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.

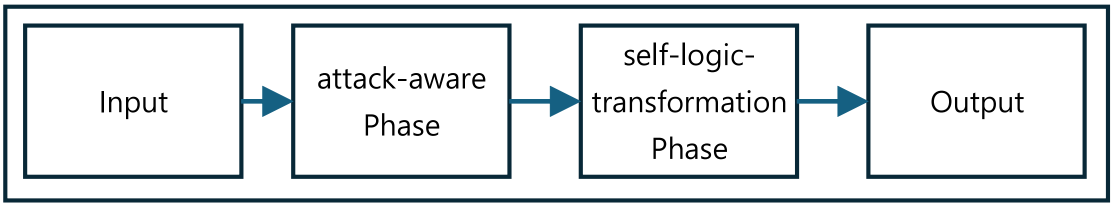

Figure 1. overview of algorithm.

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.

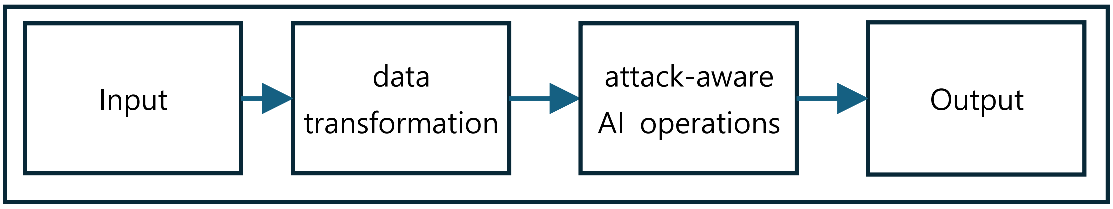

Figure 2. top view of attack-aware phase.

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.

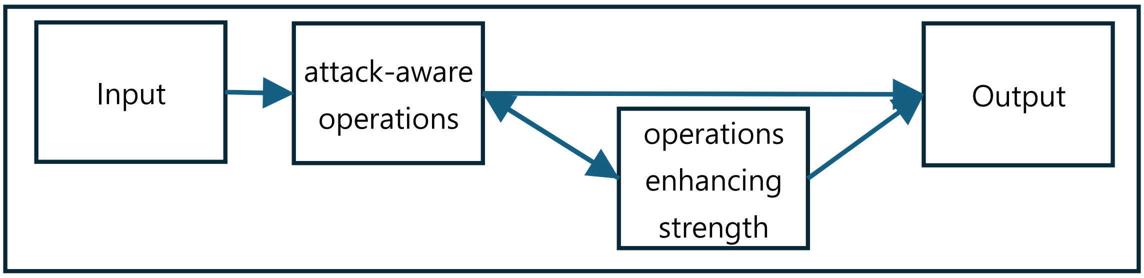

Figure 3. middle-level view of attack-aware phase.

attack-aware operations will be consisted of some layers.

self-logic-transformation phase will have below structure.

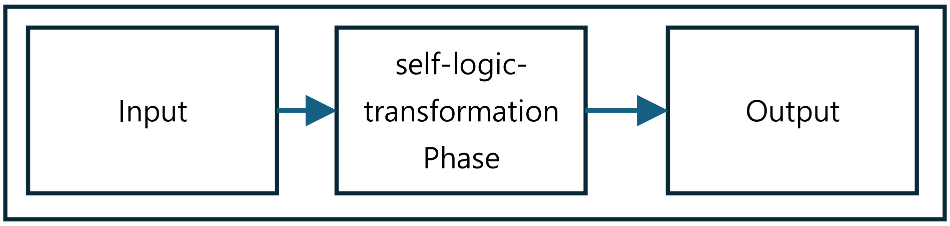

Figure 4. top view of self-logic-transformation phase.

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.

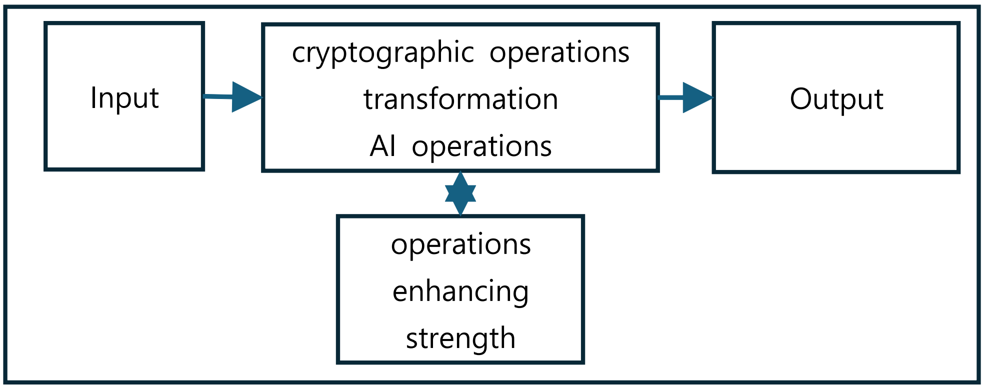
Input
cryptographic operations transformation AI operations
Output
operations enhancing strength

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.

operations enhancing strength

[!] how to format the data results from this operations.

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.

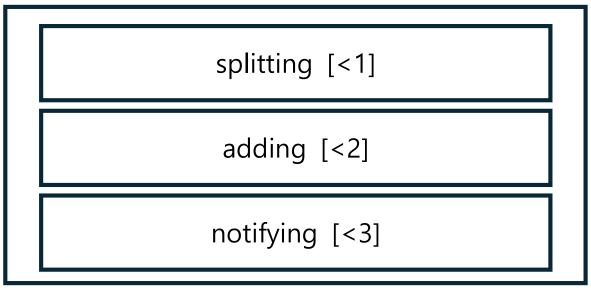

Figure 6. the data transformation module.

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.

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.

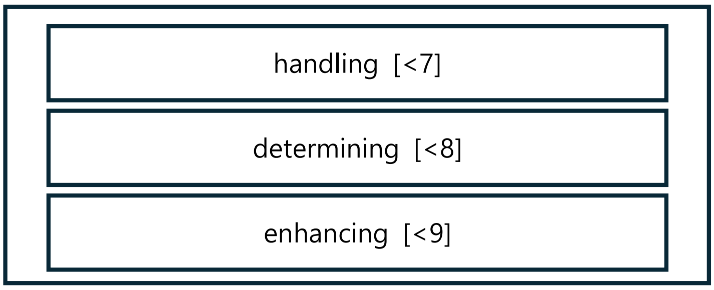

Figure 7. the attack-aware operations module.

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.

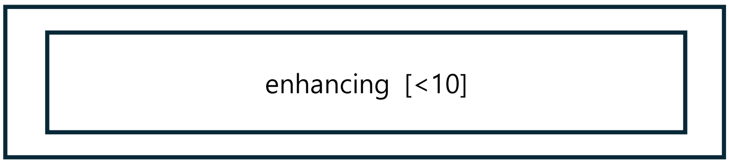

Figure 8. the operations enhancing strength module