# Fast, Scalable and Secure Onloading of Edge Functions Using AirBox

2016

Ketan Bhardwaj, Ming-Wei Shih, Pragya Agarwal, Ada Gavrilovska, Taesoo Kim, Karsten Schwan
College of Computing, Georgia Institute of Technology, Atlanta, GA

Presentation by Francis Rinaldi

## Edge Computing Improves User Experience

- Lower Access Latency to Backend Services
- Lower Cost of Accessing Backend Services
- Addresses Resource Constraints of Mobile Devices

## Client-Driven Cyber-foraging

- Offloads Computation from Client
- Challenge to Control or Predict the Internet Access Latency
- Difficult to Create for Diverse Range of Clients

## Backend-Driven Cyber-foraging

- Onloads Benefits from Server
- Can Recognize Usage Patterns to Optimize Onloading
- Can Aggregate Client Requests to Reduce Internet Traffic

Edge Function (EF)

Edge Function Platform (EFP)

## Goals of Sufficient EFP - Designing EFs

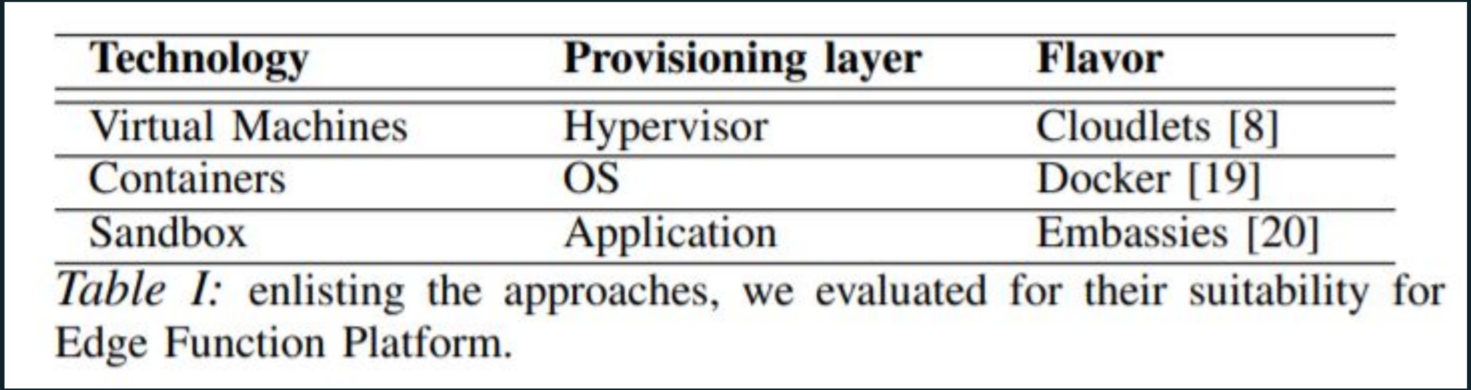
- Constraints are Few to None
- Based At Most on Standard System Libraries
- Possible Support for Common Software Patterns

## Goals of Sufficient EFP - Provisioning

- Just In Time Provisioning
- Low Compute Time
- Good Scaling

## Goals of Sufficient EFP - Security

- Critical Functionalities
  - Verification of EF Integrity
  - Confidentiality of Data
  - Secure Handling of User Traffic
- Low Overhead
- Cannot Rely on System Software



| Technology | Provisioning layer | Flavor |
|---|---|---|
| Virtual Machines | Hypervisor | Cloudlets [8] |
| Containers | OS | Docker [19] |
| Sandbox | Application | Embassies [20] |

*Table I:* enlisting the approaches, we evaluated for their suitability for Edge Function Platform.

- Developer Constraints
- Provisioning Performance
- Security and Privacy

# Design Constraints

## VM

- Develop EFs with Any OS, Libraries, and/or Applications
- Package and Send VM Image Easily

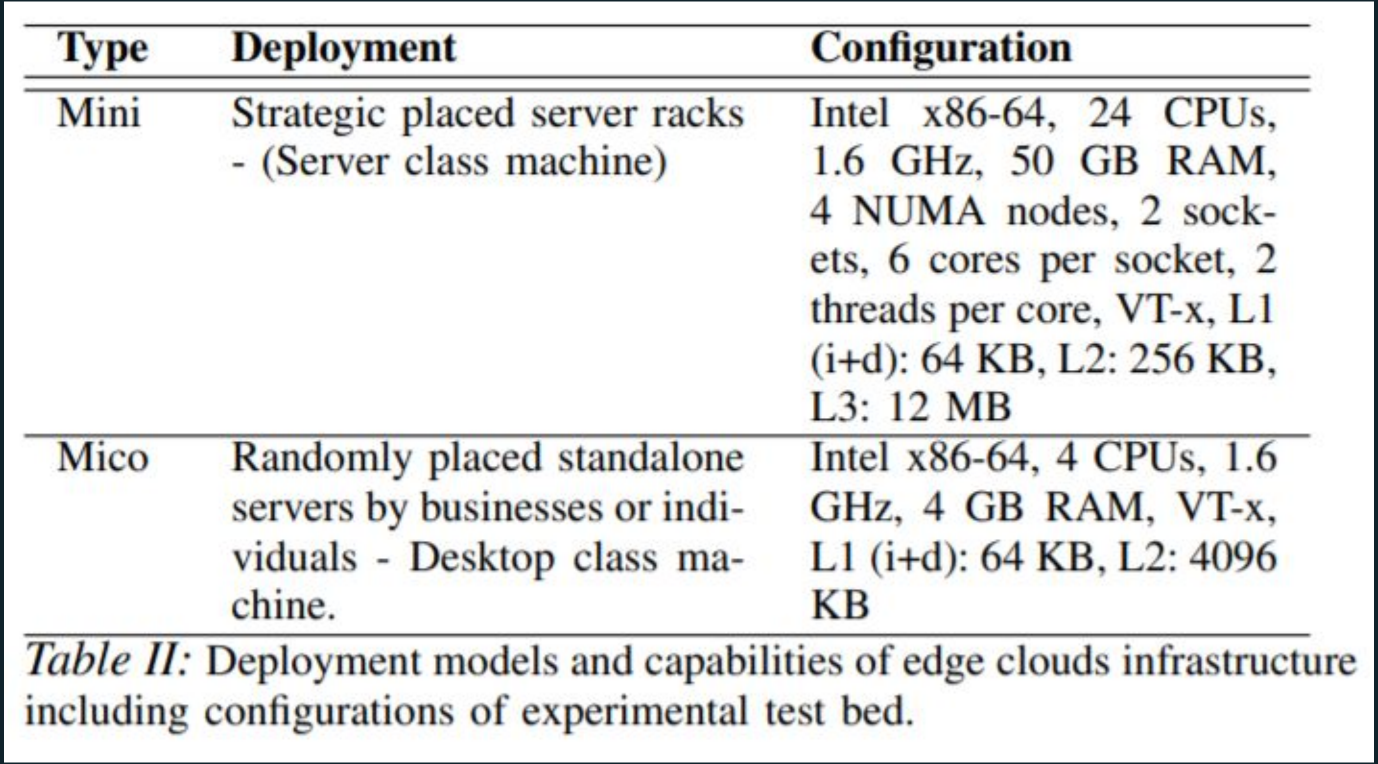
## Container

- Must Develop for a Supported OS
- Not a Large Issue

## Sandbox

- Requires Use of Specific Toolchain or Linking with a Platform-Specific ABI Library
- High Learning Curve

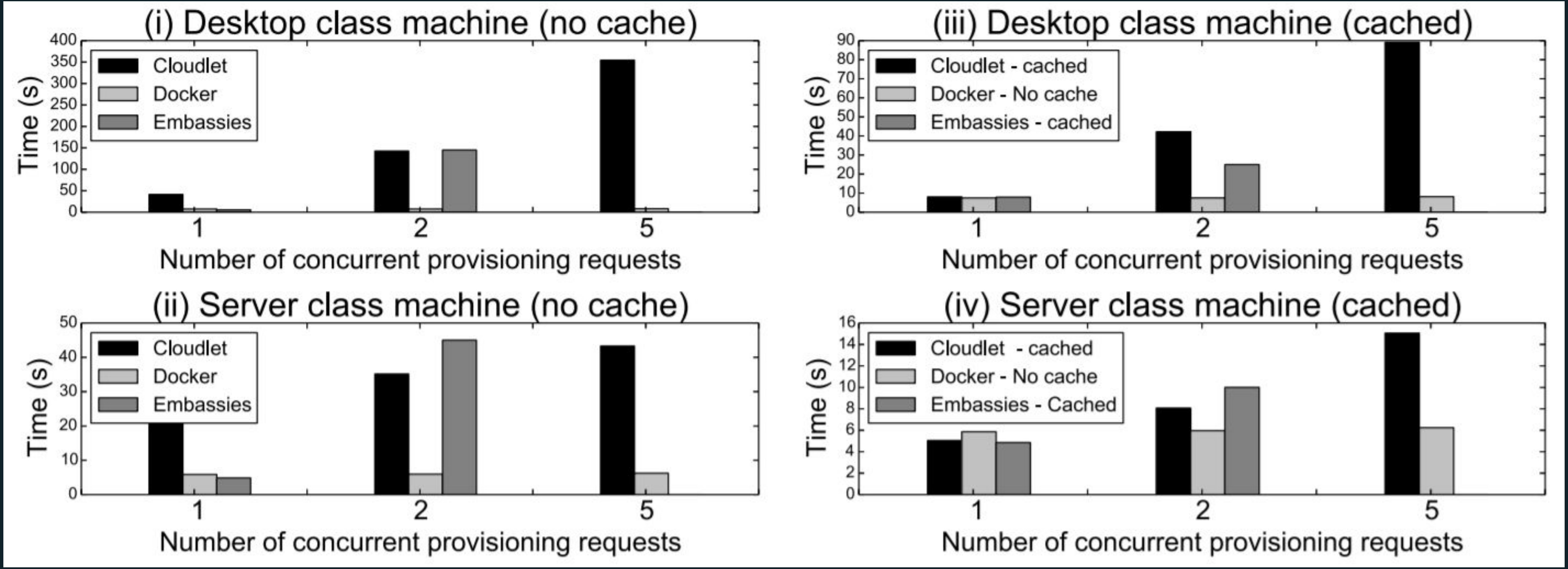
# Provisioning Performance



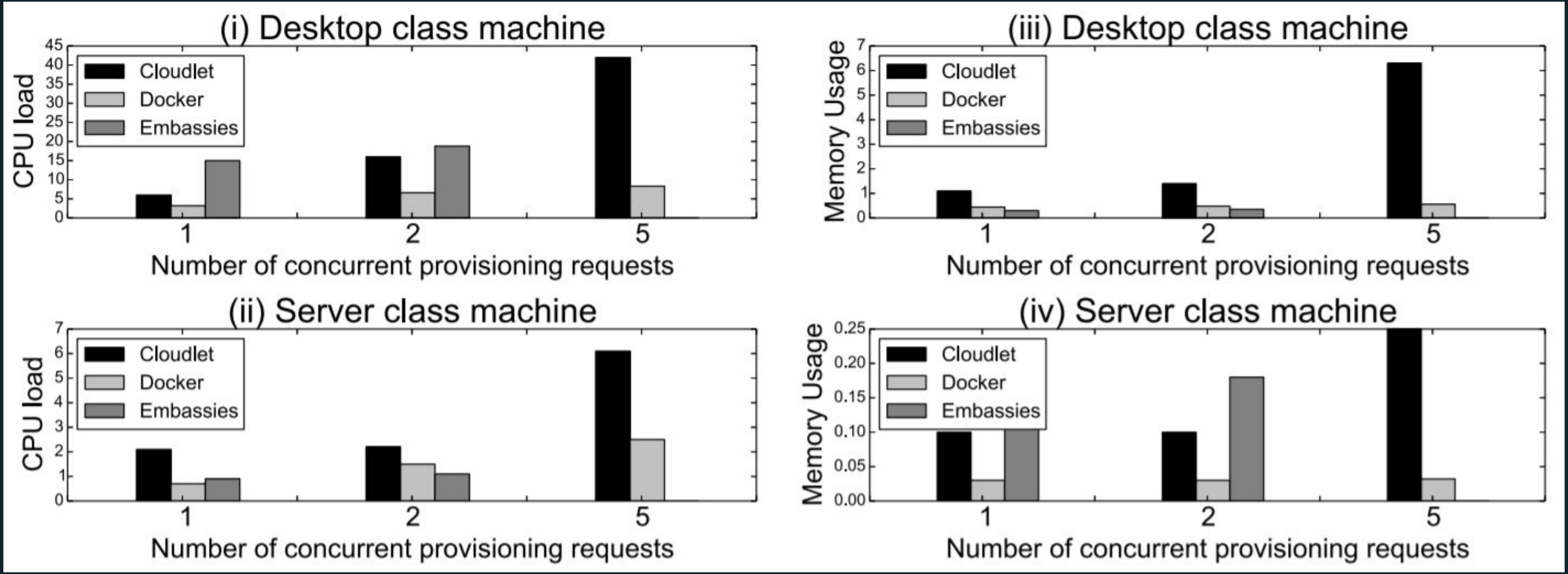
| Type | Deployment | Configuration |
|---|---|---|
| Mini | Strategic placed server racks - (Server class machine) | Intel x86-64, 24 CPUs, 1.6 GHz, 50 GB RAM, 4 NUMA nodes, 2 sockets, 6 cores per socket, 2 threads per core, VT-x, L1 (i+d): 64 KB, L2: 256 KB, L3: 12 MB |
| Mico | Randomly placed standalone servers by businesses or individuals - Desktop class machine. | Intel x86-64, 4 CPUs, 1.6 GHz, 4 GB RAM, VT-x, L1 (i+d): 64 KB, L2: 4096 KB |

*Table II:* Deployment models and capabilities of edge clouds infrastructure including configurations of experimental test bed.

# Provisioning Performance



# Provisioning Performance



# Security and Privacy

- Privileged Software “In the Wild” Cannot be Trusted by EFs
- Security Requirements:
  - Integrity Verification
  - Execution Security
  - Data Confidentiality
- Neither VMs, Containers, nor Sandboxes Can Fulfill All Requirements Alone without Trusting Privileged Software

## Security and Privacy

### VM

- No Secure Way to Verify Execution
- Cannot Use a Possibly Malicious Host Image

### Container

- Weaker Isolation
- Shared Kernel Creates a Larger Attack Surface

### Sandbox

- Stronger Security
- Shared Kernel Creates a Larger Attack Surface

# Security and Privacy

- Intel SGX Hardware Security
- Haven - Monitors all System Interaction to use SGX
  - High Overhead
  - Large Attack Surface
- VC3 - Reduces Downsides of Haven by Partitioning EFs into Trusted and Untrusted Parts

## Airbox

- Docker
  - Kernel Namespaces - Isolation
  - Cgroups - Resource Allocation and Setting Limits
  - Union File System - Application Layers

## Airbox

- Intel SGX
  - Secure ISA Extension - Enclaves
  - Remote Attestation - Remote Integrity Checks
  - Sealing - Secure Data in Non-Volatile Memory
  - Memory Protection - More Checks for Memory Access

# Airbox

- OpenSGX
  - Hardware Emulation Module
  - Operating System Emulation
  - Enclave Loader
  - User Library
  - Debugging Support
  - Performance Monitoring

## Secure Provisioning in AirBox

- Backend Service Either:
    - Creates its own EF binaries
    - Uses Available Docker Images
    - Registers a Docker Image Containing an EFT Binary and Creates a Docker File
- To Provision an EF, Sysadmins Sends Commands through AB Console to the AB Provisioners on Edge Machines
- When an EF is Booted, it Checks its Integrity with SGX’s Remote Attestation

## AirBox EF Anatomy

- EFs can be Compromised by Logged System Calls (I/O)
- SGX Enclaves have Large Overheads
- Minimizing Code Run in Enclaves is Desired
- Untrusted Part
  - Network
  - Storage
- Assumes Secure Network Protocol

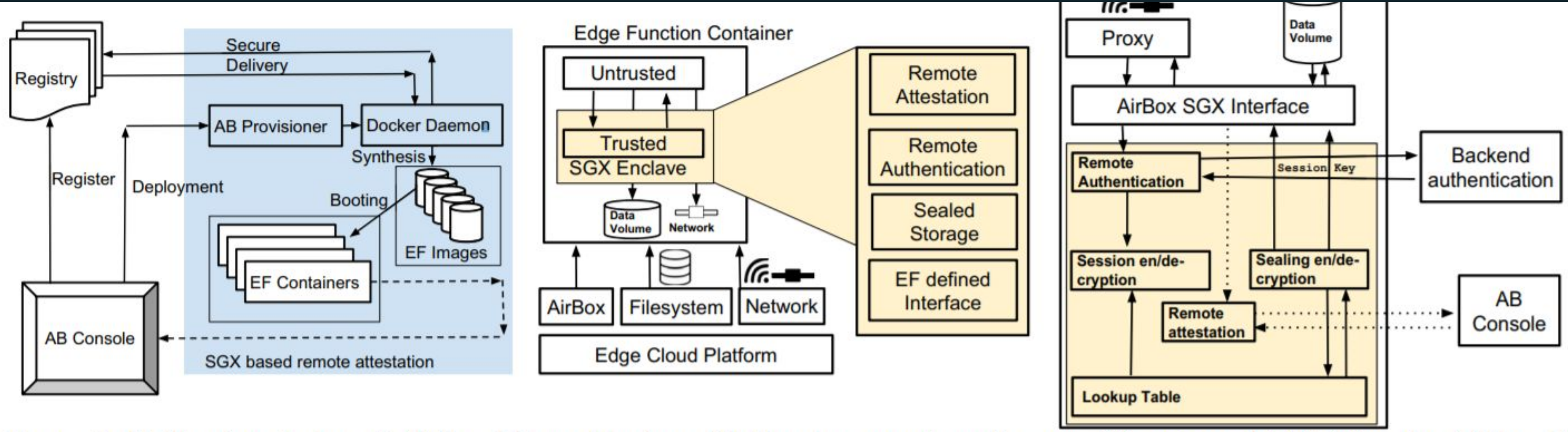


*Figure 3:* (i) Showing design of AirBox EF provisioning; (ii) Showing overview of secure EF anatomy highlighting the AirBox Open SGX interface; and (iii) implementation of a caching EF.

## State Confidentiality

- Having Trusted Storage Is Nice
- Airbox gets an Enclave Specific Sealing Key that Encrypts
- The Data can Only Be Accessed in the Enclave

## EF Implementation

- Aggregation
- Buffering
- Caching

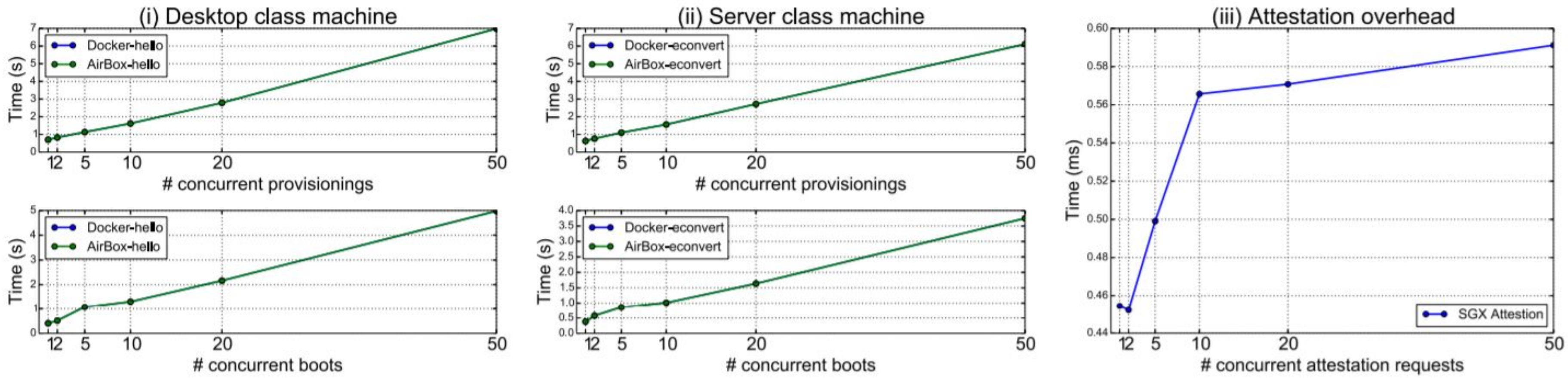


*Figure 4:* Comparing provisioning time using Docker and AirBox including SGX attestation for a hello world and an image inverting application vs. increasing number of simultaneous provisioning requests using (i) desktop class machine and (ii) server class machine; (iii) time spent in attestation command vs. number of simultaneous attestation requests using windows SDK on SGX hardware.

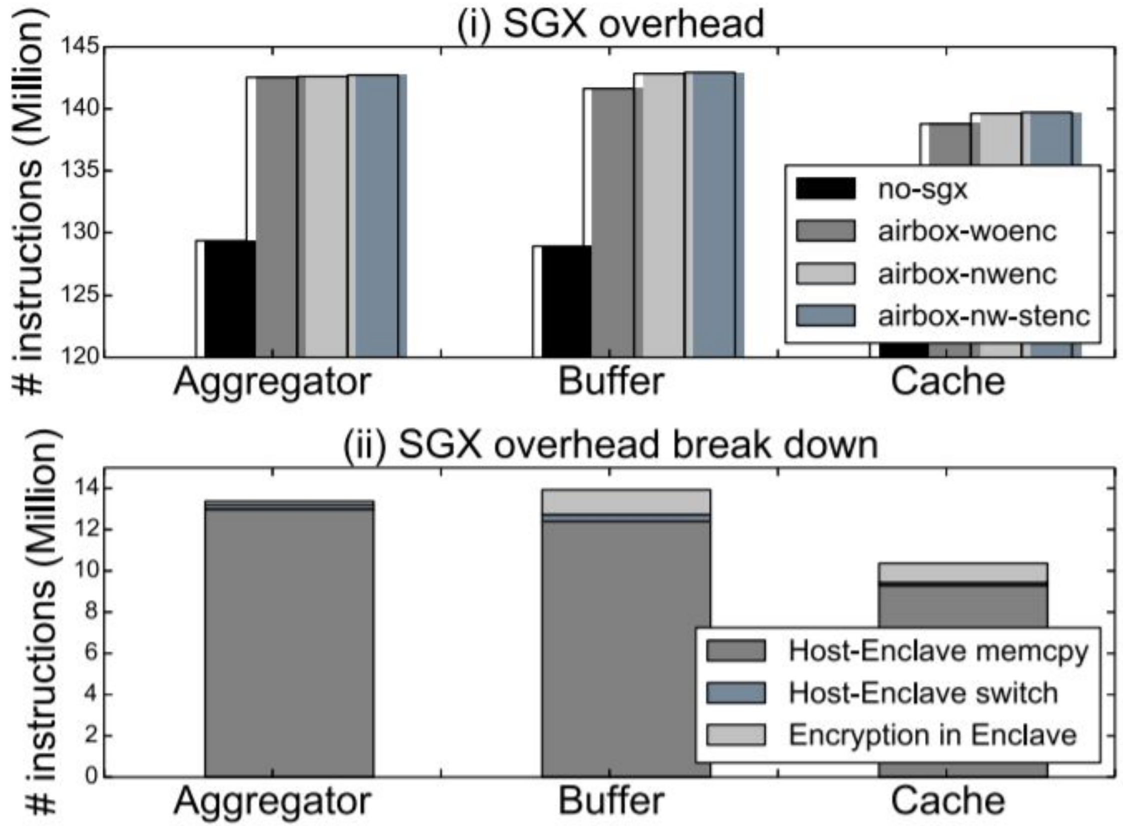


*Figure 5:* Showing (i) Airbox overhead in ABC use cases due to the use of SGX for security and privacy using OpenSGX; (ii) Showing the break down of the overhead in ABC use cases.

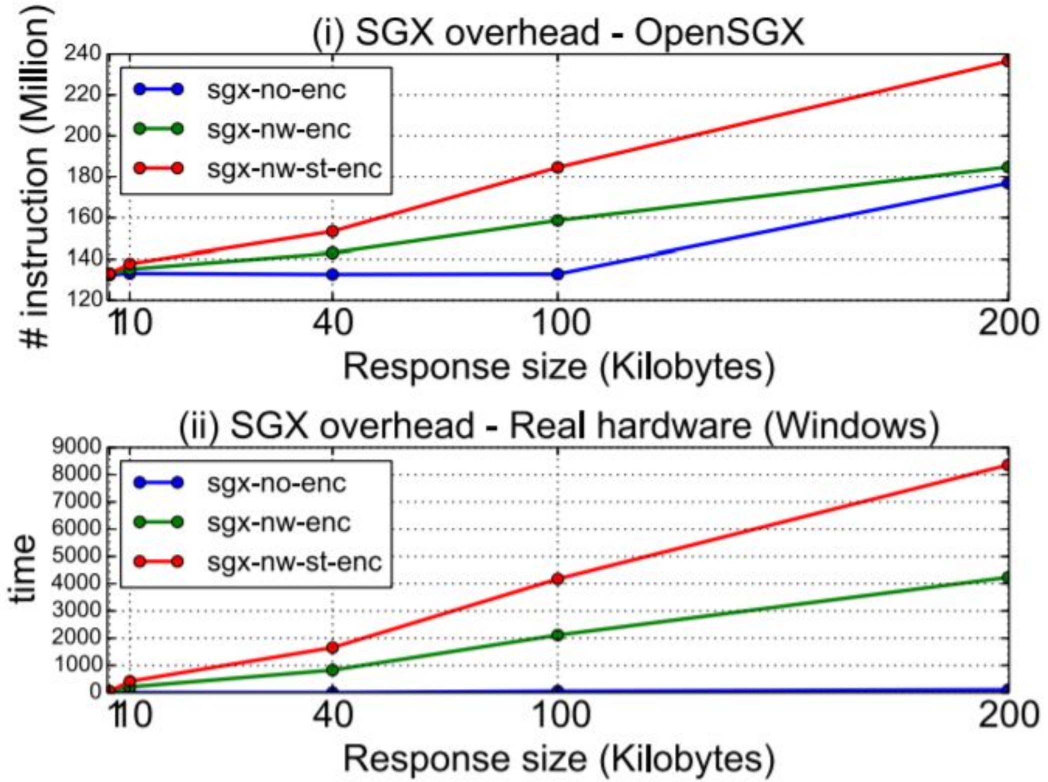


*Figure 6:* Showing SGX overhead variation with size of data transfer between host and enclave with no encryption, only network interaction encryption and with both network and storage encrypted (i) using OpenSGX and (ii) using Windows SGX SDK.

## Airbox Deployment

- Mobile Networks
- Enterprise