Length Extension Attack

Group 3

Ishita Gupta   Sri Rajitha
Rohith Krishna   Yashraj Varma
Aditya Kedar Tata   Akshat Mehra
Roopchand Parise   Parikshit Powani

Madras School of Economics

23 April 2021

Outline

Iterated Hash Functions

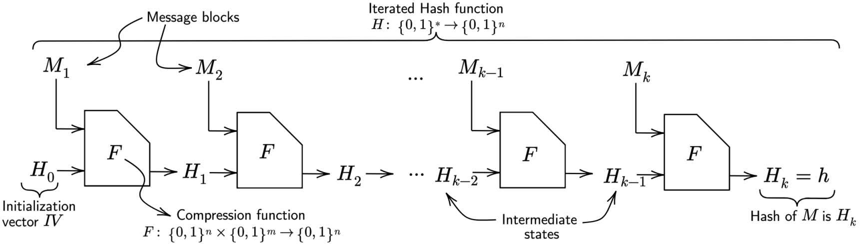
- An iterated hash function H takes an arbitrary length input message $M \in \{0,1\}^*$ and computes n-bit digest $h = H(M)$.
- Each message M is split into blocks $M_1, M_2, \ldots, M_k \in \{0,1\}^m$, with block size m.
- For a given M, the number of blocks is computed as $k = \lceil (|M|/m) \rceil$
- Let H_0 is some initialization vector IV, chosen arbitrarily.
- H iterates an underlying compression function F, and the final hash value $H(M)$ is computed as

$$H(M) = \underbrace{F\,\underbrace{(F(\ldots(F\,\overbrace{(F\,\overbrace{(F(H_0, M_1)}^{H_1}, M_2)}^{H_2}, \ldots), M_{k-1})}_{H_{k-1}}, M_k)}_{H_k}$$

Merkle-Damgård Hash Function

Given compression function $F : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$, we need to implement $H : \{0,1\}^* \to \{0,1\}^n$. We are also given H_0 and $M \in \{0,1\}^*$

- Partition message M into k blocks, such that $M = M_1||M_2||\ldots||M_{k-1}||M_k$, and $M_i \in \{0,1\}^{mk}$ for all $i \in \{1,2,\ldots,k\}$.
- Perform MD strengthening. Last block M_k takes the binary value of length $|M|$, and is subsequently padded.
- For all $i \in \{1,2,\ldots,k\}$: compute $H_i := F(H_{i-1}, M_i)$.
- Finally set the digest value, $h = H(M) = H_k$.



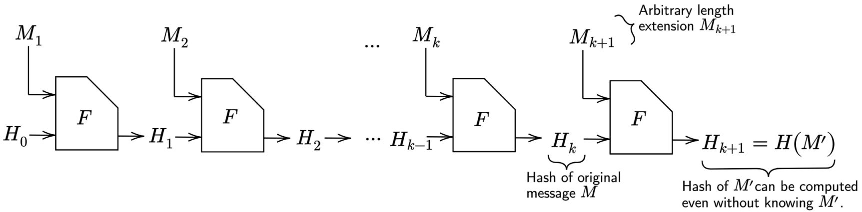
Outline

Literature on Length Extension Weaknesses

- Crypto 1989 papers by Merkle and Damgard do not mention these weaknesses.
- Soren 2009 thesis mentions that it is unlcear as to who first described length extension weaknesses. He remarks, *"They have been folklore knowledge for many years."*
- Tsudik 1992, refers to this vulnerability as padding attack in the context of MAC algorithm with a secret prefix, and observes countermeasures.
- Ferguson 2003, in the book Practical Cryptography mentions how length extensions could be harmful in poorly-constructed MACs, and elaborates why this disqualifies MD-based hash functions with respect to the random mapping based security definition.
- Lucks 2004 and 2005, identifies length extension weakness and proposes a fix called wide-pipe hash functions.

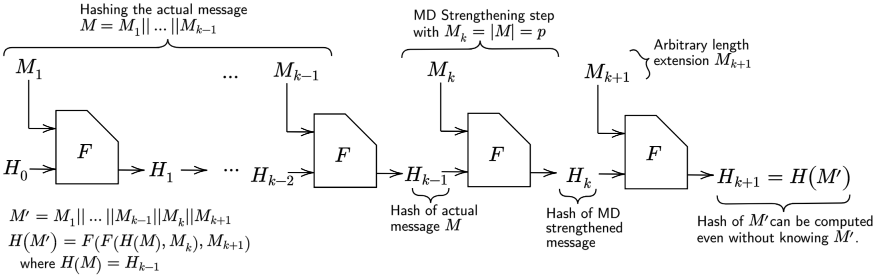
Length Extension Attack

- Let message $M = M_1||M_2||\ldots||M_k$ produce a hash $h = H(M)$.
- Choose new message M' such that $M' = M_1||M_2||\ldots||M_k||M_{k+1}$. The last term M_{k+1} is the length extension.
- The hash value $H(M)$ is merely the intermediate hash value after k blocks, in the computation of $H(M')$.
- Since $H_i = F(H_{i-1}, M_i)$, say we start with (H_0, M_1), it follows from the iterative construction that $H_{k+1} = F(H_k, M_{k+1})$.
- And by definition, $H_{k+1} = H(M')$. Thus $\boxed{H(M') = F(H(M), M_{k+1})}$



Length Extension Attack (with MD strengthening)

- If Merkle-Damgård strengthening is done, then the last block M_k is a padded block containing $|M| = p$ (usually at most 64-bit binary).
- Let $M = M_1 || \ldots || M_{k-1}$ is the actual message, and let the adversary know $H(M) = H_{k-1}$ as well as $|M| = p$ (or can guess p).
- Then adversary can initialize compression F with intermediate state as H_{k-1} and $M_k = \{0\}^{n-p} || p$.
- This produces a hash value: $H_k = F(H_{k-1}, M_k)$.
- On this length extension can be performed with arbitrary string M_{k+1}, such that $H(M') = F(\overline{F(\underline{H(M)}, M_k)}M_{k+1})$



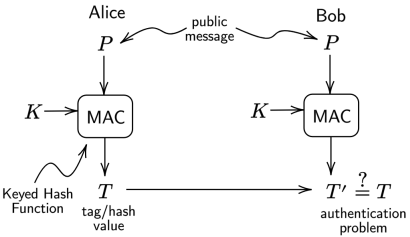
Length Extension Attack

- A major advantage of using Merkle-Damgård scheme is that:
  - Computation starts as soon as M_1 is received.
  - Hashing can happen on the fly, without any storage concerns.
- Length extension attack exists because there is no special processing at the end of the hash function computation.
- The result is that $H(M)$ provides any adversary direct information about the intermediate state after k blocks of hashing M'.
- Even when MD strengthening is done and the adversary has the hash of the actual message, length extension attack can be performed.
  - The adversary must know $|M|$ in order to use the padding block hash and then perform length extension.
  - If the adversary does not know $|M|$, even then length extension can be performed, however this time it makes his job more difficult for now he has to guess the length of the actual message.

Outline

Message Authentication Code (MAC)

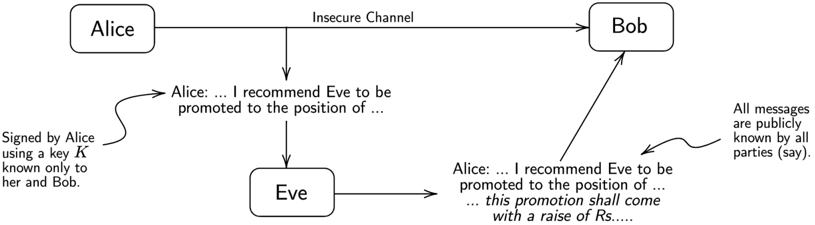
- The MAC algorithm is specific kind *keyed hash function* used for authentication.
- Let P be a publicly known message and let K be a secret key shared by Alice and Bob.



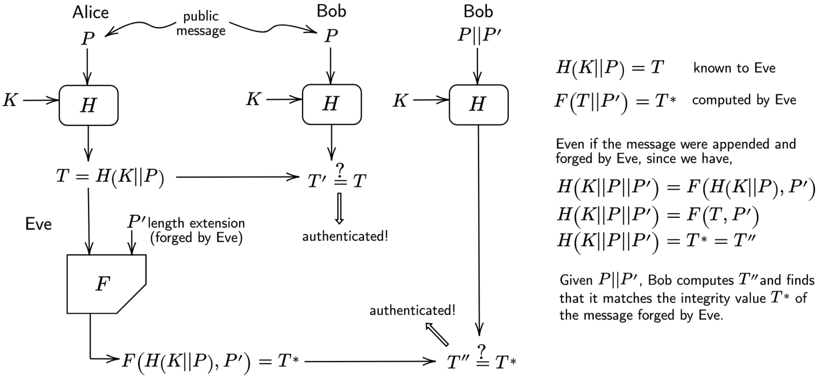
- Alice inputs both P and K into the MAC algorithm which returns a tag T (hash value), which she further sends to Bob.
- Bob also passes his P and K to the MAC, and obtains a tag T'.
- When $T' = T$ Bob can confirm the authentic source of P. Knowing T', Bob cannot however obtain P because of preimage resistance.
- We assume the secret-prefix scheme for input to the MAC.
- In hash function terminology $K||P$ is the input message corresponding to hash $T = H(K||P)$

Length Extension Attack in MAC

- Length extension attack can be harmful in the case of MAC algorithm in secret-prefix scheme, where the adversary can potentially forge messages.
- Eve can pass-off altered versions of messages and also get it authenticated even without knowing the secret K.

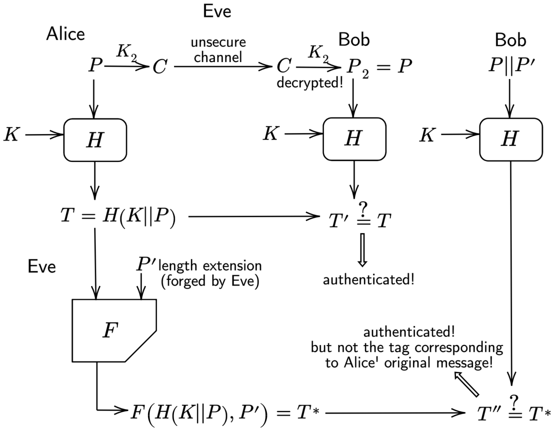


Length Extension Attack in MAC



- We observe that Eve can pass off forged (appended) message $P||P'$ and have it authenticated by Bob even without knowing the secret key K.

Length Extension in MAC when P is encrypted



$H(K||P) = T$ known to Eve

$F\big(T||P'\big) = T*$ computed by Eve

Even if the message were appended and forged by Eve, since we have,

$$H\big(K||P||P'\big) = F\big(H(K||P), P'\big)$$
$$H\big(K||P||P'\big) = F\big(T, P'\big)$$
$$H\big(K||P||P'\big) = T*$$

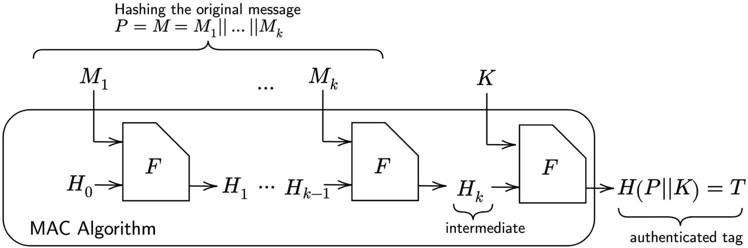
Bob decrypts C to get P_2 which is authenticated by tag T'. But Eve presents another authentic tag $T*$.

- Even though Eve can pass off a forged (appended) message $P||P'$ with the authentic tag T^*, Bob would likely catch Eve this time for he has a different authentic tag T'.

Outline

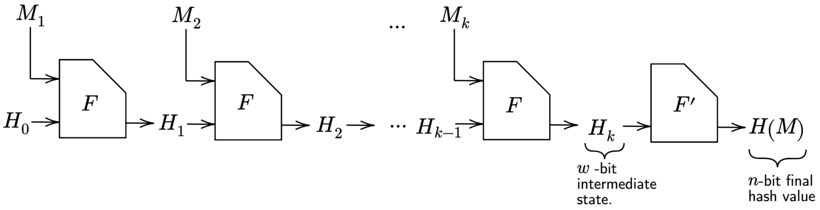
How to fix length extension attack?

- In the context a MAC algorithm, where the message P is public, a straightforward fix is to use a key suffix than a key prefix.
- Thus, the tag now takes the form $H(P||K)$. If Eve performs length extension resulting in tag $T^* = F(H(P||K), P')$. This is not authenticated because the correct tag for the extended message should be $T' = H(P||P'||K)$.
- We observe that $T^* \neq T'$.



Wide-Pipe Iterated Hash Function

- Another method is to use a wide-pipe hash construction, where the internal pipe is widened from n bits to $w > n$ bits.
- We have two compression functions:
  - $F : \{0,1\}^w \times \{0,1\}^m \to \{0,1\}^w$
  - $F' : \{0,1\}^w \to \{0,1\}^n$
- The wide-pipe iterated hash H is computed by:
  - For i in $i \in 1, 2, \ldots, k$, compute $H_i := F(H_{i-1}, M_i)$.
  - Finally set hash $H(M) = F'(H_k)$



Outline

Revisiting Security of Hash Functions

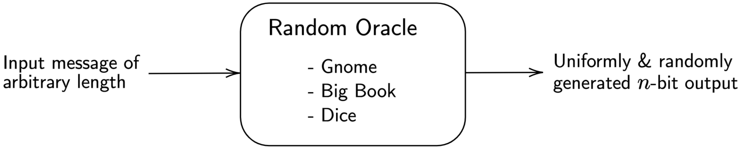
- Hash function $H : \{0,1\}^* \to \{0,1\}^n$ take an arbitrarily sized input and produce a n-bit output, called the hash value or the digest.
- Some security requirements of good hash functions: Preimage resistance, Second preimage resistance, Collision Resistance, etc.
- *(Stronger condition)*: We require an ideal hash function to be indistinguishable from a random oracle (thought of as a random mapping from all possible messages to all possible digests).
- Any mapping is truly random if and only if the design involves a physically random underlying process.
- Otherwise it is only a pseudo-random mapping - appears to be random but is in fact deterministic but convoluted so much that it is computationally infeasible to invert.

Generic attack on a Hash Function

- An attack on a hash function is a non-trivial method of distinguishing the hash function from an ideal hash function.
- This non-trivial method of distinguishing includes all generic attacks.
- For example using a birthday attack, for a n-bit digest it requires only $2^{n/2}$ steps in order to find a collision with probability $1/2$.
- Goals of different attacks vary. They exploit different security weaknesses. For instance, one goal could be to find a preimage M given hash $H(M)$. Other goals could be to find some kind of structure in hash outputs.
- Any distinguisher between and ideal hash function and a real hash function has to be more computationally efficient than a generic attack that yields similar results.

Random Oracle

- Think of a random oracle as a black box: A mythical gnome lives in it with a big book and some dice.
- We can input a string of arbitrary length into the box.
- If the input is one that the gnome has not seen prior, he uses the dice to generate a new output, uniformly and randomly, in the space of oracle outputs.
- He notes down the input and generated output in his book.
- If the gnome has seen the input before, he uses his book to recover and return the same output.



Random Oracle

- A fixed size random oracle is a function $f : \{0,1\}^a \rightarrow \{0,1\}^b$, chosen uniformly at random from a set of all such functions $\mathcal{F}$. Given $x \in \{0,1\}^a$, it computes $y = f(x) \in \{0,1\}^b$.
- A variably sized random oracle is a random function $g : \{0,1\}^* \rightarrow \{0,1\}^b$, viewed as an infinite set of fixed size random oracles, one oracle $g_a : \{0,1\}^a \rightarrow \{0,1\}^b$ for each $a \in \mathbb{N}$.
- We think of a fixed-size random oracle as an ideal compression function.
- We think of a variably-sized random oracle as an ideal hash function.
- They are an idealized version of hash functions, such that we know nothing about the output h, that we could get for any given input message M, unless we actually query the oracle.
- This is useful for security proofs because they allow to express the attack effort in terms of number of invocations/queries to the oracle.

Random Oracle

- The problem is that building a truly random oracle, or equivalently an ideal hash function is very difficult.
- A secure hash function is resilient to collisions, preimage and second preimage attacks; but these do not imply that the hash function is a random oracle.
- We saw that $H(M)$ provides an adversary direct information about the intermediate state after k blocks of hashing M'.
- Thus this disqualifies MD hash functions to be secure according to our stronger requirement of hash functions to be a random mapping (oracle).
- Any adversary with access to the oracle (hash function), knowledge of $|M|$ and $H(M)$, can imagine to create a new message M' by appending an arbitrary string to M.
- This means the adversary would end up getting $H(M')$ from the gnome in the random oracle, without even providing the message M'.

References

1. Merkle, Ralph C. "One way hash functions and DES." Conference on the Theory and Application of Cryptology. Springer, New York, NY, 1989.
2. Damgård, Ivan Bjerre. "A design principle for hash functions." Conference on the Theory and Application of Cryptology. Springer, New York, NY, 1989.
3. Ferguson, Niels, and Bruce Schneier. Practical cryptography. Vol. 141. New York: Wiley, 2003.
4. Tsudik, Gene. "Message authentication with one-way hash functions." ACM SIGCOMM Computer Communication Review 22.5 (1992): 29-38.
5. Lucks, Stefan. "Design Principles for Iterated Hash Functions." IACR Cryptol. ePrint Arch. 2004 (2004): 253.
6. Gauravaram, Praveen. Cryptographic hash functions: cryptanalysis, design and applications. Diss. Queensland University of Technology, 2007.
7. Gusev, Marjan, and Pece Mitrevski, eds. ICT Innovations 2010: Second International Conference, ICT Innovations 2010, Ohrid Macedonia, September 12-15, 2010. Revised Selected Papers. Vol. 83. Springer Science & Business Media, 2011

References

8. Canetti, Ran, Oded Goldreich, and Shai Halevi. "The random oracle methodology, revisited." Journal of the ACM (JACM) 51.4 (2004): 557-594.
9. Al-Odat, Zeyad, and Samee Khan. "Constructions and Attacks on Hash Functions." 2019 International Conference on Computational Science and Computational Intelligence (CSCI). IEEE, 2019.
10. The Story of Alice and her Boss `https://www.cits.ruhr-uni-bochum.de/imperia/md/content/magnus/rumpec05.pdf`
11. Cryptography Stack Exchange - *Understanding the Length Extension Attack*, Thomas Pornin

    `https://crypto.stackexchange.com/questions/3978/understanding-the-length-extension-attack`
12. Cryptography Stack Exchange - *What is the Random Oracle Model and why is it controversial?* , Thomas Pornin
    `https://crypto.stackexchange.com/questions/879/what-is-the-random-oracle-model-and-why-is-it-controversial`