gMBA: Expression Semantic Guided Mixed Boolean-Arithmetic Deobfuscation Using Transformer Architectures

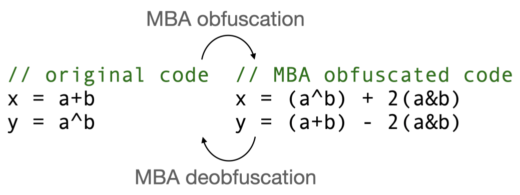
Youjeong Noh, Joon-Young Paik, Jingun Kwon*, Eun-Sun Cho*
Chungnam National University, Daejeon, Republic of Korea
`youjoeng.noh14@o.cnu.ac.kr, {lucadi, jingun.kwon, eschough}@cnu.ac.kr`

Abstract

Mixed Boolean-Arithmetic (MBA) obfuscation protects intellectual property by converting programs into forms that are more complex to analyze. However, MBA has been increasingly exploited by malware developers to evade detection and cause significant real-world problems. Traditional MBA deobfuscation methods often consider these expressions as part of a black box and overlook their internal semantic information. To bridge this gap, we propose a truth table, which is an automatically constructed semantic representation of an expression's behavior that does not rely on external resources. The truth table is a mathematical form that represents the output of expression for all possible combinations of input. We also propose a general and extensible guided MBA deobfuscation framework (gMBA) that modifies a Transformer-based neural encoder-decoder Seq2Seq architecture to incorporate this semantic guidance. Experimental results and in-depth analysis show that integrating expression semantics significantly improves performance and highlights the importance of internal semantic expressions in recovering obfuscated code to its original form. [1]

1 Introduction

Mixed Boolean-Arithmetic (MBA) obfuscation protects intellectual property by transforming code into complex forms that are difficult to analyze (Nagra and Collberg, 2009; Collberg et al., 2012; Ceccato, 2014; Bardin et al., 2017). It combines Boolean (AND, OR, XOR) and arithmetic (+, -, ×) operations to create exponentially more complex expressions (Zhou et al., 2007), making reverse engineering harder (Mohajeri Moghaddam et al., 2012). However, these techniques are increasingly exploited by malware developers to evade detection. By obfuscating internal code, malware can bypass



```
// original code   // MBA obfuscated code
x = a+b            x = (a^b) + 2(a&b)
y = a^b            y = (a+b) - 2(a&b)
```

Figure 1: MBA obfuscation and deobfuscation transform mathematically equivalent expressions into different code representations.

security analysis tools (Cohen, 1987; Kaur et al., 2024). For example, the DarkSide ransomware utilized obfuscation to evade detection (FBI National Press Office, 2021; Trend Micro Research, 2021), highlighting the need for advanced deobfuscation techniques. Figure 1 shows an example of MBA obfuscation and deobfuscation.

Traditional MBA deobfuscation methods include bitblast (Adrien Guinet and Videau, 2017; Liu et al., 2021), pattern matching (Eyrolles et al., 2016; Reichenwallner and Meerwald-Stadler, 2022), and program synthesis (Schloegel et al., 2022; Lee and Lee, 2023). Recently, Seq2Seq models using RNNs (Rumelhart et al., 1986) and Transformers (Vaswani et al., 2023) have been introduced (Feng et al., 2020a), but they consider these expressions as part of a black box and overlook their internal semantic information, leading to lower accuracy. To address this, we propose using *truth tables* as a structured semantic representation. We introduce two types: Boolean truth tables (logical equivalence) and extended truth tables (computational equivalence). These tables are automatically constructed to represent all possible input-output mappings of an expression.

We present gMBA, a guided MBA deobfuscation framework that integrates semantic information to improve deobfuscation performance. Using a Transformer-based Seq2Seq model, our approach integrates truth tables to enhance internal represen-

*Corresponding author
[1]Our code is available at https://github.com/Masterwhiece/gMBA.

tation learning. Specifically, we encode both the obfuscated source code and the truth tables into vector space representations and merge them to inject into the decoder architecture. To the best of our knowledge, this is the first approach leveraging semantic representations for MBA deobfuscation.

Comparison between our gMBA and previous methods on NeuReduce dataset (Feng et al., 2020a) showed that our method outperforms them for both accuracy for exact matching and BLEU metrics.

2 Boolean and Extended Truth Tables

MBA deobfuscation is NP-hard, meaning that no general deterministic algorithm can solve it efficiently (Zhou et al., 2007). Previous approaches have treated MBA obfuscation as a black box without understanding its mechanism. While Seq2Seq models have been applied, they rely solely on syntactic patterns. To incorporate semantic information, we introduce truth tables, representing an expression's output for all input combinations, encapsulating its semantics.

Truth Table Extraction for MBA Expressions Given Boolean variables $x_1, x_2, \dots, x_n$ where each x_i takes values from $\{0, 1\}$, we define a Boolean function derived from an MBA expression:

$$f : \{0,1\}^n \to \mathbb{Z} \tag{1}$$

which maps each possible input combination to an integer output. The truth table representation of f is constructed by evaluating $f(x_1, \dots, x_n)$ for all 2^n possible input assignments.

For a given function $f(x_1, \dots, x_n)$, we define the truth table as a vector $\mathbf{T}_f$ of length 2^n, where each entry corresponds to the function value for a specific binary input configuration:

$$\mathbf{T}_f = \begin{bmatrix} f(0,0,\dots,0) \\ f(0,0,\dots,1) \\ \vdots \\ f(1,1,\dots,1) \end{bmatrix}. \tag{2}$$

Each row of the truth table corresponds to a binary tuple $(x_1, \dots, x_n)$ sorted in lexicographical order, and the resulting vector uniquely represents the function's output across all possible inputs. For an original expression $f(x_1, \dots, x_n)$ and an MBA-obfuscated expression $g(x_1, \dots, x_n)$, we compute their truth table vectors $\mathbf{T}_f$ and $\mathbf{T}_g$, respectively.

If $\mathbf{T}_f = \mathbf{T}_g$, then f and g are functionally equivalent despite their syntactic differences. This property enables MBA deobfuscation by reducing complex expressions to their canonical form.

3 Methods

In this study, truth tables serve as features to enhance the model's understanding of Mixed Boolean-Arithmetic (MBA) expressions. To properly ground the model's reasoning in both syntactic and semantic evidence, we introduce a feature fusion mechanism that concatenates the encoder's latent embedding—capturing the syntactic patterns of obfuscated expressions—with a vectorized truth table representation that preserves raw semantic evaluation values.

In this section, we describe gMBA, where both the obfuscation code and a semantic representation of the truth table serve as inputs.

3.1 Sequence-to-sequence Architecture

Encoder The encoder processes the input sequence $\mathbf{x}$ and generates latent representations $\mathbf{H}_L$ through stacked layers.

$$\begin{aligned} \mathbf{H}_0 &= \mathbf{E}_{\text{input}} \\ \mathbf{H}_l &= \text{LN}(\mathbf{H}_{l-1} + \text{MultiHeadAttn}(\mathbf{H}_{l-1})) \\ \mathbf{H}_l &= \text{LN}(\mathbf{H}_l + \text{FFN}(\mathbf{H}_l)) \end{aligned} \tag{3}$$

where LN and FFN refers to layer normalization and feedforward networks, respectively. After passing through L layers, the final encoder output $\mathbf{H}_L$ is obtained.

Concatenation with Truth Table To integrate semantics the truth table vector $\mathbf{T}$ is projected and concatenated with the encoder output:

$$\begin{aligned} \mathbf{T} &= \text{Unsqueeze}(\text{Linear}(\mathbf{T})) \\ \mathbf{H}_{\text{final}} &= \text{Concat}(\mathbf{H}_L, \mathbf{T}) \end{aligned} \tag{4}$$

where Linear$(\cdot)$ projects $\mathbf{T}$ into the embedding space. The concatenation position varies, and a <sep> token is added when explicitly separating syntax and semantics.

Decoder The decoder generates the output sequence $\mathbf{y}$ based on $\mathbf{H}_{\text{final}}$.

$$\begin{aligned} \mathbf{Y}_0 &= \text{Embed}(\mathbf{y}) + \text{PosEnc}(\mathbf{y}) \\ \mathbf{Y}_l &= \text{LN}(\mathbf{Y}_{l-1} + \text{MaskedMultiHeadAttn}(\mathbf{Y}_{l-1})) \\ \mathbf{Y}_l &= \text{LN}(\mathbf{Y}_l + \text{MultiHeadAttn}(\mathbf{Y}_l, \mathbf{H}_{\text{final}})) \\ \mathbf{Y}_l &= \text{LN}(\mathbf{Y} + \text{FFN}(\mathbf{Y})) \end{aligned} \tag{5}$$

Output Probability Calculation The final decoder output is mapped to a probability distribution

$$\mathbf{y}_{\text{pred}} = \text{Softmax}(\text{Linear}(\mathbf{Y}_L)) \tag{6}$$

where Linear$(\cdot)$ maps the decoder output to the vocabulary space, and Softmax$(\cdot)$ generates token probabilities.

3.2 Integration of Truth Table Information

Truth Tables as Semantics of MBA Expressions To integrate semantic information, two types of truth tables were generated: Boolean truth tables (bool tt.) and extended truth tables (extended tt.).

The Boolean truth table evaluates MBA expressions for all input combinations, ensuring logical equivalence. In the function representation (Section 2), the range of f is $0, 1$, capturing the semantic relationship between obfuscated (src) and simplified (trg) expressions. This establishes a solid foundation for understanding logical behavior, independent of syntactic complexity.

On the other hand, the extended truth table encodes extended operations in MBA expressions, evaluating numeric results across input combinations to preserve computational semantics. In the function representation (Section 2), the range is $\mathbb{Z} \bmod m$, where $m > 2$. While the Boolean truth table captures binary equivalence, the extended truth table reflects operational depth and complexity, providing a complementary perspective.

Table 1 shows the Boolean truth table and extended truth table for $(a \wedge b) + 2(a\&b)$, yielding different results.

Boolean Truth Table					
a, b	$a + b$	$a \oplus b$	$a\&b$	$2(a\&b)$	$(a \oplus b) + 2(a\&b)$
0, 0	0	0	0	0	0
0, 1	1	1	0	0	1
1, 0	1	1	0	0	1
1, 1	0	0	1	0	0
Extended Truth Table					
a, b	$a + b$	$a \oplus b$	$a\&b$	$2(a\&b)$	$(a \oplus b) + 2(a\&b)$
0, 0	0	0	0	0	0
0, 1	1	1	0	0	1
1, 0	1	1	0	0	1
1, 1	2	0	1	2	2

Table 1: Example of boolean truth table (upper) and extended truth table (lower) of $(a \oplus b) + 2(a\&b)$

Integration Strategy We explore three methods to integrate the encoder output (H) with the truth table (T): addition, token-level concatenation, and hidden-dimension concatenation. These strategies are defined in Equation 7.

$$\text{Integrate}(H, T) = \begin{cases} H + T & \textit{add} \\ [H; T] & \textit{tok-level cat} \\ H \oplus T & \textit{hid-dim cat} \end{cases} \tag{7}$$

Addition: The truth table (T) is repeated along the token dimension and added element-wise to the encoder output (H), preserving its shape $B \times T \times D$. For clarity, $\mathbf{T}$ is broadcast (repeated) across the token dimension, ensuring it matches the shape of $\mathbf{H}$ before the element-wise addition.

$$H'_{i,j,k} = H_{i,j,k} + T_{i,k} \tag{8}$$

Token-level Concatenation: The truth table (T) is reshaped to $B \times 1 \times D$ and appended along the token dimension, resulting in $H' \in \mathbb{R}^{B \times (T+1) \times D}$.

$$H' = [H; T] \tag{9}$$

Hidden-dimension Concatenation: T is repeated along the token dimension and concatenated along the hidden dimension, forming $H' \in \mathbb{R}^{B \times T \times (D + D')}$.

$$H' = H \oplus T \tag{10}$$

Concatenation Strategy To optimize the concatenation strategy, we experimented with various configurations, focusing on token-level concatenation, which effectively incorporates truth table information. The tested strategies, with key variables are summarized as follows: (1) *Concat Position*: the order in which semantic (truth table) and syntactic (encoder output) information are combined, (2) *Seperator Token*: whether a special separator token is inserted to distinguish between syntax and semantics, and (3) *Semantics*: the type of semantic information used, including Boolean truth table, extended truth table, or both.

3.3 Overall Architecture

Figure 2 illustrates our model architecture, which extends the standard Transformer by integrating a truth table vector to enhance semantic understanding of mathematical expressions. The encoder follows the standard multi-head self-attention and feed-forward layers, while the truth table vector is processed through a linear transformation and concatenated with the encoder representation. This augmentation provides explicit semantic guidance to the model.

The decoder incorporates masked self-attention and attends to both the encoder output and the augmented representation, enabling it to better differentiate semantically similar expressions. This architecture improves the model's ability to recover precise mathematical expressions during deobfuscation.

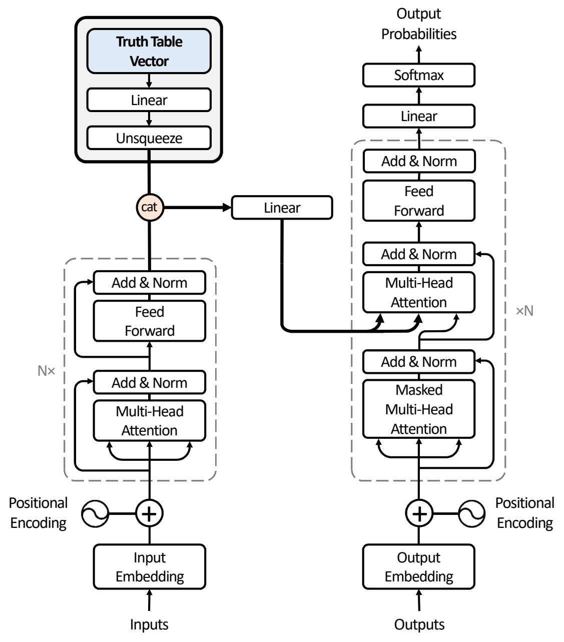


Figure 2: Model Architecture

4 Experiments

4.1 Experimental Settings

Dataset We used NeuReduce dataset for our experiments, using the same settings as in previous work (Feng et al., 2020b). Based on standard split, we divided the dataset into 80k, 20k, and 10k instances for training, validation, and test datasets, respectively. Appendix A provides further statistics.

Implementation Details The model's performance was evaluated using two metrics: exact matching accuracy and the BLEU score for n-gram matching. We used greedy decoding in all experiments. We trained the model with hidden dimension size 256 and max length 104. The baseline model is a vanilla transformer network used in the previous work that does not leverage truth tables.

4.2 Results

The experimental results on NeuReduce dataset are shown in Table 2. As shown in the table, gMBA consistently achieves higher accuracy and BLEU scores compared to the Vanilla model across all the configurations. This improvement highlights the advantage of leveraging the semantics of MBA expressions, enabling the correct deobfuscation of MBA expressions with similar syntax but different evaluation results. Specifically, gMBA outperforms the Vanilla model, achieving an accuracy improvement of at least 21.72% and up to 51.94%, along with a BLEU score increase of at least 8.78 and up to 17.49 points; the extended truth table with the <sep> token, following syntactic information, yields the highest accuracy of 92.78%.

Model	Concat Position	<sep>	Acc (%)	BLEU (%)
Vanilla	-	-	40.84	78.05
bool tt.	back	N	65.05	87.84
		Y	65.17	87.28
	front	N	66.01	87.86
		Y	63.91	87.17
	back, front of <pad>	N	62.56	86.83
		Y	65.99	87.77
extended tt.	back	N	91.69	94.96
		Y	92.59	95.39
	front	N	92.06	94.99
		Y	87.69	93.97
	back, front of <pad>	N	91.71	95.00
		Y	**92.78**	**95.53**
both	back	N	90.01	95.14
		Y	88.83	94.83
	front	N	91.01	95.48
		Y	89.19	94.98
	back, front of <pad>	N	91.57	95.51
		Y	88.03	94.49

Table 2: Impact of different truth table integration strategies on deobfuscation performance. Vanilla refers to the baseline (Feng et al., 2020b) Transformer model without truth table information.

4.3 Analysis

Impact of Truth Tables on Accuracy and Semantic Understanding In Table 2, we observe that the vanilla model achieves a relatively high BLEU score (78%) but a much lower exact-match accuracy (40%). This discrepancy suggests that while the model can generate expressions that look similar to the reference, it often fails to produce the exact target expression. In contrast, when we introduce truth table strategies, both BLEU and accuracy improve, and—crucially—the gap between these two metrics narrows. We interpret this as evidence that the added semantic information (via truth tables) helps the model better distinguish among closely related expressions. Moving from the vanilla (no semantic) setup to Boolean truth tables and then arithmetic truth tables, the model gains deeper semantic understanding, allowing it to shift from superficially similar expressions to precisely aligned original expressions. Furthermore, these results highlight the potential of neural models to learn and reason over the semantics of mathematical expressions, capturing underlying structures beyond mere syntactic similarity.

Effectiveness of the Extended Truth Tables In our observations, one of the key challenges in MBA deobfuscation lies in accurately generating the constants and coefficients within an expression. While Boolean truth tables provide only binary outcomes, the extended truth table encodes richer numerical information—specifically, multi-bit evaluation results—which can assist the model in identifying and reasoning about underlying arithmetic patterns.

This additional numerical signal enables the model to more effectively infer structural and semantic properties of the expression, including magnitude and relational cues that are not readily available from binary outputs alone. We believe that the intermediate numerical guidance provided by the extended truth table significantly enhances the learning process, ultimately leading to better deobfuscation performance.

Comparison with Fine-Tuned PLMs and LLMs One potential concern is whether pre-trained language models and large language models, with general mathematical and logical knowledge, can outperform specialized models on MBA tasks.

We additionally conducted experiments and the results are shown in Table 3. For comparison, we selected BART[2] (Lewis, 2019) and T5[3] (Raffel et al., 2020) as encoder-decoder architectured PLMs and recent LLaMA models[4] for LLMs. We fine-tuned and evaluated them with the same data described in 4.1.

Model	Metrics (%)	
	Acc	BLEU
NeuReduce [5]	40.84	78.05
LoRA Fine-tuned BART	47.96	81.91
Fine-tuned BART	52.12	74.36
LoRA Fine-tuned T5	50.90	82.30
Fine-tuned T5	54.50	83.91
LoRA Fine-tuned LLaMA 3.2 3B	37.29	75.13
LoRA Fine-tuned LLaMA 3.1 8B	38.56	79.17
gMBA (*proposed*)	**92.78**	**95.53**

Table 3: Performance comparison of fine-tuned PLMs, instruction-tuned LLMs and the proposed method on MBA deobfuscation using the dataset of NeuReduce

[2]https://huggingface.co/docs/transformers/ko/model_doc/bart

[3]https://huggingface.co/docs/transformers/model_doc/t5

[4]https://huggingface.co/meta-llama

Table 3's findings suggest that the prior mathematical and logical knowledge embedded in PLMs alone is insufficient for effective MBA deobfuscation, highlighting the necessity of specialized architectures that explicitly capture the semantics of MBA expressions. It also shows the limitation of the LLMs and we anlayzed this problem is attributed to the pretraining objective of LLMs, which optimizes for generating fluent natural language. The underlying token distribution is shaped by linguistic plausibility rather than the syntactic precision required for symbolic tasks. Consequently, even with instruction-tuning, these models struggle to meet the strict accuracy demands of MBA deobfuscation, unlike our proposed method, gMBA.

5 Conclusions

We showed that incorporating truth tables derived from semantic information is crucial for MBA deobfuscation. To incorporate truth tables, we proposed a gMBA that leverages a Transformer-based Seq2Seq framework. Our analysis showed that gMBA outperforms both traditional methods and a naive Transformer network, resulting in improving performance.

Limitations

In terms of deobfuscation accuracy, gMBA based on neural networks achieves an accuracy in the 90% range, whereas the state-of-the-art (SOTA) logic-based methods achieve 100% accuracy (Reichenwallner and Meerwald-Stadler, 2023). Despite this limitation, gMBA has potential for improvement and scalability through data augmentation, enhancing its ability to handle diverse obfuscation patterns, unlike highly accurate logic-based approaches that struggle with scalability due to the complexity of crafting rules.

Acknowledgements

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT)(RS-2024-00456953).
This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (RS-2022-00143911, AI Ex-

[5]The baseline was reimplemented based on (Feng et al., 2020b).

cellence Global Innovative Leader Education Program)

References

Ninon Eyrolles Adrien Guinet and Marion Videau. 2017. Obfuscation with mixed boolean-arithmetic expressions: reconstruction, analysis and simplification tools.

Sébastien Bardin, Robin David, and Jean-Yves Marion. 2017. Backward-bounded dse: targeting infeasibility questions on obfuscated codes. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 633–651. IEEE.

Mariano Ceccato. 2014. On the need for more human studies to assess software protection. In *Workshop on Continuously Upgradeable Software Security and Protection*, pages 55–56.

Fred Cohen. 1987. Computer viruses: theory and experiments. *Computers & security*, 6(1):22–35.

Christian Collberg, Sam Martin, Jonathan Myers, and Jasvir Nagra. 2012. Distributed application tamper detection via continuous software updates. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 319–328.

Ninon Eyrolles, Louis Goubin, and Marion Videau. 2016. Defeating mba-based obfuscation. In *Proceedings of the 2016 ACM Workshop on Software PROtection*, SPRO '16, page 27–38, New York, NY, USA. Association for Computing Machinery.

FBI National Press Office. 2021. Deputy Director Speaks at Press Conference on Colonial Pipeline Ransomware Attack.

Weijie Feng, Binbin Liu, Dongpeng Xu, Qilong Zheng, and Yun Xu. 2020a. NeuReduce: Reducing mixed Boolean-arithmetic expressions by recurrent neural network. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 635–644, Online. Association for Computational Linguistics.

Weijie Feng, Binbin Liu, Dongpeng Xu, Qilong Zheng, and Yun Xu. 2020b. Neureduce: Reducing mixed boolean-arithmetic expressions by recurrent neural network. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 635–644.

Harpreet Kaur, Dharani Sanjaiy SL, Tirtharaj Paul, Rohit Kumar Thakur, K Vijay Kumar Reddy, Jay Mahato, and Kaviti Naveen. 2024. Evolution of endpoint detection and response (edr) in cyber security: A comprehensive review. In *E3S Web of Conferences*, volume 556, page 01006. EDP Sciences.

Jaehyung Lee and Woosuk Lee. 2023. Simplifying mixed boolean-arithmetic obfuscation by program synthesis and term rewriting. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, CCS '23, page 2351–2365, New York, NY, USA. Association for Computing Machinery.

Mike Lewis. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Binbin Liu, Junfu Shen, Jiang Ming, Qilong Zheng, Jing Li, and Dongpeng Xu. 2021. {MBA-Blast}: Unveiling and simplifying mixed {Boolean-Arithmetic} obfuscation. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1701–1718.

Hooman Mohajeri Moghaddam, Baiyu Li, Mohammad Derakhshani, and Ian Goldberg. 2012. Skypemorph: protocol obfuscation for tor bridges. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, page 97–108, New York, NY, USA. Association for Computing Machinery.

Jasvir Nagra and Christian Collberg. 2009. *Surreptitious software: obfuscation, watermarking, and tamperproofing for software protection*. Pearson Education.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Benjamin Reichenwallner and Peter Meerwald-Stadler. 2022. Efficient deobfuscation of linear mixed boolean-arithmetic expressions. In *Proceedings of the 2022 ACM Workshop on Research on offensive and defensive techniques in the context of Man At The End (MATE) attacks*, pages 19–28.

Benjamin Reichenwallner and Peter Meerwald-Stadler. 2023. Simplification of General Mixed Boolean-Arithmetic Expressions: GAMBA. In *2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 427–438. ISSN: 2768-0657.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Moritz Schloegel, Tim Blazytko, Moritz Contag, Cornelius Aschermann, Julius Basler, Thorsten Holz, and Ali Abbasi. 2022. Loki: Hardening code obfuscation against automated attacks. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3055–3073.

Trend Micro Research. 2021. What We Know About Darkside Ransomware and the US Pipeline Attack.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention Is All You Need. *arXiv preprint*. ArXiv:1706.03762 [cs].

Yongxin Zhou, Alec Main, Yuan X Gu, and Harold Johnson. 2007. Information hiding in software with mixed boolean-arithmetic transforms. In *International Workshop on Information Security Applications*, pages 61–75. Springer.

A Dataset Statistics

Table 4 provides the detailed statistics of the dataset used in our experiments. The dataset consists of expressions with varying numbers of variables, operators, and lengths. The Train-small dataset contains 100K samples, while the Test set consists of 10K samples. The number of variables and operations follows a wide distribution, reflecting the complexity of mathematical expressions encountered during training and evaluation. This distribution ensures that the model is tested on expressions of varying difficulty, supporting a robust assessment of its generalization ability.

	Train-small		**Test**	
	src	trg	src	trg
Size	100K	100K	10K	10K
# of Vars	17.0 ± 15.0	4.5 ± 3.5	16.5 ± 14.5	4.5 ± 3.5
# of Ops	26.0 ± 23.0	6.0 ± 6.0	25.0 ± 23.0	6.0 ± 6.0
Length	54.0 ± 46.0	18.0 ± 17.0	52.0 ± 48.0	18.0 ± 17.0

Table 4: Statistics of the Dataset (excluding Train-large)

B Engineering

Integration Strat.	**Ver.**	**Metrics (%)** Acc	BLEU
Addition	-	64.29	87.69
Token-level Concat	v1	58.60	84.18
	v2	63.87	87.31
	v3	**65.05**	**87.84**
	v4	35.77	74.11
	v5	61.89	85.91
	v6	61.95	86.62
	v7	61.82	86.01
Hidden-dim Concat	v1	0.44	34.56
	v2	0.06	17.88
	v3	0.14	19.81
	v4	0.26	23.32
	v5	0.12	15.26
	v6	0.13	18.55
	v7	0.13	16.99

Table 5: Comparison of Accuracy and BLEU Metrics for Truth Table Integration Techniques to Assess Syntax-Semantics Fusion Methods in Transformer-based Deobfuscation

To maximize the performance of each integration strategy, we conducted extensive engineering experiments to determine the optimal placement and type of normalization layers. Using the Boolean truth table as a test case, we explored various configurations within the concatenation-based integration methods to identify settings that yield the best accuracy and BLEU scores. Table 5 summarizes the results, highlighting the effect of different normalization strategies. These insights helped in refining our final implementation, ensuring that each approach was optimized for its best possible performance.

When using the hidden-dimension concatenation strategy, we observed that the matrices encoding the expression's syntax tend to collapse, causing a marked drop in performance. Hence, we concluded that this issue cannot be overcome by engineering efforts alone.

C Analysis of Errors

```
trg1  : -2*(x|~y)-(~x)-1
pred1 : -2*(x|~y)-(~(~x)
trg2  : -3*(x&~y)-4*(~x&y)
pred2 : -3*(x&~)-4*(~x&y)
trg3  : -17*(x|y|z)
pred3 : -16*(x|y|z)
trg4  : x&y
pred4 : x-((((((((((
```

Figure 3: Samples of misprediction

In reviewing the model's mispredicted expressions, we identified several recurring error patterns that highlight the system's current limitations. In most cases, the answers started with the same subexpression but ended incorrectly. There were also many cases that had generated almost identical to the correct answer but with a slightly different constant or operator. We observed a few instances, wherein the model introduces extraneous characters or repeated segments that deviate from the expected structure.