A Simple Algebraic Representation of Rijndael

Niels Ferguson[1], Richard Schroeppel[2], and Doug Whiting[3]

[1] Counterpane Internet Security,
niels@ferguson.net
[2] Sandia National Laboratory[†]
rschroe@sandia.gov
[3] Hi/fn, Inc.,
dwhiting@hifn.com

Abstract. We show that there is a very straightforward closed algebraic formula for the Rijndael block cipher. This formula is highly structured and far simpler then algebraic formulations of any other block cipher we know. The security of Rijndael depends on a new and untested hardness assumption: it is computationally infeasible to solve equations of this type. The lack of research on this new assumption raises concerns over the wisdom of using Rijndael for security-critical applications.

1 Introduction

Rijndael has been selected by NIST to become the AES. In this paper we look at the algebraic structure in Rijndael. After RC6, Rijndael is the most elegant of the AES finalists. It turns out that this elegant structure also results in an elegant algebraic representation of the Rijndael cipher.

We assume that the reader is familiar with Rijndael. We will concentrate on the version with 128-bit block size and 128-bit keys, and occasionally mention the versions with larger key sizes. Unless otherwise noted all formulae and equations will be in the GF(2^8) field used by Rijndael.

2 Algebraic Formulae for Rijndael

In [DR98, section 8.5] the Rijndael designers note that the S-box can be written as an equation of the form

$$S(x) = w_8 + \sum_{d=0}^{7} w_d x^{255-2^d}$$

for certain constants $w_0,\ldots,w_8$.

[†] Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

S. Vaudenay and A. Youssef (Eds.): SAC 2001, LNCS 2259, pp. 103–111, 2001.

The first simplification that we make is to get rid of the constant w_8 in that formula. In a normal round the output of four S-boxes is multiplied by the MDS matrix and then four key bytes are added to the result. As the key addition and the MDS matrix are linear, we can replace the w_8 constant in the S-box by the addition of a suitable constant to the key bytes [MR00]. In the last round there is no MDS matrix but there is still a key addition, so the same trick works. This gives us the following formula for the S-box

$$S(x) = \sum_{d=0}^{7} w_d x^{255-2^d}$$

and we have to keep in mind that we now work with a modified key schedule where a suitable constant is added to each expanded key byte.

The next simplification is to rewrite the equation as

$$S(x) = \sum_{d=0}^{7} w_d x^{-2^d}$$

which is nearly equivalent as $x^{255} = 1$ for all x except $x = 0$. For the remainder of the paper we introduce the convention that $a/0 := 0$ for any value a in GF(2^8). This makes the new equation equivalent to the previous one.[1]

The form of this equation can be explained from the structure of the S-box. The S-box consists of an inversion in GF(2^8) with 0 mapped to 0, followed by a bit-linear function, followed by the addition of a constant. As noted earlier, we can move this last constant into the key schedule, so we will ignore it. Any bit-linear function can be expressed in GF(2^8) by a polynomial where all exponents are powers of two. The easiest way to see this is to observe that squaring in GF(2^8) is a bit-linear operation. After all, $(a+b)^2 = a^2+b^2$ in GF(2^8). Therefore any polynomial whose exponents are powers of two implements a bit-linear operation. None of these polynomials implements the same function, and the number of polynomials of this form equals the number of bit-linear functions. Therefore, any bit-linear function can be written as a polynomial with exponents that are powers of two.

2.1 One-Round Equation

We use the notation from [FKL+00] to discuss the internal values of a Rijndael encryption. Let $a_{i,j}^{(r)}$ be the byte at position (i, j) at the input of round r. As usual, state values in Rijndael are represented as a 4×4 square of bytes with the coordinates running from 0 to 3. For convenience we will assume that all coordinates are reduced modulo 4 so that for example $a_{8,4}^{(r)} = a_{0,0}^{(r)}$.

[1] Handling the case $x = 0$ correctly might not even be important, depending on the way we use these equations later. A single Rijndael encryption uses 160 S-box lookups. For random plaintexts there is a less than 50% chance that the case $x = 0$ will occur during the encryption. So even if we do not handle the case $x = 0$ well, the result still applies to more than half the plaintext/ciphertext pairs.

The first step in a normal round is to apply the S-box to each byte of the state in the ByteSub step. We get

$$s_{i,j}^{(r)} = S[a_{i,j}^{(r)}] = \sum_{d_r=0}^{7} w_{d_r} (a_{i,j}^{(r)})^{-2^{d_r}}$$

where $s_{i,j}^{(r)}$ is the state after ByteSub. The next step is the ShiftRow operation which we can write as

$$t_{i,j}^{(r)} = s_{i,i+j}^{(r)} = \sum_{d_r=0}^{7} w_{d_r} (a_{i,i+j}^{(r)})^{-2^{d_r}}$$

The third step in each round is the MixColumn. We can write this as

$$m_{i,j}^{(r)} = \sum_{e_r=0}^{3} v_{i,e_r} t_{e_r,j}^{(r)}$$

where the $v_{i,j}$ are the coefficients of the MDS matrix. Simple substitution now gives us

$$\begin{aligned} m_{i,j}^{(r)} &= \sum_{e_r=0}^{3} v_{i,e_r} \sum_{d_r=0}^{7} w_{d_r} (a_{e_r,e_r+j}^{(r)})^{-2^{d_r}} \\ &= \sum_{e_r=0}^{3} \sum_{d_r=0}^{7} w_{i,e_r,d_r} (a_{e_r,e_r+j}^{(r)})^{-2^{d_r}} \end{aligned}$$

for some suitable constants $w_{i,j,k}$. The final step of the round is the key addition, and results in the input to the next round.

$$\begin{aligned} a_{i,j}^{(r+1)} &= m_{i,j}^{(r)} + k_{i,j}^{(r)} \\ &= k_{i,j}^{(r)} + \sum_{e_r=0}^{3} \sum_{d_r=0}^{7} w_{i,e_r,d_r} (a_{e_r,e_r+j}^{(r)})^{-2^{d_r}} \end{aligned}$$

where $k_{i,j}^{(r)}$ is the round key of round r at position (i, j). We now have a fairly simple algebraic expression for a single round of Rijndael. We can write this formula in a couple of interesting ways.

$$a_{i,j}^{(r+1)} = k_{i,j}^{(r)} + \sum_{\substack{e_r \in \mathcal{E} \\ d_r \in \mathcal{D}}} w_{i,e_r,d_r} (a_{e_r,e_r+j}^{(r)})^{-2^{d_r}} \tag{1}$$

$$a_{i,j}^{(r+1)} = k_{i,j}^{(r)} + \sum_{f_r=0}^{31} w_{i,f_r} (a_{\lfloor f_r/8 \rfloor, \lfloor f_r/8 \rfloor + j}^{(r)})^{-2^{f_r}} \tag{2}$$

$$a_{i,j}^{(r+1)} = k_{i,j}^{(r)} + \sum_{f_r=0}^{31} w_{i,f_r} (a_{f_r,f_r+j}^{(r)})^{-2^{\lfloor f_r/4 \rfloor}} \tag{3}$$

Equation (1) is a more compact rewrite of the one we already had. We define $\mathcal{E} := \{0, \ldots, 3\}$ and $\mathcal{D} := \{0, \ldots, 7\}$ to get the same ranges. Equation (2) is derived by setting $f_r := 8e_r + d_r$. The $w_{i,j}$ are suitable constants. Note that we do not need to reduce f_r modulo 8 when using it as an exponent as this is done automatically. In GF(2^8) we have that for all k and all x, $x^k = x^{k \bmod 255}$. The exponent 2^{f_r} can thus be taken modulo 255, which makes the exponent 2^8 be equivalent to 2^0. In other words, only the ($f_r \bmod 8$) part of f_r can affect the result and we do not need to take the modulo ourselves. Equation (3) is derived in a similar manner by setting $f_r = 4d_r + e_r$, and requires a suitable rearrangements of the constants $w_{i,j}$. We find (1) the most elegant and will use that in the rest of the paper. However, one of the other formulae could also have been used with similar results.

Finally there is one more interesting way to rewrite equation (1).

$$a_{i,j}^{(r+1)} = k_{i,j}^{(r)} + \sum_{\substack{e_r \in \mathcal{E} \\ d_r \in \mathcal{D}}} \frac{w_{i,e_r,d_r}}{(a_{e_r,e_r+j}^{(r)})^{2^{d_r}}}$$

The reason that this is interesting becomes clear when we start to consider the formula for two or more rounds. Then $a_{e_r,e_r+j}^{(r)}$ is replaced by a formula with several terms, and it in turn is raised to an even power. As the field we are working in has characteristic 2 we can use the Freshman's Dream: $(a+b)^2 = a^2 + b^2$. This generalises to exponents that are powers of two, and thus it allows the exponent 2^{d_r} to be applied to each term individually instead of to the sum of terms. If we ever want to write out the full expression without the use of summation symbols this prevents the creation of many cross-product terms and thus keeps the size of the expression under control.

2.2 Multiple-Round Equations

Expressions for multiple rounds of Rijndael are easily derived by substitution. For simplicity we choose an actual value for r. For two rounds of Rijndael we get

$$a_{i,j}^{(3)} = k_{i,j}^{(2)} + \sum_{\substack{e_2 \in \mathcal{E} \\ d_2 \in \mathcal{D}}} \frac{w_{i,e_2,d_2}}{\left(k_{e_2,e_2+j}^{(1)} + \displaystyle\sum_{\substack{e_1 \in \mathcal{E} \\ d_1 \in \mathcal{D}}} \frac{w_{e_2,e_1,d_1}}{(a_{e_1,e_1+e_2+j}^{(1)})^{2^{d_1}}} \right)^{2^{d_2}}} \tag{4}$$

and the three-round version is

$$a_{i,j}^{(4)} = k_{i,j}^{(3)} + \sum_{\substack{e_3 \in \mathcal{E} \\ d_3 \in \mathcal{D}}} \frac{w_{i,e_3,d_3}}{\left(k_{e_3,e_3+j}^{(2)} + \displaystyle\sum_{\substack{e_2 \in \mathcal{E} \\ d_2 \in \mathcal{D}}} \frac{w_{e_3,e_2,d_2}}{\left(k_{e_2,e_2+e_3+j}^{(1)} + \displaystyle\sum_{\substack{e_1 \in \mathcal{E} \\ d_1 \in \mathcal{D}}} \frac{w_{e_2,e_1,d_1}}{(a_{e_1,e_1+e_2+e_3+j}^{(1)})^{2^{d_1}}} \right)^{2^{d_2}}} \right)^{2^{d_3}}}$$

Applying the Freshman's Dream to equation 4 gives us

$$a_{i,j}^{(3)} = k_{i,j}^{(2)} + \sum_{\substack{e_2 \in \mathcal{E} \\ d_2 \in \mathcal{D}}} \frac{w_{i,e_2,d_2}}{(k_{e_2,e_2+j}^{(1)})^{2^{d_2}} + \sum\limits_{\substack{e_1 \in \mathcal{E} \\ d_1 \in \mathcal{D}}} \frac{w_{e_2,e_1,d_1}^{2^{d_2}}}{(a_{e_1,e_1+e_2+j}^{(1)})^{2^{d_1+d_2}}}}$$

in which all exponentiations are on individual terms. This formula still looks rather complicated, but most of the complications are not essential to the structure of the formula. The subscripts get more complex the deeper into the recursion we go, but all subscripts are known and are independent of the key or plaintext. The same holds for the exponents, they are known and independent of the plaintext and key. We therefore introduce a somewhat sloppy notation which clarifies the structure. We write K for any expanded key byte, with the understanding that the exact position of that key byte in the key schedule is known to us. All constants are written as C even though they might not be all the same value. We replace the remaining subscripts and powers by a $*$. Again, each $*$ stands for a value that we can compute and that is independent of the plaintext and key. Finally, we use the fact that we can write the inputs to the first round by $a_{i,j}^{(1)} = p_{4j+i} + k_{i,j}^{(0)}$ where the p_i's are the plaintext bytes. All in all this gives us

$$a_{i,j}^{(3)} = K + \sum_{\substack{e_2 \in \mathcal{E} \\ d_2 \in \mathcal{D}}} \frac{C}{K^* + \sum\limits_{\substack{e_1 \in \mathcal{E} \\ d_1 \in \mathcal{D}}} \frac{C}{p_*^* + K^*}}$$

We can now write the five-round formula

$$a_{i,j}^{(6)} = K + \sum_{\substack{e_5 \in \mathcal{E} \\ d_5 \in \mathcal{D}}} \frac{C}{K^* + \sum\limits_{\substack{e_4 \in \mathcal{E} \\ d_4 \in \mathcal{D}}} \frac{C}{K^* + \sum\limits_{\substack{e_3 \in \mathcal{E} \\ d_3 \in \mathcal{D}}} \frac{C}{K^* + \sum\limits_{\substack{e_2 \in \mathcal{E} \\ d_2 \in \mathcal{D}}} \frac{C}{K^* + \sum\limits_{\substack{e_1 \in \mathcal{E} \\ d_1 \in \mathcal{D}}} \frac{C}{K^* + p_*^*}}}}} \tag{5}$$

Keep in mind that every K is some expanded key byte, each C is a known constant, and each $*$ is a known exponent or subscript, but that these values depend on the summation variables that enclose the symbol.

Equation 5 gives us the intermediate values in an encryption after five rounds as a function of the plaintext and the expanded key. It is possible to write a similar formula for the value after five rounds as a function of the ciphertext and the expanded key. The S-box is constructed from an inversion in GF(2^8) followed by a bit-linear function. The inverse S-box is constructed from a bit-linear function followed by an inversion. The inverse of the MixColumn operation is another MixColumn operation with different constants. The inverse cipher is thus constructed from the same components, and leads to a formula similar

to equation 5. (For simplicity we ignore the fact that there is no MixColumn operation in the last round which makes the last round simpler than all other rounds.)

As the results from equation 5 and the inverse equation must agree, we get a closed algebraic equation which consists of two formulae similar to equation 5. Alternatively we can write out the 10-round equation which will be about twice the size.

2.3 Fully Expanded Equations

If we try to write out equation 5 without summation symbols then we get a very large formula. Instead of a summation we simply write out 32 copies of the equation that we are summing, and substitute the appropriate summation variable values in each copy. As there are 5 summations, we end up with about 2^{25} individual terms of the form $C/(K^* + p_*^*)$. This formula would be too large to include here, but it would fit in the memory of a computer. Even the full 10-round formula would require only 2^{50} terms or so, which is certainly computable within the workload allowed for an attack on a 128-bit cipher. The 256-bit key version of Rijndael has 14 rounds. The expanded equation for half the cipher would have about 2^{35} terms, and the expanded formula for the full cipher about 2^{70} terms.

3 Other Ciphers

We know of no other 'serious' block cipher that has an algebraic description that is anywhere near as simple as the one for Rijndael. There are some general techniques that work for any block cipher, but these do not lead to practical attacks.

For example, any block cipher can be written as a boolean circuit, and then translated to a set of equations with one equation per boolean gate. However, this results in a system of equations and not a closed algebraic formula. It is equivalent to rewriting the problem of finding the cipher's key as an instance of SAT [Meh84], for which no efficient algorithms are known.

If one tries to rewrite the equations into a closed formula there is an explosion of terms. For example, in DES each output bit of the round function depends on 6 input bits. The boolean expressions for the S-boxes are fairly complicated [Kwa00], and each input bit will be used at least 16 times on average in the full boolean expressions for the output bits. A fully expanded boolean formula for DES therefore has at least around $16^{16} = 2^{64}$ terms, and due to the 'random' structure of the S-boxes this formula has no neat structures to take advantage of. Quite clearly this will not result in an attack that is faster than exhaustive search.

Another idea is to write the formula for the cipher in conjunctive normal form. This results in a simple formula: the entire function can be written using some constants and a few summation-type operators. Of course, the underlying

problem formulation for an attack is still SAT. Furthermore, a direct evaluation of this formula is impossible. First of all, the constants cannot be determined without the entire plaintext/ciphertext mapping. Even if the constants were known, the direct evaluation would cost in the order of 2^{b+k} steps where b is the block size and k the key size. This is obviously slower than an exhaustive search which requires 2^k steps.

4 An Algebraic Attack?

The real question is of course whether we can turn these formulae for Rijndael into an attack. At this moment we do not have an attack on Rijndael that uses this algebraic representation.

If the formula was a simple polynomial it would be trivial to solve, but this is nothing new. To make the formula a polynomial we would have to eliminate the $1/x$ function in the S-box. If we simply throw the $1/x$ away it makes the entire cipher affine, and there are easier ways of attacking an affine cipher. Using the equivalence $1/x = x^{254}$ and converting the formula to a polynomial leads to a polynomial with too many terms to be useful.

Another idea would be to write the formula as the quotient of two polynomials. Again, the number of terms grows very rapidly which makes this approach unpromising.

We feel that an algebraic attack would have to handle equations of the form of equation 5 directly. The form of this equation is similar to that of continued fractions, and can be seen as a generalisation. There is quite a lot of knowledge about "solving" continued fraction, but it is unclear to us whether that can be applied to these formulae. This is outside the area of expertise of the authors. We therefore have to leave this as an open problem: is there a way of solving for the key bytes K in equation 5 given enough plaintext/ciphertext pairs?

We can give a few observations. A fully expanded version of equation 5 has 2^{25} terms. If we ignore the fact that some of the key bytes in the formula must be equal we can write it as a formula in about 2^{25} individual key bytes. Computing the same intermediate state from the ciphertext gives us another formula of similar size, and setting the two equal gives us an equation in about 2^{26} expanded key bytes. From a purely information-theoretical standpoint this would require at least 2^{22} known plaintext/ciphertext pairs, but this is not a problem. The attack can even afford an algorithm of order $O(n^4)$ in the number of terms of the equation before the workload exceeds the 128-bit key size limit. Larger key sizes are even more advantageous to the attacker in finding an attack with complexity less than that of exhaustive key search. The 256-bit key version uses 14 rounds, so each equation for half the cipher would have about 2^{35} terms. An algebraic equation solver with a workload in the order of $O(n^7)$ in the number of terms might very well lead to an attack.

If the attack were to use an expanded formula for the full cipher it would have about 2^{50} terms. Again, the required plaintext/ciphertext pairs are not a problem, and an algorithm that is quadratic in the number of terms is good enough. For 256-bit keys an $O(n^3)$ algorithm would even be good enough.

Any algorithm to solve these equations can also use the fact that many of the expanded key bytes in the formula must be equal. After all, there are only 176 expanded key bytes overall, and all the key bytes in the formula are chosen from that set. As we know exactly which key value in the formula corresponds to which key byte in the expanded key, we can derive these additional equations. The Rijndael key schedule also introduces many linear equations between the various expanded key bytes which might be used.

Note that adding more rounds to Rijndael does not help as much as one would think. Each extra round adds a factor of 2^5 to the size of the fully-expanded equation. Compare this to other attacks where attacking an extra round very often involves guessing a full round key, which corresponds to a factor of 2^{128}.

5 Conclusions

The Rijndael cipher can be expressed in a very neat and compact algebraic formula. We know of no other cipher for which it is possible to derive an algebraic formula that is anywhere near as elegant. This implies that the security of Rijndael relies on a new computational hardness assumption: it is computationally infeasible to solve algebraic equations of this form. As this problem has not been studied, we do not know whether this is a reasonable assumption to make.

This puts us in a difficult situation. We have no attack on Rijndael that uses these formulae, but there might very well exist techniques for handling this type of formula that we are unaware of, or somebody might develop them in the next 20 years or so. This is a somewhat disingenuous argument; any cipher could be attacked in the future. Yet our experience teaches us that in cryptography it is best to be cautious. A system that uses Rijndael automatically bases its security on a new hardness assumption, whereas this new assumption can be avoided by using a different block cipher. In that light we are concerned about the use of Rijndael in security-critical applications.

Acknowledgements

We would like to thank John Kelsey, Mike Stay, and Yoshi Kohno for their helpful comments.

References

[DR98] Joan Daemen and Vincent Rijmen. AES proposal: Rijndael. In *AES Round 1 Technical Evaluation, CD-1: Documentation*. NIST, August 1998. See `http://www.esat.kuleuven.ac.be/~rijmen/rijndael/` or `http://www.nist.gov/aes`.

[FKL+00] Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Mike Stay, David Wagner, and Doug Whiting. Improved cryptanalysis of Rijndael. In Bruce Schneier, editor, *Fast Software Encryption 2000*, volume 1978 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000.

[Kwa00] Matthew Kwan. Reducing the gate count of bitslice DES. Cryptology ePrint Archive, Report 2000/051, 2000. `http://eprint.iacr.org/`.

[Meh84] Kurt Mehlhorn. *Data Structures and Algorithms 2: Graph Algorithms and NP-Completeness*. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1984.

[MR00] Sean Murphy and Matt Robshaw. New observations on Rijndael. Available from `http://www.isg.rhul.ac.uk/mrobshaw/`, August 2000. Preliminary Draft.