Applied Cryptography

Symmetric Cryptography, Assignment 3, Monday, March 4, 2024

Remarks:

- Hand in your answers through Brightspace.
- Hand in format: PDF. Either hand-written and scanned in PDF, or typeset and converted to PDF. Please, **do not** submit photos, Word files, LaTeX source files, or similar. Also submit code used for your assignments (as separate files).
- Assure that the name of **each** group member is **in** the document (not just in the file name).

Deadline: Sunday, March 17, 23.59

Goals: After completing these exercises you should have understanding in arguing security of hash functions, and in the usage of sponge and duplex functions.

1. **(10 points)** On 23 February 2017, researchers from CWI Amsterdam and Google Researchers broke SHA-1 in practice. In this exercise, you will generate your own SHA-1 collision. There is no need to dive into the technical details of SHA-1, the only important thing to know is that SHA-1 is a Merkle-Damgård design (see slide 10). In more detail, SHA-1 operates on a state of 160 bits, and compresses message blocks of 512 bits at a time.

   The attackers of SHA-1 derived two different messages M and M' of size 1024 bits that, if preceded by a common prefix S of size 1536 bits, resulted in SHA-1$(S\|M) =$ SHA-1$(S\|M')$. The messages S, M, M' are given at `https://www.cs.ru.nl/~bmennink/sha1attack/`, alongside a SHA-1 digest computation tool.

   (a) Compute SHA-1$(S\|M)$ and SHA-1$(S\|M')$.

   (b) Consider the following message M'':

   ```
   8B AD F0 0D 8B AD F0 0D 8B AD F0 0D 8B AD F0 0D
   8B AD F0 0D 8B AD F0 0D 8B AD F0 0D 8B AD F0 0D
   8B AD F0 0D 8B AD F0 0D 8B AD F0 0D 8B AD F0 0D
   8B AD F0 0D 8B AD F0 0D 8B AD F0 0D 8B AD F0 0D
   ```

   Compute SHA-1$(S\|M\|M'')$ and SHA-1$(S\|M'\|M'')$.

   (c) Consider a 1024-bit message block M_a defined as:

   ```
   C0 FF EE C0 FF EE C0 FF EE C0 FF EE C0 FF EE C0
   FF EE C0 FF EE C0 FF EE C0 FF EE C0 FF EE C0 FF
   EE C0 FF EE C0 FF EE C0 FF EE C0 FF EE C0 FF EE
   C0 FF EE C0 FF EE C0 FF EE C0 FF EE C0 FF EE C0
   FF EE C0 FF EE C0 FF EE C0 FF EE C0 FF EE C0 FF
   EE C0 FF EE C0 FF EE C0 FF EE C0 FF EE C0 FF EE
   C0 FF EE C0 FF EE C0 FF EE C0 FF EE C0 FF EE C0
   FF EE C0 FF EE C0 FF EE C0 FF EE C0 FF EE C0 FF
   ```

   Find a prefix P for M_a and a second message M_b such that SHA-1$(P\|M_a) =$ SHA-1(M_b), where $P\|M_a$ and M_b are distinct and of equal size. Hint: use S, M, M'.

   (d) Compute the corresponding SHA-1 hash digest for your message of question 1c.

2. **(10 points)** In this exercise you build your own sponge-based hash function, assuming you already have a permutation p on $b = 400$.

Using the KECCAK-f permutation that you can find in `permutation.py`, build the full KECCAK hash function. Here, you can assume a 10-padding, which appends the message with a single 1 and a sufficient number of 0s so that the padded message is of length a multiple of r bits. (Note that the KECCAK-f permutation operates on bytes, but still the padding is in bits.) Make sure you can control the capacity c and the rate r (with $c+r=b$), as you will use that feature in exercise 3.

You can verify your implementation against the server using

```
nc appliedcrypto.cs.ru.nl 4143
```

(Note that this is only possible from within the Radboud network, or via a VPN connection.)

3. **(20 points)** In this exercise, we look at some of the indifferentiability-based bounds for sponge constructions that were stated in the lecture slides, and show that these bounds are tight: is it possible to "break" these constructions in this number of queries? Use your implementation from exercise 2 for exercises 3b and 3d with parameters $b = 400$, $c = 40$, $r = 360$, and $n = 720$.

   (a) Sketch the generic approach to find a collision for sponge-based hash functions, and argue what the computational complexity is.

   (b) Use your implementation from exercise 2 to find an actual collision, i.e., provide two messages M, M' with $M \neq M'$ with the same hash $h(M) = h(M')$.

   (c) For the general case, e.g., if c, b, r and n are variables, what is actually the proven minimal complexity to find a collision against a sponge construction? Relate this bound to your attack.

   (d) Use your implementation from exercise 2 to find a second preimage for your name, i.e., if the encoding of your name (run 'python encoding.py your_name') is the message M, find another $M' \neq M$ such that $h(M) = h(M')$.

   (e) For the general case, e.g., if c, b, r and n are variables, what is actually the proven minimal complexity to find a second preimage against a sponge construction? Relate this bound to your attack.

4. **(10 points)** Consider SpongeWrap from lecture 5 slide 10. This construction is only a secure authenticated encryption scheme if the distinguisher cannot repeat nonces for encryption queries (it may repeat nonces for decryption queries, though).

   (a) Suppose above condition is violated and the distinguisher can repeat nonces for encryption queries. Describe an attacker that breaks the confidentiality of SpongeWrap in a constant number of queries.

   (b) Suppose we *omit* the domain separating bits 1/0 from SpongeWrap, and associated data and message are both padded with simple 10*-padding (i.e., the last, incomplete, block of both A and M is padded with a 1 and a sufficient number of 0s to get an r-bit block). Describe an attacker that breaks the authenticity of SpongeWrap in a constant number of queries. (Note: the distinguisher is *not* allowed to repeat nonces for encryption queries.)