Security

Assignment 6, Wednesday, October 14, 2015

Handing in your answers: the full story, see

`http://www.sos.cs.ru.nl/applications/courses/security2015/exercises.html`

Briefly,

- submission via Blackboard (`http://blackboard.ru.nl`);
- one single pdf file;
- make sure to write all names and student numbers and the name of your teaching assistant (Brinda or Joost).

Deadline: Thursday, October 22, 24:00 (midnight) sharp!

Goals: After completing these exercises successfully you should be able to

- correctly identify properties of hash functions;
- reason about applying hash functions in practice.

Marks: You can score a total of 100 points.

1. **(40 points)** A cryptographic hash function $\mathcal{H}$ assigns a bit string to another bit string with the following properties:

   (C) Collision resistance: it is infeasible to determine two bit strings $x \neq x'$ such that $\mathcal{H}(x) = \mathcal{H}(x')$,

   (P) Preimage resistance: given a bit string y output by $\mathcal{H}$, it is infeasible to determine a bit string x such that $\mathcal{H}(x) = y$,

   (P2) Second preimage resistance: given a bit string x it is infeasible to determine a different bit string x' ($x \neq x'$) such that $\mathcal{H}(x) = \mathcal{H}(x')$.

   (F) Fixed length: the length of output is fixed and does not depend on the input size (this is usually not a formal requirement, but it is, in practice, often the case).

   We define possible hash function candidates h. Do they meet each requirement? Explain briefly for all properties (for each hash function).

   (a) The function is defined as $h(x) := 11111011100$.

   (b) The function is defined as $h(x) := x \| x$ (where $\|$ is the concatenation operation, that is, x is written twice after each other).

   (c) Assume that $\mathcal{H}$ is a hash function that meets all four requirements (C), (P), (P2), and (F). The function is defined as $h(x) := \mathcal{H}(|x|)$ (where $|x|$ is the bit length of x).

   (d) Assume again that $\mathcal{H}$ is a hash function that meets all four requirements (C), (P), (P2), and (F). The function is defined as $h(x) := \mathcal{H}(x) \| \mathcal{H}(x)$.

   (e) Assume that $\mathcal{H}$ is a hash function with output bit-length N that meets all four requirements (C), (P), (P2), and (F). The candidate function is defined as

   $$h(x) := \begin{cases} 1^N, & \text{if } x = (0 \ldots 01) \\ \mathcal{H}(x), & \text{otherwise;} \end{cases}$$

   that is, if the input of function h is such a bit-string that only its last bit is 1, it outputs an all-1 bit-string of length N. Otherwise, it outputs the same as $\mathcal{H}$.

2. **(25 points)** In the lecture, the historical claim of the paper on vulnerabilities in the Megamos chip was discussed. In 2013, Roel Verdult published the SHA-512 hash of their banned paper. See slide 15.

(a) Why does this work, *i.e.* what does this hash prove in case the actual paper eventually gets published?

(b) Suppose Alice finds the solution to the famous computer science problem that answers 'P $\stackrel{?}{=}$ NP', but does not want to spoil the fun yet. She decides to publish the hash of either 'P is equal to NP' or 'P is not equal to NP'. Does this give away her discovery? Why?

After a dragging dispute with Volkswagen, Verdult, Garcia and Ege were finally allowed to publish their results. In the end, they did have to remove one detail from the original paper.

(c) What does this mean for the hash value published in 2013? Is it still useful? Briefly explain.

(d) Suppose the paper was hashed and published as a plaintext document, and the detail that had to be removed was a formula of only 16 lower-case alphanumerical (a-z, 0-9) characters. An attacker interested researcher might try to figure out this missing piece by using the hash. Let's assume that the attacker knows the exact position of the missing piece. Suppose he had a computer that could perform 3 billion SHA-512 computations per second. How many years would it take on average (*i.e.* with a 50% chance) to reconstruct the formula?

3. **(35 points)** Hash functions play an important role in the BitTorrent protocol. When using BitTorrent, users receive pieces of a large file from different providers (typically referred to as 'seeders'). When a user wants to obtain a certain file, he needs a '.torrent' file. Amongst other things, this file can contain a list of hashes. These hashes are used to guarantee the integrity of the individual pieces, allowing a user to check each piece when he receives it.

(a) An attacker tries to replace one piece of the original piece with a new one. What property of the hash function is important here so that the attack is detectable?

(b) Say Alice wants to download a file of 2 GB (*e.g.* an authentic Linux distribution), and it is split in pieces of 16 KB each. Suppose that the .torrent file she uses contains a list of MD5 hashes. What is the minimum size of that .torrent file? (*Hint:* How large is one MD5 hash?)

As (b) shows, that can be quite a large file to deal with. We could resolve this by increasing the size of each piece so that we do not need as many hashes. That would however make it more difficult for peers to quickly share their pieces. An alternative approach is to use *binary hash trees*.

Consider Figure 1. On the circles along the bottom (the *leaf nodes*), we place the hash values of the different pieces: $N_0^0 = h(\text{piece}_0), N_1^0 = h(\text{piece}_1), N_2^0 = h(\text{piece}_2), \ldots$. Each of the higher nodes in the tree contains the combined hash of its children. For example, the value in the first node of the next layer would be $N_0^1 = h(N_0^0 \| N_1^0) = h(h(\text{piece}_0) \| h(\text{piece}_1))$. The node above that would be $N_0^2 = h(N_0^1 \| N_1^1)$, *etc.* We continue doing this all the way to the top of the tree. The top of the tree is called the root R, which is always contained in the .torrent file.

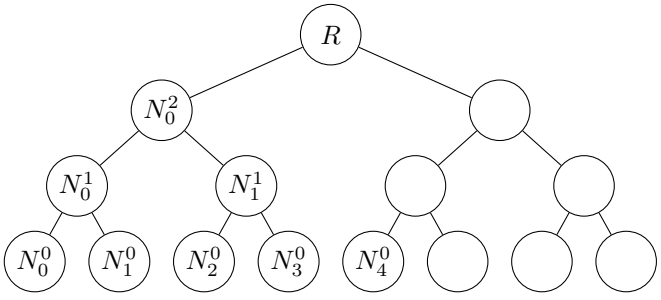

Figure 1: binary hash tree

(c) Suppose the .torrent file does not contain a list of all hashes, but only the root hash. At which stage would you be able to check the integrity of the pieces you receive from others? Why?

Instead of just sending you the piece, a seeder should now also send you a few of the nodes in the tree. This guarantees that you have enough information to calculate the root node, using his piece and the hashes he sends along. You can then compare it to the root node in the '.torrent' file. If it matches, the piece was authentic!

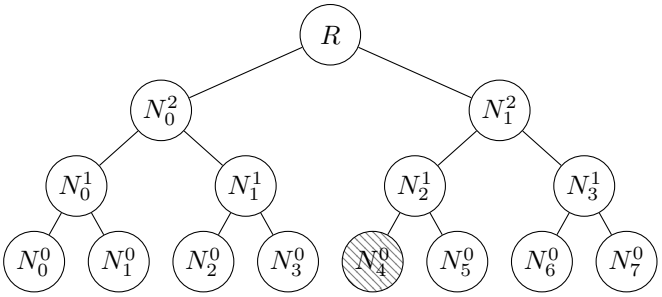

Figure 2: binary hash tree with a marked leaf node

(d) Suppose someone sends you piece_4. Now you can compute $N_4^0 = h(\text{piece}_4)$, marked gray in Figure 2. Looking at the figure, what other N-values (nodes) would he need to send you before you can compute R, to check if the piece was correct?

(e) Suppose we have a file that consists of 1024 pieces. How many hashes would need to be sent along with each of the pieces?

This exercise was only a very slight abstraction from reality. Have a look at `http://bittorrent.org/beps/bep_0030.html` if you are interested in what this looks like in practice.