Cryptography

Lecture 12

Announcements

- HW3 due 3/8 at 11:59pm. No late homeworks accepted after 2pm on 3/11

Agenda

- This time:
  - Domain Extension for CRHF
    - (Merkle-Damgard) (K/L 5.2)

Collision Resistant Hashing

Collision Resistant Hashing

Definition: A hash function (with output length ℓ) is a pair of ppt algorithms (Gen, H) satisfying the following:

- Gen takes as input a security parameter 1^n and outputs a key s. We assume that 1^n is implicit in s.
- H takes as input a key s and a string $x \in \{0,1\}^*$ and outputs a string $H^s(x) \in \{0,1\}^{\ell(n)}$.

If H^s is defined only for inputs $x \in \{0,1\}^{\ell'(n)}$ and $\ell'(n) > \ell(n)$, then we say that (Gen, H) is a fixed-length hash function for inputs of length ℓ'. In this case, we also call H a compression function.

The collision-finding experiment

$Hashcoll_{A,\Pi}(n)$:

1. A key s is generated by running $Gen(1^n)$.
2. The adversary A is given s and outputs x, x'. (If Π is a fixed-length hash function for inputs of length $\ell'(n)$, then we require $x, x' \in \{0,1\}^{\ell'(n)}$.)
3. The output of the experiment is defined to be 1 if and only if $x \neq x'$ and $H^s(x) = H^s(x')$. In such a case we say that A has found a collision.

$\ell(n)$
output length

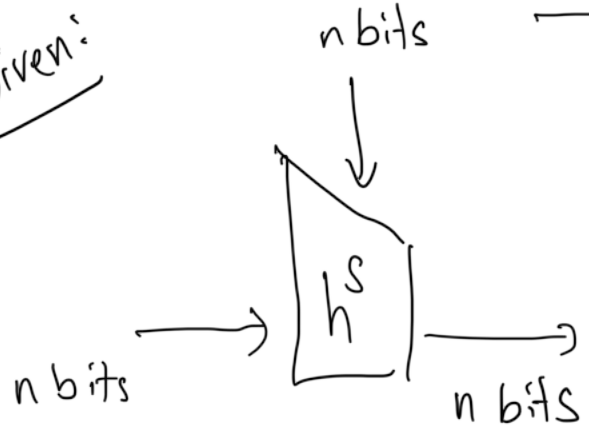
Security Definition

Definition: A hash function $\Pi = (Gen, H)$ is collision resistant if for all ppt adversaries A there is a negligible function neg such that

$$\Pr\left[Hashcoll_{A,\Pi}(n) = 1\right] \leq neg(n).$$

Domain Extension

Given:



Assume we start w/ a compression function

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

Hard to find collisions on h^s

Goal: $H^s : \{0,1\}^* \to \{0,1\}^n$

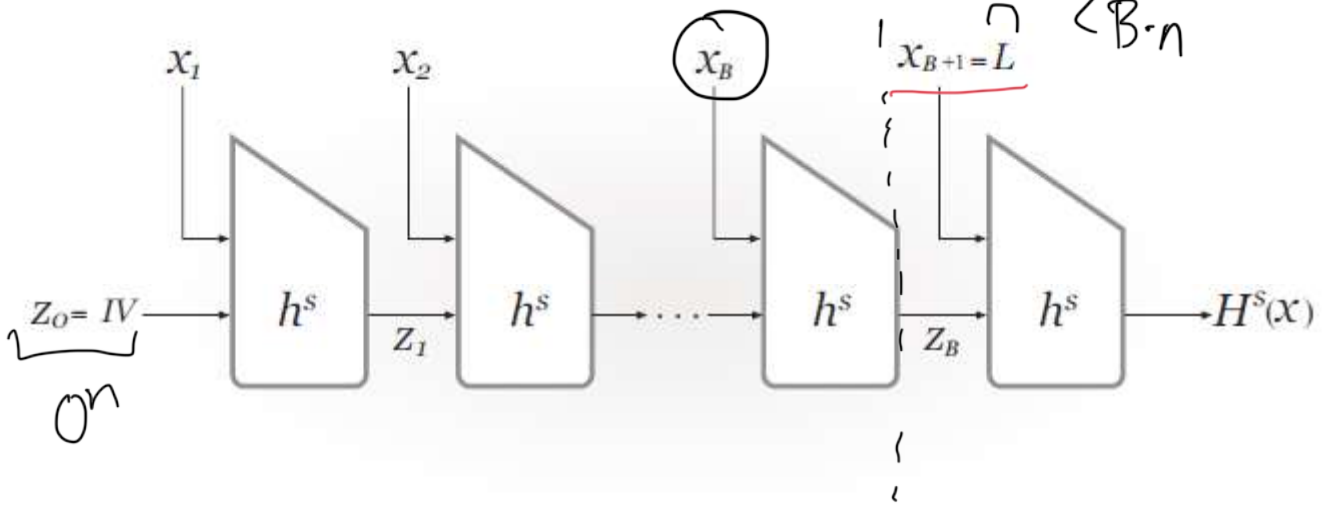
We want H^s to be collision-resistant

The Merkle-Damgard Transform

B blocks of length n

have to pad up to length $B \cdot n$

$< B \cdot n$



or

Cryptocurrencies

Blockchain

PoW

FIGURE 5.1: The Merkle-Damgård transform.

x || pad

x

H(x||pad)

H(x||pad)

Z

The Merkle-Damgard Transform

Let (Gen, h) be a fixed-length hash function for inputs of length $2n$ and with output length n. Construct hash function (Gen, H) as follows:

- Gen: remains unchanged
- H: on input a key s and a string $x \in \{0,1\}^*$ of length $L < 2^n$, do the following:
  1. Set $B := \left\lceil \frac{L}{n} \right\rceil$ (i.e., the number of blocks in x). Pad x with zeros so its length is a multiple of n. Parse the padded result as the sequence of n-bit blocks $x_1, \ldots, x_B$. Set $x_{B+1} := L$, where L is encoded as an n-bit string.
  2. Set $z_0 := 0^n$. (This is also called the IV.)
  3. For $i = 1, \ldots, B+1$, compute $z_i := h^s(z_{i-1} || x_i)$.
  4. Output z_{B+1}.

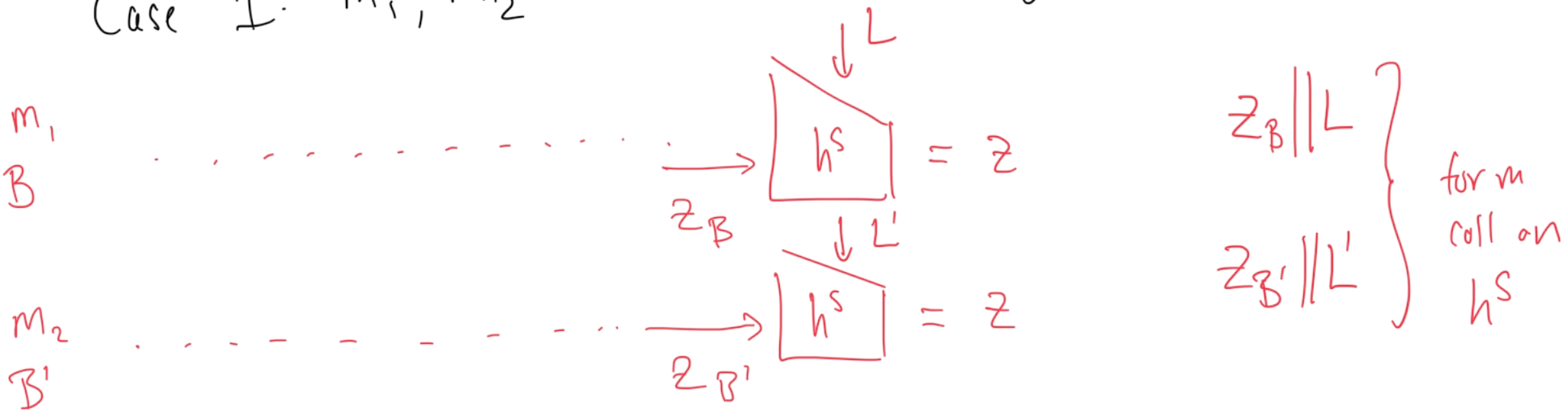
Security of Merkle-Damgard

Theorem: If (Gen, h) is collision resistant, then so is (Gen, H).

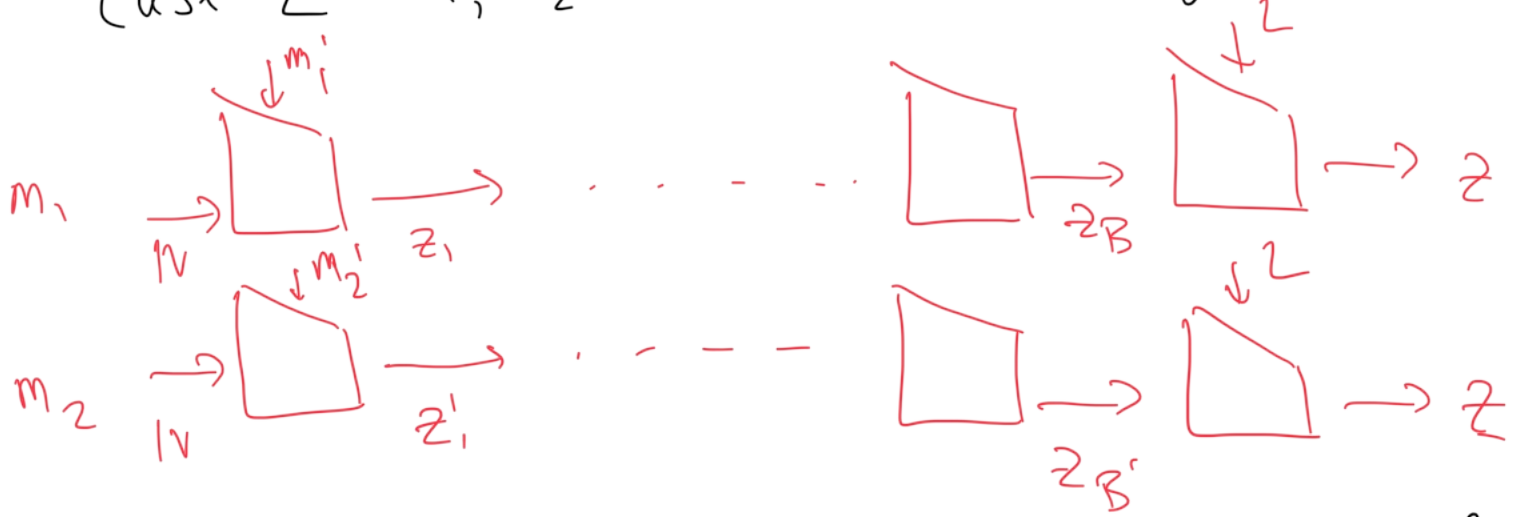
Suppose an efficient adversary finds a collision on (Gen, H) returns m_1, m_2 $m_1 \neq m_2$ s.t. $H^s(m_1) = H^s(m_2)$

Use this adversary to find a collision on (Gen, h)

Case 1: m_1, m_2 have different lengths $L \neq L'$



Case 2: m_1, m_2 have the same length L



Start at the end: for each invocation of h^S:
either: I find a collision OR step back one block.
This must succeed. Otherwise $m_1 = m_2$ (contradiction).

Try to use MD as a MAC

Key of MAC

to MAC message m output as tag: $H^s(K \| m)$

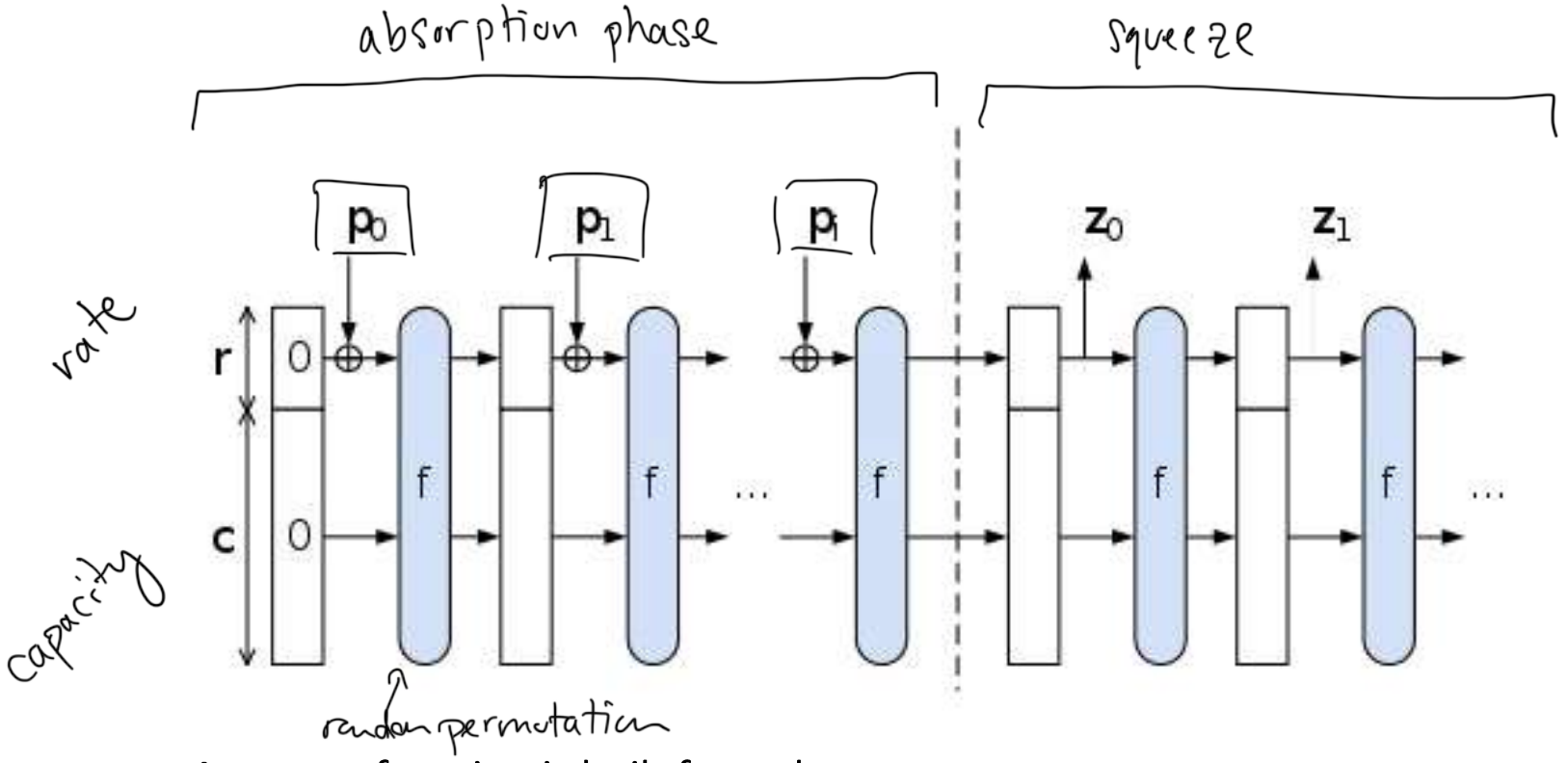
NOT secure: length extension attack.

Would like that MD behaves like a Random Oracle

SHA-1
SHA-2 (SHA-256, SHA-512) } MD

SHA-3 Uses a different

Sponge



A sponge function is built from three components:

- a state memory, *S,* containing *b* bits,
- a public, truly random permutation *f*
- a padding function *P*

The state memory is divided into two sections:

- one of size *r* (the bitrate) and
- the other of size *c* (the capacity).

These sections are denoted *R* and *C* respectively.

The padding function appends enough bits to the input string so that the length of the padded input is a whole multiple of the bitrate, *r.* The padded input can thus be broken into *r*-bit blocks.

Operation

The sponge function operates as follows:

- The state *S* is initialized to zero
- The input string is padded. This means the input p is transformed into blocks of *r* bits using the padding function P.
- R is XORed with the first *r*-bit block of padded input
- S is replaced by *f*(S)
- R is XORed with the next *r*-bit block of padded input (if any)
- S is replaced by *f*(S)

…

The process is repeated until all the blocks of the padded input string are used up ("absorbed" in the sponge metaphor).

The sponge function output is now ready to be produced ("squeezed out") as follows:

- The R portion of the state memory is the first *r* bits of output
- If more output bits are desired, S is replaced by *f*(S)
- The R portion of the state memory is the next *r* bits of output

…

The process is repeated until the desired number of output bits are produced. If the output length is not a multiple of *r* bits, it will be truncated.

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.

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

Hash Functions From Block Ciphers

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

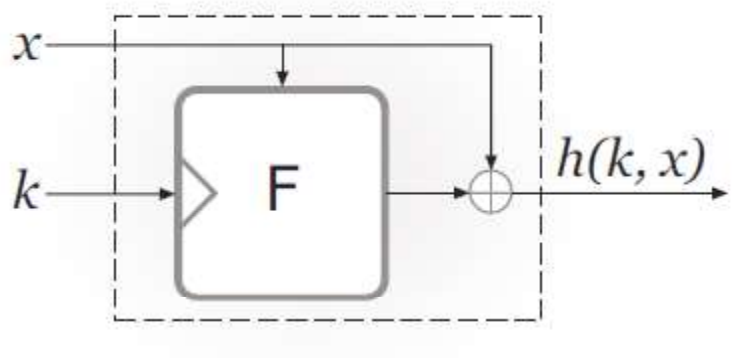


FIGURE 6.10: The Davies-Meyer construction.

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

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 $q^2/2^{\ell}$.

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.