Cryptographic Hash Functions (contd.)

Debdeep Mukhopadhyay

Assistant Professor
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
INDIA -721302

Objectives

- **Relative order of hardness of Security Criteria for Hash Functions**
- **Iterated Hash Functions:**
  - **The Merkle Damgard Construction**

Comparison of Security Criteria

- **Solving Collision is easier than solving Preimage or 2nd Preimage**
- **Can we reduce one problem to the other?**
- **We shall study two reductions:**
  - **Collision to 2nd Preimage**
  - **Collision to Preimage**

Proof Method

- **Reducing Collision to Preimage:**
  - **Assume that Preimage can be solved using a randomized algorithm**
  - **Show that then the Collision can be solved.**
- **$\text{Collision}_{\text{Hardness}} \ll \text{Preimage}_{\text{Hardness}}$**
- **Resistance against Collision => Preimage Resistance**

The first reduction

Algorithm 4.4: COLLISION-TO-SECOND-PREIMAGE(h)

external ORACLE-2ND-PREIMAGE
choose $x \in \mathcal{X}$ uniformly at random
if ORACLE-2ND-PREIMAGE$(h, x) = x'$
  then return (x, x')
  else return (failure)

- **Oracle-2nd-Preimage is an (ε,q) algorithm.**
- **Since it is a Las-Vegas algorithm, if it gives an answer it will be correct. Thus, x≠x' and h(x)=h(x'). Thus the collision is also found.**
- **Thus Collision-to-second-preimage is also an (ε,q) Las-Vegas algorithm**

The second reduction

Algorithm 4.5: COLLISION-TO-PREIMAGE(h)

external ORACLE-PREIMAGE
choose $x \in \mathcal{X}$ uniformly at random
$y \leftarrow h(x)$
if (ORACLE-PREIMAGE$(h, y) = x'$) **and** $(x' \neq x)$
  then return (x, x')
  else return (failure)

- **Assume that Oracle-Preimage is a (1,Q) Las Vegas algorithm**
- **We will make some weak assumptions on the size of X and Y, |X|≥2|Y|**

Reduction

THEOREM 4.5 *Suppose $h : \mathcal{X} \rightarrow \mathcal{Y}$ is a hash function where $|\mathcal{X}|$ and $|\mathcal{Y}|$ are finite and $|\mathcal{X}| \geq 2|\mathcal{Y}|$. Suppose* ORACLE-PREIMAGE *is a $(1, Q)$-algorithm for* ***Preimage****, for the fixed hash function h. Then* COLLISION-TO-PREIMAGE *is a $(1/2, Q+1)$-algorithm for* ***Collision****, for the fixed hash function h.*

- **Proof discussed in class.**

Point to Ponder

- **If the OraclePreimage has a success probability of ε<1, what is the minimum probability of success of CollisionToPreimage Algorithm?**

Construction of Iterated Hash Functions

- Extending a compression function to a hash function with an infinite domain
- A hash function created in this fashion is called an iterated hash function
- Consider hash functions whose inputs and outputs are bit strings
- $|x|$: length of a bit string x
- $x||y$: concatenation of strings x and y

Outline of the construction

- Given, compress: $\{0,1\}^{m+t} \rightarrow \{0,1\}^m$, $t \geq 1$
- Preprocessing:
  - an input string x, where $|x| \geq m+t+1$
  - output string y, such that $|y| \equiv 0 \pmod t$
  - $y = y_1||y_2||y_3||\ldots||y_r$, where $|y_i| = t$ for $1 \leq i \leq r$

Optional Output Transformation

- **g: $\{0,1\}^m \rightarrow \{0,1\}^l$**
- **Define $h(x)=g(z_r)$, g is a public function**
- **Sometimes, $h(x)=z_r$**

Processing

- **z_0=IV (public value, called Initialization Vector, |IV|=m)**

$$z_1=\text{compress}(z_0 || y_1)$$
$$z_2=\text{compress}(z_1 || y_2)$$
$$\ldots$$
$$\ldots$$
$$z_r=\text{compress}(z_{r-1} || y_r)$$

A typical preprocessing

- **$y = x || pad(x)$**
  - **$pad(x)$ is a padding function**
  - **it generally has the value of $|x|$, padded to the left with additional zeros (so that the sum is a multiple of t)**
- **Note that the preprocessing step has to be injective**
  - **$|y| = rt \geq |x|$**

Merkle Damgård Construction

- **Uses compress:$\{0,1\}^{m+t} \rightarrow \{0,1\}^m$, which is collision resistant to construct a collision resistant hash function, h:$\{0,1\}^* \rightarrow \{0,1\}^m$**
  - **The construction yields a proof for this result.**
- **Typically, we take $|x|=m+t+1$ (may be because we wish to keep the message length more than double that of the hash value)**

The Preprocessing

- $x = x_1 || x_2 || \ldots || x_k$,
  - where $|x_1| = |x_2| = \ldots = |x_{k-1}| = t-1$ and $|x_k| = t-1-d$, where $0 \le d \le t-2$
  - Thus,

$$k = \frac{n+d}{t-1} = \left\lceil \frac{n}{t-1} \right\rceil$$

The Algorithm

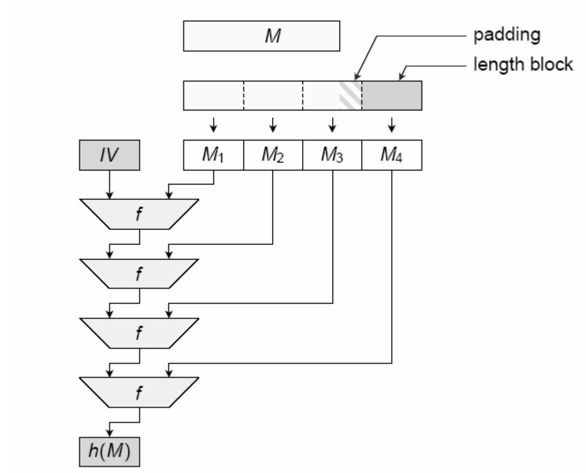
MERKLE-DAMGÅRD(x)

external compress
comment: compress : $\{0,1\}^{m+t} \to \{0,1\}^m$, where $t \ge 2$

$n \leftarrow |x|$
$k \leftarrow \lceil n/(t-1) \rceil$
$d \leftarrow k(t-1) - n$
for $i \leftarrow 1$ **to** $k-1$
  do $y_i \leftarrow x_i$
$y_k \leftarrow x_k \,||\, 0^d$
$y_{k+1} \leftarrow$ the binary representation of d
$z_1 \leftarrow 0^{m+1} \,||\, y_1$
$g_1 \leftarrow$ compress(z_1)
for $i \leftarrow 1$ **to** k
  do $\begin{cases} z_{i+1} \leftarrow g_i \,||\, 1 \,||\, y_{i+1} \\ g_{i+1} \leftarrow \text{compress}(z_{i+1}) \end{cases}$
$h(x) \leftarrow g_{k+1}$
return $(h(x))$

- This step is known as the MD strengthening
- Note that y_{k+1} is also padded to the left with zeros so that $|y_{k+1}| = t-1$
- The MD strengthening helps to make the pre-processing step injective

A Picture is better than thousand words



The Proof

Suppose **compress** : $\{0,1\}^{m+t} \rightarrow \{0,1\}^m$ *is a collision resistant compression function, where* $t \geq 2$. *Then the function*

$$h : \bigcup_{i=m+t+1}^{\infty} \{0,1\}^i \rightarrow \{0,1\}^m,$$

is a collision resistant hash function.

- **$\text{Compress}_{\text{Collision-res}}$ => $\text{Hash}_{\text{Collision-res}}$**
- **not($\text{Hash}_{\text{Collision-res}}$) => not($\text{Compress}_{\text{Collision-res}}$)**
- **If you can find a collision in the Hash function efficiently, then you can find a collision in the compression function efficiently.**

When t=1

MERKLE-DAMGÅRD2(x)

external **compress**
comment: compress : $\{0,1\}^{m+1} \to \{0,1\}^m$

$n \leftarrow |x|$
$y \leftarrow 11 \parallel f(x_1) \parallel f(x_2) \parallel \cdots \parallel f(x_n)$
denote $y = y_1 \parallel y_2 \parallel \cdots \parallel y_k$, where $y_i \in \{0,1\}, 1 \le i \le k$
$g_1 \leftarrow$ **compress**$(0^m \parallel y_1)$
for $i \leftarrow 1$ **to** $k-1$
  do $g_{i+1} \leftarrow$ **compress**$(g_i \parallel y_{i+1})$
return (g_k)

- **Here the encoding, f is done in a special way.**
  - **f(0)=0, f(1)=01**
- **The encoding is injective**
- **There does not exist two strings x≠x', such that y(x)=z||y(x'), that is no encoding is a postfix of another encoding.**

Theorems

Suppose **compress** : $\{0,1\}^{m+1} \to \{0,1\}^m$ *is a collision resistant compression function. Then the function*

$$h : \bigcup_{i=m+2}^{\infty} \{0,1\}^i \to \{0,1\}^m,$$

is a collision resistant hash function.

Suppose **compress** : $\{0,1\}^{m+t} \to \{0,1\}^m$ *is a collision resistant compression function, where* $t \ge 1$. *Then there exists a collision resistant hash function*

$$h : \bigcup_{i=m+t+1}^{\infty} \{0,1\}^i \to \{0,1\}^m.$$

The number of times **compress** *is computed in the evaluation of h is at most*

$$\begin{array}{ll} 1 + \left\lceil \frac{n}{t-1} \right\rceil & \text{if } t \ge 2 \\ 2n+2 & \text{if } t = 1, \end{array}$$

where $|x| = n$.

Assignment

- **Consider a Collision Resistant function g(x): $\{0,1\}^* \rightarrow \{0,1\}^n$**
- **Consider a hash function:**

$$h(x) = \begin{cases} 1 \| x, & \text{if x is of n bits} \\ 0 \| g(x), & \text{otherwise} \end{cases}$$

- **Discuss about the Collision and Preimage resistance of h(x). Explain the anomaly.**

Further Reading

- **Douglas Stinson, *Cryptography Theory and Practice, 2nd Edition*, Chapman & Hall/CRC**
- **Phillip Rogaway and Thomas Shrimpton, Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance, Fast Software Encryption 2004.**

Next Days Topic

- **Cryptographic Hash Functions (contd.)**