An Outline of the Stroke Theory as a Possible Method of Encipherment of the Voynich Manuscript

Elmar Vogt, Fürth*

September 22, 2010

The Stroke Theory – herafter referred to as »ST« – is my personal currently favoured idea how the Voynich Manuscript (»VM«) may have been enciphered. The ST has grown and evolved over time, thus I thought it about time to briefly summarize my current ideas on the issue.

This document is divided in two sections: The first will describe the suggested enciphering algorithm, while the second will answer questions regarding the document.

Please note that I don't claim to have actually deciphered *any* part of the manuscript.

Contents

*http://voynichthoughts.wordpress.com

0.1 A Note on Typographical Conventions

- Letters in EVA, representing ciphertext characters as found in the VM, are written in **boldface**,
- Strokes (in the sense of »graphical elements«) of the plaintext, after decomposition of the characters, are represented by »sans serif« characters enclosed in french quotation marks.
- Plaintext fragments are in general *italicized*.

1 The Algorithm

1.1 Basic Tenet

The basic assumption underlying the Stroke Theory is that in the process of enciphering the letters of the plaintext were decomposed into their constituent components – the »stroke« of the pen required to write them down –, and that each ciphertext character represents one of those plaintext strokes. A »syllable« is a group of ciphertext characters which represents one plaintext *letter*.

Consequently, each ciphertext *word* is composed of one or more »syllables«.

1.2 Idea

> For the sake of simplicity we'll assume that the plaintext was written in the latin alphabet, though the same method could be used on the cyrillic or greek alphabet.

In our alphabet, letters are composed of a limited number of different graphical elements or pen strokes. This holds also true if one looks at printed letters, which are, more or less, also built from a limited »repertoire« of strokes.

For example, the capital letter *A* can be decomposed into three strokes – namely, a slash »/«, a horizontal dash »−«, and a backslash »\«. Thus, we could write *A* as »/ − \«.

Likewise, the letter *l* consists – in its simplest form – of only a vertical stroke »|«. To write *b*, the very same stroke is used, with a subsequent ring-shaped »○«, thus *b* is rendered »| ○«. The letter *d* decomposes fairly similarly, but with »○« preceding »|«: »○ |«. The letter *o* finally is simply the ring, and nothing more: »○«.

The letter *n* shall consist – to keep matters simple – of only a single »c«-shaped crescent with the opening pointing downwards: »∩«. *m* on the other hand is the sequence of two such crescent strokes: »∩∩«.

We can go on and disassemble all letters which occur in our plaintext in this way. Depending on the exact alphabet we use (and on our taste), there is a lot of leeway in this decomposition, which also introduces some ambiguities: For example *A* might be rendered »/ − \« just as well as »/\−«. Likewise, in our simple example »∩∩« might come to mean *nn* as much as *m*.

Now, let's get down to business, assuming we want to encipher the word *Abdomen*. We start by decomposing the word letter for letter into its strokes:

A	b	d	o	m	e	n
»/ − \«	»| ○«	»○ |«	»○«	»∩∩«	»○−«	»∩«

Now we simply drop the plaintext letter boundaries and write everything as one string of strokes:

Abdomen
»/ − \	○○	○ ∩ ∩ ○ −∩«

In the next step, to get to the actual encipherment, we simply set up a table where each of the strokes is substituted with one letter from our ciphertext alphabet. (Which turns out to be the character set of the VM.) As an example, we'll use the letters in EVA transcription.

This is what a key table might look like:[1]

[1] Mind you that this is just an example, and by now means claims to be the real McCoy.

Stroke	EVA
»/«	**q**
»\«	**o**
»—«	**c**
»|«	**h**
»○«	**e**
»∩«	**y**
...	...

Thus, to encipher the word *Abdomen* again, we arrive at the following:

Plaintext:	A	b	d	o	m	e	n
Strokes:	/ — \	| ○	○ |	○	∩∩	○—	∩
Ciphertext:	**qoc**	**he**	**eh**	**e**	**yy**	**ec**	**y**

Or simply **qocheeheyyecy**.

1.3 Algorithm

1. Prepare a pattern to decompose your plaintext letters (both upper and lower case) into the set of constituent graphical elements (»strokes«).
2. Set up a substitution table to replace each plaintext stroke with one ciphertext character.
3. Remove all spaces between plaintext words as insignificant.
4. Convert your plaintext to »Camelcase«, ie alternate between upper and lower case letters. This will turn *Abdomen* into *AbDoMeN*.
5. Letter by letter, decompose the plaintext letters and encipher them according to the substitution table.
6. Introduce word breaks in the ciphertext every two or so plaintext characters.

And that's already it.

1.3.1 Why camelcase?

Step 4 probably requires a little more explanation.

If all plaintext letters were reduced to lower case before encipherment, then the complete ciphertext would consist only of 26 different syllables – 26 different *words* indeed. Clearly, this is not the case, so I hypothesized that the VM author introduced another step to »spice up« his encipherment. I think it's not implausible to assume that he experimented with his system

and found that the vanilla version (without camelcasing) left him with something which was way too easy to discern.

Yet when introducing the camelcase and some subtle wordlength rule,[2] he made the system much more difficult to crack while leaving the encipherment/decipherment effort moderately low.

1.4 Features of the VM v/s Features of the ST

1.4.1 VM »grammar« and ST »syllables«

> It has long been known that there is a set of rules, a »grammar«, governing the composition of VM ciphertext words, which are far from randomly assembled. Most »notorious« is the word-initial **qo**-group and the word-terminal **dy**. Yet, to date nobody has been able to formulate a concise set of rules which would explain the existance of all VM words as they are.

Such a behaviour would be the direct result of the ST.

Assuming in its simplest form that only latin letters were enciphered and that a ciphertext word always contains two plaintext letters, then there should be only some 50 different »chunks« or »syllables« (each corresponding to one upper or lower case letter) which would account for all of the VM words. Each VM word in this case would consist of one »prefix« and one »suffix« syllable, corresponding to one upper case and one lower case plaintext letter.

This would not be immediately obvious, because certain strokes (ie ciphertext letters) would be used both in the prefixes and the suffixes, hence blurring the distinction and the syllable boundaries. This gets aggravated under the assumption that special symbols are also enciphered, and that some words may contain only one or more than two plaintext letters.

Note that some strokes are used exclusively or predominantely for capital letters, like the vertical stroke »|«. This stroke occurs in *BDEFGHIKLNMPRT* among the upper case letters, but only in *fhkl* among the lower case letters. Thus, it would not be surprising to see the symbol for »|« ciphertext word-initial. At the same time, the vertical stroke extending *below* the baseline is only ever used in lower case letters like *pqyß*. Thus, we would expect this stroke to show up word-terminal (and more rarely).

[2] This feature would require an extra section in its own right. Clearly, something is going on with the word length – the words aren't strictly one or two plaintext characters long (ie consisting of one or two »syllables«), but seem to be composed of a fairly arbitrary number of syllables, while yet making sure that the »word-terminal« ciphertext syllables always fall on a word end. I'm not sure what's happening here and am hoping that the VM author simply introduced word breaks whenever he felt like it, ie – they don't really matter.

1.4.2 Information content

> From a statistical point of view, the average VM ciphertext word seems to contain an information content of 10 bit. This is roughly equivalent to two plaintext letters.[3]

This corresponds to the idea of the ST that a ciphertext word on average holds two plaintext letters.

1.4.3 Speed of writing, enciphering effort

The enciphering effort is fairly low. I'd expect the average person to be able to fluently encipher a text after a day or so. After all, you only need to learn the symbol set which enciphers the individual strokes (which could well correspond to the set of ca. 17 frequent VM characters).

This means that it's not an undue effort to encipher a work like the VM, and the »fluent apparency« with the absence of corrections (for what it's worth . . .) could be achieved.

1.4.4 Stringency

> There are only very few rules with the VM which are consinstently observed. For virtually every feature, there also appears to be an exception.

While this is not inherent to the ST, this effect can readily be achieved.

For example, above we have demonstrated that the letter *A* could be enciphered in different ways – as »/ − \«, or as »/\−«. Both methods are equally valid and require no change of a key on the decipherer's part. Thus, the encipherer can change these details on a whim without causing the decipherer undue trouble.

This might actually encourage a behaviour which results in a deviation from strict rules for the encipherment.

1.4.5 Repetitivity

The high degree of repetition of identical or near-identical words oberserved in the VM appears naturally as a consequence of the ST and need not be introduced *ad hoc*. (For an example of this, see 1.6.)

The simple fact is, that not only are our latin letters composed of a similar number of strokes, but their arrangement is also very similar. Letters »B«, »P«, and »R« for example only differ in a single stroke each, as do »E«, »F«, and »L«.

[3] 10 bit are just enough to encode two symbols out of a set of 32 different symbols.

This means that the ciphertext syllables will resemble each other more than if they were just random letter sequences. Furthermore, the word structure governed by the camelcase procedure will ensure that lower case syllables and upper case syllables always appear in their same respective spots, increasing the amount of perceived repetivity even more.

1.5 Arguments against the Stroke Theory

1.5.1 The line as an encipherment unit

> Several studies have shown that the line might be a relevant encipherment unit one way or another, in as far as features like wordlength etc. seem to be dependent on the word's position in the line. (Eg, line-initial words are longer on average.)

In the ST, the line plays no role, and hence those effects should not be observed.

It should be noted though that the line-positional effects mentioned aren't very pronounced.

1.6 Sample text

This is a sample of text enciphered with the ST algorithm. Plaintext language is 15th century German. Varying ciphertext word lengths have been allowed.

```
 alba hdanl habl hd hlab hada habda cbe hdacldace aea hada
hdaeab ldace fdl hdacd qpanl ldaceqo lppa qoaceabab hiae
aeaceqoac mihada hdala habl hdalafg haace qdlpace qdablhlab
adacd hdala haa hd aiace ldanl qace hkace qoac ag hdala haaceqd
qala lbhdanl miab hada ldbda qo haaceqoo hkace mialba hlfdl
hlaemifg ag hlcbemiqace a qo hkace aiace qoacae hlf qdaldcbeqace
qo cbe qphd hdalafg haaceqdqo ace fg haaceqd hlae ldace qo
anlace miacefghaaceqd hkace mibdaaeac cbeace qo mian qmi
hd aghlanl qobdahada ada hlac ag hdanlabab qala hd hdaqo
mi ace qo hkace mialba hlfanlace
```

Please note that the letters chosen bear no relationship to the EVA characters.

2 FAQs

2.1 When did you come up with the idea?

In April 2005. There was much discussion on the Voynich mailing list about the proper transcription system for the VM, and I tried my hand

at developing one as well. The basic idea of this system was to remove amibuguities by breaking down the ciphertext characters into the strokes which were drawn in writing them.

Halfway through it dawned upon me that this is actually what might have been done to *encipher* the VM.

2.2 What assumptions do you make about the plaintext language and character set?

Virtually none. The plaintext alphabet should be letter-based rather than syllable- or word-based. Trying to encipher Chinese, for example, would probably be a mess with that system.

But every language using latin, cyrillic or greek or hebrew letters should be possible.

2.3 What assumptions do you make about the origin of the VM?

Virtually none. There is nothing to contradict the mainstream assumption that the VM was written in the early 15th century somewhere in central Europe.

2.4 Do you have a precedence for the use of this enciphering scheme?

No, and I'm surprised by that, because to me it seems to be a fairly smple, fast, and moderately safe method of encipherment.

2.5 There should be only 52 constituting fragments of the VM according to your theory. But there are some 100 odd different ciphertext *characters* already.

There would not be only letters to be enciphered, but probably also all kinds of mathematical, alchemical and astrological symbols. These would require extra strokes to describe them which are not part of the »standard set«.

Actually, the ciphertext letter distribution seems to show some 17 frequent characters and a larger group of fairly to very rare characters. This would fit with the assumption that the »set of 17« is required to compose the latin letter set, while the rare characters are used in the construction of special symbols.

2.6 How do you reconcile the observation of »Currier A« and »Currier B« with your theory?

It's not unreaonable to assume that somewhen in the middle of the writing process, the author changed the basic stroke set or the plaintext alphabet

he had used. He could do so abruptly or in small steps, like with replacing different letter shapes etc.

2.7 The decipherer has no chance to see where in the ciphertext word the plaintext letter boundary is. Isn't this much too lossy?

This is partly true. On the other hand, the decipherer knows whether he sets out with an upper or lower case letter in a word, and, since there are only some 26 letters which he can be about to compose, it is not beyond his capabilities to discern the end of the current letter and the beginning of the next.

Likewise, since he is doing the decipherment »on the fly« and can see the context, something like *Aloolonnen* will quickly catch his eye and allow him to correct it to *Abdomen*.

2.8 What's the greatest obstacle to actually test the ST, ie to cracking the VM with it?

The problem is that there is a large number of degrees of freedom in the encipherment with this procedure, namely:

- Which alphabet (aka »font«) to use? For example, the letter »a« could be written with a »hook« over the top (like this: »a«), or without (like this: »*a*«).
- Which strokes are used? For one and the same alphabet, there are different possibilities which strokes are »elemental« and would be used in the breaking down.
- Which sequence? The letters can be broken down top-to-bottom, left-to-right or in an arbitrary sequence. This would generate different ciphertext letter sequences.
- What's the plaintext language? This will of course influence the letter frequencies and hence the distribution of the ciphertext characters.

Add to this the general problem with the unreliability of our transcription, where we can't be sure what constitutes two different or the same ciphertext characters, or which is one, two or actually three characters.

2.9 Then how do you plan to crack it – or determine that the ST is bogus?

The trick is to find the »syllable set« which represents the individual plaintext letters. Once the syllable set is known, the cracking of the VM is reduced to a simple substitution cipher.

But the problem is to arrive there, since we don't know where the syllable boundaries run in ciphertext words (where do the strokes of one plaintext letter end, and where do the strokes for the next letter begin?), and, judging from the length of VM ciphertext words, we can't even be sure how many plaintext letters are enciphered in one ciphertext word.

A naive computer analysis seeking to minimize the syllable set required to synthesize the ciphertext will obviously always arrive at individual ciphertext letters (because no more than 17 letters are required to synthesize the bulk of the VM ... go figure!)

I'm currently (Sept 2010) working on a slightly more sophisticated software tool to break down a text enciphered with the ST into its constituent syllables. I'm testing the tool against a known plaintext which was enciphered with the ST method, and can synthesize close to 90% of the ciphertext.

Elmar Vogt
Ludwigstr. 57
90763 Fürth
`elvogt@gmx.net`
Tel.: (++49) 173/591 29 93