# *Hardware-Oriented Security and Trust*
# *ECE 4156 HST / ECE 6156 HST*
# Spring 2024

# Assoc. Prof. Vincent John Mooney III
# Georgia Institute of Technology

# Homework 4, 75 pts. (ECE 4156) 90 pts. (ECE 6156)
# Due Friday February 9 prior to 11:55pm

1) (15 pts.) Consider the stateful variant of CBC-mode encryption where the sender simply increments the *IV* by 1 each time a message is encrypted (rather than choosing *IV* at random each time). Show that the resulting scheme is *not* CPA-secure. (NOTE: this is problem 3.20 on page 104 of Katz and Lindell.)

**Solution**

Define A as follows:
(a) Query the encryption oracle with the message $m = 0^{n-1} \| 1$. Receive in return a cipher text $\langle IV, c \rangle$.
(b) If *IV* is odd (i.e., the least significant bit is 1), then output a random bit.
(c) If *IV* is even, then output a pair of messages $m_0, m_1$ where $m_0 = 0^n$ and $m_1$ is any other message. Receive in return a challenge cipher text $\langle IV + 1, c' \rangle$.
(d) If $c == c'$ output 0; else output 1.

The probability of success for A when *IV* is odd is 50%. For even *IV*, consider the fact that *IV* is incremented by 1 for the next encryption. Therefore, since *IV* is even, the result is that $IV + 1 = IV \oplus (0^{n-1}||1)$. Thus, for even *IV*, we find the following from step (a) above:

$$c = F_k(IV \oplus m)$$
$$c = F_k(IV \oplus m \oplus 0)$$
$$c = F_k(IV \oplus m \oplus 0^{n-1}1 \oplus 0^{n-1}1)$$
$$c = F_k(IV \oplus 0^{n-1}1 \oplus m \oplus 0^{n-1}1)$$
$$c = F_k(IV + 1 \oplus m \oplus 0^{n-1}1)$$
$$c = F_k(IV + 1 \oplus 0^{n-1}1 \oplus 0^{n-1}1)$$
$$c = F_k(IV + 1 \oplus m_0)$$

The challenge cipher text (*IV* +1, c') is the encrypted text of $m_0$ if c' == c; in this case, then A correctly outputs 0. Otherwise, for even *IV*, $c' \neq c$, then A correctly outputs 1. Thus, the overall success probability of A is ¼ + ½ hence there is **75 % success probability**. The modified **CBC mode is not CPA-secure.**

2) (10 pts.) What is the effect of a single-bit error in the ciphertext when using the CBC and CTR modes of operation? (NOTE: this is slightly simpler version of problem 3.21 on page 104 of Katz and Lindell.)

## Solution

Given that a cipher text $c_1$, $c_2$, ... generated from a message $m_1$, $m_2$, ... has a bit flipped in its stream of bits (e.g., $c_i$ has a bit flipped). We look at the effect of decrypting the resulting (modified) ciphertext using each of the stated modes to obtain a message $m_1$', $m_2$', ….

**CBC mode:**
Assuming that a single bit being flipped in $c_i$ will result in a modified block $c_i$'. When decrypting, $m_i' = F_k^{-1}(c_i') \oplus c_{i-1}$. This value will be completely different than the expected value of $F_k^{-1}(c_i)$. Hence, the obtained value for the bit flipped message block is irrelevant and has no relation to $m_i$. The next message block will be computed by $m_{i+1}' = F_k^{-1}(c_{i+1}) \oplus c_i'$. The message block $m_{i+1}'$ will be almost equal to $m_{i+1}$ but with a single bit flipped. The rest of the message blocks will be unchanged.

**CTR Mode:**
A bit flip in ciphertext $c_i$ where i > 0 will only cause the message block $m_i$ to flip a single bit. But if the initial ciphertext block $c_0$ (i.e., the counter!) has a bit flipped this will cause the whole message to be decrypted incorrectly.

3) (10 pts.) What is the effect of a dropped ciphertext block (e.g., if the transmitted cipher-text $c_1, c_2, c_3$, ... is received as $c_1, c_3$, ...) when using CBC and CTR modes of operation? (NOTE: this is slightly simpler version of problem 3.22 on page 104 of Katz andLindell.)

We are given that a cipher text $c_1$, $c_2$, $c_3$, $c_4$,… from message $m_1$, $m_2$, $m_3$, $m_4$,… has a block $c_i$ dropped from its stream of blocks (e.g., each block may have 128 bits) being transmitted.

## Solution

**CBC mode:**
A single block $c_i$ being dropped will result in a decrypted message $m_1$, $m_2$, …, $m_i$, $m_{i+1}$', $m_{i+2}$ , …. In other words, message blocks before position *i* are recovered correctly, message block *i* is lost entirely, message block *i*+1 is garbled, and message blocks after position *i*+1 are recovered correctly but are shifted by one block.

**CTR Mode:**
A block dropped $c_i$ will cause the message to only recover message blocks up until just prior to position *i*. From position *i* onwards the remaining decrypted message blocks will incorrect.

4) (10 pts.) Say CBC-mode encryption is used with a block cipher having a 256-bit key and 128-bit block length to encrypt a 1024-bit message. What is the length of the resulting ciphertext? (NOTE: this is problem 3.23 on page 105 of Katz and Lindell.)

## Solution

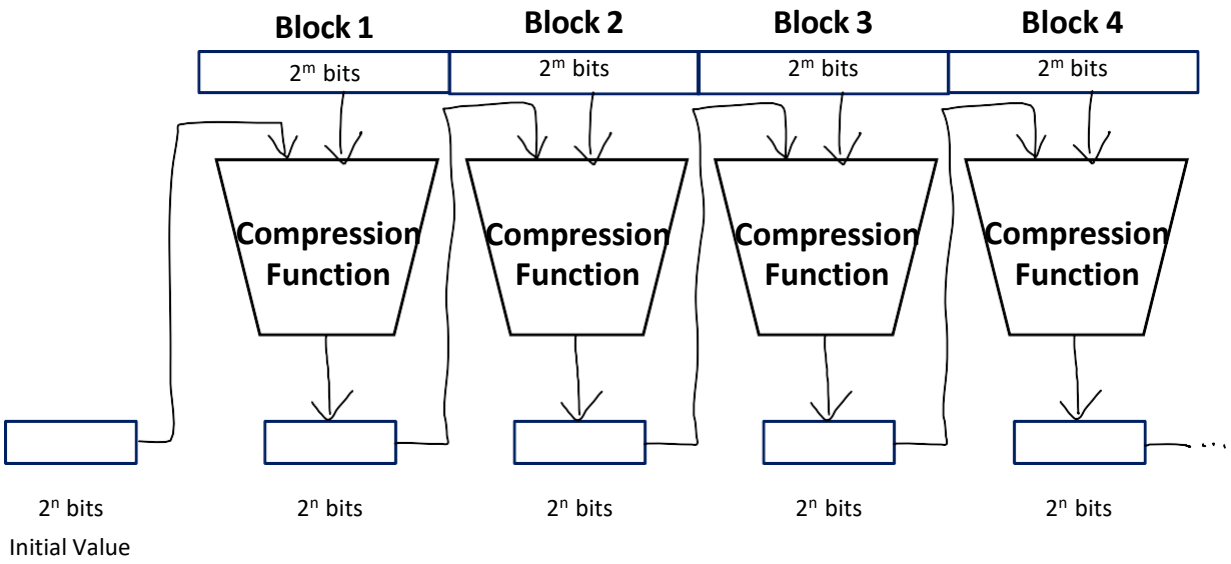
Given,

Key Length = 256
Block length = 128
Message length = 1024

Number of message blocks = 1024/128
= 8
The cipher text will have a size equal to (the number of message blocks +1) * 128
= 9 *128
= **1152**

Therefore, the **ciphertext is 1152 bits long**.

5) (20 pts.) Use your own words (do not copy from any source!) to explain the Merkle-Damgård construction. Please assume four blocks and a compression function from $<2^m,2^n>$ to $2^n$ where n = 8 and m = 9. In your answer, make sure to answer the following questions: (i) what is the goal of the Merkle-Damgård construction and (ii) what does the Merkle-Damgård construction prove? You may use the picture on the next page in your answer.




## Solution

(i) The goal of the Merkle-Damgård construction is to provide both a method and a proof for how to construct a collision-resistant hash function for arbitrary-length inputs given a collision-resistant hash function for fixed-length inputs.

(ii) The Merkle-Damgård construction proves that if the **fixed-length input hash function used is collision-resistant**, then the **arbitrary-length hash function constructed must also be collision-resistant.**

In the figure above, assume that the four blocks provided have n = 8 and m = 9. The functions shown are four repetitions of a 768-bit to 256-bit compression function. The construction results in a 2048-bit to 256-bit hash function which is provably collision resistant if the individual 768-bit to 256-bit compression functions are collision-resistant.

6) (10 pts.) Consider the case where $N = 5$ and $t = 2$. This is a secret sharing scenario, e.g., five company executives where 2 out of the 5 must be present to enter their password information in order to open a safe. Now consider the case of a 10-bit key required to carry out an action (e.g., open a safe). This key will be predicted from the inputs provided by any two of the executives. Please note that as discussed in class the key would in an actual practical scenario have a large number of bits, e.g., 128; however, in this homework we are going to use a number less than one thousand (i.e., can be represented in ten bits) for ease of providing an answer.

a. (1) Choose a key between one and one thousand, email the TA, Kevin Hutto, at khutto30@gatech.edu and then check the course webpage for homeworks http://mooney.gatech.edu/Courses/ECE4156/hwlabexam/index.html to see if your number has already been taken or not. You need to make a first attempt to choose a number prior to 10am on Tuesday Feb. 6. If your first name, first letter of your last name, and selected number appear under hw4, you are done. Otherwise, prior to 10am on Wednesday Feb. 7, you need to make a second attempt to choose a number between one and one thousand and send another email to the TA. Again, if your first name and number appears under homework 4, you are done. Otherwise, prior to 10am on Thursday Feb. 8, you need to choose a third number. Please just use a decimal representation, i.e., choose between 1 and 999.
   Choose a **key between 1-999 before the deadline to not lose a point.**

b. (9) For your given `key`, set the 10-bit key on the y-axis of a Cartesian plane consisting of zero to 1023 on the y-axis and zero to 1023 on the x-axis. Choose two points in Cartesian coordinates which define a line that intersects the y-axis at the `key` location. Specify these two 2-dimensional points `x1,y1` and `x2,y2`.

NOTE: the points you choose must all have a y-axis coordinate different than the key, i.e., `y1` ≠ `key` and `y2` ≠ `key`.

**Solution**

Assuming two points (x1,y1) and (x2,y2).

First finding the slope we get slope (m)
Finding the y intercepts, y_int = y – m*x
y_int_1= K
y_int_2= K
The two points (x1,y1) and (x2,y2) intersects the y-axis at the `key` location.

Here is a list of possible answers (**not all answers provided by the class have been included**!).

| **Key** | **x1, y1** | **x2, y2** |
|---|---|---|
| 1 | 1,2 | 2,3 |
| 17 | 1,34 | 2,51 |
| 37 | 250,23 | 500,9 |
| 100 | 40,140 | 80,180 |
| 161 | 2,162 | 4,163 |
| 227 | 100,419 | 300,803 |
| 250 | 100,300 | 350,425 |
| 276 | 300,552 | 600,828 |
| 337 | 103,543 | 251,839 |
| 448 | 100,473 | 200,498 |
| 613 | 200,473 | 400,333 |
| 672 | 300,447 | 500,297 |
| 776 | 23,385 | 42,62 |
| 777 | 290,574 | 650,322 |
| 900 | 137,626 | 322,256 |

7) [ECE 6156 only!] (15) Given your choice in 6.b, write pseudocode in a C-like syntax (or C++-like syntax or even a Java-like syntax) which, given two inputs from two of the executives, calculates the key. Use the code below as a starting point for your pseudocode. Please note that this problem will **not** be graded on *syntax* but rather on readability and correctness, so do please explain any assumptions with comments. For example, below the value `key` is the number you selected via email to the teaching assistant. Finally, please note that your pseudocode should work independently of the values of `key`, `x1`, `y1`, `x2` and `y2`.

## Solution

First of all, note that the slope of the line can be calculated by subtracting the second key from the first:

***slope = (y2 - y1)/(x2 - x1).***

Note however that if the second point is further from the y-axis, then the direction of the slope (i.e., the sign) needs to be changed:

if **(x1 < x2)** then **slope = -slope**

The calculated key occurs at the y-intercept, i.e., **x = 0**

Since **slope** = (y - y1)/(x - x1) = (y - y2)/(x - x2) = **(y2 - y1)/(x2 - x1)**,

for the point **(0, y) (Y-intercept)** the final result is that we find that

**key = y1 - slope*x1 = y2 - slope*x2**. The resulting pseudocode is as follows:

```
int key;              /* this is your key value selected in 2.a */
int testkey(x1, y1, x2, y2) {
int x1,x2,y1,y2;
Boolean condition;  /* condition is either equal to 1 or is equal to 0 */

/* fill in pseudocode here */

double slope, y_int_1, y_int_2;

slope = (y2-y1)/(x2/x1);

y_int_1 = (y1 - slope*x1);

y_int_2 = (y2 - slope*x2);

condition = 0;

if (y_int_1 == y_int_2)
   if (y_int_1 == key)
      condition =1;

if (condition)
  return (1);    /* if x1,y1 and x2,y2 intercept the y-axis at key */
else
  return (0);    /* if x1,y1 and x2,y2 do no intercept the y-axis at key */
}
```