Regulating Smart Devices in Restricted Spaces

Vinod Ganapathy

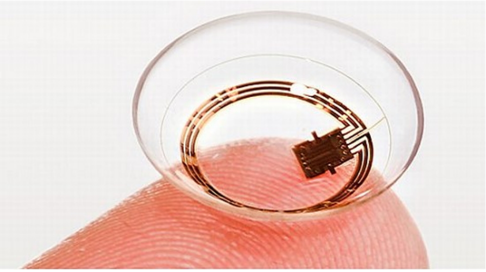
vinodg@cs.rutgers.edu

Associate Professor of Computer Science

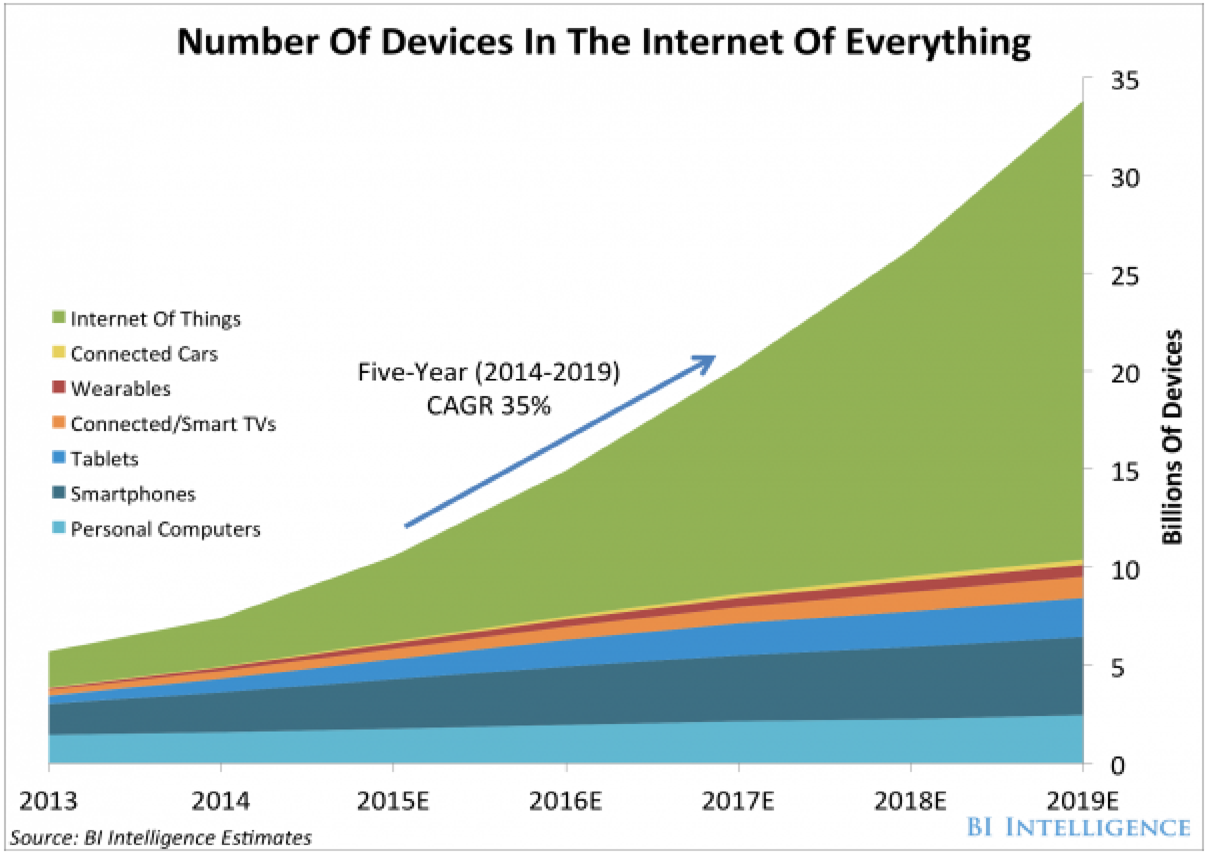
Rutgers, The State University of New Jersey

Appears in Proc. ACM MobiSys'16

Devices are everywhere!



Number of devices is increasing



- Predicted 1.2 billion new smart phones by 2018
- Predicted 50% device use increase year over year in enterprise sector until 2018 **[Gartner 2014]**

Devices are increasingly capable

Model	CPU (GHz)	Screen (1000x)	Rear camera	Front camera	Battery (mAh)	Sensors other than Camera/Microphone
iPhone	0.4	153	2MP	-	1,400	3 (light, accelerometer, proximity)
iPhone3	0.6	153	3MP	-	1,150	4 (+= compass)
iPhone4	0.8	614	5MP	0.3MP	1,420	6 (+= gyroscope, infrared)
iPhone5	1.3 (2 core)	727	8MP	1.2MP	1,560	7 (+=fingerprint)
iPhone6	2.0 (2 core)	1000	12MP	5.0MP	1,715	8 (+= barometer)

With great power …

… comes great responsibility

How can devices be misused?

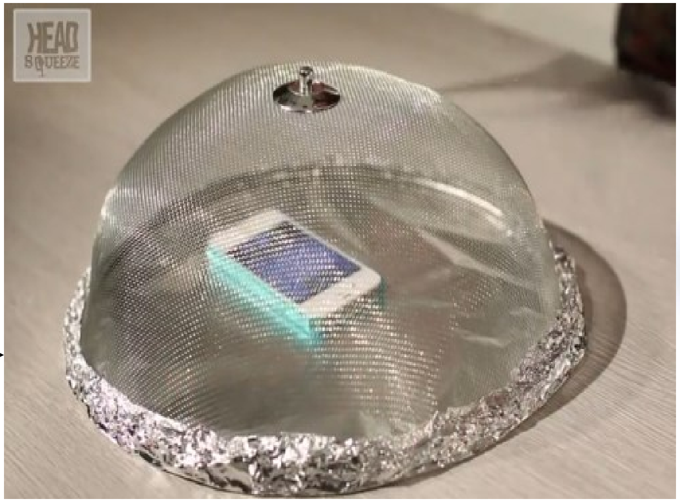
1. **Malicious end-users** can leverage sensors to exfiltrate or infiltrate unauthorized data

2. **Malicious apps** on devices can achieve similar goals even if end-user is benign

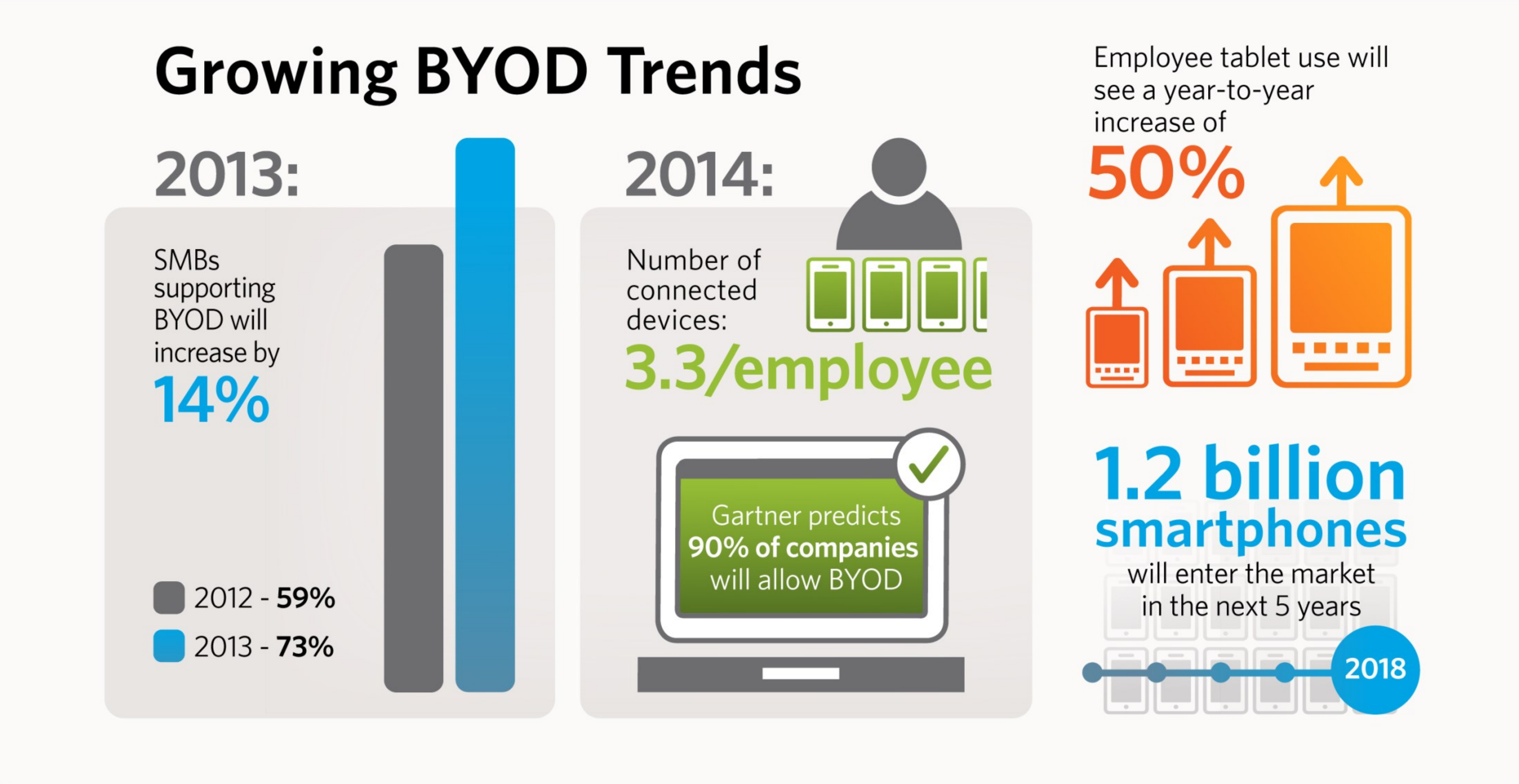
Government or corporate office

- **Problem**: Sensitive documents and meetings can be ex-filtrated using the camera, microphone and storage media
- **Current solution**: Physical security scans, device isolation

Faraday cages



Challenge: Bring your own device



Growing BYOD Trends

2013:

SMBs supporting BYOD will increase by **14%**

■ 2012 - **59%**

■ 2013 - **73%**

2014:

Number of connected devices: **3.3/employee**

Gartner predicts **90% of companies** will allow BYOD

Employee tablet use will see a year-to-year increase of **50%**

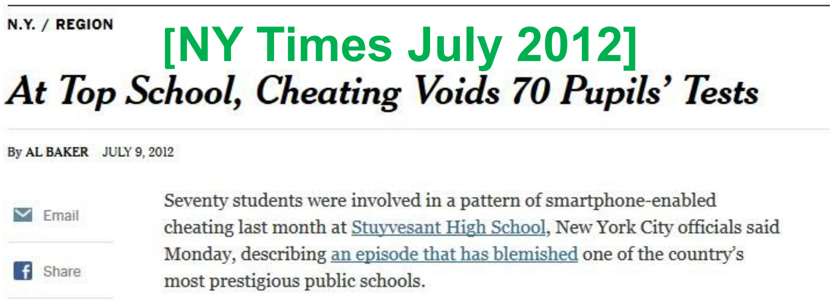
1.2 billion smartphones will enter the market in the next 5 years

Classroom and exam setting

Classroom and exam setting

- **Problem**: Personal devices can be used to infiltrate unauthorized information

[NY Times July 2012]



N.Y. / REGION

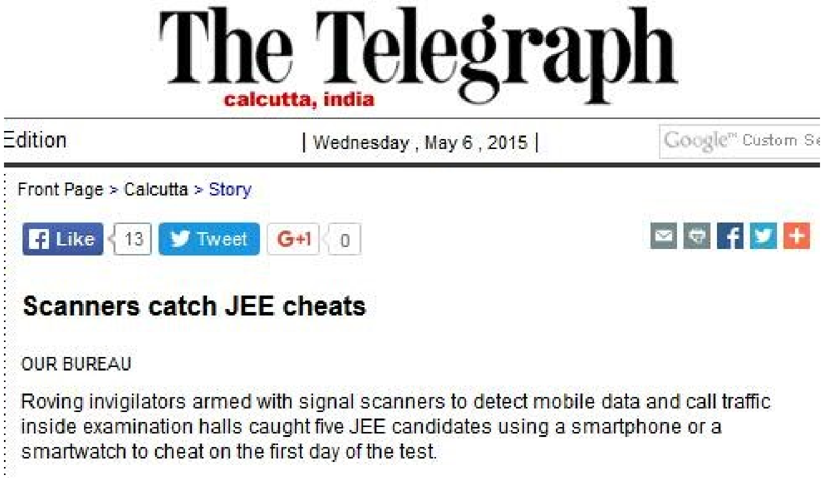
At Top School, Cheating Voids 70 Pupils' Tests

By AL BAKER JULY 9, 2012

Email

Share

Seventy students were involved in a pattern of smartphone-enabled cheating last month at Stuyvesant High School, New York City officials said Monday, describing an episode that has blemished one of the country's most prestigious public schools.



The Telegraph

calcutta, india

Edition | Wednesday , May 6 , 2015 | Google™ Custom Se

Front Page > Calcutta > Story

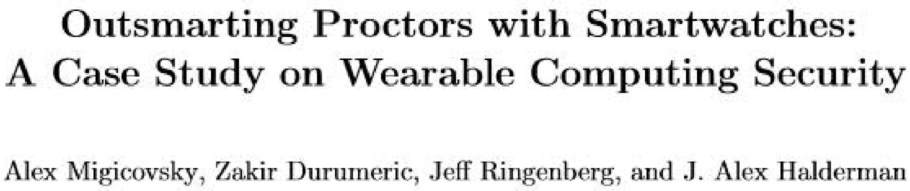
Like 13 Tweet G+1 0

Scanners catch JEE cheats

OUR BUREAU

Roving invigilators armed with signal scanners to detect mobile data and call traffic inside examination halls caught five JEE candidates using a smartphone or a smartwatch to cheat on the first day of the test.

[Financial Crypto 2014]

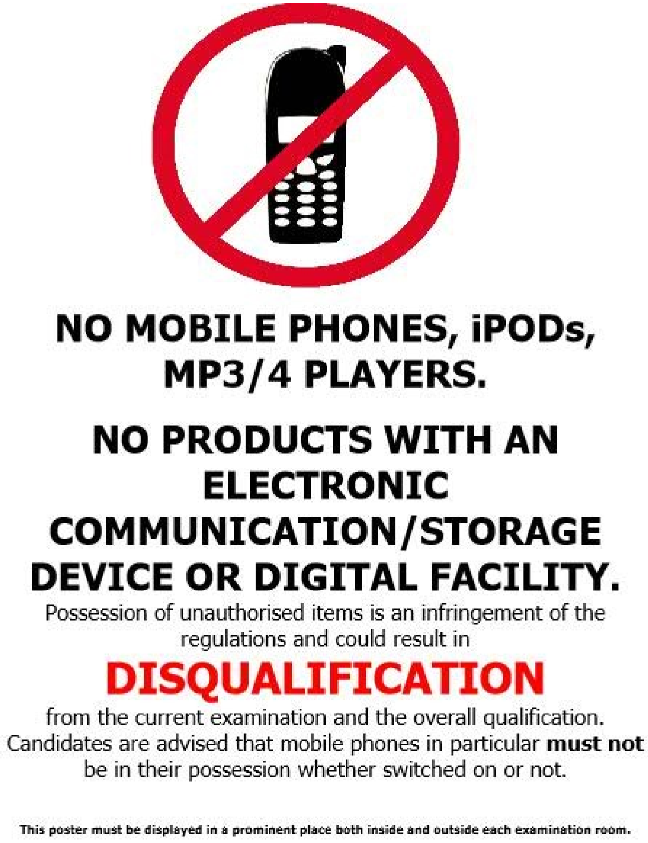


Outsmarting Proctors with Smartwatches:
A Case Study on Wearable Computing Security

Alex Migicovsky, Zakir Durumeric, Jeff Ringenberg, and J. Alex Halderman

Classroom and exam setting

- **Current solution**: Deterrence via rules and threats. Invigilation to ensure compliance



NO MOBILE PHONES, iPODs, MP3/4 PLAYERS.

NO PRODUCTS WITH AN ELECTRONIC COMMUNICATION/STORAGE DEVICE OR DIGITAL FACILITY.

Possession of unauthorised items is an infringement of the regulations and could result in

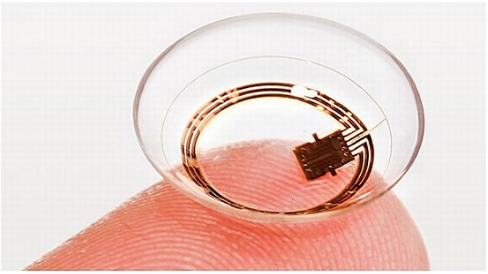
DISQUALIFICATION

from the current examination and the overall qualification. Candidates are advised that mobile phones in particular **must not** be in their possession whether switched on or not.

This poster must be displayed in a prominent place both inside and outside each examination room.

Challenge: Assistive devices

- Students may wish to use devices for legitimate reasons:
  - Smart glass or contacts for vision correction
  - Bluetooth-enabled hearing aids
  - Smart watches to monitor time

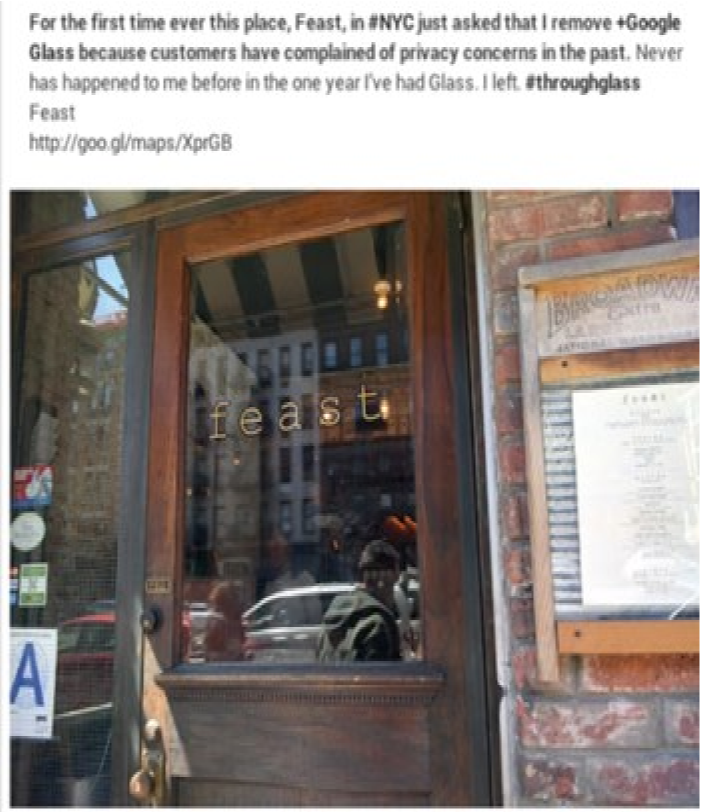


Other social settings

- Restaurants, conferences, gym locker rooms, private homes, …
- **Problems:**
  - Recording private conversations
  - Pictures of individuals taken and posted to social networks without their consent
  - Pictures and videos of otherwise private locations, e.g., private homes

Other social settings

- **Current solutions**: Informal enforcement
- **Challenge**: Social isolation ☺



For the first time ever this place, Feast, in #NYC just asked that I remove +Google Glass because customers have complained of privacy concerns in the past. Never has happened to me before in the one year I've had Glass. I left. **#throughglass**

Feast

http://goo.gl/maps/XprGB

"For the first time ever this place, Feast, in NYC just asked that I remove Google Glass because customers have complained of privacy concerns […] I left"

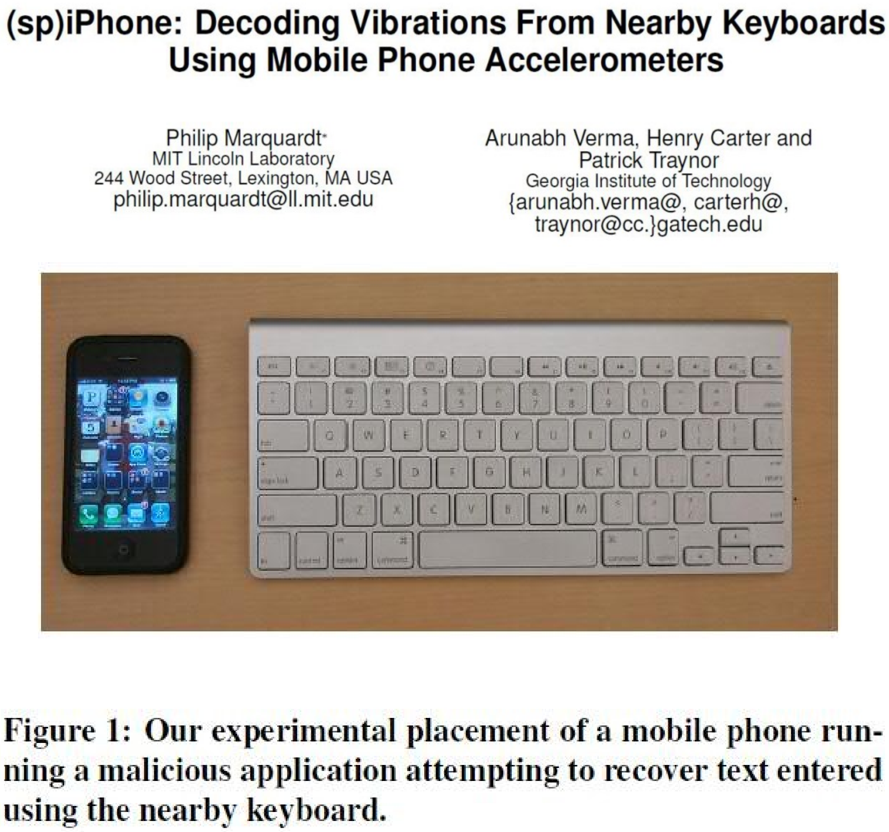
NOTICE

No wearable cameras permitted in locker room.

SmartSign.com • 800-952-1457 • S-5832

Malicious apps exploiting sensors

Sensory malware



(sp)iPhone: Decoding Vibrations From Nearby Keyboards Using Mobile Phone Accelerometers

Philip Marquardt*
MIT Lincoln Laboratory
244 Wood Street, Lexington, MA USA
philip.marquardt@ll.mit.edu

Arunabh Verma, Henry Carter and Patrick Traynor
Georgia Institute of Technology
{arunabh.verma@, carterh@, traynor@cc.}gatech.edu

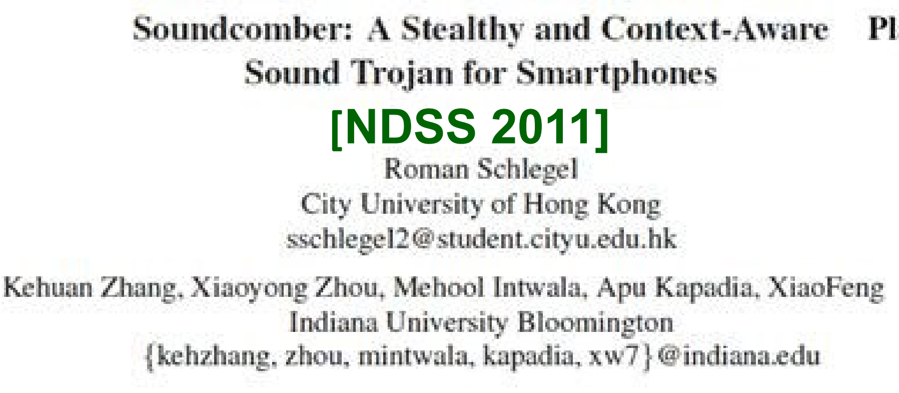
Figure 1: Our experimental placement of a mobile phone running a malicious application attempting to recover text entered using the nearby keyboard.

← Early example of sensory malware **[CCS 2011]**

- Use accelerometer and record keystroke press vibrations
- Up to 80% accuracy in word recovery

Malicious apps exploiting sensors

Sensory malware

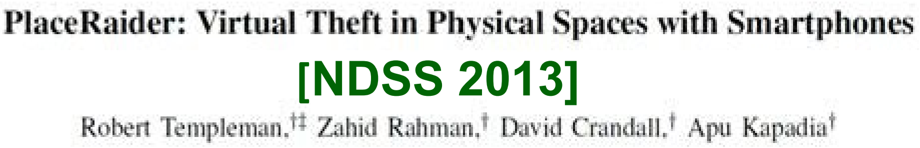


Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones

[NDSS 2011]

Roman Schlegel
City University of Hong Kong
sschlegel2@student.cityu.edu.hk

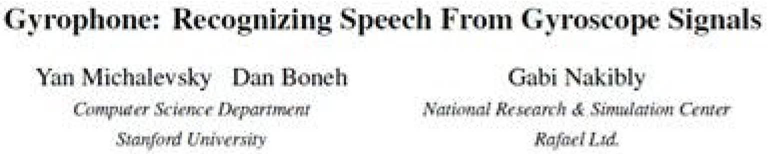
Kehuan Zhang, Xiaoyong Zhou, Mehool Intwala, Apu Kapadia, XiaoFeng
Indiana University Bloomington
{kehzhang, zhou, mintwala, kapadia, xw7}@indiana.edu



PlaceRaider: Virtual Theft in Physical Spaces with Smartphones

[NDSS 2013]

Robert Templeman,[†‡] Zahid Rahman,[†] David Crandall,[†] Apu Kapadia[†]



Gyrophone: Recognizing Speech From Gyroscope Signals

Yan Michalevsky Dan Boneh
Computer Science Department
Stanford University

Gabi Nakibly
National Research & Simulation Center
Rafael Ltd.

[USENIX Security 2014]

- Attacks have now been demonstrated using every imaginable sensor
- Attack accuracy will ***improve*** with each generation of devices and sensors

Claim

Smart devices will become integrated with daily lives → *Ad hoc* solutions, *e.g.*, banning device use, will no longer be acceptable

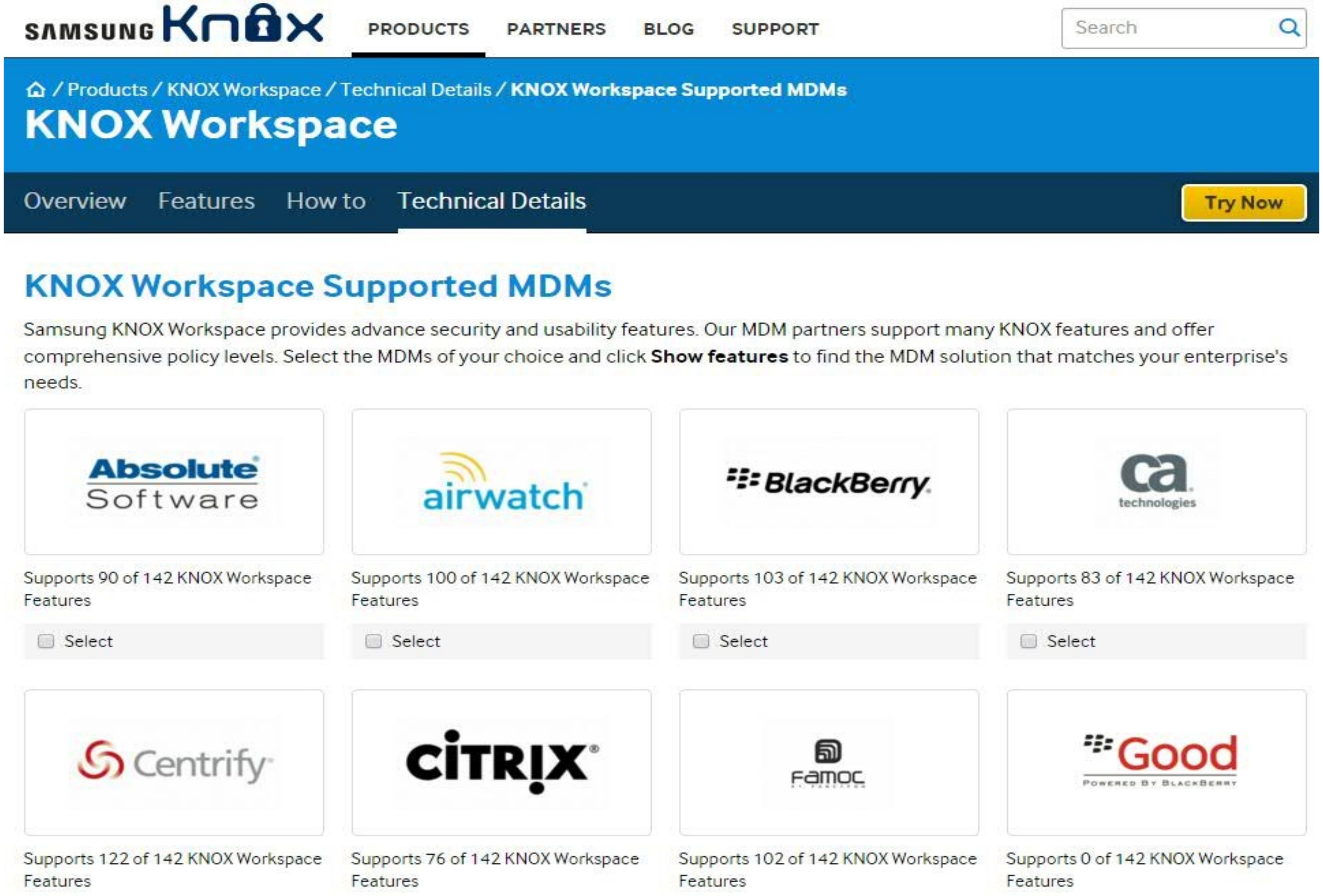
Vision

Need systematic methods to regulate devices and ensure responsible use

Discussion: Only considering **overt** device use. Covert use detection still requires traditional physical security measures.

What solutions exist today?

Mobile device management (MDM) solutions



KNOX Workspace

KNOX Workspace Supported MDMs

Samsung KNOX Workspace provides advance security and usability features. Our MDM partners support many KNOX features and offer comprehensive policy levels. Select the MDMs of your choice and click **Show features** to find the MDM solution that matches your enterprise's needs.

Absolute Software

Supports 90 of 142 KNOX Workspace Features

□ Select

airwatch

Supports 100 of 142 KNOX Workspace Features

□ Select

BlackBerry

Supports 103 of 142 KNOX Workspace Features

□ Select

ca technologies

Supports 83 of 142 KNOX Workspace Features

□ Select

Centrify

Supports 122 of 142 KNOX Workspace Features

CITRIX

Supports 76 of 142 KNOX Workspace Features

Famoc

Supports 102 of 142 KNOX Workspace Features

Good Powered by BlackBerry

Supports 0 of 142 KNOX Workspace Features

Mobile device management

- Solution for enterprises that offer *Bring your own device* (BYOD) models
- Employees are given a mobile device outfitted with a secure software stack
- Enterprise policies “pushed” to device when employee changes device persona

Mobile device management

Main shortcoming of current MDM solutions

- ➢ Enterprise must trust software stack on guest device to enforce policies correctly
- ➢ But guest devices under control of possibly malicious end-users

- Solution for enterprises that offer *Bring your own device* (BYOD) models
- Employees are given a mobile device outfitted with a secure software stack
- Enterprise policies “pushed” to device when employee changes device persona

Contributions of our work

- **Restricted space**: Location owned by a **host**, where **guest devices** must follow the host's usage policies
- Enable guest devices to **prove** policy compliance to restricted space hosts
- Use a simple, low-level API that **reduces size of trusted computing base** on guest devices

Key technical challenges

1. Guest devices are under the control of possibly malicious end-user

➢ **Solution**: Use trusted hardware on guest device

2. What constitutes proof of compliance?

➢ **Solution**: Send guest device configuration, showing policy compliance, to host

3. Doesn't that compromise guest device privacy?

➢ **Solution**: Allow guest to vet all communication to and from the host

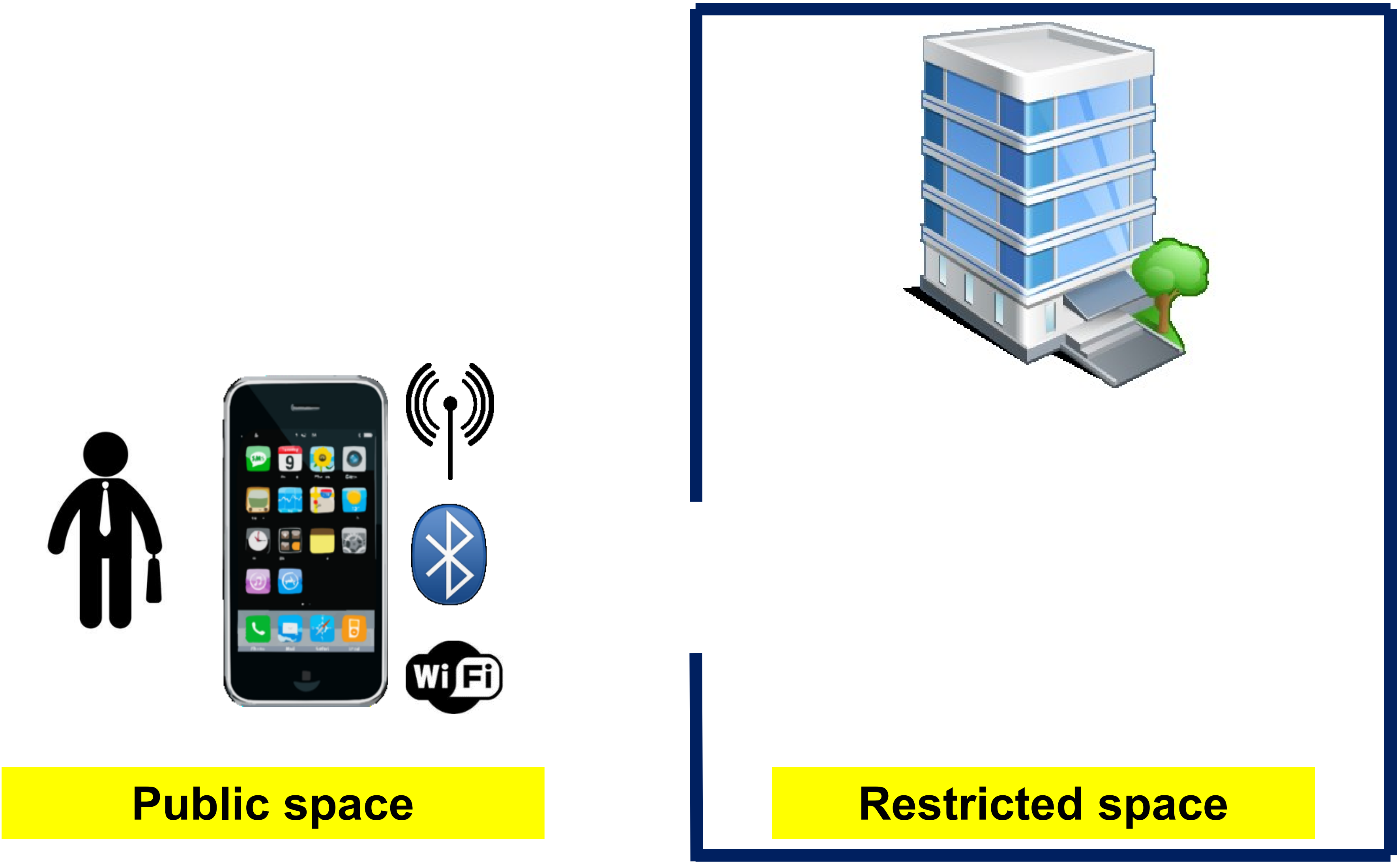
Threat model

- **Trusted hardware on guest devices**:
  - Guest devices equipped with ARM TrustZone

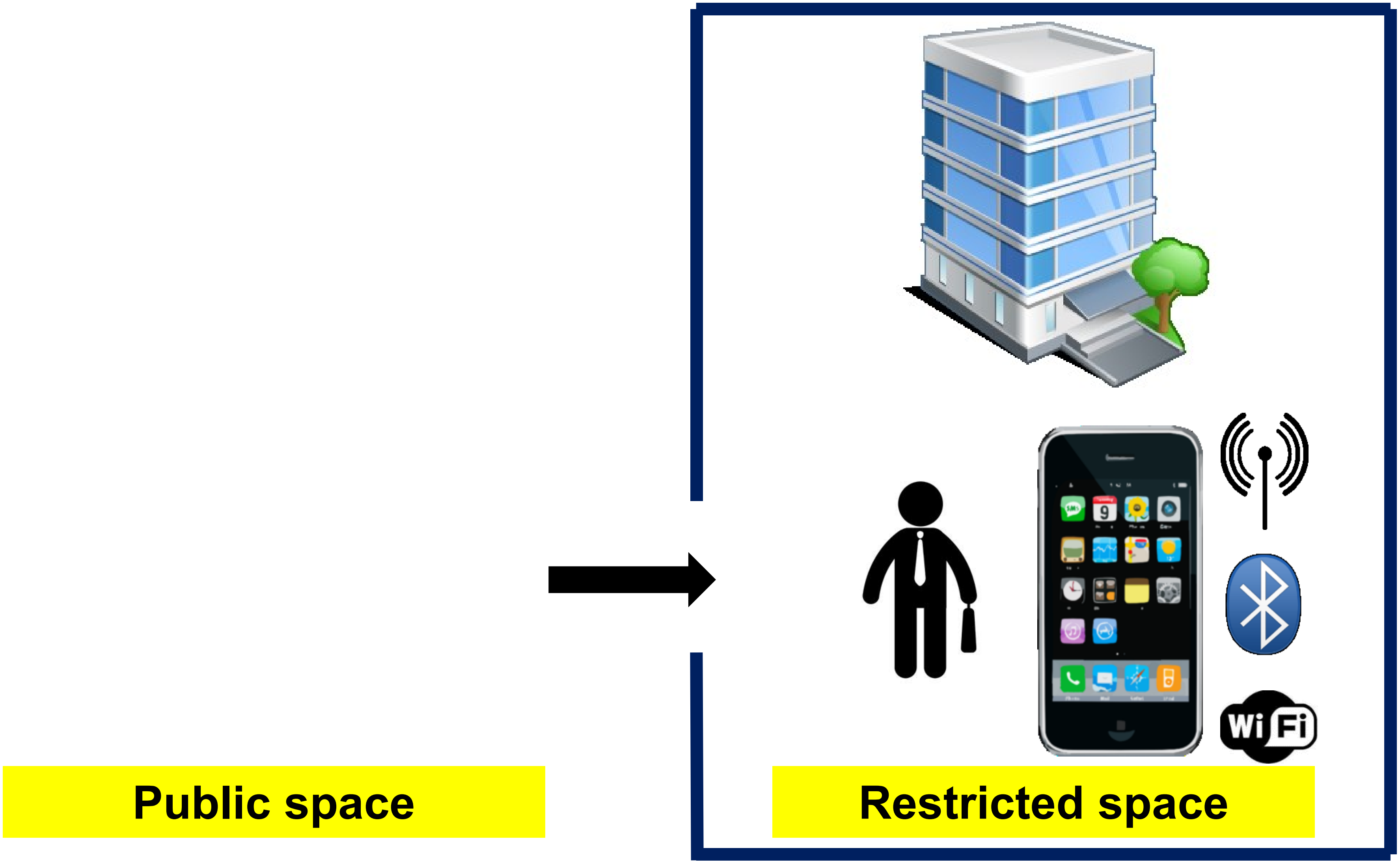
- **Hosts and guests are mutually-distrusting**:
  - Hosts do not trust end-user of guest device or its end-user software stack
  - Guests do not trust host's *reconfiguration requests* to ensure policy compliance

- **Guest devices are used overtly**:
  - Host must still use traditional physical methods to detect covert device use

Guest device check-in



Guest device check-in

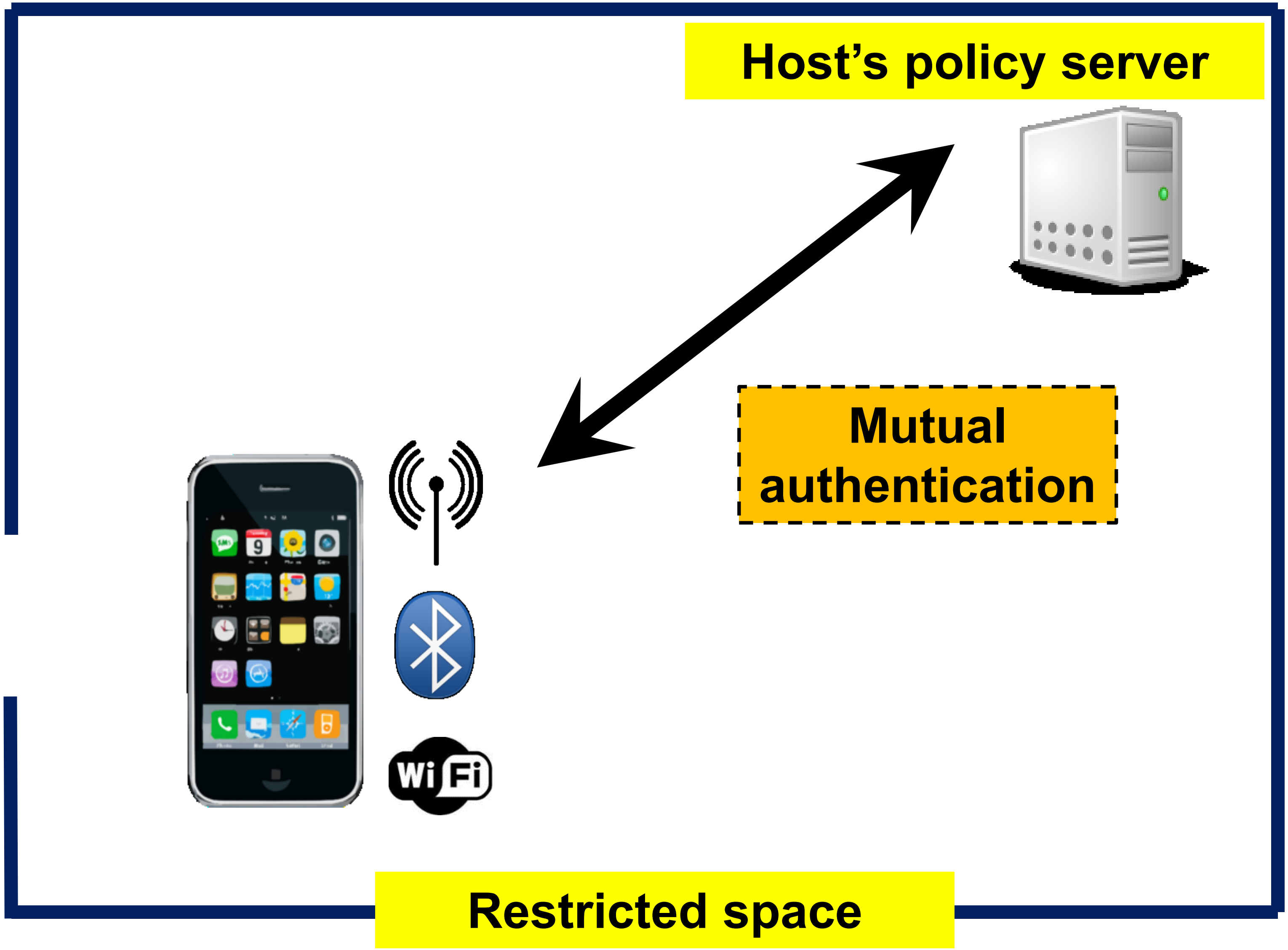


Public space

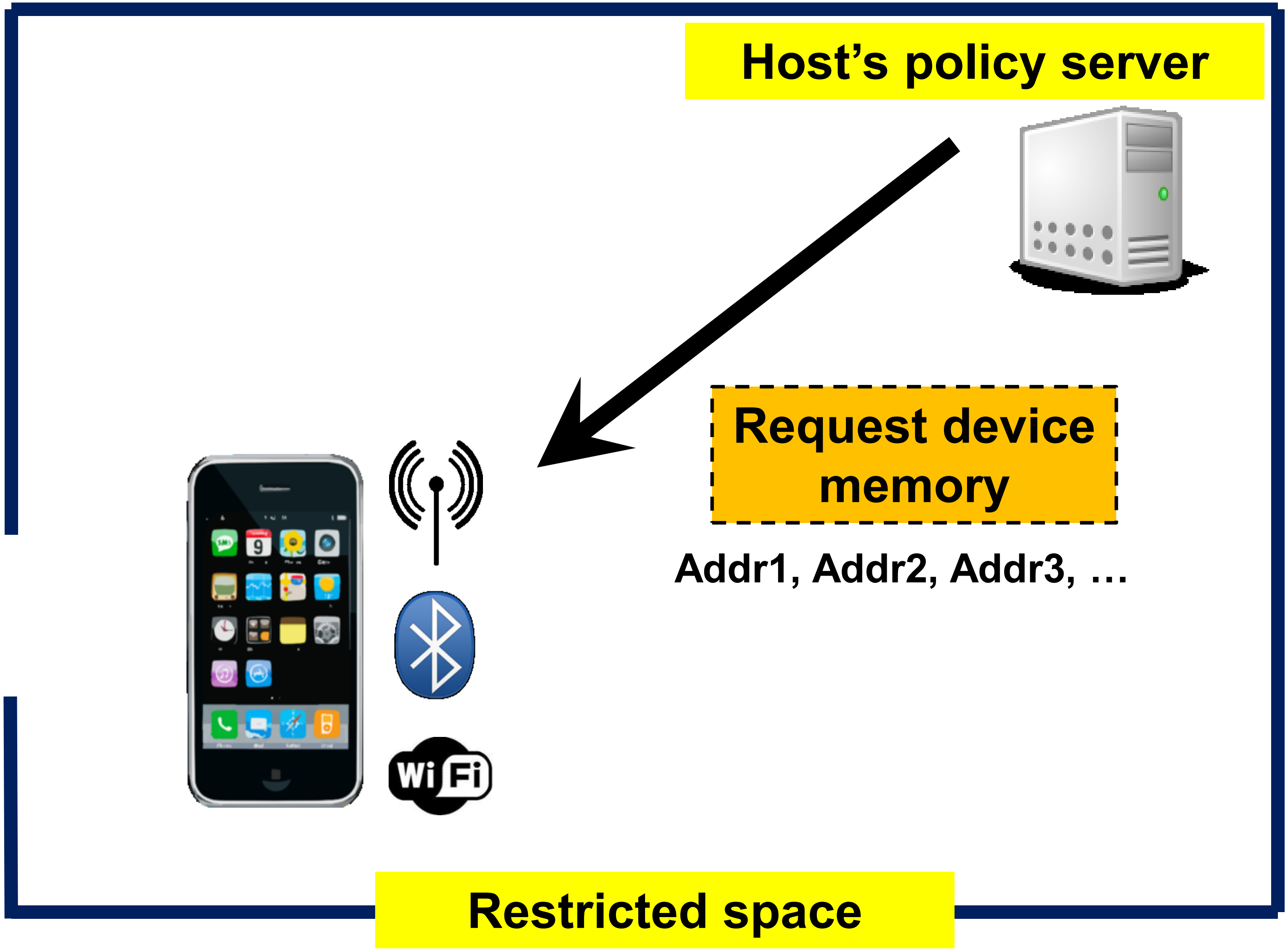
Restricted space

1

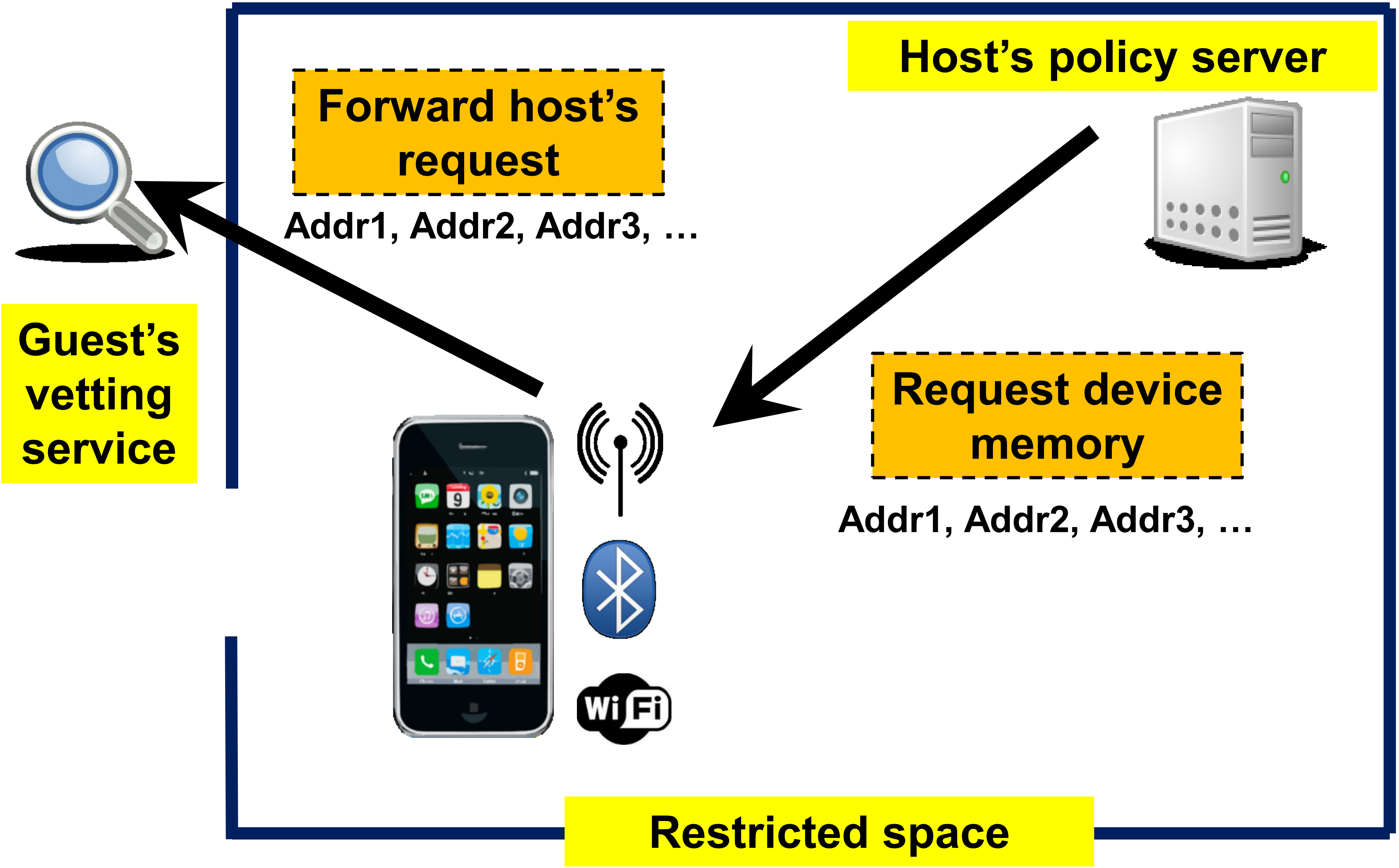
Mutual authentication



2 Host requests guest analysis



2 Guest vets host’s request



Host’s policy server

Forward host’s request

Addr1, Addr2, Addr3, …

Guest’s vetting service

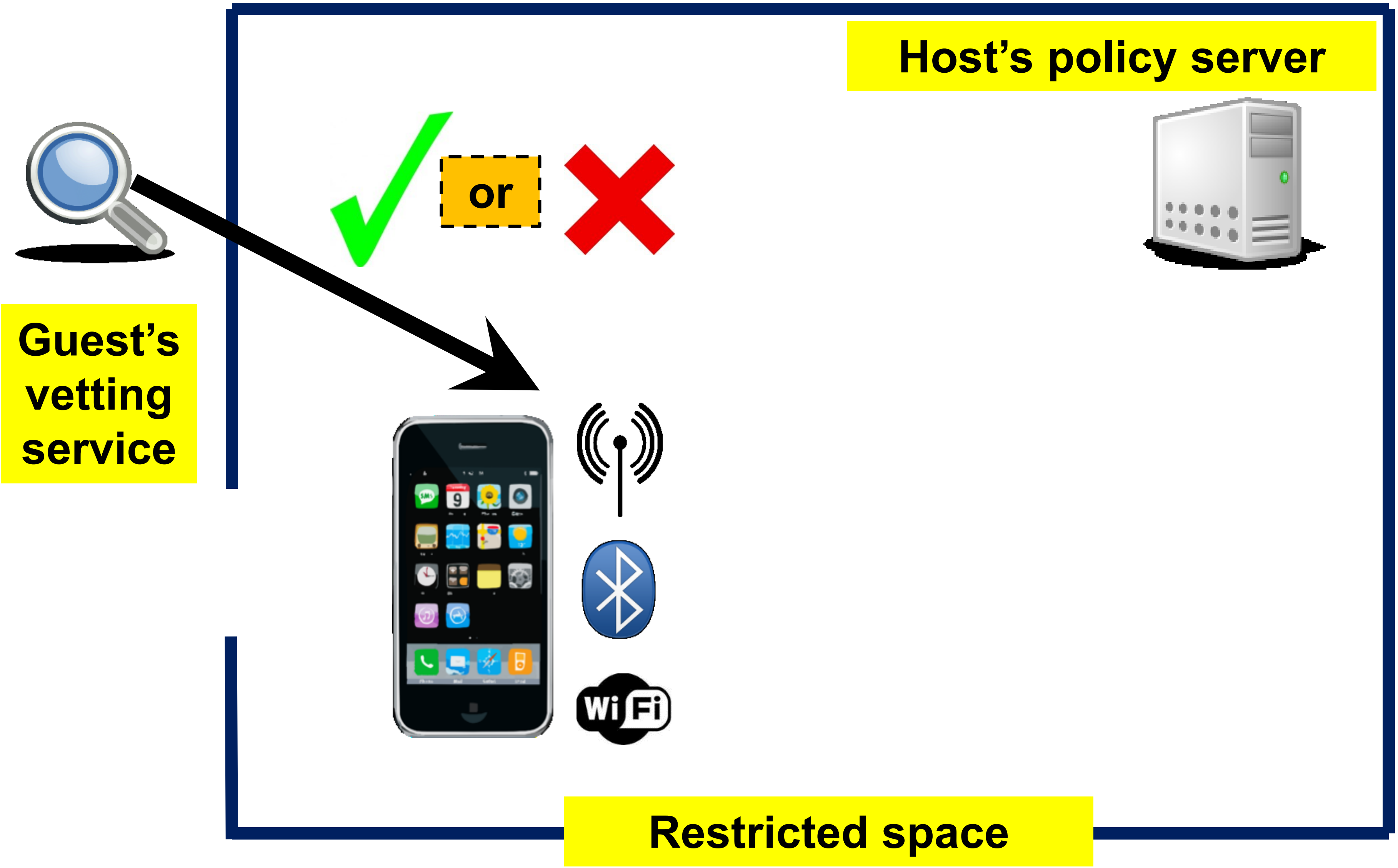
Request device memory

Addr1, Addr2, Addr3, …

Restricted space

2

Guest vets host’s request



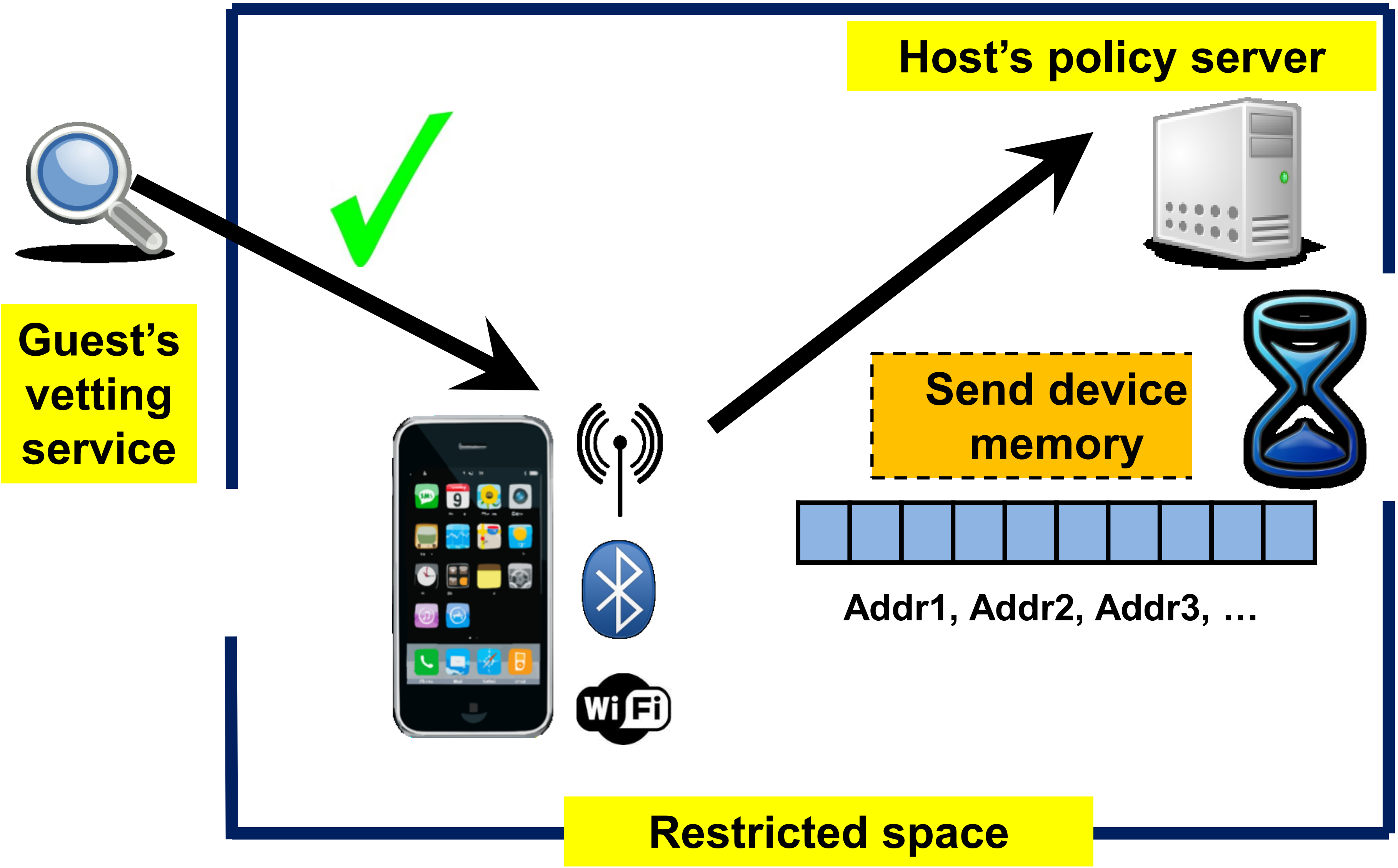
Host’s policy server

or

Guest’s vetting service

Restricted space

2 Host analyzes guest device



Host's policy server

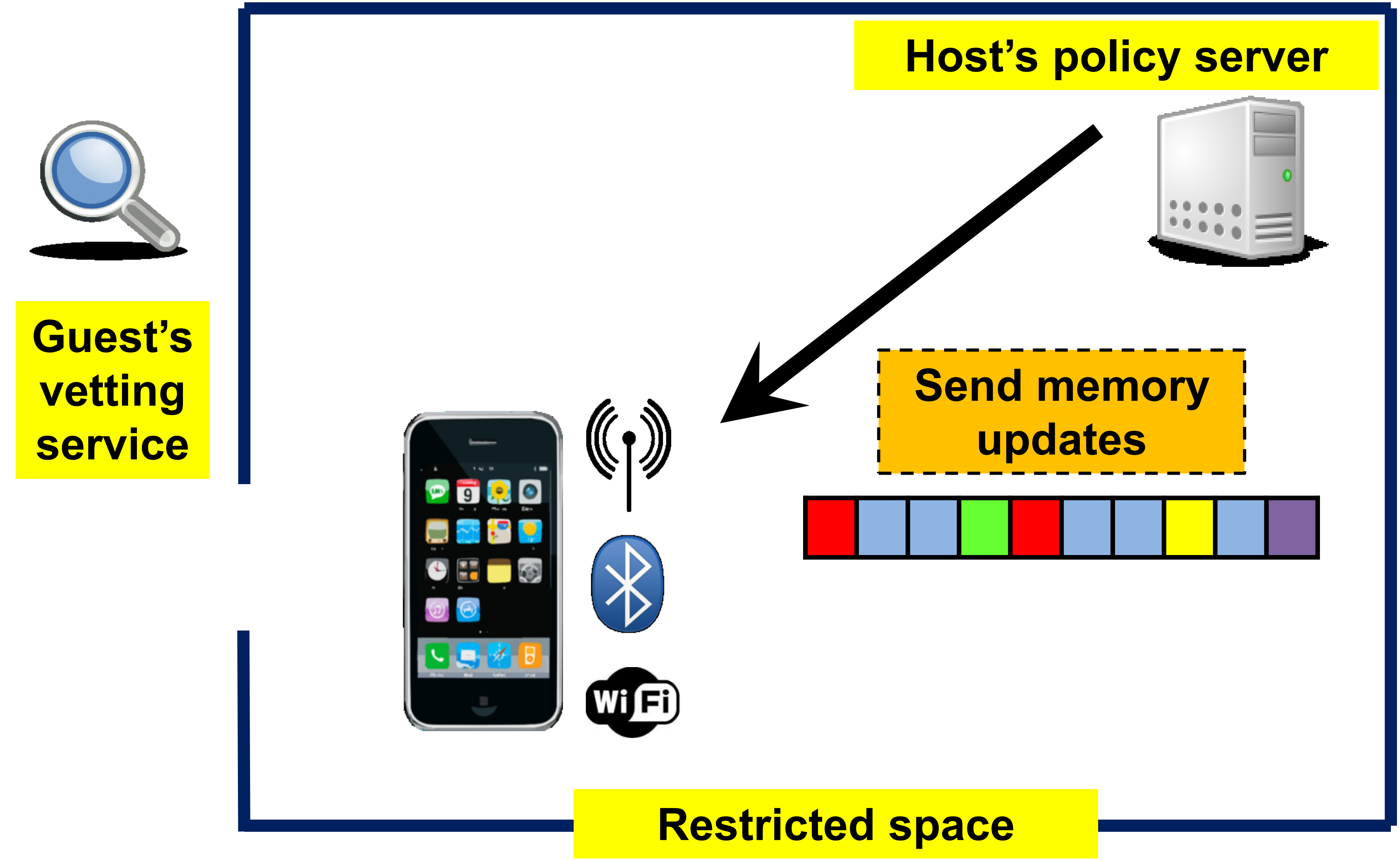
Guest's vetting service

Send device memory

Addr1, Addr2, Addr3, …

Restricted space

3 Host pushes policy to guest



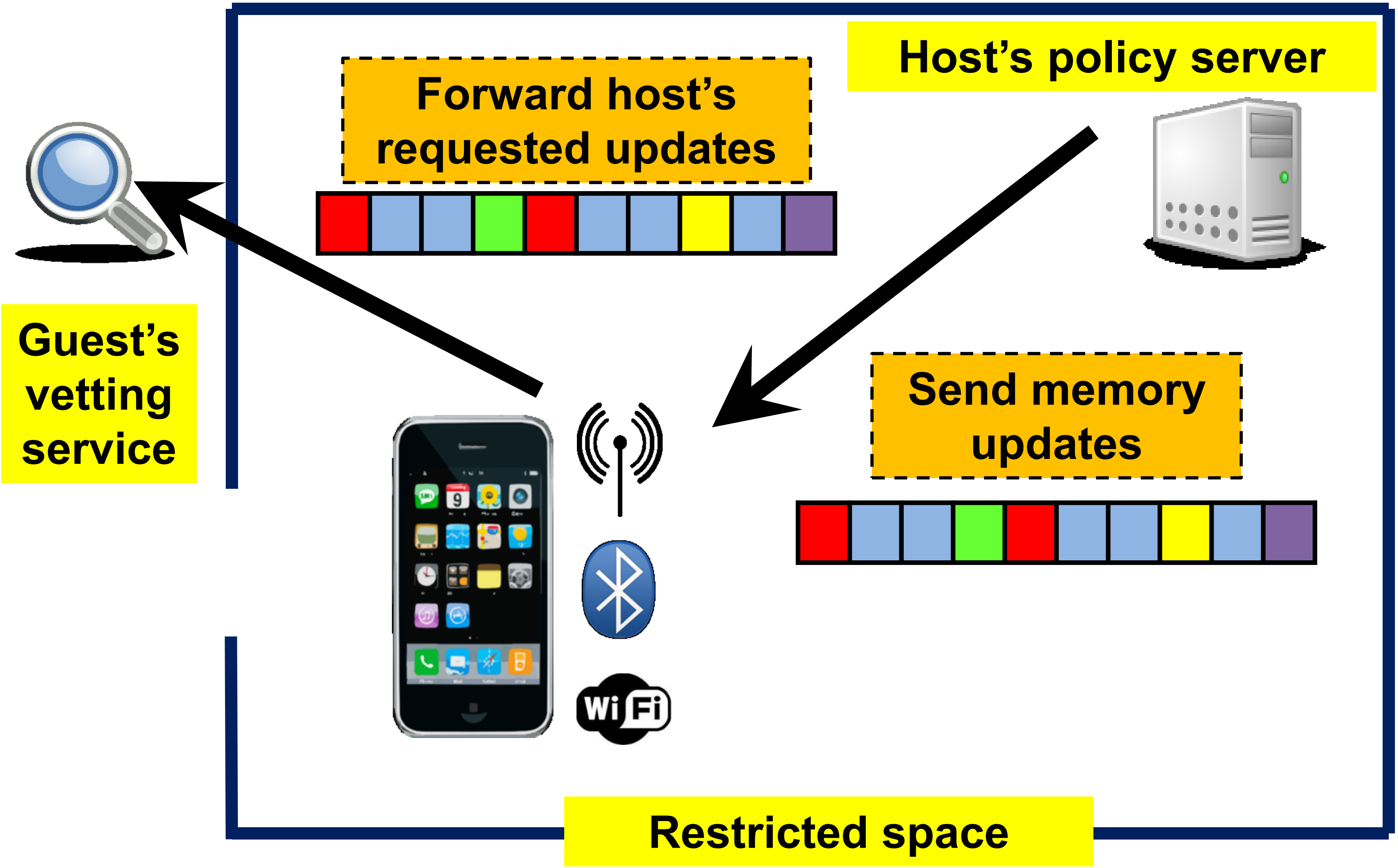
Guest's vetting service

Host's policy server

Send memory updates

Restricted space

3 Guest vets host's updates



Host's policy server

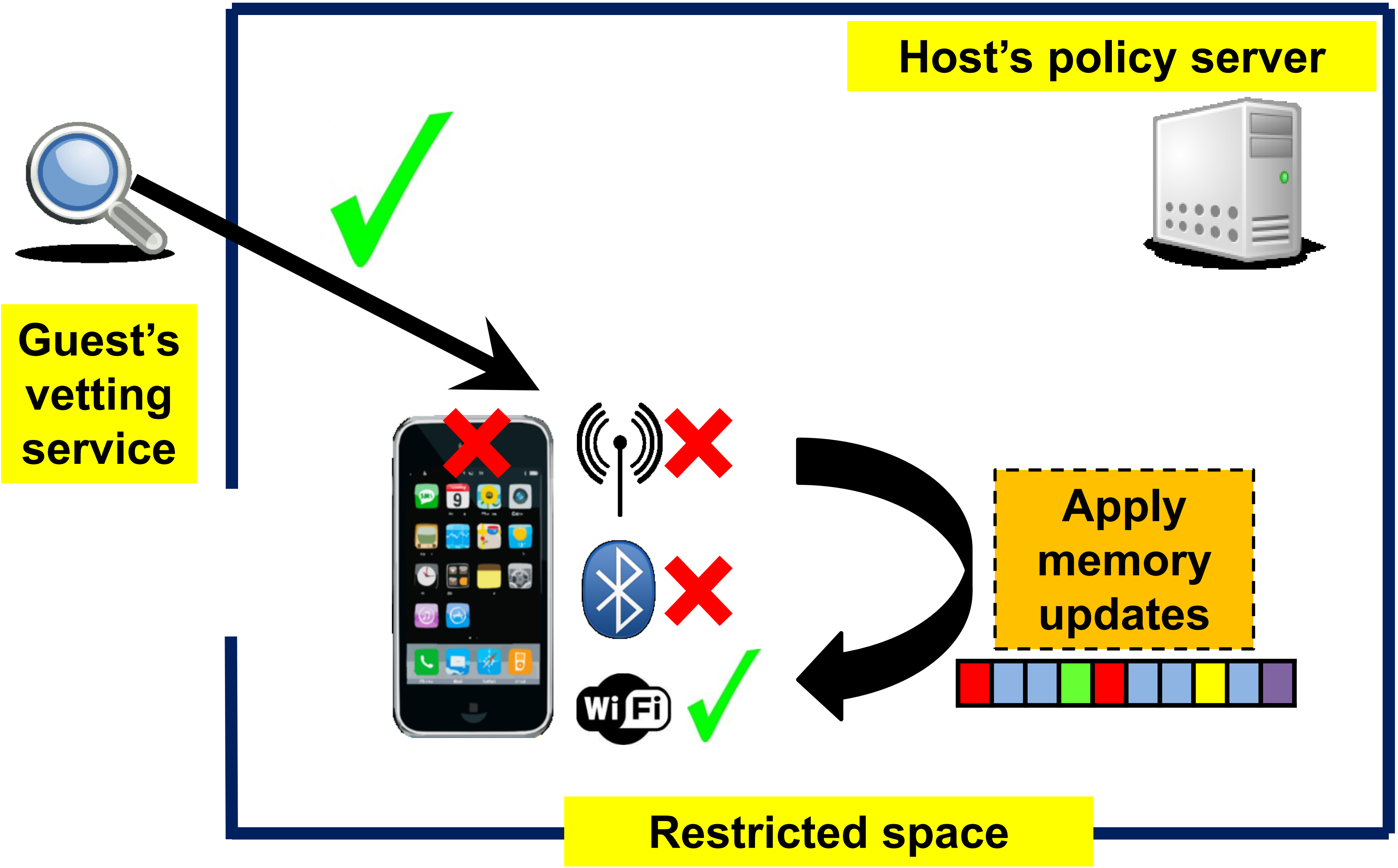
Forward host's requested updates

Send memory updates

Guest's vetting service

Restricted space

3 Guest applies host’s updates



Host’s policy server

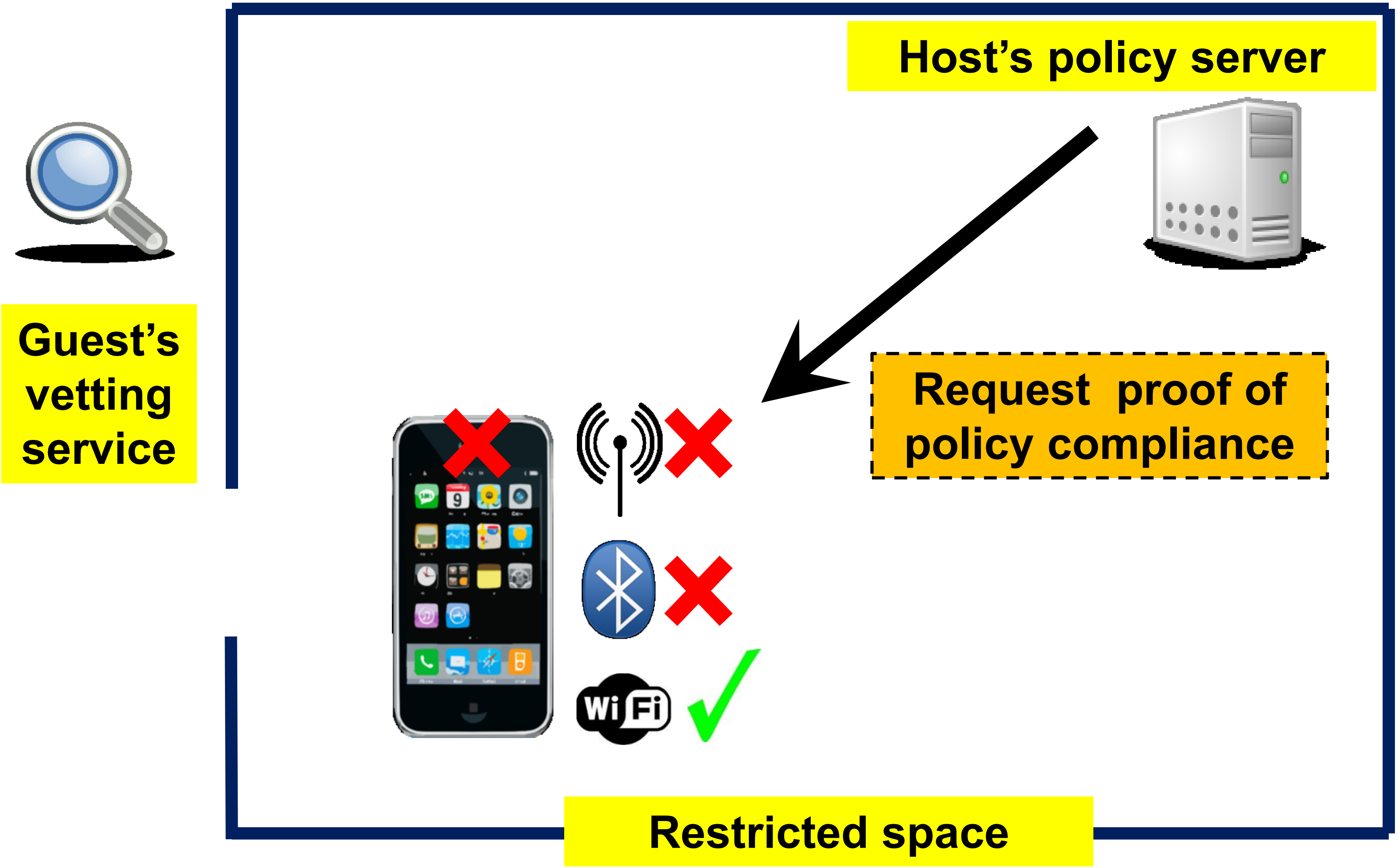
Guest’s vetting service

Apply memory updates

Restricted space

4

Host requests proof



Host’s policy server

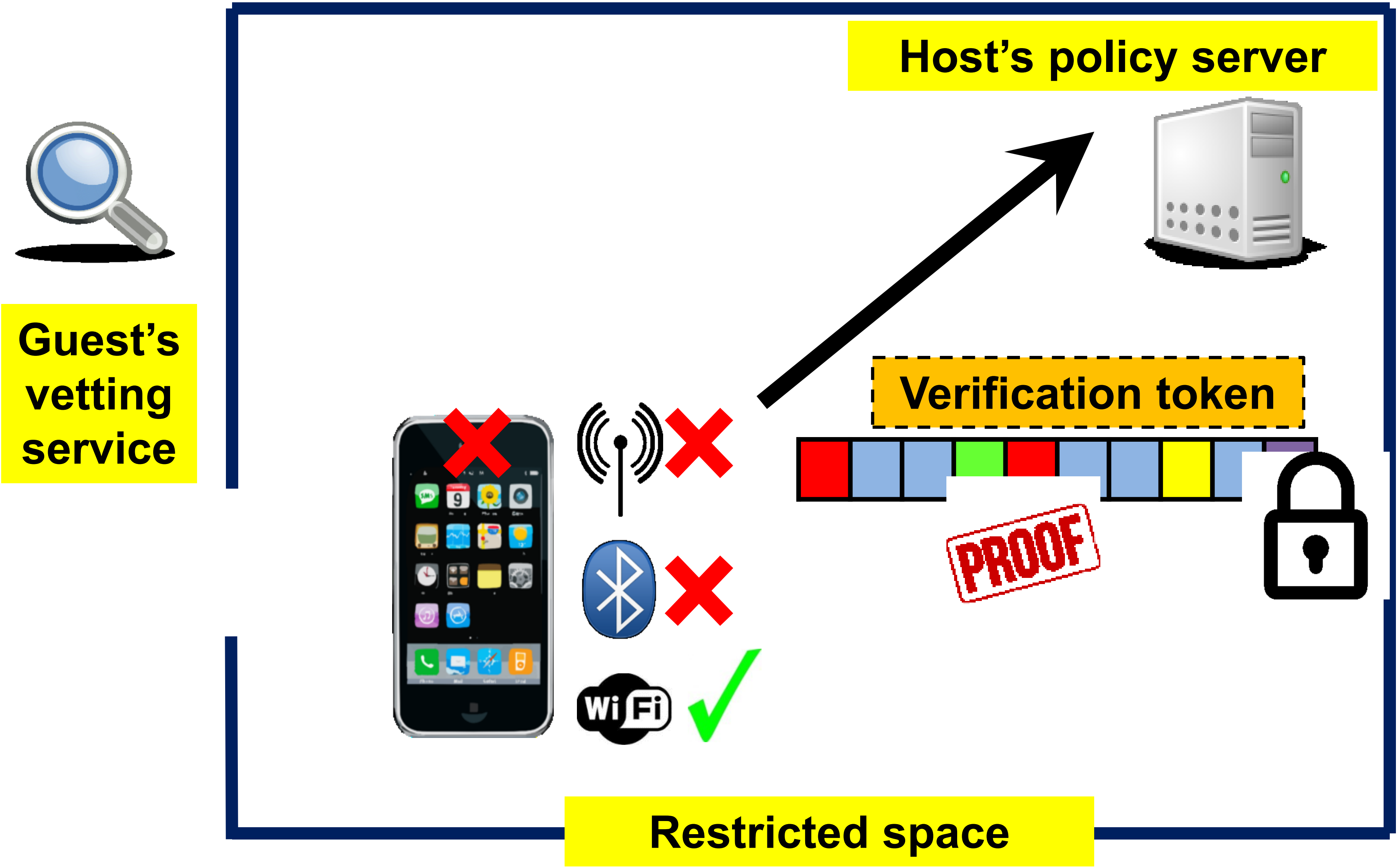
Request proof of policy compliance

Guest’s vetting service

Restricted space

4

Guest sends proof



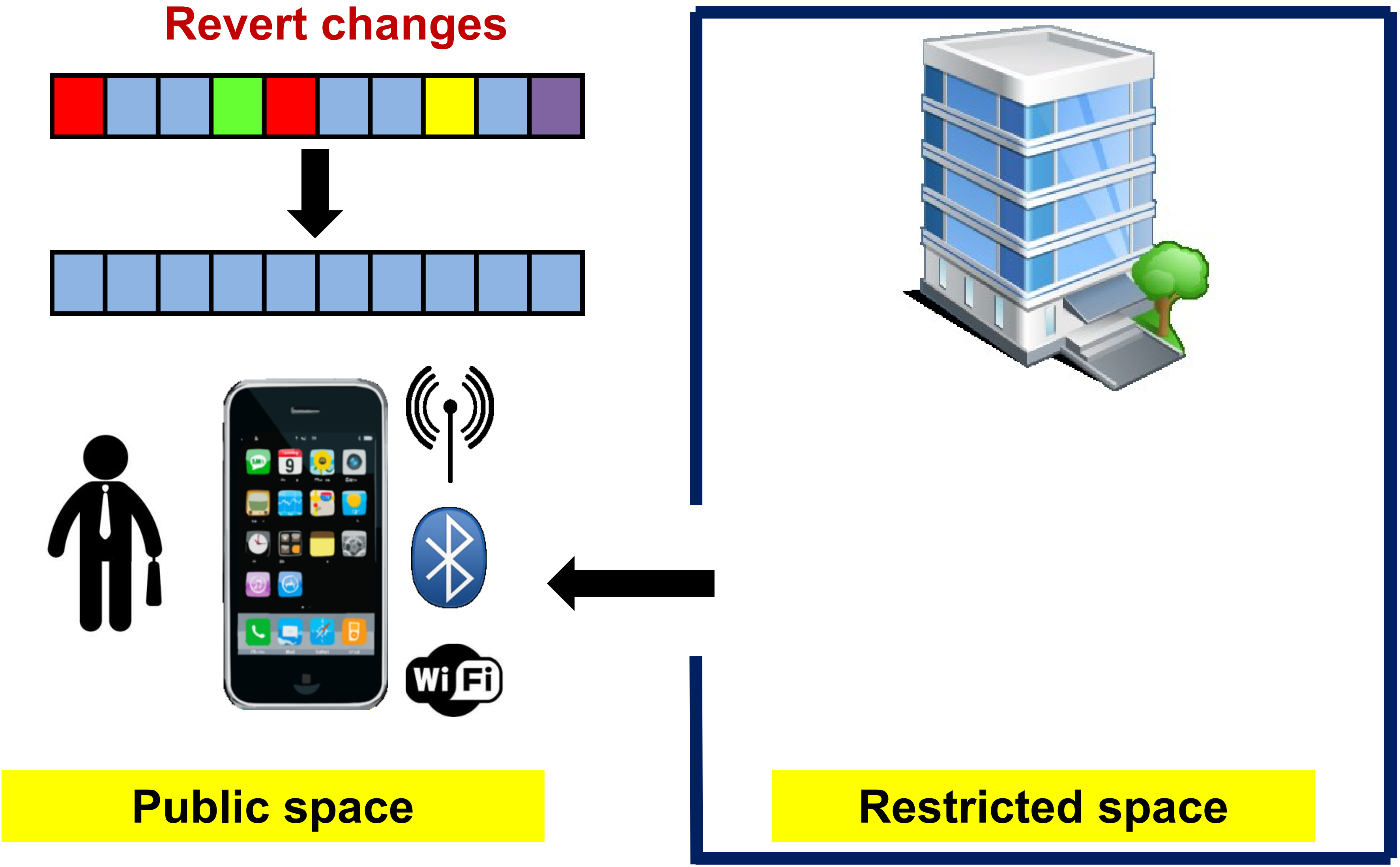
Host's policy server

Guest's vetting service

Verification token

Restricted space

Guest device check-out



Revert changes

Public space

Restricted space

Operational details

1. How can host trust guest to apply policy?

➢ **Answer**: Leverage ARM TrustZone

2. Why memory snapshots and updates?

➢ **Answer**: Powerful low-level API. Reduces TCB

3. How does vetting service ensure safety?

➢ **Answer**: Simple, conservative program analysis

4. Can't guest device simply reboot to undo?

➢ **Answer**: REM-suspend protocol

The ARM TrustZone

Guest device

Normal world (Untrusted)

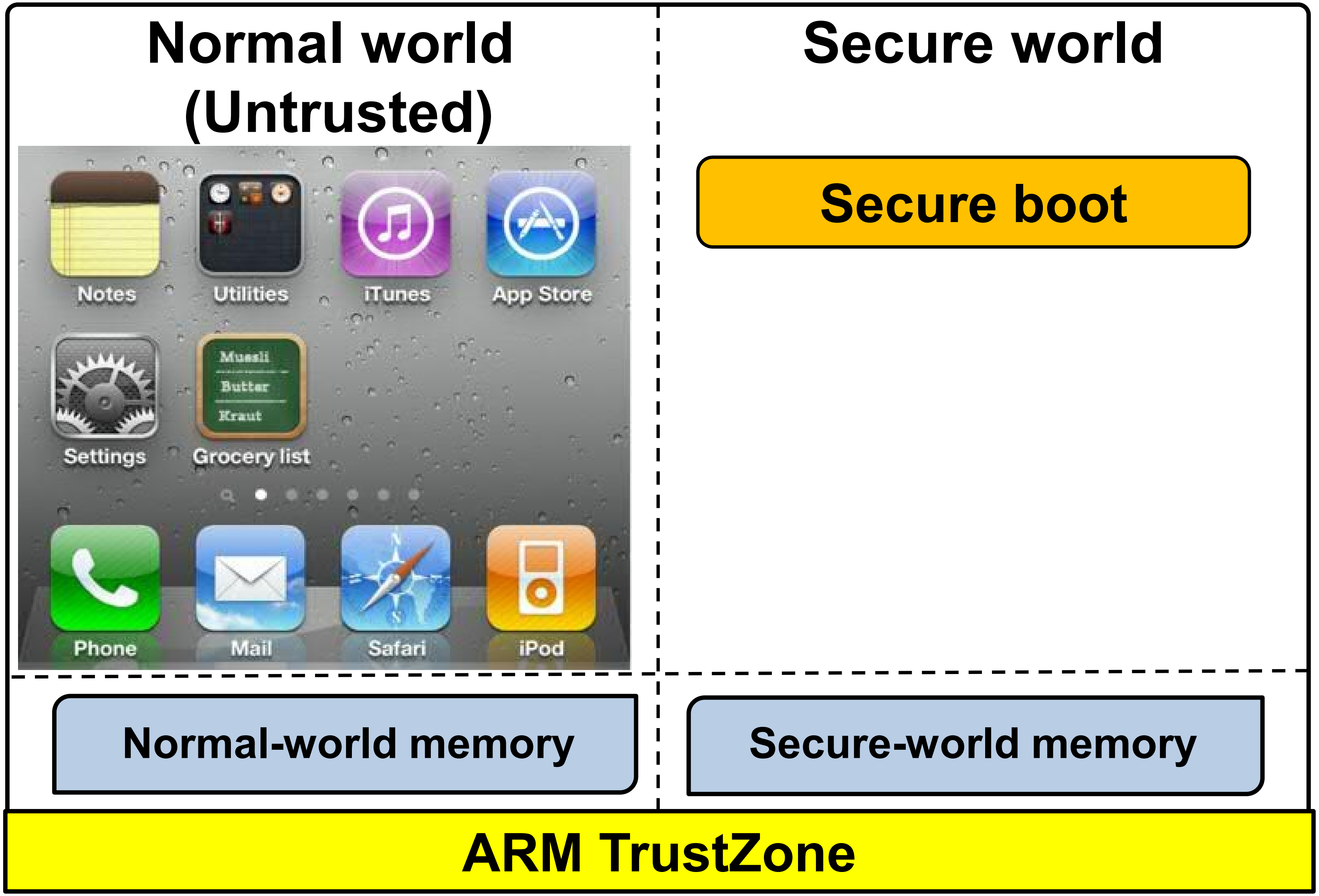
Secure world (Protected by H/W)

Normal-world memory

Secure-world memory

ARM TrustZone

Secure boot protects secure world



Normal world (Untrusted)

Secure world

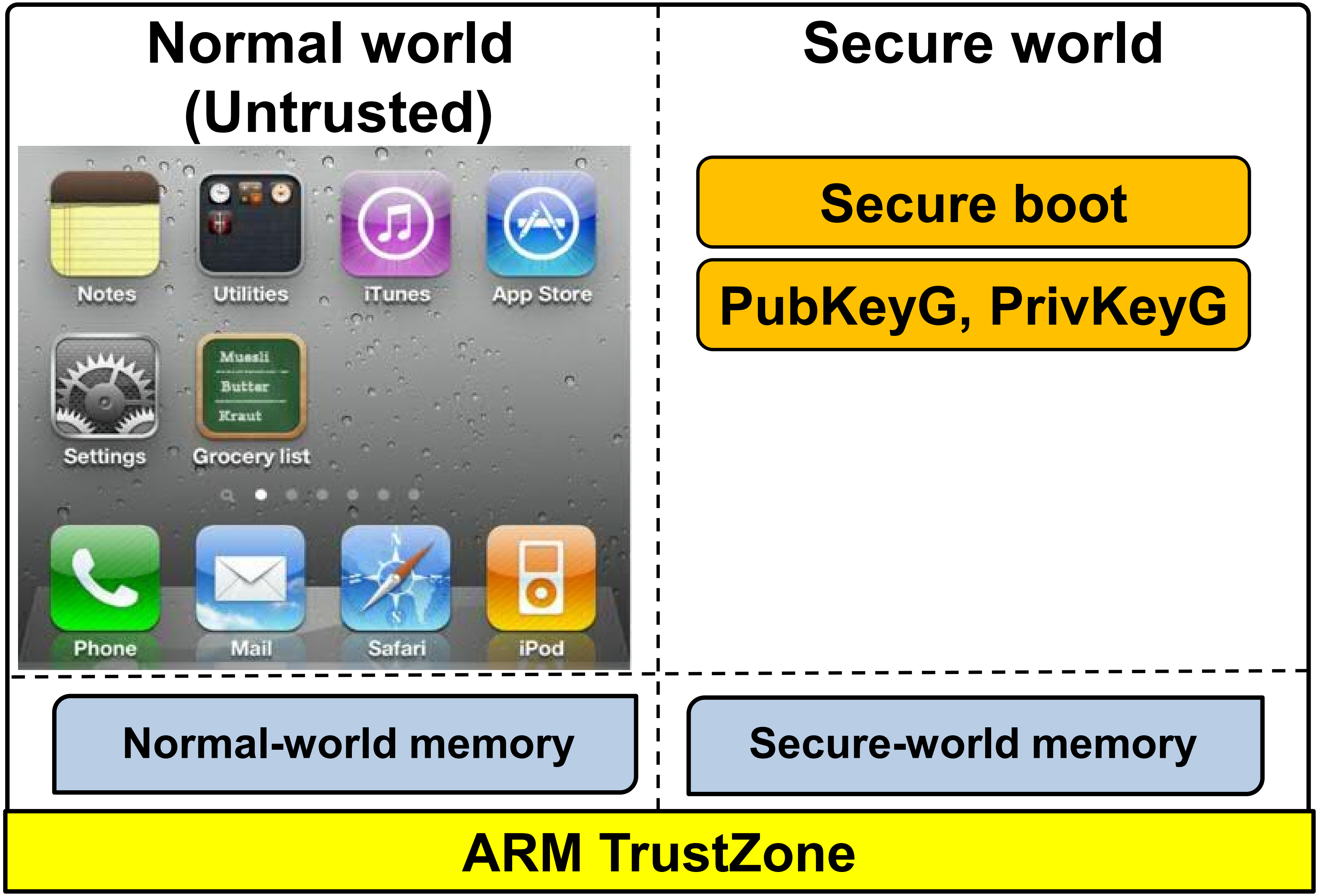
Secure boot

Normal-world memory

Secure-world memory

ARM TrustZone

Secure world stores keys



Normal world (Untrusted)

Secure world

Secure boot

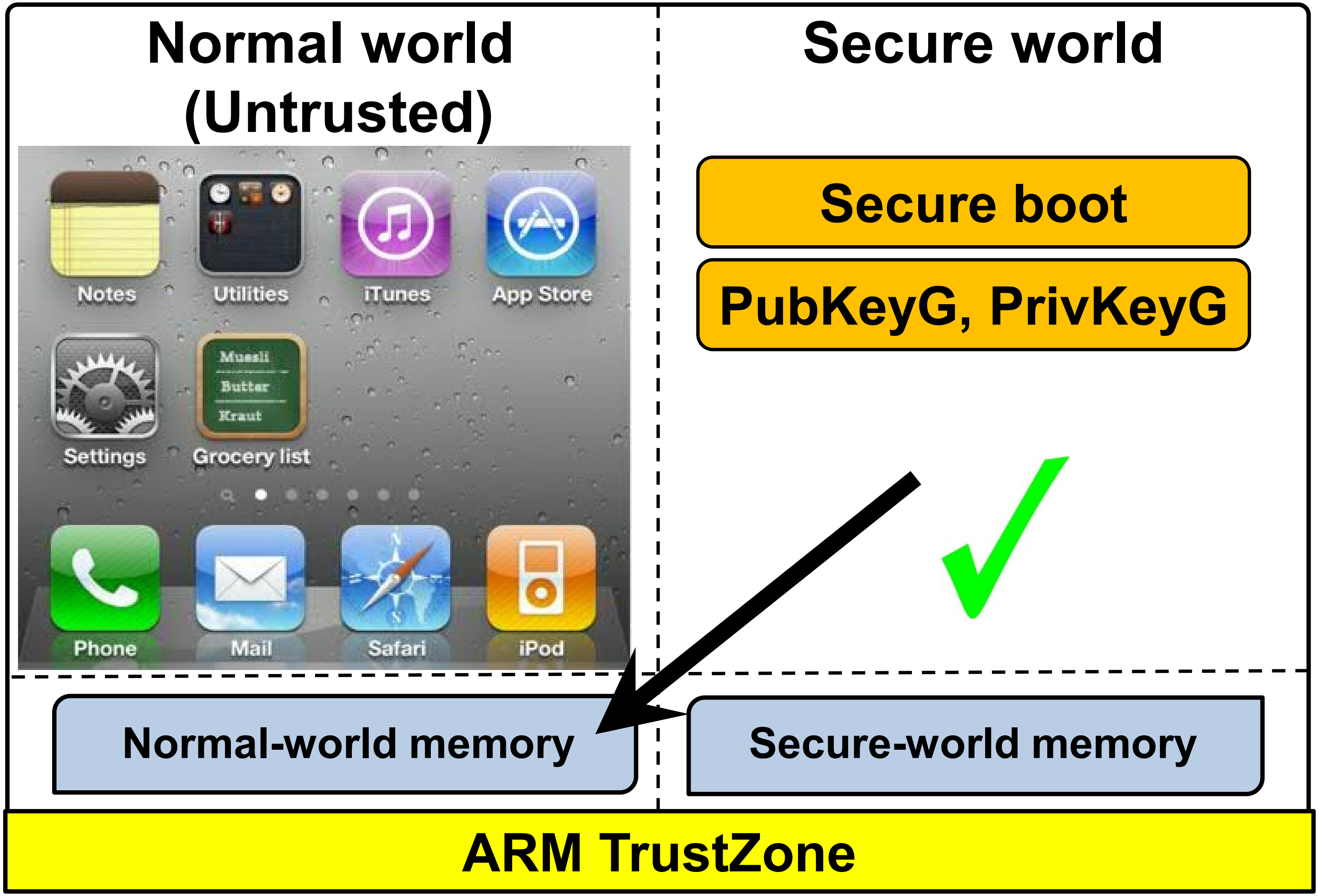
PubKeyG, PrivKeyG

Normal-world memory

Secure-world memory

ARM TrustZone

Memory is partitioned



Normal world (Untrusted)

Secure world

Secure boot

PubKeyG, PrivKeyG

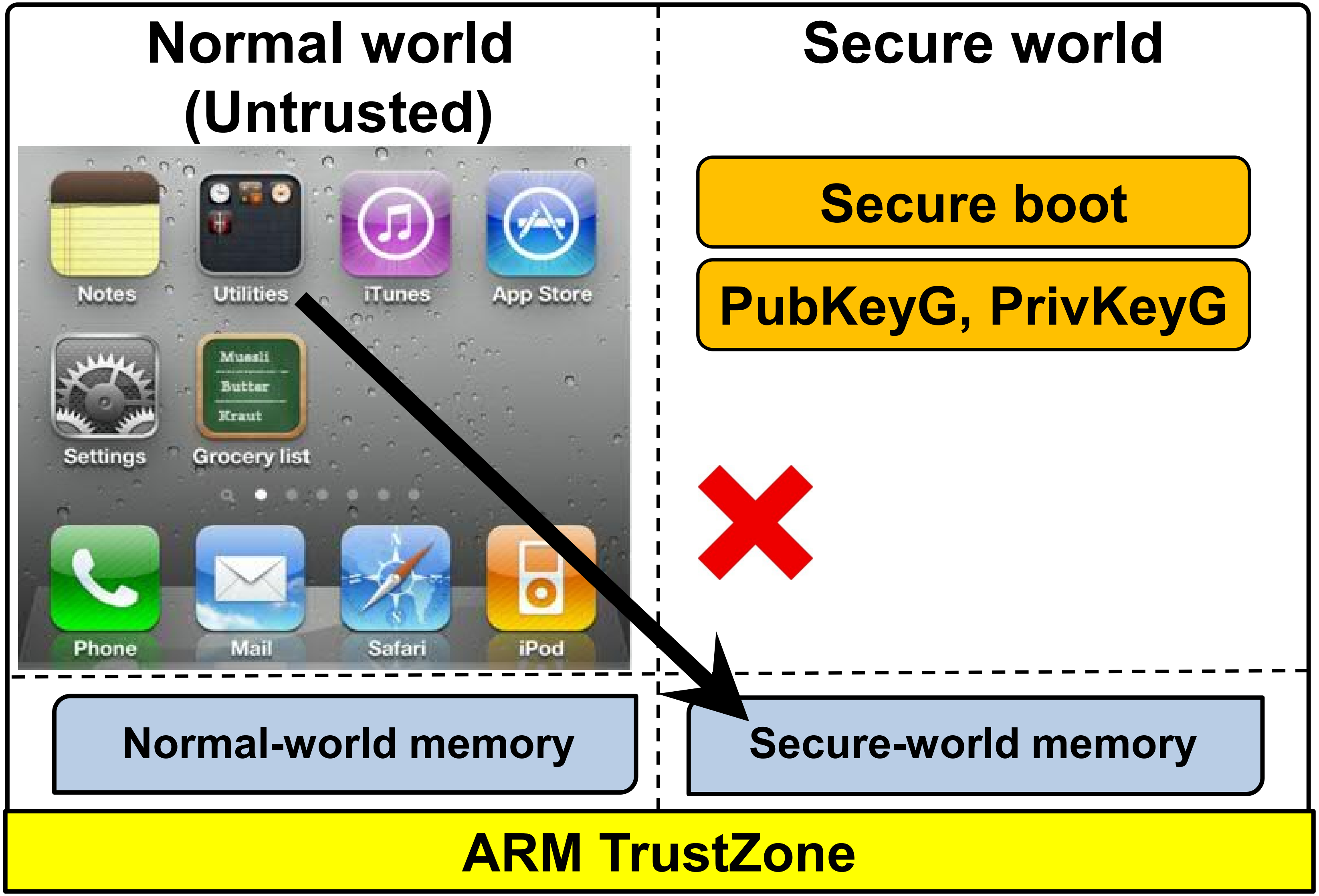
✓

Normal-world memory

Secure-world memory

ARM TrustZone

Memory is partitioned



Normal world (Untrusted)

Secure world

Secure boot

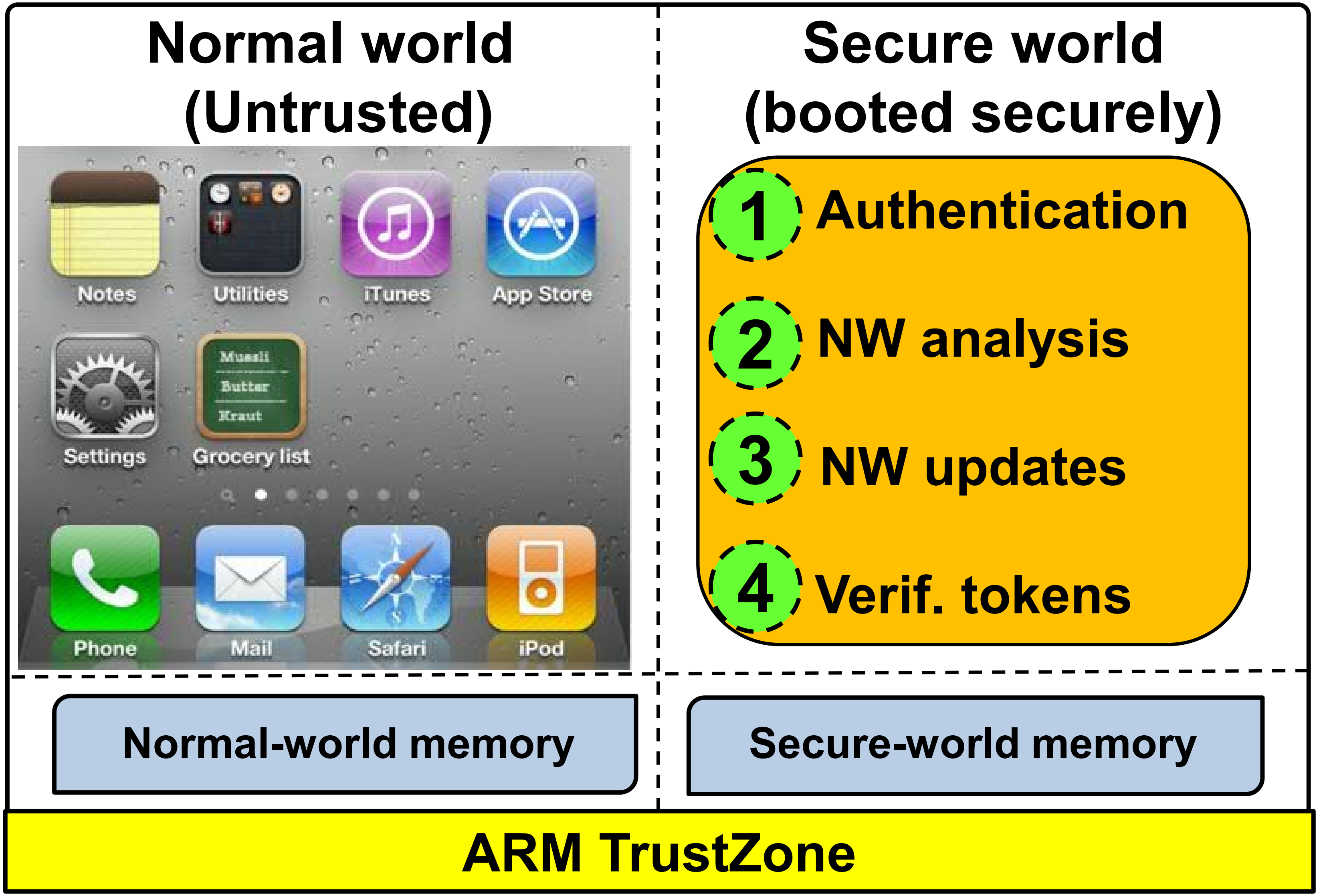
PubKeyG, PrivKeyG

Normal-world memory

Secure-world memory

ARM TrustZone

We enhance the secure world



Normal world (Untrusted)

Secure world (booted securely)

1. Authentication
2. NW analysis
3. NW updates
4. Verif. tokens

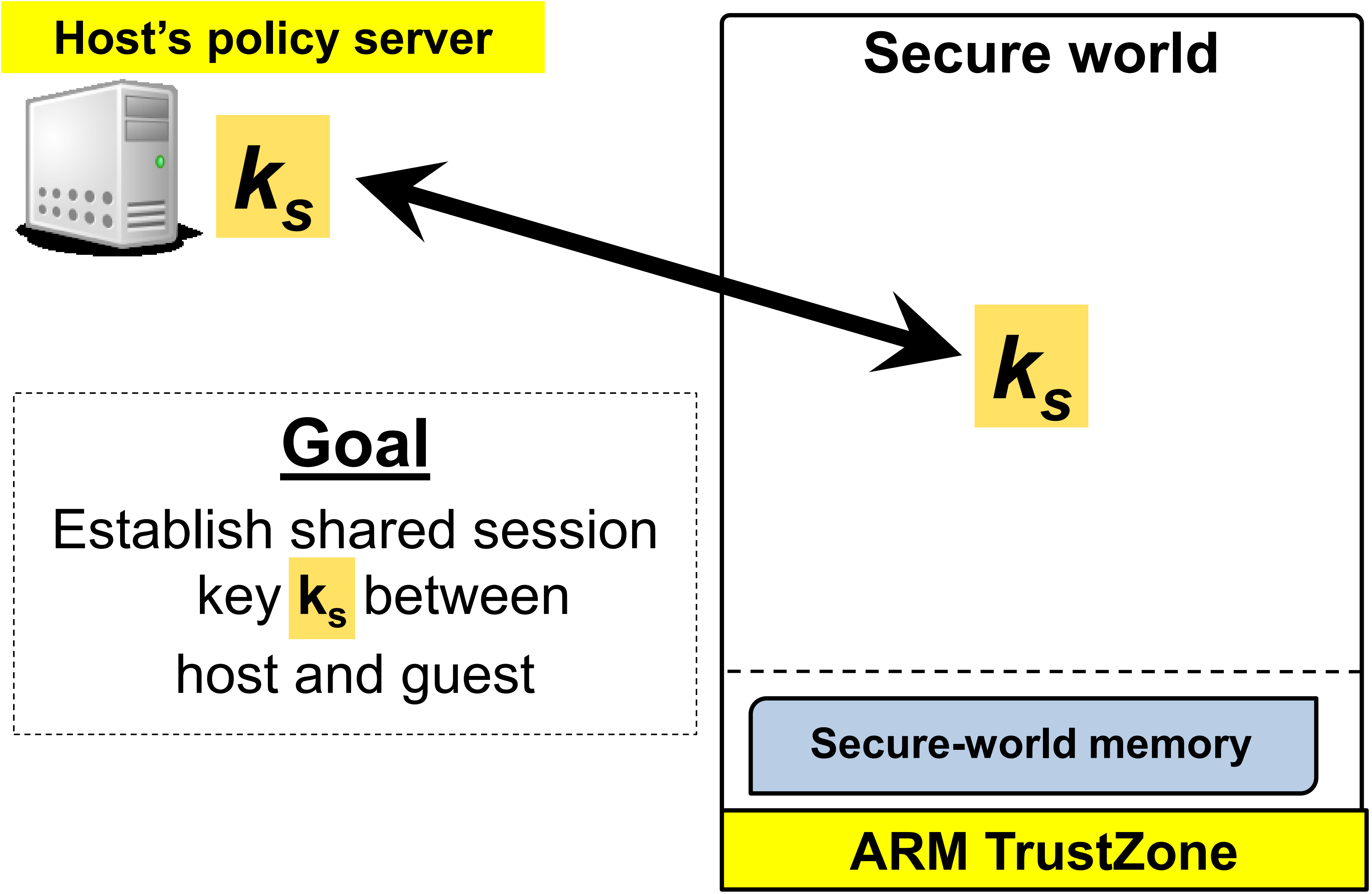
Normal-world memory

Secure-world memory

ARM TrustZone

1

Mutual authentication



Goal

Establish shared session key k_s between host and guest

1

Establishing session key $\boldsymbol{k_s}$

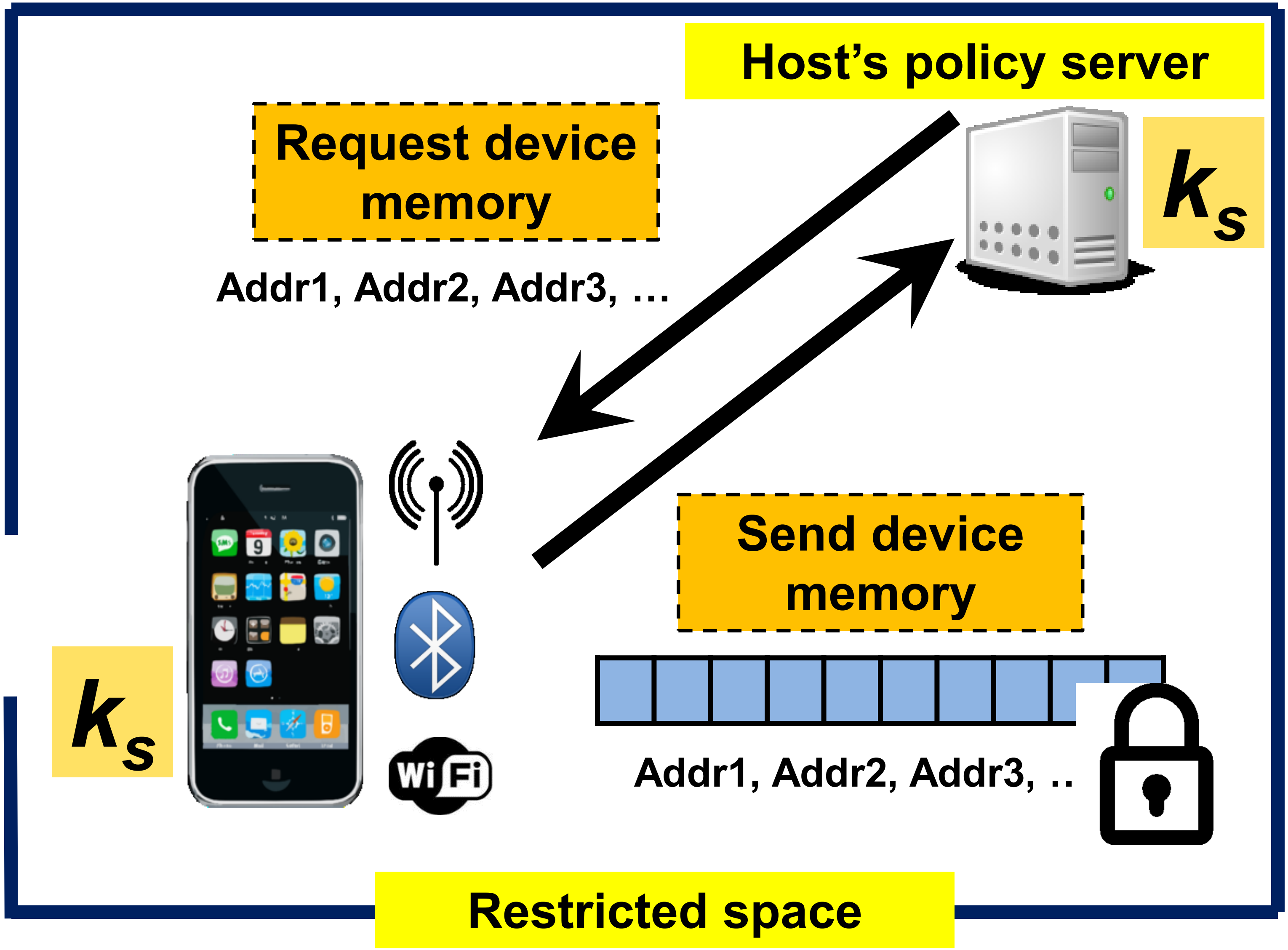
Simplified TLS/SSL handshake

- Host's keypair: **PubKeyH**, **PrivKeyH**
- Guest's keypair: **PubKeyG**, **PrivKeyG**

1. **Guest ←→ Host**: Exchange/verify public keys
2. **Host → Guest**: $Enc_{\mathbf{PubKeyG}}(\boldsymbol{k_s})$+ $Signature_{\mathbf{PrivKeyH}}$
3. **Guest (secure world)**: Verify host signature, decrypt message and obtain $\boldsymbol{k_s}$

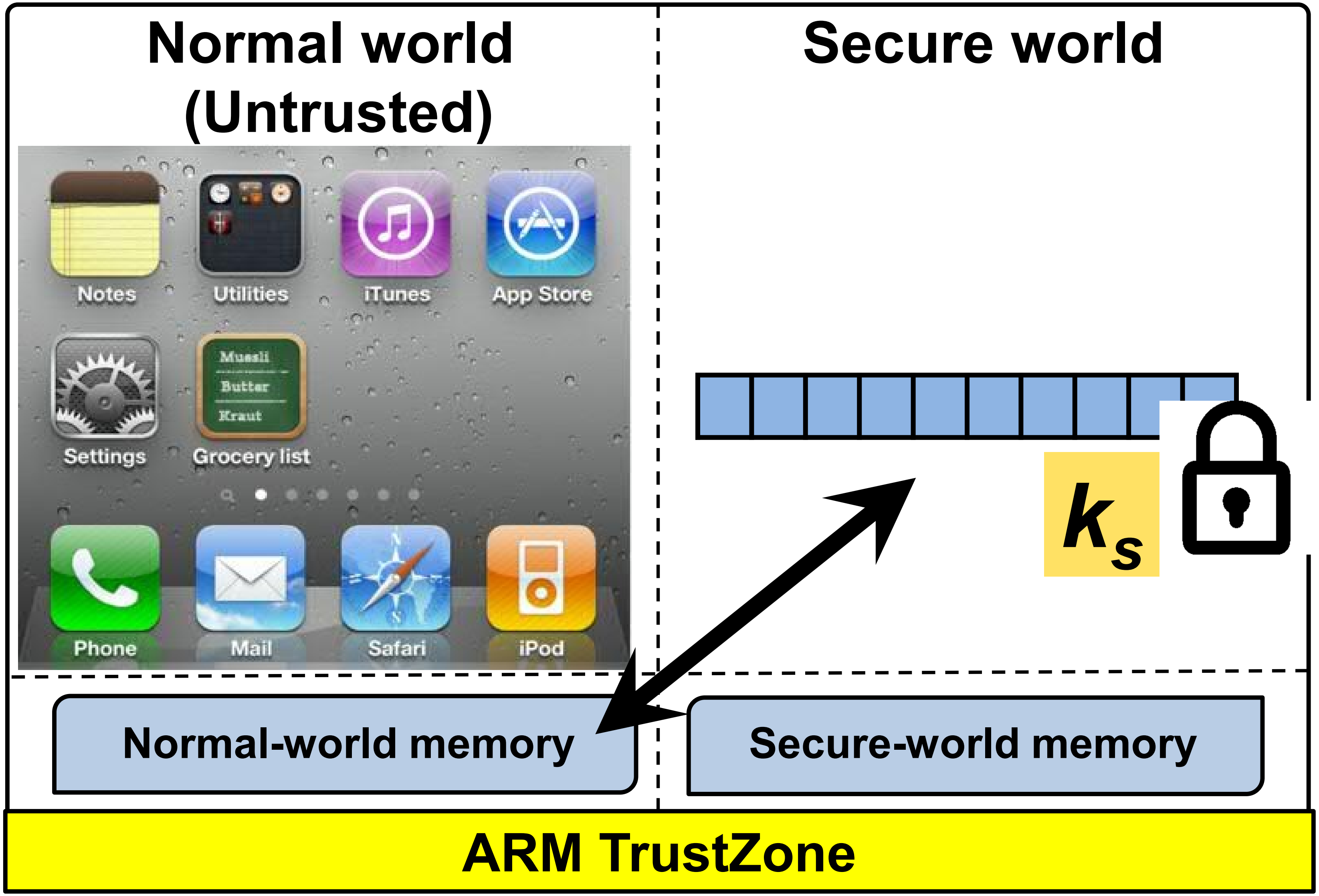
2

Guest device analysis



2

SW reads NW memory



Normal world (Untrusted)

Secure world

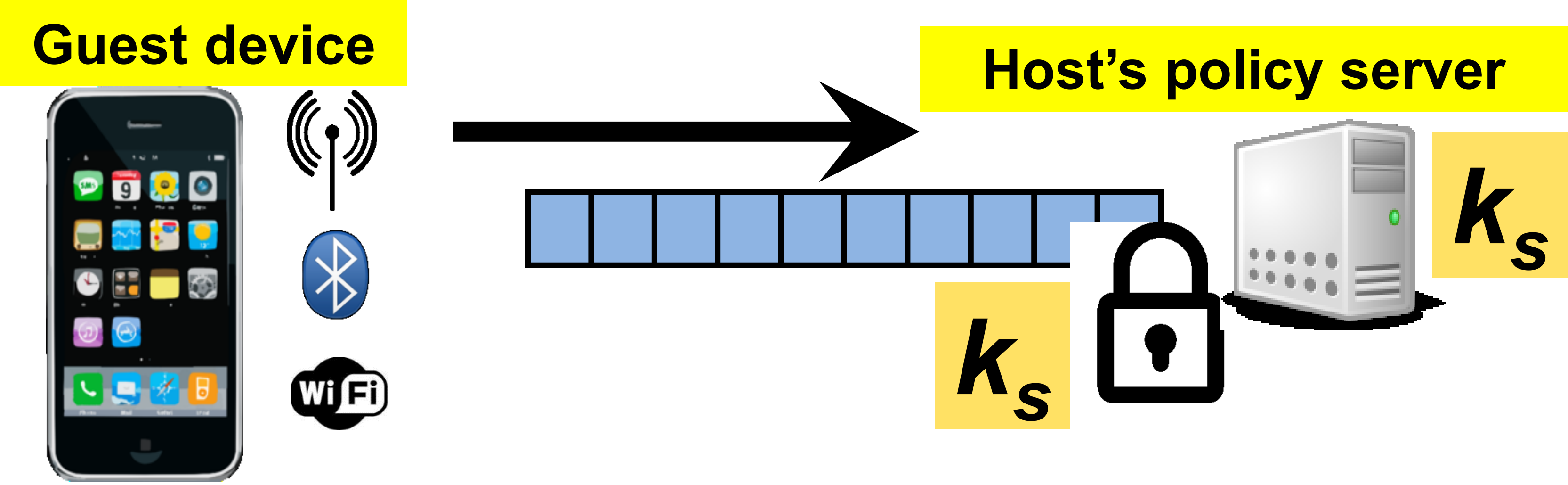
k_s

Normal-world memory

Secure-world memory

ARM TrustZone

2 Analysis of NW memory snapshot



Guest device

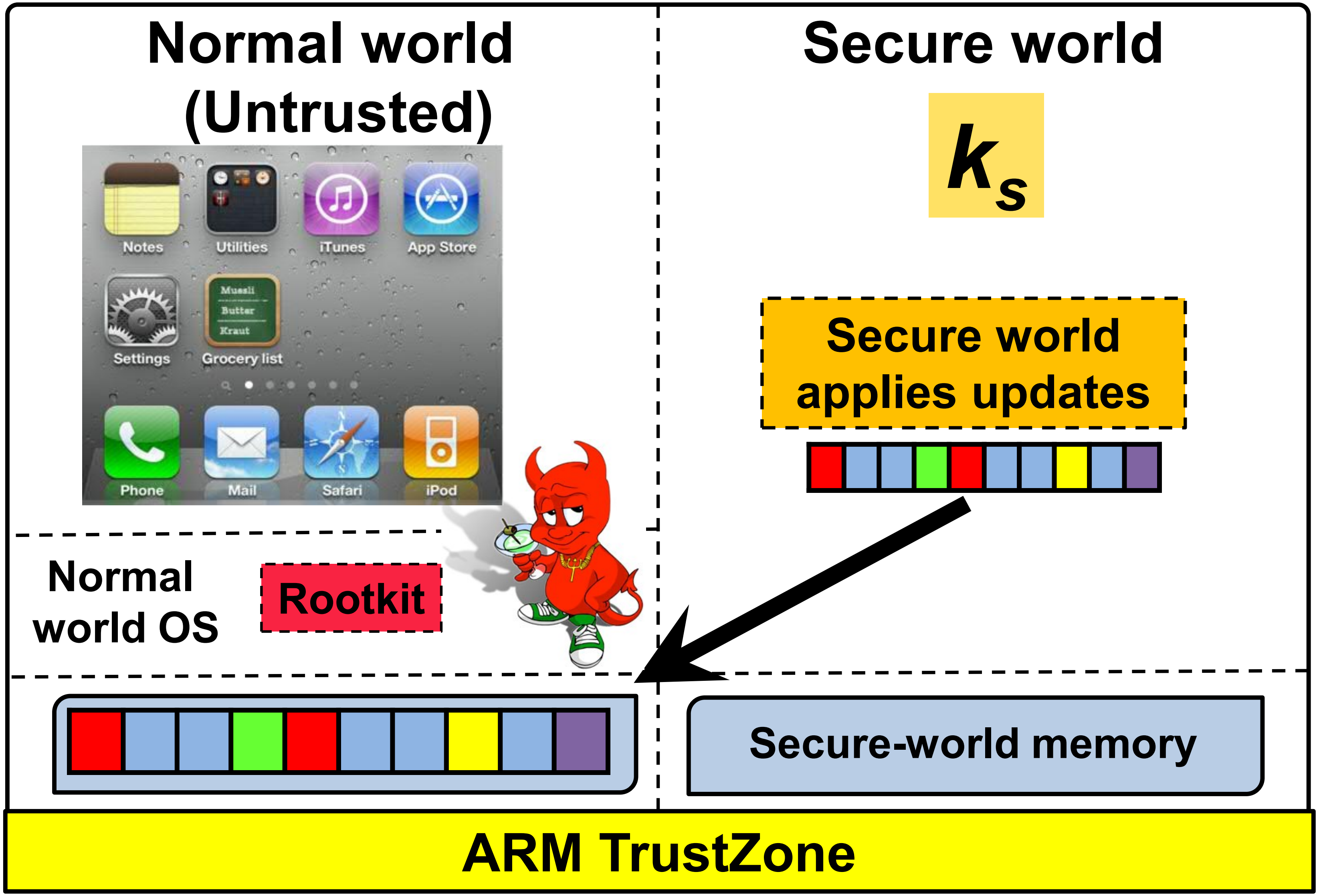
Host's policy server

- Infer what peripherals are installed, and where in memory their drivers are installed
- Detect guest device for malware infection, including kernel-level rootkits

[Baliga, Ganapathy, Iftode, ACSAC'08, TDSC'11]

2

Why look for NW rootkits?



Normal world (Untrusted)

Secure world

k_s

Secure world applies updates

Normal world OS

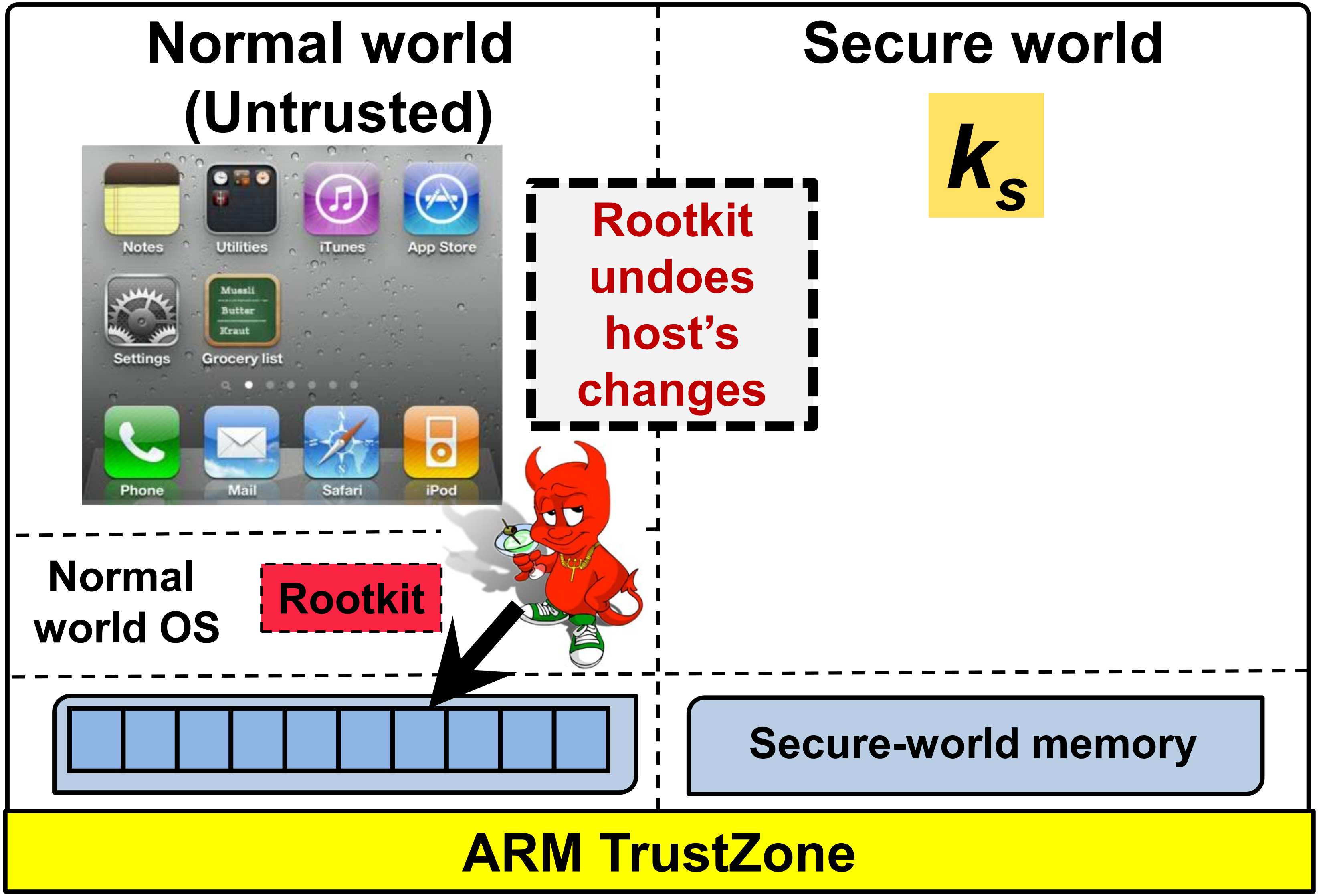
Rootkit

Secure-world memory

ARM TrustZone

2

Why look for NW rootkits?



Normal world (Untrusted)

Secure world

k_s

Rootkit undoes host's changes

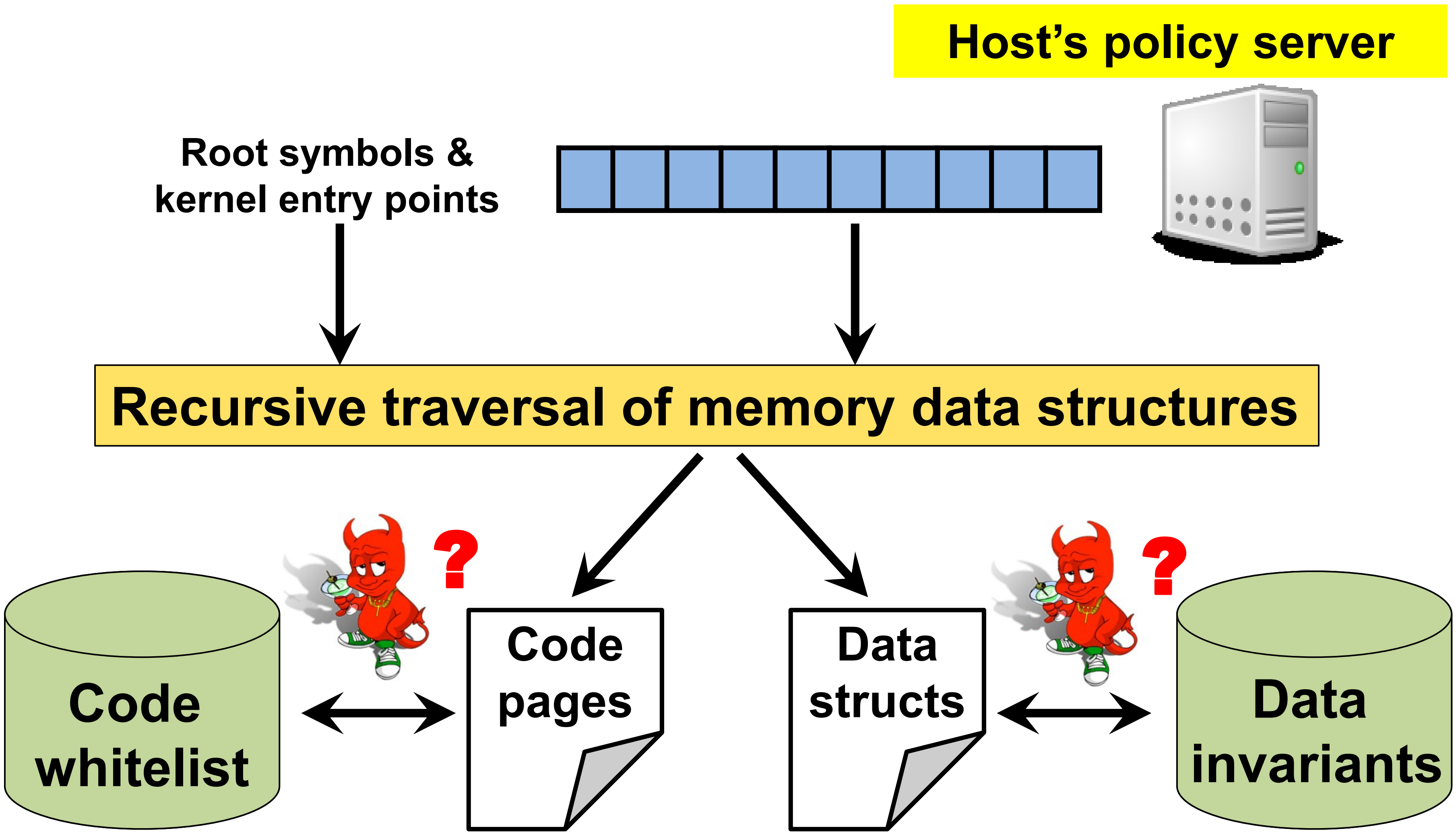
Normal world OS

Rootkit

Secure-world memory

ARM TrustZone

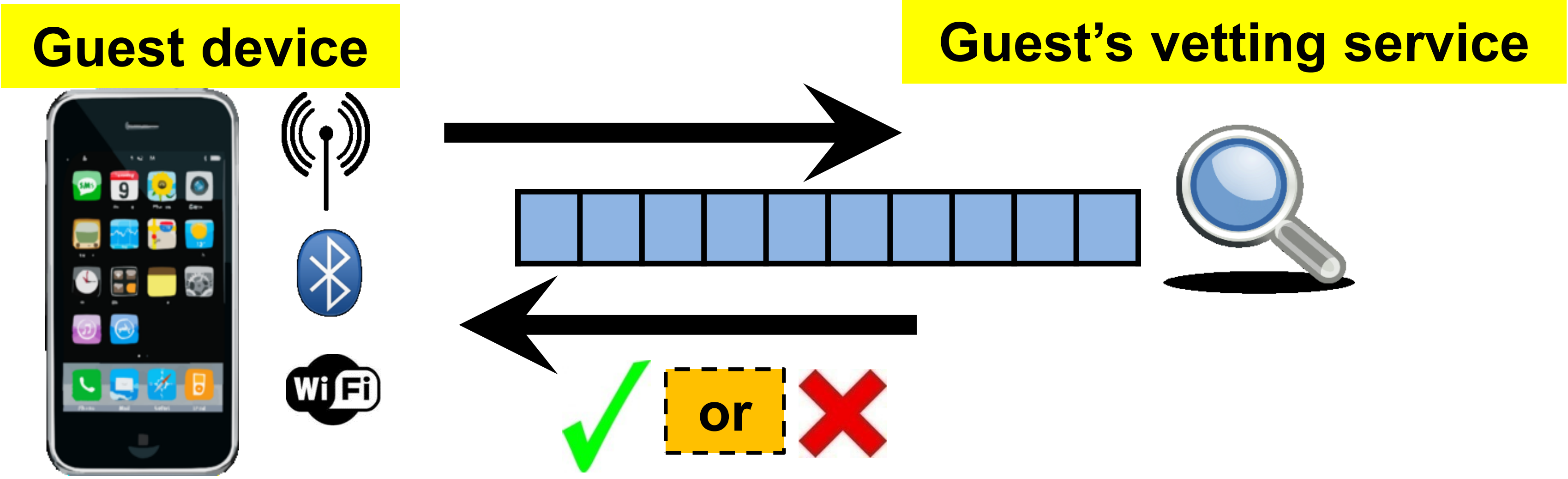
2 Analysis of NW memory snapshot



Host’s policy server
Root symbols & kernel entry points
Recursive traversal of memory data structures
Code whitelist
?
Code pages
Data structs
?
Data invariants

2

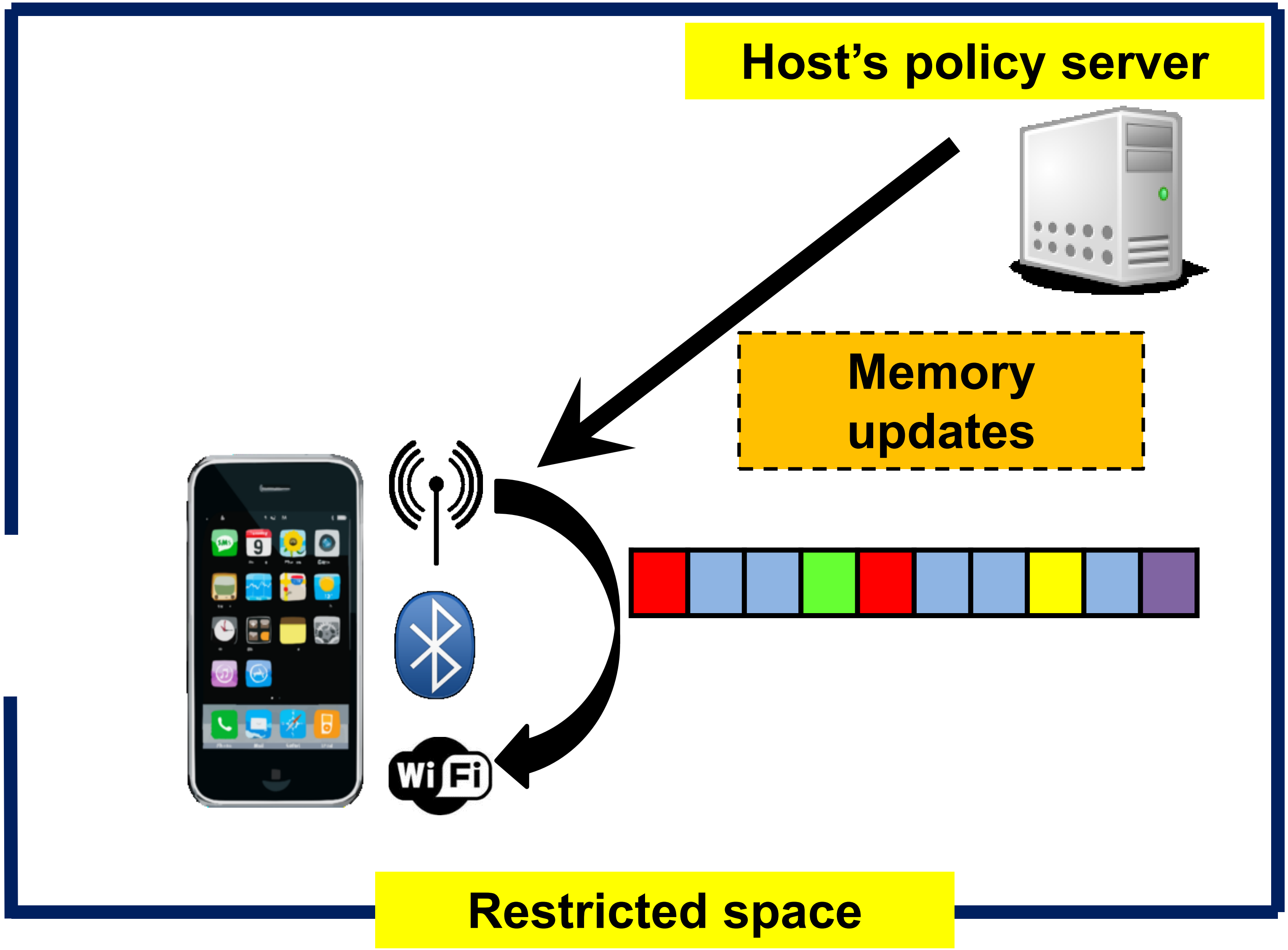
Vetting host’s requests



- Vetting server ensures that host’s requests do not compromise guest privacy
- **Vetting policy**: Host only allowed to request *guest device’s kernel memory*

3

Guest device update



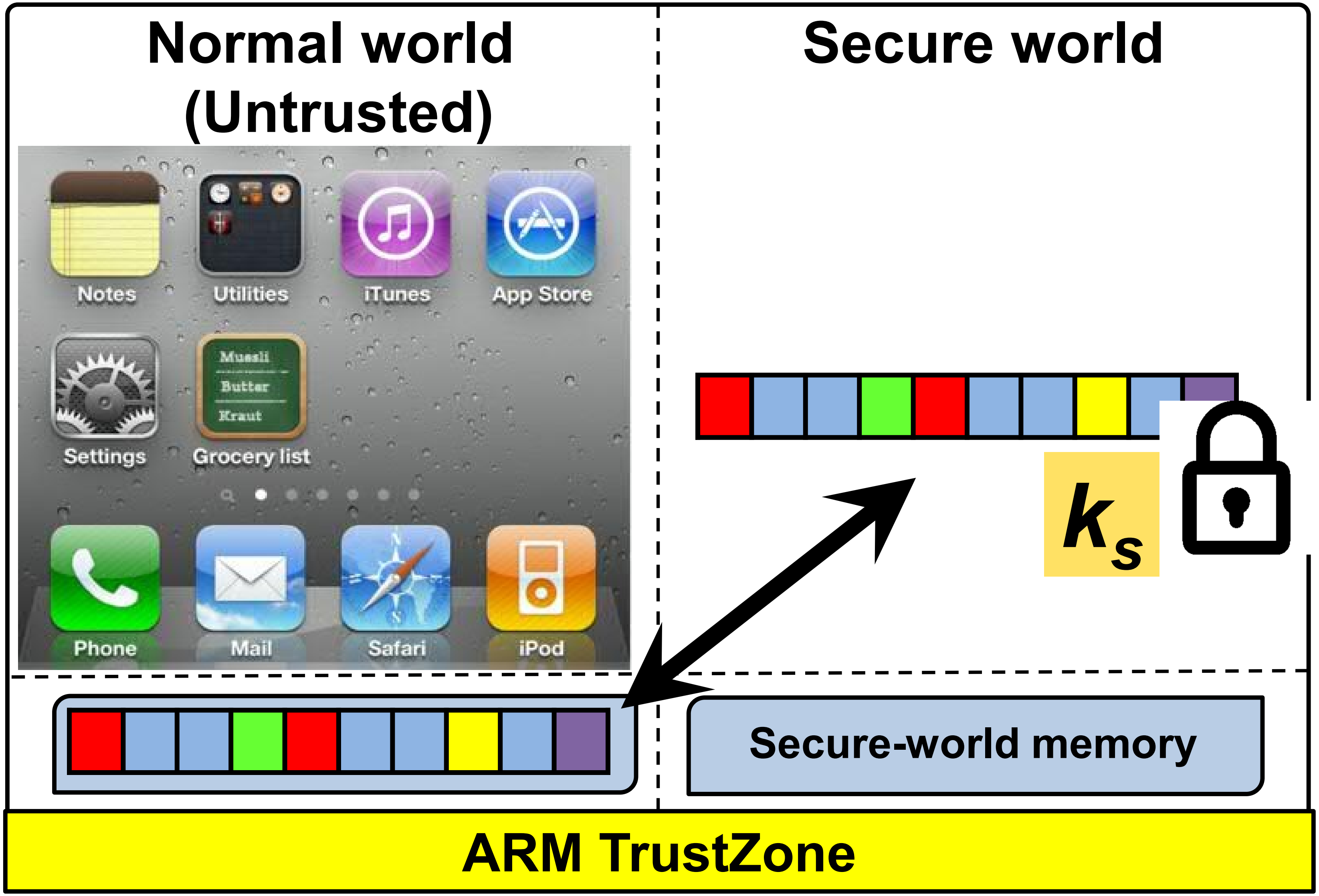
Host’s policy server

Memory updates

Restricted space

3

SW updates NW memory



Normal world (Untrusted)

Secure world

k_s

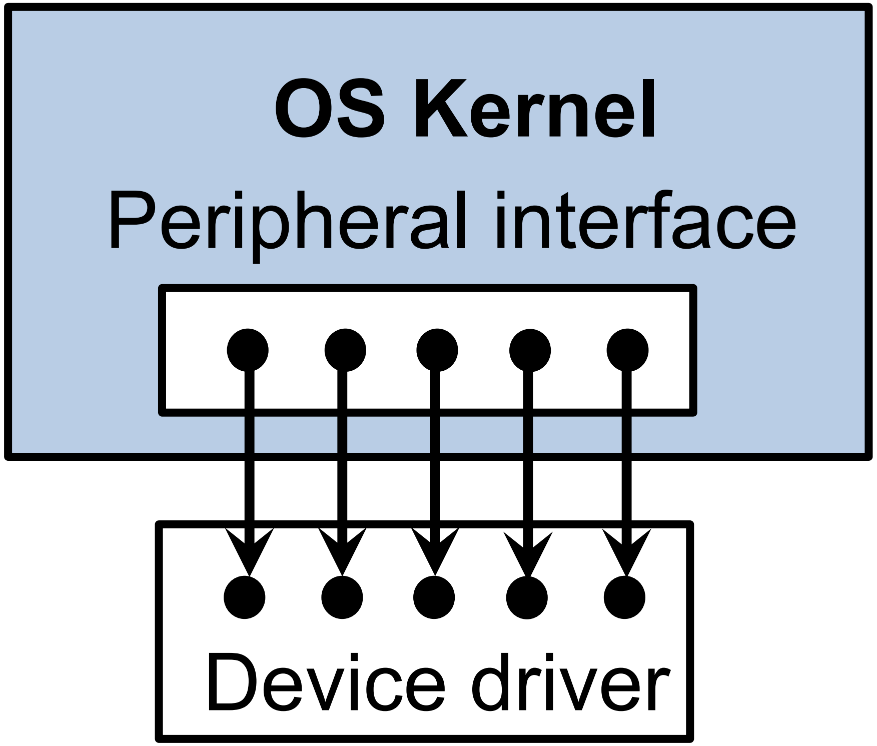
Secure-world memory

ARM TrustZone

3

Updating peripheral drivers

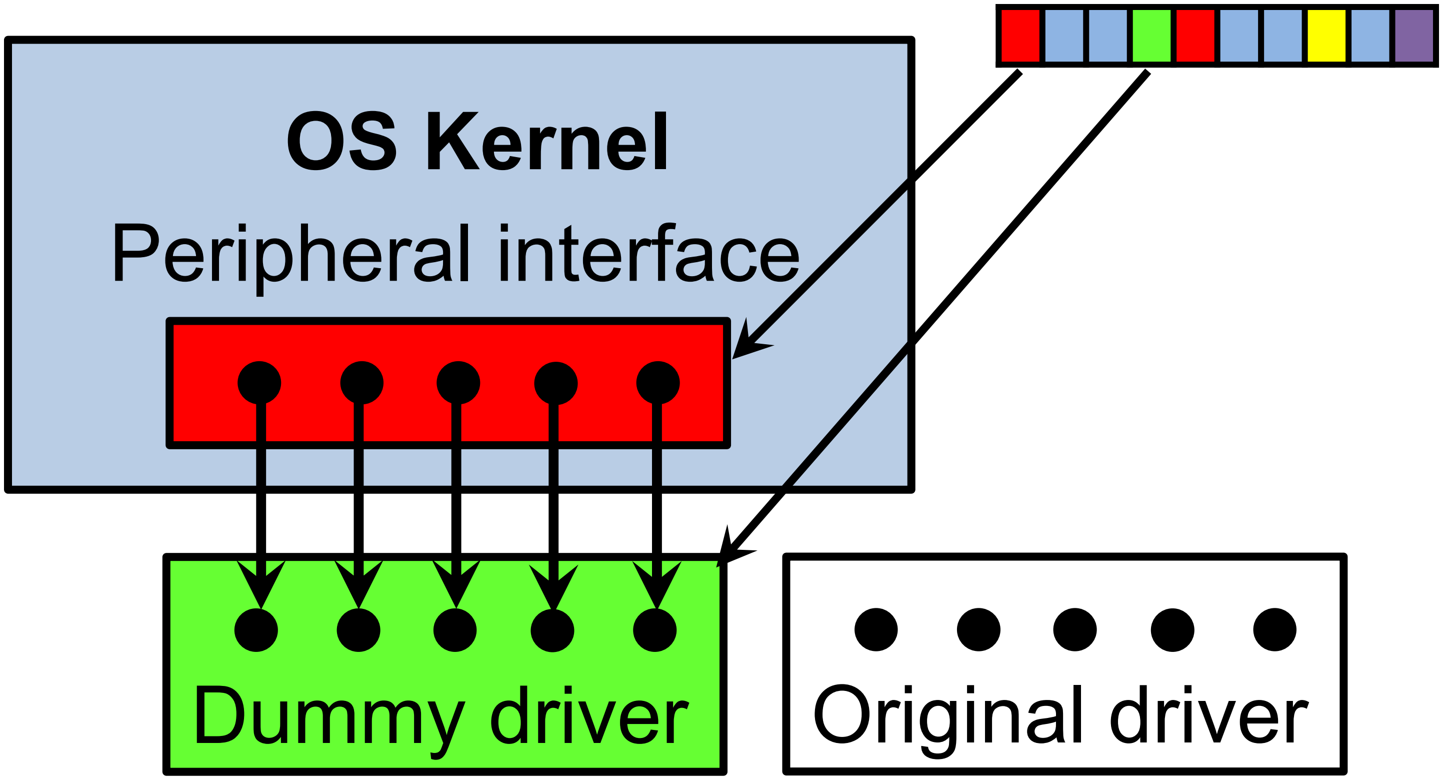
- Device drivers in normal world control execution of device peripherals



3

Updating peripheral drivers

- Introduce dummy driver to control peripheral (e.g., disable it). Update kernel driver hooks.



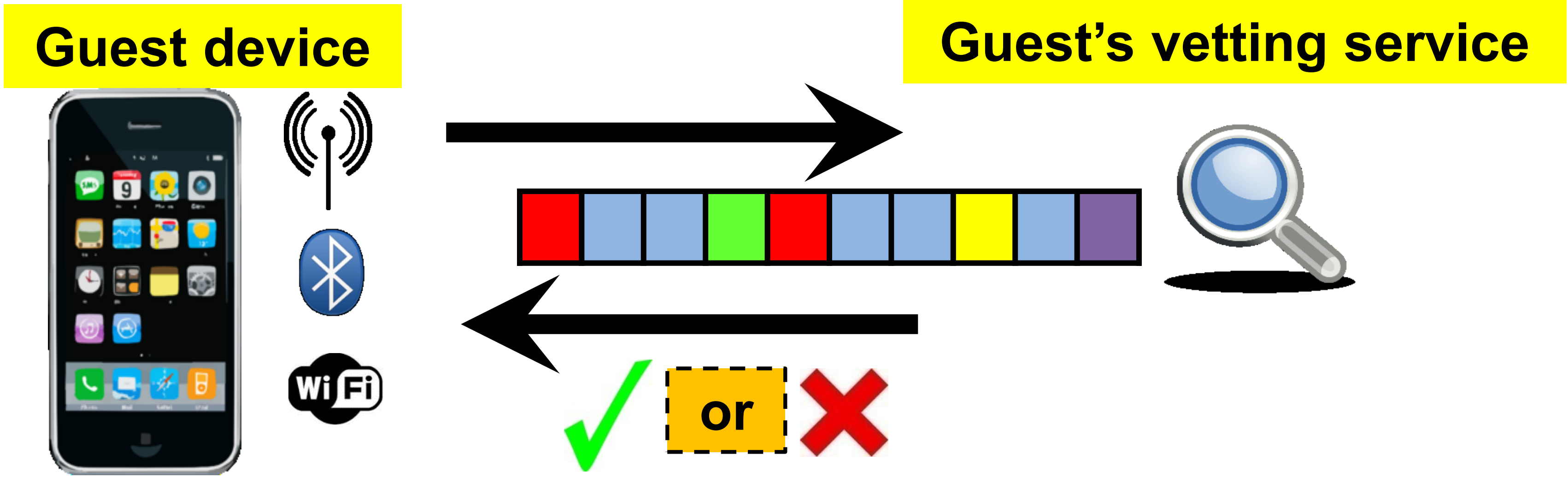
3

Are driver updates effective?

Peripheral considered	Update size (bytes)	Guest device	Peripheral disabled?
USB webcam	302	i.MX53	✓
Camera	212	Nexus phone	✓
WiFi	338	Nexus phone	✓
3G (Data)	252	Nexus phone	✓
3G (Voice)	224	Nexus phone	✓
Microphone	184	Nexus phone	✓
Bluetooth	132	Nexus phone	✓

3

Vetting host’s updates



Guest device

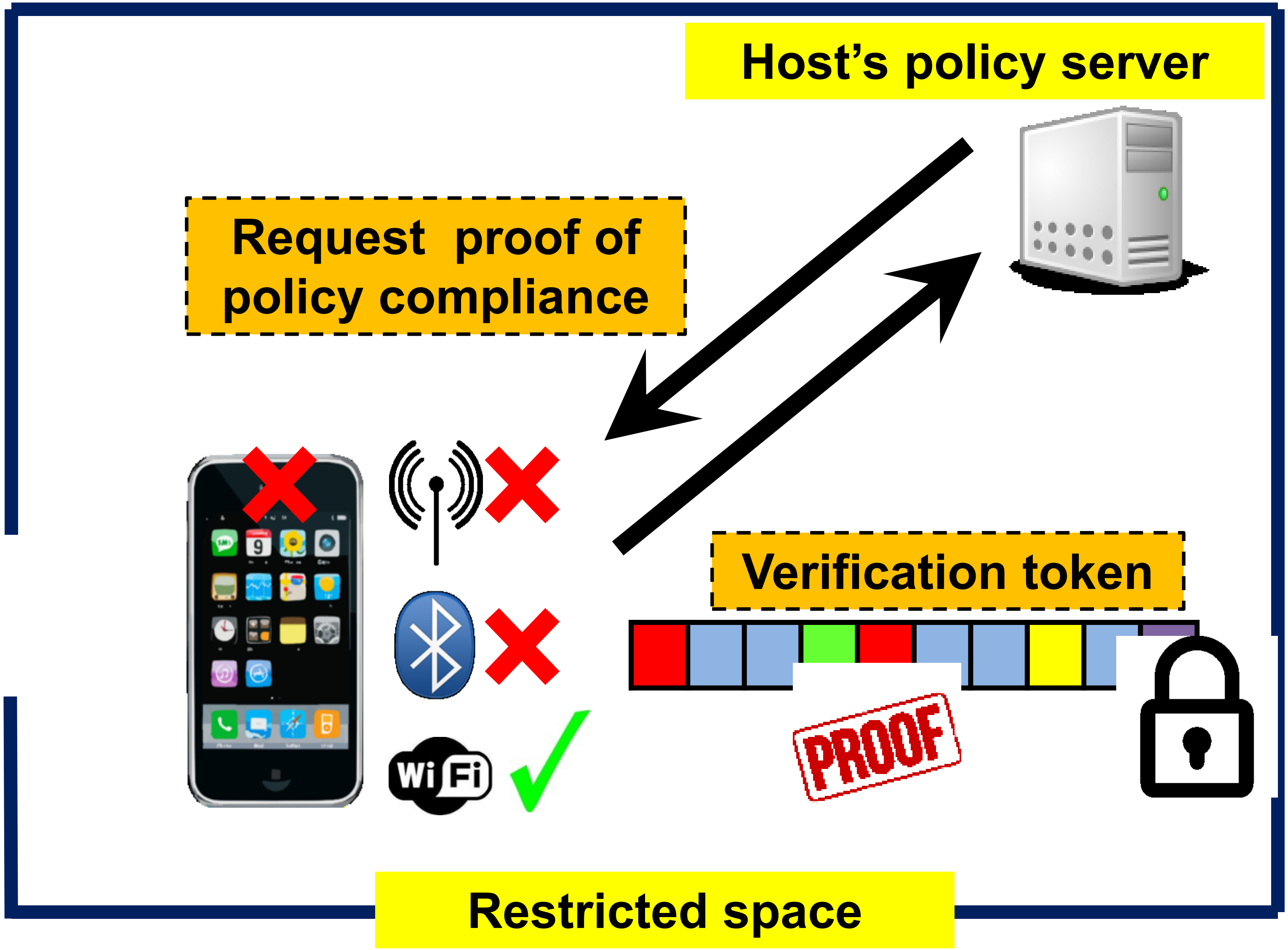
Guest’s vetting service

✓ or ✗

- An untrusted host can introduce new code into guest devices
- **Vetting policy**: Ensure that dummy drivers are a *subset* of the original drivers
  - – Via ARM-binary analysis on 🟥🟦🟦🟩🟥🟦🟦🟨🟦🟪

4

Proof of compliance



Host’s policy server

Request proof of policy compliance

Verification token

Restricted space

4

Verification tokens

- Host requests proof of compliance
- Secure world computes a fresh snapshot of all NW memory locations updated by host
- Verification token:

HMAC(🟥🟦🟦🟩🟥🟦🟦🟨🟦🟪, k_s)

- Verification token matches if and only if normal world memory still in compliance with the host’s usage policy

Memory updates are ephemeral

- Guest device can violate host’s usage policies by simply rebooting to undo host’s memory updates!

- Once device checked in, secure world must:
  - Mediate all low-battery and power-off interrupts
  - Checkpoint device memory to disk
  - Upon power up, must restore device memory from checkpoint

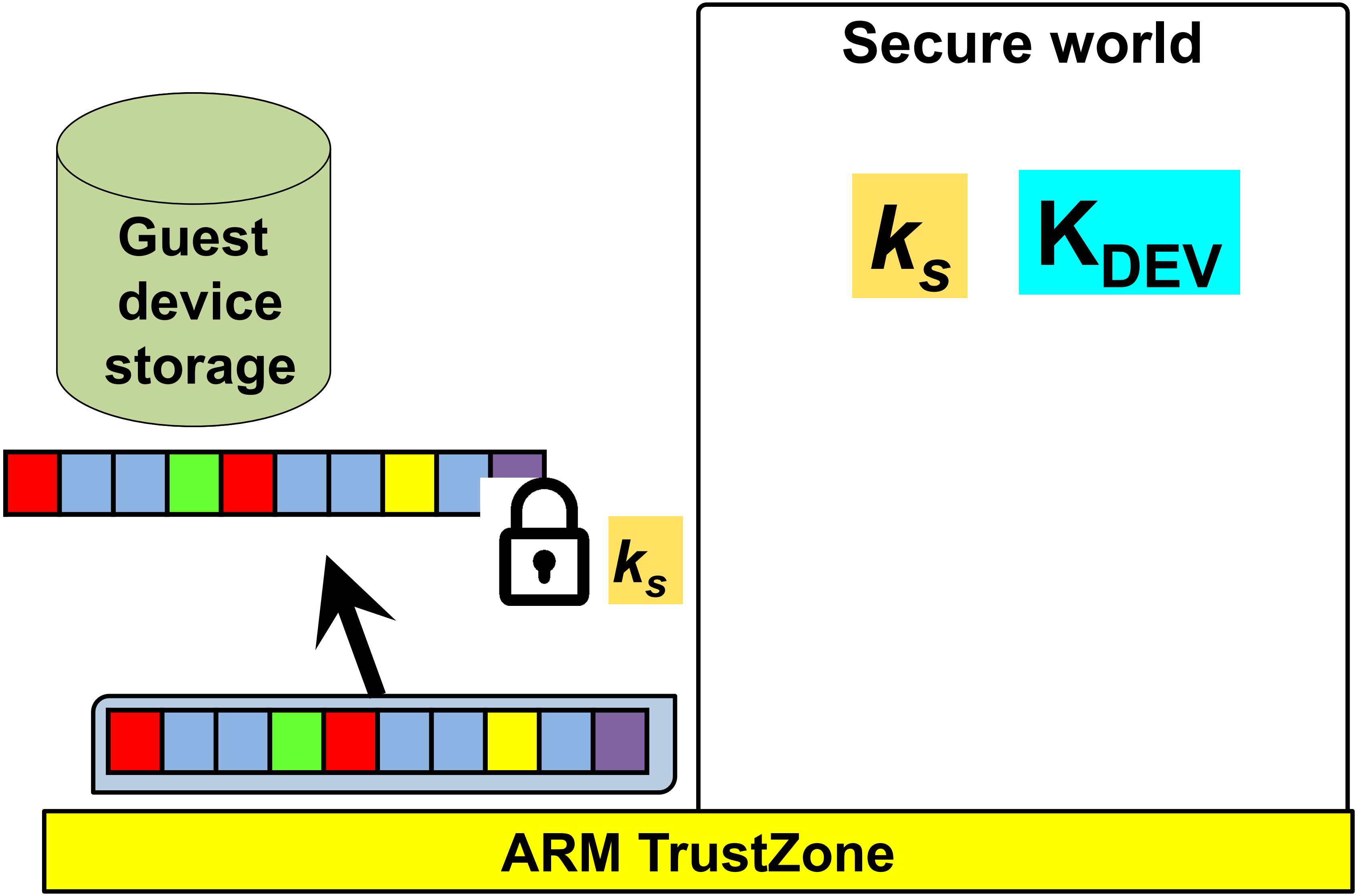
Device checkpoint

- **Problem**: Checkpoint stored on disk
  - Readable by untrusted end-user
  - But session key $\boldsymbol{k_s}$ must not be stored in clear
  - Otherwise, malicious end-user can use it to impersonate guest's trusted secure world!
- **Solution**: **REM-suspend** protocol

REM-suspend

- ARM TrustZone equips each device with a device-specific key $\mathbf{K_{DEV}}$
- The key $\mathbf{K_{DEV}}$ is only accessible from the secure world
- We use $\mathbf{K_{DEV}}$ to encrypt $\boldsymbol{k_s}$ in device checkpoint
- When device is powered again, secure world uses $\mathbf{K_{DEV}}$ to decrypt and restore $\boldsymbol{k_s}$

REM-suspend



Secure world

k_s

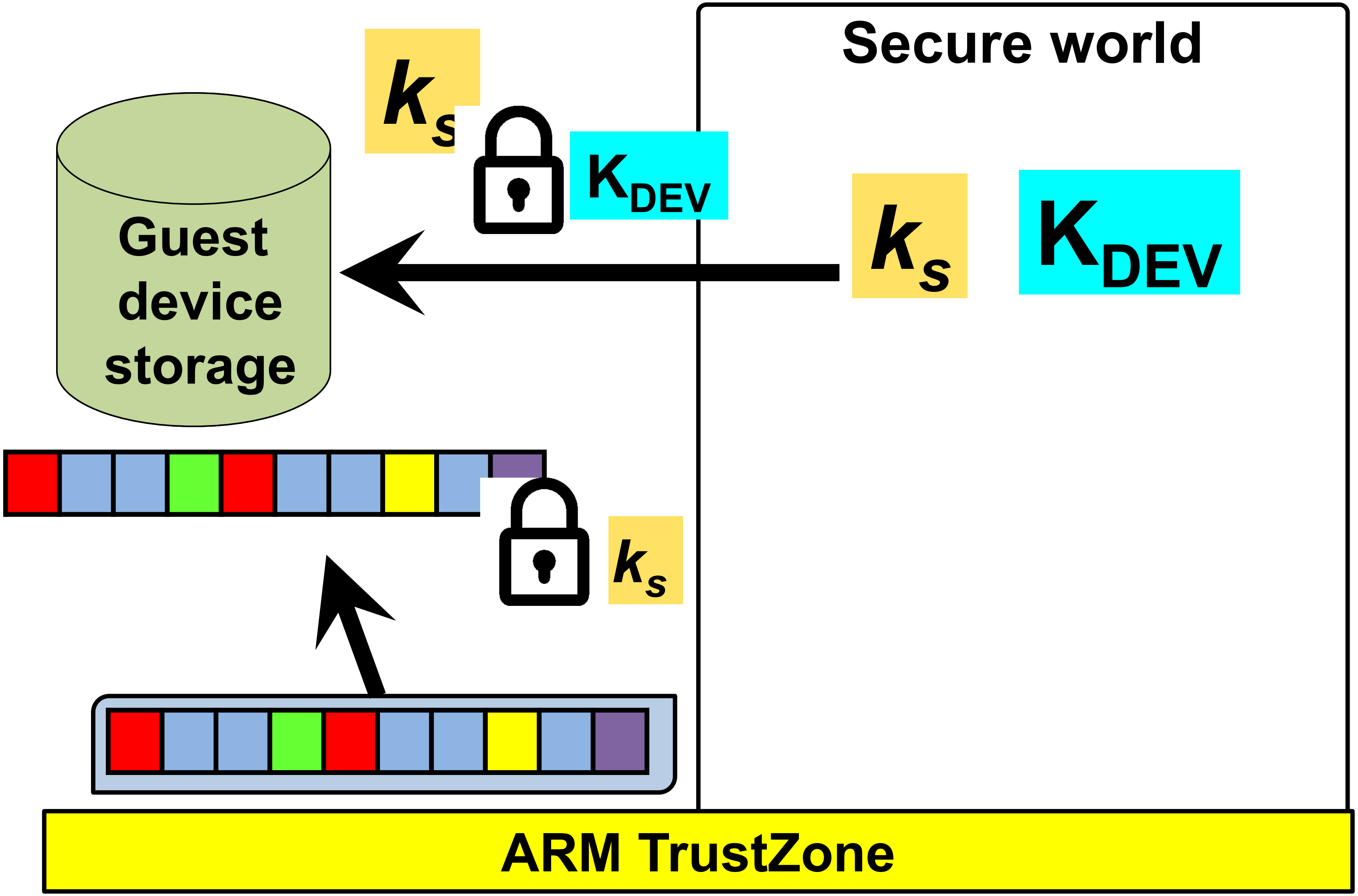
K_{DEV}

Guest device storage

k_s

ARM TrustZone

REM-suspend


Secure world
k_s
K_{DEV}
k_s
K_{DEV}
Guest device storage
k_s
ARM TrustZone

Are memory updates the right API?

- Powerful, low-level API for device control
- Simplifies design of secure world (TCB) and keeps it device-independent

TCB component	SLOC
Memory manager	1381
Authentication	1285
Memory ops., verification tokens	305
REM-suspend	609
SHA1 + HMAC	861
X509	877
RSA	2307

Do memory updates affect app stability?

Passive updates: Update memory and start the app

USB	*MobileWebCam*	*ZOOM FX*	*Retrica*	*Candy Cam*	*HD Cam Ultra*
	App Error	**Android Error**	**App Error**	**App Error**	**Android Error**
Camera	*Android Cam*	*Camera MX*	*ZOOM FX*	*Droid HD Cam*	*HD Cam Ultra*
	Android Error	**App Error**	**App Error**	**Android Error**	**Android Error**
WiFi	*Spotify*	*Play Store*	*YouTube*	*Chrome*	*Facebook*
	No Connection	**No Connection**	**No Connection**	**No Connection**	**No Connection**
3G (Data)	*Spotify*	*Play Store*	*YouTube*	*Chrome*	*Facebook*
	No Connection	**No Connection**	**No Connection**	**No Connection**	**No Connection**
3G (Voice)	*Default call application*				
	Unable to place call				
Micro-phone	*Audio rec*	*Easy voice rec*	*Smart voice rec*	*Snd/voice rec*	*Smart voice rec*
	App Error	**App Error**	**App Error**	**App Error**	**App Error**

Do memory updates affect app stability?

Active updates: Update memory with “live” app

USB	*MobileWebCam*	*ZOOM FX*	*Retrica*	*Candy Cam*	*HD Cam Ultra*
	App Error	**App Error**	**App Error**	**App Error**	**App Error**
Camera	*Android Cam*	*Camera MX*	*ZOOM FX*	*Droid HD Cam*	*HD Cam Ultra*
	Blank Screen	**App Error**	**Android Error**	**Blank Screen**	**Blank Screen**
WiFi	*Spotify*	*Play Store*	*YouTube*	*Chrome*	*Facebook*
	No Connection	**No Connection**	**No Connection**	**No Connection**	**No Connection**
3G (Data)	*Spotify*	*Play Store*	*YouTube*	*Chrome*	*Facebook*
	No Connection	**No Connection**	**No Connection**	**No Connection**	**No Connection**
3G (Voice)	*Default call application*				
	Unable to place call				
Micro-phone	*Audio rec*	*Easy voice rec*	*Smart voice rec*	*Snd/voice rec*	*Smart voice rec*
	Empty File	**Empty File**	**Empty File**	**Empty File**	**Empty File**

Related approaches

- Device virtualization:
  - Heavyweight; probably not for all devices
  - Still requires host to trust hypervisor on guest
- Mobile device management solutions:
  - No proofs to host
  - Device-dependent TCB on guest
- Context-based access control:
  - Same shortcomings as MDM solutions above

Conclusion

A systematic method to regulate devices and ensure responsible use

- Low-level API allows hosts to analyze and control guests
  - Simplifies design and size of TCB
- Hosts can obtain proofs of guest compliance
  - Relies on ARM TrustZone hardware
- Vetting service balances guest privacy with host's usage policies

Other research projects…

Generic theme: Computer Systems Security

- Improving cloud platform security
  [ACSAC’08a, RAID’10, CCS’12a, SOCC’14]
- Operating system reliability and security
  [ASPLOS’08, ACSAC’08b, ACSAC’09a, MobiSys’11, TDSC’11, TIFS’13]
- Hardware support for software and system security
  [CCS’08, ECOOP’12a, TIFS’13, MobiSys’16, RU-DCS-TR724]
- Web application and Web browser security
  [ACSAC’09b, ECOOP’12a, ECOOP’12b, ECOOP’14, FSE’14]
- Tools for cross-platform mobile app development
  [ICSE’13, ASE’15]
- Retrofitting legacy software for security
  [CCS’05, Oakland’06, ASPLOS’06, ICSE’07, CCS’08, CCS’12b]
- Validating security retrofitting transformations in optimizing compilers
  [Submitted]

A big thank you to my students

Graduated PhDs

- Dr. Mohan Dhawan (IBM Research India)
- Dr. Saman Zarandioon (Amazon.com)
- Dr. Shakeel Butt (NVidia → now at Google)
- Dr. Liu Yang (HP Labs → now at Baidu)
- Dr. Rezwana Karim (Samsung Research America)
- Dr. Amruta Gokhale (Teradata)

Former Postdocs

- Dr. Arati Baliga (AT&T Security Labs)

Graduated MS students

- Jeffrey Bickford (AT&T Research)
- Yogesh Padmanaban (Microsoft)

Current PhD students

- Jay P. Lim, Hai Nguyen, Daeyoung Kim.

Email: **vinodg@cs.rutgers.edu**
URL: **http://www.cs.rutgers.edu/~vinodg**