Midterm

October 14, 2019, 10:10–11:00 AM

There are five problems each worth five points for a total of 25 points. Show all your work, partial credit will be awarded. A blue book is provided for your work. Put your name on the blue book, sign it and return it with the test. No notes, no collaboration.

Name: ______________________________

Problem	Credit
1	
2	
3	
4	
5	
Total	

1. *Vigenere key extraction*

   The Vigenere cipher was not created with chosen plaintext attacks in mind.

   $\star$ Given a Vigenere cipher with an unknown key, explain how, with a single chosen plaintext, you can discover the key.

   Note: We have it from a very reliable source (who is now in prison for fraud and witness tampering) that the key is never longer than 10 characters.

   Also, there is no need to disambiguate between keys such as "ab" versus "abab".

   Solution: Let $V_k(m)$ represent the encryption. Set the chosen plaintext to m ='aaaaaaaaaa'. Then

   $$k' = V_k(m)$$

   is the key (in the sense $V_k = V_{k'}$, identically).

2. *Conditional probabilities*

   A desperate message goes out to an operative to bring home a necessary grocery for dinner. By previous "intel" (intelligence), Q has informed us that the probability of each grocery is:

$Pr[m]$	m
2/16	apple
4/16	butter
2/16	carrot
1/16	eggs
1/16	lettuce
1/16	pear
2/16	pepper
3/16	tomato

   While shoulder surfing, counter-intelligence (spies that spy on spies) learns that the message that was sent has 6 letters in it.

   ⋆ What is the new probability for each of the 8 messages given this conditioning event C,

$Pr[m \mid C]$	m
0	apple
4/11	butter
2/11	carrot
0	eggs
0	lettuce
0	pear
2/11	pepper
3/11	tomato

Solution: The event of a message with 6 letters is,

$$C = \{\, butter, carrot, pepper, tomato \,\}$$

and so,

$$\begin{aligned} Pr[C] &= Pr[\{\, butter, carrot, pepper, tomato \,\}] \\ &= (4+2+2+3)/16 = 11/16. \end{aligned}$$

The conditional probability is defined as,

$$Pr[m \,|\, C] = Pr[m \cap C]/Pr[C].$$

The set $m \cap C$ is empty when $m \notin C$. For for these m, $Pr[m \,|\, C] = 0$. Else $Pr[m \cap C] = Pr[m]$,

$$\begin{aligned} Pr[butter \,|\, C] &= Pr[butter]/Pr[C] \\ &= (4/16)/(11/16) \\ &= 4/11, \\ Pr[carrot \,|\, C] &= Pr[carrot]/Pr[C] \\ &= (2/16)/(11/16) \\ &= 2/11, \\ Pr[pepper \,|\, C] &= Pr[pepper]/Pr[C] \\ &= (2/16)/(11/16) \\ &= 2/11, \\ Pr[tomato \,|\, C] &= Pr[tomato]/Pr[C] \\ &= (3/16)/(11/16) \\ &= 3/11, \end{aligned}$$

3. *Protocol flaws and message forgery*

   Cryptographic expert Ive Gonetovish has created a combined encryption and authentication scheme for an authenticated encryption.

   However, he doubts whether it is necessary to have separate keys for the encryption and the authentication. After a light lunch consisting a turkey sandwich, a salad, french fries, two slices of pecan pie and a diet coke, he comes back from lunch and unwisely decides to use just one key for both. Then he falls asleep under his desk.

   You must show him that this decision was a mistake — to use only one key, not that he had a diet coke.

   The encryption is CBC with a randomly chosen IV, using the AES block cipher. The authentication is a CBC-MAC also using the AES cipher.

   $\star$ Show how you can use a chosen plaintext attack to create a (m,t) message tag pair which verifies for a message that was never signed

   Solution: The chosen plaintext is m_0. The returned ciphertext is IV, c_0, t where;

$$\begin{aligned} c_0 &= E_k(IV \oplus m_0) \\ t &= E_k(m_0). \end{aligned}$$

   The proposed forgery is on the message $m_f = IV \oplus m_0$, for which the tag is c_0,

$$\begin{aligned} \text{Vrfy}(k, m_f, c_0) &= E_k(m_f) == c_0 \\ &= E_k(IV \oplus m_0) == c_0 \\ &= c_0 == c_0 \\ &= \text{True} \end{aligned}$$

4. *Probabilistic Polynomial Time Algorithms*

   There are two ways to think about the non-determinism of a PPT.

   In one, a device delivers a random bit on request. In the other, there is a pre-drawn sequence of random bits, and the next bit in the sequence is delivered on request. The difference is that the pre-drawn sequence can run out of bits during the algorithm's run but the device delivering bits on request will not run out.

   However, this difference is unimportant and the two models are precisely equivalent.

   $\star$ Show how to create a pre-drawn sequence that, while finite, is guaranteed to be long enough for any computation of the PPT, given our security parameter n.

   Solution: A PPT runs in a polynomial time bound $q(n)$. To model of a machine with a pre-drawn sequence, begin with a query of $q(n)$ random bits, which is places into the pre-drawn sequence. Since this is a polynomial process, and provides sufficient bits for the remainder of the algorithm, the composite is a PPT on a pre-drawn tape.

   As mentioned in class, for the grade, it was acceptable to claim the creation of an exponentially long pre-drawn sequence, but one should mention the $f(n) \in O(2^n)$ that specifies that bound.

   However, that is not playing fair, since in creating so long a sequence, it is possible that sequence creator has enough time to actually solve the problem. It can then either abandon creating the sequence and just send the answer, or embed the answer into the sequence.

5. *Forgery Attack Models for Canonical Verification*

A MAC consists of a PPT `Gen(n)` that generates proper length keys; a PPT `Mac(k,m)` that calculates the tag from the key and the message; and a deterministic polynomial time algorithm `Vrfy(k,m,t)` that verifies the messsage-tag pair, given the key, message and tag.

The canonical verification is when `Vrfy` is equal to:

```
def Vrfy(k,m,t):
        return t==Mac(k,m)
```

In our forgery game we gave the adversary oracle access to the `Mac` function to make test tags on messages of its choosing.

⋆ Show that if canonical verification is used, the adversary can also be given oracle access to the `Vrfy` function to test messsage-tag pairs of its choosing, and this will not change the adversary's power.

Hint: You have to make a general argument that works for any adversary. I suggest you consider how, for this special sort of `Vrfy`, an adversary wishing to have a verify oracle can build it for itself from the oracles it already has.

Full credit needs to explain not only how to build the oracle, but how it gets used.

Solution: Let $\mathcal{A}$ be an adversary that has access to both a MAC and a VRFY oracle. Create adversary $\mathcal{A}'$ that has only access to a MAC oracle by acting as $\mathcal{A}$ until there is a query (m,t) to the VRFY oracle.

On VRFY (m,t) , $\mathcal{A}'$ simulates the verification oracle using the MAC oracle as,

$$t == MAC(m).$$

In addition, the simulation must intervene between the adversary and the protocol Π, so that when Π asks that the MAC report all the queried messages, the simulation removes from the reply those messages that were only verified.

Since any adversary with a verification oracle can be simulated by an adversary without a verification oracle (and the converse being trivial), the two models are of equivalent power.