A SIMPLIFIED RIJNDAEL ALGORITHM AND ITS LINEAR AND DIFFERENTIAL CRYPTANALYSES

MOHAMMAD MUSA, EDWARD F. SCHAEFER, AND STEPHEN WEDIG

ABSTRACT. In this paper, we describe a simplified version of the Rijndael encryption algorithm. This version can be used in the classroom for explaining Rijndael. After presentation of the simplified version, it is much easier for students to understand the real version. This simplified version has the advantage that examples can be worked by hand. We also describe attacks on this version using both linear and differential cryptanalysis. These too can be used in the classroom as a way of explaining those kinds of attacks.

ADDRESS: Edward Schaefer, Department of Mathematics and Computer Science, Santa Clara University, Santa Clara, CA 95053, USA. eschaefer@scu.edu

KEYWORDS: Rijndael, linear cryptanalysis, differential cryptanalysis

1. Introduction

A popular symmetric-key block cipher in the United States from the mid 1970's until the present has been the Data Encryption Standard (DES). As it became apparent that computer speed improvements were making the chosen keylength insecure, people started using Triple-DES instead. Triple-DES usually involves sequentially using DES with a first key in encryption mode, followed by DES with a second key in decryption mode, followed by DES with the first key again in encryption mode. But DES was not designed with this in mind. So there ought to be more efficient algorithms with the same or higher level of security as Triple-DES. In 1997, the National Institute of Standards and Technology (NIST) solicited proposals for replacements of Triple-DES. In 2001, NIST chose Rijndael to become the Advanced Encryption Standard (AES). Rijndael is a symmetric-key block cipher designed by Joan Daemen and Vincent Rijmen (see [2]).

Though Rijndael is not inordinately complicated, it would be best understood if one could work through an example by hand. However, this is not feasible. So we have designed a simplified version of Rijndael for which it is possible to work through an example by hand. In addition, we believe that we have shrunk the parameters as much as possible without losing the essence of the algorithm. The parameters were also chosen so that the linear and differential cryptanalyses are not trivial.

Though not entirely necessary, an instructor should probably present this algorithm after a discussion of finite fields of the form GF(2^r). This entire article would convert into (at least) three lectures, based on the algorithm, the linear cryptanalysis and the differential cryptanalysis. Of course each of the latter two is optional. This algorithm is similar to the simplified Data Encryption Standard algorithm presented by the second author in [4].

In Sections 2 through 6, we describe the simplified Rijndael algorithm; it has two rounds. In Section 7, we describe the real Rijndael algorithm. In Section 8, we present a linear
cryptanalytic attack on one-round simplified Rijndael. In Section 9, we present differential cryptanalytic attacks on one-round and two-round simplified Rijndael.

2. The Finite Field

Both the key expansion and encryption algorithms depend on an S-box that itself depends on the finite field with 16 elements.

The finite field $\text{GF}(2)$ consists of the set $\{0, 1\}$ where all operations work modulo 2. We use $\text{GF}(2)[x]$ to denote polynomials with coefficients in $\text{GF}(2)$. Define the field $\text{GF}(16) = \text{GF}(2)[x]/(x^4 + x + 1)$. The field $\text{GF}(16)$ is most easily thought of as consisting of the 16 polynomials of degree less than 4 where all operations work modulo $x^4 + x + 1$. That means we have $x^4 + x + 1 = 0$ or $x^4 = x + 1$ (note addition and subtraction are the same since coefficients work modulo 2 where $-1 = 1$). It is also useful to note that $x^5 = x^2 + x$ and $x^6 = x^3 + x^2$. So in $\text{GF}(16)$, we have $(x^3 + x^2 + 1)(x^3) = x^6 + x^5 + x^3 = (x^3 + x^2) + (x^2 + x) + x^3 = x$. This is a field since $x^4 + x + 1$ is irreducible over $\text{GF}(2)[x]$. Thus we can invert all non-0 elements of $\text{GF}(16)$. This is very similar to inverting elements in a finite field of the form $\text{GF}(p)$ (the integers modulo $p$) where $p$ is a prime number. That is because the Euclidean algorithm can be applied to polynomials as well. In this version, the remainder is always of lower degree than the divisor.

Let us review inversion in the more familiar setting of $\text{GF}(229)$ (229 is prime) and then see how it works in $\text{GF}(16)$. Let us invert 37 in $\text{GF}(229)$. We first use the Euclidean algorithm to find the greatest common divisor of 37 and 229 (which is 1) and then work backwards to write 1 as an integer linear combination of 37 and 229 and reduce that equation modulo 229. We will then invert $x^3 + x^2$ in $\text{GF}(16)$; the steps are essentially identical.

$$
\begin{align*}
229 &= 6 \cdot 37 + 7 \\
37 &= 5 \cdot 7 + 2 \\
7 &= 3 \cdot 2 + 1 \\
1 &= 7 - 3 \cdot 2 \\
1 &= 7 - 3 (37 - 5 \cdot 7) \\
1 &= 16 \cdot 7 - 3 \cdot 37 \\
1 &= 16 \cdot (229 - 6 \cdot 37) - 3 \cdot 37 \\
1 &= 16 \cdot 229 - 99 \cdot 37 \\
1 \equiv 16 \cdot 229 - 99 \cdot 37 \pmod{229} \\
1 \equiv 16 \cdot 0 + 130 \cdot 37 \pmod{299} \\
37^{-1} &\equiv 130 \pmod{229}
\end{align*}
$$
The word nybble refers to a four-bit string (half a byte). We will frequently associate an element \( b_0 x^3 + b_1 x^2 + b_2 x + b_3 \) of GF(16) with the nybble \( b_0 b_1 b_2 b_3 \). This notation disagrees with that in [2]. In that book, the subscripts of bits within a byte decrease from left to right and the subscripts of bytes increase from left to right. This would hamper our notation so all of our subscripts will increase from left to right.

3. The S-box

The S-box is a non-linear, invertible map from nybbles to nybbles. Here is how it operates. First, invert the nybble in GF(16). So from above, 1100 goes to 1010. The nybble 0000 is not invertible, so at this step it is sent to itself. Then associate to the nybble \( N = b_0 b_1 b_2 b_3 \) (which is the output of the inversion) the element \( N(y) = x^3 + y^2 + b_2 y + b_3 \) in GF(2)[y]/(y^4 + 1).

Let \( a(y) = y^3 + y^2 + 1 \) and \( b(y) = y^3 + 1 \) in GF(2)[y]/(y^4 + 1). The second step of the S-box is to send the nybble \( N(y) \) to \( a(y)N(y) + b(y) \). Note that \( y^4 + 1 = (y + 1)^4 \) is reducible over GF(2) so GF(2)[y]/(y^4 + 1) is not a field and not all of its non-0 elements are invertible; the polynomial \( a(y) \), however, is. Doing multiplication and addition is similar to doing so in GF(16) except that we are working modulo \( y^4 + 1 \) so \( y^4 = 1 \). The second step can also be described by an affine matrix map. That is, however, unnecessary and less concise (the same is true for the real Rijndael).
We can represent the action of the S-box in two ways (note we do not show the intermediary output of the inversion)

\[
\begin{align*}
\text{nyb} & \quad \text{S-box(nyb)} & \quad \text{nyb} & \quad \text{S-box(nyb)} \\
0000 & 1001 & 1000 & 0110 \\
0001 & 0100 & 1001 & 0010 \\
0010 & 1010 & 1010 & 0000 \\
0011 & 1011 & 1011 & 0011 \\
0100 & 1101 & 1100 & 1100 \\
0101 & 0001 & 1101 & 1110 \\
0110 & 1000 & 1110 & 1111 \\
0111 & 0101 & 1111 & 0111
\end{align*}
\]

or

\[
\begin{bmatrix}
9 & 13 & 6 & 12 \\
4 & 1 & 2 & 14 \\
10 & 8 & 0 & 15 \\
11 & 5 & 3 & 7
\end{bmatrix}
\]

The left-hand side is most useful for doing an example by hand. For the matrix on the right, we start in the upper left corner and go down, then to the next column and go down etc. The integers 0 - 15 are associated with their 4-bit binary representations. So 0000 = 0 goes to 9 = 1001, 0001 = 1 goes to 4 = 0100, …, 0100 = 4 goes to 13 = 1101, etc.

4. KEYS

For our simplified version of Rijndael, we have a 16 bit key, which we denote \( k_0 \ldots k_{15} \). That needs to be expanded to a total of 48 keybits \( k_0 \ldots k_{47} \), where the first 16 keybits are the same as the original key. Let us describe the expansion. Let \( RC[i] = x^{i+2} \in GF(16) \). So \( RC[1] = 1000 \) and \( RC[2] = 0011 \). If \( N_0 \) and \( N_1 \) are nybbles, then we denote their concatenation by \( N_0N_1 \). Let \( RCON[i] = RC[i]0000 \) (this is a byte). These are abbreviations for round constant. We define the function \( \text{RotNyb} \) to be \( \text{RotNyb}(N_0N_1) = N_1N_0 \) and the function \( \text{SubNyb} \) to be \( \text{SubNyb}(N_0N_1) = \text{S-box}(N_0)\text{S-box}(N_1) \); these are functions from bytes to bytes. Their names are abbreviations for rotate nybble and substitute nybble. Let us define an array \( W \) whose entries are bytes. The original key fills \( W[0] \) and \( W[1] \) in order. For \( 2 \leq i \leq 5 \),

\[
\begin{align*}
\text{if } i & \equiv 0(\text{mod } 2) \quad \text{then } W[i] = W[i - 2] \oplus RCON(i/2) \oplus \text{SubNyb}(\text{RotNyb}(W[i - 1])) \\
\text{if } i & \equiv 1(\text{mod } 2) \quad \text{then } W[i] = W[i - 2] \oplus W[i - 1]
\end{align*}
\]

The bits contained in the entries of \( W \) can be denoted \( k_0 \ldots k_{47} \). For \( 0 \leq i \leq 2 \) we let \( K_i = W[i]W[i + 1] \). So \( K_0 = k_0 \ldots k_{15} \), \( K_1 = k_{16} \ldots k_{31} \) and \( K_2 = k_{32} \ldots k_{47} \). For \( i \geq 1 \), \( K_i \) is the round key used at the end of the \( i \)-th round; \( K_0 \) is used before the first round.

5. THE SIMPLIFIED RIJNDAEL ALGORITHM

The simplified Rijndael algorithm operates on 16-bit plaintexts and generates 16-bit ciphertexts, using the expanded key. The encryption algorithm consists of the composition of 8 functions applied to the plaintext: \( A_{K_2} \circ SR \circ NS \circ A_{K_1} \circ MC \circ SR \circ NS \circ A_{K_0} \) (so \( A_{K_0} \) is applied first), which will be described below. Each function operates on a state. A state consists of 4 nybbles configured as in Figure 1. The initial state consists of the plaintext as in Figure 2. The final state consists of the ciphertext as in Figure 3.
5.1. **The Function** $A_{K_i}$. The abbreviation $A_K$ stands for *add key*. The function $A_{K_i}$ consists of XORing $K_i$ with the state so that the subscripts of the bits in the state and the keybits agree modulo 16.

5.2. **The Function** $NS$. The abbreviation $NS$ stands for *nybble substitution*. The function $NS$ replaces each nybble $N_i$ in a state by $S-box(N_i)$ without changing the order of the nybbles. So it sends the state

\[
\begin{align*}
N_0 &\quad N_2 \\
N_1 &\quad N_3
\end{align*}
\]
to the state

\[
\begin{align*}
S-box(N_0) &\quad S-box(N_2) \\
S-box(N_1) &\quad S-box(N_3)
\end{align*}
\]

5.3. **The Function** $SR$. The abbreviation $SR$ stands for *shift row*. The function $SR$ takes the state

\[
\begin{align*}
N_0 &\quad N_2 \\
N_1 &\quad N_3
\end{align*}
\]
to the state

\[
\begin{align*}
N_0 &\quad N_2 \\
N_1 &\quad N_3
\end{align*}
\]

5.4. **The Function** $MC$. The abbreviation $MC$ stands for *mix column*. A column $[N_i, N_j]$ of the state is considered to be the element $N_i z + N_j$ of $GF(16)[z]/(z^2 + 1)$. As an example, if the column consists of $[N_i, N_j]$ where $N_i = 1010$ and $N_j = 1001$ then that would be $(x^3 + x)z + (x^3 + 1)$. Like before, $GF(16)[z]$ denotes polynomials in $z$ with coefficients in $GF(16)$. So $GF(16)[z]/(z^2 + 1)$ means that polynomials are considered modulo $z^2 + 1$; thus $z^2 = 1$. So representatives consist of the 162 polynomials of degree less than 2 in $z$.

The function $MC$ multiplies each column by the polynomial $c(z) = x^2 z + 1$. As an example,

\[
((x^3 + x)z + (x^3 + 1))(x^2 z + 1) = (x^5 + x^3)z^2 + (x^3 + x + x^5 + x^2)z + (x^3 + 1)
\]

\[
= (x^5 + x^3 + x^2 + x)z + (x^5 + x^3 + x^3 + 1) = (x^2 + x + x^3 + x^2 + x)z + (x^2 + x + 1)
\]

\[
= (x^3)z + (x^2 + x + 1),
\]

which goes to the column $[N_k, N_l]$ where $N_k = 1000$ and $N_l = 0111$. A computation shows that $MC$ sends a column

\[
\begin{align*}
b_0 b_1 b_2 b_3 \\
b_4 b_5 b_6 b_7
\end{align*}
\]
to

\[
\begin{align*}
b_0 \oplus b_6 &\quad b_1 \oplus b_4 \oplus b_7 &\quad b_2 \oplus b_4 \oplus b_5 &\quad b_3 \oplus b_5 \\
b_2 \oplus b_4 &\quad b_0 \oplus b_3 \oplus b_5 &\quad b_0 \oplus b_1 \oplus b_6 &\quad b_1 \oplus b_7
\end{align*}
\]

Note that $z^2 + 1 = (z + 1)^2$ is reducible over $GF(16)$ so $GF(16)[z]/(z^2 + 1)$ is not a field and not all of its non-0 elements are invertible; the polynomial $c(z)$, however, is.

5.5. **The Rounds**. The composition of functions $A_{K_i} \circ MC \circ SR \circ NS$ is considered to be the $i$th round. So this simplified algorithm has two rounds. There is an extra $A_K$ before the first round and the last round does not have a $MC$; the latter will be explained in Section 6.
6. Decryption

Note that for general functions (where the composition and inversion are possible) \((f \circ g)^{-1} = g^{-1} \circ f^{-1}\). Also, if a function composed with itself is the identity map (i.e., gets you back where you started), then it is its own inverse. This is true of each \(A_K\). Although it is true for our \(SR\), this is not true for the real \(SR\) in Rijndael, so we will not simplify the notation \(SR^{-1}\). Decryption is then by \(A_{K_0} \circ NS^{-1} \circ SR^{-1} \circ MC^{-1} \circ A_{K_0} \circ NS^{-1} \circ SR^{-1} \circ A_{K_2}\).

To accomplish \(NS^{-1}\) we multiply a nybble by \(a(y)^{-1} = y^2 + y + 1\) and add \(a(y)^{-1}b(y) = y^3 + y^2\) in \(GF[2]/(y^4 + 1)\). Then invert the nybble in \(GF(16)\). Alternately, we can simply use one of the S-box tables from Section 3 in reverse.

Since \(MC\) is multiplication by \(c(z) = x^2z + 1\), the function \(MC^{-1}\) is multiplication by \(c(z)^{-1} = xz + (x^3 + 1)\) in \(GF(16)[z]/(z^2 + 1)\).

Decryption can be simply taught as above. However, to see why there is no \(MC\) in the last round, we continue. First note that \(NS^{-1} \circ SR^{-1} = SR^{-1} \circ NS^{-1}\). Let \(St\) denote a state. We have \(MC^{-1}(A_{K_i}(St)) = c(z)^{-1}(K_i \oplus St) = c(z)^{-1}(K_i) \oplus c(z)^{-1}(St) = A_{c(z)^{-1}K_i}(MC^{-1}(St))\). So \(MC^{-1} \circ A_{K_i} = A_{c(z)^{-1}K_i} \circ MC^{-1}\). Thus decryption is also \(A_{K_0} \circ SR^{-1} \circ NS^{-1} \circ A_{c(z)^{-1}K_1} \circ MC^{-1} \circ SR^{-1} \circ NS^{-1} \circ A_{K_2}\). Notice how each kind of operation appears in exactly the same order as in encryption. For the real Rijndael, this can improve implementation. This would not be possible if \(MC\) appeared in the last round.

Here is a homework exercise. The key is 1010011100111011 and the ciphertext is 0000011100111000. Find the plaintext pair of ASCII characters (note a = 01100001, . . . , z = 01111010).

7. The Real Rijndael

For simplicity, we will describe the version of Rijndael that has a 128-bit key, operates on 128-bit plaintexts and has 10 rounds. We will mostly explain the ways in which it differs from our simplified version. Each state consists of a four-by-four grid of bytes. For a description of Rijndael with longer plaintexts or longer keys, see [2].

The finite field is \(GF(2^8) = GF(2)[x]/(x^8 + x^4 + x^3 + x + 1)\). The S-box first inverts a nybble in \(GF(2^8)\) and then multiplies it by \(a(y) = y^4 + y^3 + y^2 + y + 1\) and adds \(b(y) = y^6 + y^5 + y + 1\) in \(GF(2)[y]/(y^8 + 1)\). Note \(a(y)^{-1} = y^6 + y^5 + y\) and \(a(y)^{-1}b(y) = y^2 + 1\). In [2], the authors describe multiplication by \(a(y)\) and addition by \(b(y)\) in terms of an affine matrix map.

The real ByteSub is the obvious generalization of our \(NS\) - it replaces each byte by its image under the S-box. The real ShiftRow shifts the rows left by 0, 1, 2 and 3. So it sends the state

\[
\begin{array}{cccc}
B_0 & B_4 & B_8 & B_{12} \\
B_1 & B_5 & B_9 & B_{13} \\
B_2 & B_6 & B_{10} & B_{14} \\
B_3 & B_7 & B_{11} & B_{15}
\end{array}
\]

to the state

\[
\begin{array}{cccc}
B_0 & B_4 & B_8 & B_{12} \\
B_5 & B_9 & B_{13} & B_1 \\
B_{10} & B_{14} & B_2 & B_6 \\
B_{15} & B_3 & B_7 & B_{11}
\end{array}
\]

The real MixColumn multiplies a column by \(c(z) = (x+1)z^3 + z^2 + z + x\) in \(GF(2^8)[z]/(z^4 + 1)\). Also \(c(z)^{-1} = (x^3 + x + 1)z^3 + (x^3 + x^2 + 1)z^2 + (x^3 + 1)z + (x^3 + x^2 + x)\). The step \(MC\) appears in all but the last round. The real AddRoundKey is the obvious generalization of our \(A_{K_i}\). There is an additional AddRoundKey with round key 0 at the beginning of the encryption algorithm.
For key expansion, the entries of the array W are 32 bits each. The key fills in $W[0], \ldots, W[3]$. The function RotByte cyclically rotates four bytes 1 to the left each, like the action on the second row in ShiftRow. The function SubByte applies the S-box to each byte. $RC[i] = x^i$ in GF($2^8$) and $RCON[i]$ is the concatenation of $RC[i]$ and 3 bytes of all 0’s. For $4 \leq i \leq 43$,

- if $i \equiv 0 \pmod{4}$ then $W[i] = W[i - 4] \oplus RCON(i/4) \oplus \text{SubByte}($RotByte$(W[i - 1]))$
- if $i \not\equiv 0 \pmod{4}$ then $W[i] = W[i - 4] \oplus W[i - 1]$.

The $i$th key $K_i$ consists of the bits contained in the entries of $W[4i] \ldots W[4i + 3]$.

8. Linear Cryptanalysis of One-Round Simplified Rijndael

Linear cryptanalysis was first described by Matsui in [3]. Let us assume that a single key has been used to encrypt many plaintexts and that Eve (an eavesdropper) has access to many matched plaintexts and ciphertexts from this key. Eve wants to determine this key. The idea of linear cryptanalysis is to find equations of the form

$$b \oplus \sum_{i \in S_1} p_i \oplus \sum_{j \in S_2} c_j = \sum_{l \in S_3} k_l$$

which hold with probability greater than .5; the greater the better. Here $b$ is the bit 0 or 1, $p_i$ denotes the $i$th plaintext bit, $c_j$ denotes the $j$th ciphertext bit, $k_l$ denotes the $l$th keybit and each $S_m$ is a subset of \{0, \ldots, 15\}. For each such equation, Eve evaluates the left-hand side over every matched plaintext and ciphertext. If she gets 0 more often than 1, then she assumes $\sum_{l \in S_3} k_l = 0$ and vice versa. If she can get $n$ different relations on keybits which are, in a sense described below, linearly independent, then this leaves $2^{16-n}$ possible keys. Eve can try each of these keys and see which sends the given plaintexts to the corresponding ciphertexts.

Since linear cryptanalysis requires that Eve have matched plaintexts and ciphertexts it is called a known plaintext attack. Ironically, linear cryptanalysis exploits the non-linearity of a cryptosystem. As an example, consider the linear function $x \oplus y$ on the GF(2)-vector space of dimension 2, which consists of $(x,y) = (0,0), (0,1), (1,0)$ and $(1,1)$. Note $x \oplus y = 0$ half the time and $x \oplus y = 1$ half the time. A straightforward application of linear algebra over GF(2) shows that this will occur for all linear functions over any finite-dimensional GF(2)-vector space. For comparison, the non-linear function $xy$ is 0 three-quarters of the time.

For simplicity, we will use one-round simplified Rijndael to explain linear cryptanalysis. The one round will be $A_{K_1} \circ MC \circ SR \circ NS \circ A_{K_0}$. The attack on one-round simplified Rijndael is sufficient for giving a class a reasonable understanding of linear cryptanalysis.

8.1. S-box Equations. For unsimplified encryption algorithms, trying to find such equations for the entire encryption algorithm is not feasible. So we build them up from simpler equations with probability greater than .5. The only non-linear function in simplified Rijndael is the S-box. We want to find the linear equations relating input bits to output bits of the S-box which hold with the highest probabilities. For simplified Rijndael, there are 30 equations, each holding with probability .75 and none with higher probability. Here are six
of them. Let $S\text{-box}(a_0a_1a_2a_3) = b_0b_1b_2b_3$. We have

\[
\begin{align*}
\text{equation} & \quad \text{name} \\
\text{I} & \quad a_3 \oplus b_0 = 1 \\
\text{II} & \quad a_0 \oplus a_1 \oplus b_0 = 1 \\
\text{III} & \quad a_1 \oplus b_1 = 0 \\
\text{IV} & \quad a_0 \oplus a_1 \oplus a_2 \oplus a_3 \oplus b_1 = 0 \\
\text{V} & \quad a_1 \oplus a_2 \oplus b_0 \oplus b_1 = 1 \\
\text{VI} & \quad a_0 \oplus b_0 \oplus b_1 = 1
\end{align*}
\]

It is easy to check in the S-box table in Section 3, for example, that $a_3 \oplus b_0 = 1$ twelve times and $a_3 \oplus b_0 = 0$ four times. In order to come up with equations relating $p_i$'s, $c_j$'s and $k_i$'s, we will eliminate as many unknown, intermediary bits as possible and use equations that hold with probability .75. The unknown bits, which will be denoted $m_i$ and $n_i$, are the output bits of the S-boxes which occur both in the encryption algorithm as well as in the key expansion.

### 8.2. Notation for Bits

Let us give notation to the bits appearing during encryption. After $A_{K_0}$, the state is

\[
\begin{align*}
p_0 \oplus k_0 & \quad p_1 \oplus k_1 & \quad p_2 \oplus k_2 & \quad p_3 \oplus k_3 \\
p_4 \oplus k_4 & \quad p_5 \oplus k_5 & \quad p_6 \oplus k_6 & \quad p_7 \oplus k_7
\end{align*}
\]

Next, let $S\text{-box}(p_0p_1p_2k_3) = m_0m_1m_2m_3$, and so on. After $NS$ the state is then

\[
\begin{align*}
m_0m_1m_2m_3 & \quad m_4m_5m_6m_7 & \quad m_{12}m_{13}m_{14}m_{15}
\end{align*}
\]

After $SR$ the state is then

\[
\begin{align*}
m_0m_1m_2m_3 & \quad m_8m_9m_{10}m_{11} \\
m_{12}m_{13}m_{14}m_{15} & \quad m_4m_5m_6m_7
\end{align*}
\]

After $MC$ the state is then

\[
\begin{align*}
m_0 \oplus m_{14} & \quad m_1 \oplus m_{12} & \quad m_2 \oplus m_{13} & \quad m_3 \oplus m_{11} & \quad m_6 \oplus m_8 & \quad m_4 \oplus m_7 & \quad m_9 & \quad m_4 \oplus m_5 & \quad m_10 & \quad m_5 \oplus m_{11} \\
m_2 \oplus m_{12} & \quad m_0 \oplus m_3 & \quad m_0 \oplus m_1 & \quad m_{14} & \quad m_1 \oplus m_{15} & \quad m_4 \oplus m_{10} & \quad m_5 \oplus m_8 & \quad m_{11} & \quad m_6 \oplus m_8 & \quad m_9 & \quad m_7 \oplus m_9
\end{align*}
\]

Working backwards (and exploiting the fact that if $a \oplus b = c$ then $c \oplus b = a$) we see that this state is the same as

\[
\begin{align*}
c_0 \oplus k_{16} & \quad c_1 \oplus k_{17} & \quad c_2 \oplus k_{18} & \quad c_3 \oplus k_{19} & \quad c_8 \oplus k_{24} & \quad c_9 \oplus k_{25} & \quad c_{10} \oplus k_{26} & \quad c_{11} \oplus k_{27} & \quad c_4 \oplus k_{20} & \quad c_5 \oplus k_{21} & \quad c_6 \oplus k_{22} & \quad c_7 \oplus k_{23} & \quad c_{12} \oplus k_{28} & \quad c_{13} \oplus k_{29} & \quad c_{14} \oplus k_{30} & \quad c_{15} \oplus k_{31}
\end{align*}
\]

Now let us give notation to the bits appearing during key expansion. Let $S\text{-box}(k_{12}k_{13}k_{14}k_{15}) = n_0n_1n_2n_3$ and $S\text{-box}(k_{16}k_{24}k_{27}k_{11}) = n_4n_5n_6n_7$. Then $k_{16} = k_0 \oplus n_0 \oplus 1$, and for $i = 17, \ldots, 23$ we have $k_i = k_{i-16} \oplus n_{i-16}$. For $i = 24, \ldots, 31$ we have $k_i = k_{i-16} \oplus k_{i-8}$ (see Section 4).
8.3. Finding Equations. Replacing $i$ by $j + 8$ in the previous equation, we see that for $j = 16, \ldots, 23$ we have $k_{j + 8} = k_{j - 8} + k_j$, and thus $k_j \oplus k_{j + 8} = k_{j - 8}$. This observation enables us to eliminate the unknown $n_j$’s in the following way. The equality of the two states above gives us 16 equations. If we add those equations corresponding to places 0 and 8, then 1 and 9, etc., we get

\begin{align*}
  c_0 \oplus c_8 \oplus k_8 &= m_0 \oplus m_{14} \oplus m_6 \oplus m_8 \\
  c_1 \oplus c_9 \oplus k_9 &= m_1 \oplus m_{12} \oplus m_{15} \oplus m_4 \oplus m_7 \oplus m_9 \\
  c_2 \oplus c_{10} \oplus k_{10} &= m_2 \oplus m_{12} \oplus m_{13} \oplus m_4 \oplus m_5 \oplus m_{10} \\
  c_3 \oplus c_{11} \oplus k_{11} &= m_3 \oplus m_{13} \oplus m_5 \oplus m_{11} \\
  c_4 \oplus c_{12} \oplus k_{12} &= m_2 \oplus m_{12} \oplus m_4 \oplus m_{10} \\
  c_5 \oplus c_{13} \oplus k_{13} &= m_0 \oplus m_3 \oplus m_{13} \oplus m_5 \oplus m_8 \oplus m_{11} \\
  c_6 \oplus c_{14} \oplus k_{14} &= m_0 \oplus m_1 \oplus m_{14} \oplus m_6 \oplus m_8 \oplus m_9 \\
  c_7 \oplus c_{15} \oplus k_{15} &= m_1 \oplus m_{15} \oplus m_7 \oplus m_9.
\end{align*}

The right-hand side of each of those eight equations involves all four cells of the state after $N.S$. So to use these equations as they are and combine them with equations like I through VI would give us equations with probabilities very close to .5 (we will see later where this tendency toward .5 comes from when combining equations).

Instead we notice that the bits appearing on the right-hand side of four of the equations are subsets of those appearing for the other four. So if we add the equations involving $c_3$ and $c_5$, then for $c_1$ and $c_7$, then for $c_0$ and $c_6$ and for $c_2$ and $c_4$ we get

\begin{align*}
  c_3 \oplus c_5 \oplus c_{11} \oplus c_{13} \oplus k_{11} \oplus k_{13} &= m_0 \oplus m_8 \quad \text{A} \\
  c_1 \oplus c_7 \oplus c_9 \oplus c_{15} \oplus k_9 \oplus k_{15} &= m_4 \oplus m_{12} \quad \text{B} \\
  c_0 \oplus c_6 \oplus c_8 \oplus c_{14} \oplus k_8 \oplus k_{14} &= m_1 \oplus m_9 \quad \text{C} \\
  c_2 \oplus c_4 \oplus c_{10} \oplus c_{12} \oplus k_{10} \oplus k_{12} &= m_5 \oplus m_{13} \quad \text{D}.
\end{align*}

Let us consider Equation A. Within a cell, both $m_0$ and $m_8$ correspond to the bit $b_0$ in the nybble $b_0 b_1 b_2 b_3$. So we can use Equations I and II. If we use Equation I both times we get $p_3 \oplus k_3 \oplus m_0 = 1$ and $p_{11} \oplus k_{11} \oplus m_8 = 1$, each with probability .75. Thus $p_3 \oplus p_{11} \oplus k_3 \oplus k_{11} \oplus m_0 \oplus m_8 = 0$ with probability $(.75)^2 + (.25)^2 = .625$. That is because the left-hand side is 0 if both $p_3 \oplus k_3 \oplus m_0$ and $p_{11} \oplus k_{11} \oplus m_8$ are 1, which occurs with probability $(.75)^2$ or are both 0, which occurs with probability $(.25)^2$. We can rewrite the last equation as $p_3 \oplus p_{11} \oplus k_3 \oplus k_{11} = m_0 \oplus m_8$ with probability .625. Combining that with Equation A we get $p_3 \oplus p_{11} \oplus c_3 \oplus c_5 \oplus c_{11} \oplus c_{13} = k_3 \oplus k_{13}$ with probability .625. This is our first of the kind of equation needed to do linear cryptanalysis.

Let us continue using Equation A. Equation I gives us $p_3 \oplus k_3 \oplus m_0 = 1$ and Equation II gives us $p_8 \oplus k_8 \oplus p_9 \oplus k_9 \oplus m_8 = 1$, each with probability .75. Combining we get $p_3 \oplus p_8 \oplus p_9 \oplus c_3 \oplus c_5 \oplus c_{11} \oplus c_{13} = k_3 \oplus k_8 \oplus k_9 \oplus k_{11} \oplus k_{13}$ with probability .625. If we use Equation II on both we get $p_0 \oplus p_1 \oplus p_8 \oplus p_9 \oplus c_3 \oplus c_5 \oplus c_{11} \oplus c_{13} = k_0 \oplus k_1 \oplus k_8 \oplus k_9 \oplus k_{11} \oplus k_{13}$.
Using Equation II on the first and I on the second is now redundant as it gives the same information as we get by combining the former three.

For Equation B we can also use Equations I and I, I and II, and II and II. For Equation C we can use Equations III and III, III and IV, and IV and IV. For Equation D we can use Equations III and III, III and IV, and IV and IV.

If we add A and C we get $c_0 \oplus c_3 \oplus c_5 \oplus c_6 \oplus c_8 \oplus c_{11} \oplus c_{13} \oplus c_{14} \oplus k_8 \oplus k_{11} \oplus k_{13} \oplus k_{14} = m_0 \oplus m_1 \oplus m_8 \oplus m_9$. So we can use V and V. It is tempting to use VI until one notices that it is the same as II \(\oplus\) III, and hence is redundant. If we add B and D we can also use V and V.

The 14 equations are

$$
\begin{align*}
& p_3 \oplus p_{11} \oplus c_4 \oplus c_5 \oplus c_{11} \oplus c_{13} = k_3 \oplus k_{13} \\
& p_4 \oplus p_8 \oplus p_9 \oplus c_3 \oplus c_5 \oplus c_{11} \oplus c_{13} = k_4 \oplus k_5 \oplus k_9 \oplus k_{11} \oplus k_{13} \\
& p_0 \oplus p_1 \oplus p_8 \oplus p_9 \oplus c_3 \oplus c_5 \oplus c_{11} \oplus c_{13} = k_3 \oplus k_4 \oplus k_5 \oplus k_9 \oplus k_{11} \oplus k_{13} \\
& p_7 \oplus p_{15} \oplus c_1 \oplus c_7 \oplus c_9 \oplus c_{15} = k_7 \oplus k_9 \\
& p_7 \oplus p_{12} \oplus p_{13} \oplus c_1 \oplus c_7 \oplus c_9 \oplus c_{15} = k_7 \oplus k_5 \oplus k_{12} \oplus k_{13} \oplus k_{15} \\
& p_4 \oplus p_5 \oplus p_{12} \oplus p_{13} \oplus c_1 \oplus c_7 \oplus c_9 \oplus c_{15} = k_4 \oplus k_5 \oplus k_9 \oplus k_{12} \oplus k_{13} \oplus k_{15} \\
& p_1 \oplus p_9 \oplus c_0 \oplus c_6 \oplus c_8 \oplus c_{14} = k_1 \oplus k_8 \oplus k_9 \oplus k_{14} \\
& p_1 \oplus p_8 \oplus p_9 \oplus p_{10} \oplus p_{11} \oplus c_0 \oplus c_6 \oplus c_8 \oplus c_{14} = k_1 \oplus k_9 \oplus k_{10} \oplus k_{11} \oplus k_{14} \\
& p_0 \oplus p_1 \oplus p_2 \oplus p_3 \oplus p_8 \oplus p_9 \oplus p_{10} \oplus p_{11} \oplus c_0 \oplus c_6 \oplus c_8 \oplus c_{14} = k_0 \oplus k_1 \oplus k_2 \oplus k_3 \oplus k_9 \oplus k_{10} \oplus k_{11} \oplus k_{14} \\
& p_5 \oplus p_{13} \oplus c_2 \oplus c_4 \oplus c_{10} \oplus c_{12} = k_5 \oplus k_{10} \oplus k_{12} \oplus k_{13} \\
& p_5 \oplus p_{12} \oplus p_{13} \oplus p_{14} \oplus p_{15} \oplus c_2 \oplus c_4 \oplus c_{10} \oplus c_{12} = k_5 \oplus k_{10} \oplus k_{13} \oplus k_{14} \oplus k_{15} \\
& p_4 \oplus p_5 \oplus p_6 \oplus p_7 \oplus p_{12} \oplus p_{13} \oplus p_{14} \oplus p_{15} \oplus c_2 \oplus c_4 \oplus c_{10} \oplus c_{12} = k_4 \oplus k_5 \oplus k_6 \oplus k_7 \oplus k_{10} \oplus k_{13} \oplus k_{14} \oplus k_{15} \\
& p_1 \oplus p_2 \oplus p_9 \oplus p_{10} \oplus c_0 \oplus c_3 \oplus c_5 \oplus c_6 \oplus c_8 \oplus c_{11} \oplus c_{13} \oplus c_{14} = k_1 \oplus k_2 \oplus k_8 \oplus k_9 \oplus k_{10} \oplus k_{11} \oplus k_{13} \oplus k_{14} \\
& p_5 \oplus p_6 \oplus p_{13} \oplus p_{14} \oplus c_1 \oplus c_2 \oplus c_4 \oplus c_7 \oplus c_9 \oplus c_{10} \oplus c_{12} \oplus c_{15} = k_5 \oplus k_6 \oplus k_9 \oplus k_{10} \oplus k_{12} \oplus k_{13} \oplus k_{14} \oplus k_{15};
\end{align*}
$$

they each hold with probability .625.

To the right-hand side of each of the 14 equations, we can associate a vector of length 16. For example, the first is \([0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]\). The 14 associated vectors are linearly independent over GF(2). Thus there is no redundancy of information. The eight equations we omitted (by not using II then I, for example) are all dependent on these 14.

8.4. The Attack. As described above, Eve now takes the matched plaintexts and ciphertexts and evaluates the left-hand side of each of the 14 equations. For each equation, whichever resulting bit occurs more often, that is assumed to be equal to the right-hand sum of keybits. For simplicity, let us assume that Eve has gotten the 14 correct choices for those resulting bits. In other words, the bit which was most often $p_3 \oplus p_{11} \oplus c_3 \oplus c_5 \oplus c_{11} \oplus c_{13}$ is actually equal to the value of $k_3 \oplus k_{13}$ for the given key, etc. We have 14 independent equations and 16 unknowns, so there are $2^{16-14} = 4$ possible keys. If we take the 14 vectors described above and include the two vectors associated to $k_0$ and $k_4$, then we get a set of 16 linearly independent vectors (a basis). So to determine the four possible keys, Eve can make arbitrary choices for $k_0$ and $k_4$ and then use the equations to determine the other 14 bits.
A straightforward exercise in probability shows that if Eve has $n = 109$ different matched plaintexts and ciphertexts, then she can be 95% certain that all 14 of the bit choices were correct. For a certainty level of $p$ with $0 < p < 1$ (we chose $p = .95$) we have $n = 15z^2$ where $z$ is the value in a $Z$-table corresponding to $1 - \sqrt{p}$. In fact only $n = 42$ matched plaintexts and ciphertexts are needed for the certainty level to go above $p = .5$. If none of the keys works, she can get more matched plaintexts and ciphertexts. Alternately, she can try switching some of her 14 choices for the bits. Her first attempt would be to switch the one that had occurred the fewest number of times.

8.5. Speed of attack. If Eve has a single matched plaintext and ciphertext, then she can try all $2^{16} = 65536$ keys to see which sends the plaintext to the ciphertext. If more than one key works, she can check the candidates on a second matched plaintext and ciphertext. Undoubtedly only one key will work both times. This is called a pure brute force attack. On average, she expects success half way through, so the expected success occurs after $2^{15}$ encryptions.

For our linear cryptanalytic attack to succeed with 95% certainty, Eve needs to compute the values of the 14 different $\sum_{i \in S_1} p_i \oplus \sum_{j \in S_2} c_j$’s for each of the 109 matched plaintexts and ciphertexts. Then she only needs to do a brute force attack with four possible keys.

Thus this linear cryptanalytic attack seems very attractive compared to a pure brute force attack for one-round simplified Rijndael. However, when you add rounds, you have to do more additions of equations in order to eliminate unknown, intermediary bits and the probabilities associated to the equations then tend toward .5 (as we saw our probabilities go from .75 to .625). The result is that many more matched plaintexts and ciphertexts are needed in order to be fairly certain of picking the correct bit values for the $\sum_{l \in S_3} k_l$’s.

9. Differential Cryptanalysis of Simplified Rijndael

Differential cryptanalysis was first described by Biham and Shamir in [1]. This is a chosen plaintext attack. That means there is a single key and Eve (an eavesdropper) gets to choose the plaintexts that will be encrypted with that key. The key is unknown to her and she wants to determine it. Of course the matching ciphertexts are assumed to be available to Eve also. The premise of a chosen plaintext attack seems a little unrealistic at first, but there have been instances where an attacker has been able to provide plaintexts to a cryptosystem. In addition, with an enormous number of known plaintexts, some of the plaintexts that would have been chosen will appear and a chosen plaintext attack can be launched.

For differential cryptanalysis, Eve typically chooses a pair of plaintexts that differ at specified bits and are the same at the rest. She then looks at the difference in the corresponding two ciphertexts and deduces information about the key. In its simplest form, this is accomplished by finding equations of the type $S$-box$(\text{string}_1 \oplus \text{keybits}) \oplus S$-box$(\text{string}_2 \oplus \text{keybits}) = \text{string}_3$, where each string$_i$ is known and the two sets of keybits are unknown and the same (for example $k_0k_1k_2k_3$). The XOR of the inputs is called the input XOR and string$_3$ is called the output XOR. Everything in the equation is known except for the set of keybits. So Eve finds a pair of matched plaintexts and ciphertexts where the plaintexts agree and differ at the appropriate bits. She then determines the keybits that make the equation valid.
There may be several candidates. If that is the case, then she finds another pair of matched plaintexts and ciphertexts. Again she gets a set of candidate keybits. The correct keybits will be in the intersection of the two sets of candidates. With enough such pairs of matched plaintexts and ciphertexts, the intersection of these sets should just be the correct set of keybits. An example of this is given in Section 9.1 for one-round simplified Rijndael.

With enough rounds, it is impossible to find such equations as above that hold with probability $p$. So instead, equations of that type are found which hold with some lesser probability $p$. Again, for each pair of matched plaintexts and ciphertexts, Eve finds the set of candidate keybits. However, in this case, the correct keybits will appear in about $p$ of these sets. The incorrect values of the keybits should appear randomly in these sets and there are many incorrect values. So only the correct keybits will appear with a proportion around $p$ and the other possibilities will appear less frequently. That way the correct keybits can be identified. This will be described in Section 9.2 for two-round simplified Rijndael.

9.1. Differential Cryptanalysis of One-Round Simplified Rijndael. The round will be as in Section 8. We will use the notation of Section 8.2. Choose a pair of plaintexts $p_0 \ldots p_{15}$ and $p_0^* \ldots p_{15}^*$ such that $p_i = p_i^*$ for $0 \leq i \leq 7$. In addition, choose the plaintexts so that $p_0 p_1 p_2 p_3 \neq p_0^* p_1^* p_2^* p_3^*$ and $p_4 p_5 p_6 p_7 \neq p_4^* p_5^* p_6^* p_7^*$ as nybbles (so it is OK if some bits agree in each nybble). There is only one key used so $k_i = k_i^*$ for each $i$. Therefore $m_i = m_i^*$ for $0 \leq i \leq 7$.

Now we go to the end of the round and work our way back using only the first and fourth nybbles. From Section 8.2, we see that

$$c_0 c_1 c_2 c_3 \oplus c_0^* c_1^* c_2^* c_3^* = (c_0 \oplus k_{16}, \ldots, c_4 \oplus k_{19}) \oplus (c_0^* \oplus k_{16}^*, \ldots, c_3^* \oplus k_{19}^*)$$

$$= (m_0 \oplus m_{14}, \ldots, m_3 \oplus m_{13}) \oplus (m_0^* \oplus m_{14}^*, \ldots, m_3^* \oplus m_{13}^*) = m_0 m_1 m_2 m_3 \oplus m_0^* m_1^* m_2^* m_3^*$$

$$= \text{S-box}(p_0 p_1 p_2 p_3 \oplus k_0 k_1 k_2 k_3) \oplus \text{S-box}(p_0^* p_1^* p_2^* p_3^* \oplus k_0^* k_1^* k_2^* k_3^*)$$

(with parentheses and commas added for clarity). This gives us Equation I: S-box($p_0 p_1 p_2 p_3 \oplus k_0 k_1 k_2 k_3$) $\oplus$ S-box($p_0^* p_1^* p_2^* p_3^* \oplus k_0^* k_1^* k_2^* k_3^*$). Similarly, looking at the fourth nybble of ciphertext, we get Equation II: S-box($p_4 p_5 p_6 p_7 \oplus k_4 k_5 k_6 k_7$) $\oplus$ S-box($p_4^* p_5^* p_6^* p_7^* \oplus k_4^* k_5^* k_6^* k_7^*$).

Now choose a pair of plaintexts $p_0 \ldots p_{15}$ and $p_0^* \ldots p_{15}^*$ such that $p_i = p_i^*$ for $0 \leq i \leq 7$ and $p_8 p_9 p_{10} p_{11} \neq p_8^* p_9^* p_{10}^* p_{11}^*$ and $p_{12} p_{13} p_{14} p_{15} \neq p_{12}^* p_{13}^* p_{14}^* p_{15}^*$. We have $m_i = m_i^*$ for $0 \leq i \leq 7$. As above, we start at the end of the first round with the third nybble and get Equation III: S-box($p_8 p_9 p_{10} p_{11} \oplus k_8 k_9 k_{10} k_{11}$) $\oplus$ S-box($p_8^* p_9^* p_{10}^* p_{11}^* \oplus k_8^* k_9^* k_{10}^* k_{11}^*$) $= c_{0} c_{1} c_{2} c_{3} \oplus c_{0}^* c_{1}^* c_{2}^* c_{3}^*$. Similarly, looking at the second nybble we get Equation IV: S-box($p_{12} p_{13} p_{14} p_{15} \oplus k_{12} k_{13} k_{14} k_{15}$) $\oplus$ S-box($p_{12}^* p_{13}^* p_{14}^* p_{15}^* \oplus k_{12}^* k_{13}^* k_{14}^* k_{15}^*$) $= c_{4} c_{5} c_{6} c_{7} \oplus c_{4}^* c_{5}^* c_{6}^* c_{7}^*$.

As an example, let us say that Alice and Bob use the key 11011100 11101111 (we include spaces to make it easier to read). Eve does not know the key, of course. Eve encrypts the ASCII plaintext $\text{No}=01001110 01101111$ and gets 00100010 01001101. She encrypts $\text{to}=01110100 01101111$ and gets 00001010 00010011. Equation I is then S-box(0100 $\oplus$ k0k1k2k3) $\oplus$ S-box(0111 $\oplus$ k0k1k2k3) = 0010 $\oplus$ 0000 = 0010. The only strings k0k1k2k3 that satisfy this are in Set1 $= \{0100, 0111, 1101, 1110\}$. Note that the set of 16 pairs of nybbles of the form (0100 $\oplus$ k0k1k2k3, 0111 $\oplus$ k0k1k2k3) is equal to the set of 16 pairs of nybbles of the
form \((l_0l_1l_2l_3, 0011 \oplus l_0l_1l_2l_3)\) (since 0011 = 0100 \(\oplus\) 0111). So it would be a waste to pick another pair of plaintexts for which \(p_0p_1p_2p_3 \oplus p_0^*p_1^*p_2^*p_3^* = 0011\) and \(c_0c_1c_2c_3 \oplus c_0^*c_1^*c_2^*c_3^* = 0010\). This dependence only on the input and output XORs of the S-box makes it simpler to create tables giving the solutions of equations like Equation I.

The same key encrypts \(Mr = 01001101\ \ 01110010\ \ 11010101\ \ 11101000\ \ \text{and} \ \ or = 01101111\ \ 01110011\ \ 01001111\). Equation I for this pair is \(S-box(0100 \oplus k_0k_1k_2k_3) \oplus S-box(0110 \oplus k_0k_1k_2k_3) = 1101 \oplus 1100 = 0001\). The only strings \(k_0k_1k_2k_3\) that satisfy this are in \(\{111, 1111\}\). The intersection of \(Set_1\) and \(Set_2\) is 111, so that must be \(k_0k_1k_2k_3\).

From the encryptions of \(No\) and to as well as \(Mr\) and \(or\), Eve uses Equation II and gets that \(k_4k_5k_6k_7\) is in \(Set_3 = \{0110, 1100\}\) and \(Set_4 = \{1100, 1110\}\), respectively. Thus she knows \(k_4k_5k_6k_7 = 1100\).

From the encryptions of if and is as well as \(PM\) and \(Pa\), Eve uses Equation III and gets that \(k_8k_9k_{10}k_{11}\) is in \(Set_5 = \{1110, 1111\}\) and \(Set_6 = \{1100, 1110\}\), respectively. Thus she knows \(k_8k_9k_{10}k_{11} = 1110\). She uses Equation IV and gets that \(k_{12}k_{13}k_{14}k_{15}\) is in \(Set_7 = \{1010, 1111\}\) and \(Set_8 = \{0001, 0011, 1101, 1111\}\), respectively. Thus she knows \(k_{12}k_{13}k_{14}k_{15} = 1111\).

### 9.2. Differential Cryptanalysis of Two-Round Simplified Rijndael

A computation shows that for any given input XOR to the S-box (other than 0000), there are always seven different output XORs. For example, there are two pairs with XOR equal to 1000 whose output XOR is 1111. The other six pairs with XOR 1000 each have a different output XOR.

#### 9.2.1. S-box Equations

Choose a pair of plaintexts \(p_0 \ldots p_{15}\) and \(p_0^* \ldots p_{15}^*\) where \(p_0 \neq p_0^*\) but \(p_i = p_i^*\) otherwise. So \(p_0p_1p_2p_3 \oplus p_0^*p_1^*p_2^*p_3^* = 1000\). These are encrypted with the same key \(k_0 \ldots k_{15}\) resulting in ciphertexts \(c_0 \ldots c_{15}\) and \(c_0^* \ldots c_{15}^*\). We will use the notation of Section 8.2. Let \(S_0\) denote the first nybble of a state. After the first application of \(NS\) during encryption, only \(S_0\) and \(S_0^*\) will differ. We have \(m_i = m_i^*\) for \(4 \leq i \leq 15\). From the previous paragraph, we see that with probability 1/4 we have \(m_0m_1m_2m_3 \oplus m_0^*m_1^*m_2^*m_3^* = 1111\).

We denote the nybble in \(S_0\) after \(AK_1\) as \(x_0x_1x_2x_3\) = \(m_0 \oplus m_14 \oplus k_16, m_1 \oplus m_12 \oplus m_15 \oplus k_17, m_2 \oplus m_12 \oplus m_13 \oplus k_18, m_3 \oplus m_13 \oplus k_19\) (with commas added for clarity). We see \(x_0^*x_1^*x_2^*x_3^*\) in \(S_0^*\) is identical except that the \(m_i^*\)’s, for \(0 \leq i \leq 3\), may differ. The XOR of these two nybbles is also 1111 with probability 1/4. These two nybbles are the inputs of the S-box from the second \(NS\). Since \(SR\) does not affect \(S_0\) and \(AK_2\) adds the same key both times, the outputs of the S-box have the same XOR as \(c_0c_1c_2c_3 \oplus c_0^*c_1^*c_2^*c_3^*\).

This gives us Equation V: \(S-box(x_0x_1x_2x_3) \oplus S-box(x_0^*x_1^*x_2^*x_3^*) = c_0c_1c_2c_3 \oplus c_0^*c_1^*c_2^*c_3^*\). We also know \(x_0x_1x_2x_3 \oplus x_0^*x_1^*x_2^*x_3^* = 1111\) with probability 1/4. We will always assume that \(x_0x_1x_2x_3 \oplus x_0^*x_1^*x_2^*x_3^* = 1111\); we will correct 1/4 of the time. This gives us Equation VI: \(S-box(x_0x_1x_2x_3) \oplus S-box(x_0x_1x_2x_3 \oplus 1111) = c_0c_1c_2c_3 \oplus c_0^*c_1^*c_2^*c_3^*(\text{which holds with probability 1/4})\). Since the ciphertext is known, this equation will produce possibilities for \(x_0x_1x_2x_3\). A computation (related to that in the first paragraph of Section 9.2) shows that 1/4 of the time we will get four possibilities and 3/4 of the time we will get two possibilities for an average of 2.5 possibilities for \(x_0x_1x_2x_3\).

Let \(g(b_0b_1b_2b_3) = b_2, b_0 \oplus b_3, b_0 \oplus b_1, b_1, b_1\). We have

\[
x_0x_1x_2x_3 = m_0m_1m_2m_3 \oplus g(m_{12}m_{13}m_{14}m_{15}) \oplus k_{16}k_{17}k_{18}k_{19}
\]
= m_0m_1m_2m_3 \oplus g(m_1m_2m_{14}m_{15}) \oplus k_0k_1k_2k_3 \oplus n_0n_1n_2n_3 \oplus 1000 \\
= S-box(p_0p_1p_2p_3 \oplus k_0k_1k_2k_3) \oplus g(S-box(p_1p_2p_{13}p_{14}p_{15} \oplus k_{12}k_{13}k_{14}k_{15})) \oplus k_0k_1k_2k_3 \\
\oplus S-box(k_{12}k_{13}k_{14}k_{15}) \oplus 1000 \\
(see Section 8.2). Summarizing, we get Equation VII:

\[
S-box(p_0p_1p_2p_3 \oplus k_0k_1k_2k_3) \oplus g(S-box(p_1p_2p_{13}p_{14}p_{15} \oplus k_{12}k_{13}k_{14}k_{15})) \oplus k_0k_1k_2k_3 \\
\oplus S-box(k_{12}k_{13}k_{14}k_{15}) \oplus 1000 = x_0x_1x_2x_3.
\]

When we have the correct possibility for \( x_0x_1x_2x_3 \), the only unknowns in Equation VII are 8 of the keybits. A computation shows that the average number of candidates for the unknown keybyte \( k_0k_1k_2k_3k_{12}k_{13}k_{14}k_{15} \), arising from such an equation, is 16. For a fixed plaintext, the keybits determine the left-hand, and hence the right-hand side of Equation VII. Thus, the correct keybyte can not arise as a candidate for a wrong choice of \( x_0x_1x_2x_3 \).

For 3/4 of the plaintext pairs chosen, we are wrong when we assume that \( x_0x_1x_2x_3 \oplus x_0^*x_1^*x_2^*x_3^* = 1111 \). In those cases, it is not possible for the correct \( x_0x_1x_2x_3 \) to arise as a possibility (from Equation VI). Otherwise we would have from Equation V that \( S-box(x_0x_1x_2x_3) \oplus S-box(x_0x_1x_2x_3 \oplus b_0b_1b_2b_3) = c_0c_1c_2c_3 \oplus c_0^*c_1^*c_2^*c_3^* \) for \( b_0b_1b_2b_3 \neq 1111 \). But the possibilities for \( x_0x_1x_2x_3 \) satisfy Equation VI, which is \( S-box(x_0x_1x_2x_3) \oplus S-box(x_0x_1x_2x_3 \oplus 1111) = c_0c_1c_2c_3 \oplus c_0^*c_1^*c_2^*c_3^* \). Solving simultaneously we get \( S-box(x_0x_1x_2x_3 \oplus 1111) = S-box(x_0x_1x_2x_3 \oplus b_0b_1b_2b_3) \) which is impossible, since the S-box is a one-to-one function. Therefore, the correct keybyte will never arise as a candidate in these cases.

9.2.2. Probabilities. For 1/4 of the plaintext pairs chosen, we are correct when we assume that \( x_0x_1x_2x_3 \oplus x_0^*x_1^*x_2^*x_3^* = 1111 \). Now 3/4 of the time, we have two possibilities for \( x_0x_1x_2x_3 \) when we solve Equation VI. Exactly one will be correct and this case occurs with probability \( (1/4)(3/4)(1/2) = 3/32 \). In this case, the correct keybyte is guaranteed to occur as a candidate. For the incorrect choice for \( x_0x_1x_2x_3 \), the correct keybyte can not occur as a candidate. The other 1/4 of the time, we have four possibilities for \( x_0x_1x_2x_3 \). Exactly one will be correct and this case occurs with probability \( (1/4)^3 = 1/64 \). In this case, the correct keybyte is guaranteed to occur as a candidate. There will be three incorrect choices for \( x_0x_1x_2x_3 \) and so, for those choices, the correct keybyte can not occur as a candidate.

As explained earlier, for the 3/4 of the plaintext pairs chosen for which \( x_0x_1x_2x_3 \oplus x_0^*x_1^*x_2^*x_3^* \neq 1111 \), the correct keybyte can not occur as a candidate.

So among all candidate sets for the keybyte, the correct keybyte is expected to occur in 3/32 + 1/64 = 7/64 of the lists. Each incorrect keybyte will occur randomly. Since there are 256 possible keybytes and the candidate sets have average size 16, an incorrect keybyte should occur in (about) \( 16/256 = 1/16 \) of the candidate sets. We will exploit the fact that \( 7/64 > 1/16 \) in order to determine the correct keybyte.
9.2.3. The Attack and its Speed. Equation VII depends only on the key and the byte $p_0 p_1 p_2 p_3 p_{12} p_{13} p_{14} p_{15}$. So Eve chooses pairs of plaintexts that differ only at $p_0$ and such that the subsets $p_1 p_2 p_3 p_{12} p_{13} p_{14} p_{15}$ are different for different pairs. She uses 70 pairs; this number will be explained later. Encrypting these 70 plaintext pairs requires 140 steps. For the 70 plaintext pairs we expect $2^{2.5 \cdot 70} = 175$ possibilities for $x_0 x_1 x_2 x_3$ (and matching plaintext). Each determines a candidate set. To generate a candidate set, Eve first finds the 16 input/output pairs for the function $f_1(k_0 k_1 k_2 k_3) = S-box(p_0 p_1 p_2 p_3 \oplus k_0 k_1 k_2 k_3) \oplus k_0 k_1 k_2 k_3 \oplus 1000 \oplus x_0 x_1 x_2 x_3$ (see Equation VII). Then she finds the 16 input/output pairs for the function $f_2(k_{12} k_{13} k_{14} k_{15}) = g(S-box(p_{12} p_{13} p_{14} p_{15} \oplus k_{12} k_{13} k_{14} k_{15}) \oplus S-box(k_{12} k_{13} k_{14} k_{15}))$. Evaluating $f_1$ or $f_2$ is almost as bad as an encryption, so we will consider this 32 steps. A correct keybyte is one for which $f_1(k_0 k_1 k_2 k_3) = f_2(k_{12} k_{13} k_{14} k_{15})$. The inputs of both $f_1$ and $f_2$ can be sorted by output. Then it takes a trivial amount of time to tally the candidates; so we consider this to be part of the 32 steps above (as mentioned earlier, the average candidate set contains 16 candidate bytes for $k_0 k_1 k_2 k_3 k_{12} k_{13} k_{14} k_{15}$). So far we have $140 + 175 \cdot 32 = 5740$ steps.

A straightforward exercise shows that for 175 candidate sets, the probability of a given wrong byte appearing more often than the right keybyte is .0582. Since there are 255 wrong byte candidates for the keybyte, the expected ranking (in terms of frequency of appearance) of the right keybyte will be 14.84. At this point, Eve assumes the most frequently appearing candidate for the keybyte is correct and does a brute force attack on the remaining eight keybits (this typically requires a single matched plaintext and ciphertext, as explained in Section 8.5). If none of these 256 keys work, she moves to the second most frequently appearing candidate for the keybyte. On average, this will require a brute force attack using $14.84 \cdot 256 = 3799$ keys.

This gives a total of $5740 + 3799 \approx 2^{13.2}$ steps. It turns out that 70 pairs minimizes this sum. This is, of course, not much better than the pure brute force attack.

We can start again, but instead make the plaintexts differ only in one bit which is $p_4$, $p_8$ or $p_{12}$. Each leads to an equation like VII. However these equations differ in that they contain 12 or 16 unknown keybits (instead of 8). Again the correct and incorrect keybit strings will appear in 7/64 and 1/16 of the candidate sets, respectively. These appear to have the advantage that one is solving for more keybits and thus needs to brute force fewer. However, it takes so many steps to determine a candidate set (which will average $2^8$ or $2^{12}$ candidates), that an attack using one of these would take longer than a pure brute force attack.

References