Course Business

- **Homework 3 Released**
  - Due: Tuesday, October 31st.

- I will be travelling early next week to attend a workshop on data-privacy

- Guest Lecture on 10/24 (Professor Spafford)

Cryptography
CS 555

Week 9:

- One Way Functions
- Number Theory

Readings: Katz and Lindell Chapter 7, B.1, B.2, 8.1-8.2

CS 555: Week 8: Topic 1: One Way Functions

What are the minimal assumptions necessary for symmetric key-cryptography?

One-Way Functions (OWFs)

$$f(x) = y$$

Definition: A function $f\colon \{0,1\}^* \rightarrow \{0,1\}^*$ is one way if it is

1. **(Easy to compute)** There is a polynomial time algorithm (in |x|) for computing f(x).
2. **(Hard to Invert)** Select $x \leftarrow \{0,1\}^n$ uniformly at random and give the attacker input 1^n, f(x). The probability that a PPT attacker outputs x' such that $f(x') = f(x)$ is negligible.

Hard Core Predicates

- Recall that a one-way function f may potentially reveal lots of information about input

- **Example**: $f(x_1,x_2)=(x_1,g(x_2))$, where g is a one-way function.
- **Claim**: f is one-way (even if $f(x_1,x_2)$ reveals half of the input bits!)

Hard Core Predicates

Definition: A predicate $\mathrm{hc}: \{0,1\}^* \to \{0,1\}$ is called a hard-core predicate of a function f if

1. (Easy to Compute) hc can be computed in polynomial time
2. (Hard to Guess) For all PPT attacker A there is a negligible function negl such that we have

$$\mathbf{Pr}_{x \leftarrow \{0,1\}^n}[A(1^n, f(x)) = \mathrm{hc}(x)] \leq \frac{1}{2} + negl(n)$$

Attempt 1: Hard-Core Predicate

Consider the predicate

$$\text{hc}(\text{x}) = \bigoplus_{i=1}^{n} x_i$$

Hope: hc is hard core predicate for any OWF.

Counter-example:

$$\text{f(x)} = (\text{g(x)}, \bigoplus_{i=1}^{n} x_i)$$

Trivial Hard-Core Predicate

Consider the function

$$f(x_1,\ldots,x_n) = x_1,\ldots,x_{n-1}$$

f has a trivial hard core predicate

$$\mathrm{hc}(\mathrm{x}) = x_n$$

Not useful for crypto applications (e.g., f is not a OWF)

Attempt 3: Hard-Core Predicate

Consider the predicate

$$\mathrm{hc}(\mathrm{x}, \mathrm{r}) = \oplus_{i=1}^{n} x_i r_i$$

(the bits r_1,…, r_n will be selected uniformly at random)

Goldreich-Levin Theorem: (Assume OWFs exist) For any OWF f, hc is a hard-core predicate of g(x,r)=(f(x),r).

Using Hard-Core Predicates

Theorem: Given a one-way-permutation f and a hard-core predicate hc we can construct a PRG G with expansion factor $\ell(n) = n + 1$.

Construction:

$$G(s) = f(s) \parallel \mathrm{hc}(s)$$

Intuition: f(s) is actually uniformly distributed

- s is random
- f(s) is a permutation
- Last bit is hard to predict given f(s) (since hc is hard-core for f)

Arbitrary Expansion

Theorem: Suppose that there is a PRG G with expansion factor $\ell(n) = n + 1$. Then for any polynomial p(.) there is a PRG with expansion factor p(n).

Construction:

- $\mathbf{G(x) = y||b.}$ **(n+1 bits)**
- $\mathbf{G^{i+1}(x) = G(z)||b}$ **where** $\mathbf{G^{i}(x) = z||b}$ **(n+i bits)**

Any Beyond

Theorem: Suppose that there is a PRG G with expansion factor $\ell(n) = n + 1$. Then for any polynomial p(.) there is a PRG with expansion factor p(n).

Theorem: Suppose that there is a PRG G with expansion factor $\ell(n) = 2n$. Then there is a secure PRF.

Theorem: Suppose that there is a secure PRF then there is a strong pseudorandom permutation.

Any Beyond

Corollary: If one-way functions exist then PRGs, PRFs and strong PRPs all exist.

Corollary: If one-way functions exist then there exist CCA-secure encryption schemes and secure MACs.

PRFs from PRGs

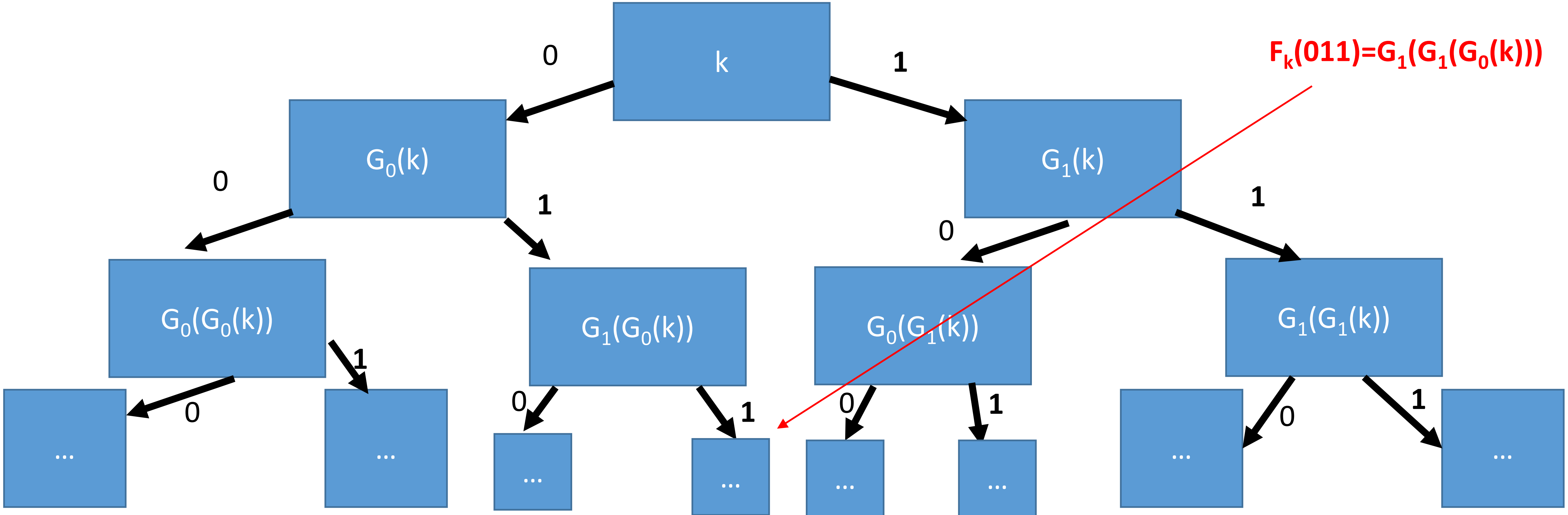
Theorem: Suppose that there is a PRG G with expansion factor $\ell(n) = 2n$. Then there is a secure PRF.

Let $G(x) = G_0(x)||G_1(x)$ (first/last n bits of output)

$$\boldsymbol{F_K(x_1, \dots, x_n) = G_{x_n}\left(\dots\left(G_{x_2}\left(G_{x_1}(K)\right)\right)\dots\right)}$$

PRFs from PRGs

Theorem: Suppose that there is a PRG G with expansion factor $\ell(n) = 2n$. Then there is a secure PRF.



PRFs from PRGs

Theorem: Suppose that there is a PRG G with expansion factor $\ell(n) = 2n$. Then there is a secure PRF.

Proof:

Related Claim: For any t(n) and any PPT attacker A we have

$$\left|\boldsymbol{Pr}\left[\boldsymbol{A}\left(\boldsymbol{r_1} \parallel \cdots \parallel \boldsymbol{r_{t(n)}}\right)\right] - \boldsymbol{Pr}\left[\boldsymbol{A}\left(\boldsymbol{G}(\boldsymbol{s_1}) \parallel \cdots \parallel \boldsymbol{G}\left(\boldsymbol{s_{t(n)}}\right)\right)\right]\right| < \boldsymbol{negl(n)}$$

(recall Homework 2!)

PRFs from PRGs

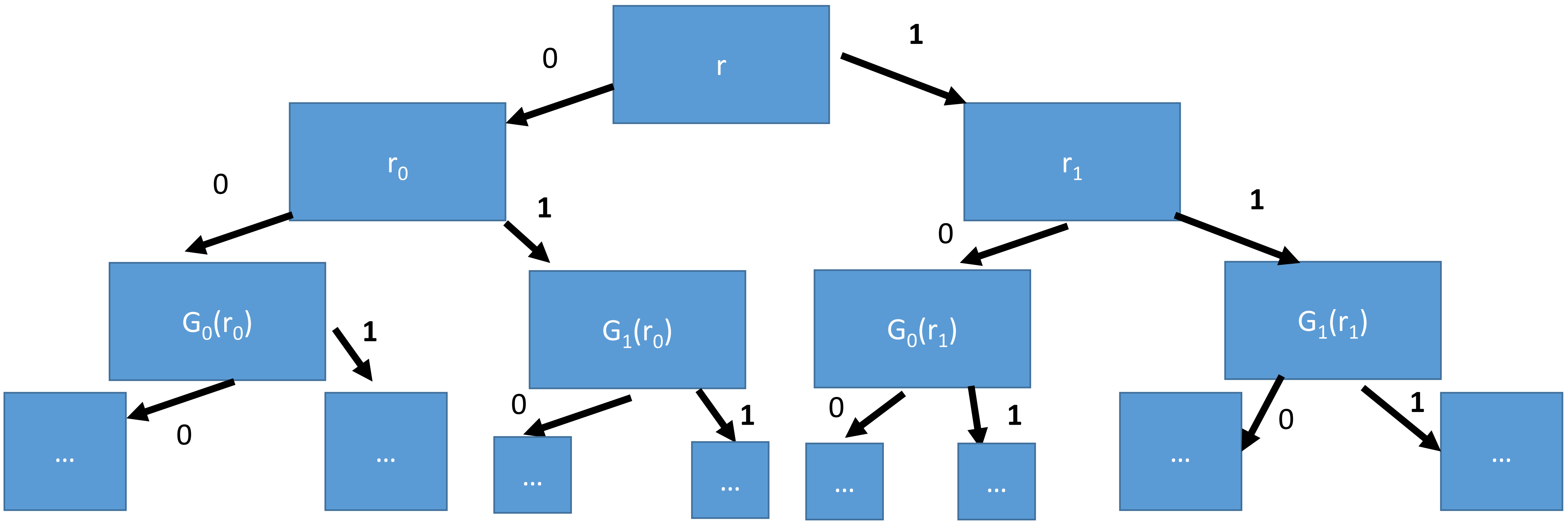
Claim 1: For any t(n) and any PPT attacker A we have

$$\left|Pr\left[A\left(r_1 \parallel \cdots \parallel r_{t(n)}\right)\right] - Pr\left[A\left(G(s_1) \parallel \cdots \parallel G\left(s_{t(n)}\right)\right)\right]\right| < negl(n)$$

Proof by Hybrids: Fix j

$$Adv_j = \left|Pr\left[A\left(r_1 \parallel \cdots \parallel r_{j+1} \parallel G\left(s_{j+2}\right) \ldots \parallel G\left(s_{t(n)}\right)\right)\right]\right.$$

Hybrid H_1



From OWFs (Recap)

Theorem: Suppose that there is a PRG G with expansion factor $\ell(n) = n + 1$. Then for any polynomial p(.) there is a PRG with expansion factor p(n).

Theorem: Suppose that there is a PRG G with expansion factor $\ell(n) = 2n$. Then there is a secure PRF.

Theorem: Suppose that there is a secure PRF then there is a strong pseudorandom permutation.

OWFs/OWPs are Sufficient for Symmetric Crypto

Corollary: If one-way permutations exist then PRGs, PRFs and strong PRPs all exist.

Corollary: If one-way permutations exist then there exist CCA-secure encryption schemes and secure MACs.

Remark: Can obtain all of the above results from OWFs as well

Are OWFs Necessary for Private Key Crypto?

- Previous results show that OWFs are sufficient.

- Can we build Private Key Crypto from weaker assumptions?

- **Short Answer:** No, OWFs are also *necessary* for most private-key crypto primitives

PRGs → OWFs

Proposition 7.28: If PRGs exist then so do OWFs.

Proof: Let G be a secure PRG with expansion factor $\ell(n) = 2n$.

Question: why can we assume that we have an PRG with expansion 2n?

Answer: We already showed that a PRG with expansion factor $\ell(n) = n + 1$. Implies the existence of a PRG with expansion p(n) for any polynomial.

PRGs → OWFs

Proposition 7.28: If PRGs exist then so do OWFs.

Proof: Let G be a secure PRG with expansion factor $\ell(n) = 2n$.

Claim: G is also a OWF!

(Easy to Compute?) ✓

(Hard to Invert?)

Intuition: If we can invert G(x) then we can distinguish G(x) from a random string.

PRGs → OWFs

Proposition 7.28: If PRGs exist then so do OWFs.

Proof: Let G be a secure PRG with expansion factor $\ell(n) = 2n$.

Claim 1: Any PPT A, given G(s), cannot find s except with negligible probability.

Reduction: Assume (for contradiction) that A can invert G(s) with non-negligible probability p(n).

Distinguisher D(y): Simulate A(y)

Output 1 if and only if A(y) outputs x s.t. G(x)=y.

PRGs → OWFs

Proposition 7.28: If PRGs exist then so do OWFs.

Proof: Let G be a secure PRG with expansion factor $\ell(n) = 2n$.

Claim 1: Any PPT A, given G(s), cannot find s except with negligible probability.

Intuition for Reduction: If we can find x s.t. G(x)=y then y is not random.

Fact: Select a random 2n bit string y. Then (whp) there does not exist x such that G(x)=y.

Why not?

PRGs → OWFs

Proposition 7.28: If PRGs exist then so do OWFs.

Proof: Let G be a secure PRG with expansion factor $\ell(n) = 2n$.

Claim 1: Any PPT A, given G(s), cannot find s except with negligible probability.

Intuition: If we can invert G(x) then we can distinguish G(x) from a random string.

Fact: Select a random 2n bit string y. Then (whp) there does not exist x such that G(x)=y.

- Why not? Simple counting argument, 2^{2n} possible y's and 2^n x's.
- Probability there exists such an x is at most 2^{-n} (for a random y)

What other assumptions imply OWFs?

- PRGs → OWFs
- (Easy Extension) PRFs → PRGs → OWFs

- Does secure crypto scheme imply OWFs?
  - CCA-secure? (Strongest)
  - CPA-Secure? (Weaker)
  - EAV-secure? (Weakest)
    - As long as the plaintext is longer than the secret key
  - Perfect Secrecy? X (Guarantee is information theoretic)

EAV-Secure Crypto → OWFs

Proposition 7.29: If there exists a EAV-secure private-key encryption scheme that encrypts messages twice as long as its key, then a one-way function exists.

Recap: EAV-secure.

- Attacker picks two plaintexts m_0,m_1 and is given c=$Enc_K(m_b)$ for random bit b.
- Attacker attempts to guess b.
- No ability to request additional encryptions (chosen-plaintext attacks)
- In fact, no ability to observe any additional encryptions

EAV-Secure Crypto → OWFs

Proposition 7.29: If there exists a EAV-secure private-key encryption scheme that encrypts messages twice as long as its key, then a one-way function exists.

Reduction: $\boldsymbol{f(m,k,r) = Enc_k(m;r) \| m}$.

Input: 4n bits

(For simplicity assume that $\mathbf{Enc}_k$ accepts n bits of randomness)

Claim: f is a OWF

EAV-Secure Crypto → OWFs

Proposition 7.29: If there exists a EAV-secure private-key encryption scheme that encrypts messages twice as long as its key, then a one-way function exists.

Reduction: $\boldsymbol{f(m, k, r) = Enc_k(m; r) \| m}$.

Claim: f is a OWF

Reduction Intuition: Inverting f involves finding secret key k consistent with known message-ciphertext pair.

MACs→ OWFs

In particular, given a MAC that satisfies MAC security (Definition 4.2) against an attacker who sees an arbitrary (polynomial) number of message/tag pairs.

Conclusions: OWFs are necessary and sufficient for all (non-trivial) private key cryptography.

→OWFs are a minimal assumption for private-key crypto.

Public Key Crypto/Hashing?

- OWFs are known to be necessary
- Not known (or believed) to be sufficient.

Computational Indistinguishability

- Consider two distributions X_ℓ and Y_ℓ (e.g., over strings of length ℓ).
- Let D be a distinguisher that attempts to guess whether a string s came from distribution X_ℓ or Y_ℓ.

The advantage of a distinguisher D is

$$Adv_{D,\ell} = \left|Pr_{s\leftarrow X_\ell}[D(s) = 1] - Pr_{s\leftarrow Y_\ell}[D(s) = 1]\right|$$

Definition: We say that an ensemble of distributions $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ are <u>computationally indistinguishable</u> if for all PPT distinguishers D, there is a negligible function negl(n), such that we have

$$Adv_{D,n} \leq negl(n)$$

Computational Indistinguishability

The advantage of a distinguisher D is

$$Adv_{D,\ell} = \left|Pr_{s \leftarrow X_\ell}[D(s) = 1] - Pr_{s \leftarrow Y_\ell}[D(s) = 1]\right|$$

- Looks similar to definition of PRGs
  - X_n is distribution $G(U_n)$ and
  - Y_n is uniform distribution $U_{\ell(n)}$ over strings of length $\ell(n)$.

Computational Indistinguishability

Definition: We say that an ensemble of distributions $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ are computationally indistinguishable if for all PPT distinguishers D, there is a negligible function negl(n), such that we have

$$Adv_{D,n} \leq negl(n)$$

Theorem 7.32: Let t(n) be a polynomial and let $P_n = X_n^{t(n)}$ and $Q_n = Y_n^{t(n)}$ then the ensembles $\{P_n\}_{n\in\mathbb{N}}$ and $\{Q_n\}_{n\in\mathbb{N}}$ are computationally indistinguishable

Computational Indistinguishability

Definition: We say that an ensemble of distributions $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ are computationally indistinguishable if for all PPT distinguishers D, there is a negligible function negl(n), such that we have

$$Adv_{D,n} \leq negl(n)$$

Fact: Let $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ be computationally indistinguishable and let $\{Z_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ be computationally indistinguishable

Then

$\{X_n\}_{n\in\mathbb{N}}$ and $\{Z_n\}_{n\in\mathbb{N}}$ are computationally indistinguishable

CS 555: Week 9: Topic 2 Number Theory/Public Key-Cryptography

Public Key Cryptography

- **Key-Exchange Problem:**
  - Obi-Wan and Yoda want to communicate securely
  - Suppose that
    - Obi-Wan and Yoda don’t have time to meet privately and generate one
    - Obi-Wan and Yoda share an asymmetric key with Anakin
    - Can they use Anakin to exchange a secret key?



Public Key Cryptography

- Key-Exchange Problem:
  - Obi-Wan and Yoda want to communicate securely
  - Suppose that
    - Obi-Wan and Yoda don’t have time to meet privately and generate one
    - Obi-Wan and Yoda share an asymmetric key with Anakin
    - Can they use Anakin to exchange a secret key?
    - **Remark**: Obi-Wan and Yoda both trust Anakin, but would prefer to keep the key private just in case.

Public Key Cryptography

- Key-Exchange Problem:
  - Obi-Wan and Yoda want to communicate securely
  - Suppose that
    - Obi-Wan and Yoda don't have time to meet privately and generate one
    - Obi-Wan and Yoda share an asymmetric key with Anakin
    - Can they use Anakin to exchange a secret key?
    - **Remark**: Obi-Wan and Yoda both trust Anakin, but would prefer to keep the key private just in case.
- Need for Public-Key Crypto
  - We can solve the key-exchange problem using public-key cryptography.
  - No solution is known using symmetric key cryptography alone

Public Key Cryptography

- Suppose we have n people and each pair of people want to be able to maintain a secure communication channel.
  - How many private keys per person?
  - **Answer**: n-1

- Key Explosion Problem
  - n can get very big if you are Google or Amazon!

Number Theory

- Key tool behind public key-crypto
  - RSA, El-Gamal, Diffie-Hellman Key Exchange

- Aside: don’t worry we will still use symmetric key crypto
  - It is more efficient in practice
  - First step in many public key-crypto protocols is to generate symmetric key
    - Then communicate using authenticated encryption

Polynomial Time Factoring Algorithm?

FindPrimeFactor

Input: N

For i=1,…,N

  if N/i is an integer then

    Output I

Running time: O(N) steps

Correctness: Always returns a factor

Did we just break RSA?

Polynomial Time Factoring Algorithm?

FindPrimeFactor

Input: N

For i=1,…,N

  if N/i is an integer then

    Output I

Running time: O(N) steps

Correctness: Always returns a factor

We measure running time of an arithmetic algorithm (multiply, divide, GCD, remainder) in terms of the number of bits necessary to encode the inputs.

How many bits $||N||$ to encode N?
Answer: $||N||= \log_2(N)$

Polynomial Time Operations on Integers

Polynomial time in $||a||$and $||b||$

- Addition
- Multiplication
- Division with Remainder
  - **Input: a** and divisor **b**
  - **Output**: quotient q and remainder r < **b** such that

$$\boldsymbol{a} = q\boldsymbol{b} + r$$

Convenient Notation: r = **a** mod **b**

- Greatest Common Divisor
  - **Example:** gcd(9,15) = 3
- Extended GCD(**a**,**b**)
  - Output integers X,Y such that

$$X\boldsymbol{a} + Y\boldsymbol{b} = \gcd(\boldsymbol{a}, \boldsymbol{b})$$

Polynomial Time Operations on Integers

- Division with Remainder
  - **Input:** a and b
  - **Output**: quotient q and remainder r < b such that

$$\boldsymbol{a} = q\boldsymbol{b} + r$$

- Greatest Common Divisor
  - **Key Observation:** if $\boldsymbol{a} = q\boldsymbol{b} + r$

  Then gcd(**a,b**) = gcd(r, **b**)=gcd(**a** mod **b**, **b**)

  Proof:

  - Let d = gcd(**a**,**b**). Then d divides both a and b. Thus, d also divides r=a-qb.
    →d=gcd(**a**,**b**) $\leq$ gcd(r, **b**)
  - Let d' = gcd(r, **b**). Then d' divides both b and r. Thus, d' also divides a = qb+r.
    →gcd(**a**,**b**) $\geq$ gcd(r, **b**)=d'
  - Conclusion: d=d'.

More Polynomial Time Operations on Integers

- **(Modular Arithmetic)** The following operations are polynomial time in $\|a\|$ and $\|b\|$ and $\|N\|$.

1. Compute [**a** mod **N**]
2. Compute sum [(**a**+**b**) mod **N**], difference [(**a**-**b**) mod **N**] or product [**ab** mod **N**]
3. Determine whether **a** has an inverse $\mathbf{a}^{-1}$ such that 1=[$\mathbf{a}\mathbf{a}^{-1}$ mod **N**]
4. Find $\mathbf{a}^{-1}$ if it exists
5. Compute the exponentiation [$\mathbf{a}^{\mathbf{b}}$ mod **N**]

More Polynomial Time Operations on Integers

- (Modular Arithmetic) The
  in |

1. Compute [**a** mod **N**]
2. Compute sum [(
   [**ab** mod **N**]
3. Determine whether **a** has an inverse $\mathbf{a}^{-1}$ such that 1=[$\mathbf{aa}^{-1}$ mod **N**]
4. Find $\mathbf{a}^{-1}$ if it exists
5. Compute the exponentiation [$\mathbf{a}^{\mathbf{b}}$ mod **N**]

Remark: Part 3 and 4 use extended GCD algorithm

More Polynomial Time Operations on Integers

- (Modular Arithmetic) The following operations are polynomial time in in $||a||$ and $||b||$ and $||N||$.

1. Compute the exponentiation [$\mathbf{a}^{\mathbf{b}}$ mod **N**]

Attempt 1:

X =1

For i=1,…,b

X = X*a

What is wrong?

More Polynomial Time Operations on Integers

(Modular Arithmetic) The following operations are polynomial time in $\|a\|$, $\|b\|$ and $\|N\|$.

1. Compute the exponentiation [$\mathbf{a}^{\mathbf{b}}$ mod **N**]

Attempt 2:

If (b=0) return 1

X[0]=a;

For i=1,…,$\log_2$(b)+1

    X[i] = X[i-1]*X[i-1]   // invariant: X[i] = $\boldsymbol{a}^{2^i}$

$$[\mathbf{a}^{\mathbf{b}} \text{ mod } \mathbf{N}] = \boldsymbol{a}^{\sum_i \boldsymbol{b}[i] 2^i} \text{mod } \mathbf{N}$$

$$= \prod_i \mathbf{b}[\mathrm{i}]\ \mathrm{X[i]}\ \text{ mod } \mathbf{N}$$

What is wrong?

The number of bits in $a^{2^{\|b\|+1}}$ is O($2^{\|b\|+1}$).

More Polynomial Time Operations on Integers

(Modular Arithmetic) The following operations are polynomial time in $\|a\|$, $\|b\|$ and $\|N\|$.

1. Compute the exponentiation [$\mathbf{a}^{\mathbf{b}}$ mod **N**]

Fixed Algorithm:

If (b=0) return 1

X[0]=a;

For i=1,…,$\log_2(b)+1$

    X[i] = X[i-1]*X[i-1] mod **N**   // Invariant: X[i] = $\boldsymbol{a}^{2^i}$ mod **N**

$$[\mathbf{a}^{\mathbf{b}} \bmod \mathbf{N}] = \boldsymbol{a}^{\sum_i \boldsymbol{b}[i] 2^i} \bmod \mathbf{N}$$

$$= \prod_i \mathbf{b}[\mathrm{i}]\ \mathrm{X}[\mathrm{i}] \bmod \mathbf{N}$$

More Polynomial Time Operations on Integers

(Sampling) Let

$$\mathbb{Z}_N = \{1, \ldots, N\}$$
$$\mathbb{Z}_N^* = \{x \in \mathbb{Z}_N \mid \gcd(N, x) = 1\}$$

Examples:

$$\mathbb{Z}_6^* = \{1,5\}$$

$$\mathbb{Z}_7^* = \{1,2,3,4,5,6\}$$

More Polynomial Time Operations on Integers

(Sampling) Let

$$\mathbb{Z}_N = \{1, \dots, N\}$$
$$\mathbb{Z}_N^* = \{x \in \mathbb{Z}_N \mid \gcd(N, x) = 1\}$$

- There is a probabilistic polynomial time algorithm (in |N|) to sample from $\mathbb{Z}_N^*$ and $\mathbb{Z}_N$
- Algorithm to sample from $\mathbb{Z}_N^*$ is allowed to output "fail" with negligible probability in |N|.
- Conditioned on not failing sample must be uniform.

Useful Facts

$$x, y \in \mathbb{Z}_N^* \rightarrow [xy \bmod \mathrm{N}] \in \mathbb{Z}_N^*$$

Example 1: $\mathbb{Z}_8^* = \{1,3,5,7\}$

$$[3 \times 7 \bmod 8] = [21 \bmod 8] = [5 \bmod 8] \in \mathbb{Z}_N^*$$

Proof: gcd(xy,N) = d

Suppose d>1 then for some prime p and integer q we have d=pq.

Now p must divide N and xy (by definition) and hence p must divide either x or y.

(WLOG) say p divides x. In this case gcd(x,N)=p > 1, which means $x \notin \mathbb{Z}_N^*$

More Useful Facts

$$x, y \in \mathbb{Z}^*_{\mathrm{N}} \rightarrow [xy \bmod \mathrm{N}] \ \in \mathbb{Z}^*_{\mathrm{N}}$$

Fact 1: Let $\boldsymbol{\phi}(\boldsymbol{N}) = |\mathbb{Z}^*_{\mathrm{N}}|$ then for any $x \in \mathbb{Z}^*_{\mathrm{N}}$ we have

$$\left[x^{\boldsymbol{\phi}(\boldsymbol{N})} \bmod \mathrm{N}\right] = 1$$

Example: $\mathbb{Z}^*_8 = \{1,3,5,7\}$, $\phi(8) = 4$

$$\left[3^{\mathbf{4}} \bmod 8\right] = [9 \times 9 \bmod 8] = 1$$

$$\left[5^{\mathbf{4}} \bmod 8\right] = [25 \times 25 \bmod 8] = 1$$

$$\left[7^{\mathbf{4}} \bmod 8\right] = [49 \times 49 \bmod 8] = 1$$

More Useful Facts

$$x, y \in \mathbb{Z}_{\mathrm{N}}^{*} \rightarrow [xy \bmod \mathrm{N}] \in \mathbb{Z}_{\mathrm{N}}^{*}$$

Fact 1: Let $\boldsymbol{\phi}(\boldsymbol{N}) = |\mathbb{Z}_{\mathrm{N}}^{*}|$ then for any $x \in \mathbb{Z}_{\mathrm{N}}^{*}$ we have $\left[x^{\boldsymbol{\phi}(\boldsymbol{N})} \bmod \mathrm{N}\right] = x$

Fact 2: Let $\boldsymbol{\phi}(\boldsymbol{N}) = |\mathbb{Z}_{\mathrm{N}}^{*}|$ and let $N = \prod_{i=1}^{m} p_i^{e_i}$, where each p_i is a distinct prime number and $e_i > 0$ then

$$\boldsymbol{\phi}(\boldsymbol{N}) = \prod_{i=1}^{m} (p_i - 1) p_i^{e_i - 1} = n \prod_{i=1}^{m} \left(1 - \frac{1}{p_i}\right)$$

Recap

- Polynomial time algorithms (in bit lengths $||\boldsymbol{a}||$, $||\boldsymbol{b}||$ and $||\mathbf{N}||$) to do important stuff
  - GCD(**a**,**b**)
  - Find inverse $\mathbf{a}^{-1}$ of **a** such that 1=[$\mathbf{a}\mathbf{a}^{-1}$ mod **N**] (if it exists)
  - PowerMod: [$\mathbf{a}^{\mathbf{b}}$ mod **N**]
  - Draw uniform sample from $\mathbb{Z}_N^* = \{x \in \mathbb{Z}_N | \gcd(N, x) = 1\}$
    - Randomized PPT algorithm

More Useful Facts

$$x, y \in \mathbb{Z}_{\mathrm{N}}^{*} \rightarrow [xy \bmod \mathrm{N}] \in \mathbb{Z}_{\mathrm{N}}^{*}$$

Fact 1: Let $\boldsymbol{\phi}(\boldsymbol{N}) = |\mathbb{Z}_{\mathrm{N}}^{*}|$ then for any $x \in \mathbb{Z}_{\mathrm{N}}^{*}$ we have

$$\left[x^{\boldsymbol{\phi}(\boldsymbol{N})} \bmod \mathrm{N}\right] = 1$$

Example: $\mathbb{Z}_{8}^{*} = \{1,3,5,7\}$, $\phi(8) = 4$

$$\left[3^{\mathbf{4}} \bmod 8\right] = [9 \times 9 \bmod 8] = 1$$
$$\left[5^{\mathbf{4}} \bmod 8\right] = [25 \times 25 \bmod 8] = 1$$
$$\left[7^{\mathbf{4}} \bmod 8\right] = [49 \times 49 \bmod 8] = 1$$

More Useful Facts

$$x, y \in \mathbb{Z}_{\mathrm{N}}^{*} \rightarrow [xy \bmod \mathrm{N}] \in \mathbb{Z}_{\mathrm{N}}^{*}$$

Fact 1: Let $\boldsymbol{\phi}(\boldsymbol{N}) = |\mathbb{Z}_{\mathrm{N}}^{*}|$ then for any $x \in \mathbb{Z}_{\mathrm{N}}^{*}$ we have $\left[x^{\boldsymbol{\phi}(\boldsymbol{N})} \bmod \mathrm{N}\right] = 1$

Fact 2: Let $\boldsymbol{\phi}(\boldsymbol{N}) = |\mathbb{Z}_{\mathrm{N}}^{*}|$ and let $N = \prod_{i=1}^{m} p_i^{e_i}$, where each p_i is a distinct prime number and $e_i > 0$ then

$$\boldsymbol{\phi}(\boldsymbol{N}) = \prod_{i=1}^{m} (p_i - 1) p_i^{e_i - 1} = n \prod_{i=1}^{m} \left(1 - \frac{1}{p_i}\right)$$

More Useful Facts

Fact 2: Let $\boldsymbol{\phi(N)} = |\mathbb{Z}_N^*|$ and let $N = \prod_{i=1}^{m} p_i^{e_i}$, where each p_i is a distinct prime number and $e_i > 0$ then

$$\boldsymbol{\phi(N)} = \prod_{i=1}^{m} (p_i - 1) p_i^{e_i - 1} = n \prod_{i=1}^{m} \left(1 - \frac{1}{p_i}\right)$$

Example 0: Let p be a prime so that $\mathbb{Z}_p^* = \{1, \dots, p - 1\}$

$$\boldsymbol{\phi(p)} = p\left(1 - \frac{1}{p}\right) = p - 1$$

More Useful Facts

Fact 2: Let $\boldsymbol{\phi}(\boldsymbol{N}) = |\mathbb{Z}_N^*|$ and let $N = \prod_{i=1}^{m} p_i^{e_i}$, where each p_i is a distinct prime number and $e_i > 0$ then

$$\boldsymbol{\phi}(\boldsymbol{N}) = \prod_{i=1}^{m} (p_i - 1) p_i^{e_i - 1} = n \prod_{i=1}^{m} \left(1 - \frac{1}{p_i}\right)$$

Example 1: N = 9 = 3^2 (m=1, e_1=2)

$$\boldsymbol{\phi}(\boldsymbol{9}) = \prod_{i=1}^{1} (p_i - 1) p_i^{2-1} = 2 \times 3$$

More Useful Facts

Example 1: N = 9 = 3^2 (m=1, e_1=2)

$$\boldsymbol{\phi}(\mathbf{9}) = \prod_{i=1}^{1}(p_i - 1)p_i^{2-1} = 2 \times 3$$

Double Check: $\mathbb{Z}_9^* = \{1,2,4,5,7,8\}$

More Useful Facts

Fact 2: Let $\boldsymbol{\phi}(\boldsymbol{N}) = |\mathbb{Z}_N^*|$ and let $N = \prod_{i=1}^{m} p_i^{e_i}$, where each p_i is a distinct prime number and $e_i > 0$ then

$$\boldsymbol{\phi}(\boldsymbol{N}) = \prod_{i=1}^{m} (p_i - 1) p_i^{e_i - 1} = n \prod_{i=1}^{m} \left(1 - \frac{1}{p_i}\right)$$

Example 2: N = 15 = 5×3 (m=2, e_1=e_2=1)

$$\boldsymbol{\phi}(\mathbf{15}) = \prod_{i=1}^{2} (p_i - 1) p_i^{1-1} = (5-1)(3-1) = 8$$

More Useful Facts

Example 2: N = 15 = 5×3 (m=2, e_1=e_2=1)

$$\boldsymbol{\phi}(\mathbf{15}) = \prod_{i=1}^{2} (p_i - 1) p_i^{1-1} = (5-1)(3-1) = 8$$

Double Check: $\mathbb{Z}_{15}^{*} = \{1,2,4,7,8,11,13,14\}$

I count 8 elements in $\mathbb{Z}_{15}^{*}$

More Useful Facts

Fact 2: Let $\boldsymbol{\phi}(\boldsymbol{N}) = |\mathbb{Z}_N^*|$ and let $N = \prod_{i=1}^{m} p_i^{e_i}$, where each p_i is a distinct prime number and $e_i > 0$ then

$$\boldsymbol{\phi}(\boldsymbol{N}) = \prod_{i=1}^{m} (p_i - 1) p_i^{e_i - 1} = n \prod_{i=1}^{m} \left(1 - \frac{1}{p_i}\right)$$

Special Case: N = pq   (p and q are distinct primes)

$$\boldsymbol{\phi}(\boldsymbol{N}) = (p-1)(q-1)$$

More Useful Facts

Special Case: N = pq   (p and q are distinct primes)

$$\boldsymbol{\phi}(\boldsymbol{N}) = (p-1)(q-1)$$

Proof Sketch: If $x \in \mathbb{Z}_N$ is not divisible by p or q then $x \in \mathbb{Z}_N^*$**.** How many elements are not in $\mathbb{Z}_N^*$?

- **Multiples of p:** p, 2p, 3p,…,pq   (q multiples of p)
- **Multiples of q:** q, 2q,…,pq   (p multiples of q)
- **Double Counting?** N=pq is in both lists. Any other duplicates?
- No! cq = dp → q divides d (since, gcd(p,q)=1) and consequently $\mathrm{d} \geq q$
  - Hence, $\mathrm{dp} \geq pq = N$

More Useful Facts

Special Case: N = pq (p and q are distinct primes)

$$\boldsymbol{\phi(N)} = (p-1)(q-1)$$

Proof Sketch: If $x \in \mathbb{Z}_N$ is not divisible by p or q then $x \in \mathbb{Z}_N^*$. How many elements are not in $\mathbb{Z}_N^*$?

- **Multiples of p:** p, 2p, 3p,…,pq (q multiples of p)
- **Multiples of q:** q, 2q,…,pq (p multiples of q)
- **Answer:** p+q-1 elements are not in $\mathbb{Z}_N^*$

$$\boldsymbol{\phi(N) = N - (p + q - 1)}$$
$$\mathbf{= pq - p - q + 1 = (p-1)(q-1)}$$

Groups

Definition: A (finite) group is a (finite) set $\mathbb{G}$ with a binary operation $\circ$ (over G) for which we have

- (**Closure**:) For all $\mathrm{g}, \mathrm{h} \in \mathbb{G}$ we have $\mathrm{g} \circ \mathrm{h} \in \mathbb{G}$
- **(Identity**:) There is an element $\mathrm{e} \in \mathbb{G}$ such that for all $\mathrm{g} \in \mathbb{G}$ we have
$$\mathrm{g} \circ \mathrm{e} = \mathrm{g} = \mathrm{e} \circ \mathrm{g}$$
- **(Inverses**:) For each element $\mathrm{g} \in \mathbb{G}$ we can find $\mathrm{h} \in \mathbb{G}$ such that $\mathrm{g} \circ \mathrm{h} = \mathrm{e}$. We say that h is the inverse of g.
- (**Associativity:**) For all $\mathrm{g}_1, \mathrm{g}_2, \mathrm{g}_3 \in \mathbb{G}$ we have
$$(\mathrm{g}_1 \circ \mathrm{g}_2) \circ \mathrm{g}_3 = \mathrm{g}_1 \circ (\mathrm{g}_2 \circ \mathrm{g}_3)$$

We say that the group is **abelian** if

- (**Commutativity:**) For all $\mathrm{g}, \mathrm{h} \in \mathbb{G}$ we have $\mathrm{g} \circ \mathrm{h} = \mathrm{h} \circ \mathrm{g}$

Abelian Groups (Examples)

- **Example 1:** $\mathbb{Z}_N$ when $\circ$ denotes addition modulo N
- Identity: 0, since $0 \circ x = [0+x \bmod N] = [x \bmod N]$.
- Inverse of x? Set $x^{-1}=N-x$ so that $[x^{-1}+x \bmod N] = [N-x+x \bmod N] = 0$.

- **Example 2:** $\mathbb{Z}_N^*$ when $\circ$ denotes multiplication modulo N
- Identity: 1, since $1\circ x = [1(x) \bmod N] = [x \bmod N]$.
- Inverse of x? Run extended GCD to obtain integers a and b such that

$$ax + bN = \gcd(x, N) = 1$$

Observe that: $x^{-1} = a$. Why?

Abelian Groups (Examples)

- **Example 1:** $\mathbb{Z}_N$ when ∘ denotes addition modulo N
- Identity: 0, since 0 ∘ x =[0+x mod N] = [x mod N].
- Inverse of x? Set x^{-1}=N-x so that [x^{-1}+x mod N] = [N-x+x mod N] = 0.

- **Example 2:** $\mathbb{Z}_N^*$ when ∘ denotes multiplication modulo N
- Identity: 1, since 1∘ x =[1(x) mod N] = [x mod N].
- Inverse of x? Run extended GCD to obtain integers a and b such that

$$ax + bN = \gcd(x, N) = 1$$

Observe that: x^{-1} = a, since [ax mod N] = [1-bN mod N] = 1

Groups

Lemma 8.13: Let $\mathbb{G}$ be a group with a binary operation $\circ$ (over G) and let $\mathrm{a}, \mathrm{b}, \mathrm{c} \in \mathbb{G}$. If $\mathrm{a} \circ \mathrm{c} = \mathrm{b} \circ \mathrm{c}$ then $\mathrm{a} = \mathrm{b}$.

Proof Sketch: Apply the unique inverse to c^{-1} both sides.

$\mathrm{a} \circ \mathrm{c} = \mathrm{b} \circ \mathrm{c}$ → $(\mathrm{a} \circ \mathrm{c}) \circ c^{-1} = (\mathrm{b} \circ \mathrm{c}) \circ c^{-1}$

→ $\mathrm{a} \circ (\mathrm{c} \circ c^{-1}) = \mathrm{b} \circ (\mathrm{c} \circ c^{-1})$

→ $\mathrm{a} \circ (e) = \mathrm{b} \circ (e)$

→ $\mathrm{a} = \mathrm{b}$

(**Remark**: it is not to difficult to show that a group has a *unique* identity and that inverses are *unique*).

Group Exponentiation

Definition: Let $\mathbb{G}$ be a group with a binary operation $\circ$ (over G) let m be a positive integer and let $\mathrm{g} \in \mathbb{G}$ be a group element then we define

$$g^m = \underbrace{\mathrm{g} \circ \cdots \circ \mathrm{g}}_{\text{m times}}$$

Theorem: Let $\mathbb{G}$ be finite group with size $\mathrm{m} = |\mathbb{G}|$ and let $\mathrm{g} \in \mathbb{G}$ be a group element then g^m=1 (where 1 denotes the unique identity of $\mathbb{G}$).

Group Exponentiation

Theorem 8.14: Let $\mathbb{G}$ be finite group with size $\mathrm{m} = |\mathbb{G}|$ and let $\mathrm{g} \in \mathbb{G}$ be a group element then g^m=1 (where 1 denotes the unique identity of $\mathbb{G}$).

Proof: (for abelian group) Let $\mathbb{G} = \{g_1, \dots, g_m\}$ then we claim

$$g_1 \circ \cdots \circ g_m = (g \circ g_1) \circ \cdots \circ (g \circ g_m)$$

Why? If $(g \circ g_i) = \left(g \circ g_j\right)$ then $g_j = g_i$ (by Lemma 8.13)

Group Exponentiation

Theorem 8.14: Let $\mathbb{G}$ be finite group with size $\mathrm{m} = |\mathbb{G}|$ and let $\mathrm{g} \in \mathbb{G}$ be a group element then g^m=1 (where 1 denotes the unique identity of $\mathbb{G}$).

Proof: (for abelian group) Let $\mathbb{G} = \{g_1, \ldots, g_m\}$ then we claim

$$g_1 \circ \cdots \circ g_m = (g \circ g_1) \circ \cdots \circ (g \circ g_m)$$

Because $\mathbb{G}$ is abelian we can re-arrange terms

$$g_1 \circ \cdots \circ g_m = (g_1 \circ \cdots \circ g_m)(g^m)$$

By Lemma 8.13 we have $1 = g^m$. QED

Group Exponentiation

Theorem 8.14: Let $\mathbb{G}$ be finite group with size $\mathrm{m} = |\mathbb{G}|$ and let $\mathrm{g} \in \mathbb{G}$ be a group element then g^m=1 (where 1 denotes the unique identity of $\mathbb{G}$).

Corollary 8.15: Let $\mathbb{G}$ be finite group with size $\mathrm{m} = |\mathbb{G}| > 1$ and let $\mathrm{g} \in \mathbb{G}$ be a group element then for any integer x we have $g^x = g^{[x \bmod m]}$.

Proof: $g^x = g^{qm+[x \bmod m]} = g^{[x \bmod m]}$, where q is unique integer such that x=qm+ $[x \bmod m]$

Group Exponentiation

Special Case: $\mathbb{Z}_N^*$ is a group of size $\boldsymbol{\phi(N)}$ so we have now proved

Corollary 8.22: For any $g \in \mathbb{Z}_\mathrm{N}^*$ and integer x we have

$$[g^x \text{mod N}] = \left[g^{[x \bmod \boldsymbol{\phi(N)}]} \text{mod N}\right]$$

Chinese Remainder Theorem

Theorem: Let N = pq (where gcd(p,q)=1) be given and let $f: \mathbb{Z}_N \rightarrow \mathbb{Z}_p \times \mathbb{Z}_q$ be defined as follows

$$f(x) = ([x \; mod \; p], [x \; mod \; q])$$

then

- f is a bijective mapping (invertible)
- f and its inverse $f^{-1}: \mathbb{Z}_p \times \mathbb{Z}_q \rightarrow \mathbb{Z}_N$ can be computed efficiently
- $f(x+y) = f(x) + f(y)$
- The restriction of f to $\mathbb{Z}_N^*$ yields a bijective mapping to $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$
- For inputs $x, y \in \mathbb{Z}_N^*$ we have $f(x)f(y) = f(xy)$

Chinese Remainder Theorem

Application of CRT: Faster computation

Example: Compute [11^{53} mod 15]

f(11)=([-1 mod 3],[1 mod 5])

f(11^{53}) =([$(-1)^{53}$ mod 3],[1^{53} mod 5])= (-1,1)

f^{-1}(-1,1)=11

Thus, 11=[11^{53} mod 15]