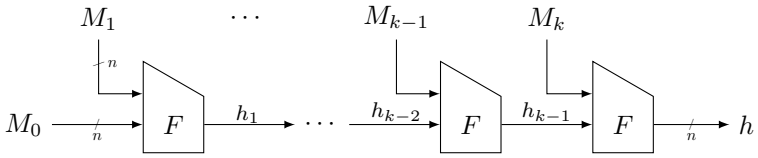
Applied Cryptography

Symmetric Cryptography, Assignment 2, Wednesday, March 2, 2022

Exercises with answers and grading.

1. **(10 points)** In lecture 4 slides 11 (and also at Introduction to Cryptography), we have learned that the Merkle-Damgård hash function is collision resistant if the underlying compression function F is collision resistant. However, for this to hold, we *require* the initial state value to be a constant IV or the message length $\text{len}(M)$ to be encoded. Suppose that we would not do so, i.e., fill the initial state with message data and omitting the length encoding:



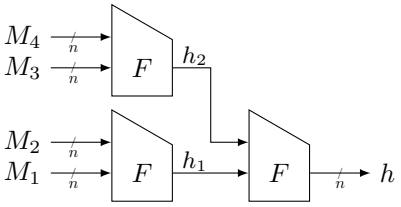
Mount a generic collision attack against this mode.

Begin Secret Info:
The attack is quite simple:

- *Pick arbitrary* $M_1 \| M_2 \| M_3$*;*
- *Compute the hash* $h = \text{MD}(M_1 \| M_2 \| M_3) = h$*;*
- *Compute the compression function call* $h_1 = F(M_1 \| M_2)$*;*
- *Compute the hash* $h = \text{MD}(h_1 \| M_3) = h'$*.*

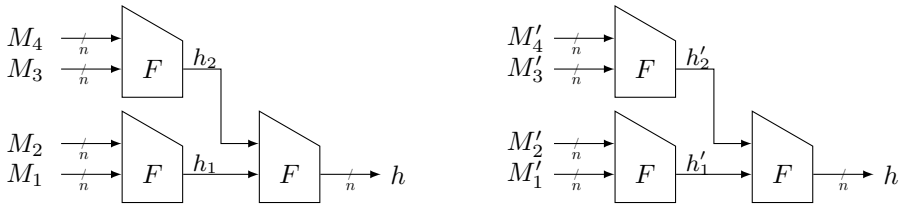
By design, $h = h'$*.*
End Secret Info

2. **(10 points)** Merkle-Damgård is a *sequential* hashing mode, an alternative is tree-based hashing. This exercise is about proving collision resistance of a *simplified* tree-based hashing mode. Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a compression function, and consider the following hash function $\mathcal{H} : \{0,1\}^{4n} \to \{0,1\}^n$:



Argue that $\mathcal{H}$ is collision resistant if F is collision resistant. Note that in the lecture we did not state a formal security model (for technical reasons); therefore, you do not have to derive a formal security bound in this exercise.

Begin Secret Info:
Suppose we find two different messages (M_1, M_2, M_3, M_4) *and* (M'_1, M'_2, M'_3, M'_4) *that yield the same hash:*



- *Clearly, if $h_1 \neq h'_1$ or $h_2 \neq h'_2$, then $F(h_1, h_2) = h = F(h'_1, h'_2)$ forms a non-trivial collision for F.*
- *Otherwise, if both $h_1 = h'_1$ and $h_2 = h'_2$, we distinguish between two cases:*
  - *$(M_1, M_2) \neq (M'_1, M'_2)$: then $F(M_1, M_2) = h_1 = F(M'_1, M'_2)$ forms a non-trivial collision for F.*
  - *$(M_3, M_4) \neq (M'_3, M'_4)$: then $F(M_3, M_4) = h_2 = F(M'_3, M'_4)$ forms a non-trivial collision for F.*

End Secret Info .....

3. **(0 points, as this question already appeared in Introduction to Cryptography)** The sponge construction is proven to be indifferentiable from random with bound around $N^2/2^{c+1}$. The bound is, in fact, tight: it is possible to "break" the construction by making $N \approx 2^{c/2}$ evaluations of the permutation. Describe an attack that obtains a collision for the sponge construction with approximately $N \approx 2^{c/2}$ evaluations of the permutation. For simplicity, restrict your focus to data that is of length 3 blocks, where padding appends an encoding of the length into the last block:

$$\mathsf{pad}(M) = M \| 0^* \| \langle \mathsf{length}(M) \rangle_r \,.$$

This means that you can exactly consider the sponge construction as depicted on lecture 4 slide 25.

Remarks: the precise number of permutation calls in your attack may differ by a constant factor (this is not a problem), and you do not need to compute the exact success probability (intuition suffices).

Begin Secret Info: .....
Focus on data of padded length four blocks, and denote the state AFTER absorption of the third block but before the evaluation of P by $S_r \| S_c$. The attack can operate as follows:

- *Find two messages M and M' of equal length such that*
  - *$\mathsf{pad}(M) = (M_1, M_2, M_3, M_4)$ and $\mathsf{pad}(M') = (M'_1, M'_2, M'_3, M'_4)$ (with necessarily $M_4 = M'_4$),*
  - *$M_1 \| M_2 \neq M'_1 \| M'_2$,*
  - *$S_c = S'_c$.*

  This step takes approximately $N \approx 2^{c/2}$ work.
- *Define $M'' = M' \oplus 0^r \| 0^r \| (S_r \oplus S'_r)$. Then, $\mathsf{sponge}(M)$ and $\mathsf{sponge}(M'')$ give colliding outputs.*

End Secret Info .....

4. **(10 points)** The sponge construction is proven to be indifferentiable from random with bound around $N^2/2^{c+1}$. The bound is, in fact, tight: it is possible to "break" the construction by making $N \approx 2^{c/2}$ evaluations of the permutation. However, it is not immediate to use such "break" to obtain second preimages for the construction.

(a) Consider a sponge construction with $b = 320$, $c = 256$, $r = 64$, and $n = 256$. This particularly means that digests are generated through 4 squeezes. What is the average number of primitive evaluations N you need to make in order to find a second preimage? Explain your answer.

(b) For the setting of question (a), suppose you are given a message M and its image $h \in \{0,1\}^n$. Describe informally an attack that obtains a second preimage for h with approximately N evaluations of the permutation, where N is your answer to question (a).

(c) Consider a sponge construction with $b = 300$, $c = 256$, $r = 44$, and $n = 88$. This particularly means that digests are generated through 2 squeezes. What is the average number of primitive evaluations N you need to make in order to find a second preimage? Explain your answer.

(d) For the setting of question (c), suppose you are given a message M and its image $h \in \{0,1\}^n$. Describe informally an attack that obtains a second preimage for h with approximately N evaluations of the permutation, where N is your answer to question (c).

Begin Secret Info:

By lecture 4 slide 26, the sponge construction is second preimage resistant up to approximately $N^2/2^{c+1} + N/2^n$. *This means that a second preimage can be found in* $\min\{2^{c/2}, 2^n\}$ *evaluations.*

- *By above observation, the required number of primitive evaluations is* $\min\{2^{256/2}, 2^{256}\} = 2^{128}$.
- *Given the first preimage* M, *evaluate* $\mathsf{sponge}(M)$. *This gives* h, *obviously, but all we are interested in, is the state BEFORE the last permutation call BEFORE squeezing,* Z. *From this state, vary* M_3 *and make an inverse permutation call,* $Y = P^{-1}(Z \oplus (M_3 \| 0^c))$. *Similarly, from the initial state, vary* M_1 *and make a forward permutation call* $X = P(M_1 \| 0^c)$. *After approximately* $2^{c/2}$ *attempts for each, you found* (M_1, M_2) *such that* $X \oplus Y = M_2 \| 0^c$ *for some* M_2 *(i.e., you have an inner collision). The second preimage is* $M_1 \| M_2 \| M_3$.
- *By above observation, the required number of primitive evaluations is* $\min\{2^{256/2}, 2^{88}\} = 2^{88}$.
- *This is in fact just a generic preimage attack for* h: *vary* M' *and compute* $\mathsf{sponge}(M') = h'$. *After approximately* 2^n *attempts, you found an* M' *such that* $h' = h$.

End Secret Info

5. **(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.)

Begin Secret Info:

(a) *The attacker can make two SpongeWrap encryption queries with identical N and A, and differing first message blocks $M_1 \neq M_1'$:*

- $\mathit{SpongeWrap}(N, A, M_1 \| M_{rest}, r) = (C_1 \| C_{rest}, T)$,
- $\mathit{SpongeWrap}(N, A, M_1' \| M_{rest}', r) = (C_1' \| C_{rest}', T)$.

The resulting ciphertexts satisfy $C_1 \oplus M_1 = C_1' \oplus M_1'$. This is unlikely to hold for a random oracle \$.

Note: the attacker can also simply ask for two different tag lengths for the same message. In literature, this would not be considered a valid attack, but here, I will.

(b) *The attacker can make a single SpongeWrap encryption query, move the first block of M to the last block of A, and obtain a valid forgery. Below, we already consider data to be padded, and associated data and message are explicitly described in terms of their blocks:*

- $\mathit{SpongeWrap}(N, A, M_1 \| M_2, r) = (C_1 \| C_2, T)$,
- $\mathit{SpongeWrap}^{-1}(N, A \| M_1, C_2, T) = M_2$.

End Secret Info

6. **(10 points)** This question is about the keyed sponge/duplex in a leaky setting. In this setting, we assume that every evaluation of P on secret state leaks λ bits of information about this state. This happens in a non-adaptive setting. I.e., if S is a secret state, and the attacker can eavesdrop an evaluation $P(S \oplus (M \| 0^c))$, λ bits of the input $S \oplus (M \| 0^c)$ leak to the adversary. If it can also eavesdrop an evaluation $P(S \oplus (M' \| 0^c))$, for a different M', λ different bits (in the worst case) of the input $S \oplus (M' \| 0^c)$ are leaked. If a permutation evaluation is repeated, this yields *no new* information. (The adversary would typically be assumed to have set up its eavesdropping attack so as to maximize the leakage, so the smartest possible attacker would, in above example, have learned 2λ bits of S.)

Consider SpongeWrap from lecture 5 slide 10. For the sake of simplicity, we assume that $b = 3k$, $c = 2k$, and thus, $r = k$. K is a k-bit key, N a k-bit nonce, and $\mathsf{init}(K, N) = K \| N \| 0^k$.

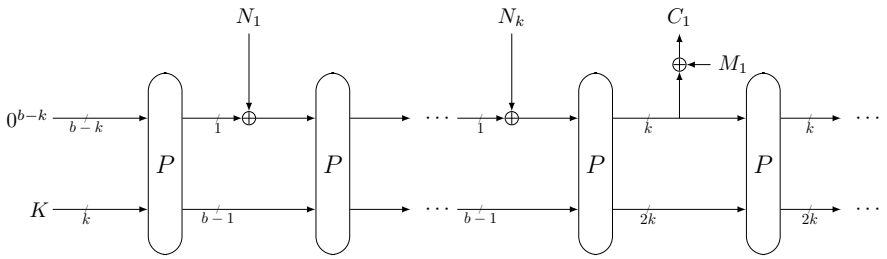
(a) Describe informally an attack that obtains the secret k-bit key, assuming λ bits leakage per primitive evaluation. How many construction queries does your attack make (in terms of k and λ)?

We reconsider SpongeWrap by *not* gluing together key and nonce, but *instead* inserting the nonce as A_1 (and move up associated data by one permutation call).

(b) Suppose the adversary can make 2 authenticated encryption evaluations for different nonces. How many bits of leakage does it learn of the key K? (Hint: it is important to remember that if a permutation is evaluated twice for the same input value, it does *not* leak 2λ bits but rather only λ bits.)

(c) It is nevertheless possible to recover the key by making $\lceil b/\lambda \rceil$ authenticated encryption evaluations. Describe informally an attack that obtains the secret key K with approximately this many number of construction evaluations.

(d) Consider the following stream encryption mode based on the sponge. As before, we have a k-bit key K and k-bit nonce N. This time, the nonce is cut into k bits $N_1 \| \cdots \| N_k$, that are processed one by one. Keystream generation happens with r-bit blocks at a time.



Explain informally why leakage is no problem here (neither at the absorbing phase nor at the squeezing phase).

Begin Secret Info:.....

(a) *For each nonce, λ bits of information about K are leaked. After $\lceil k/\lambda \rceil$ nonces, the entropy of the key reduced to* 0 *and we can consider it lost. The attack is basically already described: Evaluate the scheme for $\lceil k/\lambda \rceil$ different nonces, each evaluation (in the worst case) leaking λ bits of information about the key. That's it.*

(b) *Focusing on the first two permutation calls per construction evaluation, there are three* distinct *calls: $S = P(K\|0^{2k})$, $S' = P(S \oplus N\|0^c))$, and $S'' = P(S \oplus (N'\|0^c))$. Thus, only λ bits of information about the key are leaked.*

(c) *In previous answer, it is obvious that, while only λ bits of information about the key are leaked, 2λ bits of information are leaked about S. After making $\lceil b/\lambda \rceil$ authenticated encryption evaluations, the attacker can learn S. From this, it can compute $P^{-1}(S)$ offline, and recover K.*

(d) *As the key K is never evaluated twice with different side-inputs, at most $\lambda \ll k$ bits are leaked. For any state S in the nonce absorption phase, the state is evaluated at most* 2 *times so leaks at most $2\lambda \ll b$ bits. As nonces are all different, w.h.p., the states before stream generation are always different and never collide, so all leak at most $\lambda \ll b$ bits.*

End Secret Info.....