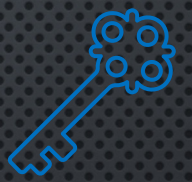
FOR YOUR KEYS ONLY



Presented by

Sai mitra jandhyala

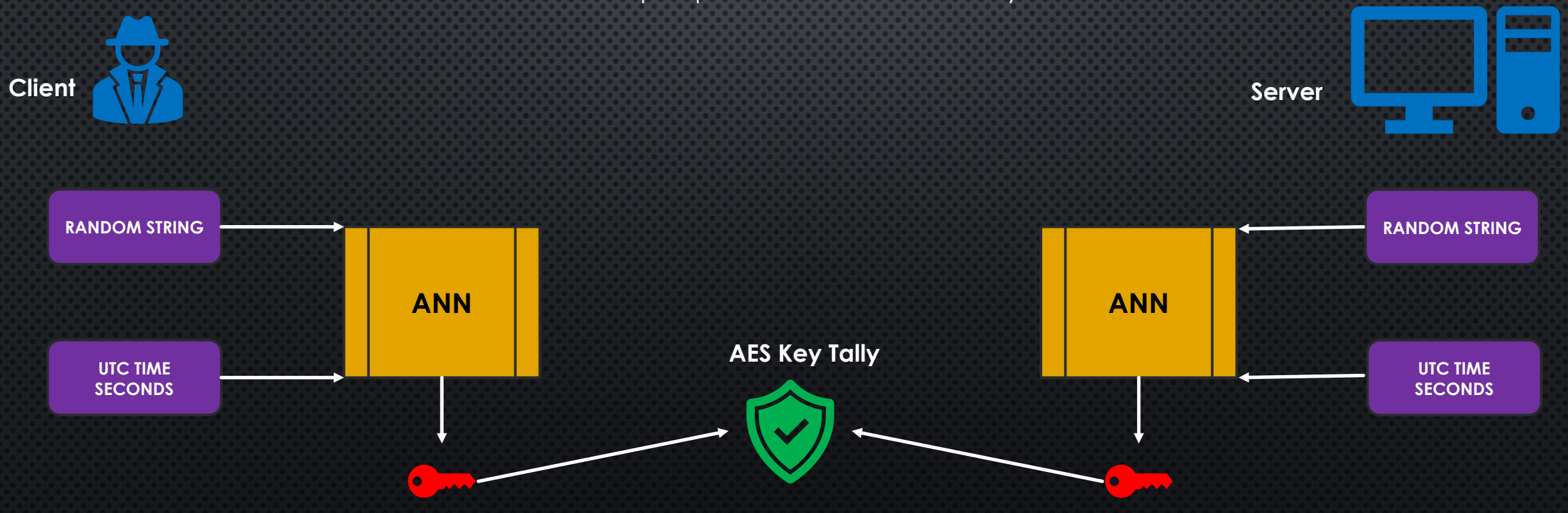
Student ID: 1061239

MSc in artificial intelligence and machine learning

Liverpool john moores university

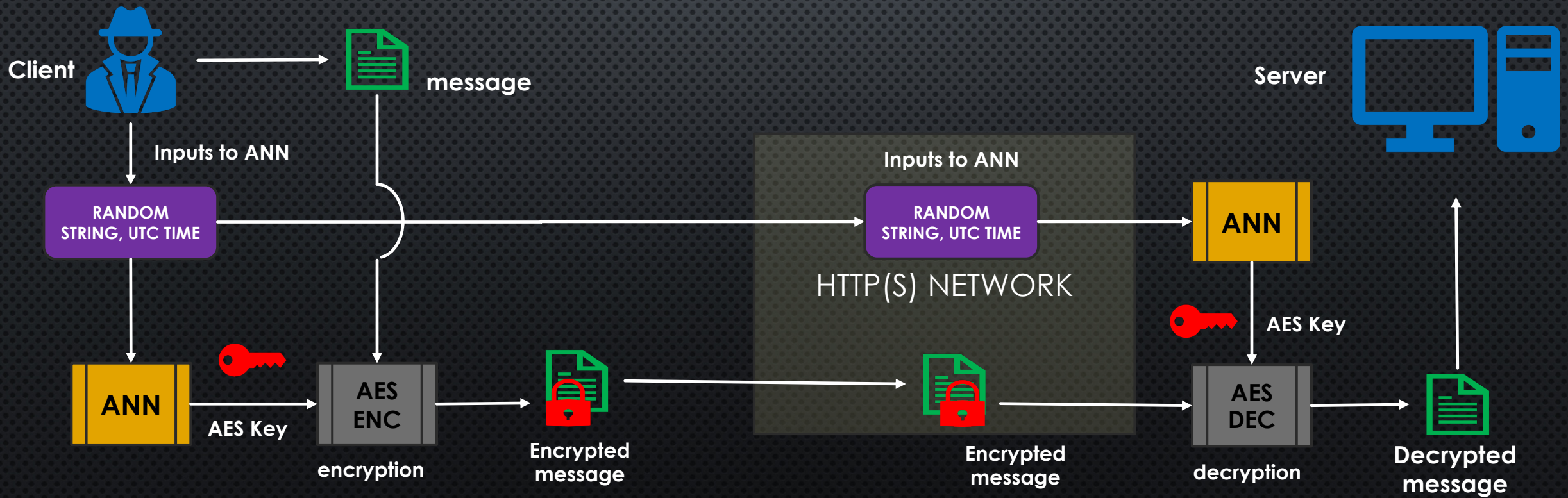
INTRODUCTION

- Advantages of using ANNs for AES Key Generation
  - Dynamic AES Key generation
  - UTC Time Stamp sensitivity to prevent Replay attacks
  - AES Key Hidden from the network
  - Hard to guess AES Key from inputs fed into ANN
  - Same ANN when fed with same inputs produces the same AES Key



INTRODUCTION CONTD….

- Keep a shared ANN between Client and the Server.
- Generate AES Keys using Artificial Neural Networks (ANNs) and encrypt message to be sent.
- Don't' share AES Cipher Keys; share the "recipe" to generate those keys(Random String & UTC Time Seconds)

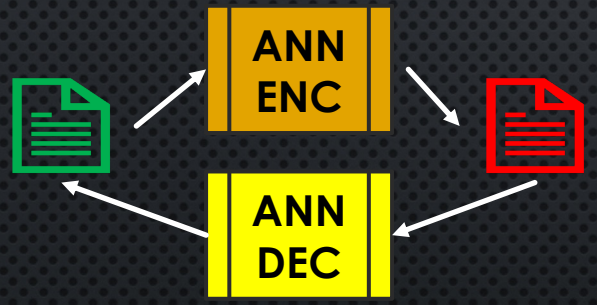


AIM & OBJECTIVES

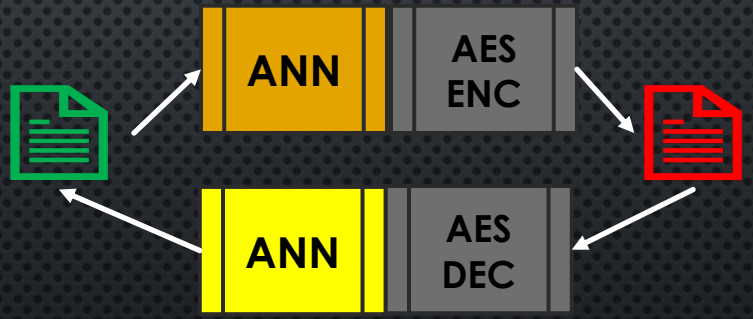
- Aim: Use ANNs for Client-Server Secure communication
- Objectives inline with the aim:
  - Server to generate ANN upon Client request and deliver it securely with the Client
  - Non-repeatability of AES Keys and non-collision of AES Key between two ANNs
  - Non-Correlation of ANN inputs vs Outputs
  - Replay attack prevention (reject stale messages)
  - Non feasibility of Learner-ANN Model

LITERATURE REVIEW

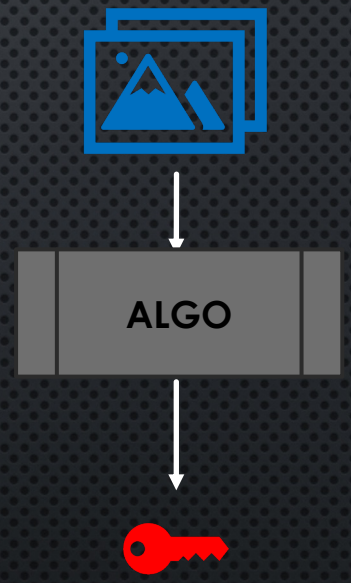
- Two decades of research work with ANNs in the world of Cyber Security.
- More specifically lot of research around AES/DES Encryption mechanisms in combination with ANNs.
- Research work categorization



ANN Used for Encryption and Decrypted on a fixed dataset

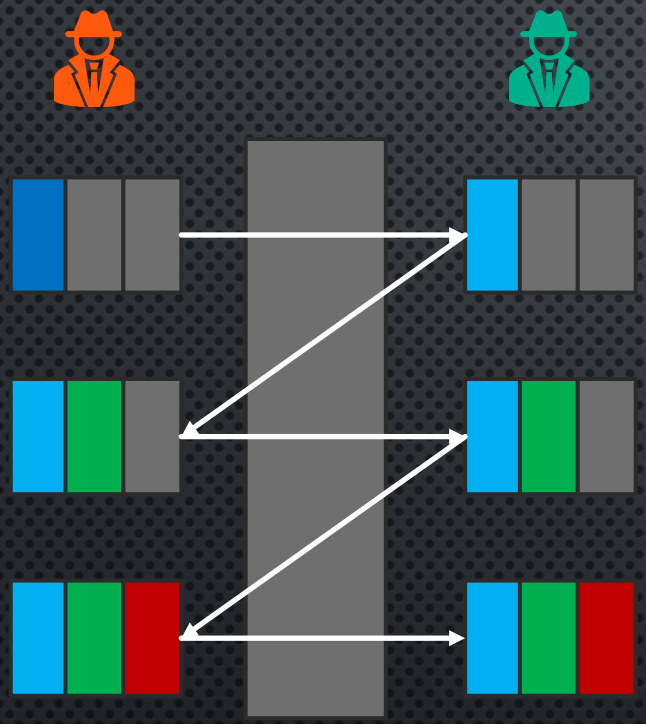


ANN Outputs Encrypted with AES layer and decrypted back using AES layer before processing

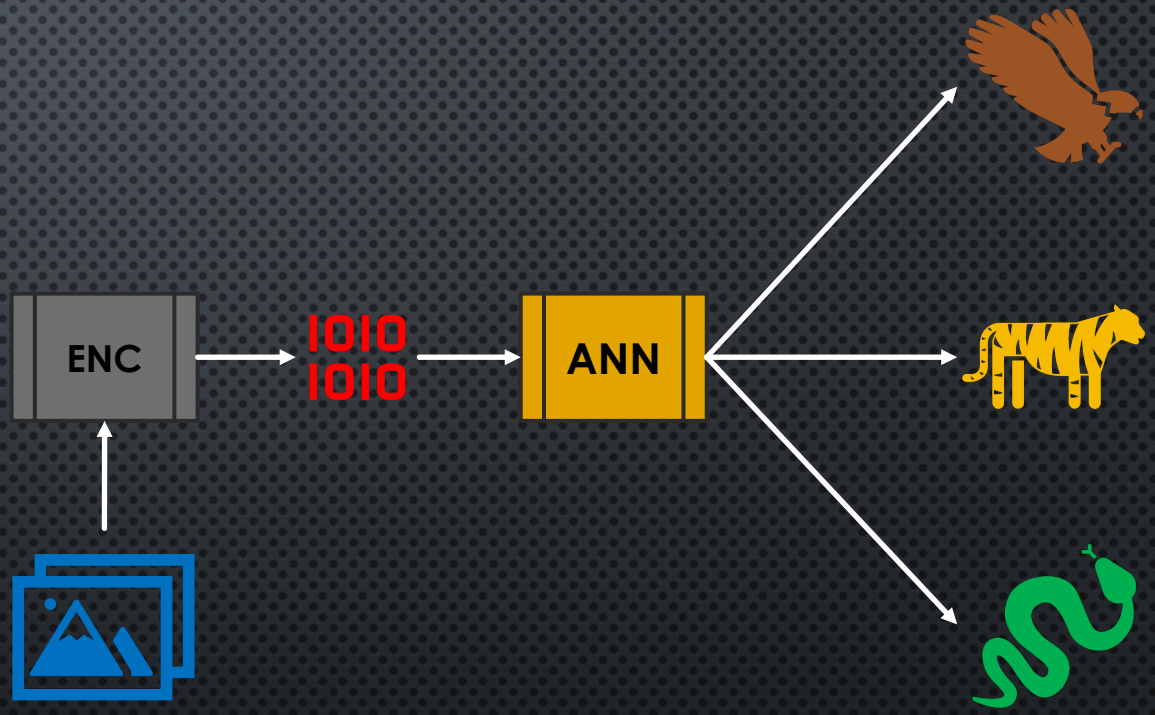


Logic/Algo used in producing Cipher Keys using multimedia (Images)

LITERATURE REVIEW CONTD...

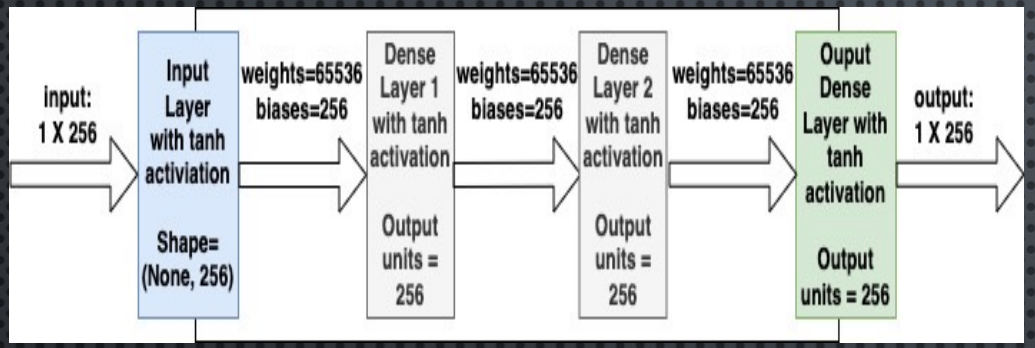


Two Clients realizing same ANNs weights and biases through Mutual Synchronization over unsecure Network

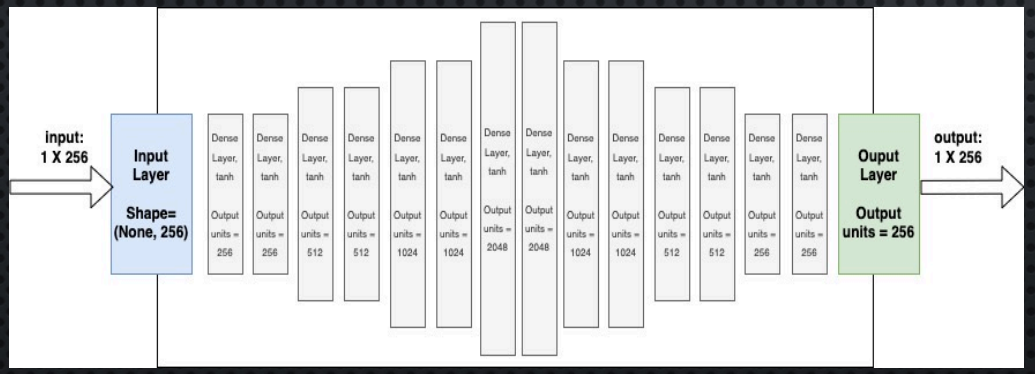


ANNs accepting encrypted inputs for making prediction/categorization of data

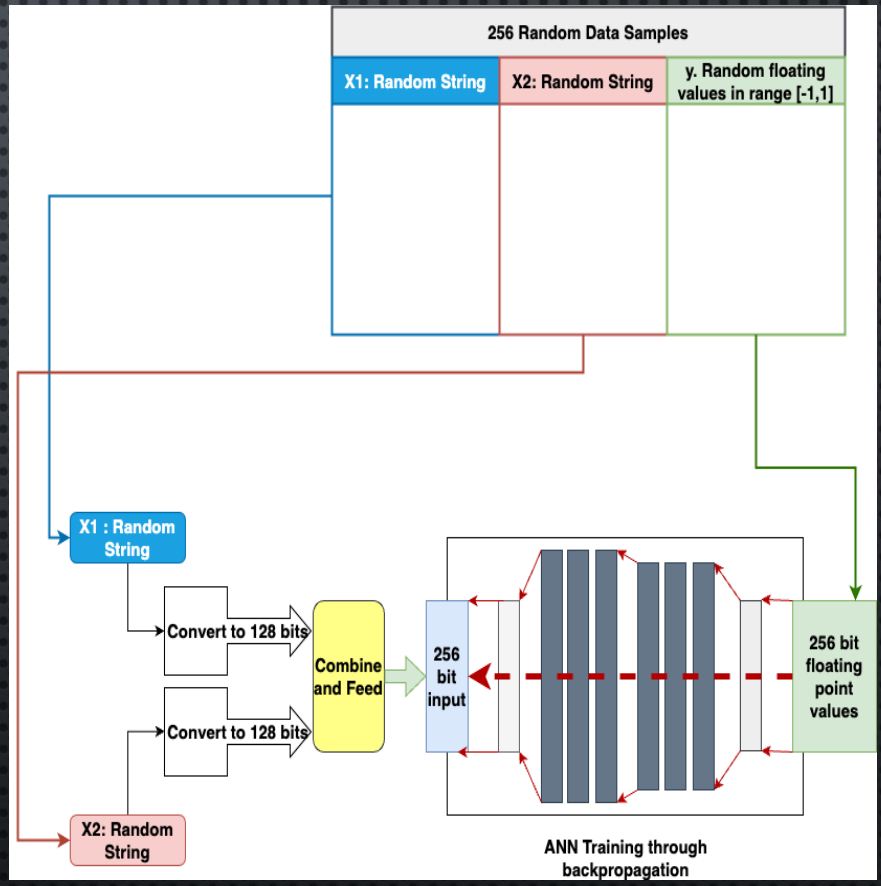
METHODOLOGY



Simple ANN

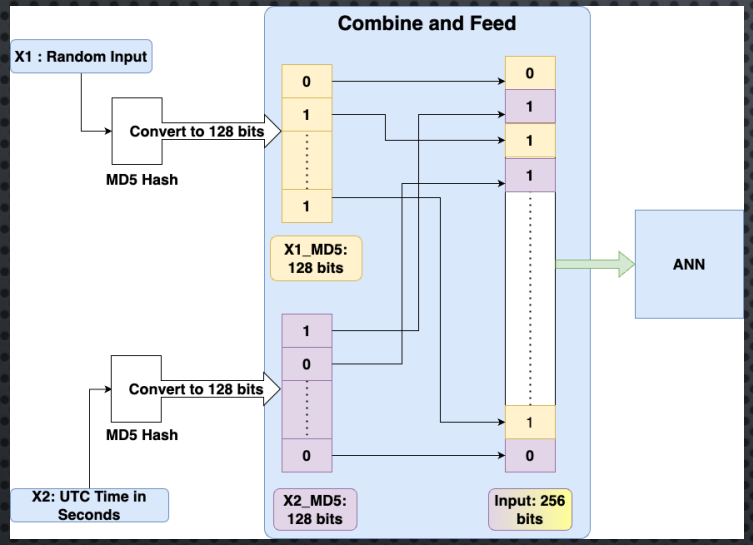


Complex ANN

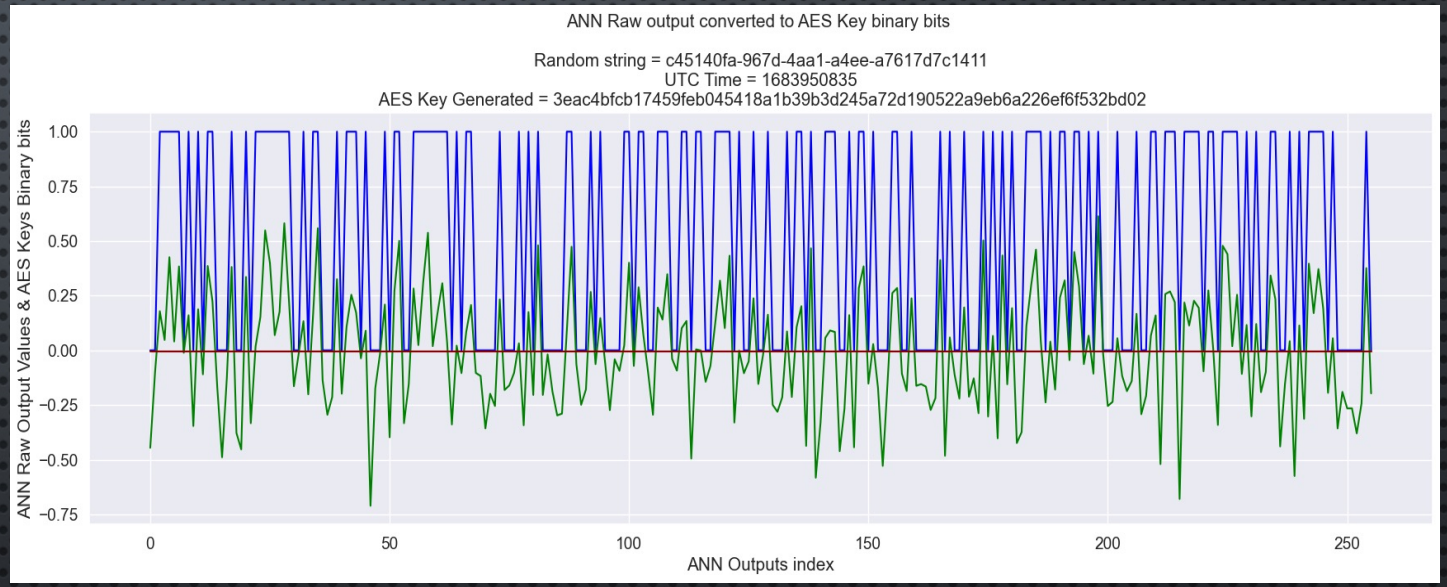


Quick Train ANN

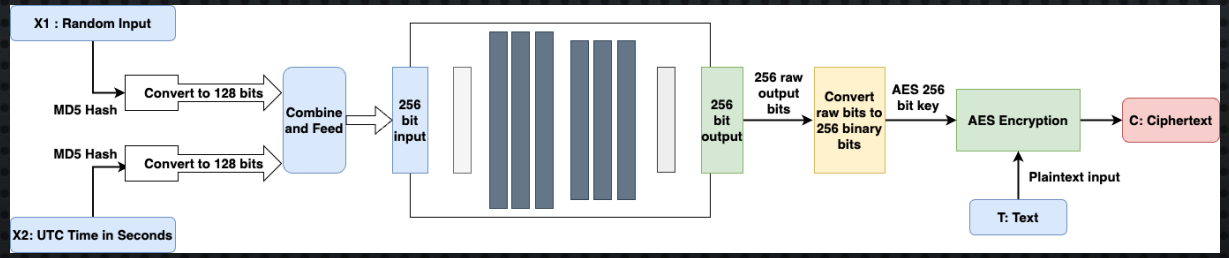
METHODOLOGY CONTD…



Combine and Feed: Two 128-bit MD5 hashes fed into ANN

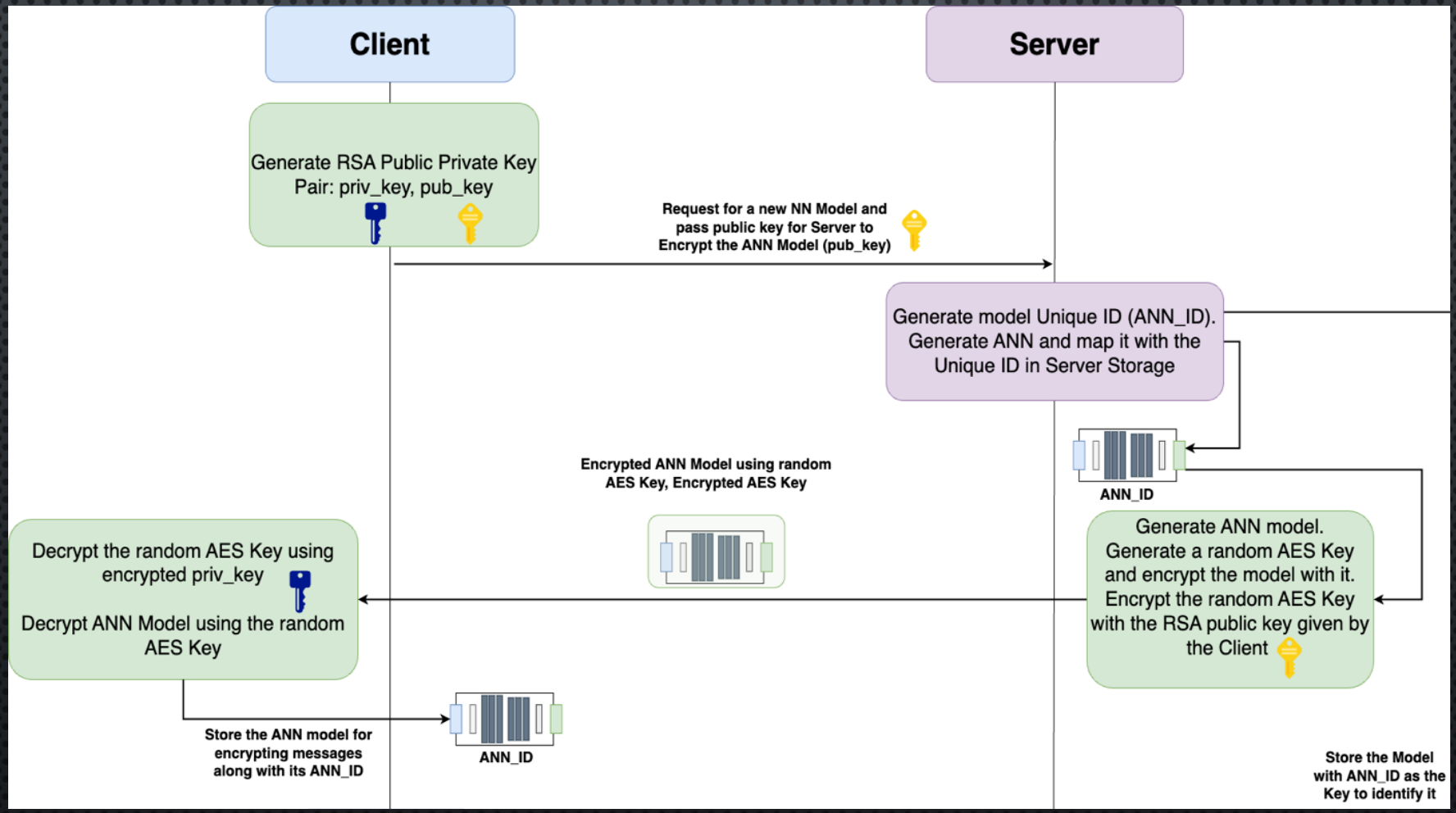


ANN floating points vector output converted into AES Key bits using mean value method



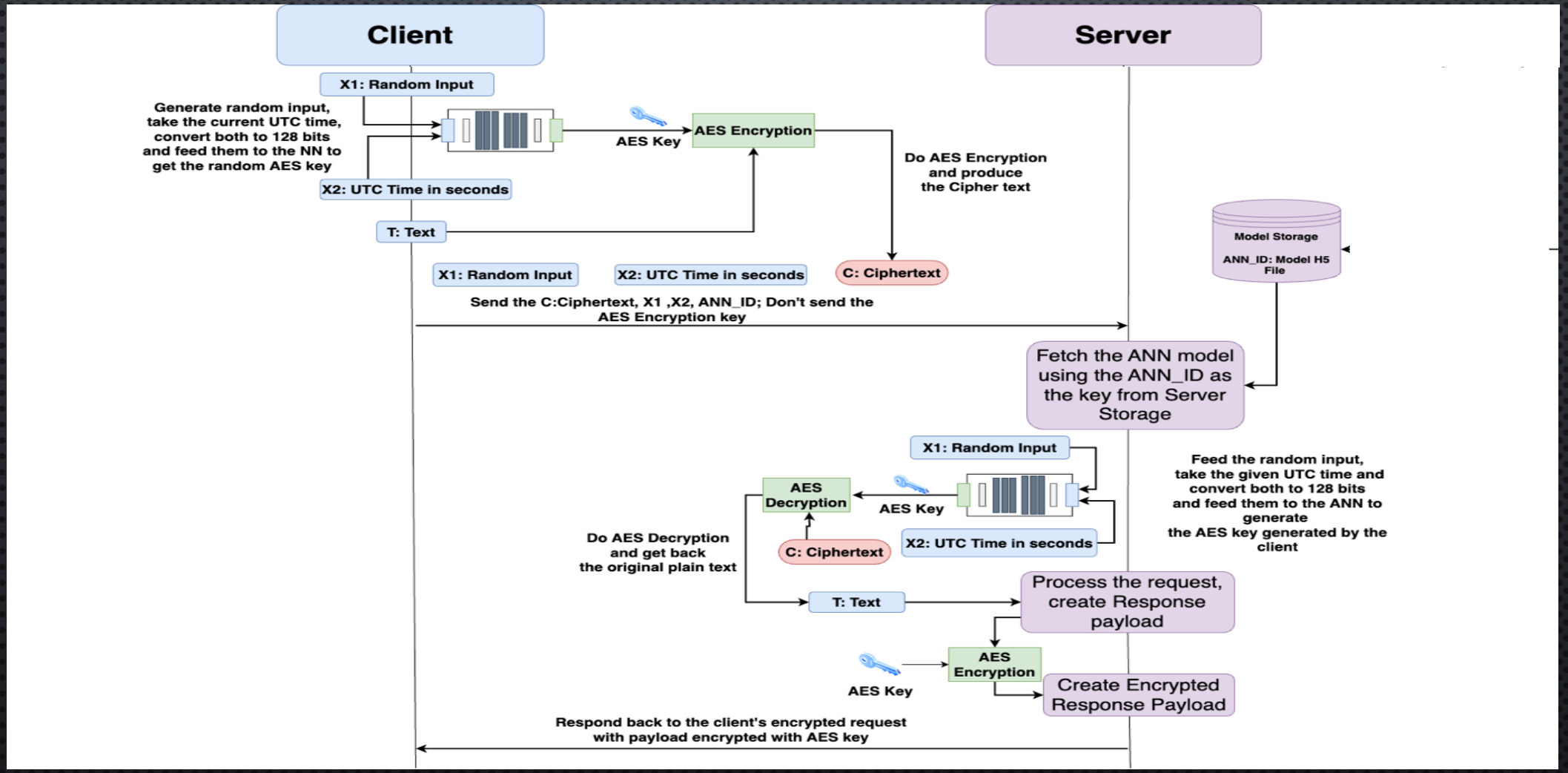
Feed ANN with X1 (random string md5) and X2 (UTC time seconds md5) to get random output vector

METHODOLOGY CONTD...



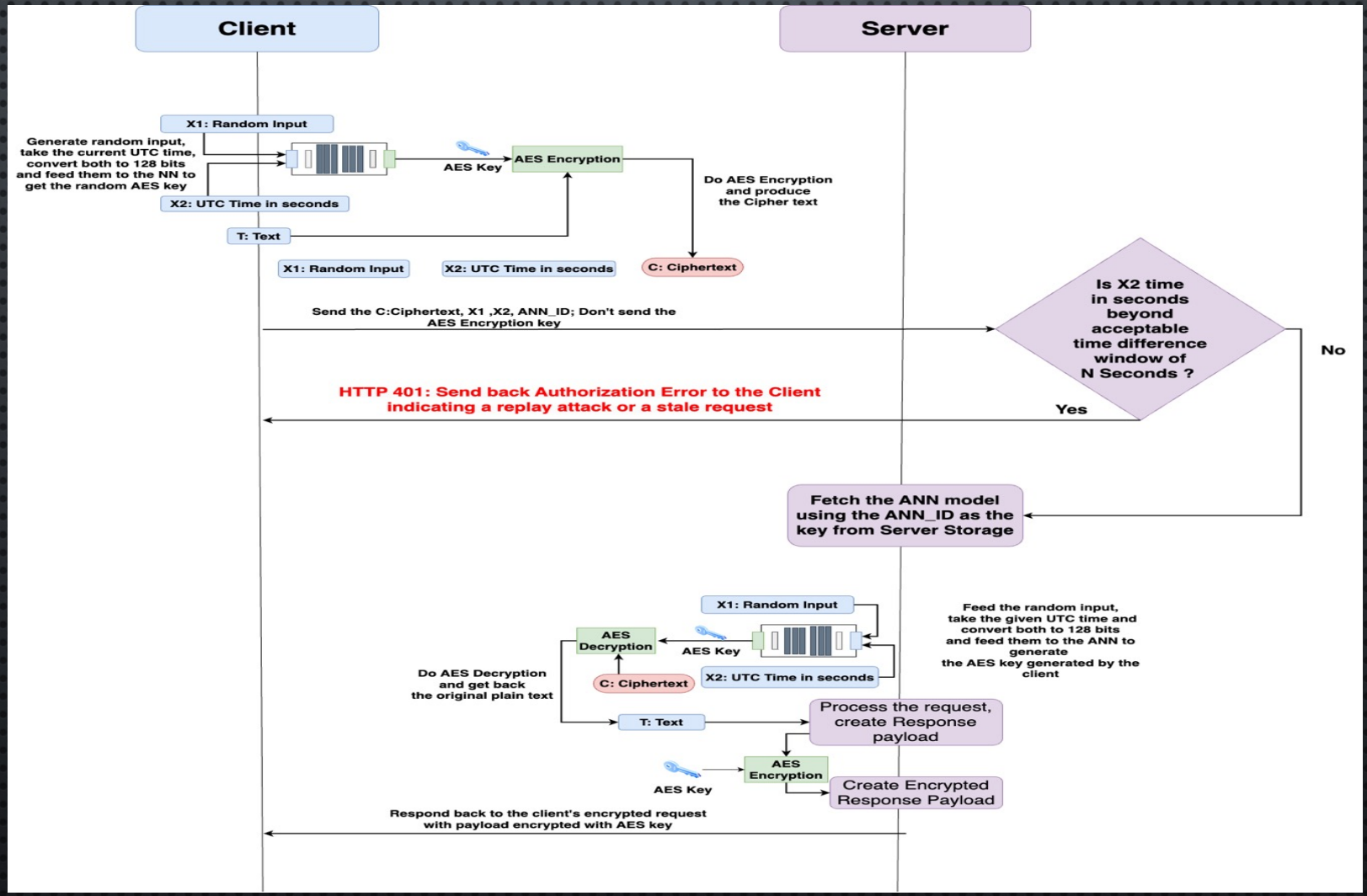
Client fetching ANN securely from Server by sending its RSA Public Key

METHODOLOGY CONTD…



Client and Server using shared ANN for Secure Communication

METHODOLOGY CONTD…



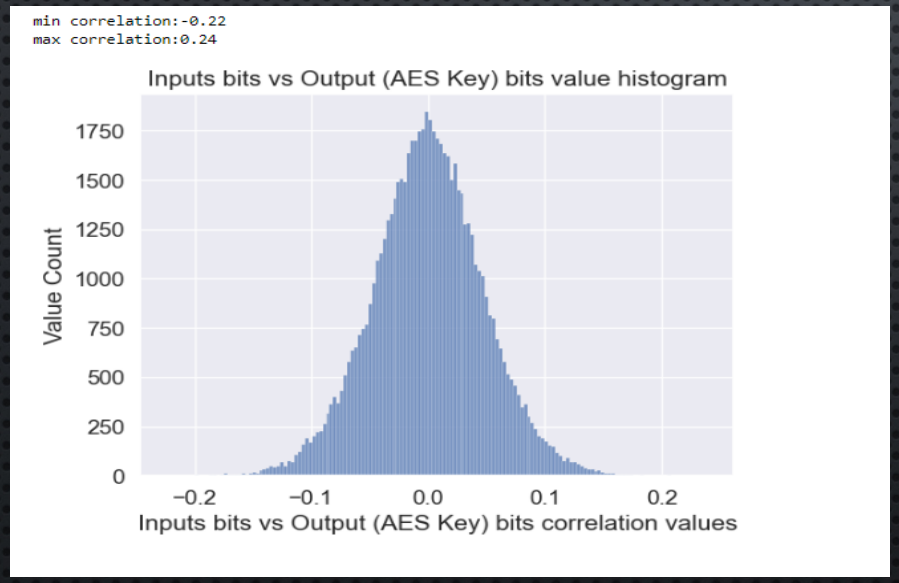
Replay Attack Scenario: Send an old UTC Time stamp or change UTC time stamp of an old message; The Server rejects the Client payload

RESULTS & DISCUSSIONS

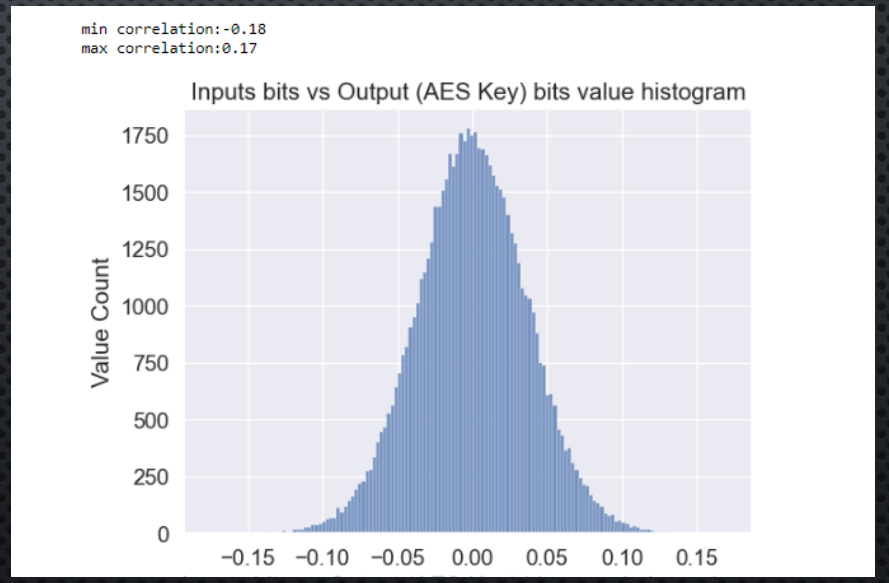
Client	Server	ANN Model Generation, Quick Training and sharing in format specific to the client	Secure ANN Delivery with encryption using Client generated RSA Public key encryption	Encrypted Communication using dynamically generated AES Key (without sharing those Keys on network)	Replay Attack Scenarios
Python Client	Python Flask Application Server	Simple ANN generated with H5 format upon Client request	Encrypted H5 with RSA 2048-bit public key and shared with the Client	Client could decrypt and load the H5 file and use it for secure communication with the Server.	Handled
Java Client	Python Flask Application Server	Simple ANN generated with H5 format upon Client request	Encrypted H5 with RSA 2048-bit public key and shared with the Client	Client could decrypt and load the H5 file and use it for secure communication with the Server.	Handled
JavaScript Client	Python Flask Application Server	Simple ANN generated in TFJS format (folder with model.json and .bin files) upon Client request	Encryption not supported with TensorflowJS http(s) model loading	Client could load the ANN model using TensorflowJS and use it for secure communication with the Server.	Handled

RESULTS & DISCUSSIONS CONTD…

ANN Model Type	Quick Trained	Number of Data Samples to Example Correlation	Correlation values between input bits and output bits (AES Key bits)	Conclusion
Simple ANN	Yes	10000	-0.22 to +0.24	Weak Correlation
Complex ANN	Yes	10000	-0.18 to +0.17	Weak Correlation



Simple ANN: Correlation values range between input bits and output bits; range (-0.22 to 0.24) centering around zero



Complex ANN: Correlation values range between input bits and output bits; range(-0.18 to 0.17) centering around zero

RESULTS & DISCUSSIONS CONTD…

ANN Model Type	Quick Trained	Iterations	Conclusion
Simple ANN	Yes	100000 (One Lakh)	No Repetition of AES Keys
Simple ANN	No	100000 (One Lakh)	No Repetition of AES Keys
Complex ANN	Yes	100000 (One Lakh)	No Repetition of AES Keys
Complex ANN	No	100000 (One Lakh)	No Repetition of AES Keys

Test for non repetition of AES Keys

ANN Model Type	Quick Trained	Iterations	Any common Key generated by both the models	In any instance, with the same inputs, do both the models produce a same AES Key
Simple ANN	Yes	100000 (One Lakh)	No	No
Complex ANN	Yes	100000 (One Lakh)	No	No

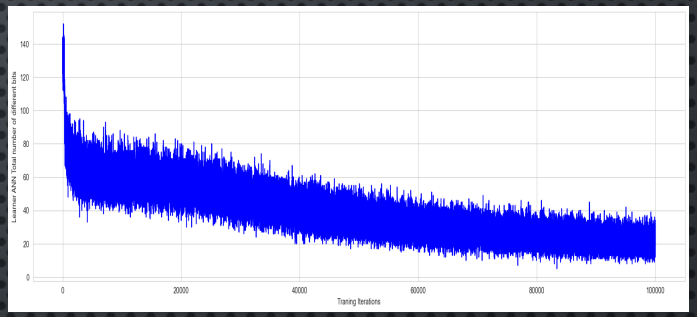
Test for non conflict of AES Keys between two ANN models

ANN Model Type	Format	Approximate File Size
Simple ANN	H5	800 KB to 1.5 MB
Complex ANN	H5	60 to 120 MB
Simple ANN	TFJS	Model.json file 3 KB One .bin shard file 80 KB
Complex ANN	TFJS	Model.json file 12 KB Around fifteen .bin shards 4 MB each

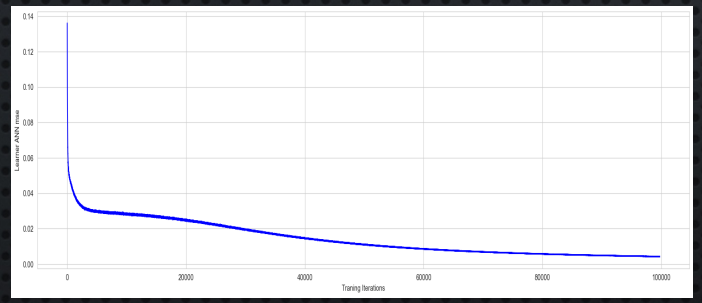
Model Formats and Sizes

RESULTS & DISCUSSIONS CONTD…

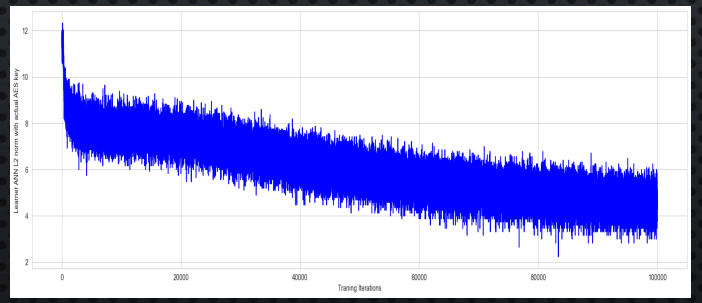
ANN Model Type	Quick Trained	Iterations	Learner ANN Metrics	Observation
Simple ANN, Learner ANN which is another Simple ANN	Yes	100000 (One Lakh)	MSE	Initially reduces but flattens out
			Accuracy	Rises but does in a range of values. Does not narrow down.
			L2 Norm	Initially reduces but flattens out. Does not touch zero
			Total number of different bits between predicted and actual AES	Reduces but does not reach zero



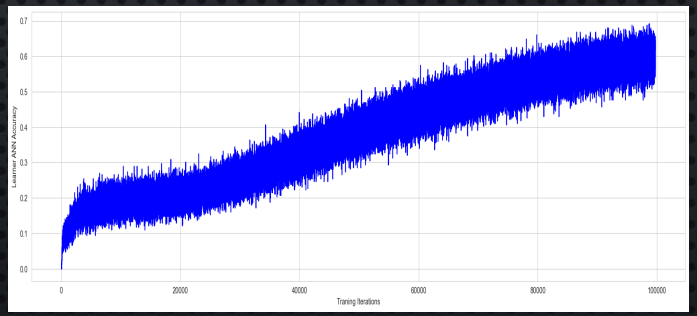
Learner ANN Total number of different bits of AES Key with subject ANN's AES Key bits: Never touches zero



Learner ANN MSE

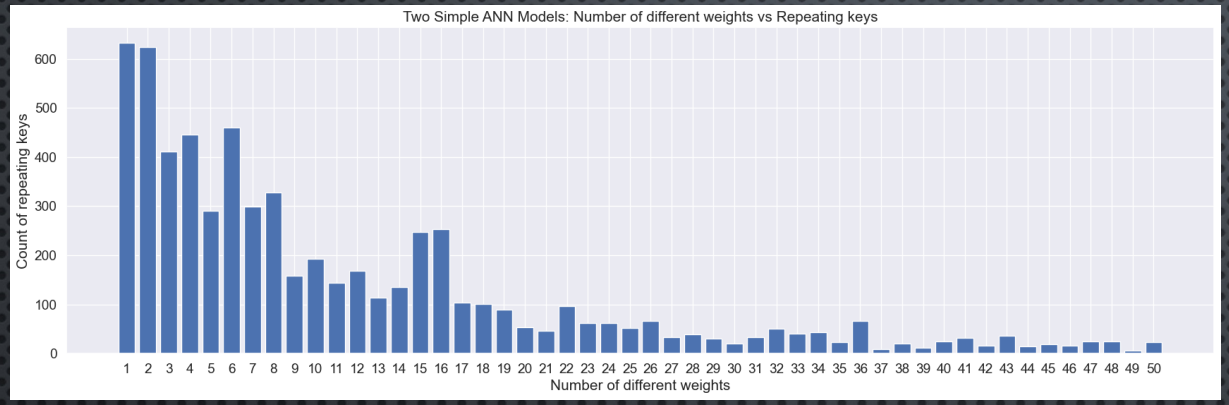


Learner ANN L2 Norm of AES Key with subject ANN's AES Key: Never touches zero

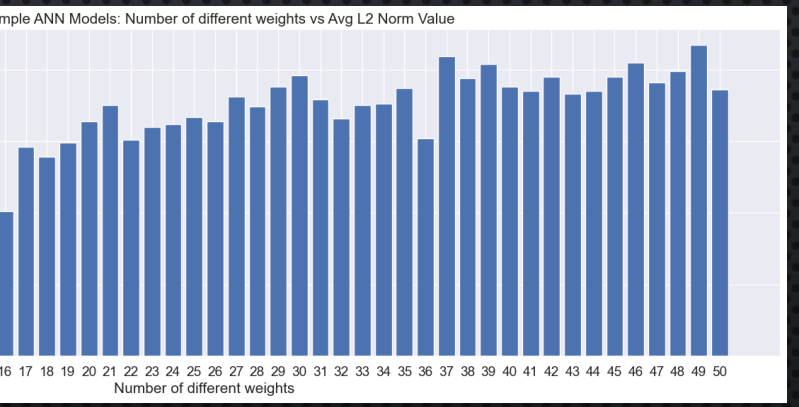


Learner ANN accuracy reaches 70% only

RESULTS & DISCUSSIONS CONTD…



Effect of number of different weights between two ANNs on ANN Key repetitions. Even at as low of 50 different weights repetition of AES Keys drastically reduces



Effect of number of different weights between two models on the L2 Norm between AES Keys they generate.

RESULTS & DISCUSSIONS CONTD…

ANN Model Type	Number of Datapoints	Number of AES Key Mismatches	Mismatch percentage	levenshtein distance between Remote Server's AES Key vs Local machine AES Key
Simple ANN	50000	2	0.004%	1 (single binary bit difference)
Complex ANN	50000	10	0.02%	1 (single binary bit difference)

AES KEY MISMATCH ILLUSTRATION:

Remote Server's AES Key:

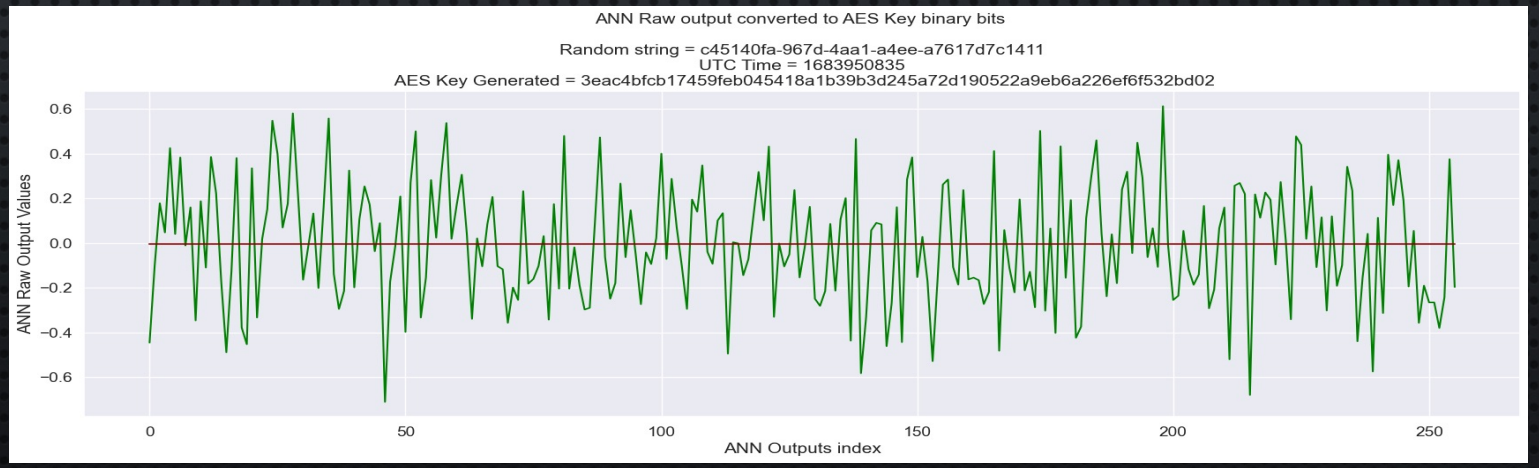
c9a628d65f3ed515f124628251715eb481ff9a125f7dc8ce2b188a2bef84c15c

Locally Generated AES Key:

C9a628d65f3ed515f124628251715eb481ff9a125f7dc8ce2b188a29ef84c15c

b in binary is 1011 => Server produced 1

9 in binary in 1001 => Java Client produced 0



CONCLUSIONS & FUTURE RECOMMENDATIONS

- CONCLUSIONS:
  - Is secure communication with ANNs achieved ?
  - Applicable to different types of Clients ?
  - Secure delivery of ANN using Client generated RSA Keypair?
    - Secure ANN deliver JS Client using Tensorflow JS ? Only on HTTPS
  - Error rate of AES Key mismatch with same model across different machine ? Tolerable (just 2/50000 for Simple ANN, 10/50000 for Complex ANN
    - Remediation on AES Key mismatch:
      - Re-generate a new AES Key
      - Define margin around ANN output floating point vector mean value within which all the points would be treated as fuzzy (tried for both 1 and 0)

CONCLUSIONS & FUTURE RECOMMENDATIONS CONTD..

- RECOMMENDATIONS:
  - More randomization: By creating ANNs with varying number of layer and architectural changes.
  - Better metric than MEAN value for ANN floating point values for converting to AES Key bits.
  - Secure Delivery of ANNs even with JS Client: Intercept XHR Encrypted Model Responses before Tensorflow JS loads the model. Decrypt the responses before handing over to Tensorflow JS.
  - Pre-Generate ANNs to avoid on demand generation. Maintain ANN inventory for Clients.
  - Build mobile applications with pre-fed ANNs in application binaries instead of serving ANNs over network.
  - ANNs as document encryptors.
  - ANNs as protocol layer security (like TLS)

GITHUB CODEBASE: FOR YOUR KEYS ONLY

- GitHub repository link: https://github.com/JSMitra/ForYourKeysOnly
  - Jupyter based Python notebook implementation: https://github.com/JSMitra/ForYourKeysOnly/blob/main/FYKO_JupyterNotebook/FYKO.ipynb
    - Contains Complete Code, POC, Experiments, Results and a Sample Python Client
  - Flask Application Server which can server ANNs and do secure communication with Clients: https://github.com/JSMitra/ForYourKeysOnly/tree/main/FYKO_FlaskApplication
    - Contains JS Client: https://github.com/JSMitra/ForYourKeysOnly/tree/main/FYKO_FlaskApplication/FYKO_JS
  - Java Client: https://github.com/JSMitra/ForYourKeysOnly/tree/main/FYKO_Clients/FYKO_Java
  - Sample Outputs using Java, JavaScript and Python Clients: https://github.com/JSMitra/ForYourKeysOnly/tree/main/FYKO_Client_SampleOutputs

THANK YOU !