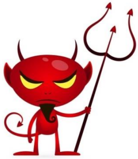
Introduction to Security Reduction

Lecture 2: Preliminaries

(Field, Group, Pairing, and Hash Function)



Adversary

My IQ is up to 186.

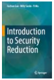
My interest is breaking schemes.

You want me to help you solve problem?

Fool me first!

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 Finite Field

2 Cyclic Groups

- Definition and Description
- Easy Problem and Hard Problem
- Two Group Choices
- Computations Over Group

3 Bilinear Pairings

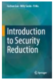
- Symmetric and Asymmetric
- Computations Over Pairing

4 Hash Functions

- Security-Based Classification
- Application-Based Classification

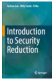
5 *(Pseudo)Random Number Generator

6 *Insecure Schemes



Outline

1. **Finite Field**
2. **Cyclic Groups**
   - Definition and Description
   - Easy Problem and Hard Problem
   - Two Group Choices
   - Computations Over Group
3. **Bilinear Pairings**
   - Symmetric and Asymmetric
   - Computations Over Pairing
4. **Hash Functions**
   - Security-Based Classification
   - Application-Based Classification
5. ***(Pseudo)Random Number Generator**
6. ***Insecure Schemes**



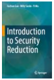
Definition of Finite Field

Definition (Finite Field)

A finite field (Galois field), denoted by $(\mathbb{F}, +, *)$, is a set containing a finite number of elements with two binary operations "+" (addition) and "$*$" (multiplication) defined as follows.

- $\forall u, v \in \mathbb{F}$, we have $u + v \in \mathbb{F}$ and $u * v \in \mathbb{F}$.
- $\forall u_1, u_2, u_3 \in \mathbb{F}$, $(u_1 + u_2) + u_3 = u_1 + (u_2 + u_3)$ and
  $$(u_1 * u_2) * u_3 = u_1 * (u_2 * u_3).$$
- $\forall u, v \in \mathbb{F}$, we have $u + v = v + u,\ u * v = v * u$
- $\exists 0_{\mathbb{F}}, 1_{\mathbb{F}} \in \mathbb{F}$ (identity elements), $\forall u \in \mathbb{F}$, we have
  $u + 0_{\mathbb{F}} = u$ and $u * 1_{\mathbb{F}} = u$.
- $\forall u \in \mathbb{F}$, $\exists -u \in \mathbb{F}$ such that $u + (-u) = 0_{\mathbb{F}}$.
- $\forall u \in \mathbb{F}^*$, $\exists u^{-1} \in \mathbb{F}^*$ such that $u * u^{-1} = 1_{\mathbb{F}}$. Here, $\mathbb{F}^* = \mathbb{F} \backslash \{0_{\mathbb{F}}\}$.
- $\forall u_1, u_2, v \in \mathbb{F}$, we have $(u_1 + u_2) * v = u_1 * v + u_2 * v$.

Note: The binary operations $\neq$ "+, $\times$" in elementary arithmetic.



Field Operations

The two binary operations "addition and multiplication" can be extended to subtraction and division through their inverses described as follows.

- $\forall u, v \in \mathbb{F}$, we have

  $$u - v = u + (-v),$$

  which is the addition of u and the additive inverse of v.

- $\forall u \in \mathbb{F}, v \in \mathbb{F}^*$, we have

  $$u/v = u * v^{-1},$$

  which is the multiplication of u and the multiplicative inverse of v.

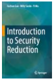
Definition Explanations

Let $(\mathbb{F}_{q^n}, +, *)$ be a finite field.

- n is a positive integer, and q is a prime number called characteristic.
- This finite field has q^n elements.

$$\underbrace{q \times q \times \cdots \times q \times q}_{n}$$

- Each element in the finite field can be seen as an n-length vector, where each scalar in the vector is from the finite field $\mathbb{F}_q$.
- The bit length of each element in this finite field is $n \cdot |q|$.

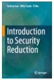


Special Finite Field: Prime Field $\mathbb{F}_q$

$(\mathbb{F}_q, +, *)$

- There are q elements in this field $\mathbb{Z}_q = \{0, 1, 2, \cdots, q-1\}$.
- $u + v = u+v \mod q$.
- $u * v = u*v \mod q$.
- $-u = q-u$.
- $u^{-1} = u^{q-2} \mod q$.

Note: Prime field is important due to the use of a group of prime order.



Outline

1 Finite Field

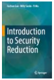
2 **Cyclic Groups**
- Definition and Description
- Easy Problem and Hard Problem
- Two Group Choices
- Computations Over Group

3 Bilinear Pairings
- Symmetric and Asymmetric
- Computations Over Pairing

4 Hash Functions
- Security-Based Classification
- Application-Based Classification

5 *(Pseudo)Random Number Generator

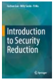
6 *Insecure Schemes



Names of Group

There are three types of groups from basic to advanced.

1.Abelian Group
↓
2.Abelian Group with Cyclic
↓
3.Abelian Group with Cyclic of Prime Order



Outline

1 Finite Field

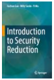
2 Cyclic Groups
- Definition and Description
- Easy Problem and Hard Problem
- Two Group Choices
- Computations Over Group

3 Bilinear Pairings
- Symmetric and Asymmetric
- Computations Over Pairing

4 Hash Functions
- Security-Based Classification
- Application-Based Classification

5 *(Pseudo)Random Number Generator

6 *Insecure Schemes

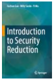


Definition of Group (1)

Definition (Abelian Group)

An abelian group, denoted by $(\mathbb{H}, \cdot)$, is a set of elements with one binary operation "$\cdot$" defined as follows.

- $\forall u, v \in \mathbb{H}$, we have $u \cdot v \in \mathbb{H}$.
- $\forall u_1, u_2, u_3 \in \mathbb{H}$, we have $(u_1 \cdot u_2) \cdot u_3 = u_1 \cdot (u_2 \cdot u_3)$.
- $\forall u, v \in \mathbb{H}$, we have $u \cdot v = v \cdot u$.
- $\exists 1_{\mathbb{H}} \in \mathbb{H}$, $\forall u \in \mathbb{H}$, we have $u \cdot 1_{\mathbb{H}} = u$.
- $\forall u \in \mathbb{H}$, $\exists u^{-1} \in \mathbb{H}$, such that $u \cdot u^{-1} = 1_{\mathbb{H}}$.



Definition of Group (2)

Definition (Abelian Group with Cyclic)

An abelian group $\mathbb{H}$ is a cyclic group if there exists (at least) one generator, denoted by h, which can generate the group $\mathbb{H}$:

$$\mathbb{H} = \left\{ h^1, h^2, \cdots, h^{|\mathbb{H}|} \right\} = \left\{ h^0, h^1, h^2, \cdots, h^{|\mathbb{H}|-1} \right\},$$

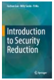
where $|\mathbb{H}|$ denotes the group order of $\mathbb{H}$ and $h^{|\mathbb{H}|} = h^0 = 1_{\mathbb{H}}$.

Definition (Abelian Group with Cyclic of Prime Order)

A group $\mathbb{G}$ is a cyclic subgroup of prime order if it is a subgroup of a cyclic group $\mathbb{H}$ and $|\mathbb{G}|$ is a prime number, where

- $|\mathbb{G}|$ is a divisor of $|\mathbb{H}|$;
- There exists a generator $g \in \mathbb{H}$, which generates $\mathbb{G}$.

Abelian Group with Cyclic of Prime Order is short as Cyclic Group.



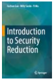
Why Cyclic and Prime Order?

$$\mathbb{G} = \{g^0, g^1, \cdots, g^{p-1}\} \text{ for a prime } p.$$

- The group $\mathbb{G}$ is the smallest subgroup without confinement attacks.
- Any group element except g^0 is a generator of $\mathbb{G}$.
- Any integer in $\{1, 2, \cdots, p-1\}$ has a modular multiplicative inverse. For any $x \in \mathbb{Z}_p^* = \{1, 2, \cdots, p-1\}$, we can definitely compute

$$g^{\frac{1}{x}}.$$

Note: We don't have to use a group (in prime order) in cryptography.



Cyclic Group in Description

To define a group for scheme constructions, we need to specify

- The space of the group, denoted by $\mathbb{G}$.
- The generator of the group, denoted by g.
- The order of the group, denoted by p.

$(\mathbb{G}, g, p)$ are the basic components when describing a group.

Note: We could need more information when describing a group.

Size of Group Element

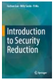
What is the representation size of each group element?

$$\mathbb{G} = \{g^0, g^1, \cdots, g^{p-1}\} \text{ for a prime } p.$$

- p group elements and each group element has the same size.
- Each group element can be encoded into a bit string.
- Each group element must be represented with a different bit string.
- To represent $p = 2^{160}$ elements, we need at least 160-bit strings.
- It could be hard to achieve optimal size.

We therefore have the representation of group element

$$|g| \geq 160$$



Outline

1 Finite Field

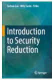
2 Cyclic Groups
- Definition and Description
- Easy Problem and Hard Problem
- Two Group Choices
- Computations Over Group

3 Bilinear Pairings
- Symmetric and Asymmetric
- Computations Over Pairing

4 Hash Functions
- Security-Based Classification
- Application-Based Classification

5 *(Pseudo)Random Number Generator

6 *Insecure Schemes

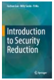


Computing Problems Over Group

When we use a cyclic group to build cryptography,

- Some computing problems must be easy. Otherwise, cryptography is not usable. A group only defines the group operation "$\cdot$", but it can be extended to group exponentiation.
- Some computing problems must be hard. Otherwise, cryptography is not secure. The most fundamental hard problem over a group is the discrete logarithm problem.

Note: DL problem is hard in some well-constructed groups only.



Easy: Group Exponentiation (1)

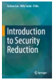
Let $(\mathbb{G}, g, p)$ be a cyclic group and x be a positive integer from $\mathbb{Z}_p$. We denote by g^x the group exponentiation.

- The group exponentiation g^x is defined as

$$g^x = \underbrace{g \cdot g \cdots g \cdot g}_{x}.$$

- The group exponentiation is composed of $x-1$ copies of the group operations from the above definition. It is impractical to conduct $x-1$ copies of computations when x is as large as 2^{160}.

- There exist algorithms that can compute the group exponentiation very fast. For example, the square-and-multiply algorithm.



Easy: Group Exponentiation (2)

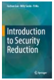
Given $g \in \mathbb{G}$ and $x \in \mathbb{Z}_p$, we can compute g^x as follows.

- Convert x into an n-bit string x:

$$x = x_{n-1} \cdots x_1 x_0 = \sum_{i=0}^{n-1} x_i 2^i.$$

- Let $g_i = g^{2^i}$. Compute $g_i = g_{i-1} \cdot g_{i-1}$ for all $i \in [1, n-1]$.
- Compute g^x by

$$g^x = \prod_{i=0}^{n-1} g_i^{x_i} = g^{\sum_{i=0}^{n-1} x_i 2^i}.$$



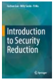
Hard: Discrete Logarithm

Suppose we are given $g, h \in \mathbb{G} \setminus 1_{\mathbb{G}}$

- The integer x satisfying $g^x = h$ is called the discrete logarithm.
- Computing x is known as the discrete logarithm (DL) problem.

If $\mathbb{G}$ is a group of prime order, then for any two group element $g, h \in \mathbb{G} \setminus 1_{\mathbb{G}}$, the discrete logarithm x must exist! (Another reason why we need a group of prime order)

Note: If DLP is easy, all schemes over such a group must be insecure.

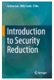


Hardness of DL (1)

Let $(\mathbb{G}, g, p)$ be any group. The most efficient algorithm for solving DLP requires $\Omega(\sqrt{p})$ steps (exponentiation). Roughly speaking, at least $\sqrt{p}$.

- $\Omega(\sqrt{p})$ steps means "lower bound" $\sqrt{p}$ (at least).
- $O(\sqrt{p})$ steps means "upper bound" $\sqrt{p}$ (at most).
- This algorithm can solve DL problem over any group.
- DLP over some specific groups could take less than $\sqrt{p}$ steps.
- DLP over some specific groups could be easy. $O(1)$ steps.

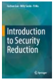
Note: The step number should be $c \cdot \sqrt{p}$ for some positive coefficient c in computational complexity.



Hardness of DL (2)

To implement (design) a scheme constructed over a group $(\mathbb{G}, g, p)$, where the adversary must take at least 2^{80} steps to break the scheme, we must consider generic attack and specific attack in solving the DL problem.

- The parameter must satisfy $p \geq 2^{160}$ to resist generic attacks.
- All other parameters for specific group constructions, such as the size of group element, must be large enough to resist specific attacks for solving DLP.



Outline

1 Finite Field

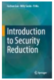
2 Cyclic Groups
- Definition and Description
- Easy Problem and Hard Problem
- Two Group Choices
- Computations Over Group

3 Bilinear Pairings
- Symmetric and Asymmetric
- Computations Over Pairing

4 Hash Functions
- Security-Based Classification
- Application-Based Classification

5 *(Pseudo)Random Number Generator

6 *Insecure Schemes



Cyclic Groups and Finite Fields

- A finite field $(\mathbb{F}_{q^n}, +, *)$ already implies two groups.

$$(\mathbb{F}_{q^n}, +), \quad (\mathbb{F}^*_{q^n}, *)$$

- We still need other advanced groups for various reasons.
- For example, with short representation of group element.

Group Choice 1: Multiplicative Group

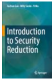
A multiplicative group is defined as $(\mathbb{G}, g, q, p)$.

- **Group Elements.** The group elements are integers from

$$\mathbb{Z}_q^* = \{1, 2, \cdots, q-1\}, \quad |g| = \log^q .$$

- **Group Generator.** g is from $\mathbb{Z}_q^*$ (some integers from this set are not the generators of $\mathbb{G}$).
- **Group Order.** p satisfying $p|(q-1)$.
- **Group Operation.** We have $u \cdot v = u \times v \bmod q$.

Note: The integer q significantly affects the hardness of DLP in this special construction and q must be at least 1024 bits. Otherwise, solving its DLP takes less than 2^{80} steps.

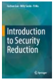


Group Choice 2: Elliptic Curve Group

An elliptic curve group is defined as $(\mathbb{G}, g, p)$.

- **Group Elements.** The group elements are points (represented with x-coordinate and y-coordinate) on the elliptic curve. When the curve is given, we can use the x-coordinate and one more bit only to represent a group element.
- **Group Generator.** g is also a point.
- **Group Order.** p a prime order.
- **Group Operation.** We have $u \cdot v$ defined by elliptic curves.

Note: The size of group element can be as short as the group order. That is, $|g| = |p| = 160$ where solving its DLP requires 2^{80} steps.



Outline

1 Finite Field

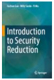
2 Cyclic Groups
- Definition and Description
- Easy Problem and Hard Problem
- Two Group Choices
- Computations Over Group

3 Bilinear Pairings
- Symmetric and Asymmetric
- Computations Over Pairing

4 Hash Functions
- Security-Based Classification
- Application-Based Classification

5 *(Pseudo)Random Number Generator

6 *Insecure Schemes

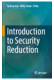


Computations Over Group (Prime Order)

- **Group Operation.** Given $g, h \in \mathbb{G}$, compute $g \cdot h$.
- **Group Inverse.** Given $g \in \mathbb{G}$, compute $\frac{1}{g} = g^{-1}$.
  Since $g^p = g \cdot g^{p-1} = 1$ (not the integer 1), we have $g^{-1} = g^{p-1}$.
- **Group Division.** Given $g, h \in \mathbb{G}$, compute $\frac{g}{h} = g \cdot h^{-1}$.
- **Group Exponentiation.** Given $g \in \mathbb{G}$ and $x \in \mathbb{Z}_p$, compute g^x.

Question: Given $g \in \mathbb{G}, (x, y, z) \in \mathbb{Z}_p$, do you know how to compute

$$g^{-\frac{y-z}{x+z}}?$$



Outline

1 Finite Field

2 Cyclic Groups
- Definition and Description
- Easy Problem and Hard Problem
- Two Group Choices
- Computations Over Group

3 **Bilinear Pairings**
- Symmetric and Asymmetric
- Computations Over Pairing

4 Hash Functions
- Security-Based Classification
- Application-Based Classification

5 *(Pseudo)Random Number Generator

6 *Insecure Schemes

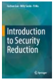
Pairing Overview

- Bilinear pairing maps two group elements in elliptic curve groups to a third group element in a multiplicative group without losing its isomorphic property.
- Bilinear pairing was originally introduced to solve hard problems in elliptic curve groups by mapping its problem instance into a problem instance in a multiplicative group.

Bilinear pairing ($\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$) falls into the following three types.

- Symmetric. $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$. Denoted by $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$.
- Asymmetric 1. $\mathbb{G}_1 \neq \mathbb{G}_2$ with homomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$.
- Asymmetric 2. $\mathbb{G}_1 \neq \mathbb{G}_2$ with no efficient homomorphism.

Note: Homomorphism might be needed in scheme construction.



Outline

1 Finite Field

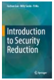
2 Cyclic Groups
- Definition and Description
- Easy Problem and Hard Problem
- Two Group Choices
- Computations Over Group

3 **Bilinear Pairings**
- **Symmetric and Asymmetric**
- Computations Over Pairing

4 Hash Functions
- Security-Based Classification
- Application-Based Classification

5 *(Pseudo)Random Number Generator

6 *Insecure Schemes



Symmetric Pairing (Definition)

Let $\mathbb{PG} = (\mathbb{G}, \mathbb{G}_T, g, p, e)$ be a symmetric-pairing group. Here, $\mathbb{G}$ is an elliptic curve group, $\mathbb{G}_T$ is a multiplicative subgroup, $|\mathbb{G}| = |\mathbb{G}_T| = p$, g is a generator of $\mathbb{G}$, and e is a map satisfying the following three properties.

- For all $u, v \in \mathbb{G}, a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$.
- $e(g, g)$ is a generator of group $\mathbb{G}_T$.
- For all $u, v \in \mathbb{G}$, there exist efficient algorithms to compute $e(u, v)$.

Symmetric Pairing (Size)

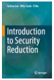
Two types of DLP:

- Compute x from g and g^x.
- Compute x from $e(g,g)$ and $e(g,g)^x$.

To make sure solving any DLP takes at least 2^{80} steps, it requires that

$$|g| \geq 512\text{(bits)}, \quad |e(g,g)| \geq 1024\text{(bits)}.$$

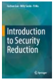
Note: 1024 is just a textbook size. We need a larger parameter now.



Asymmetric Pairing (Definition)

Let $\mathbb{PG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, p, e)$ be an asymmetric-pairing group. Here, $\mathbb{G}_1, \mathbb{G}_2$ are elliptic curve groups, $\mathbb{G}_T$ is a multiplicative subgroup, $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T| = p$, g_1 is a generator of $\mathbb{G}_1$, g_2 is a generator of $\mathbb{G}_2$, and e is a map satisfying the following three properties.

- For all $u \in \mathbb{G}_1, v \in \mathbb{G}_2, a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$.
- $e(g_1, g_2)$ is a generator of group $\mathbb{G}_T$.
- For all $u \in \mathbb{G}_1, v \in \mathbb{G}_2$, there exist efficient algo. to compute $e(u, v)$.



Asymmetric Pairing (Size)

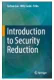
Three types of DLP:

- Compute x from g_1 and g_1^x.
- Compute x from g_2 and g_2^x.
- Compute x from $e(g_1, g_2)$ and $e(g_1, g_2)^x$.

To make sure solving any DLP takes at least 2^{80} steps, it requires that

$$|g_1| \geq 160\text{(bits)}, |g_2| \geq 1024\text{(bits)}, \quad |e(g,g)| \geq 1024\text{(bits)}.$$

Note: We have to set $|g_2| = |e(g_1, g_2)|$ due to asymmetric construction.



Outline

1 Finite Field

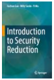
2 Cyclic Groups
- Definition and Description
- Easy Problem and Hard Problem
- Two Group Choices
- Computations Over Group

3 **Bilinear Pairings**
- Symmetric and Asymmetric
- **Computations Over Pairing**

4 Hash Functions
- Security-Based Classification
- Application-Based Classification

5 *(Pseudo)Random Number Generator

6 *Insecure Schemes



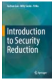
Basic Computations

A symmetric-pairing group is composed of groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order p and a bilinear map e. All computations over a pairing group are summarized as follows.

- All modular operations over $\mathbb{Z}_p$ (prime field).
- All group operations over the groups $(\mathbb{G}, \mathbb{G}_T)$.
- The pairing computation $e(u, v)$ for all $u, v \in \mathbb{G}$.

Question: Given $g, g^a \in \mathbb{G}, (x, y) \in \mathbb{Z}_p$, do you know how to compute

$$e(g, g)^{(a+x)(a+y)}?$$



Outline

1 Finite Field

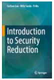
2 Cyclic Groups
- Definition and Description
- Easy Problem and Hard Problem
- Two Group Choices
- Computations Over Group

3 Bilinear Pairings
- Symmetric and Asymmetric
- Computations Over Pairing

4 **Hash Functions**
- **Security-Based Classification**
- **Application-Based Classification**

5 *(Pseudo)Random Number Generator

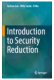
6 *Insecure Schemes



Hash Functions: $H(\cdot)$

- A hash function takes an arbitrary-length string as an input and returns a much shorter string as an output.
- In scheme construction, we cannot embed all values into $\mathbb{Z}_p$ or $\mathbb{G}$ due to limited space. We compute $g^{H(m)}$ instead of g^m when $m \notin \mathbb{Z}_p$.
- In security reduction, hash function might be set as random oracle.

Note: Most public-key cryptography schemes need a hash function.



Outline

1 Finite Field

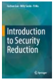
2 Cyclic Groups
- Definition and Description
- Easy Problem and Hard Problem
- Two Group Choices
- Computations Over Group

3 Bilinear Pairings
- Symmetric and Asymmetric
- Computations Over Pairing

4 **Hash Functions**
- **Security-Based Classification**
- Application-Based Classification

5 *(Pseudo)Random Number Generator

6 *Insecure Schemes

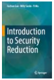


One-Way and Collision-Resistant

Hash functions can be classified into the following two main types according to the security definition.

- **One-Way Hash Function.** Given a one-way hash function H and an output string y, it is hard to find a pre-image input x satisfying $y = H(x)$.
- **Collision-Resistant Hash Function.** Given a collision-resistant hash function H, it is hard to find two different inputs x_1 and x_2 satisfying $H(x_1) = H(x_2)$.

We can simply call a hash function *cryptographic hash function* that is one-way hash function, or a collision-resistant hash function satisfying applications.



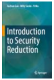
Outline

1 Finite Field
2 Cyclic Groups
- Definition and Description
- Easy Problem and Hard Problem
- Two Group Choices
- Computations Over Group

3 Bilinear Pairings
- Symmetric and Asymmetric
- Computations Over Pairing

4 Hash Functions
- Security-Based Classification
- **Application-Based Classification**

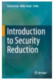
5 *(Pseudo)Random Number Generator
6 *Insecure Schemes



The Space of Hashing Outputs

Hash functions can be classified into the following three types according to the output space, where the input can be any arbitrary strings.

- $H : \{0,1\}^* \rightarrow \{0,1\}^n$. The output space is the set containing all n-bit strings. We mainly use this kind of hash function to generate a symmetric key from the key space $\{0,1\}^n$ for hybrid encryption.
- $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$. The output space is $\{0,1,2,\cdots,p-1\}$, where p is the group order. We use this kind of hash function to embed hashing values in group exponents such as $g^{H(m)}$.
- $H : \{0,1\}^* \rightarrow \mathbb{G}$. The output space is a cyclic group. That is, this hash function will hash the input string into a group element. This hash function exists for some groups only.



Outline

1 Finite Field

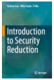
2 Cyclic Groups
- Definition and Description
- Easy Problem and Hard Problem
- Two Group Choices
- Computations Over Group

3 Bilinear Pairings
- Symmetric and Asymmetric
- Computations Over Pairing

4 Hash Functions
- Security-Based Classification
- Application-Based Classification

5 *(Pseudo)Random Number Generator

6 *Insecure Schemes



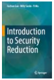
(Pseudo)Random Number Generator (1)

In many scheme constructions, algorithms need to choose random numbers from a space to perform computations, such as

- A random n-bit string from $\{0,1\}^n$.
- A random integer from $\mathbb{Z}_p$.
- A random element from such as $\mathbb{G}$.

Let x be the random variable and w_1, w_2 be any two possible random numbers from the space. The action "randomly choose" means that

$$\Pr[x = w_1] = \Pr[x = w_2].$$

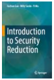


(Pseudo)Random Number Generator (2)

Something different here:

- In scheme algorithms, algorithms can choose real random numbers satisfying the equal probability.
- In real world, algorithms could choose peudorandom numbers on with a pseudorandom number generator.

Security reductions also assume that all chosen random numbers are truly random.



Outline

1 **Finite Field**

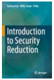
2 **Cyclic Groups**
- Definition and Description
- Easy Problem and Hard Problem
- Two Group Choices
- Computations Over Group

3 **Bilinear Pairings**
- Symmetric and Asymmetric
- Computations Over Pairing

4 **Hash Functions**
- Security-Based Classification
- Application-Based Classification

5 ***(Pseudo)Random Number Generator**

6 ***Insecure Schemes**

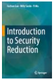


How to Forge Signatures

Backgroup: This lecture will give some insecure signature schemes.

- The adversary is given a public key and some signatures.
- The adversary is asked to forge a signature on a new message.

Questions: How to forge signature on a new message?

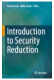


Insecure Scheme (1)

- The public key is $pk = (g, g^{\alpha})$ and the signing key is $sk = \alpha \in \mathbb{Z}_p$.
- Suppose the signature on m is defined as

$$\sigma_m = g^{\alpha \cdot m}.$$

Question: How to forge a signature when given (pk, m, σ_m)?



Insecure Scheme (2)

- The public key is $pk = (g, g^\alpha)$ and the signing key is $sk = \alpha \in \mathbb{Z}_p$.
- Suppose the signature on m is defined as

$$\sigma_m = g^{\alpha+m}.$$

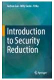
Question: How to forge a signature when given (pk, m, σ_m)?

Insecure Scheme (3)

- The public key is $pk = (g, g^{\alpha}, g^{\beta})$ and $sk = (\alpha, \beta) \in \mathbb{Z}_p$.
- Suppose the signature on m is defined as

$$\sigma_m = \alpha + m\beta \mod p.$$

Question: How to forge a signature when given $(pk, m_1, \sigma_{m_1}, m_2, \sigma_{m_2})$?



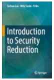
Insecure Scheme (4)

- The public key is $pk = (g, g^{\alpha}, g^{\beta})$ and $sk = (\alpha, \beta) \in \mathbb{Z}_p$.
- Suppose the signature on m is defined as

$$\sigma_m = \left(g^{\alpha\beta + mr}, \quad g^r \right),$$

where r is a random number chosen from $\mathbb{Z}_p$.

Question: How to forge a signature when given (pk, m, σ_m)?



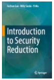
Insecure Scheme (5)

- The public key is $pk = (g, g^{\alpha}, g^{\beta})$ and $sk = (\alpha, \beta) \in \mathbb{Z}_p$.
- Suppose the signature on m is defined as

$$\sigma_m = \left(g^{\alpha\beta + mr\cdot\beta}, \;\; g^{r} \right),$$

where r is a random number chosen from $\mathbb{Z}_p$.

Question: How to forge a signature when given (pk, m, σ_m)?

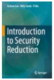


Insecure Scheme (6)

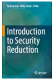
- The public key is $pk = (g, g^{\alpha})$ and the signing key is $sk = \alpha \in \mathbb{Z}_p$.
- Suppose the signature on m is defined as

$$\sigma_m = g^{\frac{1}{\alpha \cdot m}}.$$

Question: How to forge a signature when given (pk, m, σ_m)?




END


Introduction
to Security
Reduction