Watermarking Java Bytecode

Anya Berdichevskaya

Advisors: Samuel Wagstaff

Mikhail Atallah

Why Watermark Java Bytecode?

- With all the possibilities of moving Java software around the internet, there is a great need in use of software that will:
  - Protect one's intellectual property
  - Discourage intellectual property theft
  - Disallow unauthorized access
  - Prove ownership

What is a Watermark in a Java Bytecode?

- Watermark is a secret that
  - Is embedded into a Java class file
  - Does not change the behavior of the executable
  - Is not easily found or removed
  - Is not noticed by the user

Where to Insert a Watermark?

- Possibilities: Java bytecode class file has two major arrays:
  - Constant_pool – constants stored here
  - Attributes – code blocks are here
- We chose:
  - Attributes – largest part of the class file; easier to “hide”

How to Insert a Watermark?

- Get a hash of a method
- Compare the last bit of the hash to the bit of the watermark
  - If same: repeat for the next method
  - If different: use BLOAT to add instructions and rehash

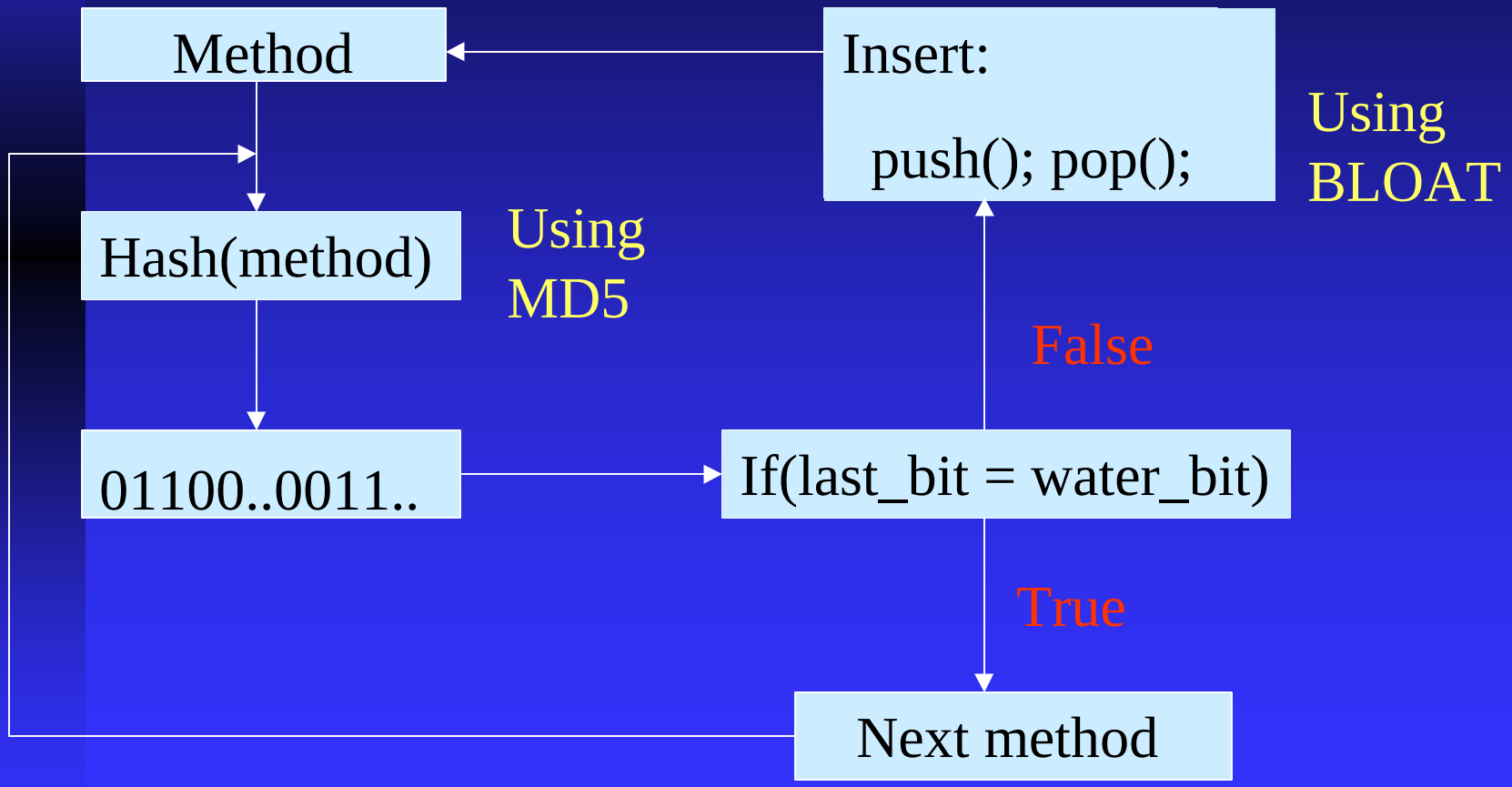
What is a Hash Function?

- Mathematical function that takes an input string and converts it to a shorter output string
- Works one direction: very hard to generate a string that hashes to the same hash value
- Slight change in input string = radical change in output string
- We use MD5

What is BLOAT?

- BLOAT: Bytecode-Level Optimization and Analysis Tool
- Useful for optimizing Java class files
- Able to reorder and modify instructions
- We use it to add instructions:
  - Push()
  - Pop()

Algorithm


Method
Insert:
push(); pop();
Using
BLOAT
Hash(method)
Using
MD5
False
01100..0011..
If(last_bit = water_bit)
True
Next method

Drawbacks

- Not all methods can be watermarked:
  - Native methods
  - Abstract methods
- Not all parts of Java bytecode are watermarked
- Reorder of methods -> invalid watermark
- Only hides one bit, per method