



## OBA2: An Onion Approach to Binary Code Authorship Attribution

*By*

**Saed Alrabaae, Noman Saleem, Stere Preda,  
Lingyo Wang, Mourad Debbabi**

*Presented At*

The Digital Forensic Research Conference  
**DFRWS 2014 EU** Amsterdam, NL (May 7<sup>th</sup> - 9<sup>th</sup>)

DFRWS is dedicated to the sharing of knowledge and ideas about digital forensics research. Ever since it organized the first open workshop devoted to digital forensics in 2001, DFRWS continues to bring academics and practitioners together in an informal environment. As a non-profit, volunteer organization, DFRWS sponsors technical working groups, annual conferences and challenges to help drive the direction of research and development.

**<http://dfrws.org>**

# *OBA2: An Onion approach to Binary code Authorship Attribution*

Saed Alrabaee, Noman Saleem, Stere Preda, Lingyu Wang, Mourad Debbabi

Computer Security Laboratory, Concordia Institute for Information Systems  
Engineering, Concordia University

Saed Alrabaee

May, 08, 2014

- Introduction
- Challenges
- Related Work
- Proposed Solution (OBA2)
- Implementation
- Results
- Conclusion

- What is OBA2?
  - Multilayer approach to binary authorship attribution
- Why do we need OBA2?
  - Authorship Attribution
  - Reverse Engineering
- How does OBA2 work?
  - Code Filtration
  - Syntax-based attribution
  - Semantic-based attribution

# Introduction

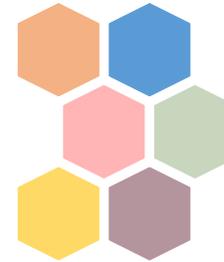


*"On the Internet, nobody knows you're a dog."*

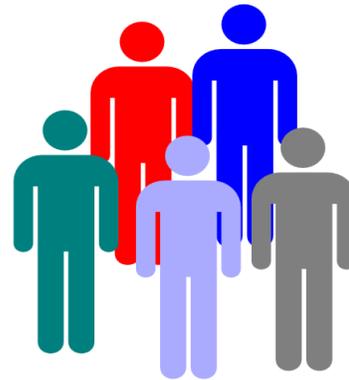
.bin  
.dll  
.so  
.lib  
.asm

Abstraction

Stylometric  
Features



Mapping

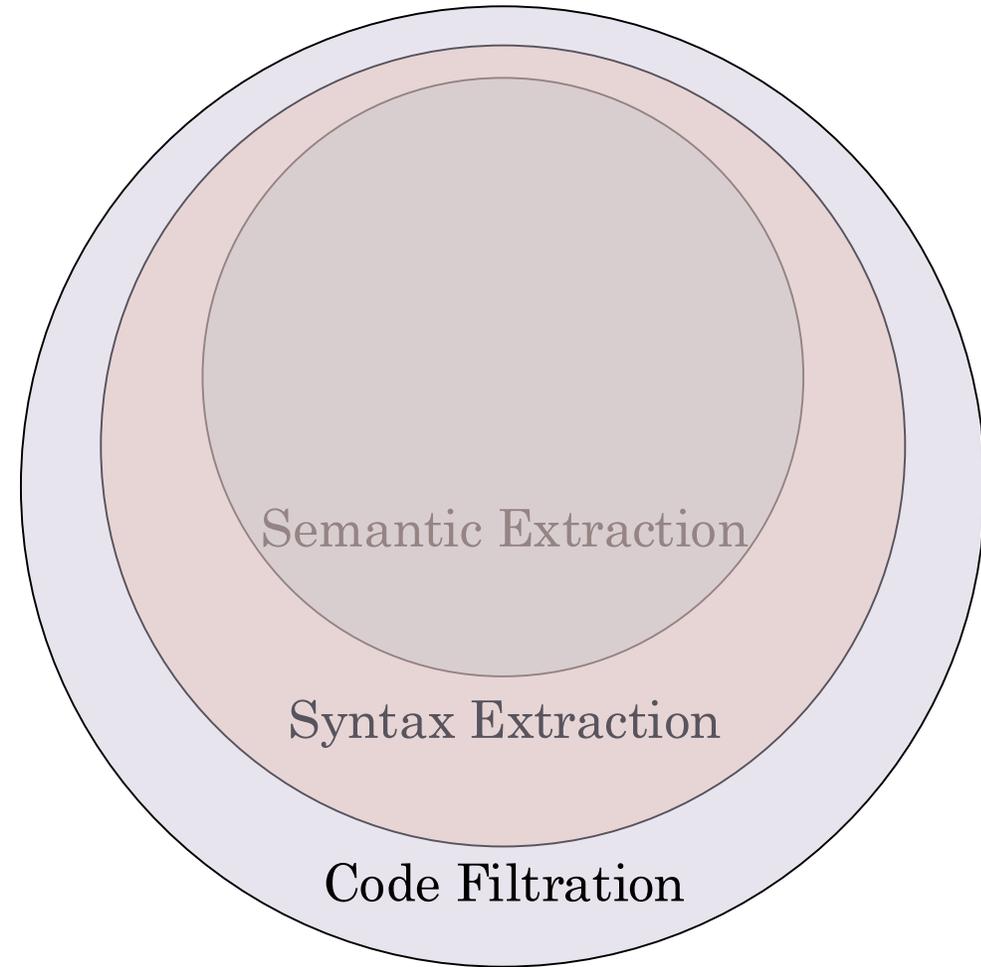


- Tool validation
  - Candidate methods (statistics analysis) – same results
  - Style vs. functionality
- “Good” validation dataset
  - Code reuse
  - Several authors per file
- Heterogeneous dev. environments

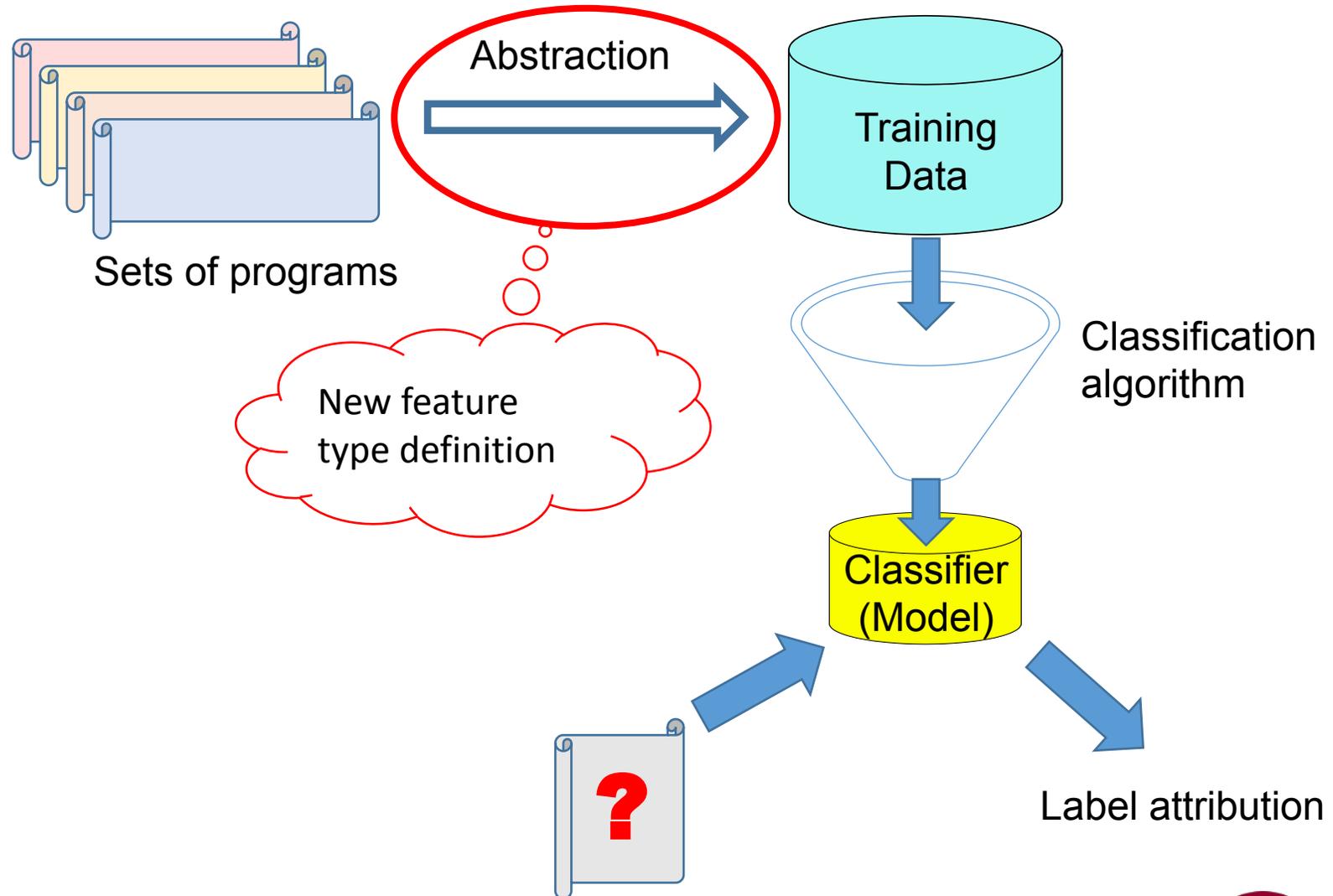
- Rosenblum et al extracts features using five pre-defined templates
  - Idioms
    - ✓ short sequences of instructions
  - Graphlets
    - ✓ 3-node subgraph of the Control Flow Graph (CFG)
  - Supergraphlets
    - ✓ collapsing and merging neighbor nodes of the CFG
  - Libcalls and Call Graphlets
    - ✓ function names of imported libraries
  - N-grams:

# Proposed Solution (OBA2)

- Why do we choose onion?

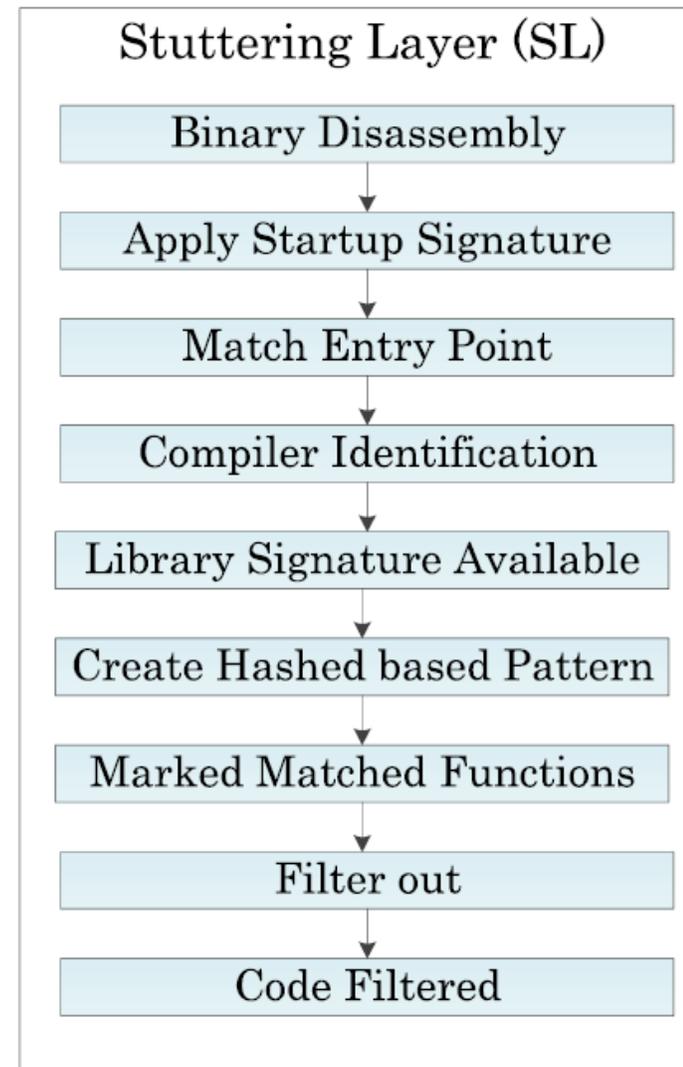


# Proposed Solution (OBA2)



# Proposed Solution (OBA2)

- Stuttering Layer



# Proposed Solution (OBA2)

## Code Analysis Layer

STL					
Category	Class	Exact	Inexact	No	
Container	Pair	x		2	
	Vector				
	List		x	1	
	Queue				
	Stack		x		1
	Hash sets				
Algorithm	Valarray		x	2	
	Sort				
	Depth-first search		x	1	
	Dijkstra's algorithm				
	Minimum spanning Breadth-first search			x	1
Socket	Data transfer mech				
	Options management		x	2	
	Network addressing	x		3	
I/O	Connection setup	x		3	
	stdin	x		3	
	printf	x		4	
	scanf	x		2	
	puts	x		1	
	gets				
Encryption	getche		x	1	
	TEA				
	RC4		x	2	
	AES				
	MD5				
	RSA				

# Proposed Solution (OBA2)

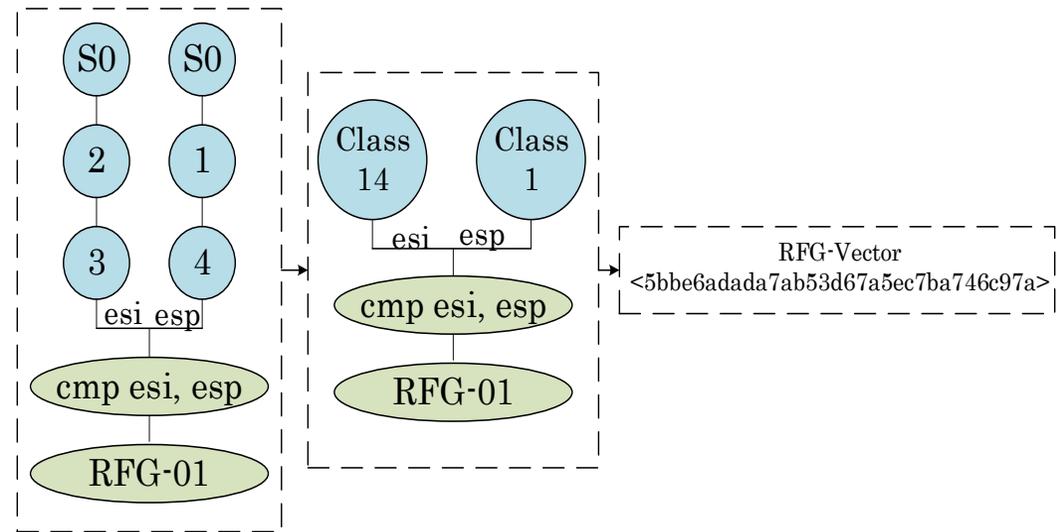
- Register Flow  
Analysis Layer

Class	Arithmetic	Logical	Generic	Stack
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5	1	1	0	0
6	1	0	1	0
7	1	0	0	1
8	1	1	1	0
9	1	1	0	1
10	1	0	1	0
11	1	1	1	1
12	0	1	1	0
13	0	1	0	1
14	0	0	1	1
15	0	1	1	1

- Register Flow Graph
  - “CMP op1, op2”
  - RFG root encodes the operands combination
  - RFG leaves encode instruction classes
  - < root, color, dissim. >
- ASM instruction and register classes

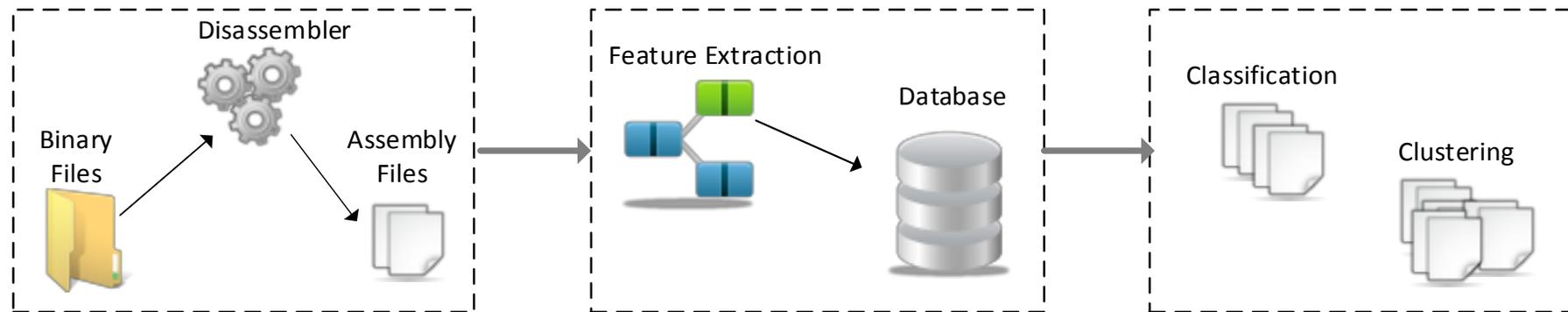
# Proposed Solution (OBA2)

```
1.  sub    esp, 0D8h
2.  push  esi
3.  mov    esi, esp
4.  add    esp, 8
5.  cmp    esi, esp
```



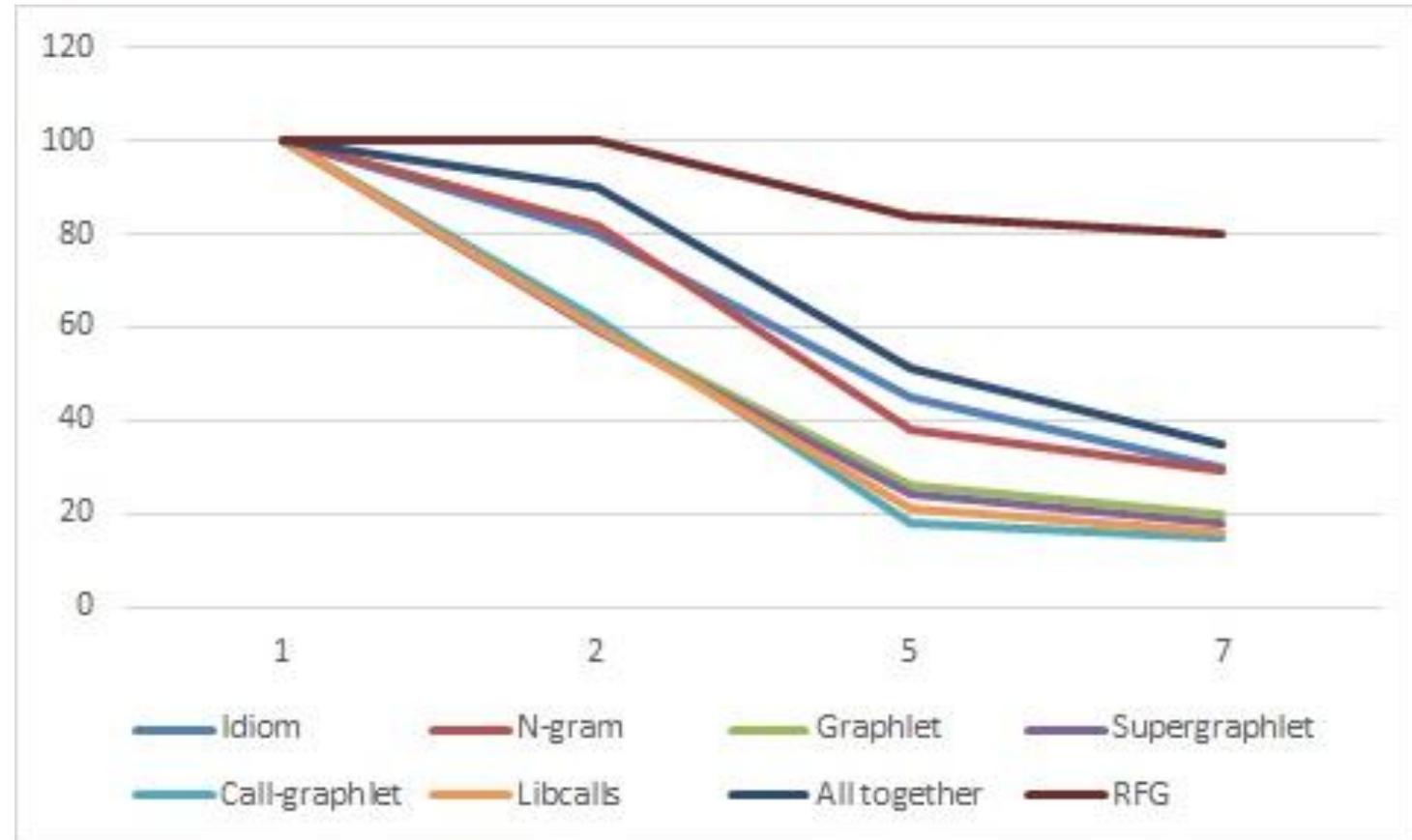
# Implementation

- Current state of implementation
  - PostgreSQL & RFG extraction algorithm
  - Preliminary results
    - e.g., for 7 authors x 10 programs
    - 3044 singletons, 10% different roots



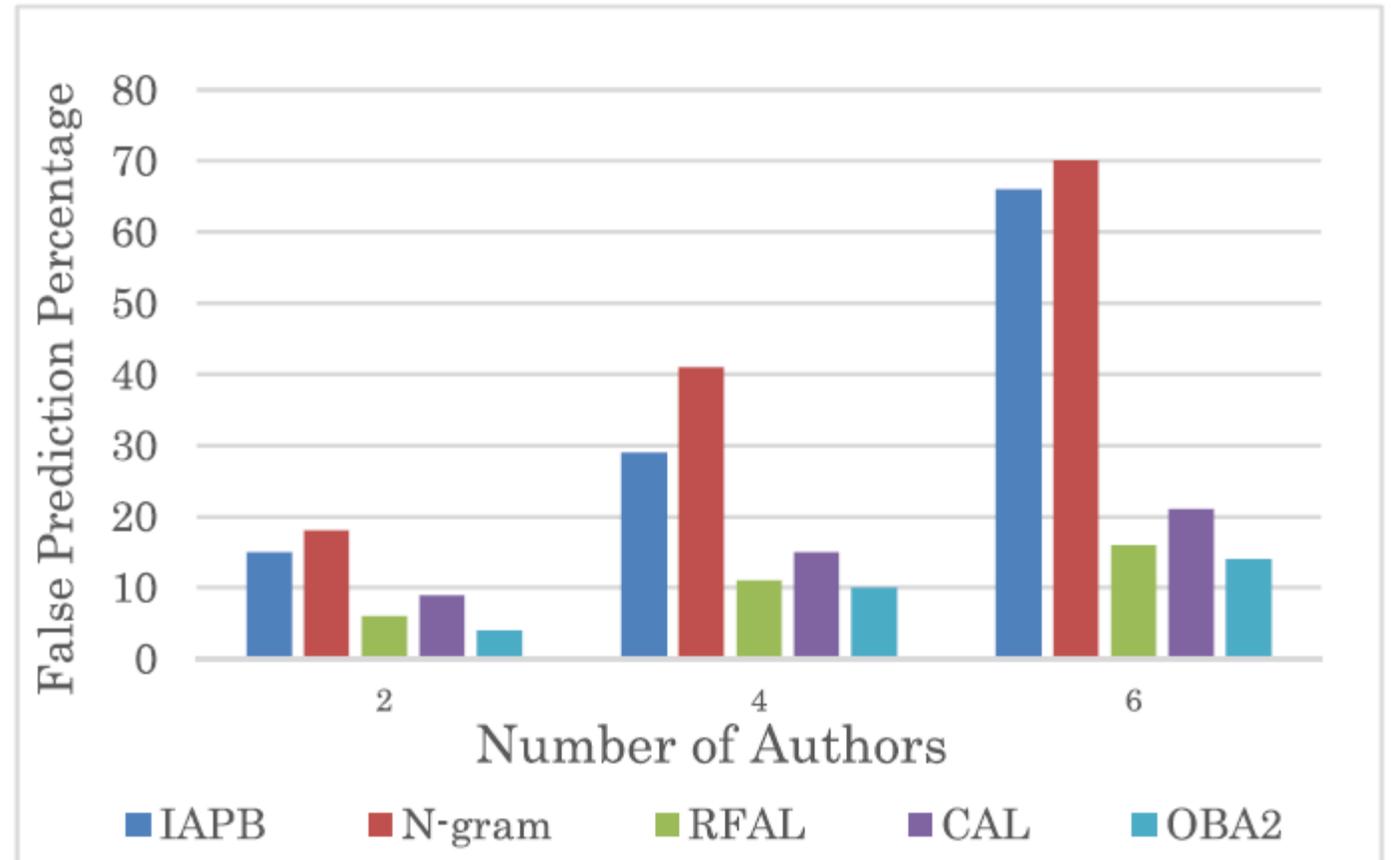
# Results

- Accuracy of Features



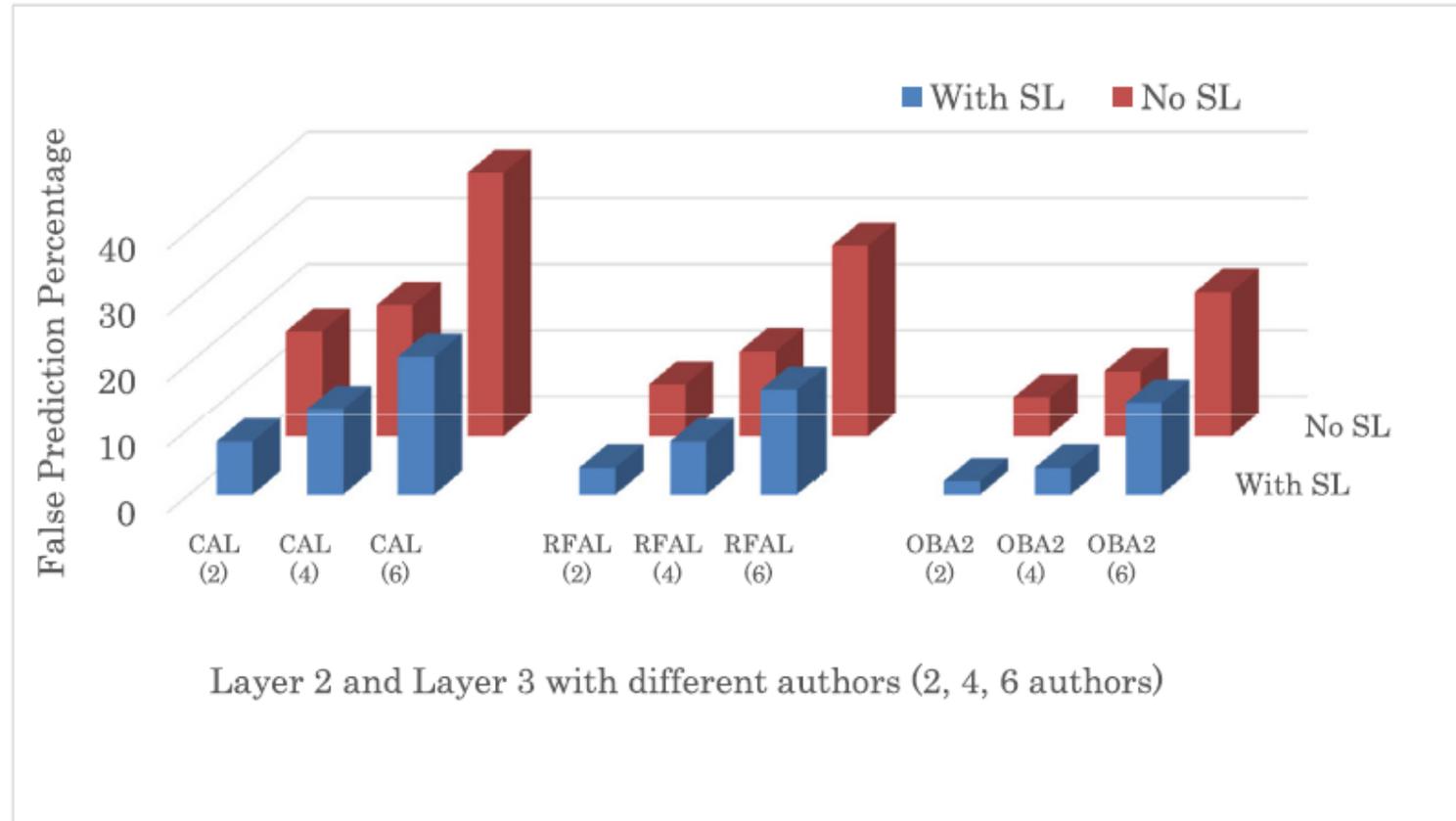
# Results

- False Predication



# Results

- Effect of Code Filtration



- Authorship attribution framework
  - Code filtration
  - Semantic signatures
  - Plugins for other feature types
- Limitation of Rosenblum et al
  - Filtered Code
  - Meaningful Features
  - Functionality related features
- Future work
  - RFG theory improvement (e.g., CFG)?

# Thank You

