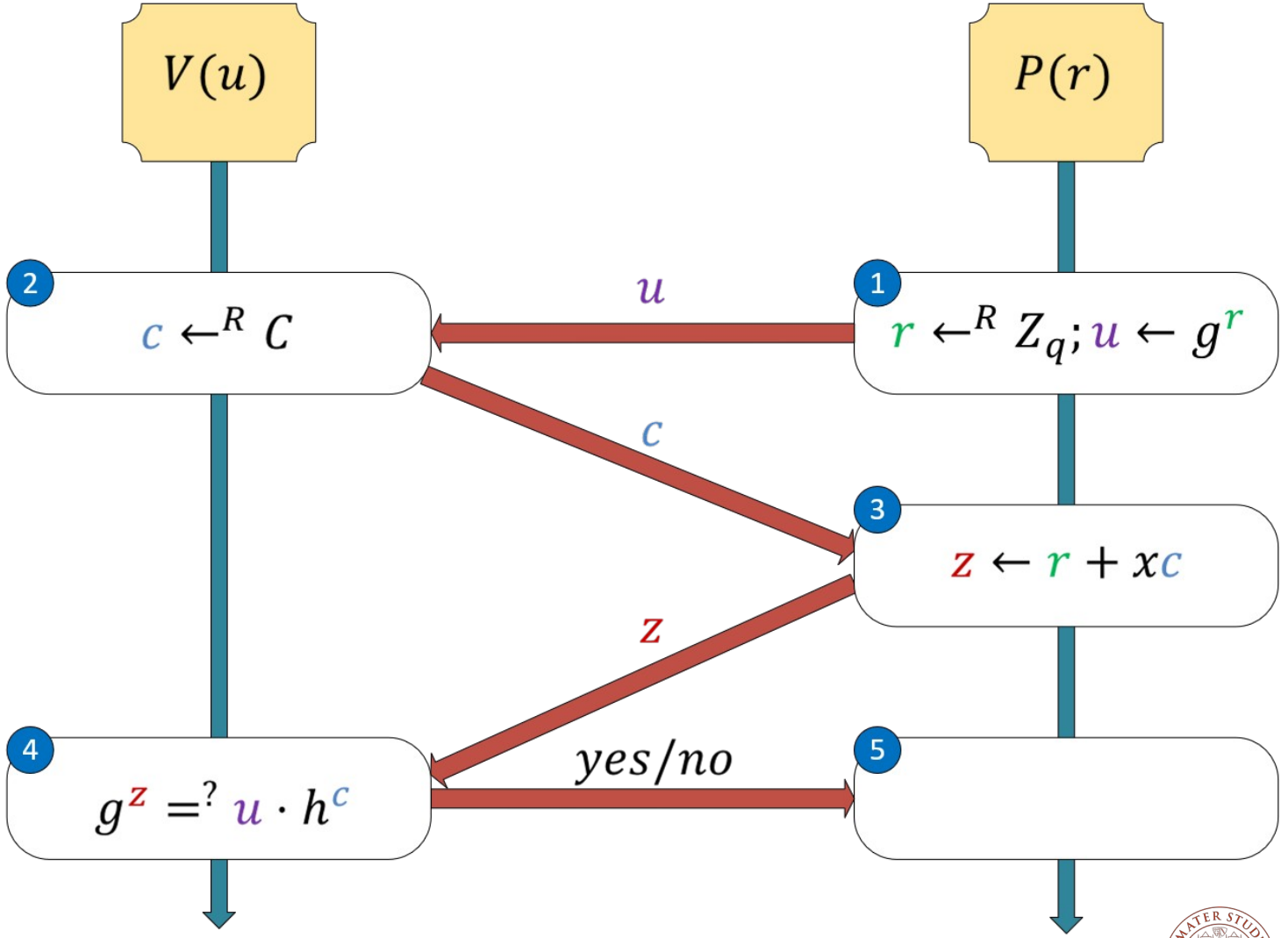
The malicious verifier has a special power: **rewinding**. While performing the Schnorr identification protocol, at any point he can keep the data and rollback the protocol to any of the previous steps.



Schnorr's Identity Protocol: Soundness (2)

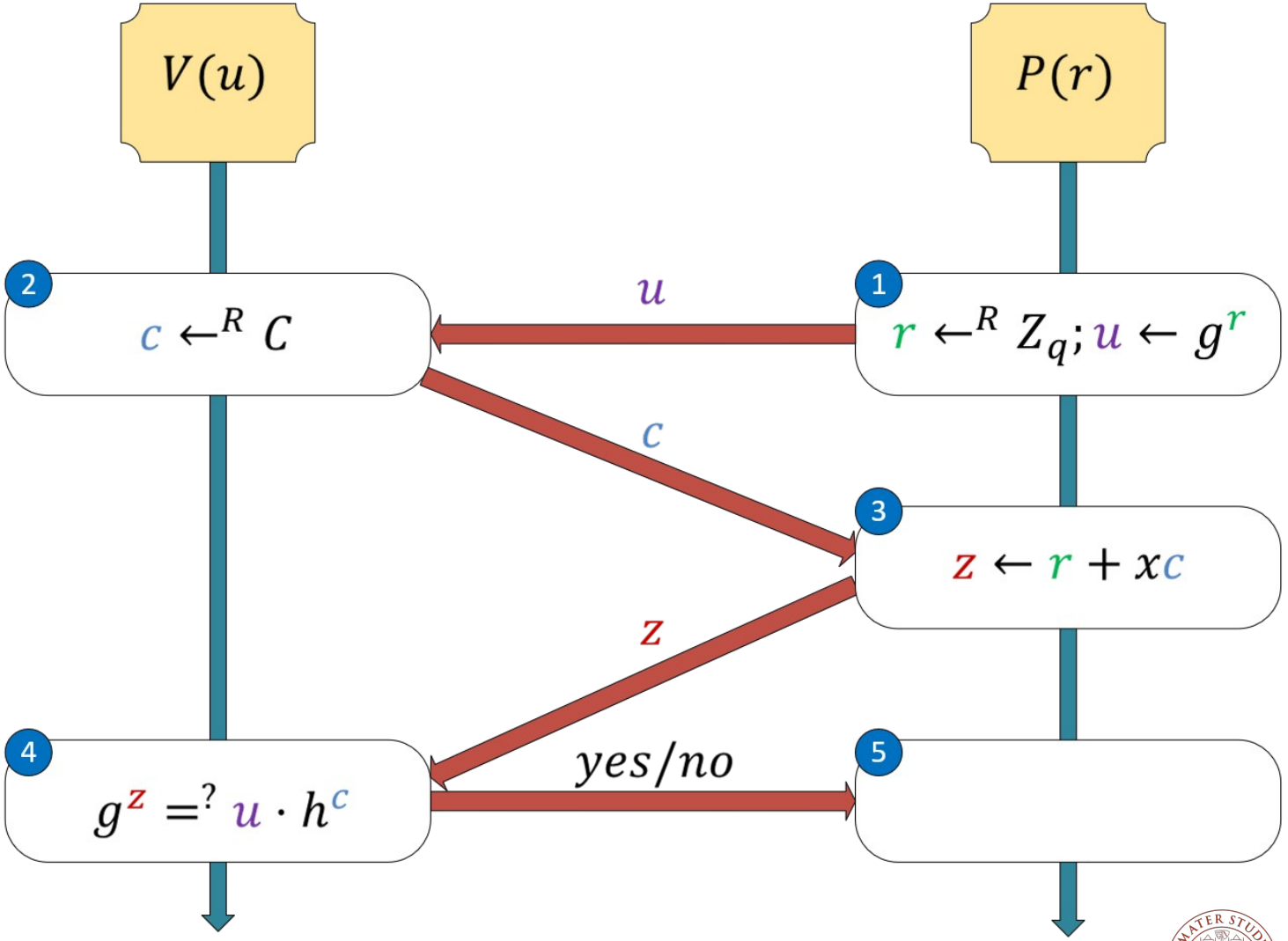
1. At first, he obtains a valid conversation by executing steps 1, 2, and three 3 with the prover.
2. The verifier then **rewinds** the state of the algorithm to step 2. He generates a new and sends it to the prover, which, in turn, responds with a new :

   .



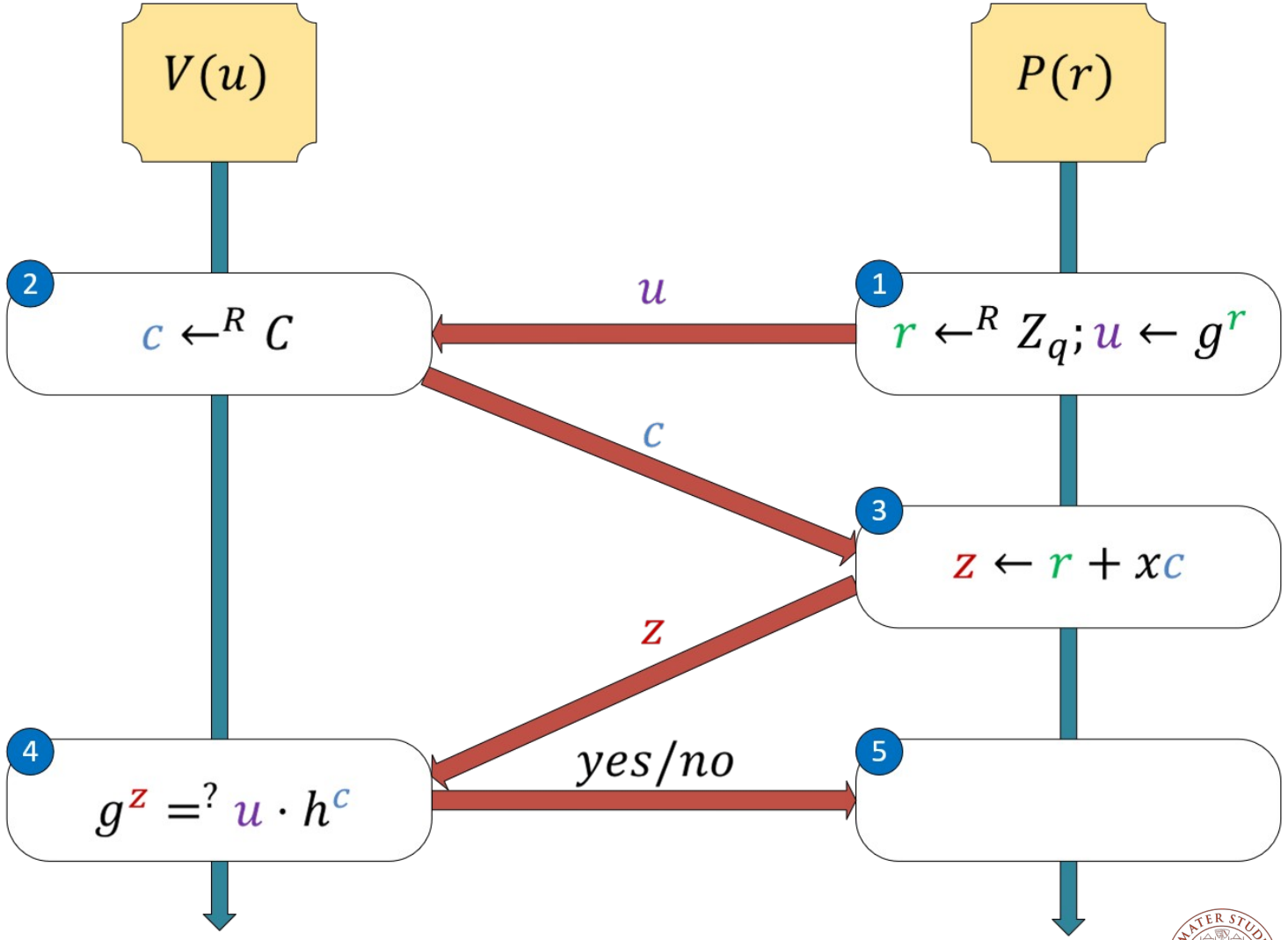
Schnorr's Identity Protocol: Soundness (3)

3. Since and he knows that:



Schnorr's Identity Protocol: Soundness (3)

3.

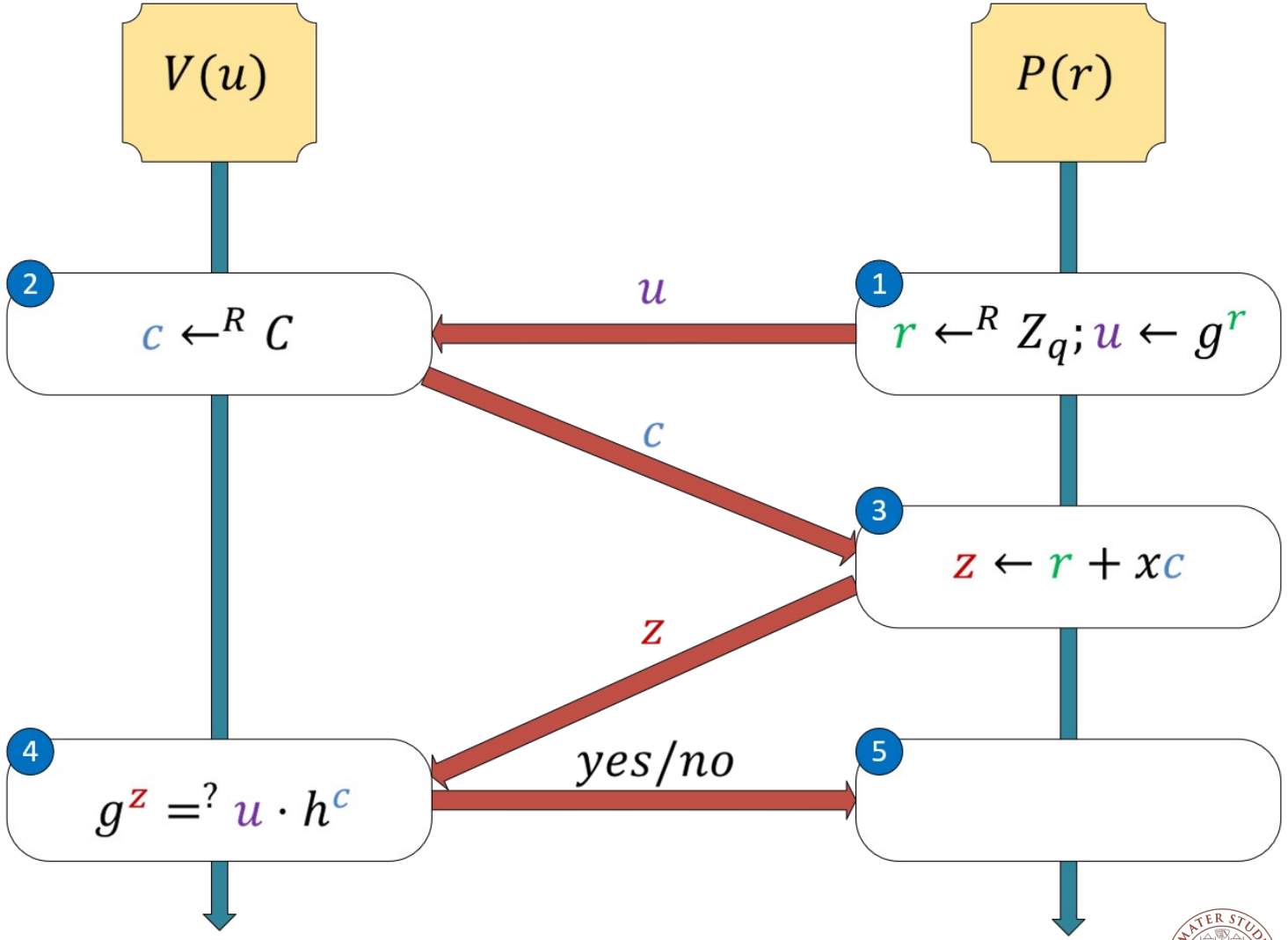


The verifier has successfully extracted the secret .
This is why in real scenarios, **rewinding** is strictly **forbidden**.

Schnorr's Identity Protocol: Zero-Knowledge (1)

It's now time to prove that the Schnorr identification protocol is, in fact, **Zero-Knowledge**.

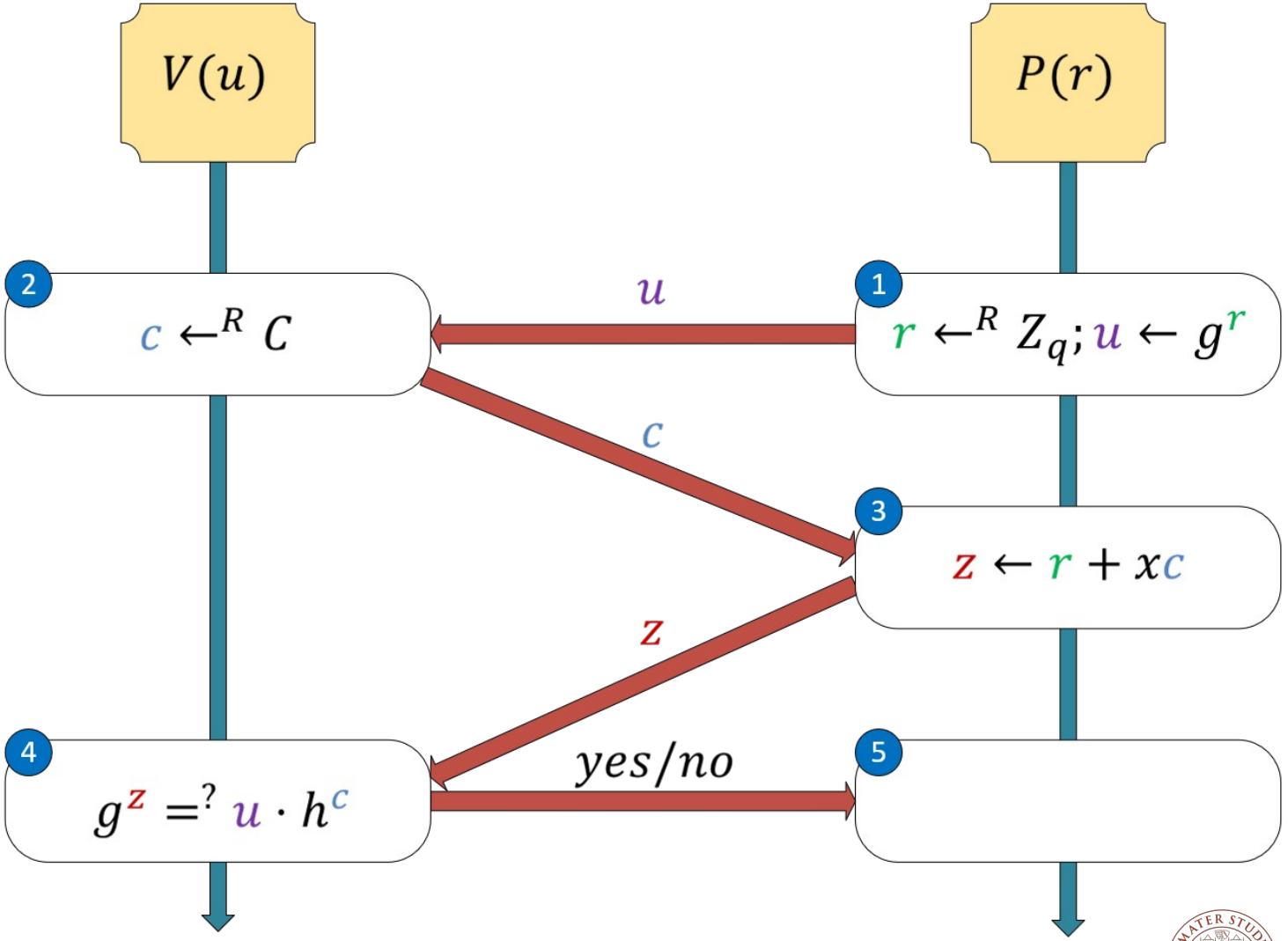
This is done theoretically by creating a **Simulator** algorithm that simulates a valid conversation without knowledge of the secret .



Schnorr's Identity Protocol: Zero-Knowledge (2)

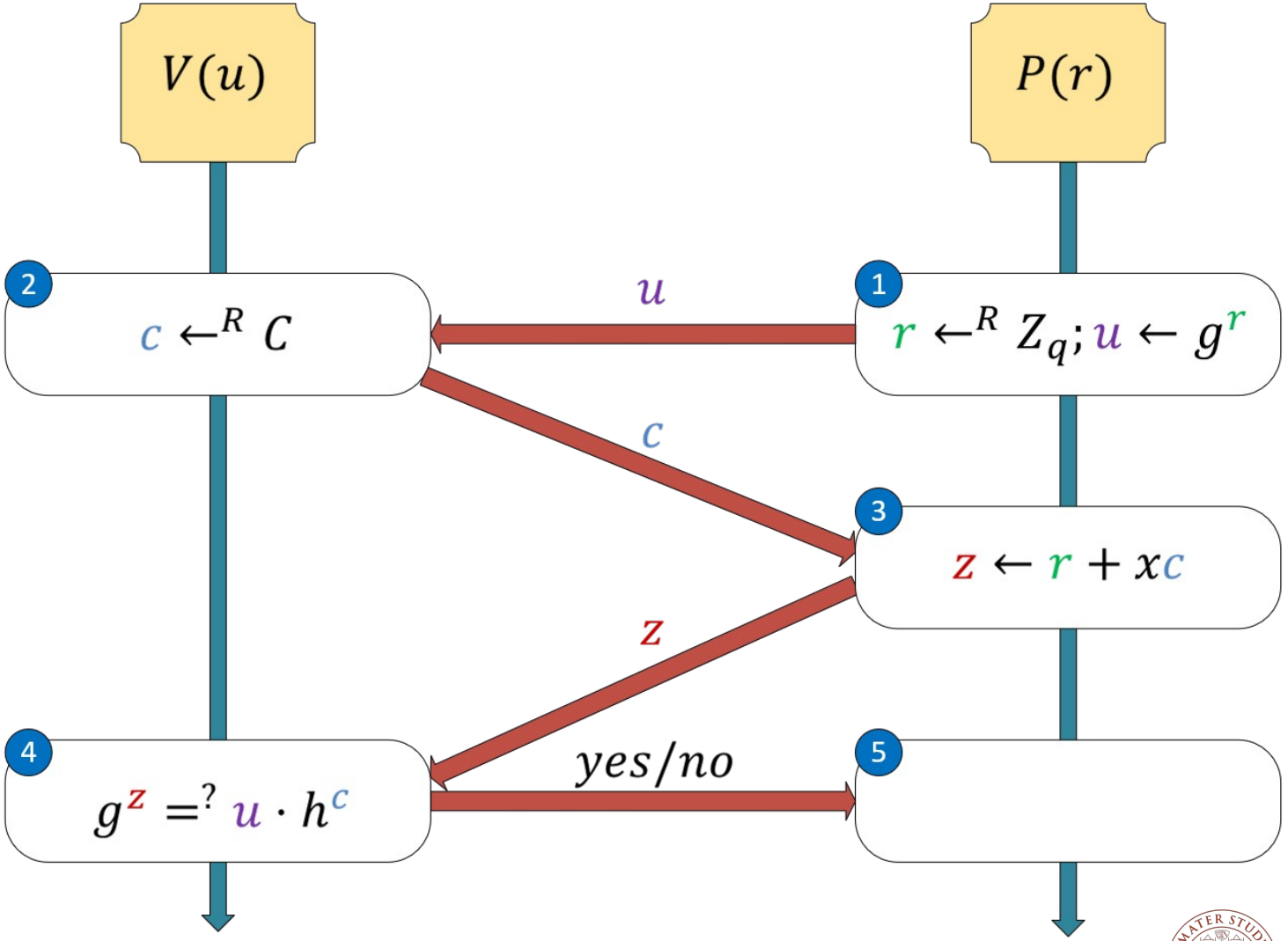
The Simulator acts in a **<u>backwards</u>** way with respect to the original algorithm.

1.
2.
3.



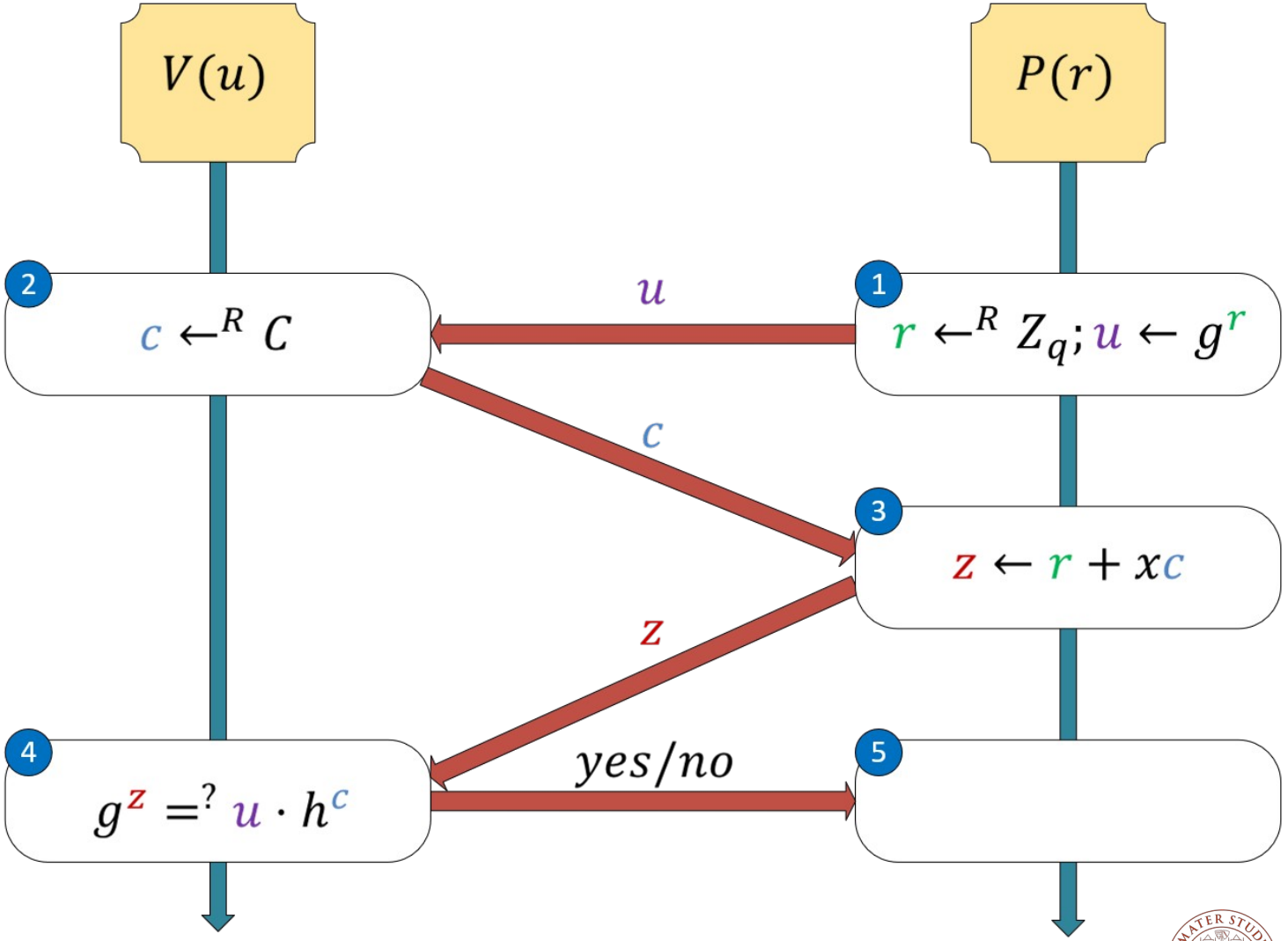
Schnorr's Identity Protocol: Zero-Knowledge (2)

Proof:



Schnorr's Identity Protocol: Zero-Knowledge (2)

Cont.



Hence, we proved that Simulator produces a valid conversation without knowledge of .

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Credits:

Alessandro Buldini

alessandro.buldini@unibo.it

www.unibo.it