Deep Learning-Based Differential Distinguishers for Cryptographic Sequences

Amrita Bose, Debranjan Pal, Dipanwita Roy Chowdhury

Department of Computer Science and Engineering,
Indian Institute of Technology Kharagpur

INDOCRYPT 2024, CHENNAI, Tamil Nadu, India

December 2024

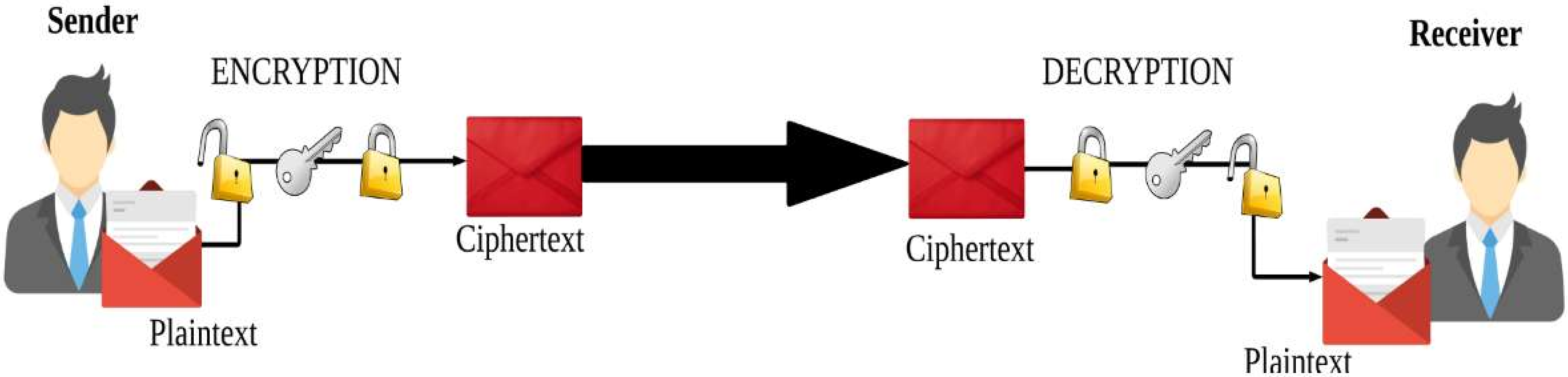
Outline

- Introduction
- A Brief History
- Deep Learning for Sequence Classification
- Neural Distinguishers based on Sequence Classification
- Results and Observation
- Conclusion

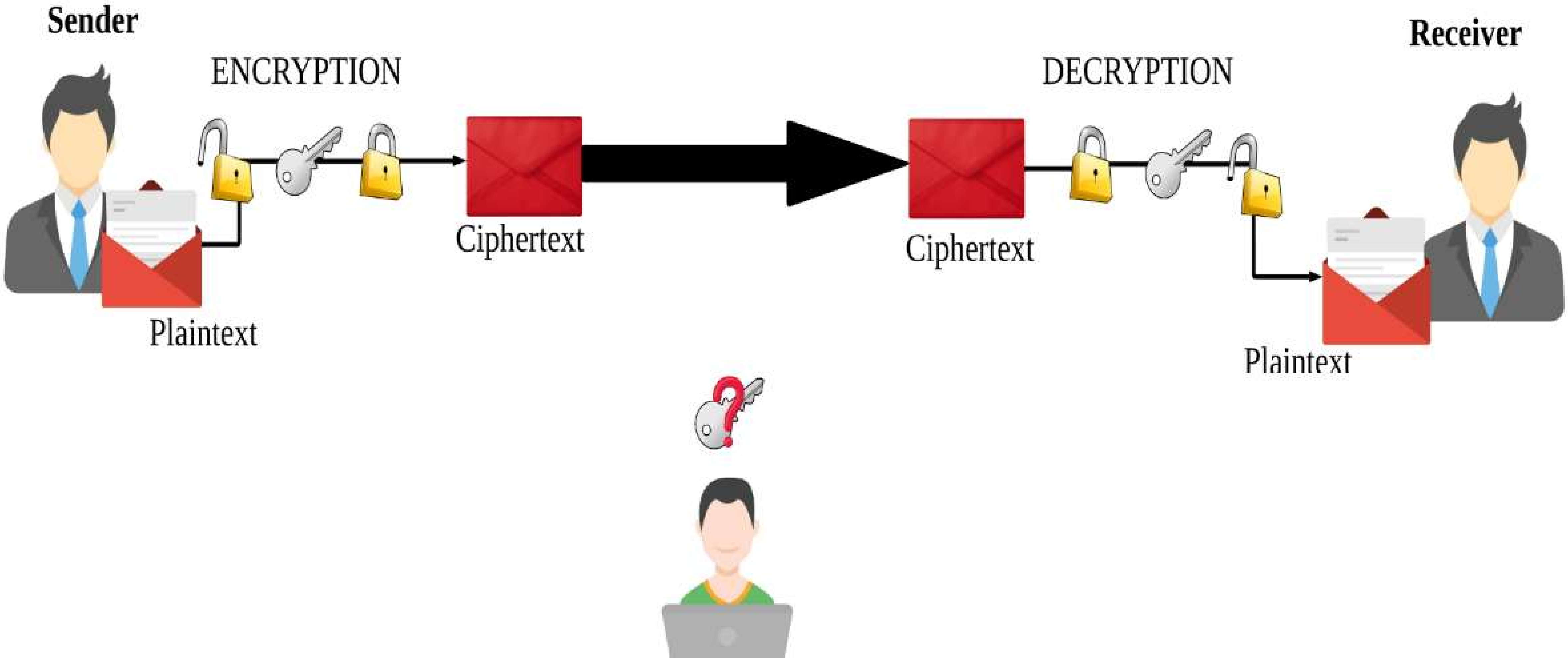
Outline

- Introduction
- A Brief History
- Deep Learning for Sequence Classification
- Neural Distinguishers based on Sequence Classification
- Results and Observation
- Conclusion

Introduction

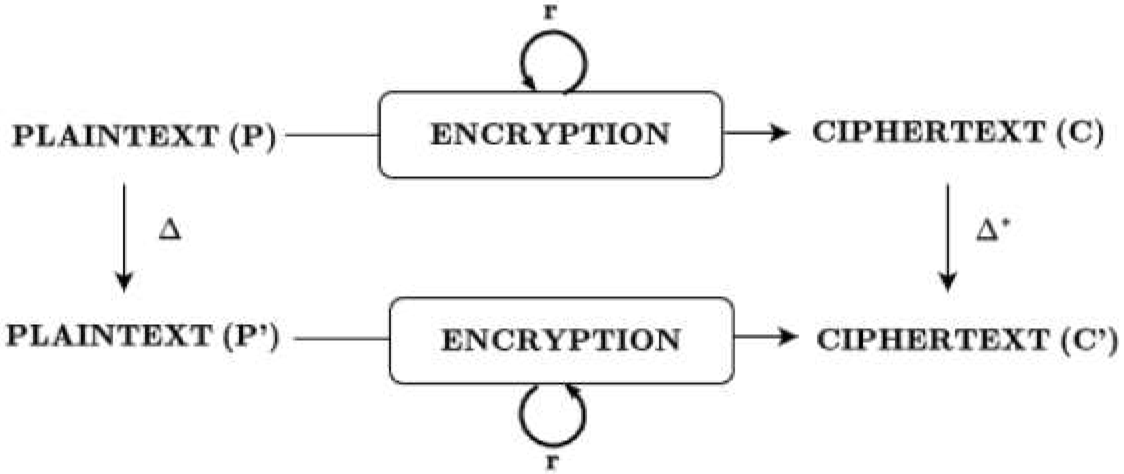


Introduction



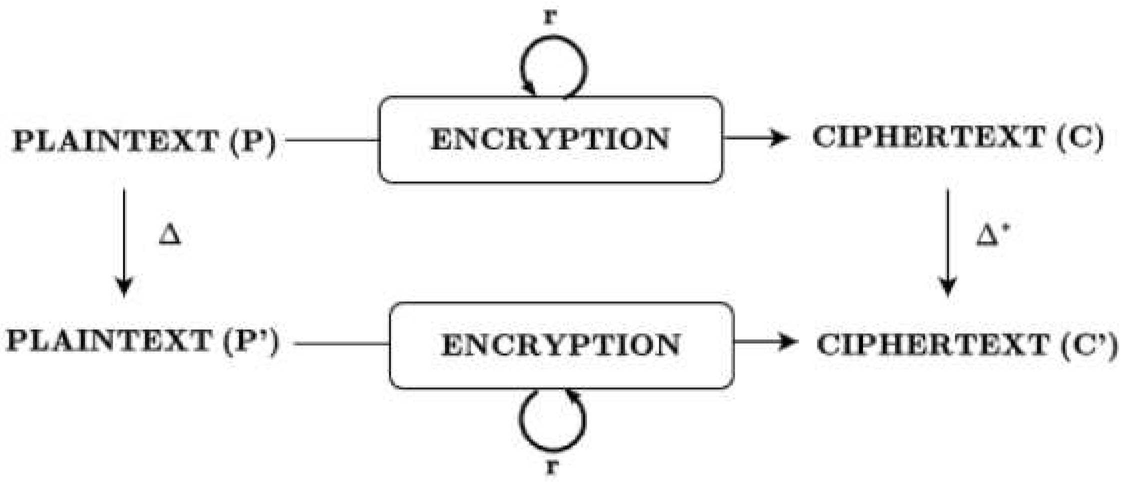
Introduction

- Differential Cryptanalysis:



Introduction

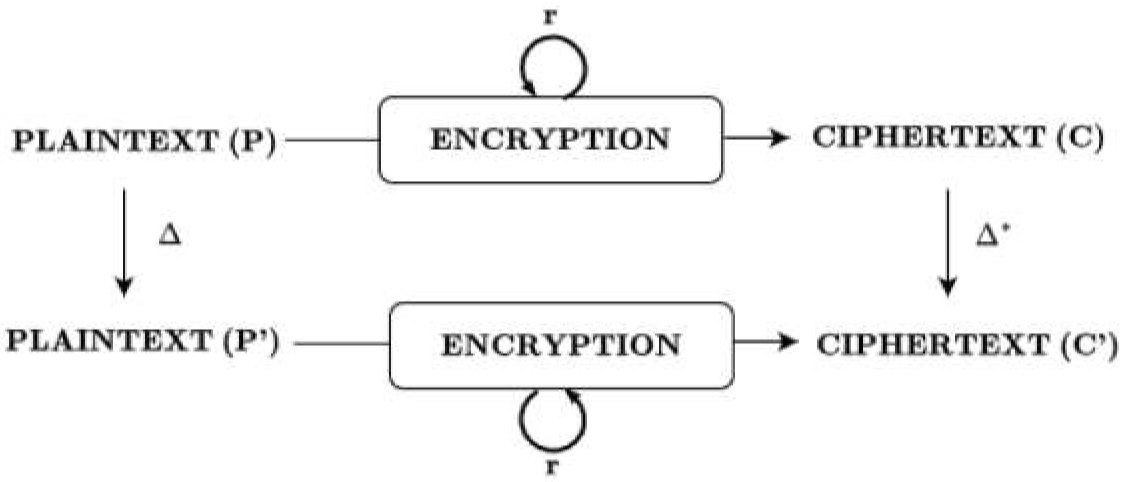
- **Differential Cryptanalysis:**

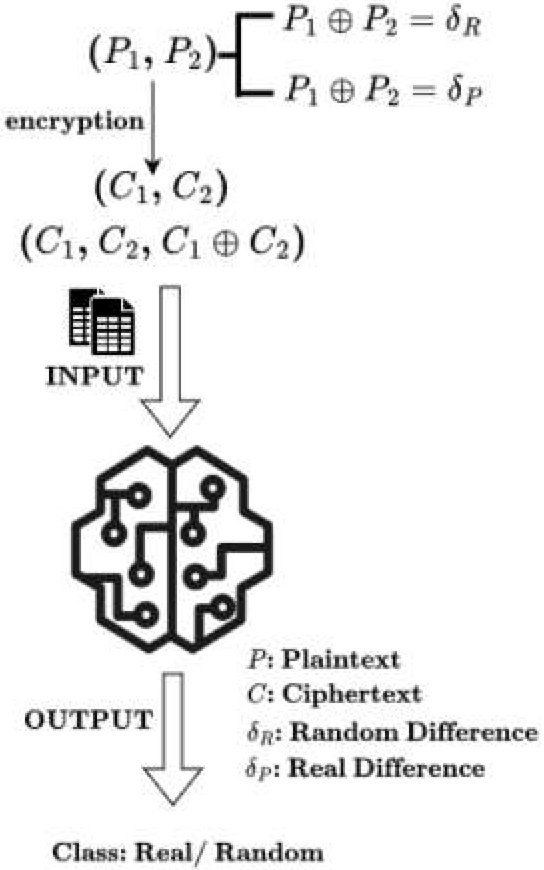


- Conventional Method →
  relies on Difference Distribution Table (DDT)

Introduction

- **Differential Cryptanalysis:**



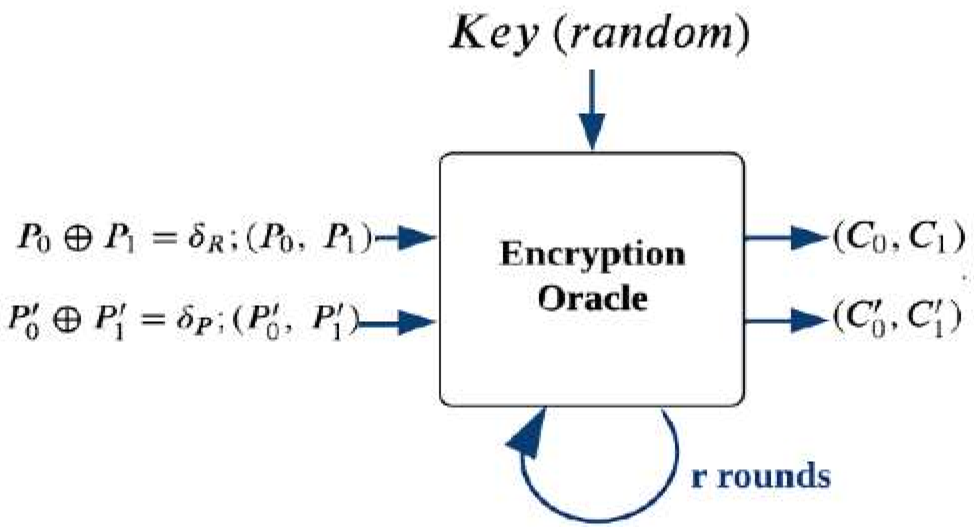


- Conventional Method →
  relies on Difference Distribution Table (DDT)

- New Research Direction → utilize deep learning model
  to get a distinguisher [Aron Gohr]

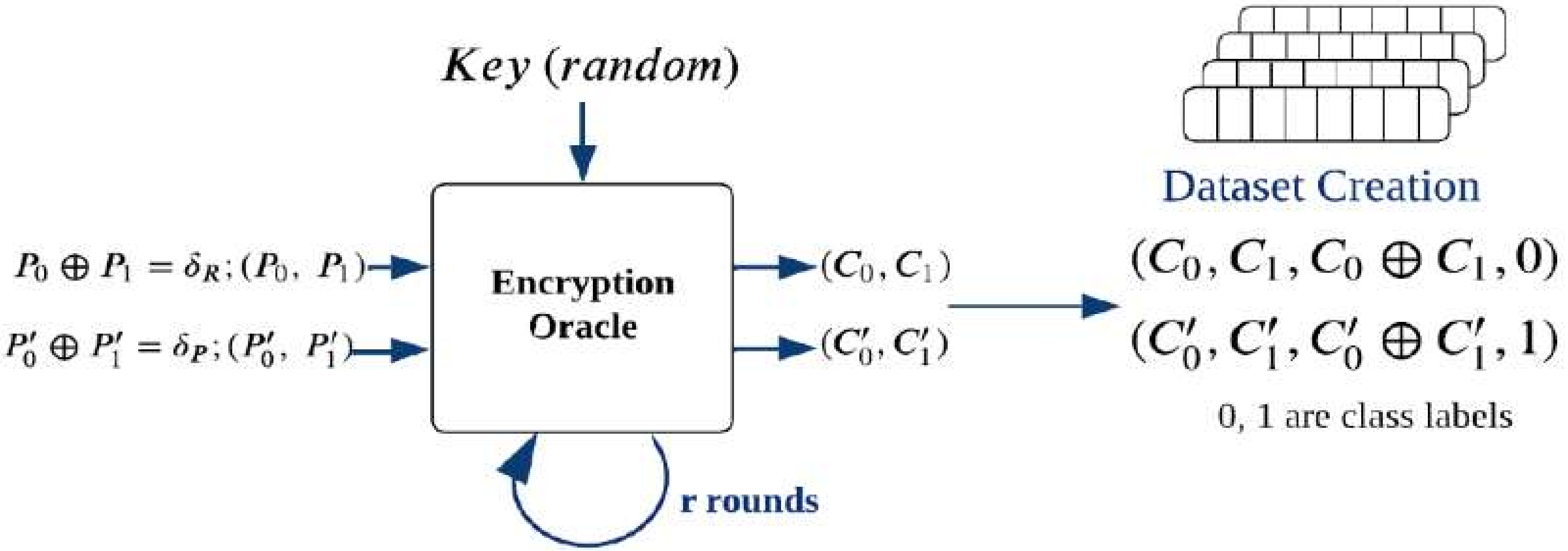
Introduction

- Deep Learning-based Distinguisher



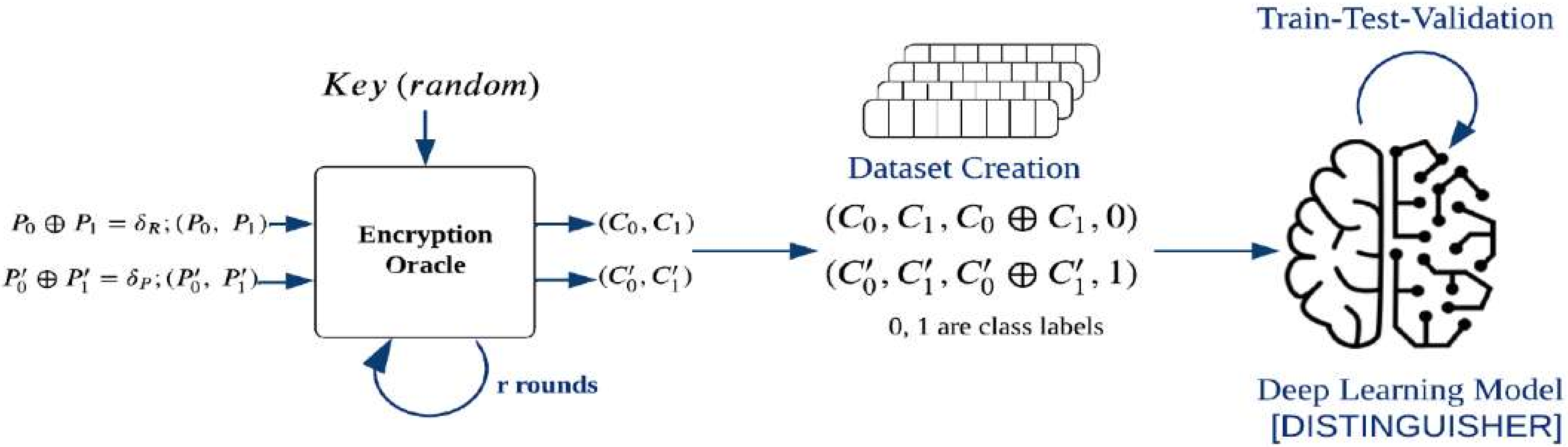
Introduction

- Deep Learning-based Distinguisher



Introduction

- Deep Learning-based Distinguisher



What is the purpose of a good distinguisher?

- a distinguisher can highlight whether an encryption scheme deviates from the expected random behavior
- some subtle patterns that may go unnoticed otherwise can be captured by an automated approach
- a good distinguisher is a prerequisite for a successful key recovery attack

Outline

- Introduction
- A Brief History
- Deep Learning for Sequence Classification
- Neural Distinguishers based on Sequence Classification
- Results and Observation
- Conclusion

A Brief History

1. Aron Gohr proposed distinguisher for *SPECK* [*ResNet*]
2. Zhang et al. proposed distinguisher for *TweGIFT-128* [*MLP*]
3. Wang et al. obtained distinguishers for *SPECK32/64* and *PRESENT64/80* [*Adaboost, DT, KNN, Logistic Regression, RFC, SVM, MLP, CNN, RNN, LSTM*]
4. Baksi et al. presented the notion of multiple input differences to define neural distinguishers [*MLP, CNN, LSTM*]
5. Mishra et al. proposed distinguisher for *GIFT64* and *PRIDE* [*MLP, CNN*]
6. Bacuieti et al. proposed using *Convolutional Autoencoders* as preprocessors before training the model.
7. Liu et al. obtained improved accuracy for *SPECK* [*CNN*]
8. Deng et al. obtained improved results for *SPECK* [*Attention+CNN*]
9. Bellini et al. proposed a *DBitNet* network [*CNN based*]
10. Shen et al. proposed a score distribution based technique [*MLP*]

Outline

- Introduction
- A Brief History
- Deep Learning for Sequence Classification
- Neural Distinguishers based on Sequence Classification
- Results and Observation
- Conclusion

Deep Learning for Sequence Classification

- *MLP* architectures $\rightarrow$ struggle to capture the spatial or temporal dependencies

Deep Learning for Sequence Classification

- *MLP* architectures → struggle to capture the spatial or temporal dependencies
- *CNN* → can capture local patterns in data but fails to interpret long-range dependencies

Deep Learning for Sequence Classification

- *MLP* architectures → struggle to capture the spatial or temporal dependencies
- *CNN* → can capture local patterns in data but fails to interpret long-range dependencies
- the position of a data point, with respect to other data points carry some vital dependencies

Deep Learning for Sequence Classification

- *MLP* architectures → struggle to capture the spatial or temporal dependencies
- *CNN* → can capture local patterns in data but fails to interpret long-range dependencies
- the position of a data point, with respect to other data points carry some vital dependencies
- text classification, language translation, time-series applications and speech recognition are examples of sequential data

Deep Learning for Sequence Classification

- *MLP* architectures $\rightarrow$ struggle to capture the spatial or temporal dependencies
- *CNN* $\rightarrow$ can capture local patterns in data but fails to interpret long-range dependencies
- the position of a data point, with respect to other data points carry some vital dependencies
- text classification, language translation, time-series applications and speech recognition are examples of sequential data
- in the present context: the ciphertexts can be considered a sequence of characters encoded by *ASCII* (American Standard Code for Information Interchange)

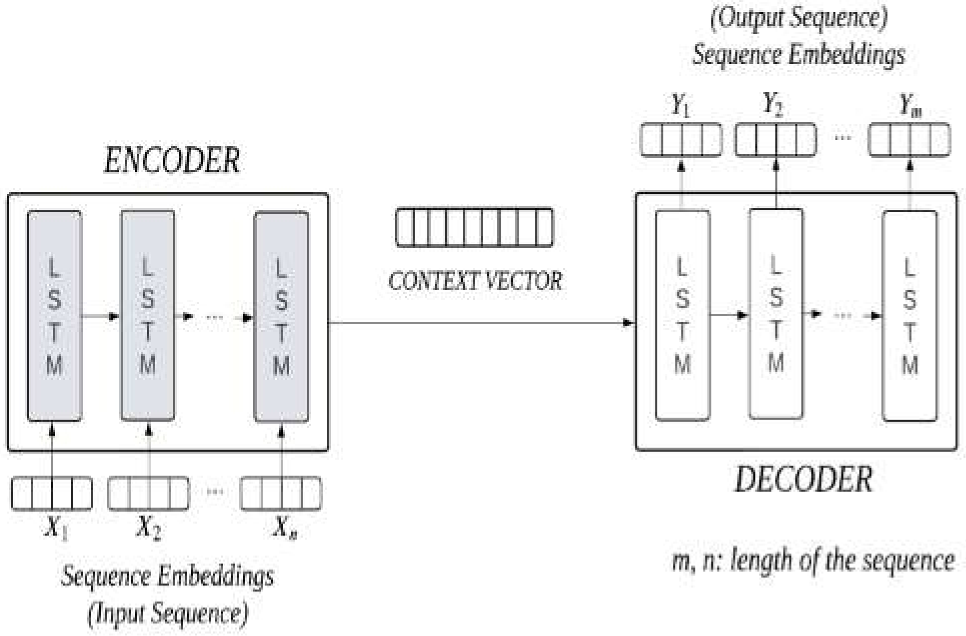
Deep Learning for Sequence Classification

- *MLP* architectures → struggle to capture the spatial or temporal dependencies
- *CNN* → can capture local patterns in data but fails to interpret long-range dependencies
- the position of a data point, with respect to other data points carry some vital dependencies
- text classification, language translation, time-series applications and speech recognition are examples of sequential data
- in the present context: the ciphertexts can be considered a sequence of characters encoded by *ASCII* (American Standard Code for Information Interchange)

e.g., for a 64−bit blocksize cipher : [28, 121, 9, 250, 30, 66, 12, 97]

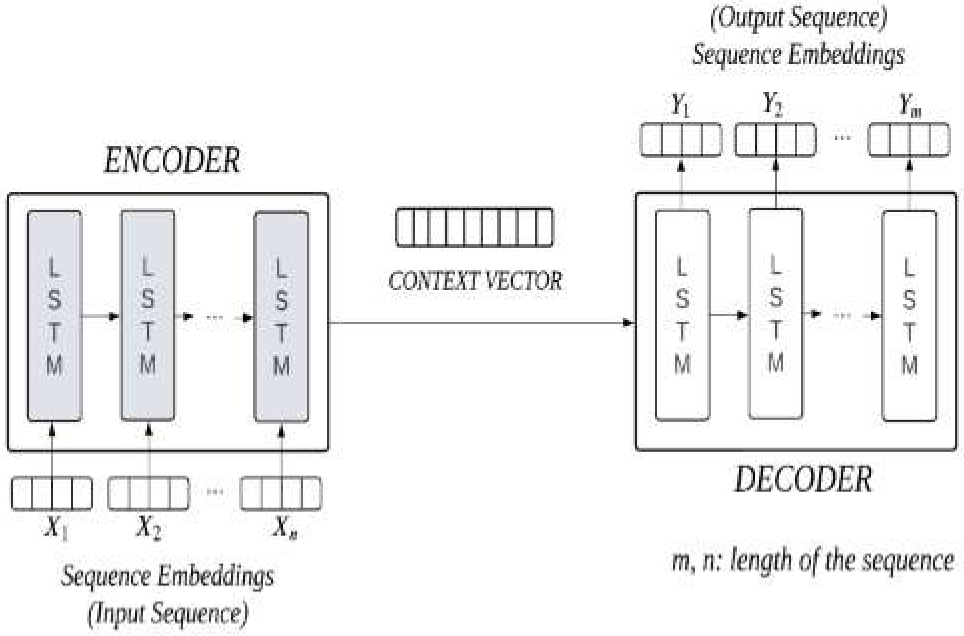
Sequence Detection Models

- Encoder-Decoder Network: proposed in 2014, a team led by Ilya Sutskever from Google

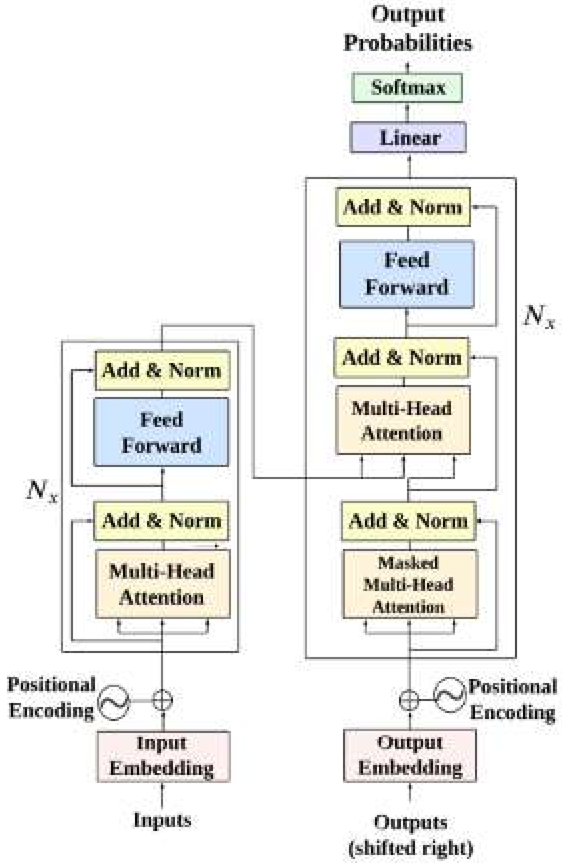


Sequence Detection Models

- Encoder-Decoder Network: proposed in 2014, a team led by Ilya Sutskever from Google



- Transformer: published in 2017 by Google Brain titled "Attention is all you need"



Outline

- Introduction
- A Brief History
- Deep Learning for Sequence Classification
- Neural Distinguishers based on Sequence Classification
- Results and Observation
- Conclusion

The Neural Distinguisher Pipeline

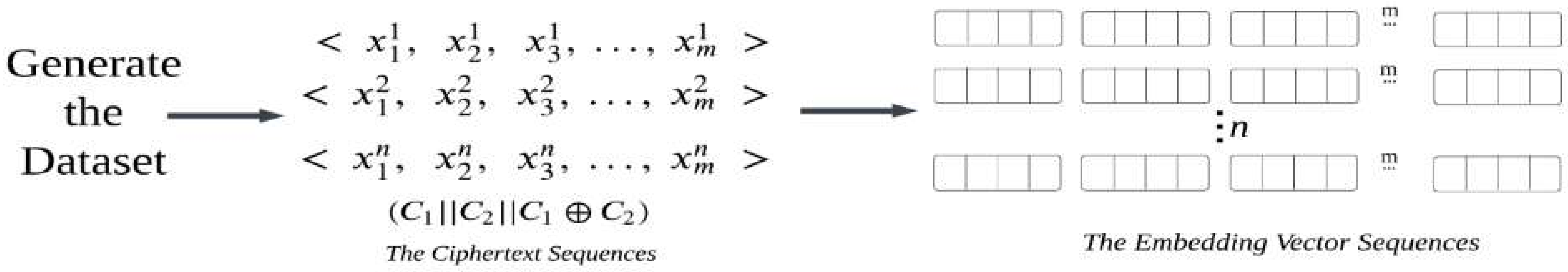
Generate the Dataset →

$$< x_1^1,\ x_2^1,\ x_3^1,\ \ldots,\ x_m^1 >$$
$$< x_1^2,\ x_2^2,\ x_3^2,\ \ldots,\ x_m^2 >$$
$$< x_1^n,\ x_2^n,\ x_3^n,\ \ldots,\ x_m^n >$$

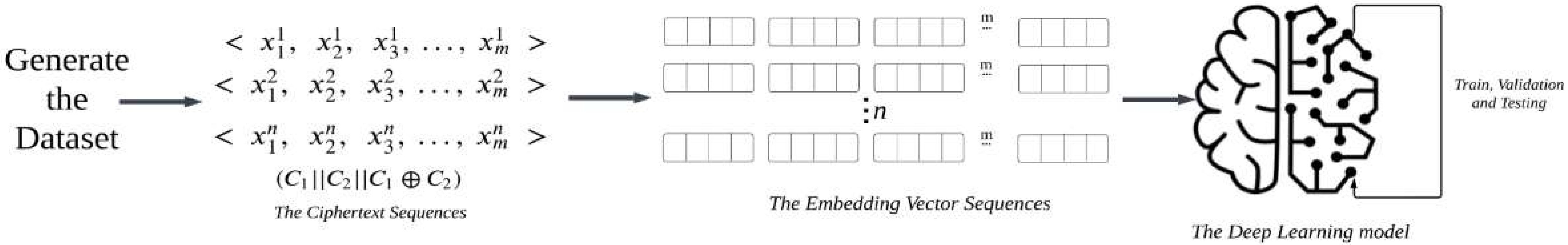
$(C_1 || C_2 || C_1 \oplus C_2)$

The Ciphertext Sequences

The Neural Distinguisher Pipeline



The Neural Distinguisher Pipeline



Data Generation

The Algorithm used for creating dataset:

Algorithm 1 Data Generation Algorithm: Differential Distinguisher

Input: Plaintext difference, δ_P; number of encryption rounds, r; Dataset size, $\mathcal{N}$

Output: Dataset, $\mathcal{D}$

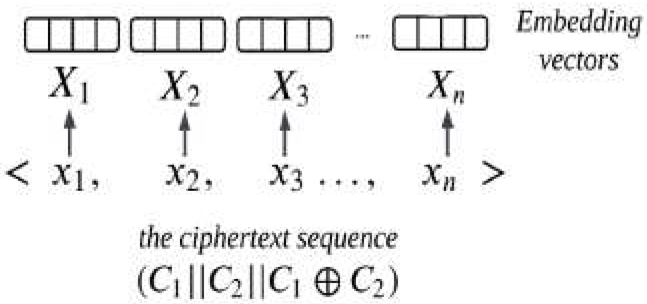
$\mathcal{D} \leftarrow \emptyset$
for $i = 1$ to $\mathcal{N}$ **do**
    $\mathcal{K} \leftarrow RANDOM\_BYTES()$ ▷ generating random key, $\mathcal{K}$
    $\mathcal{P}_1 \leftarrow RANDOM\_BYTES()$ ▷ generating random plaintext, $\mathcal{P}$
    $\mathcal{C}_1 \leftarrow ENCRYPT(\mathcal{P}_1, \mathcal{K}, r)$ ▷ call the encryption oracle, returns ciphertext, $\mathcal{C}$
    $choice \leftarrow random\_number(0, 1)$
    if $(choice = 0)$ **then** ▷ random difference
        $\mathcal{P}_2 \leftarrow RANDOM\_BYTES()$
        $Label \leftarrow 0$
    else ▷ real difference
        $\mathcal{P}_2 = \mathcal{P}_1 \oplus \delta_{\mathcal{P}}$
        $Label \leftarrow 1$
    end if
    $\mathcal{C}_2 \leftarrow ENCRYPT(\mathcal{P}_2, \mathcal{K}, r)$
    $\mathcal{C}_{XOR} \leftarrow \mathcal{C}_1 \oplus \mathcal{C}_2$
    $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_{XOR}, Label)$ ▷ include in the dataset
end for
return $\mathcal{D}$

Deep Learning Models

- *LSTM-based Encoder Classifier (LbEC)*

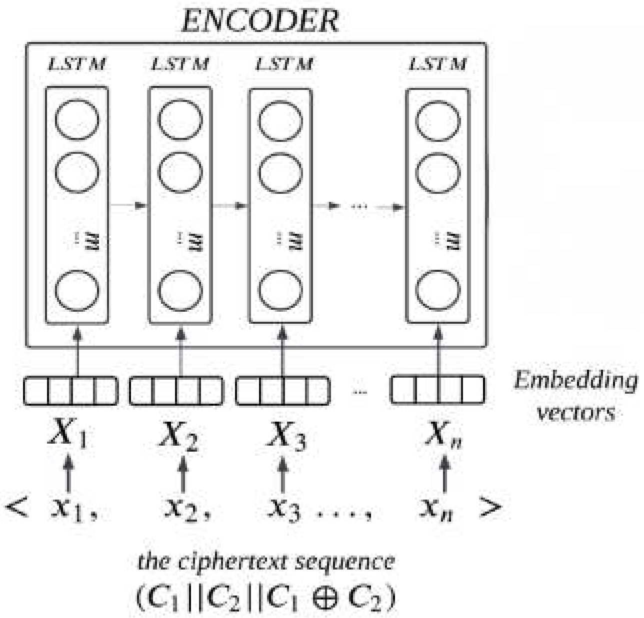
Deep Learning Models

- *LSTM-based Encoder Classifier (LbEC)*



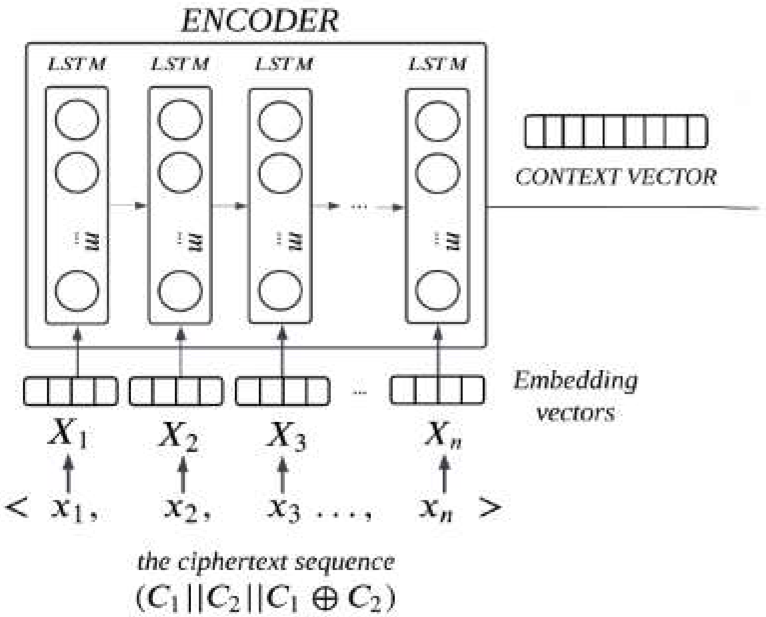
Deep Learning Models

- *LSTM-based Encoder Classifier (LbEC)*



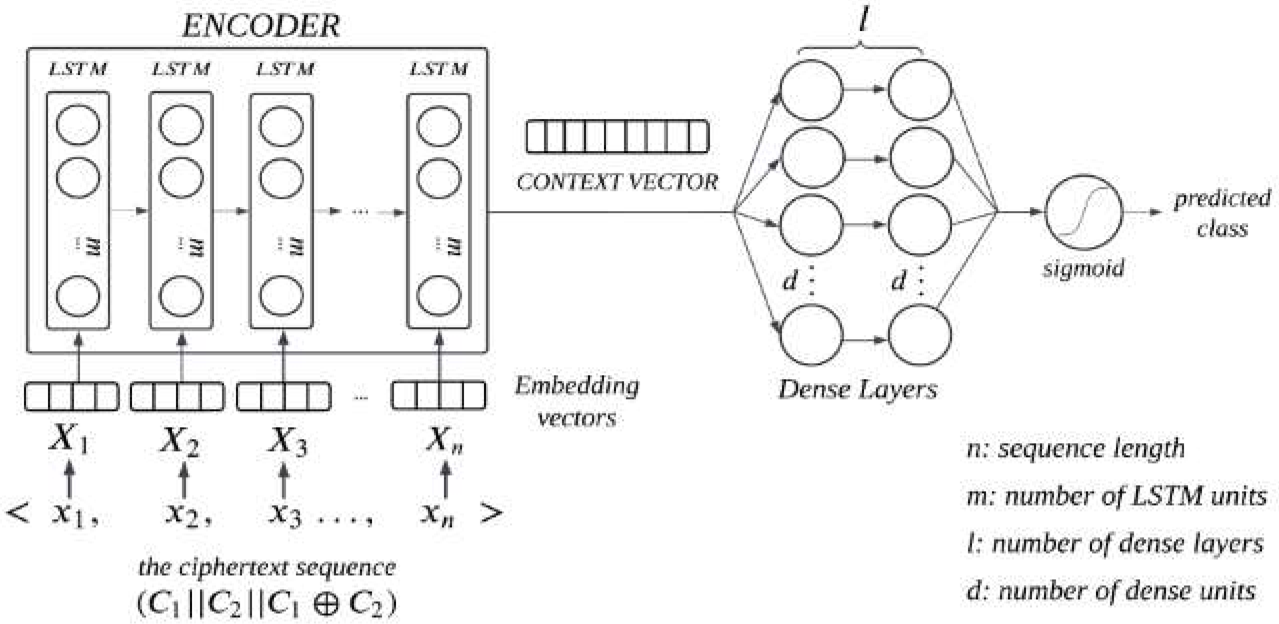
Deep Learning Models

- *LSTM-based Encoder Classifier (LbEC)*



Deep Learning Models

- *LSTM-based Encoder Classifier (LbEC)*

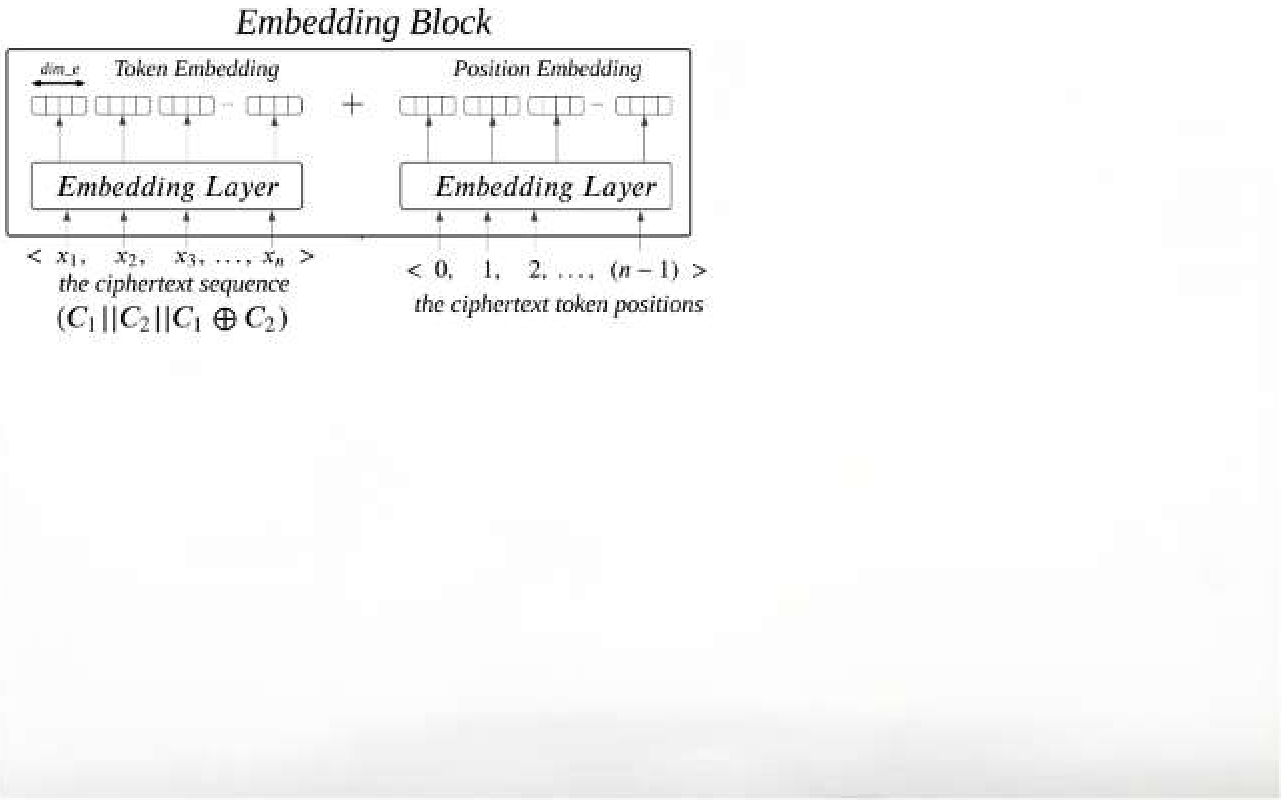


Deep Learning Models

- *Transformer based Encoder-only Classifier (TbEC)*

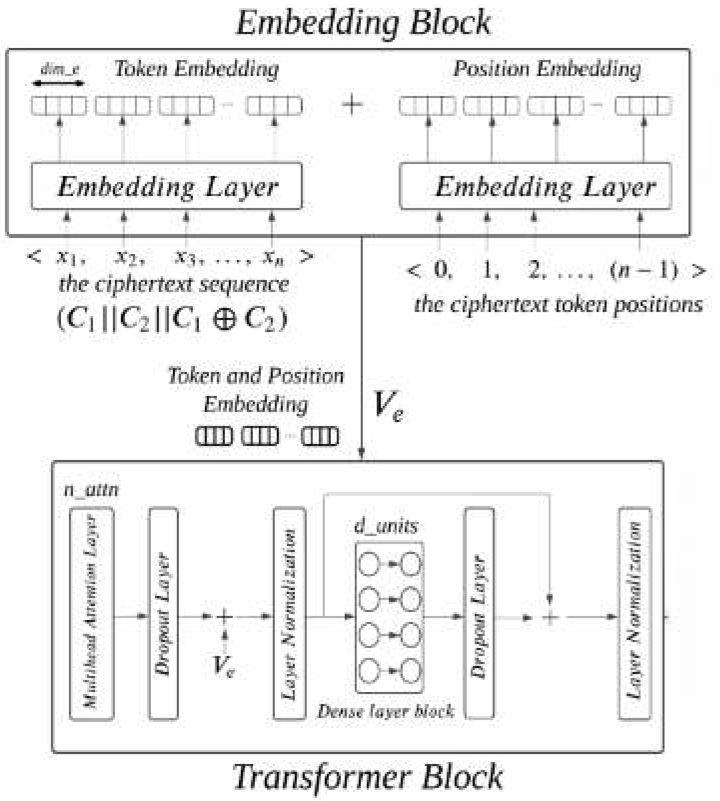
Deep Learning Models

- *Transformer based Encoder-only Classifier (TbEC)*



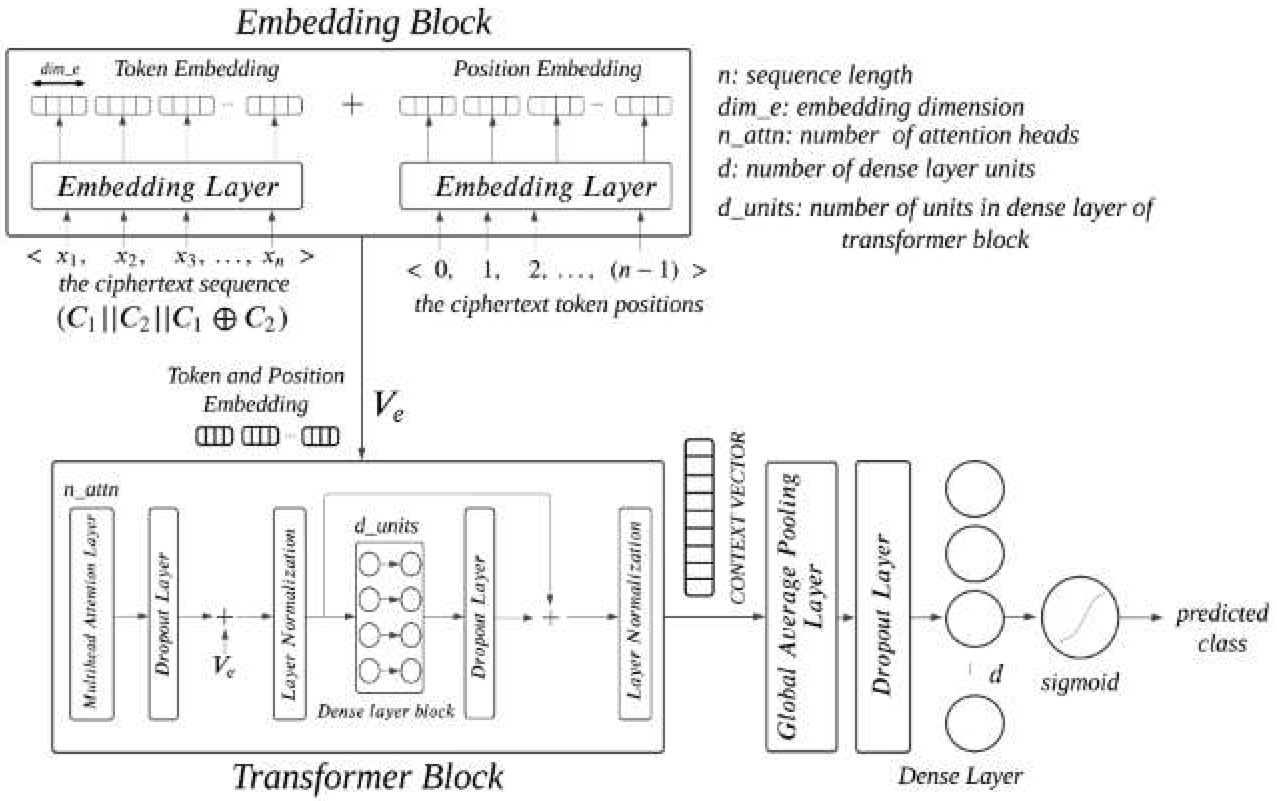
Deep Learning Models

- *Transformer based Encoder-only Classifier (TbEC)*



Deep Learning Models

- *Transformer based Encoder-only Classifier (TbEC)*



Searching the Distinguisher

The Algorithm used for Searching the Distinguisher

Algorithm 2 Distinguisher Search

Input: Dataset, $\mathcal{D}$

Output: Distinguisher found/ not found

$\mathcal{D}_{train}, \mathcal{D}_{val}, \mathcal{D}_{test} \leftarrow SPLIT\_DATASET(\mathcal{D})$ ▷ split the dataset, $\mathcal{D}$ into train, validation & test sets
$model \leftarrow DEFINE\_MODEL()$ ▷ calls the function definition of the model
$Accuracy_{(train)} \leftarrow TRAIN(model, \mathcal{D}_{train}, \mathcal{D}_{val})$ ▷ train the $model$ on D_{train} & D_{val}
if $Accuracy_{(train)} > 50\%$ **then**
    $Accuracy_{(test)} \leftarrow TEST(model, \mathcal{D}_{test})$ ▷ test the $model$ on D_{test}
    if $Accuracy_{(test)} > 50\%$ **then**
        return "Distinguisher Found"
    else
        return "Distinguisher Not Found"
    end if
else
    return "Distinguisher Not Found"
end if

Outline

- Introduction
- A Brief History
- Deep Learning for Sequence Classification
- Neural Distinguishers based on Sequence Classification
- Results and Observation
- Conclusion

Brief Overview of the Ciphers

HIGHT

- *ARX* based, Generalized Fiestel Structure
- block size → 64 bits
- key size → 128 bits
- number of rounds → 32

PRESENT

- *SPN* based
- block size → 64 bits
- key size → 80 bits
- number of rounds → 31

LEA

- *ARX*-based
- block size → 128 bits
- key size → 128 bits
- number of rounds → 24

SPARX

- *ARX* based
- block size → 64 bits
- key size → 128 bits
- number of rounds → 24

Piccolo-80

- *GFN* based
- block size → 64 bits
- key size → 80 bits
- number of rounds → 25

Obtained Results

HIGHT

- $LbEC \to 16$ rounds distinguisher (Accuracy $= 50.02\%$)
- $TbEC \to 11$ rounds distinguisher (Accuracy $= 50.18\%$)

PRESENT

- $LbEC \to 12$ rounds distinguisher (Accuracy $= 50.14\%$)
- $TbEC \to 12$ rounds distinguisher (Accuracy $= 50.01\%$)

LEA

- $LbEC \to 11$ rounds distinguisher (Accuracy $= 50.15\%$)
- $TbEC \to 13$ rounds distinguisher (Accuracy $= 50.12\%$)

SPARX

- $LbEC \to 6$ rounds distinguisher (Accuracy $= 50.62\%$)
- $TbEC \to 6$ rounds distinguisher (Accuracy $= 50.18\%$)

Piccolo-80

- $LbEC \to 6$ rounds distinguisher (Accuracy $= 50.18\%$)
- $TbEC \to 9$ rounds distinguisher (Accuracy $= 50.05\%$)

Comparisons of the Result

Block cipher	Model Architecture	Rounds	Test Accuracy
HIGHT	LGBM[1]	10	50.01%
	CNN[1]	10	50.30%
	LSTM[1]	8	66.36%
	DBitNet[2]	10	75.10%
	LbEC	**16**	**50.02%**
	TbEC	**11**	**50.18%**
PRESENT	CNN based[3]	7	72.05%
	DBitNet[2]	9	50.90%
	LbEC	**12**	**50.14%**
	TbEC	**12**	**50.01%**
LEA	LGBM[1]	9	50.18%
	CNN[1]	9	50.45%
	LSTM[1]	8	59.81%
	DBitNet[2]	11	51.10%
	LbEC	**11**	**50.15%**
	TbEC	**13**	**50.12%**
SPARX	LGBM[1]	5	50.64%
	CNN[1]	5	50.38%
	LSTM[1]	5	50.45%
	LbEC	**6**	**50.62%**
	TbEC	**6**	**50.18%**
Piccolo-80	**LbEC**	**6**	**50.18%**
	TbEC	**9**	**50.05%**

[1] D. Pal, U. Mandal, M. Chaudhury, A. Das, and D. R. Chowdhury, "A deep neural differential distinguisher for ARX based block cipher." Cryptology ePrint Archive, Paper 2022/1195, 2022.

[2] E. Bellini, D. G erault, A. Hambitzer, and M. Rossi, "A cipher-agnostic neural training pipeline with automated finding of good input differences," IACR Trans. Symmetric Cryptol., vol. 2023, no. 3, pp. 184–212, 2023.

[3] L. Zhang and Z. Wang, "Improving differential-neural distinguisher model for des, chaskey, and present," 2022.

Outline

- Introduction
- A Brief History
- Deep Learning for Sequence Classification
- Neural Distinguishers based on Sequence Classification
- Results and Observation
- Conclusion

Conclusion

- the problem of building differential distinguishers aided by deep learning is observed from a unique standpoint
- embedding vectors have represented the sequence tokens
- the embedding vectors representing the ciphertext sequences have been used for the classification tasks
- one hot encoding $\rightarrow LbEC$
- the model learns the embedding vectors during the training phase$\rightarrow TbEC$
- this is a generalized approach $\rightarrow$ the approach is clearly not specific to any particular design
- future scope $\rightarrow$ further improvements and extension of the method to other ciphers
- perform a key recovery with the help of the proposed distinguishers

THANK YOU