MysteryTwister C3

THE CRYPTO CHALLENGE CONTEST

ASAC – A Strong(er) ADFGVX Cipher – Part 1

Author: Stefan Fendt

May 2016

Introduction (1/2)

Even today it could be useful for short texts to know a strong cipher which is doable by hand. A computer could be compromized without noticing it. Our own head, a pen, and a sheet of paper (for this cipher some more sheets) are unlikely to be compromised unnoticed.

The cipher in this series of challenges consists of three steps: A Polybius square, a pseudorandom number generator ('PRNG') out of a Polybius square, and a double-column transposition.

This series consists of 5 challenges that are based on each other. The first part only involves the Polybius square and serves as introduction, part 2 adds the PRNG, and part 3 uses the complete cipher but substitutes the double-column transposition with a single-column transposition (as in ADFGVX). Part 4 works with the complete ASAC cipher. At last, part 5 is a "bonus challenge" which just modifies the PRNG step in part 2 of the series.

Introduction (2/2)

The goal of the author was to get a cipher that – only just – can be done by hand, ensures a hopefully high degree of security even against computerized attacks and that can be easily remembered and built from memory.
The cipher is similar to ADFGVX, but has three major changes:

1. The Polybius square is a little bit bigger. This allows (by approximation) a compensation for the frequency of single characters.
2. A simple PRNG has been added.
3. Instead of the single-column transposition a double-column transposition is used.

Challenge Description

The first part of the series for ASAC only uses the first step of the three-step cipher, the Polybius square.
A description of the complete cipher together with a detailed example can be found starting from page 5.
The goal of this challenge (part 1 of the series) is to decrypt the given ciphertext that can be found in the text file (inside of the additional zip archive):
`mtc3-fendt-01-asac-01-ciphertext.txt`.

As solution, please enter the numbers written as digits you find in the plaintext in the order of their occurence in the text, parted by commas and without spaces.

Example (1/11) – Overview of the Cipher

The ASAC cipher is based on ADFGVX and consists of 3 steps:

1. Extended Polybius square (10x10 instead of 5x5).
2. The output of a PRNG from another Polybius square (10x10) is added modulo 10 to the result of step 1.
3. Double-column transposition (instead of single-column transposition).

For each steps 1 and 2 we need one key, and two keys for step 3.

Example (2/11) – Overview of the Cipher

Now we want to apply this method to the following little sample plaintext: **MysteryTwister C3 macht Spaß!**
The plaintext alphabet consists of the numbers 0 to 9, the blank, and the characters A to Z. Because of that, we ignore punctuation marks and substitute the 'ß' with 'ss'. Likewise, we would substitute 'ä', 'ö' and 'ü' with 'ae', 'oe' and 'ue'. All lower case letters become upper case letters.

Thus, our plaintext reads:
MYSTERYTWISTER C3 MACHT SPASS

Example (3/11) – Step 1: Filling the Polybius Square

First, enter the numbers 0 to 9 and a space into an empty 10x10 Polybius square, beginning in the top left corner. After that, we enter the alphabet (each ending with a space) until the square is filled. To compensate the frequency of the vowels approximately, 'A', 'I', 'O' and 'U' will always be written twice and 'E' will always be written four times. The key is used as column and row identifiers. If the key is shorter than 20 characters it will be repeated, if it is longer it will be cut after 20 characters.
The password in our example is 'KRYPTOGRAFENHAUSBOOT'.
(Hint: This password has nothing to do with the actual challenges).

Example (4/11) – Step 1: Filling the Polybius Square

After this our Polybius square looks as follows:

		0	1	2	3	4	5	6	7	8	9
		K	**R**	**Y**	**P**	**T**	**O**	**G**	**R**	**A**	**F**
0	**E**	0	1	2	3	4	5	6	7	8	9
1	**N**		A	A	B	C	D	E	E	E	E
2	**H**	F	G	H	I	I	J	K	L	M	N
3	**A**	O	O	P	Q	R	S	T	U	U	V
4	**U**	W	X	Y	Z		A	A	B	C	D
5	**S**	E	E	E	E	F	G	H	I	I	J
6	**B**	K	L	M	N	O	O	P	Q	R	S
7	**O**	T	U	U	V	W	X	Y	Z		A
8	**O**	A	B	C	D	E	E	E	E	F	G
9	**T**	H	I	I	J	K	L	M	N	O	O

Example (5/11) – Step 1: Shifting Elements in the Polybius Square

Respective to the value of the characters in the password ('A'=0, 'B'=1,...), first the elements of the rows and then the elements of the columns will be shifted modulo 10 to the right or to the bottom. For above password we get the following permuted square after shifting the row elements:

		0	1	2	3	4	5	6	7	8	9
		K	**R**	**Y**	**P**	**T**	**O**	**G**	**R**	**A**	**F**
0	**E**	6	7	8	9	0	1	2	3	4	5
1	**N**	E	E	E		A	A	B	C	D	E
2	**H**	I	I	J	K	L	M	N	F	G	H
3	**A**	O	O	P	Q	R	S	T	U	U	V
4	**U**	W	X	Y	Z		A	A	B	C	D
5	**S**	E	E	F	G	H	I	I	J	E	E
6	**B**	S	K	L	M	N	O	O	P	Q	R
7	**O**	Y	Z		A	T	U	U	V	W	X
8	**O**	E	E	F	G	A	B	C	D	E	E
9	**T**	I	I	J	K	L	M	N	O	O	H

Example (6/11) – Step 1: Shifting Elements in the Polybius Square

Afterwards, we shift the column elements and get the following permuted square as result:

		0	1	2	3	4	5	6	7	8	9
		K	**R**	**Y**	**P**	**T**	**O**	**G**	**R**	**A**	**F**
0	**E**	6	O	L	G	A	O	A	U	4	E
1	**N**	E	X		M	L	U	I	B	D	R
2	**H**	I	E	F	A	R	B	O	J	G	X
3	**A**	O	K	J	G		M	U	P	U	E
4	**U**	W	Z	8	K	H	1	C	V	C	H
5	**S**	E	E	E	9	N	A	N	D	E	S
6	**B**	S	I	J		T	M	2	O	Q	E
7	**O**	Y	7	P	K	A	S	B	3	W	H
8	**O**	E	E	Y	Q	L	A	N	C	E	V
9	**T**	I	I	F	Z	0	I	T	F	O	D

Example (7/11) – Step 1: Substitution of Plaintext with the Polybius Square

With the help of this square we now substitute the single characters, the digits, and the spaces with two-digit numbers. Each character has multiple possible substitutions. For instance, for 'O' we have the choice between 10, 50, 62, 03, 76 and 89, where the first number is the column number and the second one is the row number. The substitution of a given character shall choose one of the possibilities as random as possible.
One possible substitution for our plaintext is the following numerical sequence:

```
31 07 57 69 12 42 28 69 87 09 06 46 25 91 21 78 77 21
56 47 78 97 69 43 06 73 32 57 06
```

This is the substituted plaintext at the end of step 1.

Example (8/11) – Step 2: Creating the PRNG

Analog to the first step, we create another Polybius square where each row is filled with '0123456789'. Afterwards, we shift the elements of the rows and the columns with the help of the second key. Here we can see why it is important to shift the row elements first. In case we shift the columnn elements first, we have only identical elements in each column and our shifting would change nothing.

For our example, we choose 'PASSWORTZWEI' as second key.

Example (9/11) – Step 2: Creating the PRNG

After both shiftings our Polybius square looks as follows:

		0	1	2	3	4	5	6	7	8	9
		P	**A**	**S**	**S**	**W**	**O**	**R**	**T**	**Z**	**W**
0	**E**	2	7	7	8	7	3	6	9	0	2
1	**I**	8	3	2	3	5	1	8	2	6	0
2	**P**	6	6	4	5	0	8	8	7	4	5
3	**A**	3	1	4	5	6	6	4	9	1	1
4	**S**	1	3	0	1	9	1	2	9	9	4
5	**S**	6	3	8	9	4	7	9	5	4	9
6	**W**	2	9	5	6	6	0	7	3	0	1
7	**O**	5	7	3	4	6	5	2	0	3	1
8	**R**	0	4	8	9	2	7	8	8	8	7
9	**T**	2	2	4	5	0	7	1	3	0	5

Example (10/11) – Step 2: Applying the PRNG

Now we read the digits row by row and get a pseudorandom number with 100 digits. We add the single digits to the digits of the substituted plaintext modulo 10. For instance, the computation of the first 12 digits looks like this:

```
     31 07 57 69 12 42...
 +   27 78 73 69 02 83...
 ------------------------
     58 75 20 28 14 25...
```

If the substituted plaintext is longer than 100 digits we simply repeat the pseudorandom sequence. The result of the addition of the sequence with our example text looks as follows:

```
58 75 20 28 14 25 41 10 69 69 62 81 23 78 66 09 12 87
95 58 81 98 50 62 90 36 11 94 91
```

Example (11/11) – Step 3: Double-Column Transposition

For the double-column transposition we need another two passwords whose lenghts should be greater than $\lceil \sqrt{m} \rceil$, where m is the length of the (substituted) plaintext. Also, the lengths of the keywords shouldn't share any prime factors.

A detailed description of the double-column transposition (DCT) can be found in the first part of the series to the Double-Column Transposition/GRANIT (see step 2 in the example). So DCT will not be explained here again. The respective challenge can be found in [3].

For our example we used the passwords 'HALLOWELT' and 'VOGELFEDER' with the lengths 9 and 10. After applying the double-column transposition we get the following ciphertext:

```
29589 36561 91489 31100 48766 26019
66980 54588 12958 11292 17827 025
```

Links

(1) https://en.wikipedia.org/wiki/Polybius_square

(2) https://en.wikipedia.org/wiki/ADFGVX_cipher

(3) https://www.mysterytwisterc3.org/images/challenges/mtc3-drobick-01-doppelwuerfel-01-en.pdf

(4) You can find a sample implementation of this cipher in C++ inside the additional zip archive.