Attack-aware Self-logic-transformation Cryptographic Algorithm

Introduction.

this is a preview of the cryptographic algorithm that is based on AI technology. existing cryptography such as DES, AES, ECDSA, it is based on mathmathics and from it designed algorithms, encounters their limitation from the quauntum computing's powerful operation ability.

the algorithms based on mathmathics and its utilization with computing for the cryptography on the system is proved method by many expert. and currently, in many area in which systems need data encryption and decryption from that another area such as public key infra has been used for the purpose.

but, they can encounter expected limitation from internals of themselves.
many experts say that the algorithms utilized in various area to improve the security level of HW or SW based system can be broken by the quauntum computing's ability. the ability is based on the qubit, it has possibility to be determined by the instrument device, that is unchanged but its status being determined by the device can be changed. and from it, the ability is decided depending on the number of qubit.

AI technology based on the complicate operations and massive datas, the AI algorithm model use them to design and train the model and infer something for the purpose it want to achieve, is already utilizing in diverse industry.
with them, also parameter tuning is perceived as one of consideration to improve the model.
the operations, AI technology uses internals, highly complicated-form at the entire view. it is combined each other together and many operations are used to form the AI model.

under this circumstances, I want to propose this algorithm.
the main idea starts with combining the cryptography and AI to overcome the limitation of existing cryptographic algorithm from quauntum computing.

Expected Effect.

first of all, if this algorithm could have the proved ability to overcome the limitation and achieve to the goal, I think, this could be one of way to advance the cryptography.

this algorithm should meet these condition to achieve the goal of it.

attack-aware

with the AI operations and internal's operations, this algorithm should have ability to determine coming data or input could be attack or not.

self-logic-transformation

with the AI operations and internal's operations, this algorithm should have ability to transform internal's operations or its algorithmic logic.

cryptographic operation

with the AI operations and internal's operations, this algorithm should have ability to correctly decrypt the encrypted-data.

from under this conditions, expected effect will be below.

enhanced cryptographic performance
the complicate AI operations could be useful to hide internals operation comparing the existing algorithm.
it can means that AI makes internals more densely complicate.

also, this algorithm makes crpytographic strength more stronger.
one of features change itself when attack is coming.
it means the complexity of this algorithm could be self-changed from it the difficulty to break crpytographic operations will be higher.

low running-time comparing with existing algorithms
I cant accurately predict the running-time of this algorithm.
but I think the time of this algorithm needs more than existing algorithms.

encryption/decryption without cryptographic key
the key of cryptography is on how to safely keep the crpytographic key used for

the algorithm.
but, AI operations need a lot of parameters so also it could hide the key inside the model.
it means the parameters will be used by itself as key and the key will be changed by itself.

Expected Restriction.

at top view, this algorithm should have these ability as its functionalities to fully run and defend itself from attack.

attack-aware
in this phase, when data comes to this algorithm, the algorithm should get the data as itself and determine whether the data is attack or needs to be encrypted or decrypted.

if this determines the data as attack, it should pass the output to the next.
if this determines the data to be encrypt or decrypt, it should pass the data as itself to the next.

also, it should have ability to make itself more stronger by this ability it have to defend itself from the another expected-attack by malicious attacker to break this phase.

self-logic-transformation for cryptographic operation
in this phase, when the datas passed by the front phase comes to, this algorithm should get the data as itself and determine whether it have to change the operations inside this phase or have to encrypt or decrypt the data.

if the passed data is attack, it should change the operations in this phase.
if the passed data is pure, it should run cryptographic operations to encrypt or decrypt the data.

to continuously defend the attack, when it change the operations its goals should focus on how to enhance strength.

from under this conditions, expected problem will be below.

identifying attack
the data fed into this algorithm is considered as pure itself.
it should be determined as one of attack or data to be processed by this

algorithm.
under this condition, the data should be fixed-size.
if data is bigger or smaller, necessary tasks should be.

the factors to determine whether the data is attack or not should only be on this algorithm itself.
if others are combined with this algorithm for attack-aware, it will impact on this algorithm by changing the operations in each phase so entire operations in this algorithm will be changed.
therefore, the algorithm in attack-aware should be fixed-form or be redesigned if there is needs to customize this.

the output passed to next phase doesn't have to be complicate but should be fixed-size for processing in next phase.
its purpose is notifying the type of input data to next phase.

enhancing strength of each phase
the factors utilized to enhance strength of each phase should be on this algorithm itself and they should interact with each other at the entire operations.

the operations should be changed to enhance strength of this algorithm.
therefore, feedback should be at each phase.

self-transformation with purpose of correctly decrypting the data
when attack comes, the operations or algorithmic logic at each phase will be changed by itself.

the main purpose of this functionality is not on changing itself but on how to correctly decrypt the encrypted data under with changing operations and parameters used as key.

DESIGN.

under this conditions, this algorithm will have below structure but it can be changed.

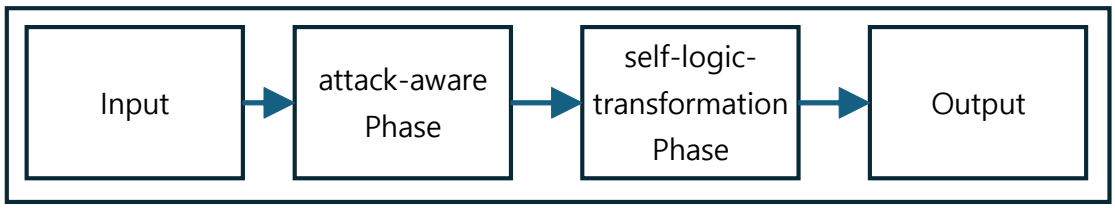


Figure 1. overview of algorithm.

the Input is pure data but it can be attack by malicious attacker.
the Output is outcome of this algorithm and it can be one of encrypted data or decrypted data by this algorithm.

attack-aware phase will have below structure.

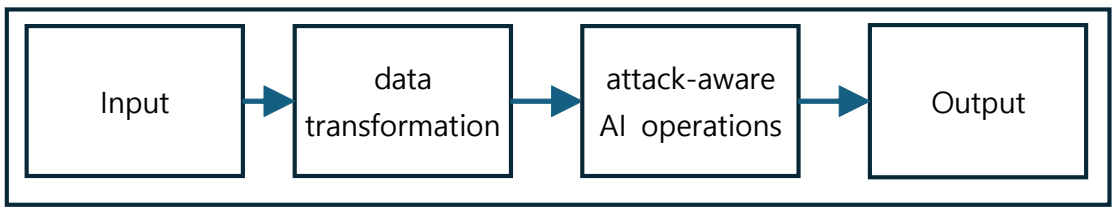


Figure 2. top view of attack-aware phase.

at top view of attack-aware phase, it runs AI operations for the purpose.
in this phase, main purpose is determining that attack from data exists.
then if the attack appears it should run operations to enhance its strength.
if attack doesn't appear, pass the data to next phase.

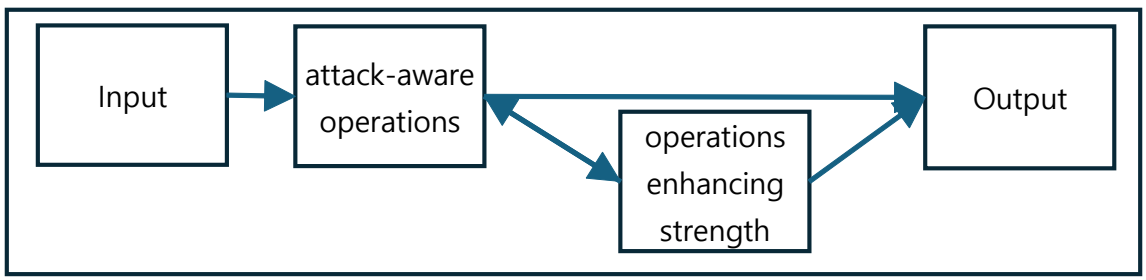


Figure 3. middle-level view of attack-aware phase.

attack-aware operations will be consisted of some layers.

self-logic-transformation phase will have below structure.

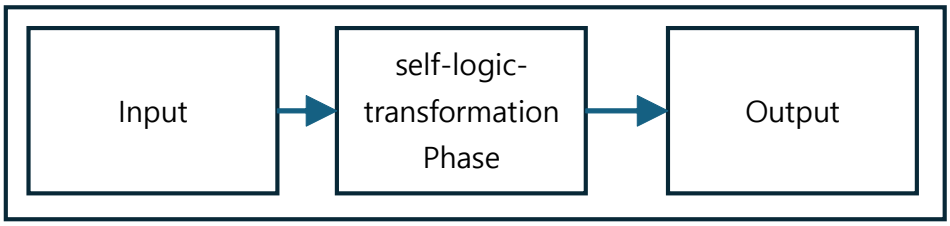


Figure 4. top view of self-logic-transformation phase.

at top view of self-logic-transformation phase, it runs AI operations for the purpose.
in this phase, main tasks are cryptographic operation transformation for encryption and decryption, operation enhancing strength.

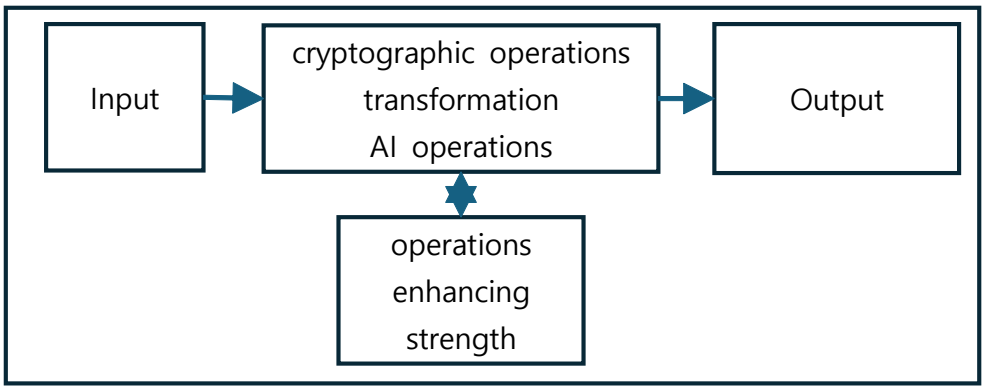


Figure 5. middle-level view of self-logic-transformation phase.

low-level view of each phase will be added.

Expected Problem.

from design, there are a lot of expected problems before go inside to implement this algorithm.

the problems are described below.
almost all of problems exist at the main phases, attack-aware and self-logic-transformation,

[*], represents each stage or phase at the top-level view.
[], represents a set of data-types can be existed in stage or phase.
[!], represents one of the expected problems.
[#], represents one of the solutions or options in each stage or phase.

[*] under Input, Output

nothing to be problem.
Input passes the data from the input device or system to the attack-aware phase and Output displays the output from the self-logic-transformation phase.

[data types of Input]
data to be encrypt, encrypted data to be decrypt, attack-data.

[data types of Output]
encrypted data, decrypted data.

[*] under attack-aware phase

many problems or various difficult level to implement can be in this phase.

Input

nothing to be problem.
Input passes the data from the front stage to the data transformation stage.

data transformation

[!] how to structure the data.
[1] data length per block.
data length per block can be arbitrary value.
but, it should matched with the attack-aware operations stage.

for the compatibility with existing algorithms, options can be same with the algorithms.

[!] how to pass the data.
[2] pass data-block in serial.
[3] pass data-block in parallel.
if the data-type is attack-data and pass the data in parallel, operations inside the attack-aware operations stage can be weak.
although serial's running-time is longer than parallel but it is proper for purpose.
the effect from attack to this algorithm should diffuse entirely and by that this algorithm should be protected from various attack.

attack-aware operations

[!] how to structure the layers and operations to determine the data-type.
[4] multi-layer and complicate operation.
to achieve high security level, operations inside this stage should be complicate.
but the trade-off between operations and strength should be considered.

[!] how to get the output from the operations enhancing strength stage and use it to enhance strength of attack-aware operations.
[5] pass the output in all-in-one.
[6] effect of output and how the output impact strength.
it is very difficult to predict the relation between the operation or output and strength of this stage.
one of the solutions is analyzing the relation with various experiments.

[!] how to structure the output from this stage and pass it to the Output and the operations enhancing strength stage.
[7] pass the output itself to each stage
even though memory usage will be higher, passing the output as itself is proper.

operations enhancing strength

[!] how to format the output from this operations.
[8] it is similar with [6].
if this output itself highly impacts strength, the trade-off should be considered.
if not, new approach will be required.

[!] how to design the internals.
[9] this stage's purpose is enhancing strength of the attack-aware operations stage.
therefore, it doesn't have to be complicate like the front stage.
but, the possibility should be opened.

although the internals of this stage opens to attacker, the security of the algorithm can't be broken.
because the activity to analyzing this algorithm with data will change the internals of this algorithm.

Output
nothing to be problem.
Output passes the data from the front stage to the self-logic-transformation phase with the data-type determined by the attack-aware operations.

[data type of Output]
original data and its type.

[*] under self-logic-transformation phase
many problems or various difficult level to implement can be in this phase.

Input
nothing to be problem.
Input passes the data from the front phase to the cryptographic operations transformation AI operations stage.

cryptographic operations transformation AI operations

[!] how to design the internals.
[10] the main is how to design the cryptographic operations using AI operations.
the AI operations are very complicate and its structure is in multi-layers.
it interact with each other and if it is moving from layer to layer, the parameters also change and effect the output.

the difficulty to implement this is more higher because the transformation is added to this stage.
therefore, it makes maintaining the consistency for decryption more difficult.

the difficulty to implement this is also more higher because the original data doesn't have any information about this stage regardless of its type.

the difficulty to implement this is also very higher because when the attack has detected the operations or parameters in this stage will be changed.
it could means the attack breaks the consistency randomly.

in perspective of cryptography, although the data is encrypted by this algorithm, if

there is no ability to correctly decrypt it, this algorithm is useless.

under this conditions, the detail design of this stage doesn't have meaning.
maintaining the consistency for decryption in this algorithm is highly complicate technical-issue.
but, having entire view to this stage is necessary.

[!] does this algorithm supports the data encrypted by other algorithms.
[11] decrypting the data encrypted by other algorithms is except from this algorithm.

[!] how to make the data structure used in AI operation.
[12] first, original data split as unit will be needed.
some parameters or information are also needed in the structure but details should be described after implementing this stage.

[!] how much layers will be used
[13] the number of layers will be decided depending on the data structure.
some experiments might be needed.

[!] how to encrypt the data and how to output the encrypted data.
[14] details should be described after implementing this stage.
but some tricks might be needed before the encrypted data is outputted.

[!] how to decrypt the encrpyted data by this algorithm without any information.
[15] details should be described after implementing this stage.

[!] how to transform internals of this stage.
[16] it is same with [15].

operations enhancing strength

[!] how to format the output from this operations.
[17] it depends on [12].

Output
nothing to be problem.
Output passes the data from the front stage to the end-Output stage.

[data type of Output]
encrypted data, decrypted data.

MODULE DESIGN.

until this, overall terms used in top-level and middle-level has came out.
from it, the modules in each phase, attack-aware and self-logic-transformation, are described with the definition of terms used in the phase and its internal structure.

[>#], represents one of conditions or reasons to be considered at each phase or stage.
[<#], represents approach or solution for [>#].

attack-aware phase

this phase's input length is same with the Input's length.
this phase will be consisted with three main stage, data transformation and attack-aware operations, and operations enhancing strength.

below, the term "module" means the term "stage".

data transformation

this module exists to process the input data.
the input data's format is considered as binary data and could be one of various encoding format such as UTF-8, UTF-16, UTF-32 and etc.

the reasons this module exists in front of the attack-aware operations module are described below.

[>1] its length could be longer than block length.
[>2] its length could be shorter than block length.
[>3] its encoding format could be one of UTF-8, UTF-16, UTF-32 and etc.

from it, this module's tasks are described below.
[<1] split the input data into block units.
[<2] add arbitrary number to the input data to form block.
[<3] notify its encoding format to the attack-aware operations module.

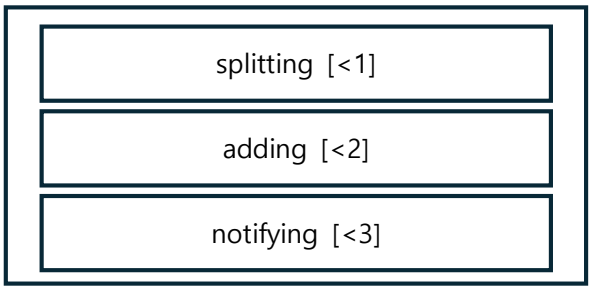


Figure 6. the data transformation module.

attack-aware operations
this module exists to determine the input data's format as one of three three types, data to be encrypted, encrypted data to be decrypted, and attack-data. with the purpose, strength of this module should be more stronger if attack has detected.

the meaning of enhancing strength in this module should meet below conditions.

[>4] sensitivity, adjusting the operations or parameters in this module entirely effects internals in this module.
[>5] complexity, adjusting the operations or parameters in this module should be more complicate to attacker.
[>6] consistency, the output of this module is consistent after adjusting the operations or parameters in this module.
[*] resistance, although attacker passes attack-data to this module and gets the output, predicting or inferring entire structure of this module and its internals should be difficult.

the reasons this module exists in this phase are described below.

[>7] handle the input's encoding format.
[>8] determine the input's type.
[>9] enhance strength of this module.

from it, this module's tasks are described below.

[<7] adjust the number of operation and parameter depending on the input's encoding format.
[<8] run the operations in multi-layer.
[<9] get the output from the operations enhancing strength module and apply it to this module.

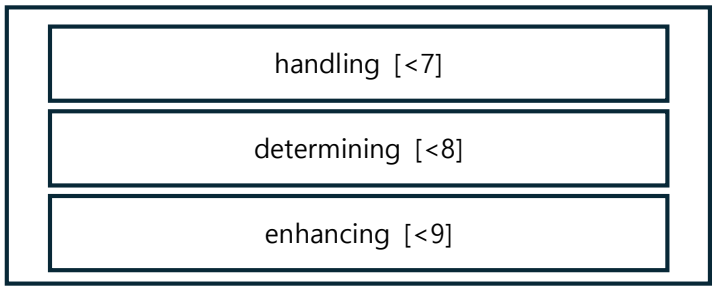


Figure 7. the attack-aware operations module.

operations enhancing strength
the module exists to enhance strength of the attack-aware operations module.

the reason this module exists in this phase is described below.

[>10] enhance strength of the attack-aware operations module.

from it, this module's task is described below.

[<10] get the output from the attack-aware operations module and run operations and return the output.

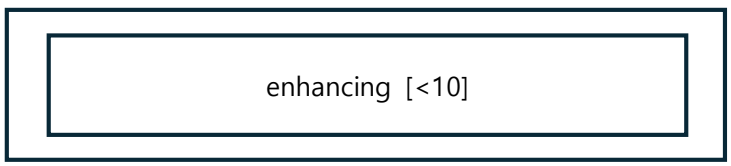


Figure 8. the operations enhancing strength module

self-logic-transformation phase

this phase will be consisted with two main stage, cryptographic operations transformation AI operations and operations enhancing strength.
but, some of stages could be added in this phase.

below, the term "module" means the term "stage".

cryptographic operations transformation AI operations
this module exists for data encryption, data decryption.
with the purpose, strength of this module should be more stronger if attack has detected.

the meaning of enhancing strength in this module should meet below conditions.

[>11] sensitivity, adjusting the operations or parameters in this module entirely effects internals in this module.
[>12] consistency, the output of this module is consistent after adjusting the operations or parameters in this module.
[*] replicabililty, although internals of this module is opened to attacker, entire structure of this module and its internals should not be replicated by the attacker.
[*] resistance, although attacker knows data and output from the data by this module, predicting or inferring entire structure of this module and its internals should be difficult.
[*] duplicability, although attacker copies the output by this module from other system, if the system's entire structure and its internals doesn't same with the system that makes the output, the output should not be correctly decrypted.
[*] flexibility, although attacker uses many data and knows the outputs by this module, from it decrypting other data copied or gotten from other system should be difficult.

the reasons this module exists in this phase are described below.

[>13] encrypt or decrypt the data.
[>14] enhance strength of this module.

from it, this module's tasks are described below.

[<13] run the operations in multi-layer.
[<14] get the output from the operations enhancing strength module and apply it to this module.

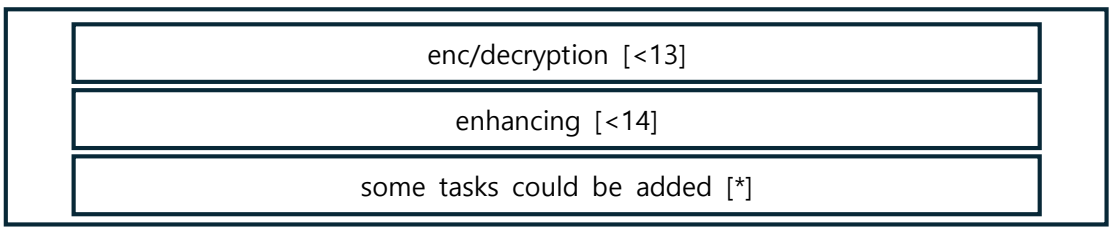


Figure 9. the cryptographic operations transformation AI operations module.

operations enhancing strength
the module exists to enhance strength of the cryptographic operations transformation AI operations module.

the reason this module exists in this phase is described below.

[>15] enhance strength of the cryptographic operations transformation AI operations module.

from it, this module's task is described below.

[<15] get the output from the cryptographic operations transformation AI module and run operations and return the output.

Figure 10. the operations enhancing strength module

TASK DESIGN.

from the module design, overall tasks in each module has came out.
before go inside more deeper, overall logical sequence or procedure of tasks used in the each module is required to be described although it can be adjusted in the implementation stage.

[#], is same with the task [<#].
[<<#], represents a logic of the task.

[1] splitting.

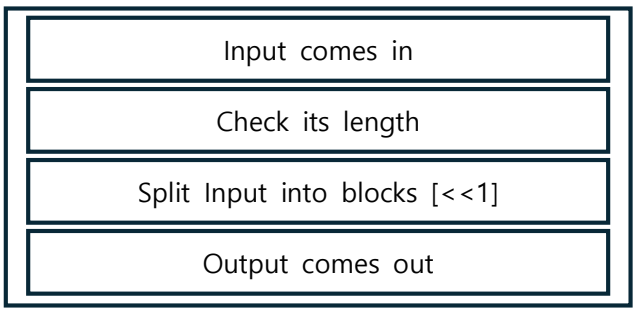


Figure 11. logical procedure of the task [<1]

[2] adding.

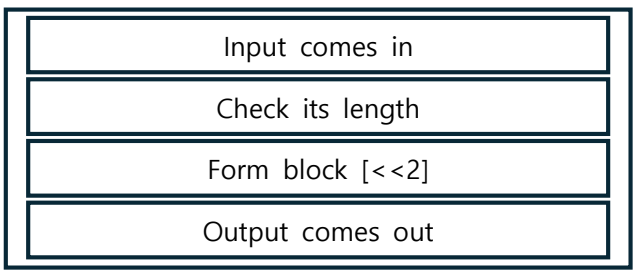


Figure 12. logical procedure of the task [<2]

[3] notifying.

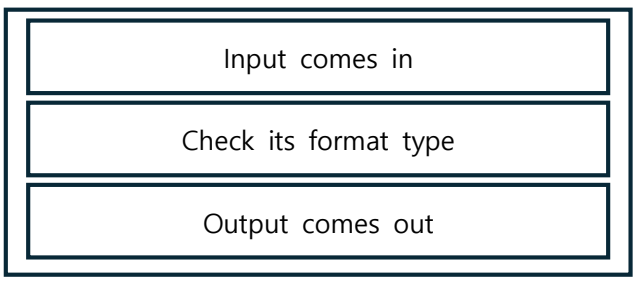


Figure 13. logical procedure of the task [<3]

[7] handling.

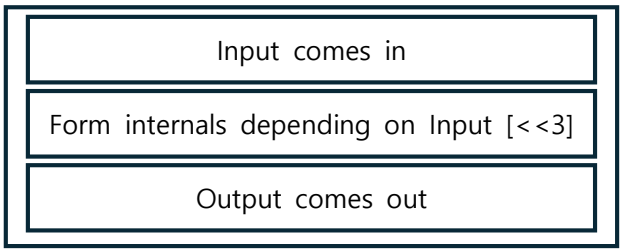


Figure 14. logical procedure of the task [<7]

[8] determining.

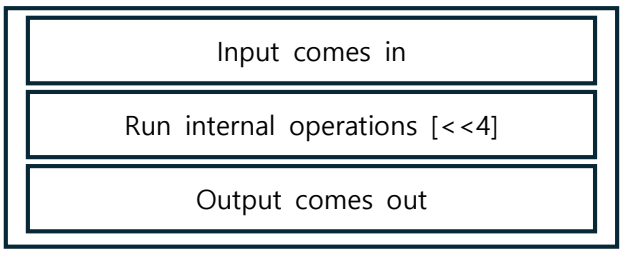


Figure 15. logical procedure of the task [<8]

[9] enhancing.

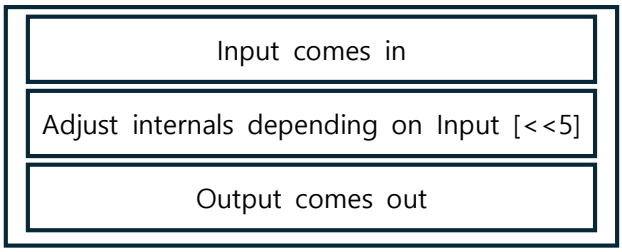


Figure 16. logical procedure of the task [<9]

[10] enhancing.

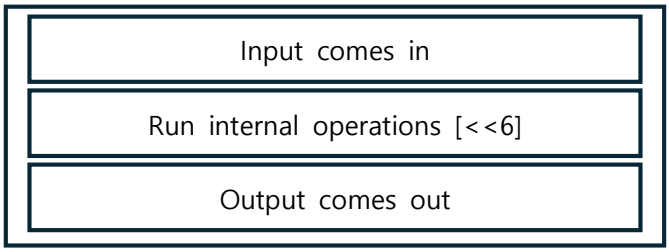


Figure 17. logical procedure of the task [<10]

[13] enc/decryption.

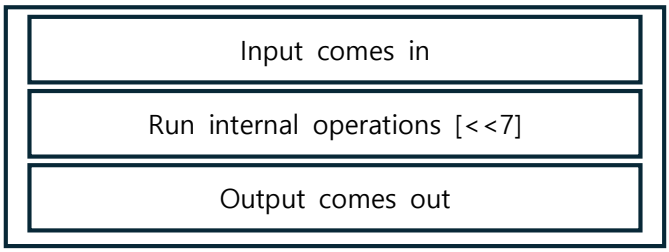


Figure 18. logical procedure of the task [<13]

[14] enhancing.

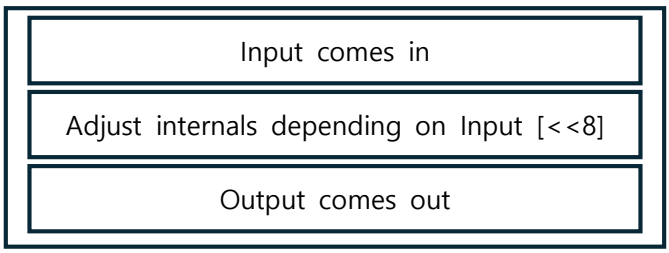


Figure 19. logical procedure of the task [<14]

[15] enhancing.

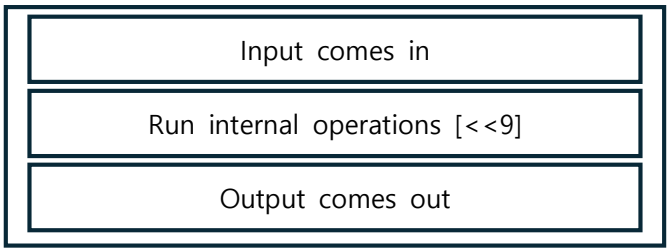


Figure 20. logical procedure of the task [<15]

LOGIC DESIGN.

from the task design, overall logic in each task has came out.
before go inside more deeper, overall algorithm sequence or procedure of logic used in the each task is required to be described although it can be adjusted in the implementation stage.

[#], is same with the logic [<<#].

[I], represents that instruction complexity should be considered.
[M], represents that memory complexity should be considered.

[I*], represents that operation layout or operation in the layout should be transformed.
[D*], represents that data layout or data and property in the layout should be transformed.

[>#-#], represents one of subtitle or issue should be considered in the logic.
[<#-#], represents approach or solution for [>#-#].
[*], represents one of considerations that isn't decided yet.

[*1], means the module "attack-aware operations".
[*2], means the module "cryptographic operations transformation AI operations".

[1] Split Input into blocks

[>1-1] block size
[<1-1] the block size will affect the entire number of operations in internals of the modules [*1], [*2] by the logic [3].
from it, the entire number of layers and data structures for the operations will be decided.

[2] Form block

[>2-1] block size
[<2-1] same with [<1-1].

[3][I][M] Form internals depending on Input

[>3-1] the number of layer to be formed and their relation
[<3-1] the layer's functionalities should have at least below purpose for the logic

[4], [7].
Accepts block, this means that the one of layers in the modules [*1], [*2] should have purpose to accept the block as input.
Runs the logic [4], [7], this means that some of the layers in internals of the modules [*1], [*2] should have purpose to process the logic [4], [7].
[*] Backs or Tunes output, this means that some of the layers in internals of the modules [*1], [*2] should have purpose to back the output to the one of front layers or tune the output.
[*] Tricks, this means that some of layers in internals of the module [*2] should have purpose to do some tricks for the logic [7].

[>3-2] the number of operations to be formed in the layer and their relation
[<3-2] from [<3-1], the operation's functionalities should have at least below purpose.
Accepts processing-unit, this means the operations should have purpose to accept the processing-unit from the block.
Runs operation with processing-unit, this means the operations in some layers should have purpose to process the unit for the modules [*1], [*2].
Owns data structure for the operation, this means the operation should have purpose to own its data structure.
Passes the data structure, this means the operation should have purpose to pass the data structure to other operation in the next layer.

[>3-3] data structure of parameters to be formed
[<3-3] from [<3-2], the data structures should have at least below purpose.
Owns processing-unit, the means the data structure should have purpose to own its processing-unit.
[*] Owns some information, this means the data structure should have purpose to own some information for the logic [4], [5], [7], [8].

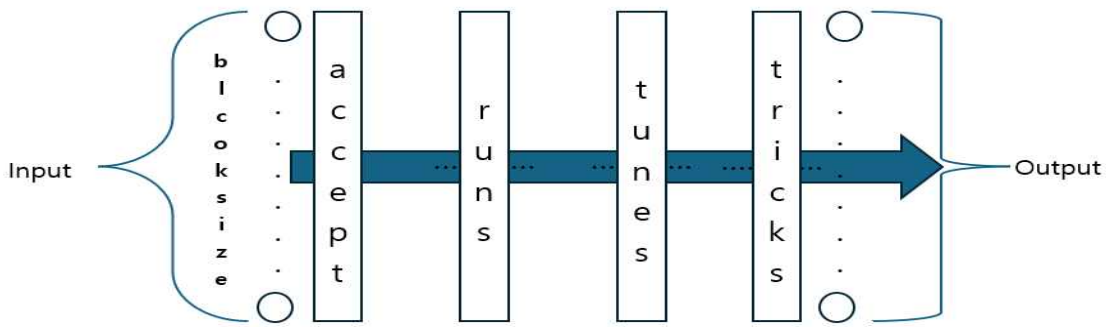


Figure 21. internals to be formed by the logic [3].

[4] Run internal operations

[>4-1] the relation between output from operations and output to be determined
[<4-1] the operations should have at least below purpose.
Owns the data structure, this means this entire operations in this logic should have purpose to own the data structure from the logic [3], [5].
Determines the output, this means the entire operations in this logic should have purpose to determine the output.
Passes the output, this means this logic should have purpose to pass the output to one of the logic [5], [7]
Requests the enhancement from the logic [6], this means that attack has came in and from it to defend internals of the module [*1] new internal's information will be needed.
Adjusts internals, this means that attack has came in and from it to defend internals of the module [*1] new internal's structure will be needed.

[5][I*][D*] Adjust internals depending on Input

[>5-1] the number of layer to be transformed and their relation
[<5-1] different with [<3-1], the layers to be transformed should have at least below purpose.
Transforms the number of layer with the information from the logic [6], this means the number of entire layer used in internals of the module [*1] will be transformed with the output from the logic [6].

[>5-2] the number of operations to be transformed and their relation
[<5-2] different with [<3-2], the operations to be transformed should have at least below purpose.
Transforms the number of operation with the information from the logic [6], this means the number of entire operation used in internals of the module [*1] is consistent but the number of operation used in the each layer will be transformed with the output from the logic [6].

[>5-3] data structure of parameters to be transformed
[<5-3] different with [<3-3], the data structure to be transformed should have at least below purpose.
Transforms the data structure with the information from the logic [6], this means the information of data structure used in internals of the module [*1] will be transformed with the output from the logic [6].

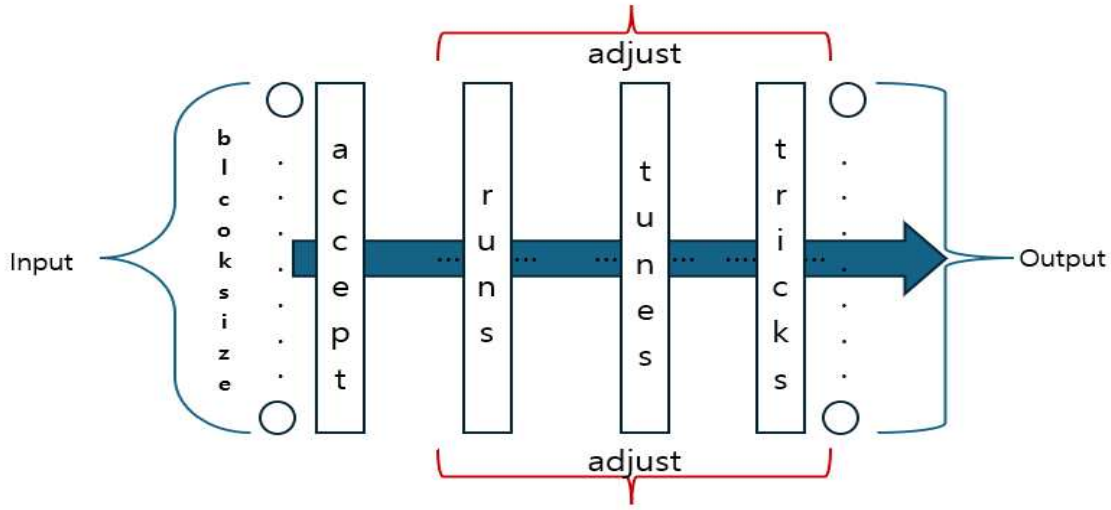


Figure 22. the location to be adjusted by the logic [5].

the tricks in Figure 22. should be removed.

[6] Run internal operations

[>6-1] the relation between input and output
[<6-1] this logic should have at least below purpose.
Owns some information, this means this logic should have purpose to have some of data structures passed from the logic [4].
Determines the output, this means the output from the entire operations should have purpose to determine the output to enhance strength of internals in the module [*1].

[7] Run internal operations

[>7-1] the relation between output from operations and output to be enc/decrypted
[<7-1] the operations should have at least below purpose.
Owns the data structure, this means this entire operations in this logic should have purpose to own the data structures from the logic [3], [8].
Encrypts or Decrypts the data in the data structure, this means the entire operations in this logic should have purpose to encrypt or decrypt the data in the data structure.
Requests the enhancement from the logic [9], this means that attack has came in and from it to defend internals of the module [*2] enhancement will be needed.
Adjusts internals, this means that attack has came in and from it to defend internals of the module [*2] new internal's structure will be needed.

[8][I*][D*] Adjust internals depending on Input

[>8-1] the number of layer to be transformed and their relation
[<8-1] different with [<3-1], the layers to be transformed should have at least below purpose.
Transforms the number of layer with the information from the logic [9], this means the number of entire layer used in internals of the module [*2] will be transformed with the output from the logic [9].

[>8-2] the number of operations to be transformed and their relation
[<8-2] different with [<3-2], the operations to be transformed should have at least below purpose.
Transforms the number of operation with the information from the logic [9], this means the number of entire operation used in internals of the module [*2] is consistent but the number of operation used in the each layer will be transformed with the output from the logic [9].

[>8-3] data structure of parameters to be transformed
[<8-3] different with [<3-3], the data structure to be transformed should have at least below purpose.
Transforms the data structure with the information from the logic [9], this means the information of data structure used in internals of the module [*2] will be transformed with the output from the logic [9].

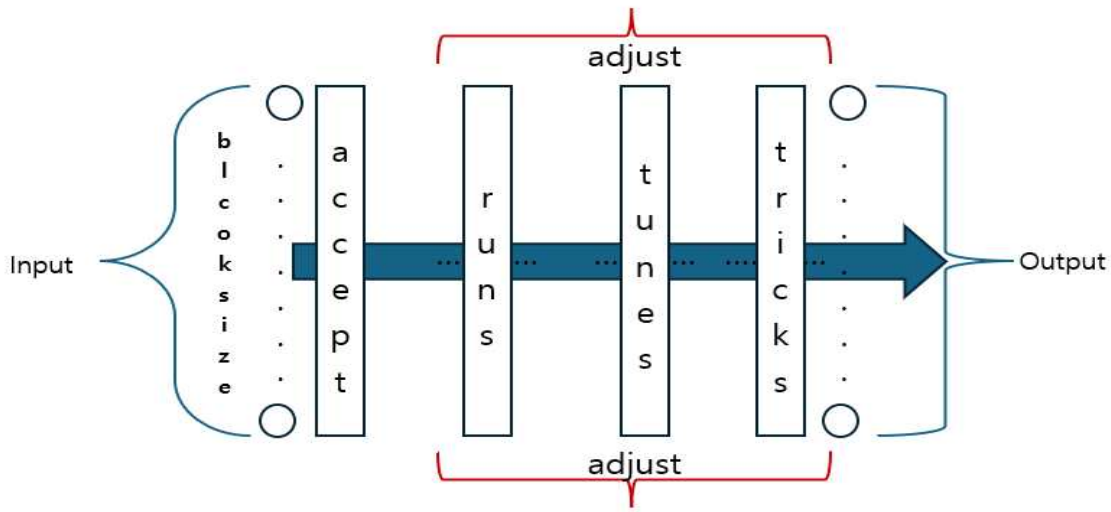


Figure 23. the location to be adjusted by the logic [8].

[9] Run internal operations

[>9-1] the relation between input and output
[<9-1] this logic should have at least below purpose.
Owns some information, this means this logic should have purpose to have some of the data structures passed from the logic [7].
Determines the output, this means the output from the entire operations should have purpose to determine the output to enhance strength of internals in the module [*2].

Unexpected Problem.

from the logic design, overall internal structure has came out.
but, from it new problems that should be described have came out.
before design the operation and data structure used in the logic, defining the problems is required.
also, some terms should be corrected to clear their definition although revising this will be.

Definition of Term.

block, this means a grouped data that is consisted of small data to be processed in the aware unit and cryptographic unit.
block length, this means the number of small data that consists a block.

aware unit, this means a unit that has a small data and some information for the operation.
aware information, this means some information used in the aware unit.

cryptographic unit, this means a unit that has a small data and operation to encrypt or decrypt the data and some information for the operation.
cryptographic information, this means some information used in the cryptographic unit.

with this definition, below terms used in the logic design should be corrected.

the term “processing-unit” has to be replaced with small data.
the term “operation” has to be replaced with aware unit or cryptographic unit.
the term “some information” has to be replaced with aware information or crytographic information.

Problem.

[!], represents one of the expected problems.
[#] and [#-#], represents one of the solutions or options.

[*1], means the module “attack-aware operations”.
[*2], means the module “cryptographic operations transformation AI operations”.

[!] show the possibility to solve this or suggest other alternative.
when module[1] = A(), module[2] = B(), trick = T(), input = x, attack = c = 0(false)

or 1(true), feeds input with c(0) to this algorithm.
then, the result will be below.
module[2] = B(), y(x)[enc] = T(B(x)), module[1] = A().

after that, when module[1] = A(), module[2] = B(), x = T(B(x)), c = 0.
then, the result will be below.
module[2] = B(), y(x)[dec] = T(B(T(B(x))), module[1] = A(),

after that, when module[1] = A(), module[2] = B(), x = T`(B(x)), c = 1.
then, the result will be below.
module[2] = B`(), y(x)[res] = T(B`(T`(B(x)))), module[1] = A`().

after that, when module[1] = A`(), module[2] = B`(), x = T(B(x)), c = 0.
then, the result will be below.
module[2] = B`(), y(x)[dec] = T(B`(T(B(x)))), module[1] = A`().

[1] remind that this is deep learning that has complicate structure that is consisted of units in each layer and it uses massive data.
some parameters of information in each unit will be adjusted automatically by the logic and data.

if it could be done to take the approach based on mathematics to solve this, if it is possible, this algorithm is useless.
one of this algorithm's purpose is hiding internal operation in the structure and from it calculating the equation to break this algorithm or do something from the equation should be difficult.

the structure in the module [*1] exists to infer data's type and other structure in the module [*2] exists for cryptographic operation.
from the purpose of each modules, first thing I have to do is defining the operation of each unit.

R(x) of the module [*2].
Encrypt, this means to encrypt small data x by the logic.

	A	B	C	D	E	F	G	H
1	Encrypt	small data	layer	unit numb	variance	value	output	number
2		48	1	1	0.062069	255	48.06207	9
3		97	1	2	0.062069	255	194.1241	
4		48.06207	2	1	32.22222	255	40.14215	
5		194.1241	2	2	32.22222	255	226.3464	
6	Decrypt	40.14215	1	1	0.033773	255	40.17592	
7		226.3464	1	2	0.033773	255	197.7603	
8		40.17592	2	1	59.21967	255	49.69779	
9		197.7603	2	2	59.21967	255	1.979933	

Figure 24. example of encrypting a small data x

Decrypt, this means to decrypt small data x by the logic.

A	B	C	D	E	F	G	H
Encrypt	small data	layer	unit numb	variance	value	output	number
	48	1	1	0.013793	255	48.01379	2
	97	1	2	0.013793	255	194.0276	
	48.01379	2	1	145	255	96.5069	
	194.0276	2	2	145	255	84.02759	
Decrypt	96.5069	1	1	0.011078	255	96.51797	
	84.02759	1	2	0.011078	255	168.0773	
	96.51797	2	1	180.5345	255	138.5262	
	168.0773	2	2	180.5345	255	93.61181	

Figure 25. example of decrypting a small data x

above Figure 24 and Figure 25 show there is possibility to implement R(x) of the module [*2].
but the variance should be carefully considered and for R(x) of the module [*2] the structure in module [*2] should learn the variance from the structure and data.

the equations used for R(x) of the module [*2] are below.

Encrypt, [(number of unit in each layer / the layer's depth) * (small data in the unit + variance)] mod (small data's maximum value).

Decrypt, [(number of unit in each layer / the layer's depth) * (small data in the unit + variance)] mod (small data's maximum value).

T(x) of the module [*2].
Trick, this means to do something on encrypted data like digital signing by the logic.

[!] find a generalized equation for T(x)
[2] 2025.10.26. try and result.
for this, I take a small data 48 and other data 97 that are same number in Figure 24 and detail is described below.

x	E(x)	output(x)	Trick	half trick	real output(x) = output +half trick
48	96.5069	6.24853	164.742	82.3708	88.6193
97	84.0276	29.5091	200.482	100.241	129.75

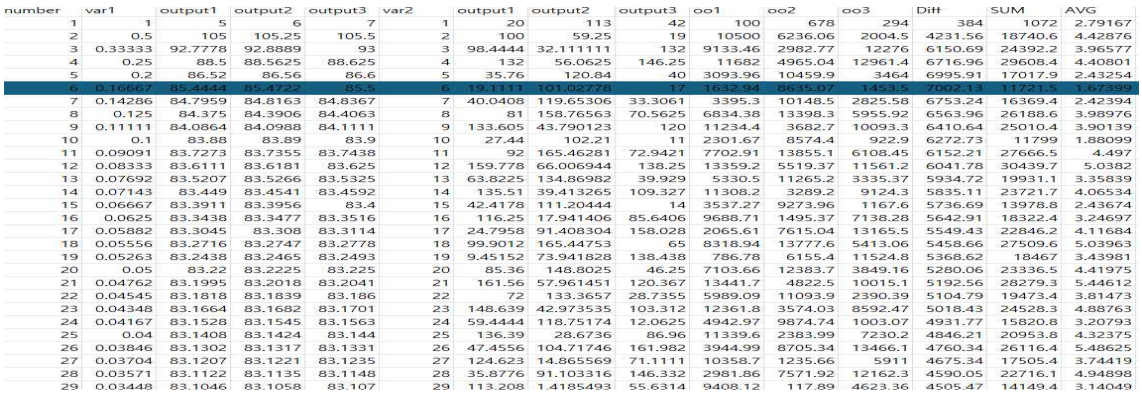
Figure 26. try and result of [2]

output(x) = the result got after doing something on E(x).
Trick = (output(x)-E(x)) mod maximum value in the number space of small data.
half Trick = Trick / 2.
real output(x) = output(x) + half Trick.

with this values, what I have to do is getting the output(x) from the real output(x).
the reason why I do this is to get the information from the number as much as possible.
because there is no information or relation between the E(x) and output(x) just looking at the values.

one of my approach is reversing the calculation to get E(x) reducing the space of number to search.

this could be applied on the small data 48 but not on 97.
therefore, I have to find a generalized equation for T(x).

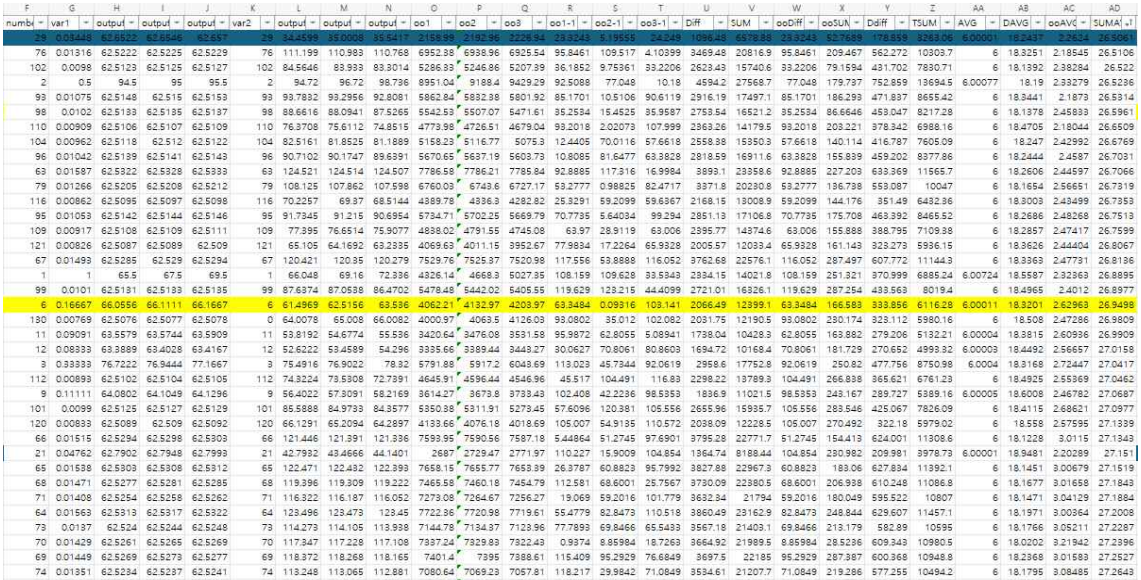


number	var1	output1	output2	output3	var2	output1	output2	output3	oo1	oo2	oo3	Diff	SUM	AVG
1	1	5	6	7	1	20	113	42	100	678	294	384	1072	2.79167
2	0.5	105	105.25	105.5	2	100	59.25	19	10500	6236.06	2004.5	4231.56	18740.6	4.42876
3	0.33333	92.7778	92.8889	93	3	98.4444	32.111111	132	9133.46	2982.77	12276	6150.69	24392.2	3.96577
4	0.25	88.5	88.5625	88.625	4	132	56.0625	146.25	11682	4965.04	12961.4	6716.96	29608.4	4.40801
5	0.2	86.52	86.56	86.6	5	35.76	120.84	40	3093.96	10459.9	3464	6995.91	17017.9	2.43254
6	0.16667	85.4444	85.4722	85.5	6	19.1111	101.02778	17	1632.94	8635.07	1453.5	7002.13	11721.5	1.67399
7	0.14286	84.7959	84.8163	84.8367	7	40.0408	119.65306	33.3061	3395.3	10148.5	2825.58	6753.24	16369.4	2.42394
8	0.125	84.375	84.3906	84.4063	8	81	158.76563	70.5625	6834.38	13398.3	5955.92	6563.96	26188.6	3.98976
9	0.11111	84.0864	84.0988	84.1111	9	133.605	43.790123	120	11234.4	3682.7	10093.3	6410.64	25010.4	3.90139
10	0.1	83.88	83.89	83.9	10	27.44	102.21	11	2301.67	8574.4	922.9	6272.73	11799	1.88099
11	0.09091	83.7273	83.7355	83.7438	11	92	165.46281	72.9421	7702.91	13855.1	6108.45	6152.21	27666.5	4.497
12	0.08333	83.6111	83.6181	83.625	12	159.778	66.006944	138.25	13359.2	5519.37	11561.2	6041.78	30439.7	5.0382
13	0.07692	83.5207	83.5266	83.5325	13	63.8225	134.86982	39.929	5330.5	11265.2	3335.37	5934.72	19931.1	3.35839
14	0.07143	83.449	83.4541	83.4592	14	135.51	39.413265	109.327	11308.2	3289.2	9124.3	5835.11	23721.7	4.06534
15	0.06667	83.3911	83.3956	83.4	15	42.4178	111.20444	14	3537.27	9273.96	1167.6	5736.69	13978.8	2.43674
16	0.0625	83.3438	83.3477	83.3516	16	116.25	17.941406	85.6406	9688.71	1495.37	7138.28	5642.91	18322.4	3.24697
17	0.05882	83.3045	83.308	83.3114	17	24.7958	91.408304	158.028	2065.61	7615.04	13165.5	5549.43	22846.2	4.11684
18	0.05556	83.2716	83.2747	83.2778	18	99.9012	165.44753	65	8318.94	13777.6	5413.06	5458.66	27509.6	5.03963
19	0.05263	83.2438	83.2465	83.2493	19	9.45152	73.941828	138.438	786.78	6155.4	11524.8	5368.62	18467	3.43981
20	0.05	83.22	83.2225	83.225	20	85.36	148.8025	46.25	7103.66	12383.7	3849.16	5280.06	23336.5	4.41975
21	0.04762	83.1995	83.2018	83.2041	21	161.56	57.961451	120.367	13441.7	4822.5	10015.1	5192.56	28279.3	5.44612
22	0.04545	83.1818	83.1839	83.186	22	72	133.3657	28.7355	5989.09	11093.9	2390.39	5104.79	19473.4	3.81473
23	0.04348	83.1664	83.1682	83.1701	23	148.639	42.973535	103.312	12361.8	3574.03	8592.47	5018.43	24528.3	4.88763
24	0.04167	83.1528	83.1545	83.1563	24	59.4444	118.75174	12.0625	4942.97	9874.74	1003.07	4931.77	15820.8	3.20793
25	0.04	83.1408	83.1424	83.144	25	136.39	28.6736	86.96	11339.6	2383.99	7230.2	4846.21	20953.8	4.32375
26	0.03846	83.1302	83.1317	83.1331	26	47.4556	104.71746	161.982	3944.99	8705.34	13466.1	4760.34	26116.4	5.48625
27	0.03704	83.1207	83.1221	83.1235	27	124.623	14.865569	71.1111	10358.7	1235.66	5911	4675.34	17505.4	3.74419
28	0.03571	83.1122	83.1135	83.1148	28	35.8776	91.103316	146.332	2981.86	7571.92	12162.3	4590.05	22716.1	4.94898
29	0.03448	83.1046	83.1058	83.107	29	113.208	1.4185493	55.6314	9408.12	117.89	4623.36	4505.47	14149.4	3.14049

Figure 27. try and result of [2]

although I didn't capture and paste all detail information related with Fig.27 here, but in the number space of 89, number 6 has the minimum AVG in all number from 1 to 88.

[3] 2025-10-27. try and result.
today from the number 130 I could get 29. it was really hard.
I have made a excel file for calculation to get the generalized equation for T(x).



numb	var1	output	output	output	var2	output	output	output	oo1	oo2	oo3	oo1-1	oo2-1	oo3-1	Diff	SUM	ooDiff	ooSUM	Ddiff	TSUM	AVG	DAVG	ooAVG	SUMA
29	0.03448	62.6522	62.6546	62.657	29	34.4599	35.0008	35.5417	2158.99	2192.96	2226.94	233.243	5.19559	24.249	1096.48	6578.88	233.243	52.7689	178.559	3263.06	6.00001	18.2437	2.2624	26.5061
76	0.01316	62.5222	62.5225	62.5229	76	111.199	110.983	110.768	6952.88	6938.96	6925.54	95.8461	109.517	4.10399	3469.48	20816.9	95.8461	209.467	562.272	10303.7	6	18.3251	2.18545	26.5106
102	0.0098	62.5123	62.5125	62.5127	102	84.5646	83.933	83.3014	5286.33	5246.86	5207.39	36.1852	9.75361	33.2206	2623.43	15740.6	33.2206	79.1594	431.702	7830.71	6	18.1392	2.38284	26.522
2	0.5	94.5	95	95.5	2	94.72	96.72	98.736	8951.04	9188.4	9429.29	92.5088	77.048	10.18	4594.2	27568.7	77.048	179.737	752.859	13694.5	6.00077	18.19	2.33279	26.5236
93	0.01075	62.5148	62.515	62.5153	93	93.7832	93.2956	92.8081	5862.84	5832.38	5801.92	85.1701	105.106	90.6119	2916.19	17497.1	85.1701	196.293	471.837	8655.42	6	18.3441	2.1873	26.5314
98	0.0102	62.5133	62.5135	62.5137	98	88.6616	88.0941	87.5265	5542.59	5507.07	5471.61	35.2534	15.4525	35.9587	2753.54	16521.2	35.2534	86.6646	453.047	8217.28	6	18.1378	2.45833	26.5961
110	0.00909	62.5106	62.5107	62.5109	110	76.3708	75.6112	74.8515	4773.98	4726.51	4679.04	93.2018	2.02073	107.999	2363.26	14179.5	93.2018	203.221	378.342	6988.16	6	18.4705	2.18044	26.6509
104	0.00962	62.5118	62.512	62.5122	104	82.5161	81.8525	81.1889	5158.23	5116.77	5075.3	12.4405	70.0116	57.6618	2558.38	15350.3	57.6618	140.114	416.787	7605.09	6	18.247	2.42992	26.6769
96	0.01042	62.5139	62.5141	62.5143	96	90.7102	90.1747	89.6391	5670.65	5637.19	5603.73	10.8085	81.6477	63.3828	2818.59	16911.6	63.3828	155.899	459.202	8377.86	6	18.2444	2.4587	26.7031
63	0.01587	62.5322	62.5328	62.5333	63	124.521	124.514	124.507	7786.58	7786.21	7785.84	92.8885	117.316	16.9984	3893.1	23358.6	92.8885	227.203	683.369	11565.7	6	18.2606	2.44597	26.7066
79	0.01266	62.5205	62.5208	62.5212	79	108.125	107.862	107.598	6760.03	6743.6	6727.17	53.2777	0.98825	82.4717	3371.8	20230.8	53.2777	136.738	553.087	10047	6	18.1654	2.56651	26.7319
116	0.00862	62.5095	62.5097	62.5098	116	70.2257	69.37	68.5144	4389.78	4336.3	4282.82	25.3291	59.2099	59.6367	2168.15	13008.9	59.2099	144.176	351.49	6432.36	6	18.3003	2.48499	26.7353
95	0.01053	62.5142	62.5144	62.5146	95	91.7345	91.215	90.6954	5734.71	5702.25	5669.79	70.7735	5.64034	99.294	2851.13	17106.8	70.7735	175.708	463.392	8465.52	6	18.2686	2.48268	26.7513
109	0.00917	62.5108	62.5109	62.5111	109	77.395	76.6514	75.9077	4838.02	4791.55	4745.08	63.97	28.9119	63.006	2395.77	14374.6	63.006	155.888	388.795	7109.38	6	18.2857	2.47417	26.7599
121	0.00826	62.5087	62.5089	62.509	121	65.105	64.1692	63.2335	4069.63	4011.15	3952.67	77.9834	17.2264	65.9328	2005.57	12033.4	65.9328	161.143	323.273	5936.15	6	18.3626	2.44404	26.8067
67	0.01493	62.5285	62.529	62.5294	67	120.421	120.35	120.279	7529.76	7525.37	7520.98	117.556	53.8888	116.052	3762.68	22576.1	116.052	287.497	607.772	11144.3	6	18.3363	2.47731	26.8186
1	1	65.5	67.5	69.5	1	66.048	69.16	72.336	4326.14	4668.3	5027.35	108.159	109.628	83.5343	2384.15	14021.8	108.159	251.321	370.999	6885.24	6.00724	18.5587	2.32363	26.8895
99	0.0101	62.5181	62.5183	62.5185	99	87.6374	87.0538	86.4702	5478.48	5442.02	5405.55	119.629	123.215	44.4099	2721.01	16326.1	119.629	287.254	433.563	8019.4	6	18.4965	2.4012	26.8977
6	0.16667	66.0556	66.1111	66.1667	6	61.4969	62.5156	63.536	4062.21	4132.97	4203.97	63.3484	0.09316	103.141	2066.49	12399.1	63.3484	166.583	333.856	6116.28	6.00011	18.3201	2.62963	26.9498
130	0.00769	62.5076	62.5077	62.5078	0	64.0078	65.008	66.0082	4000.97	4069.5	4126.03	93.0802	35.012	102.082	2031.75	12190.5	93.0802	230.174	323.112	5980.16	6	18.508	2.47286	26.9809
11	0.09091	63.5579	63.5744	63.5909	11	53.8192	54.6774	55.536	3420.64	3476.08	3531.58	95.9872	62.8055	5.08941	1738.04	10428.3	62.8055	163.882	279.206	5132.21	6.00004	18.3815	2.60936	26.9909
12	0.08333	63.3889	63.4028	63.4167	12	52.6222	53.4589	54.296	3335.66	3389.44	3443.27	30.0627	70.8061	80.8603	1694.72	10168.4	70.8061	181.729	270.652	4993.32	6.00003	18.4492	2.56657	27.0158
3	0.33333	76.7222	76.9444	77.1667	3	75.4916	76.9022	78.32	5791.88	5917.2	6048.69	113.023	45.7344	92.0619	2958.6	17752.8	92.0619	250.82	477.756	8750.98	6.0004	18.3168	2.72447	27.0417
112	0.00893	62.5102	62.5104	62.5105	112	74.3224	73.5308	72.7391	4645.91	4596.44	4546.96	45.517	104.491	116.83	2298.22	13789.3	104.491	266.838	365.621	6761.23	6	18.4925	2.55369	27.0462
9	0.11111	64.0802	64.1049	64.1296	9	56.4022	57.3091	58.2159	3614.27	3673.8	3733.43	102.408	42.2236	98.5353	1836.9	11021.5	98.5353	248.167	289.727	5389.16	6.00005	18.6008	2.46782	27.0687
101	0.0099	62.5125	62.5127	62.5129	101	85.5888	84.9733	84.3577	5350.88	5311.91	5273.45	57.6096	120.381	105.556	2655.96	15935.7	105.556	283.546	425.067	7826.09	6	18.4115	2.68621	27.0977
120	0.00833	62.5089	62.509	62.5092	120	66.1291	65.2094	64.2897	4133.66	4076.18	4018.69	105.007	54.9135	110.572	2038.09	12228.5	105.007	270.492	322.18	5979.02	6	18.558	2.57595	27.1339
66	0.01515	62.5294	62.5298	62.5303	66	121.446	121.391	121.336	7593.95	7590.56	7587.18	5.44864	51.2745	97.6901	3795.28	22771.7	51.2745	154.413	624.001	11308.6	6	18.1228	3.0115	27.1343
21	0.04762	62.7902	62.7948	62.7993	21	42.7932	43.4666	44.1401	2687	2729.47	2771.97	110.227	15.9009	104.854	1364.74	8188.44	104.854	230.982	209.981	3978.79	6.00001	18.9481	2.20289	27.151
65	0.01538	62.5303	62.5308	62.5312	65	122.471	122.432	122.393	7658.15	7655.77	7653.39	26.3787	60.8823	95.7992	3827.88	22967.3	60.8823	183.06	627.834	11392.1	6	18.1451	3.00679	27.1519
68	0.01471	62.5277	62.5281	62.5285	68	119.396	119.309	119.222	7465.58	7460.18	7454.79	112.581	68.6001	25.7567	3730.09	22380.5	68.6001	206.938	610.248	11086.8	6	18.1677	3.01659	27.1843
71	0.01408	62.5254	62.5258	62.5262	71	116.322	116.187	116.052	7273.08	7264.67	7256.27	19.069	59.2016	101.779	3632.84	21794	59.2016	180.049	595.522	10807	6	18.1471	3.04129	27.1884
64	0.01563	62.5313	62.5317	62.5322	64	128.496	128.473	128.45	7722.36	7720.98	7719.61	55.4779	82.8473	110.518	3860.49	23162.9	82.8473	248.844	629.607	11457.1	6	18.1971	3.00364	27.2008
73	0.0137	62.524	62.5244	62.5248	73	114.273	114.105	113.938	7144.78	7134.37	7123.96	77.7893	69.8466	65.5483	3567.18	21403.1	69.8466	213.179	582.89	10595	6	18.1766	3.05211	27.2287
70	0.01429	62.5261	62.5265	62.5269	70	117.347	117.228	117.108	7397.24	7329.83	7322.43	0.9874	8.85984	18.7263	3664.92	21989.5	8.85984	28.5236	609.343	10980.5	6	18.0202	3.21942	27.2396
69	0.01449	62.5269	62.5273	62.5277	69	118.372	118.268	118.165	7401.4	7395	7388.61	115.409	95.2929	76.6849	3697.5	22185	95.2929	287.387	600.368	10948.8	6	18.2368	3.01583	27.2527
74	0.01351	62.5234	62.5237	62.5241	74	113.248	113.065	112.881	7080.64	7069.23	7057.81	118.217	29.9842	71.0849	3534.61	21207.7	71.0849	219.286	577.255	10494.2	6	18.1795	3.08485	27.2643
115	0.0087	62.5097	62.5098	62.51	115	71.2499	70.4102	69.5705	4453.81	4401.33	4348.85	83.497	23.702	52.0617	2200.67	13204	52.0617	159.261	358.101	6522.37	6	18.2138	3.05908	27.2729
16	0.0625	63	63.0078	63.0156	16	48.128	48.8881	49.6485	3032.06	3080.33	3128.63	52.1762	91.7445	81.849	1540.17	9241.03	81.849	225.77	243.053	4507.63	6.00002	18.5459	2.75837	27.3043
87	0.01149	62.5169	62.5172	62.5174	87	99.9293	99.5379	99.1464	6247.27	6222.83	6198.38	35.646	31.7968	47.171	3111.41	18668.5	35.646	114.614	512.628	9276.93	6	18.0968	3.21533	27.3122
8	0.125	64.5	64.5313	64.5625	8	57.856	58.7925	59.73	3731.71	3793.95	3856.32	26.9295	56.0121	87.8816	1896.98	11382	56.0121	170.823	306.827	5605.58	6.00006	18.2695	3.04976	27.3193
81	0.01235	62.5195	62.5198	62.5201	81	106.076	105.781	105.485	6631.82	6613.38	6594.95	101.582	67.7561	44.9298	3306.69	19840.2	67.7561	214.268	539.823	9812.94	6	18.1781	3.16234	27.3404
117	0.00855	62.5094	62.5095	62.5096	117	69.2016	68.3299	67.4582	4325.75	4271.27	4216.79	98.4024	105.091	81.7819	2135.63	12813.8	98.4024	285.276	339.538	6264.26	6	18.4493	2.89907	27.3484
78	0.01282	62.521	62.5214	62.5217	78	109.15	108.902	108.655	6824.14	6808.72	6793.29	102.079	109.288	124.252	3404.36	20426.1	109.288	335.619	549.178	10045.3	6	18.2914	3.07097	27.3624
7	0.14286	65.1122	65.1531	65.1939	7	59.5069	60.4792	61.4527	3874.63	3940.4	4006.84	67.3922	62.4146	75.6758	1970.2	11821.4	67.3922	205.483	317.135	5807.95	6.00008	18.3138	3.04905	27.3629
72	0.01389	62.5247	62.5251	62.5255	72	115.297	115.146	114.995	7208.93	7199.52	7190.11	44.7141	121.226	75.7472	3599.76	21598.6	75.7472	241.687	587.335	10678.4	6	18.1812	3.19071	27.3719
75	0.01333	62.5228	62.5231	62.5235	75	112.223	112.024	111.825	7016.51	7004.09	6991.67	40.9252	1.56411	92.2956	3502.05	21012.3	40.9252	134.785	576.853	10438.7	6	18.096	3.29345	27.3895
83	0.01205	62.5186	62.5189	62.5192	83	104.027	103.7	103.372	6503.62	6483.18	6462.75	52.4567	53.6982	68.4234	3241.59	19449.6	53.6982	174.578	531.316	9637.49	6	18.1389	3.2511	27.39

Figure 28. try and result of [3]

the important thing is just changing one variable in the excel file I could get both 6 and 29 without changing any variable except for the number data.

test data
128,130,132,2
88,89,90,15

T(x) of the module [*1].

from Fig 24, 25, 27, 28, it is just two small data and there is other problems to solve but before going to the problems I have to do this first.

first, let's summarize how this algorithm works.

suppose that two small data to be encrypted, 48 and 97, they are encrypted to 96.5069, 84.0276 by R(x) of the module [*2] also they are changed as 88.6193, 129.75 by T(x) of the module [*2] then this algorithm will output the values.
after that, this is under assumption, T(x) of the module [*1] do something to detect attack and determines this values is encrypted by this T(x) of the module [*1] and then notifies that this values not attack with values 88.6193, 129.75.

after that the module [*2] runs the logic and decrypt the values 88.6193, 129.75 and get values 48, 97.

more specifically, let's see how the module [*2] decrypt the value 88.6193.

from [2], equation for decryption are same like this.

output(88.6193) = temp output(x) + half trick, in this example the temp output(x) will be 6.24583 by T(x) of the module [*2] from it half trick will be calculated as 82.3708. then trick will be 164.742.

Trick(164.742) = output(x) - E(x), it is same with E(x) = output(x) - Trick(164.742)
here, output(x) will be 6.24583 so -158.49617 will be came out.
because the value should be in the number space of 255, -158.49617 mod 255 = 96.50383.
finally, the value 96.50383 will be 48 by the T(x) of the module [*2].

this is assumption.
because my test data is all integer value.

next thing I have to do is this.

Detect, this means to detect attack from the data and determine the result by the logic.

[!] determinism.
[4] determinism for this algorithm should meet these conditions.
generality
first, this means this algorithm should have ability to encrypt many data by the module [*2] and also decrypt them by the module [*2] even though at least one attack exists in the middle time.
second, this means the module [*1] should have ability to determine that attack exist on the data even though the module [*2] encrypts many data and at least one attack exists in the middle time.

under this condition, traditional substitution can not be adopted.

before going to deeper, determinism for this algorithm should be defined.
the meaning it has should satisfy these thing to achieve the purpose of this algorithm.

the meaning data have in this algorithm doesn't have nothing.
from it, although data encrypted by the module [*2] could be changed on same data by the logic and changed structure by it, data decrypted by the module [*2] should be determined as original data.
also, output by the module [*1] should be determined.

this is very difficult and I'm not sure that this could be done.
but I have to find at least one solution to solve this problem for this project.

$A = A_1, A_2, A_3, \dots, A_N$ | A = Attack aware Function, N = Attack Count.
$E = E_1, E_2, E_3, \dots, E_N$ | E = Encrypt Function, N = Attack Count.
$E(x) = E_1(x), E_2(x), E_3(x), \dots, E_K(x)$ | x = Input Data, K = cycle number.

$\overrightarrow{A} E_N(x_K)$ | A = Attack.
$D = D_1, D_2, D_3, \dots, D_N$ | D = Decrypt Function, N = Attack Count.
$D(x) = D_1(x), D_2(x), D_3(x), \dots, D_K(x)$ | x = Input Data, K = cycle number.
$T(x)_N = (O_1(x) - E(x)) \bmod (M)$ | T = Trick Function, $O_1(x)$ = output(x), M = Maximum in the number space of N.
$O_2(x) = O_1(x) + T(x)/2$ | $O_2(x)$ = real output(x).

$\overrightarrow{A} O_2(x_K)$ | A = Attack.
$O_2'(x_K)$ | O_2' = reverse calculation of O_2.
$S = s \epsilon ACK, ENC, DEC$ | s = Status, ACK = Attack, ENC = Encrypt, DEC = Decrypt.

$x = x_1, x_2, x_3, \ldots, x_K$ | x = Input Data, K = cycle number.

$\check{x}(S) = \check{x}_1(s), \check{x}_2(s), \check{x}_3(s), \ldots, \check{x}_K(s)$ | x = Input Data has Status s, K = cycle number.

$\hat{x}(S) = \hat{x}_1(s), \hat{x}_2(s), \hat{x}_3(s), \ldots, \hat{x}_K(s)$ | x = Input Data has Status s determined by the module [*1], K = cycle number.

under this definition and from the conditions for this algorithm, below things has made.

$(1)\, E_N(x_K) = E_N(x_K) \neq E_N(x_K)$

$(1-1)\, E_N(x_P) = E_N(x_Q) \neq E_N(x_P)$

$(1-2)\, E_P(x_K) = E_Q(x_K) \neq E_P(x_K)$

$(2)\, T(E_N(x_K))_Z = T(E_N(x_K))_Z \neq T(E_N(x_K))_Z$

$(3)\, O_1(x_K) = O_1(x_K) \neq O_1(x_K)$

$(4)\, O_2(x_K) = O_2(x_K) \neq O_2(x_K)$

$(4-1)\, O_2'(E_N(x_K)) = E_N(x_K)$

$(5)\, D_N(E_N(x_K)) = D_N(E_N(x_K)) = x_K$

$(5-1)\, D_P(E_P(x_K)) = D_Q(E_Q(x_K)) = x_K$

$(5-2)\, D_P(E_Q(x_K)) = x_K$

$(6)\, A_N(\check{x}_K(ACK)) = \hat{x}_K(ACK)$

$(7)\, A_N(\check{x}_K(ENC)) = \hat{x}_K(ENC)$

$(8)\, A_N(\check{x}_K(DEC)) = \hat{x}_K(DEC)$

$(9) A_N(\check{x}_K(s)) = A_{N-1}(\check{x}_K(s)) = A_{N-2}(\check{x}_K(s)) = \ldots = A_1(\check{x}_K(s)) = \hat{x}_K(s)$

below mixed process is ENC on Input x_K and DEC on from ENC output.

x_K
$\downarrow \rightarrow \check{x}_K(DEC)$
$A_N(x_K)$
$\downarrow \rightarrow \hat{x}_K(DEC)$
$E_N(x_K)$
$\downarrow$
$O_1(E_N(x_K)),\ T(E_N(x_K))_Z$
$\downarrow$
$O_2(E_N(x_K))$
$\downarrow \rightarrow x_K = O_2(E_N(x_K)), \check{x}_K(ENC)$
$A_N(x_K)$
$\downarrow \rightarrow \hat{x}_K(ENC)$
$O_2'(x_K), D_N(E_N(x_K))$
$\downarrow$
x_K

[!] determination of $A_N(x_K)$ for $\hat{x}_K(s)$ from $\check{x}_K(s)$.

[4-1] case $x_K(ACK)$, $x_K(ENC)$

because there is no information used to compare with x_K the difficulty exists.

one of solutions for this case are using two data, $(O_2(E_N(x_K)), E_N(x_K))$.

although this solution doesn't fit with this process I have to write this to memorize it.

this is based on [3] under assumption.

there I could get two values similar with $O_1(E_N(x_K))$ from $O_2(E_N(x_K))$.

if $A_N(x_K)$ has ability to get $O_1(E_N(x_K))$ from $O_2(E_N(x_K))$ then it can calculate $E_N(x_K)$ from $O_2(E_N(x_K))$ and compare with $E_N(x_K)$.

from it, $A_N(x_K)$ can determine s of $E_N(x_K)$ as one of ACK and ENC.

the important thing is unique relation between $E_N(x_K)$ and $O_1(E_N(x_K))$.

but remind that this is not fit with this process.

[!] determination of $O_1(x_K)$ for $T(E_N(x_K))_Z$.

[4-2] if $O_1(x_K)$ isn't always same on x_K, $T(E_N(x_K))_Z$ is also change although $E_N(x_K)$ is same on x_K.

one of solutions is using unit information when calculate $O_1(x_K)$.

but I don't know how to do that right now.

$\int_{i=1}^{n} I(i)$ | $\int$ = temporary notation, I = Unit's information,

[!] determination of $D_N(E_N(x_K))$ for x_K from $E_N(x_K)$.

[4-3] if I could find a solution for [4-2], especially I could define $\int_{i=1}^{n} I(i)$, I could solve this problem.

the reason why $D_N(E_N(x_K)) \rightarrow x_K$ is diffcult now is that there is no unique relation between $E_N(x_K)$ and $D_N(E_N(x_K)) = x_K$ and also $E_N(x_K)$ doesn't always same on x_K.

if I can define $O_1(x_K)$ specifically then I could solve this.

in below process, a Input x_K is a encrypted data and it has modified by attack.

$\xrightarrow[A]{} O_2(E_N(x_K))$

$\downarrow \rightarrow x_K = \xrightarrow[A]{} O_2(E_N(x_K)), \check{x}_K(ACK)$

$A_N(x_K)$

$\downarrow \rightarrow \hat{x}_K(ACK)$

$D_{N+1}(x_K)$ or $E_{N+1}(x_K)$

$\downarrow$

$D_{N+1}(x_K)$

below process explains that after a Input x_K encrypts, someone wants to decrypt it. but in the middle time there was at least one attack.

$O_2(E_N(x_K))$
$\downarrow \rightarrow x_K = O_2(E_N(x_K)), \check{x}_K(ENC)$
$A_N(x_K)$
$\downarrow \rightarrow \hat{x}_K(ENC)$
$O_2^{'}(x_K), D_{N+T}(E_N(x_K))$
$\downarrow$
x_K

[4-4] determination of $D_{N+T}(x_K)$ for x_K from $E_N(x_K)$.

this process explains that after one attack has detected how this works.

x_K
$\downarrow \rightarrow \check{x}_K(DEC)$
$A_N(x_K)$
$\downarrow \rightarrow \hat{x}_K(DEC)$
$E_{N+1}(x_K)$
$\downarrow$
$O_1(E_{N+1}(x_K)),\ T(E_{N+1}(x_K))_Z$
$\downarrow$
$O_2(E_{N+1}(x_K))$
$\downarrow \rightarrow x_K = O_2(E_{N+1}(x_K)), \check{x}_K(ENC)$
$A_N(x_K)$
$\downarrow \rightarrow \hat{x}_K(ENC)$
$O_2^{'}(x_K), D_{N+1}(E_{N+1}(x_K))$
$\downarrow$
x_K

[!] what is attack to break this algorithm
[5] in this algorithm attack means to modify the data encrypted by the module [*2]. but because under this algorithm the meaning of algorithm doesn't exists on the data, although data is encrypted by the module [*2] it is just one of data.
so [4] should be solved before defining the attack specifically.

[!] from the definition of attack, how to detect it on the data without any information.
[6] this problem should be considered with [4], [5] together.

[!] how to process prior data encrypted by the module [*2] if attack occurs after the process of encryption for the data.
[7] one important thing that was out of consideration and I should think about is this. this problem should be considered with [4] together.

some solutions I have for [7] are below.
discard the data, this means to discard the prior data.
but this solution should not be adopted if the data is very important and must be used.
temporarily reformatting internals of the module [*2], this means to reconstruct internals of the module [*2] based on the data.
I think this is one of solutions to be considered but to adopt this the solution [4] should be solved and I have to consider that this could be done.

R(x) of the module [*1].