Security Lab – Work Factor und Block Cipher Modes

VMware

Dieses Lab können Sie mit dem **Ubuntu-Image** durchführen. In der Aufgabenstellung wird angenommen, dass Sie mit diesem Image arbeiten.

Alternativ können Sie das Lab auch auf Ihrem eigenen Laptop bearbeiten. Dazu benötigen Sie Java und eine Entwicklungsumgebung. Weiter unten ist erklärt, wie Sie das bereitgestellte Projekt in *NetBeans* oder *Eclipse* importieren können. Ebenfalls benötigen Sie ein Programm zum Entpacken von ZIP Files.

1 Einleitung

In diesem Praktikum berechnen Sie verschiedene Work Factors. Insbesondere studieren Sie den Einfluss von nicht gleichmässig verteilten Schlüsseln auf die Schwierigkeit, einen Schlüssel durch ausprobieren zu raten. Zudem betrachten Sie einen zentralen Unterschied der beiden Betriebsmodi Cipher Block Chaining (CBC) und Electronic Code Book (ECB) von blockbasierten Verschlüsselungsverfahren (Block Cipher). Diese Modi werden von den meisten symmetrischen Verschlüsselungsalgorithmen angeboten.

Hinweis: Ist in diesem Praktikum eine Zahl ohne Kommastellen angegeben, handelt es sich um den exakten Wert. Beispiel: 0, 4, 17/2. Ist eine Zahl mit Kommastellen angegeben, ist der korrekte Wert auf die angegebene Anzahl Kommastellen gerundet. Beispiel: korrekter Wert 1.9994872 wird zu 2.00.

Hinweis: In diesem Praktikum werden einige Rechnungen verlangt. Oft ist bei den Aufgaben das korrekte Ergebnis bereits angegeben. Das soll Ihnen dabei helfen, Ihre Rechnung auf Korrektheit zu prüfen. Denn es kommt bei der Bewertung der Aufgabe mehr darauf an, ob Sie die Rechnung richtig durchgeführt haben, als dass Sie das numerisch richtige Ergebnis präsentieren können. In Aufgabe 3 wird beispielsweise verlangt, dass Sie nachweisen, dass die Summe bestimmter ein einer Tabelle angegebenen Wahrscheinlichkeiten 1 ist. Da die vorgegebene Summe mit «1» angegeben ist und nicht mit «1.00», ist klar, dass hier *exakt* 1 herauskommen muss und nicht nur *ungefähr* 1. Weiterhin reicht es nicht aus, als Antwort einfach beispielsweise zu schreiben $\sum_{k=1}^{16} 1/16 = 1$, denn das formuliert bloss die Frage um. Weiterhin wird es nicht akzeptiert, die Gleichung in Wolfram Alpha oder ein ähnliches System einzugeben und die Antwort ungeprüft zu übernehmen. Sie müssen die Rechnung selbst durchführen, von Hand, und solange es geht ohne Zuhilfenahme eines Taschenrechners.

Hinweis: Zwei für dieses Praktikum hilfreiche Gleichungen sind

$$\sum_{k=0}^{n} x^k = \frac{x^{n+1} - 1}{x - 1}$$

$$\sum_{k=0}^{n} k = \frac{n(n+1)}{2}$$

2 Work Factor

Der Work Factor bezeichnet die mittlere Anzahl Versuche, die man braucht, um ein Geheimnis zu erraten. Formal sei $X = \{x_1, \ldots, x_n\}$ eine endliche Menge möglicher Ergebnisse eines Experiments. Das können beispielsweise die sechs Seiten eines Würfels sein, die nach einem Wurf oben liegen können; die zwei Seiten einer Münze; die verwendeten Schlüssel eines Kryptosystems oder die verschiedenen Blöcke eines Ciphertexts. Laut random oracle model sollten diese Blöcke ja zufällig gewählt sein. Es sei nun weiter für $1 \leq i \leq n$ mit p_i die Wahrscheinlichkeit bezeichnet, dass x_i als Ergebnis des Experiments auftaucht. Für einen fairen Würfels ist beispielsweise $X = \{1,2,3,4,5,6\}$ und es ist wegen der

Fairness $p_i = 1/6$. Ist der Würfel nicht fair, ist zwar X unverändert, aber die p_i sind dann nicht mehr alle gleich. Will man nun das Ergebnis des Experiments raten, und tut man das in der Reihenfolge $x_1, \ldots, x_n$, dann ist der Work Factor die erwartete Anzahl der Versuche:

$$\mathrm{WF}(X) = \sum_{k=1}^{n} k\, p_k.$$

Der Work Factor wird manchmal in bit angegeben. Der Work Factor in bit ist $\log_2 \mathrm{WF}(X)$.

Im Folgenden betrachten wir eine nicht näher ausgeführte Spielzeug-Verschlüsselung mit 4 bit Schlüssellänge. Sie machen nun mit zwei verschiedenen Systemen G (für *good*) und B (für *bad*) Experimente und stellen fest, dass die verschiedenen Schlüssel bei den beiden Systemen mit verschiedenen Wahrscheinlichkeiten $p_{i,G}$ und $p_{i,B}$ ausgewählt werden:

Schlüssel	$p_{i,G}$	$p_{i,B}$	Schlüssel	$p_{i,G}$	$p_{i,B}$
0000	$1/16$	$1/2^1$	**1000**	$1/16$	$1/2^9$
0001	$1/16$	$1/2^2$	**1001**	$1/16$	$1/2^{10}$
0010	$1/16$	$1/2^3$	**1010**	$1/16$	$1/2^{11}$
0011	$1/16$	$1/2^4$	**1011**	$1/16$	$1/2^{12}$
0100	$1/16$	$1/2^5$	**1100**	$1/16$	$1/2^{13}$
0101	$1/16$	$1/2^6$	**1101**	$1/16$	$1/2^{14}$
0110	$1/16$	$1/2^7$	**1110**	$1/16$	$1/2^{15}$
0111	$1/16$	$1/2^8$	**1111**	$1/16$	$1/2^{15}$
	8/16			8/16	

Wie bei jeder Wahrscheinlichkeitsverteilung sollte auch hier die Summe der Wahrscheinlichkeiten Eins ergeben. Verifizieren Sie das für G und B.

G = 16 * 1/16 = 1
B = 1/2^1 + 1/2^2 + ... + 1/2^15 + 1/2^15 ~= 1

Versetzen Sie sich jetzt in die Lage eines Angreifers auf System G. Sie wollen das System knacken, müssen dazu aber Schlüssel einen nach dem anderen ausprobieren. Begründen Sie, warum die Reihenfolge, in der Sie die Schlüssel ausprobieren, keinen Einfluss auf die zu erwartende Anzahl der Proben hat, die Sie brauchen, bis Sie den richtigen Schlüssel herausgefunden haben.

Berechnen Sie nun den work factor von G (korrekte Antwort: 17/2). Berechnen Sie diesen work factor auch in bit (korrekte Antwort: 3.09).

Da jeder Schlüssel gleich wahrscheinlich ist, kommt die Reihenfolge nicht mehr drauf an.

work factor:
(N + 1)/2 -> 17/2 = 8.5

work factor in bits :
log2(8.5) = 3.0874

Die Schlüssel von G haben also etwa 3.1 bit work factor.

Versetzen Sie sich jetzt in die Lage eines Angreifers auf System B. Sie wollen wieder das System knacken, müssen dazu aber wieder Schlüssel einen nach dem anderen ausprobieren. Begründen Sie, warum die Strategie guten Erfolg verspricht, Schlüssel in absteigender Reihenfolge ihrer Wahrscheinlichkeit auszuprobieren.

Berechnen Sie nun den work factor von B (korrekte Antwort: 2.00). Berechnen Sie diesen work factor auch in bit (korrekte Antwort: 1.00).

Da die anfangs Keys eine höhere Wahrscheinlichkeit besitzen.

work factor:
(1*1/2^1) + (2*1/2^2) + ... + (15*1/2^15) + (16*1/2^15) = 1.99996 ~= 2

work factor in bits:
log2(2) = 1 bits

Das System B hat also nur noch etwa 1.0 bit work factor.

Wir haben also gesehen, dass je nach Verteilung der Schlüssel und daher natürlich auch je nach Wissenstand des Angreifers ein System einen deutlich niedrigeren work factor haben kann, als es nach der Schlüssellänge allein eigentlich zu erwarten war.

Stellen Sie nun eine Vermutung auf, bei welcher Verteilung der Schlüssel der work factor am grössten ist. Eine Begründung ist nicht nötig, aber Ihre Vermutung muss zu den beobachteten Fakten passen.

Im Idealfall verwandelt eine Verschlüsselung einen Plaintext in einen Ciphertext, der von rein zufälligem Text nicht zu unterscheiden ist. In diesem Fall sei $X = \{0,1\}$ die Menge der im Text auftretenden bits. Was gilt für die Wahrscheinlichkeiten p_0 und p_1, mit denen ein Null- bzw Eins-bit auftritt? Welchen Work Factor in bit hat also ein bit des Klartexts in diesem Fall? Das Ergebnis wird in Abschnitt 4 noch gebraucht. (Korrektes Ergebnis: etwa 0.6 bit.)

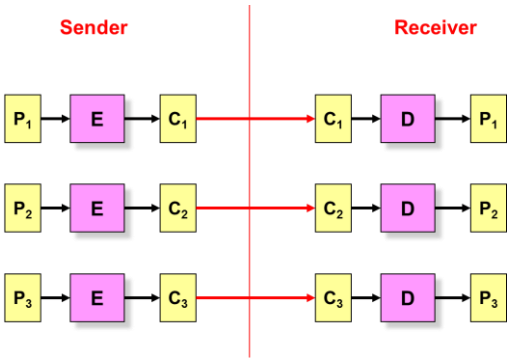
work factor:
(N+1)/2 = (2+1)/2 = 1.5

work factor in bits:
log2(1.5) = 0.6 bit

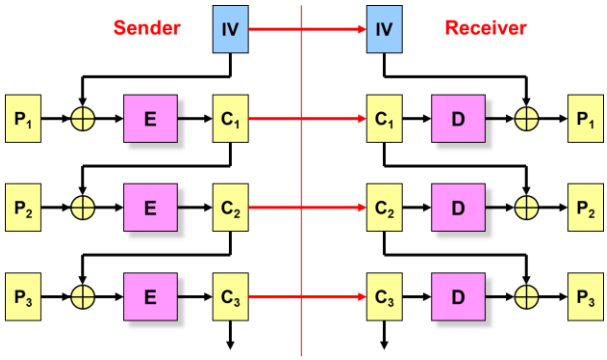
3 Block Cipher Modes

In der Vorlesung haben wir als eine informationstheoretisch sichere Verschlüsselung das One-Time-Pad kennengelernt. Das ist aber reichlich impraktikabel, da die Schlüssellänge so lang wie der Plaintext sein muss und zudem nur für eine Nachricht verwendbar ist. Deshalb werden typischerweise blockbasierte Verschlüsselungsverfahren (*block ciphers*) eingesetzt. Eine Block Cipher ist ein deterministisches Verschlüsselungsverfahren, bei dem ein Plaintext fester Länge auf einen Ciphertext fester Länge abgebildet wird. Die genaue Transformation wird dabei durch einen Schlüssel bestimmt und lässt sich durch das *random oracle model* beschreiben, bei dem durch den Schlüssel eine zufällige Permutation der Eingabewerte auf die Ausgabewerte bestimmt wird. Im Gegensatz zu einer *Stream Cipher* kann ein Block Cipher nur ganze Blöcke verschlüsseln. Zur Verschlüsselung grösserer Datenmengen wird ein Betriebsmodus verwendet, der festlegt, wie die Block Cipher wiederholt anzuwenden

ist. Zwei dieser Modi sind Electronic Code Book (ECB) und Cipher Block Chaining (CBC). Die nachfolgenden Grafiken zeigen die grundlegenden Funktionsweisen:



Electronic Code Book Mode



Cipher Block Chaining Mode

Sicherheitsexperten raten typischerweise dringend von der Verwendung von Block Ciphers im ECB Mode ab. Einer der Hauptgründe für diesen Rat sollte Ihnen nach der Durchführung des folgenden Experiments klar werden. **Hinweis**: Je nach Plaintext und je nach Angreifer kann auch die Verwendung von CBC problematisch sein. Näheres entnehmen Sie bitte der Vorlesung.

Verwenden Sie die ausführbare jar-Datei *AesTool.jar* (diese finden Sie im Verzeichnis *Sec-Lab_BlockCipherModes*, das Sie zu Beginn des Praktikums beim Entpacken des ZIP Files erzeugt haben) um die Bilddatei *image.bmp* (diese finden Sie im Unterverzeichnis *files*) einmal im ECB Mode und einmal im CBC Mode zu verschlüsseln. Die verschlüsselte Datei enthält dabei den Namen *image.bmp.enc* und wird im Unterverzeichnis *files* abgelegt. Benennen Sie diese Datei nach dem Verschlüsseln jeweils um in *image.bmp.ecb.enc* bzw. *image.bmp.cbc.enc*. Wenn Sie das Tool ohne Argumente aufrufen erhalten Sie Informationen, wie es zu verwenden ist.

Als nächstes modifizieren Sie die beiden verschlüsselten Dateien so, dass diese von einem Programm, das BMP Dateien anzeigen kann, angezeigt werden können. Dazu überschreiben Sie den nun verschlüsselten BMP Header wieder mit dem originalen BMP Header. Für die Beispieldatei müssen Sie hierzu die ersten 54 Bytes der verschlüsselten Dateien mit den ersten 54 Bytes der Originaldatei ersetzen. Die Datei *image.header.bmp* (ebenfalls im Unterverzeichnis *files*) enthält genau diese 54 Bytes. Das Ersetzen können Sie je nach Plattform auf folgende Weisen erledigen (hier für den ECB Mode gezeigt, CBC funktioniert analog):

```
java -jar .\AesTool.jar e .\files\image.bmp ECB 00112233445566778899AABBCCDDEEFF
java -jar .\AesTool.jar e .\files\image.bmp CBC 00112233445566778899AABBCCDDEEFF
```

- Linux (und Mac):
  - Mit Hilfe eines Hex Editors für Linux (resp. Mac), z.B. bless.
  - Empfohlen: Mit Bordmitteln mittels dem Tool `dd`:

    ```
    cp image.bmp.ecb.enc image.bmp.ecb.enc.bmp

    dd conv=notrunc if=image.header.bmp of=image.bmp.ecb.enc.bmp
    ```

- Windows:
  - Mit Hilfe eines Hex Editors. Z.B. mittels der freien Version des Hex Editors Neo: http://www.hhdsoftware.com/free-hex-editor
  - Mit Bordmitteln. Dann allerdings „unsauber“. Anstatt die ersten 54 Bytes der verschlüsselten Datei durch den unverschlüsselten Header zu ersetzen, hängen Sie den unverschlüsselten Header vorne an die verschlüsselte Datei an. Dies können Sie mit folgenden Aufrufen tun:

    ```
    copy /B image.header.bmp + image.bmp.ecb.enc
    image.bmp.ecb.enc.bmp
    ```

Bemerkung: Dies führt zu einer etwas „verzerrten“ Darstellung, da der verschlüsselte Header nun auch als Bildinformation interpretiert wird. Den gewünschten Effekt werden Sie aber dennoch gut erkennen können.

Betrachten Sie anschliessend die beiden so erzeugten Bitmapdateien. Was beobachten Sie? Haben Sie eine Erklärung für die Ursache Ihrer Beobachtung?

Praktikumspunkte

In diesem Praktikum können Sie **2 Praktikumspunkte** erreichen:

- Zwei Punkte erhalten Sie, wenn Sie dem Betreuer Ihre Antworten auf die Fragen in der Praktikumsanleitung zeigen und diese Antworten mehrheitlich korrekt sind. Ebenfalls müssen Sie allfällige Kontrollfragen des Betreuers richtig beantworten. Zudem müssen Sie die beiden ECB- und CBC-verschlüsselten Bilder aus der letzten Aufgabe zeigen.

Beim ECB ist das Bild einigermassen noch erkennbar, da beim ECB jeder Block mit dem gleichen Key transformiert wird, ist der Ciphertext einigermassen noch lesbar.

Beim CBC hingegen kann man nichts mehr erkennen (static), da beim CBC über IV und noch über das "chaining" eine Transformation ausgeführt wird, ist der Ciphertext ziemlich zufällig.