1. {Fk}k∈ {0,1} n: {0, 1} l(n) → {0, 1} l(n) be a PRF family.

F ′k(x) = Fk(x) if x is even

Fk(x + 1) if x is odd

MAC scheme works as follows:

Mack(m) outputs t = F ′k(m).

Vrfyk(m, t) outputs 1 if and only if t = F ′k(m).

Adversary A:

* A queries t1: = Mack (1n)
* A outputs (z, t1).
* A wins if Vrfyk(m, t) = 1 and m has not been queried yet.
* Since, z has not been queried yet and t will be the same for F’k(1) and F’k(2),

Pr[A wins] = 1.

Hence, this MAC scheme is insecure because the adversary can easily forge a MAC for any message m' without knowing secret key k.

1. To show that the given CBC-MAC scheme with a random IV is insecure, we have to create an adversary that wins with non-negligible probability.

Adversary A:

* A queries t1 := Mack(1n).
* A aborts if t1 != 0n || F’k(1n).
* If t1 = 0n || F’k(1n), then A continues the experiment.
* Now, A changes t1 = 1n || F’k(1n) by flipping the first bit of the MAC and outputs (0n, t1).
* A wins if Vrfyk(0n, t1) = 1 and m which is 0n has not been queried yet.

Pr[A wins] = ½.

Hence, the given CBC-MAC scheme with a random IV is insecure.

1. To prove that the given CBC-MAC scheme with a random IV is insecure, we have to create an adversary that wins with non-negligible probability.

Adversary A:

* A queries t1 := Mack(m[1] || m[2]) where |m[1]| = |m[2]| = n and |m| = 2n. A receives back from the oracle t where t:= r || Fk(Fk(r ⊕ m[1]) ⊕ m[2]); where |r| = |m[1]| = |m[2]| = n and |m| = 2n.

Get length 2n 🡪 r || m[2];

We can simply replace all occurrences of r with m[1] and vice versa. This results in the expression m[1] || F'k(F'k(m[1] ⊕ r) ⊕ m[2]).

So, the new expression after swapping r and m[1] is:

m[1] || F'k(F'k(m[1] ⊕ r) ⊕ m[2])

* A outputs (r || m[2], t) to the challenger.
* A wins if Vrfyk(r || m[2], t) = 1 and m which is r || m[2] has not been queried and Vrfyk = 1. There is this chance that if r = m[2], probability that this occurs and A fails to win is 2-n.

This is something we also consider while analyzing A’s winning probability.

So, Pr[A wins] = 1 - 2-n

which is non-negligible for large values of n.

Hence, the given MAC scheme is insecure.

1. To prove that the given CBC-MAC scheme is insecure, we construct a distinguisher that can distinguish between the MAC of a randomly chosen message and the MAC of a message chosen by the adversary with a non-negligible advantage.

Adversary A has access to MAC oracle Mack(.) which returns a valid tag t’ for any queried message m.

O(.) evaluates for a given message m, the adversary A chooses a random n-bit string s and queries the oracle for the tag t = s || Hs(k || m). Let z0 = 0n and let x = (k || m) || 0n.

Then, A can construct another tag t' as follows:

* A chooses a random n-bit string s.
* A makes a MAC oracle query on the message m = 0n, and obtains the response s' || H(k||0n), where s' in {0,1}n.
* Input x🡪 {0,1}n and parsed x || (|x|) as x1 || x2 || x3, where |x1| = |x2| = |x3| = n.
* For first MAC, |x| = 2n. So, x|| (2n)2 will be used as input. Add 0s as padding in the end, till (2n)2 becomes a multiple of n.
* Compute z1 = Hs(z0 || x1) = Hs(k || m).
* Compute z2 = Hs(z1 || x2) = Hs(Hs(k || m) || 0n || m).
* Compute z3 = Hs(z2 || x3) = Hs(Hs(Hs(k || m) || 0n || m) || 0n).
* Output t' = z3 || z1.
* A wins if Vrfyk(m, t’) = 1 and m has not been queried yet.

For another input, we choose |x| = 3n. So, x|| (3n)2 will be used as input. Add 0s as padding in the end, till (3n)2 becomes a multiple of n and repeat similar step.

t' has the same length as t, and that k || m is a prefix of x. z1 = Hs(k || m) is used as an intermediate value while computing Hs(x).

Now, since t' and t are constructed in a similar way using the same k and m. Therefore, the probability that A forges a valid tag without knowing key k is ½.

i.e., Pr[A wins] = ½.

which is non-negligible.

Hence, the given variable length MAC scheme is insecure.