Hash functions in post-quantum cryptography

Christian Majenz
CWI

QuSoft
Research Center for Quantum Software

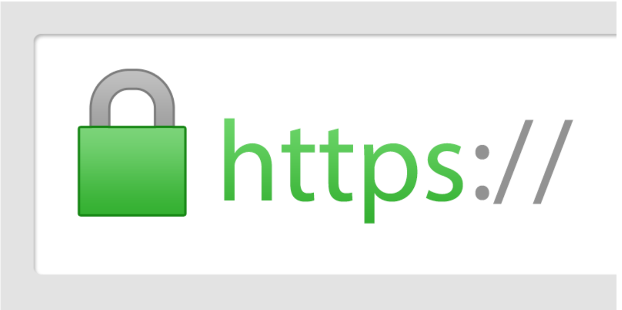
CWI
Centrum Wiskunde & Informatica

Cryptography is everywhere

Cryptography is everywhere

https://

Cryptography is everywhere



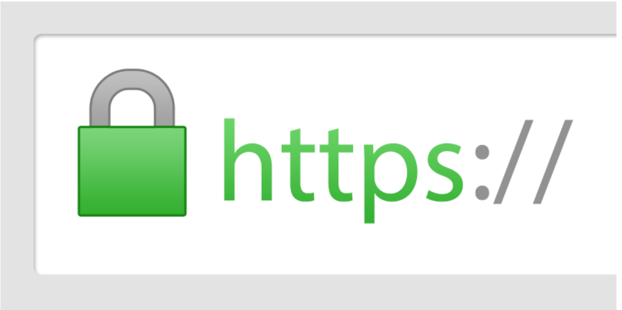
https://

Cryptography is everywhere

https://

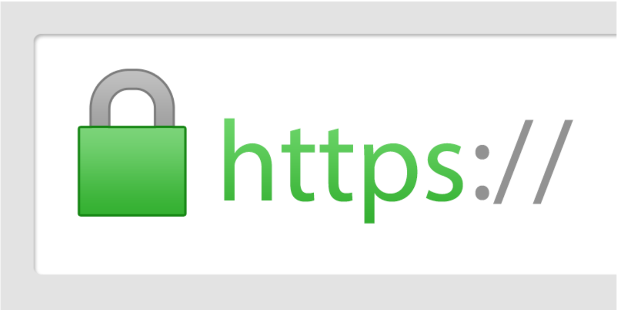
DigiD

Cryptography is everywhere



https://

Cryptography is everywhere



BITCOIN·DIGITAL·DECENTRALIZED·PEER TO PEER
MONETARY METALS
MJB COPPER 999 FINE 1 TROY OZ

Quantum computers

Quantum computers

- Accelerating effort to build a quantum computer

Quantum computers

- Accelerating effort to build a quantum computer
- Major investments:

Google
IBM
CHINESE ACADEMY OF SCIENCES
intel
Microsoft
QUANTUM
FLAGSHIP

Quantum computers

- Accelerating effort to build a quantum computer
- Major investments:

Google IBM

intel Microsoft

CHINESE ACADEMY OF SCIENCES

QUANTUM FLAGSHIP

We need to prepare cryptography for the arrival of quantum computers!

Quantum computers

- Accelerating effort to build a quantum computer
- Major investments:

Google IBM

CHINESE ACADEMY OF SCIENCES

intel Microsoft

QUANTUM FLAGSHIP

We need to prepare cryptography for the arrival of quantum computers!

- Security against quantum attackers

nationale wetenschaps agenda

- Quantum cryptography

Quantum computers

- Accelerating effort to build a quantum computer
- Major investments:

We need to prepare cryptography for the arrival of quantum computers!

- **This talk: Security against quantum attackers (post-quantum cryptography)**

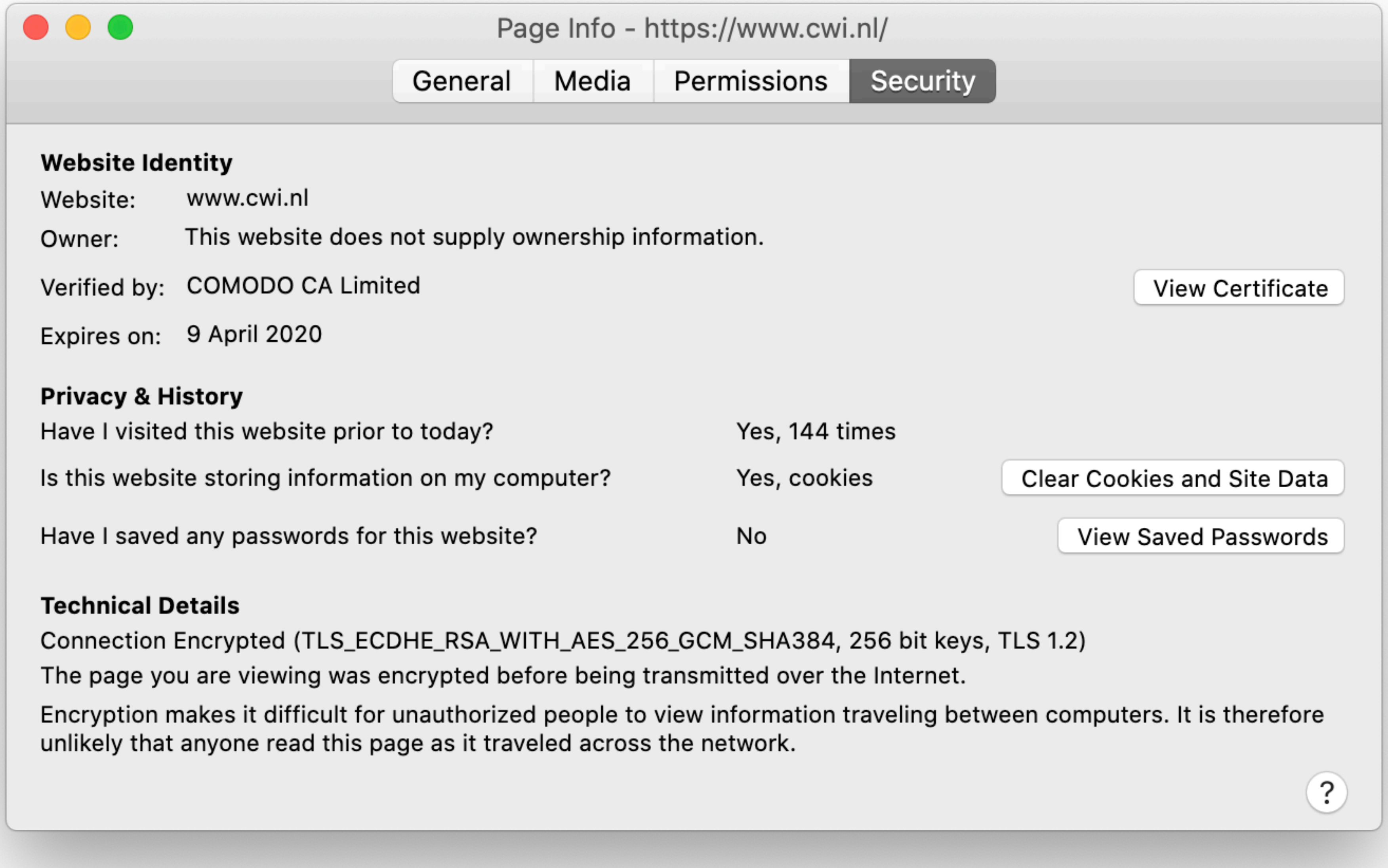
Elements of post-quantum crypto

Elements of post-quantum crypto

- Quantum Cryptanalysis

Elements of post-quantum crypto

- Quantum Cryptanalysis



Website Identity

Website:	www.cwi.nl
Owner:	This website does not supply ownership information.
Verified by:	COMODO CA Limited
Expires on:	9 April 2020

Privacy & History

Have I visited this website prior to today?	Yes, 144 times
Is this website storing information on my computer?	Yes, cookies
Have I saved any passwords for this website?	No

Technical Details

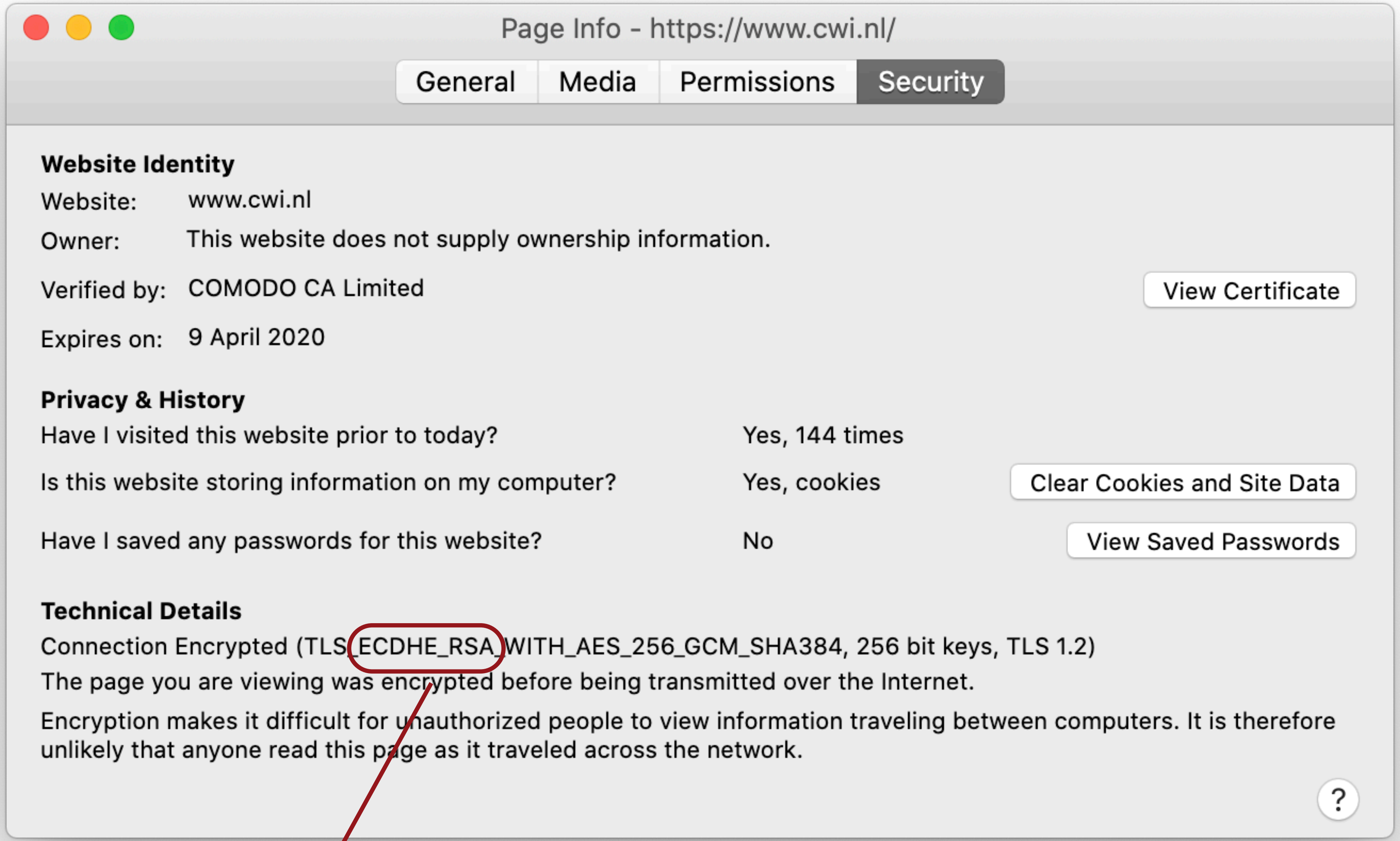
Connection Encrypted (TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, 256 bit keys, TLS 1.2)

The page you are viewing was encrypted before being transmitted over the Internet.

Encryption makes it difficult for unauthorized people to view information traveling between computers. It is therefore unlikely that anyone read this page as it traveled across the network.

Elements of post-quantum crypto

- Quantum Cryptanalysis



Website Identity

Website: www.cwi.nl

Owner: This website does not supply ownership information.

Verified by: COMODO CA Limited

Expires on: 9 April 2020

Privacy & History

Have I visited this website prior to today? Yes, 144 times

Is this website storing information on my computer? Yes, cookies

Have I saved any passwords for this website? No

Technical Details

Connection Encrypted (TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, 256 bit keys, TLS 1.2)

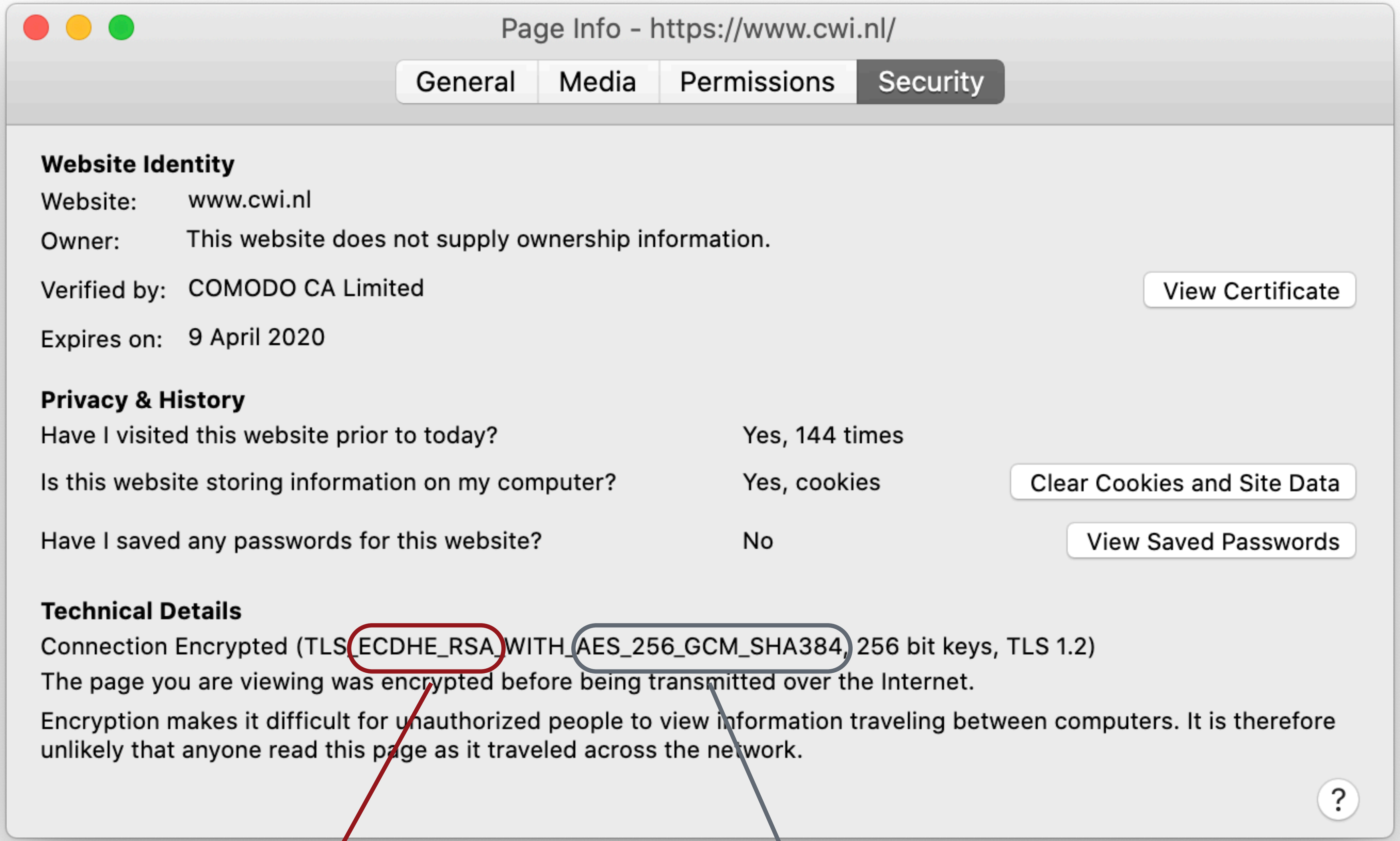
The page you are viewing was encrypted before being transmitted over the Internet.

Encryption makes it difficult for unauthorized people to view information traveling between computers. It is therefore unlikely that anyone read this page as it traveled across the network.

Shor's algorithm:
Complete break

Elements of post-quantum crypto

- Quantum Cryptanalysis



Website Identity

Website:	www.cwi.nl
Owner:	This website does not supply ownership information.
Verified by:	COMODO CA Limited
Expires on:	9 April 2020

Privacy & History

Have I visited this website prior to today?	Yes, 144 times
Is this website storing information on my computer?	Yes, cookies
Have I saved any passwords for this website?	No

Technical Details

Connection Encrypted (TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, 256 bit keys, TLS 1.2)

The page you are viewing was encrypted before being transmitted over the Internet.

Encryption makes it difficult for unauthorized people to view information traveling between computers. It is therefore unlikely that anyone read this page as it traveled across the network.

Shor's algorithm:
Complete break

Grover's algorithm:
Might necessitate increased key lengh

Elements of post-quantum crypto

- Quantum Cryptanalysis: Shor, Grover

- Quantum-secure computational assumptions
  - Lattice problems
  - Decoding random codes
  - Inverting multivariate polynomials
  - Secure hash functions
  - Supersingular isogeny Diffie-Hellman

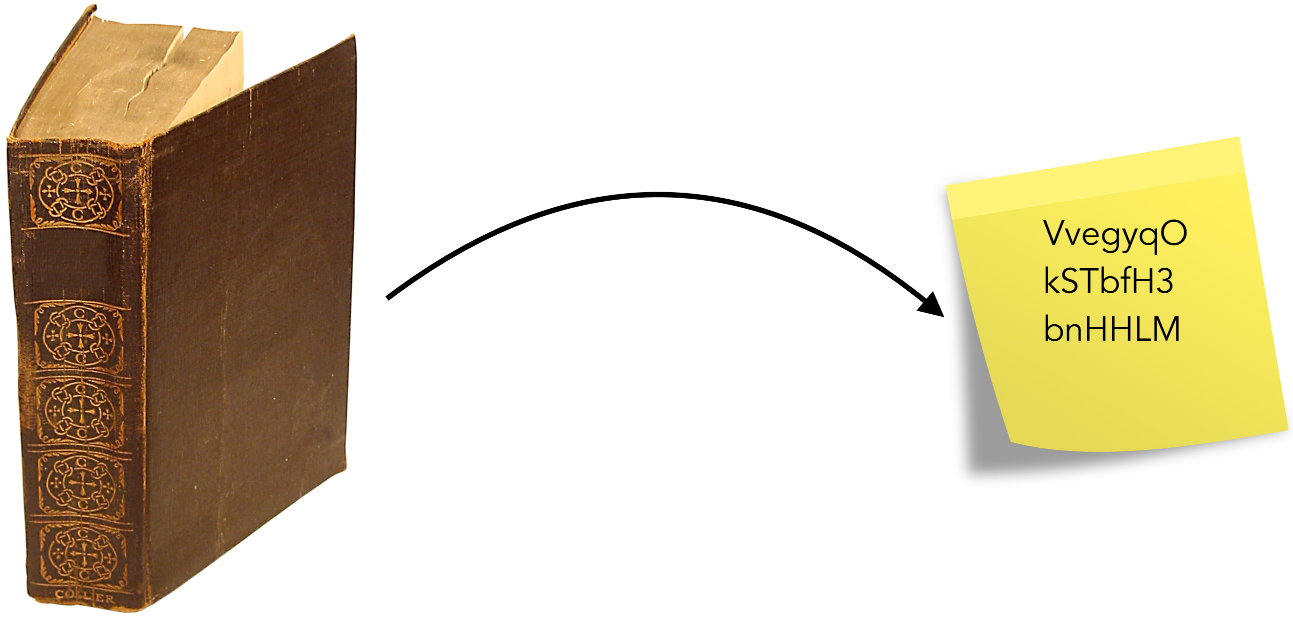
Elements of post-quantum crypto

- Quantum Cryptanalysis: Shor, Grover

- Quantum-secure computational assumptions
    - Lattice problems
    - Decoding random codes
    - Inverting multivariate polynomials
    - Secure hash functions
    - Supersingular isogeny Diffie-Hellman

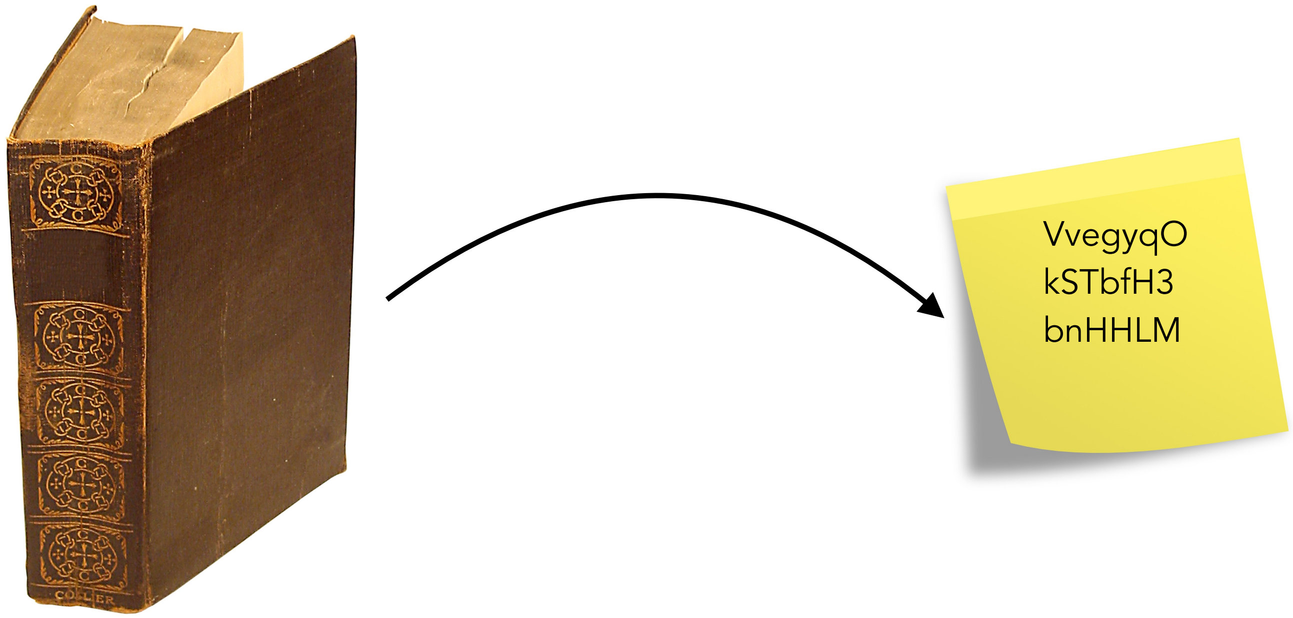
- Models: Quantum Random Oracle Model (QROM)

Hash functions



VvegyqO
kSTbfH3
bnHHLM

Hash functions



Ubiquitous in cryptography. Example: digital signatures

The (Q)ROM

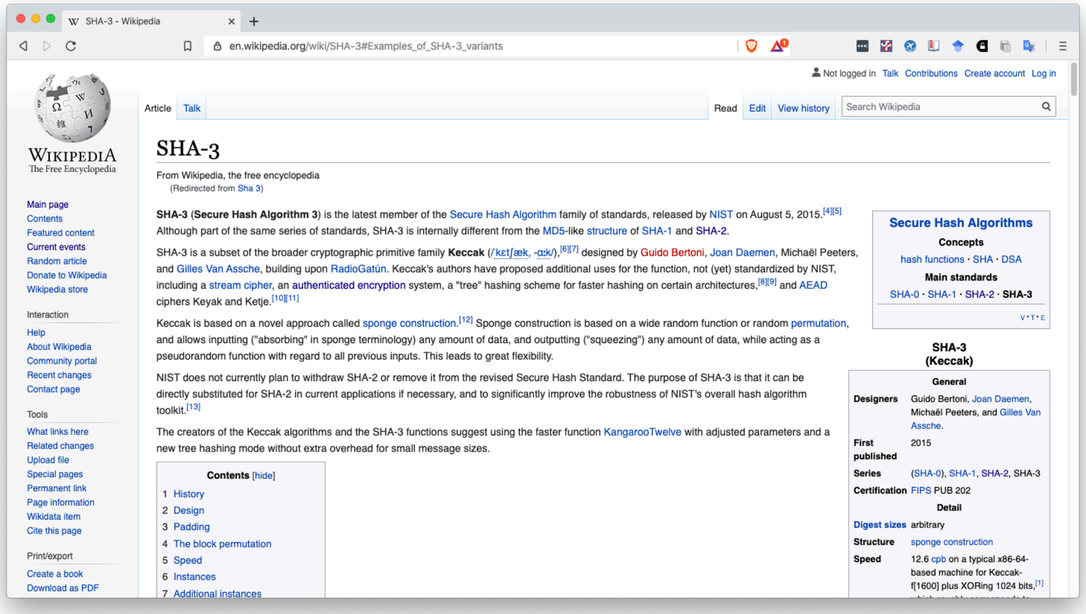
The (Q)ROM

Reality

Model

The (Q)ROM

Reality



SHA-3

From Wikipedia, the free encyclopedia
(Redirected from Sha 3)

SHA-3 (**Secure Hash Algorithm 3**) is the latest member of the Secure Hash Algorithm family of standards, released by NIST on August 5, 2015.[4][5] Although part of the same series of standards, SHA-3 is internally different from the MD5-like structure of SHA-1 and SHA-2.

SHA-3 is a subset of the broader cryptographic primitive family **Keccak** (/ˈkɛtʃæk, -ɑːk/),[6][7] designed by Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche, building upon RadioGatún. Keccak's authors have proposed additional uses for the function, not (yet) standardized by NIST, including a stream cipher, an authenticated encryption system, a "tree" hashing scheme for faster hashing on certain architectures,[8][9] and AEAD ciphers Keyak and Ketje.[10][11]

Keccak is based on a novel approach called sponge construction.[12] Sponge construction is based on a wide random function or random permutation, and allows inputting ("absorbing" in sponge terminology) any amount of data, and outputting ("squeezing") any amount of data, while acting as a pseudorandom function with regard to all previous inputs. This leads to great flexibility.

NIST does not currently plan to withdraw SHA-2 or remove it from the revised Secure Hash Standard. The purpose of SHA-3 is that it can be directly substituted for SHA-2 in current applications if necessary, and to significantly improve the robustness of NIST's overall hash algorithm toolkit.[13]

The creators of the Keccak algorithms and the SHA-3 functions suggest using the faster function KangarooTwelve with adjusted parameters and a new tree hashing mode without extra overhead for small message sizes.

Contents [hide]

Secure Hash Algorithms

Concepts

hash functions · SHA · DSA

Main standards

SHA-0 · SHA-1 · SHA-2 · **SHA-3**

V · T · E

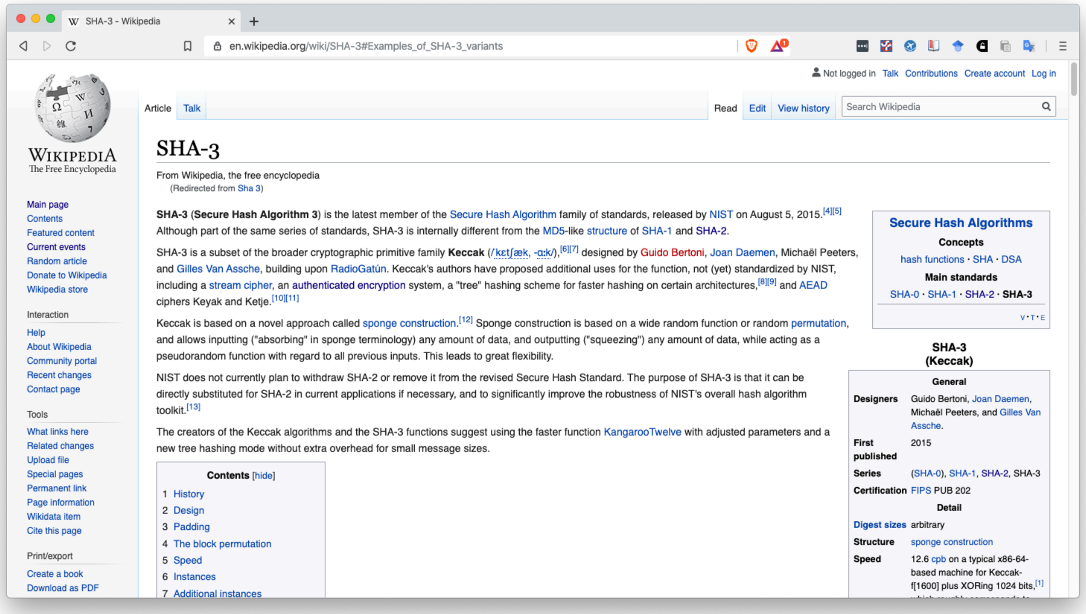
SHA-3
(Keccak)

General	
Designers	Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.
First published	2015
Series	(SHA-0), SHA-1, SHA-2, SHA-3
Certification	FIPS PUB 202
Detail	
Digest sizes	arbitrary
Structure	sponge construction
Speed	12.6 cpb on a typical x86-64-based machine for Keccak-f[1600] plus XORing 1024 bits,[1]

Model

The (Q)ROM

Reality



SHA-3

From Wikipedia, the free encyclopedia
(Redirected from Sha 3)

SHA-3 (**Secure Hash Algorithm 3**) is the latest member of the Secure Hash Algorithm family of standards, released by NIST on August 5, 2015.[4][5] Although part of the same series of standards, SHA-3 is internally different from the MD5-like structure of SHA-1 and SHA-2.

SHA-3 is a subset of the broader cryptographic primitive family **Keccak** (/ˈkɛtʃæk, -ɑːk/),[6][7] designed by Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche, building upon RadioGatún. Keccak's authors have proposed additional uses for the function, not (yet) standardized by NIST, including a stream cipher, an authenticated encryption system, a "tree" hashing scheme for faster hashing on certain architectures,[8][9] and AEAD ciphers Keyak and Ketje.[10][11]

Keccak is based on a novel approach called sponge construction.[12] Sponge construction is based on a wide random function or random permutation, and allows inputting ("absorbing" in sponge terminology) any amount of data, and outputting ("squeezing") any amount of data, while acting as a pseudorandom function with regard to all previous inputs. This leads to great flexibility.

NIST does not currently plan to withdraw SHA-2 or remove it from the revised Secure Hash Standard. The purpose of SHA-3 is that it can be directly substituted for SHA-2 in current applications if necessary, and to significantly improve the robustness of NIST's overall hash algorithm toolkit.[13]

The creators of the Keccak algorithms and the SHA-3 functions suggest using the faster function KangarooTwelve with adjusted parameters and a new tree hashing mode without extra overhead for small message sizes.

Contents [hide]

Secure Hash Algorithms

Concepts

hash functions · SHA · DSA

Main standards

SHA-0 · SHA-1 · SHA-2 · **SHA-3**

V · T · E

SHA-3
(Keccak)

General	
Designers	Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.
First published	2015
Series	(SHA-0), SHA-1, SHA-2, SHA-3
Certification	FIPS PUB 202
Detail	
Digest sizes	arbitrary
Structure	sponge construction
Speed	12.6 cpb on a typical x86-64-based machine for Keccak-f[1600] plus XORing 1024 bits,[1]

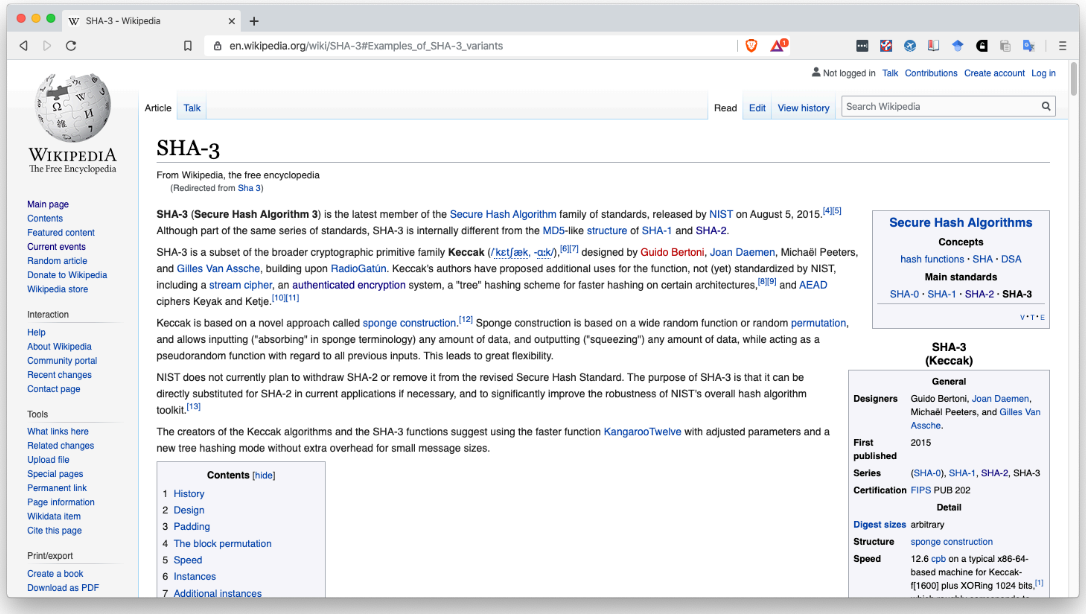
Model

$$H : \{0,1\}^* \rightarrow \{0,1\}^n$$

Uniformly random

The (Q)ROM

Reality



SHA-3

From Wikipedia, the free encyclopedia
(Redirected from Sha 3)

SHA-3 (**Secure Hash Algorithm 3**) is the latest member of the Secure Hash Algorithm family of standards, released by NIST on August 5, 2015.[4][5] Although part of the same series of standards, SHA-3 is internally different from the MD5-like structure of SHA-1 and SHA-2.

SHA-3 is a subset of the broader cryptographic primitive family **Keccak** (/ˈkɛtʃæk, -ɑːk/),[6][7] designed by Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche, building upon RadioGatún. Keccak's authors have proposed additional uses for the function, not (yet) standardized by NIST, including a stream cipher, an authenticated encryption system, a "tree" hashing scheme for faster hashing on certain architectures,[8][9] and AEAD ciphers Keyak and Ketje.[10][11]

Keccak is based on a novel approach called sponge construction.[12] Sponge construction is based on a wide random function or random permutation, and allows inputting ("absorbing" in sponge terminology) any amount of data, and outputting ("squeezing") any amount of data, while acting as a pseudorandom function with regard to all previous inputs. This leads to great flexibility.

NIST does not currently plan to withdraw SHA-2 or remove it from the revised Secure Hash Standard. The purpose of SHA-3 is that it can be directly substituted for SHA-2 in current applications if necessary, and to significantly improve the robustness of NIST's overall hash algorithm toolkit.[13]

The creators of the Keccak algorithms and the SHA-3 functions suggest using the faster function KangarooTwelve with adjusted parameters and a new tree hashing mode without extra overhead for small message sizes.

Contents [hide]

Secure Hash Algorithms

Concepts

hash functions · SHA · DSA

Main standards

SHA-0 · SHA-1 · SHA-2 · **SHA-3**

V · T · E

SHA-3
(Keccak)

General	
Designers	Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.
First published	2015
Series	(SHA-0), SHA-1, SHA-2, SHA-3
Certification	FIPS PUB 202
Detail	
Digest sizes	arbitrary
Structure	sponge construction
Speed	12.6 cpb on a typical x86-64-based machine for Keccak-f[1600] plus XORing 1024 bits,[1]

Model

$H : \{0,1\}^* \to \{0,1\}^n$

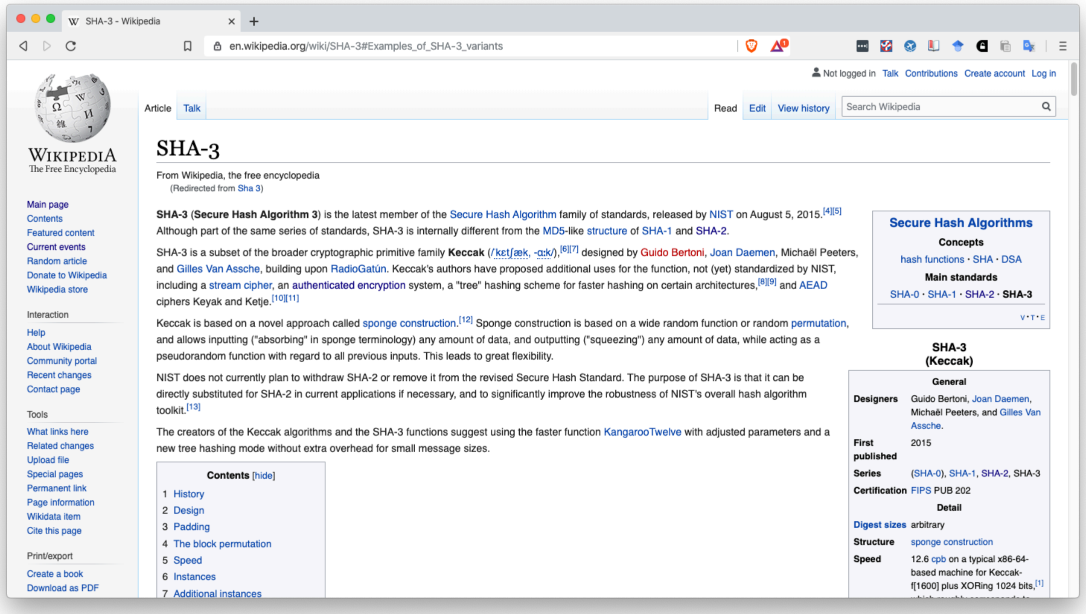
Uniformly random

All agents have (quantum) oracle access to H

$$(x, y) \mapsto (x, y \oplus H(x))$$

The (Q)ROM

Reality



SHA-3

From Wikipedia, the free encyclopedia
(Redirected from Sha 3)

SHA-3 (**Secure Hash Algorithm 3**) is the latest member of the Secure Hash Algorithm family of standards, released by NIST on August 5, 2015.[4][5] Although part of the same series of standards, SHA-3 is internally different from the MD5-like structure of SHA-1 and SHA-2.

SHA-3 is a subset of the broader cryptographic primitive family **Keccak** (/ˈkɛtʃæk, -ɑːk/),[6][7] designed by Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche, building upon RadioGatún. Keccak's authors have proposed additional uses for the function, not (yet) standardized by NIST, including a stream cipher, an authenticated encryption system, a "tree" hashing scheme for faster hashing on certain architectures,[8][9] and AEAD ciphers Keyak and Ketje.[10][11]

Keccak is based on a novel approach called sponge construction.[12] Sponge construction is based on a wide random function or random permutation, and allows inputting ("absorbing" in sponge terminology) any amount of data, and outputting ("squeezing") any amount of data, while acting as a pseudorandom function with regard to all previous inputs. This leads to great flexibility.

NIST does not currently plan to withdraw SHA-2 or remove it from the revised Secure Hash Standard. The purpose of SHA-3 is that it can be directly substituted for SHA-2 in current applications if necessary, and to significantly improve the robustness of NIST's overall hash algorithm toolkit.[13]

The creators of the Keccak algorithms and the SHA-3 functions suggest using the faster function KangarooTwelve with adjusted parameters and a new tree hashing mode without extra overhead for small message sizes.

Contents [hide]

Secure Hash Algorithms

Concepts

hash functions · SHA · DSA

Main standards

SHA-0 · SHA-1 · SHA-2 · **SHA-3**

v · t · e

SHA-3
(Keccak)

General	
Designers	Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.
First published	2015
Series	(SHA-0), SHA-1, SHA-2, SHA-3
Certification	FIPS PUB 202
Detail	
Digest sizes	arbitrary
Structure	sponge construction
Speed	12.6 cpb on a typical x86-64-based machine for Keccak-f[1600] plus XORing 1024 bits,[1]

- Outrageosly optimistic

Model

$H : \{0,1\}^* \to \{0,1\}^n$

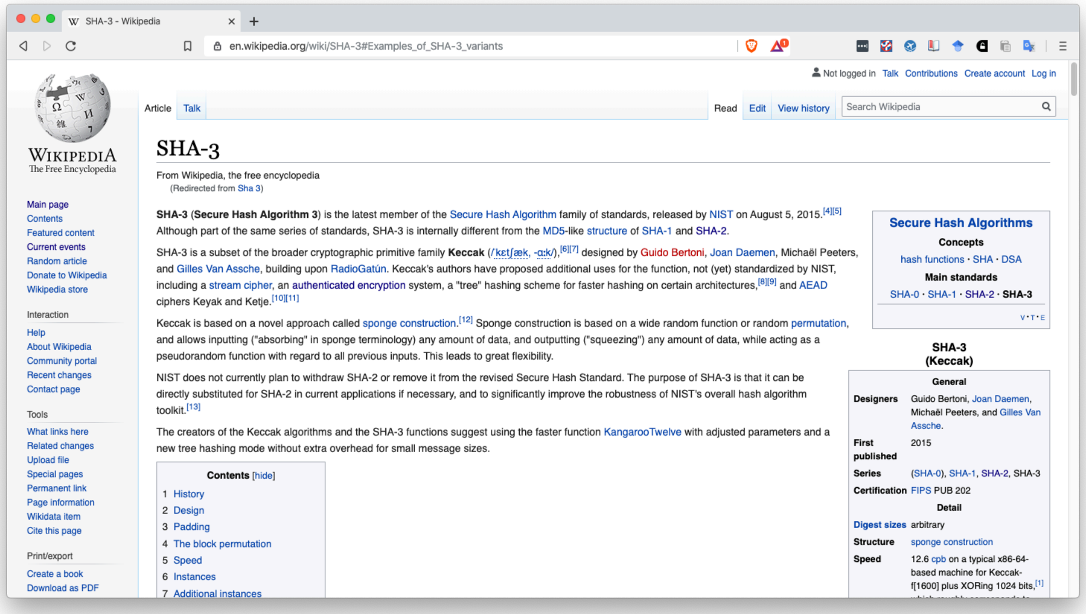
Uniformly random

All agents have (quantum) oracle access to H

$$(x, y) \mapsto (x, y \oplus H(x))$$

The (Q)ROM

Reality



SHA-3

From Wikipedia, the free encyclopedia
(Redirected from Sha 3)

SHA-3 (**Secure Hash Algorithm 3**) is the latest member of the Secure Hash Algorithm family of standards, released by NIST on August 5, 2015.[4][5] Although part of the same series of standards, SHA-3 is internally different from the MD5-like structure of SHA-1 and SHA-2.

SHA-3 is a subset of the broader cryptographic primitive family **Keccak** (/ˈkɛtʃæk, -ɑːk/),[6][7] designed by Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche, building upon RadioGatún. Keccak's authors have proposed additional uses for the function, not (yet) standardized by NIST, including a stream cipher, an authenticated encryption system, a "tree" hashing scheme for faster hashing on certain architectures,[8][9] and AEAD ciphers Keyak and Ketje.[10][11]

Keccak is based on a novel approach called sponge construction.[12] Sponge construction is based on a wide random function or random permutation, and allows inputting ("absorbing" in sponge terminology) any amount of data, and outputting ("squeezing") any amount of data, while acting as a pseudorandom function with regard to all previous inputs. This leads to great flexibility.

NIST does not currently plan to withdraw SHA-2 or remove it from the revised Secure Hash Standard. The purpose of SHA-3 is that it can be directly substituted for SHA-2 in current applications if necessary, and to significantly improve the robustness of NIST's overall hash algorithm toolkit.[13]

The creators of the Keccak algorithms and the SHA-3 functions suggest using the faster function KangarooTwelve with adjusted parameters and a new tree hashing mode without extra overhead for small message sizes.

Secure Hash Algorithms
Concepts
hash functions · SHA · DSA
Main standards
SHA-0 · SHA-1 · SHA-2 · **SHA-3**
V · T · E

SHA-3
(Keccak)

General	
Designers	Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.
First published	2015
Series	(SHA-0), SHA-1, SHA-2, SHA-3
Certification	FIPS PUB 202
Detail	
Digest sizes	arbitrary
Structure	sponge construction
Speed	12.6 cpb on a typical x86-64-based machine for Keccak-f[1600] plus XORing 1024 bits,[1]

- Outrageosly optimistic
- Seems to work in practice

Model

$H : \{0,1\}^* \rightarrow \{0,1\}^n$

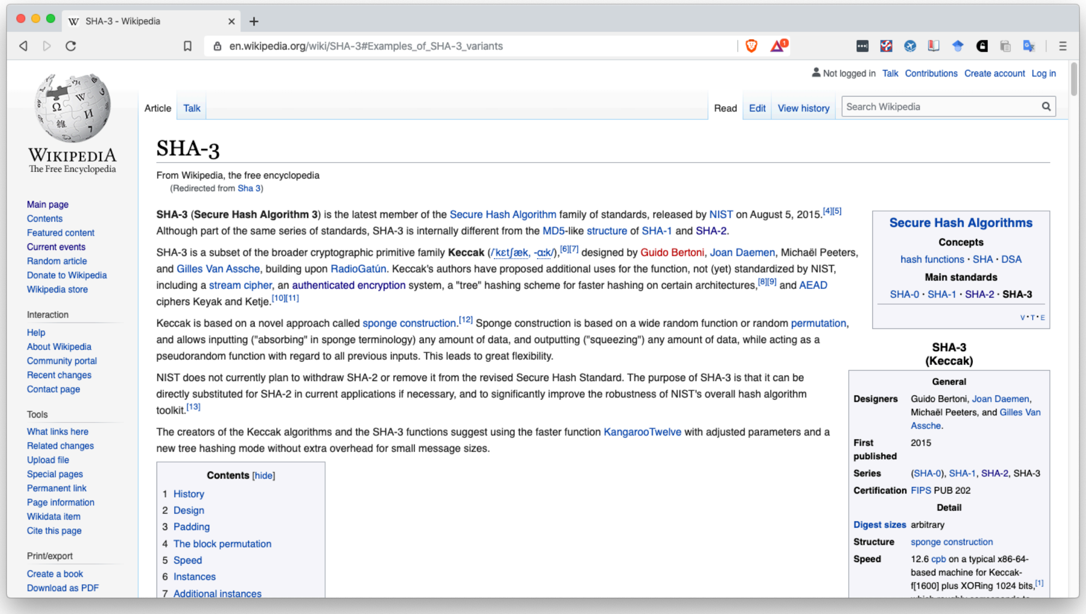
Uniformly random

All agents have (quantum) oracle access to H

$$(x, y) \mapsto (x, y \oplus H(x))$$

The (Q)ROM

Reality



SHA-3

From Wikipedia, the free encyclopedia
(Redirected from Sha 3)

SHA-3 (**Secure Hash Algorithm 3**) is the latest member of the Secure Hash Algorithm family of standards, released by NIST on August 5, 2015.[4][5] Although part of the same series of standards, SHA-3 is internally different from the MD5-like structure of SHA-1 and SHA-2.

SHA-3 is a subset of the broader cryptographic primitive family **Keccak** (/ˈkɛtʃæk, -ɑːk/),[6][7] designed by Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche, building upon RadioGatún. Keccak's authors have proposed additional uses for the function, not (yet) standardized by NIST, including a stream cipher, an authenticated encryption system, a "tree" hashing scheme for faster hashing on certain architectures,[8][9] and AEAD ciphers Keyak and Ketje.[10][11]

Keccak is based on a novel approach called sponge construction.[12] Sponge construction is based on a wide random function or random permutation, and allows inputting ("absorbing" in sponge terminology) any amount of data, and outputting ("squeezing") any amount of data, while acting as a pseudorandom function with regard to all previous inputs. This leads to great flexibility.

NIST does not currently plan to withdraw SHA-2 or remove it from the revised Secure Hash Standard. The purpose of SHA-3 is that it can be directly substituted for SHA-2 in current applications if necessary, and to significantly improve the robustness of NIST's overall hash algorithm toolkit.[13]

The creators of the Keccak algorithms and the SHA-3 functions suggest using the faster function KangarooTwelve with adjusted parameters and a new tree hashing mode without extra overhead for small message sizes.

Contents [hide]

Secure Hash Algorithms

Concepts

hash functions · SHA · DSA

Main standards

SHA-0 · SHA-1 · SHA-2 · **SHA-3**

V · T · E

SHA-3
(Keccak)

General	
Designers	Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.
First published	2015
Series	(SHA-0), SHA-1, SHA-2, SHA-3
Certification	FIPS PUB 202
Detail	
Digest sizes	arbitrary
Structure	sponge construction
Speed	12.6 cpb on a typical x86-64-based machine for Keccak-f[1600] plus XORing 1024 bits,[1]

Model

$H : \{0,1\}^* \to \{0,1\}^n$

Uniformly random

All agents have (quantum) oracle access to H

$$(x, y) \mapsto (x, y \oplus H(x))$$

- Outrageosly optimistic
- Seems to work in practice
- Enables very efficient crypto

QROM challenges

QROM challenges

ROM techniques:

1. Query transcripts

QROM challenges

ROM techniques:

1. Query transcripts
2. Rewinding

QROM challenges

ROM techniques:

1. Query transcripts
2. Rewinding
3. Reprogramming

QROM challenges

ROM techniques:

1. Query transcripts
2. Rewinding
3. Reprogramming

Quantum theory makes things difficult! No-cloning, Measurement disturbance

QROM challenges

ROM techniques:

1. Query transcripts
2. Rewinding
3. Reprogramming

Quantum theory makes things difficult! No-cloning, Measurement disturbance

QROM:

QROM challenges

ROM techniques:

1. Query transcripts
2. Rewinding
3. Reprogramming

Quantum theory makes things difficult! No-cloning, Measurement disturbance

QROM:

1. ~~Query transcripts~~

QROM challenges

ROM techniques:

1. Query transcripts
2. Rewinding
3. Reprogramming

Quantum theory makes things difficult! No-cloning, Measurement disturbance

QROM:

1. ~~Query transcripts~~
2. ~~Rewinding~~ $\Longrightarrow$ 3 specialized rewinding techniques that don't cover all applications

QROM challenges

ROM techniques:

1. Query transcripts
2. Rewinding
3. Reprogramming

Quantum theory makes things difficult! No-cloning, Measurement disturbance

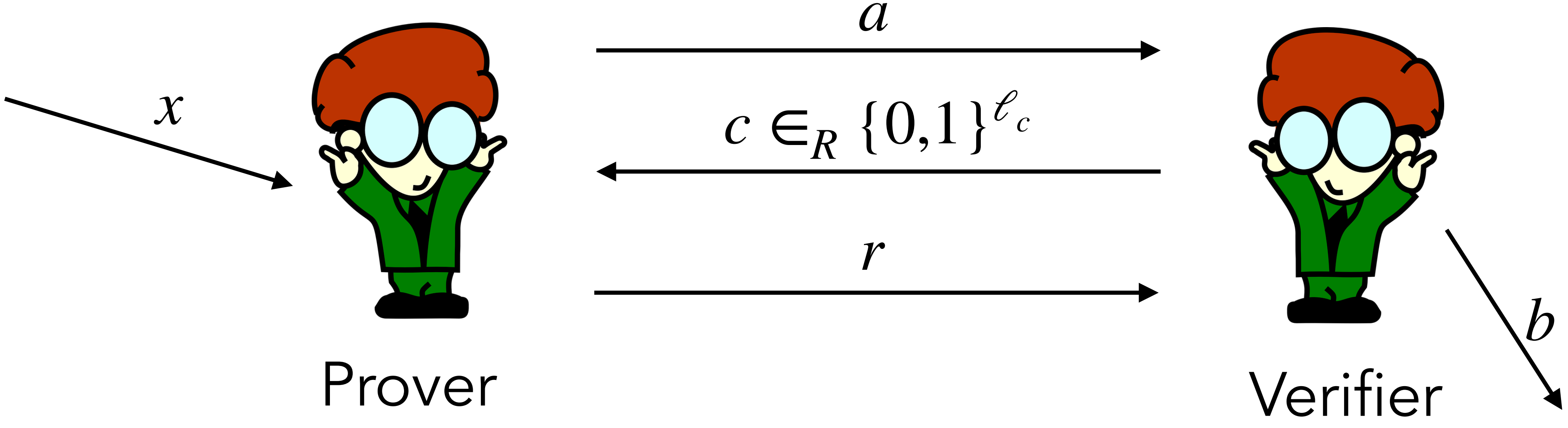
QROM:

1. ~~Query transcripts~~
2. ~~Rewinding~~ $\Longrightarrow$ 3 specialized rewinding techniques that don't cover all applications
3. Reprogramming: Sure, if you know how without 1. and 2.

The Fiat Shamir transformation

The Fiat Shamir transformation

Σ-protocol: Interactive proof system



x

a

$c \in_R \{0,1\}^{\ell_c}$

r

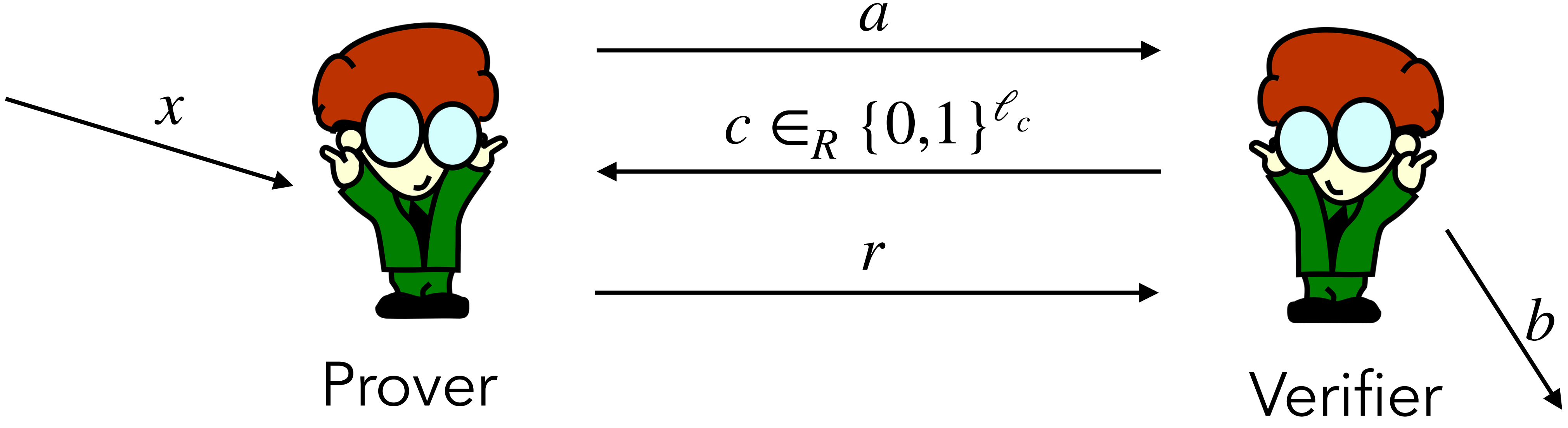
b

Prover

Verifier

The Fiat Shamir transformation

Σ-protocol: Interactive proof system



x

a

$c \in_R \{0,1\}^{\ell_c}$

r

b

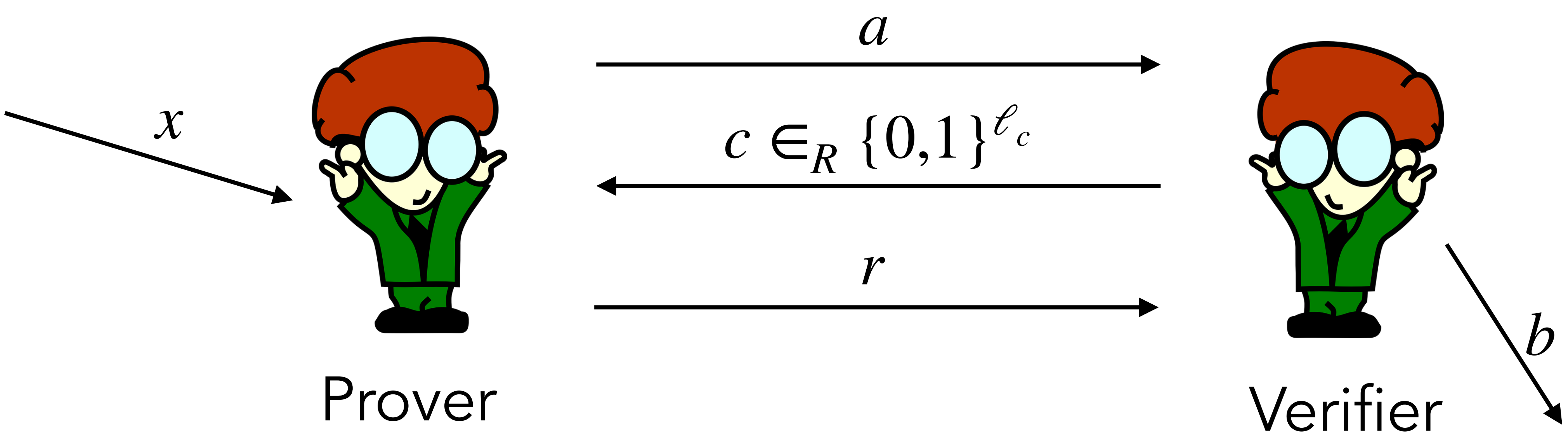
Prover

Verifier

Fiat Shamir (FS) transformation: $c = H(x, a)$

The Fiat Shamir transformation

Σ-protocol: Interactive proof system



x

a

$c \in_R \{0,1\}^{\ell_c}$

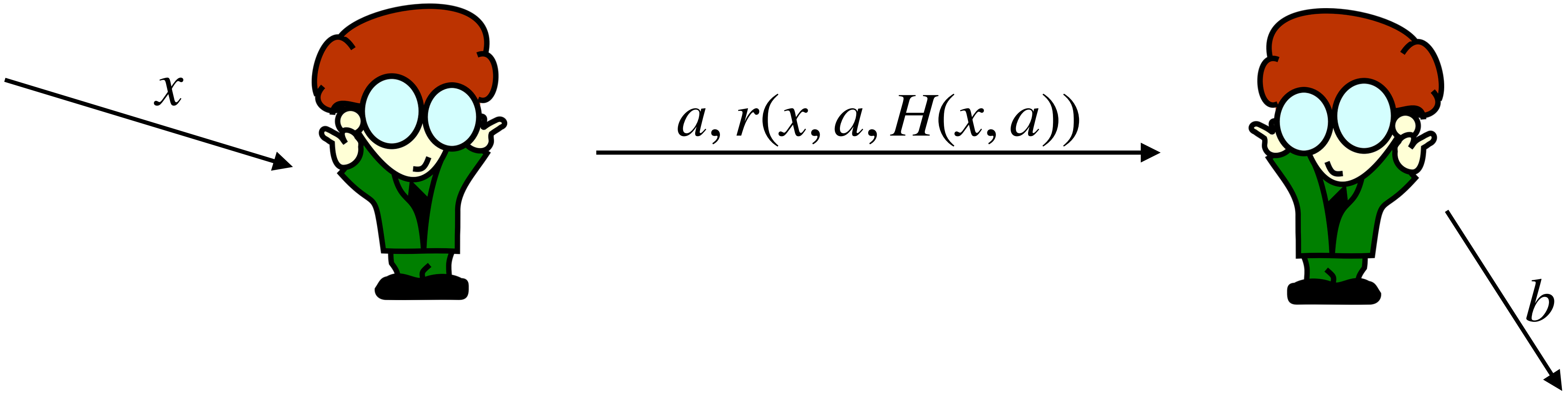
r

b

Prover

Verifier

Fiat Shamir (FS) transformation: $c = H(x, a)$



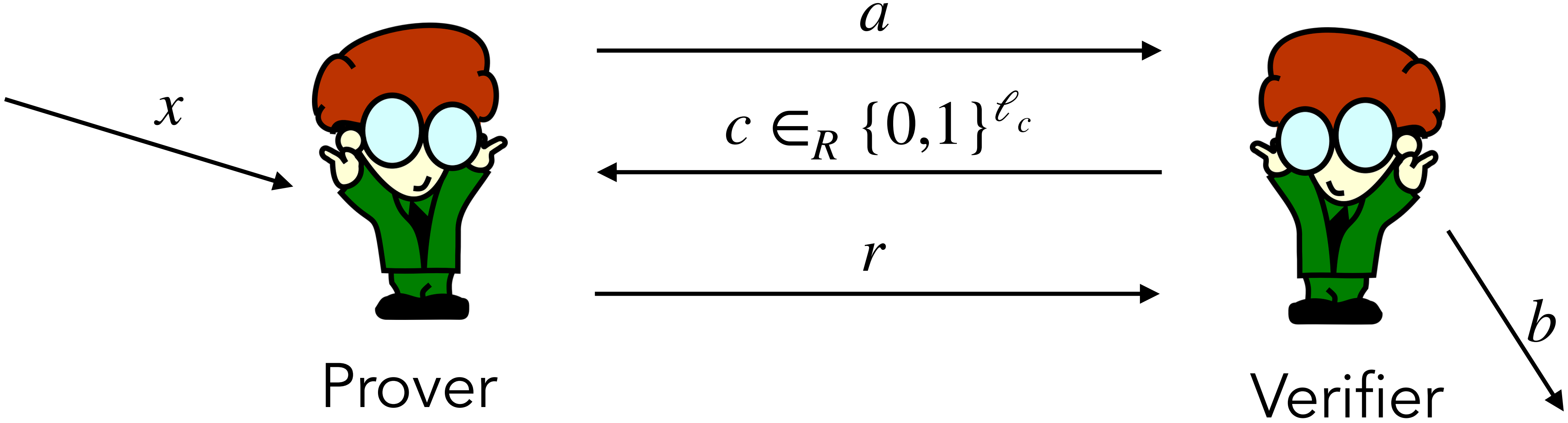
x

$a, r(x, a, H(x, a))$

b

The Fiat Shamir transformation

Σ-protocol: Interactive proof system



x

a

$c \in_R \{0,1\}^{\ell_c}$

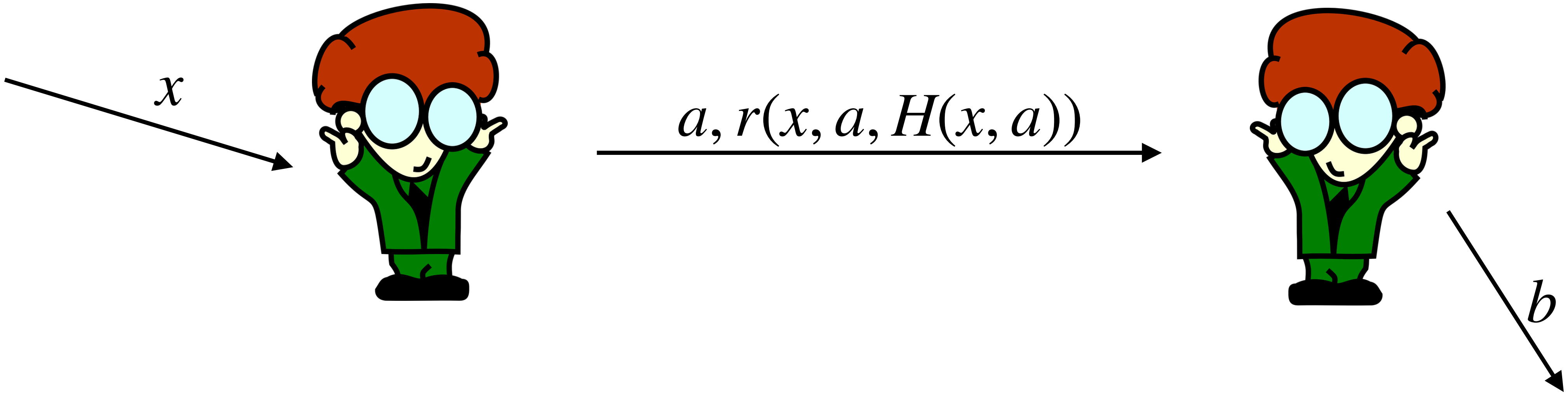
r

b

Prover

Verifier

Fiat Shamir (FS) transformation: $c = H(x, a)$



x

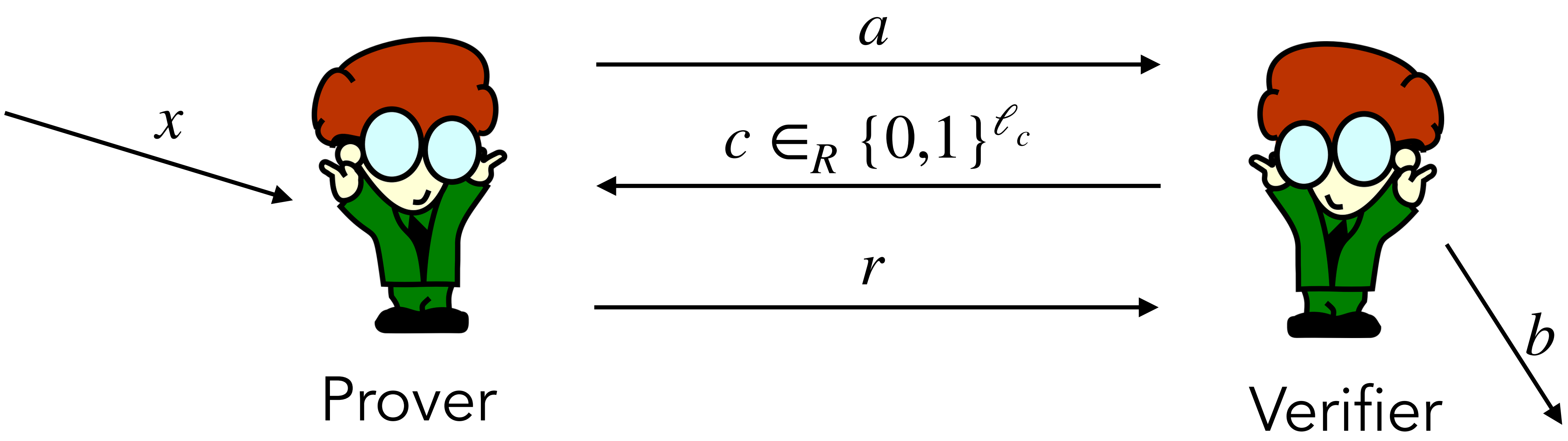
$a, r(x, a, H(x, a))$

b

non-interactive!!!

The Fiat Shamir transformation

Σ-protocol: Interactive proof system



x

a

$c \in_R \{0,1\}^{\ell_c}$

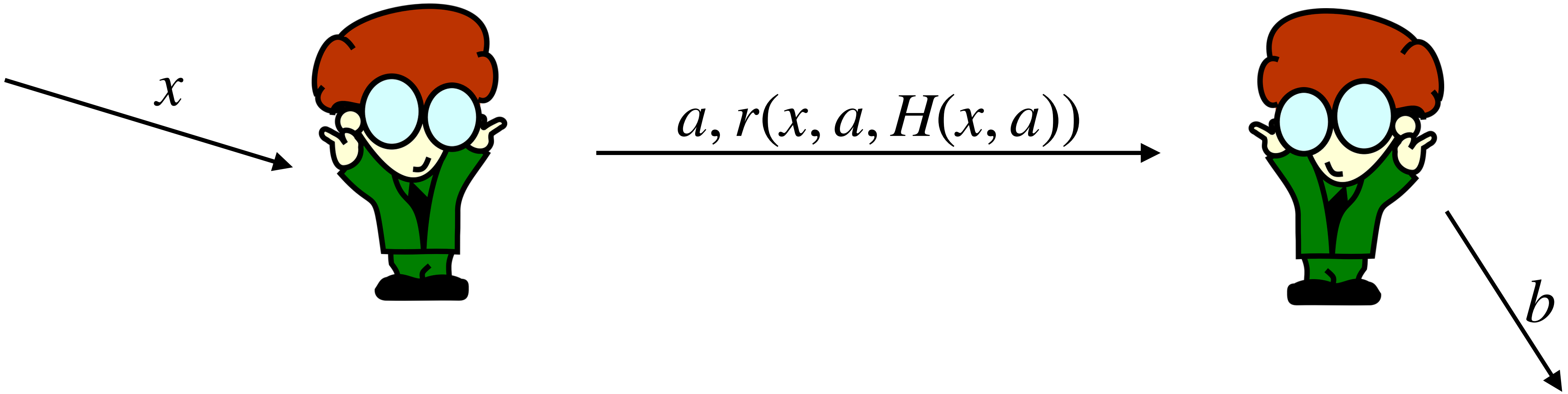
r

b

Prover

Verifier

Fiat Shamir (FS) transformation: $c = H(x, a)$



x

$a, r(x, a, H(x, a))$

b

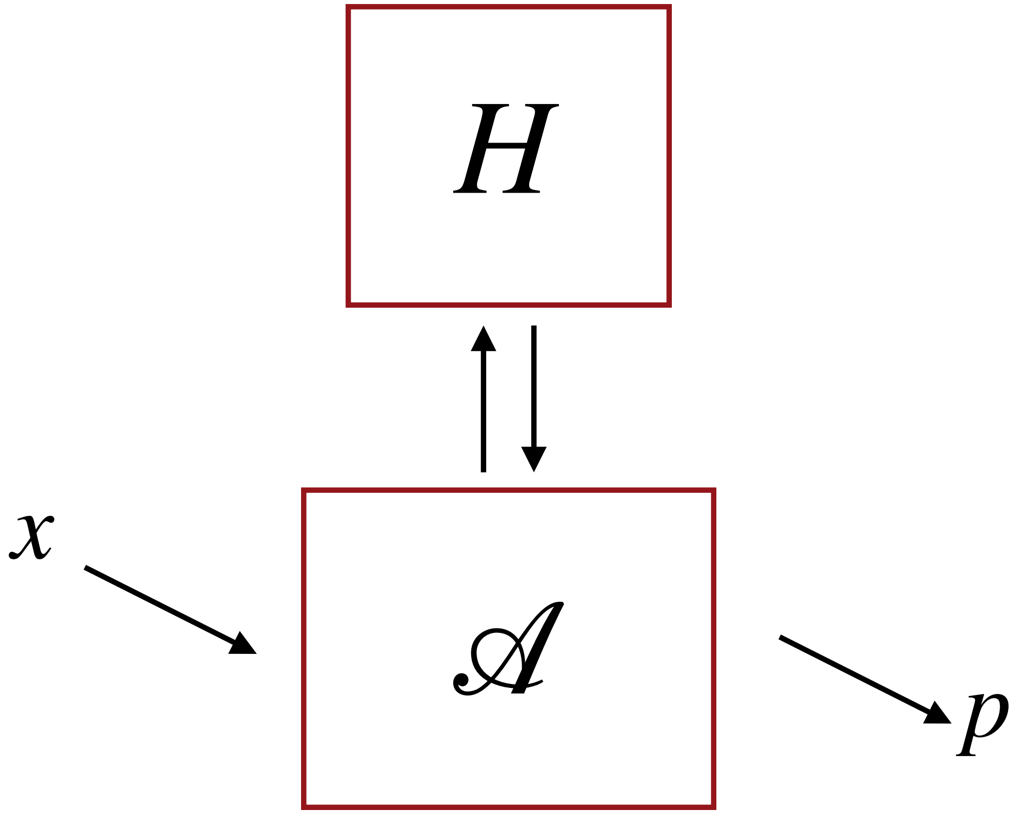
non-interactive!!! ⟹used for efficient digital signatures

ROM security

The FS transformation is secure in the ROM (Pointcheval, Stern 96):

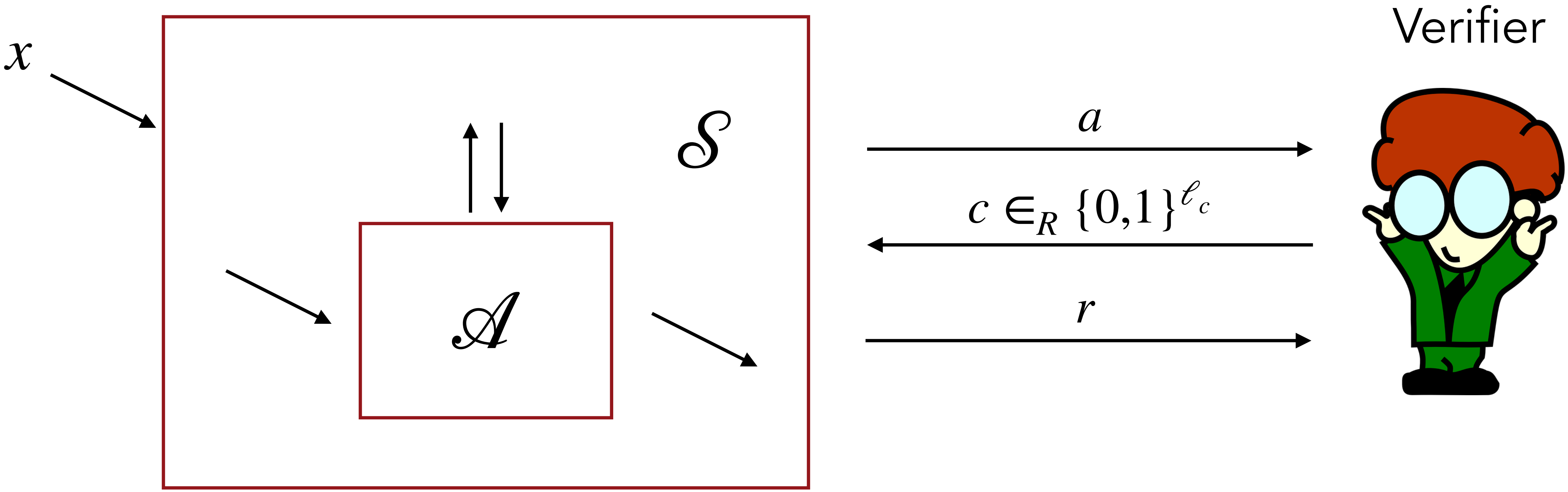
ROM security

The FS transformation is secure in the ROM (Pointcheval, Stern 96):



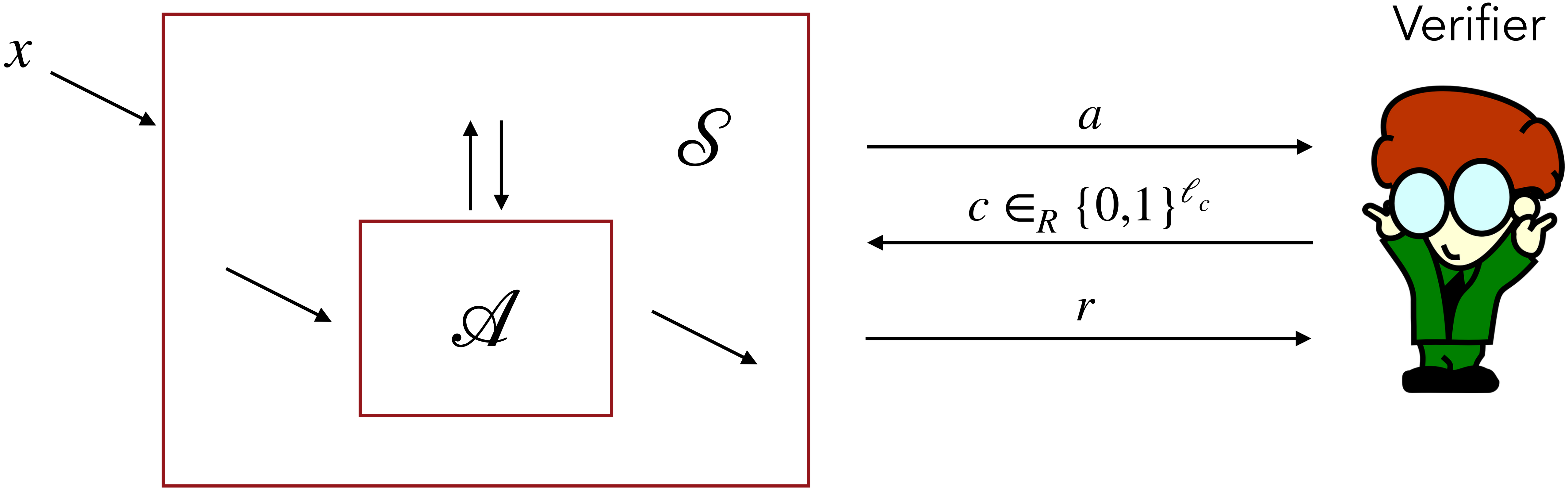
ROM security

The FS transformation is secure in the ROM (Pointcheval, Stern 96):



ROM security

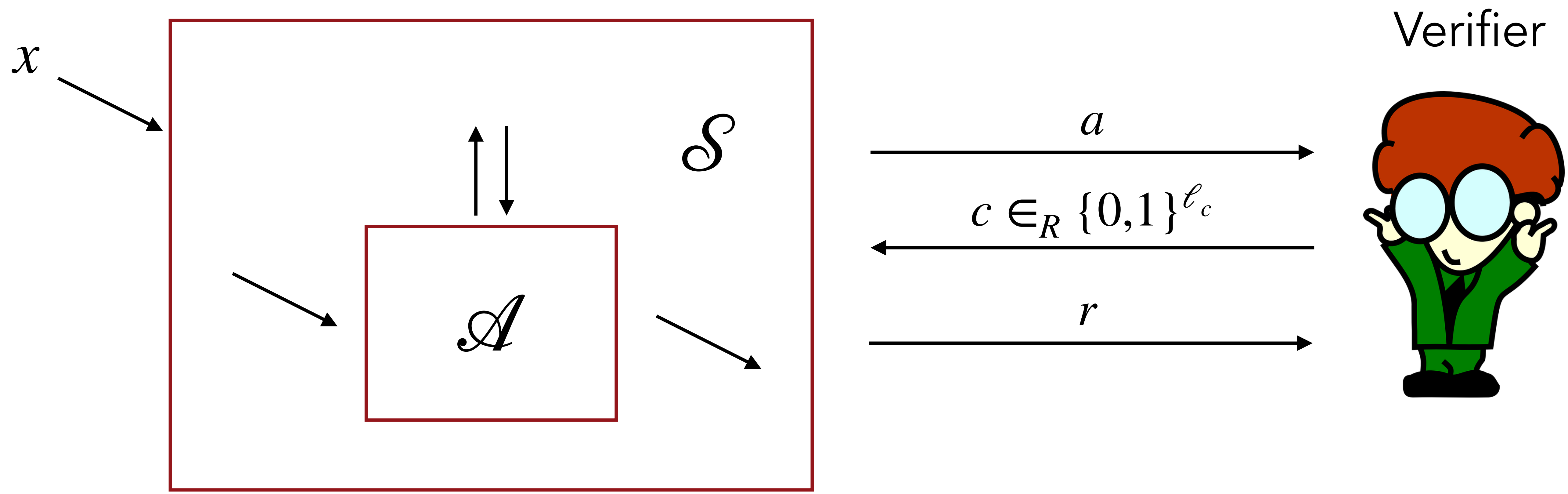
The FS transformation is secure in the ROM (Pointcheval, Stern 96):



Success probability: $\varepsilon(\mathscr{S}[\mathscr{A}]) \geq \dfrac{\varepsilon(\mathscr{A})}{O(q)}$

ROM security

The FS transformation is secure in the ROM (Pointcheval, Stern 96):

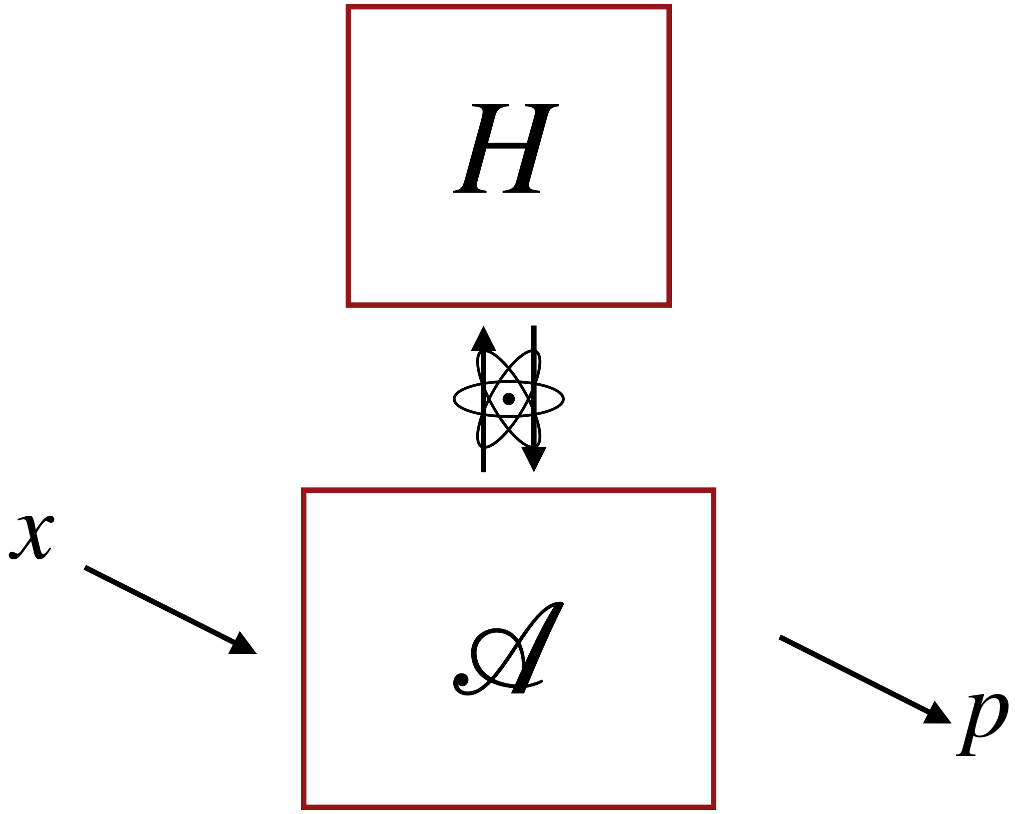


Success probability: $\varepsilon(\mathcal{S}[\mathcal{A}]) \geq \dfrac{\varepsilon(\mathcal{A})}{O(q)}$

# of queries $\mathcal{A}$ makes to H

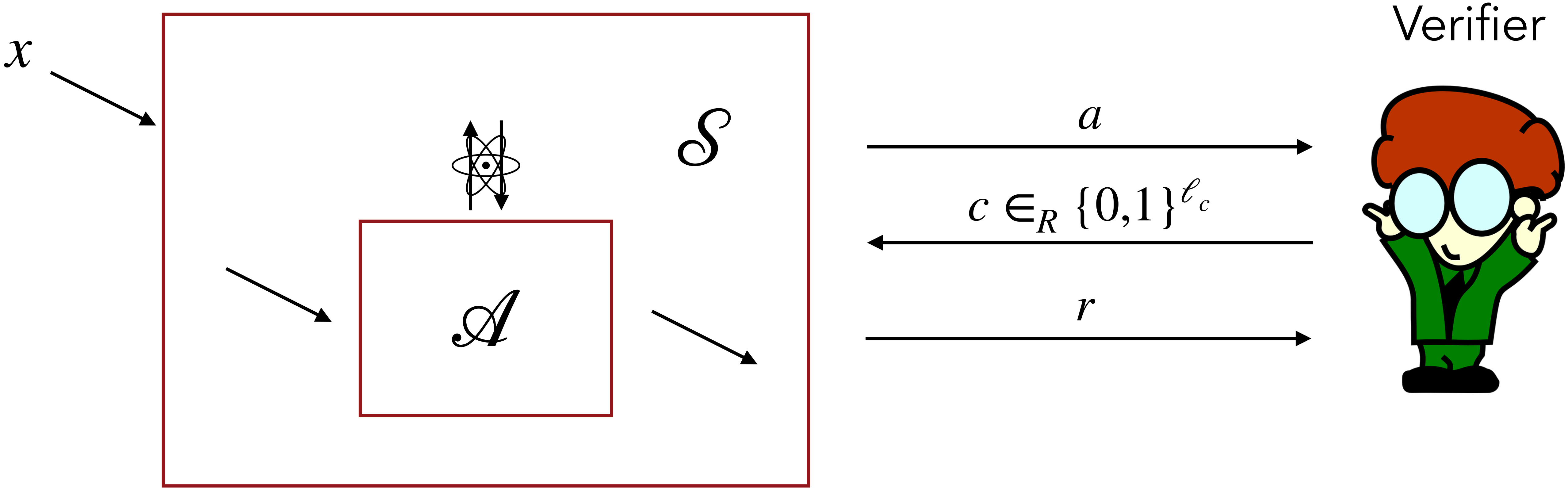
QROM security

The FS transformation is secure in the QROM (Don, Fehr, M, Schaffner ’19):



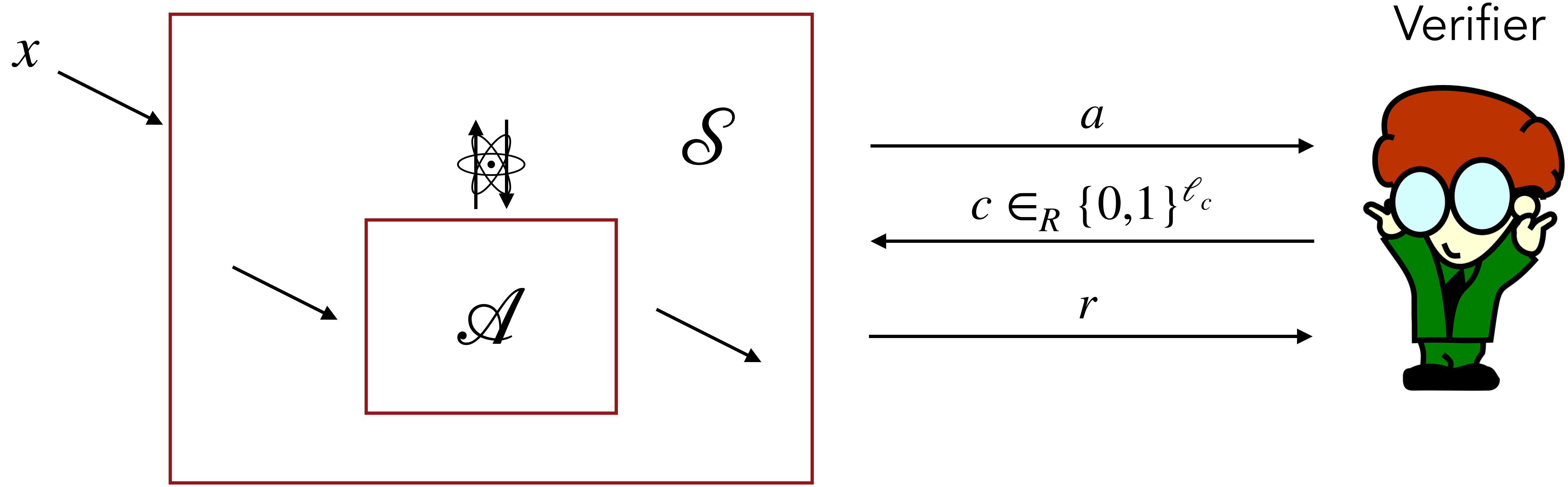
QROM security

The FS transformation is secure in the QROM (Don, Fehr, M, Schaffner ’19):



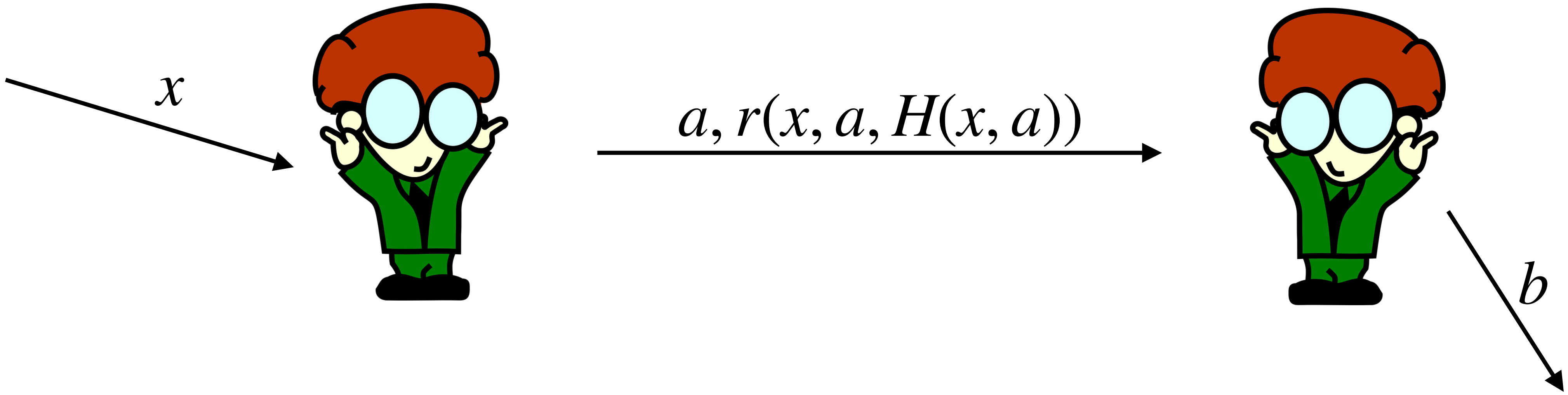
QROM security

The FS transformation is secure in the QROM (Don, Fehr, M, Schaffner '19):

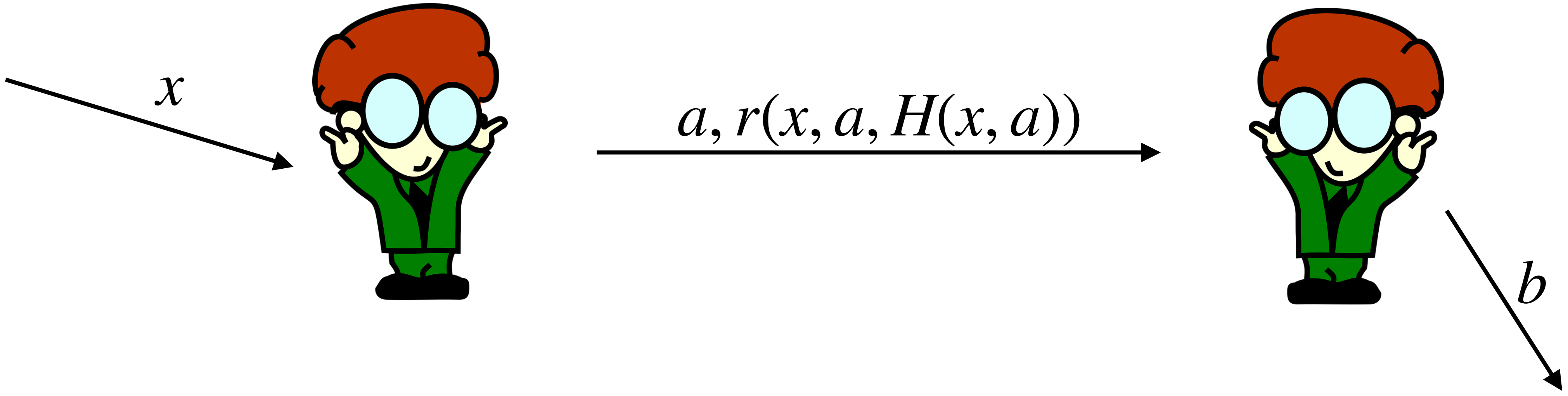


Success probability: $\varepsilon(\mathscr{S}[\mathscr{A}]) \geq \dfrac{\varepsilon(\mathscr{A})}{O(q^2)}$

Technique



Technique



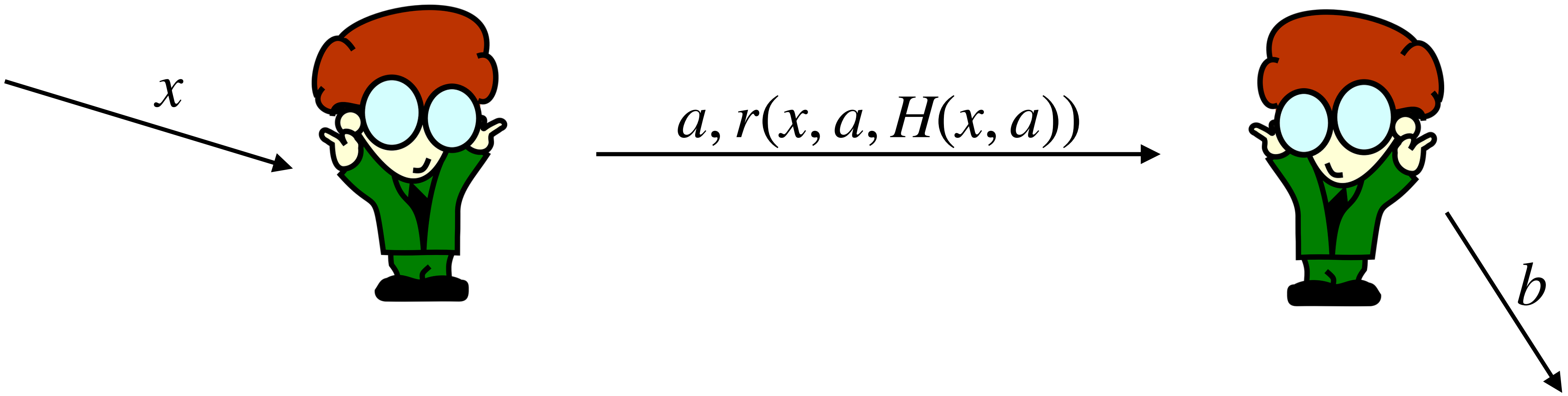
x

$a, r(x, a, H(x, a))$

b

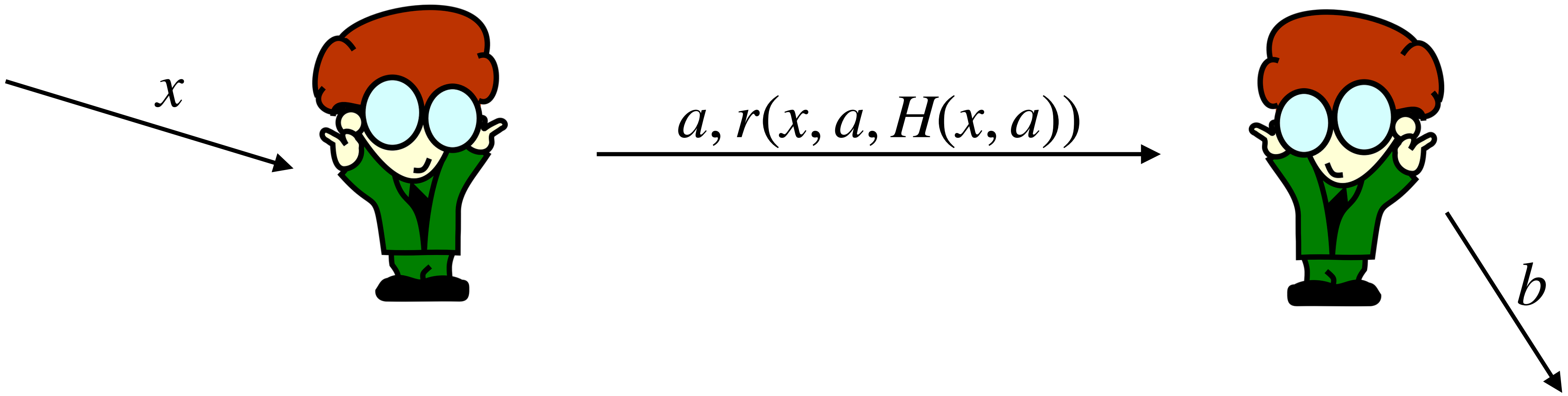
Suppose r was injective $\implies$ $\mathscr{A}$ essentially needs to classically query H on (x, a).

Technique



Suppose r was injective $\implies$ $\mathscr{A}$ essentially needs to classically query H on (x, a).

Technique



Suppose r was injective $\Longrightarrow$ $\mathscr{A}$ essentially needs to classically query H on (x, a).

Measure-and-Reprogram: Pick a random query, measure it and reprogram with c from the Σ-protocol.

Long term goal

Popular belief about QROM: Grover speed-up is as good as it gets.

Long term goal

Popular belief about QROM: Grover speed-up is as good as it gets.

$\Longrightarrow$ Dream: QROM-to-ROM reduction should solve all our problems!

Long term goal

Popular belief about QROM: Grover speed-up is as good as it gets.

$\Longrightarrow$ Dream: QROM-to-ROM reduction should solve all our problems!

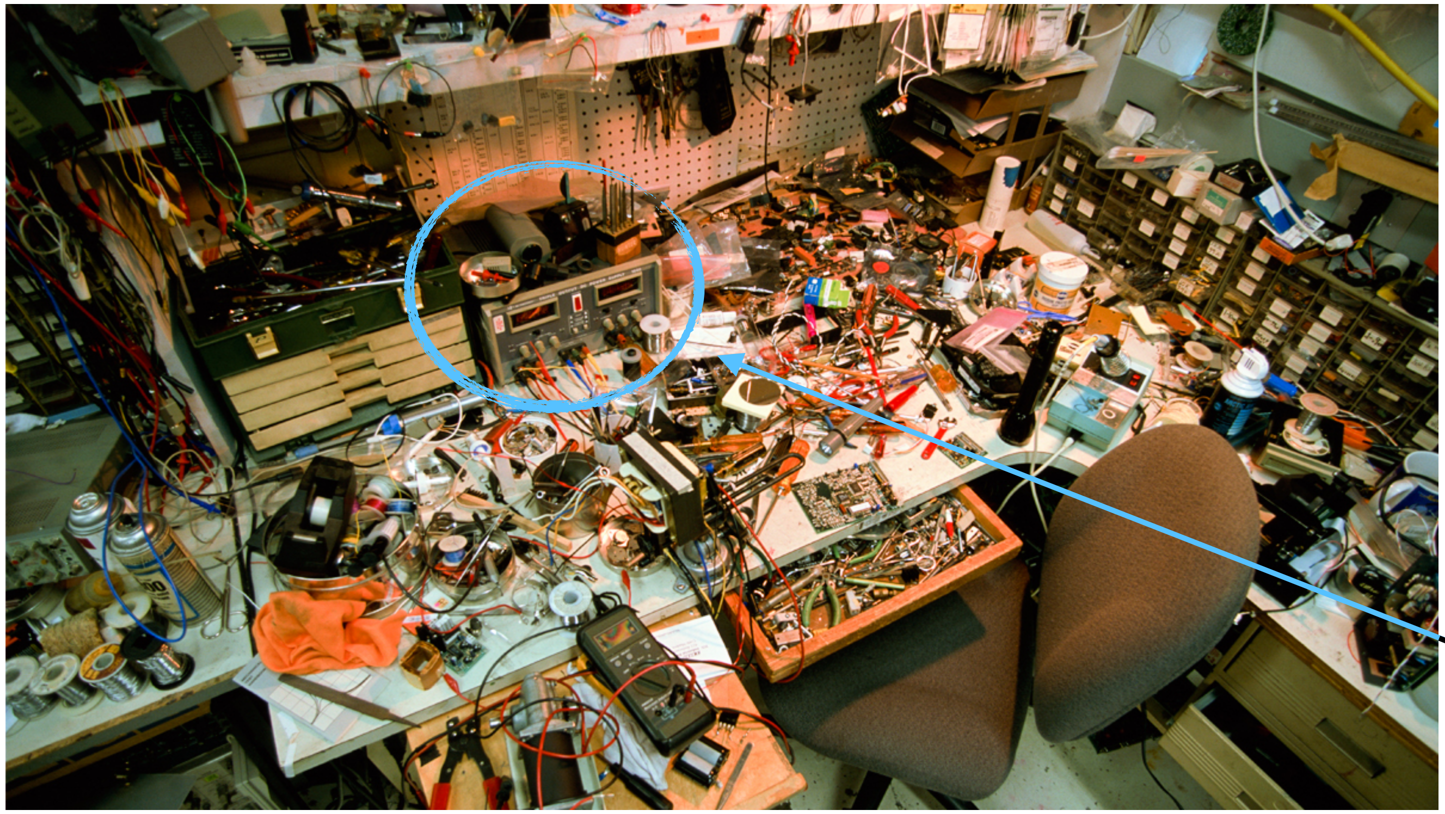
Reality:

Long term goal

Popular belief about QROM: Grover speed-up is as good as it gets.

$\Longrightarrow$ Dream: QROM-to-ROM reduction should solve all our problems!

Reality:



QROM security of FS

Summary

- The (Q)ROM is extremely useful for efficient cryptography
- Quantum theory complicates things, much less coherent picture of QROM security
- Important cases solved, e.g. Fiat Shamir
- General reduction from QROM to ROM would be nice to have!