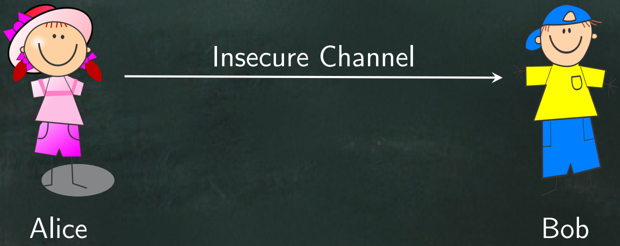
CS 6160 Cryptology Lecture 1: Overview of the Field of Cryptography

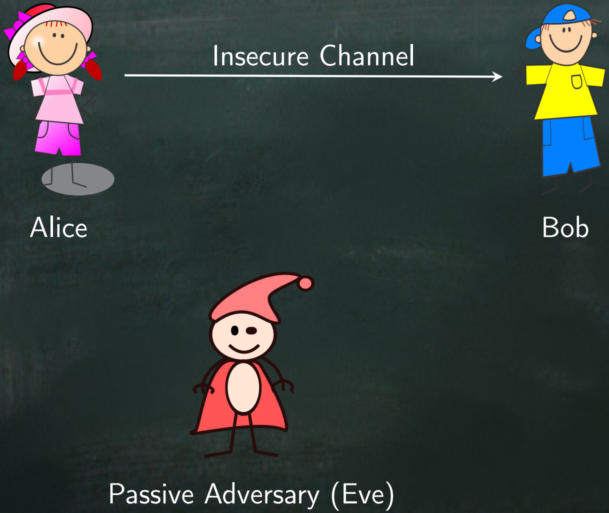
Maria Francis

August 20, 2021

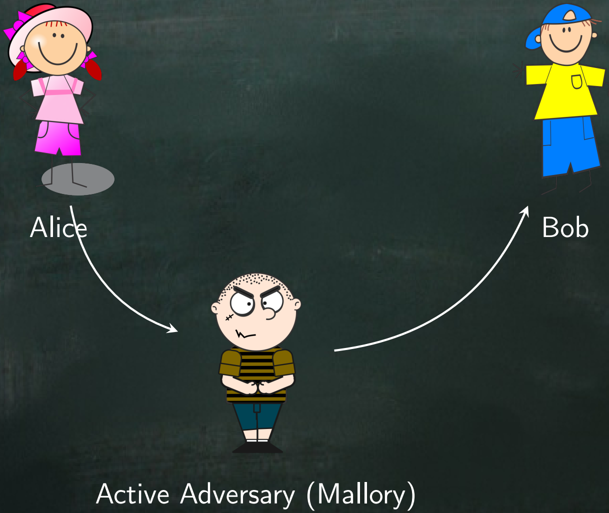
The need for cryptography



The need for cryptography



The need for cryptography



The need for cryptography

- Communicating or computing over a channel where there are adversaries.
- There are information systems(PCs, cellphones, network of computers, ATMs, cars, smart grids, etc.)
- Our aim is to control access to the information – we are looking to see if we can control who sees and modifies the information.
- Some examples of such rules :
  - Only XX can read the contents of the file
  - The contents of this file has not been changed after XX send it.
  - The recipient of this email can authenticate the sender.

Our Aims and the Tools at Hand

What we aim to achieve:

- Data Confidentiality
- Data Integrity
- Authentication –
- Non-repudiation – the sender cannot claim that she/he did not send it

How do we achieve them? Tools such as:

- Encryption
- Hash Functions
- Digital Signatures
- Zero knowledge Proofs

What about the Adversary?

- Could be anybody – insider/outsider
- Assume he knows system design including implementation details
- Resources – powerful computers, ability to intercept messages, ability to collude with some participants
- Generally generous assumptions about adversary's abilities but we assume a computationally bounded adversary.

A Simple Solution

Alice

Key k
Encryption Algorithm Enc
Decryption Algorithm Dec

All previously agreed upon

Bob

A Simple Solution

Alice

$$c = Enc_k(m)$$

Bob

Retrieving the message:
$m = Dec_k(c)$

A Simple Solution

$c = Enc_k(m)$ $\longrightarrow$

Alice

Bob

Retrieving the message:
$m = Dec_k(c)$

Questions: Which of these are secret here?
How are the secrets agreed upon? Parties may not even have met before?

Formalizing the Solution

Symmetric key encryption (SKE) scheme consists of:

- $\mathcal{M}$: a set of possible plaintexts
- $\mathcal{C}$: a set of possible ciphertexts
- $\mathcal{K}$: a set of possible keys.
- Gen (called the key generation algorithm) is a randomized algorithm that returns a key k such that $k \in \mathcal{K}$.
- A family of encryption functions, $Enc_k : \mathcal{M} \to \mathcal{C}, \forall k \in \mathcal{K}$
- A family of decryption functions, $Dec_k : \mathcal{C} \to \mathcal{M}, \forall k \in \mathcal{K}$, such that $Dec_k(Enc_k(m)) = m$ for all $m \in \mathcal{M}$ and $k \in \mathcal{K}$

Timeline of Cryptography as a field

0 2021

Ancient times to 1900s:
Classical
Ciphers

1900s :
Mechanical
Ciphers

1970s:
Modern Ciphers
- Symmetric keys
/Public Key Crypto

Classical Ciphers

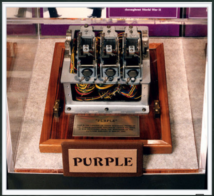
The first idea that comes up when you need secret communication.
Popular upto 1900s :

- Shift/Ceaser cipher
- Substitution cipher
- Vigenère cipher, etc

S	E	N	D	R	E	I	N	F	O	R	C	E	M	E	N	T	S
V	I	G	E	N	E	R	E	V	I	G	E	N	E	R	E	V	I
N	M	T	H	E	I	Z	R	A	W	X	G	R	Q	V	R	O	A

Mechanical Ciphers - Enigma, Purple, etc

- Motor devices
- Electromechanical devices
- Cryptography performed by (typically, rotor) machines.
- Alan Turing and others at Bletchley Park, William Friedman and others in the USA – all helped break these ciphers



Classical Cryptosystems

- As time proceeds the cryptosystems get more sophisticated but they are all broken!

Classical Cryptosystems

- As time proceeds the cryptosystems get more sophisticated but they are all broken!
- The main idea was security by obscurity.
- *Gen*, *Enc*, *Dec* and the generated key k were secret.
- Less information we give to the adversary, the harder it is to break the scheme, right??

Classical Cryptosystems

- As time proceeds the cryptosystems get more sophisticated but they are all broken!
- The main idea was security by obscurity.
- *Gen*, *Enc*, *Dec* and the generated key k were secret.
- Less information we give to the adversary, the harder it is to break the scheme, right??
- Not really! – Kerchoff's principle (1884)

Kerchoff's principle

- The only thing that should be private is the key k, Gen, Enc, Dec should be assumed to be public.

Kerchoff's principle

- The only thing that should be private is the key k, Gen, Enc, Dec should be assumed to be public.
- Conservative approach – guarantees that security is preserved even if everything but the key is known to the adversary.
- First step to formally defining the security of encryption schemes.

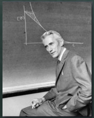
Kerchoff's principle

- The only thing that should be private is the key k, *Gen*, *Enc*, *Dec* should be assumed to be public.
- Conservative approach – guarantees that security is preserved even if everything but the key is known to the adversary.
- First step to formally defining the security of encryption schemes.
- Immediate consequence – all of the algorithms (*Gen*, *Enc*, *Dec*) cannot be deterministic.
- If so, then Eve would be able to compute everything that Alice and Bob could compute and would thus be able to decrypt anything that Bob can decrypt.

Kerchoff's principle

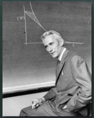
- The only thing that should be private is the key k, *Gen*, *Enc*, *Dec* should be assumed to be public.
- Conservative approach – guarantees that security is preserved even if everything but the key is known to the adversary.
- First step to formally defining the security of encryption schemes.
- Immediate consequence – all of the algorithms (*Gen*, *Enc*, *Dec*) cannot be deterministic.
- If so, then Eve would be able to compute everything that Alice and Bob could compute and would thus be able to decrypt anything that Bob can decrypt.
- *To prevent this we require Gen to be randomized.*

Perfect Secrecy (C. Shannon (1916 - 2001))



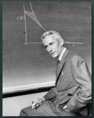
- Can we perfect secrecy (information theoretic security)? -

Perfect Secrecy (C. Shannon (1916 - 2001))



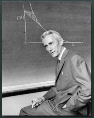
- Can we perfect secrecy (information theoretic security)? - i.e, adversary has infinite computational resources and still not be able to break it?

Perfect Secrecy (C. Shannon (1916 - 2001))



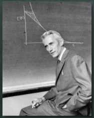
- Can we perfect secrecy (information theoretic security)? - i.e, adversary has infinite computational resources and still not be able to break it?
- Yes!

Perfect Secrecy (C. Shannon (1916 - 2001))



- Can we perfect secrecy (information theoretic security)? - i.e, adversary has infinite computational resources and still not be able to break it?
- Yes!
  - Key is perfectly random
  - Key is at least as long as the message
  - Key is never re-used!

Perfect Secrecy (C. Shannon (1916 - 2001))



- Can we perfect secrecy (information theoretic security)? - i.e, adversary has infinite computational resources and still not be able to break it?
- Yes!
  - Key is perfectly random
  - Key is at least as long as the message
  - Key is never re-used!
- One Time Pad/ Vernam Cipher - unbreakable!

Computational Complexity Theory

- A systematic study of what computationally bounded parties can and cannot do
- Started with Alan Turing but formal study with Cook (Turing Award '82), Karp (Turing Award '85) and Blum (Turing Award '95)
- Notions that are critical in this area – polynomial time reductions, NP-completeness

Computational Complexity Theory

- A systematic study of what computationally bounded parties can and cannot do
- Started with Alan Turing but formal study with Cook (Turing Award '82), Karp (Turing Award '85) and Blum (Turing Award '95)
- Notions that are critical in this area – polynomial time reductions, NP-completeness
- Many ideas are still "conjectured" or believed

Computational Complexity Theory

- A systematic study of what computationally bounded parties can and cannot do
- Started with Alan Turing but formal study with Cook (Turing Award '82), Karp (Turing Award '85) and Blum (Turing Award '95)
- Notions that are critical in this area – polynomial time reductions, NP-completeness
- Many ideas are still "conjectured" or believed
- The foundation of modern cryptography

Computationally Bounded Adversaries

- Shannon's results made us feel that encryption can succeed only if the key size is at least as long as the message!

Computationally Bounded Adversaries

- Shannon's results made us feel that encryption can succeed only if the key size is at least as long as the message!
- Key thing is assumption!

Computationally Bounded Adversaries

- Shannon's results made us feel that encryption can succeed only if the key size is at least as long as the message!
- Key thing is assumption! Eve has unbounded computing power - too strong!

Computationally Bounded Adversaries

- Shannon's results made us feel that encryption can succeed only if the key size is at least as long as the message!
- Key thing is assumption! Eve has unbounded computing power - too strong!
- So we make a reasonable assumption – Eve has only probabilistic polynomial time (PPT) computing power.

Computationally Bounded Adversaries

- Shannon's results made us feel that encryption can succeed only if the key size is at least as long as the message!
- Key thing is assumption! Eve has unbounded computing power - too strong!
- So we make a reasonable assumption – Eve has only probabilistic polynomial time (PPT) computing power. To be fair, we restrict Alice and Bob to PPT as well.

Probabilistic Polynomial Time

Definition (Polynomial Time Algorithm)

If an algorithm A gets an input of size k it is considered polynomial time if it runs in $O(k^c)$ time where c is a constant.

Probabilistic Polynomial Time

Definition (Polynomial Time Algorithm)

If an algorithm A gets an input of size k it is considered polynomial time if it runs in $O(k^c)$ time where c is a constant.

We write $y = A(x)$ to denote the output of A on input x.

Probabilistic Polynomial Time

Definition (Polynomial Time Algorithm)

If an algorithm A gets an input of size k it is considered polynomial time if it runs in $O(k^c)$ time where c is a constant.

We write $y = A(x)$ to denote the output of A on input x.

Definition (Probabilistic Polynomial Time Algorithm (PPT))

It is a polynomial time algorithm A that is randomized.

Probabilistic Polynomial Time

Definition (Polynomial Time Algorithm)

If an algorithm A gets an input of size k it is considered polynomial time if it runs in $O(k^c)$ time where c is a constant.

We write $y = A(x)$ to denote the output of A on input x.

Definition (Probabilistic Polynomial Time Algorithm (PPT))

It is a polynomial time algorithm A that is randomized.

- What does that mean? We flip coins during the computation of the algorithm and the output depends on the coin toss outcome.
- $y \leftarrow A(x)$ is the random variable y, the randomized output of A on input x, i.e. r was chosen at random and $y = A(x; r)$ was computed.

Probabilistic Polynomial Time

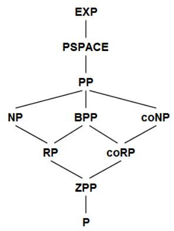
- Ability to toss coins may provide additional power – if the scheme is secure against PPT adversaries, then it is secure against deterministic poly-times ones.

Probabilistic Polynomial Time

- Ability to toss coins may provide additional power – if the scheme is secure against PPT adversaries, then it is secure against deterministic poly-times ones.
- It is " conjectured" that class of problems solvable by probabilistic PT machines is strictly larger than the class of problems solvable by deterministic ones, i.e. $\mathrm{P} \subsetneq \mathrm{BPP}$.

Probabilistic Polynomial Time

- Ability to toss coins may provide additional power – if the scheme is secure against PPT adversaries, then it is secure against deterministic poly-times ones.
- It is " conjectured" that class of problems solvable by probabilistic PT machines is strictly larger than the class of problems solvable by deterministic ones, i.e. P $\subsetneq$ BPP.



Another Assumption

- Eve should only have negligible chance to guess the value and Eve cannot break the system with significant probability even after poly no of chances.

Another Assumption

- Eve should only have negligible chance to guess the value and Eve cannot break the system with significant probability even after poly no of chances.

Definition ($\mathrm{negl}(k)$)

A function $v(k)$ is called negligible, $\mathrm{negl}(k)$, if:

$$(\forall c > 0)(\exists k^{'})(\forall k \geq k^{'})[v(k) < \frac{1}{k^c}]$$

Another Assumption

- Eve should only have negligible chance to guess the value and Eve cannot break the system with significant probability even after poly no of chances.

Definition (negl(k))

A function $v(k)$ is called negligible, negl(k), if:

$$(\forall c > 0)(\exists k')(\forall k \geq k')[v(k) < \frac{1}{k^c}]$$

A negl function is one that is asymptotically smaller than any inverse poly. function.

Another Assumption

- Eve should only have negligible chance to guess the value and Eve cannot break the system with significant probability even after poly no of chances.

Definition (negl(k))

A function $v(k)$ is called negligible, negl(k), if:

$$(\forall c > 0)(\exists k^{'})(\forall k \geq k^{'})[v(k) < \frac{1}{k^c}]$$

A negl function is one that is asymptotically smaller than any inverse poly. function.
2^{-k}, k^{-logk} are negl but $1/k^{1000}$ is not.
Negligible function stays negligible even after poly. attempts : $\mathrm{poly}(k) \cdot \mathrm{negl}(k) = \mathrm{negl}(k)$.

Security Parameter

- It is a measure of both the resource requirements of the cryptographic protocol/algorithm as well as the probability of the adversary breaking the system.

Security Parameter

- It is a measure of both the resource requirements of the cryptographic protocol/algorithm as well as the probability of the adversary breaking the system.
- Represented as k and security guarantees are given asymptotically as a function of k.

Security Parameter

- It is a measure of both the resource requirements of the cryptographic protocol/algorithm as well as the probability of the adversary breaking the system.
- Represented as k and security guarantees are given asymptotically as a function of k.
- One example is key size given in bits. For RSA it is the length of the modulus n in bits.

Security Parameter

- It is a measure of both the resource requirements of the cryptographic protocol/algorithm as well as the probability of the adversary breaking the system.
- Represented as k and security guarantees are given asymptotically as a function of k.
- One example is key size given in bits. For RSA it is the length of the modulus n in bits.
- Alice and Bob and the adversary are assumed to run in time polynomial in k and the adversary can break the system with probability negligible in k.

Security Parameter

- It is a measure of both the resource requirements of the cryptographic protocol/algorithm as well as the probability of the adversary breaking the system.
- Represented as k and security guarantees are given asymptotically as a function of k.
- One example is key size given in bits. For RSA it is the length of the modulus n in bits.
- Alice and Bob and the adversary are assumed to run in time polynomial in k and the adversary can break the system with probability negligible in k.
- Increasing k we get higher security but a degraded efficiency. What is the correct value of k?

1970s - Public Key Revolution

- Merkle, and independently Hellman and Diffie, invented the notion of public-key cryptography.
- In November 1976, Diffie and Hellman published New Directions in Cryptography, proclaiming *We are at the brink of a revolution in cryptography.*

PKE - Diffie/Hellman Ideas

- Every party A has a public key PK_A and a private key SK_A.

PKE - Diffie/Hellman Ideas

- Every party A has a public key PK_A and a private key SK_A.
- Anybody who wants to send a message m to A uses PK_A to encrypt:

$$c = Enc_{PK_A}(m).$$

- A will use SK_A to decrypt:

$$m = Dec_{SK_A}(c).$$

PKE - Diffie/Hellman Ideas

- Every party A has a public key PK_A and a private key SK_A.
- Anybody who wants to send a message m to A uses PK_A to encrypt:

$$c = Enc_{PK_A}(m).$$

- A will use SK_A to decrypt:

$$m = Dec_{SK_A}(c).$$

- Should be efficient to compute PK_A, SK_A pairs.

PKE - Diffie/Hellman Ideas

- Every party A has a public key PK_A and a private key SK_A.
- Anybody who wants to send a message m to A uses PK_A to encrypt:

$$c = Enc_{PK_A}(m).$$

- A will use SK_A to decrypt:

$$m = Dec_{SK_A}(c).$$

- Should be efficient to compute PK_A, SK_A pairs.
- Publishing PK_A does not give any information about SK_A – computationally infeasible to get SK_A from PK_A.

PKE - Diffie/Hellman Ideas

- Every party A has a public key PK_A and a private key SK_A.
- Anybody who wants to send a message m to A uses PK_A to encrypt:

$$c = Enc_{PK_A}(m).$$

- A will use SK_A to decrypt:

$$m = Dec_{SK_A}(c).$$

- Should be efficient to compute PK_A, SK_A pairs.
- Publishing PK_A does not give any information about SK_A – computationally infeasible to get SK_A from PK_A.
- Digital Signatures – sign with SK_A and verify with PK_A.

Public Key Encryption

c

Alice

Bob

Encrypting
using Bob's public key:
$c = Enc_{PK_B}(m)$

Decrypting
using Bob's private key:
$m = Dec_{SK_B}(c)$

Digital Signatures using Public Keys

(m, s)

Bob

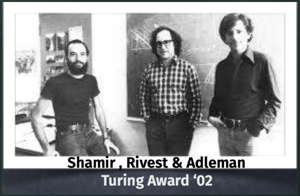
Alice

Signing
using Bob's private key:
$s = Sign_{SK_B}(m)$

Verifying
using Bob's public key:
$Verify_{PK_B}(s, m)$

Public Key Encryption – RSA

How to implement Diffie/Hellman ideas? – Rivest, Shamir, Adleman in 1977 came with the RSA algorithm.



Public Key Encryption – RSA

How to implement Diffie/Hellman ideas? – Rivest, Shamir, Adleman in 1977 came with the RSA algorithm.

Shamir , Rivest & Adleman
Turing Award '02

Note: In 1999, it was revealed that Ellis, Cocks and Williamson had invented PKC in the British secret service, before their invention outside.

Public Key Encryption – RSA

- Idea : reduce the problem of finding the secret key to some known hard mathematical problem.

Public Key Encryption – RSA

- Idea : reduce the problem of finding the secret key to some known hard mathematical problem.
- RSA security relies (in part) on the conjectured hardness of factoring n, a product of two very large primes p, q.

Public Key Encryption – RSA

- Idea : reduce the problem of finding the secret key to some known hard mathematical problem.
- RSA security relies (in part) on the conjectured hardness of factoring n, a product of two very large primes p, q.
- What we look for in any PKC are one-way functions with a trapdoor:
  - Function f must be invertible so as to decrypt encrypted messages.
  - Efficient to encrypt
  - Difficult to invert so that Eve cannot compute m knowing $f(m)$
  - Has a trapdoor: given some information (SK_A) finding m is easy given $f(m)$.

PKC revolution

- Huge implications - exponential rise in study and usage of cryptography.

PKC revolution

- Huge implications - exponential rise in study and usage of cryptography.
- Wide scale deployment of crypto systems

PKC revolution

- Huge implications - exponential rise in study and usage of cryptography.
- Wide scale deployment of crypto systems
- More Turing award winners in theoretical crypto - Goldwasser & Micali (2012)

PKC revolution

- Huge implications - exponential rise in study and usage of cryptography.
- Wide scale deployment of crypto systems
- More Turing award winners in theoretical crypto - Goldwasser & Micali (2012)
- More interesting ideas like for e.g:
  - secret sharing : collaboration between distrusting parties
  - zero knowledge proofs (ZKPs) : revealing nothing but the validity of the statement
  - fully homomorphic encryption : computation on encrypted data
- IACR - Intl. Assn. for Crypto Research. Sponsors the big crypto conferences – Crypto, Eurocrypt and Asiacrypt

Principles of Modern Cryptography

- The aim is to move away from art of solving codes to science/mathematics for securing digital information against adversarial attacks.

Principles of Modern Cryptography

- The aim is to move away from art of solving codes to science/mathematics for securing digital information against adversarial attacks.
-
- Principle 1 - Formal Definitions

Principles of Modern Cryptography

- The aim is to move away from art of solving codes to science/mathematics for securing digital information against adversarial attacks.
-
- Principle 1 - Formal Definitions
- Clear description of threat model and security guarantees before the design process begins.

Principles of Modern Cryptography

- The aim is to move away from art of solving codes to science/mathematics for securing digital information against adversarial attacks.
-
- Principle 1 - Formal Definitions
- Clear description of threat model and security guarantees before the design process begins.
  - You need to know what you have to achieve before beginning!

Principles of Modern Cryptography

- The aim is to move away from art of solving codes to science/mathematics for securing digital information against adversarial attacks.
-
- Principle 1 - Formal Definitions
- Clear description of threat model and security guarantees before the design process begins.
  - You need to know what you have to achieve before beginning! You do not want to work with intuitive ideas.

Principles of Modern Cryptography

- The aim is to move away from art of solving codes to science/mathematics for securing digital information against adversarial attacks.
-
- Principle 1 - Formal Definitions
- Clear description of threat model and security guarantees before the design process begins.
  - You need to know what you have to achieve before beginning! You do not want to work with intuitive ideas.
- It helps analyze and evaluate the scheme and some cases prove security too!

Formal Definitions - An example

- Example : What does one mean by secure encryption?
- *It should be impossible for an attacker to recover the key.*

Formal Definitions - An example

- Example : What does one mean by secure encryption?
- *It should be impossible for an attacker to recover the key.* $Enc_k(m) = m$ is not secure.

Formal Definitions - An example

- Example : What does one mean by secure encryption?
- *It should be impossible for an attacker to recover the key.* $Enc_k(m) = m$ is not secure.
- *It should be impossible for an attacker to recover the entire plaintext from the ciphertext.*

Formal Definitions - An example

- Example : What does one mean by secure encryption?
- *It should be impossible for an attacker to recover the key.* $Enc_k(m) = m$ is not secure.
- *It should be impossible for an attacker to recover the entire plaintext from the ciphertext.* Not good enough!

Formal Definitions - An example

- Example : What does one mean by secure encryption?
- *It should be impossible for an attacker to recover the key.* $Enc_k(m) = m$ is not secure.
- *It should be impossible for an attacker to recover the entire plaintext from the ciphertext.* Not good enough!
- *It should be impossible for an attacker to recover any character of the plaintext from the ciphertext.*

Formal Definitions - An example

- Example : What does one mean by secure encryption?
- *It should be impossible for an attacker to recover the key.* $Enc_k(m) = m$ is not secure.
- *It should be impossible for an attacker to recover the entire plaintext from the ciphertext.* Not good enough!
- *It should be impossible for an attacker to recover any character of the plaintext from the ciphertext.* What about some relations?

Formal Definitions - An example

- Example : What does one mean by secure encryption?
- *It should be impossible for an attacker to recover the key.* $Enc_k(m) = m$ is not secure.
- *It should be impossible for an attacker to recover the entire plaintext from the ciphertext.* Not good enough!
- *It should be impossible for an attacker to recover any character of the plaintext from the ciphertext.* What about some relations?
- A right answer: *regardless of any information an attacker already has, a ciphertext should leak no additional information about the underlying plaintext.*

Formal Definitions - An example

- Example : What does one mean by secure encryption?
- *It should be impossible for an attacker to recover the key.* $Enc_k(m) = m$ is not secure.
- *It should be impossible for an attacker to recover the entire plaintext from the ciphertext.* Not good enough!
- *It should be impossible for an attacker to recover any character of the plaintext from the ciphertext.* What about some relations?
- A right answer: *regardless of any information an attacker already has, a ciphertext should leak no additional information about the underlying plaintext.*
- What is prior knowledge? What is leak?

Principle 2 - Precise Assumptions

- Typically security is not proved unconditionally as we have seen in this lecture so we rely on assumptions.

Principle 2 - Precise Assumptions

- Typically security is not proved unconditionally as we have seen in this lecture so we rely on assumptions.
- They are not proven but conjectured to be true.

Principle 2 - Precise Assumptions

- Typically security is not proved unconditionally as we have seen in this lecture so we rely on assumptions.
- They are not proven but conjectured to be true.
- This means the assumptions should be examined and tested to be true.

Principle 2 - Precise Assumptions

- Typically security is not proved unconditionally as we have seen in this lecture so we rely on assumptions.
- They are not proven but conjectured to be true.
- This means the assumptions should be examined and tested to be true.
- If it is not refuted for many years our confidence in it is increased.

Principle 2 - Precise Assumptions

- Typically security is not proved unconditionally as we have seen in this lecture so we rely on assumptions.
- They are not proven but conjectured to be true.
- This means the assumptions should be examined and tested to be true.
- If it is not refuted for many years our confidence in it is increased. This means precisely and simply stating it or else no one will study it!

Principle 2 - Precise Assumptions

- Typically security is not proved unconditionally as we have seen in this lecture so we rely on assumptions.
- They are not proven but conjectured to be true.
- This means the assumptions should be examined and tested to be true.
- If it is not refuted for many years our confidence in it is increased. This means precisely and simply stating it or else no one will study it!
- Assumptions give us a way of comparing two schemes based on two different assumptions.

Principle 2 - Precise Assumptions

- Typically security is not proved unconditionally as we have seen in this lecture so we rely on assumptions.
- They are not proven but conjectured to be true.
- This means the assumptions should be examined and tested to be true.
- If it is not refuted for many years our confidence in it is increased. This means precisely and simply stating it or else no one will study it!
- Assumptions give us a way of comparing two schemes based on two different assumptions.
- If an assumption is broken (like factoring on a quantum computer) the schemes built on the assumption (such as RSA, ECDSA) will break!

Principle 2 - Proofs of Security

- Rigorous proof shows a scheme satisfies a certain definition under certain assumptions.

Principle 2 - Proofs of Security

- Rigorous proof shows a scheme satisfies a certain definition under certain assumptions. I.e. no adversary with certain specified resources can break the scheme because that would entail the underlying assumption to be false.

Principle 2 - Proofs of Security

- Rigorous proof shows a scheme satisfies a certain definition under certain assumptions. I.e. no adversary with certain specified resources can break the scheme because that would entail the underlying assumption to be false.
- Cryptography is still an art!

Principle 2 - Proofs of Security

- Rigorous proof shows a scheme satisfies a certain definition under certain assumptions. I.e. no adversary with certain specified resources can break the scheme because that would entail the underlying assumption to be false.
- Cryptography is still an art! Creativity needed for developing new definitions, new schemes and proving their security.

Principle 2 - Proofs of Security

- Rigorous proof shows a scheme satisfies a certain definition under certain assumptions. I.e. no adversary with certain specified resources can break the scheme because that would entail the underlying assumption to be false.
- Cryptography is still an art! Creativity needed for developing new definitions, new schemes and proving their security. and for attacking deployed cryptosystems even if they are proven secure.

Principle 2 - Proofs of Security

- Rigorous proof shows a scheme satisfies a certain definition under certain assumptions. I.e. no adversary with certain specified resources can break the scheme because that would entail the underlying assumption to be false.
- Cryptography is still an art! Creativity needed for developing new definitions, new schemes and proving their security. and for attacking deployed cryptosystems even if they are proven secure. I.e. there is a difference between provable security and real-world security.

Principle 2 - Proofs of Security

- Rigorous proof shows a scheme satisfies a certain definition under certain assumptions. I.e. no adversary with certain specified resources can break the scheme because that would entail the underlying assumption to be false.
- Cryptography is still an art! Creativity needed for developing new definitions, new schemes and proving their security. and for attacking deployed cryptosystems even if they are proven secure. I.e. there is a difference between provable security and real-world security.
- Not a drawback. It means the definitions or assumptions are being broken!

Principle 2 - Proofs of Security

- Rigorous proof shows a scheme satisfies a certain definition under certain assumptions. I.e. no adversary with certain specified resources can break the scheme because that would entail the underlying assumption to be false.
- Cryptography is still an art! Creativity needed for developing new definitions, new schemes and proving their security. and for attacking deployed cryptosystems even if they are proven secure. I.e. there is a difference between provable security and real-world security.
- Not a drawback. It means the definitions or assumptions are being broken!
- Pegasus shows us that well-resourced targeted attack is still impossible to prevent! Read : `https://blog.cryptographyengineering.com/2021/07/20/a-case-against-security-nihilism/`.

The future of the field

- Post Quantum Cryptography – In 1994, Peter Shor invented a fast factorization algorithm that runs in polynomial time on a (hypothetical) quantum computer.

The future of the field

- Post Quantum Cryptography – In 1994, Peter Shor invented a fast factorization algorithm that runs in polynomial time on a (hypothetical) quantum computer. So now we need alternatives to RSA.

The future of the field

- Post Quantum Cryptography – In 1994, Peter Shor invented a fast factorization algorithm that runs in polynomial time on a (hypothetical) quantum computer. So now we need alternatives to RSA.
- Security over Blockchains (using ZKPs)

The future of the field

- Post Quantum Cryptography – In 1994, Peter Shor invented a fast factorization algorithm that runs in polynomial time on a (hypothetical) quantum computer. So now we need alternatives to RSA.
- Security over Blockchains (using ZKPs)
- Voting systems

The future of the field

- Post Quantum Cryptography – In 1994, Peter Shor invented a fast factorization algorithm that runs in polynomial time on a (hypothetical) quantum computer. So now we need alternatives to RSA.
- Security over Blockchains (using ZKPs)
- Voting systems
- Attacks on different cryptographic primitives like hash functions.

Conclusion

- Cryptography is not THE solution to security issues.

Conclusion

- Cryptography is not THE solution to security issues.
- Security involves a lot of other things like practical and good software and hardware engineering, policy, human factors, etc.

Conclusion

- Cryptography is not THE solution to security issues.
- Security involves a lot of other things like practical and good software and hardware engineering, policy, human factors, etc.
- Cryptography is an essential component of any solution when correctly used can help us greatly enhance security.

Conclusion

- Cryptography is not THE solution to security issues.
- Security involves a lot of other things like practical and good software and hardware engineering, policy, human factors, etc.
- Cryptography is an essential component of any solution when correctly used can help us greatly enhance security.
- Lots of cryptographic tools are not used in practical systems.

Conclusion

- Cryptography is not THE solution to security issues.
- Security involves a lot of other things like practical and good software and hardware engineering, policy, human factors, etc.
- Cryptography is an essential component of any solution when correctly used can help us greatly enhance security.
- Lots of cryptographic tools are not used in practical systems.
- But all in all, cryptography is an interesting field that borrows abstract ideas from mathematics and builds secure systems using the tools of hardware and software engineering.

Conclusion

- Cryptography is not THE solution to security issues.
- Security involves a lot of other things like practical and good software and hardware engineering, policy, human factors, etc.
- Cryptography is an essential component of any solution when correctly used can help us greatly enhance security.
- Lots of cryptographic tools are not used in practical systems.
- But all in all, cryptography is an interesting field that borrows abstract ideas from mathematics and builds secure systems using the tools of hardware and software engineering.
- And of course, there is cryptanalysis!