De-anonymizing Programmers via Code Stylometry

Aylin Caliskan-Islam
Drexel University

Richard Harang
U.S. Army Research Laboratory

Andrew Liu
University of Maryland

Arvind Narayanan
Princeton University

Clare Voss
U.S. Army Research Laboratory

Fabian Yamaguchi
University of Goettingen

Rachel Greenstadt
Drexel University

24th USENIX
Security Symposium
AUGUST 12–14, 2015 • WASHINGTON, D.C.
usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Presenter : Taehyeon Lee

CySecLab KAIST

KAIST CySecLab

Introduction

Source code stylometry & de-anonymization

Stylometry

Source Code

Stylistic fingerprints

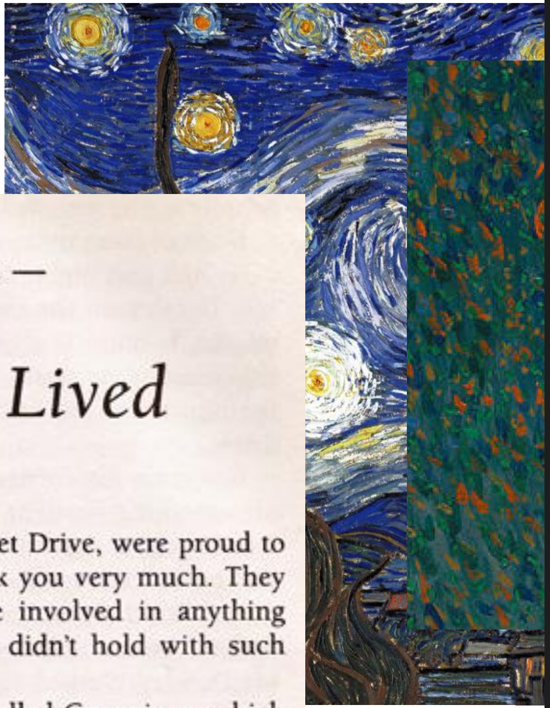
- Fine-art
- Unconventional text
- Translated text

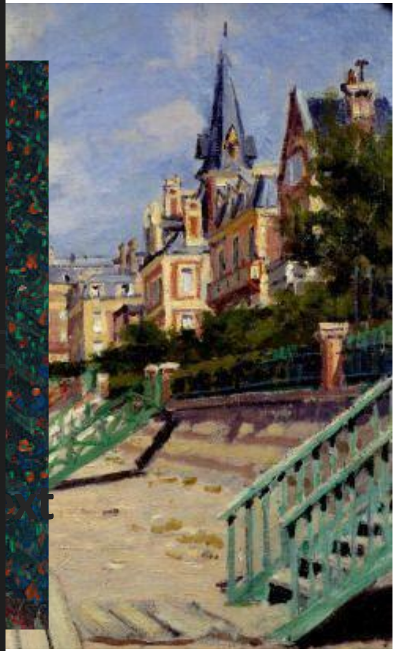
Books

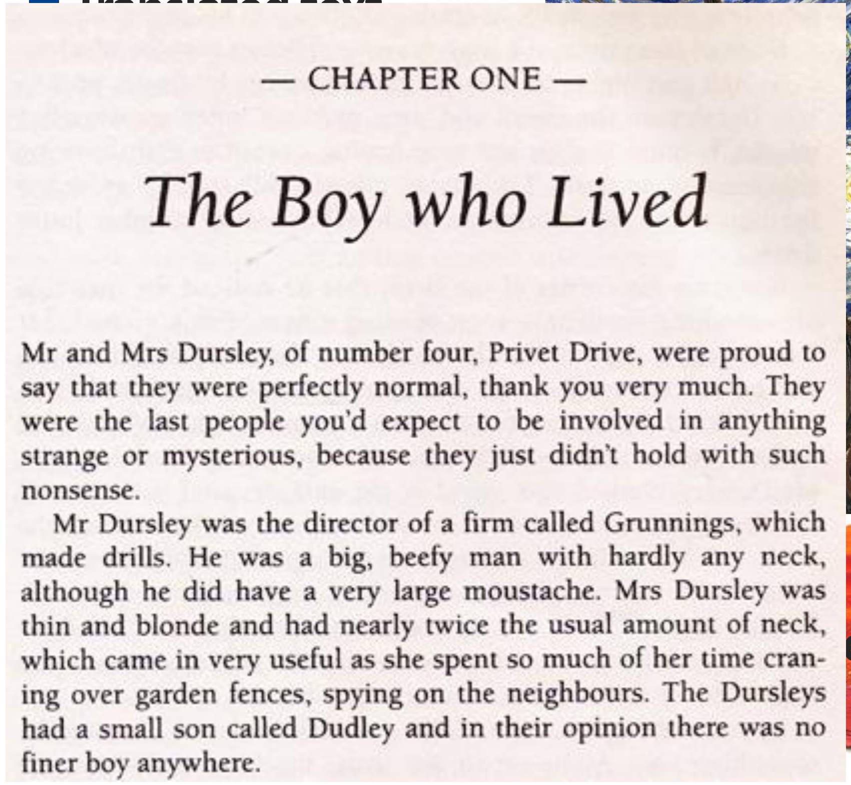
Vincent Van G

e Monet

r eyes





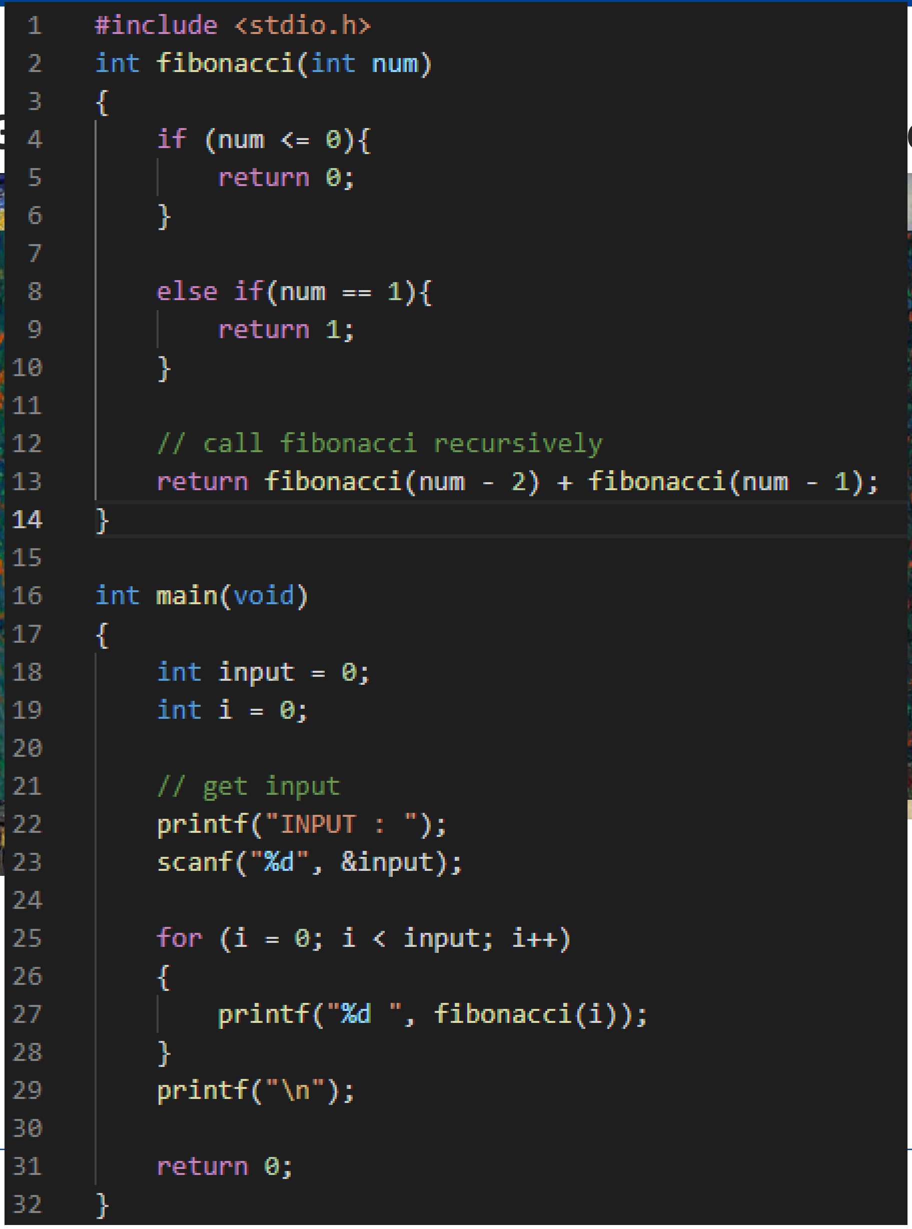


— CHAPTER ONE —

The Boy who Lived

Mr and Mrs Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much. They were the last people you'd expect to be involved in anything strange or mysterious, because they just didn't hold with such nonsense.

Mr Dursley was the director of a firm called Grunnings, which made drills. He was a big, beefy man with hardly any neck, although he did have a very large moustache. Mrs Dursley was thin and blonde and had nearly twice the usual amount of neck, which came in very useful as she spent so much of her time craning over garden fences, spying on the neighbours. The Dursleys had a small son called Dudley and in their opinion there was no finer boy anywhere.



```c
#include <stdio.h>
int fibonacci(int num)
{
    if (num <= 0){
        return 0;
    }

    else if(num == 1){
        return 1;
    }

    // call fibonacci recursively
    return fibonacci(num - 2) + fibonacci(num - 1);
}

int main(void)
{
    int input = 0;
    int i = 0;

    // get input
    printf("INPUT : ");
    scanf("%d", &input);

    for (i = 0; i < input; i++)
    {
        printf("%d ", fibonacci(i));
    }
    printf("\n");

    return 0;
}
```

Source Code Stylometry

Everyone code

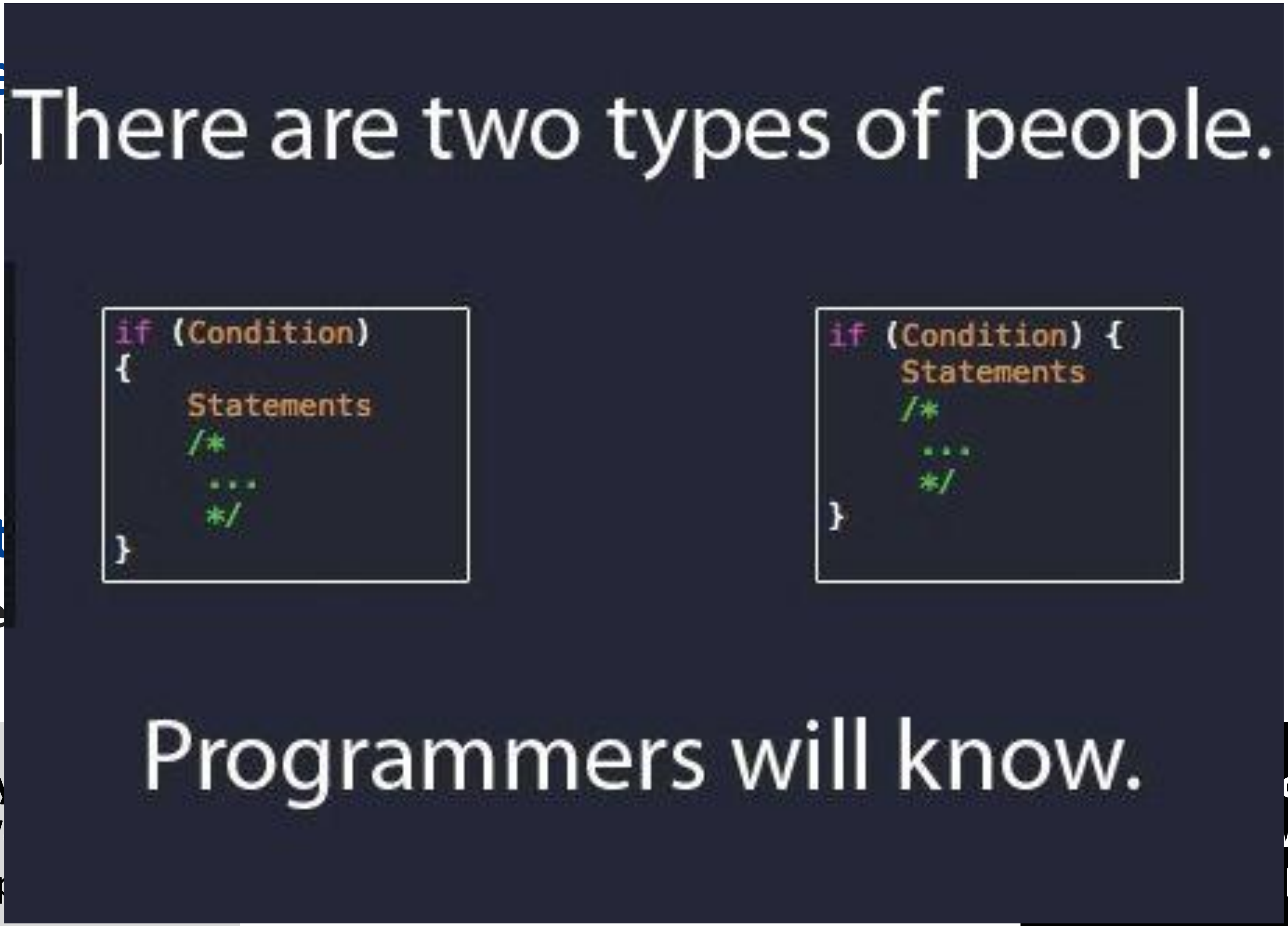
■ Example : do-whil

Code stylomet

■ More sophisticate

Easy
“Hello W
Examp

d or
World
lems


There are two types of people.

```
if (Condition)
{
    Statements
    /*
     ...
     */
}
```

```
if (Condition) {
    Statements
    /*
     ...
     */
}
```

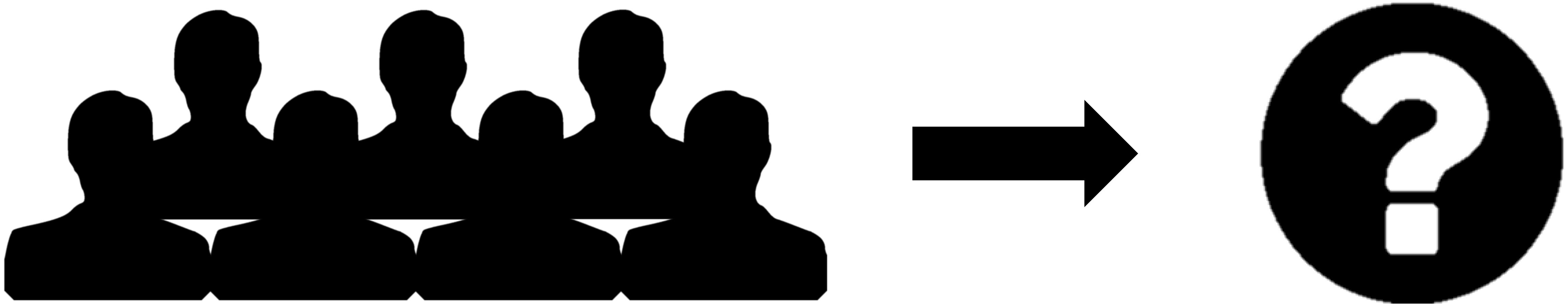
Programmers will know.

The Anonymity

The anonymity comes from indistinguishability in some (domain) set

- Better anonymity = indistinguishable in size of 100 elements than size of 10 elements in domain

Therefore, the de-anonymizing means figuring out someone within some set.



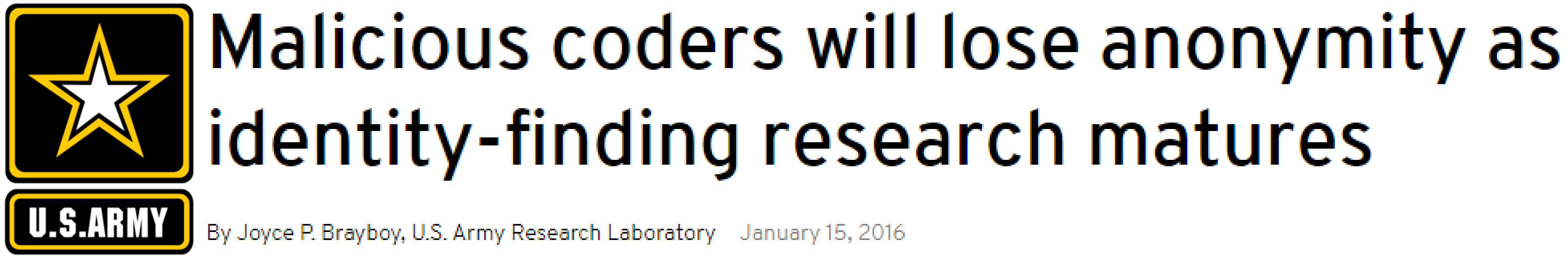
KAIST CySecLab

Why De-anonymizing is needed?

Real world necessity of de-anonymization of programmers

Why De-anonymizing is needed?

De-anonymizing can be used…



Malicious coders will lose anonymity as identity-finding research matures

By Joyce P. Brayboy, U.S. Army Research Laboratory January 15, 2016

In military, knowing about enemy and friendly is important!

De-anonymization scenarios

Programmer de-anonymization

- Who is Satoshi Nakamoto?
- Who wrote this malicious code?

Malicious programmers of D.P.R.K (North Korea)

WANTED
BY THE FBI
PARK JIN HYOK
KIM IL
JON CHANG HYOK
Conspiracy to Commit Wire Fraud and Bank Fraud; Conspiracy to Commit Computer-Related Fraud (Computer Intrusion)

Code plagiarize detection / Ghostwriting detection

- In case of Homework assignment
- Coding test for hiring developers

SAMSUNG SW Expert Academy

kakao

programmers

KAIST CySecLab

De-anonymizing Programmers

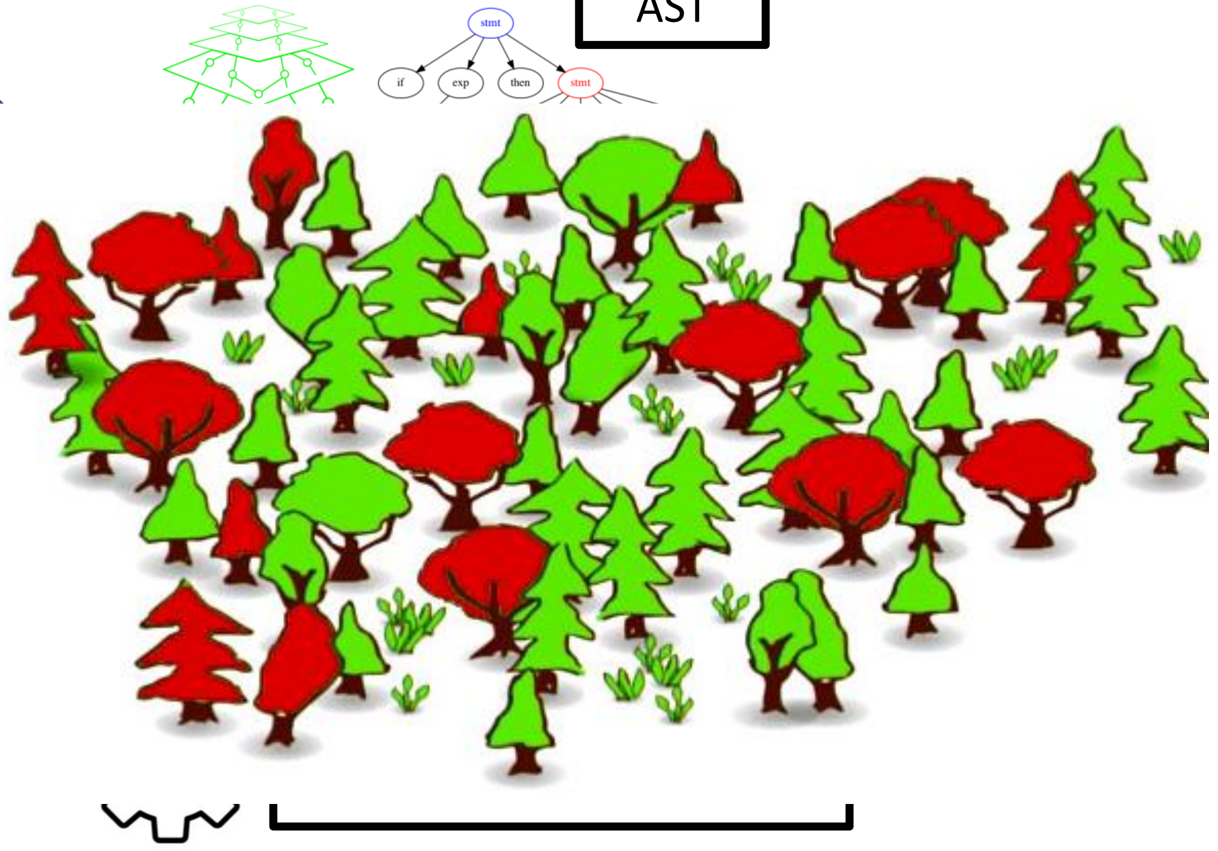
Fuzzy AST + Code Stylometry Feature Set + Machine Learning (classifier)

Identifying author of source

Method Overview

Google
code jam

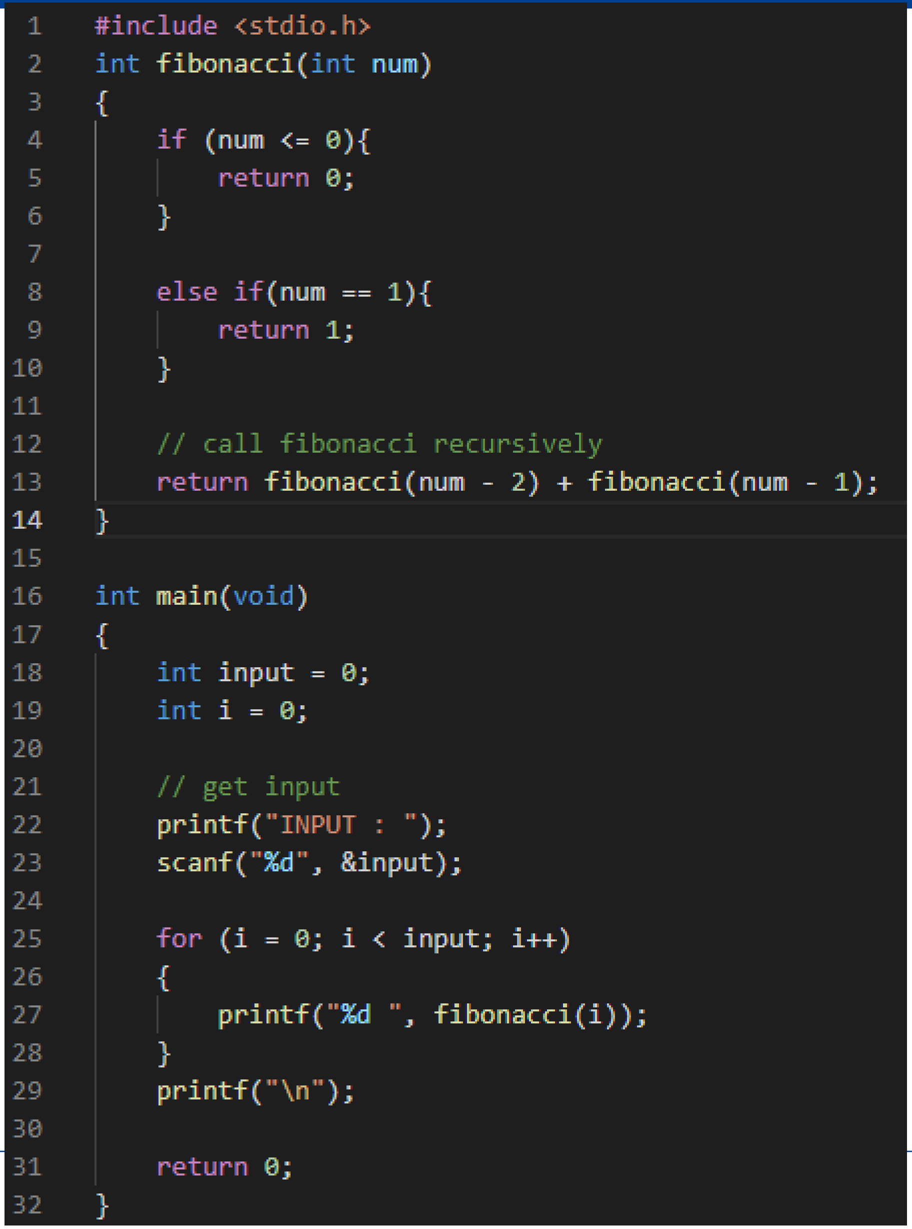
AST



RANDOM FOREST

Train & Classification
With majority rule

Code Stylometry Feature Set (CSFS)



```c
#include <stdio.h>
int fibonacci(int num)
{
    if (num <= 0){
        return 0;
    }

    else if(num == 1){
        return 1;
    }

    // call fibonacci recursively
    return fibonacci(num - 2) + fibonacci(num - 1);
}

int main(void)
{
    int input = 0;
    int i = 0;

    // get input
    printf("INPUT : ");
    scanf("%d", &input);

    for (i = 0; i < input; i++)
    {
        printf("%d ", fibonacci(i));
    }
    printf("\n");

    return 0;
}
```

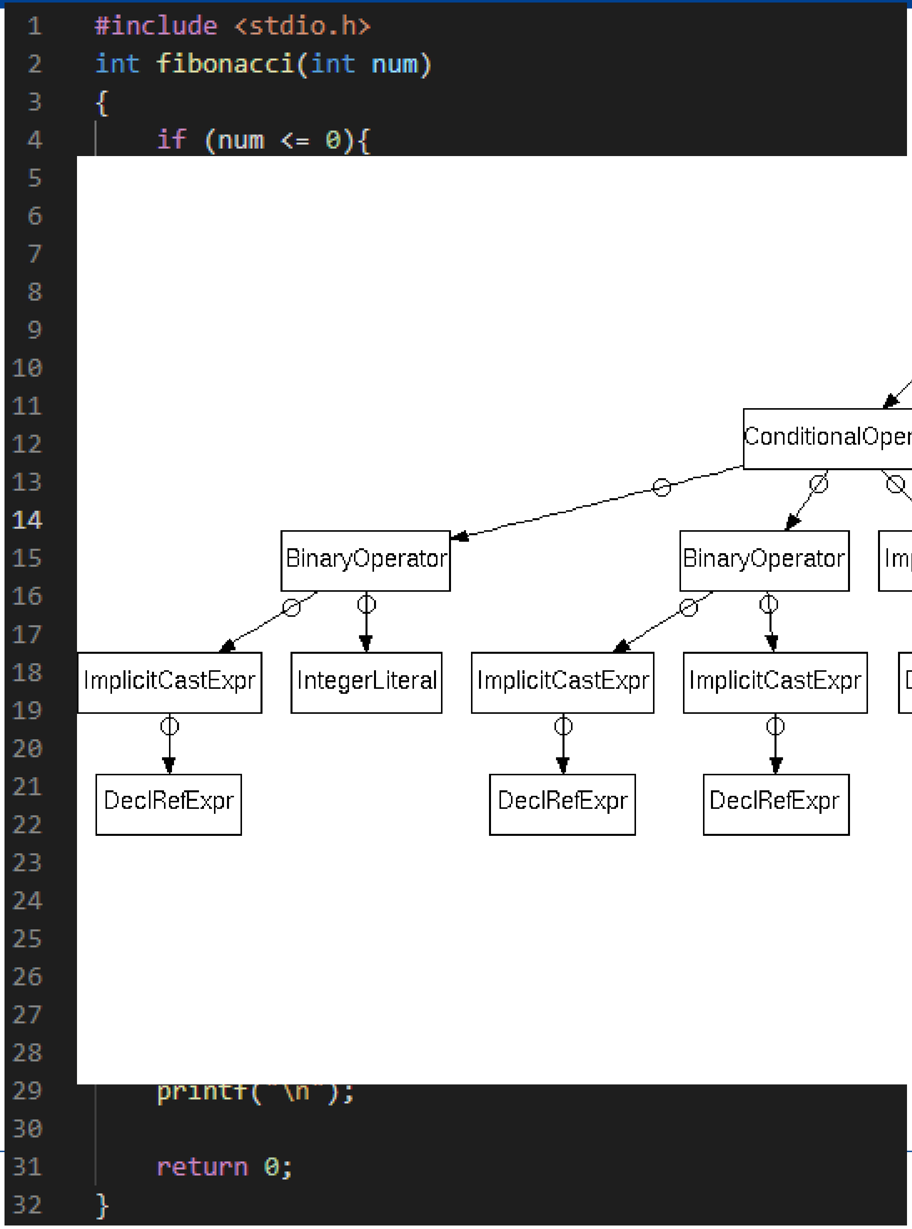
Lexical feature

- # of ternary operations
- # of comments
- # of literals
- ...

Layout feature

- # of tabs
- # of spaces
- # of empty line
- Bracelets before/after newline
- ...

Code Stylometry Feature Set (CSFS)



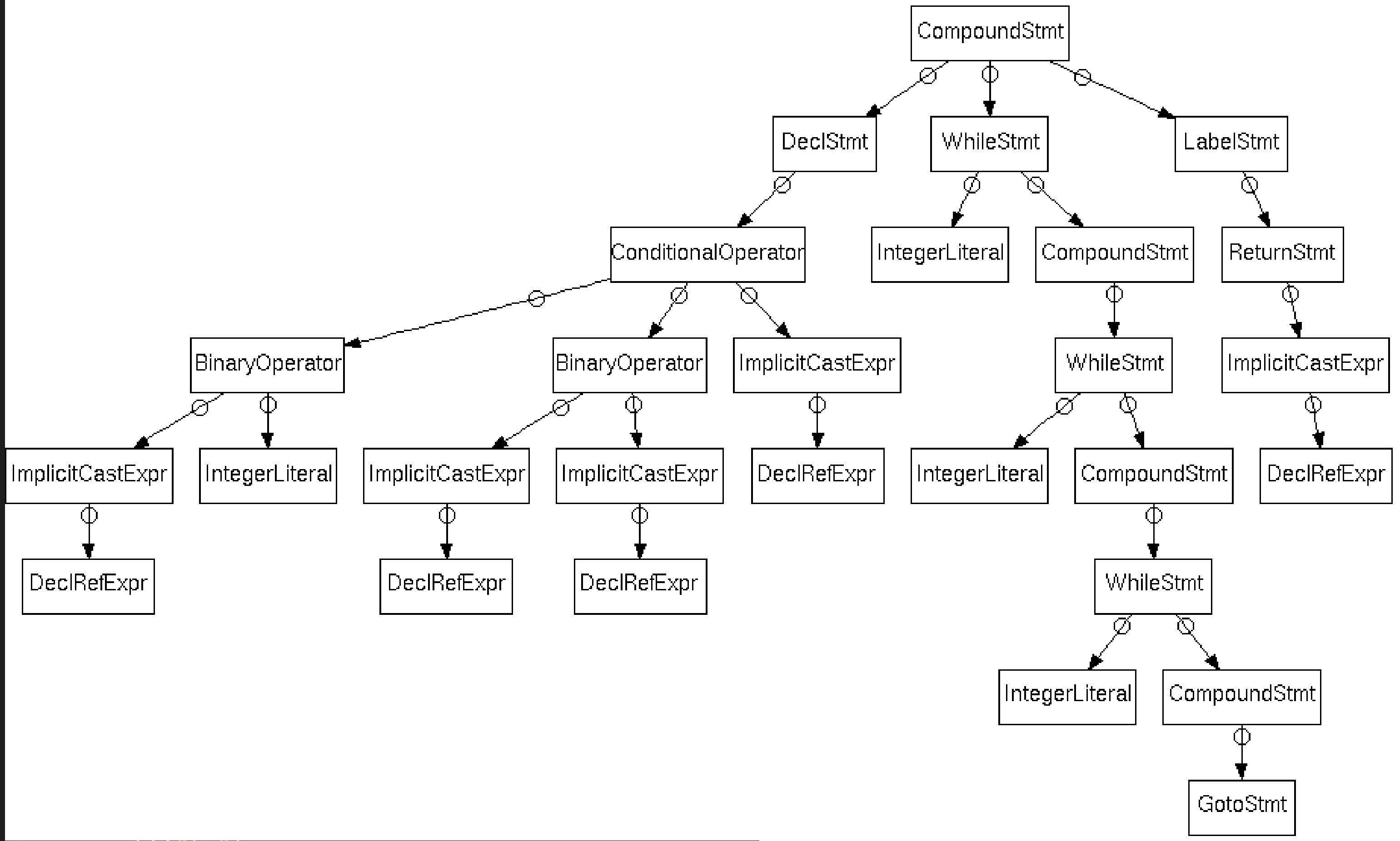
```
#include <stdio.h>
int fibonacci(int num)
{
    if (num <= 0){
    printf("\n");

    return 0;
}
```

Syntatic feature

e depth

rd



KAIST CySecLab

Evaluation

Can this method really de-anonymize programmers?

How about obfuscation?

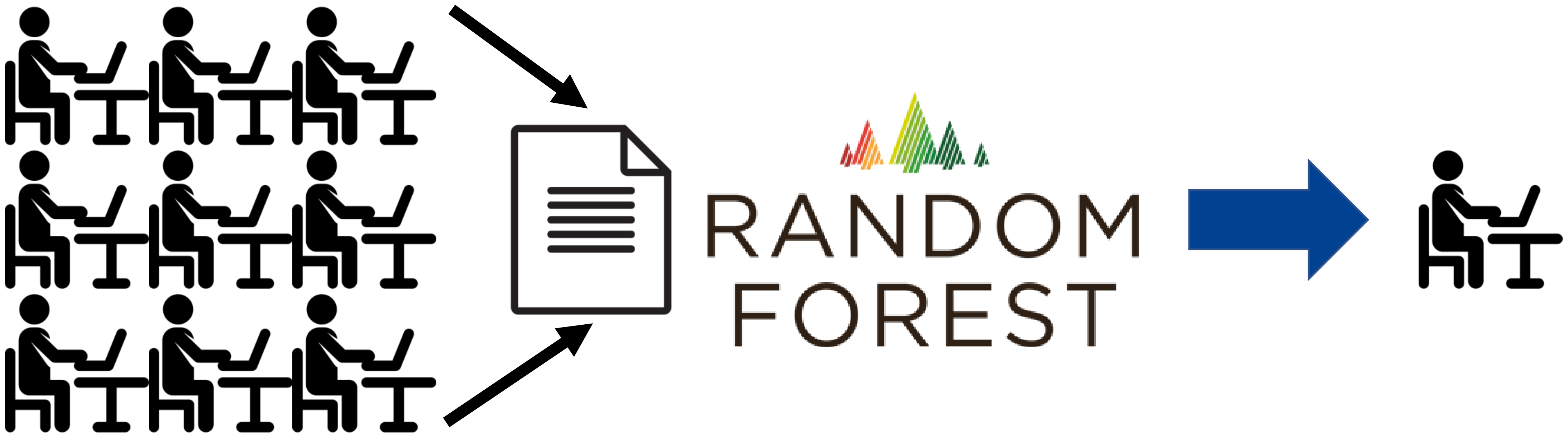
Is this method supports only C/C++?

Any limitation?

Authorship attribution experiments

It works?

- 94% accuracy in identifying 1,600 authors of 14,400 anonymous program files



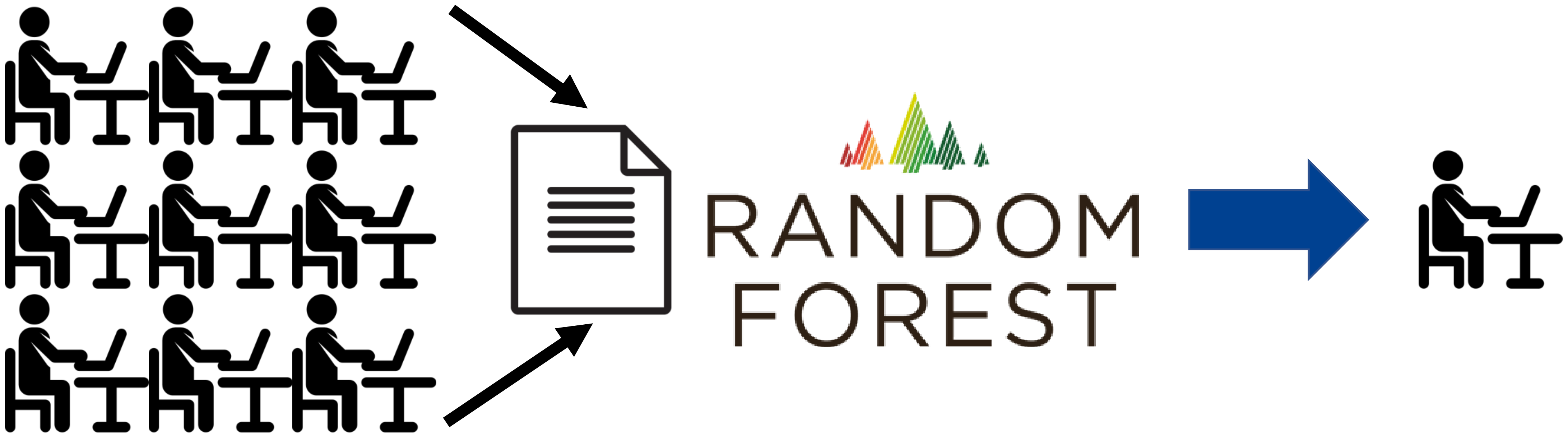
1600 authors

Classified programmer with in 14400 anonymous program file

Authorship attribution experiments

Is the coding style remains throughout years?

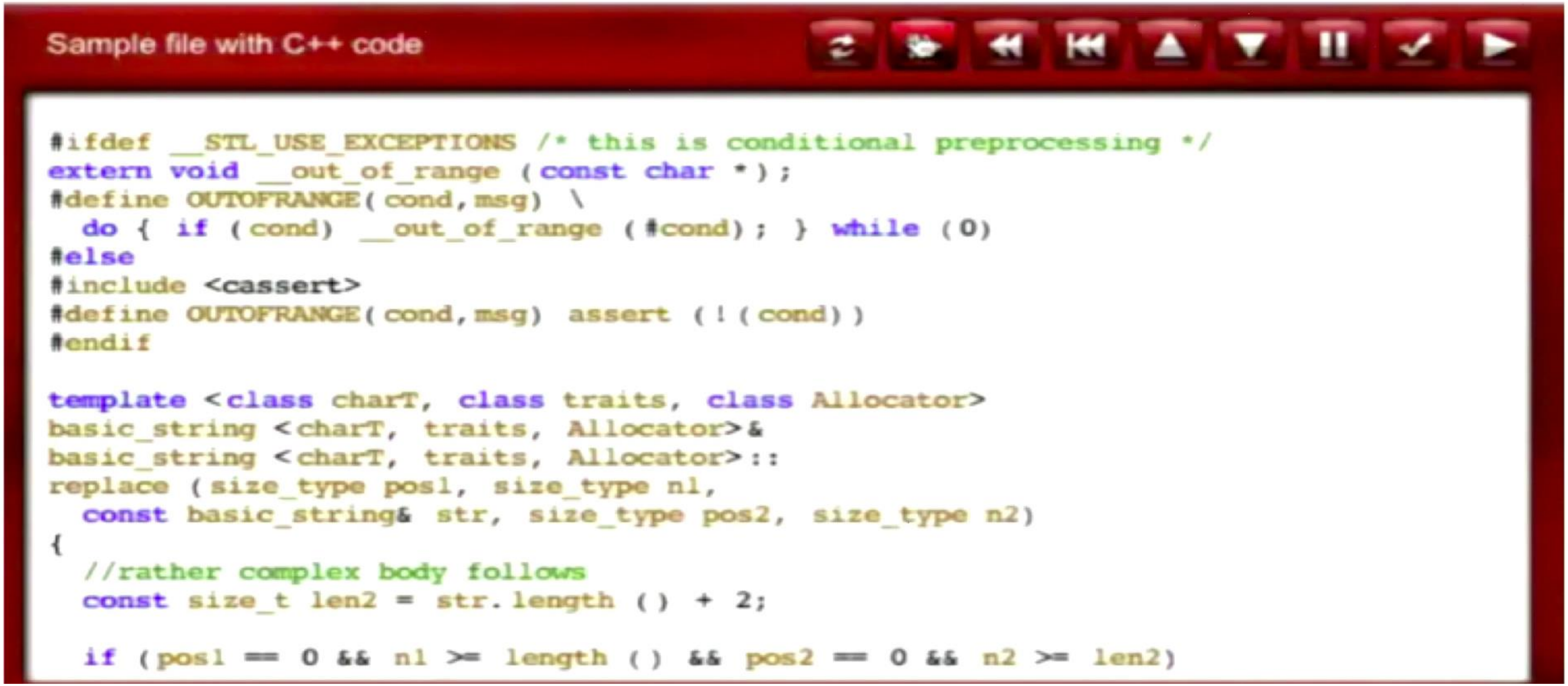
- 98% auccuracy, train and test in 2014 (same year)
- 96% accuracy, train on 2012, test on 2014



25 authors

Classified programmer with in 25 anonymous program file

Obfuscation – STUNNIX

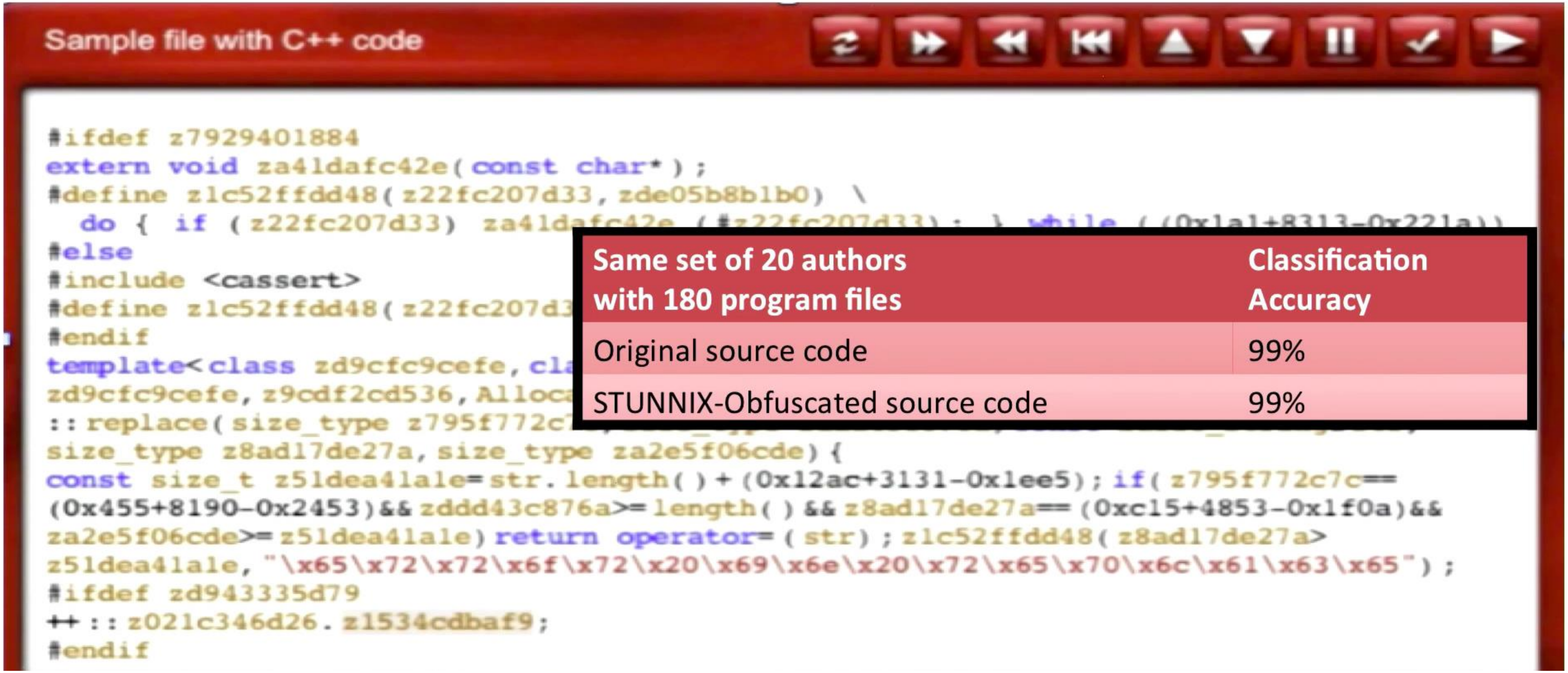


```
#ifdef __STL_USE_EXCEPTIONS /* this is conditional preprocessing */
extern void __out_of_range (const char *);
#define OUTOFRANGE(cond,msg) \
  do { if (cond) __out_of_range (#cond); } while (0)
#else
#include <cassert>
#define OUTOFRANGE(cond,msg) assert (!(cond))
#endif

template <class charT, class traits, class Allocator>
basic_string <charT, traits, Allocator>&
basic_string <charT, traits, Allocator>::
replace (size_type pos1, size_type n1,
  const basic_string& str, size_type pos2, size_type n2)
{
  //rather complex body follows
  const size_t len2 = str.length () + 2;

  if (pos1 == 0 && n1 >= length () && pos2 == 0 && n2 >= len2)
```

Obfuscation – STUNNIX



```
#ifdef z7929401884
extern void za41dafc42e(const char*);
#define z1c52ffdd48(z22fc207d33,zde05b8b1b0) \
  do { if (z22fc207d33) za41dafc42e (#z22fc207d33); } while ((0x1a1+8313-0x221a))
#else
#include <cassert>
#define z1c52ffdd48(z22fc207d3
#endif
template<class zd9cfc9cefe, cla
zd9cfc9cefe, z9cdf2cd536, Alloca
::replace(size_type z795f772c7
size_type z8ad17de27a, size_type za2e5f06cde){
const size_t z51dea41a1e=str.length()+(0x12ac+3131-0x1ee5);if(z795f772c7c==
(0x455+8190-0x2453)&& zddd43c876a>=length()&& z8ad17de27a==(0xc15+4853-0x1f0a)&&
za2e5f06cde>=z51dea41a1e) return operator=(str);z1c52ffdd48(z8ad17de27a>
z51dea41a1e,"\x65\x72\x72\x6f\x72\x20\x69\x6e\x20\x72\x65\x70\x6c\x61\x63\x65");
#ifdef zd943335d79
++::z021c346d26.z1534cdbaf9;
#endif
```

Same set of 20 authors with 180 program files	Classification Accuracy
Original source code	99%
STUNNIX-Obfuscated source code	99%

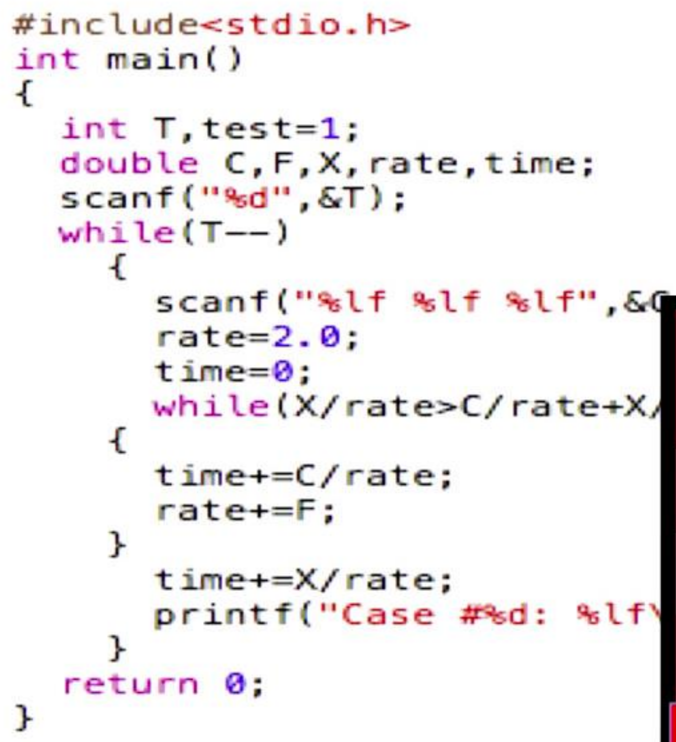
KAIST CySecLab

Obfuscation – TIGRESS

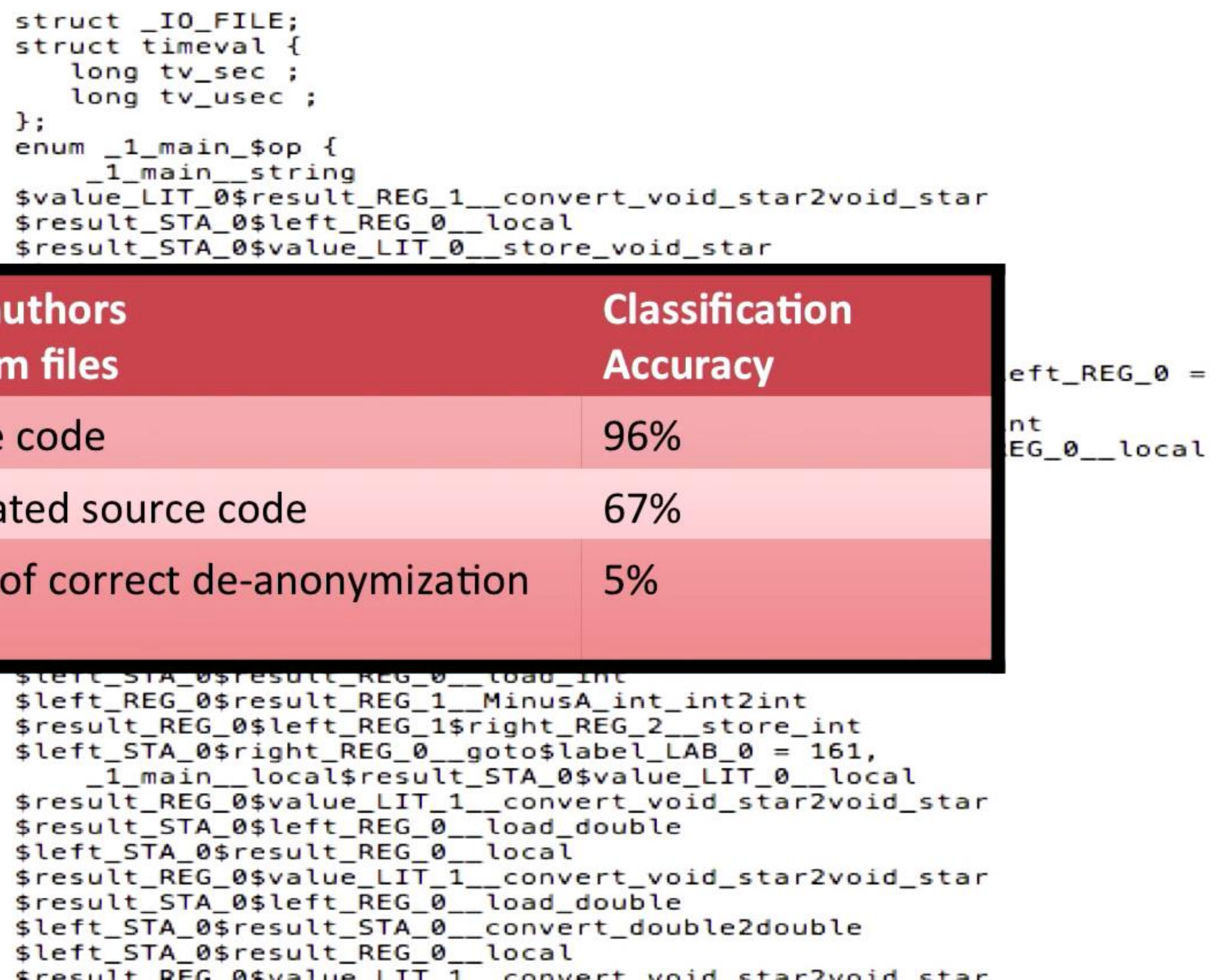
```c
#include<stdio.h>
int main()
{
  int T,test=1;
  double C,F,X,rate,time;
  scanf("%d",&T);
  while(T--)
    {
      scanf("%lf %lf %lf",&C,&F,&X);
      rate=2.0;
      time=0;
      while(X/rate>C/rate+X/(rate+F))
    {
      time+=C/rate;
      rate+=F;
    }
      time+=X/rate;
      printf("Case #%d: %lf\n",test++,time);
    }
  return 0;
}
```

```c
struct _IO_FILE;
struct timeval {
   long tv_sec ;
   long tv_usec ;
};
enum _1_main_$op {
    _1_main__string
$value_LIT_0$result_REG_1__convert_void_star2void_star
$result_STA_0$left_REG_0__local
$result_STA_0$value_LIT_0__store_void_star
$left_STA_0$right_STA_1__local
$result_STA_0$value_LIT_0__convert_void_star2void_star
$left_STA_0$result_REG_0__local
$result_REG_0$value_LIT_1__convert_void_star2void_star
$result_STA_0$left_REG_0__store_void_star$right_STA_0$left_REG_0 =
46,
    _1_main__local$result_REG_0$value_LIT_1__constant_int
$result_STA_0$value_LIT_0__store_int$right_STA_0$left_REG_0__local
$result_STA_0$value_LIT_0__convert_void_star2void_star
$left_STA_0$result_REG_0__string
$value_LIT_0$result_REG_1__convert_void_star2void_star
$result_STA_0$left_REG_0__store_void_star
$right_STA_0$left_REG_0__local
$result_REG_0$value_LIT_1__convert_void_star2void_star
$result_STA_0$left_REG_0 = 44,
    _1_main__convert_void_star2void_star
$left_STA_0$result_REG_0__load_int
$left_REG_0$result_REG_1__MinusA_int_int2int
$result_REG_0$left_REG_1$right_REG_2__store_int
$left_STA_0$right_REG_0__goto$label_LAB_0 = 161,
    _1_main__local$result_STA_0$value_LIT_0__local
$result_REG_0$value_LIT_1__convert_void_star2void_star
$result_STA_0$left_REG_0__load_double
$left_STA_0$result_REG_0__local
$result_REG_0$value_LIT_1__convert_void_star2void_star
$result_STA_0$left_REG_0__load_double
$left_STA_0$result_STA_0__convert_double2double
$left_STA_0$result_REG_0__local
$result_REG_0$value_LIT_1__convert_void_star2void_star
```

Obfuscation – TIGRESS



```
#include<stdio.h>
int main()
{
   int T,test=1;
   double C,F,X,rate,time;
   scanf("%d",&T);
   while(T--)
      {
         scanf("%lf %lf %lf",&C
         rate=2.0;
         time=0;
         while(X/rate>C/rate+X/
      {
         time+=C/rate;
         rate+=F;
      }
         time+=X/rate;
         printf("Case #%d: %lf\
      }
   return 0;
}
```



```
struct _IO_FILE;
struct timeval {
   long tv_sec ;
   long tv_usec ;
};
enum _1_main_$op {
   _1_main__string
$value_LIT_0$result_REG_1__convert_void_star2void_star
$result_STA_0$left_REG_0__local
$result_STA_0$value_LIT_0__store_void_star
eft_REG_0 =
nt
EG_0__local
$left_REG_0$result_REG_1__MinusA_int_int2int
$result_REG_0$left_REG_1$right_REG_2__store_int
$left_STA_0$right_REG_0__goto$label_LAB_0 = 161,
   _1_main__local$result_STA_0$value_LIT_0__local
$result_REG_0$value_LIT_1__convert_void_star2void_star
$result_STA_0$left_REG_0__load_double
$left_STA_0$result_REG_0__local
$result_REG_0$value_LIT_1__convert_void_star2void_star
$result_STA_0$left_REG_0__load_double
$left_STA_0$result_STA_0__convert_double2double
$left_STA_0$result_REG_0__local
```

Same set of 20 authors with 180 program files	Classification Accuracy
Original C source code	96%
TIGRESS-Obfuscated source code	67%
Random chance of correct de-anonymization	5%

Method Generalization - Python

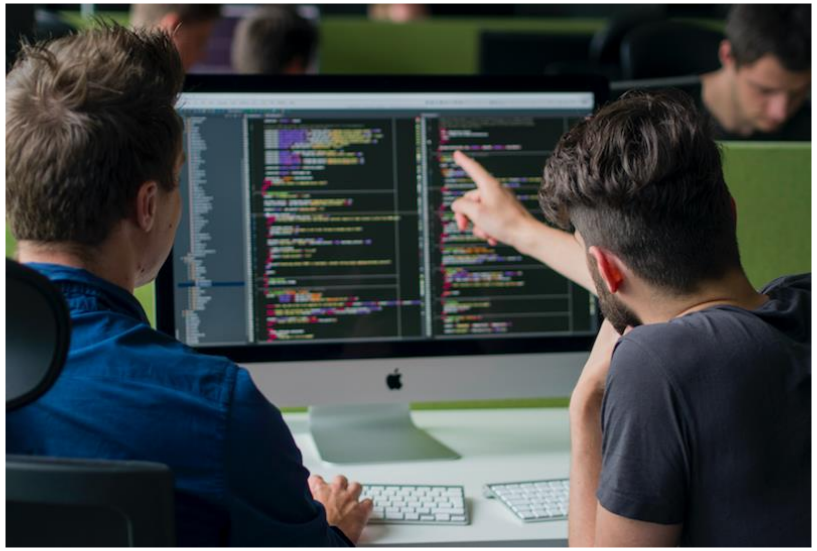
Using 'only' Python equivalents of syntactic features

Application	Programmers	Instances	Result
Python programmer de-anonymization	229	2061	53.9%
Top-5 relaxed classification	229	2,061	75.7%
Python programmer de-anonymization	23	207	87.9%
Top-5 relaxed classification	23	207	99.5%

Limitations

Multi-authorship problem

- Source code cloning
- Ex) Use of Stack-Overflow code
- Pair coding example



Coding style normalized problem

- Open-source projects, like Linux kernel, and enterprise internal projects have strict coding style

Conclusion

Conclusion

First principled use of syntactic features to investigate style in source code

Reach 94% accuracy in classifying 1600 authors in GCJ

For future research, going binary to de-anonymizing programmers!

KAIST CySecLab

Related works

Source code stylometry research timeline

Related works

Works	# Author	LOC	Features	Result
MacDonell	7	148	Lexical & Layout	88.0%
Frantzeskou	8	145	Lexical & Layout	100.0%
Elenbogen	12	100	Lexical & Layout	74.7%
Shevertalov	20	N/A	Lexical & Layout	80.0%
Frantzeskou	30	172	Lexical & Layout	96.9%
Ding	46	N/A	Lexical & Layout	67.2%
Ours	35	68	**Lexical & Layout & Syntactic**	100.0%
	250	77		98.0%
	1600	70		93.6%

KAIST CySecLab

Related works

Large-Scale and Language-Oblivious Code Authorship Identification (CCS '18)

- Proposes deep learning based code authorship identification system for code authorship attribution.
- Works on mixed language environments (JAVA/C++, Python/C++).

Misleading Authorship Attribution of Source Code using Adversarial Learning (USENIX '19)

- Attacking authorship attribution of source code using machine learning approach with preserving semantic structure.

KAIST CySecLab

Related Questions

Related Questions

김한나(Best)

- **They define Code Stylometry Feature Set consists of Lexical, Layout, Syntactic features. I think this is defined manually by the authors. I think there might be more important features for classification and it is time costly. Is there any method to find those features automatically?**
  - Yes. Maybe. I think the Lexical and Layout feature can automatically extracted by checking 'diff' of parsed (normalized) source code and original source code.

이용화

- **Can other features like program usage (ex. history like command usage, process occupancy, idle time between various processes) be the features to discriminate individuals? - And how and where can this technology be used?**
  - I think this work will be very hard to collect data. Also, I cannot imagine that abstracting program usage behavior. Therefore, I think it is hard to use program usage feature to discriminate individuals.

Related Questions

김경태, 장준하

- **(김경태) I wonder how the detecting the stylistic features works in the binary code?**
- **(장준하) I wonder if we can use intermediate language (IL) to de-anonymize the program. Because many data abstracted when compilation to binary**
  - After this work, the authors checked binary de-anonymization. To do that, they used decompiled code and disassembled code. Also, AST from decompiled code and Control Flow Graph (CFG) from disassembled code features were used.
  - Also, the generalization slides showed de-anonymization is possible even only the syntactic features used.

안준호

- **For software forensic part, is there sufficient data for malware developer? I think if there isn't, then this application will not work because it is supervised learning.**
  - I think the use of this technique provides another option for software forensic, not silver bullet.
    When software forensic, there is no formal way to investigate.
    Like "Sony Pictures hack" incident, the DPRK used sophisticated malware that hard to identify origin.
    Therefore, the investigators specified origin as DPRK by showing the compiler language option was Korean.

VirusTotal

KAIST CySecLab

Thanks!

Taehyeon Lee