Cryptography Today

Federico Pintore[1]

[1]Mathematical Institute,
University of Oxford

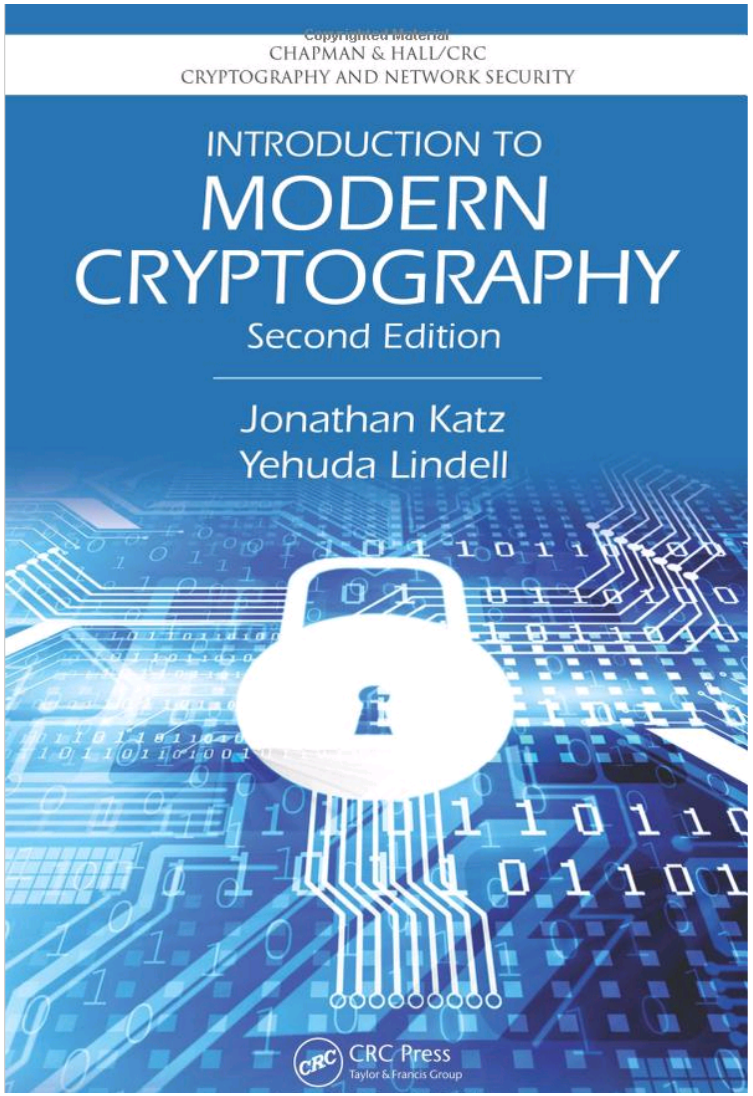
About the Course

- Regular classes with worksheets, to work with some concrete examples (every Friday from second week. Second, fourth and sixth week at 11 am in Room C2. Third, fifth and seventh week at 12:30 in Room C2. Eighth week at 11 am in Room C4).
- Every other week, write a short summary ($\approx$ 500 words) about one research paper (suggested in the further reading sections in the slides).
- You hand in your worksheets/summaries every Tuesday. You hand in/solve sheet-0 in week-2, and so on.
- One class (Friday, 29 Nov) to give presentations (in groups) about a chosen research paper (not graded).

About the Course

- Mini project
- Reading research papers!

Course Main Reference



Course Material

- slides for each lecture
- slides courtesy of

Dr. Ali El Kaafarani

Mathematical Instute, PQShield Ltd

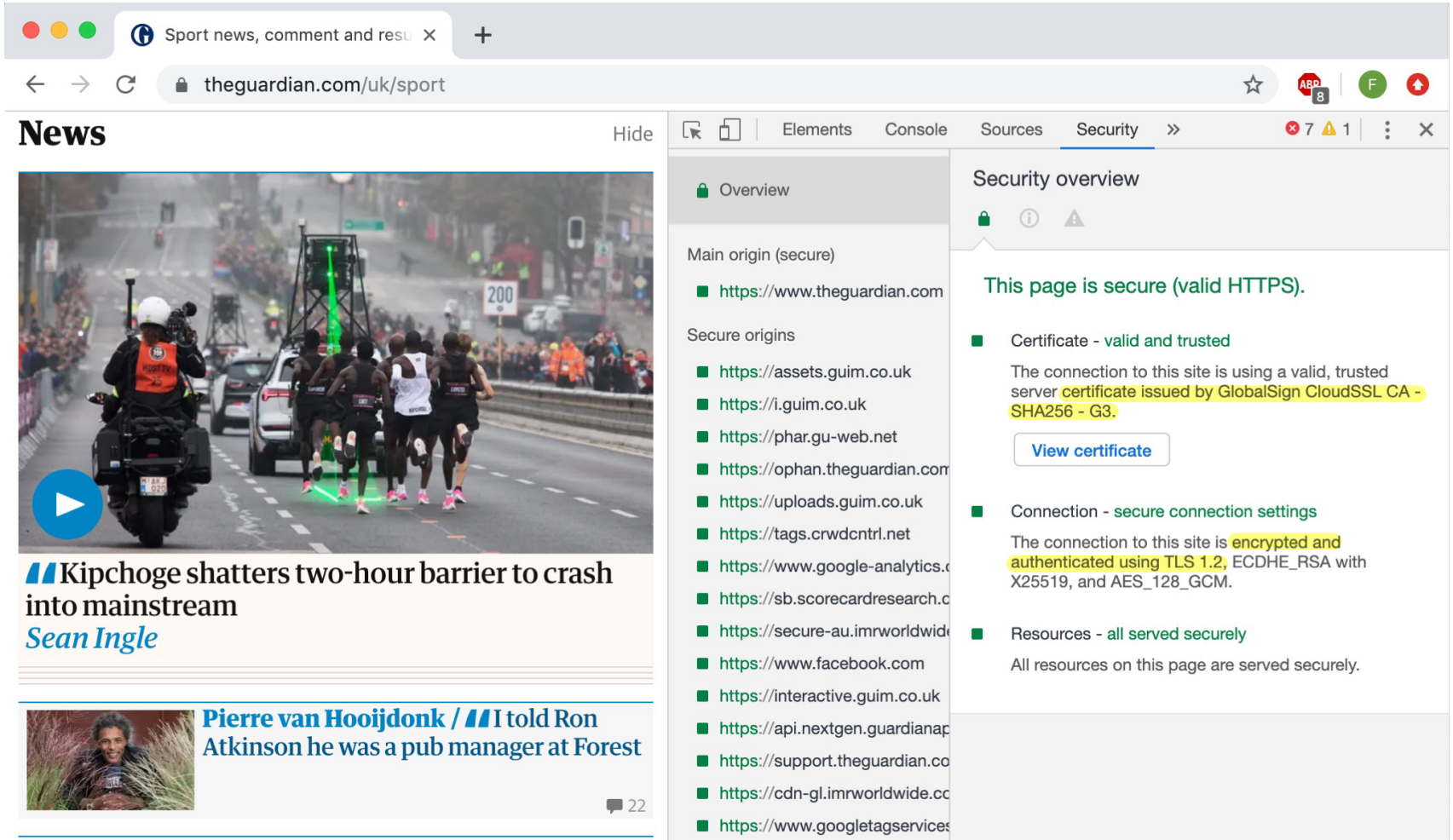
Outline

1. **Cryptography Usage in the Real World: Do we use it? Where?**

2. **Modern Cryptography**
   - Provable Security
   - Symmetric Key Cryptosystems
   - Hash Functions
   - Public Key Cryptosystems
   - Digital Signatures

3. **Advanced Cryptographic Tools/Schemes**

Web Browsers



Messaging Systems

Mobile Applications

An Empirical Study of Cryptographic Misuse in Android Applications

Manuel Egele, David Brumley
Carnegie Mellon University
{megele,dbrumley}@cmu.edu

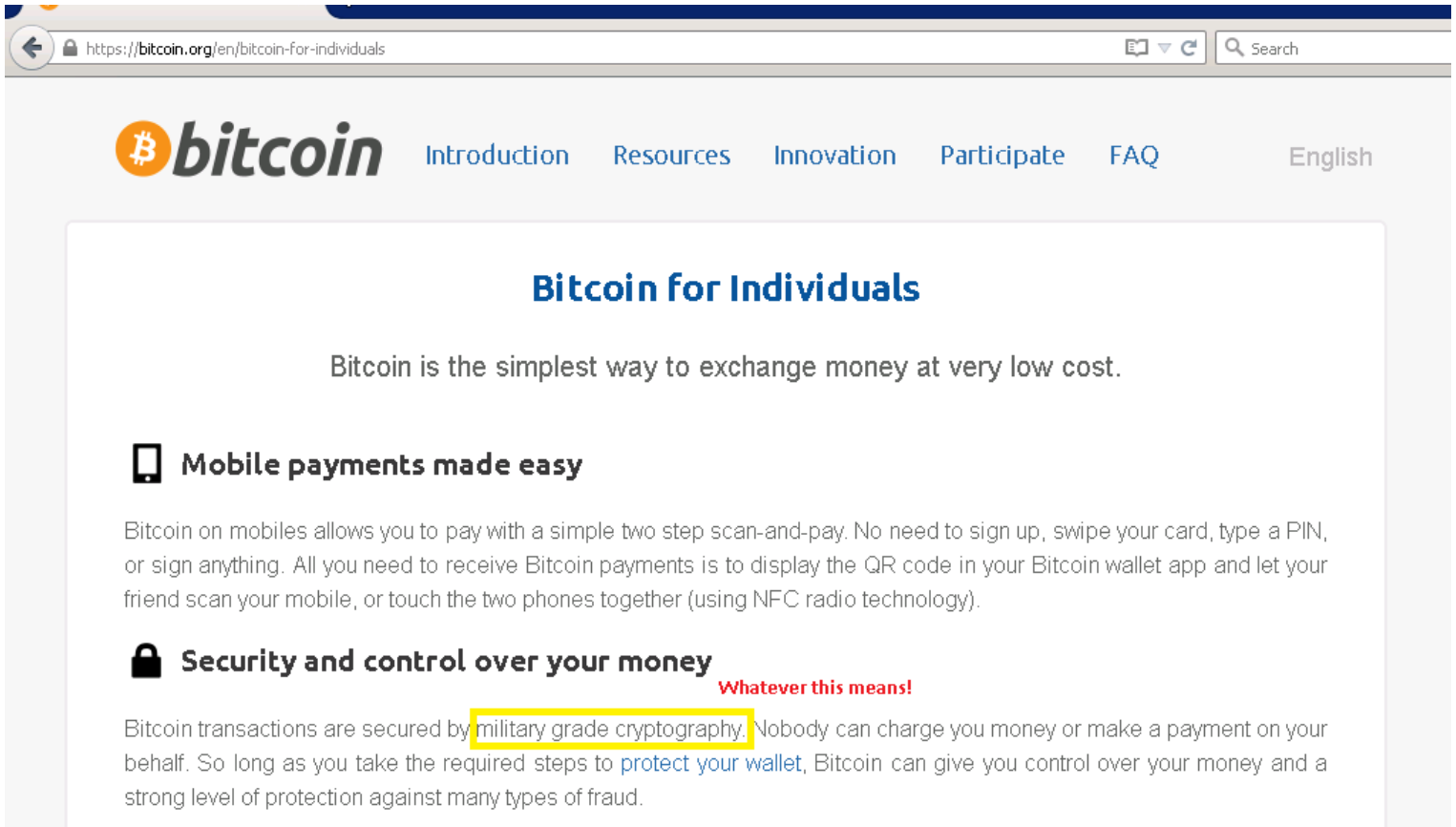
Yanick Fratantonio, Christopher Kruegel
University of California, Santa Barbara
{yanick,chris}@cs.ucsb.edu

ABSTRACT

Developers use cryptographic APIs in Android with the intent of securing data such as passwords and personal information on mobile devices. In this paper, we ask whether developers use the cryptographic APIs in a fashion that provides typical cryptographic notions of security, e.g., IND-CPA security. We develop program analysis techniques to automatically check programs on the Google Play marketplace, and find that 10,327 out of 11,748 applications that use cryptographic APIs – 88% overall – make at least one mistake. These numbers show that applications do not use cryptographic APIs in a fashion that maximizes overall security. We then suggest specific remediations based on our analysis towards improving overall cryptographic security in Android applications.

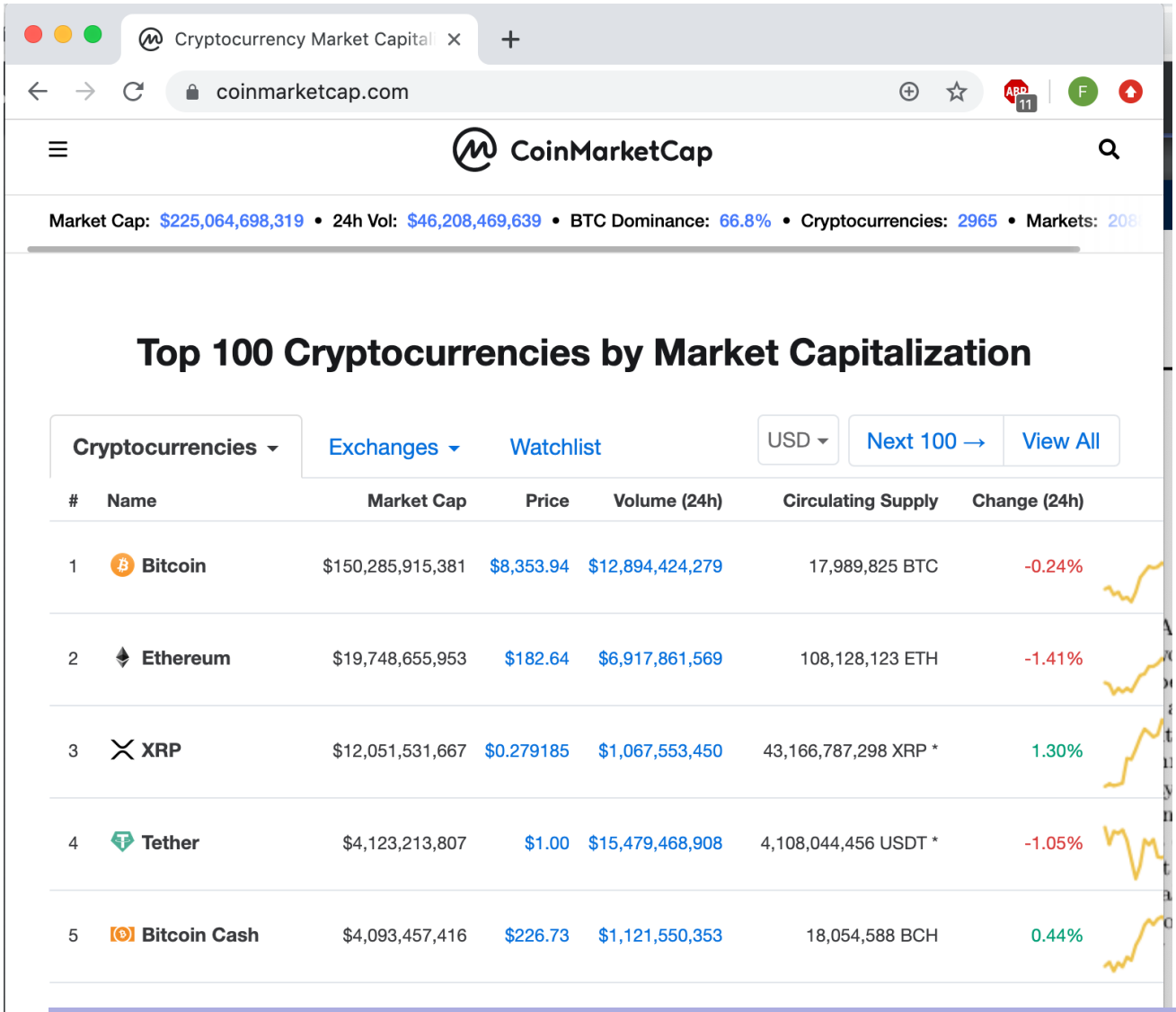
Crypto Makes the Headlines

Bitcoin

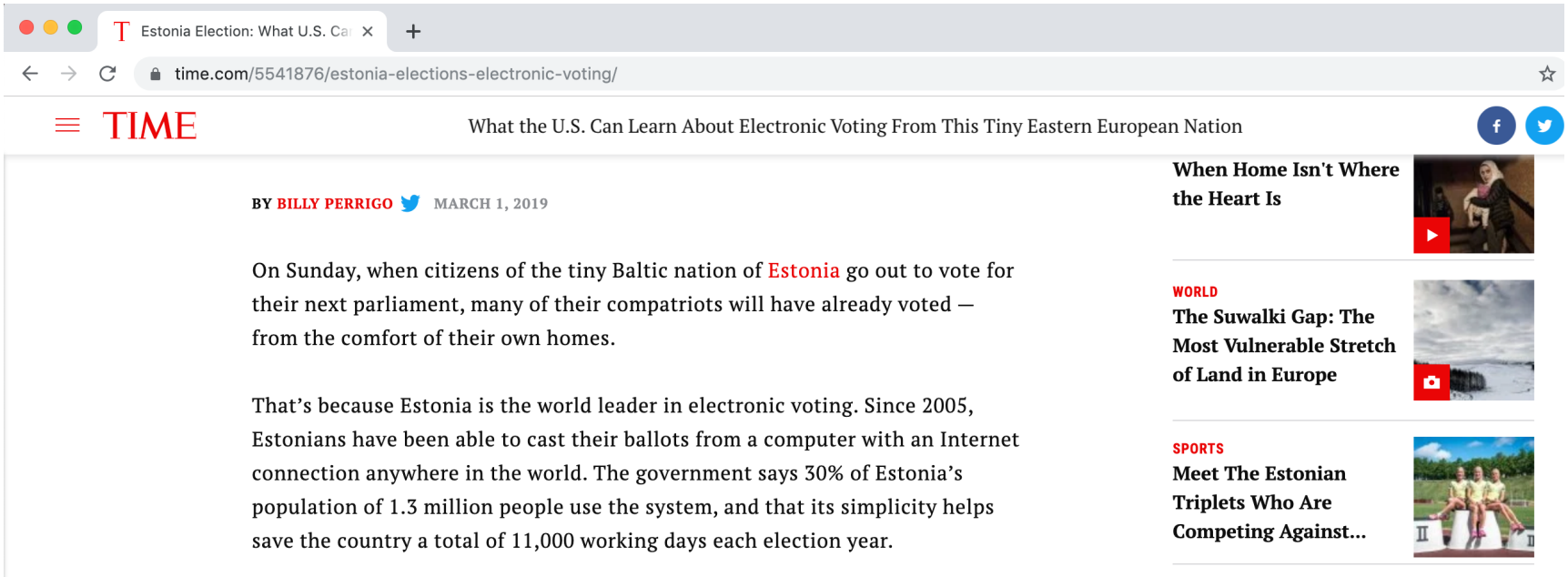


https://bitcoin.org/en/

Altcoins


Cryptocurrency Market Capital
coinmarketcap.com
CoinMarketCap
Market Cap: $225,064,698,319 • 24h Vol: $46,208,469,639 • BTC Dominance: 66.8% • Cryptocurrencies: 2965 • Markets:
Top 100 Cryptocurrencies by Market Capitalization
Cryptocurrencies
Exchanges
Watchlist
USD
Next 100 →
View All
Name Market Cap Price Volume (24h) Circulating Supply Change (24h)
1 Bitcoin $150,285,915,381 $8,353.94 $12,894,424,279 17,989,825 BTC -0.24%
2 Ethereum $19,748,655,953 $182.64 $6,917,861,569 108,128,123 ETH -1.41%
3 XRP $12,051,531,667 $0.279185 $1,067,553,450 43,166,787,298 XRP * 1.30%
4 Tether $4,123,213,807 $1.00 $15,479,468,908 4,108,044,456 USDT * -1.05%
5 Bitcoin Cash $4,093,457,416 $226.73 $1,121,550,353 18,054,588 BCH 0.44%

E-voting


Estonia Election: What U.S. Ca
time.com/5541876/estonia-elections-electronic-voting/
TIME
What the U.S. Can Learn About Electronic Voting From This Tiny Eastern European Nation
BY BILLY PERRIGO MARCH 1, 2019
On Sunday, when citizens of the tiny Baltic nation of Estonia go out to vote for their next parliament, many of their compatriots will have already voted — from the comfort of their own homes.
That's because Estonia is the world leader in electronic voting. Since 2005, Estonians have been able to cast their ballots from a computer with an Internet connection anywhere in the world. The government says 30% of Estonia's population of 1.3 million people use the system, and that its simplicity helps save the country a total of 11,000 working days each election year.
When Home Isn't Where the Heart Is
WORLD
The Suwalki Gap: The Most Vulnerable Stretch of Land in Europe
SPORTS
Meet The Estonian Triplets Who Are Competing Against...

Cryptography Usage in the Real World: a Summary

Cryptography Usage in the Real World: a Summary

- On-line banking, e-commerce

Cryptography Usage in the Real World: a Summary

- On-line banking, e-commerce
- SSH: to remotely login and to transfer files.

Cryptography Usage in the Real World: a Summary

- On-line banking, e-commerce
- SSH: to remotely login and to transfer files.
- Emails, cloud computing, etc.

Cryptography Usage in the Real World: a Summary

- On-line banking, e-commerce
- SSH: to remotely login and to transfer files.
- Emails, cloud computing, etc.
- Streaming media providers

Cryptography Usage in the Real World: a Summary

- On-line banking, e-commerce
- SSH: to remotely login and to transfer files.
- Emails, cloud computing, etc.
- Streaming media providers
- ATM machines, etc.

What Cryptography Was

- largely an art
- historically exploited to enable secret communications
- until the 1970s, mainly used for military purposes

What Cryptography Is

Definition (From Katz, Lindell's book)

Cryptography is the study of mathematical techniques for securing digital information, systems, and distributed computations against adversarial attacks.

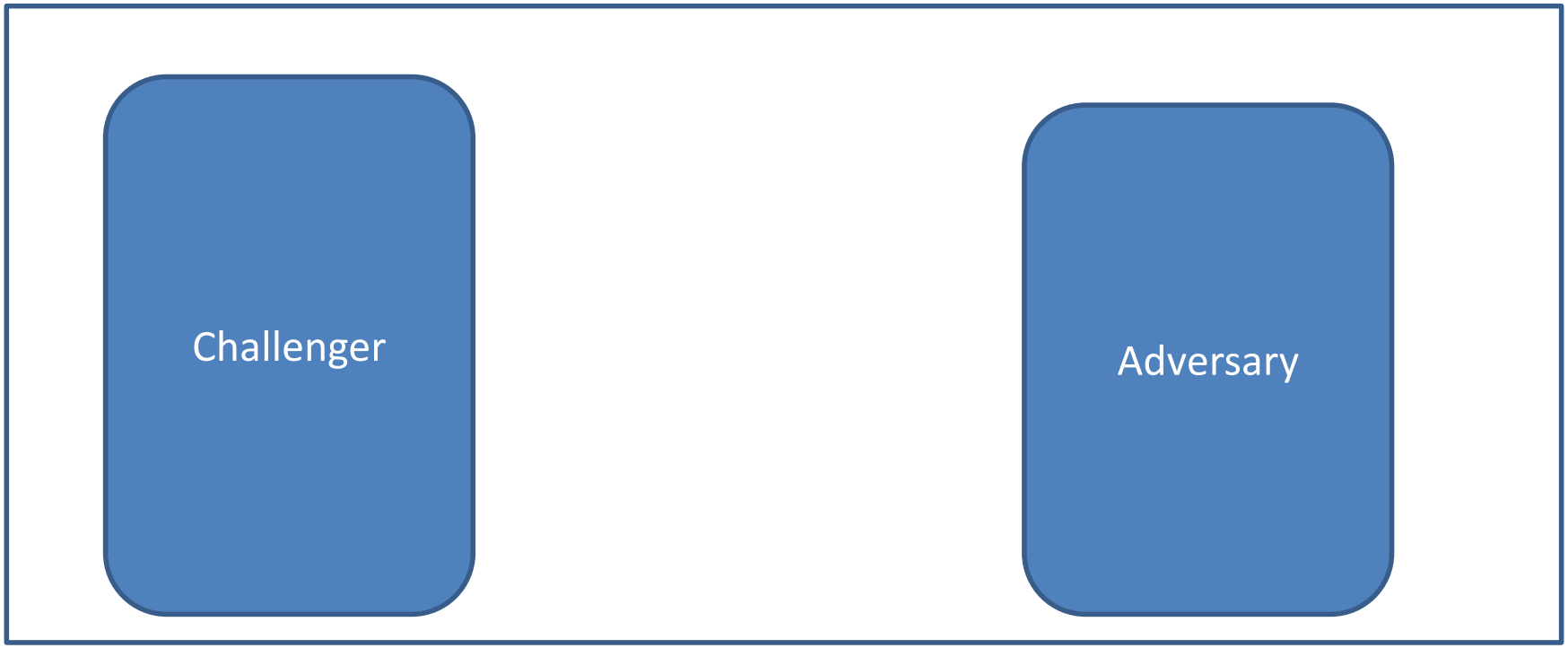
- a science
- exploited for countless real world applications
- ubiquitous in our everyday life

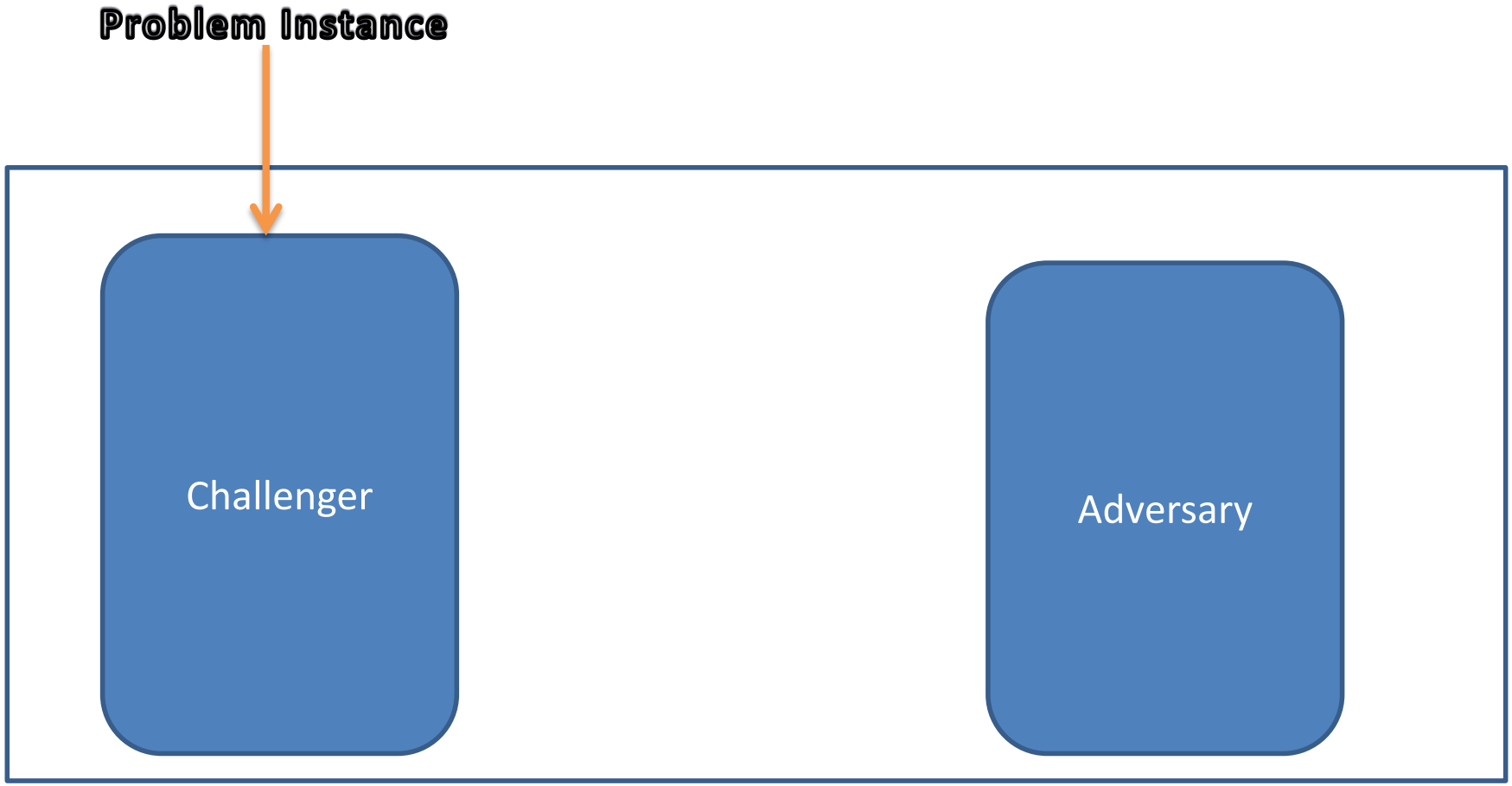
Outline

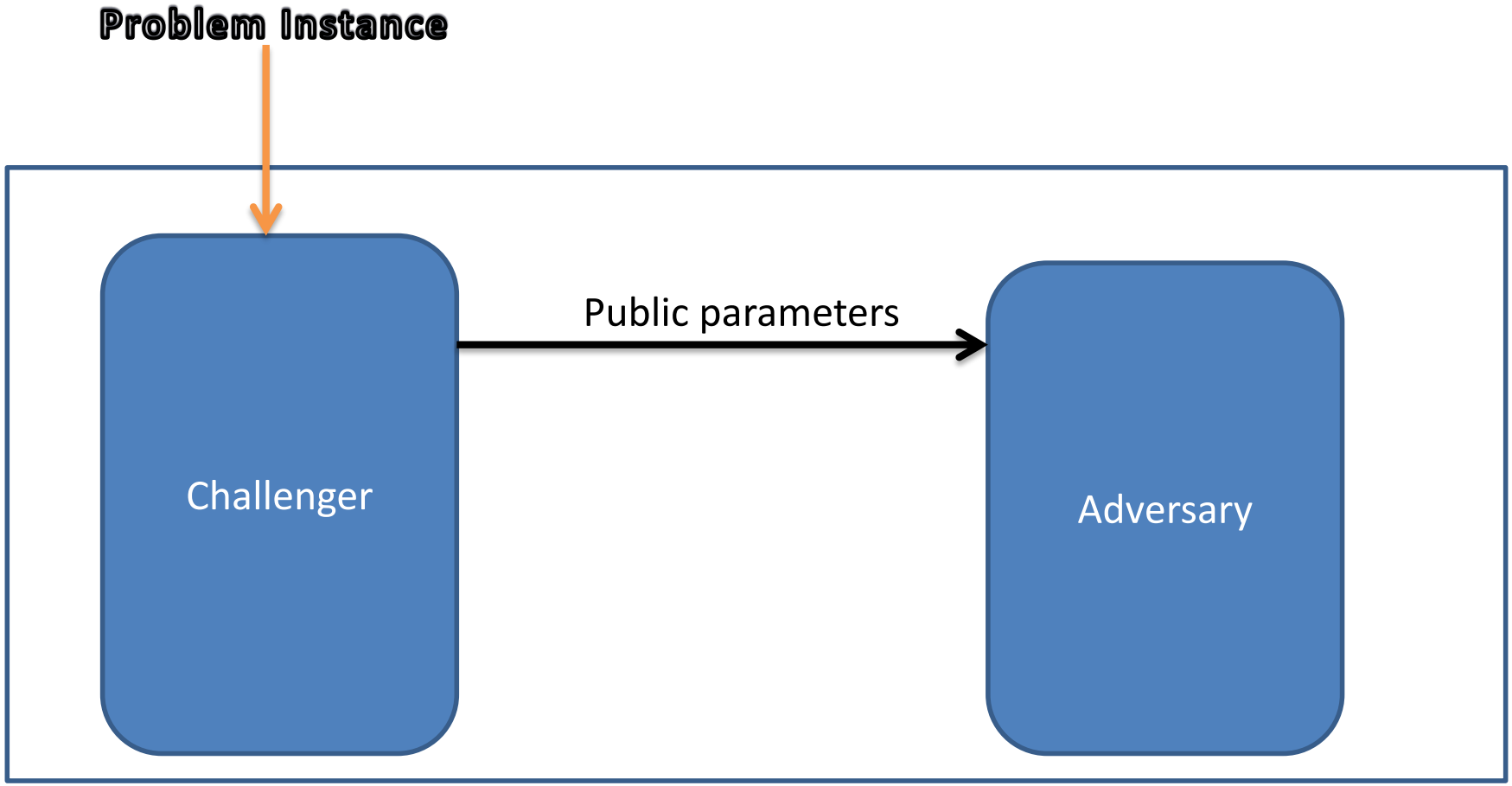
Provable Security

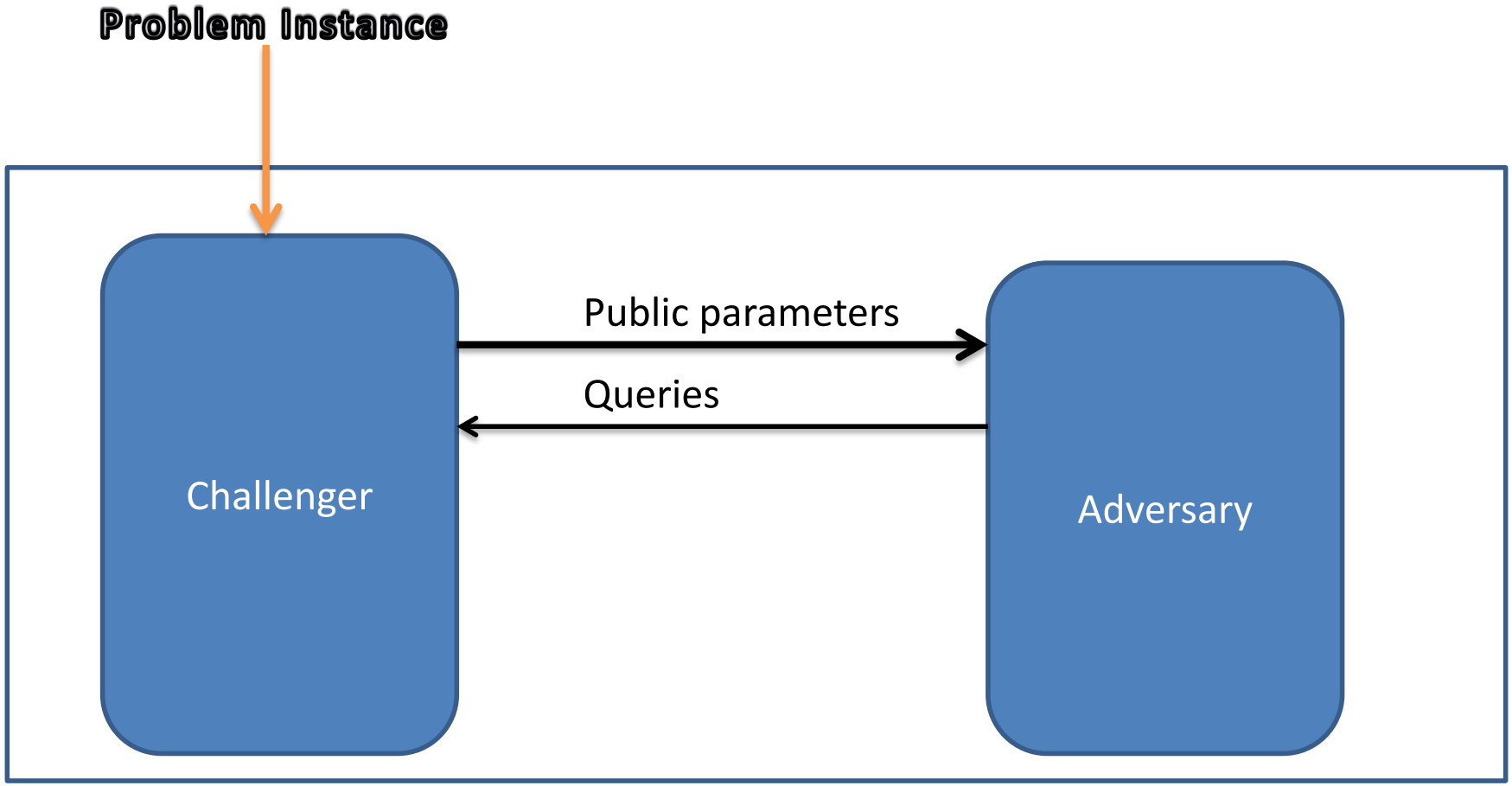
- Proofs of security of cryptographic schemes are among the features distinguishing Modern Cryptography from Classical Cryptography.
- How can we prove the security of our cryptosystems?
- Proofs by reduction: an efficient attacker on the cryptographic scheme is turned (by a **reduction**) into an efficient algorithm solving an assumed-to-be hard mathematical problem. It is a proof by contradiction.

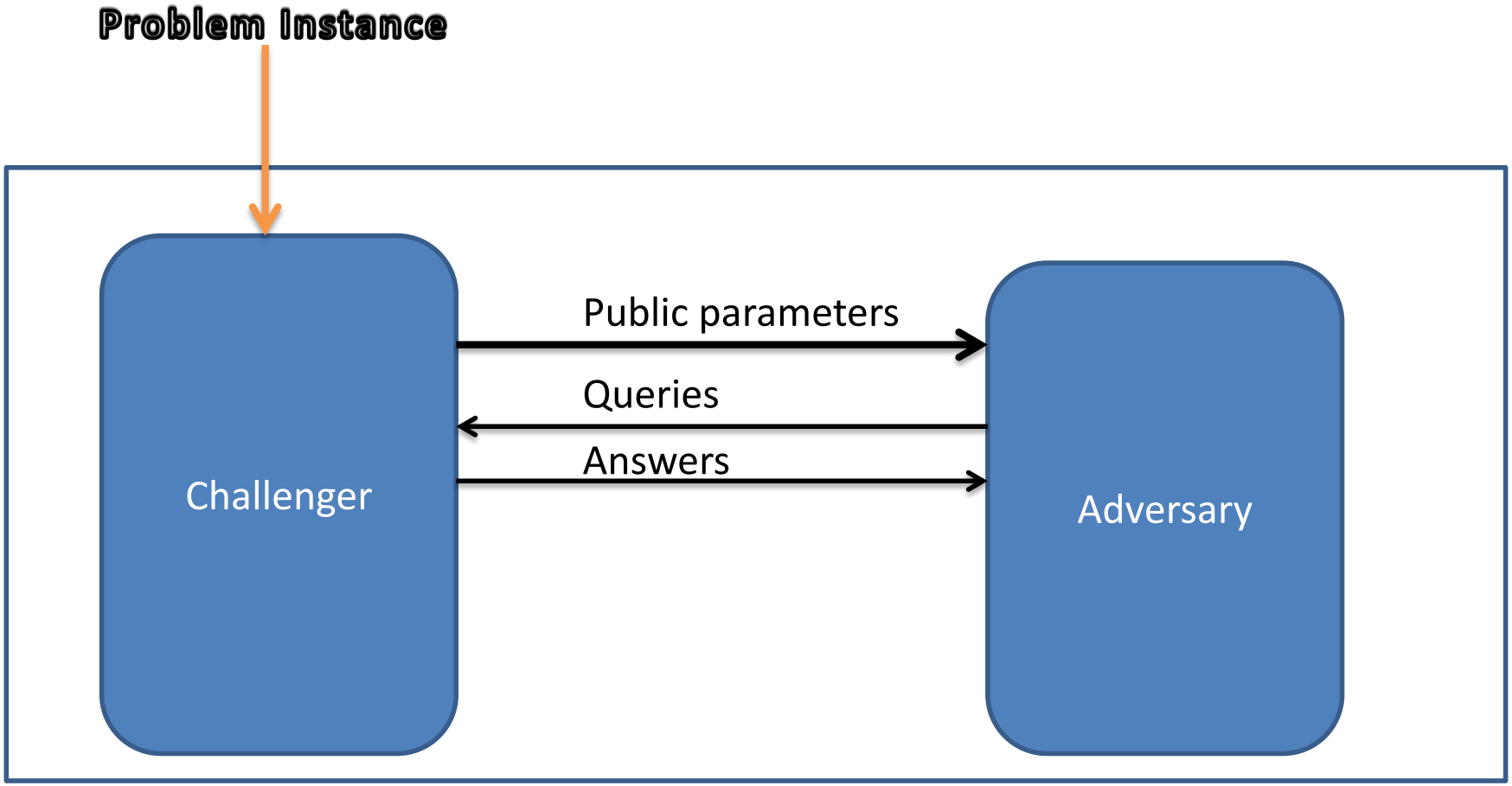
Security Games: Proofs by Reduction

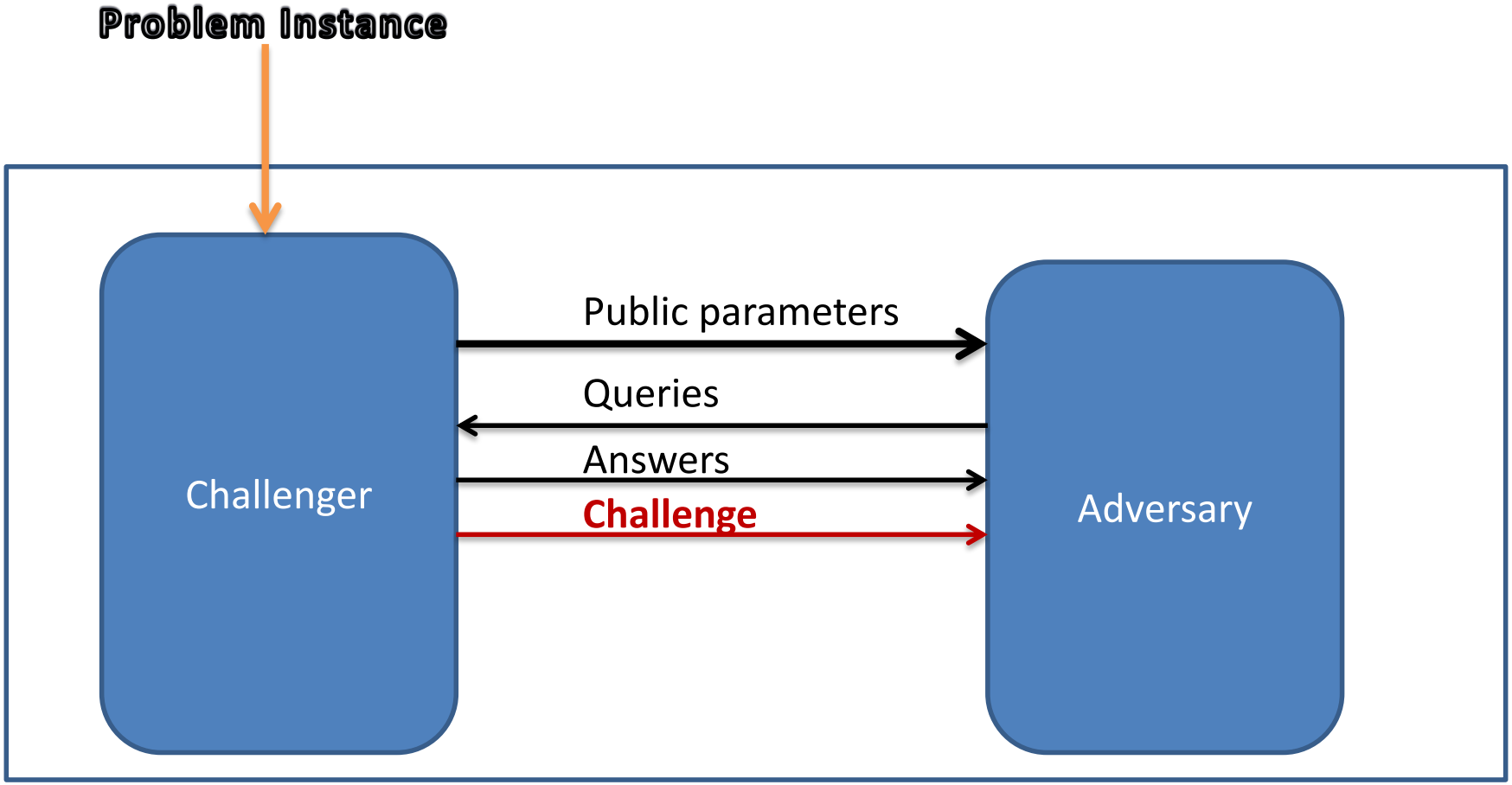


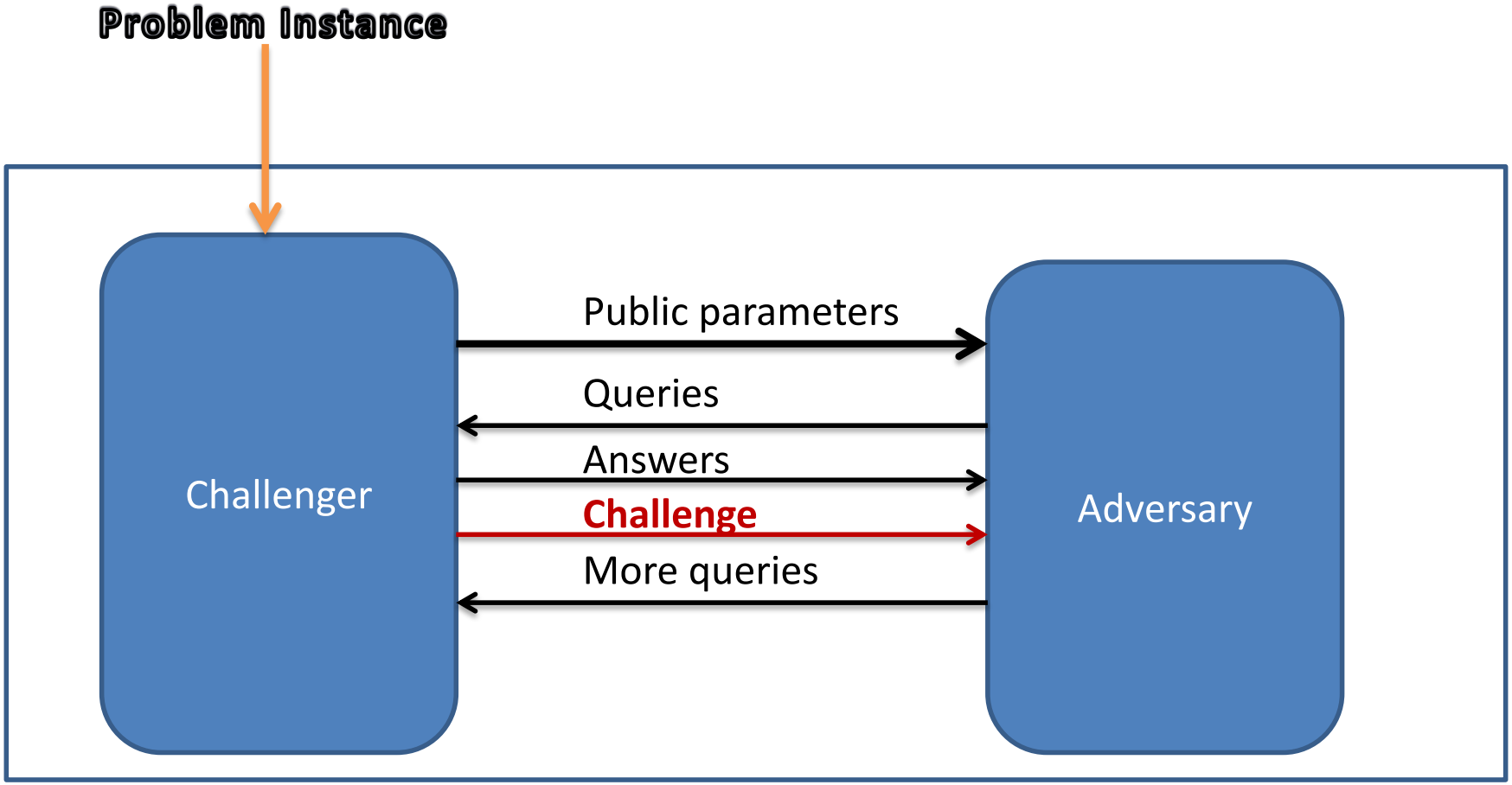

Problem Instance
Challenger
Adversary

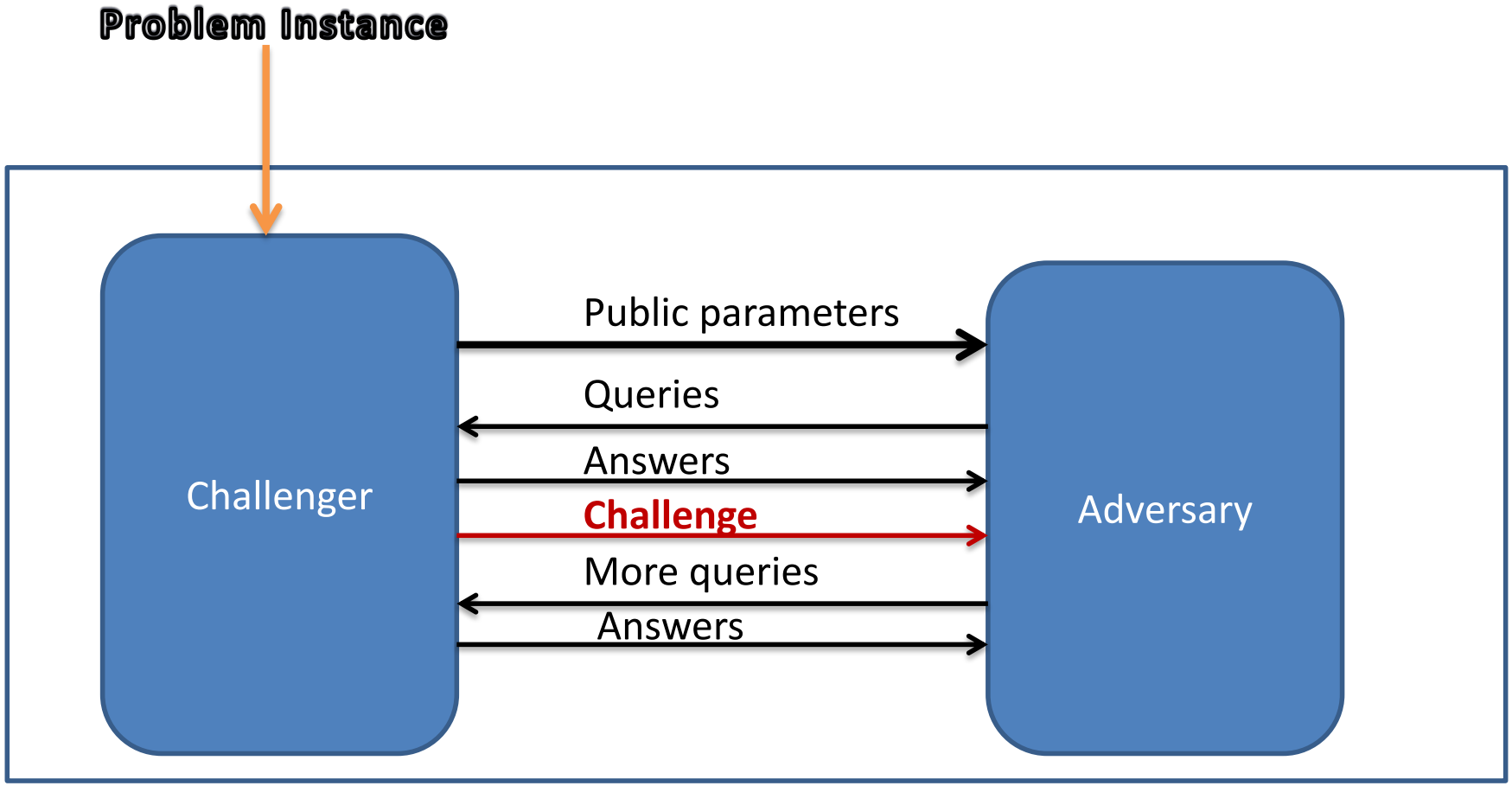

Problem Instance
Challenger
Public parameters
Adversary

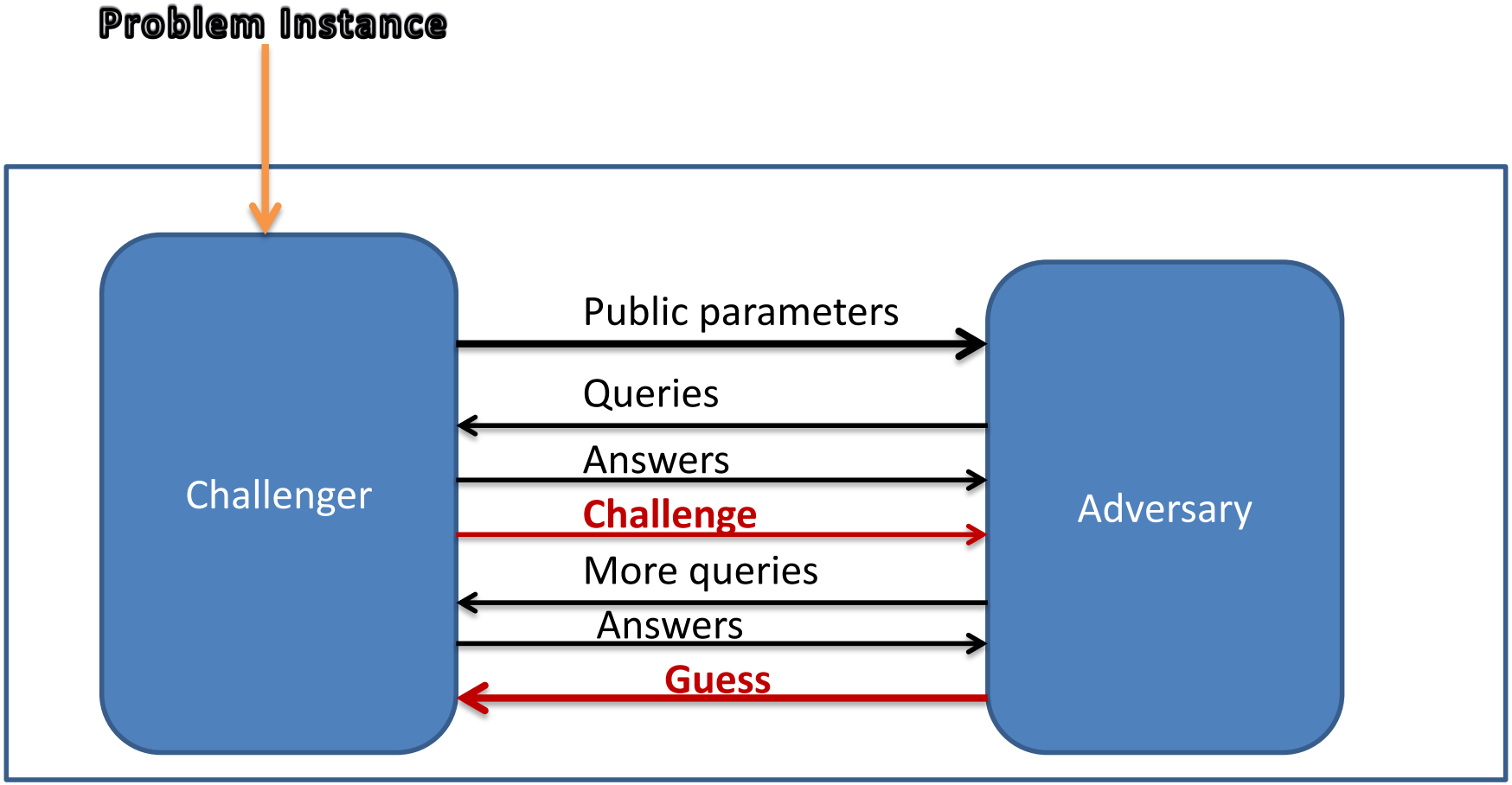

Problem Instance
Public parameters
Queries
Challenger
Adversary

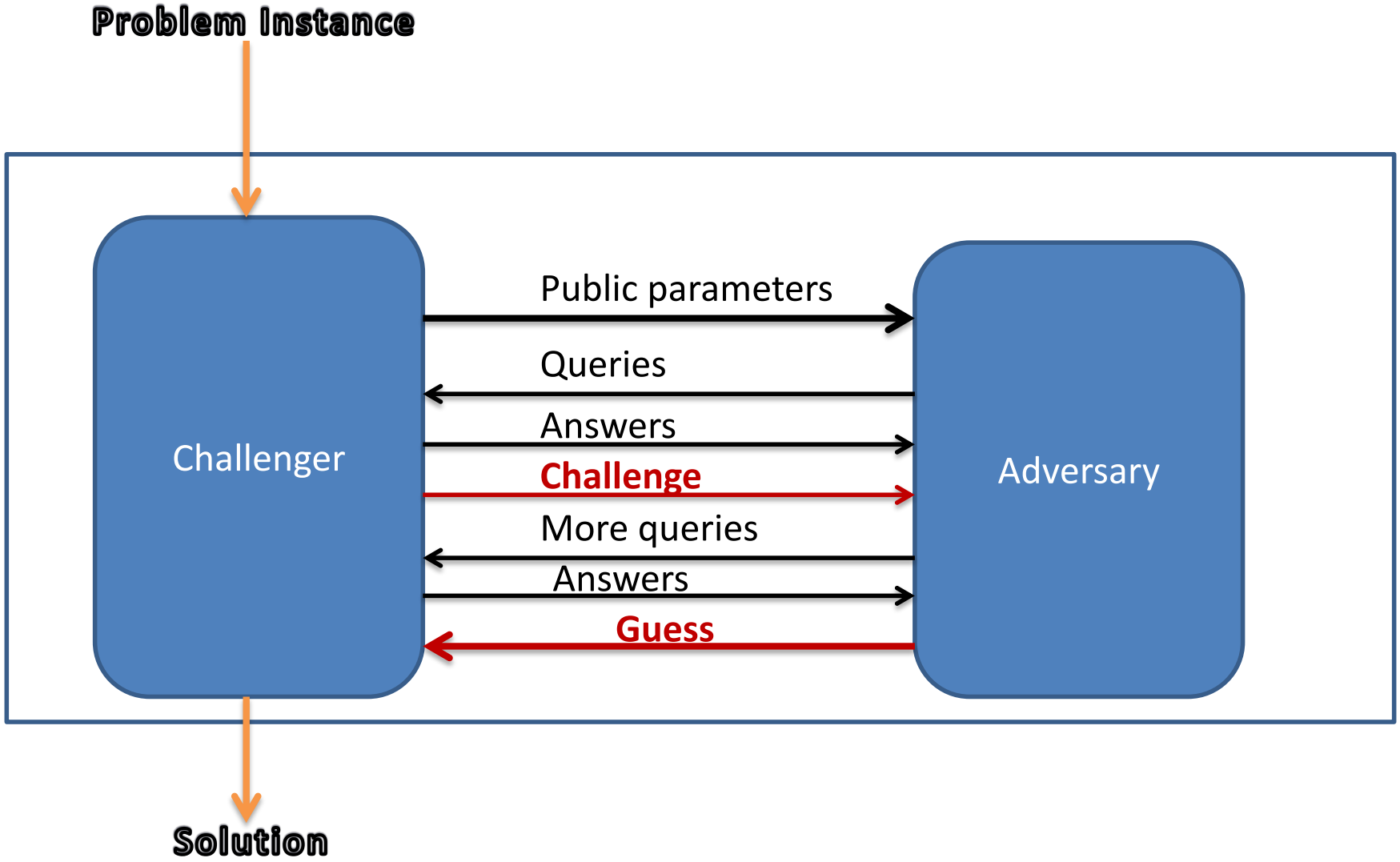

Problem Instance
Challenger
Public parameters
Queries
Answers
Adversary


Problem Instance
Challenger
Public parameters
Queries
Answers
Challenge
Adversary


Problem Instance
Challenger
Public parameters
Queries
Answers
Challenge
More queries
Adversary


Problem Instance
Challenger
Public parameters
Queries
Answers
Challenge
More queries
Answers
Adversary


Problem Instance
Challenger
Public parameters
Queries
Answers
Challenge
More queries
Answers
Guess
Adversary


Problem Instance
Challenger
Public parameters
Queries
Answers
Challenge
More queries
Answers
Guess
Adversary
Solution

Hard Mathematical Problems

Some mathematical problems are believed to be computationally hard (to different extents):

- Integer Factorization: given a composite number n, compute its (unique) factorization $n = \prod p_i^{e_i}$, where p_i are prime numbers.
- It is *believed* to be hard if $n = pq$ for well-chosen $p \neq q$.

Hard Mathematical Problems

Some mathematical problems are believed to be computationally hard (to different extents):

- Integer Factorization: given a composite number n, compute its (unique) factorization $n = \prod p_i^{e_i}$, where p_i are prime numbers.
- It is *believed* to be hard if $n = pq$ for well-chosen $p \neq q$.
- Discrete Logarithm: given a cyclic group $(G = \langle g \rangle, \circ)$, $h \in G$, compute $k \in \mathbb{Z}_{|G|}$ such that $g^k = h$
- Dlog is *believed to be* hard in $G = \mathbb{F}_p^*$ and even harder in groups of points on (well-chosen) elliptic/hyperelliptic curves.

Hard Mathematical Problems

Short Vector Problem (SVP) in Lattices:

- Given n *linearly independent* vectors $\vec{b}_1, \ldots, \vec{b}_n \in \mathbb{R}^m$, the lattice generated by them is defined as

$$\mathcal{L}(\vec{b}_1, \ldots, \vec{b}_n) \stackrel{\text{def}}{=} \left\{ \sum_{i=1}^{n} x_i \vec{b}_i \mid x_i \in \mathbb{Z} \right\}$$

- SVP: it is hard to determine the smallest non-zero vector in an arbitrary lattice (easy in low dimensions).

Hardness Assumptions: Average vs. Worst Cases

Do we have the same confidence in different cryptosystems that are based on different hardness assumptions?

Hardness Assumptions: Average vs. Worst Cases

Do we have the same confidence in different cryptosystems that are based on different hardness assumptions?

- *average-case* assumption: hardness of solving a random instance (drawn from a given probability distribution) of a problem;

Hardness Assumptions: Average vs. Worst Cases

Do we have the same confidence in different cryptosystems that are based on different hardness assumptions?

- *average-case* assumption: hardness of solving a random instance (drawn from a given probability distribution) of a problem;
- *worst-case* assumption: hardness of solving an arbitrary instance of a problem (even the worst instances)

Hardness Assumptions: Average vs. Worst Cases

Do we have the same confidence in different cryptosystems that are based on different hardness assumptions?

- *average-case* assumption: hardness of solving a random instance (drawn from a given probability distribution) of a problem;
- *worst-case* assumption: hardness of solving an arbitrary instance of a problem (even the worst instances)

Breaking a lattice-based cryptographic scheme is at least as hard as solving several hard lattice problems in the worst case.

Outline

Symmetric Key Cryptosystems

A symmetric encryption scheme consists of three algorithms that are (KeyGen, Enc, Dec). Let $\mathcal{M}$ be the message space, whereas the key space is $\mathcal{K}$.

- KeyGen(n): is a randomized algorithm that, given the security parameter n, returns a key SK $\in \mathcal{K}$.
- Enc(SK, m): is a randomized algorithm that, on input a key SK $\in \mathcal{K}$ and a plaintext $m \in \mathcal{M}$, outputs a ciphertext c.
- Dec(SK, c): is a deterministic algorithm that, on input a key SK and a ciphertext c, outputs a message $m \in \mathcal{M} \cup \perp$.

Correctness:

$$\forall m \in \mathcal{M}, \Pr[\text{SK} \leftarrow \text{KeyGen}(n) : \text{Dec}(\text{SK}, \text{Enc}(\text{SK}, m)) = m] = 1$$

Outline

Hash Functions

- Informally speaking, hash functions take a long input string and output a shorter string of a fixed length called a *digest*.
- They are used to achieve *integrity* (or *authenticity*) in the private-key setting.
- They are used almost everywhere in Cryptography, e.g. HMAC, commitment schemes, saved passwords, etc.
- If you *imagine* that hash functions are truly random (modelled as *random oracle model*), then proving the security of some cryptographic schemes becomes achievable (e.g. RSA-OAEP).
- A debate/controversy over the soundness of the random oracle model.

Outline

Public Key Cryptosystems

An asymmetric encryption scheme consists of the following algorithms:

- KeyGen(n): is a randomized algorithm that takes the security parameters as input and returns a pair of keys $(\mathsf{PK}, \mathsf{SK})$, the public key PK and its matching secret key SK, respectively.
- Enc(PK, m): A randomized algorithm that takes a public key PK, a plaintext m and returns a ciphertext c.
- Dec(SK, c): A deterministic algorithm that takes the secret key SK and a ciphertext c, and returns a message $m \in \mathcal{M} \cup \perp$.

Correctness:

$$\forall m \in \mathcal{M}, \Pr[(\mathsf{SK}, \mathsf{PK}) \leftarrow \mathsf{KeyGen}(n) : \mathsf{Dec}(\mathsf{Enc}(\mathsf{PK}, m), \mathsf{SK}) = m] = 1$$

Outline

Digital Signatures

- Are used to achieve *integrity* (or *authenticity*) in the public key setting.

Digital Signatures

- Are used to achieve *integrity* (or *authenticity*) in the public key setting.
- If a signature σ on a message m is verified correctly against a given public key PK, it ensures that the message was indeed sent by the owner of this public key (already known to potential verifiers) and the message was NOT modified in transit.

Digital Signatures

- Are used to achieve *integrity* (or *authenticity*) in the public key setting.
- If a signature σ on a message m is verified correctly against a given public key PK, it ensures that the message was indeed sent by the owner of this public key (already known to potential verifiers) and the message was NOT modified in transit.
- More importantly, signers cannot deny having signed a message, also known as *non-repudiation*.

Secret Sharing

- schemes for distributing a secret amongst a group of participants
- each participant receives a share of the secret
- the secret can be reconstructed only when a sufficient number of shares are combined together

Lagrange Interpolating Polynomial: given n points $(x_1, y_1), \cdots, (x_n, y_n)$, one can construct the polynomial

$$P(x) = \sum_{j=1}^{n} y_j P_j(x)$$

of degree $\leq (n-1)$ that passes through them, setting:

$$P_j(x) = y_j \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{(x - x_k)}{(x_j - x_k)}$$

Shamir Secret Sharing (1979)

It works in two phases as follows:

- Distribute the shares: pick a random polynomial $Q(x) \in \mathbb{F}_p[x]$ of degree $\ell < n$ (where n is the number of participants) s.t. $Q(0) = s$. Compute the shares

$$S_i = Q(i) \pmod{p} \quad i = 1, \cdots, n$$

  and send them over to the participants $A_1, \cdots, A_n$.
- Reconstruct the secret: using Lagrange interpolation, any $\ell + 1$ participants can *together* compute $Q(0) \mod p$ which is the secret s.

Multi-Party Computation

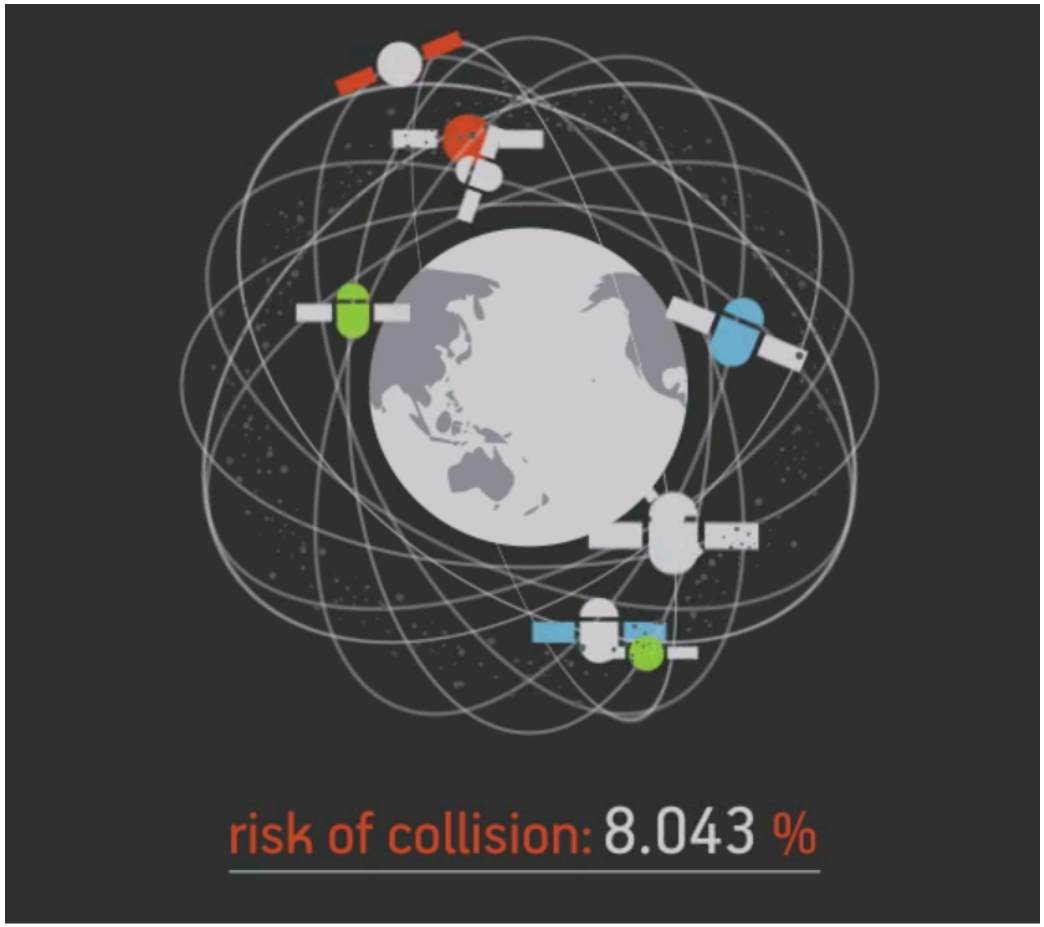
- Suppose that each of the n parties $P_1, \ldots, P_n$ has a secret input s_i. They all want to evaluate a public function f on input $(s_1, \cdots, s_n)$ to learn the output and yet keep their secret inputs hidden from each other.

Multi-Party Computation

- Suppose that each of the n parties $P_1, \ldots, P_n$ has a secret input s_i. They all want to evaluate a public function f on input $(s_1, \cdots, s_n)$ to learn the output and yet keep their secret inputs hidden from each other.
- *Secure Multi-Party Computation* is the solution!

Multi-Party Computation: an Application

https://www.youtube.com/watch?v=bAp_aZgX3B0



Zero-Knowledge Proofs

To convince a Verifier that some mathematical statement is true, without revealing anything else.

Zero-Knowledge Proofs

To convince a Verifier that some mathematical statement is true, without revealing anything else.

Statements can be about *facts* (e.g. the number N is squarefree) or about *knowledge* (e.g. I know the factorisation of N).

Zero-Knowledge Proofs

To convince a Verifier that some mathematical statement is true, without revealing anything else.

Statements can be about *facts* (e.g. the number N is squarefree) or about *knowledge* (e.g. I know the factorisation of N).

- Completeness: If a given statement is true, the Prover can always convince a verifier

Zero-Knowledge Proofs

To convince a Verifier that some mathematical statement is true, without revealing anything else.

Statements can be about *facts* (e.g. the number N is squarefree) or about *knowledge* (e.g. I know the factorisation of N).

- Completeness: If a given statement is true, the Prover can always convince a verifier
- Soundness: Prover cannot convince the verifier if the statement is false

Zero-Knowledge Proofs

To convince a Verifier that some mathematical statement is true, without revealing anything else.

Statements can be about *facts* (e.g. the number N is squarefree) or about *knowledge* (e.g. I know the factorisation of N).

- Completeness: If a given statement is true, the Prover can always convince a verifier
- Soundness: Prover cannot convince the verifier if the statement is false
- Zero-Knowledge: The proof doesn't reveal any extra information beyond the validity of the statement

Zero-Knowledge Proofs

- Blog: `http://blog.cryptographyengineering.com/2014/11/zero-knowledge-proofs-illustrated-primer.html`
- Online demo: `http://web.mit.edu/~ezyang/Public/graph/svg.html`

Fully Homomorphic Encryption

- Cloud computing is a hot topic nowadays!
- Companies want to store their huge data on the clouds and let the cloud companies do the computation on their data.

Fully Homomorphic Encryption

- Cloud computing is a hot topic nowadays!
- Companies want to store their huge data on the clouds and let the cloud companies do the computation on their data.
- But they want to preserve data confidentiality, so they decide to encrypt their data (and not give away the encryption keys!)

Fully Homomorphic Encryption

- Cloud computing is a hot topic nowadays!
- Companies want to store their huge data on the clouds and let the cloud companies do the computation on their data.
- But they want to preserve data confidentiality, so they decide to encrypt their data (and not give away the encryption keys!)
- How can the cloud companies do computation on encrypted data and give back the result in an encrypted format!

Fully Homomorphic Encryption

- Some encryption schemes are naturally *partially* homomorphic, i.e., $\text{Enc}(A) \times \text{Enc}(B) = \text{Enc}(A \times B)$.
- *Fully* homomorphic encryption allows for *arbitrary* computation on ciphertexts. You can write a program of any functionality and run it on a given ciphertext to get the desirable result in an encrypted format!
- In theory, this was proven possible in 2009. In practice, it is still far away from being practical!

Classical Vs Post-Quantum Cryptography

- What would happen to Dlog and Factorisation based Cryptosystems if quantum computers existed?

Classical Vs Post-Quantum Cryptography

- What would happen to Dlog and Factorisation based Cryptosystems if quantum computers existed?
  - Shor's algorithm solves both problems *efficiently*

Classical Vs Post-Quantum Cryptography

- What would happen to Dlog and Factorisation based Cryptosystems if quantum computers existed?
  - Shor's algorithm solves both problems *efficiently*
- Any alternatives?
- New hard problems for new cryptographic branches:

Classical Vs Post-Quantum Cryptography

- What would happen to Dlog and Factorisation based Cryptosystems if quantum computers existed?
  - Shor's algorithm solves both problems *efficiently*
- Any alternatives?
- New hard problems for new cryptographic branches:
  - Lattice-Based Cryptography (e.g. fully homomorphic encryption)

Classical Vs Post-Quantum Cryptography

- What would happen to Dlog and Factorisation based Cryptosystems if quantum computers existed?
  - Shor's algorithm solves both problems *efficiently*
- Any alternatives?
- New hard problems for new cryptographic branches:
  - Lattice-Based Cryptography (e.g. fully homomorphic encryption)
  - Code-Based Cryptography (e.g. McEliece cryptosystem)

Classical Vs Post-Quantum Cryptography

- What would happen to Dlog and Factorisation based Cryptosystems if quantum computers existed?
  - Shor's algorithm solves both problems *efficiently*
- Any alternatives?
- New hard problems for new cryptographic branches:
  - Lattice-Based Cryptography (e.g. fully homomorphic encryption)
  - Code-Based Cryptography (e.g. McEliece cryptosystem)
  - Hash-Based Cryptography (e.g. Merkle signature)

Classical Vs Post-Quantum Cryptography

- What would happen to Dlog and Factorisation based Cryptosystems if quantum computers existed?
  - Shor's algorithm solves both problems *efficiently*
- Any alternatives?
- New hard problems for new cryptographic branches:
  - Lattice-Based Cryptography (e.g. fully homomorphic encryption)
  - Code-Based Cryptography (e.g. McEliece cryptosystem)
  - Hash-Based Cryptography (e.g. Merkle signature)
  - Multivariate Cryptography (e.g. Rainbow signature)

Classical Vs Post-Quantum Cryptography

- What would happen to Dlog and Factorisation based Cryptosystems if quantum computers existed?
  - Shor's algorithm solves both problems *efficiently*
- Any alternatives?
- New hard problems for new cryptographic branches:
  - Lattice-Based Cryptography (e.g. fully homomorphic encryption)
  - Code-Based Cryptography (e.g. McEliece cryptosystem)
  - Hash-Based Cryptography (e.g. Merkle signature)
  - Multivariate Cryptography (e.g. Rainbow signature)
  - Isogeny-based Cryptography (e.g. SIDH)

Further Reading (1)

- Jean-Jacques Quisquater, Myriam Quisquater, Muriel Quisquater, Michaël Quisquater, Louis Guillou, Marie Guillou, Gaïd Guillou, Anna Guillou, Gwenolé Guillou, and Soazig Guillou.
  How to explain zero-knowledge protocols to your children.
  In *Advances in Cryptology—CRYPTO'89 Proceedings*, pages 628–631. Springer, 1990.
- Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov.
  The first collision for full sha-1.
  IACR Cryptology ePrint Archive, 2017:190, 2017.