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.

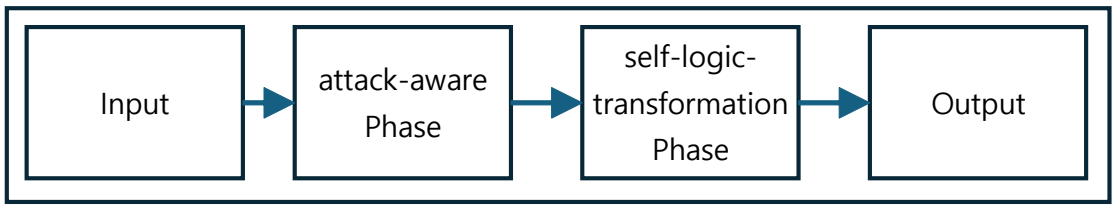

Figure 1. overview of algorithm.

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.

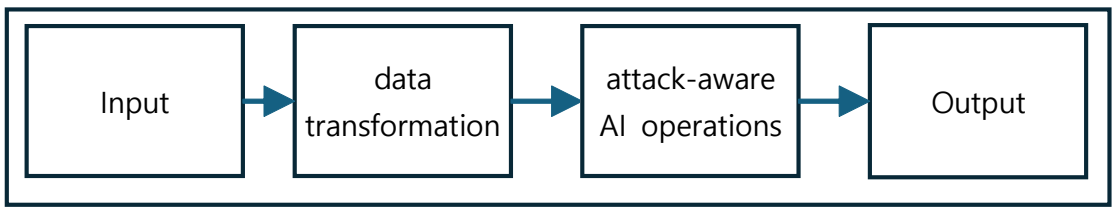

Figure 2. top view of attack-aware phase.

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.

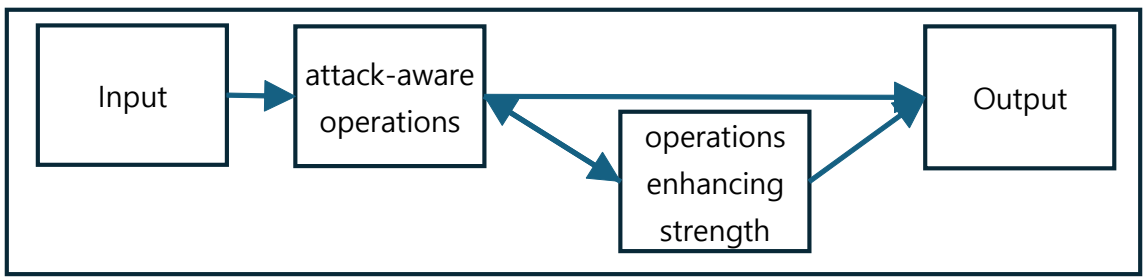

Figure 3. middle-level view of attack-aware phase.

attack-aware operations will be consisted of some layers.

self-logic-transformation phase will have below structure.

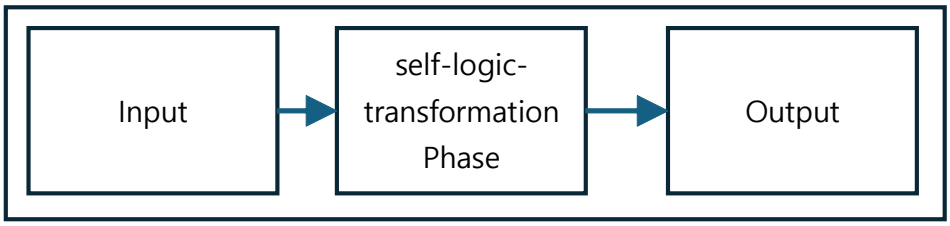

Figure 4. top view of self-logic-transformation phase.

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.

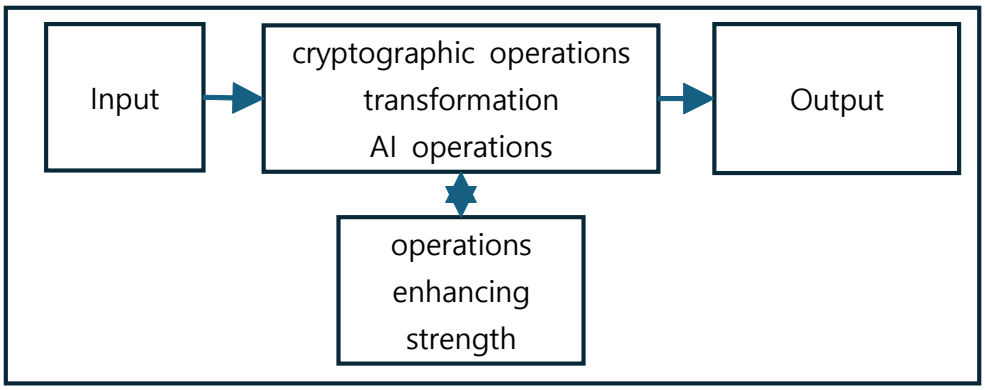

Figure 5. middle-level view of self-logic-transformation phase.

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.

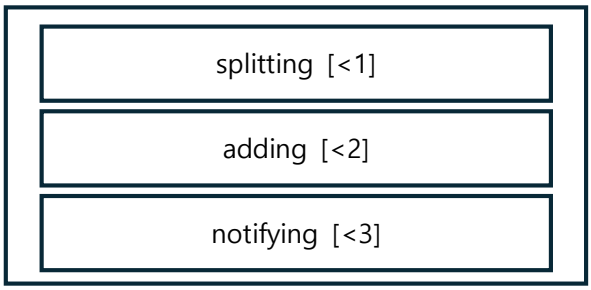

Figure 6. the data transformation module.

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.

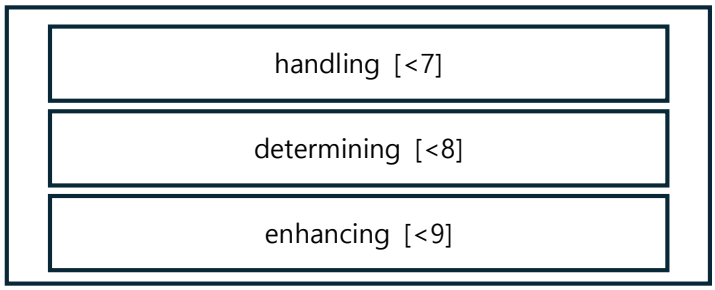

Figure 7. the attack-aware operations module.

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.

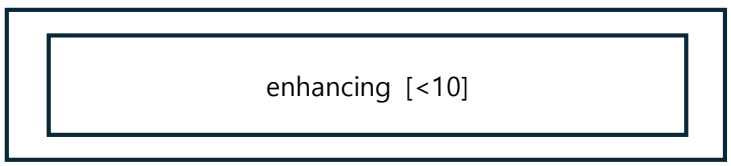

Figure 8. the operations enhancing strength module

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.

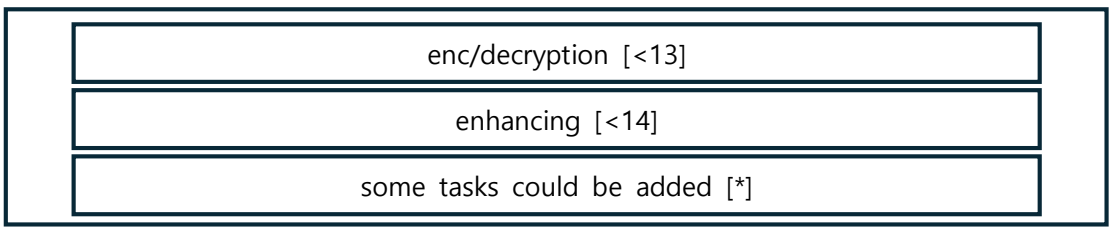

Figure 9. the cryptographic operations transformation AI operations module.

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.

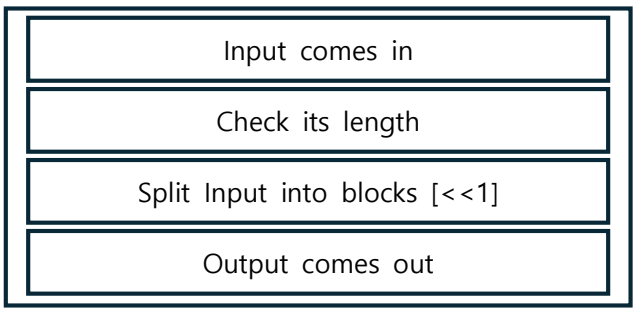

Figure 11. logical procedure of the task [<1]

[2] adding.

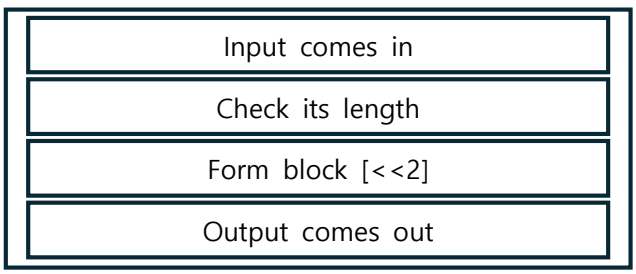

Figure 12. logical procedure of the task [<2]

[3] notifying.

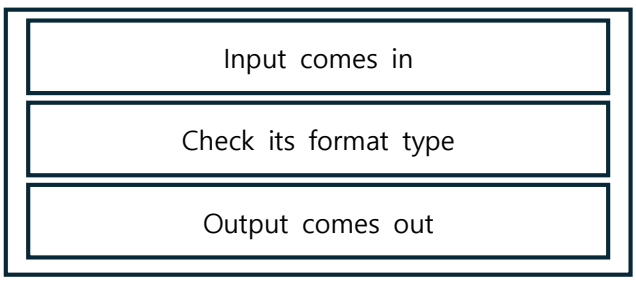

Figure 13. logical procedure of the task [<3]

[7] handling.

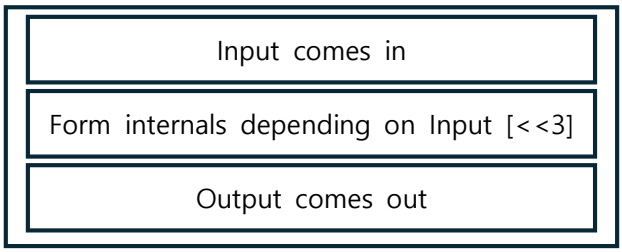

Figure 14. logical procedure of the task [<7]

[8] determining.

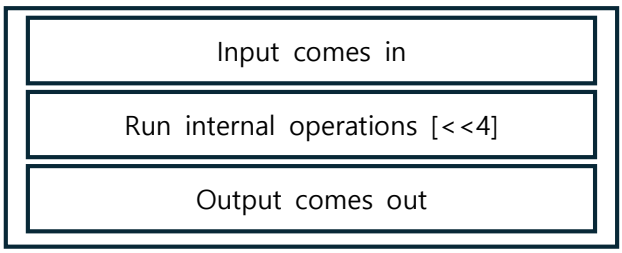

Figure 15. logical procedure of the task [<8]

[9] enhancing.

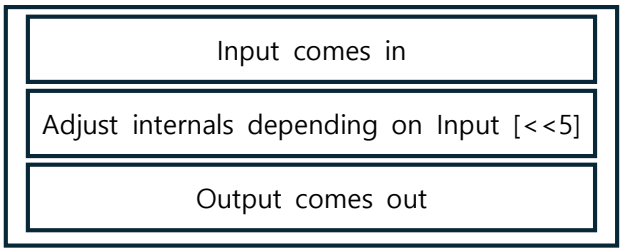

Figure 16. logical procedure of the task [<9]

[10] enhancing.

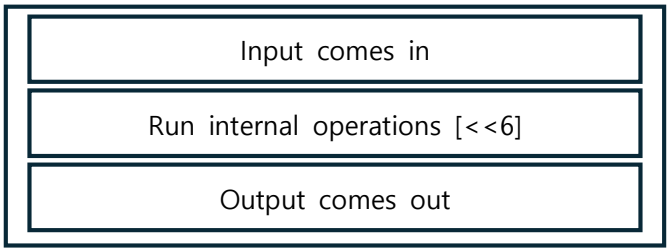

Figure 17. logical procedure of the task [<10]

[13] enc/decryption.

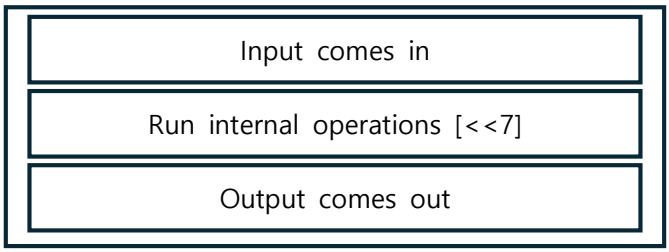

Figure 18. logical procedure of the task [<13]

[14] enhancing.

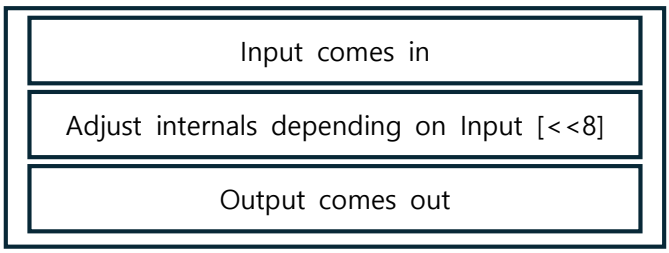

Figure 19. logical procedure of the task [<14]

[15] enhancing.

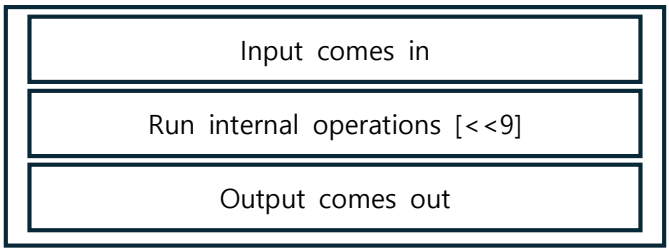

Figure 20. logical procedure of the task [<15]

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].

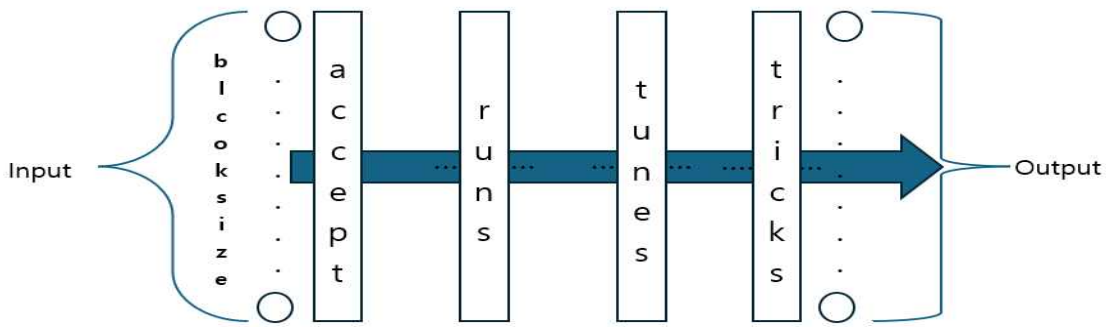

Figure 21. internals to be formed by the logic [3].

[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].

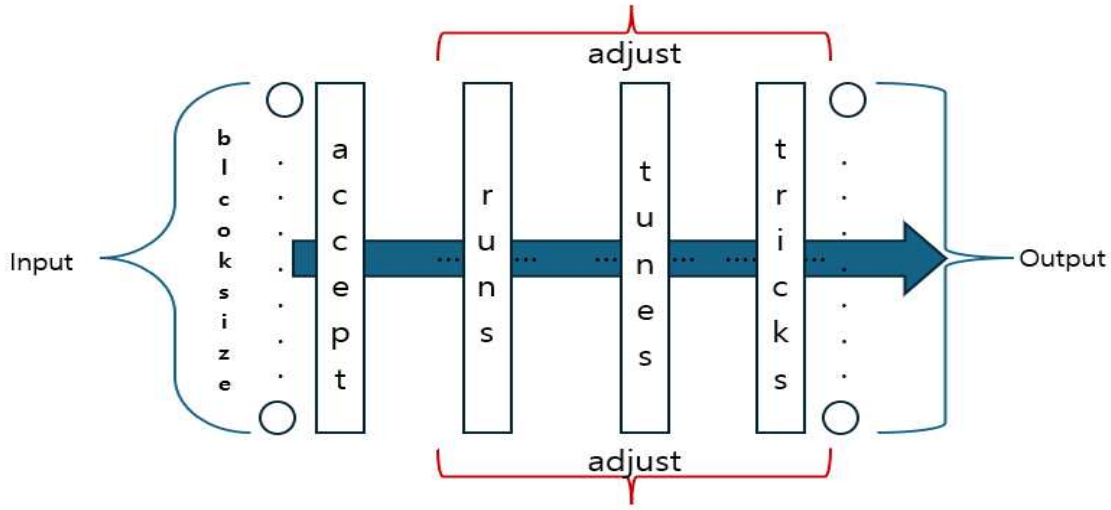

Figure 22. the location to be adjusted by the logic [5].

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].

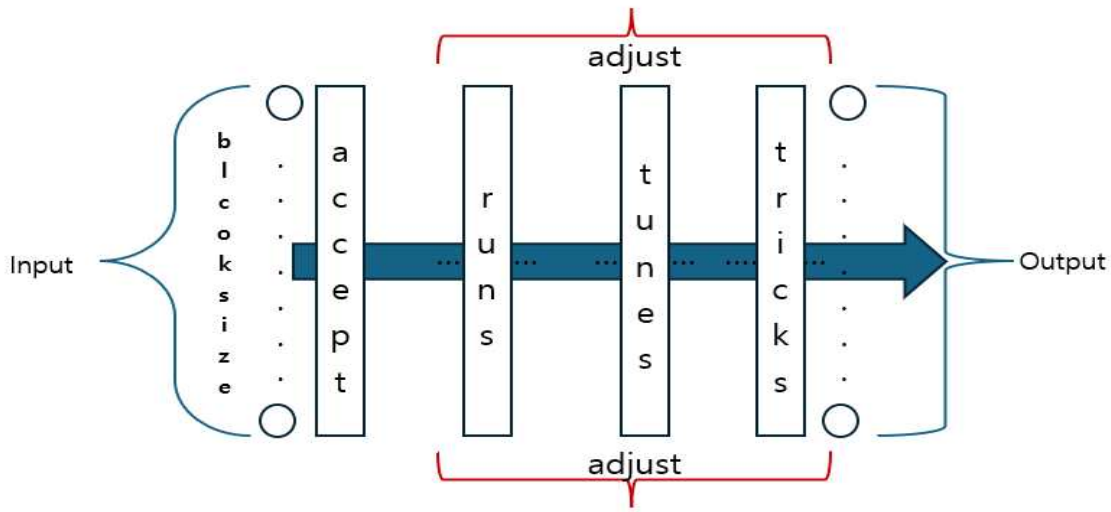

Figure 23. the location to be adjusted by the logic [8].

[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.