On inverting the VMPC one-way function

KAMIL KULESZA
Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Cambridge, UK[1],
Institute of Fundamental Technological Research, Polish Academy of Sciences, Warsaw, Poland,
e-mails: *K.Kulesza@damtp.cam.ac.uk, Kamil.Kulesza@ippt.gov.pl*

Draft version 5.2

Abstract.
We investigate the possibilities of inverting the VMPC one-way function, which was proposed at FSE 2004. First, we describe the function using the language of permutation theory. Next, easily invertible instances of VMPC are derived. We also show that no VMPC function is one-to-one. Implications of these results for cryptographic applications of VMPC conclude the paper.
Keywords: one-way function, cryptoanalysis, permutations

1. Introduction

In 2004 at the 11th Fast Software Encryption workshop Bartosz Zoltak proposed a candidate for a one-way function that he named *VMPC* ([1]). The name is an acronym for Variably Modified Permutation Composition, which nicely captures the main idea behind construction of the family of the functions concerned. In [1] the *VMPC* is defined as follows (original wording):

Definition 1 [Zoltak 2004]

A k-level VMPC function, referred to as $VMPC_k$, is such a transformation of n-element permutation P into n-element permutation Q, where

$VMPC_k = Q[x] = \mathrm{P}[\mathrm{P}_k[\mathrm{P}_{k-1}[:::[\mathrm{P}_1[\mathrm{P}[\mathrm{x}]]]:::]]]$

$x \in \{0,1,...,n-1\}$, $k < n$,

P_i is the n-element permutation such that $P_i[x] = f_i(P[x])$, where f_i is any function such that $P_i[x] \neq P[x] \neq P_j[x]$ for $i \in \{1,2,...,k\}$, $j \in \{1,2,...,k\}$, $i \neq j$.

For simplicity of further references f_i is assumed to be $f_i(x) = x + i$ ■

After stating the general definition, the paper describes a few particular instances of the functions: $VMPC_1$, $VMPC_2$, $VMPC_3$. In [1] these functions have the following form:

$$VMPC_1 = \mathrm{P}[\mathrm{P}[\mathrm{P}[\mathrm{x}]]+1] \quad (1.1)$$

$$VMPC_2 = \mathrm{P}[\mathrm{P}[\mathrm{P}[\mathrm{P}[\mathrm{x}]]+1]+2] \quad (1.2)$$

$$VMPC_3 = \mathrm{P}[\mathrm{P}[\mathrm{P}[\mathrm{P}[\mathrm{P}[\mathrm{x}]]+1]+2]+3] \quad (1.3)$$

The *VMPC* function is easy to compute, since it requires only three basic MOV instructions on Intel 80x86 processors.

Zoltak [1] claims that the *VMPC* function is difficult to invert for permutations of sufficiently many elements. Permutations of degree 256 are proposed as appropriate for cryptographic applications. The *VMPC*'s author claims that the most efficient inverting method has a complexity of about 2^{260} for 256-element permutations. This refers to the $VMPC_1$, with much greater complexities for higher-level functions.

This difficulty in inverting attracted our attention and motivated the work below.

In this work, following the original paper [1], we focus mainly on the $VMPC_1$, making

[1] Part of the work described in this paper was done when the author was visiting scholar at DAMTP.

occasional references to the $VMPC_2$, $VMPC_3$. From now on, when we refer to a VMPC function, we actually mean the $VMPC_1$ function, unless stated otherwise.
The paper is organized as follows: Section 2 is devoted to mathematical notation and description of the $VMPC$ using the language of permutation theory. In Section 3 we present instances for which the $VMPC$ function can be easily inverted (so-called "weak keys"). Next, we describe results on finding the general solution to the problem and comment on the inverting algorithm. Section 4 describes computational results concerning the image of the permutation group under the $VMPC$. We summarize the paper in Section 5; we indicate possible directions of further research and final conclusions.

2. VMPC in the language of permutation theory

First, we write the $VMPC$ using permutation theory terms, in order to benefit from it's powerful toolbox. In this paper we follow the most popular notation in the field, which can be traced back at least to Herstein's works, see [2]. For more detailed treatment of permutation theory, the interested reader is referred to appendix A. Let:

S_n be the symmetric group of degree n;
$|S_n|$ be the cardinality of S_n;
A be an element of S_n, $A \in S_n$;
I be the identity permutation;
A^{-1} be the inverse of permutation A;
A^m is multiplication of m permutations A;
A_n is alternating group of degree n (a normal subgroup of S_n).

There are many ways to represent the permutation A. In this paper we use the short matrix notation: $A = \begin{pmatrix} 1 & 2 & \dots & n \\ a & b & \dots & x \end{pmatrix}$, where letters in the lower row are assigned different numbers from the set $\{1,2,\dots,n\}$. For instance, consider a permutation of degree 4: $A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 4 & 2 \end{pmatrix}$.
In the short matrix notation the left-shift of A's lower row by one element ($<<1$) can be described using multiplication by the permutation $Q = \begin{pmatrix} 1 & 2 & \dots & n \\ 2 & \dots & n & 1 \end{pmatrix}$. For instance, consider the permutation A from above example, then the left-shift by one can be written as: $QA = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix}\begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 4 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 2 & 3 \end{pmatrix}$.
Using letter Q to denote above operation may at first create some confusion with Zoltak definition ([1]). However, Q has been associated with such a permutation long before the $VMPC$ paper.
Left-shift of A's lower row by m ($<<m$)[•] elements can be described using multiplication by the permutation Q^m.
Recall that every permutation can be written as a product of disjoint cycles.
Finally, cycles of length 2 are called transpositions.
Having all notations in place it is the time to write down the equations for the $VMPC_k$. Let $VMPC_k(A) = D_k$, then the following equations correspond to equations 1.1-1.3:

[•] For simplicity we consider $m < n$.

$$D_1 = A^2QA \tag{2.1}$$

$$D_2 = A^2QAQ^2A = D_1Q^2A \tag{2.2}$$

$$D_3 = A^2QAQ^2AQ^3A = D_2Q^3A \tag{2.3}$$

The $VMPC_k$ problem can be formulated as follows: given D_k which was computed from the permutation A using the $VMPC_k$ find A.

3. Inverting VMPC

In this section we concentrate on the $VMPC_1$. First, we show cases where given D, A can be quickly calculated analytically. Following convention from Section 1 we can write the equation 2.1 as: $D = AAQA$.

3.1 Special cases (a.k.a. *weak keys*)

1. Consider the situation when

$$A^2 = I. \tag{3.1}$$

In such a case:

$$D = QA, \tag{3.2}$$

which yields:

$$Q^{-1}D = A. \tag{3.3}$$

In order to test whether A meets condition from the equation 3.1, one has to check whether the following equation holds:

$$\left(Q^{-1}D\right)^2 = I. \tag{3.4}$$

If this is the case, the permutation A can be calculated from the equation 3.3. The question is how often such a solution happens.

The equation 3.1 holds when A consists exclusively of transpositions and cycles of length 1. Let Δ_1 be the number of permutations that meet this condition. It holds that:

$$\Delta_1 = 1 + \sum_{\substack{m=2 \\ m/2}}^{n} \left[\binom{n}{m} \ (m-1)!! \right], \tag{3.5}$$

where m is the number of elements that form transpositions in a permutation of degree n. For sufficiently large n (but still much lower than 256) , it can be seen that:

$$\Delta_1 < \frac{|S_n|}{n} = (n-1)!. \tag{3.6}$$

2. Consider the situation when

$$D = Q^{3r+1}, \tag{3.7}$$

where $r < n$. Then the following solution always exist:

$$A = Q^r. \tag{3.8}$$

Permutations D of the type described by the equation 3.7 are easy to spot, because they correspond to a left-shift by m-elements, where $m = 3r + 1$. Let's denote the number of such D's, which are unique by Δ_2. Then $\Delta_2 < n - 1$.

3.2 General solution - theoretical considerations

Special cases, which are described above, apply to no more than the $\frac{1}{n}th$ of all elements from S_n. The remaining ones are not so easy to handle. Searching for a general solution requires examining properties of the symmetric group S_n.

When searching for a general solution the first idea that comes to one's mind is to commute permutations Q and A in equations for the *VMPC* (eq. 2.1-2.3). If this were possible, equation 2.1 would have the following form:

$$D = A^3 Q, \tag{3.9}$$

Similary, the equation for the $VMPC_k$ would have the form:

$$D_k = A^{k+2} Q^x, \tag{3.10}$$

where $x = \sum_{m=1}^{k} m$.

Equations of such a type are relatively easy to solve by a deterministic polynomial time algorithm. We named such a method of solving the *VMPC* problem as an "abelian" solution. However, it will not work in a general case because S_n is non-abelian for all $n > 2$.

Although, the above fact eliminates an "abelian" solution for a general case, one may still be interested in some particular cases:

1. Checking the center of S_n ($Z(S_n)$), since all elements belonging to $Z(S_n)$ would commute with Q. Unfortunately, it is trivial for all $n > 2$ ($Z(S_n) = \{I\}$ for $n > 2$).
2. Checking normal subgroups of S_n which contains Q. Let's call such a subgroup H_Q. Then for some $X \in H_Q$ the equation 2.1 can be written as:

$$D = A^3 X. \tag{3.11}$$

Equations for the $VMPC_k$ would be treated in a similar way, leading to expressions analogous to the equation 3.10. Solving such equations requires checking $X \in H_Q$ and solving for A, very much like in case of equations 3.9 and 3.10. Such a procedure could be feasible provided that $|H_Q|$ is small. Unfortunately, the only normal subgroup of S_n containing Q is A_n (for odd n). Since $|A_n| = \frac{|S_n|}{2}$, this approach is not practical.

All of the above show that properties of S_n are not of great help. So, maybe it would be possible to find a homomorphism from S_n to some more useful group?

For instance, homomorphism from S_n to an abelian group would allow us to "transport the equation" to an abelian group, solve it in such a group and map solutions back to the relevant elements of S_n. Unfortunately, the only such homomorphism for $n > 2$ is (up to the representation) the homomorphism from S_n to Z_2. All that can be obtained using this is the parity of A. However, A's parity can be calculated in an easier way, which will be described in Section 3.3.1.

Using methods from classical permutation theory we have not been able to obtain a general solution for the *VMPC*. However, we have checked tools from the classical toolbox and eliminated those as possible candidates. Still, there are situations where classical group theory is useful.

3.3 In search of a general solution – what can be done?

In this section we describe what can be done using existing tools.

3.3.1 A's parity

Consider the equation 2.1, multiplying both sides of the equation by Q, to yield

$$DQ = A^2QAQ, \tag{3.12}$$

which can be written as:

$$DQ = A(AQ)^2. \tag{3.12a}$$

Since $(AQ)^2$ is always even, the parity of A is the same as the parity of DQ.
Similar reasoning, together with the known parity of Q (which is always opposite to n's parity), allows us to find A's parity for other functions from the $VMPC$ family.
Knowing A's parity is important, since it reduces the number of possible candidates by a factor of 2.

3.3.2 n-cycle solutions

Recall the idea of commuting Q and A in equations for the $VMPC$. The idea presented in this section is very much the same as the one that was behind an "abelian" solution. It amends to finding some X such that:

$$QA = AX. \tag{3.13}$$

Now, the question is whether we can say anything about X assuming that any $A \in S_n$ can be used in the equation. It turns out that this is the case.

Consider $A = \begin{pmatrix} 1 & 2 & \dots & n \\ a & b & \dots & x \end{pmatrix}$, the equation 3.13 can be written as:

$$QA = \begin{pmatrix} 1 & 2 & \dots & n \\ 2 & 3 & \dots & 1 \end{pmatrix}\begin{pmatrix} 1 & 2 & \dots & n \\ a & b & \dots & x \end{pmatrix} = \begin{pmatrix} 1 & 2 & \dots & n \\ a & b & \dots & x \end{pmatrix}X. \tag{3.14}$$

Solving equation 3.14 for X yields:

$$X = \begin{pmatrix} a & b & \dots & x \\ b & \dots & x & a \end{pmatrix}. \tag{3.15}$$

Next, observe that the permutation X is made of only one cycle of length n. Such a permutation is called an n-cycle. There is whole family of n-cycles (e.g., Q belongs to it). One of the interesting properties of such permutations is that the inverse of an n-cycle is also an n-cycle. The cardinality of the set of n-cycles is $(n-1)!$. By substituting equation 3.13 into equation 2.1 one obtains:

$$D = A^3X, \tag{3.16}$$

which can be written as:

$$DX^{-1} = A^3. \tag{3.16a}$$

Again, solving such equations requires checking all n-cycles and solving for A. Although it is not feasible for a head-on attack, it significantly reduces number of possible candidates to search (by factor n). In this capacity it can be useful for some algorithmic constructions.
It is interesting to note that the same result, as presented above, can obtained by considering orbits of the group generated by Q([3]).

3.3.3 Special cases applied in a more general situation

The cases described in Section 3.1 have a very useful feature that, given the permutation of D, it is possible to decide quickly whether it belongs to a special case. In addition, positive outcome immediately yields the solution to the $VMPC$ problem.
It should be observed that the conditions stated in the equations 3.1 or 3.7 need not apply to a whole permutation A in order to be valid. In such a situation, one will not obtain immediate solution for whole permutation A, but may quickly uncover its parts which meet one of these conditions. For this mechanism to work at least two elements from A (preferably at least 3) should form a *block* in the short matrix notation.

A *block* in the short matrix notation is a sequence of m index elements (upper row) such that corresponding permutation elements (lower row) contain only elements from m index element sequence (possibly permuted). For instance consider:

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 6 & 8 & 4 & 3 & 5 & 1 & 2 & 7 \end{pmatrix}. \tag{3.17}$$

The block is formed by index elements 3,4,5 with corresponding permutation elements 4,3,5. Given a permutation A, its elements that form a block can be found as described in Section 3.1. When block consists m elements, then at least $m-1$ of them can be uncovered. Elements that belong to the block form cycles, which are disjoint with all other cycles in that permutation. Hence, remaining A's elements can be considered separately, as a permutation on $n-m+1$ elements. This observation allows to decrease size of the problem being solved. As a result, in some cases problem might be vulnerable to attack by available algorithms. Let's denote:

y as the degree of permutation which can be feasibly attacked with available algorithms;

Δ_3 as the number of permutations which can be feasibly attacked by using block method outlined above.

Recall that n is the degree of permutation A and assume $y < n$. Then taking into account number of weak keys (see Section 3.1) the value of Δ_3 can be described as follows:

$$\Delta_3 = (\Delta_1 + \Delta_2) \cdot y! \tag{3.18}$$

Substituting values for Δ_1 and Δ_2 yields:

$$\Delta_3 < [(n-y-1)! + (n-1)] \cdot y! \tag{3.19}$$

3.3.4 Comments on an original inverting algorithm

Zoltak [1] provided only a rough estimate for the computational complexity of the inverting algorithm. The algorithm was based on phenomenological observation that when certain number of A's elements are known, the function can be inverted. This number depends upon the level of the *VMPC* function and the degree of permutation concerned (n). For instance, for $n = 256$ and the first level function, the number of known elements should be around 34. In [1], the inverting method for above parameters has the complexity of about 2^{260}. After going through reasoning presented, we found that the estimate provided concerned the complexity of guessing sufficient number of permutation elements, rather then the algorithm itself.

In the absence of any further information we tried to assess the complexity of the inverting algorithm and also to implement it. Unfortunately, the description provided was insufficient to complete any of these tasks. As a results, we ended up with the program which we believe is close to what the *VMPC* author described. We ran some numerical experiments for the first level *VMPC* function and $n = 256$ in order to check results on a number of known elements. Experiments were carried out on randomly selected permutations. Because of quite limited computing power our sample cannot be considered statistically significant (having in mind $|S_n|$), so original results were neither definitely proved or disproved. We observed a wide range for numbers of known elements required in the individual cases. In addition, many permutations could not be successfully inverted with less then 42 elements and some required even 50.

While working on the inverting algorithm, we designed our own inverting procedure. Let's start its brief description from the observation that equation 2.1 can be rewritten as the following set of equations:

$$\begin{cases} D = BQA \\ B = A^2 \end{cases}. \tag{3.20}$$

Next, the set of equations has to be solved for A.
The main ideas behind our procedure are:

1. Observe that in the set of equations (3.20) the equation for D is very easy to solve. If we assign any value to the particular element of the permutation B we immediately obtain a corresponding value for an element from the permutation A.
2. Apply the second equation (for B) to test whether such a value of the element from the permutation A is valid.
3. If value is not valid, we add it to the list of forbidden values for the element concerned.
4. If value is valid, it will provide the element in B with which it is linked.

This routine is applied to all elements in A in order to build the list of forbidden values for every element (see, 3 above), which narrows the space of possible permutations A.
In addition, for every valid value of an element from A, the list of linked elements from B is created (see, 4 above).
For every element of the permutation of degree n, n values have to be tested. Hence, the complexity of this operation is n^2.
Both lists (forbidden values and linked elements) can be used to check the validity of values of a permutation's elements fed into procedure. This check is performed in a linear time.
Hence, the complexity of inverting the $VMPC$ by our procedure is n^2 times the complexity of guessing the sufficient number of permutation elements. However, it should be emphasized that permutations elements are guessed from a smaller set than in the original algorithm ([1]). This should result in a smaller number of permutation's elements that have to be guessed.
There is a more room for further improvement, for instance one may make use of the facts that $B \in A_n$ and A's parity can also be easily established (Section 3.1.1). Taking these two factors into account further decreases the number of possible A's.
We programmed a crude implementation of our procedure and tested it numerically. It performed a little better than the original method, but again our sample cannot be considered statistically significant.
To summarize: we are still not sure whether our procedure is equivalent to the original one or whether it differs substantially. This is mainly due to lack of information regarding the original inverting algorithm. Even, if both methods reduce to the same thing, our procedure is much more transparent and offers simplicity with quantifiable computational complexity.

4. The image of S_n under the $VMPC_k$

We start from the fact the $VMPC$ is never one-to-one. While proof of this result is somewhat tedious, its essence lies in the observation that permutations having specific cycle decompositions cannot be created when $VMPC$ acts on S_n ([4]). Since, the proof has rather a qualitative character, the quantitative question regarding the cardinality of S_n's image under the $VMPC$ remains open. Let's denote:

$D_{n,k}$ as the image of S_n under the $VMPC_k$;

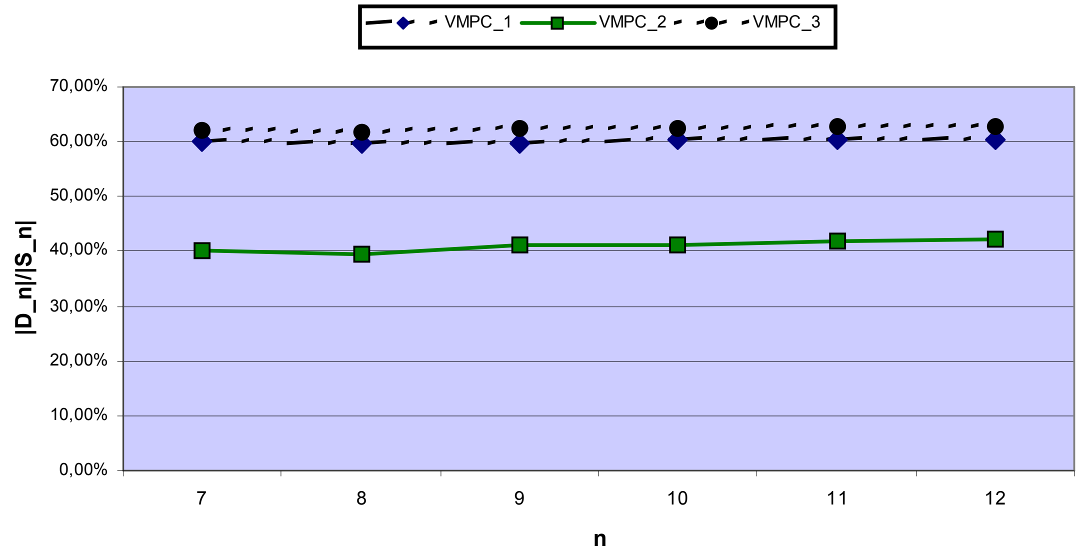
$\left|D_{n,k}\right|$ as the cardinality of $D_{n,k}$, in other words the number of distinct elements of $D_{n,k}$.

We calculated all $D_{n,k}$ for $k \in \{1,2,3\}$ and $n \in \{3,...,12\}$. First, we investigated cardinalities of sets $D_{n,k}$ and compared them with cardinality of S_n. As an example, in the table 4.1 we present values for $D_{n,1}$.

n	3	4	5	6	7	8	9	10	11	12
$\|S_n\|$	6	24	120	720	5040	40320	362880	3628800	39916800	479001600
$\|D_{n,1}\|$	4	16	80	460	3031	24072	216522	2190520	24103024	289062962
$\|D_{n,1}\|/\|S_n\|$	0,6667	0,6667	0,6667	0,638889	0,601389	0,597024	0,596677	0,603649	0,603832	0,6034697

Table 4.1. $|D_{n,1}|$ ***compared with*** $|S_n|$ ***for*** $n \in \{3,...,12\}$

From the table 4.1 it is clear that $D_{n,1}$ is always much smaller than S_n. In the tested data range the ratio $\frac{|D_{n,1}|}{|S_n|}$ converges to 0.6. The size of $|D_{n,k}|$ as a percentage of $|S_n|$ for the $VMPC_1$, $VMPC_2$ and $VMPC_3$ is presented on the graph 4.1. In the graph, values for $n < 7$ are omitted for legibility.



Graph 4.1. Cardinality of S_n ***'s image under the*** $VMPC_1$, $VMPC_2$, $VMPC_3$ ***as a percentage of*** S_n ***'s cardinality.***

From the graph it is clear that the $VMPC_1$ and the $VMPC_3$ produce images of similar size, about 60% of $|S_n|$. In turn the $VMPC_2$ behaves differently, it produces an image with a size cardinality of 40% of $|S_n|$. We return to this issue later in the paper.

Now, it is time to ask the question: how many permutations A ($A \in S_n$) give the same image under the $VMPC_k$.

Let's denote set of all elements of $D_{n,k}$ which can be computed from more than one permutation A as $D_{n,k}^{++}$. The cardinality of such a set will be denoted as $|D_{n,k}^{++}|$.

In the table below we present the ratio $\frac{\left|D_{n,k}^{++}\right|}{\left|D_{n,k}\right|}$. Again, we omit values for $n < 7$ for clarity.

N	7	8	9	10	11	12
$\left\|D_{n,1}^{++}\right\| / \left\|D_{n,1}\right\|$	46,02%	43,17%	44,21%	43,50%	43,45%	43,35%
$\left\|D_{n,2}^{++}\right\| / \left\|D_{n,2}\right\|$ |	71,53%	64,96%	65,93%	66,63%	66,99%	67,21%
$\left\|D_{n,3}^{++}\right\| / \left\|D_{n,3}\right\|$	41,96%	43,09%	42,12%	42,18%	41,99%	41,96%

Table 4.2. Ratio between $\left|D_{n,k}^{++}\right|$ and $\left|D_{n,k}\right|$, for $n \in \{7,...,12\}$

The table 4.2 shows that there is a significant number of permutations A ($A \in S_n$), that give the same image under the $VMPC_k$. Unsurprisingly, percentages for the $VMPC_1$ and $VMPC_3$ are quite similar, while the percentage for the $VMPC_2$ is much higher.
In addition, the distribution of permutations A that map to the same image is non-trivial. We illustrate it in the next table by the mean of example for $n = 12$.
For the fixed values of n and k, every element from $D_{n,k}$ can be assigned a number of permutations A ($A \in S_n$) that map to it. Let's use acronym $A2D$ to denote such a number.

#	$A2D$	$VMPC_1$	$VMPC_2$	$VMPC_3$
1	1	163742508	66058416	175048468
2	2	80992932	60905484	87497172
3	3	30331358	39126988	29519168
4	4	9736512	20073546	7582284
5	5	2942448	8809416	1586082
6	6	881000	3576438	287124
7	7	274092	1394696	47280
8	8	97842	595899	7584
9	9	35356	275400	1770
10	10	15756	170450	608
11	11	6564	100152	144
12	12	3600	72225	120
13	13	1236	47260	104
14	14	828	41820	24
15	15	614	29760	12
16	16	156	27250	16
17	17	72	15864	12
18	18	52	16104	0
19	20	24	9384	0
20	24	4	12366	0
21	27	8	7520	0
...	...	0	...	0
181	140376	0	1	0

Table 4.3.Number of image elements of S_{12} under the $VMPC_k$ that can be computed from multiple A.

The case of the $VMPC_3$ distribution is made of 16 consecutive values of $A2D > 1$. The maximum number of different permutations A, which give the same element from $D_{12,3}$ is 17. There are 12 different elements from $D_{12,3}$ with $A2D = 17$. The $VMPC_1$ distribution is similar, with some gaps towards the end. There are total 20 values of $A2D > 1$. The maximum number of different permutations A, which give the same element from $D_{12,1}$ is 27. There are 8 different elements from $D_{12,1}$ with $A2D = 27$. As previously, the $VMPC_2$ is different. At the beginning the distribution is continuous up to about #100 (not presented in the table 4.3). This is followed by increasing gaps. There are total of 180 values for $A2D > 1$. The maximum number of different permutations A, which give the same element from $D_{12,2}$, is 140376. There is only one element from the image of S_n under the $VMPC_2$ that can be computed from 140376 different permutations A. Next on the list (not presented in the table 4.3) are 12 elements of $D_{12,2}$ with $A2D = 1540$.

To summarize relations between the $VMPC_k$ for various k values, we state a conjecture.

Conjecture 4.1

Given two natural numbers k_1 and k_2 such that $k_1 \neq k_2$ and both have the same parity, two functions $VMPC_{k1}$ and $VMPC_{k2}$ will produce images of S_n with similar characteristics (e.g., the cardinality and the decomposition into $A2D$). ■

Observation 4.1

High number of permutations A, such that $VMPC_k(A) \in D_{n,k}^{++}$, has far reaching consequences for the difficulty of inversing the $VMPC_k$. For instance, consider brute force guessing (see Section 3.3.4). In such a case the attacker has to guess a set of correct values (say 34) for one out of $A2D$ permutations. This makes the task $A2D$ times easier than originally estimated. ■

Observation 4.2

For the $VMPC_1$ we found that permutations A, which fall into special cases (see Section 3.1), usually yield elements from $D_{n,1}$ with $A2D > 1$. Hence, the probability of successful attacks making use of weak keys increase, maybe even by an order of magnitude. This is not so surprising, since the value of Δ_1 was first estimated in relation to $|S_n|$ (eq. 3.6), not to $|D_{n,1}|$. ■

<u>Concluding remarks on S_n's image under the $VMPC_k$.</u>

1. Our numerical experiments were performed on all elements from S_n for $n \in \{3,...,12\}$. During calculations for $n = 12$ we approached the limit of today's PC technology. With some extra effort one could make calculations on all elements from S_n for $n = 13$ and maybe $n = 14$[••]. However, we do not think that such calculations would bring qualitatively different results. Obtained results are sufficient for extrapolation.

[••] Supercomputer processing power might allow to achieve n=17, as long as someone is willing to pick up the bill. Even in such a case, we do not expect more significant results to be collected.

2. The $VMPC_k$ function is not one-to-one. In this section we showed that this not an issue of few elements from S_n, but a basic property of the function. The cardinality of an image of S_n under the $VMPC_1$ and $VMPC_3$ is about 60% of $|S_n|$. The same cardinality under the $VMPC_2$ is about 40% of $|S_n|$. We expect these percentages to be stable (or even decrease) as n increases.
3. In all cases investigated there was substantial number of elements of the image of S_n under the $VMPC_k$, such that different elements of S_n would map into the same element of the image. Again, we expect percentages of such elements for the particular $VMPC_k$ to be stable (or even increase) as n increases.
4. We expect that the $VMPC_k$ for $k > 3$ would behave much in the same way as functions described in this section. This view is formalized in conjecture 1.

5. Conclusions

First, we summarize our results, simultaneously outlining possibilities for future research. We conclude with remarks on the usefulness of the $VMPC$ in cryptography.

5.1 Summary and further research

We started by describing $VMPC$ in the language of permutation theory. This allowed us to write the definition of $VMPC$ as permutations equation. The $VMPC$ problem was formulated in terms of finding solutions to an equation. For the $VMPC_1$, we showed that there exist permutations for which the function can be inverted in linear time. Moreover, the number of such permutations is non negligible. Similar methods can be given for other members of the $VMPC$ family. Searching for more weak keys is an interesting path for future research. Next, we discussed applications of classical permutation theory methods to finding a general solution to the $VMPC$ problem. We eliminated some seemingly promising lines of research and obtained a few partial results, which can be useful for the computational approach, exhhausting methods of classical permutation theory in the process. The future theoretical research should rather focus on methods from representation theory (e.g., [5]) or computational group theory [6]. Another promising line of research may be algebraic attacks, for instance see [7].
On the algorithmic front we proposed our own inverting procedure, which is transparent and strait forward. Also, we were able to state and justify the computational complexity of our procedure. Comparison with the traditional inverting algorithm seems to be in our favor. A further development of the proposed procedure and its optimization seems to an interesting task. Finally, we investigated the image of S_n under the $VMPC_k$. It was confirmed that family of the $VMPC$ functions is not one-to-one and S_n's image under the $VMPC_k$ takes up about 40%-60% of S_n's cardinality. As a result, some inverting methods can be significantly enhanced. Further research in this area could focus on conjecture 4.1 and qualitative results based on observations 4.1 and 4.2.

5.2 Remarks on usefulness of $VMPC$ in cryptography

We have not been able to find a general solution to the $VMPC$ problem and we still do not have an efficient inverting algorithm for every case. However, our results indicate that the $VMPC$ is not a good candidate for a cryptographic one-way function.
A non-negligible number of weak keys (Sections 3.1 and 3.3.3), coupled with the peculiar structure of S_n's image under the $VMPC_k$, which is not fully understood, disqualifies it at

present. In addition, other researchers have not thoroughly investigated this family of functions. In fact we do not know any other papers which investigate basic properties of the *VMPC*. Since there are many promising paths for further research, we would be very much surprised if no more weaknesses were found.
Zoltak [1] proposed a stream cipher, a key scheduling algorithm and a MAC scheme all built using the *VMPC* function. It is well known fact (e.g., [8]) that all those functionalities can be built from a one-way function. However, using *VMPC* to build them is asking for repeating the Enigma story. Its weaknesses known today already make it vulnerable to attack methods similar to ones used against Enigma (see [9], [10]), not to mention more sophisticated ones ([11]).

Acknowledgement

Author wants to thank Prof. Jerzy Browkin for numerous discussion on the *VMPC* and group theory. Many thanks go to all students participating in Summer Research Lab Project for the fun we had while playing with the *VMPC*. Especially, I want to thank Adrian Orzepowski for his help in obtaining results in Section 3.3.4 and 4. Lat but not least, I want to thank Prof. Zbigniew Kotulski for his suggestions and a final encouragement to write it the paper.

References

[1] B. Zoltak. 'VMPC One-Way Function and Stream Cipher'. Proceeding of FSE 2004, Roy, Bimal; Meier, Willi (Eds.), LNCS vol. 3017, Springer-Verlag, Berlin, Heilderberg, New York. 2004.

[2] I.N. Herstein. Topics in Algebra. Blaisdell Publishing Company. London. 1964.

[3] J. Browkin. Orbits of the group generated by Q. Private communication on 5.08.2005.

[4] J. Browkin. Proof that the VMPC is not one-to-one. Private communication on 26.07.2005.

[5] B. Simon. Representations of Finite and Compact Groups. American Mathematical Society. 1996.

[6] C.C. Sims. Computation with finitely presented groups. Cambridge University Press. Cambridge.1994.

[7] Nicolas Courtois. 'General Principles of Algebraic Attacks and New Design Criteria for Components of Symmetric Ciphers'. Proceeding of AES 2004, Dobbertin, Hans; Rijmen, Vincent; Sowa, Aleksandra (Eds.), LNCS vol. 3373, Springer-Verlag, Berlin, Heilderberg, New York. 2005.

[8] A.J. Menezes, P. van Oorschot and S.C. Vanstone. Handbook of Applied Cryptography, CRC Press, Boca Raton 1997.

[9] K. Gaj. Enigma cipher, breaking methods. WKŁ, Warsaw 1989.

[10] A.Orłowski, K. Gaj. 'Facts and Myths of Enigma: Breaking Stereotypes'. Advances in Cryptology - Eurocrypt 2003, Biham, Eli (Ed.), LNCS vol. 2656, Springer-Verlag, Berlin, Heilderberg, New York. 2003.

[11] Y. Tsunoo, T. Saito, H. Kubo, M. Shigeri, T. Suzaki, and T. Kawabata, 'The Most Efficient Distinguishing Attack on VMPC and RC4A', eSTREAM, the ECRYPT Stream Cipher Project: http://www.ecrypt.eu.org/stream/papersdir/037.pdf

Appendix A – selected information from permutation theory

In the paper we use the most popular notation in field, which can be traced back at least to Herstein's works, for instance see [2].

Definitions

S_n is a set of all permutations on n elements. S_n forms a group, which is called symmetric group of degree n. $|S_n|$ is a cardinality of S_n, $|S_n| = n!$.

A is the element of S_n, $A \in S_n$.

I is the identity permutation, $AI = IA = A$ for all $A \in S_n$.

A^{-1} be the inverse of permutation A, $AA^{-1} = A^{-1}A = I$ for all $A \in S_n$.

A^m is multiplication of m permutations A, for instance when $m = 3$ $A^3 = AAA$.

Theorem A.1

Every permutation can be written as product of disjoint cycles. ■

Theorem A.2

If the pair of cycles does not have entries in common, then they commute. ■

Definitions (c.d.)

Cycles of length 2 are called transpositions.

Theorem A.3

Every permutation in S_n, $n > 1$, is a product of transpositions. ■

Theorem A.4

Every permutation in S_n can be represented as a product of either even or odd number of transpositions. A permutation, which is a product of even number of transpositions, cannot be a product of odd number of transpositions and vice versa. . ■

Definitions (c.d.)

A_n is a set of all even permutations in S_n. A_n is a normal subgroup of S_n. It is called alternating group of degree n, $|A_n| = \frac{n!}{2}$.

$Z(G)$ is center of a group G. $Z(G)$ is the subset of elements in G that commute with every element in G.