July 18, 1995

Robert J. Miller
4471 Pacific Coast Hwy, Apt. A-308
Torrance, CA 90505

(310) 334-3267 (W)
(310) 378-4772 (H)

Commissioner of Patents and Trademarks,

I wish to file this Disclosure Document for the following:

(1a) The invention of the "vectorlite" enciphering / deciphering key table, for the use of enciphering and deciphering data.

(1b) The invention of various improvements to (1a), reference above, in form of special key table attributes. The attributes signify data elements within the key table that cause special actions to be taken by the enciphering and deciphering process.

(2) The process of "vectorlite" key table "paging", by which different "pages" of the "vectorlite" key table are brought into and out of context and usage (much like computer memory paging within virtual memory computer operating systems).

(3) The process by which to encipher data, utilizing the enciphering and deciphering key table refered to within this disclosure as the "vectorlite" key table.

(4) The invention of a process by which to obtain series of random numbers, for use by the processes of constructing the vectorlite key table and to perform the enciphering process.

Items 1a, 1b, 2, 3 and 4 further described on the following pages.

A prototype enciphering and deciphering series of programs was constructed and successfully operated on July 4, 1995 utilizing a two dimensional "vectorlite" key table and "vectorlite" enciphering / deciphering process.

Formal patent applications will be filed within two years, to claim protection and rights to these inventions and processes.

Thank You,

R. J. Miller

Robert J. Miller
18-July-95

Invention Disclosure Document: "Vectorlite" encyphering and deciphering system
Robert J. Miller, 18-Jul-95

Item 1a:	The "vectorlite key" table,	Inception of idea:	Approx Nov 1994
		First successful usage:	July 4, 1995

Terms:	Plaintext	Original, non protected, data
	Ciphertext	Protected / enciphered data

A "vectorlite" key table, for the use of enciphering and deciphering data, is a multidimensional table, of at least dimension 2, containing "cells". Each cell consists of the following three items:

(1) a plaintext value
(2) a direction vector value
(3) an attribute value

Figures 1, 2 and 3 illustrate a single page two dimensional "vectorlite" key table example. For the illustrated two dimensional example, key table item [0,0] (the origin being based at the top, left of the page) would consist of the ASCII binary value for the letter "S", a key table direction vector "row +" (i.e. to the right, or "east"), and no special attributes (blank - no entry).

"Vectorlite" key table pages may be of arbitrary dimensions, constrained only by the practicallity of implementation.

A "vectorlite" key table consists of key table "pages". Each page is an equal sized segment of the key table. For practicallity of implementation, each key table page is equally dimensioned and equally sized within each dimension. The number of key table pages within a "vectorlite" key table is unlimitted.

A "vectorlite" key table may be stored onto a comupter file. The computer file would consist of the "vectorlite" key table, preceeded by fields specifing various aspects of the key table, such as date of creation, author, size, number of dimensions, key "name", etc....

The plaintext item within each key table "cell" represents various portions of the plaintext original binary data. The binary data cell may be of any arbitrary "depth". Example depths are: a single 8 bit comupter byte; a 16 bit computer word; or a 4 bit nibble. Plaintext "cell" values are assigned randomly within a "vectorlite" key table during the construction process of a "vectorlite" key table. The Plaintext cell values may represent some or all potential values possible. The number of instances for each potential value within the table may be equal, random, or wieghted according to the probability of occurance within the plaintext data to be enciphered. There are typically many occurrances of each value. There are no restrictions to the placement of values, but typically some rules will apply so as to ensure the encipherability of all possible plaintext occurances.

Each potential value of plaintext is typically represented once within the table without any active "vectorlite" attributes (see below).

The "vectorlite" key table cell direction vector values represent directions (dimension and + or -). They are used by the enciphering and deciphering system to "change direction" while navigating through the key table. A two dimensional table has four values: up, down, left, and right, (or N, E, S, and W) to traverse through the a single row or column in accending or decending order. A table of dimension three would have six values, to traverse "up" or "down" while fixed within a single row and column in the other two dimensions. The concept is carried forward for higher order key table dimensions.

The attribute value identifies "vectorlite" key table cells of special meaning. The plaintext value of each cell with special meanings are ignored.

Item 1b: "vectorlite key table attributes"Inception of idea: Approx Nov 1994
First successful usage (#1) July 4, 1995

The attributes are improvements to the basic "vectorlite" key table. They enable the enciphering and deciphering system to perform special operations to make cryptographic analysis of the cyphertext significantly more difficult.

No.	Attribute Name	Description and Meaning
1	Re-vector	Absolute key table location values of plaintext follow within ciphertext. Ciphertext contains a number of values, equal to the dimension of the key table into the cipher text, to indicate the next location of a plaintext item. Revector cells are guarenteed "always" reachable. (Successfully implemented July 4th)
2	Page	Page in a segment of the key table. The key table page number follows as the next data item in the ciphertext.
3	Valid	Plaintext cell values obtained by subsequent ciphertext displacements into the key table are valid plaintext items.
4	Invalid	Plaintext cell values obtained by subsequent ciphertext displacements into the key table are NOT valid plaintext items (and are deceptions or false data...).
5	Trap	Constrain key table movement within bounded region. Bounded region specified by values following within ciphertext. The number of values is equal to the dimension of the key table.
6	Escape	Disable any key table Trap constraints which may be active (active key table region defaults back to the full size of each key table dimension).
7	Validate	A flag to indicate a special sequence of ciphertext / plaintext data is to follow to validate and verify the ciphertext is not tampered or altered in transmission.

Key table attributes (cont)

No. Attribute Name Description and Meaning

8 V-ADD

Data item following within ciphertext is to be "modulo-ed" to twice the key table dimension, added to each cell's direction vector, the result of which is "modulo-ed" again to twice the key table's dimension, and then used as a resultant key table direction vector.

Hence, the enciphering / deciphering system has an active key table direction "bias" to apply to each cell's direction vector. By default, the V-ADD value at start is zero.

The intent of the bias is to cause many more potential paths within the key table for each of the same sequences of binary data items.

9 P-ADD

Data item following within ciphertext is to be added and "modulo-ed" to each following plaintext cell value. The resultant value is the actual plaintext.

The enciphering / deciphering system has an active key table plaintext "bias" to apply to each key table cell's plaintext value. By default, at the start, the value is zero.

The intent of the P-ADD is to create many more potential paths for the same sequence of plaintext items at different occurance points within the plaintext.

10 Modify

Data items within ciphertext to follow will modify the current active key table page's plaintext values. The following ciphertext items follow:

1. Number of cells to subsitute
2. Dimension to apply subsitution within.
3. 1st cell item to change within the dimension
4. <plaintext data items to substitute>

A vectorlite key table remains unchanged on permanent storage, but may be modified by the ciphertext while active. A page operation disables a key table modification. If the same key table page were to page in and activate again, the original non-modified page would activate (and is hence a way to disable any and all modifications currently active).

Key table attributes (cont)

No. Attribute Name Description and Meaning

11 B-Change

Change enciphering and deciphering boundry condition behavior while navigating at key table dimension boundries.

Possible values indicate:
(1) "Circular wrap"
(2) Bounce, non-inclusive count
(3) Bounce, inclusive count
(4) Deflect +
(5) Deflect -

"Circular wrap" causes a soft boundry, where a row's or column's max and min values are adjectent to each other, and the enciphering and decipher system moves freely from min to max or max to min.

"Bounce non-inclusive" causes a hard boundry at the key table min and max values. Navigation beyond the boundry causes the movement to reflect back. The min or max cell's location is counted only once.

"Bounce inclusive" is the same as above, but the boundry cell's location is counted twice.

"Deflect +" causes a hard boundry at the key table's min and max cell locations. Attempts to navigate past the boundry causes a "+" dimension direction change. For a two dimensional example - Deflecting at the "East" boundry would cause additional movement to take place in the "South" direction.

"Deflect -" is the same as above, the dimension change is in the opposite direction (one dimension / direction negative).

Item 2: Vectorlite Key table paging	Inception of idea:	Approx Nov 1994
	First successful usage:	In development

Vectorlite Key table paging is the process of the enciphering and deciphering system activiating various portions of a very large key table. It is best illustrated by an example:

1. Take a vectorlite key table of dimension 2, each dimension of 256 elements in size. The depth of table's plaintext elements is one byte - 8 bits.

2. There may be as many as 256 key table pages for this key, each of dimension 2, and the size of each page being 256x256. The number of pages is limitted by the max value of the plaintext, which for one 8 bit byte is 256.

3. The enciphering system activiates any key table page by moving to any cell within the key table marked with the "PAGE" attribute. The value which follows within the ciphertext is the page number of the key to "activiate".

The two dimensional 8 bit deep 256 x 256 key table page is a very practical key table implementation for a "vectorlite" style table. It presents 16.7 million total potential plaintext cells, while only 16K are active at a time. It makes for efficient usage of computer memory and cache, provide the enciphering program stays within a single page for a time long enough to take advantage of this...

Item 3: "vectorlite enciphering"	Inception of idea:	Approx Nov 1994
and deciphering process	First successful usage (#1)	July 4, 1995

The "vectorlite" enciphering and deciphering process uses a "vectorlite" key table to encipher and decipher plaintext data.

The process is simple and best described (at this time) by example for a two dimensional key table of size 256 x 256, with 1 page (page 0):

1. Read in the 1st key table page. Key page 0 at location 0,0 is active.

2. The direction vector for key table item [0,0] will be the first direction to navigate

3. Take the first plaintext item to be enciphered

4. Find the first occurance of the plaintext item within the row 0 or column 0, searching in the direction obtained in step 2

5. The number of cells required to navigate through the table in the current direction, to find the first occurance of the plaintext, becomes the ciphertext.

6. If a key table boundry is reached, the currently active boundry condition is applied. By default, circular wrapping until the enciphering process directs otherwise. So, starting at 0,0 with direction North / Up would cause 0,0 's plaintext item to be checked first (current location) and the next item checked would be 0, 255

7. If the item is not found (not all bytes can be within a row or column containing special attribute cells, since their plaintext field is invalid), then find the displacement to a "revector" cell. Revector cells are guarenteed to be within reach. Insert the displacement into the ciphertext. Search for any cell with a plaintext value matching the plaintext value. Insert the table coordinates into the ciphertext.

8. The current location (in the same row or column, or absolute "jump" via step 7) become the current location. The direction vector of this cell is the next direction to search within a row or column for a matching plaintext value for the next item. Repeat step 5-7 as necessary to the end of the message.

The enciphering system uses the special attributes to manipulate the key table, change active key table boundries, insert false data, and bias the directional changing and plaintext all to provide many more permutations of potential paths for common repeated sequences of data. There are no restrictions applied to their usage.

The enciphering and deciphering programs utilize the VALID special attribute and assume all starting data is invalid. A random number of "lead-in" ciphertext elements may be inserted into the datastream until the it causes a VALID cell to be encountered. This can help prevent successfull cryptoanalysis of the begining portions of key tables where they are suseptable to repeat sequences.

Item 4: "Random Sequences"	Inception of idea:	Approx Nov 1994
	First successful usage (#1)	Nov 4, 1994 (?)

The "vectorlite" key tables require truely random sequences of direction vectors and plaintext data to be inserted into the key tables to prevent successful cryptoanalysis.

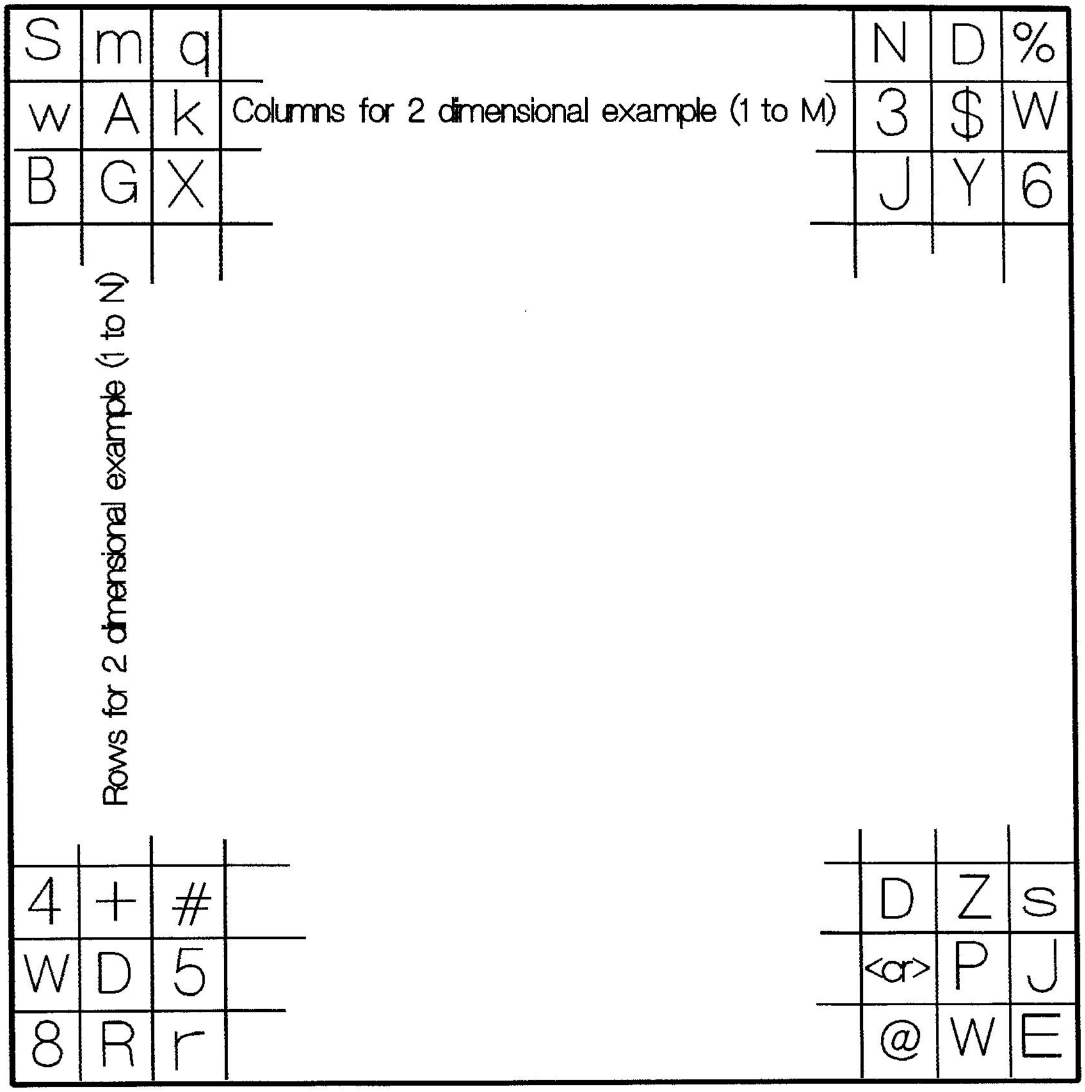
A method / process has been invented by which to obtain these sequences for the use within crypto key generation.

The process involves using a PC (or equivalent) windowing operating system, a special program, and a PC's mouse. The program captures the mouse's coordinates as a user moves the mouse within the window. The coordinates are modulo-ed to the maxminum value needed in the random sequence. The capture continues the required data elements needed in the random sequence are captured.

Figure 4 illustrates how a user may move the mouse as the program captures the coordinates, modulates them the max value needed, and stores them until all that are needed are obtained.

For example, a two dimensional "vectorlite" key table requires direction vector values 0, 1, 2, and 3. The mouse coordinates in this case would be captured, and the values "modulo-ed" to 4 (in "C" program language x-mouse-coord % 4).

It is claimed that this is an invention for the use to obtain truely random sequences of numbers for the use in the construction of cryptographic enciphering and deciphering key tables.



S	m	q		N	D	%
w	A	k	Columns for 2 dimensional example (1 to M)	3	$	W
B	G	X		J	Y	6
	Rows for 2 dimensional example (1 to N)					
4	+	#		D	Z	s
W	D	5		<cr>	P	J
8	R	r		@	W	E

Figure 1: Plain Text Substitution Table
(2 Dimensional Organization
example of Table and Cells)

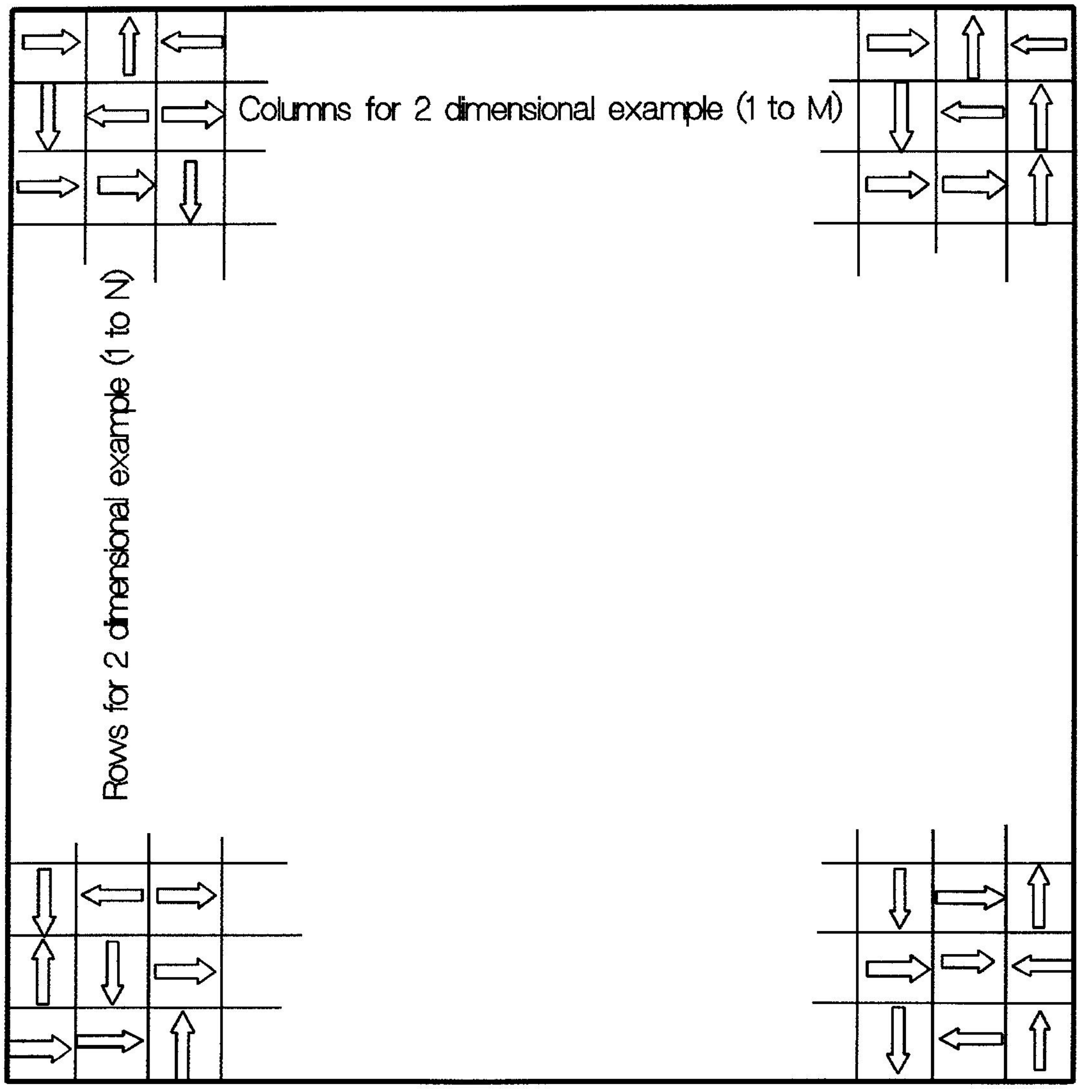

Columns for 2 dimensional example (1 to M)
Rows for 2 dimensional example (1 to N)

Figure 2: Vector Direction Change Table
(2 Dimensional Organization
example of Table and Cells)

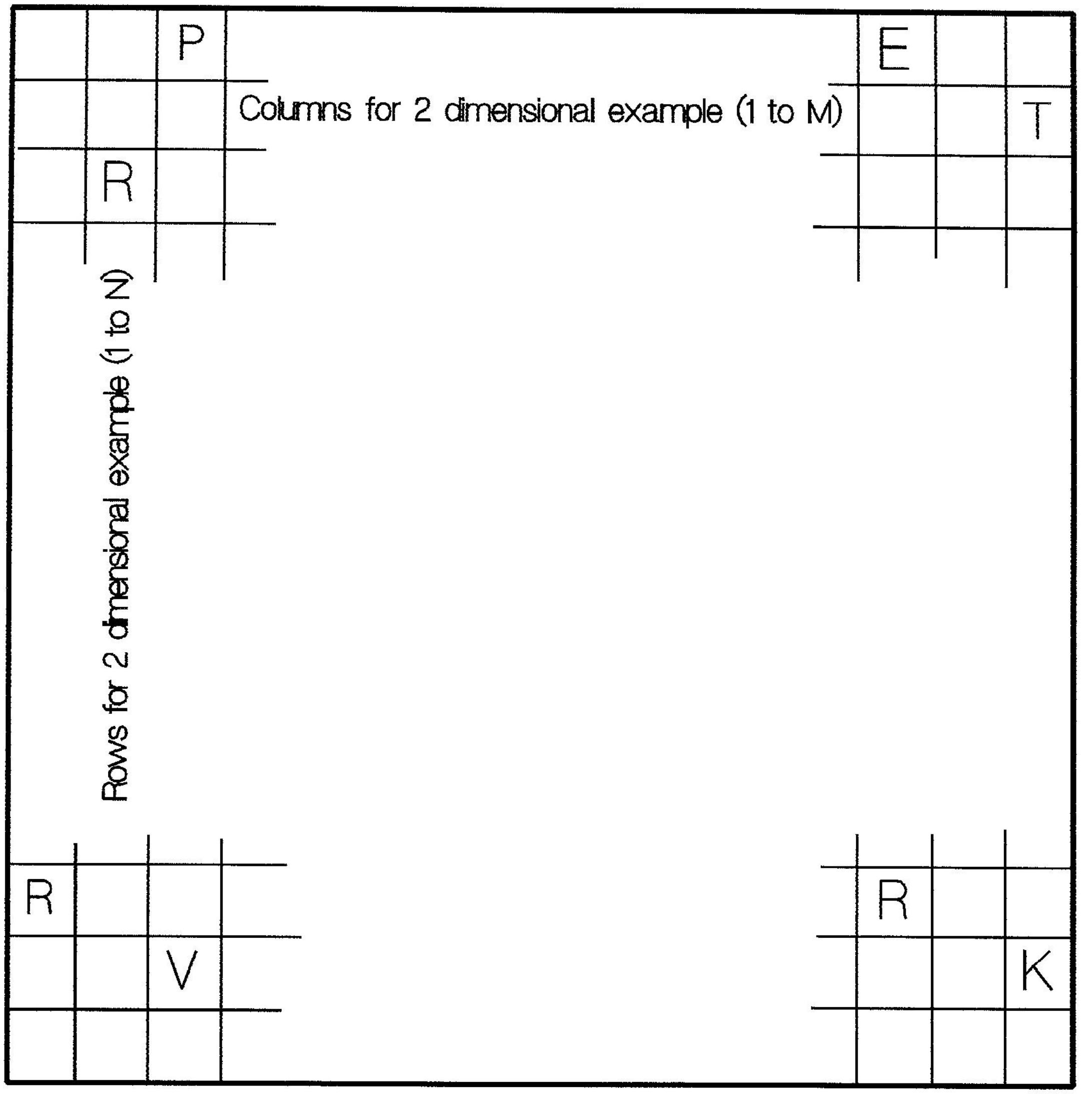

P
E
T
R
Columns for 2 dimensional example (1 to M)
Rows for 2 dimensional example (1 to N)
R
V
R
K

Figure 3: Substitution Cell Attribute Table
(2 Dimensional Organization
example of Table and Cells)