Cryptography

Lecture 16

Announcements

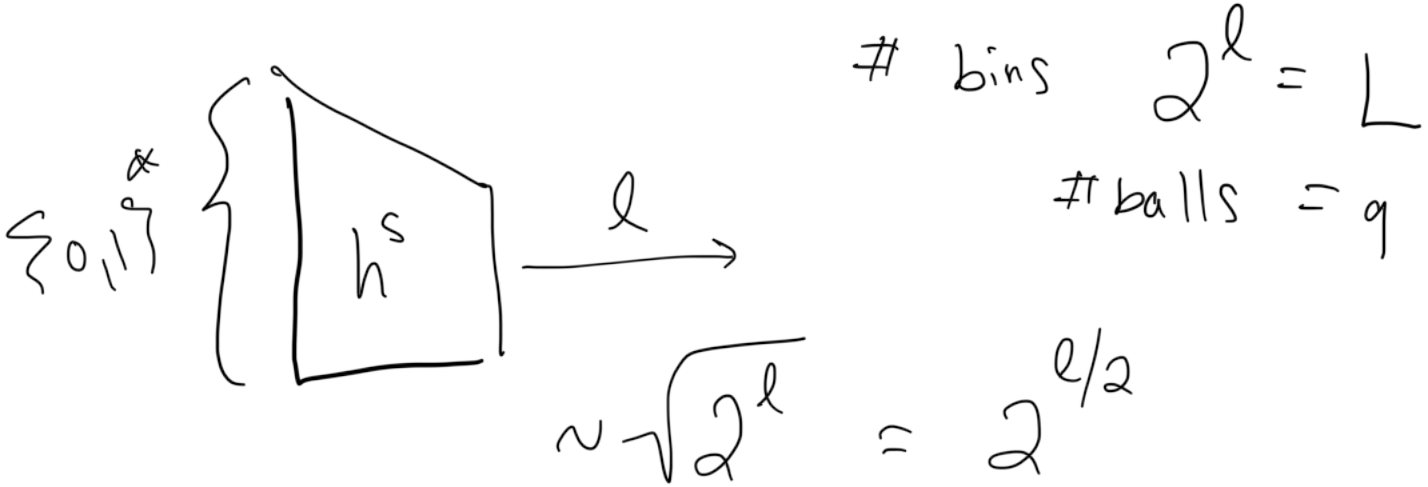
- HW4 due on 4/10

Agenda

- Finish Birthday bound + CRHF
- New Unit: Number Theory!

Preliminaries

- How much security can we hope for from a CRHF that outputs ℓ bits?
- Discuss the "**birthday bound**"
  - No matter what function is used, collisions can be found with high probability after making $2^{\ell/2}$ queries.



Given q balls, L bins what is the expected # of collisions

(2^ℓ)

Our goal: Find a q s.t. expected # of collisions ≥ 1.

To analyze. for every pair $i, j \in [q]$, $i < j$.

$$I_{ij} = \begin{cases} 1 & \text{if balls } i,j \text{ collide} \\ 0 & \text{o/w.} \end{cases}$$

$$\mathbb{E}[\#\text{ of coll.}] = \mathbb{E}\left[\sum_{\substack{i,j \\ i<j}} I_{ij}\right] \overset{\text{lin. of expectation}}{=} \sum_{\substack{i,j \\ i<j}} \mathbb{E}[I_{ij}] = \sum_{\substack{i,j \\ i<j}} \Pr[I_{ij}=1]$$

For fixed i,j

Analyze $\Pr[I_{ij}=1] = \frac{1}{L}$

$$= \sum_{\substack{i,j \\ i<j}} \frac{1}{L}$$

$$= \frac{\binom{q}{2}}{L} \approx \frac{q^2}{L} = 1, \text{ set } q \approx \sqrt{L}.$$

Weaker Notions of Security

- Second preimage or target collision resistance: Given s and a uniform x it is infeasible for a ppt adversary to find $x' \neq x$ such that $H^s(x') = H^s(x)$.
- Preimage resistance: Given s and uniform y it is infeasible for a ppt adversary to find a value x such that $H^s(x) = y$.

Hash Functions From Block Ciphers

- Hash functions are generally constructed in two steps:
  - First, a compression function (fixed-length hash function) h is designed
  - Next, some mechanism (e.g. Merkle-Damgard) is used to extend h so as to handle arbitrary input lengths
- We will focus on the first step

$h : \{0,1\}^{2n} \rightarrow \{0,1\}^{n}$

✓ previously

Hash Functions From Block Ciphers

- Davies-Meyer construction:
  - F is a block-cipher will n-bit key and ℓ-bit block length.

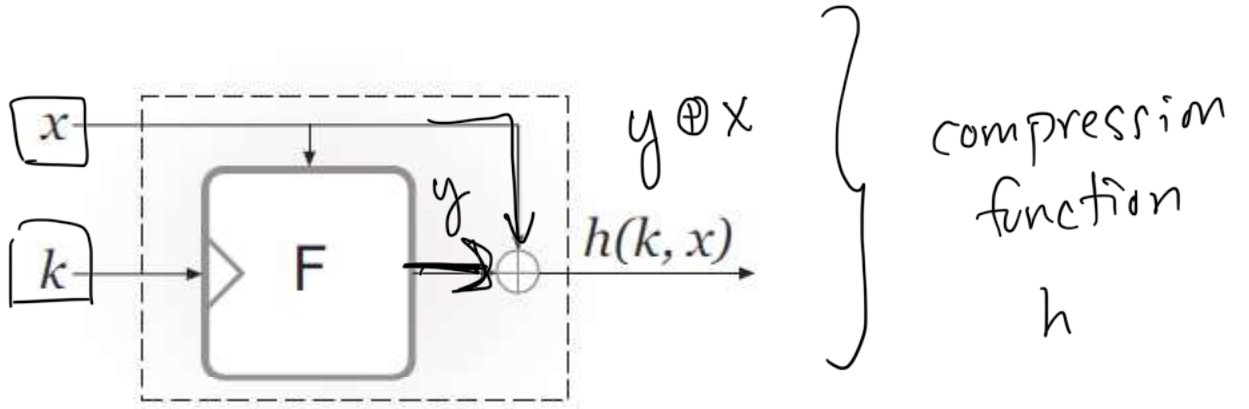


FIGURE 6.10: The Davies-Meyer construction.

  - Above forms a compression function from $n + \ell$ bits to n bits.

$y \oplus x_2$

1st attempt: Use a block ciphn as a compression function

$F: \{0,1\}^n \times \{0,1\}^\ell \longrightarrow \{0,1\}^\ell$

$x = x_1 \| x_2$

$h(x) = F_{x_1}(x_2) = y$

$$x_2 = F_{x_1}^{-1}(y).$$

Security Analysis

- We do not know how to prove collision-resistance of the compression function based on the assumption that F is a strong PRP.
- Requires stronger assumption that F behaves like an **ideal cipher**.
  - Like a truly random permutation, except can query oracle on **different keys**.
  - Each key $k \in \{0,1\}^n$ specifies an independent, uniform permutation $F(k,\cdot)$ on ℓ-bit strings.

Security Analysis

- Theorem: If F is modeled as an ideal cipher, then the Davies-Meyer construction yields a collision-resistant compression function. Concretely, any attacker making $q < 2^{\ell/2}$ queries to its ideal-cipher oracles finds a collision with probability at most $q^2/2^{\ell}$.

random hash function

MD5

- 128-bit output length.
- Designed in 1991, and for several years was believed to be collision-resistant. Over a period of several years, various weaknesses began to be found in MD5 but
- these did not appear to lead to any easy way to find collisions.
- In 2004 a team of Chinese cryptanalysts presented a new method for finding
- collisions in MD5 and were able to demonstrate an explicit collision!
- Since then, the attack has been improved—collisions can be found in under a minute on a desktop PC—and extended so that even "controlled collisions" (e.g., two postscript files generating arbitrary viewable content) can be found.
- Due to these attacks, MD5 should no longer be used today for any application requiring cryptographic security.

SHA-0, SHA-1, SHA-2

- The Secure Hash Algorithm (SHA) refers to a series of cryptographic hash functions standardized by NIST.
- SHA-1, was introduced in 1995. This algorithm has a 160-bit output length, and supplanted a predecessor called SHA-0 which was withdrawn due to unspecified flaws discovered in that algorithm.
- Theoretical analysis over the past few years indicates that collisions in SHA-1 can be found using significantly fewer than the 280 hash function evaluations that would be necessary using a birthday attack.
- Recently an explicit collision has been found.
- It is therefore recommended to migrate to SHA-2, which does not currently appear to have the same weaknesses.
- SHA-2 comprises two related functions: SHA-256 and SHA-512, with 256- and 512-bit output lengths, respectively.

SHA-0, SHA-1, SHA-2

- All hash functions in the SHA family are constructed using the same basic design:
  - A compression function is first defined using the Davies-Meyer construction as applied to some block cipher
  - Extended to support arbitrary length inputs using the Merkle-Damg°ard transform.
- The block cipher in each case was designed specifically for building the compression function.
  - Block ciphers SHACAL-1 (for SHA-1) and SHACAL-2 (for SHA-2). Have large block lengths (160 and 256 bits respectively) and 512-bit key lengths.

SHA-3 (Keccak)

- NIST announced in late 2007 a public competition to design a new cryptographic hash function to be called SHA-3.
- Submitted algorithms were required to support both 256- and 512-bit output lengths.
- 51 first-round candidates were narrowed down to 14 in December, 2008, and these were further reduced to five finalists in 2010. The remaining candidates were subject to intense scrutiny by the cryptographic community over the next two years.
- In October, 2012, NIST announced the selection of Keccak as the winner of the competition.
- This algorithm is currently undergoing standardization as the next-generation replacement for SHA-2.

SHA-3 (Keccak)

- Keccak is unusual in several respects.
  - One of the reasons Keccak was chosen is because its structure is very different from that of SHA-1 and SHA-2.
- It is based on an **unkeyed** permutation f with a large block length of 1600 bits; this is radically different from, e.g., the Davies-Meyer construction which relies on a keyed permutation.
- Keccak does not use the Merkle-Damgard transform to handle arbitrary input lengths. Instead, it uses a newer approach called the **sponge** construction.
- Keccak—and the sponge construction more generally—can be analyzed in the random-permutation model
  - Here parties have access to an oracle for a random permutation $f : \{0,1\}^{\ell} \rightarrow \{0,1\}^{\ell}$ (and possibly its inverse).
  - This is weaker than the ideal-cipher model.

Number Theory

Modular Arithmetic

15 % 13 = 2
a P b

Definition of modulo:

We say that two integers a, b are congruent modulo p denoted by

$$a \equiv b \; mod \; p$$

$a = b \bmod p$

If

15 13

$$p \mid (a - b)$$

$13 \mid (15-2)$

13 ✓

(i.e. p divides $(a - b)$).

Modular Arithmetic

Examples: All of the following are true

$$2 \equiv 15 \; mod \; 13$$

$$28 \equiv 15 \; mod \; 13$$

$$41 \equiv 15 \; mod \; 13$$

$$-11 \equiv 15 \; mod \; 13$$

$13 \mid (15 - (-11))$

26 ✓

Modular Arithmetic

Operation: addition mod p

Regular addition, take modulo p.

Example: $8 + 10 \; mod \; 13 \equiv 18 \; mod \; 13 \equiv 5 \; mod \; 13$.

Properties of Addition mod p (p does not have to be prime)

$Z_p = \{0, \ldots, p-1\}$

Closure. $a, b \in Z_p$ $\Rightarrow$ $(a+b) \bmod p$

Identity: $\forall a \in Z_p$, $\exists$ an identity $\Rightarrow 0$ (id)
s.t. $(a + id) \bmod p \equiv a$.

Inverse: $\forall a \in Z_p$, there exists an inverse $b \in Z_p$
s.t. $(a+b) \bmod p = 0$.

$-a \equiv (p-a) \bmod p$

Associativity: $([(a+b) \bmod p] + c) \bmod p \equiv$

$(a + [(b+c) \bmod p]) \bmod p$.

Properties of Addition mod p

Consider the set Z_p of integers $\{0,1, \ldots, p-1\}$ and the operation addition mod p.

- Closure: Adding two numbers in Z_p and taking mod p yields a number in Z_p.
- Identitiy: For every $a \in Z_p$, $[0 + a] mod\ p \equiv a\ mod\ p$.
- Inverse: For every $a \in Z_p$, there exists a $b \in Z_p$ such that $a + b \equiv 0\ mod\ p$.
  - b is simply the negation of a $(b = -a)$.
  - Note that using the property of inverse, we can do subtraction. We define $c - d\ mod\ p$ to be equivalent to $c + (-d) mod\ p$.
- Associativity: For every $a, b, c \in Z_p$: $(a + b) + c = a + (b + c) mod\ p$.

Z_p is a group with respect to addition!

Definition of a Group

$\mathbb{Z}_p = \{0, \ldots p-1\}$ $+ \bmod p$

A group is a set G along with a binary operation $\circ$ for which the following conditions hold:

- Closure: For all $g, h \in G$, $g \circ h \in G$.
- Identity: There exists an identity $e \in G$ such that for all $g \in G, e \circ g = g = g \circ e$.
- Inverse: For all $g \in G$ there exists an element $h \in G$ such that $g \circ h = e = h \circ g$. Such an h is called an inverse of g.
- Associativity: For all $g_1, g_2, g_3 \in G, (g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$.

When G has a finite number of elements, we say G is finite and let $|G|$ denote the order of the group.

# of elements

$|\mathbb{Z}_p| = p$

Abelian Group

commutative

A group G with operation $\circ$ is abelian if the following holds:

- Commutativity: For all $g, h \in G, g \circ h = h \circ g$.

$+ \bmod p$

We will always deal with finite, abelian groups.

Other groups over the integers

- We will be interested mainly in multiplicative groups over the integers, since there are computational problems believed to be hard over such groups.
  - Such hard problems are the basis of number-theoretic cryptography.
- Group operation is multiplication mod p, instead of addition mod p.

Multiplication mod p

Example:

$$3 \cdot 8 \; mod \; 13 \equiv 24 \; mod \; 13 \equiv 11 \; mod \; 13.$$

Multiplicative Groups

Is Z_p a group with respect to multiplication mod p?

- Closure—YES
- Identity—YES (1 instead of 0)
- Associativity—YES
- Inverse—NO
  - 0 has no inverse since there is no integer a such that $0 \cdot a \equiv 1 \; mod \; p$.

Multiplicative Group

For p prime, define $Z_p^* = \{1, \dots, p-1\}$ with operation multiplication mod p.

We will see that Z_p^* is indeed a multiplicative group!

To prove that Z_p^* is a multiplicative group, it is sufficient to prove that every element has a multiplicative inverse (since we have already argued that all other properties of a group are satisfied).

This is highly non-trivial, we will see how to prove it using the Euclidean Algorithm.

Inefficient method of finding inverses mod p

Example: Multiplicative inverse of $9\ mod\ 11$.

$$9 \cdot 1 \equiv 9\ mod\ 11$$
$$9 \cdot 2 \equiv 18 \equiv 7\ mod\ 11$$
$$9 \cdot 3 \equiv 27 \equiv 5\ mod\ 11$$
$$9 \cdot 4 \equiv 36 \equiv 3\ mod\ 11$$
$$9 \cdot 5 \equiv 45 \equiv 1\ mod\ 11$$

What is the time complexity?

Brute force search. In the worst case must try all 10 numbers in Z^*_{11} to find the inverse.

This is exponential time! Why? Inputs to the algorithm are $(9,11)$. The length of the input is the length of the binary representation of $(9,11)$. This means that input size is approx. $\log_2 11$ while the runtime is approx. $2^{\log_2 11} = 11$. The runtime is exponential in the input length.

Fortunately, there is an efficient algorithm for computing inverses.

Euclidean Algorithm

Theorem: Let a, p be positive integers. Then there exist integers X, Y such that $Xa + Yp = \gcd(a, p)$.

Given a, p, the Euclidean algorithm can be used to compute $\gcd(a, p)$ in polynomial time. The extended Euclidean algorithm can be used to compute X, Y in polynomial time.

Proving Z_p^* is a multiplicative group

In the following we prove that every element in Z^*_p has a multiplicative inverse when p is prime. This is sufficient to prove that Z_p^* is a multiplicative group.

Proof. Let $a \in Z_p^*$. Then $\gcd(a, p) = 1$, since p is prime.

By the Euclidean Algorithm, we can find integers X, Y such that $aX + pY = \gcd(a, p) = 1$.

Rearranging terms, we get that $pY = (aX - 1)$ and so $p \mid (aX - 1)$.

By definition of modulo, this implies that $aX \equiv 1 \, mod \, p$.

By definition of inverse, this implies that X is the multiplicative inverse of a.

Note: By above, the extended Euclidean algorithm gives us a way to compute the multiplicative inverse in polynomial time.

Extended Euclidean Algorithm Example

Find: X, Y such that $9X + 23Y = \gcd(9,23) = 1$.

$$23 = 2 \cdot 9 + 5$$
$$9 = 1 \cdot 5 + 4$$
$$5 = 1 \cdot 4 + 1$$
$$4 = 4 \cdot 1 + 0$$

$$1 = 5 - 1 \cdot 4$$
$$1 = 5 - 1 \cdot (9 - 1 \cdot 5)$$
$$1 = (23 - 2 \cdot 9) - \big(9 - (23 - 2 \cdot 9)\big)$$
$$1 = 2 \cdot 23 - 5 \cdot 9$$

$-5 = 18\ mod\ 23$ is the multiplicative inverse of $9\ mod\ 23$.

Time Complexity of Euclidean Algorithm

When finding $\gcd(a, b)$, the "b" value gets halved every two rounds.

Why?

Time complexity: $2\log(b)$.

This is polynomial in the length of the input.

Why?

Getting Back to Z_p^*

Group $Z_p^* = \{1, \ldots, p-1\}$ operation: multiplication modulo p.

Order of a finite group is the number of elements in the group.

Order of Z_p^* is $p - 1$.