Datum : 17.05.2023

Minimax Machine

Miniprojekt Hardware-Praktikum

Gruppe 6:

Yassine Bibi - 10047454

Adem Ben Rejeb - 10047407

Cing Cong - 10032176

Aufgabenstellung :

Durch eine Ransomware-Attacke ist eine Datei verschlüsselt worden, die nun wieder entschlüsselt werden muss. Der verschlüsselnde Algorithmus ist bekannt und der Schlüssel wurde auf dem Hauptspeicher gefunden, sodass der Algorithmus nun auf der Minimax Maschine umzusetzen ist, um die Datei wieder lesbar zu machen.

Unsere Minimax-Maschine:

Hinzugefügte Register:

i_Value: Wert des ersten Index
j_Value: Wert des zweiten Index
x_Value: Wird Als Byte index genutzt

Adresse_i: Wird nur in PRGA genutzt als Output Adresse genutzt

S[i]_Value: Wert im S-Array des Indexwerts i
S[j]_Value: Wert im S-Array des Indexwerts j

ACCU2: Temporäre/Zwischen Werte
K_Value: KSA: Speichert die Key Byte
PRGA: Speichert die Output Byte

Hinzugefügte Konstanten:

4: Die Größe des Schlüssels in Bytes

8,16,24: Shift/Rotate für die manipulation von Bytes(Bit Operationen)

255: Der letzte Index in der S-Box oder im S-Array. Dies kann als Endbedingung für eine Schleife im Algorithmus verwendet werden.

256: Key Adresse und Wird Auch als Data-1 Adresse genutzt

63: Kontrolliert die Decompressing von Sbox

2473: Lange von Encrypted_data

16776704: Intermediate Wert für die Decompressing von Sbox

2732: Output Adresse

Hinzugefügte Operationen:

B S.R A/B R.R A/ B R.L A / B S.L A / A S.L B :

A OR B:

A XOR B:

A AND B:

B SUB A, A SUB B, A ADD B, B DIV A, B MOD A, A MUL B:

INV B

Setup der Minimax-Maschine:

Damit die Maschine betriebsbereit ist, müssen wir den Speicher wie folgt einstellen und initialisieren:

0x000000 → 0x00003F : Sbox

0x000100 : Schlussel

0x000101 → 0x000AAA : Encrypted Data.

0x000AAC → 0x001454 : Decrypted Data

Nachdem die Maschine ausgeführt ist. Kann man die Speicherdaten von 0x000AAC → 0x001454 exportieren und kriegt die entschlüsselten Daten.

Um Daten von anderer Länge zu verwenden, muss man die Konstanten im Register entsprechend anpassen.

Beschreibung der Steuersignale Tabelle:

1-Array-Permutation mit Hilfe des Key(KSA):

Als erster Schritt des RC4-Algorithmus wird das Array einer Permutation unterzogen, indem der bereitgestellte Key-Value und der folgende Algorithmus verwendet werden:

```
j := 0
for i from 0 to 255
{
j := (j + S[i] + key[i mod key length]) mod 256
swap values of S[i] and S[j]
}
```

Der erste Schritt des Prozesses besteht darin, eine S-Box zu handhaben, die vom Professor bereitgestellt wird und sich innerhalb eines spezifischen Speicherbereichs befindet: von 0x00 bis 0x40 in hexadezimaler Notation.

Diese spezielle S-Box ist komprimiert, wobei jede ihrer Adressen vier Bytes enthält. Um diese Daten effektiv zu manipulieren, implementiert das Programm einen Dekompressionsprozess.

0-10: Das Programm liest den vier Byte-Wert von der entsprechenden Adresse der S-Box. Um die einzelnen Bytes aus diesem Wert zu extrahieren, verwendet das Programm bitweise Operationen: eine Rechtsverschiebung und eine UND-Operation mit dem Wert 255. Diese Operationen dekomprimieren den vier Byte-Wert effektiv in vier einzelne ein Byte-Werte, die vorübergehend in spezifischen Registern gespeichert werden.

11-28: Sobald jeder dieser Register mit den dekomprimierten Bytes gefüllt ist, schreibt das Programm den Inhalt dieser Register in einen leeren Speicherbereich. Dieser Bereich beginnt bei der Adresse 0xFFFE00, wieder in hexadezimaler Notation. Die spezifische Position innerhalb dieses Bereichs, an der jedes Byte geschrieben wird, wird durch eine Formel bestimmt: 4 multipliziert mit dem aktuellen Iterationswert plus dem Index des zu schreibenden Bytes.

29: Zurücksetzen der Register.

30-36: Anschließend wird eine zweite Schleife verwendet. Diese Schleife wird 256-mal durchlaufen, um sicherzustellen, dass jedes dekomprimierte Byte angesprochen wird. Während jeder Iteration liest das Programm den Wert an der entsprechenden Adresse im Speicher, beginnend bei der Adresse 0xFFFE00. Der gelesene Wert wird effektiv als 0 geschrieben, wobei das Programm nach jeder Operation die Speicheradresse um eins erhöht. Diese Operation kopiert effektiv die gesamte dekomprimierte S-Box und ordnet sie von 0x00 bis 0xFF an.

38-42: Nach der Duplizierung führt das Programm eine weitere Schleife aus, um den Speicherbereich bei 0xFFFE00 zu löschen und alle Werte auf Null zu setzen. Diese Operation dient dazu, den Speicherbereich zurückzusetzen und ihn auf den nächsten wesentlichen Schritt des Programms vorzubereiten.

KSA Beginnt:

44-60: Zunächst lädt das Programm den Schlüssel, der an einer bestimmten Adresse gespeichert ist, in das Memory Data Register (MDR). Der Schlüssel wird aus dem Memory Address Register (MAR) extrahiert, wo er gespeichert ist. Anschließend wird ein Index des Bytes im Schlüssel mithilfe des Modulo-Operators berechnet, wodurch der Wert key_index erzeugt wird.

61-63: Das Programm fährt fort, den S-Box-Wert, bezeichnet als S[i], in das MDR zu laden. Hierbei entspricht i dem aktuellen Iterationswert. Der S-Box-Wert wird aus dem MAR abgerufen, in das MDR geladen und dann in einer Variable namens S[i]_value gespeichert.

63-67: Eine Variable namens j_value wird dann aktualisiert, indem der Modulo 256 der Summe aus dem aktuellen j_value, s_value und dem extrahierten Byte des Schlüssels berechnet wird.

68-74: Die Austauschoperation beinhaltet zunächst das Speichern des aktuellen S[i]-Werts im Akkumulator (ACCU). Das Programm lädt dann den S[j]-Wert in das MAR, ruft den Wert an dieser Adresse in das MDR ab und schreibt den im ACCU gespeicherten Wert an diese Stelle. Dadurch wird S[i] effektiv an die Adresse S[j] platziert. Die Operation wird abgeschlossen, indem S[j] im ACCU gespeichert wird und anschließend ACCU an die Adresse S[i] geschrieben wird.

75-76: Diese Schleife wird 256 mal wiederholt.

2-Pseudo Random Generation Algorithm Beginnt(PRGA):

Die PRGA (Pseudo-Random Generation Algorithm) im RC4-Algorithmus ist dafür verantwortlich, einen Strom von pseudo-zufälligen Bytes zu generieren, die zur Verschlüsselung oder Entschlüsselung von Daten verwendet werden. Sie verwendet den Schlüsselplanungsalgorithmus und die modifizierte Permutation der S-Box, um den Schlüsselstrom zu erzeugen.

77: Alle Variablen zurücksetzen.

78-84: Die Adresse der neuen unverschlüsselten Daten festlegen und die erforderlichen Operationen zur Bestimmung des Endes durchführen.

85-103: Das Lesen von S[i] und S[j], das Vertauschen der beiden Werte und das Generieren eines Ausgabebits, das als (S[i] + S[j]) mod 255 berechnet wird.

104-Ende:

-Um den Fortschritt zu verfolgen, verwendet die PRGA einen Zähler namens xvalue. Dieser Zähler wird nach jeder Bit-Manipulation erhöht. Um zu entscheiden, welches Byte der Algorithmus gerade verarbeitet, werden eine Reihe von Überprüfungen durchgeführt, die den xvalue involvieren. Durch Auswertung der Bedingungen xvalue = 0 und ACCU = xvalue - 1 und Überprüfung, ob der resultierende Wert null ist, kann das Programm feststellen, ob es in die Schleife eintreten muss, die auf dieses Byte wirkt. Wenn nicht, wird ACCU dekrementiert, bis das richtige Byte erreicht ist. Insgesamt gibt es drei Überprüfungen und vier mögliche Byte-Platzierungen.

-Nachdem die entsprechende Byteschleife betreten wurde, identifiziert das Programm die Speicheradresse zum Schreiben und speichert sie in ACCU. Abhängig vom aktuellen Byte führt der Algorithmus einen unterschiedlichen Satz von Operationen durch.

-Wenn das Byte 0 ist, ruft der Algorithmus einfach das Ausgabebyte ab und schreibt es an die identifizierte Adresse. Wenn das Byte jedoch 1, 2 oder 3 ist, liest das Programm den Speicher an der Schreibadresse und wendet eine Rechtsrotation um 8, 16 oder 24 Bits an, je nach Bytenummer. Anschließend führt es eine ODER-Operation mit dem Ausgabebyte durch, rotiert das Ergebnis um 8, 16 oder 24 Bits nach links (wiederum abhängig von der Bytenummer) und schreibt dies in den Speicher.

-Nach jeder Iteration, außer für das dritte Byte, speichert das Programm den Zustand, erhöht xvalue und kehrt zum Anfang der PRGA zurück. Wenn der Vorgang des dritten Bytes abgeschlossen ist, geht das Programm auch zum Anfang zurück, jedoch mit einem zusätzlichen Schritt: Es liest die entschlüsselten vier Bytes und führt eine XOR-Operation mit den vier Bytes aus, die aus dem vorherigen Prozess erhalten wurden. Der resultierende Wert wird dann an der Zieladresse geschrieben.

-Die Zieladresse und die aktuelle Adresse werden inkrementiert und der Prozess wiederholt sich. Dies setzt sich fort, bis die verbleibende Datenlänge minus die Anzahl der Iterationen Null erreicht, was bedeutet, dass der gesamte Datensatz von der PRGA verarbeitet wurde.