Cryptography Today

Ali El Kaafarani[1,2]

[1]Mathematical Institute
[2] PQShield Ltd.

About the Course

- Regular classes with worksheets so you can work with some concrete examples (every Friday at 11 am in Room C4 first week, then in C5).
- 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 Wednesday by 4 pm. First week, you solve sheet-0 with the tutor and investigate some useful crypto-tools, then you hand in/solve sheet-1 in week-2, and so on.
- One class (Friday, 16 Nov) to give presentations (in groups) about a chosen research paper (not graded).

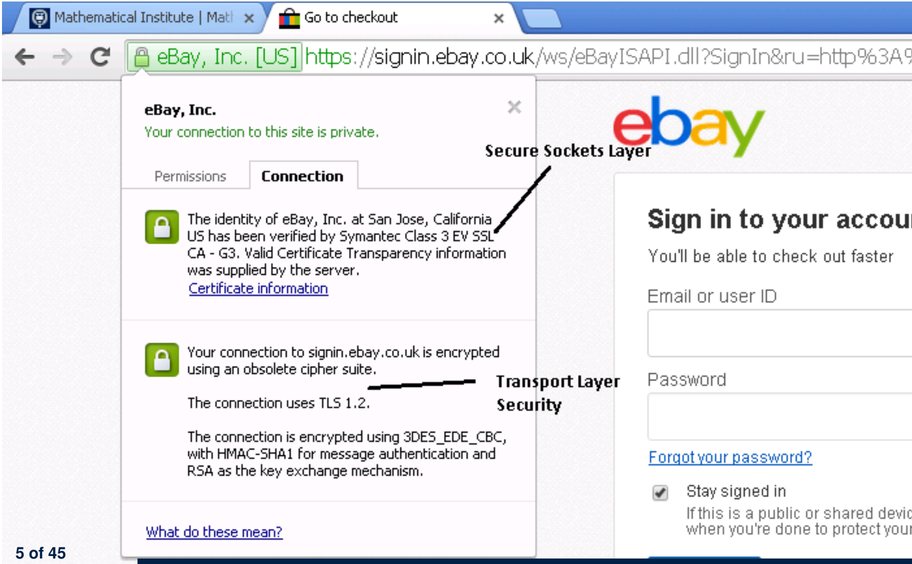
About the Course

- Mini project.
- Reading research papers!

Outline

1. **Cryptography Usage in the Real World: Do we use it? Where?**
2. **Modern Cryptography**
   - Provable Security
   - Symmetric Key Cryptosystems
   - Public Key Cryptosystems
   - Hash Functions
   - Digital Signatures
   - Advanced Cryptographic Tools/Schemes
3. **Classical Vs Post-Quantum Cryptography**

Web Browsers



Some Recent Cryptanalysis

PRESS RELEASE

From: Centrum Wiskunde & Informatica (CWI) in the Netherlands, Inria in France and Nanyang Technological University in Singapore (NTU Singapore)

Thursday 8 October 2015

RESEARCHERS URGE: INDUSTRY STANDARD SHA-1 SHOULD BE RETRACTED SOONER

International policy

The research team says: “In 2012, security expert Bruce SCHNEIER estimated the SHA-1 attack costs to be around 700,000 dollar in 2015. This would decrease by 2018 to about 173,000 dollar, which he deemed to be within the resources of criminals. However, we showed that graphics cards are much faster for these attacks and we now estimate that a full SHA-1 collision will cost between 75,000 and 120,000 dollar renting Amazon EC2 cloud over a few months today, in early autumn 2015. This implies that collisions are already within the resources of criminal syndicates, almost two years earlier than previously expected, and one year before SHA-1 will be marked as unsafe in modern Internet browsers. Therefore we recommend that SHA-1 based signatures should be marked as unsafe much sooner than current international policy prescribes. In particular, we strongly urge against a recent proposal to extend issuance of SHA-1 certificates with another year in the CA/Browser Forum, for which the discussion closes tomorrow, on 9 October.”

Some Recent Cryptanalysis (Feb 2017)

SHATTERED

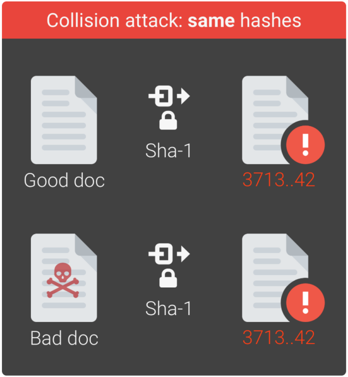
We have broken SHA-1 in practice.

This industry cryptographic hash function standard is used for digital signatures and file integrity verification, and protects a wide spectrum of digital assets, including credit card transactions, electronic documents, open-source software repositories and software updates.

It is now practically possible to craft two colliding PDF files and obtain a SHA-1 digital signature on the first PDF file which can also be abused as a valid signature on the second PDF file.

For example, by crafting the two colliding PDF files as two rental agreements with different rent, it is possible to trick someone to create a valid signature for a high-rent contract by having him or her sign a low-rent contract.

Infographic | Paper



E-voting

Mathematical Institute | Matl
Election 2015: How feasible
www.bbc.co.uk/news/magazine-32421086

BBC Sign in News Sport Weather iPlayer TV Rad

NEWS

Home | UK | World | Business | Politics | Tech | Science | Health | Education | Entertai

Magazine

Election 2015: How feasible would it be to introduce online voting?

By Tom de Castella
BBC News Magazine

27 April 2015 | Magazine

Mobile Applications

An Empirical Study of Cryptographic Misuse in Android Applications

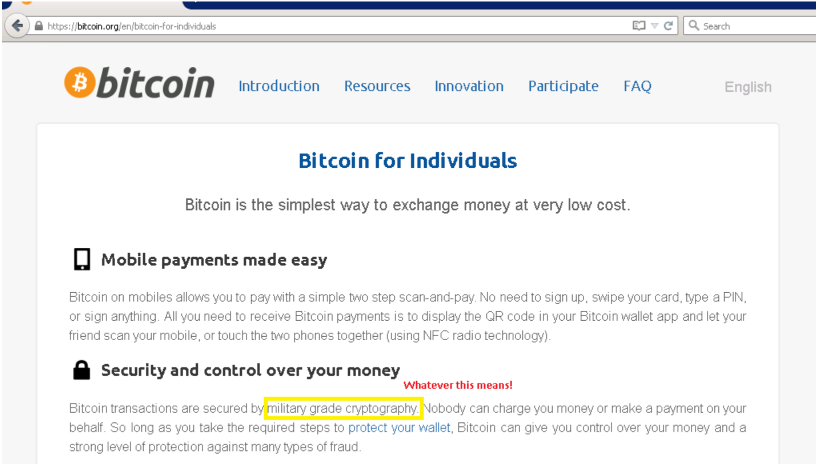
Manuel Egele, David Brumley
Carnegie Mellon University
{megele,dbrumley}@cmu.edu

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.

Bitcoin



https://bitcoin.org/en/

Cryptography Usage in the Real World: a Summary

Cryptography Usage in the Real World: a Summary

- On-line banking, e-commerce (https:)

Cryptography Usage in the Real World: a Summary

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

Cryptography Usage in the Real World: a Summary

- On-line banking, e-commerce (https:)
- SSH: to remotely login and to transfer files.
- Bitcoin: a decentralized digital currency

Cryptography Usage in the Real World: a Summary

- On-line banking, e-commerce (https:)
- SSH: to remotely login and to transfer files.
- Bitcoin: a decentralized digital currency
- Electronic Voting.

Cryptography Usage in the Real World: a Summary

- On-line banking, e-commerce (https:)
- SSH: to remotely login and to transfer files.
- Bitcoin: a decentralized digital currency
- Electronic Voting.
- Emails, cloud computing, etc.

Cryptography Usage in the Real World: a Summary

- On-line banking, e-commerce (https:)
- SSH: to remotely login and to transfer files.
- Bitcoin: a decentralized digital currency
- Electronic Voting.
- Emails, cloud computing, etc.
- Mobile applications.

Cryptography Usage in the Real World: a Summary

- On-line banking, e-commerce (https:)
- SSH: to remotely login and to transfer files.
- Bitcoin: a decentralized digital currency
- Electronic Voting.
- Emails, cloud computing, etc.
- Mobile applications.
- ATM machines, etc.

Outline

Hardness Assumptions

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 $p \neq q$.

Hardness Assumptions

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.

Hardness Assumptions

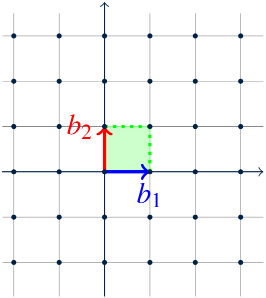
Short Vector Problem in Lattices (SVP)

- A lattice is a discrete version of a vector subspace, more formally;
- Given n *linearly independent* vectors $\vec{b}_1, \ldots, \vec{b}_n \in \mathbb{R}^m$, the lattice generated by them is defined as

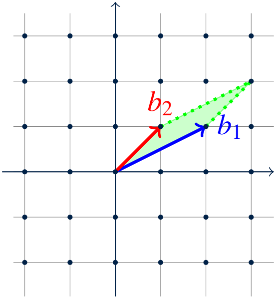
$$\mathcal{L}(\vec{b}_1, \ldots, \vec{b}_n) \stackrel{\text{def}}{=} \left\{ \sum_{i=1}^{n} x_i \vec{b}_i \mid x_i \in \mathbb{Z} \right\}$$

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

Hardness Assumptions



(a) The lattice $\mathbb{Z}^2$ with $B = \{(0,1),(1,0)\}$



(b) The lattice $\mathbb{Z}^2$ with a $B' = \{(1,1),(2,1)\}$

Hardness Assumptions: Average vs. Worst Cases [1]

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

[1] Post-Quantum Cryptography, Daniel Bernstein et al.

Hardness Assumptions: Average vs. Worst Cases [1]

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

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

[1] Post-Quantum Cryptography, Daniel Bernstein et al.

Hardness Assumptions: Average vs. Worst Cases [1]

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

- Breaking a lattice-based cryptographic scheme is at least as hard as solving several hard lattice problems in the worst case.
- Breaking a cryptographic scheme that is based on factoring might imply the ability to factor some numbers chosen according to a certain distribution, but not the ability to factor all numbers!

[1] Post-Quantum Cryptography, Daniel Bernstein et al.

Hardness Assumptions: Average vs. Worst Cases [1]

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

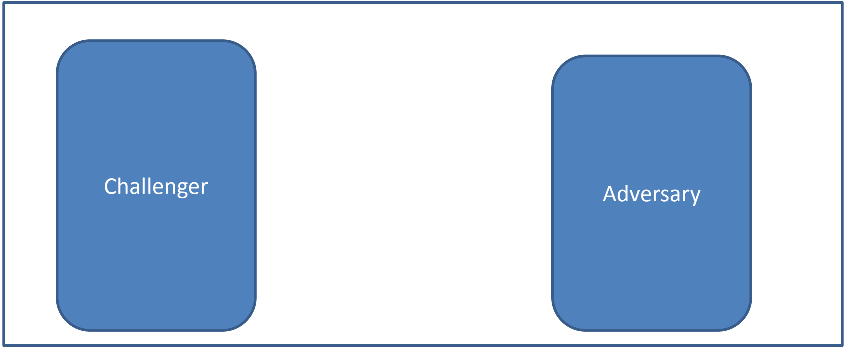
- Breaking a lattice-based cryptographic scheme is at least as hard as solving several hard lattice problems in the worst case.
- Breaking a cryptographic scheme that is based on factoring might imply the ability to factor some numbers chosen according to a certain distribution, but not the ability to factor all numbers!

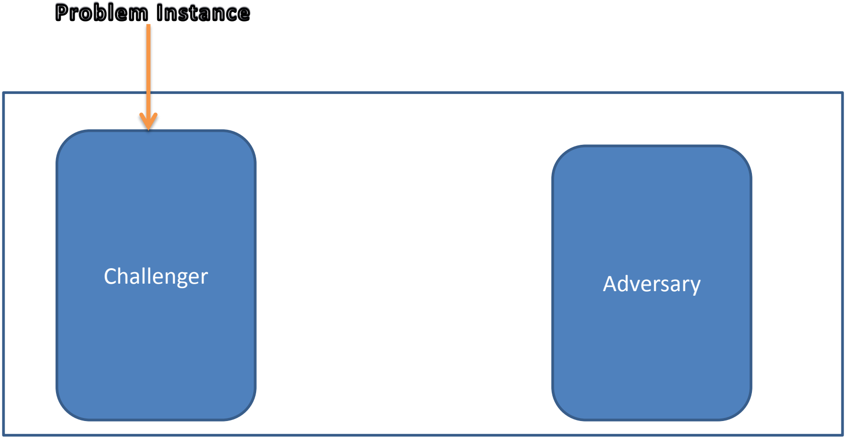
How can we prove the security of our cryptosystems?

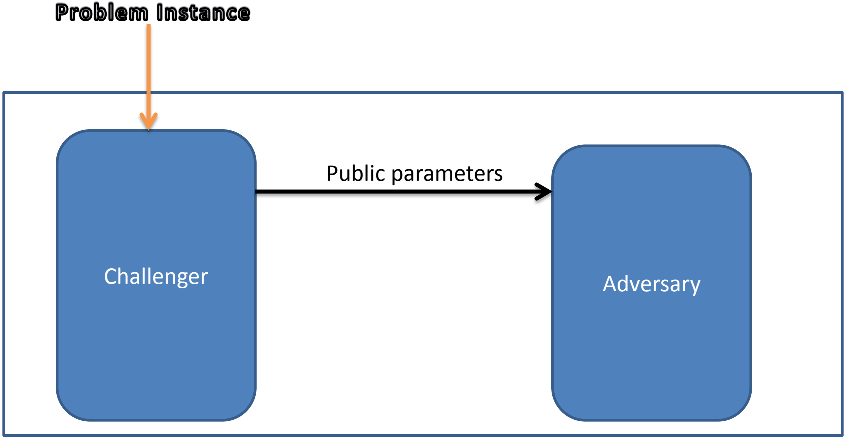
- Proofs by reduction!

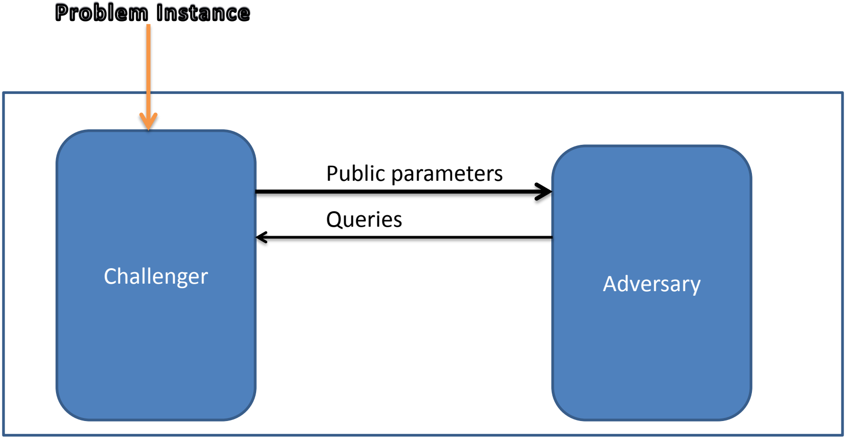
[1] Post-Quantum Cryptography, Daniel Bernstein et al.

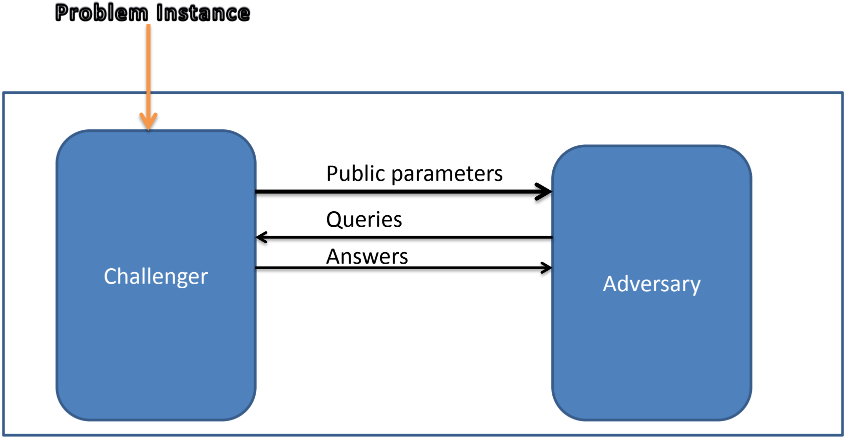
Security Games: Proofs by Reduction

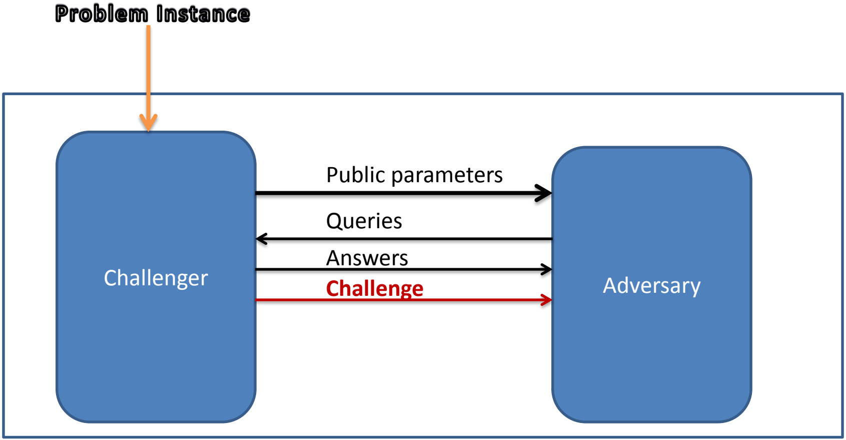


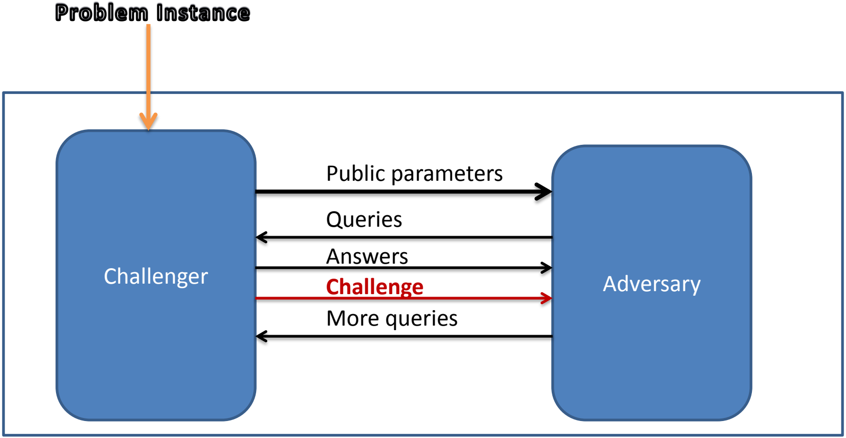

Problem Instance
Challenger
Adversary

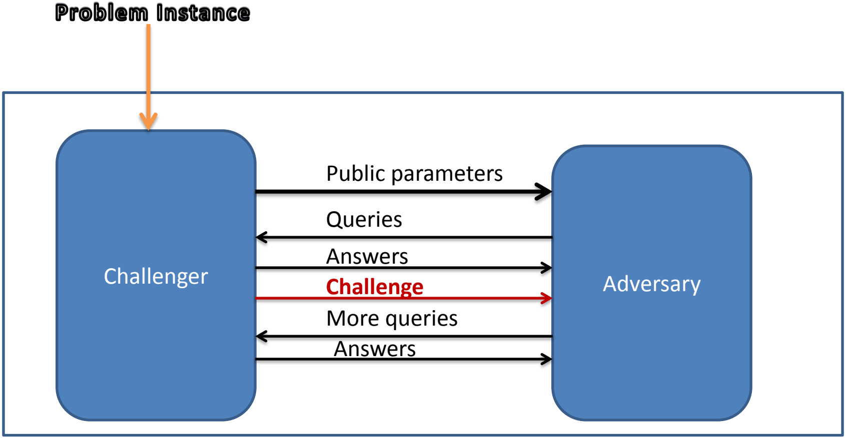

Problem Instance
Public parameters
Challenger
Adversary

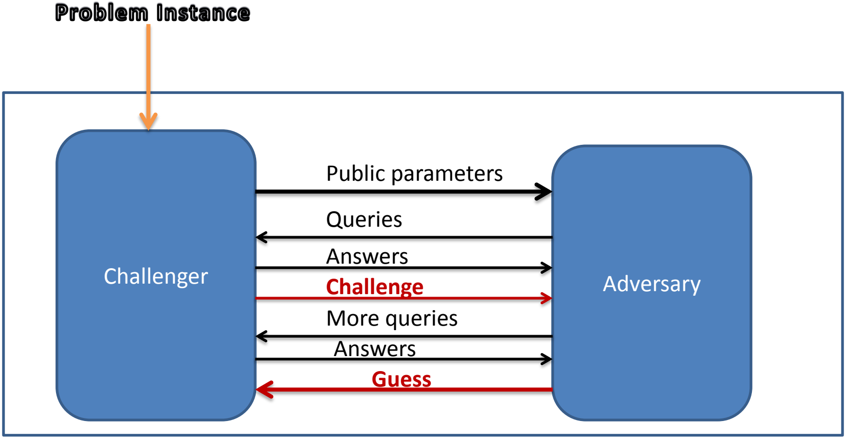

Problem Instance
Public parameters
Queries
Challenger
Adversary

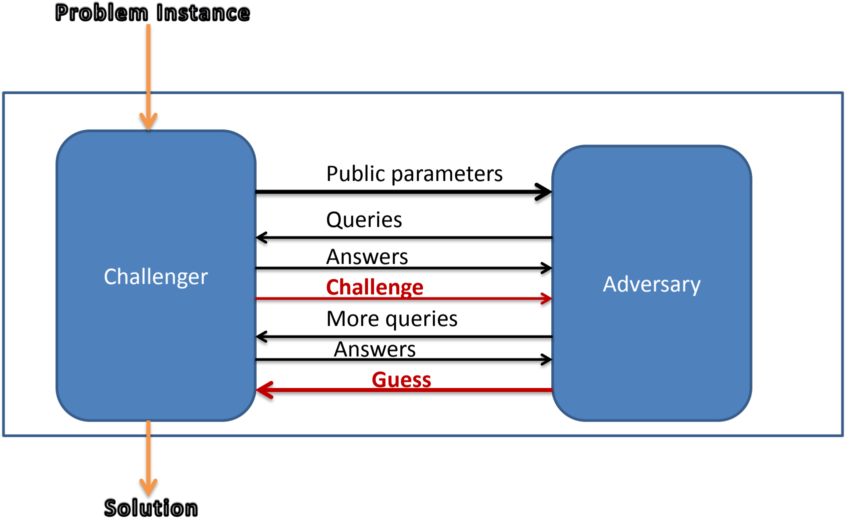

Problem Instance
Challenger
Public parameters
Queries
Answers
Adversary


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

Outline

Symmetric Key Cryptosystems

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

- KeyGen(n):[2] 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:

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

[2] You often equivalently see KeyGen(1^n), which emphasizes that the algorithm runs in time polynomial in the length of its input.

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

1. Cryptography Usage in the Real World: Do we use it? Where?
2. **Modern Cryptography**
   - Provable Security
   - Symmetric Key Cryptosystems
   - Public Key Cryptosystems
   - **Hash Functions**
   - Digital Signatures
   - Advanced Cryptographic Tools/Schemes
3. Classical Vs Post-Quantum Cryptography

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.
- Cryptographic hash functions are much harder to design than those used to build *hash tables* in data structures.

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*.

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*.
- This is how you verify that the update sent by a certain company is authentic.

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*.
- This is how you verify that the update sent by a certain company is authentic.
- In comparison to message authentication codes (MAC);
  - Key distribution and management is hugely simplified.
  - Signatures are *publicly verifiable!*

Outline

Secret Sharing

- Lagrange Interpolating Polynomial: given n points $(x_1, y_1), \cdots, (x_n, y_n)$, one can construct the polynomial $P(x)$ of degree $\leq (n-1)$ that passes through them as follows:

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

Shamir Secret Sharing

- Shamir Secret Sharing works in two phases as follows:
- Distribute the shares: first pick a random polynomial $Q(x) \in \mathbb{F}_p[x]$ of degree $\ell < n$ (where n is the number of participants) s.t. $Q(0) = s$. Then, compute the shares

$$S_i = Q(i) \mod p \text{ for } i = 1, \cdots, n$$

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

Multi-Party Computation

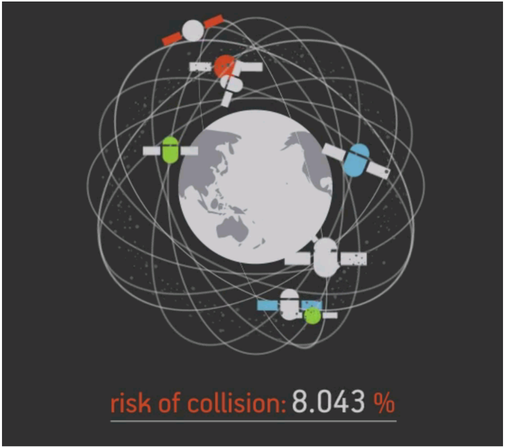
- Suppose that we have n parties $P_1, \cdots, P_n$, each has a secret input s_i. They all want to evaluate a public function f on inputs $(s_1, \cdots, s_n)$ to learn the output and yet keep their inputs hidden from each other.

Multi-Party Computation

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

- Completeness: Prover can always convince a verifier that a given statement is true

Zero-Knowledge Proofs

- Completeness: Prover can always convince a verifier that a given statement is true (A valid vote will always be accepted).

Zero-Knowledge Proofs

- Completeness: Prover can always convince a verifier that a given statement is true (A valid vote will always be accepted).
- Soundness: Prover cannot convince the verifier if the statement is false

Zero-Knowledge Proofs

- Completeness: Prover can always convince a verifier that a given statement is true (A valid vote will always be accepted).
- Soundness: Prover cannot convince the verifier if the statement is false (You cannot fool the verifier and vote with -1000).

Zero-Knowledge Proofs

- Completeness: Prover can always convince a verifier that a given statement is true (A valid vote will always be accepted).
- Soundness: Prover cannot convince the verifier if the statement is false (You cannot fool the verifier and vote with -1000).
- Zero-Knowledge: The proof doesn't reveal any extra information beyond the validity of the statement

Zero-Knowledge Proofs

- Completeness: Prover can always convince a verifier that a given statement is true (A valid vote will always be accepted).
- Soundness: Prover cannot convince the verifier if the statement is false (You cannot fool the verifier and vote with -1000).
- Zero-Knowledge: The proof doesn't reveal any extra information beyond the validity of the statement (The vote is still secret!).

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., $\mathsf{Enc}(A) \times \mathsf{Enc}(B) = \mathsf{Enc}(A \times B)$.
- *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! (using quantum Fourier transform).

Classical Vs Post-Quantum Cryptography

- What would happen to Dlog and Factorisation based Cryptosystems if quantum computers existed?
  - Shor's algorithm! (using quantum Fourier transform).
- Any alternatives?

Classical Vs Post-Quantum Cryptography

- What would happen to Dlog and Factorisation based Cryptosystems if quantum computers existed?
  - Shor's algorithm! (using quantum Fourier transform).
- Any alternatives?
  - 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! (using quantum Fourier transform).
- Any alternatives?
  - 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! (using quantum Fourier transform).
- Any alternatives?
  - 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! (using quantum Fourier transform).
- Any alternatives?
  - Lattice-Based Cryptography (e.g. fully homomorphic encryption)
  - Code-Based Cryptography (e.g. McEliece cryptosystem)
  - Hash-Based Cryptography (e.g. Merkle signature)
  - Multivariate-based Cryptography (e.g. Rainbow signature)

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.