Security

Assignment 6, Friday, October 14, 2016

Handing in your answers: For the full story, see

`http://www.sos.cs.ru.nl/applications/courses/security2016/exercises.html`

To summarize:

- Include your name and student number **in** the document (they will be printed!), as well as the name of your teaching assistant (Hans or Joost). When working together, include **both** your names and student numbers.
- Submit one single **pdf** file – when working together, only hand in **once**.
- Hand in via Blackboard, before the deadline.

Deadline: Monday, October 24, 09:00 sharp!

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

- correctly identify properties of hash functions;
- apply hash functions in practical settings;
- reason about the use of salted hashes.

Marks: You can score a total of 100 points.

In these exercises we consider the following properties for hash functions $\mathcal{H}$:

(C) Collision resistance: it is infeasible to find 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 find a bit string x such that $\mathcal{H}(x) = y$,

(P2) Second preimage resistance: given a bit string x it is infeasible to find 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).

1. **(35 points)** We define possible hash function candidates h. Do they meet each of the properties (C), (P), (P2) and (F)? 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 concatenation, that is, x is repeated twice).
   (c) Assume that $\mathcal{H}$ is a hash function that meets all four requirements (C), (P), (P2), and (F). The function is defined as $h(x) := \mathcal{H}(x) \| \mathcal{H}(x)$.
   (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 $h(x) := \mathcal{H}(|x|)$ (where $|x|$ is the bit length of 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. **(35 points)** Alice and Bob play a game where they need to answer a few multiple-choice questions with either I, II, III or IV. Both of them get the same list of questions by e-mail from a server. Since the answers have to be checked, they decide to e-mail their answers to each other. However, this poses a problem; who is going to send the answers first? Seeing Alice's answers, Bob might change his answers after receiving hers. Similarly, the same problem arises if they swap the order.

   Alice and Bob decide to use a secure hash function h that has the properties (C), (P) and (P2) described above. Their intention is to hide their actual answer and, at the same time, commit to an answer such that they cannot change it afterwards.

   Let $a_i \in \{\text{I}, \text{II}, \text{III}, \text{IV}\}$ be the answers of Alice and $b_i \in \{\text{I}, \text{II}, \text{III}, \text{IV}\}$ be the answers of Bob for question i. Their scheme runs as follows:

1.	$A \longrightarrow B$	:	$h(a_i) = x_i$	*phase 1: commitment*
2.	$B \longrightarrow A$	:	$h(b_i) = y_i$	
3.	$B \longrightarrow A$	:	b_i	*phase 2: revealing and verification*
4.	A	:	**verify** $h(b_i) = y_i$	
5.	$A \longrightarrow B$	:	a_i	
6.	B	:	**verify** $h(a_i) = x_i$	
7.	...	:	continue at step 1 with the next question $(i + 1)$	

   When they successfully go through all steps they both think that cheating is impossible.

   (a) Does this scheme prevent cheating? If yes, explain why. If no, give an attack.

   (b) In another game the questions are no longer multiple choice. Now Alice and Bob have to send their own essays as answers instead of just I, II, III or IV. They still want to use the same scheme. Does this scheme, in this new setting, prevent cheating (yes/no)? If yes, explain why. If no, give an attack.

   (c) Can we switch steps 3 and 4 with 5 and 6 (i.e. $1, 2, 5, 6, 3, 4$), or would this break the scheme? Consider the effect for both multiple-choice and the essay questions.

   (d) Modify the scheme to make it suitable for multiple-choice. Describe the new scheme using arrow notation.

   (e) For each property (P), (P2) and (C), explain why it is important in this scenario.

3. **(30 points)** Lately there have been a number of high-profile database leaks, with millions of accounts hitting the internet. Most prominently, this summer saw leaks resulting from attacks on Last.fm and Dropbox[1] that took place back in 2012.

   (a) Last.fm simply stored MD5 hashes of passwords. Suppose you can compute 3 billion hashes per second, and you know that someone's password only contains digits and letters (both lowercase and uppercase). How long would it take to break a fully random 10-character password?

   (b) Suppose you're not interested in one specific account, but you just want access to any account. Knowing that the leak included roughly 43 million entries, how long would it take on average before you find a match? Assume that comparing two hashes is free.

   (c) Luckily, Dropbox had 'salted' their passwords before hashing them, preventing such a multi-target attack. On top of that, they used the bcrypt hash function, which is designed to be **slow**. Assume this allows you to do only 10 hashes per second. How long would you expect it to take to break a 6-character alphanumeric[2] password?

.... **The assignment continues on the next page!** ....

[1]For more background information on password leaks, see Troy Hunt's blog (e.g. `https://www.troyhunt.com/the-dropbox-hack-is-real/`, and his database of leaks at `https://haveibeenpwned.com/`).

[2]So, as before: only containing a-z, A-Z and 0-9.

For simplicity, we will briefly ignore salts and multi-target attacks for the remainder of this exercise.

In a typical (web) application, users are asked to submit their password when logging in. This is then hashed and compared to a stored value $\mathcal{H}(password)$. The plaintext password itself is never stored.

(d) Dropbox was transitioning from SHA1 to bcrypt hashes: roughly half the passwords were still stored as SHA1 hashes. In order to upgrade from a SHA1 hash to a bcrypt hash, the user had to log in at least once. Explain why this is necessary: which property of SHA1 (i.e. P, P2 or C) prevents Dropbox from simply updating the hashes themselves, without the user logging in?

(e) The reason why Dropbox wanted to upgrade to bcrypt is because it is much more time-consuming to brute force bcrypt hashes, making it less of a problem if they ever leak. Assume that some users never log in, but Dropbox does want to protect all passwords using bcrypt. How could they have done this, assuming they only have the SHA1 hash of the passwords? Explain what they would need to compute, and mention what they need to check when a user logs in.