# *Quasi-Algorithms / Quasi-Functions*
# *Polymorph Encryption*
# *[Joined Thesis version 0.9]*

Kostadin Bajalcaliev
http://eon.pmf.ukim.edu.mk/~kbajalc
kbajalc@eon.pmf.ukim.edu.mk

## Abstract

**Quasi-Algorithms are new theoretical approach in understanding computation and can have a profound practical usage. Redefinition of term "Algorithm" is presented. Benefactions in cryptography are discussed, especially concerning block cipher design trough the theory of Quasi Algorithms.**

## 1. Introduction

In my prior works, especially Block cipher designs the idea of Polymorph Encryption is introduced. [To note that there is no similarity between my theory of Polymorph Encryption and the similar named Virus design technique]. Since ANIGMA cipher, published in 1997, Polymorph Encryption is the building block in my Block-Cipher designs and a center of my research work. Acknowledging the need to generalize the concept and to make precise definition as well to establish the terminology exhaustive research has been done. This thesis is the summary of the results.

Polymorph Encryption is a strategy of designing a secure Block-Cipher. The main idea is to design dynamic algorithms using data-depend operations, data-depend operand chousing etc. As a good illustration here is the F function used in SQ2:

```
unsigned long F(unsigned long x)
  {
   int i; unsigned long r;
    for(i=0;i<RP;i++)
     {
      r=(x*(x+11));
      switch((int)(r>>30))
       {
           case 0: x+=S[(int)((r>>26)%16)]; break;
           case 1: x*=S[(int)((r>>26)%16)]; break;
           case 2: x^=S[(int)((r>>26)%16)]; break;
           case 3: x-=S[(int)((r>>26)%16)]; break;
       }
     }
    return x;
  }
```

`S[]` is the key array. The two most-significant bits of r=(x*(x+11)) are used to chouse the operation, and the next four the operand. The effect of this design principle can be summarized as: No Key = No Algorithm. If F(x) and S are known can x be computed? Generally No, because some of the bits from x are used to chouse the operation and the operand participating in computation of y. If k (the number of subsequent operation, operand pairs) is some small number brute force attack is possible. This is the same as trying to find an inverse function. But generally, it is unknown is F bijective. Later this will be explained in more details.

RC5 is one of the most innovative block ciphers, for the first time data-depend rotations are introduced. In Blowfish the idea of Key-Depend S-boxes is presented. In DES two bits from the block are used to chouse the S-box to be used. CRAB introduces usage of more different F functions. The BIG question is what is in common between them, what is common between secure designs already published? This thesis objective is to answer that question.

## 2. Polymorph Encryption (more math)

The cornerstone of mathematics is the idea of function. Analogy between function and algorithms exist, any algorithm is a function but not all functions are algorithms. This thesis is concerned with functions as algorithms. Later the properties of this 'receipt' will be discussed. This SQ2 round function written in mathematical stile is:

$$F(x) = ((...(x_0 \circ_{x_0} S_{x_0}) \circ_{x_1} S_{x_1}) \circ_{x_2} S_{x_2} ...) \circ_{x\,n-1} S_{x_{n-1}}$$

$$F_k(x,S) = \begin{cases} x_0 = x \\ x_k = x_{k-1} \circ_{[x_{k-1}]} S_{[x_{k-1}]} \end{cases}$$

Here o is one of the basic arithmetical operations, and S is the key array. Nothing similar to this has ever entered math books; there are recurrent functions but functions with variable operations are out of the question. But in cryptography this king of functions are more than welcomed. Let "analyze" the properties of the hypothetical encryption system using this kind of F-function (SQ2 for example). The existence of inverse function SQ2 like functions is unknown. If $F_k(x)$ and S can x be computed? Let k=2 which means:

$$F(x) = (x_0 \circ_{x_0} S_{x_0}) \circ_{x_1} S_{x_1}$$

As implemented in SQ2 the two most significant bits are used to chouse the operation and the next 4 to chouse the operand. The value of F(x)=$x_2$ is available to the attacker, in order to find x, $x_1$ should be calculated first, but here comes the problem, the 6 most significant bits of $x_1$ were used to calculate $x_2$. In SQ2 the operation set is {+, -, *, XOR}. Reversing the function is only possible by guessing those 6 MSB and testing the assumption. For example if the assumption about the 6 MSB was 101011 then F' is assumed to be:

$$F'(x) = (x_0 \circ_{x_0} S_{x_0}) - S_{11}$$

The last operation is inverted:

$$x_1' = x_2 + S_{11}$$

Testing the assumption about the 6 MSB is by checking the 6-MSB of x'. If the 6 MSB of $x_1$' are not the same with the assumption then $x_1$ is certainly not what we have computed as $x_1$'. If the assumption shown to be correct there is no guarantee that $x_1$' is the correct value of $x_1$. It can happened some values of S to produce the same result for two distinct values of x. Generally F(x)=F(y) does not means x = y; F is not a bijective function and such functions are not reversible. It is very likely that randomly chosen values of S are going to result in non-bijective function. Let an assumption is made that most of the values of S result in bijective functions (even I can not prove that there is at least one such an array S). Since the only way to reverse F is by brute force, it is needed to estimate the effort; in this particular function it is $O(2^{6k})$. This is not a small problem for k=4 of bigger. Further discussion is pointless since knowing of the result and the key by definition means knowledge of x this is the decryption function. The other case, x and F(x) are known, S should be discovered. The same method is applied again. First an assumption is made for the 6 MSB of $x_1$, but here comes the BIG problem, now an assumption for S should be made but there are $2^{26}$ values of S[11] which subtracted from $x_2$ will result in a $x_1$ having the 6 MSB as assumed. This is simply too mach for brute force attack. So finding S having enough (x, F(x)) pairs is hard task. This is of great cryptographic value, since S is the Key.

## 3. General model for Polymorph Encryption F-function

Since SQ2 is not the only encryption algorithm that has [or going to] implement the Polymorph Encryption definition of general model for Polymorph F-function is needed. The simplest mathematical notation for this model will be:

$$f(x,K) = \begin{cases} c(x) = 0 \rightarrow h_0(x,K) \\ c(x) = 1 \rightarrow h_1(x,K) \\ . \\ . \\ . \\ c(x) = n \rightarrow h_n(x,K) \end{cases}$$

c(x) is a condition function and h is our set of primitive/elementary functions. There is an example.

$$f(x,K)=\begin{cases} c(x)=0 \rightarrow h_0(x,K) \\ c(x)=1 \rightarrow h_1(x,K) \\ c(x)=2 \rightarrow h_2(x,K) \\ c(x)=3 \rightarrow h_3(x,K) \end{cases}$$

$$H=\begin{Bmatrix} (x+k)(x-k) \\ (x+kx)x \\ (x+k)xor(x-kx) \\ x+((x+k)xor(k-kx)) \end{Bmatrix}$$

$$c(x)=x \bmod 4$$

Using this template and the Sq2 F-function is given below.

$$F_k(x,S)=\begin{cases} x_0=x \\ x_k=f(x_{k-1},S) \end{cases}$$

$$f(x,S)=h_{c(x)}(x,S)$$

$$H=\begin{Bmatrix} x+S_0, x-S_0, x*S_0, x\oplus S_0 \\ x+S_1, x-S_1, x*S_1, x\oplus S_1 \\ . \\ . \\ . \\ x+S_{15}, x-S_{15}, x*S_{15}, x\oplus S_{15} \end{Bmatrix}$$

The idea is simple to be described in math world. The Domain set is divided into n subsets according to the Condition function and to each subset different transformation rule is assigned. Instead of constructing H from simple functions entire algorithms can be used as given below:

$$f(x,K)=\begin{cases} c(x)=0 \rightarrow h_0(x,K) \\ c(x)=1 \rightarrow h_1(x,K) \\ c(x)=2 \rightarrow h_2(x,K) \\ c(x)=3 \rightarrow h_3(x,K) \end{cases}$$

$$H=\{DES(x,k), RC5(x,k), Blowfish(x,k), GOST(x,k)\}$$

$$c(x)=x \bmod 4$$

By incorporating this function in a standard Feistel Network a new block cipher is born.

## 4. Cryptoanalysis of Polymorph Encryption

The mathematical model of the Polymorph Encryption is discussed within Quasi Algorithms section. Today there is a lot of different attacks develop to extract the key form the knowledge of large quantity of (x, E(x,k)) pairs, where E(x,k) is encryption under key *k*. Differential and Linear

Cryptoanalysis are certainly the most commonly used. Differential Cryptoanalysis examine the difference propagation between pairs of plaintexts and ciphertexts. In order to discover the key the encryption function is carefully examined for non-uniformity in difference propagation. There is a significantly different circumstances in Polymorph Encryption. Because F is key-determined examining difference propagation in general is impossible. The difference propagation analysis of function F for a fixed key is possible. But analyzing the behavior of F for a given key will not lead to any general conclusions. Data-depend operations used in Polymorph Encryption cause the algorithm to be indeterminate. Instead of designing strong non-linear functions a.k.a. S-box Polymorph Encryption is making a totally different approach. The goal is not to make the system resistant to the given attack but to make the system incompatible to the attack [plant virus is not going to cause any damage on the ants, but the biggest tree can die from the infection and vise versa]. The same stand for Linear Cryptoanalysis, we can not discover the eventual linear dependencies between the plaintext, ciphertext and the key when the key is the algorithm itself. Shortly Polymorph Encryption stands for Encryption Algorithm = KEY. Different keys determine different encryption algorithms.

Here is small review of some well known design solutions that incorporate Polymorph Encryption – like solutions. One of design solutions in DES is especially worth of mention. Before entering the F function the block is expanded from 32 to 48 bits and it is split in 4 sub-blocks each 6 bits. The MSB and LSB of those sub-blocks are used to chouse one of 4 S-boxes available. This is very simple implementation of data dependent-operation but as the 'history' has shown just enough to defeat Differential Cryptoanalysis. In RC5 Ron Rivest used data-depend rotations; this is the first explicit Polymorph-like design solution. As we know RC5 had survive a lot of analysis and still stand secure. Blowfish is an example of well-realized key-depend S-boxes. Key-depend S-boxes are nothing more that realization of Polymorph Encryption credo Encryption Algorithm = Key. Can Blowfish be analyzed without fixing the S-boxes?

Polymorph Encryption is nothing new, but certainly genius, even no one has ever formulated it as separate theory. The building blocks of security of some of the widest used ciphers are the principles here described under the name of Polymorph Encryption.

As an illustration a real live situation [not so real, let say form some American psychology institute]. You are placed in a new European car, an exclusive model, and the doctor tells you "You are part of our new experiment. There is a Russian nuclear bomb in your car, if you don't find the way to the Nuclear Tech Faculty in Iakoato, you are going to …. Now you are in Siberia and you have 120 hours to reach your goal" You the lab mouse, have never been in Russia and Russian is not your field of best interest. And here we go, what you are going to do? Start the car quickly. There is a cross-way but which way to chouse? Somehow you chouse one and the next cross-way bring the same dilemma. You try to examine the environment, there is a cow crossing one of your potential directions and the other one have a sleeping elephant on the road. You chouse one, the cow is more attractive. And so on, cossway after crossway. After 119 hours and 50 minutes you decide to stop and to wait for your destiny, there is no Iaokato city in Russia. Bum. You are awaked, this was just a simulation. Your shrink seem more than happy he have diagnosed you syndrome. Your choices on the crossways have shown what you fair and what you prefer. Your personality has driven you and you have drive the car just as intermediator.

The same thing happens when a block is entering the Polymorph Encryption Algorithm. There are crossways, the F-function, something must be done and the block value determine what it is going to be. After that the block came out of the function changed. The next F-function execution produces the same must-to-do-something situation. The output is a mix from the personality of the patient (the plaintext) and environment (the key).

## 5. Quasi Algorithms / Quasi Functions

Even this part of the thesis has a little practical value it is important to define the rules of the game. The general model has been already described. The starting point in theoretical definition of what have been presented as Polymorph Encryption is the definition of Algorithm. As the functions are the building block of mathematics Algorithms are the building block of everyday activity of ordinary people.

*Algorithm is a calculation procedure having these properties: It is consisted of finite number of steps / primitive operations [usually arithmetical and logical]. Each step is fully defined, the execution order is strictly determined [there is begging and end]. There is a finite need of space and time in order to get the work done [complexity of the algorithm].*

It is well known that each algorithm is a function, what is going to be described can be named 'quasi function' for the mathematicians and for everybody else as 'quasi algorithm'.

$$f(x) = x^3 + ax^2 + x/b - c$$

The first question, what is the algorithm to calculate this equation?

| | |
|---|---|
| $S \coloneqq 0;$ | $S \coloneqq 0;$ |
| $T \coloneqq x^3$ | $T \coloneqq x.o.3$ |
| $S \coloneqq S + T$ | $S \coloneqq S.o.T$ |
| $T \coloneqq x^2$ | $T \coloneqq x.o.2$ |
| $T \coloneqq aT$ | $T \coloneqq a.o.T$ |
| $S = S + T$ | $S = S.o.T$ |
| $T = x/b$ | $T = x.o.b$ |
| $S = S + T$ | $S = S.o.T$ |
| $S = S - c$ | $S = S.o.c$ |
| $f(x) = S$ | $f(x) = S$ |

Each step in the algorithm is one of the basic arithmetical operations. The essence of the algorithm is the step order, if you mix the steps you will gain something that maybe unuseful. Because of this the algorithm can be defined as Order-of-steps + definition-of-each-step = algorithm. Each step carries information about several things, the operands, the operation and where to store the result. One more abstraction can be done, the skeleton of the algorithm can be extracted by abstracting the operations. The other case, abstracting the operands is absurd. All the arithmetical operation (addition, multiplication, subtraction, division, xor, and …) can be substituted with 'o' – general operation.

The result is not an algorithm, we have no idea what should happened in those steps, but we do know the order of steps, we know which data is affected in each step. This generalized algorithm or skeleton is just a template telling us everything about the conditions and environment in which we should calculate something but on our free will is to chouse some meaningful set of operations.

If we write the example function in prefix notation we have:

$$F(x) = -((+(+(+(\wedge(x,3),*(a,\wedge(x,2))),*(a,\wedge(x,2))),/(x,b))),c)$$

With an abstraction of the operations substituting tham with the general operation o, we get:

$$F'(x) = o_1(o_2(o_3(o_4(o_5(x,3), o_6(a, o_7(x,2))), o_8(a, o_9(x,2)), o_{10}(x,b))), c)$$

In order to turn this skeleton in a function we need to substitute those $o_j$ with some of the available binary operations, which our famous mathematicians have canonized thousands of years ago {+,-,*,/,^,q}. After this substitution is done our skeleton is turned into algorithm / function. Our example function have 10 o, *operation places* where we can insert the operations we chouse. Our set of operations is very small only 6 of them. However on each operation place we can put any of them which result in having $6^{10}$ different functions with the given skeleton. The function we started with is just one of them. Having this short discussion in mind we can define Quasi Algorithm:

*Quasi Algorithm is a calculation procedure which is determined by the order of step of primitive arithmetical transformations, for each step it is uniquely determined which data [operands] are involved in the transformation and where the result is stored. Actual primitive operations taking place in every step are not specified.*

The difference between algorithm and quasi algorithm is obvious. As an input to the quasi algorithm besides the argument x [maybe more of them] we pass an information about the primitive operations which should be executed in each step. The quasi function conjugated to our example is:

$$Q(x) = x\ o\ 3\ o\ a\ o\ x\ o\ 3\ o\ a\ o\ x\ o\ 2\ o\ x\ o\ b\ o\ c$$

Because this is not a function concerning our mathematical heritage we need to make a little expansion. We define a general overlay function, Φ, which play the role of quasi algorithm processor. The quasi function, the arguments and the information about the primitive operation to be executed in each step are the arguments passed to the overlay function.

$$\Phi(F, \sigma, x)$$

Here sigma is a variation over the set of available primitive operations {+,-,*,/,^,q} [q is notation for n-th root function, ^ is power function] For each possible value of sigma Φ present a function that have the F skeleton. Our example function is writen in this notation as: Φ( F,(-,+,+,+,^,*,^,*,^,/),x). Just to note that the variation sigma is of n-th degree where n is the number of operation places in the skeleton F. Expanding the notation our f is now:

$$\Phi(F, \sigma, x) = F_\sigma(x) = x\ \sigma_1\ 3\ \sigma_2\ a\ \sigma_3\ x\ \sigma_4\ 3\ \sigma_5\ a\ \sigma_5\ x\ \sigma_6\ 2\ \sigma_7\ x\ \sigma_8\ b\ \sigma_9\ c$$

For a given value of sigma Φ is a function in a form we are used to see in our math books. Further generalization is possible, instead of sigma we can place any variation-generator-function which as arguments accept the set of possible operation, the degree of the variation and a number, the result is a variation in form we need.

$$\Phi(F, \Sigma(A, |F|, g), x) = F_\Sigma(x)$$

The set A can be the default set of operations {+,-,*,/,^,q} or any other set of binary operations which can be user defined. Binary functions can also be included in the set if we chouse to accept them as primitive operations. The second argument is |F| or absolute value of the template or degree of freedom which is defined as number of operation places in the template need to be filed.

The overlay function Φ is a family of functions in respect to the different values of sigma parameter. Here is example number 2.

$$F(x) = (1 \circ 1 \circ x) \circ x$$

Using this template the overlay function is:

$$\Phi(F, \sigma, x) = F_{\sigma}(x) = (1 \sigma_1 1 \sigma_2 n) \sigma_3 n$$

Mathematics has developed a wide specter of methods and boring theorems how to analyze the behavior of functions. It is now my intention to show how useless that grandiose theory can be. Just a simple task to find:

$$\lim_{x \to \infty} \Phi(F, \sigma, x) = ?$$

Certainly sigma can have any of the possible values. How to calculate this limit? The easiest solution to run over all the possible values of sigma, filling the set of possible limits for the given skeleton. The solution is:

$$\lim_{x \to \infty} \Phi(F, \sigma, x) = \{-\infty, 0, 2, e, +\infty\}$$

But brute-force approach is exactly what the math wants to avoid, instead of empirical conclusions about something we usually tend to develop a model to calculate the result instead of guess-and-check approach. Simply Brute force is out of question, just imagine the example 1 function and the quest for integral.

## Conclusion about Quasi Algorithms

Even here nothing more than basic idea and little math notation have been presented I am sure Quasi Algorithm are worth of interest and research. Quasi Algorithms Theory is very brave term for what is here described but future research are certainly going to show the importance of this ‘theory’. A quick summary:

1. Quasi Algorithms are real thing they exist.
2. Every algorithm belong to a class of functions determined by its skeleton
3. Algorithm = Quasi Algorithm + Operations-to-be-inserted
4. Today stage of development of math can not cope with Quasi Algorithms
5. We hope that will be the case in near future
6. Cryptography is number one benefic of this theory

I am not so sure about other fields of science but cryptography can certainly get the most form Quasi Algorithms. Polymorph Encryption described in this thesis is just a example how to use the theory of Quasi Algorithms. If we figure out how to analyze Quasi Functions that will be a plus for every possible field of science [at least those involving math]. Fining templates and patterns is what we are trying so hard to teach our computers to do so. Quasi Algorithm is theory that separates the pattern from the actual implementation, and that is really a simple task. Analyzing those patterns will enable to investigate the behavior of whole class of problems instead of separate specimens. Dividing problems in classes is certainly the main objective of every science. There so very interesting question that can be derived from the theory of Quasi Algorithms:

1. If there are inverse functions for all or part of the function having skeleton F, are these inverse functions share a common skeleton?

2. Is it possible to reconstruct the function if we have its skeleton and finite number of pairs $(x,F_s(x))$ [there is similar problem when we have the n-th epitome of the function and some finite number of pairs $(x,f(x))$ ]?
3. Can we expand the available mathematical theory in order to include Quasi Algorithms?
4. Are there classes of Quasi Algorithms that show some special properties?

This thesis not tend to be complete nor present some class-one-importance discovery, in contrary I am trying to formulate the idea and to scratch the potential use of it.

## Cryptoanalysis of Polymorph Encryption trough Quasi Algorithms

The revolutionary concept introduced by Quasi Algorithms in cryptography is not just a change in notation but a deep movement in understanding the process of encryption at first place. Standard model of encryption using S-box or some other non-linear structure tend to use fixed functions incorporated in Feistel Network and here is a new 'secure' block cipher. But 'secure' was certainly not a characteristic for more than 99% of block cipher proposed until today. Polymorph Encryption introduces something totally different. Using the equation of general overlay function $\Phi$:

$$\Phi(\mathrm{C}, \Sigma(\mathrm{A},|\mathrm{C}|,(\mathrm{x},\mathrm{k})),(\mathrm{x},\mathrm{k})) = C_{\Sigma}(x,k)$$

The sigma function use the same argument as C. The direct consequence is that every value of x and k make the decision which function from the family C [the encryption Quasi Algorithm] they are going to be processed trough. This is similar to the situation to encrypt a plaintext under the plaintext itself as a key. As stated before the main strategy against the cryptoanalysis is to make cipher incompatible to them. This certainly does not imply that there will be no attack discovered against Polymorph Encryption but our current mathematical understanding of things is just not enough to even start a basic analysis of Polymorph Encryption and Quasi Functions.

In order to understand Quasi functions as implemented in Polymorph Encryption here is a little illustration. The set of input values is divide into subsets according to the function C. To each subset different function from H is ascribed. Since F function is executed more than once we there is hoping from one subset to other. Let $x_k=\Phi(F,\Sigma(A,|F|,(x_{k-1},K),(x_{k-1},K))$ naturally $x_{k-1}$ belong to some subset according to the division in respect of C. After the execution of the corresponding function $x_k$ may not belong to the same subset any more. If this is the case, and in practice this effect is exactly what make the cipher secure, then calculating $x_{k+1}$ from $x_k$ is going to be according to a different rule than $x_{k-1}$ to $x_k$. All the functions in H should be distinct and all of them equally probable to be chosen. Also they must have different difference propagation, clearly it is not the same to multiply x with other number or just to add them or xor them. The effect from this subset hoping produces the uncertainty in deference propagation shown by the Polymorph F-function.

## Supporting info and contact

This thesis make a extensive reference to algorithm and other thesis, since it will be very long and boring reading to include all of them in one place here is a index of my prior works somehow referenced in this thesis:

1. ANIGMA cipher, there are three separate documents an outcome from that project: ANIGMA, MEX128 and Variable Function Technology, in the last one is the original description of what is now Polymorph Encryption
2. A2 cipher, containing some discussion how to implement Polymorph Encryption into the F-function
3. SQ1 there an interesting survey on implementing Polymorph Encryption in Stream Cipher Design even the thesis major objective is information loss theory.
4. SQ2 used here as an example of Polymorph encryption. This one of my major works implementing all the theory explained here.
5. SQ3 a.k.a. Z876 is the newest design also using Polymorph encryption but in some alternative approach than discerned here. I recommend you to read the documentation about this cipher together with SQ2 they are typical specimens of Polymorph Encryption. SQ3 have no F-function as usually used instead there is change in Feistel Network.

All this document are publicly available trough my web sites in research section:

http://kbajalc.8m.com
http://eon.pmf.ukim.edu.mk
http://home.cyberarmy.com/kbajalc

I hope you are going to find this interesting, any help is welcome in order to speed up the research. Contact information is listed below:

Kostadin Bajalcaliev
Ul. April 26th Street #14
1480 Gevgelija
Macedonia
Europe

Kbajalc@freemail.org.mk
Kbajalc@eon.pmf.ukim.edu.mk

Please write a message for any suggestion you may have.