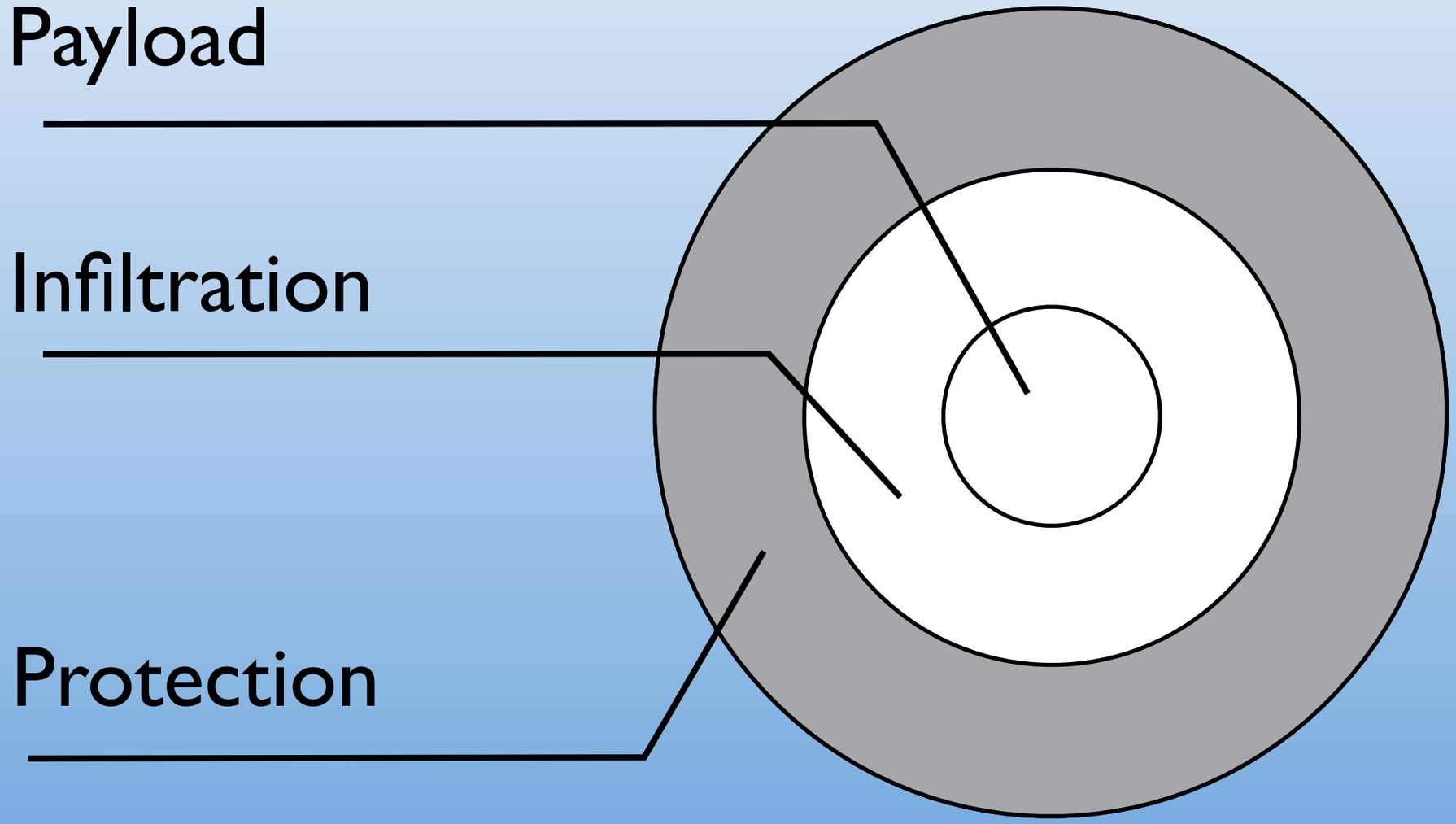
THE UNIVERSITY OF ARIZONA

Weaknesses in Defenses against Web-Borne Malware

Gen Lu and Saumya Debray
Department of Computer Science
The University of Arizona

July, 2013

Web-based Malware



Motivation

Existing detectors are tied to today's malware protection techniques

- *Can we get around the detectors by breaking the assumptions they made?*

Outline

- Current techniques
- Thwarting detection/analysis
- Experimental evaluation
- Discussion

Current Techniques

Static Analysis

Zozzle:

Curtsinger et al., 2011

Cujo:

Rieck et al., 2010

Assumption:

Malware only apply code unpacking obfuscation

Dynamic Analysis

JSAND:

Cova et al., 2010

Cujo:

Rieck et al., 2010

Assumption:

Malicious behavior can be observed under monitoring

Multi-Path Exploration

Rozzle:

Kolbitsch et al., 2012

Kudzu:

Saxena et al., 2010

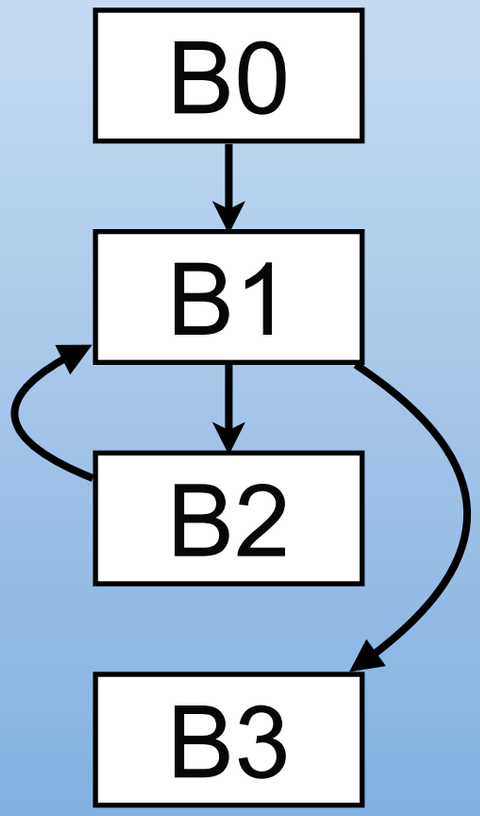
Assumption:

Branching is done by conditional statements

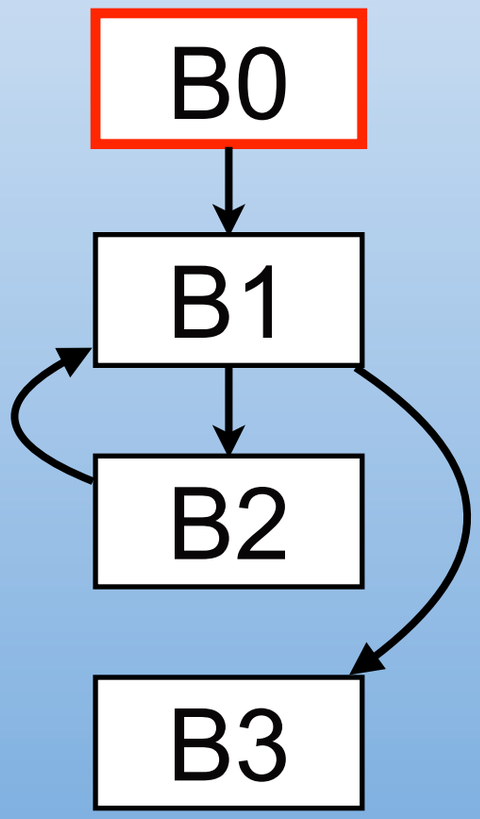
Thwarting Analysis

Defense		**Exploitation**
<u>Static analysis</u> simple deobfuscation	hiding malicious logic	Emulation-Based Obfuscation
<u>Dynamic analysis</u> single execution path	hiding malicious behavior	Anti-Analysis Defense
<u>Multi-path exploration</u> explore conditional statements	hiding branch logic	Implicit Conditionals

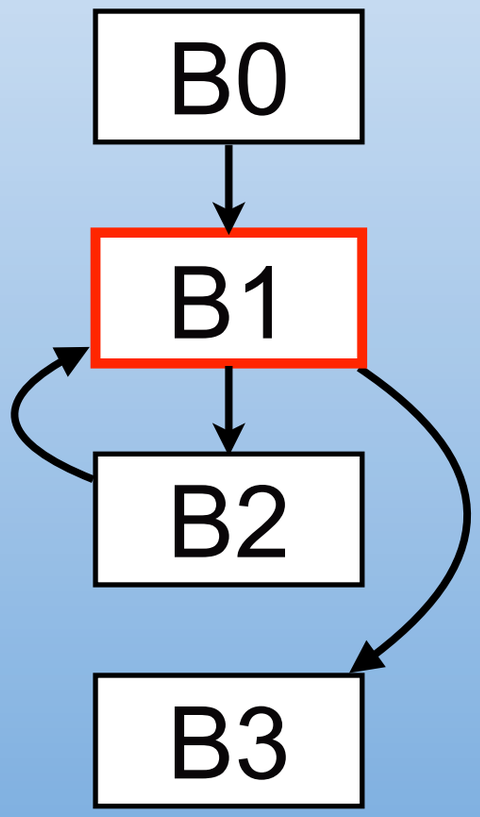
Emulation-Based Obfuscation



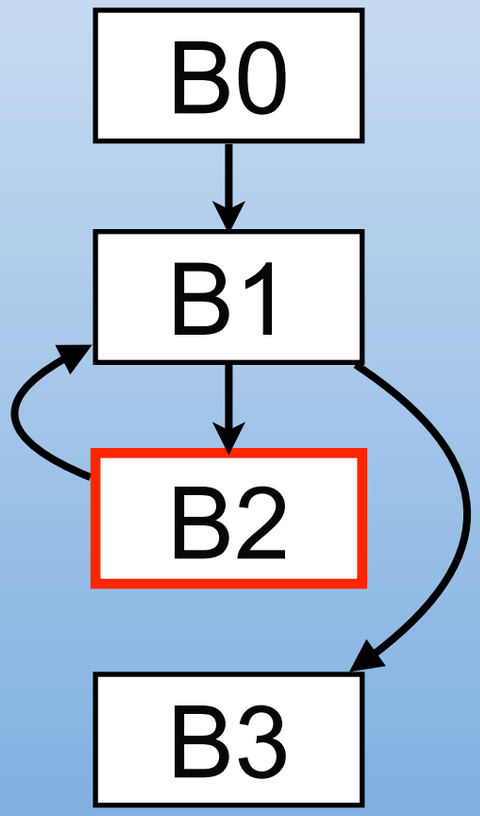
Emulation-Based Obfuscation



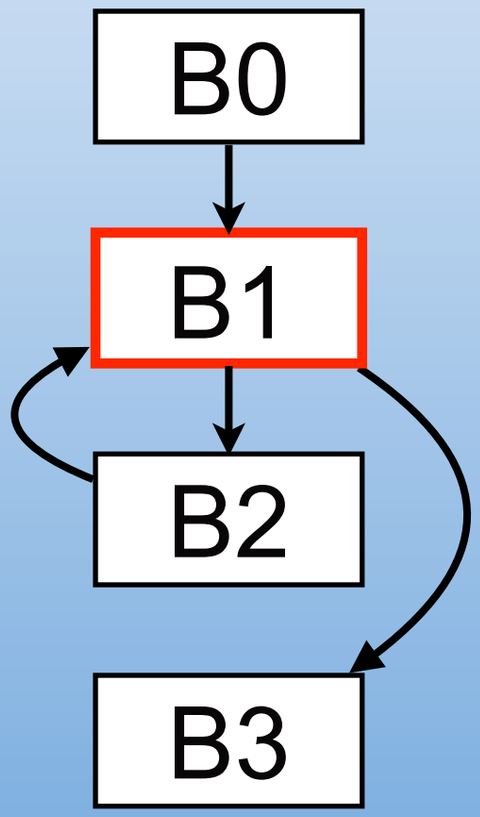
Emulation-Based Obfuscation



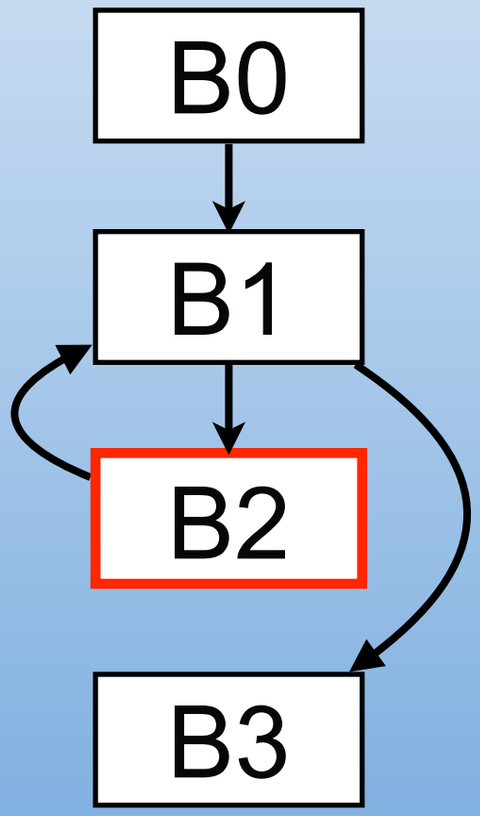
Emulation-Based Obfuscation



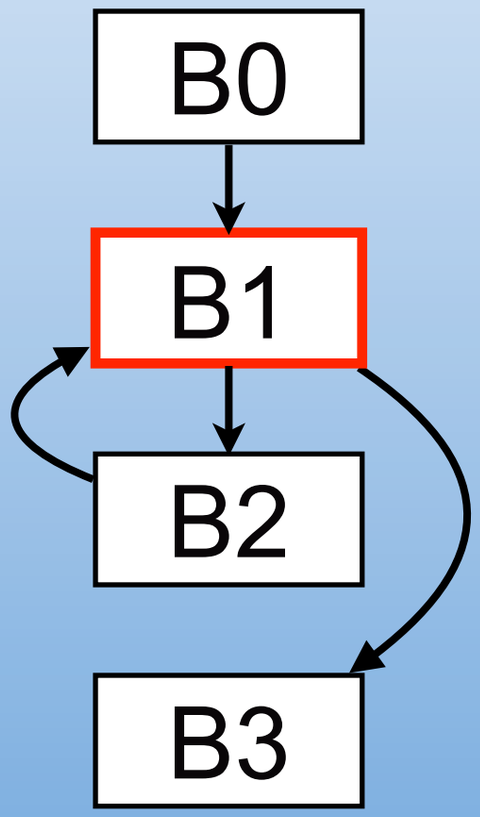
Emulation-Based Obfuscation



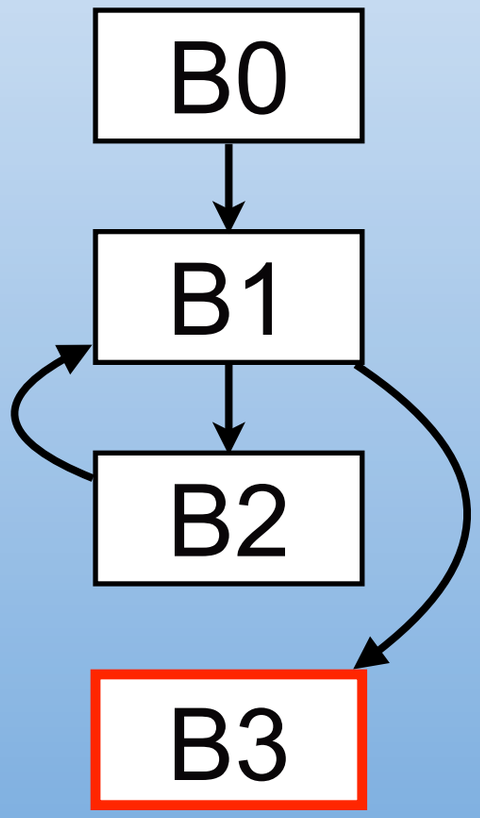
Emulation-Based Obfuscation



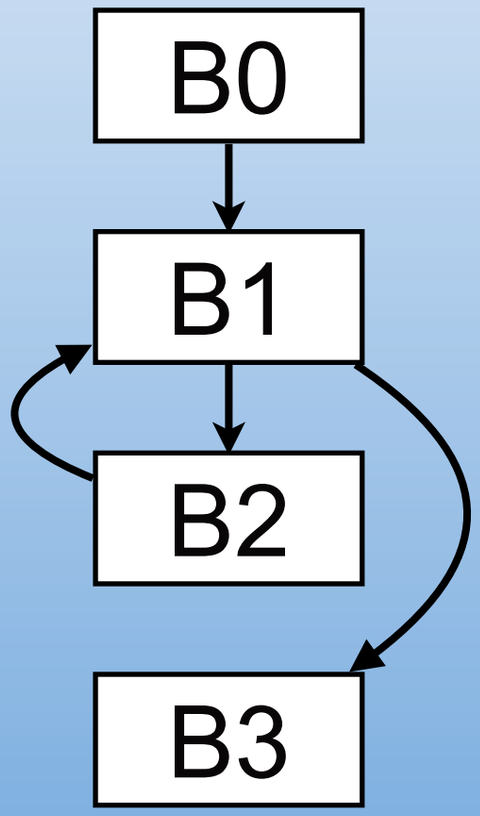
Emulation-Based Obfuscation



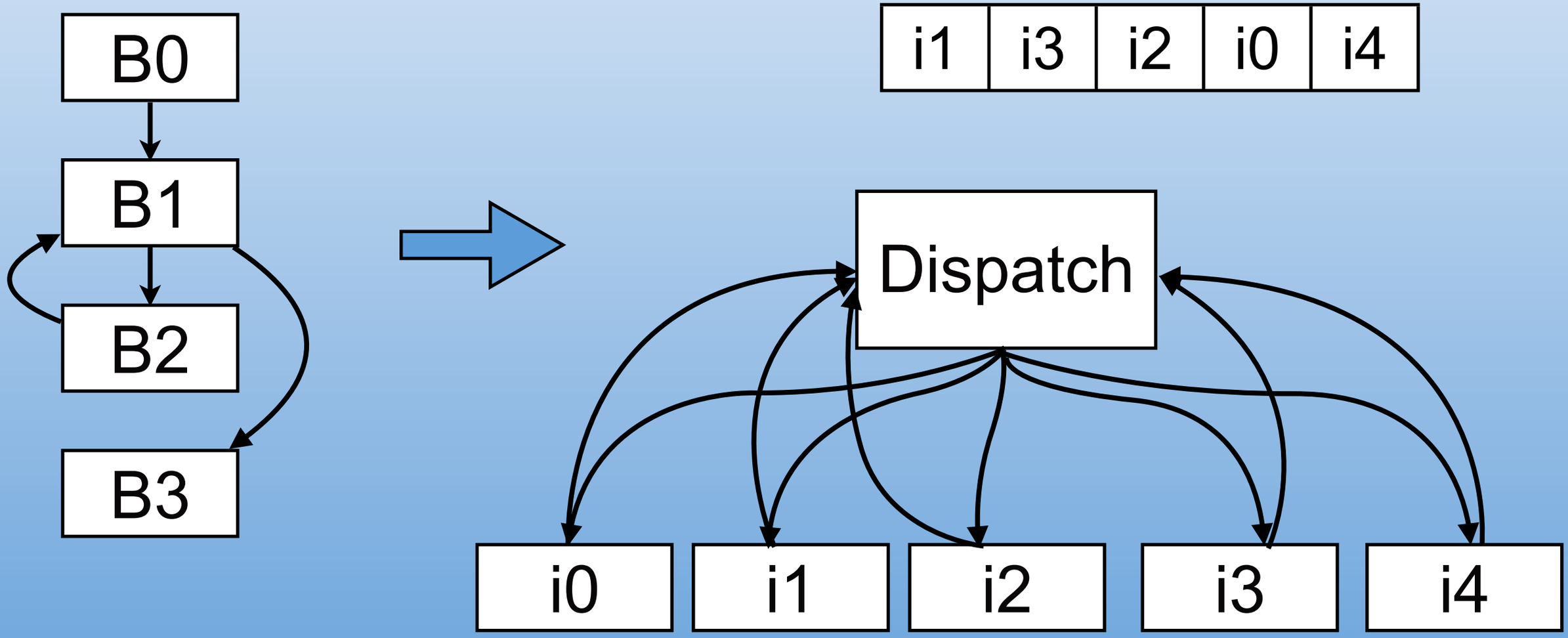
Emulation-Based Obfuscation



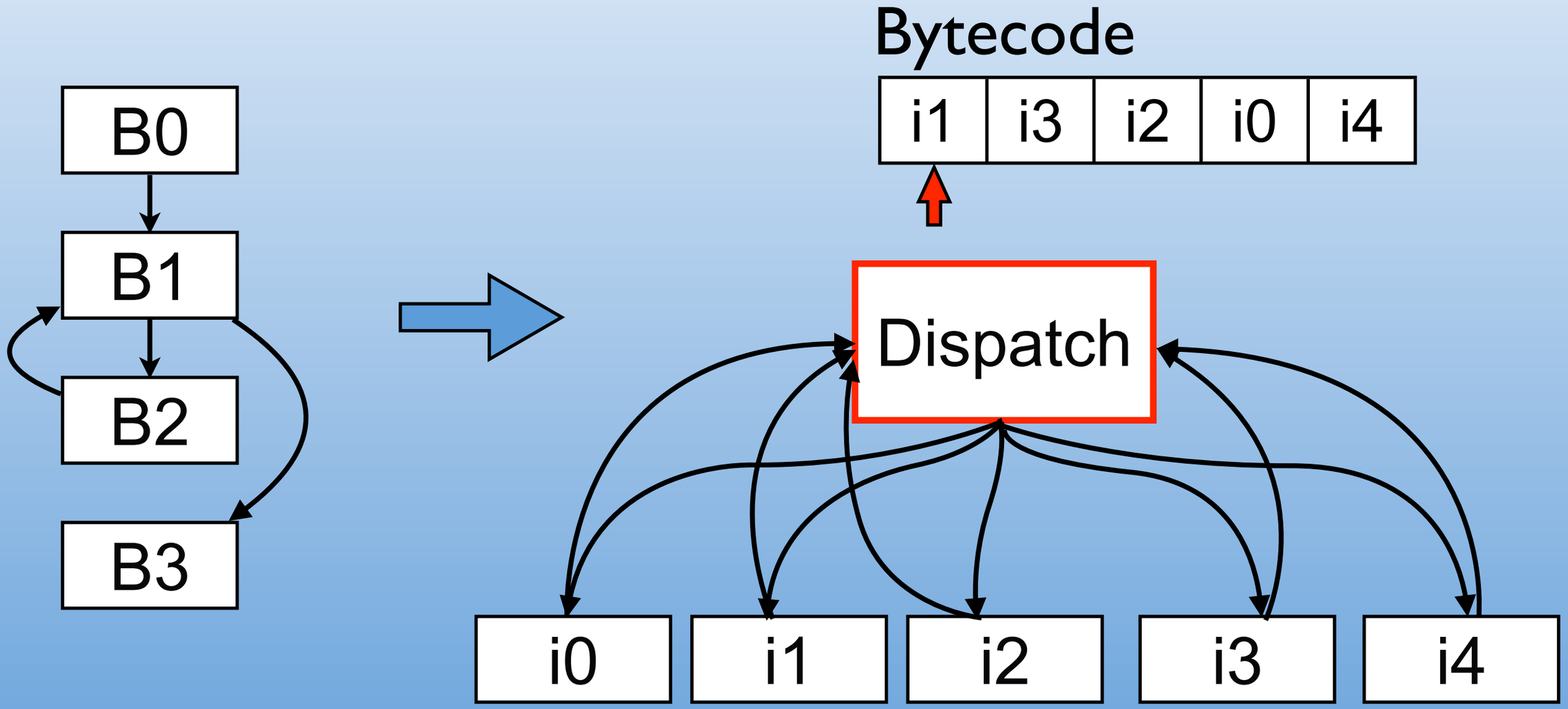
Emulation-Based Obfuscation



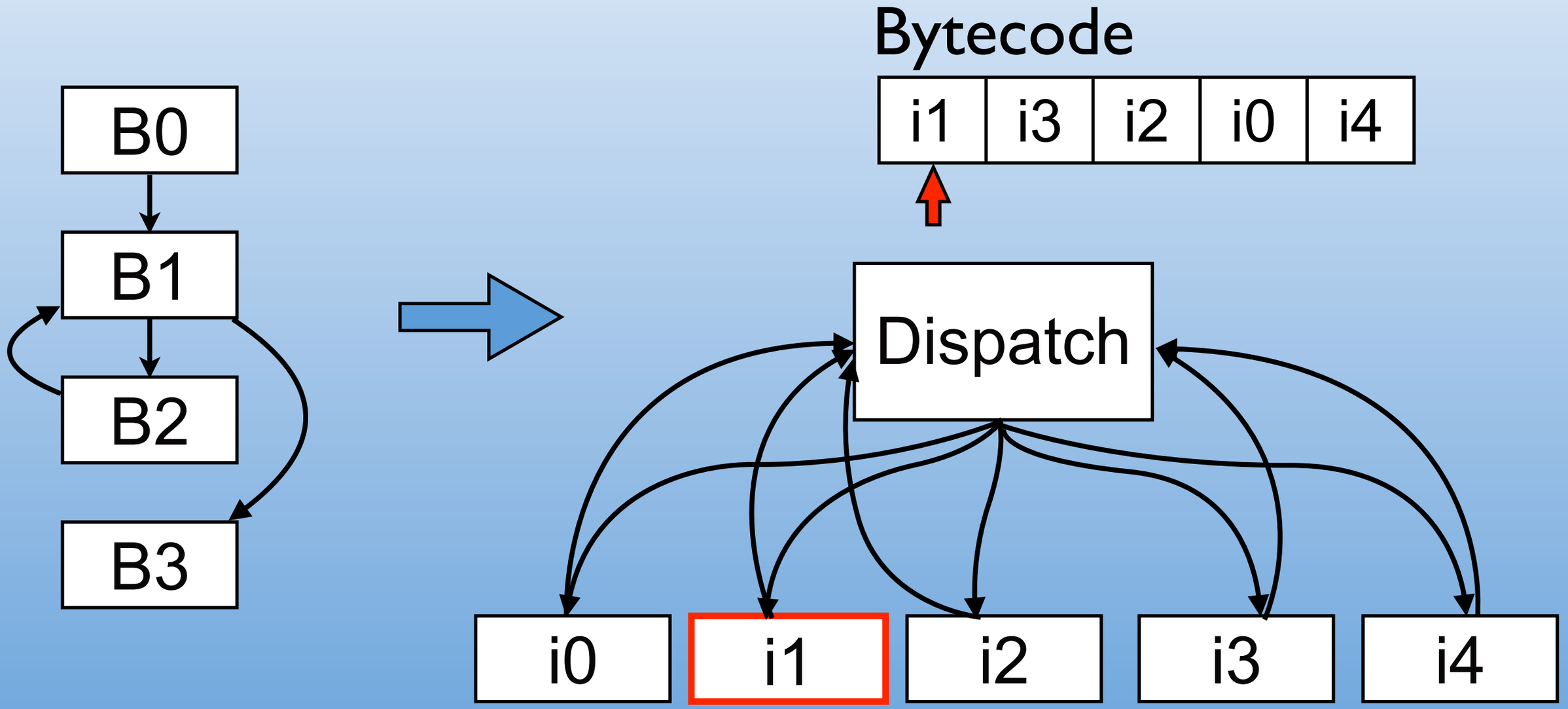
Emulation-Based Obfuscation



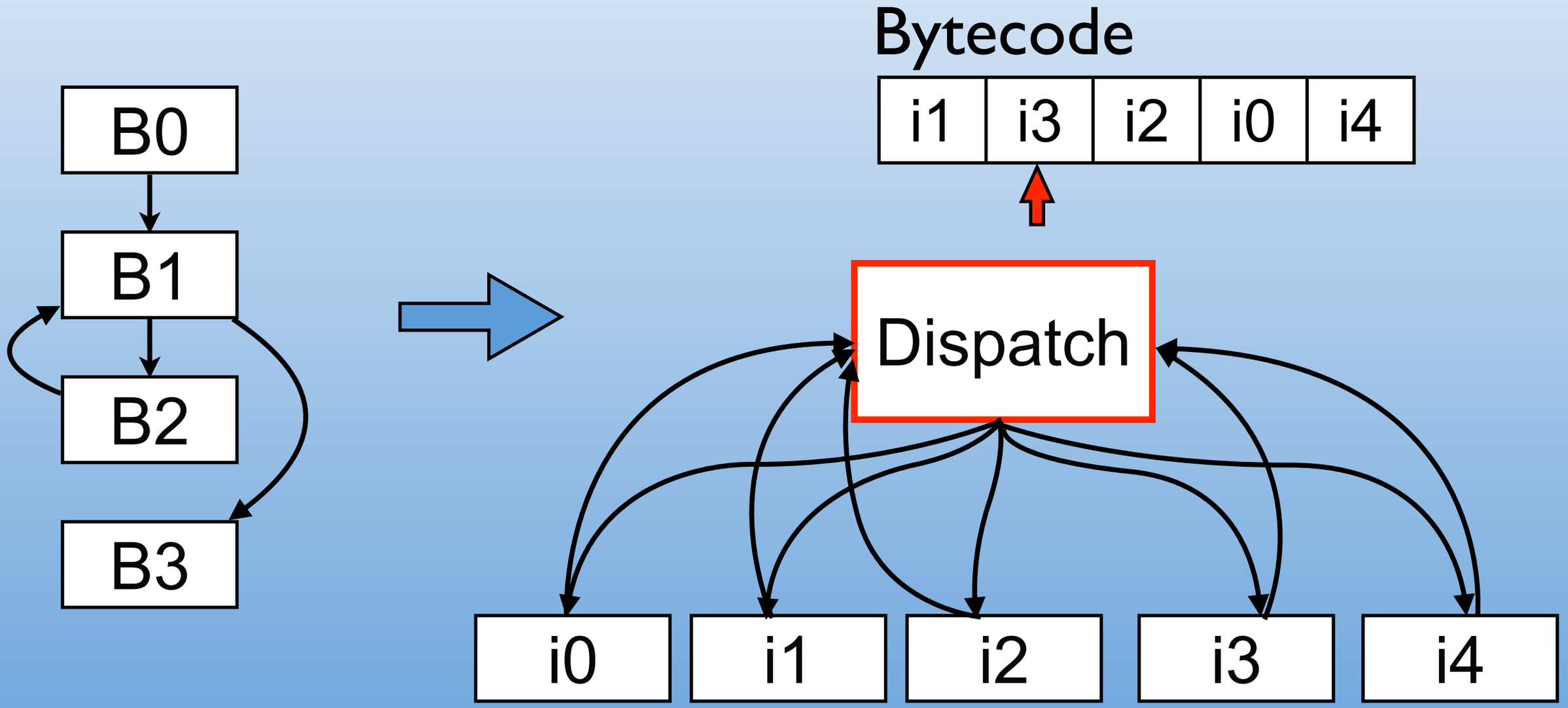
Emulation-Based Obfuscation



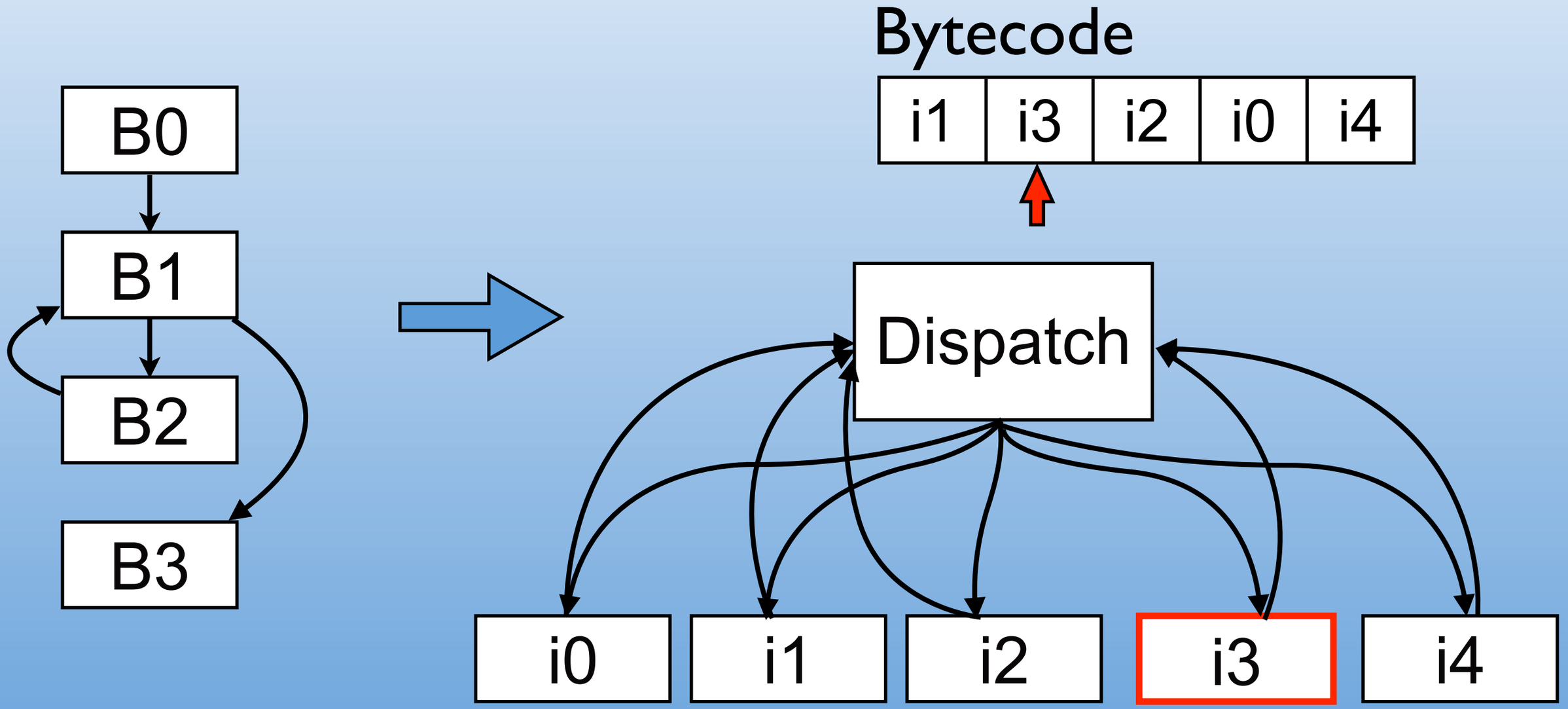
Emulation-Based Obfuscation



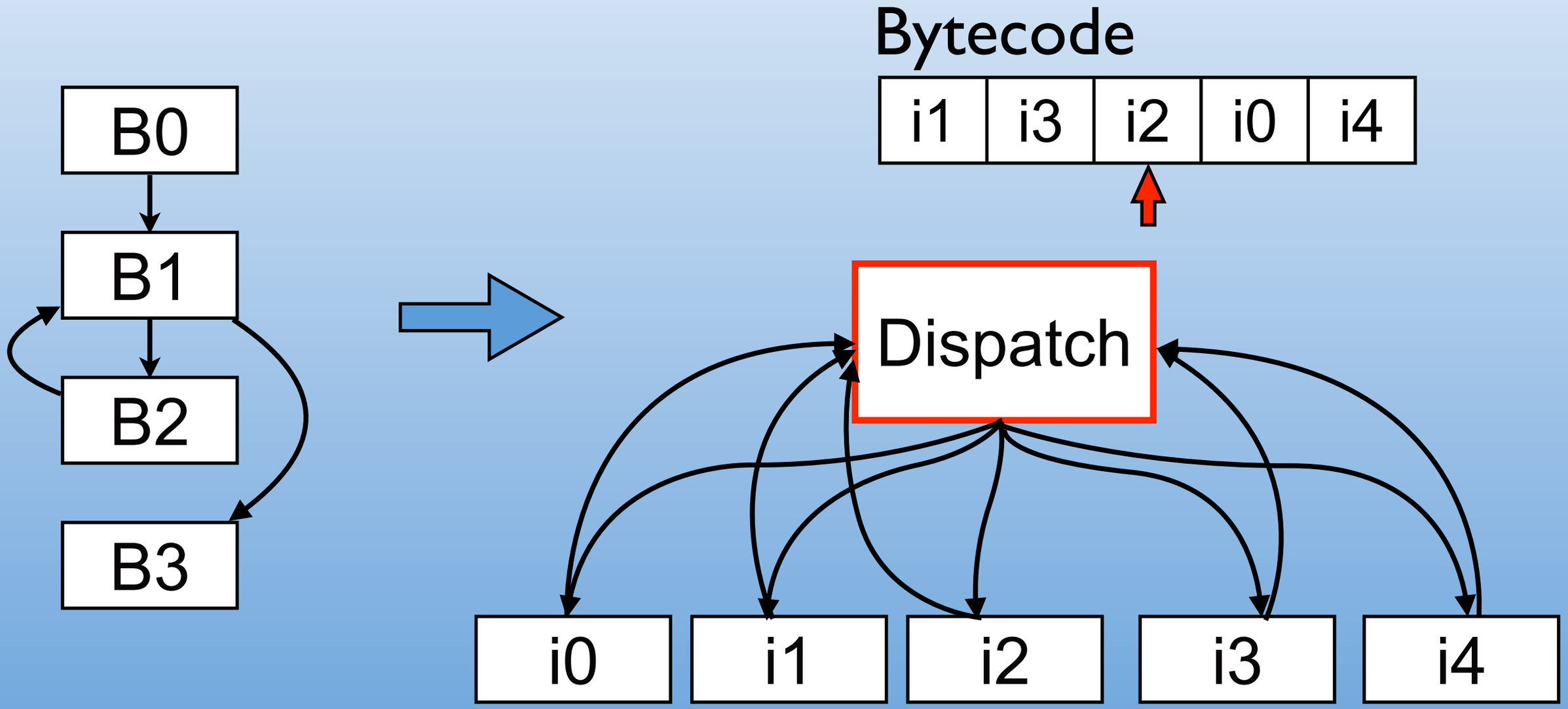
Emulation-Based Obfuscation



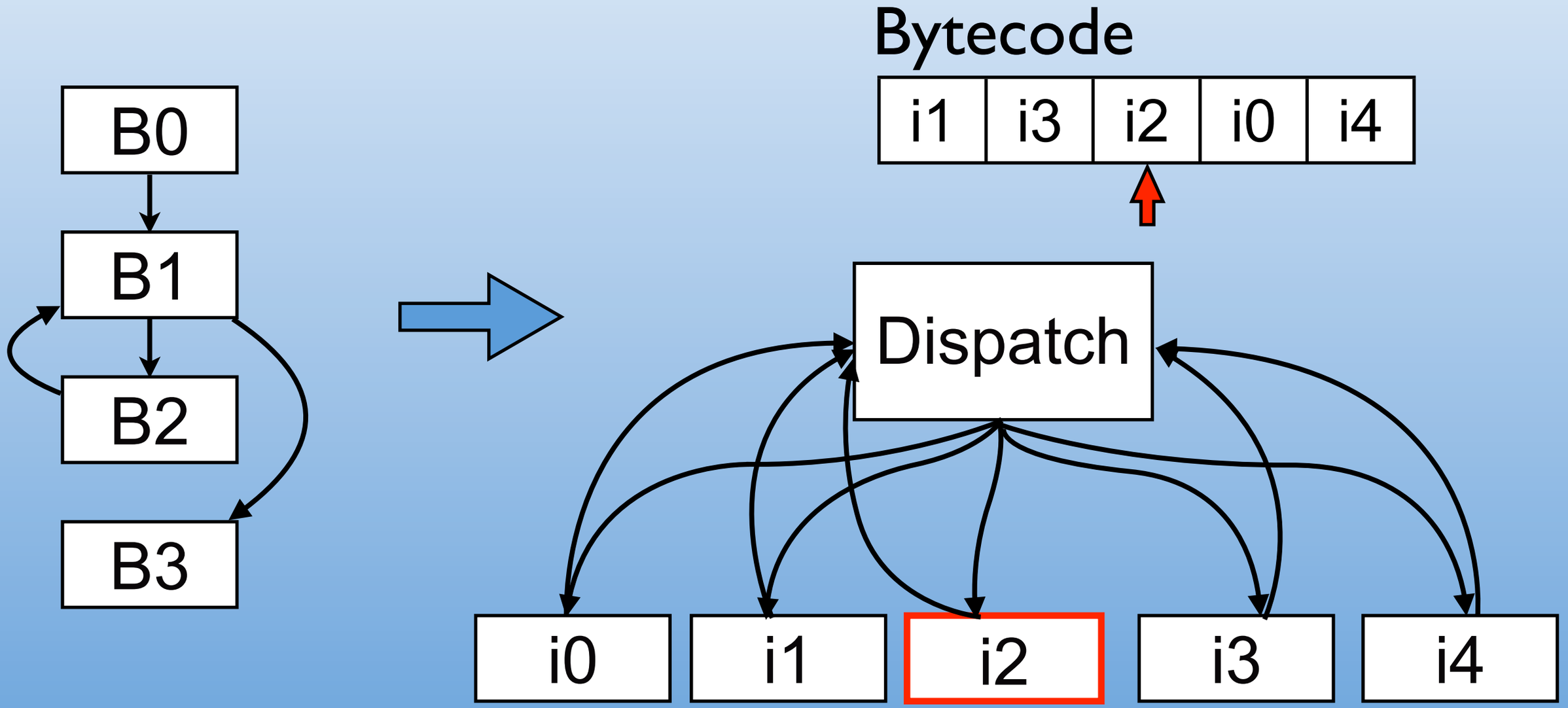
Emulation-Based Obfuscation



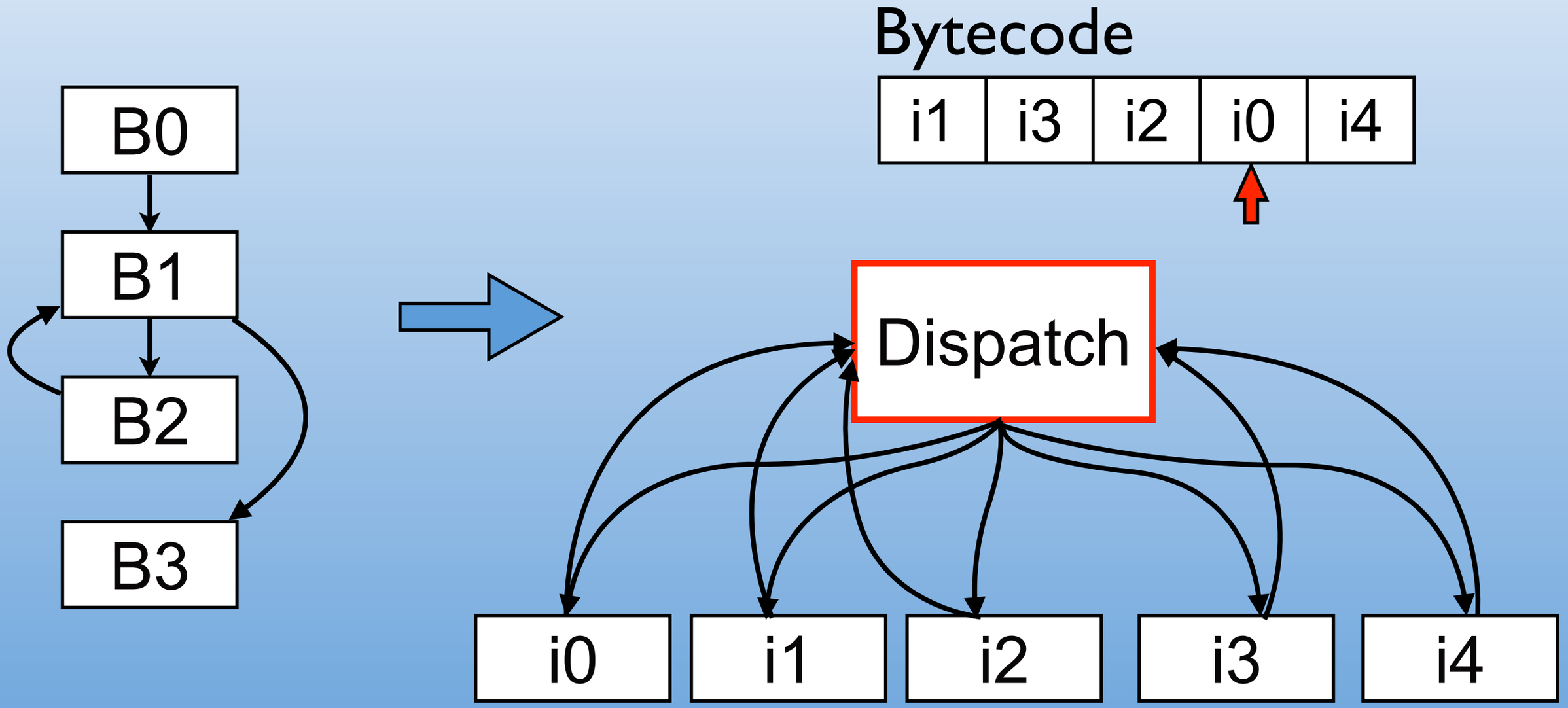
Emulation-Based Obfuscation



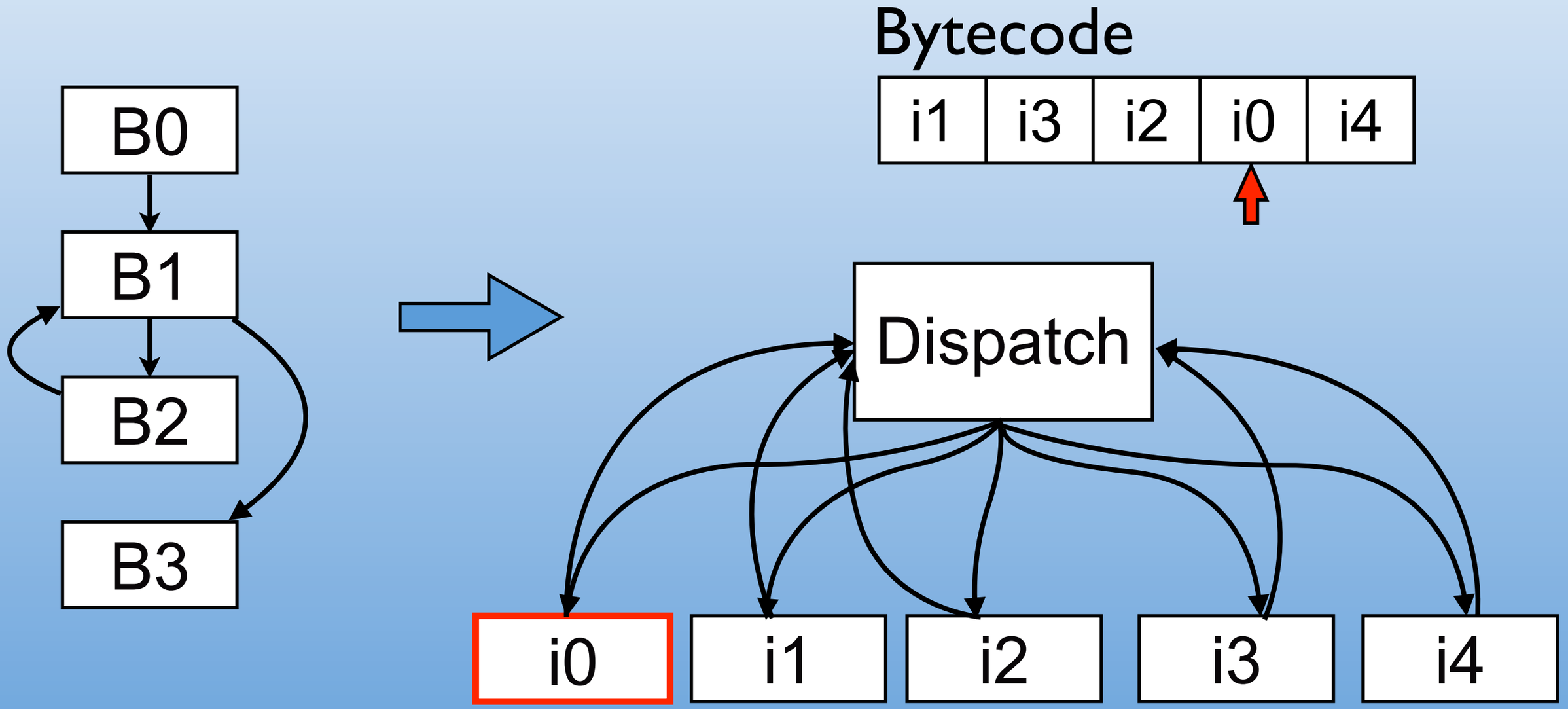
Emulation-Based Obfuscation



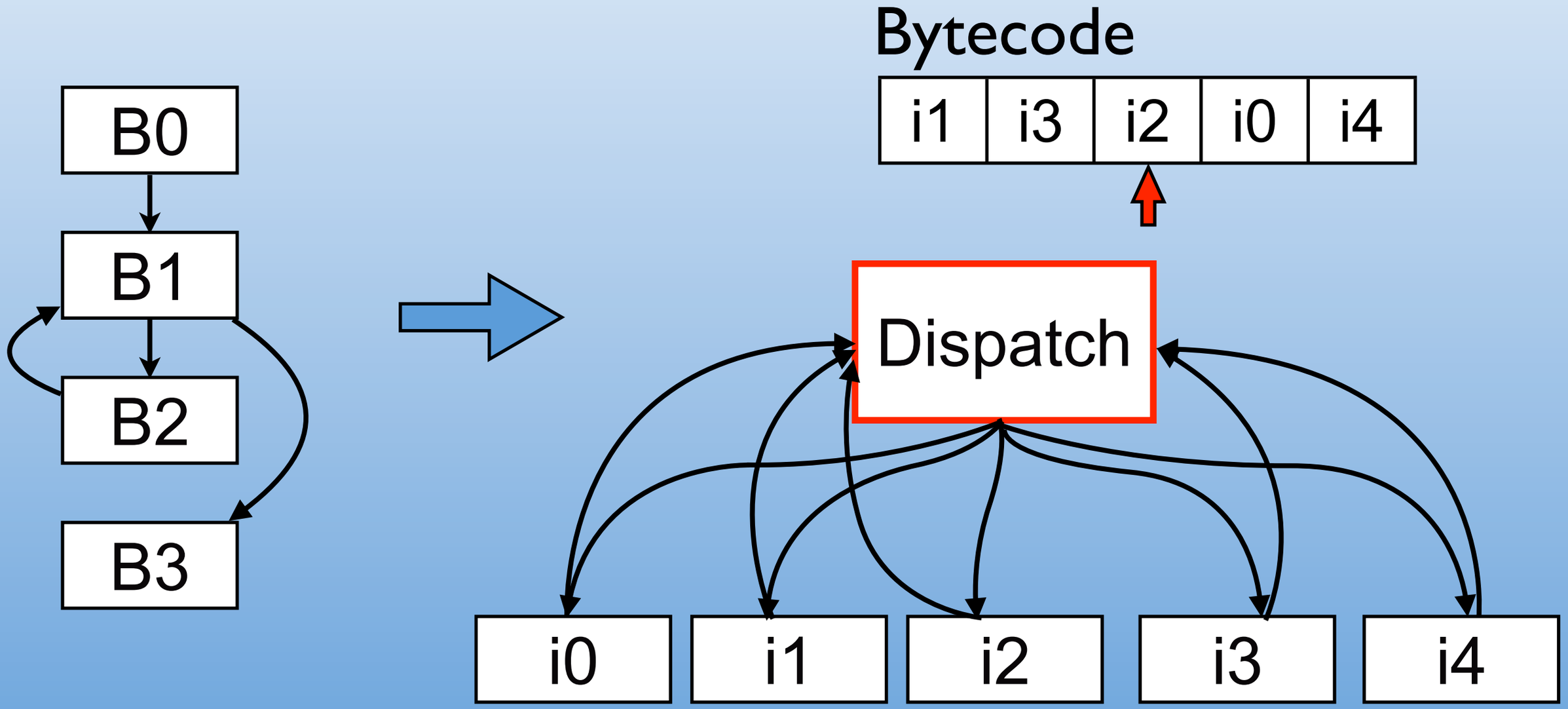
Emulation-Based Obfuscation



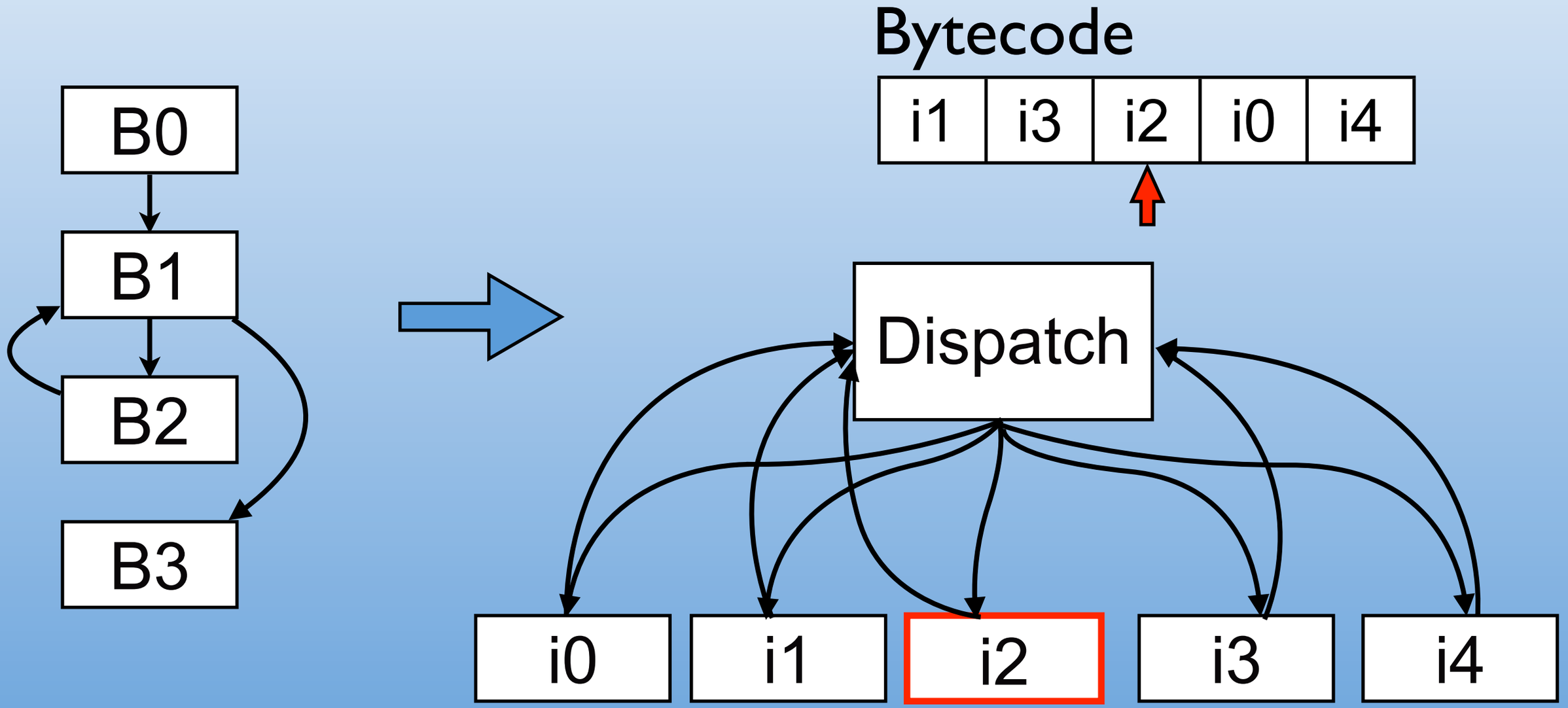
Emulation-Based Obfuscation



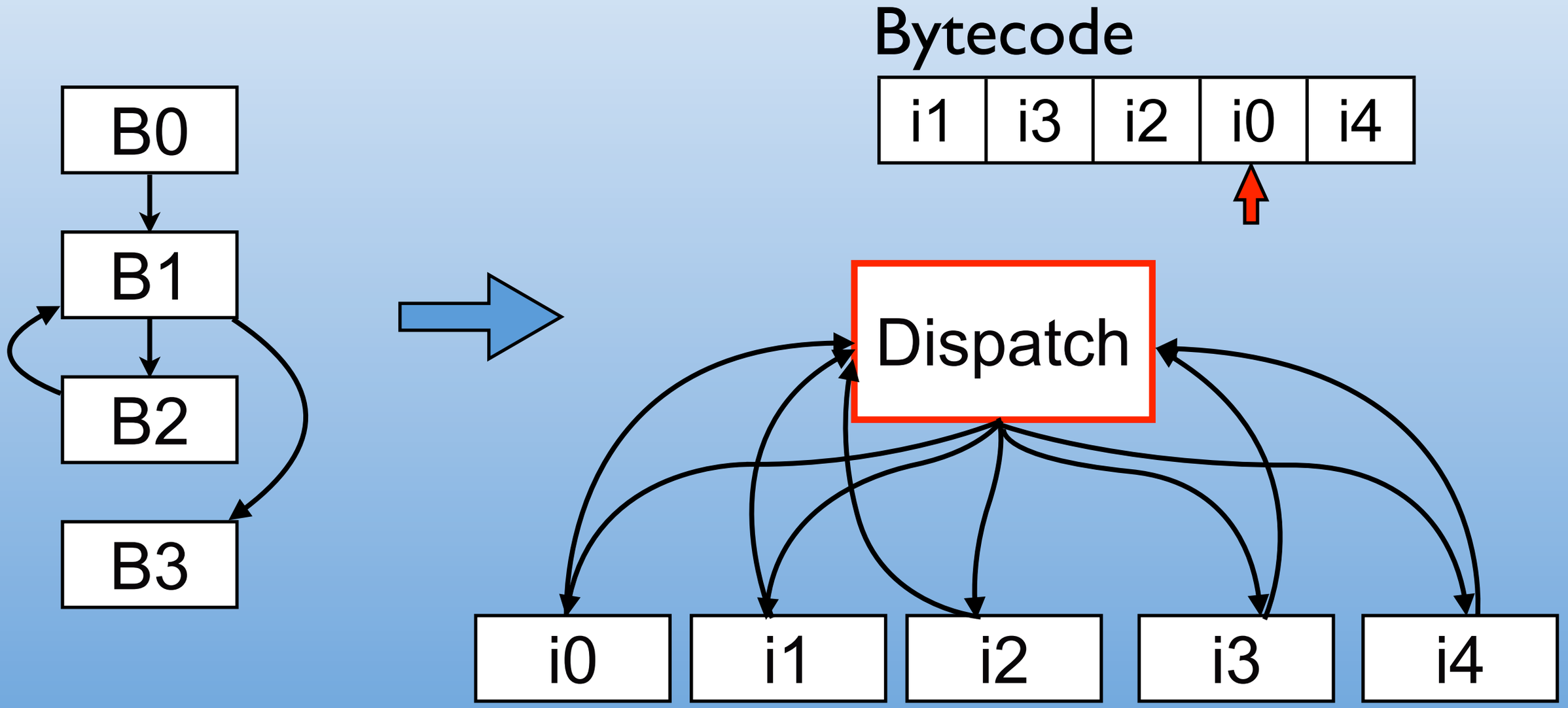
Emulation-Based Obfuscation



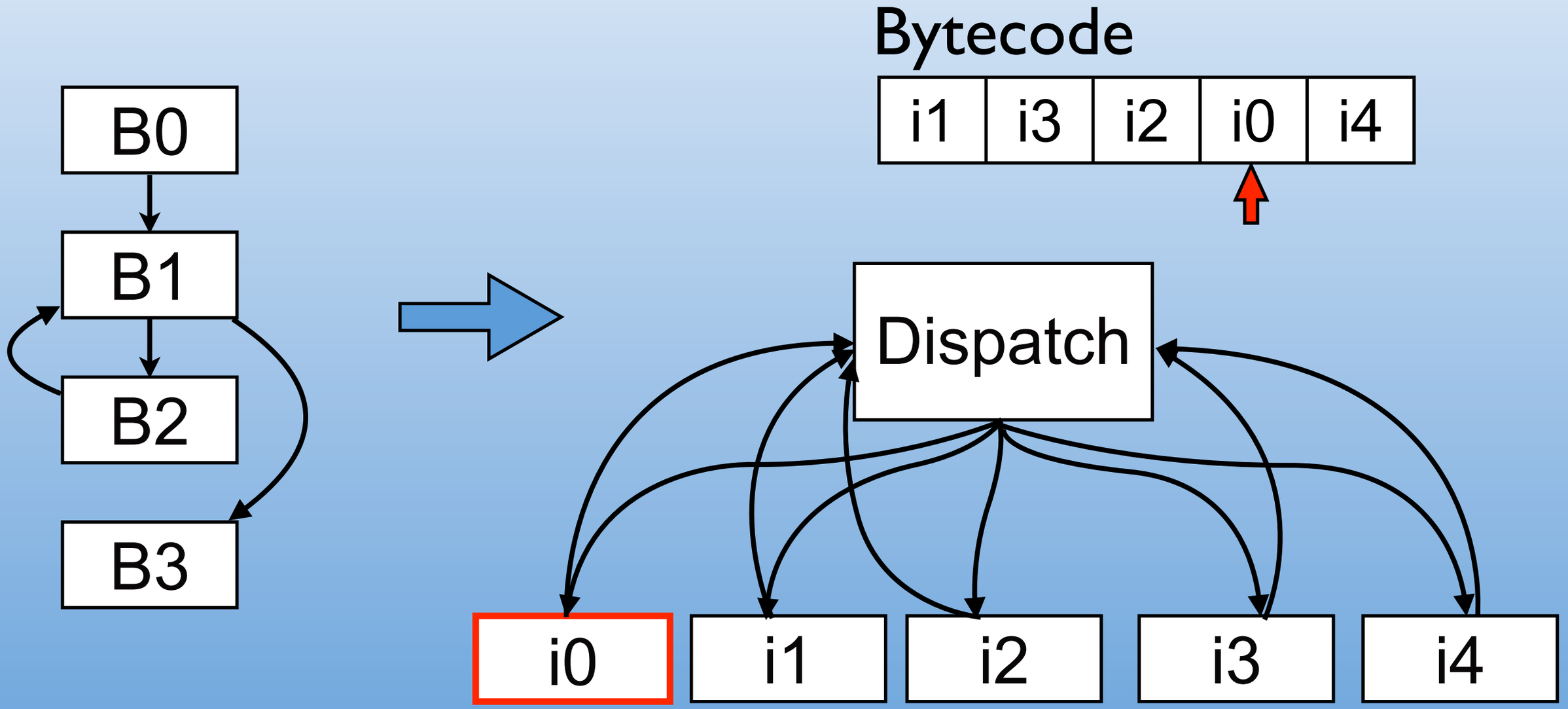
Emulation-Based Obfuscation



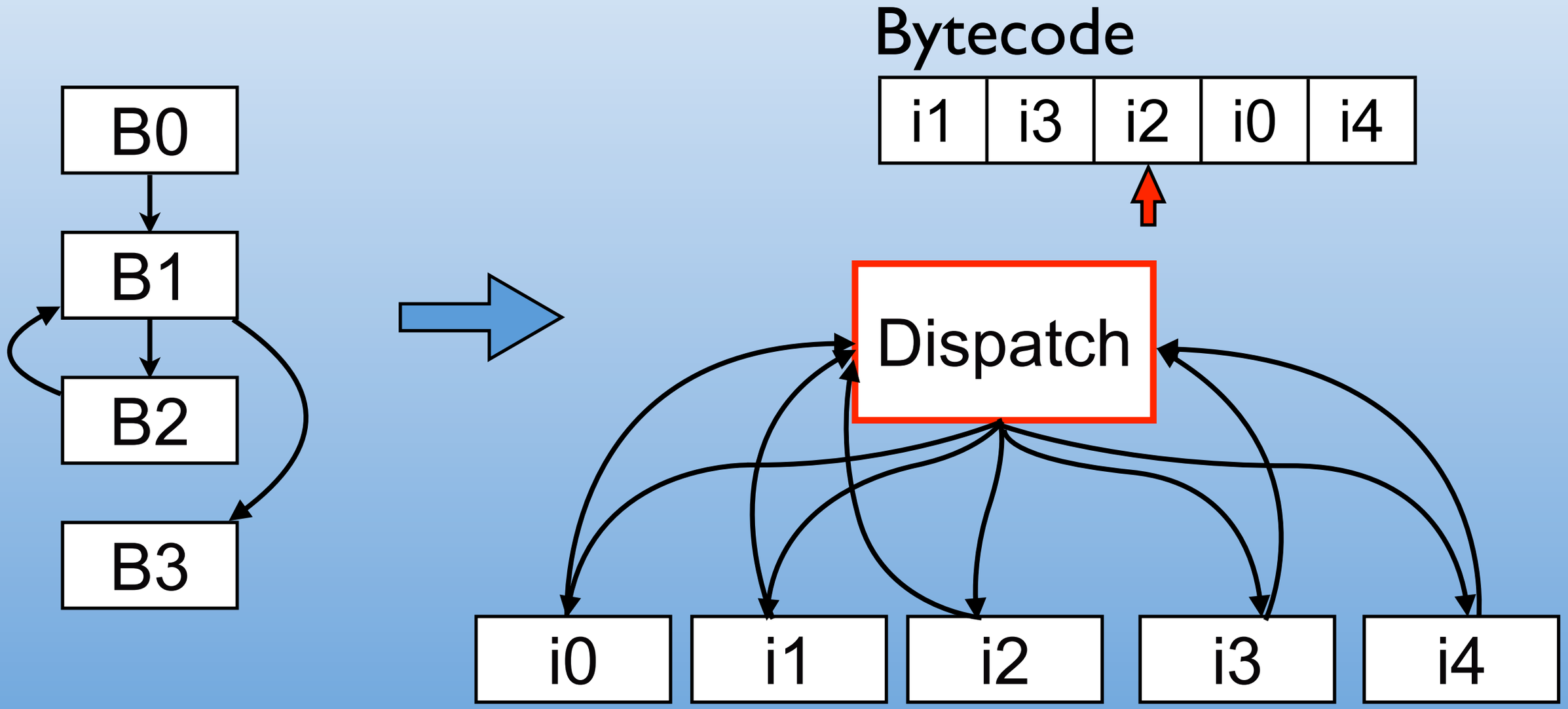
Emulation-Based Obfuscation



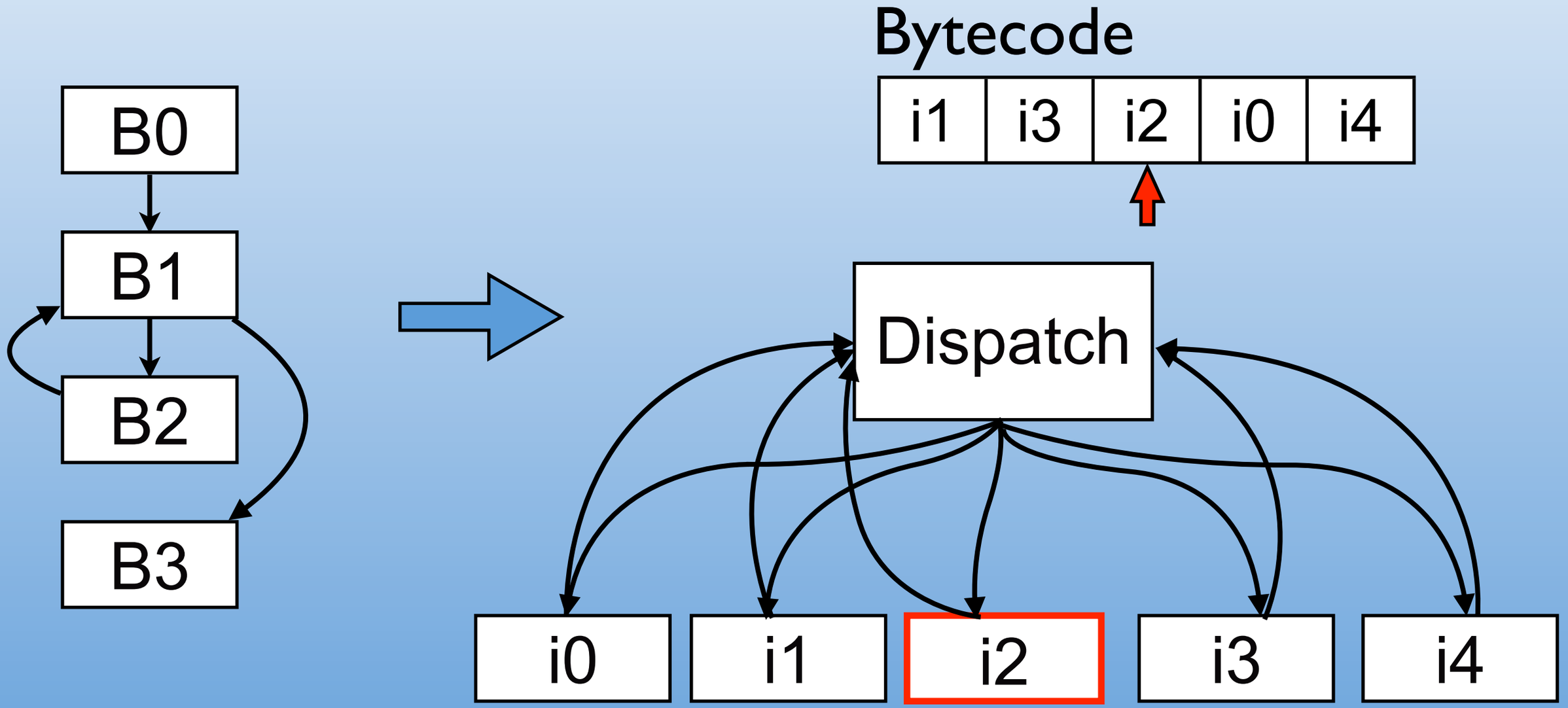
Emulation-Based Obfuscation



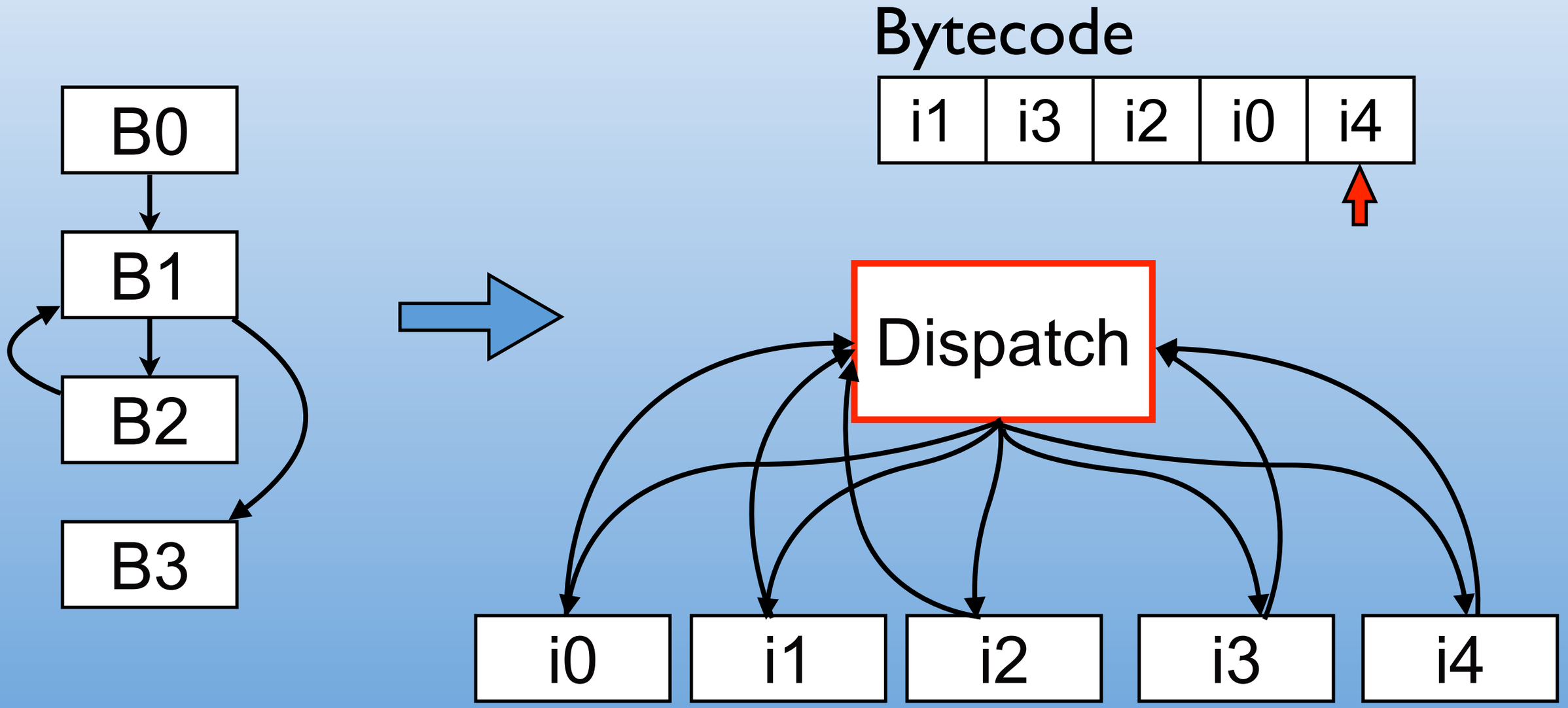
Emulation-Based Obfuscation



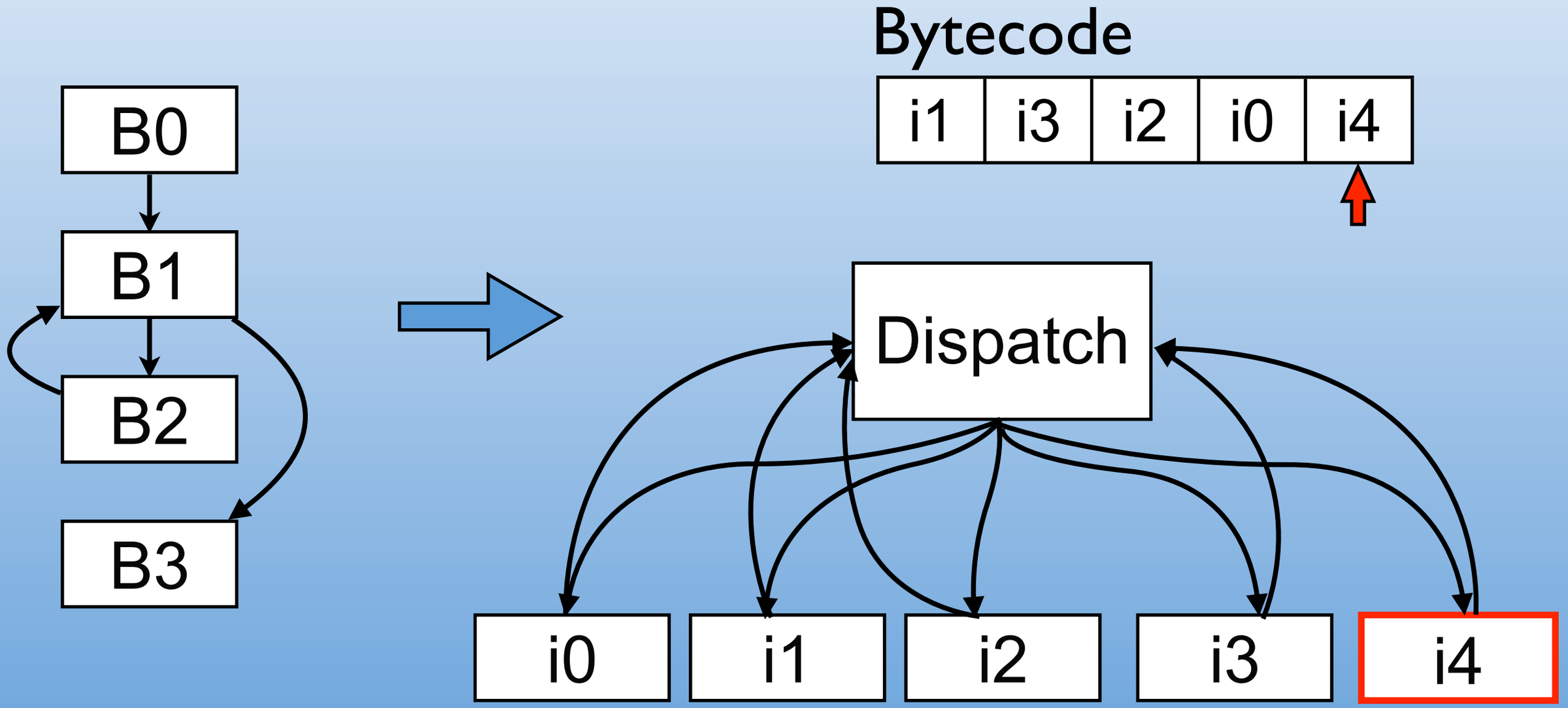
Emulation-Based Obfuscation



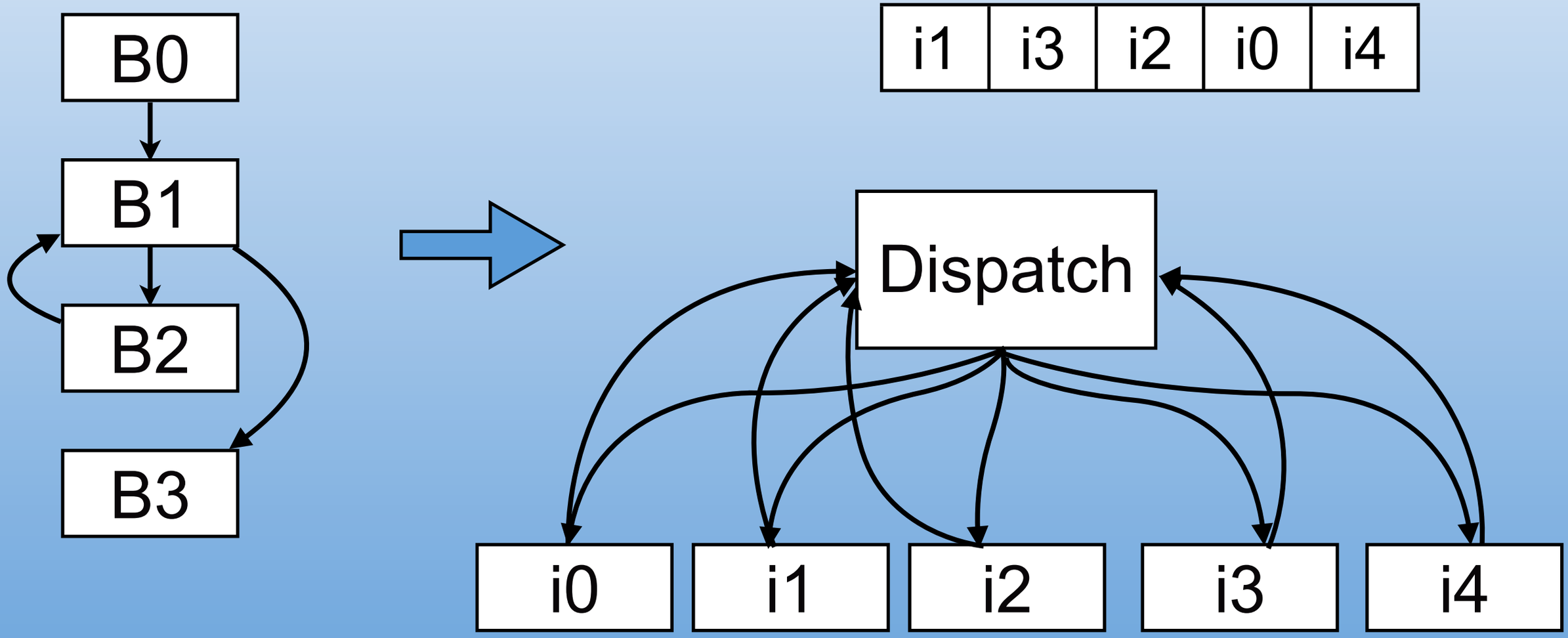
Emulation-Based Obfuscation



Emulation-Based Obfuscation



Emulation-Based Obfuscation



Emulation-Based Obfuscation

- Fundamentally different from string obfuscation
- Original program is hidden under interpreter's logic
- Looks like normal, unobfuscated program

Anti-analysis Defense

- Whether to conduct attack is determined at runtime
- Detecting detection/analysis systems:
  - *behavioral differences*
  - *execution overhead*
  - ...

Implicit Conditionals

- Hiding possible branch targets from multi-path exploration
  - branch without conditionals
- Emulation-based obfuscation?

Implicit Conditionals

- Replacing conditionals with calculation of parameters used by the interpreter
  - *conditional-free*
- Emulation-based obfuscation
  - *semantics-preserving*

Implicit Conditionals

```
if test(v)
   then C
```

transform →

```
p = comp(v)
interpret(p,code)
```

- comp(v) returns desired value *iff* test(v) holds `true`
- `code` is the bytecode representation of `C`, which is arranged based on the desired value of `p`

Possible choices of parameter:

- Bytecode entry point
- Instruction pointer increment value

Implicit Conditionals

Example:

```
bytecode = [b1, b2, ...];
p = hash(navigator.userAgent);
ip = parseInt(p[4] + p[11]);
inc = parseInt(p[13] + p[7]);
while (ip < bytecode.length) {
    execute(bytecode[ip]);
    ip = ip + inc;
}
```

Experimental Evaluation

- Target systems
  - VirusTotal
    www.virustotal.com
  - Zozzle
    Curtsinger et al.
  - Wepawet
    Cova et al.
- Testcases
  - 7 malware samples
  - 2 versions for each sample
    - code unpacking
    - proposed protection

Detection Results

malware Sample	Code Unpacking			Proposed Protection		
	VirusTotal	Wepawet	Zozzle	VirusTotal	Wepawet	Zozzle
M_1	5/40	detected	not detected	0/42	not detected	not detected
M_2	4/41	detected	not detected	0/42	not detected	not detected
M_3	5/42	detected	detected	0/42	not detected	not detected
M_4	5/42	detected	detected	0/42	not detected	not detected
M_5	5/42	detected	not detected	0/42	not detected	not detected
M_6	5/42	detected	not detected	0/42	not detected	not detected
M_7	10/42	detected	n/a	2/42	not detected	n/a

Discussion

- Can we adjust existing tools against proposed protection?
- Possible attack models

Conclusion

- Existing detection techniques are closely tied to current malware and protection.
- Can we defend future attacks regardless of obfuscation/protection?

THE UNIVERSITY OF ARIZONA

Thank you