MobileFindr: Function Similarity Identification for Reversing Mobile Binaries

Yibin Liao, Ruoyan Cai, Guodong Zhu, Yue Yin, Kang Li

UNIVERSITY OF GEORGIA

Reverse Engineering

- **The process of taking a software program's binary code and recreating it so as to trace it back to the original source code**

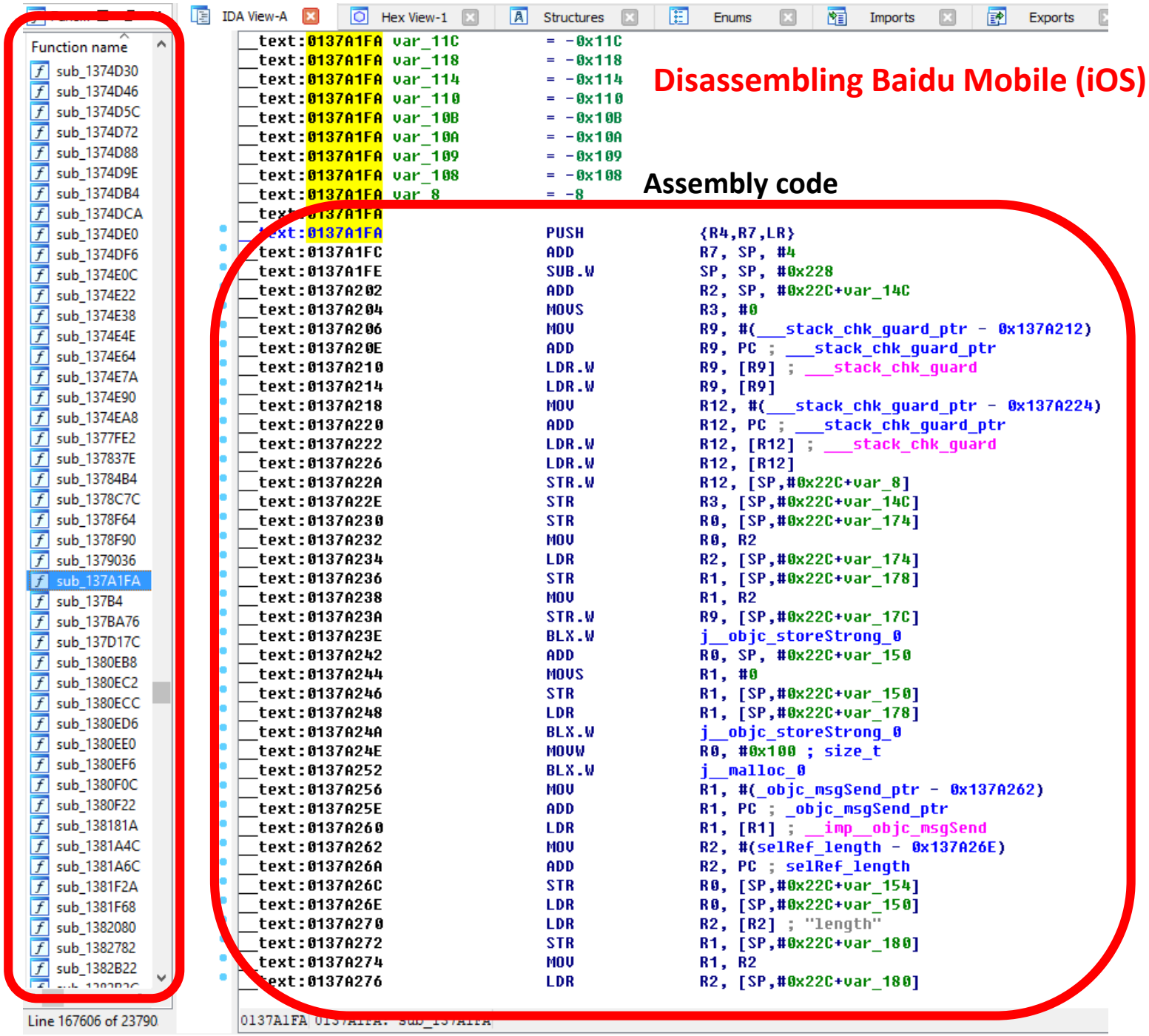
Reverse Engineering

- **Purpose**
  - Find security flaws
  - Test code compatibility
  - Enhance new features without source
  - Understand the design of malicious code

Tools for Reversing Mobile Apps

lldb/gdb	• **Debugging iOS/Android native code**
Apktools / Baksmali	• **Convert Android executable (DEX) to human-readable dalvik bytecode**
IDA Pro	• **Combines debugger, disassembler and decompiler**
Etc.	

Function Window



Disassembling Baidu Mobile (iOS)

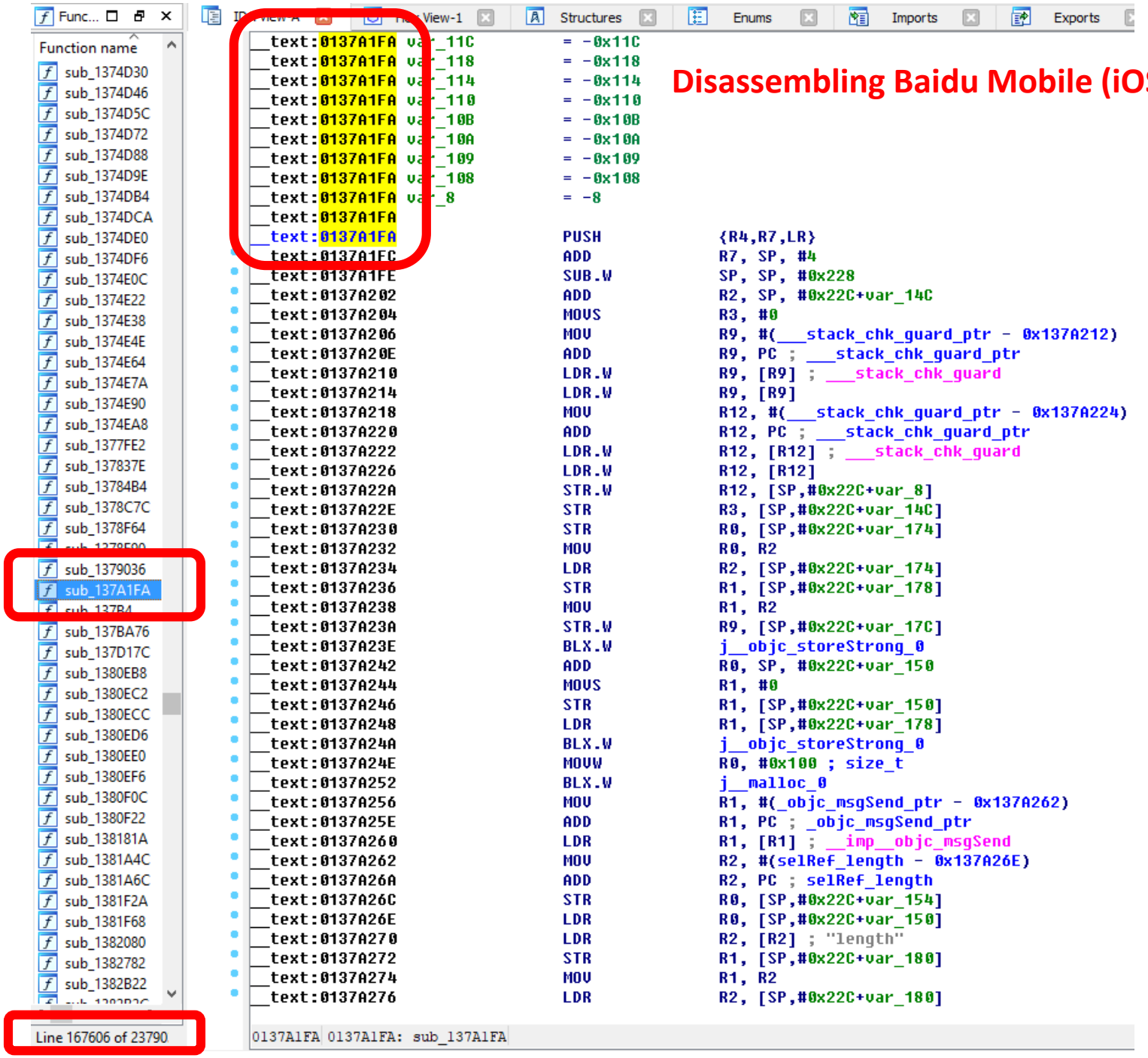
Assembly code

Disassembling Baidu Mobile (iOS)

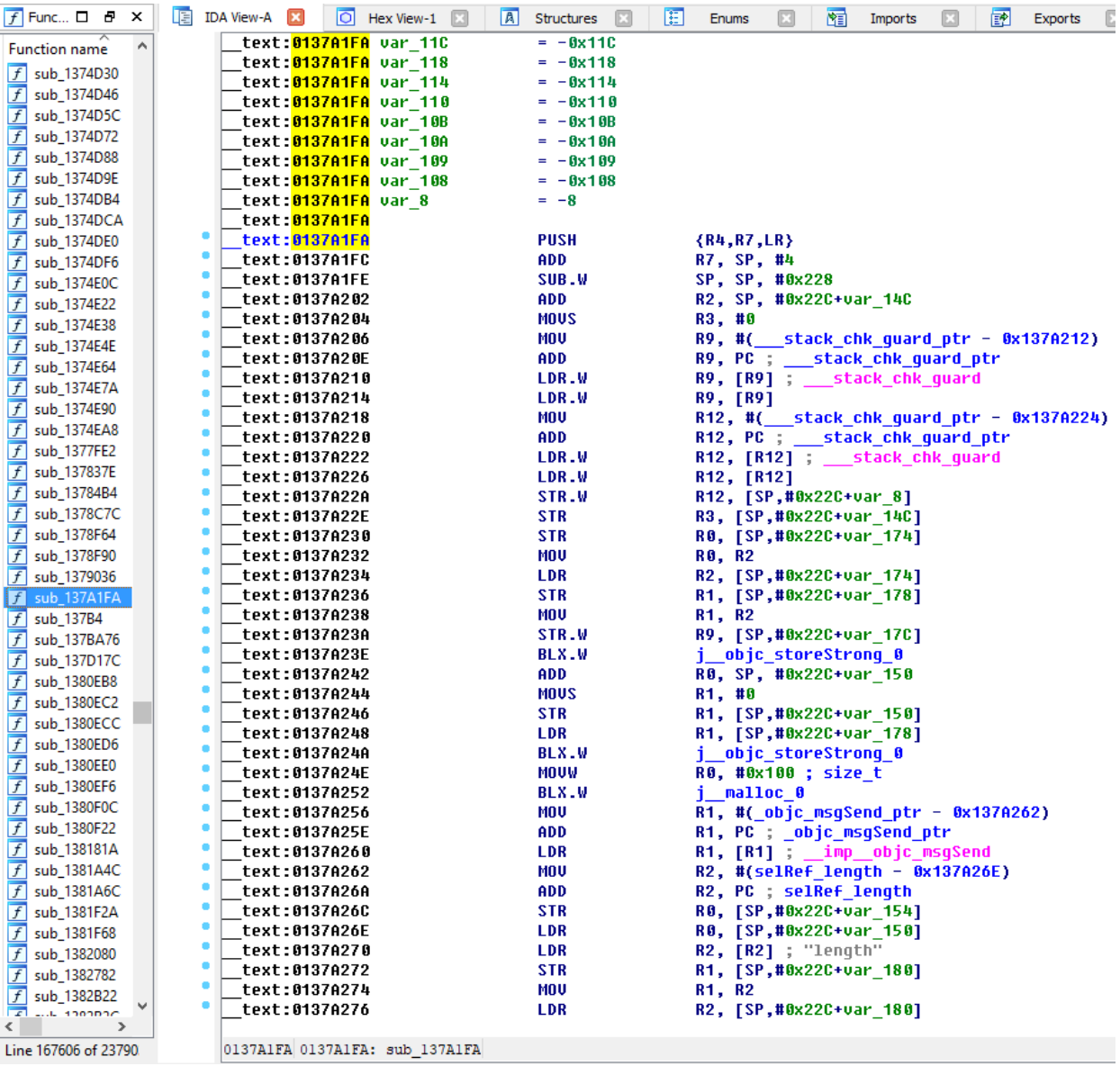
No debug symbols

Need to determine where to add breakpoints

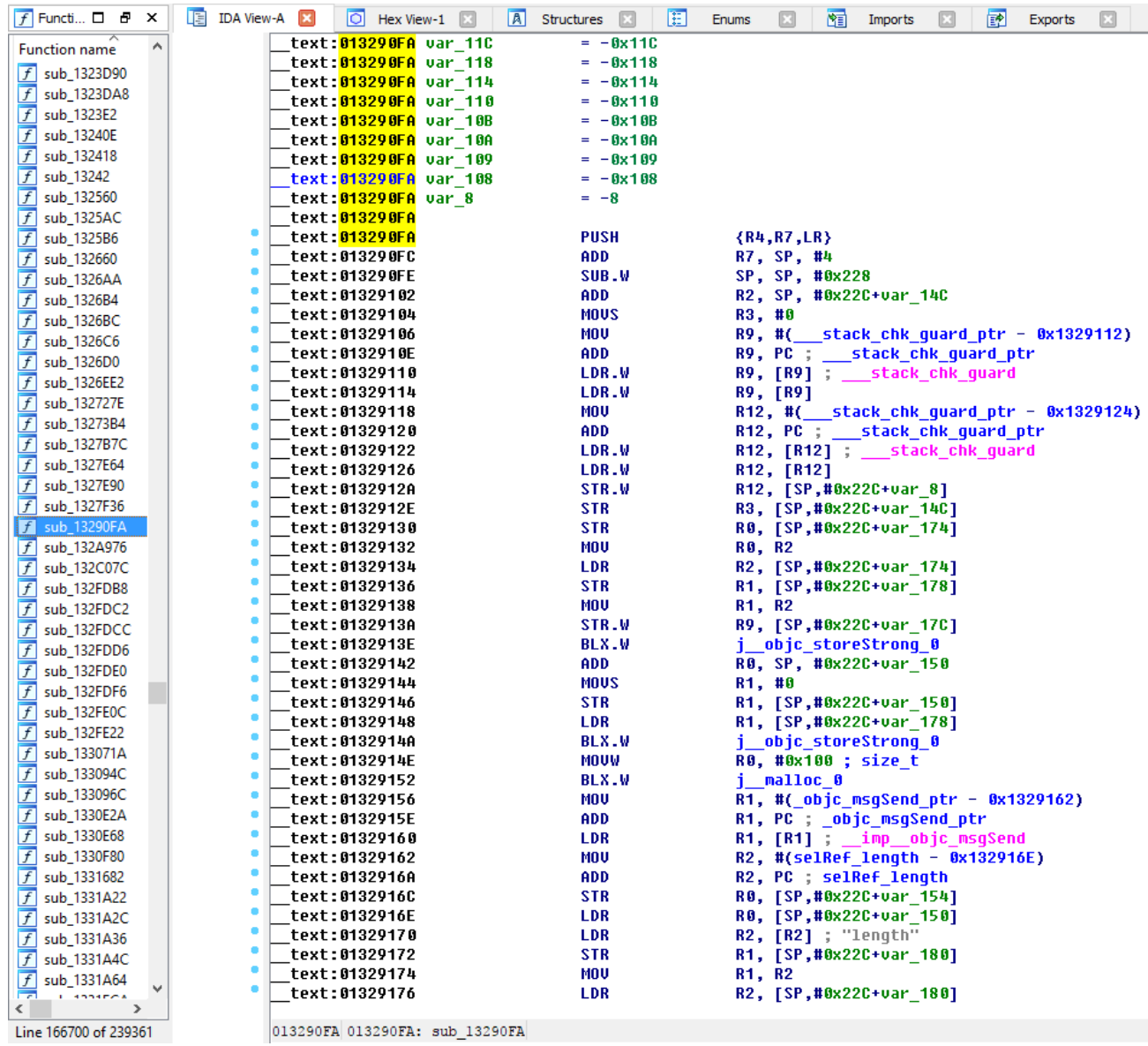
- In 167,606 functions



Baidu mobile (version 9.30)
Function: sub_137A1FA



Baidu mobile (version 9.35)
Function: sub_13290FA



Different version, different function name and address, but same code!

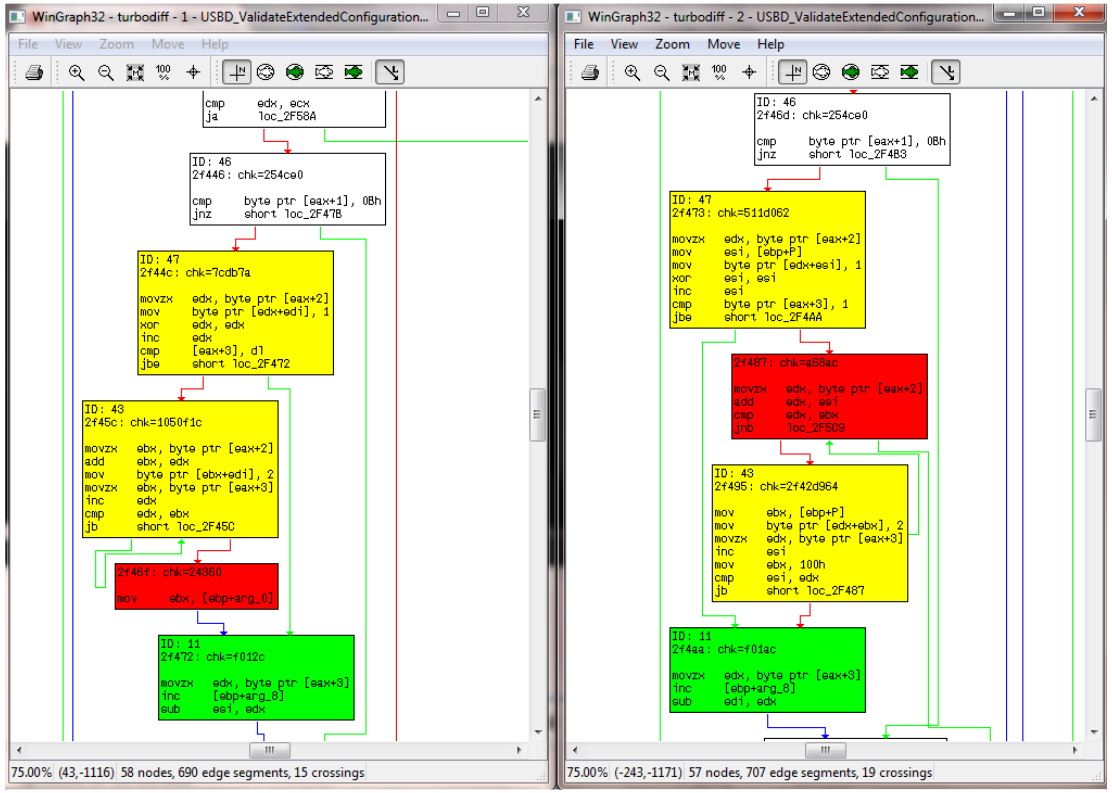
Picture Yourself as an Analyst

- **You just identified a function of interest**

- **Questions:**
  - Have I seen an equivalent or similar function before?
  - How can I find binaries that contain similar functions?

Security Applications of Code Similarity Detection

Patch analysis



“patch based exploit” generation

Software plagiarism detection

Malware analysis

Existing Works / Tools

Focus on control flow graph (CFG) extraction	• E.g., BinDiff, BinGrep, discovRe, Genius, Gemini
Based on basic block modeling	• E.g., BinHunt, Cop
Run in the emulator	• E.g., BinSim
Rely on Pin tool	• E.g., BLEX, CryptHunt

Existing Works / Tools Limitations

Focus on control flow graph (CFG) extraction	• Exclude obfuscated binaries
Based on basic block modeling	• Theorem prover is computationally expensive. • Impractical for large code bases
Run in the emulator	• Emulator detection
Rely on Pin tool	• Not for iOS

Example (Objective-C)

```
1.  - (NSString *)encrypt1:(NSString *)message {
2.      if ([message length] == 0) {
3.          return @"NULL";
4.      }
5.
6.      NSString *key = [self makeKey1];
7.      NSString *encryptedMsg = [self xorWithString:key withMessage:message];
8.
9.      NSLog(@"encrypt1: %@", encryptedMsg);
10.     return encryptedMsg;
11. }
```

Code Obfuscation (IDA decompiled code)

```
// ViewController - (id)encrypt1:(id)
id __cdecl -[ViewController encrypt1:](struct ViewController *self, SEL a2, id a3)
{
  struct ViewController *v3; // r5@1
  void *v4; // r4@1
  int v5; // r1@1
  void *v6; // r0@2
  int v7; // r0@2
  int v8; // r6@2
  void *v9; // r0@2
  int v10; // r5@2

  v3 = self;
  v4 = (void *)objc_retain(a3, a2);
  if ( objc_msgSend(v4, "length") )
  {
    v6 = objc_msgSend(v3, "makeKey1");
    v7 = objc_retainAutoreleasedReturnValue(v6);
    v8 = v7;
    v9 = objc_msgSend(v3, "xorWithString:withMessage:", v7, v4);
    v10 = objc_retainAutoreleasedReturnValue(v9);
    NSLog(66020, v10);
    objc_release(v8);
  }
  else
  {
    v10 = 66004;
    objc_retain(66004, v5);
  }
  objc_release(v4);
  return (id)j__objc_autoreleaseReturnValue(v10);
}
```

No Obfuscation

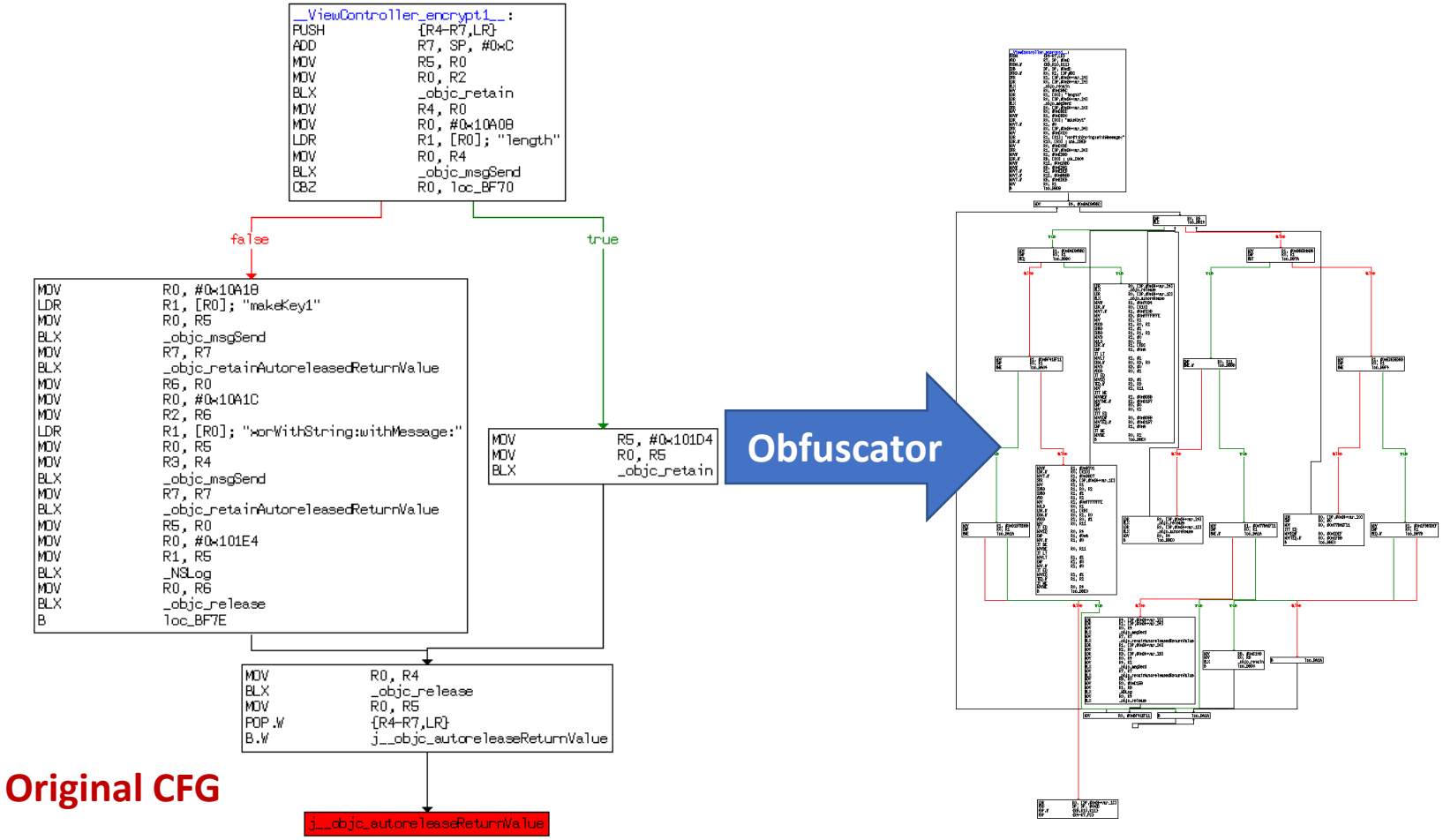
LLVM-Obfuscator

```
// ViewController - (id)encrypt1:(id)
id __cdecl -[ViewController encrypt1:](struct ViewController *self, SEL a2, id a3)
{
  int v3; // r6@0
  signed int v4; // r0@1
  void *v5; // r0@4
  int v6; // r4@4
  void *v7; // r0@4
  int v15; // r2@14
  signed int v16; // r1@16
  signed int v17; // r2@19
  signed int v19; // r2@26
  signed int v22; // r3@28
  int v23; // r0@28
  signed int v24; // r2@30
  struct ViewController *v26; // [sp+8h] [bp-2Ch]@1
  id v27; // [sp+10h] [bp-24h]@1
  void *v28; // [sp+14h] [bp-20h]@1
  struct objc_object *v29; // [sp+18h] [bp-1Ch]@0

  v26 = self;
  v27 = a3;
  objc_retain(a3, a2);
  v28 = objc_msgSend(v27, "length");
  v4 = -303701101;
  while ( 1 )
  {
    while ( 1 )
    {
      while ( v4 > -303701102 )
      {
        if ( v4 > 1801001627 )
        {
          if ( v4 != 1801001628 )
          {
            if ( v4 != 2008731425 )
              goto LABEL_41;
            v5 = objc_msgSend(v26, "makeKey1");
            v6 = objc_retainAutoreleasedReturnValue(v5);
            v7 = objc_msgSend(v26, "xorWithString:withMessage:");
            v3 = objc_retainAutoreleasedReturnValue(v7);
            NSLog(49496, v3);
            objc_release(v6);
            goto LABEL_5;
          }
          objc_release(v27);
          objc_autorelease(v29);
          v4 = -1698061380;
        }
        else if ( v4 == -303701101 )
        {
          v4 = 2008731425;
          if ( !v28 )
```

Obfuscation

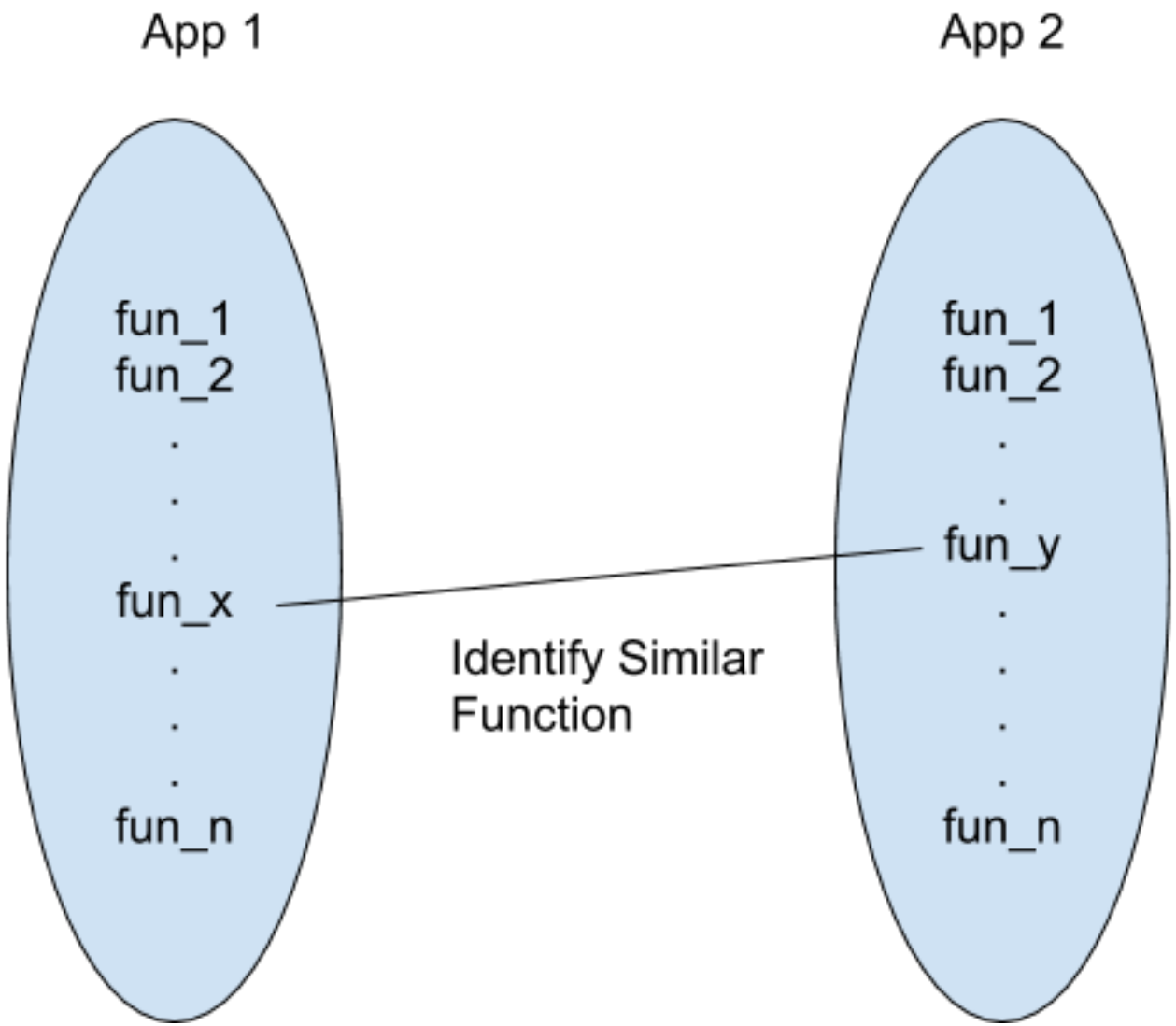
Code Obfuscation



Original CFG

Obfuscated CFG

Goal


App 1
App 2
fun_1
fun_2
fun_x
fun_n
fun_1
fun_2
fun_y
fun_n
Identify Similar
Function

Problems

- **What features are useful for semantic similarity comparison?**
- **How these features are captured on mobile platform?**
- **How to characterize a function with such features?**

MobileFindr

On-device and Trace-based Similarity Mapping Framework for iOS Apps

Idea

- **What features are useful for semantic similarity comparison?**
  - dynamic behavior features
- **How these features are captured on mobile platform?**
  - Dynamic instrumentation
- **How to characterize a function with such features?**
  - Thread backtrace

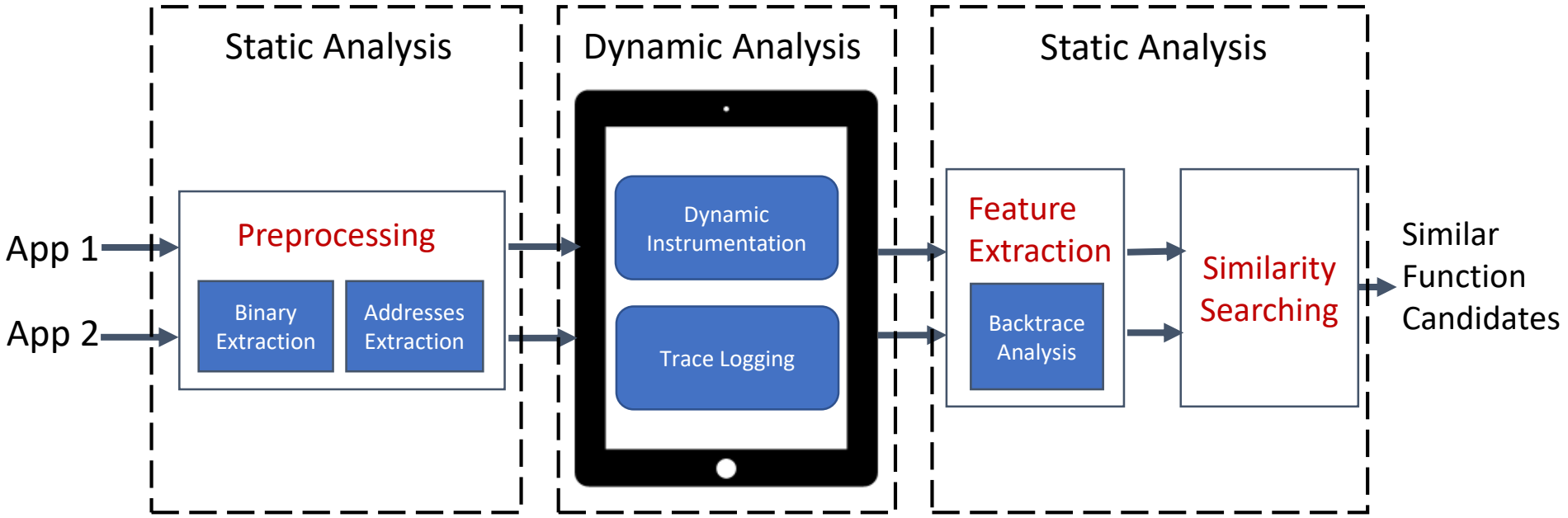
Dynamic Behavior Features

We define the concept of "dynamic behavior features" broadly to include any information that can be derived from observations made during execution.

Dynamic Behavior Features

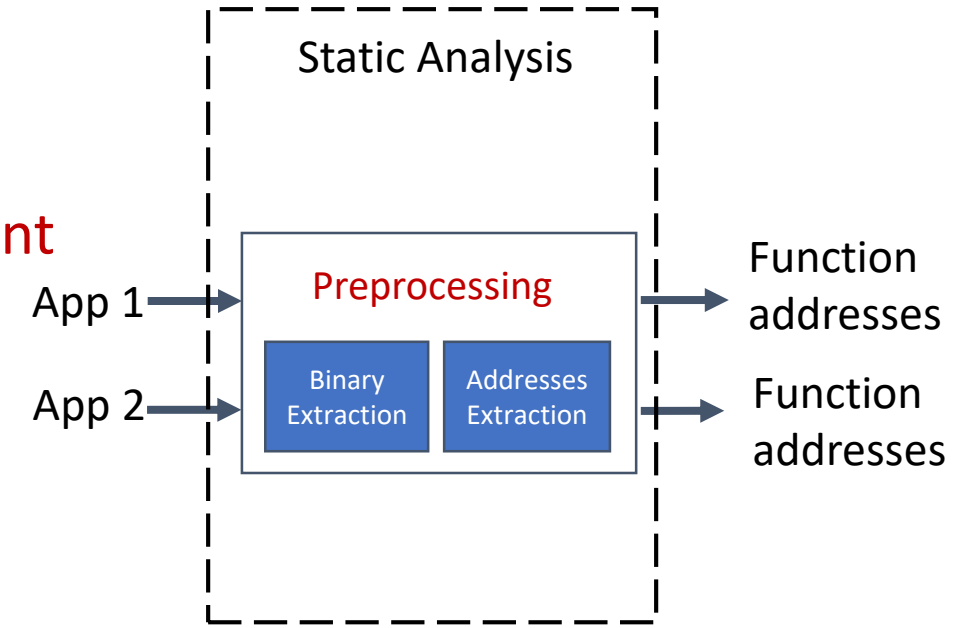
- **System calls**
  - E.g., read, write, open, etc. (libsystem_kernel.dylib)
- **Library calls**
  - Memset, memcpy, free. Etc (libSystem.B.dylib)
  - _objc_getClass, _objc_getProtocol, etc. (libobjc.A.dylib)
  - Etc.
- **Framework API calls**
  - E.g., OpenGLES, CoreMedia, UIKit, etc.

System Design



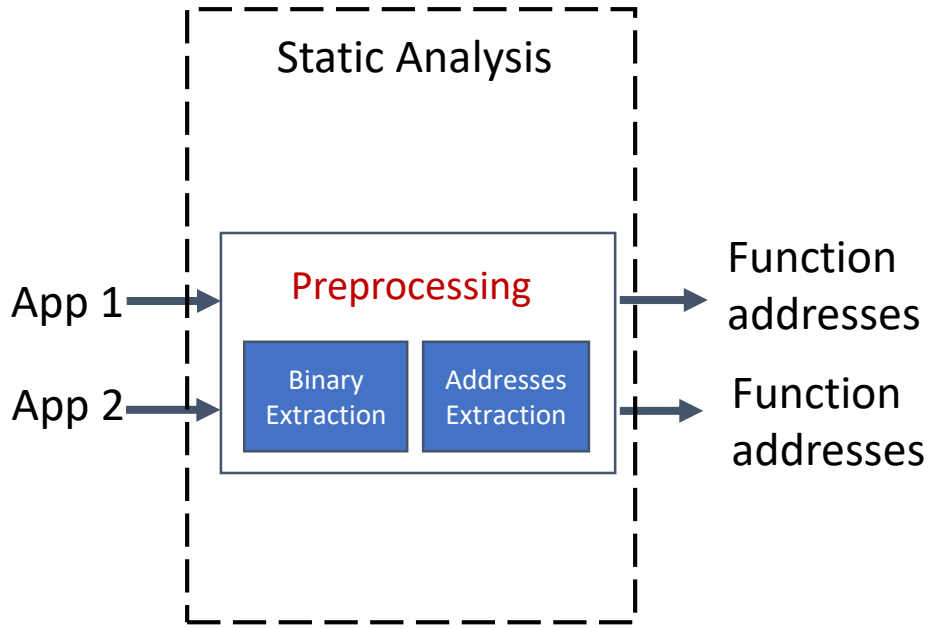
Preprocessing

- **Binary extraction**
  - Extract app binaries
    - Clutch 2.0
  - Remove PIE (Position-independent executable)
    - Armv7: offset 0x401a
      - 0x00 -> 0x01
    - Non-armv7(Arm64): offset 0x18
      - 0x00 -> 0x01
  - Re-sign the binary and copy it back to app folder



Preprocessing

- **Address extraction**
  - Implemented with IDAPython API



Listing 1.1: Function Addresses

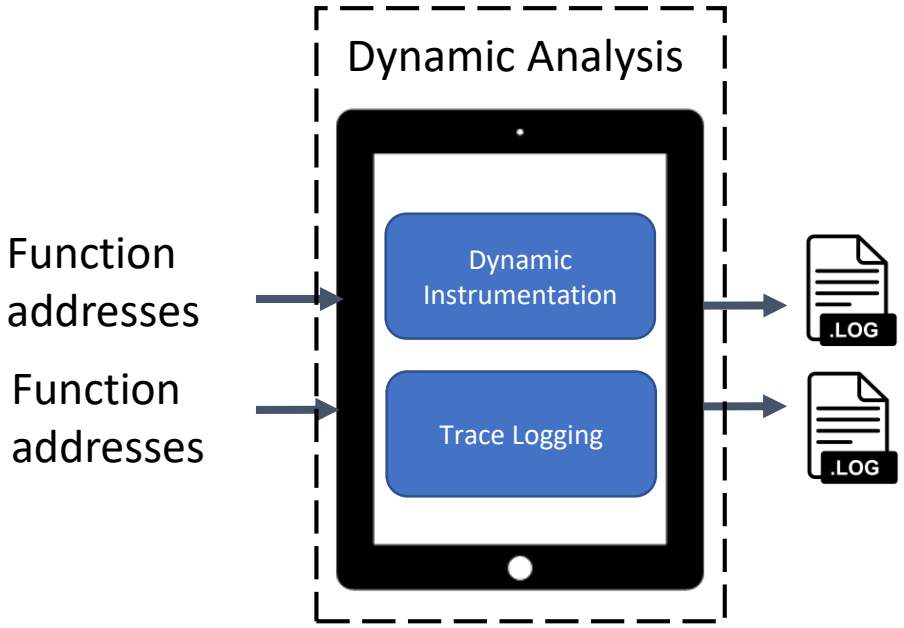
```
...
0xb7ea,0xb964,-[VideoSnakeViewController
    toggleRecording:]
0xe2cc,0xe51c,-[VideoSnakeSessionManager
    startRecording]
0x111d8,0x1128c,-[MovieRecorder initWithURL:]
0x1161c,0x116a8,-[MovieRecorder delegate]
0x11834,0x11980,-[MovieRecorder prepareToRecord]
0x11d48,0x11ebc,-[MovieRecorder finishRecording]
...
```

Listing 1.2: Library Addresses

```
...
0x1606c,__Block_copy
0x1607c,__Block_object_assign
0x1608c,__Block_object_dispose
0x1609c,__Unwind_SjLj_Register
0x160ac,__Unwind_SjLj_Resume
0x160bc,__Unwind_SjLj_Unregister
...
```

On-device Dynamic Analysis

- **Dynamic instrumentation**
  - Implemented with FRIDA API: https://www.frida.re/
    - Hook functions
    - Inject JavaScript
- **Trace logging**
  - Framework API invocations
  - Library function invocations
  - System call invocations
  - Thread backtrace



Listing 1.1: Function Addresses

```
...
0xb7ea,0xb964,-[VideoSnakeViewController
    toggleRecording:]
0xe2cc,0xe51c,-[VideoSnakeSessionManager
    startRecording]
0x111d8,0x1128c,-[MovieRecorder initWithURL:]
0x1161c,0x116a8,-[MovieRecorder delegate]
0x11834,0x11980,-[MovieRecorder prepareToRecord]
0x11d48,0x11ebc,-[MovieRecorder finishRecording]
...
```

Listing 1.2: Library Addresses

```
...
0x1606c,__Block_copy
0x1607c,__Block_object_assign
0x1608c,__Block_object_dispose
0x1609c,__Unwind_SjLj_Register
0x160ac,__Unwind_SjLj_Resume
0x160bc,__Unwind_SjLj_Unregister
...
```

Listing 1.3: Stack Backtrace: Address

```
...
0x1609c,0x11892,0xe498,0xb92e,0xb15a
0x1621c,0x118c0,0xe498,0xb92e,0xb15a
0x1620c,0x118fc,0xe498,0xb15a
...
```

Listing 1.4: Stack Backtrace: Name

```
...
__Unwind_SjLj_Register,-[MovieRecorder
    prepareToRecord],-[VideoSnakeSessionManager
    startRecording],-[VideoSnakeViewController
    toggleRecording:],sub_B120
_dispatch_get_global_queue,-[MovieRecorder
    prepareToRecord],-[VideoSnakeSessionManager
    startRecording],-[VideoSnakeViewController
    toggleRecording:],sub_B120
_dispatch_async,-[MovieRecorder prepareToRecord],-[
    VideoSnakeSessionManager startRecording],-[
    VideoSnakeViewController toggleRecording:],
    sub_B120
...
```

Feature Extraction

- **Analyze the stack backtrace information**
  - Match the library calls/system calls/api calls to corresponding caller functions

Listing 1.5: Function Features

```
{
        "name" : "-[MovieRecorder prepareToRecord]",
        "features" : [
                [
                        "__Unwind_SjLj_Register",
                        "_dispatch_get_global_queue",
                        "_dispatch_async",
                        "__Block_object_assign",
                        "__Unwind_SjLj_Unregister"
                ]
        ]
}
```

Similarity Searching

- **Calculate similarity score**
  - Edit distance: sim(fun_x, fun_y)
    - E.g., reference function **fun_x** and target function **fun_y**

Evaluation

- **Experiment Setup**
  - On-device analysis
    - 32GB Apple Jailbroken iPad
    - iOS 8.3
  - Feature extraction & similarity searching
    - CPU: Intel Core i7-6700K Processor (8-core with 4.00 GHz)
    - 64 GB memory
    - Ubuntu Linux 14.04 TLS
    - Python 2.7.12
    - IDA Pro 6.6

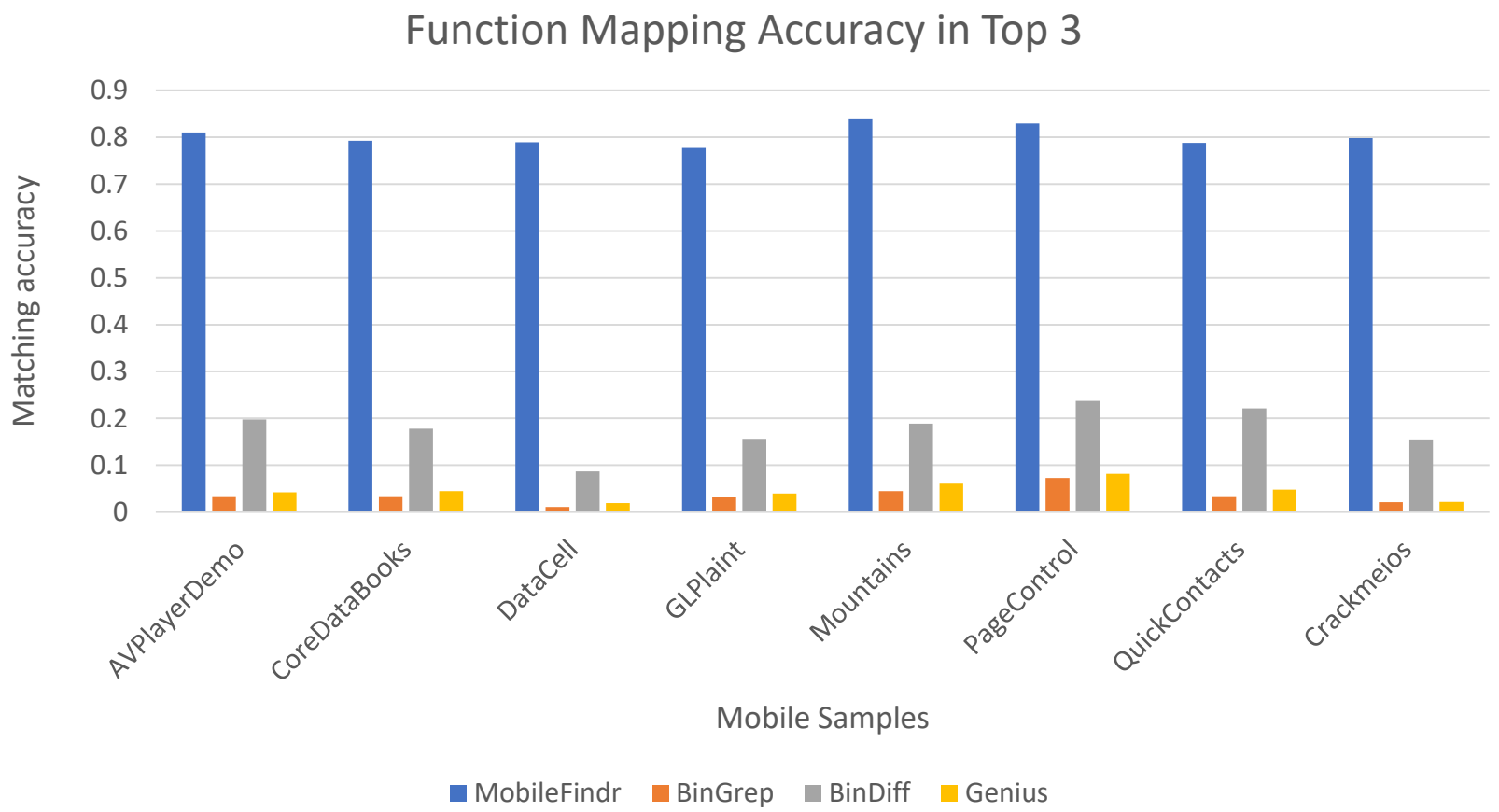
Evaluation

- **Can MobileFindr detect similar functions in different versions of same mobile apps?**
- **Can MobileFindr detect similar functions used by different mobile apps**
- **Can MobileFindr be used for analyzing real-world mobile apps?**

Ground Truth Dataset I

- **8 Open source iOS app samples:** https://developer.apple.com/library/content/navigation/
- **Obfuscated by llvm-obfuscator 4.0**
  - 8 non-obfuscated vs 8 obfuscated version
  - Select functions from non-obfuscated version as reference functions
- **Keep Debug symbols**
  - ground truth through function names

Comparative Results



Evaluation

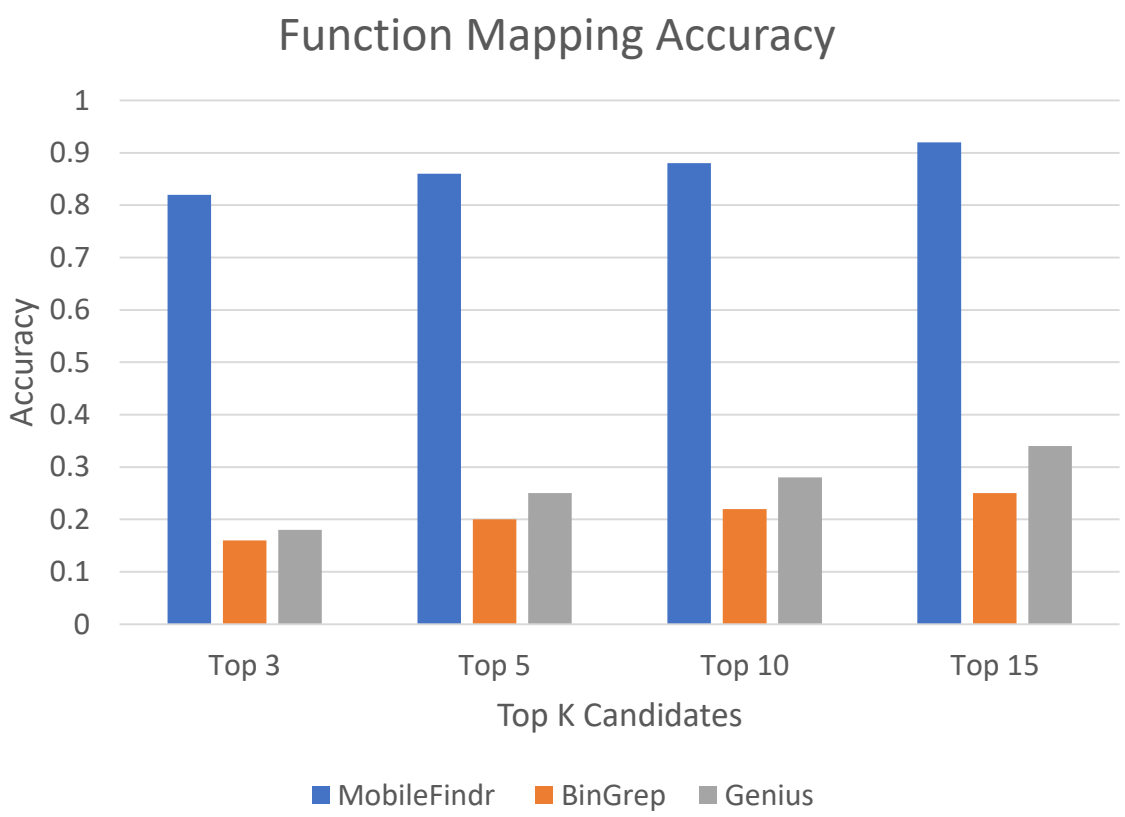
- **Can MobileFindr detect similar functions in different versions of same mobile apps?**
- **Can MobileFindr detect similar functions used by different mobile apps**
- **Can MobileFindr be used for analyzing real-world mobile apps?**

Ground Truth Dataset II

- **Top used third-part iOS framework**
  - AFNetworking 3.0
- **8 opensource projects from GitHub that use the framework above**
- **Three obfuscation options: (**https://github.com/obfuscator-llvm/obfuscator/wiki/Features**)**
  - Control flow flattening (-fla)
  - Instruction substitution (-sub)
  - Bogus control flow (-bcf)
- **Result in 64 apps (56 obfuscated apps + 8 non-obfuscated)**
- **Debug symbols ->** ground truth through function names

Comparative Results

	Obfuscation Type
1	NONE
2	-sub
3	-bcf
4	-fla
5	-sub, -bcf
6	-sub, -fla
7	-bcf, -fla
8	-sub, -bcf, -fla



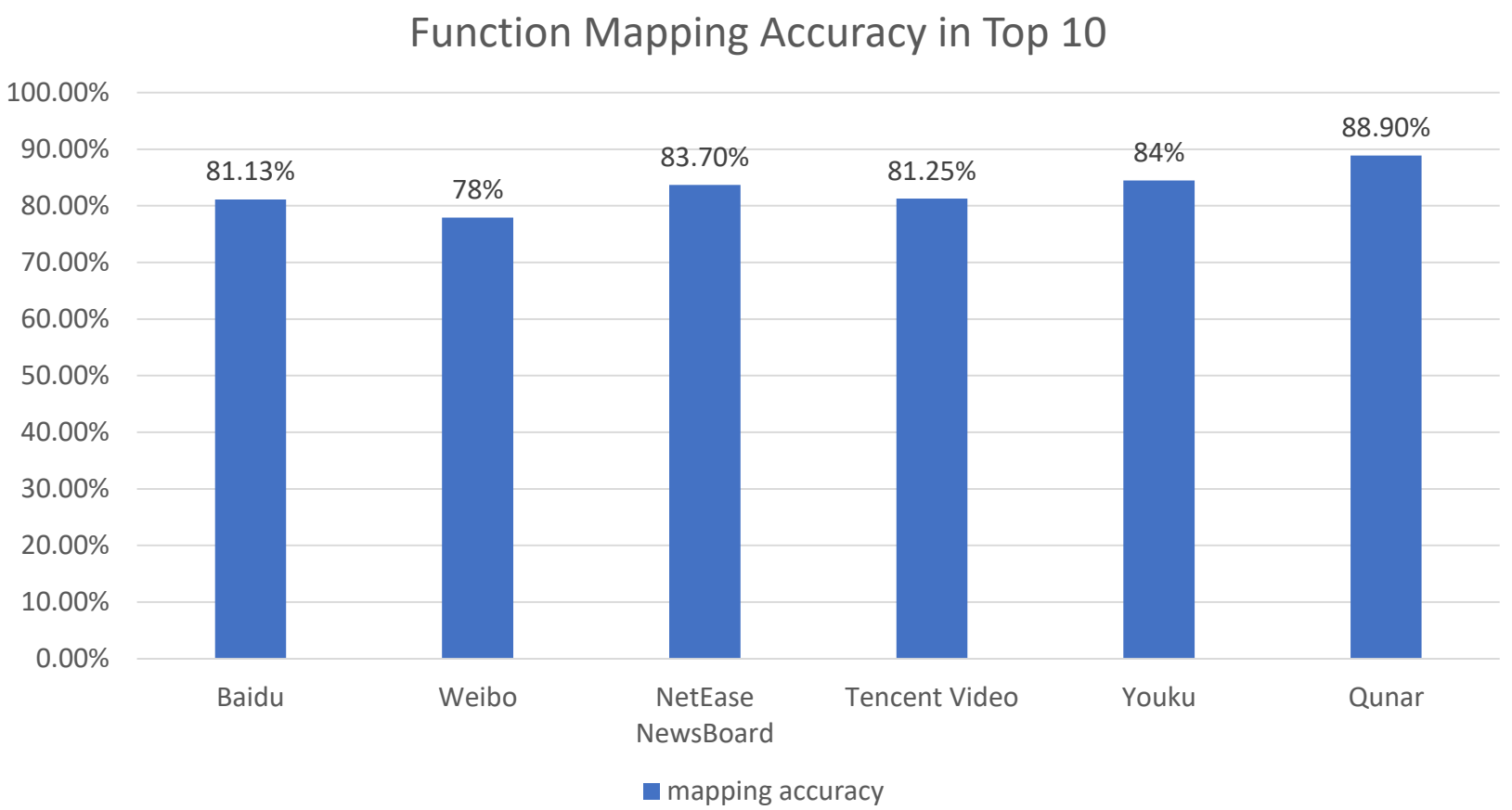
Evaluation

- **Can MobileFindr detect similar functions in different versions of same mobile apps?**
- **Can MobileFindr detect similar functions used by different mobile apps**
- **Can MobileFindr be used for analyzing real-world mobile apps?**

Analysis Real World Apps

- **Collect top-ranked apps from iOS App Store**
  - Compare different versions of each apps
    - E.g., baidu_v9.3 vs baidu_v9.35
  - Perform same action for each pair of apps to collect traces
    - E.g., perform web searching with same keywords
  - List top 10 similar function candidates
    - Based on the similarity score

Analysis Real World Apps



Limitations & Discussions

- **Path coverage issue**
  - Incomplete path coverage for dynamic analysis
  - May use offloading mechanism for path exploration (presented by Malton, USENIX 2017)
- **Small function issue**
  - We only evaluate functions contain more than 5 basic block

Summary

- Designed a trace-based function similarity mapping system for mobile apps
  - Resilience to various anti-reverse engineering techniques
- Demonstrated the viability of our approach for multiple top-ranked real-world iOS frameworks and apps

THANK YOU!

Q&A?