Hash Functions — Introduction

Orr Dunkelman

Computer Science Department

7 March, 2012

אוניברסיטת חיפה
University of Haifa

Outline

What?

- This is a seminar about cryptanalytic techniques on hash functions.
- Seminar:
  - I shall give a few introductory lectures,
  - Each one will present one paper in a 45-minute time slot.
- The papers are real life research papers.
- You shall present them to the class.
- Which means: you need to know the material, and you need to pass it on to your peers.

Why?

- Hash functions are a really hot topic.
- There is even a competition for selecting the next generation cryptographic hash functions at the moment.
- New ideas and techniques emerged in the last few years, with applications to widely used hash functions.

Where, When, and Who?

- Location: TBD
- Wed., 16:15-17:45.
- Lecturer:
  - Orr Dunkelman
  - Email: orrd (at-sign) cs (dot) haifa (dot) ac (dot) il
  - Office: Jacobs 408.
  - Phone: 8447

Grades

- 60% — Lecturer's evaluation,
- 20% — Participation in classes (it is mandatory to attend at least 10 meetings),
- 20% — Peers' evaluation.

Perquisites

- Probabilistic Methods (203.2480),
- Computational Models (203.6510)

It is highly recommended to take a look at the slides of the introduction to cryptography course.

What is a Cryptographic Hash Function?

A cryptographic hash function is a function that accepts an input of indefinite length, and outputs a digest of fixed length. securely.

First Introduction to Cryptography

[DH76] There is, however, a modification which eliminates the expansion problem when N is roughly a megabit or more. Let g be a one-way mapping from binary N-space to binary n-space where n is approximately 50. Take the N bit message m and operate on it with g to obtain the n bit vector m'. Then use the previous scheme to send m'...

Digital Signatures

- Digital signatures are method to authenticate the source of a message, and assure its completeness.
- The security requirements are:
  - Only the signer can generate a legitimate signatures.
  - Everybody can verify that the signature is valid.
  - Any adversary, even with access to many signatures, cannot generate a new pair of message and signature.

Digital Signatures using RSA

- The first digital signature algorithm was based on RSA:
  1. The user U chooses two large primes p, q,
  2. Then he computes $n = pq$, and finds two numbers e, d such that $e \cdot d \equiv 1 \bmod \varphi(n)$.
  3. The public key is (n, e) and the private one is (n, d).
  4. To sign a message $0 \leq m \leq n-1$, the user computes $sig = m^d \bmod n$.
  5. To verify a signature sig on a message m, compute $m' = sig^e \bmod n$, and accept if $m' = m$.

Why you should NEVER use RSA for signatures in this way

- The signature on 0 and 1 is 0 and 1, respectively.
- Given two messages m_1, m_2 and their corresponding signatures sig_1, sig_2, you can compute the signature on $m_1 \cdot m_2$ as $sig_1 \cdot sig_2$:

$$(sig_1 \cdot sig_2)^e = sig_1^e \cdot sig_2^e = m_1 \cdot m_2$$

- You can pick a random string sig and compute $m = sig^e \bmod n$ to obtain a valid pair of message and signature.
- And many other reasons ...

The Standard RSA with Hash Functions

- It is possible to solve the previous issues* by signing a hash of the message.
- Namely, to compute a signature *sig*, compute $sig = h(m)^d \bmod n$.
- To verify the signature *sig* on a message *m*, check whether $sig^e \stackrel{?}{\equiv} h(m) \bmod n$.
- What $h(\cdot)$ should satisfy so this will be a secure signature scheme?

What is a Hash Function? (cont.)

- (Cryptographic) Hash Functions are means to **securely** reduce a string m of arbitrarily length into a fixed-length digest.
- The main problem is the definition of securely.
- For signature schemes, twothree basic requirements exist:
  1. Preimage resistance: given $y = h(x)$, it is hard to find x (or x', s.t., $h(x') = y$).
  2. Second preimage resistance: given x, it is hard to find x' s.t. $h(x) = h(x')$.
  3. Collision resistance: it is hard to find x_1, x_2 s.t. $h(x_1) = h(x_2)$.

Where else can you Find Hash Functions?

- Hash functions were quickly adopted in other places:
  - Password files (storing $h(pwd, salt)$ instead of pwd).
  - Bit commitments schemes (commit — $h(b, r)$, reveal — b, r).
  - Key derivation functions (take $k = h(g^{xy} \bmod p)$).
  - MACs (long story).
  - Tags of files (to detect changes).
  - Inside PRNGs.
  - In certificates (in the signatures).
  - Inside protocols (used in many “imaginative” ways).
  - ...

What do we Want out of Our Hash Functions?

As hash functions are widely used, various requirements are needed to ensure the security of construction based on hash functions:

- Collision resistance — signatures, bit commitment (for binding), MACs.
- Second preimage resistance — signatures.
- Preimage resistance — signatures (RSA, or other TD-OWP), password files, bit commitment (for hiding).
- Pseudo Random Functions — key derivation, MACs.
- Pseudo Random Oracle — protocols, PRNGs.

What do we Really Want out of Hash Functions?

We want the hash function to behave in a manner which would prevent any adversary from doing anything malicious to the hash function:

- One-wayness (no inversion).
- No collisions (up to the birthday bound).
- No second preimages.
- Outputs which are nicely distributed.
- ...

Therefore, the ideal hash function attaches for each possible message M a random value as $h(M)$. And voilá — a random oracle.

What about Security?

- Collisions exist. Also second preimages. Also preimages.
- Finding them is possible.
- But should be hard.

which raises the question:

How hard?

Optimal Security of a Hash Function

If $h(\cdot)$ is the ideal hash function (a random oracle):

- Finding a preimage — $O(2^n)$ work (exhaustive search).
- Finding a second preimage — $O(2^n)$ work (exhaustive search).
- Finding a collision — $O(2^{n/2})$ work (birthday attack) [can be done with small memory overhead (Floyd or Nivasch)].

for an n-bit digest size.

The Birthday Paradox

How many people should be in a room, such that two of them share their birthday with probability of at least 50%? (assume no leap years)

- 366 — Ensure that there are two with such a birthday, by the pigeonhole principle.
- 183 — Probability of more than 99.999%.
- 23 — Probability of 50.730%.

Why?

The Birthday Paradox (cont.)

Let's look at the probability p_k that k people had k unique birthdays.

- The probability that the first person has a birthday different from all previous birthdays — 1.
- The probability that the second person has a birthday different from all previous birthdays — $364/365$.
- For the third (assuming the first two have unique birthdays) — $363/365$.
- For the $(\ell + 1)$ person (assuming the first ℓ have unique birthdays) — $(365 - \ell)/365$.

Hence,

$$p_k = \prod_{i=0}^{k-1} \frac{365 - i}{365} = \prod_{i=0}^{k-1} \left(1 - \frac{i}{365}\right)$$

The Birthday Paradox (cont.)

- As $1 - x \leq e^{-x}$:

$$p_k = \prod_{i=0}^{k-1} \left(1 - \frac{i}{365}\right) \leq \prod_{i=0}^{k-1} e^{-i/365} = e^{-k(k-1)/(2 \cdot 365)}.$$

As long as $1/2 > e^{-k(k-1)/(2 \cdot 365)} > p_k$, the probability of a collision is more than $1/2$.

Exercise:

1. Assuming there are n possible birthdays, what should be the number of people such that two have a common birthday with probability $1/2$?
2. With probability p?
3. Assume that the probability of being born in each day of December is twice as for other days. How many people are needed in the room to have a collision?

The Birthday Paradox — A Variant

- Another variant of the birthday paradox: There are two sets of people A and B.
- What should be the sizes of A and B such that there will be a collision in the birthday between one person from A and one from B with probability $1/2$.
- The probability of the first person from A to collide is $|B|/365$.
- The probability of the second person from A is $|B|/365$.
- ...
- So the probability of a collision is

$$1 - \left(1 - \frac{|B|}{365}\right)^{|A|}.$$

Collision Resistance of Hash Functions

Let us try to define the meaning of $h(\cdot)$ being collision resistant.

- It is computationally infeasible to find a collision. Formally: There is no efficient algorithm which given h finds collisions.
- $h(\cdot)$ is a hash function. Therefore, necessarily there exist a, b such that $h(a) = h(b)$. Consider the algorithm:

  print a, b.

What Should We Do?

Collision Resistance of Hash Functions (cont.)

- Practical solution — a and b are unknown. For any specific function finding them takes $O(1)$ anyway. So who cares?
- Theoretical solution (I) — let us define a *family* of hash functions, and bundle the collision resistance of one of them to the collision resistance of the family.
- Theoretical solution (II) — we do not know the value of a, b for a specific hash function. Thus, let us define a protocol Π, which uses a hash function $h(\cdot)$, such that we can show that every adversary A against Π yields an attack on $h(\cdot)$ [R05].

How to Build a Hash Function

?

How to Build a Symmetric-Key Block Cipher-Based Encryption Scheme

1. Design a block cipher. (a primitive that accepts a key of fixed length, and encrypts plaintexts of a fixed length).
2. Find a good mode of operation. (a method to encrypt messages whose length is different than the block size).
3. Combine the two together.

Examples of modes of operation: ECB, CBC, CTR, ...

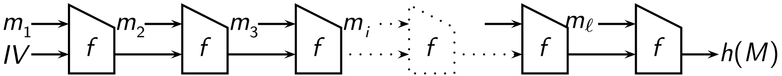
How to Build a Hash Function (part II)

- Design a compression function (a black box that accepts $n + b$ bits and produces n bits).
- Find a good mode of iteration (a way to handle messages of length longer (or shorter) than $n + b$).
- Combine the two.

The Merkle-Damgård Construction

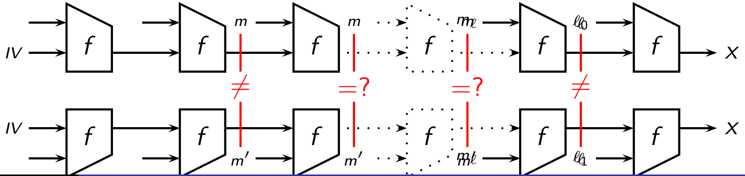
Given a compression function $f : \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$, the Merkle-Damgård hash function H_f is defined as:

1. Pad the message M to a multiple of b (with 1, and as many 0's as needed and the length of the message).
2. Divide the padded message into ℓ blocks $m_1 m_2 \dots m_\ell$.
3. Set $h_0 = IV$.
4. For $i = 1$ to ℓ, compute $h_i = f(h_{i-1}, m_i)$.
5. Output h_ℓ (or some function of it).



Collision Resistance of Merkle-Damgård

- Assume that the compression function is optimal.
- Let assume that there is an adversary A which can find collisions in $MD^f(\cdot)$ efficiently, and we transform it into A' which finds collisions in $f(\cdot)$.
- Examine the collision produced by A. If the messages are not of the same length, then, necessarily there is a pair of inputs $(h, m) \neq (h', m')$ s.t. $f(h, m) = f(h', m')$.
- If the messages are of the same length, start from the last block and go backwards, until you find the block which differs. And voilá — a collision in $f(\cdot)$.



The Security of the Merkle-Damgård Construction

- Finding a collision in H_f means finding a collision in f.
- Thus, if f is collision-resistant, so is H_f.

- Also, finding a second preimage in H_f means finding a collision in f.
- The same is true for finding a preimage (because you can use it to find a second preimage).

To conclude, if f is collision resistant (i.e., it takes $O(2^{n/2})$ invocations to find a collision), then H_f is collision resistant and (second) preimage resistant with security level of $O(2^{n/2})$.

Recall that we target security of $O(2^n)$ for (second) preimage resistance!

A Few Solutions

- Widepipe/ChopMD — throw away some of the bits of the output (e.g., $n/2$ bits, which would result in collision resistance of $O(2^{n/4})$ and (second) preimage resistance of $O(2^{n/2})$).
- Additional inputs — if the round function has some dithering inputs, salts, or counters, one can prove the $O(2^n)$ (second) preimage resistance.
- Sponges — have a hige internal state (with a very light update permutation and a light message injection to the internal state).
- Keyed Hash Functions — if the key is selected at random, one can prove the $O(2^n)$ (second) preimage resistance.

Some Concluding Remarks

- Hash functions are the only cryptographic primitive which is not keyed.
- When the hash function is "keyed", the key is given to the adversary (or even chosen by him).
- In other words — this is a cryptographic primitive with no keys nor secrets.
- Unlike other cryptographic schemes, for which the security definitions are mostly accepted, hash functions have many sets of security definitions.
- During this course, we shall concentrate on the main three ones: collisions, (second) preimages, and preimages.
- However, invalidation of any other security property is sufficient to call the hash function "broken".