Applied Cryptography: Assignment 2

Group number 16

Elwin Tamminga s1013846

Lucas van der Laan s1047485

1

We can mount a collision attack by doing to following steps:

1. Choose any message M with a length being a multiple of the length of the output of the hash function, and query this to the hash function resulting in h.
2. Choose any message M' of any length and append it to M and to h, resulting in two new messages $M||M'$ and $h||M'$.
3. We now have two different messages ($M||M'$ and $h||M'$) that will result in the same hash h' when queried to the hash function ($h(M||M') = h(h||M')$).

2

If F is collision resistant, then both $F(M_1, M_2)$ and $F(M_3, M_4)$ will always result in a unique hash value h_1 and h_2. Thus $F(h_1, h_2)$ will also result in a unique hash value h, as the input values of h_1 and h_2 are unique and F is collision resistant. This leads to the conclusion that $\mathcal{H}$ is indeed collision resistant.

4

(a) The security strength for the (2nd) preimage resistance of the sponge construction is $\min(c/2, n)$. In this case, the security strength is $\min(256/2, 256) = 128$ bits. This means that the average number of primitive evaluations you need to make in order to find a 2nd preimage is $N = 2^{128}$.

(b) Given M and $h = h(M)$:

(1) Let $M' = M$, then modify the first r bits such that $M' \neq M$.

(2) Then evaluate the hash function and check if the output $h' = h(M')$ is $h' = h$.

(3) If not, then modify the next r bits in M' and check again. This will test each permutation for a collision.

(4) After modifying all bits in M', try again starting with different r bits. In the end it should take $N = 2^{128}$ primitive evaluations until a hash is found such that $h(M') = h(M)$ with $M' \neq M$.

(c) Just like in (a), the security strength for the (2nd) preimage resistance of the sponge construction is $\min(c/2, n)$. In this case, the security strength is $\min(256/2, 88) = 88$ bits. This means that the average number of primitive evaluations you need to make in order to find a 2nd preimage is $N = 2^{88}$.

(d) Given M and $h = h(M)$:

(1) Choose a message M' of length 88 bits.

(2) Evaluate all possible M' with $M' \neq M$ until $h(M') = h(M)$.

(3) On average, this should take 2^{88} primitive evaluations.

5

(a) If the nonce is repeated and the same key is used, we can do a known plaintext attack to break the confidentiality of SpongeWrap.

(1) We first do 2 online queries using the key K, associated data A, and nonce N with 2 separate messages M and M' which results in ciphertexts C and C'.

(2) We can then XOR the ciphertext C' with M' to get the keystream.

(3) We can now XOR the keystream with the ciphertext C, which will always result in the message M. This would not be the case if we were talking to a Random Oracle, and thus breaking the confidentiality of SpongeWrap.

(b) When the frame bits are replaced by 10^*-padding, the tag creation block and the ciphertext creation block are almost the same, with the only difference that the output is XOR-ed with the corresponding message block part. We can break the authenticity of a SpongeWrap of rate r by doing the following:

(1) First we do an online query with a known message M and associated data A to generate a ciphertext C and tag T.

(2) We then truncate C by r bits to create C'.

(3) We can then XOR the last r bits of C with the last r bits of M to create T'.

(4) We now do an online decryption query with C' and A and get back M' and T', which means that we successfully created a forgery.

6

(a) Every construction query will leak bits of the key in the first primitive evaluation P. So for every construction query with a difference nonce it will leak λ bits of $(K||N||0^k)$. This means it will only leak $\lambda/3$ bits of K. So in order to obtain the secret k-bit key, we need to make $3 \cdot k/\lambda$ queries.

(b) In this case it will leak λ bits of $(K||0^k)$ per evaluation, or $\lambda/2$ of K. Because the nonce is not in the initial state, it will only leak $\lambda/2$ bits of K after 2 evaluations.

(c) We couldn't find a way to obtain the key if the nonce is not included in the initial state.

(d) Because the nonce gets added separately, the key does not leak as the same information of the key gets leaked every single time. Because the same information gets leaked every construction query, there is not enough information leaked to do a successful attack, thus it is not a problem.