Selected Areas in Cryptology
Cryptanalysis
Week 6

Marc Stevens

stevens@cwi.nl

https://homepages.cwi.nl/~stevens/mastermath/

Asymmetric from symmetric cryptography

- Can we build asymmetric cryptography from symmetric cryptography?

- Benefits:
  - Symmetric cryptography seems generally to resist quantum cryptanalysis
  - No number-theoretic assumptions needed

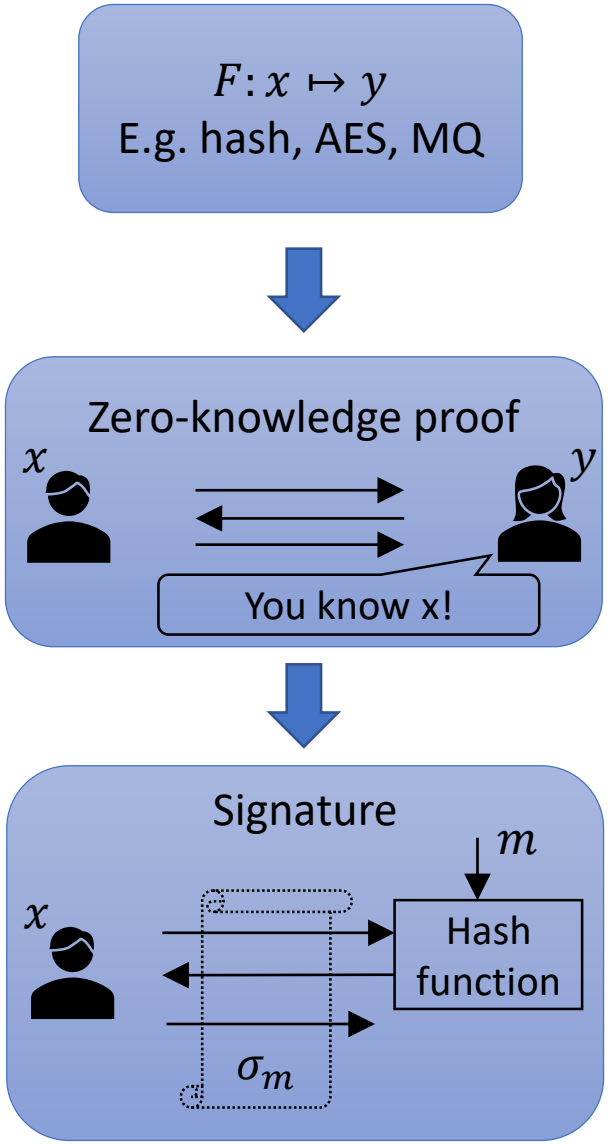
- This week:
  - Hash-based signatures (continued): making schemes more practical
  - MPC-in-the-head on symmetric cryptography

Signatures from Symmetric Crypto

- Consider any one-way function F
  - Hash function: $F(x) = f(x)$
  - Block cipher: $F(x) = Enc_x(0)$
  - MQ system: $F(\vec{x}) = \left(p_1(\vec{x}), \dots, p_k(\vec{x})\right)$, where $p_i(\vec{x}) \in F_q[x_1, \dots, x_n]$

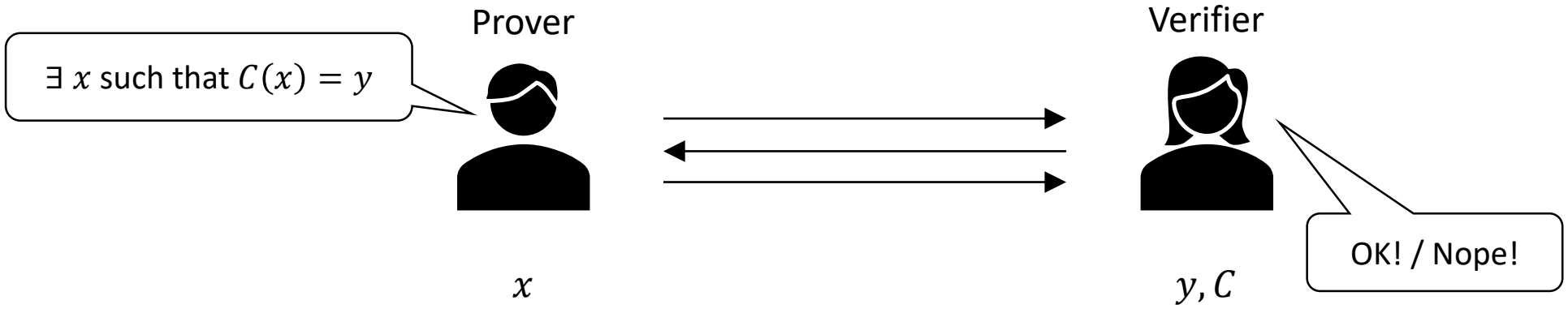
- Consider a protocol P for a circuit C_y:
  - Protocol to prove a Prover knows a secret witness x such that $C_y(x) = 1$
  - Instantiation $C_y(x) = F(x) =? y$ for secret x and public $y = F(x)$
  - Protocols can be made non-interactive using hash function call

- Combine F and P into a signature scheme:
  - Add message to hash function call
  - ⇒ binds non-interactive proof to message
  - The non-interactive proof proves signer knows secret x for public key y



Interactive Proofs

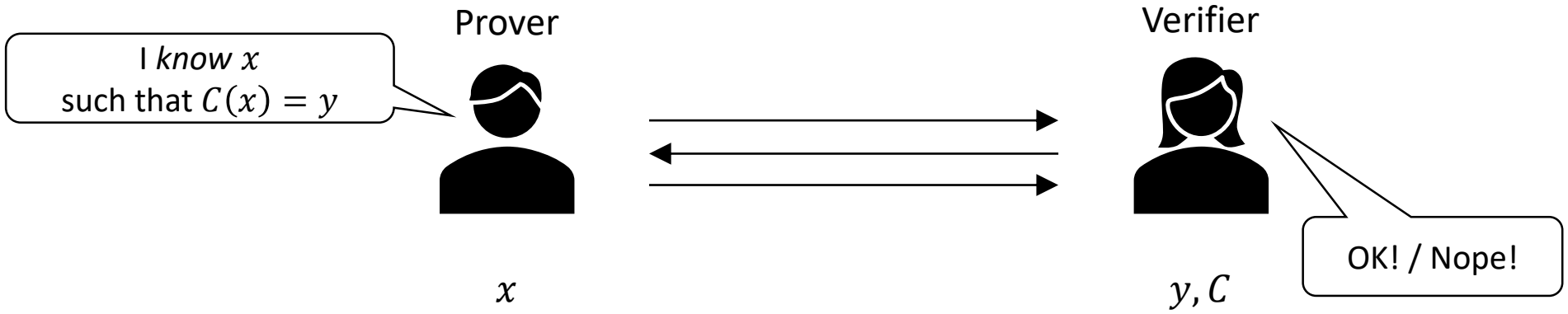
- Multi-round protocol to prove a statement



- C, y define the *statement*, x is called the *witness*
- Security Properties
  - Completeness : $\Pr[\text{ OK! } | \exists\ x\ s.t.\ C(x) = y] = 1$
  - Soundness : $\Pr[\text{ OK! } | \neg\exists\ x\ s.t.\ C(x) = y] \leq \epsilon$ ← soundness error
  - Soundness amplification: repeat protocol n times ⇒ soundness error ϵ^n

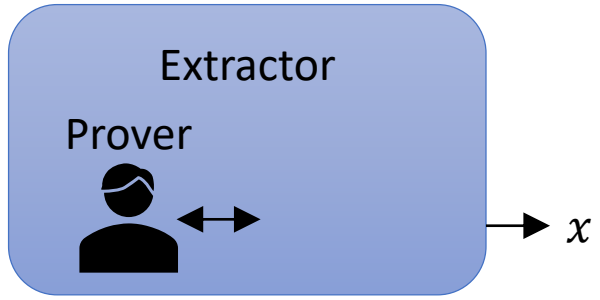
Interactive Proofs of Knowledge (PoK)

- Multi-round protocol to prove *knowledge*



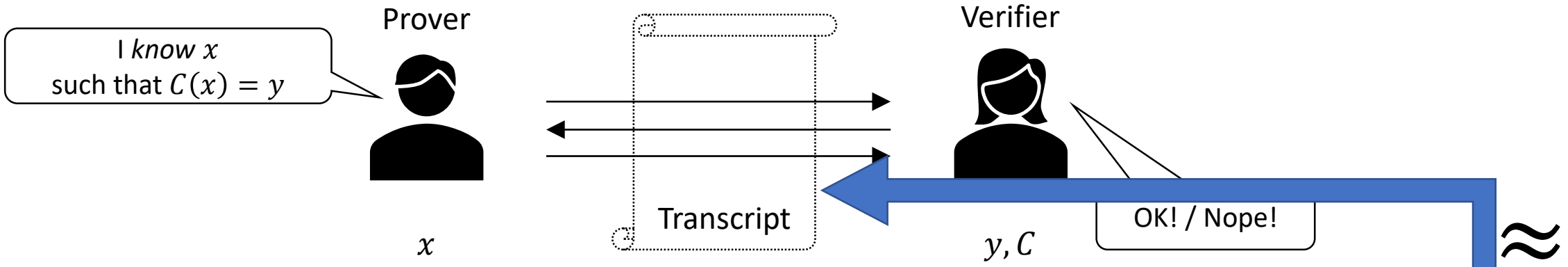
- How to define *knowledge soundness*?
  - Informal: $\Pr[\text{ OK! } | \text{ Prover doesn't know } x \; s.t. \; C(x) = y] \leq \epsilon$

- Formally: prove there exists an Extractor such that
  - The Extractor can extract x from any prover P with $\Pr[\text{ OK! }] > \epsilon$
  - Thus for any prover P that doesn't know x we have $\Pr[\text{ OK! }] \leq \epsilon$

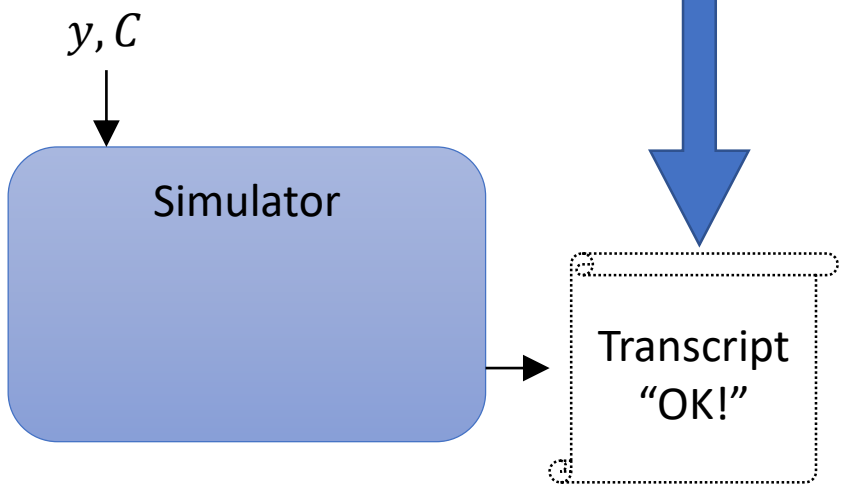


Interactive Zero-Knowledge (ZK) PoK

- Multi-round protocol to prove *knowledge*

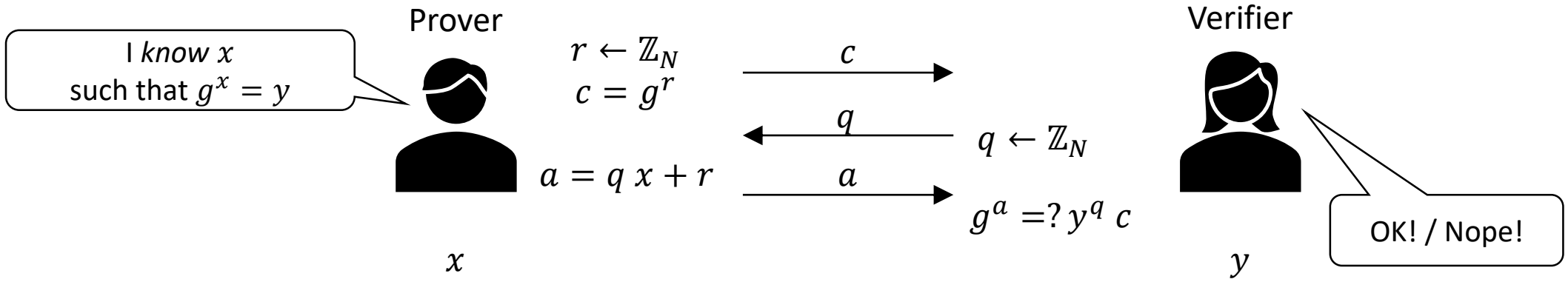


- Trivial protocol: Prover simply sends x
- But we don't want that: x is the secret key!
- We want a *Zero Knowledge* Interactive Proof of Knowledge:
  - Informally: Verifier learns *nothing* about x
- Formally: prove there exists a Simulator that
  - Given y, C produces Transcripts
    - where the Verifier accepts
    - That are indistinguishable from actual valid Transcripts

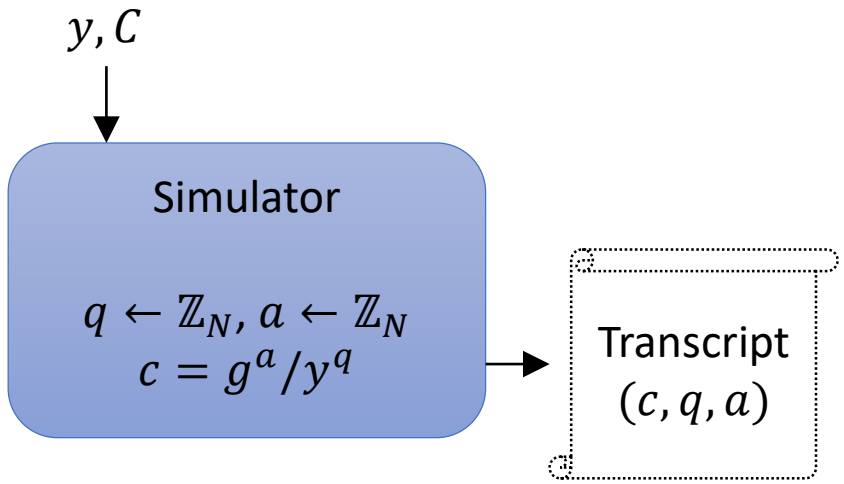


Example Interactive ZK-PoK

- Consider secure Elliptic Curve with generator g of order N



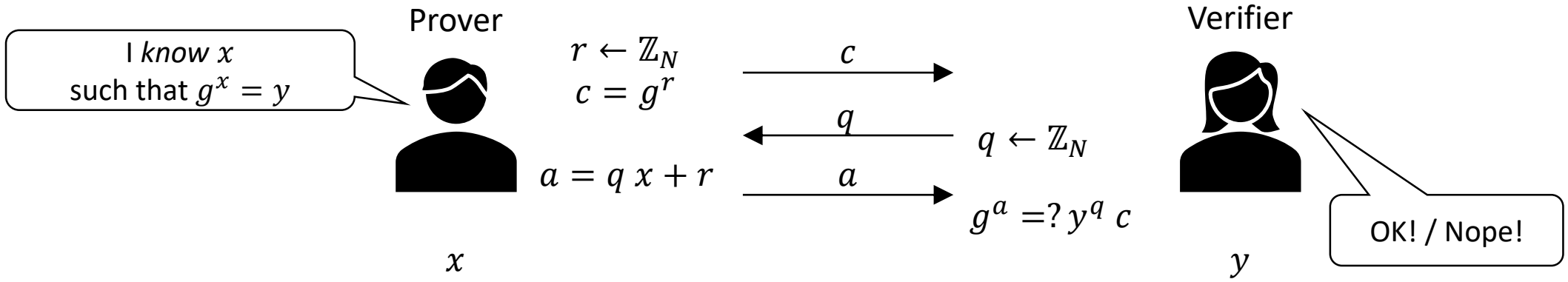
- Zero-Knowledge Simulator:
  - $q \leftarrow \mathbb{Z}_N, a \leftarrow \mathbb{Z}_N$
  - Output (c, q, a) with $c = g^a / y^q$
- Note that Transcript distributions match *perfectly*
  - Given q there is a bijection between a and c
  - a uniform random ⇒ c uniform random
  - q uniform random over $\mathbb{Z}_N$
  - $g^a = y^q \cdot c$



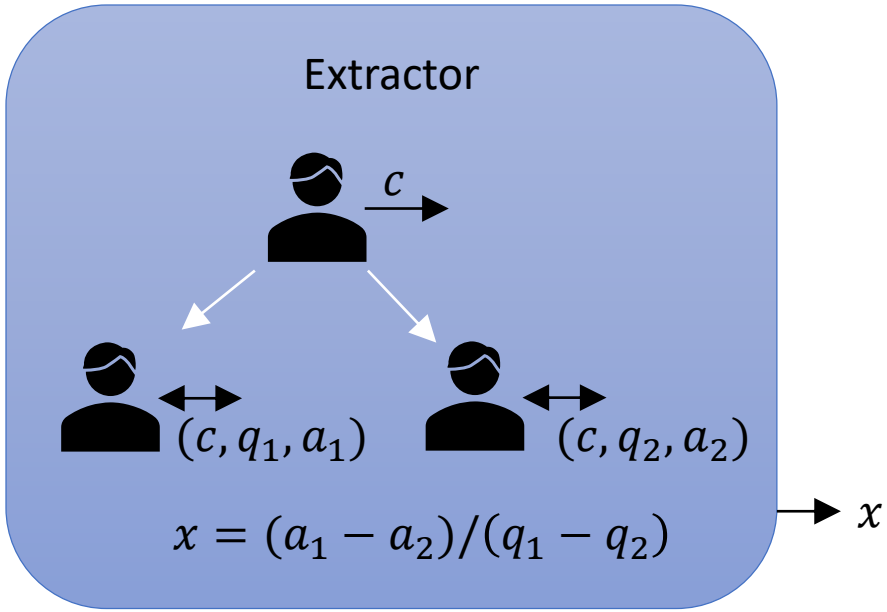
- ⇒ *Perfect* zero-knowledge
  - (vs statistical / computational in case of negligible statistical distance / indistinguishability by any PPT A)

Example Interactive ZK-PoK

- Consider secure Elliptic Curve with generator g of order N

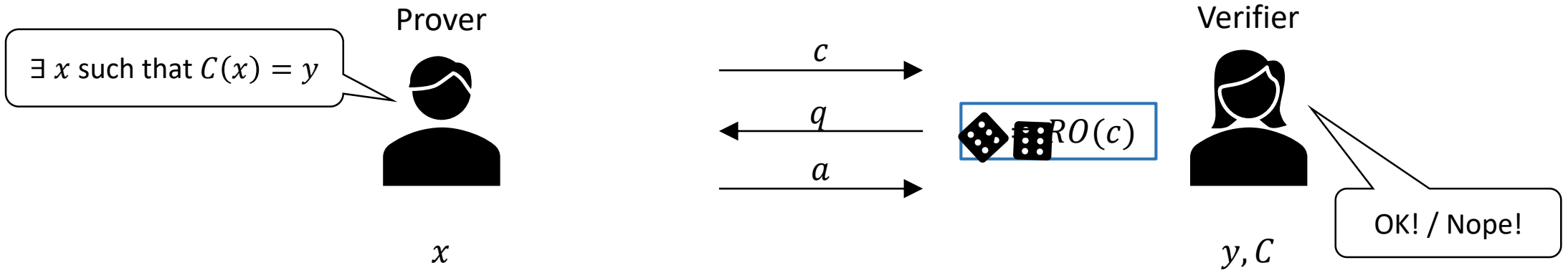


- Knowledge Soundness Extractor:
  - Wait till Prover P outputs c
  - Now *clone* P into P_1 and P_2
  - Send q_1 to P_1 and q_2 to P_2, with $q_2 \neq q_1$
  - Valid answers a_1 and a_2 give $a_1 - a_2 = (q_1 - q_2) \cdot x$
  - Called *2-Special Knowledge Soundness*
- For any c, if Prover can produce answers to 2 distinct queries then extractor obtains x !
- Note that adversary successful for any y is discrete log oracle!



Fiat-Shamir Transform

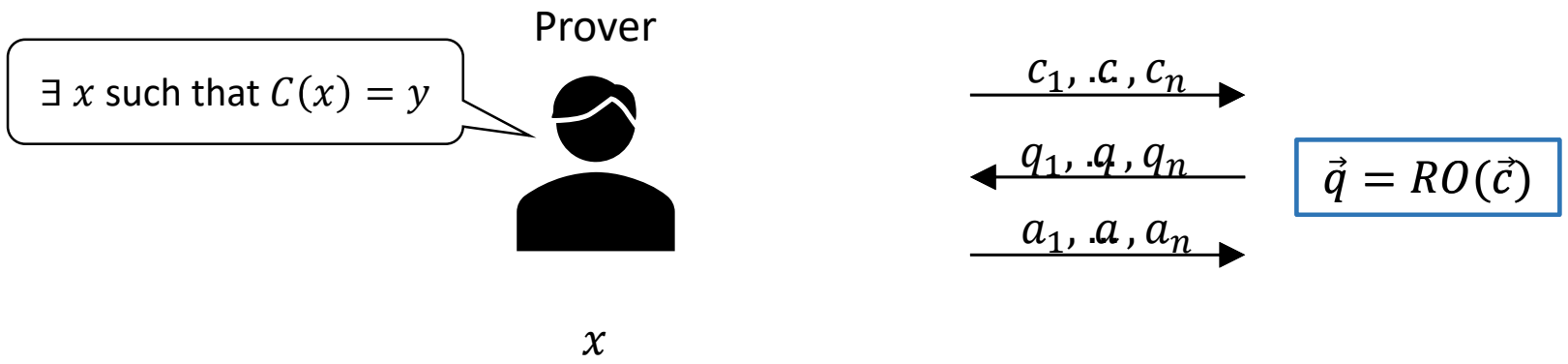
- Given any multi-round protocol with *public-coin* verifier



- Public-coin verifier:
  - Messages from the verifier contain only public coin tosses: uniform random bits

- Fiat-Shamir Transform:
  - Replace Verifier with public coin tosses with Random Oracle (instantiated with hash function)
  - Prover can generate transcripts $\pi = (c, q, a)$ without knowing random q before committed to c
  - *Non-Interactive proof*: π can be made public and verified by any verifier

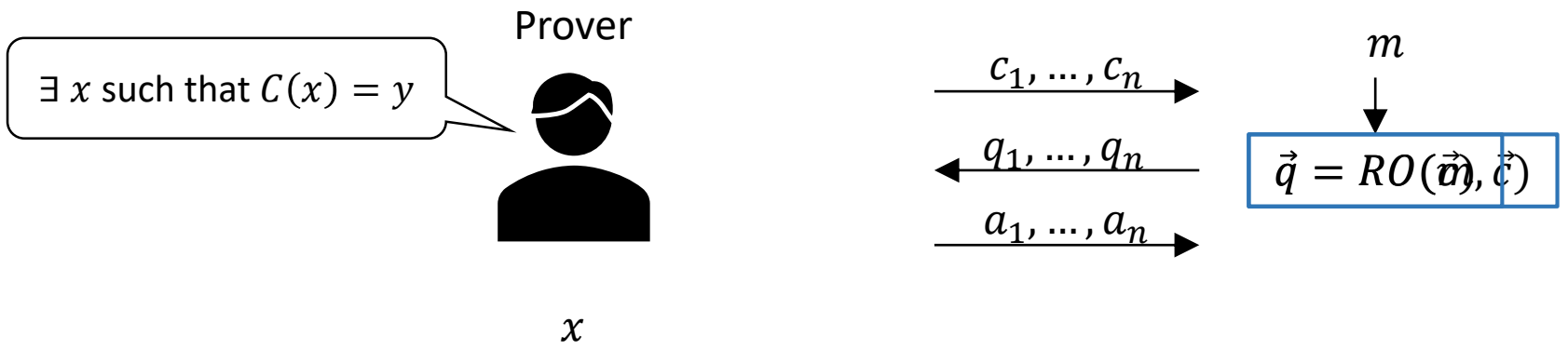
Fiat-Shamir Soundness Amplification



- Interactive ZK-PoK Soundness Amplification
  - Repeat protocol with soundness error ϵ a total of n times
  - $\Rightarrow$ soundness error $\leq \epsilon^n$
- Non-interactive ZK-PoK Forgery
  - An adversary A can attempt many instances until it finds a proof that verifies properly
  - Generating n proofs $\pi_1, \ldots, \pi_n$ independently costs only a factor n more !!
- Non-interactive ZK-PoK Soundness Amplification
  - Instead, to amplify soundness, the n protocol executions should be done together
  - The verifier's queries should depend on all commitments: $q_1, \ldots, q_n \leftarrow RO(c_1, \ldots, c_n)$
  - Now all instances need succeed to generate a valid proof $\pi_{[1,n]}$: soundness error $\leq \epsilon^n$

Signatures from non-interactive ZK-PoK

- Consider non-interactive ZK-PoK



- Transformation into signature scheme:
  - Simply add message m to sign to *RO* input
  - Private key: x
  - Public key: $y = C(x)$
  - Signature = transcript $\pi_{[1,n]}$ which can be compressed:
    - $(q_1, \dots, q_n)$ can be omitted as they can be computed by both signer & verifier
    - Thus $\sigma = (c_1, \dots, c_n, a_1, \dots, a_n)$
  - Signing & Verifying is straightforward from non-interactive ZK-PoK
  - Security reduces in ROM to security of non-interactive ZK-PoK & finding x' s.t. $C(x') = y$

Summary

- Interactive Proofs
- “Proof of Knowledge”
  - Extractor that can extract witness/secret from successful adversary
- “Zero-Knowledge”
  - Simulator that produces transcripts indistinguishable from real transcripts
- Fiat-Shamir Transformation
  - Replace public-coin verifier with Random Oracle
  - Transformation Interactive Proof ⇒ Non-interactive Proof
  - Parallel Soundness Amplification
- Signature scheme
  - Add message to Random Oracle input
  - Signature is compressed transcript