Security

Assignment 7, Monday, October 19, 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, November 12 (i.e. after the exam weeks), 24:00 (midnight) sharp!

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

- analyse the use of a hash function in a protocol;
- identify vulnerabilities of hash functions and implement attacks.

Marks: You can score a total of 100 points.

1. **(30 points)** The following authentication protocol makes use of the Lamport hash. The Lamport hash basically is repeatedly hashing an input value. We write $h^n(x)$ to denote hashing x n times, *e.g.* $h^3(x) = h(h(h(x)))$. Each user chooses a password pw and hashes this n times and sends it to the server. Let us assume that in a system $n = 10\,000$ initially. The server then stores a tuple (user, n, $Y = h^n(pw)$) for each user in a database. Users can now authenticate to the server using the following protocol:

$$\begin{array}{llcl} 1. & A \longrightarrow S & : & A \\ 2. & S \longrightarrow A & : & n \\ 3. & A \longrightarrow S & : & X = h^{n-1}(pw) \end{array}$$

   The server checks if $h(X) = Y$, then decrements n and sets $Y := X$. So, after a successful run of this protocol the server holds a new tuple (user, $n-1$, $Y = h^{n-1}(pw)$).

   (a) Can you think of an attack (here, relay or simple man-in-the-middle is not considered an attack) after which the adversary can gain access multiple times without the user being present? Use the arrow notation ($A \longrightarrow B$: message) in your explanation.

   (b) At some point $n = 0$. Is it safe to start over again and put n back to $10\,000$? Briefly explain your answer.

   (c) One way to construct a one-time pad that uses a Lamport hash is by starting from the hash of a random starting value r and generating an infinite sequence of $h(r), h(h(r)), \ldots$. Is this way to construct a one-time pad secure? If yes, briefly motivate your answer, otherwise describe an attack.

   (d) Another way to construct a one-time pad using a Lamport hash is to 'reverse' the above approach, to obtain $h^{1000}(r), h^{999}(r), h^{998}(r), \ldots, h(r)$. (The length of the pad in this case is 1000 times the output length of the hash function, after which a new random value is chosen for a new pad.) What is the security problem with this construction?

2. **(25 points)** Hashes are often used as fingerprints on files. Digital signatures (later in the lecture) typically sign the hash of a document. A hash collision allows an attacker to produce two different files with the same hash – a digital signature on (the hash of) one of these files is also valid for the other.

   It is often believed that this is not very much of a problem because two colliding messages (messages with the same hash) are extremely unlikely to be any meaningful or useful messages. However, an attacker can use an *arbitrary* collision to produce two files with content *of his choice* that have the same hash value. Details and examples for this attack can be found on `http://th.informatik.uni-mannheim.de/People/lucks/HashCollisions/`.

(a) Produce two postscript files that have the same MD5 hash value, one file displaying the text "The 2016 US President is a woman. [S-number(s)]", the other one displaying the text "The 2016 US President is a man. [S-number(s)]". (Replace [S-number(s)] with your student number(s).)

Upload these two files with the respective filenames `woman.ps` and `man.ps` to Blackboard, along with your solutions.

Hint: Start with the postscript files from the above website. Modify these files in an editor that does not destroy the rest of the file (such as Notepad++, vim..). NOTE: Some editors might add other characters (e.g. white space), thus making the hash collision impossible. So try with different editors until it works ;)

(b) Determine for each of the following hash constructions whether it is susceptible to the same MD5 weakness that was used here? (**Hint:** Read carefully the text on the given webpage and use Wikipedia.)

- SHA-1
- SHA-2
- SHA-3

3. **(45 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 *pre-image resistance*, *2nd pre-image resistance* and *collision resistance*. 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:

$$\begin{array}{rrcl}
1. & A \longrightarrow B & : & h(a_i) = x_i \\
2. & B \longrightarrow A & : & h(b_i) = y_i \\
3. & B \longrightarrow A & : & b_i \\
4. & A & : & \textbf{verify } h(b_i) = y_i \\
5. & A \longrightarrow B & : & a_i \\
6. & B & : & \textbf{verify } h(a_i) = x_i \\
7. & \dots & : & \text{continue at step 1 with the next question } (i+1)
\end{array}$$

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

(a) Does this scheme prevent cheating (yes/no)? 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) This scheme consists of two phases. First Alice and Bob commit to an answer in steps 1 and 2 and in steps 3 to 6 they reveal and verify them. Is the order of the steps *within* these phases important to prevent cheating (by Alice or Bob)? For example, can we just switch the combination of steps 3 and 4 with the combination of steps 5 and 6, or would this break the scheme? Consider this for both the multiple choice game as well as the one with open questions. Briefly explain your answer.

(d) Which of the standard hash function properties – that is *pre-image resistance*, *2nd pre-image resistance* and *collision resistance* – are important in this scheme? Explain you answer.

(e) Assume that Alice and Bob play a coin-flipping game. Alice wins if they flip the same side, Bob wins if the sides are different. Alice's flip is denoted by a, Bob's by b. Explain why the original scheme (assuming only one round there) is not applicable if Alice and Bob want to play this game via e-mails using the original scheme, that is,

$$\begin{array}{rrcl}
1. & A \longrightarrow B & : & h(a) = x \\
2. & B \longrightarrow A & : & h(b) = y \\
3. & B \longrightarrow A & : & b \\
4. & A & : & \textbf{verify } h(b) = y \\
5. & A \longrightarrow B & : & a \\
6. & B & : & \textbf{verify } h(a) = x
\end{array}$$

(f) Modify the scheme to make it suitable for the coin-flipping game. Describe the new scheme and make it explicit which hash function properties are relevant in this case.