S C I E N C E ▪ P A S S I O N ▪ T E C H N O L O G Y

TU Graz

Institute for Technical Informatics ▪ Inffeldgasse 16/I, 8010 Graz, Austria ▪ *office@iti.tugraz.at*

Memory-Efficient On-Card Byte Code Verification for Java Cards

Reinhard Berlach, Michael Lackner and Christian Steger

Institute for Technical Informatics, Graz University of Technology
{reinhard.berlach, michael.lackner, steger}@tugraz.at

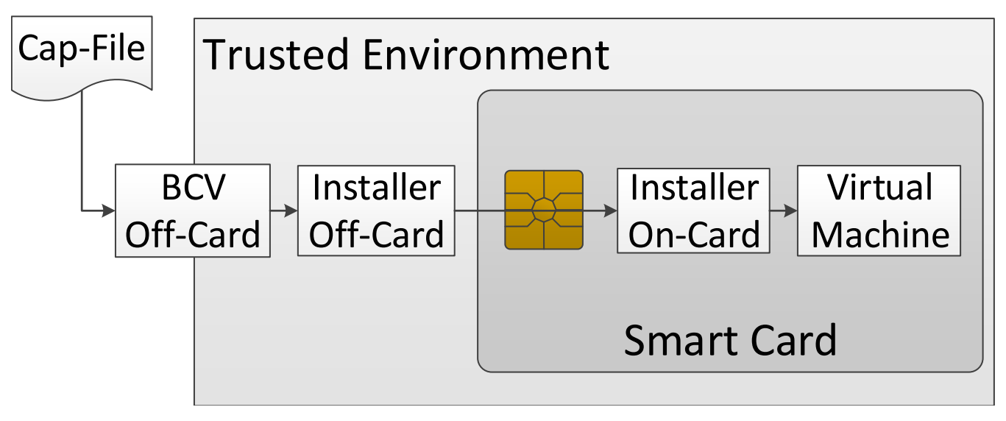
Johannes Loinig and Ernst Haselsteiner

NXP Semiconductors
{johannes.loinig, ernst.haselsteiner}@nxp.com

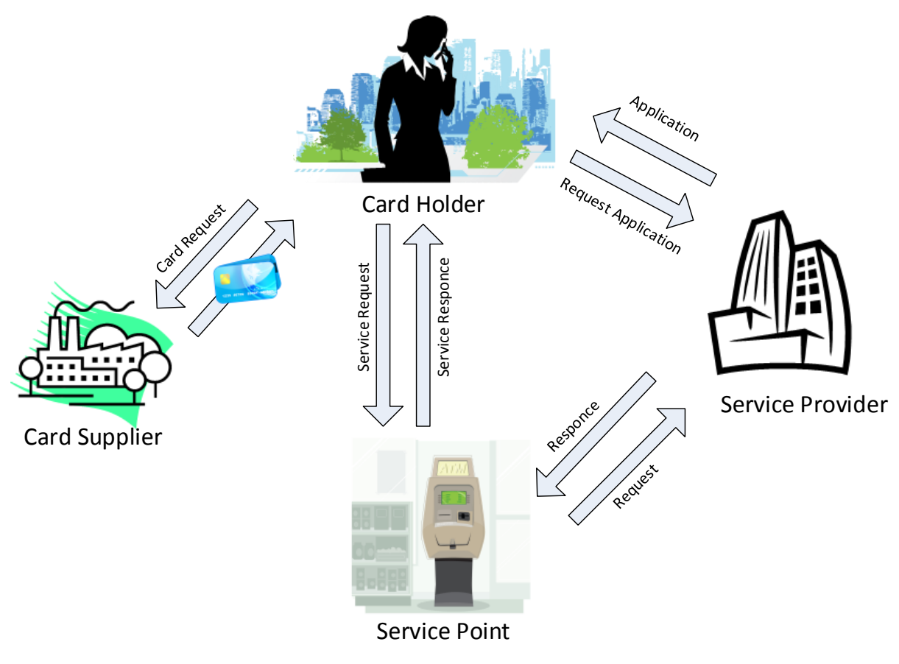
Motivation

Java Card Security [7, 10]

- Bytecode
  - Verification (BCV) [4, 8]
  - Off-Card
  - Resource intense algorithm
- Secure Loading
  - Off and On-Card Component
  - Done by Cryptographic Signature
    - Key-exchange between Card Supplier and Issuer

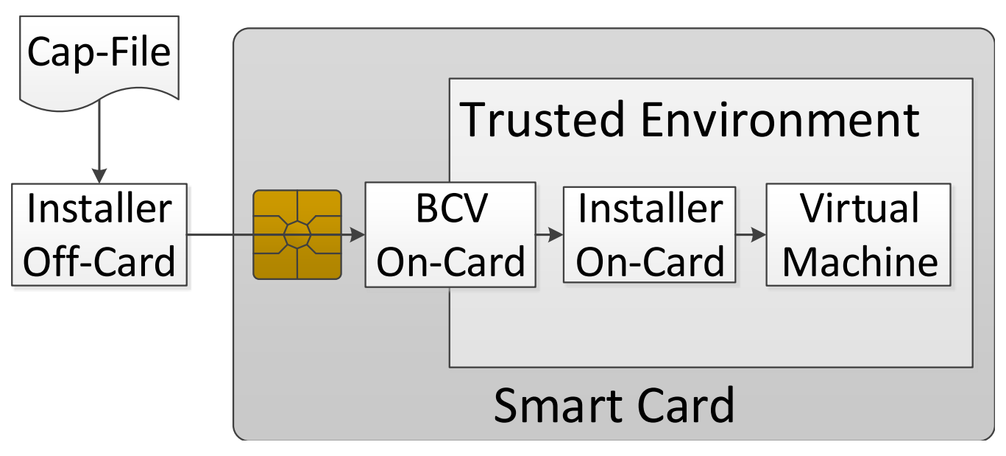


User Centric Ownership Model [1]



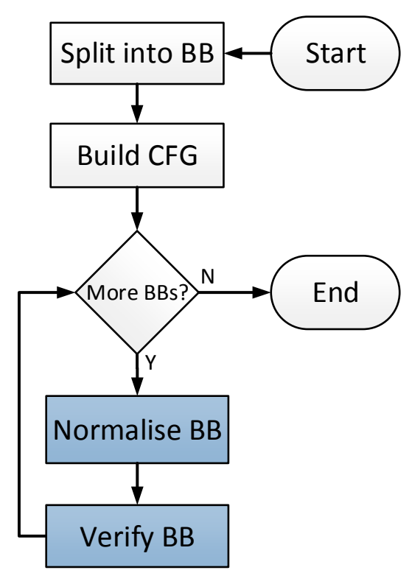
Overview of the User Centric Ownership Model [1]

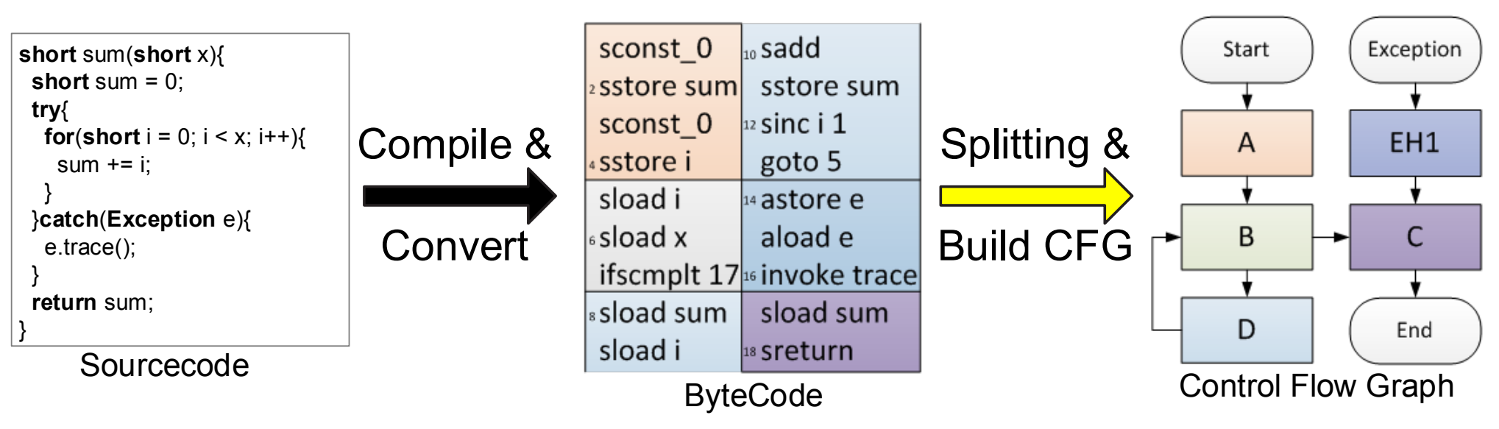
- No Secure Loading
  - No Business relationship between Card Supplier and Issuer
- Needs On-card BCV



Memory-Efficient BCV

- Working on Basic Blocks
  - Combining Normalising and CFG
  - BB is smallest verifiable unit
- Building CFG
  - On-Card
  - In linear time
  - Reuse of Objects to minimize memory usage





```
short sum(short x){
  short sum = 0;
  try{
    for(short i = 0; i < x; i++){
      sum += i;
    }
  }catch(Exception e){
    e.trace();
  }
  return sum;
}
```

Sourcecode

Compile & Convert

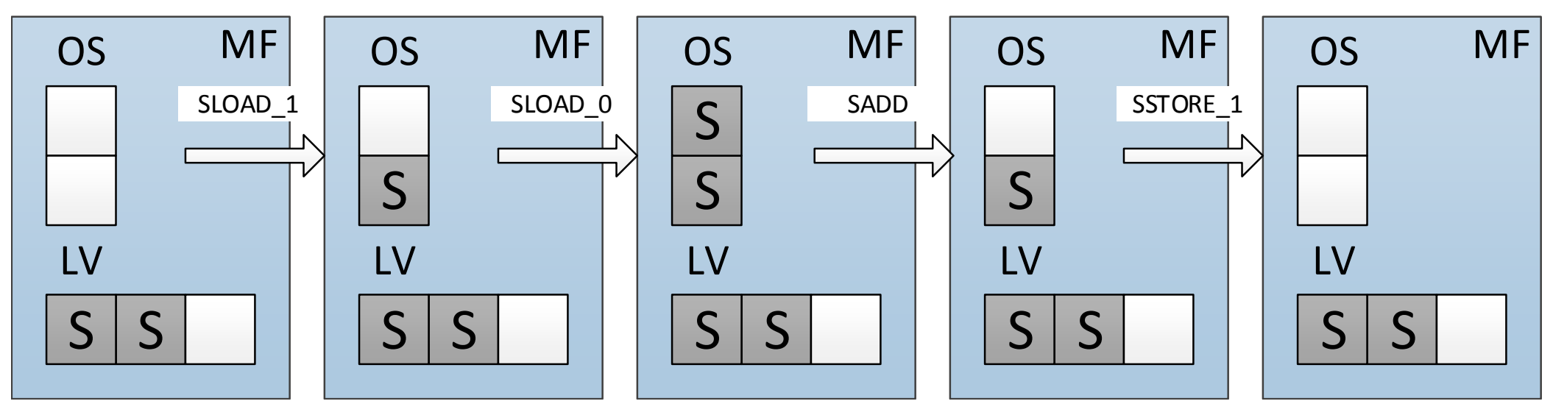
```
  sconst_0       10 sadd
2 sstore sum        sstore sum
  sconst_0       12 sinc i 1
4 sstore i          goto 5
  sload i        14 astore e
6 sload x           aload e
  ifscmplt 17    16 invoke trace
8 sload sum         sload sum
  sload i        18 sreturn
```

ByteCode

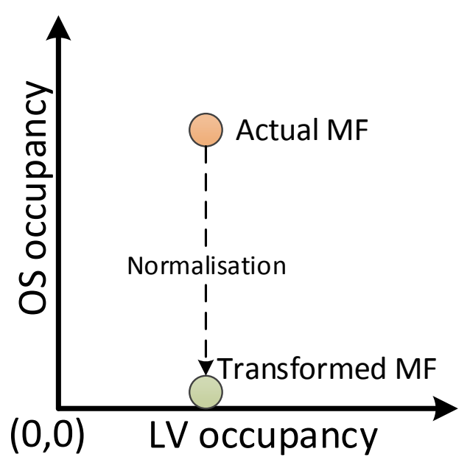
Splitting & Build CFG

Control Flow Graph

- Abstract Interpretation
  - On-Card
  - Working on BB

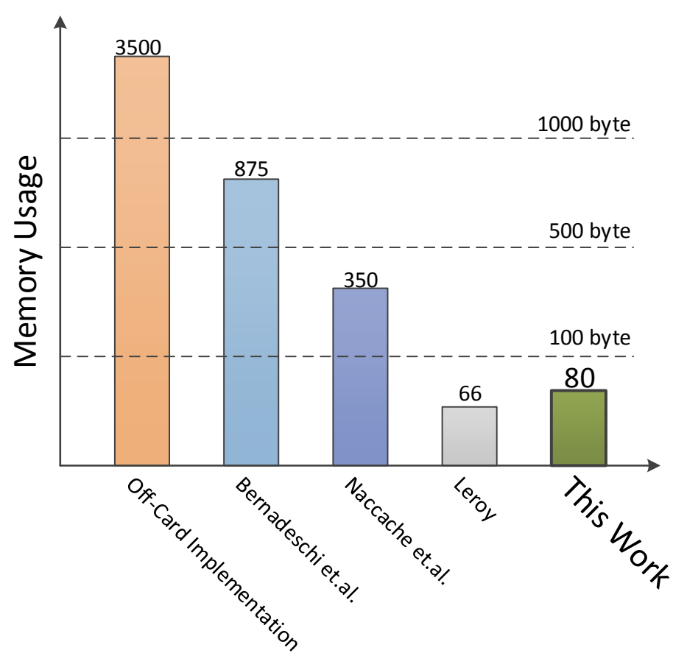


- Temporary Normalisation
  - On-Card
  - Not changing execution of Application



Conclusion

- On-Card
  - Algorithm running on-card
  - Standard Compliance
- Temporary Normalisation
  - Reducing Memory conumption
  - Usable also on low-cost Smart Cards



References

[1] R. Akram, K. Markantonakis, and K. Mayes. A Paradigm Shift in Smart Card Ownership Model. In International Conference on Computational Science and Its Applications (ICCSA), March 2010.

[2] C. Bernardeschi, L. Martini, and P. Masci. Java bytecode verification with dynamic structures. In International Conference on Software Engineering and Applications (SEA), Cambridge, MA, USA, 2004.

[3] D. Deville and G. Grimaud. Building an „impossible" verifier on a java card. In Proceedings of the 2nd conference on Industrial Experiences with Systems Software - Volume 2, Berkeley, CA, USA, 2002. USENIX Association.

[4] J. Gosling. Java intermediate bytecodes. ACM SIGPLAN workshop on intermediate representations (IR'95), 30(3):111-118, 1995.

[5] X. Leroy. Bytecode verification on Java smart cards. Software: Practice and Experience, 32(4):319-340, 2002.

[6] D. Naccache, A. Tchoulkine, C. Tymen, and E. Trichina. Reducing the memory complexity of type-inference algorithms. In Information and Communications Security, volume 2513 of Lecture Notes in Computer Science, pages 109-121. Springer Berlin / Heidelberg, 2002.

[7] Oracle. Virtual Machine Specification. Java Card Platform, Version 3.0.4, Classic Edition, 2011.

[8] Oracle. Java card 3 platform off-card verification tool specification, classic edition. Beta Draft Version 1.0, Oracle, February 2012.

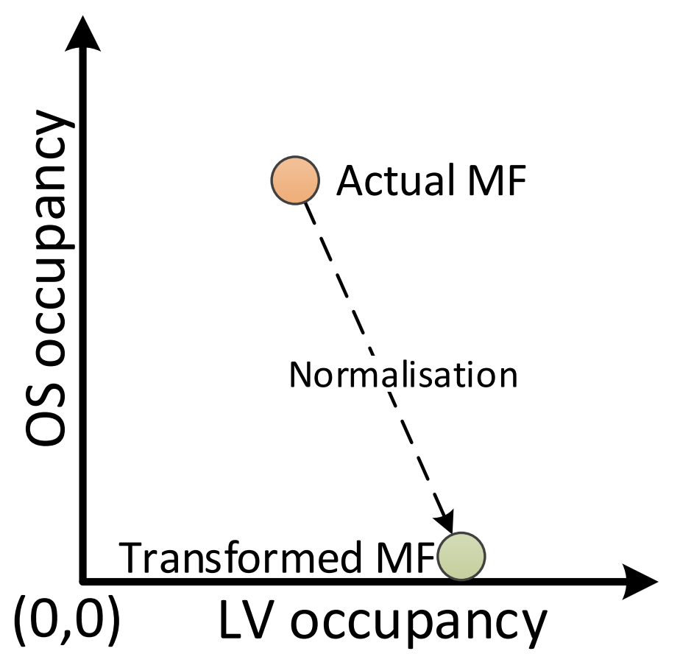
[9] E. Rose and K. H. Rose. Lightweight Bytecode Verification. Journal of Automated Reasoning, 31:303-334, 2003.

[10] M. Witteman. Java Card Security. Information Security Bulletin, July 2003.

Related Work

Byte Code Verification

- Original BCV [4, 8]
  - Off-Card
  - Resource intense algorithm
  - Abstract interpretation
  - Part of the Sandbox Concept of Java



Normalising in the MF-Plane [5]

On Card BCV

- Proof Carrying Code (PCC) [9]
  - Needs Off-Card Components
  - Verification in Single pass
  - +50% size for PCC
- Normalising [5]
  - Needs Off-Card Components
  - Same memory consumption as execution
- Reducing the Dictionary [2, 6]
  - Using Control Flow Graphs
  - Minimizing saved elements of Dictionary

This project is sponsored by . . .

NXP

FFG

NFC

www.iti.tugraz.at