ASBE Defeats Statistical Analysis and Other Cryptanalysis

By: Prem Sobel, CTO MerlinCryption, Inc.

Introduction

ASBE, Anti-Statistical Block Encryption [Ref-0], uses blocks [Ref 33, p.43-82] as part of the algorithm. These blocks are manipulated in ways different from all currently known and published existing encryption algorithms in a variable way which depends on the key.

The ASBE algorithm is designed to be exponentially stronger than existing encryption algorithms and to defeat known Statistical Analysis and Cryptanalysis. The purpose of this white paper is to describe how ASBE succeeds in these goals.

Statistical Analysis

Early encryption techniques are typically variations of substitution ciphers. Statistical analysis attempts to defeat such substitution ciphers, based on statistical knowledge of the language (for example, English) [Ref-1] [Ref-2] [Ref-3] [Ref-32, p.10-13]. These techniques include substitution ciphers such as:

- Letter frequency,
- Letter frequency of first letter of a word,
- Most likely duplicate letter pairs (such as **LL EE SS OO TT FF RR NN PP CC** in English),
- Most likely different letter pairs (in English: **TH HE AN RE ER IN ON AT ND ST ES EN OF TE ED OR TI HI AS TO**) and non-occurring letter pairs (for example, **XJ QG HZ** in English),
- Most likely letter triples and non-occurring letter triples,
- Most likely letter quadruples and non-occurring letter quadruples, and
- Most frequent words.

Clearly these statistics and patterns of multi-letter sequences are language-dependent. In the early days of cryptography, there were several limitations and/or assumptions made that are no longer valid today, due to the availability of digital communications. The following list describes several of these invalid assumptions and their limitations:

- There is no distinction made between upper and lower case (resulting in a limit of only 26 letters in English),

- Digits (0..9) are not used except in forms of words (such as **FOUR**),
- Apostrophe and underscore do not appear in words,
- No punctuation is used, for example: period is replaced by the word **STOP** (if it is used at all),
- Spaces between words are not handled properly, they are either:
  - not encrypted, or
  - shown "as is," or
  - are broken into fixed-length groups of letters (such as: five letters of cyphertext). This requires that the recipient must find the actual word boundaries after decrypting the letters.
- Other white space control, such as **TAB** and *new line* are not used in messages and are not expected in cyphertext. (The single character **LF** in some operating systems can be replaced by the two-character sequence **CR LF** in other operating systems).

There is no reason for any of these assumed limitations to be true today. These invalid assumptions and their limitations complicate the statistics [Ref-35].

The "subject" of the encrypted text can also significantly alter the statistics. Taking a large enough sample of text can produce statistics which are more generic [Ref-4] [Ref-5] [Ref-6] [Ref-7]. In addition, there are specialty languages, for example, it is well known that the statistics for characters used in different programming languages are different from each other and from natural language.

How the characters are encoded is significant. The most common encoding is ASCII which uses one byte for each character. An older encoding is EBCIDIC. A more modern encoding is UNICODE [Ref-8] which has a variable number of bytes per character and allows for multiple, natural languages in one message.

ASCII uses 8 bits per character. Typically, the most significant bit is always 0, and some of the remaining 128 byte values are not used. **Table 1** is a table of all byte values including the ASCII character codes, showing:

- Hexadecimal, for example 0x01, when the character is never used,
- The character glyph for the printable characters, and
- A two- or three-letter code abbreviation (with grey background for some of the white space and special characters), such as: ***TAB***, (0x09 in hexadecimal).

Table 1: Byte Values and Corresponding Characters

0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	TAB	LF	0x0B	0x0C	CR	0x0E	0x0F
0x10	0x11	0x12	0x13	0x14	0x15	0x16	0x17	0x18	0x19	0x1A	ESC	0x1C	0x1D	0x1E	0x1F
SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	0x7F
0x80	0x81	0x82	0x83	0x84	0x85	0x86	0x87	0x88	0x89	0x8A	0x8B	0x8C	0x8D	0x8E	0x8F
0x90	0x91	0x92	0x93	0x94	0x95	0x96	0x97	0x98	0x99	0x9A	0x9B	0x9C	0x9D	0x9E	0x9F
0xA0	0xA1	0xA2	0xA3	0xA4	0xA5	0xA6	0xA7	0xA8	0xA9	0xAA	0xAB	0xAc	0xAD	0xAE	0xAF
0xB0	0xB1	0xB2	0xB3	0xB4	0xB5	0xB6	0xB7	0xB8	0xB9	0xBA	0xBB	0xBC	0xBD	0xBE	0xBF
0xC0	0xC1	0xC2	0xC3	0xC4	0xC5	0xC6	0xC7	0xC8	0xC9	0xCA	0xCB	0xCC	0xCD	0xCE	0xCF
0xD0	0xD1	0xD2	0xD3	0xD4	0xD5	0xD6	0xD7	0xD8	0xD9	0xDA	0xDB	0xDC	0xDD	0xDE	0xDF
0xE0	0xE1	0xE2	0xE3	0xE4	0xE5	0xE6	0xE7	0xE8	0xE9	0xEA	0xEB	0xEC	0xED	0xEE	0xEF
0xF0	0xF1	0xF2	0xF3	0xF4	0xF5	0xF6	0xF8	0xF9	0xFA	0xFB	0xFC	0xFC	0xFD	0xFE	0xFF

Using the same layout for the position of byte values, a sample text file was read by the free **MerlinCryption®** program named **MerlinFileView™**. The resulting histogram of the byte values, whose entries are shown as percentages, are displayed in **Table 2**.

Table 2: Histogram of Byte Values of a Sample Text File

0	0	0	0	0	0	0	0	0	0	0.68	0	0	0.68	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16.66	0	0	0	0	0	0	0.03	0.31	0.31	0	0	0.55	0.12	0.62	0
0.03	0	0	0	0	0	0	0	0	0	0.12	0	0	0	0	0.22
0	0.09	0.03	0.06	0	0.03	0.03	0	0	0.25	0	0	0	0.03	0.09	0
0	0	0.06	0.03	0.18	0	0.03	0	0	0	0	0	0	0	0	0
0	6.38	0.92	3.02	1.54	10.66	2.19	0.99	3.79	6.38	0.06	0.28	3.17	1.36	6.38	5.82
0.99	0.28	4.07	5.27	9.02	2.13	0.92	1.26	0.37	1.51	0.03	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The byte values which do not occur are show as the single value **0**. As is expected, the most frequent letter is lower case **e**, 10.66%, yet its frequency is exceeded by the space character **SP** with a frequency of 16.66%. Upper case **E** occurs with a frequency of only 0.03%, which is not too surprising. What is important to recognize is, the two characters **e** and **E** are distinct, and their frequency of occurrence is different.

In **Table 2** there are no **TAB** or **ESC** characters in the encrypted text, but the **CR LF** sequence occurs with significant frequency, 0.68%. This frequency of occurrence is greater than: some of the letters, many of the punctuation or special characters (such as: **, - + = ? /**), and all of the digits. As might be expected in the vast majority of cases, the frequency for these two bytes (**CR LF**) is exactly equal (when using character counts, rather than a rounded-off percentage).

The same text file used to generate the histogram of byte values in **Table 2** was encrypted using **MerlinEZ®** and a particular **CryptoFile®** key-in-a-file, which in this case is a picture.

The MerlinCryption encryption technology uses a “CryptoFile” key-in-a-file instead of a single (string of characters as a) key. A CryptoFile (key-in-a-file) can be a file of any type, including pure random data. A CryptoFile contains many possible keys within it, where the number of possible keys is proportional to the square of the size of the CryptoFile.

The histogram of the byte values of the encrypted file, using a picture as a CryptoFile (key-in-a-file), is shown in **Table 3**.

Table 3: Encrypted Sample Text File Histogram Using a Specific CryptoFile and Key

0.40	0.37	0.37	0.23	0.46	0.46	0.60	0.52	0.34	0.37	0.43	0.29	0.43	0.40	0.32	0.49
0.34	0.43	0.34	0.32	0.40	0.23	0.52	0.32	0.46	0.43	0.46	0.40	0.46	0.52	0.32	0.43
0.26	0.26	0.34	0.54	0.46	0.40	0.37	0.54	0.37	0.63	0.29	0.37	0.37	0.40	0.32	0.46
0.40	0.43	0.32	0.43	0.43	0.32	0.43	0.40	0.46	0.34	0.23	0.17	0.46	0.49	0.52	0.23
0.49	0.46	0.34	0.34	0.29	0.37	0.54	0.29	0.26	0.54	0.52	0.40	0.46	0.46	0.26	0.49
0.40	0.26	0.29	0.46	0.34	0.34	0.46	0.43	0.43	0.20	0.43	0.46	0.11	0.49	0.37	0.37
0.60	0.40	0.34	0.43	0.37	0.37	0.32	0.46	0.29	0.23	0.32	0.46	0.46	0.32	0.43	0.43
0.57	0.54	0.37	0.37	0.43	0.40	0.54	0.43	0.34	0.49	0.46	0.57	0.40	0.49	0.34	0.57
0.49	0.29	0.34	0.54	0.49	0.60	0.32	0.37	0.40	0.34	0.43	0.34	0.52	0.26	0.49	0.52
0.46	0.17	0.34	0.63	0.37	0.29	0.46	0.40	0.34	0.20	0.29	0.40	0.26	0.37	0.34	0.32
0.32	0.46	0.23	0.32	0.34	0.46	0.60	0.34	0.34	0.40	0.37	0.37	0.54	0.29	0.34	0.32
0.26	0.23	0.29	0.40	0.52	0.34	0.23	0.34	0.37	0.17	0.26	0.43	0.37	0.34	0.26	0.29
0.23	0.37	0.34	0.37	0.80	0.49	0.40	0.54	0.43	0.54	0.52	0.32	0.49	0.37	0.32	0.34
0.52	0.54	0.40	0.26	0.11	0.40	0.37	0.32	0.37	0.26	0.40	0.34	0.52	0.49	0.49	0.43
0.37	0.40	0.20	0.17	0.57	0.40	0.34	0.37	0.60	0.37	0.29	0.43	0.23	0.46	0.37	0.29
0.54	0.37	0.26	0.34	0.29	0.37	0.32	0.43	0.34	0.32	0.34	0.57	0.46	0.63	0.40	0.40

The exact same text file using the exact same **CryptoFile** and key was again encrypted using **MerlinEZ**. As a distinct benefit of ASBE encryption, this encrypted file differs by nearly 100% from the previous encrypted file, yet will decrypt to the same plain text file. The histogram displayed in **Table 4** shows that the sample text file, encrypted this second time, is noticeably different.

Table 4: Encrypted Sample Text File Histogram with the Same CryptoFile and Key – Encrypted a Second Time

1.38	0.58	0.41	0.28	0.44	0.39	0.25	0.66	0.39	0.30	0.25	0.36	0.22	0.25	0.44	0.52
0.47	0.50	0.25	0.30	0.30	0.55	0.25	0.36	0.17	0.28	0.28	0.47	0.47	0.64	0.55	0.58
0.36	0.30	0.36	0.30	0.41	0.36	0.30	0.28	0.33	0.17	0.33	0.36	0.28	0.22	0.39	0.25
0.22	0.44	0.22	0.19	0.30	0.30	0.17	0.33	0.28	0.36	0.33	0.30	0.39	0.41	0.14	0.33
0.69	0.52	0.39	0.33	0.50	0.39	0.30	0.39	0.44	0.36	0.36	0.30	0.19	0.50	0.28	0.36
0.47	0.47	0.30	0.25	0.86	6.08	0.41	0.64	0.52	0.25	0.50	0.22	0.33	0.50	0.28	0.44
0.22	0.33	0.25	0.28	0.36	0.44	0.41	0.41	0.06	0.22	0.30	0.33	0.39	0.19	0.47	0.25
0.33	0.30	0.39	0.61	0.28	0.41	0.47	0.39	0.41	0.33	0.36	0.41	0.47	0.52	0.36	0.47
0.50	0.41	0.36	0.33	0.39	0.17	0.30	0.28	0.47	0.19	0.33	0.41	0.41	0.33	0.44	0.50
0.33	0.30	0.19	0.17	0.19	0.39	0.25	0.17	0.36	0.39	0.36	0.19	0.22	0.28	0.30	0.39
0.22	0.36	0.64	0.19	0.33	0.30	0.28	0.41	0.28	0.28	1.33	0.44	0.30	0.08	0.41	0.44
0.36	0.30	0.25	0.11	0.30	0.30	0.36	0.33	0.52	0.52	0.66	0.41	0.39	0.28	0.39	0.33
0.36	0.28	0.30	0.28	0.17	0.41	0.22	0.39	0.22	0.36	0.33	0.39	0.39	0.30	0.36	0.30
0.25	0.30	0.33	0.28	0.47	0.72	0.33	0.39	0.44	0.55	0.22	0.22	0.50	0.44	0.25	0.44
0.52	0.30	0.36	0.39	0.28	0.28	0.33	0.33	0.25	0.47	0.47	0.30	0.11	0.47	0.39	0.30
0.47	0.33	0.28	0.17	0.19	0.55	0.30	0.33	0.33	0.41	0.25	0.52	0.44	0.41	0.36	1.99

The histograms in **Table 3** and **Table 4** of the encrypted sample text file use the same picture as **CryptoFile** and key. The encrypted file is different every time, even with identical input, and depends on the **CryptoFile** and key used.

The following histogram in **Table 5** results from encrypting the same text file using **MerlinEZ** with a *different* picture used as a **CryptoFile** (key-in-a-file) and key.

Table 5: Encrypted Sample Text File Histogram Using a Different Picture as CryptoFile and Key

1.38	0.64	0.41	0.30	0.44	0.39	0.22	0.75	0.39	0.30	0.33	0.39	0.28	0.22	0.41	0.50
0.52	0.44	0.25	0.33	0.36	0.55	0.28	0.36	0.22	0.36	0.25	0.55	0.50	0.58	0.52	0.61
0.39	0.30	0.33	0.33	0.39	0.39	0.25	0.19	0.39	0.11	0.36	0.41	0.28	0.14	0.33	0.19
0.25	0.41	0.30	0.19	0.36	0.25	0.25	0.41	0.28	0.36	0.28	0.33	0.44	0.41	0.17	0.36
0.66	0.50	0.30	0.22	0.44	0.44	0.30	0.41	0.41	0.39	0.30	0.30	0.19	0.44	0.25	0.33
0.50	0.41	0.33	0.28	0.86	6.08	0.44	0.64	0.47	0.28	0.39	0.22	0.41	0.39	0.22	0.39
0.19	0.36	0.22	0.25	0.39	0.47	0.39	0.39	0.08	0.17	0.30	0.39	0.36	0.25	0.44	0.22
0.33	0.36	0.55	0.41	0.22	0.39	0.41	0.44	0.39	0.25	0.30	0.41	0.39	0.61	0.39	0.47
0.44	0.39	0.41	0.33	0.41	0.11	0.41	0.28	0.44	0.25	0.30	0.39	0.44	0.39	0.44	0.39
0.39	0.30	0.19	0.25	0.22	0.33	0.33	0.17	0.33	0.30	0.33	0.19	0.25	0.30	0.28	0.44
0.22	0.30	0.69	0.22	0.41	0.28	0.22	0.39	0.33	0.33	1.30	0.39	0.25	0.08	0.41	0.50
0.39	0.28	0.25	0.19	0.22	0.33	0.36	0.36	0.50	0.52	0.66	0.39	0.39	0.28	0.44	0.30
0.33	0.30	0.19	0.25	0.25	0.44	0.17	0.33	0.25	0.28	0.28	0.39	0.39	0.30	0.33	0.36
0.30	0.22	0.36	0.33	0.41	0.72	0.28	0.36	0.50	0.52	0.17	0.30	0.52	0.41	0.33	0.44
0.50	0.25	0.36	0.41	0.30	0.28	0.25	0.33	0.28	0.52	0.50	0.33	0.14	0.41	0.47	0.28
0.50	0.36	0.28	0.14	0.17	0.55	0.36	0.30	0.28	0.47	0.30	0.55	0.41	0.44	0.39	2.02

The **Table 6** histogram results from encrypting the same text file with **MerlinEZ** using a **CryptoFile** (key-in-a-file) generated by **MerlinMetaGen™**.

Table 6: Encrypted Sample Text File Histogram
Using a CryptoFile, Key-in-a-File, Produced by MerlinMetaGen

0.48	0.31	0.42	0.50	0.67	0.42	0.39	0.31	0.34	0.39	0.36	0.28	0.34	0.56	0.45	0.36
0.36	0.36	0.28	0.50	0.45	0.42	0.39	0.59	0.31	0.34	0.36	0.36	0.39	0.28	0.53	0.42
0.42	0.45	0.45	0.42	0.56	0.36	0.36	0.28	0.56	0.48	0.31	0.42	0.28	0.48	0.28	0.34
0.56	0.39	0.36	0.36	0.34	0.34	0.39	0.53	0.56	0.50	0.31	0.36	0.34	0.31	0.25	0.50
0.56	0.36	0.39	0.64	0.28	0.22	0.31	0.34	0.31	0.36	0.42	0.31	0.34	0.42	0.39	0.48
0.42	0.59	0.48	0.42	0.22	0.31	0.14	0.34	0.25	0.39	0.25	0.42	0.25	0.39	0.28	0.36
0.42	0.53	0.39	0.28	0.31	0.36	0.39	0.20	0.42	0.36	0.28	0.45	0.50	0.50	0.50	0.42
0.31	0.31	0.28	0.36	0.42	0.39	0.39	0.53	0.45	0.34	0.42	0.48	0.36	0.50	0.45	0.42
0.17	0.42	0.39	0.48	0.45	0.31	0.42	0.36	0.34	0.48	0.45	0.39	0.53	0.45	0.34	0.48
0.34	0.36	0.39	0.34	0.31	0.36	0.20	0.31	0.22	0.17	0.48	0.64	0.45	0.48	0.59	0.42
0.34	0.59	0.28	0.42	0.34	0.48	0.36	0.39	0.53	0.34	0.36	0.59	0.70	0.22	0.39	0.53
0.36	0.36	0.53	0.53	0.28	0.42	0.28	0.31	0.39	0.31	0.36	0.39	0.36	0.36	0.48	0.31
0.48	0.48	0.50	0.45	0.45	0.39	0.34	0.25	0.42	0.39	0.48	0.36	0.36	0.22	0.28	0.45
0.53	0.34	0.22	0.25	0.31	0.45	0.22	0.42	0.25	0.31	0.42	0.34	0.48	0.36	0.17	0.28
0.59	0.42	0.28	0.48	0.34	0.31	0.22	0.42	0.39	0.42	0.45	0.56	0.48	0.42	0.34	0.28
0.31	0.50	0.31	0.42	0.39	0.50	0.34	0.31	0.42	0.48	0.31	0.53	0.39	0.42	0.31	0.31

The **Table 6** histogram is, by design of the patent pending **MerlinMetaGen** algorithm, much closer to uniform than the histograms resulting from the use of a picture as a **CryptoFile** and key. This is because, among other reasons, a picture very rarely has a uniform distribution of pixel and byte values. A **MerlinFileView** histogram of the picture file would show if any byte values did not occur. Note that all byte values do occur for both pictures used above. For a perfectly uniform distribution, the average value as a percentage for each byte value is:**100/256 = 0.390625**.

Mathematical Analysis

- **RSA Encryption**, which is not symmetric, depends on factoring large numbers into a small number of primes. The efficacy of this encryption algorithm depends on the non-symmetry of the ease of multiplication vs. the much harder steps needed to factor a number. Because of steady advances in mathematics in factoring of large numbers, RSA keys up 1000 bits can relatively easily be broken [Ref-10] [Ref-11] [Ref-32, p.159-162, 255-261, 470-474] [Ref-33, p.185-206, 223-244].

- **Elliptical Curve Cryptography** [Ref-30], is not symmetric. It is based on the algebraic structure of elliptical curves. The efficacy of this encryption algorithm depends on the (unproven) assumption that finding the discrete logarithm of a random elliptic curve element with respect to a publicly known base point is infeasible. Because of continual advances in mathematics, this assumption is likely to be proven invalid, and will increasingly require larger and larger keys, as is evident for RSA [Ref-32, p.261-263, 480-481, 496-498].

- **Simple XOR and Counting Coincidences** [Ref-32, p.13-15]. Simple XOR is symmetric. This approach attempts to determine, first the length of the key, and then eliminate the key without knowing its value, to determine the plaintext. This approach does not apply to ASBE because: the key is not used in a predictably cyclical fashion.

Key Exchange Issues

One of the biggest weaknesses of many encryption algorithms is the fixed and well-defined key exchange mechanism. With ASBE, there is no fixed or defined key exchange mechanism or protocol [Ref 33, p.245-278]. In fact, it is not necessary to exchange keys when setting up or using any of the **MerlinCryption platforms**. This results in the benefit that the personnel overhead, cost to manage keys, and key lifecycles can be completely eliminated [Ref 33, p.297-346].

Cryptanalysis

Over the last decades, multiple forms of cryptanalysis [Ref-9] [Ref-32, p.5-7] have been developed to break or reverse engineer unknown encrypted messages. These include:

A.1 Boomerang attack	**A.10** Chosen-plaintext attack
A.2 Brute force attack	**A.11** Adaptive-chosen-plaintext attack
A.3 Davies' attack	**A.12** Chosen-cyphertext attack
A.4 Meet-in-the-middle attack	**A.13** Chosen-key attack
A.5 Related-key attack	**C.1** Differential cryptanalysis
A.6 Slide attack	**C.2** Impossible differential cryptanalysis
A.7 XSL attack	**C.3** Integral cryptanalysis
A.8 Cyphertext-only attack	**C.4** Linear cryptanalysis
A.9 Known-plaintext attack	**C.5** Mod-n cryptanalysis

A.1 Boomerang Attack

See section **C.1 Differential cryptanalysis**, below.

A.2 Brute Force Attack

This approach tries every possible key to decrypt an intercepted encrypted message. It is only useful for small keys. Brute force attacks use a combination of:

- a very fast computer,
- many computers, and/or
- special purpose hardware.

Examples of Brute Force Attacks include:

- DES encryption, with its 56 bit keys, is broken in a matter of days or hours by custom hardware costing about $250,000 [Ref-12] [Ref-13] [Ref-32, p.265-285]. **NIST** has since declared "DES inadequate" (among others), see [Ref-14].
- SSL encryption is broken by 112 computers in hours [Ref-15]. SSL is broken by 200 PSPs with added factor of weakness of MD5 hash [Ref-16].

Note that the age of the universe is about 13.5 billion years, which is about 4.34×10^{17} seconds [Ref-17]. The current fastest computer can execute just under 10^{15} instructions per second, and it can take many thousands to millions of instructions to try each key. This means this fastest computer could try about 10^{11} to 10^{14} keys per second, with the time doubling for each additional bit in the key. It would take this computer a few minutes to less than hour to brute force crack a 56 bit DES encrypted message.

MerlinCryption's ASBE Algorithm, with its scalable keys (with lengths of 2008 bits up to 2GB), make this kind of attack astronomically and exponentially impractical. Each additional key bit *doubles the number of possible keys*, an exponential growth.

Even the smallest ASBE key would take more than a *billion times a billion 58 times longer* to try all keys than trying to brute force guess a 256 bit key.

2008-256 = 1752 more bits, increasing the time by $2^{1752} = 10^{9*58.4}$

There are 2^{256} or about $10^{9*8.5}$ (a billion times a billion less than 9 times) possible keys of length 256 bits.

Using every available computer (over 10^{12} computers) it would still take a billion times a billion (a number of times left as an exercise for the reader) times the age of the universe to brute force guess the smallest ASBE key [Ref-32, p.153].

A.3 Davies' Attack

This attack is specific to DES encryption and does not apply to ASBE because the ASBE key length is not only larger than the DES key, but is also variable in length [Ref-18].

A.4 Meet-in-the-Middle Attack

The Meet-in-the-Middle Attack attempts to work forward from the plaintext and backwards from the cyphertext, for the purpose of "meeting in the middle" to reduce the amount of work needed to deduce the key. There is an assumption that this same key will be used again. This attack does not apply to ASBE because for any given input, (plaintext, key, and password), there is no single output cyphertext: the cyphertext is *different every time*, and yet it always decrypts correctly [Ref-19] [Ref-32, p.48-49]. In addition, there is no need to use the same key repeatedly, and with the **MerlinCryption platform,** the key can be automatically changed for every encryption.

A.5 Related-Key Attack

The Related-Key Attack attempts to use different keys with known mathematical relationships as a method to mathematically deduce a target key. This attack does not apply to ASBE because there is no predictable mathematical relationship between keys used in different encryptions that mathematically map to changes in the cyphertext [Ref-20].

- There is no predicable relationship between keys used in different encryptions.
- The length of the encrypted password is likely to be different.
- The length of encrypted data depends on the key (and other variables).

A.6 Slide Attack

The Slide Attack attempts to defeat multi-round encryption by exploiting the sub key used in each round which may be used in a repetitive way, which then renders the number of rounds irrelevant. This attack does not apply to ASBE because fixed sliding in a block is not used, and there is no inherent sub-key related to a round or a block [Ref-21].

A.7 XSL Attack

An XSL Attack attempts to find and solve quadratic simultaneous equations which describe the algorithm, and then use eXtended Sparse Linearization to solve for the key. This attack, while effective against AES encryption, does not apply to ASBE. Because of the polymorphic nature of the ASBE algorithm, there is no fixed inverse function [Ref-22].

A.8 Cyphertext-Only Attack

Given multiple encrypted messages (or ciphertexts), the goal is to deduce the plaintext, and/or the key [Ref-32, p.5-6]. This method is not effective against ASBE encrypted data or files for multiple reasons:

- Each encrypted message contains an encrypted password of unknown length.
- The length of encrypted data depends on the key (and other variables).
- Each cyphertext may be encrypted by ASBE using different custom versions of the algorithm or with different initialization vectors.

A.9 Known Plaintext Attack

In this attack, [Ref-32, p.6, 7], the cryptographic analyst has both the encrypted message and the resulting cyphertext, with the goal to deduce the key. This method is not effective against ASBE encrypted data or files for multiple reasons:

- Each encrypted message contains an encrypted password of unknown length.
- The length of encrypted data depends on the key (and other variables).
- Each cyphertext may be encrypted by ASBE using different custom versions of the algorithm or with different initialization vectors.

A.10 Chosen Plaintext Attack, and

A.11 Adaptive-Chosen Plaintext Attack

These attacks, [Ref-32, p.6], combine the known plaintext attack with the ability to choose the plaintext to encrypt and see the resulting cyphertext. This method is not effective against ASBE encrypted data or files for multiple reasons:

- Each encrypted message contains an encrypted password of unknown length.
- The length of encrypted data depends on the key (and other variables).
- Each cyphertext may be encrypted by ASBE using different custom versions of the algorithm or with different initialization vectors.

A.12 Chosen-Cyphertext Attack

In this attack, [Ref-32, p.6], the cryptographic analyst can choose different cyphertext to be decrypted and has access to resulting cyphertext, with the goal to deduce the key. This method is not effective against ASBE encrypted data or files for multiple reasons:

- Each encrypted message contains an encrypted password of unknown length.
- The length of encrypted data depends on the key (and other variables).
- Each cyphertext may be encrypted by ASBE using different custom versions of the algorithm or with different initialization vectors.

A.13 Chosen-Key Attack

In this attack, [Ref-32, p.7], the cryptographic analyst doesn't actually choose the key, but knows something about the relationship between the keys. This method is not effective against ASBE encrypted data or files for multiple reasons:

- Each encrypted message contains an encrypted password of unknown length.
- The length of encrypted data depends on the key (and other variables).
- Each cyphertext may be encrypted by ASBE using different custom versions of the algorithm or with different initialization vectors.

C.1 Differential Cryptanalysis

Differential Cryptanalysis is based on the correlation between changes in the input and the resulting changes to the output after encryption. This is done by knowledge of the transforms performed by the encryption algorithm. The skipjack encryption algorithm, [Ref-23], as well as DES encryption (among others), have been broken by differential cryptanalysis. Differential cryptanalysis, [Ref-24], and the Boomerang Attack, [Ref-25], do not apply to ASBE for multiple reasons:

- the length of the resulting cyphertext is a variable depending on the key (among other variables),
- differences in the input plaintext or key have no linear correlation with the resulting cyphertext, and
- each cyphertext may be encrypted by ASBE using different custom versions of the algorithm or with different initialization vectors.

C.2 Impossible Differential Cryptanalysis

Impossible differential cryptanalysis exploits the impossibility (that is, changes having probability 0) of differences in every stage of the encryption algorithm to improve what differential cryptanalysis predicts as possible [Ref-26]. Impossible differential cryptanalysis does not apply to ASBE for multiple reasons. One of these reasons is because every possible block value *can* occur, independent of the input.

C.3 Integral Differential Cryptanalysis

Integral cryptanalysis uses sets of plaintexts where some of the content is held constant and some of the content is varied through all possibilities, with the intent to deduce how the algorithm works. Integral differential cryptanalysis, [Ref-27], does not apply to ASBE for multiple reasons. Two reasons are because:

- every possible block value *can* occur, and
- the change of a single bit in the input plaintext, does not result in a change in the cyphertext having a fixed or predictable position.

C.4 Linear Cryptanalysis

Linear cryptanalysis [Ref-28] is based on finding affine [Ref-34] approximations to the algorithm. Linear cryptanalysis does not apply to ASBE for multiple reasons which include:

- the length of resulting cyphertext is a variable depending on the key (among other variables),
- for any given input (plaintext, key, or password) there is no single output cyphertext (it is different every time – yet always decrypts correctly), and
- there is nothing even remotely similar to an affine transformation in the algorithm between the inputs (plaintext, key, and password) and the output (cyphertext).

C.5 Mod-n Cryptanalysis

Mod-n cryptanalysis [Ref-29] exploits unevenness in how an encryption algorithm operates over congruence classes modulo n. Mod-n cryptanalysis does not apply to ASBE for multiple reasons. The change of a single bit in the input plaintext or key, does not result in a:

- change in the cyphertext having a fixed position, and a
- change in a single or fixed number of positions in the cyphertext.

The Anti-Statistical Block Encryption (ASBE) Algorithm

This white paper describes how ASBE defeats known Statistical Analysis and Cryptanalysis. The unrivaled ASBE algorithm is designed to be: exponentially stronger than other known encryption algorithms and completely unpredictable to hackers. The goal of ASBE is to keep data safe wherever it is: at rest, in change, in use, or in motion.

ASBE Algorithm on Wikipedia:
http://en.wikipedia.org/wiki/Anti-Statistical_Block_Encryption

Paul H. “Prem” Sobel on Wikipedia:
http://en.wikipedia.org/wiki/Paul_%28Prem%29_Sobel

White Paper by Prem Sobel, *A Newly Clarified Fourth State of Data*:
http://merlincryption.com/assets/papers/FourthStateOfData20111129.pdf

References:

[Ref-0] http://en.wikipedia.org/wiki/Anti-Statistical_Block_Encryption
[Ref-1] http://en.wikipedia.org/wiki/Letter_frequency
[Ref-2] http://en.wikipedia.org/wiki/Frequency_analysis
[Ref-3] http://www.letterfrequency.org/
[Ref-4] http://www.math.cornell.edu/~mec/2003-2004/cryptography/subs/frequencies.html
[Ref-5] http://mtgap.bilfo.com/letter_frequency.html
[Ref-6] http://www.cryptograms.org/letter-frequencies.php
[Ref-7] http://www.ekatetra.com/downloads/LF_Report_Chords_LetterFrequencyAnalysis.pdf
[Ref-8] http://unicode.org/
[Ref-9] http://en.wikipedia.org/wiki/Cryptanalysis
[Ref-10] http://techie-buzz.com/tech-news/1024-bit-rsa-cracked.html
[Ref-11] http://arstechnica.com/uncategorized/2007/05/researchers-307-digit-key-crak-endangers-1024-bit-rsa
[Ref-12] http://en.wikipedia.org/wiki/Brute_force_attack
[Ref-13] http://en.wikipedia.org/wiki/EFF_DES_cracker
[Ref-14] http://www.nist.gov/manuscript-publication-search.cfm?pub_id=907517
[Ref-15] http://www.marktaw.com/technology/HowlongdoesittaketocrackS.html
[Ref-16] http://hackaday.com/2008/12/30/25c3-hackers-completely-break-ssl-using-200-ps3s/
[Ref-17] http://en.wikipedia.org/wiki/Age_of_the_universe
[Ref-18] http://en.wikipedia.org/wiki/Davies'_attack
[Ref-19] http://en.wikipedia.org/wiki/Meet-in-the-middle_attack
[Ref-20] http://en.wikipedia.org/wiki/Related-key_attack
[Ref-21] http://en.wikipedia.org/wiki/Slide_attack
[Ref-22] http://en.wikipedia.org/wiki/XSL_attack
[Ref-23] http://en.wikipedia.org/wiki/Skipjack_(cipher)
[Ref-24] http://en.wikipedia.org/wiki/Differential_cryptanalysis
[Ref-25] http://en.wikipedia.org/wiki/Boomerang_attack
[Ref-26] http://en.wikipedia.org/wiki/Impossible_differential_cryptanalysis
[Ref-27] http://en.wikipedia.org/wiki/Integral_cryptanalysis
[Ref-28] http://en.wikipedia.org/wiki/Linear_cryptanalysis
[Ref-29] http://en.wikipedia.org/wiki/Mod-n_cryptanalysis
[Ref-30] http://en.wikipedia.org/wiki/Elliptic_curve_cryptography
[Ref-31] http://en.wikipedia.org/wiki/Discrete_logarithm_records#Elliptic_curves
[Ref-32] "Applied Cryptography", by Bruce Schneier, 2nd Ed. 1996, John Wiley & Sons, ISBN 047117099
[Ref-33] Practical Cryptography", by Niels Ferguson and Bruce Schneier, 2003, John Wiley & Sons, ISBN 0471223573
[Ref-34] http://en.wikipedia.org/wiki/Affine_transformation
[Ref-35] http://www.nsa.gov/public_info/_files/military_cryptanalysis/mil_crypt_I.pdf
http://www.nsa.gov/public_info/_files/military_cryptanalysis/mil_crypt_IIpdf
http://www.nsa.gov/public_info/_files/military_cryptanalysis/mil_crypt_III.pdf
http://www.nsa.gov/public_info/_files/military_cryptanalysis/mil_crypt_IV.pdf