

Fuzzy Cryptanalysis:

Applying Fuzzy Logic to Cryptanalysis

John Washburn

Phone: (262) 238-8940

Fax: none

crypto@WashburnResearch.org

<http://www.washburnresearch.org/cryptography>

This research was self funded, but benefits from the remarkable energy and clarity of Bart Kosko; one of the great evangelists of fuzzy and multi-state logic.

Abstract

This paper discusses the extension of fuzzy logic to the discipline of cryptanalysis and cryptography. [Fuzzy logic](#)¹ is the branch of logic in which the truth value of a logical proposition (or set membership) is represented as a real value on unit interval $[0,1]$. Most cryptographic papers make little distinction between the hardware bit used in a cryptographic implementation and the analyst's knowledge of the state that hardware bit. In this paper, a fuzzy bit is defined as the rational number which is the probability the value of the underlying hardware bit is on. With this abstraction the value of a fuzzy bit represents the analyst's knowledge or lack of knowledge concerning a particular hardware bit within a cryptographic system.

Most cryptographic primitives are readily adapted to the concept of a “fuzzy bit”. These primitives include:

- bits,
- ordered vectors of bits (e.g. bytes, words, longs, etc),
- bitwise operators (e.g. XOR, AND, OR, NOT, etc.),
- logical operations (equal, less than, greater than, etc.),
- substitution (s-boxes),
- permutation,
- addition, and
- matrices

Building on the concept of a single fuzzy bit, the author created these and the other cryptographic primitives needed to apply the DES and AES algorithms using fuzzy bits throughout. This gives rise to a powerful new approach to cryptanalysis because quantitative measures of confusion, diffusion, and avalanche can be obtained. With such measurements, it is possible to quantitatively compare the cryptographic features of various algorithms.

Keywords: Fuzzy Logic, Cryptanalysis, Cryptography, Fuzzy Cryptanalysis, and Fuzzy Cryptography.

Introduction

The initial impetus for the author to apply the concepts of fuzzy logic to cryptography was to create an improved [known-plaintext attack](#)² to recover the cryptographic key. While this particular and initial application of fuzzy cryptanalysis did not work, the author believes the value of fuzzy cryptanalysis remains. The result of applying fuzzy cryptanalysis to the Data Encryption Standard (DES) algorithm was the finding that even a single bit of uncertainty among the 56-key bits created a 64-bit output of perfect uncertainty. None of the information contained in 64-bit DES input block was present in the 64-bit output block after 16 rounds. This is not true of reduced DES implementations of fewer rounds.

Fuzzy Cryptanalysis makes such precise and quantifiable statements possible. Further, because of the general nature of fuzzy bits, fuzzy cryptanalysis should prove a powerful tool for cryptographic research for evaluating and comparing the strengths of cryptographic algorithms.

Fuzzy Bits

Imagine there are 3 binary bits implemented in hardware. These 3 hardware bits can assume one of 8 discrete values: 001, 001, 010, 011, 100, 101, 110, or 111. These 8 discrete values are represented in the fuzzy logic literature as the corners of the unit cube as shown in Illustration 1.

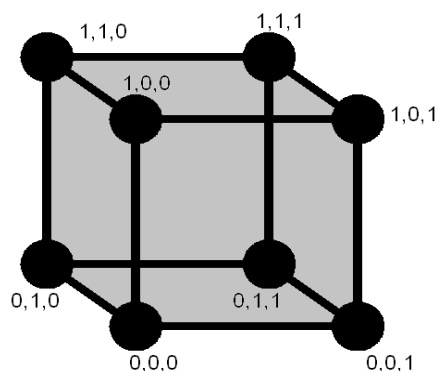


Illustration 1: 3 bits, Binary and Fuzzy

The basic unit of fuzzy cryptanalysis is the fuzzy bit. For a vector of 3 fuzzy bits, the three truth values would form coordinates within the unit cube. These coordinates represent the

analyst's knowledge of the state of the three underlying hardware bits. The fuzzy values of $(0,1,1)$ indicate the analyst is certain the first bit is off, the second bit is on and the third bit is on. The fuzzy values of $\left(0, 1, \frac{1}{2}\right)$ indicate the analyst is certain the first bit is off, is certain the second bit is on and the analysts has no knowledge if the third bit is on or off. The fuzzy values of $\left(\frac{7}{8}, \frac{3}{4}, \frac{1}{16}\right)$ indicate the 3 hardware bits are most likely $(1,1,0)$. The exact probability is $\frac{315}{512}$ that the 3 underlying hardware bits are indeed $(1,1,0)$.in the unit cube.

These coordinates represent the eight potential information items about the analyst's knowledge of the state of the three underlying hardware bits. The fuzzy values of $(0,1,1)$ indicate the analyst is certain the first bit is off, the second bit is on, and the third bit is on.

The fuzzy values of $0,1,\frac{1}{2}$ indicate the analyst is certain the first bit is off, is certain the second bit is on, and has no knowledge if the third bit is on or off. The fuzzy values of $\frac{7}{8},\frac{3}{4},\frac{1}{16}$ indicate the 3 hardware bits are most likely $(1,1,0)$. The exact probability is $\frac{315}{512}$ that the 3 underlying hardware bits are indeed $(1,1,0)$.

Fuzzy Logic

Fuzzy logic not poor reasoning, but instead is the system of logical reasoning where membership in a set is not merely yes or no. Set membership is instead part of a continuum ranging from certainly not a member of the set (Zero) and certainly a member of the set (One). The values between zero and one represent varying degrees of partial membership in the set. Fuzzy Logic is designed to handle imprecisions such as uncertainty, ambiguity, and vaguenessto certainly a member of the set (One). The values between zero and one represent varying degrees of partial membership in the set. Fuzzy Logic is designed to handle imprecisions such as uncertainty, ambiguity, and vagueness. Because of this, an axiom of predicate logic, the Law of the Excluded Middle, is not true under fuzzy logic. An item can simultaneously be both somewhat Tall and somewhat Not Tall. If I am 2 meters tall, my membership in the set, "Tall People", varies depending on whether I am among horse jockeys or NBA players.

Within the discipline of fuzzy logic there is some dispute as to on the nature of truth values. Some contend the truth value of the nature of truth values. Some contend the truth value of a fuzzy variable is strictly the probability an item is a member of a given set. This is uncertainty. Others contend concepts such as vagueness and ambiguity are well modeled by fuzzy logic and are separate and distinct from simple probabilistic uncertainty. For more information on this fascinating aspect of fuzzy logic I would refer you to the book, [*Fuzzy Thinking*](#)³ by [*Bart Kosco*](#)⁴.

Cryptography, though, admits only one form of fuzzy; probabilistic uncertainty. A hardware bit in a cryptographic system is either on or off (One or Zero). The cryptanalyst is uncertain which value is the correct value of the bit, but there is one and only one correct value. Uncertainty of this form is correctly modeled by Bayesian probability theory.

The Mathematics of a Fuzzy Bit

Thus, a fuzzy bit, for the purposes, of this paper is defined as follows:

A fuzzy bit is a rational value, ρ , on the closed interval, $[0,1]$ where ρ represents the probability the underlying hardware bit is on (has a value of one).

If a fuzzy bit has a value of 0, then the value of the underlying hardware bit is certainly off.

If a fuzzy bit has a value of 1, then the underlying hardware bit is certainly on. If a fuzzy bit

has a value of $\frac{1}{2}$ then it is equally likely the underlying hardware bit is on or off. The bit is

perfectly uncertain. If the fuzzy bit has a value of $\frac{17}{52}$, then underlying hardware bit is probably zero since there only a 1 in 3 chance the underlying hardware bit is on. .

The only operations used to calculate probabilities from other probabilities are addition, subtraction, multiplication and division. These mathematical operations are all closed on the set of rational numbers. Thus, if the initial values of the fuzzy bits are constrained to the set of rational numbers, then all subsequent values for the fuzzy bits will also be rational numbers. This initial condition is reasonably easily met in a cryptography setting. Initial values for fuzzy bits in most cryptographic applications would be one, zero, or unknown

$$\left(\rho = \frac{1}{2} \right).$$

A Simple Example

The question is how can the values of the fuzzy bits assume values other than one or zero when every bit in a cryptographic system is either on or off? The values of the fuzzy bits change and become more uncertain as the bits are transformed by a cryptographic algorithm. Illustration 2 shows a very simple encryption system. It is not a strong system, but was created for its simplicity. Illustration 2 shows a very simple encryption system. It is not a strong system, but was created for its simplicity.

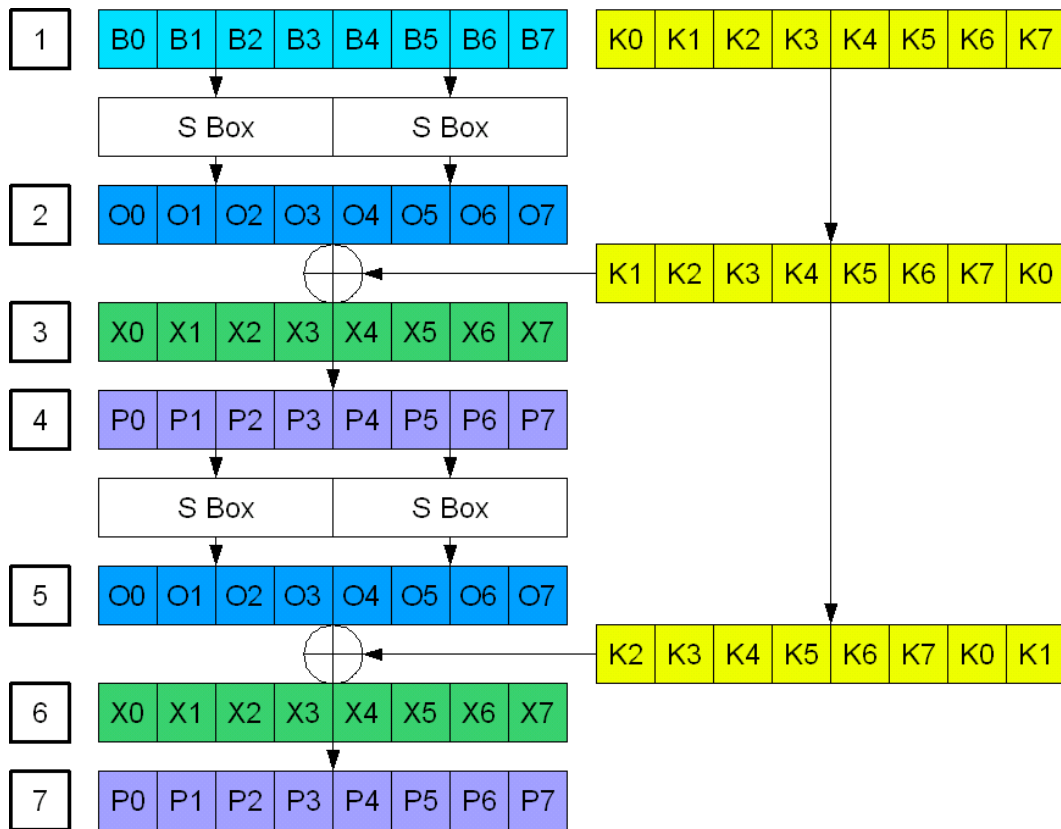


Illustration 2: Simple Encryption System

The system illustrated has two rounds, encrypts 8-bit blocks, and takes an 8-bit key. Each round consists of the following:

- A 4-bit substitution on the left and right halves of the input block
- Exclusive-or (XOR) with the 8-bit round key
- an 8-bit to 8-bit permutation.

The two round keys are generated by rotating the key right 1 bit for each round.

The 4-bit to 4-bit substitution box is defined as: 4,2,5,7,D,1,6,A,0,9,B,3,8,F,C,E

The 8-bit to 8-bit permutation is defined as: 3,5,0,4,2,1,6,7; where an input of ABCDEFGH would be permuted into an output of DFAECBGH.

With an 8-bit key of 0x57, the simple system above would encrypt the 8-bit input block, 0x4A, 0x4A as 0xB9. The round keys would be 0xAB and 0xD5 and the seven intermediate values are: 0x4A, 0xDB, 0x70, 0x4A, 0xDB, 0x9E, 0xB9.

Assume we are guessing at the key. With a guess that the least significant bit of the key is 1,

the fuzzy bits of the key would be: $\left[\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 1 \right]$. Using this key of 8 fuzzy

bits, the example encryption system would encrypt the 8-bit input block of 0x4A as

$\left[\frac{1}{2}, \frac{3}{8}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2} \right]$ with seven intermediate values of:

1. $[0, 1, 0, 0, 1, 0, 1, 0]$
2. $[1, 1, 0, 1, 1, 0, 1, 1]$
3. $\left[0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2} \right]$
4. $\left[0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2} \right]$
5. $\left[\frac{1}{4}, \frac{5}{8}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2} \right]$
6. $\left[\frac{1}{2}, \frac{3}{8}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2} \right]$
7. $\left[\frac{1}{2}, \frac{3}{8}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2} \right]$

and 2 round keys of:

1. $\left[1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2} \right]$
2. $\left[\frac{1}{2}, 1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2} \right]$

Cryptographic Primitives

The simple example system above used the following cryptographic primitives: substitution boxes, exclusive or, permutation and rotate right. These and other cryptographic primitives need to be defined so as to manipulate fuzzy bits as well as simple binary hardware bits.

Design Considerations

Each of the formulas or algorithm defined for fuzzy bits must meet the following design constraints:

1. If all the fuzzy bits are binary (fuzzy bits exactly equal 1 or exactly equal to 0), then the fuzzy result of the cryptographic primitive should also be composed of binary, fuzzy bits.
2. If all the fuzzy bits are binary, then the result of the cryptographic primitive is identical to the result obtained for the cryptographic primitive if ordinary hardware bits had been used throughout.

Appendix A describes how each of the following cryptographic primitive was defined and then implemented using Java.

- Bit
- Bitwise operators; NOT, AND, OR, and XOR
- Vector of N Bits (includes bytes, 64-bit DES blocks, AES words, etc.)
- Permutation of N bits to M bits (includes expansions and contractions as well as simple permutations.)
- Shifts and rotations
- Substitution boxes
- Logical operators (includes equal, greater than, less than, etc.)
- Addition
- Addition modulo 2^N

The one operation for which the author could find no suitable design was for multiplication in any form; multiplication, multiplication modulo 2^N or multiplication in a Galois field, $GF(2^N)$. The Java implementation has a definition for multiplication in $GF(2^8)$ for use by the AES algorithm, but this implementation is computationally expensive and not

particularly useful. The author welcomes suggestions for and improved design for fuzzy multiplication.

Conclusions

The attempt to create an improved known-text attack was not successful. The research proved useful even with this failure. It is illuminating to observe and document the introduction and spread of uncertainty within a cryptographic system. This allows the cryptanalyst to more precisely quantify the information theory concepts of confusion, diffusion, and avalanche. Such precision provides the opportunity to use fuzzy cryptanalysis to construct better cryptographic primitives such as s-boxes. It also allows for measurement of the strength or weakness of various cryptographic systems.

Also, the focus of this paper was on encryption in electronic code book mode. There seems to be no restriction to applying fuzzy cryptanalysis to other areas of cryptographic research such as other encryption modes, zero knowledge transfer protocols, blinding protocols or hashing algorithms.

Appendix A – Cryptographic Primitives

In this appendix the rational values of the probabilities are represented by Greek letters. The fuzzy variables are represented by capital roman letters. Formulas will be used to represent a fuzzy variable and its associated probability simultaneously. Thus fuzzy bit A with a probability of ρ is represented by $A(\rho)$.

Bitwise Operators

The resulting probabilities for the resulting fuzzy bit(s) for the basic bitwise and logical operators common to cryptography are given in the following table.

Operator	Boolean Formula	Probability
Bitwise Not	$B = \bar{A}$	$B(\alpha) = \bar{A}(\rho)$ $\alpha = 1 - \rho$
Bitwise AND	$C = AB$	$C(\beta) = A(\rho)B(\alpha)$ $\beta = \alpha\rho$
Bitwise OR	$C = A + B$	$C(\beta) = A(\rho) + B(\alpha)$ $\beta = \alpha + \rho - \alpha\rho$
Bitwise XOR	$C = A \oplus B$	$C(\beta) = A(\rho) \oplus B(\alpha)$ $\beta = \alpha + \rho - 2\alpha\rho$ $= \alpha(1 - \rho) + (1 - \alpha)\rho$

Ordered vectors of N bits are represented as a simple array. The index of the array is used as the bit position. The most common values for the vector lengths are 8 (bytes), 16 (shorts), 32 (longs), 48 (DES sub-keys), 64, and 128.

Logical Operators

The resulting probabilities for the resulting fuzzy bit(s) for the basic logical operators are given in the table below:

Logical Operator	Formula	Probability
Equal (One Bit)	Does bit A equal bit B?	$\beta = [A(\rho) = B(\alpha)]$ $\beta = 1 - \alpha - \rho + 2\alpha\rho$ $= \alpha\rho + (1 - \alpha)(1 - \rho)$
Not Equal (One Bit)	Does bit A not equal bit B?	$\beta = [A(\rho) \neq B(\alpha)]$ $\beta = (1 - \rho)\alpha + \rho(1 - \alpha)$ $\beta = \rho + \alpha - 2\rho\alpha$
Greater Than (One Bit)	Is bit A greater than B?	$\beta = [A(\rho) > B(\alpha)]$ $\beta = \rho(1 - \alpha)$
Less Than (One Bit)	Is bit A less than B?	$\beta = [A(\rho) < B(\alpha)]$ $\beta = (1 - \rho)\alpha$
Greater Than or equal to (One Bit)	Is bit A greater than or equal B?	$\beta = [A(\rho) \geq B(\alpha)]$ $\beta = \rho(1 - \alpha) + (1 - \rho - \alpha + 2\rho\alpha)$ $\beta = 1 - \alpha + \rho\alpha$
Less Than or equal to (One Bit)	Is bit A less than or equal B?	$\beta = [A(\rho) \leq B(\alpha)]$ $\beta = (1 - \rho)\alpha + (1 - \rho - \alpha + 2\rho\alpha)$ $\beta = 1 - \rho + \rho\alpha$
Logical Equal (Vector of N Bits)	Is one bit vector equal to a second bit vector of equal length? $A_i = B_i$	$\beta = \prod_{i=0}^{N-1} [A_i(\rho_i) = B_i(\alpha_i)]$ $\beta = \prod_{i=0}^{N-1} [1 - \alpha_i - \rho_i + 2\alpha_i\rho_i]$

Permutation, Shifts, Rotation, Expansion and Contraction

Seven common to cryptography are Permutation, Shift Left, Shift Right, Rotation Left, Rotate Right, Expansions and Contractions. All of these operations are a [mapping](#) of N bits to M bits. The number of output bits, M, may or may not be equal to the number of input bits, N. Also, as is the case with the 2 shift operators and contractions, there is a loss of information. Permutation, Rotation, and Expansions do not lose information. Permutations and rotations map N bits to N bits. Shift operators map N bits to N-1 bits with the single bit replaced by either a One or Zero. Expansions map N bits to N+k bits and Contractions map N bits to N-k bits; where $k > 0$.

Regardless of the precise details of the mapping, all mapping involves the rearrangement of bits without changing the value of the bits transported. Because of this, all these cryptographic primitives can be implemented as fuzzy operations.

Within the Java implementation of fuzzy cryptanalysis, all of the mappings discussed above are covered by slightly abusing the term permutation to include any N-bit to M-bit mapping.

Substitution (S-Boxes)

A substitution box (S-Box) selects and output of M bits based on an input of N-bit. Given a input vector of N fuzzy bits it is possible to determine the probability output of M bits based on an input of N-bit. Given an input vector of N fuzzy bits, it is possible to determine the probability that the input to the S-Box is a particular constellation of N binary bits. This probability can then be applied to scale the output of M binary bits. By enumerating all 2^N possible inputs, calculating the probability of that particular input, the resulting 2^N possible outputs (suitably scaled) can be summed to produce an M-bit fuzzy result of the S-Box.

Addition

What does it mean to add on vector of fuzzy bits to a second vector of fuzzy bits? Brute enumeration would seem to be an obvious answer, but it is computationally infeasible. It turns out though that by blindly applying the algorithm of addition to fuzzy bits the resulting answer matches the value derived from complete enumeration and can be done in a computationally efficient manner. The truth table for the addition of any 2 bits and a carry e

vector of fuzzy bits to a second vector of fuzzy bits? Brute enumeration would seem to be an obvious answer, but it is computationally infeasible. It turns out, though, that by blindly applying the algorithm of addition to fuzzy bits the resulting answer matches the value derived from complete enumeration and can be done in a computationally efficient manner. The truth table for the addition of any 2 bits and a carry-in bit is given by:

Carry In	A	B	Sum	Carry Out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

The sum column is given by the formula: $\text{Sum} = A \text{ XOR } B \text{ XOR } (\text{Carry In})$. The Carry out column is given by the formula: $\text{Carry out} = B \text{ AND } C + A \text{ XOR } C$. Using the bitwise operations (AND and XOR) are already defined for fuzzy bits, it is possible to extend the definition of single-bit sum and carry out to fuzzy bits as well. The algorithm for adding two vectors of N fuzzy bits is to set the initial value of Carry In to Zero and add the fuzzy bits from each vector pairwise from least significant to most significant. The carry out bit from the addition of bits A_{i-1} and B_{i-1} is the carry in bit for the addition of bits A_i and B_i . If the addition is modulo 2^N , then the final carry -In). The Carry-out column is given by the formula: $\text{Carry-out} = B \text{ AND } C + A \text{ XOR } C$. Using the bitwise operations, AND and XOR, already defined for fuzzy bits, it is possible to extend the definition of single-bit sum and carry out to fuzzy bits as well. The algorithm for adding two vectors of N fuzzy bits is to set the initial value of Carry-In to Zero and add the fuzzy bits from each vector pairwise from least significant to most significant. The carry-out bit from the addition of bits A_{i-1} and B_{i-1} is the carry-in bit for the addition of bits A_i and B_i . If the addition is modulo 2^N , then the final carry-out bit is ignored.

Multiplication

No method other than brute force enumeration of all multiplicands and multipliers has been found by the author for multiplication in any form (multiplication modulo 2^N or multiplication in a Galois Field $GF(2^N)$.) enumeration of all multiplicands and multipliers has been found by the author for multiplication in any form (multiplication modulo 2^N or multiplication in a Galois Field $GF(2^N)$.) JOHN, YUCKY
FORMULA AT THE END

Appendix B – Failed Bit -Picking Scheme

The general scheme was to carefully track the flow of information as it was transformed by various cryptographic primitives. In a known text attack the input and output texts are known with perfect certainty. By setting all of the bits of the key to perfect uncertainty (truth value is equal $\frac{1}{2}$), one could set and clear each bit of the key in succession. For each of the 2^N keys where only 1 bit is known with certainty, the cryptographic algorithm is applied to the perfectly known input to produce an output where each output bit has a value in the real interval $[0,1]$. The set of fuzzy bits could then be compared to the expected output. The value of this comparison is the probability the vector of fuzzy bits is equal to the expected output bits. From these comparisons, it was hoped that based on the fuzzy output which was “closest” to the expected output, a single bit of the key could then be set or cleared. Once one bit is clear the vector of fuzzy bits is equal to the expected output bits. From these comparisons, it was hoped that based on the fuzzy output which was “closest” to the expected output, a single bit of the key could then be set or cleared. Once one bit is clear, the process would be repeated where each possible second key bit is set or cleared in succession.

The author’s visualization of this was that such fuzzy cryptanalysis was similar to picking a physical lock. Cryptographic attacks designed to recover keys, operate on the whole key. The thought was to attack the cryptographic key one bit at a time much as lock picking attacks a physical lock one tumbler at a time.

The cryptographic algorithm settled on was the DES as there is much research and several successful attacks known. By applying fuzzy cryptanalysis to the DES it would be possible to compare and contrast the fuzzy cryptanalysis to other more traditional breaks in the DES algorithm

¹ http://en.wikipedia.org/wiki/Fuzzy_logic

² http://en.wikipedia.org/wiki/Known-plaintext_attack

³ http://www.amazon.com/Fuzzy-Thinking-New-Science-Logic/dp/078688021X/sr=8-1/qid=1170519560/ref=pd_bbs_1/102-4481497-2248112?ie=UTF8&s=books

⁴ <http://sipi.usc.edu/~kosko/>