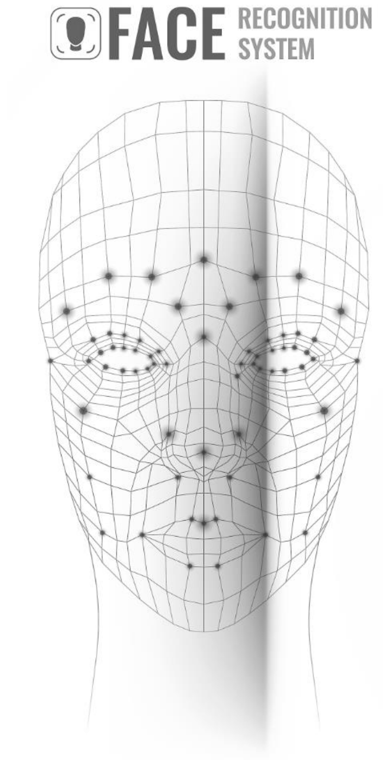
# o-glassesX: Compiler Provenance Recovery with Attention Mechanism from a Short Code Fragment

Yuhei Otsubo*†, Akira Otsuka†, Mamoru Mimura‡†,
Takeshi Sakaki§, and Hiroshi Ukegawa*

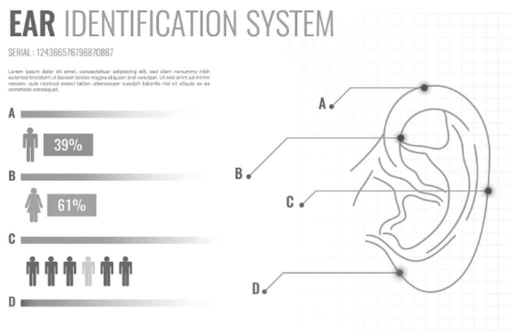
*National Police Agency, Tokyo, Japan
†Institute of information Security, Kanagawa, Japan
‡National Defense Academy, Kanagawa, Japan
§The University of Tokyo, Tokyo, Japan

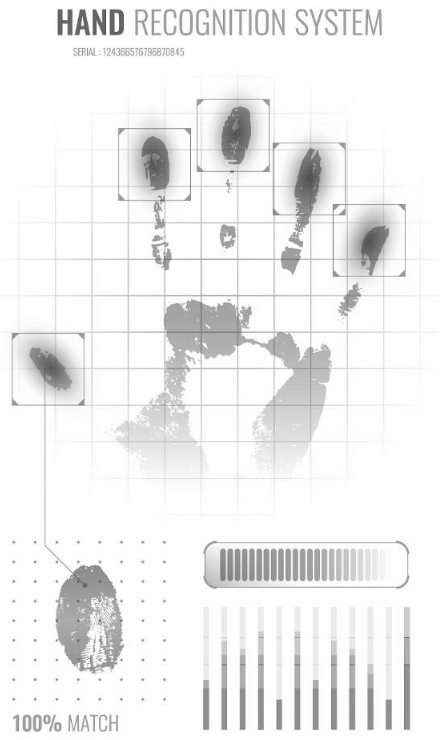
# Introduction

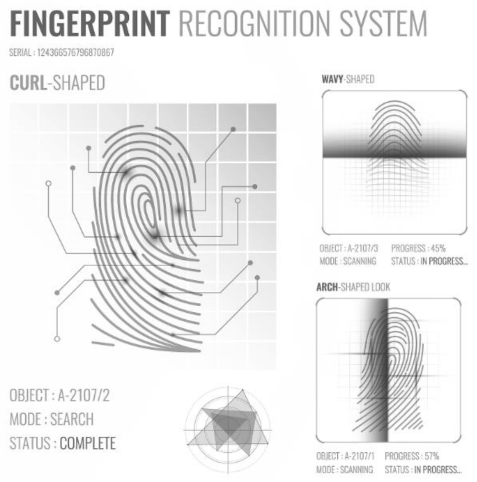
# Forensic Scientists

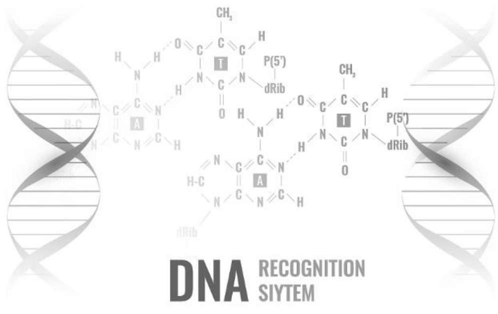


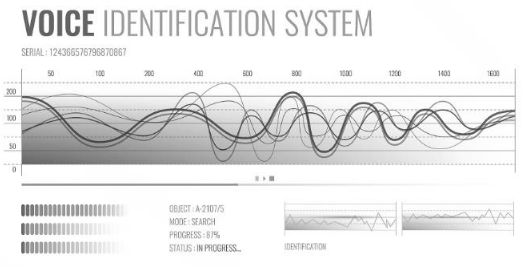




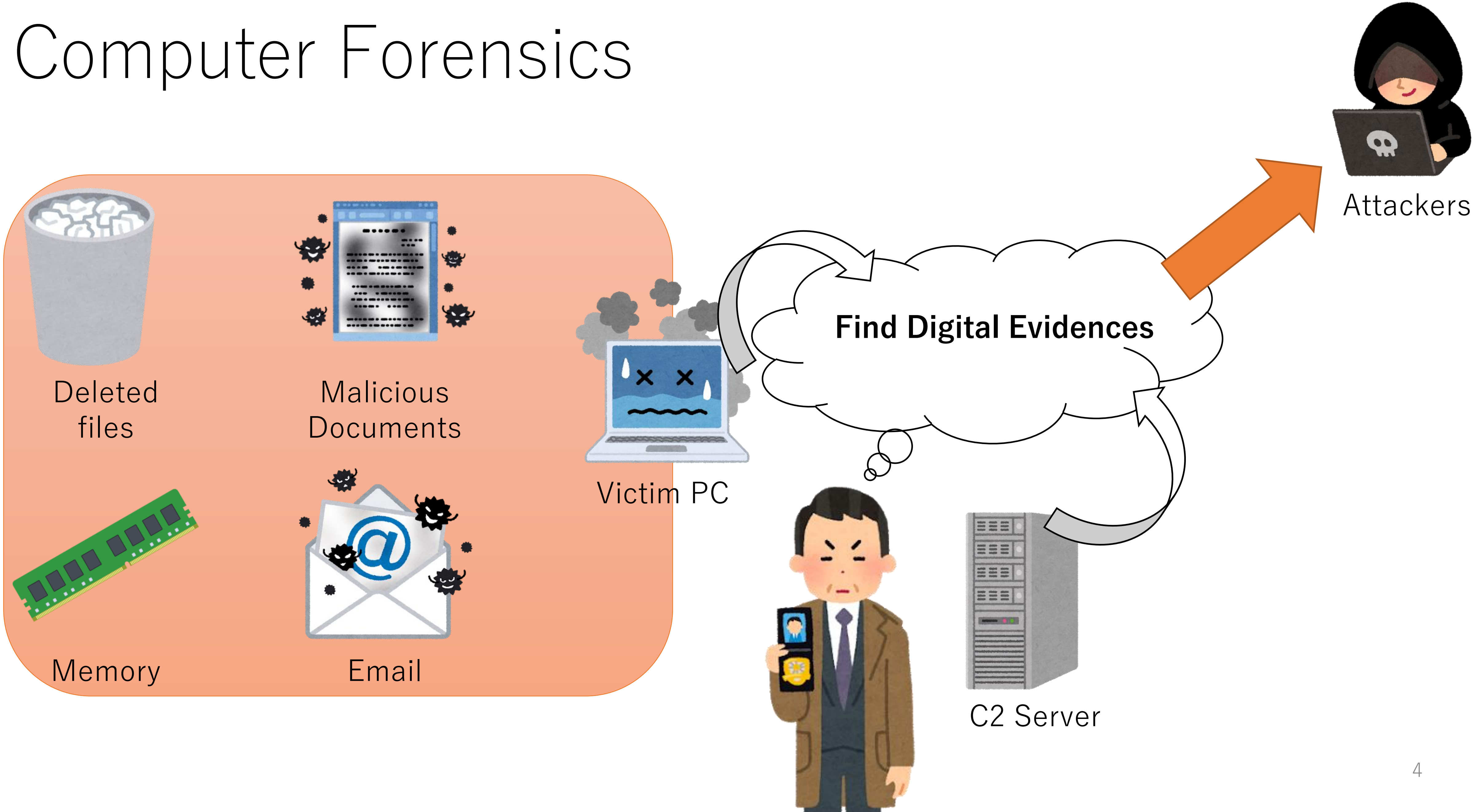




Designed by macrovector / Freepik (http://www.freepik.com)

# Computer Forensics

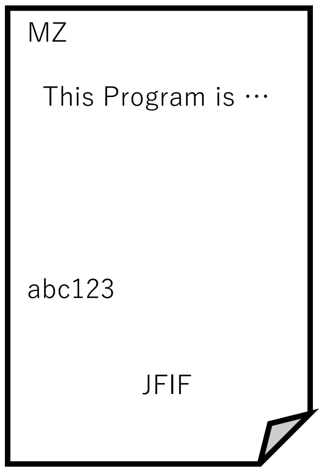


# Author Identification

## Compiler Provenance

Compiler
Family
Version
Optimization level

Malicious EXE




Specific Strings

Secret key

Resources

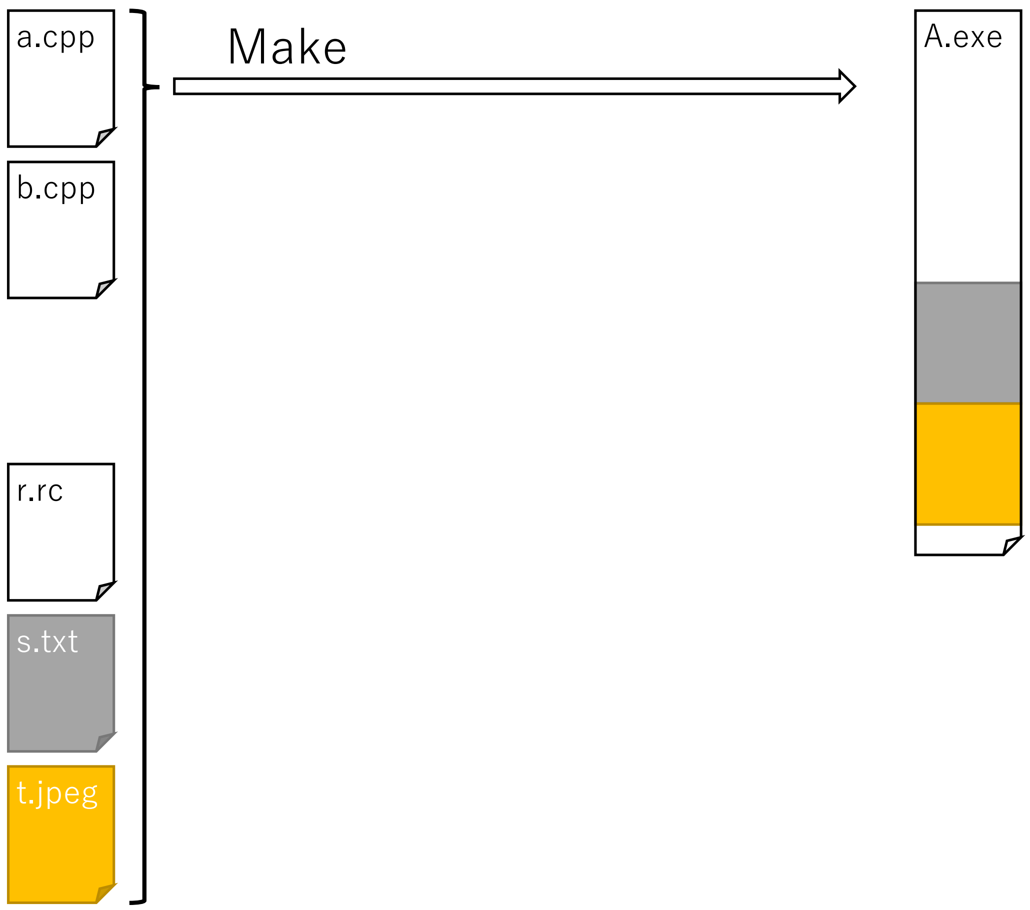
Language

API name

Compile time

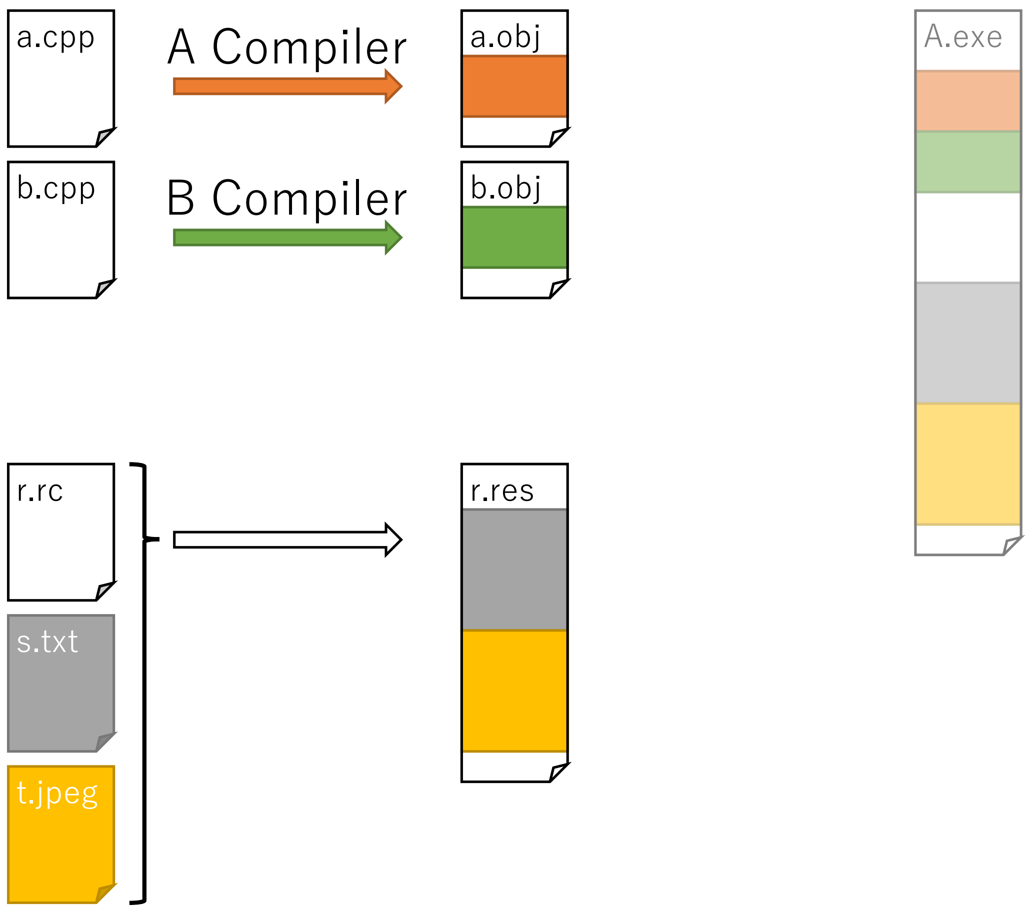
Static Link Libraries

# Multiple Compiler Binary




**What is the truth label of A.exe?**

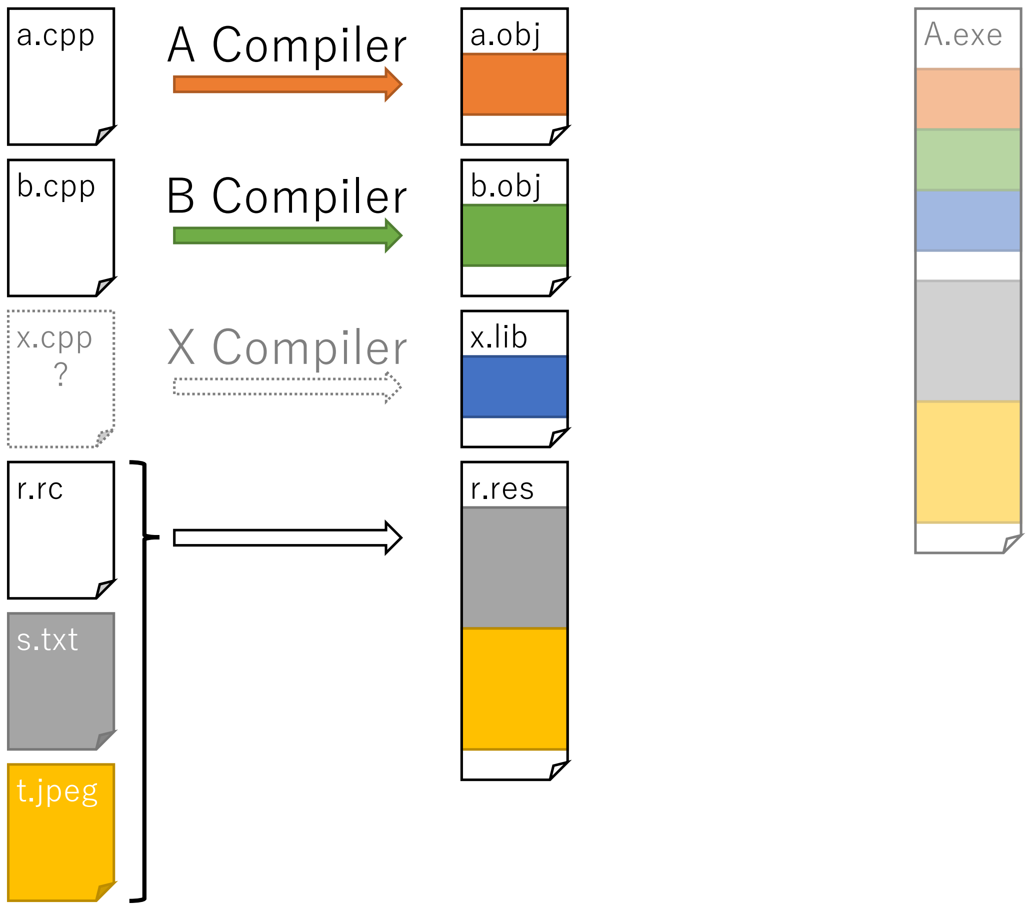
# Multiple Compiler Binary




**What is the truth label of A.exe?**

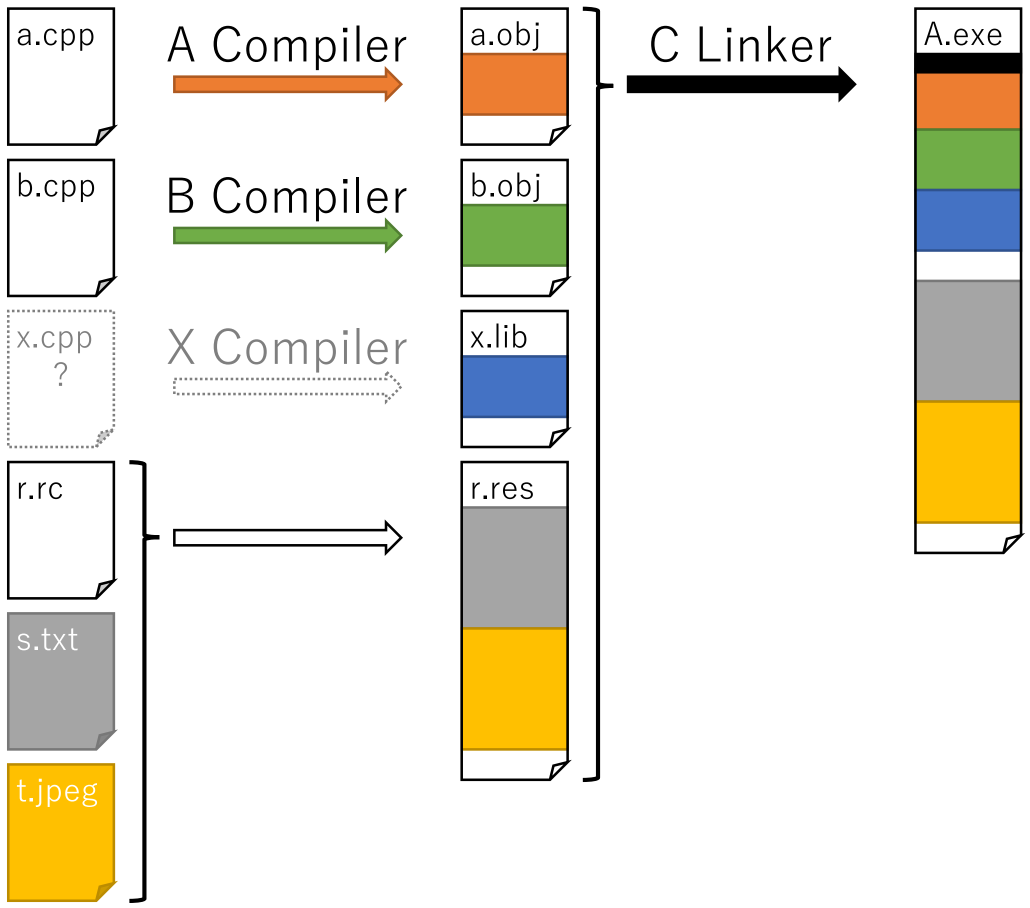
- **A Compiler?**
- **B Compiler?**

# Multiple Compiler Binary



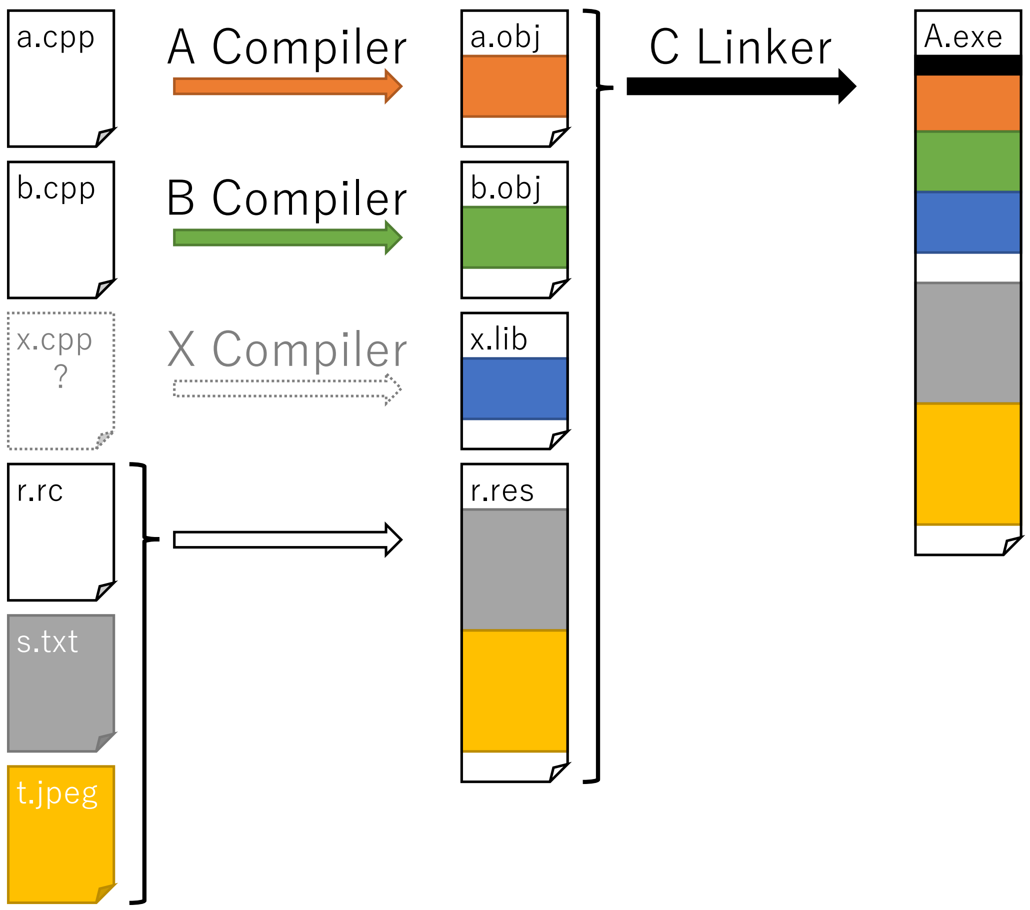

**What is the truth label of A.exe?**
- **A Compiler?**
- **B Compiler?**
- **X Compiler?**

# Multiple Compiler Binary




**What is the truth label of A.exe?**
- **A Compiler?**
- **B Compiler?**
- **X Compiler?**
- **C Linker?**

# Multiple Compiler Binary



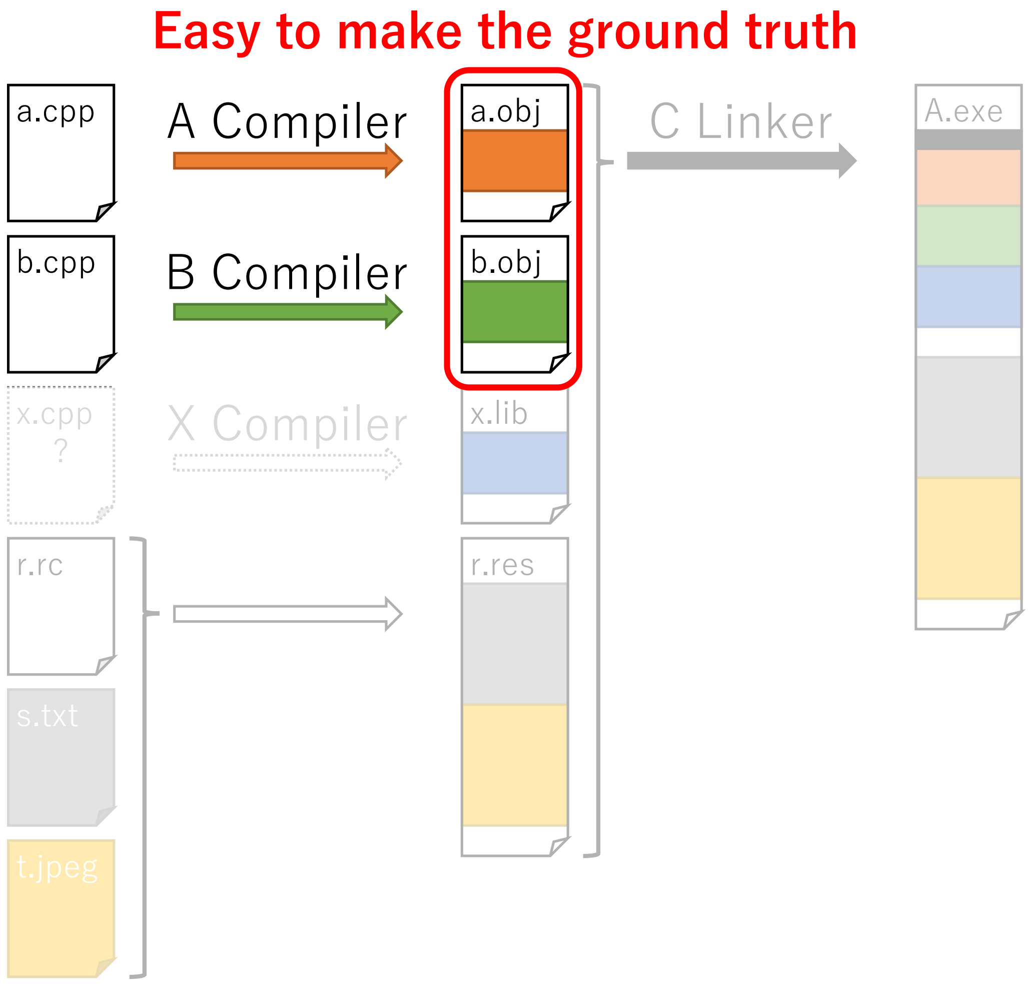

**What is the truth label of A.exe?**
- **A Compiler?**
- **B Compiler?**
- **X Compiler?**
- **C Linker?**

**What is the truth label of a.obj?**
- **A Compiler**

**What is the truth label of b.obj?**
- **B Compiler**

**What is the truth label of x.lib?**
- **Hmm… I think VC, because MS provide it!**

# Multiple Compiler Binary




What is the truth label of A.exe?
- A Compiler?
- B Compiler?
- X Compiler?
- C Linker?

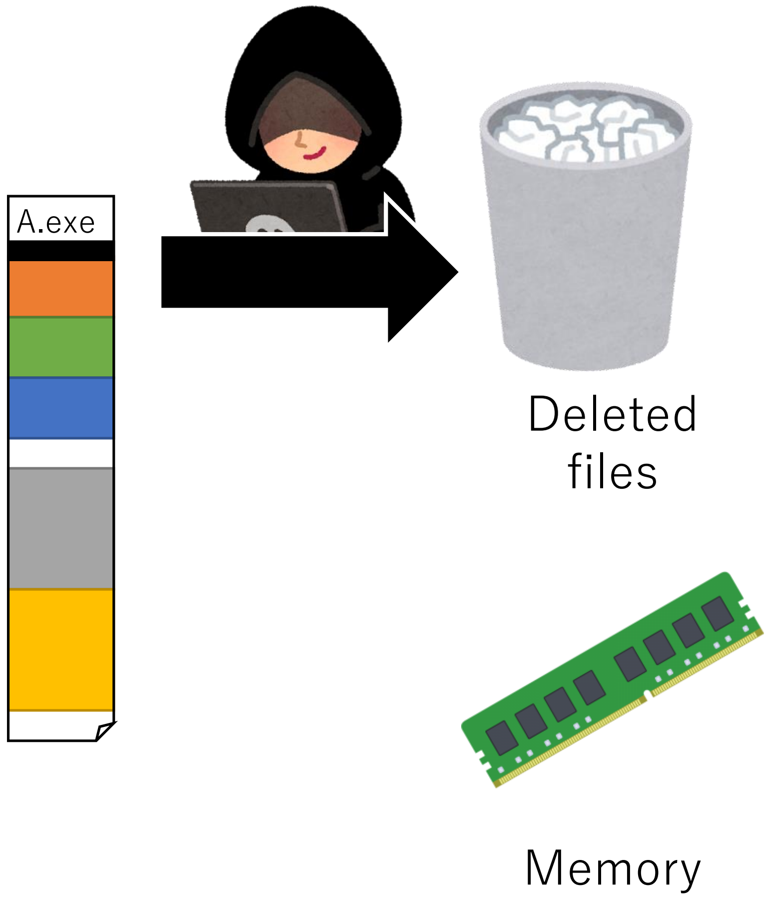
**What is the truth label of a.obj?**
- **A Compiler**

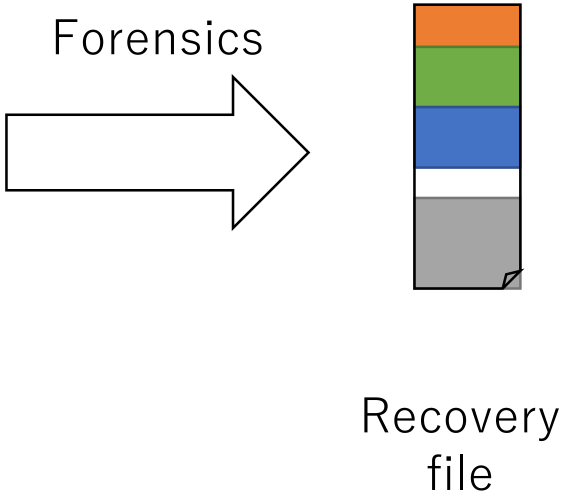
**What is the truth label of b.obj?**
- **B Compiler**

What is the truth label of x.lib?
- Hmm… I think VC, because MS provide it!

# Fragmented Files

**Collect as much of the attacker's trace as possible even from fragmented files.**

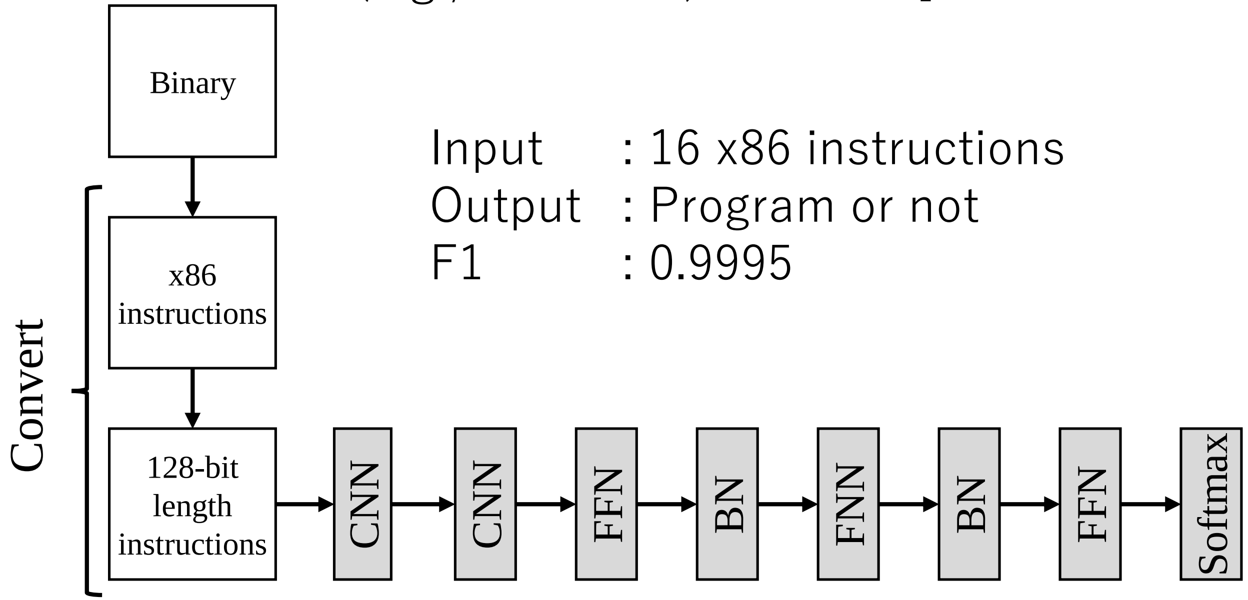


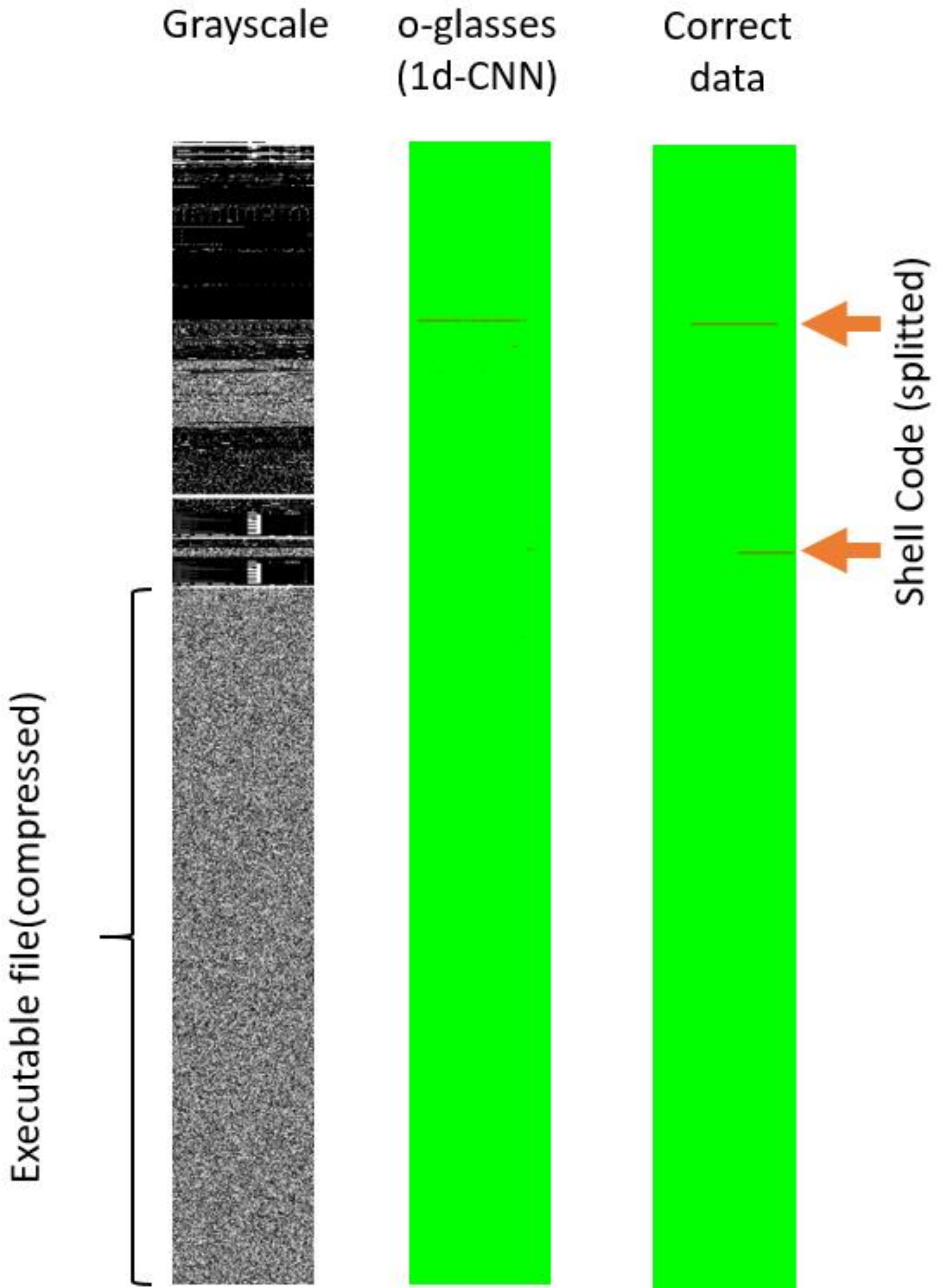
# Preliminaries

# o-glasses

x86 code (e.g., shellcode) detector [arXive.1806.05328]

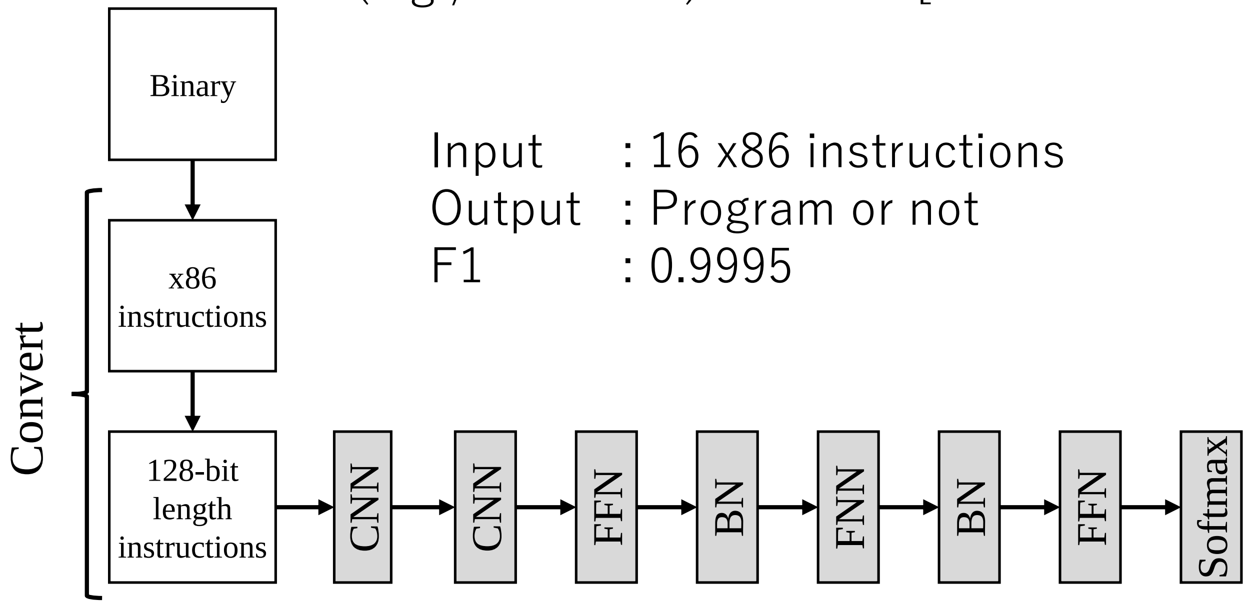



Input : 16 x86 instructions
Output : Program or not
F1 : 0.9995

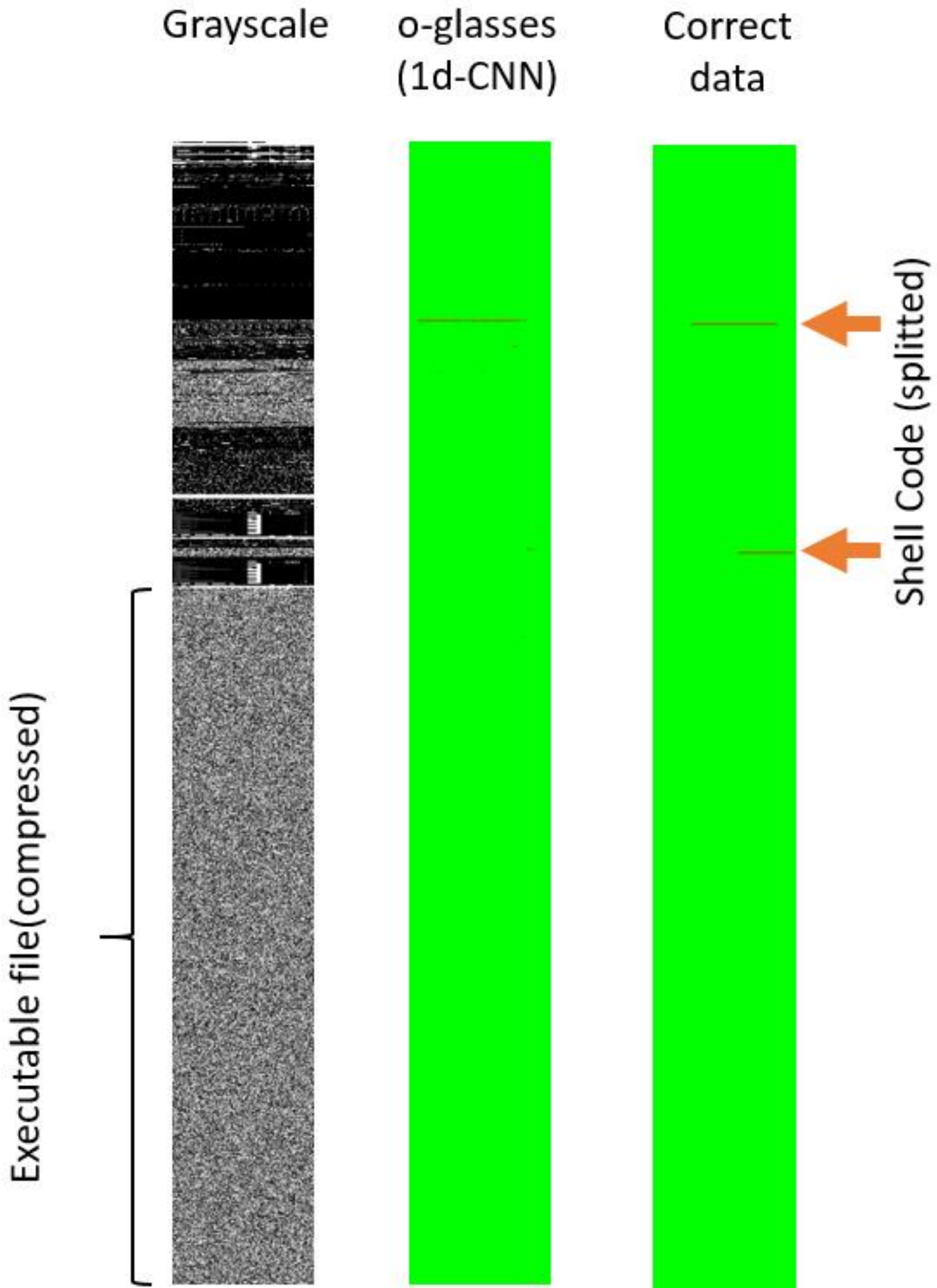


# o-glasses

x86 code (e.g., shellcode) detector [arXive.1806.05328]




Input : 16 x86 instructions
Output : Program or not
F1 : 0.9995




- **Applying to compiler identification**
- **Black Box Problem**

# Attention Is All You Need

[Łukasz Kaiser et al., NIPS, 2017]

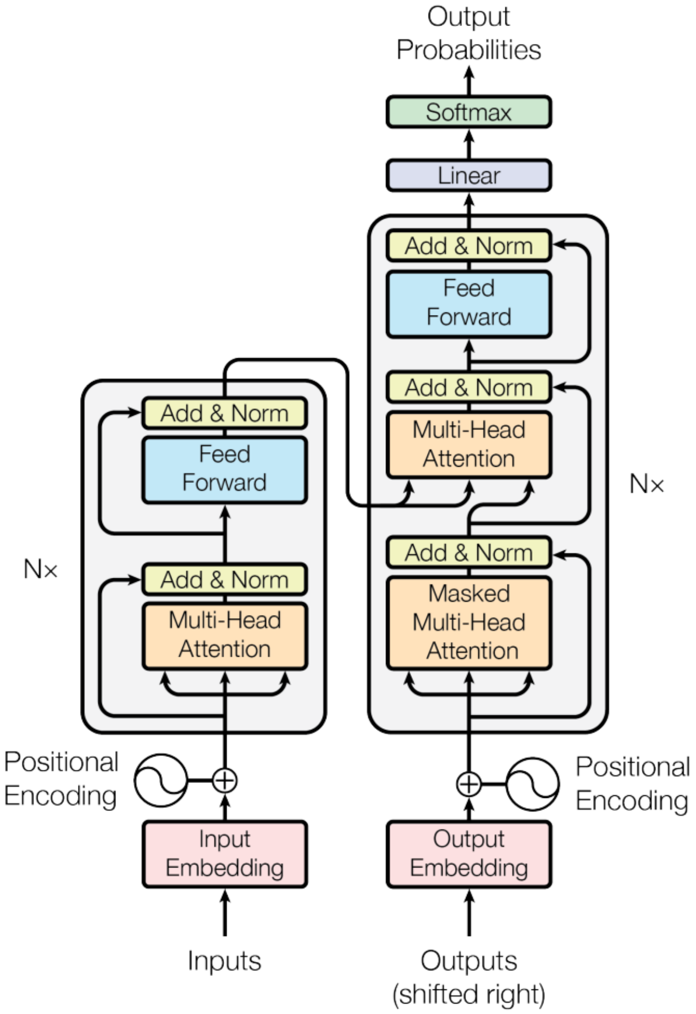



Figure 1: The Transformer - model architecture.

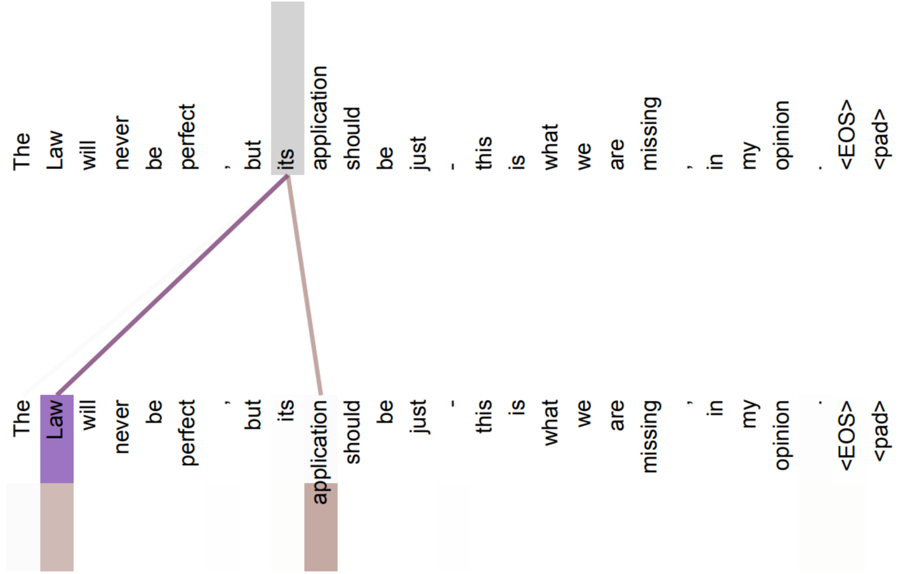
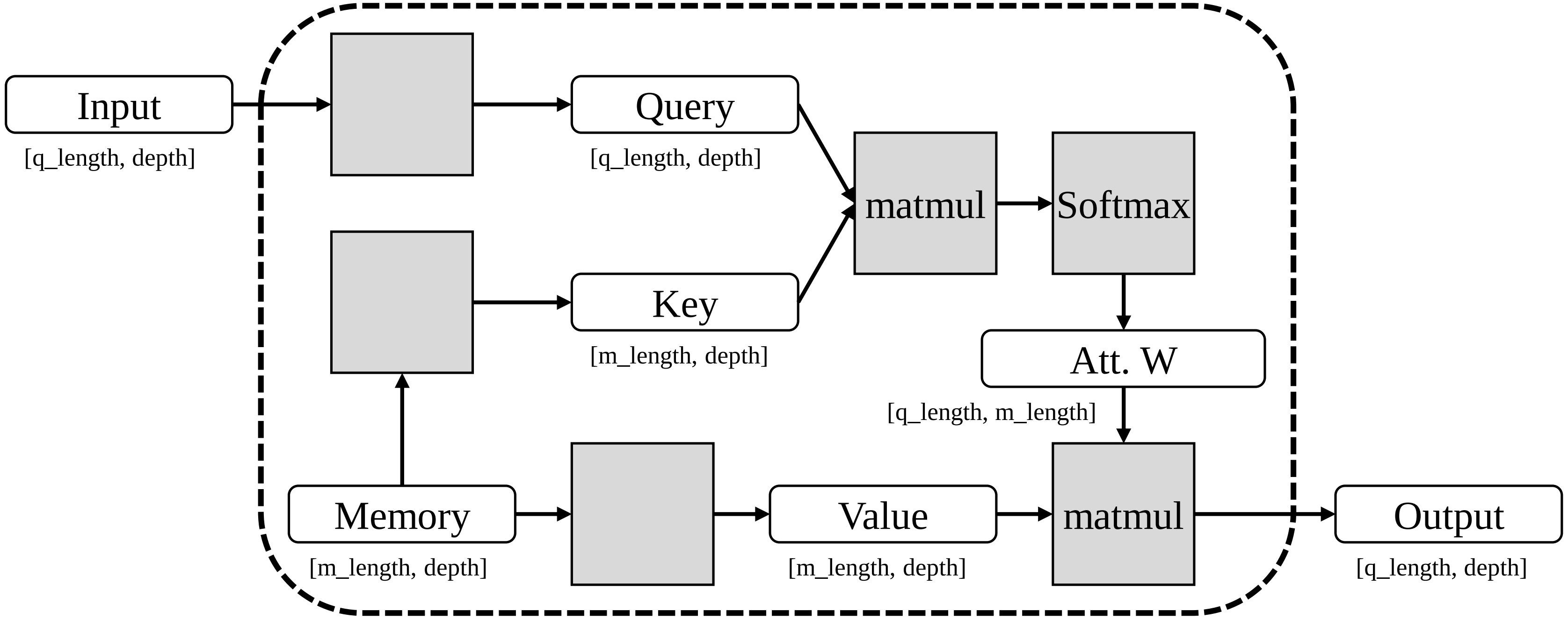



Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

| Model | BLEU | | Training Cost (FLOPs) | |
|---|---|---|---|---|
| | EN-DE | EN-FR | EN-DE | EN-FR |
| ByteNet [15] | 23.75 | | | |
| Deep-Att + PosUnk [32] | | 39.2 | | $1.0 \cdot 10^{20}$ |
| GNMT + RL [31] | 24.6 | 39.92 | $2.3 \cdot 10^{19}$ | $1.4 \cdot 10^{20}$ |
| ConvS2S [8] | 25.16 | 40.46 | $9.6 \cdot 10^{18}$ | $1.5 \cdot 10^{20}$ |
| MoE [26] | 26.03 | 40.56 | $2.0 \cdot 10^{19}$ | $1.2 \cdot 10^{20}$ |
| Deep-Att + PosUnk Ensemble [32] | | 40.4 | | $8.0 \cdot 10^{20}$ |
| GNMT + RL Ensemble [31] | 26.30 | 41.16 | $1.8 \cdot 10^{20}$ | $1.1 \cdot 10^{21}$ |
| ConvS2S Ensemble [8] | 26.36 | **41.29** | $7.7 \cdot 10^{19}$ | $1.2 \cdot 10^{21}$ |
| Transformer (base model) | 27.3 | 38.1 | $\mathbf{3.3 \cdot 10^{18}}$ | |
| Transformer (big) | **28.4** | **41.0** | $2.3 \cdot 10^{19}$ | |

# Basic of Attention

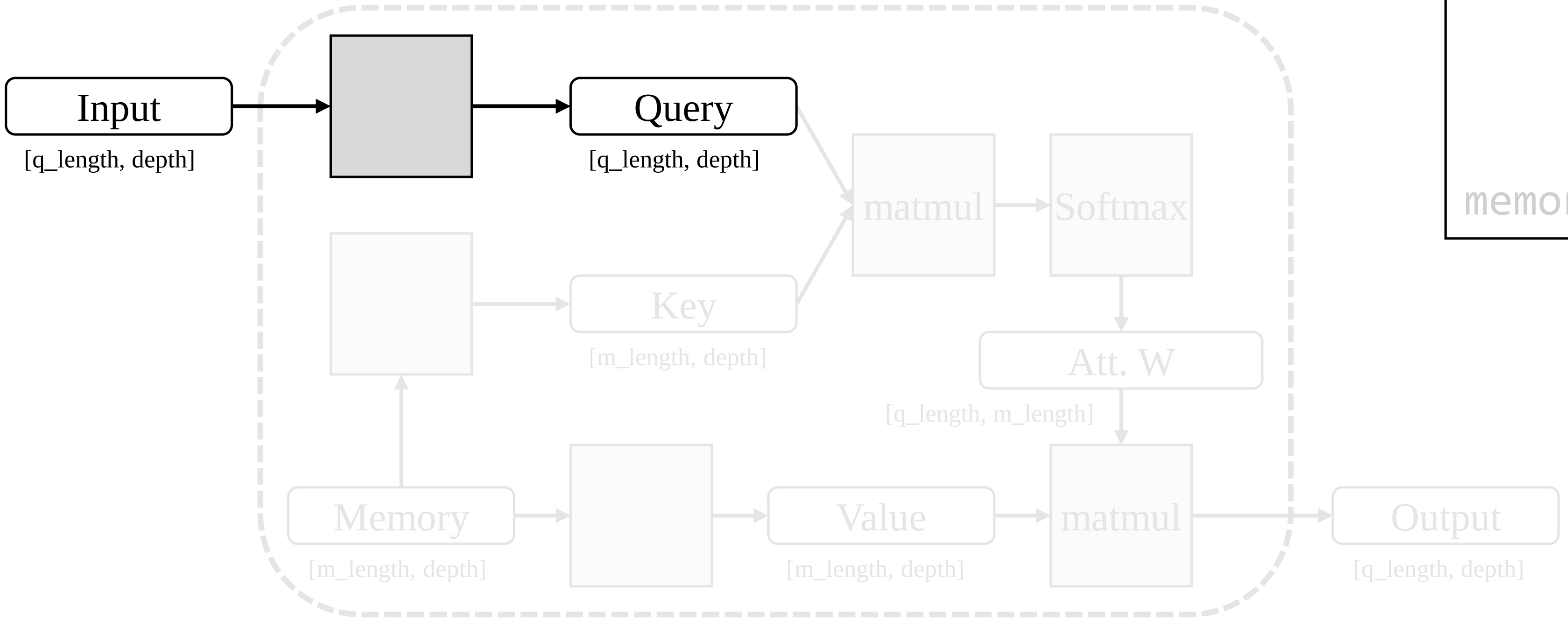


# Basic of Attention



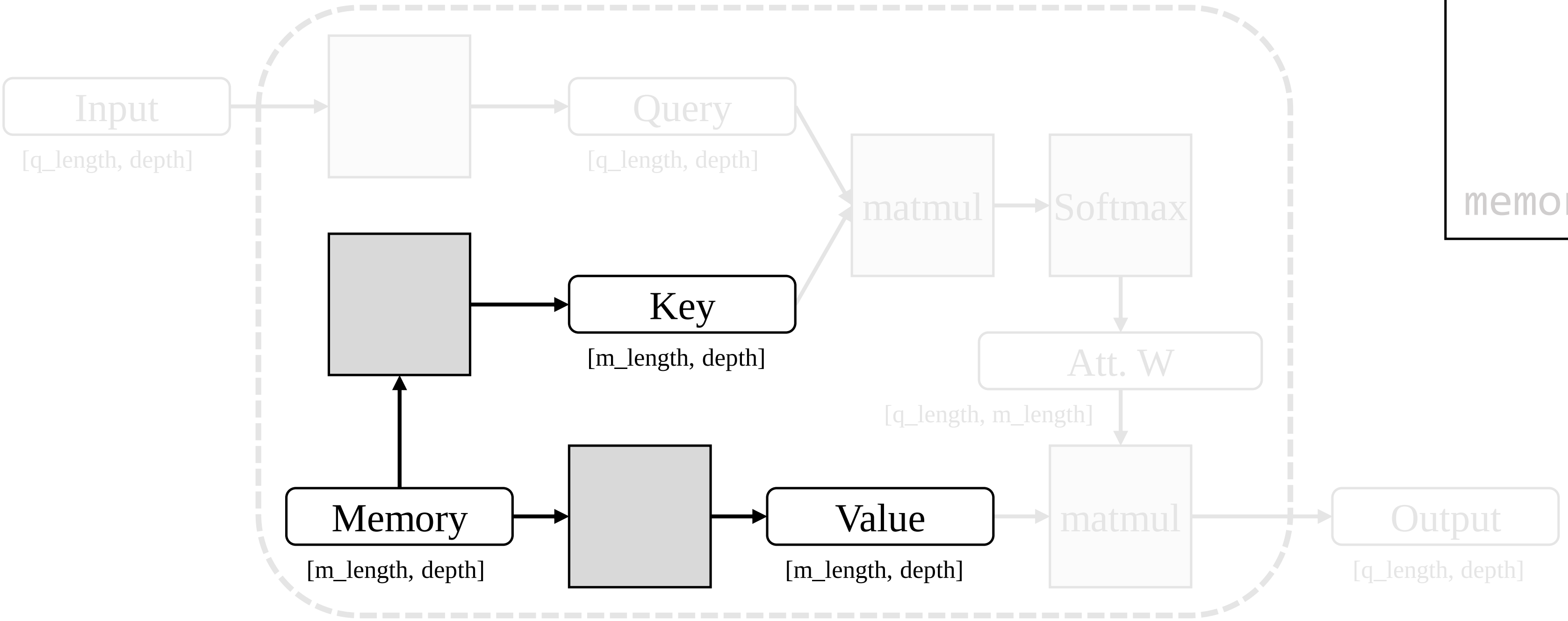

```
query = 'key2'
memory = {'key1':'value2',
             'key2':'value2',
             'key3':'value3',
             'key4':'value4'}
memory[query] = 'value2'
```

# Basic of Attention



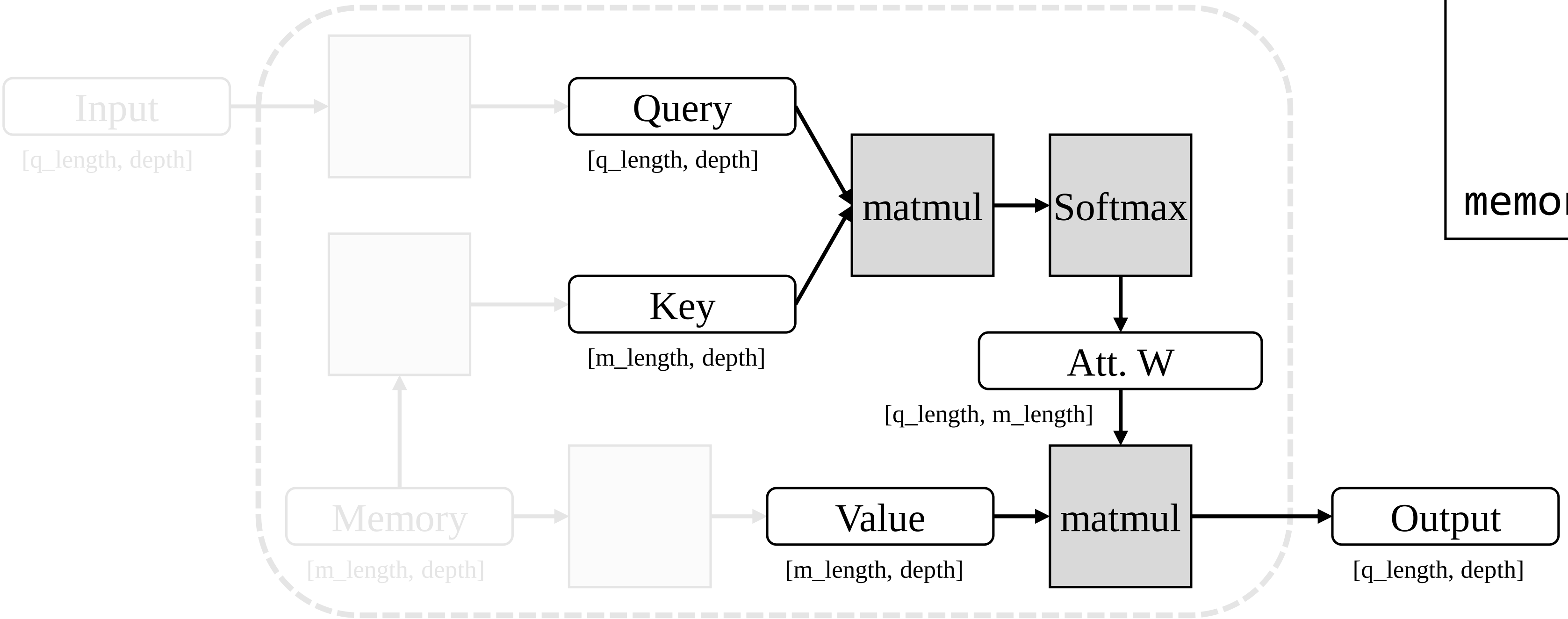

```
query = 'key2'
memory = {'key1':'value2',
          'key2':'value2',
          'key3':'value3',
          'key4':'value4'}
memory[query] = 'value2'
```

# Basic of Attention



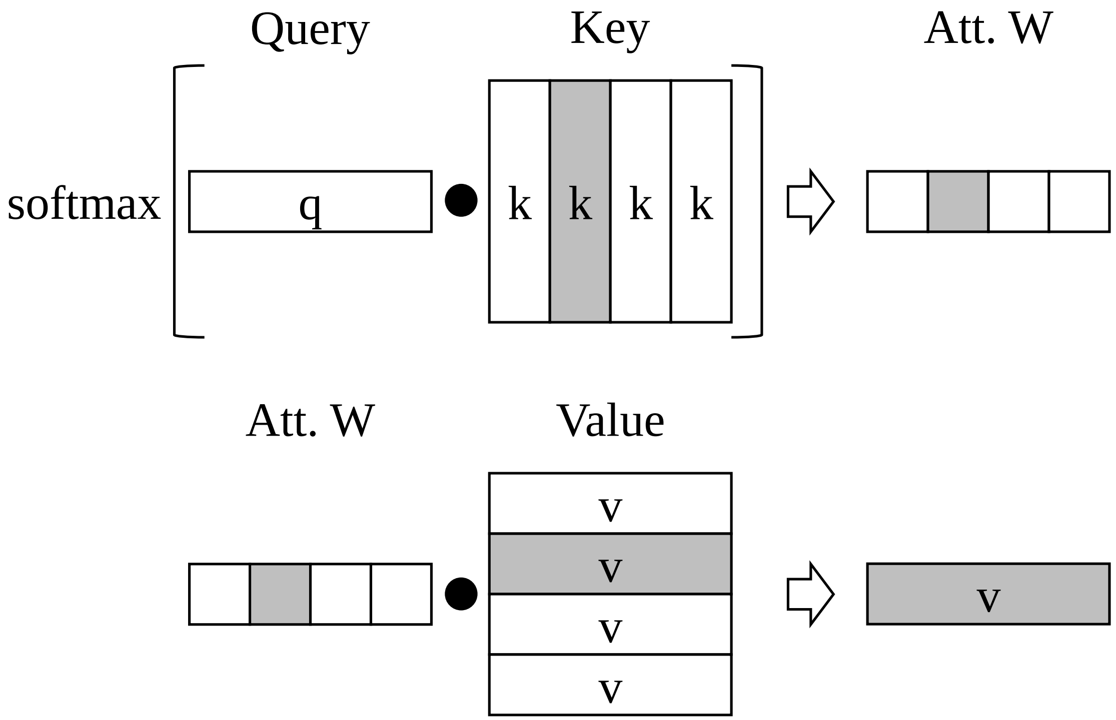

```
query = 'key2'
memory = {'key1':'value2',
          'key2':'value2',
          'key3':'value3',
          'key4':'value4'}
memory[query] = 'value2'
```

# Basic of Attention

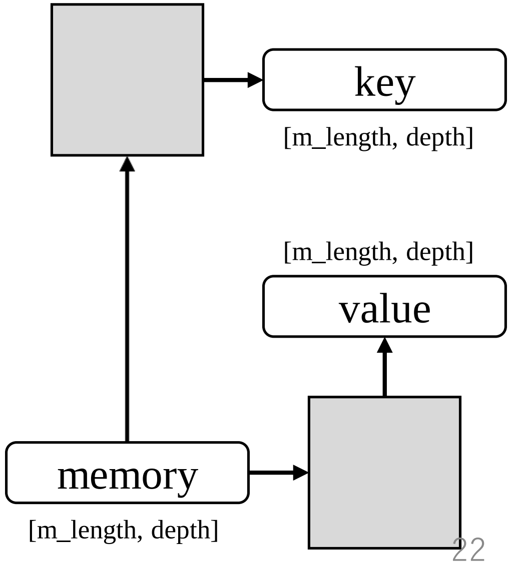



```
query = 'key2'
memory = {'key1':'value2',
          'key2':'value2',
          'key3':'value3',
          'key4':'value4'}
memory[query] = 'value2'
```

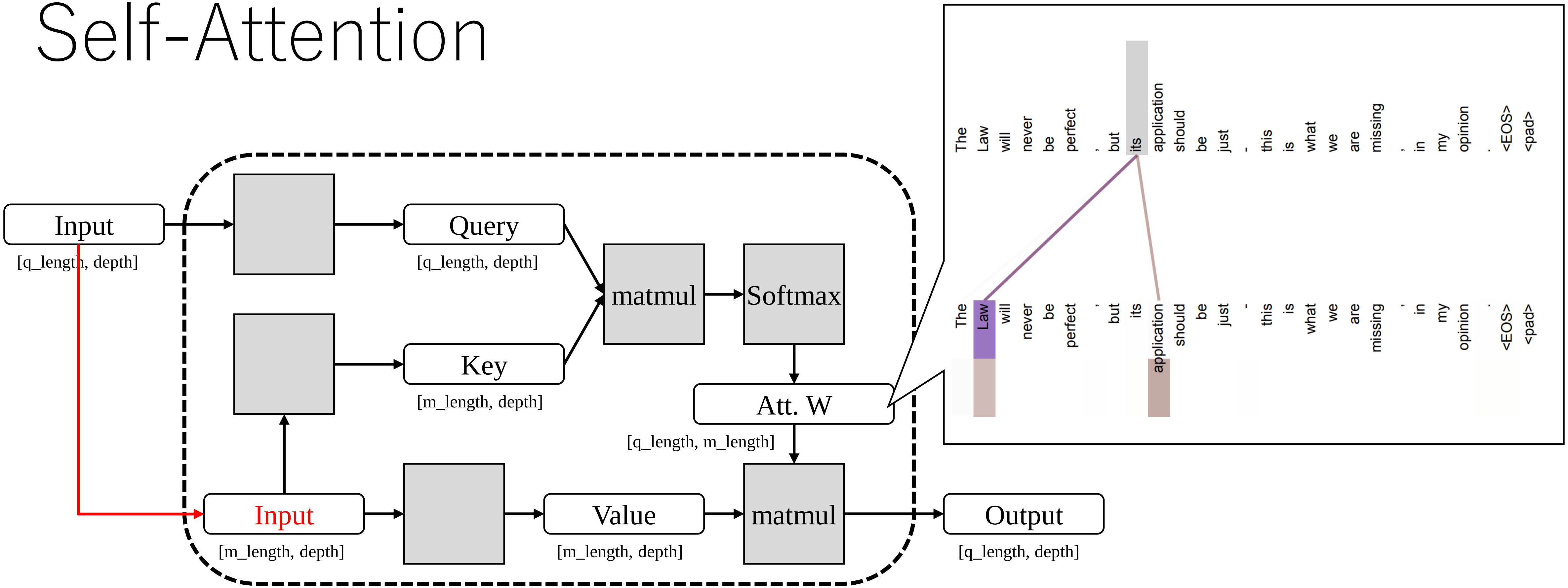
# Dot-Product Attention vs. Dictionary Object




```
query = 'key2'
memory = {'key1':'value2',
          'key2':'value2',
          'key3':'value3',
          'key4':'value4'}
memory[query] = 'value2'
```



# Self-Attention



# Positional Encoding (PE)

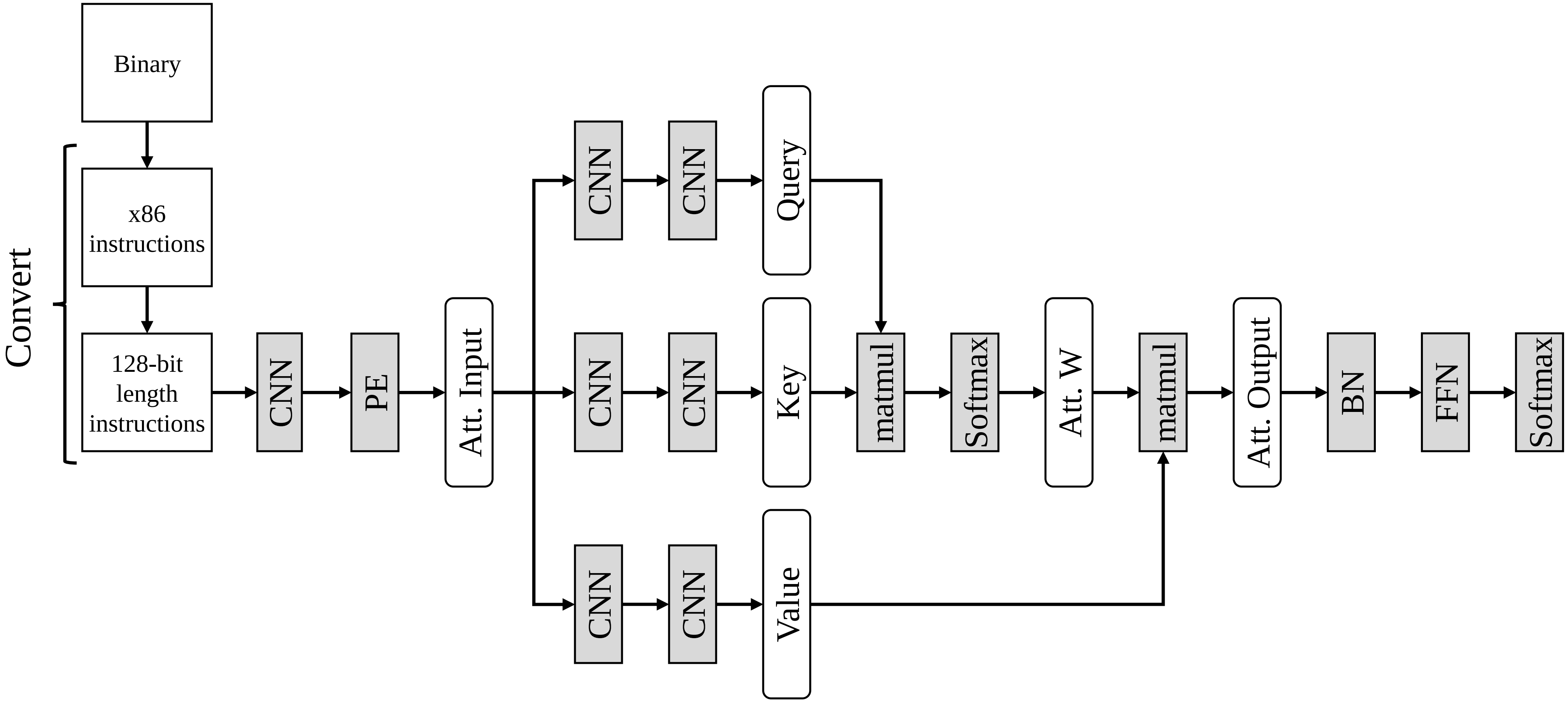
PE (Positional Encoding) adds information about the word position to the input word vectors for learning the context of words.

$$Y(X) = X + \alpha PE$$

$$PE_{(pos,2i)} = \sin\left(pos/10000^{2i/d_{model}}\right)$$
$$PE_{(pos,2i+1)} = \cos\left(pos/10000^{2i/d_{model}}\right)$$

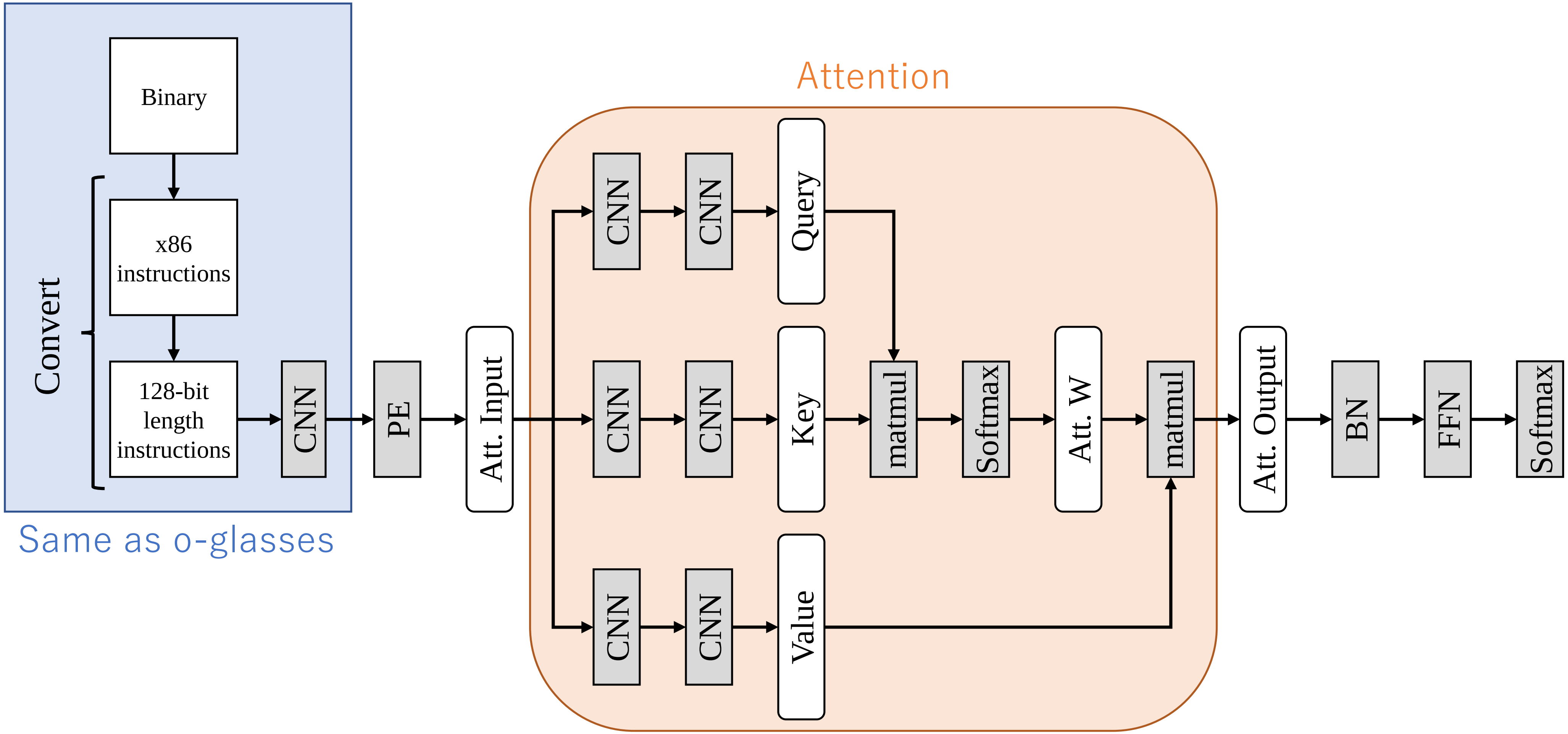
# Proposed Method

# o-glassesX



Binary
x86 instructions
128-bit length instructions
Convert
CNN
PE
Att. Input
CNN
CNN
Query
CNN
CNN
Key
CNN
CNN
Value
matmul
Softmax
Att. W
matmul
Att. Output
BN
FFN
Softmax

# o-glassesX


Binary
x86 instructions
128-bit length instructions
Convert
CNN
Same as o-glasses
PE
Att. Input
Attention
CNN
CNN
Query
CNN
CNN
Key
CNN
CNN
Value
matmul
Softmax
Att. W
matmul
Att. Output
BN
FFN
Softmax

# Preprocessing details

**Binary**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 60 | B9 | 67 | 01 | 00 | 00 | EB | 0F |

**x86 instructions**

| Bytes | Instruction |
|---|---|
| 60 | PUSHA |
| B9 67 01 00 00 | MOV ECX,0x167 |
| EB 0F | JMP loc_17 |

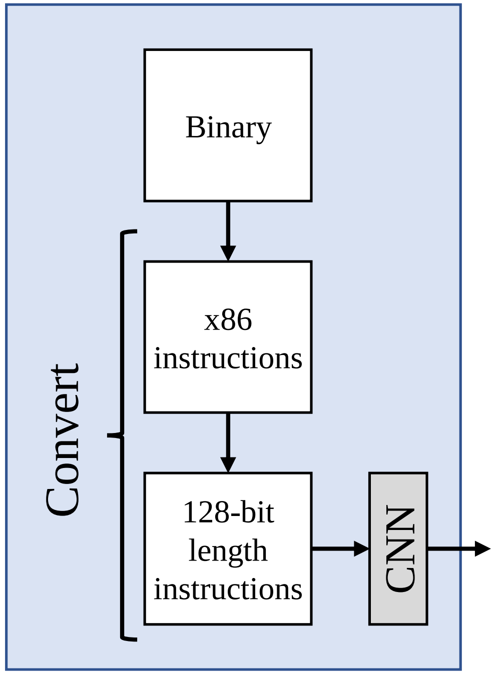
**128-bit length instructions**

16 bytes

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 60 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| B9 | 67 | 01 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| EB | 0F | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

128 bits

```
0000011000000000… … … … …00000000
1001110111100110… … … … …00000000
1101011111110000… … … … …00000000
```

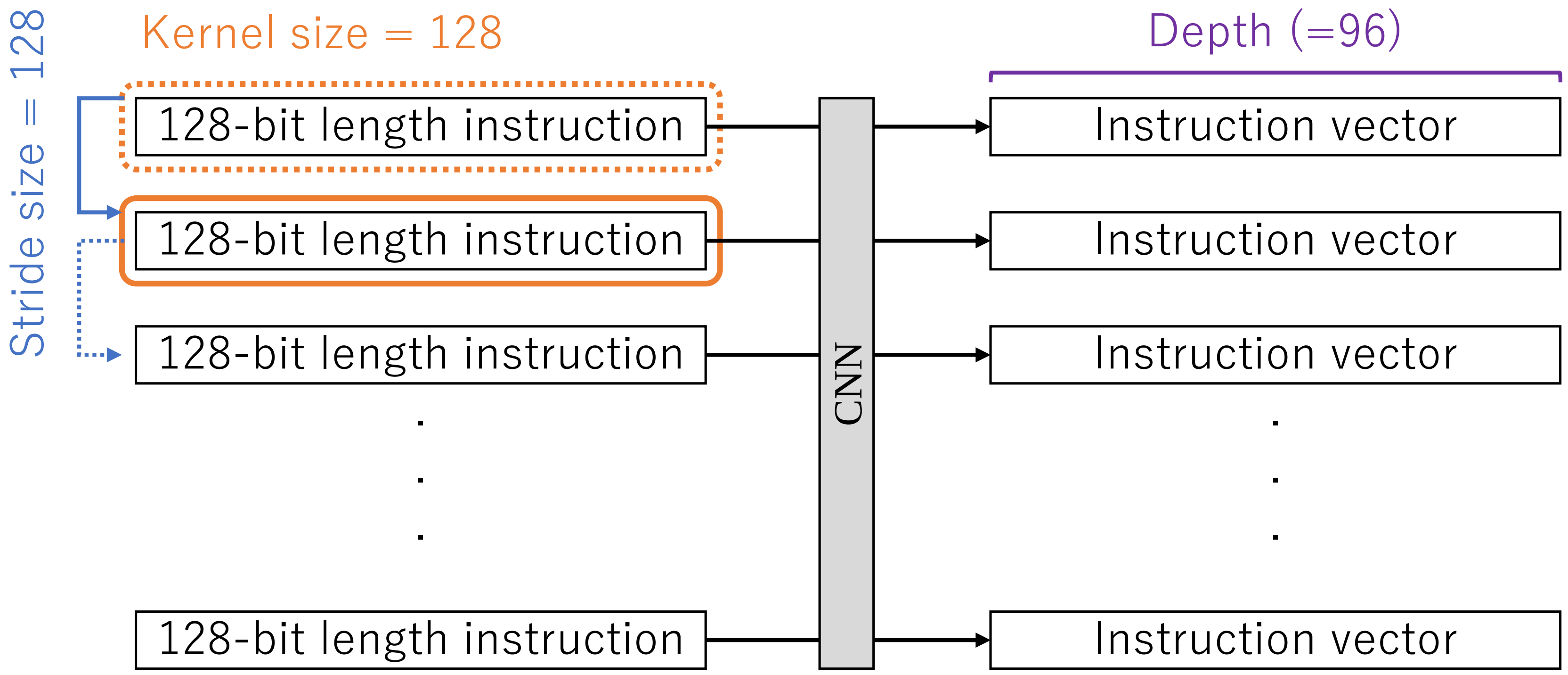


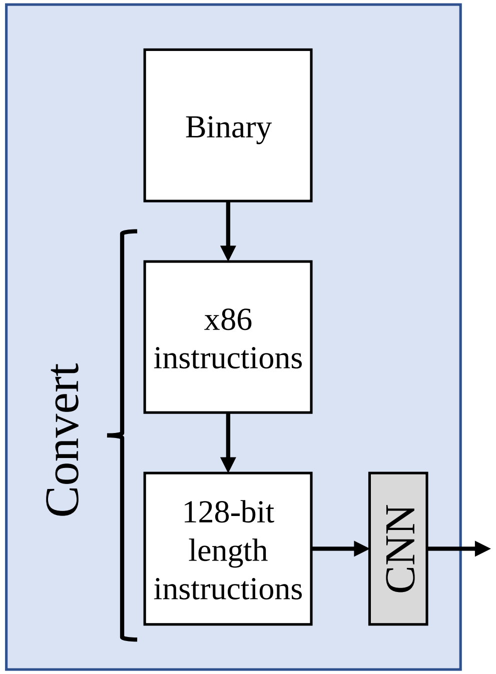

Same as o-glasses

# The 1st CNN Layer

Each unit in CNN has specially local connections to the input units, called a Kernel.
Every kernel shares the weight parameters with the others in the same layer.

Each kernel covers a single instruction by adjusting the hyperparameters.




Same as o-glasses

# Evaluation

# Training Dataset

Collecting source code files from GitHub
Compiling various compilers and options

**Total** : 19 labels
**Compiler** : 4 families
Visual C++, GCC, Clang and Intel C++ Compiler
**Opt. level** : 2 types
maximum or not
**CPU Arc.** : 2 types
x86 or x86-64

TABLE II. DATASET COMPILATION SETTINGS.

| Compiler family | Version | Architecture x86 | Architecture x86-64 | Optimization Level Low | Optimization Level High |
|---|---|---|---|---|---|
| VC | 2003 | ✓ | | -Od | -Ox |
| | 2017 | ✓ | ✓ | -Od | -Ox |
| GCC | 6.3.0 | ✓ | ✓ | -O0 | -O3 |
| Clang | 5.0.2 | ✓ | ✓ | -O0 | -O3 |
| ICC | 19.0.0.117 | ✓ | ✓ | -O0 | -O3 |

| | | | Label | #Binaries | #Code |
|---|---|---|---|---|---|
| Program | VC2017 | x86 | VC17,32,none(Od) | 1,170 | 369,605 |
| | | | VC17,32,max(Ox) | 1,147 | 255,143 |
| | | x86-64 | VC17,64,none(Od) | 1,456 | 540,568 |
| | | | VC17,64,max(Ox) | 1,242 | 542,020 |
| | VC2003 | x86 | VC03,32,none(Od) | 1,350 | 292,277 |
| | | | VC03,32,max(Ox) | 1,306 | 270,743 |
| | | x86-64 | - | - | - |
| | | | - | - | - |
| | GCC | x86 | GCC,32,none(O0) | 2,111 | 227,004 |
| | | | GCC,32,max(O3) | 1,844 | 239,821 |
| | | x86-64 | GCC,64,none(O0) | 1,582 | 283,276 |
| | | | GCC,64,max(O3) | 1,580 | 287,775 |
| | Clang | x86 | Clang,32,none(O0) | 1,205 | 101,024 |
| | | | Clang,32,max(O3) | 1,196 | 86,521 |
| | | x86-64 | Clang,64,none(O0) | 1,892 | 332,278 |
| | | | Clang,64,max(O3) | 1,883 | 246,500 |
| | ICC | x86 | ICC,32,none(Od) | 1,761 | 1,494,677 |
| | | | ICC,32,max(Ox) | 1,724 | 1,161,499 |
| | | x86-64 | ICC,64,none(Od) | 1,796 | 1,419,705 |
| | | | ICC,64,max(Ox) | 1,728 | 1,046,958 |
| | | | Others | 101 | 912,855 |
| | | | Total | 28,074 | 10,110,249 |

# 4-fold Cross Validation (Input: 64 instructions)

| Predict / Train | | VC | | | | | | GCC | | | | Clang | | | | ICC | | | | Others | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | R | P | F1 |
| VC03,32,none(Od) | 1 | .9604 | .0355 | .0026 | .0002 | .0002 | .0002 | .0001 | .0000 | .0000 | .0000 | .0001 | .0000 | .0000 | .0000 | .0001 | .0002 | .0000 | .0000 | .0004 | .9500 | .9604 | .9552 |
| VC17,32,none(Od) | 2 | .0463 | .9517 | .0002 | .0007 | .0001 | .0001 | .0000 | .0000 | .0000 | .0000 | .0001 | .0000 | .0000 | .0000 | .0000 | .0002 | .0000 | .0000 | .0006 | .9625 | .9517 | .9570 |
| VC03,32,max(Ox) | 3 | .0026 | .0001 | .9875 | .0061 | .0001 | .0003 | .0000 | .0002 | .0000 | .0000 | .0000 | .0002 | .0000 | .0000 | .0000 | .0022 | .0000 | .0000 | .0006 | .9786 | .9875 | .9830 |
| VC17,32,max(Ox) | 4 | .0004 | .0010 | .0144 | .9774 | .0001 | .0005 | .0000 | .0001 | .0000 | .0001 | .0001 | .0008 | .0000 | .0001 | .0000 | .0044 | .0000 | .0000 | .0007 | .9887 | .9774 | .9830 |
| VC17,64,none(Od) | 5 | .0002 | .0001 | .0002 | .0001 | .9978 | .0008 | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 | .0001 | .0000 | .0001 | .0001 | .0001 | .0004 | .9977 | .9978 | .9977 |
| VC17,64,max(Ox) | 6 | .0002 | .0001 | .0004 | .0004 | .0013 | .9931 | .0000 | .0000 | .0000 | .0005 | .0000 | .0000 | .0001 | .0006 | .0000 | .0001 | .0000 | .0023 | .0008 | .9955 | .9931 | .9943 |
| GCC,32,none(O0) | 7 | .0002 | .0000 | .0001 | .0000 | .0000 | .0000 | .9973 | .0009 | .0004 | .0001 | .0003 | .0004 | .0000 | .0000 | .0002 | .0000 | .0000 | .0000 | .0000 | .9972 | .9973 | .9973 |
| GCC,32,max(O3) | 8 | .0001 | .0000 | .0005 | .0002 | .0000 | .0000 | .0011 | .9921 | .0000 | .0003 | .0002 | .0051 | .0000 | .0000 | .0000 | .0004 | .0000 | .0000 | .0000 | .9946 | .9921 | .9933 |
| GCC,64,none(O0) | 9 | .0001 | .0001 | .0000 | .0000 | .0000 | .0000 | .0008 | .0000 | .9970 | .0006 | .0000 | .0000 | .0010 | .0002 | .0000 | .0000 | .0001 | .0000 | .0000 | .9979 | .9970 | .9975 |
| GCC,64,max(O3) | 10 | .0000 | .0000 | .0000 | .0000 | .0001 | .0002 | .0001 | .0006 | .0003 | .9800 | .0000 | .0003 | .0003 | .0168 | .0000 | .0001 | .0001 | .0012 | .0000 | .9819 | .9800 | .9809 |
| Clang,32,none(O0) | 11 | .0003 | .0002 | .0000 | .0000 | .0000 | .0000 | .0004 | .0004 | .0000 | .0001 | .9972 | .0006 | .0007 | .0000 | .0001 | .0000 | .0000 | .0000 | .0000 | .9959 | .9972 | .9965 |
| Clang,32,max(O3) | 12 | .0001 | .0000 | .0006 | .0011 | .0000 | .0001 | .0004 | .0068 | .0000 | .0003 | .0006 | .9869 | .0000 | .0019 | .0000 | .0012 | .0000 | .0000 | .0000 | .9790 | .9869 | .9829 |
| Clang,64,none(O0) | 13 | .0000 | .0000 | .0000 | .0000 | .0000 | .0001 | .0000 | .0000 | .0009 | .0003 | .0010 | .0000 | .9973 | .0003 | .0000 | .0000 | .0000 | .0001 | .0000 | .9979 | .9973 | .9976 |
| Clang,64,max(O3) | 14 | .0000 | .0000 | .0001 | .0001 | .0001 | .0006 | .0000 | .0001 | .0002 | .0147 | .0000 | .0015 | .0002 | .9810 | .0000 | .0001 | .0001 | .0013 | .0001 | .9796 | .9810 | .9803 |
| ICC,32,none(Od) | 15 | .0001 | .0000 | .0001 | .0000 | .0000 | .0000 | .0002 | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 | .9994 | .0002 | .0000 | .0000 | .0000 | .9995 | .9994 | .9994 |
| ICC,32,max(Ox) | 16 | .0002 | .0001 | .0029 | .0028 | .0000 | .0001 | .0000 | .0003 | .0000 | .0001 | .0000 | .0005 | .0000 | .0000 | .0001 | .9930 | .0000 | .0001 | .0000 | .9916 | .9930 | .9923 |
| ICC,64,none(Od) | 17 | .0000 | .0000 | .0000 | .0000 | .0001 | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 | .9996 | .0001 | .0000 | .9995 | .9996 | .9995 |
| ICC,64,max(Ox) | 18 | .0000 | .0000 | .0001 | .0000 | .0001 | .0016 | .0000 | .0001 | .0000 | .0012 | .0000 | .0000 | .0000 | .0013 | .0000 | .0001 | .0002 | .9952 | .0000 | .9948 | .9952 | .9950 |
| Others | 19 | .0000 | .0000 | .0000 | .0001 | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 | .9997 | .9964 | .9997 | .9980 |

# Comparison of Performance of Related Work

| | | o-glassesX (L=64) | o-glassesX (L=16) | o-glasses[23] (L=64) | o-glasses[23] (L=16) | Rosenblum's[29] | ORIGIN[26] (SVM) | ORIGIN[26] (CRF) | BinComp[25] |
|---|---|---|---|---|---|---|---|---|---|
| Accuracy | All components | .9884 (19) | .9555 (19) | .9421 (19) | .9323 (19) | .924 (3) | .604 (18) | .918 (18) | .801 (6) |
| | Compiler family | .9886 (4) | .9670 (4) | .9140 (4) | .9271 (4) | .924 (3) | .983 (3) | .999 (3) | – |
| | Optimization | .9989 (2) | .9943 (2) | .9830 (2) | .9864 (2) | – | .971 (2) | .999 (2) | .917 (2) |
| | Architecture | .9997 (2) | .9985 (2) | .9959 (2) | .9940 (2) | – | – | – | – |
| | Code/Non-code | .9999 (2) | .9995 (2) | .9991 (2) | .9987 (2) | – | – | – | – |
| ML model | | Attention | Attention | CNN | CNN | CRF | SVM | CRF | (k-means) |
| Dataset | Features | 64 Instructions | 16 Instructions | 64 Instructions | 16 Instructions | Byte Seq. | 1 Function | Function Seq. | 1 File |
| | #Samples | 1,793,478 | 1,900,000 | 471,124 | 1,886,521 | 81,886,169 | 955,000 | 955,000 | 1,177 |
| | #Binaries | 28,074 | 28,074 | 28,074 | 28,074 | 1,119 | 2,686 | 2,686 | 1,177 |
| | #Variations VC | 6 | 6 | 6 | 6 | 1 | 6 | 6 | 2 |
| | #Variations GCC | 4 | 4 | 4 | 4 | 1 | 8 | 8 | 2 |
| | #Variations Clang | 4 | 4 | 4 | 4 | - | - | - | - |
| | #Variations ICC | 4 | 4 | 4 | 4 | 1 | 4 | 4 | 2 |
| | #Variations Non-code | 1 | 1 | 1 | 1 | - | - | - | - |
| K-fold cross-validation | | 4 | 4 | 4 | 4 | (Anomalous) | (Anomalous) | (Anomalous) | 10 |

# Calculating ‘Why’ with the Attention Mechanism

| Input | |
|---|---|
| ENTER 0x558b, 0x8 | .222 |
| MOV [EDX+0xc], ECX | .115 |
| CMP DWORD [EBP+0x10], 0x80 | .029 |
| JNZ 0x40001b7 | .016 |
| MOV EAX, [EBP+0x8] | .019 |
| MOV ECX, [EAX+0xc] | .079 |
| MOV [EBP-0x8], ECX | .036 |
| MOV EDX, [EBP-0x8] | .041 |
| SHR EDX, 0x10 | .063 |
| AND EDX, 0xff | .255 |
| MOV EAX, [EDX*4+0x0] | .153 |
| AND EAX, 0xff000000 | .044 |
| MOV ECX, [EBP+0x8] | .062 |
| XOR EAX, [ECX] | .207 |
| MOV EDX, [EBP-0x8] | .039 |
| SHR EDX, 0x8 | .063 |

**Algorithm 1** Automatic feature extraction procedure

```
InputFile ← Compile(SourceFile,Optimization).pullCode()
Offset ← 0
while Offset < InputFile.EndOffset do
    Input ← InputFile.pull(Offset,L)
    Result ← model.predictor(Input)
    if Result.Label = RealLabel then
        if Result.Confidence > 0.99 then
            i ← argsmax(L2(Result.AttentionWeight))
            Count[input[i]] ++
        end if
    end if
    Offset ++
end while
Print(Ranking(Count))
```

# Typical Instructions for each compiler

e.g.,
when focusing on the NOP instruction…

| | VC03,32,none(Od) | VC17,32,none(Od) | GCC,32,none(O0) | Clang,32,none(O0) | ICC,32,none(Od) |
|---|---|---|---|---|---|
| 1 | XOR ECX, [EAX*4+0x0] | IMUL ECX, EAX, 0x0 | AND EAX, 0xff000000 | AND EAX, 0xff000000 | IMUL EAX, EAX, 0x4 |
| 2 | AND EAX, 0xff000000 | IMUL ECX, EAX, 0x9 | MOVZX EAX, AL | NOP WORD [EAX+EAX+0x0] | AND EAX, 0xff000000 |
| 3 | XOR EDX, [EAX*4+0x0] | IMUL EDX, ECX, 0xc | LEA EDX, [EAX*4+0x0] | AND ECX, 0xff0000 | IMUL EDX, EDX, 0x4 |
| 4 | AND ECX, 0xff0000 | XOR ECX, EBP | ADD DWORD [EBP-0x8], 0x1 | XOR EAX, [ECX*4+0x1028] | CALL 0x4003c76 |
| 5 | XOR EAX, [EDX*4+0x0] | IMUL ECX, EAX, 0x3 | AND EAX, 0xff0000 | NOP DWORD [EAX+0x0] | MOV EAX, 0xffffffff |

| | VC03,32,max(Ox) | VC17,32,max(Ox) | GCC,32,max(O3) | Clang,32,max(O3) | ICC,32,max(Ox) |
|---|---|---|---|---|---|
| 1 | AND ESI, 0xff0000 | NOP | LEA ESI, [ESI+0x0] | NOP WORD [CS:EAX+EAX+0x0] | NOP DWORD [EAX+0x0] |
| 2 | AND ECX, 0xff0000 | MOVZX ECX, BYTE [EAX+EBP] | LEA EDI, [EDI+0x0] | MOV EBP, 0xff000000 | LEA EBX, [EDX+EDX] |
| 3 | AND EDI, 0xff | PUSH DWORD [ESP+0x24] | AND EAX, 0xff000000 | MOV EDX, 0xff00 | LEA ECX, [ESI+ESI] |
| 4 | XOR ECX, EBP | SUB DWORD [ESP+0x2c], 0x1 | AND EAX, 0xff0000 | MOV DL, [ESP+0x15] | LEA EDX, [ESI+ESI] |
| 5 | AND ECX, 0xff | XOR EAX, ESP | AND EBX, 0xff0000 | MOV ESI, 0xff00 | PXOR XMM0, [ESP+0x20] |

Fig. 5. Typical instructions for each compiler against aes.c

# Case Study: Various Optimization Levels

OfficeMalScanner.exe (OMS)

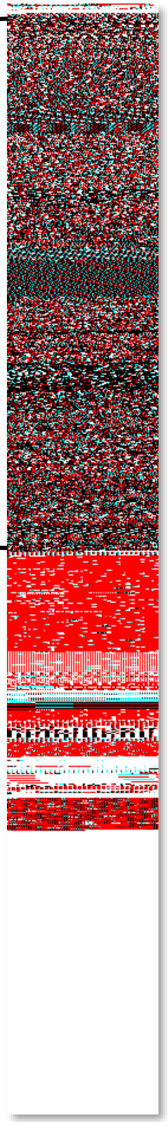
| Function name | Segment | Start | Length |
|---|---|---|---|
| sub_401010 | .text | 00401010 | 0000014C |
| sub_401160 | .text | 00401160 | 0000023A |
| sub_4013A0 | .text | 004013A0 | 00000147 |
| sub_4014F0 | .text | 004014F0 | 000000BF |
| sub_4015B0 | .text | 004015B0 | 0000039D |
| **_main** | **.text** | **00401950** | **00004129** |
| sub_405AA0 | .text | 00405AA0 | 00000046 |
| sub_405AF0 | .text | 00405AF0 | 00000085 |
| sub_405B80 | .text | 00405B80 | 0000042D |
| sub_405FB0 | .text | 00405FB0 | 000001E7 |
| sub_4061A0 | .text | 004061A0 | 00000015 |
| sub_4061C0 | .text | 004061C0 | 0000002D |
| MakeMnemonic | .text | 004061EE | 00000006 |
| InstrDasm | .text | 004061F4 | 00000006 |
| InstrDecode | .text | 004061FA | 00000006 |
| ___crtCorExitProcess | .text | 00406200 | 0000002B |
| ___crtExitProcess | .text | 0040622B | 00000017 |
| __lockexit | .text | 00406243 | 00000009 |
| __unlockexit | .text | 0040624C | 00000009 |
| __init_pointers | .text | 00406255 | 00000033 |
| __initterm_e | .text | 00406288 | 00000024 |
| __cinit | .text | 004062AC | 00000097 |
| _doexit | .text | 00406343 | 00000140 |
| **_exit** | **.text** | **00406483** | **00000016** |
| **__exit** | **.text** | **00406499** | **00000016** |
| **__cexit** | **.text** | **004064AF** | **0000000F** |
| **__c_exit** | **.text** | **004064BE** | **0000000F** |
| __amsg_exit | .text | 004064CD | 0000001D |
| **_printf** | **.text** | **004064EB** | **000000A7** |

Original Code
(no optimization)

Static Link Library
(maximum optimization)

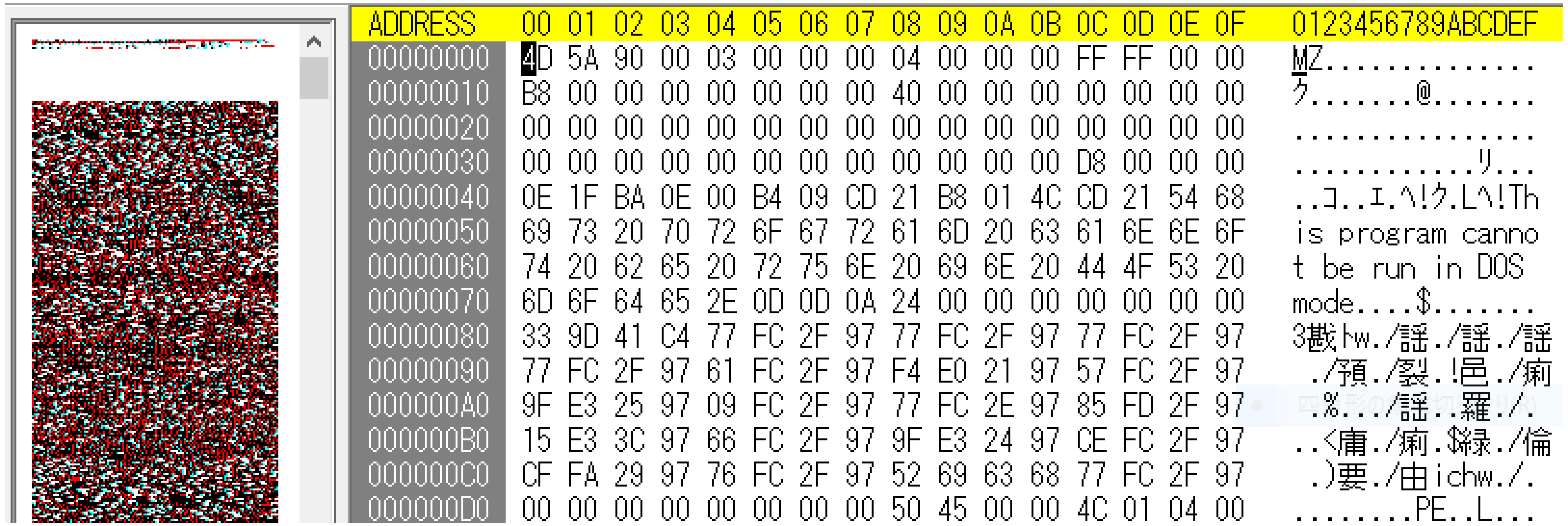
# Bit-Image of OMS

.text segment
(Program Code)



NULL (0x00)
Control Characters (0x01-0x1F)
Printable Characters (0x20-0x7F)
Others (0x80-0xFF)

Stirling[10] : a hex editor



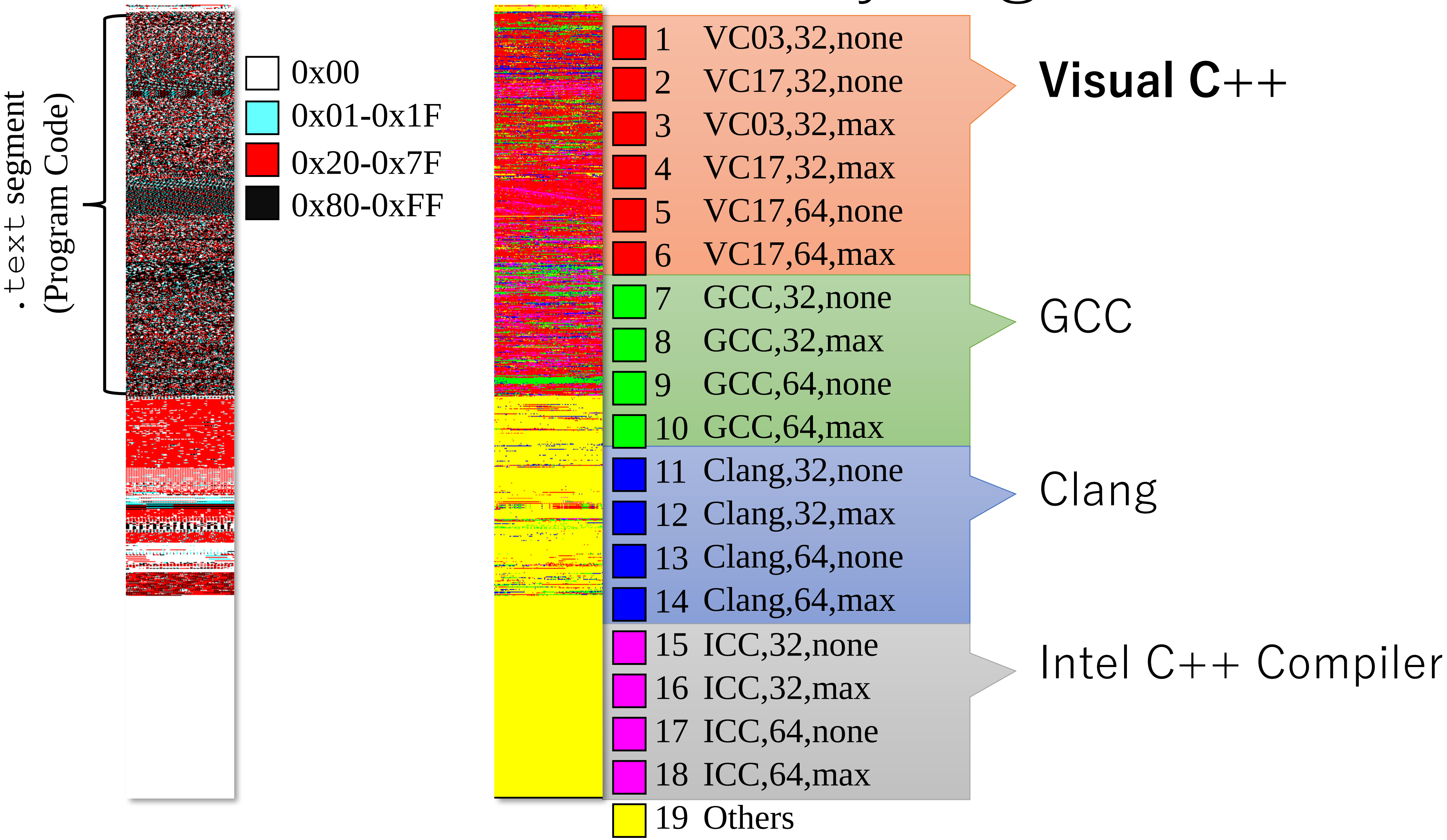

Bit-Image View

Hex View

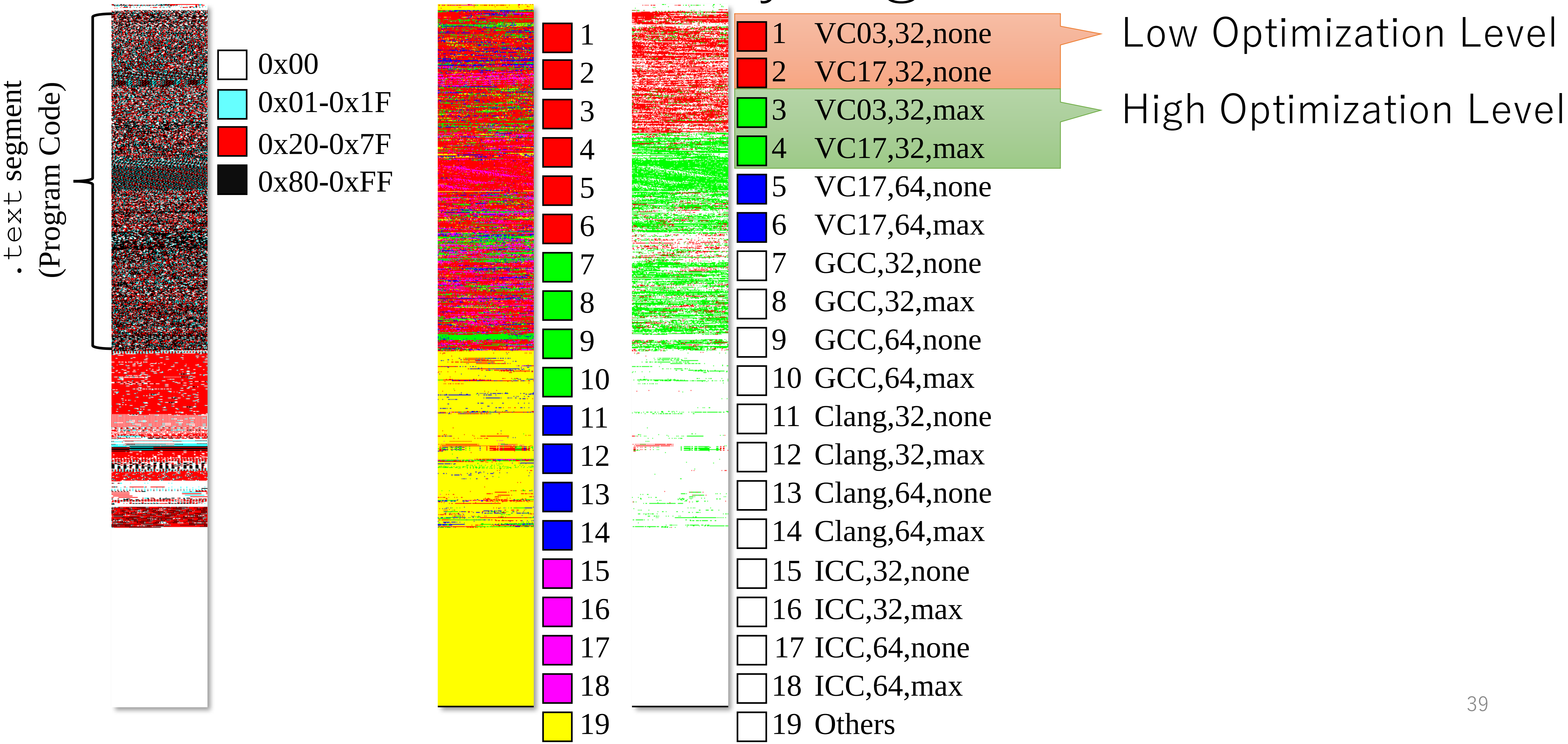


[10] K. Goto, "Stirling," https://www.vector.co.jp/soft/win95/util/se079072.html, 1998.

# Visualization of OMS by o-glassesX



# Visualization of OMS by o-glassesX



# Case Study: Tracking Emdivi RATs in Dev. Env.

**EMDIVI** malware family is used in targeted email attacks against Japanese organizations.
It allows machines to be remotely controlled by attackers for malicious commands and other activities.

# The Version of Emdivi

```
sub_4282BD proc near
push    offset aT17_08_26_kenp ; "t17.08.26.KENPO0202"
mov     ecx, offset unk_437590
call    sub_401F54
push    offset sub_42905B
call    _atexit
pop     ecx
retn
sub_4282BD endp
```

- Almost attached malware is compiled just before used
  So, the sender and the developer may be in the same group

- Frequently Updated
  t17 : For initial compromise
  t19,t20 : For expanding the intrusion (High stealth performance)



https://www.macnica.net/security/report_01.html/

# Emdivi dataset

Analysis Report made by Macnica Netwoks
https://www.macnica.net/security/report_01.html/


163 MD5 Hashes of the Emdivi Family

**Appendix** Emdivi RAT ハッシュ値

| MD5 | Version | Compile Time (UTC+8) |
|---|---|---|
| 7fa87d1adc06bb19dde13689afe8f8ef | t9_4_sender | 2012/05/28 10:02:45 |
| 2f210e5e55eb90880c12019e358c43fb | t9_5_system | 2012/05/30 12:49:25 |
| 66680364d2f006db747dd640b044efe3 | t9_5_system | 2012/05/30 11:51:39 |
| d953cadc4be2ab27219ef87a6a1aad87 | t9_5_system | 2012/05/30 14:47:22 |
| 3a68b60202787c4c779f8534ea186c75 | t9_5_system | 2012/05/30 12:49:25 |
| b1f967dfe09603844a2354977356165f | t9_5_system | 2012/05/30 14:47:22 |
| 4aa0d9c2b300d627c1f5abd048331597 | t9_6 | 2012/06/01 22:07:15 |
| 094d87782555477fdc6325c56c28ff30 | t9_6 | 2012/06/01 22:07:15 |
| a219e2c31784bec4fc159400b229f4e0 | t9_6 | 2012/06/01 18:27:44 |
| dfb0ad1e22d60716512855602d47392d | t9_6 | 2012/06/01 19:24:13 |
| e01e34660211bb8c7c746a6819f81c2b | t9_6 | 2012/06/01 19:24:13 |
| d2f46428e1651ab6555d6f5ee87b04e9 | t11.05 | 2012/09/13 12:58:32 |
| 1c462660b33130f5e9c2ad664eeedb40 | t15.07 | 2013/04/22 13:17:43 |
| 402406d85bcec25b14e2baa1c7b9584c | t16.19 | 2013/07/25 16:21:30 |

# Difference of Emdivi Rats in Dev. Env.

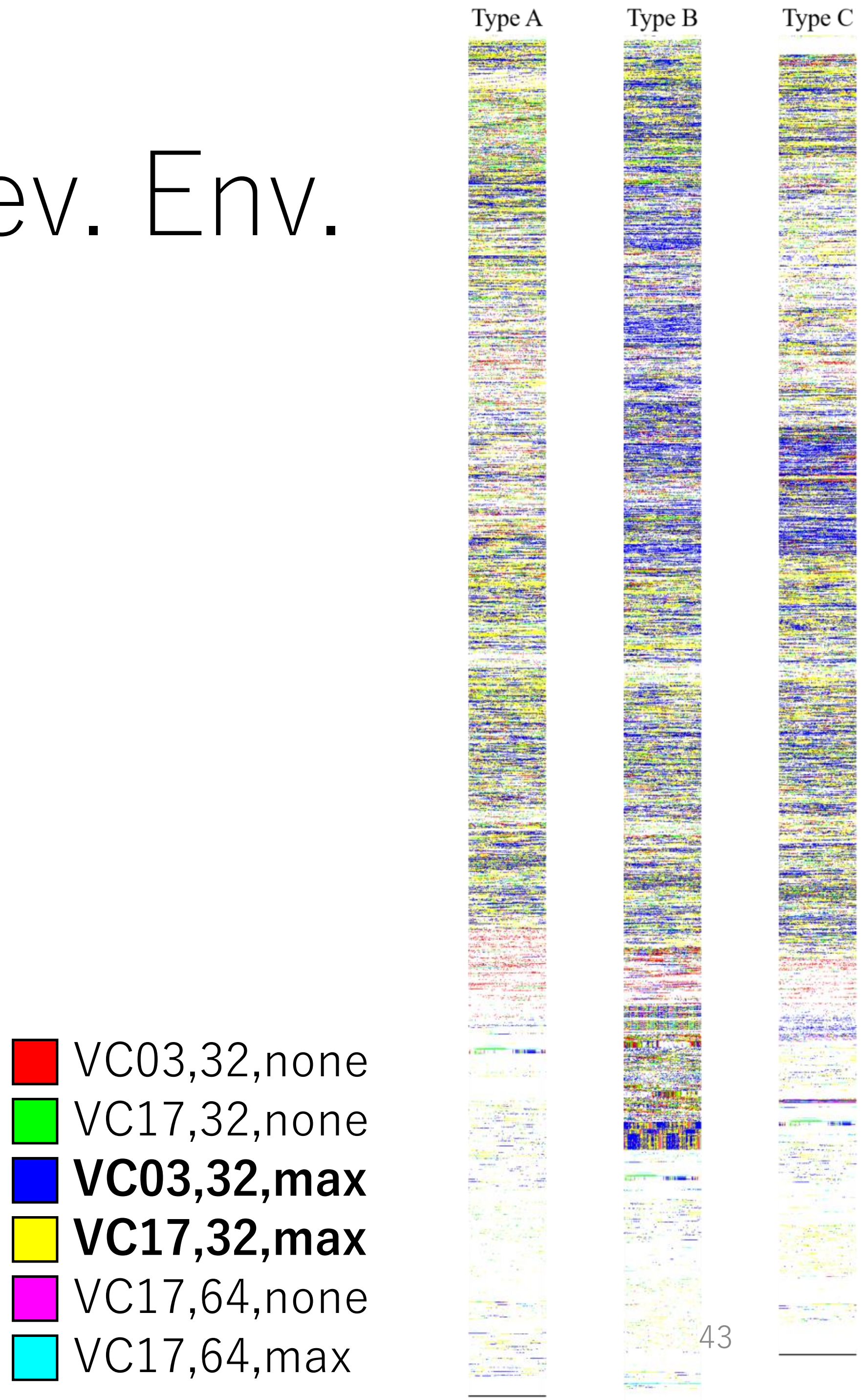
## All Sample

Architecture : 32-bit (x86)
Compiler family : Visual C++
Optimization level : max

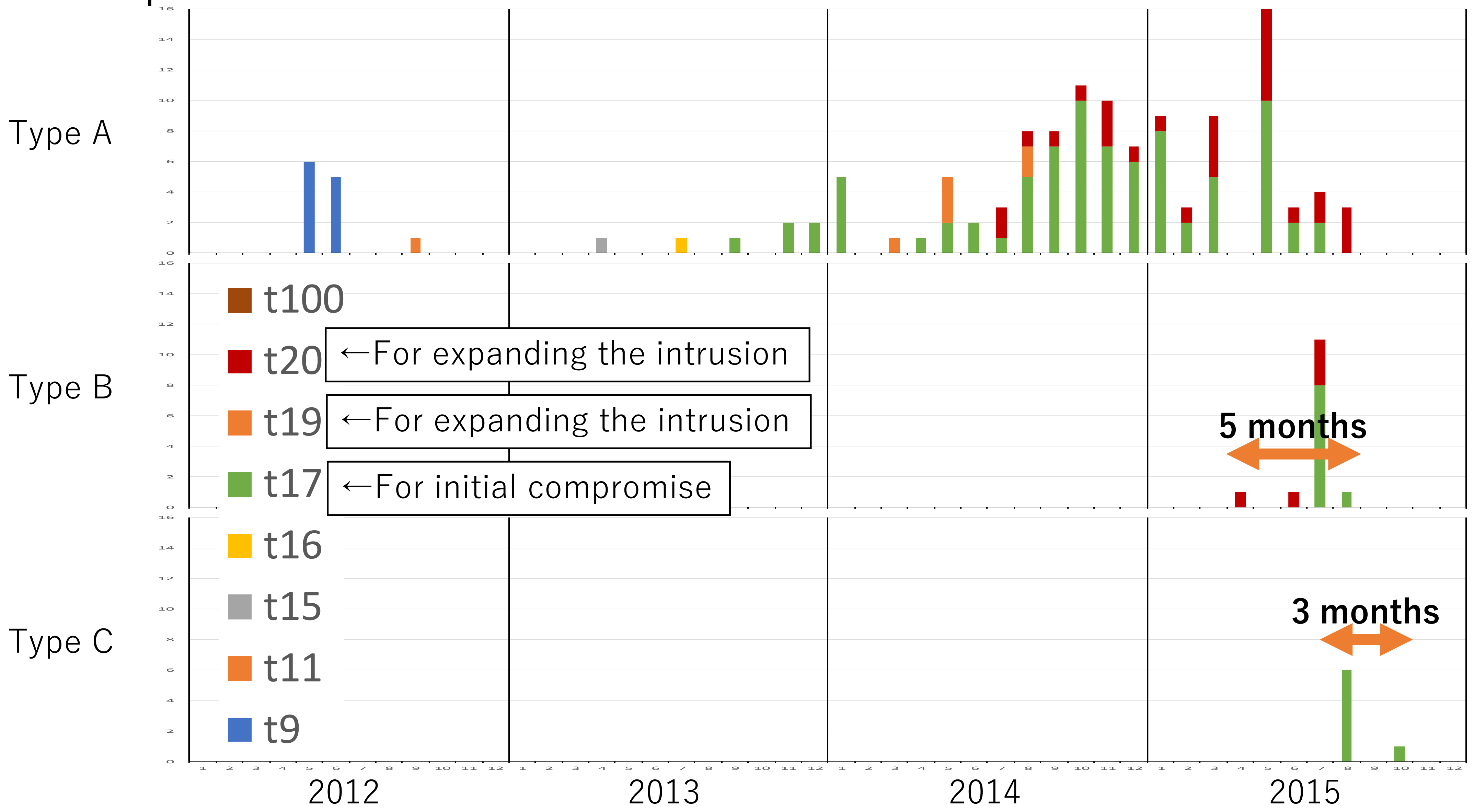
## Focusing on Compiler Version

Yellow : relatively new compiler
Blue : relatively old compiler

Type A: Yellow
Type B: Blue -> Yellow
Type C: Yellow -> Blue -> Yellow



# Compile-time and version of Emdivi RATs



# Limitation

## Obfuscated Code

Input machine code need to be already de-obfuscated.

## Multi CPU Architectures

This method may be applied to many CPU architecture besides x86.
Splitting binary by instruction is difficult in two more CPU architectures inputs at the same time.
This limitation will be resolved, if new bin2vec method supporting multi CPU Arc. is released…

# Conclusion

## High Recognition Rate for Stripped Machine Code

16-instruction input: .956 accuracy
64-instruction input: .988 accuracy

## Solution to Black Box Problem

o-glassesX can calculate how much input data contributes to output in units of instructions

## Case Study: Emdivi

It has been revealed that there are three attackers in the same attack group.

o-glassesX and our dataset are available at

**https://github.com/yotsubo/o-glassesX**