Introduction to Security Reduction

Lecture 3: Preliminaries

(Hard Problem and Secure Scheme)

Adversary

My IQ is up to 186.

My interest is breaking schemes.

You want me to help you solve problem?

Fool me first!

Journal of WoCrypt

Lecture 12: Flaws in Papers
Lecture 11: Revision of Security Reduction
Lecture 10: Security Proofs for Encryption (Computational)
Lecture 9: Security Proofs for Encryption (Decisional)
Lecture 8: Security Proofs for Digital Signatures
Lecture 7: Analysis (Towards A Correct Reduction)
Lecture 6: Simulation and Solution
Lecture 5: Difficulties in Security Reduction
Lecture 4: Entry to Security Reduction
Lecture 3: Preliminaries (Hard Problem and Secure Scheme)
Lecture 2: Preliminaries (Field, Group, Pairing, and Hash Function)
Lecture 1: Definitions (Algorithm and Security Model)

Computational Complexity Theory

Outline

1 Preliminaries
- Computational and Decisional
- Deterministic and Probabilistic
- Polynomial and Exponential
- Negl. and Non-Negl.
- Probability and Advantage

2 Hard Problem
- Hardness Definition
- Examples
- Hardness Analysis

3 Secure Scheme
- Security Parameter
- Security Level
- Definition of Security

Outline

1 Preliminaries
- Computational and Decisional
- Deterministic and Probabilistic
- Polynomial and Exponential
- Negl. and Non-Negl.
- Probability and Advantage

2 Hard Problem
- Hardness Definition
- Examples
- Hardness Analysis

3 Secure Scheme
- Security Parameter
- Security Level
- Definition of Security

Outline

1 Preliminaries
- Computational and Decisional
- Deterministic and Probabilistic
- Polynomial and Exponential
- Negl. and Non-Negl.
- Probability and Advantage

2 Hard Problem
- Hardness Definition
- Examples
- Hardness Analysis

3 Secure Scheme
- Security Parameter
- Security Level
- Definition of Security

Computing Problem

A computing problem defined over a mathematical primitive is a mathematical object representing certain questions and answers.

- It has an infinite number of instance-solution pairs (x, y).
- x is called instance and y is called solution.
- In cryptography, we consider those pairs with the same length $|x|$.

We propose algorithms to solve computing problems. The computing cost (i.e. time cost) of finding y increases when the length of x grows.

Computing problems = computational problems + decisional problems.

Computational Problem

In a computational problem,

- The solution y to the problem instance x is from a large space. For example, the space size is up to $2^{|x|}$.
- CDH problem: instance $x = (g, g^a, g^b) \in \mathbb{G}$; solution $y = g^{ab} \in \mathbb{G}$.
- The computational problem is also called search problem in computational complexity.

Decisional Problem

In a decisional problem,

- The solution y to the problem instance x is from the set $\{0, 1\}$ with only two answers.
- If $y = 1$, the instance x is also called true instance. Otherwise, $y = 0$ and x is called false instance.
- DDH problem: instance $x = (g, g^a, g^b, Z) \in \mathbb{G}$; solution y

$$y = \begin{cases} 1 & Z = g^{ab} \\ 0 & Z \neq g^{ab} \end{cases}$$

- The set containing all true instances is a formal language L and the decisional problem is to decide whether $x \in L$.

Outline

1 Preliminaries
- Computational and Decisional
- Deterministic and Probabilistic
- Polynomial and Exponential
- Negl. and Non-Negl.
- Probability and Advantage

2 Hard Problem
- Hardness Definition
- Examples
- Hardness Analysis

3 Secure Scheme
- Security Parameter
- Security Level
- Definition of Security

Deterministic and Probabilistic

All algorithms can be classified into deterministic algorithms and probabilistic algorithms.

- A deterministic algorithm is an algorithm where, given as an input a problem instance x, it will always return a correct result (solution y).
- A probabilistic (randomized) algorithm is an algorithm where given as an input a problem instance, it will return a correct result with some likelihood.
- Probabilistic algorithms are believed to be more efficient than deterministic algorithms in solving problems

We denote by (t, ϵ) that an algorithm returns a correct result in time t with success probability ϵ.

Note: We also call the same symbol ϵ in (t, ϵ) advantage in some scenarios.

Algorithms Classifications

All objects (construct scheme, break scheme, solve problem, and program reduction) are all to propose algorithms.

In cryptography, algorithms can be classified into

- **Scheme Algorithm.** To construct a cryptosystem (scheme).
- **Attack Algorithm.** To break a scheme.
- **Solution Algorithm.** To solve a hard problem.
- **Reduction Algorithm.** To describe a security reduction.

Note: These are dummy concepts for understanding security reduction.

Outline

1 Preliminaries
- Computational and Decisional
- Deterministic and Probabilistic
- Polynomial and Exponential
- Negl. and Non-Negl.
- Probability and Advantage

2 Hard Problem
- Hardness Definition
- Examples
- Hardness Analysis

3 Secure Scheme
- Security Parameter
- Security Level
- Definition of Security

Polynomial and Exponential (1)

Let $f(\lambda)$ be a function in positive number λ.

- Polynomial means that there exists $c, k > 0$ such that

$$f(\lambda) \leq c \cdot \lambda^k.$$

- Exponential means that there exists $c > 0$ such that

$$2^{c \cdot \lambda} \leq f(\lambda)$$

Note: Computational complexity has different definitions on polynomial and exponential but the above is what we want in cryptography.

Polynomial and Exponential (2)

Let $f(\lambda)$ be a function in positive number λ.

- In cryptography, we care time cost and hence care polynomial time and exponential time.
- Polynomial time should be exactly understood as "polynomial speed". How fast $f(\lambda)$ increases when the length of λ grows.

Note: 2^{160} is not an exponential time but 2^{λ} is!

Outline

1 Preliminaries
- Computational and Decisional
- Deterministic and Probabilistic
- Polynomial and Exponential
- Negl. and Non-Negl.
- Probability and Advantage

2 Hard Problem
- Hardness Definition
- Examples
- Hardness Analysis

3 Secure Scheme
- Security Parameter
- Security Level
- Definition of Security

Negligible and Non-Negligible

- The function $negl(\lambda)$ is negligible if
  - $negl(\lambda) = \frac{1}{f(\lambda)}$, and
  - $f(\lambda)$ cannot be bounded by any polynomial in λ (super-polynomial).
- The function $non\text{-}negl(\lambda)$ is non-negligible if
  - $negl(\lambda) = \frac{1}{f(\lambda)}$, and
  - $f(\lambda)$ is a polynomial in λ .
- These two concepts are also not about value but speed.

Note: non-negligible is sometimes called noticeable.

Outline

1 Preliminaries
- Computational and Decisional
- Deterministic and Probabilistic
- Polynomial and Exponential
- Negl. and Non-Negl.
- Probability and Advantage

2 Hard Problem
- Hardness Definition
- Examples
- Hardness Analysis

3 Secure Scheme
- Security Parameter
- Security Level
- Definition of Security

Probability (1)

- Probability is to measure the likelihood that an event will occur.
- The event is returning an output
  - that is a problem solution to a given problem instance (in a computing problem), or
  - that is a successful attack on a scheme (in breaking a scheme).
- We have to analyze probability due to attack algorithm or solution algorithm by the adversary is a probabilistic algorithm.

Note: We stress again that we have to define with probabilistic algorithms because they are more efficient than deterministic algorithms.

Probability (2)

The probabilities have different ranges in solving a hard problem or in breaking a secure scheme. For example,

- **Digital Signatures.** Let $\Pr[\text{Win}_{Sig}]$ be the probability that the adversary successfully forges a valid signature.

$$0 \leq \Pr[\text{Win}_{Sig}] \leq 1.$$

- **Encryption.** Let $\Pr[\text{Win}_{Enc}]$ be the probability of correctly guessing message in the indistinguishability security model for encryption.

$$\frac{1}{2} \leq \Pr[\text{Win}_{Enc}] \leq 1.$$

Probability (3)

Let $\Pr[\mathrm{Win}_D]$ be the probability of correctly guessing the target Z in DDH problem instance (g, g^a, g^b, Z).

- Suppose Z is randomly chosen from a space having two elements where one element is true and the other is false. Then, we have

$$\frac{1}{2} \leq \Pr[\mathrm{Win}_D] \leq 1.$$

- Suppose Z is randomly chosen from $\mathbb{G}$ having p elements where only one element is true. Then, we have

$$1 - \frac{1}{p} \leq \Pr[\mathrm{Win}_D] \leq 1.$$

  Since only one of elements from $\mathbb{G}$ is equal to g^{ab} and Z is randomly chosen, we have that Z is false with probability $\frac{p-1}{p} = 1 - \frac{1}{p}$.

Advantage (1)

- The concept “advantage” was motivated by the inconsistent probabilities in different events.
- The probability cannot be set as a value to distinguish successful attack from failed attack.
- Advantage is an adjusted probability to replaced probability.
- The minimal advantage in definition must be zero.

Advantage (2)

Let P_{ideal} be the maximal probability of breaking a secure scheme or solving a hard problem. P_{ideal} can also be understood as a natural probability in some trivial attack algorithms or solution algorithms.

- If P_{ideal} is non-negligible, we define the advantage as

$$\text{Advantage} = \text{Probability of Successful Attack} - P_{ideal}.$$

- If P_{ideal} is negligible, we define the advantage as

$$\text{Advantage} = \text{Probability of Successful Attack}.$$

Advantage (3)

When defining an advantage,

- The maximal advantage could be different and associated with definitions. For example, we can define

$$\text{Advantage} = \text{Probability of Successful Attack} - P_{ideal}.$$

$$\text{Advantage} = 2(\text{Probability of Successful Attack} - P_{ideal}).$$

The minimal advantage is the same, but the maximal one is different.

- However, we don't care the maximal advantage. We only care whether the advantage is negligible or non-negligible.

Note: We recommend the advantage definition where its maximum is 1.

Advantage (4)

Advantage is well defined when the following condition holds.

Roughly speaking:

Every adversary in t polynomial time has negligible advantage ϵ in breaking scheme or solving problem.

$\Updownarrow$

The scheme is secure or the problem is hard.

Note: The value ϵ here is not probability but advantage. The above saying means that no matter how you propose an attack algorithm, its advantage should be negligible.

Outline

1 Preliminaries
- Computational and Decisional
- Deterministic and Probabilistic
- Polynomial and Exponential
- Negl. and Non-Negl.
- Probability and Advantage

2 Hard Problem
- Hardness Definition
- Examples
- Hardness Analysis

3 Secure Scheme
- Security Parameter
- Security Level
- Definition of Security

Outline

1 Preliminaries
- Computational and Decisional
- Deterministic and Probabilistic
- Polynomial and Exponential
- Negl. and Non-Negl.
- Probability and Advantage

2 Hard Problem
- Hardness Definition
- Examples
- Hardness Analysis

3 Secure Scheme
- Security Parameter
- Security Level
- Definition of Security

Computational Hardness

Let x be a random instance of a computational problem.

We say that the computational problem is hard if for any probabilistic polynomial time adversary (PPT) $\mathcal{A}$,

$$\Pr[\mathcal{A}(x) = y] = \epsilon(\lambda).$$

- The adversary runs a probabilistic algorithm.
- The adversary stops in polynomial time.
- The value $\epsilon(\lambda)$ is a negligible function in λ.
- The problem instance x has length λ.

Decisional Hardness (1)

Let x be a random instance of a decisional problem.

We say that the decisional problem is hard if for any probabilistic polynomial time adversary (PPT) $\mathcal{A}$,

$$\left| \Pr\left[\mathcal{A}(x) = True | x = True\right] - \Pr\left[\mathcal{A}(x) = True | x = False\right] \right| = \epsilon(\lambda).$$

- The adversary runs a probabilistic algorithm.
- The adversary stops in polynomial time.
- The value $\epsilon(\lambda)$ is a negligible function in λ.
- The problem instance x has length λ.

Decisional Hardness (2)

$$\left| \Pr\left[\mathcal{A}(x) = True | x = True\right] - \Pr\left[\mathcal{A}(x) = True | x = False\right] \right| = \epsilon(\lambda).$$

Let $x = (g, g^a, g^b, Z)$.

- $\Pr\left[\mathcal{A}(x) = True | x = True\right]$ is the probability of correctly guessing Z on the condition that $Z = g^{ab}$.
- $\Pr\left[\mathcal{A}(x) = True | x = False\right]$ is the probability of wrongly guessing Z on the condition that $Z \neq g^{ab}$.

The above advantage definition is suitable no matter how Z is chosen.

Remark (1)

The concepts of hard problem and hardness assumption are equivalent. However, the descriptions of these two concepts are slightly different.

- We can say that breaking a proposed scheme implies solving an underlying hard problem. Therefore, the scheme is secure under the hardness assumption.
- We can also say that a hardness assumption is a weak assumption or a strong assumption. Weak or strong is not related to the problem but to the strength of the hardness.

Remark (2)

All hardness assumptions can be classified into *weak assumptions* and *strong assumptions*, but the classification is not very precise.

- Weak assumptions =a computing problem is hard if and only if a few conditions hold.
- Strong assumptions =a computing problem is hard if and only if lots of conditions hold.

Breaking a strong assumption is easier than breaking a weak assumption.

Remark (3)

"Weak" and "Strong" have completely different meanings in applications.

	Good to us	Bad to us
Hardness Assumption	Weak Assumption	Strong Assumption
Computational Model	Strong Model	Weak Model

That is, a scheme secure under a weak hardness assumption in a strong security model is better (called good to us).

Outline

1 Preliminaries
- Computational and Decisional
- Deterministic and Probabilistic
- Polynomial and Exponential
- Negl. and Non-Negl.
- Probability and Advantage

2 Hard Problem
- Hardness Definition
- Examples
- Hardness Analysis

3 Secure Scheme
- Security Parameter
- Security Level
- Definition of Security

Computationally Hard Problems (1)

Let $\mathbb{G}$ be the group from $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ unless specified otherwise.

Discrete Logarithm Problem (DL)

Instance: $g, g^a \in \mathbb{G}$, where $\mathbb{G}$ is a general cyclic group
Compute: a

Computational Diffie-Hellman Problem (CDH)

Instance: $g, g^a, g^b \in \mathbb{G}$, where $\mathbb{G}$ is a general cyclic group
Compute: g^{ab}

q-Strong Diffie-Hellman Problem (q-SDH)

Instance: $g, g^a, g^{a^2}, \cdots, g^{a^q} \in \mathbb{G}$
Compute: $\left(s, g^{\frac{1}{a+s}}\right) \in \mathbb{Z}_p \times \mathbb{G}$ for any s

Computationally Hard Problems (2)

Let $\mathbb{G}$ be the group from $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ unless specified otherwise.

q-Strong Diffie-Hellman Inversion Problem (q-SDHI)

Instance: $g, g^a, g^{a^2}, \cdots, g^{a^q} \in \mathbb{G}$

Compute: $g^{\frac{1}{a}}$

q-Bilinear Diffie-Hellman Inversion Problem (q-BDHI)

Instance: $g, g^a, g^{a^2}, \cdots, g^{a^q} \in \mathbb{G}$

Compute: $e(g,g)^{\frac{1}{a}}$

q-Bilinear Diffie-Hellman Problem (q-SDH)

Instance: $g, g^a, g^{a^2}, \cdots, g^{a^q}, g^{a^{q+2}}, g^{a^{q+3}}, \cdots, g^{a^{2q}}, h \in \mathbb{G}$

Compute: $e(g,h)^{a^{q+1}}$

Decisionally Hard Problems (1)

Let $\mathbb{G}$ be the group from $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ unless specified otherwise.

Decisional Diffie-Hellman Problem (DDH)

Instance: $g, g^a, g^b, Z \in \mathbb{G}$, where $\mathbb{G}$ is a general cyclic group

Decide: $Z \stackrel{?}{=} g^{ab}$

Variant Decisional Diffie-Hellman Problem (Variant DDH)

Instance: $g, g^a, g^b, g^{ac}, Z \in \mathbb{G}$, where $\mathbb{G}$ is a general cyclic group

Decide: $Z \stackrel{?}{=} g^{bc}$

Decisional Bilinear Diffie-Hellman Problem (DBDH)

Instance: $g, g^a, g^b, g^c \in \mathbb{G}, Z \in \mathbb{G}_T$

Decide: $Z \stackrel{?}{=} e(g, g)^{abc}$

Decisionally Hard Problems (2)

Let $\mathbb{G}$ be the group from $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ unless specified otherwise.

Decisional Linear Problem

Instance: $g, g^a, g^b, g^{ac_1}, g^{bc_2}, Z \in \mathbb{G}$

Decide: $Z \stackrel{?}{=} g^{c_1+c_2}$

q-DABDHE Problem

Instance: $g, g^a, g^{a^2}, \cdots, g^{a^q}, h, h^{a^{q+2}} \in \mathbb{G}, Z \in \mathbb{G}_T$

Decide: $Z \stackrel{?}{=} e(g,h)^{a^{q+1}}$

Decisional (f, g, F)-GDDHE Problem

Instance: $g, g^a, g^{a^2}, \cdots, g^{a^{n-1}}, \ g^{af(a)}, \ g^{b \cdot af(a)} \in \mathbb{G}$

$h, h^a, h^{a^2}, \cdots, h^{a^{2k}}, \qquad h^{b \cdot g(a)} \in \mathbb{G}$

$Z \in \mathbb{G}_T$

$f(x), g(x)$ are co-prime polynomials of degree n, k.

Decide: $Z \stackrel{?}{=} e(g,h)^{b \cdot f(a)}$

Outline

1 Preliminaries
- Computational and Decisional
- Deterministic and Probabilistic
- Polynomial and Exponential
- Negl. and Non-Negl.
- Probability and Advantage

2 Hard Problem
- Hardness Definition
- Examples
- Hardness Analysis

3 Secure Scheme
- Security Parameter
- Security Level
- Definition of Security

How to Prove "Hardness" (1)

- Public key cryptography is based on the complexity result $\mathcal{NP} \neq \mathcal{P}$.
- We have not yet proved $\mathcal{NP} \neq \mathcal{P}$.
- A proved-to-be hard problem implies $\mathcal{NP} \neq \mathcal{P}$.
- All hard problems are believed-to-be hard only.
- However, we can still somewhat prove hardness.

Note: Some problems analyzed to be hard could be actually easy.

(Because the analysis is wrong!)

How to Prove "Hardness" (2)

Suppose we are going to prove the hardness of computing problem A.

Two common ways in proving hardness.

- Reduction. We prove that solving problem A implies solving an existing problem B that is well believed to be hard. (Will be introduced more in next lecture).
- Weak Computational Model. We prove that solving problem A in a weak computational model is hard. One of well-known models is the generic group model for proving hard problems defined over a cyclic group.

Note: The standard computational model is the Turing machine.

How to Prove "Hardness" (3)

Suppose we are going to prove the hardness of computing problem A.

One special way in proving hardness: membership proof.

- A family of computing problems are "proved" to be hard.
- We prove that problem A belongs to this family.

A Family of decisionally hard problem :

Decisional (P, Q, f)-GDHE Problem

Instance: $g^{P(x_1,x_2,\cdots,x_m)} \in \mathbb{G}$, $e(g,g)^{Q(x_1,x_2,\cdots,x_m)}, Z \in \mathbb{G}_T$

$P = (p_1, p_2, \cdots, p_s) \in \mathbb{Z}_p[X_1, \cdots, X_m]^s$ is an s-tuple of m-variate polynomials

$Q = (q_1, q_2, \cdots, q_s) \in \mathbb{Z}_p[X_1, \cdots, X_m]^s$ is an s-tuple of m-variate polynomials

$f \in \mathbb{Z}_p[X_1, X_2, \cdots, X_m]$

$f \neq \sum a_{i,j} p_i p_j + \sum b_i q_i$ holds for $\forall a_{i,j}, b_i$

Decide: $Z \stackrel{?}{=} e(g,g)^{f(x_1,x_2,\cdots,x_m)}$

Outline

1 Preliminaries
- Computational and Decisional
- Deterministic and Probabilistic
- Polynomial and Exponential
- Negl. and Non-Negl.
- Probability and Advantage

2 Hard Problem
- Hardness Definition
- Examples
- Hardness Analysis

3 Secure Scheme
- Security Parameter
- Security Level
- Definition of Security

Outline

1 Preliminaries
- Computational and Decisional
- Deterministic and Probabilistic
- Polynomial and Exponential
- Negl. and Non-Negl.
- Probability and Advantage

2 Hard Problem
- Hardness Definition
- Examples
- Hardness Analysis

3 Secure Scheme
- Security Parameter
- Security Level
- Definition of Security

Security Parameter

- Taking as input a security parameter λ, the key generation algorithms outputs a public key and a secret key.
- The security parameter is the bit length of input string.
- Security parameter can be roughly seen as instance length.
- In group-based cryptography, the security parameter λ refers in particular to the bit length of a group element, such as 160 bits or 1,024 bits.
- The size n in $H : \{0,1\}^* \rightarrow \{0,1\}^n$ is a security parameter.
- Breaking a scheme is becoming hard when the length of λ grows.

Note: Security parameter $\neq$ Public key size.

Outline

1 Preliminaries
- Computational and Decisional
- Deterministic and Probabilistic
- Polynomial and Exponential
- Negl. and Non-Negl.
- Probability and Advantage

2 Hard Problem
- Hardness Definition
- Examples
- Hardness Analysis

3 Secure Scheme
- Security Parameter
- Security Level
- Definition of Security

Security Level

- The security level indicates the strength of an adversary in breaking a scheme or solving a problem, which can be seen as the time cost of breaking a scheme or solving a problem.
- When λ is security parameter, we have $f(\lambda)$ is called security level.
- We say that a scheme or a problem has k-bit security if an adversary must take 2^k steps/operations to break the scheme or to solve the problem (with success probability at least 2/3).

Note: A scheme generated with security parameter λ cannot achieve λ-bit security but at most half.

Security Level (from the view of attack algorithm)

Suppose all potential attack algorithms to break a proposed scheme have been found with the following distinct time cost and advantage

$$(t_1, \epsilon_1), \quad (t_2, \epsilon_2), \quad \cdots, \quad (t_l, \epsilon_l).$$

For a simple analysis, we say that the proposed scheme has k-bit security if

$$2^k = \min\left\{\frac{t_1}{\epsilon_1}, \quad \frac{t_2}{\epsilon_2}, \quad \cdots, \quad \frac{t_l}{\epsilon_l}\right\}.$$

The security level of a proposed scheme is not fixed (from the view of attack algorithms). When a better attack algorithm is proposed, the security level decreases.

Security Level (Upper Bound)

Suppose a scheme is constructed over a cyclic group.

- Solving the Discrete Log problem can definitely break the scheme.
- The security level of scheme is not higher than DL problem.
- The discrete log hardness is the upper bound security level.

Summary: If solving problem B can immediately help breaking the scheme, the hardness of problem B is the upper bound security of scheme.

Security Level (Lower Bound)

Suppose there is a scheme and a security reduction.

- Suppose breaking the scheme implies solving a hard problem A.
- The security level of scheme is "higher" than the problem A.
- The hardness of problem A is the lower bound security of scheme.

Note: "higher" has some conditions that will be explained later when introducing concrete security.

Security Level

Suppose there is a scheme.

- The known security level is based on all known attack algorithms.
- The exact security level is based on the best attack algorithm (it is hard to find the best one. Otherwise, proving $\mathcal{NP} \neq \mathcal{P}$ is easy). Therefore, the exact security is usually unknown.
- The upper bound security level is based on a computing problem harder than breaking the scheme, such as the DL problem.
- The lower bound security level is based on the underlying hardness assumption adopted in security reduction.

Note: Which security level do you think we must know?

Outline

1 Preliminaries
- Computational and Decisional
- Deterministic and Probabilistic
- Polynomial and Exponential
- Negl. and Non-Negl.
- Probability and Advantage

2 Hard Problem
- Hardness Definition
- Examples
- Hardness Analysis

3 Secure Scheme
- Security Parameter
- Security Level
- Definition of Security

Security Definition

- Security model defines what the adversary knows.
- The hardness of breaking a scheme in different security models is not the same. For example,

  Easy: forge a signature with a secret key.
  Hard: Forge a signature without knowing public key.
- The security level of a scheme is associate with security model.
- Security definition must clearly define its security model.

Security Definition

Definition (Security)

We say that a scheme generated with security parameter λ is secure in a security model if every PPT adversary has advantage $\epsilon(\lambda)$ only, where $\epsilon(\lambda)$ is a negligible function in λ.

Note:

- The advantage must be well defined in the security model.
- Security definition doesn't use the concept of security level.
- Security level is used to understand the concrete security of scheme when λ is set with a concrete number.

Hardness Definition

Definition (Hardness)

We say that a computing problem is hard if every PPT adversary has advantage $\epsilon(\lambda)$ only, where

- $\epsilon(\lambda)$ is a negligible function in λ.
- The instance length is λ.

Note:

- The symbol λ refers to instance length not the security parameter.
- Computing problem doesn't need a security model.


END


Introduction
to Security
Reduction