ECE 455: CYBERSECURITY

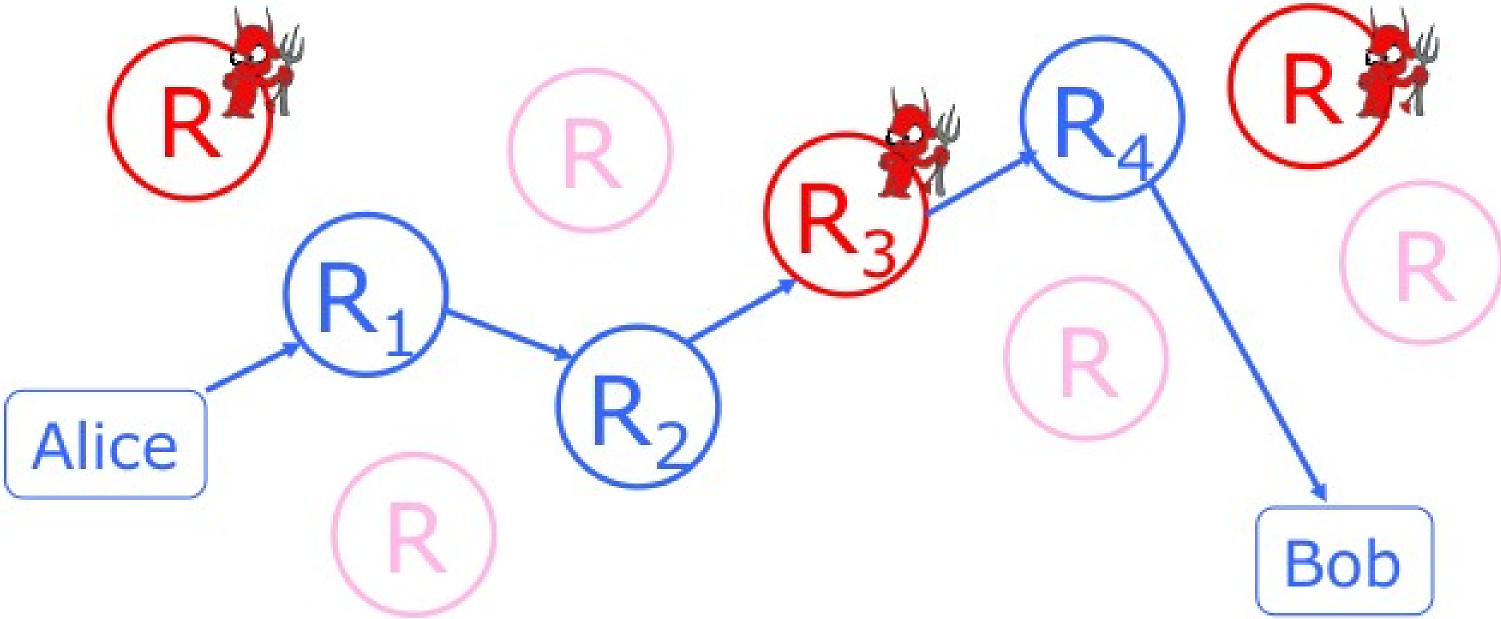
Lecture #11

Daniel Gitzel

TOR

Review: Onion Routing

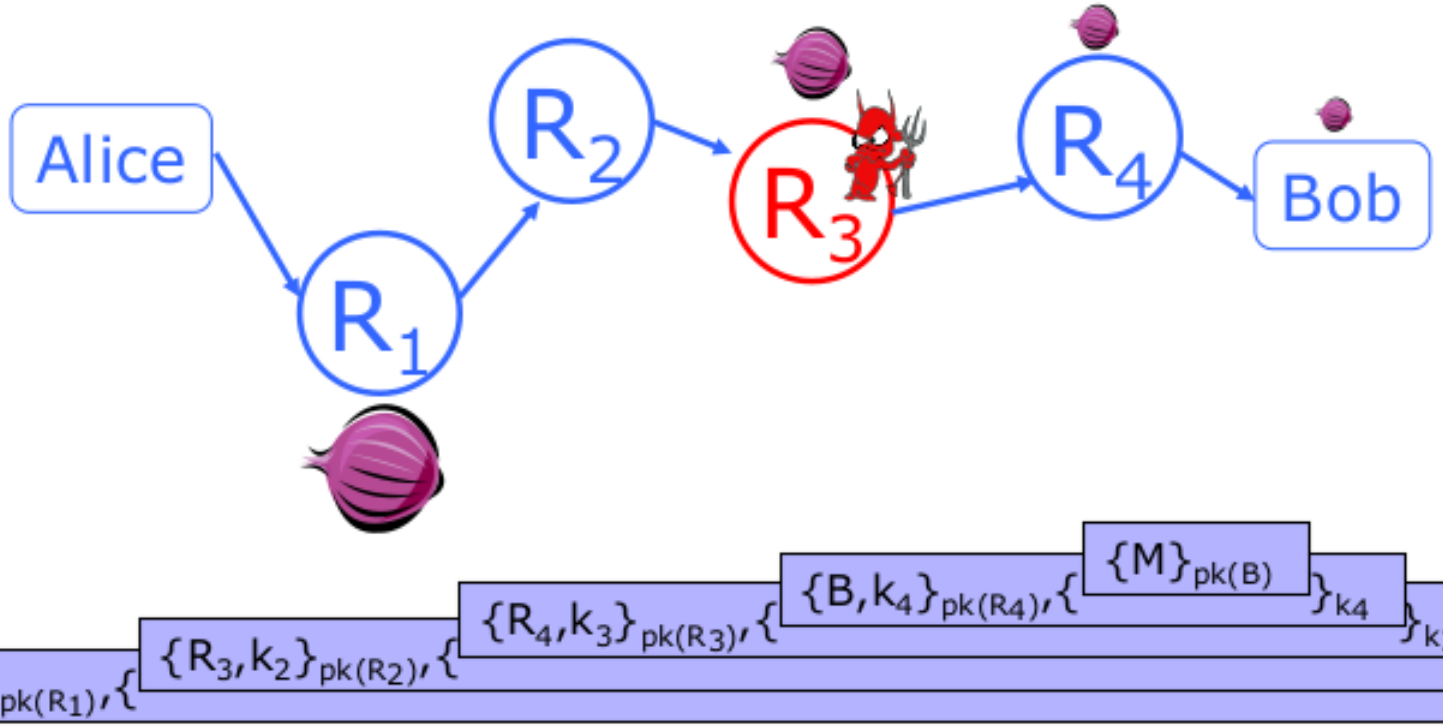
- **Sender chooses a random sequence of routers**
- **Some routers are honest, some controlled by attacker**
- **Sender controls the length of the path**



[Reed, Syverson, Goldschlag 1997]

Route Establishment

- **Routing info for each link encrypted with router's public key**
- **Each router learns only the identity of the next router**

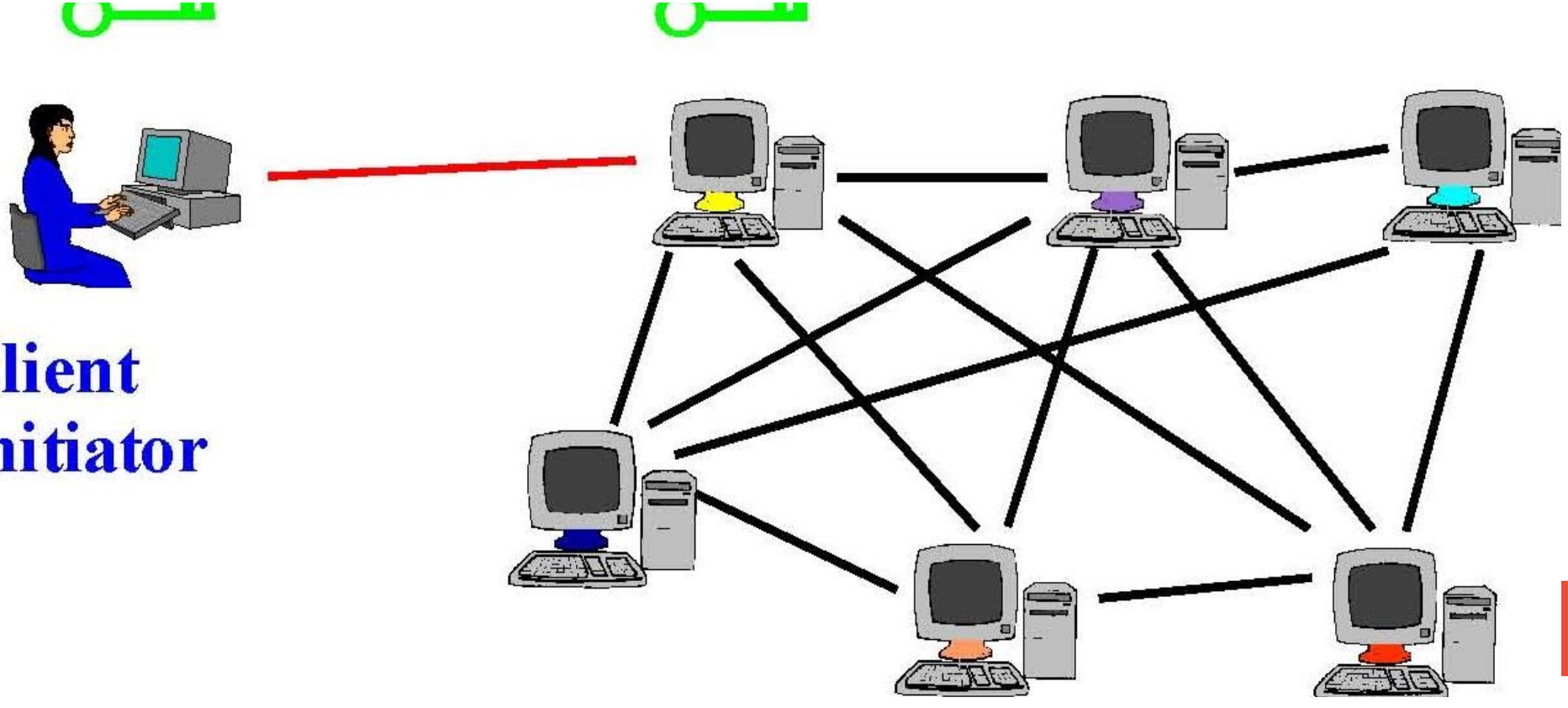


Tor

- **Second-generation onion routing network**
  - https://tor.eff.org
  - Developed by Roger Dingledine, Nick Mathewson and Paul Syverson
  - Specifically designed for low-latency anonymous Internet communications
- **Running since October 2003**
  - “Easy-to-use” client proxy
  - Freely available, can use it for anonymous browsing

Tor Circuit Setup (1)

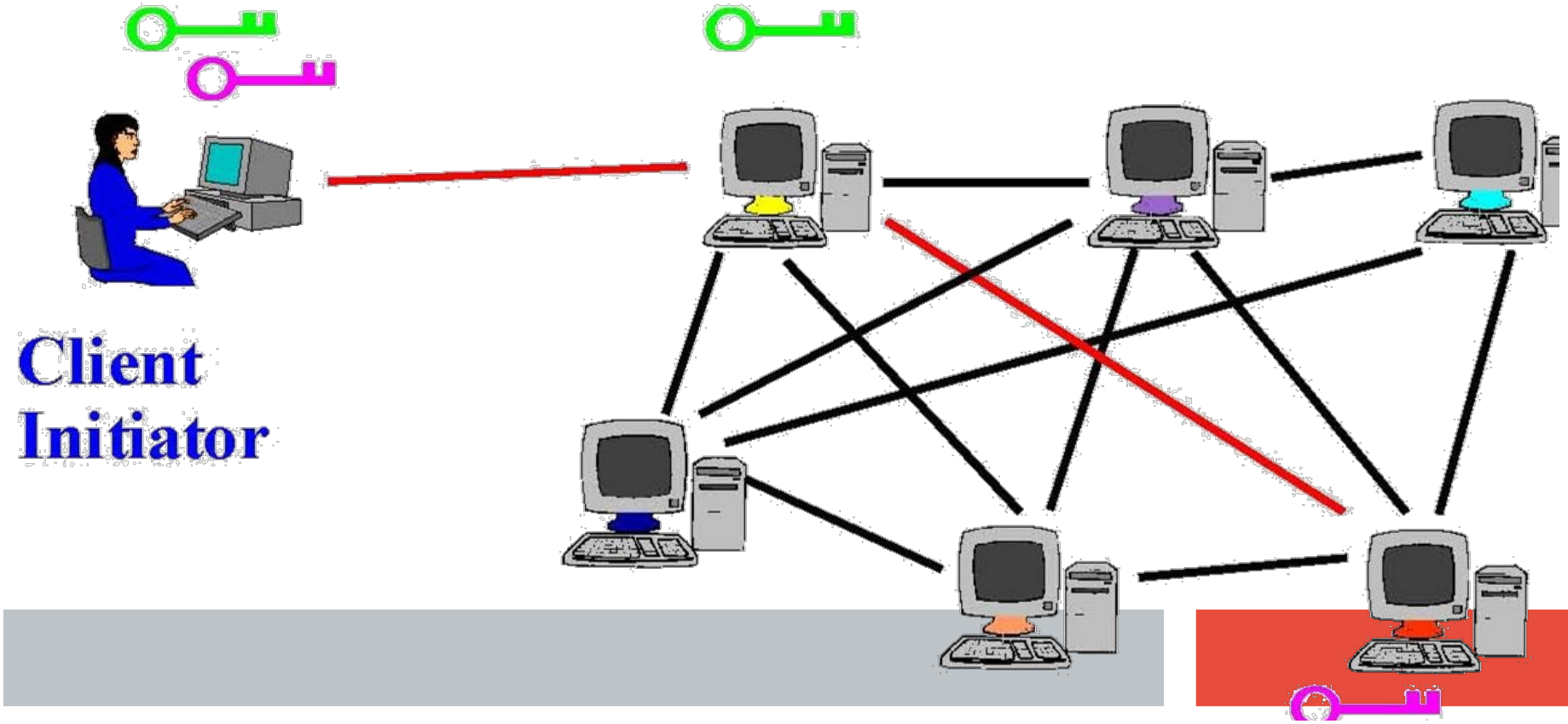
Client proxy establishes a symmetric session key and circuit with Onion Router #1



Tor Circuit Setup (2)

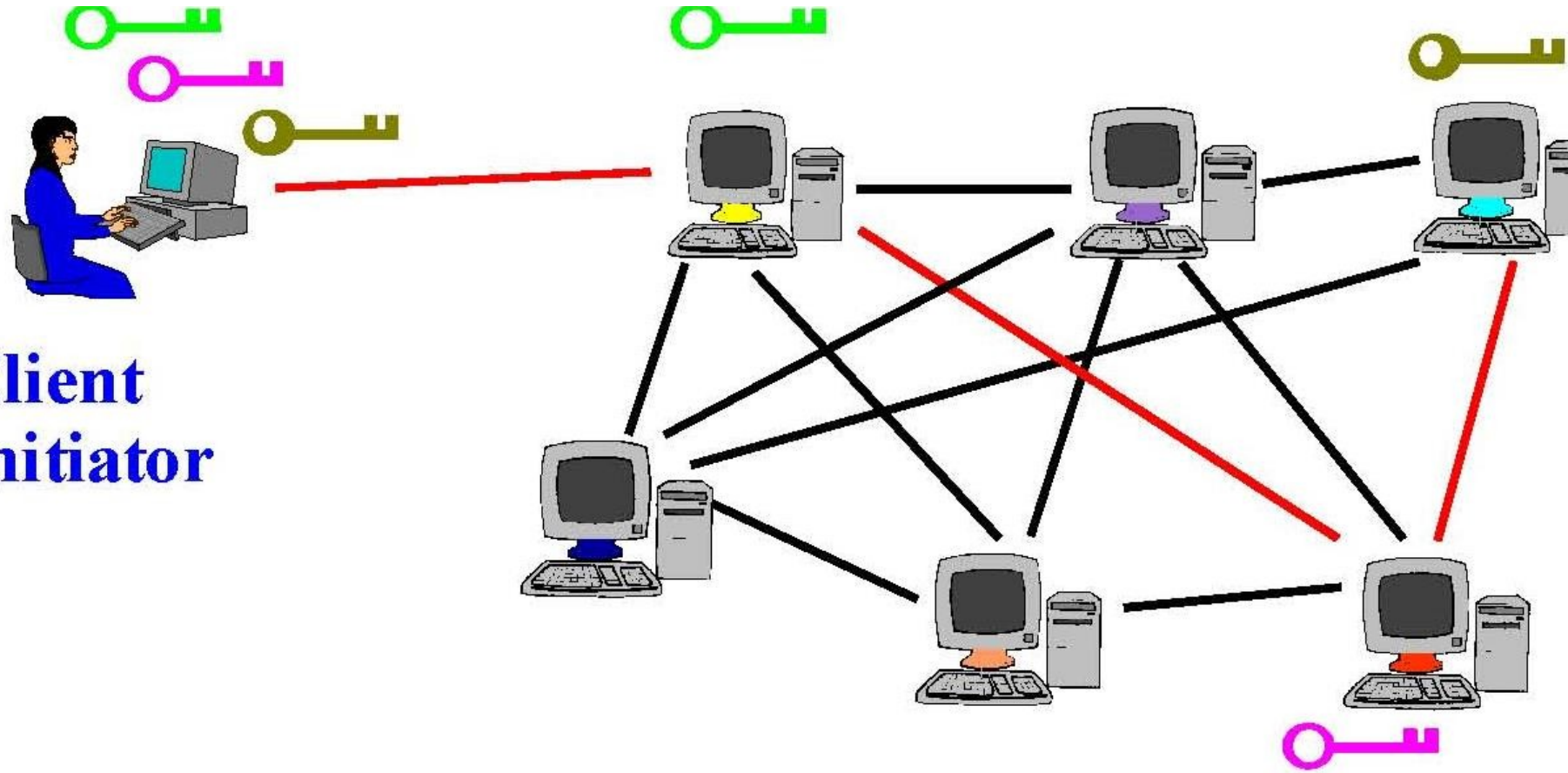
- **Client proxy extends the circuit by establishing a symmetric session key with Onion Router #2**

  Tunnel through Onion Router #1



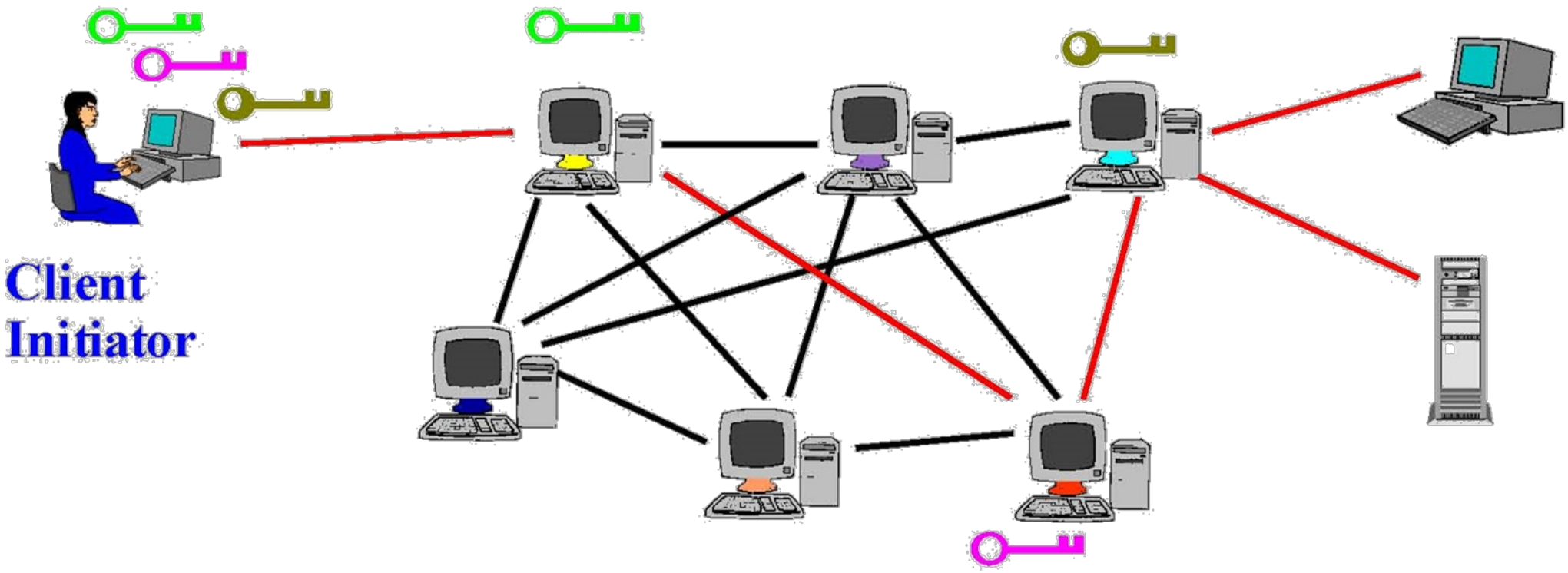
Tor Circuit Setup (3)

- **Client proxy extends the circuit by establishing a symmetric session key with Onion Router #3**
  - Tunnel through Onion Routers #1 and #2



Using a Tor Circuit

Client applications connect and communicate over the established Tor circuit.



Tor Management

- **Many applications can share one circuit**
  - Multiple TCP streams over one anonymous connection
- **Tor router doesn’t need root privileges**
  - Encourages people to set up their own routers
  - More participants = better anonymity for everyone
- **Directory servers**
  - Maintain lists of active onion routers, their locations, current public keys, etc.
  - Control how new routers join the network
    - “Sybil attack”: attacker creates a large number of routers
  - Directory servers’ keys ship with Tor code

Is Tor Perfect?

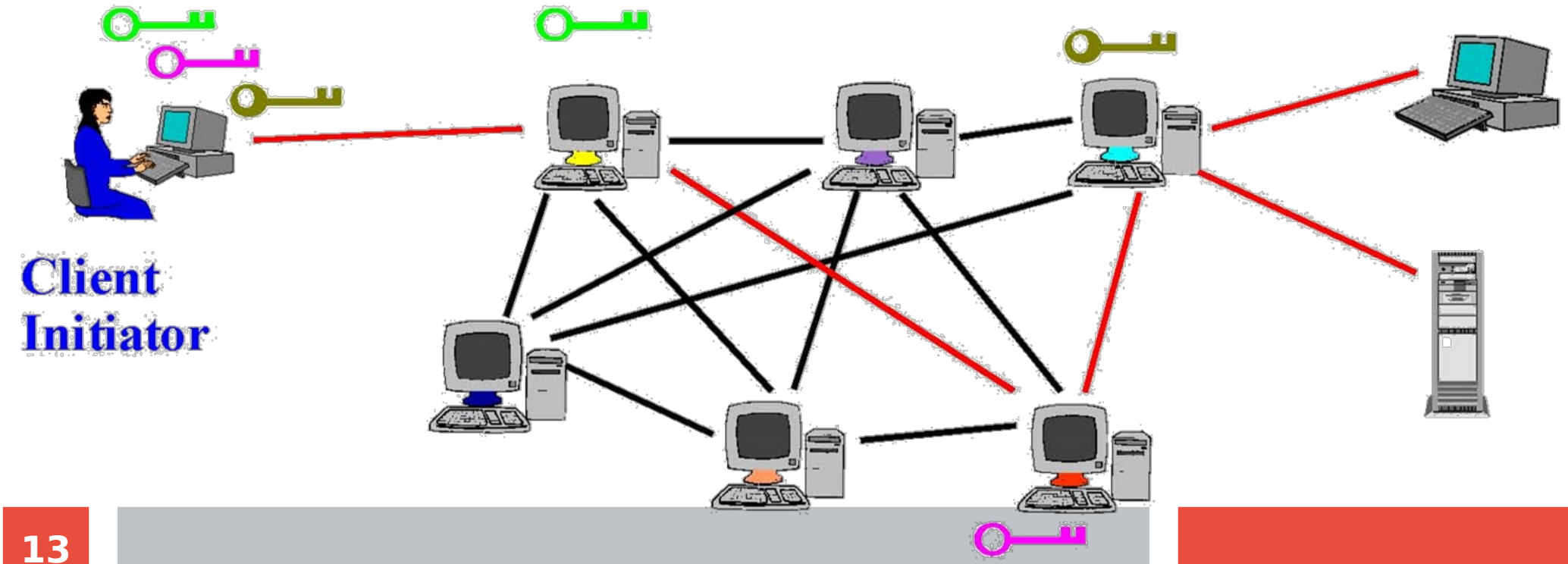
What can “go wrong” with the use of Tor?

Issues and Caveats

- **Passive traffic analysis**
  - Infer from network traffic who is talking to whom
  - To hide your traffic, must carry other people's traffic!
- **Active traffic analysis**
  - Inject packets or put a timing signature on packet flow
- **Compromise of network nodes**
  - Attacker may compromise some routers
  - And powerful adversaries may have "too many" routers (e.g., a super computer at a national lab)
- **It is not obvious which nodes have been compromised**
  - Attacker may be passively logging traffic
  - Better not to trust any individual router
  - Assume that some fraction of routers is good, don't know which

Issues and Caveats

- **Tor isn’t completely effective by itself**
  - Tracking cookies, fingerprinting, etc.
  - Exit nodes can see everything!



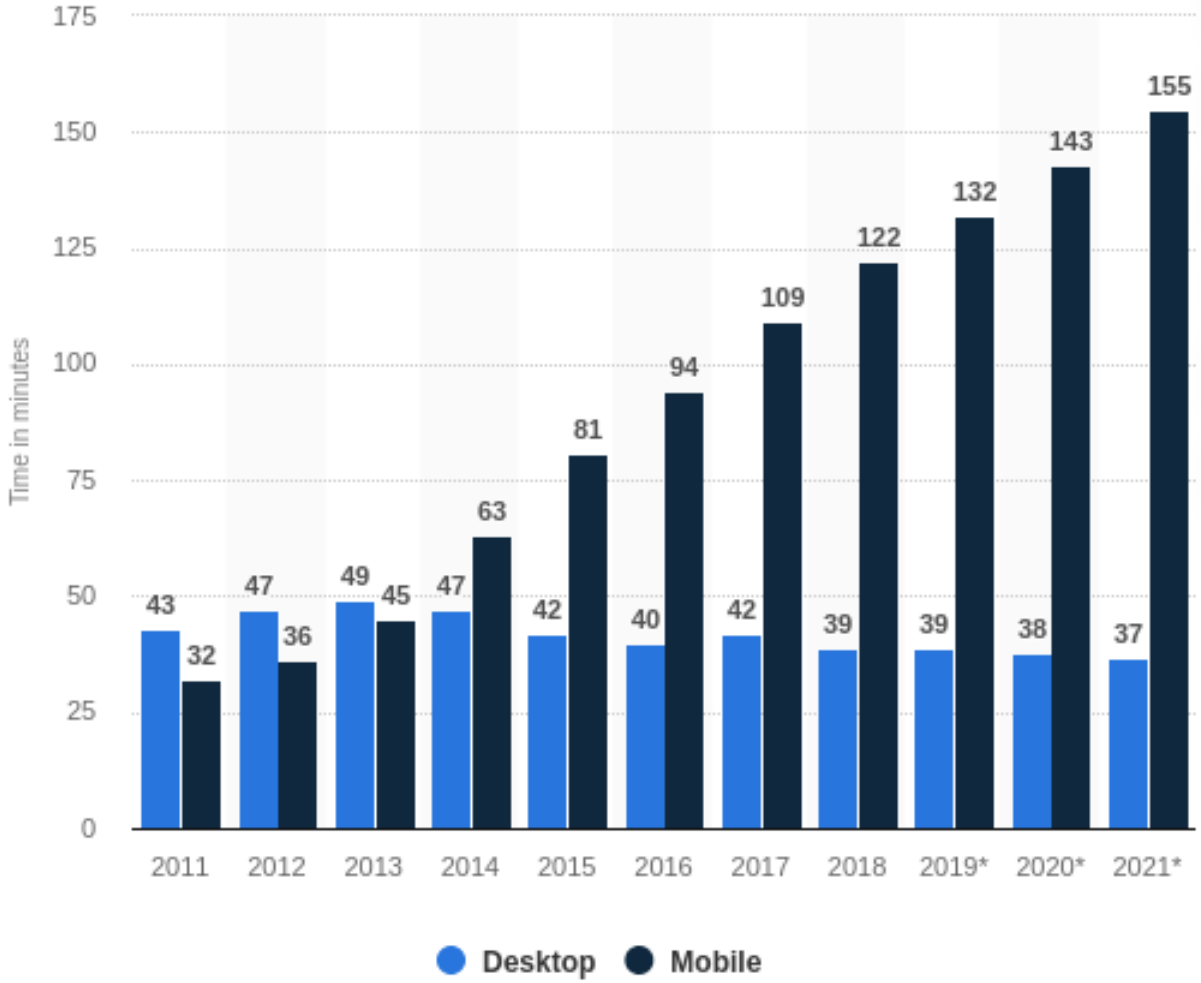
Issues and Caveats

- **The simple act of using Tor could make one a target for additional surveillance**
- **Hosting an exit node could result in illegal activity coming from your machine**

MOBILE SECURITY

Mobile is huge and still growing

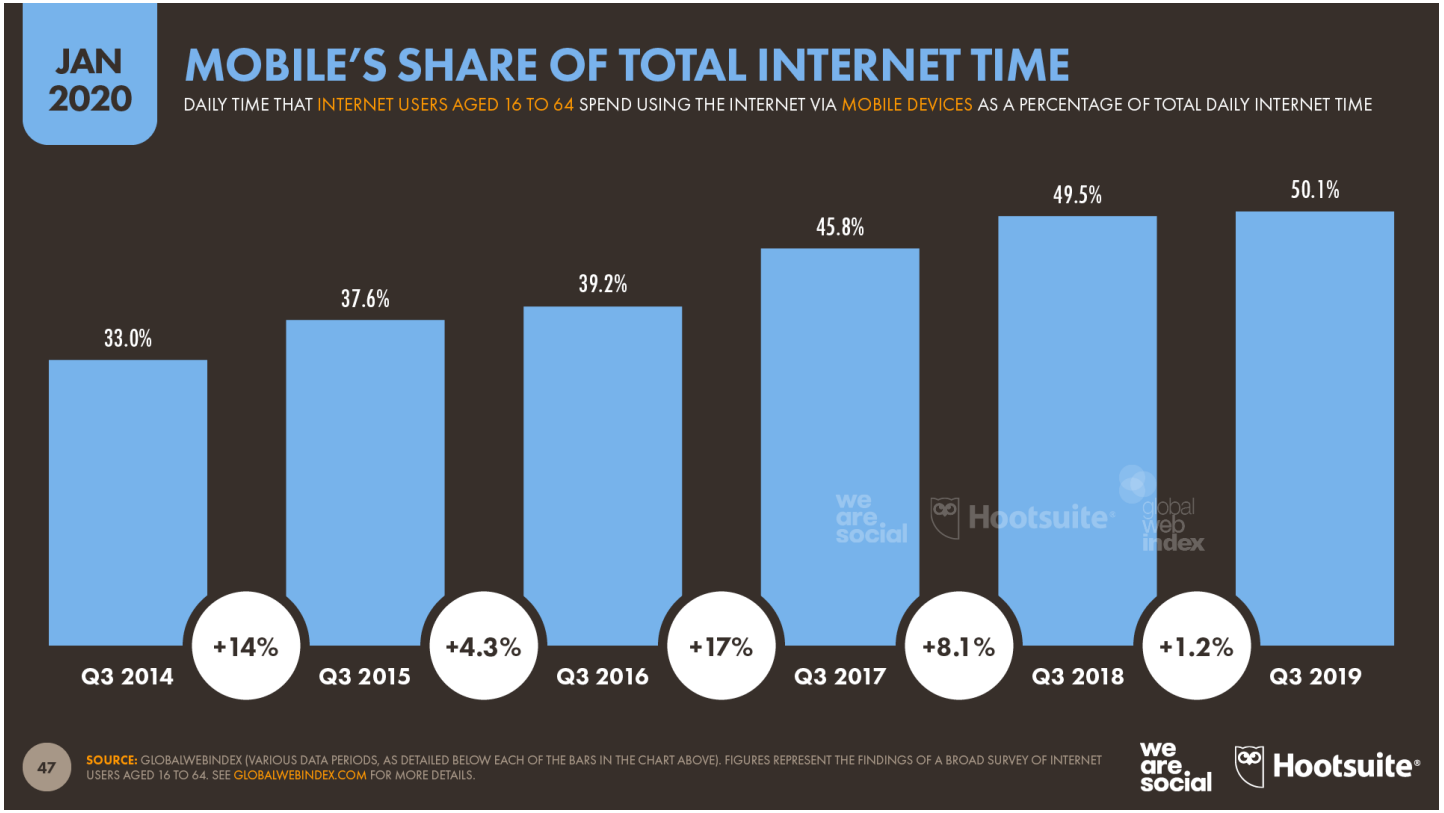
Daily time spent per user worldwide, by device:



https://www.statista.com/statistics/319732/daily-time-spent-online-device/

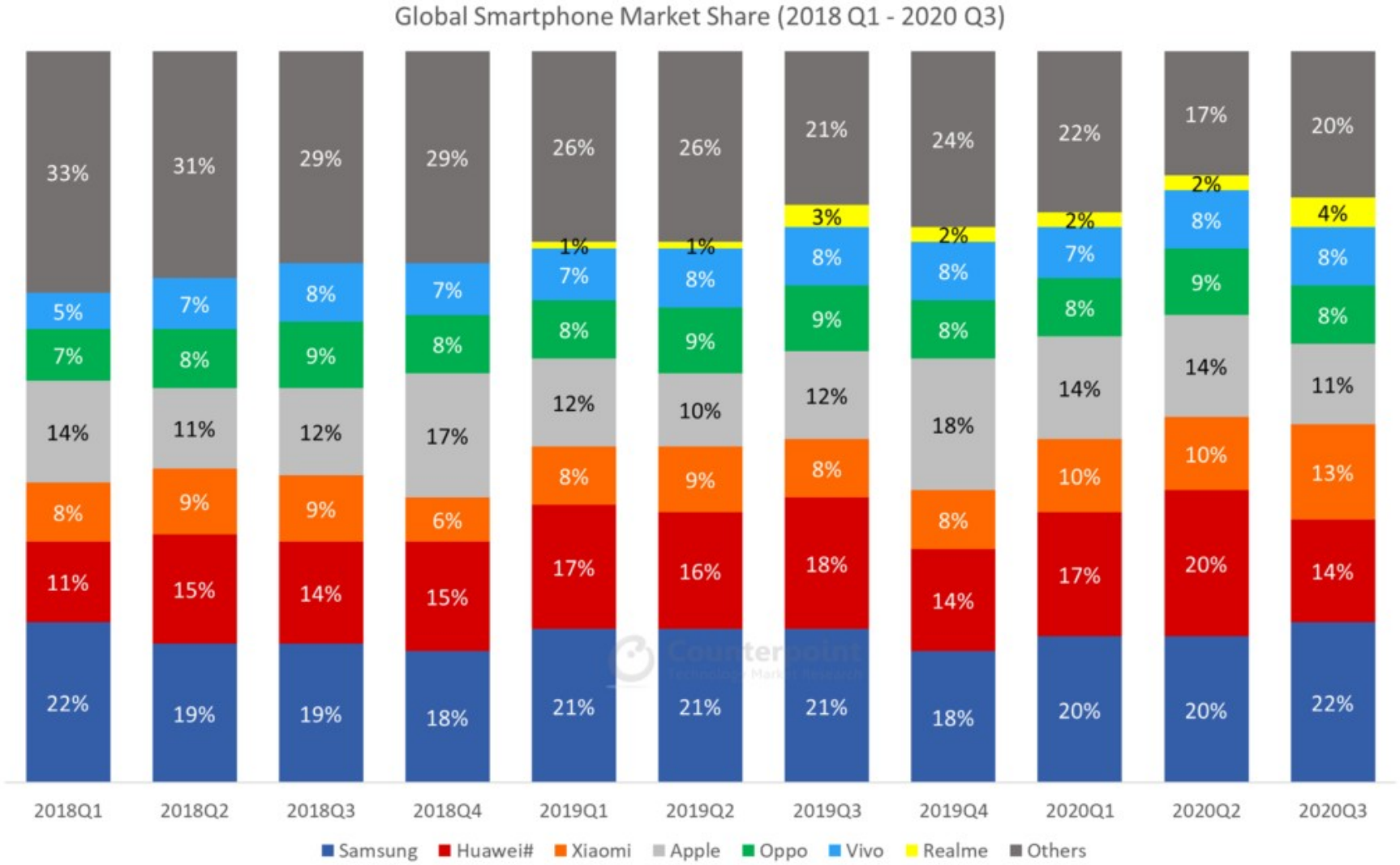
More mobile devices are online

- **2.5B active Android devices (2019)**
- **1.4B active Apple devices (2019)**



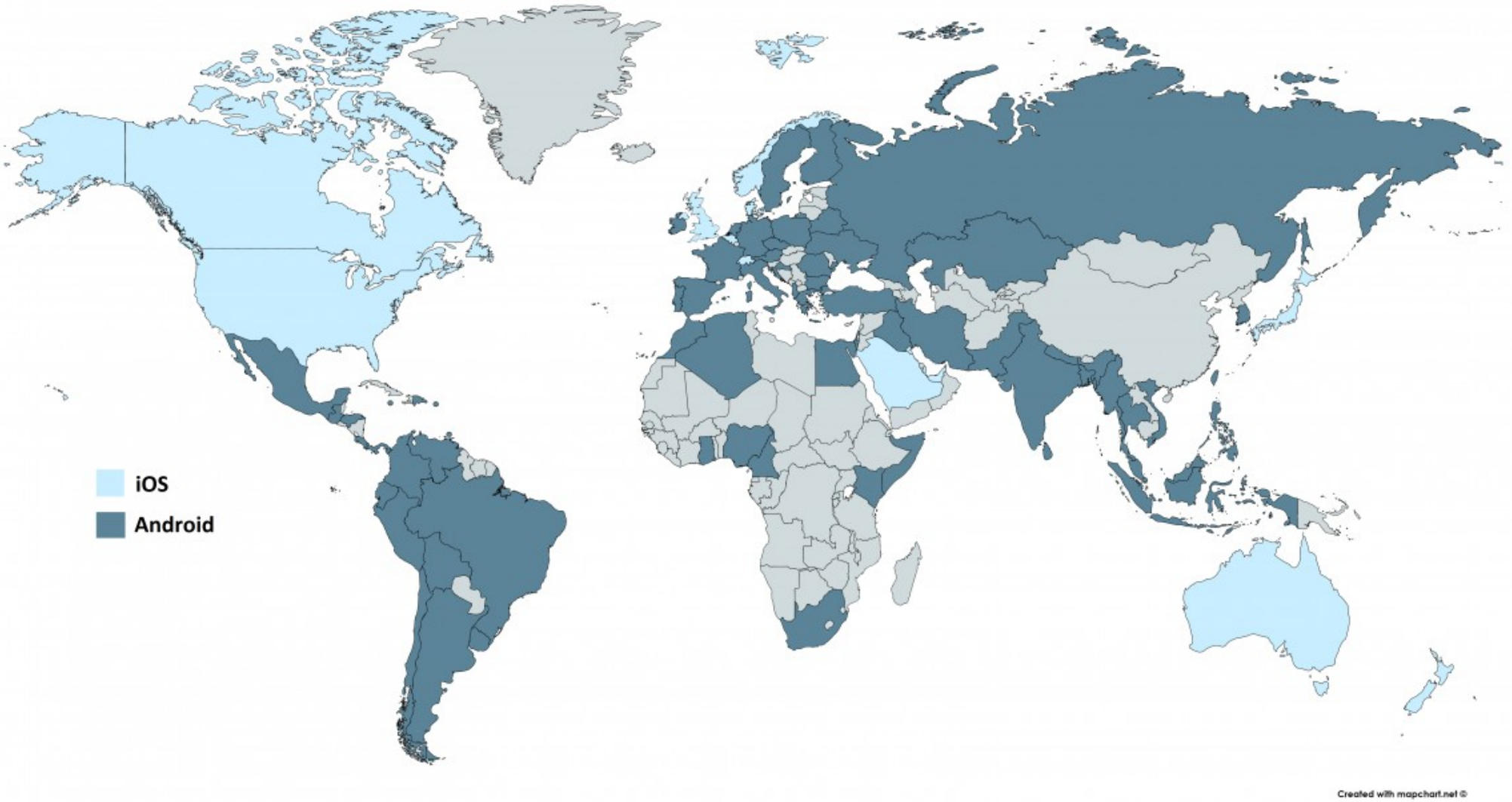
https://www.globalwebindex.com/

Mobile Market Share



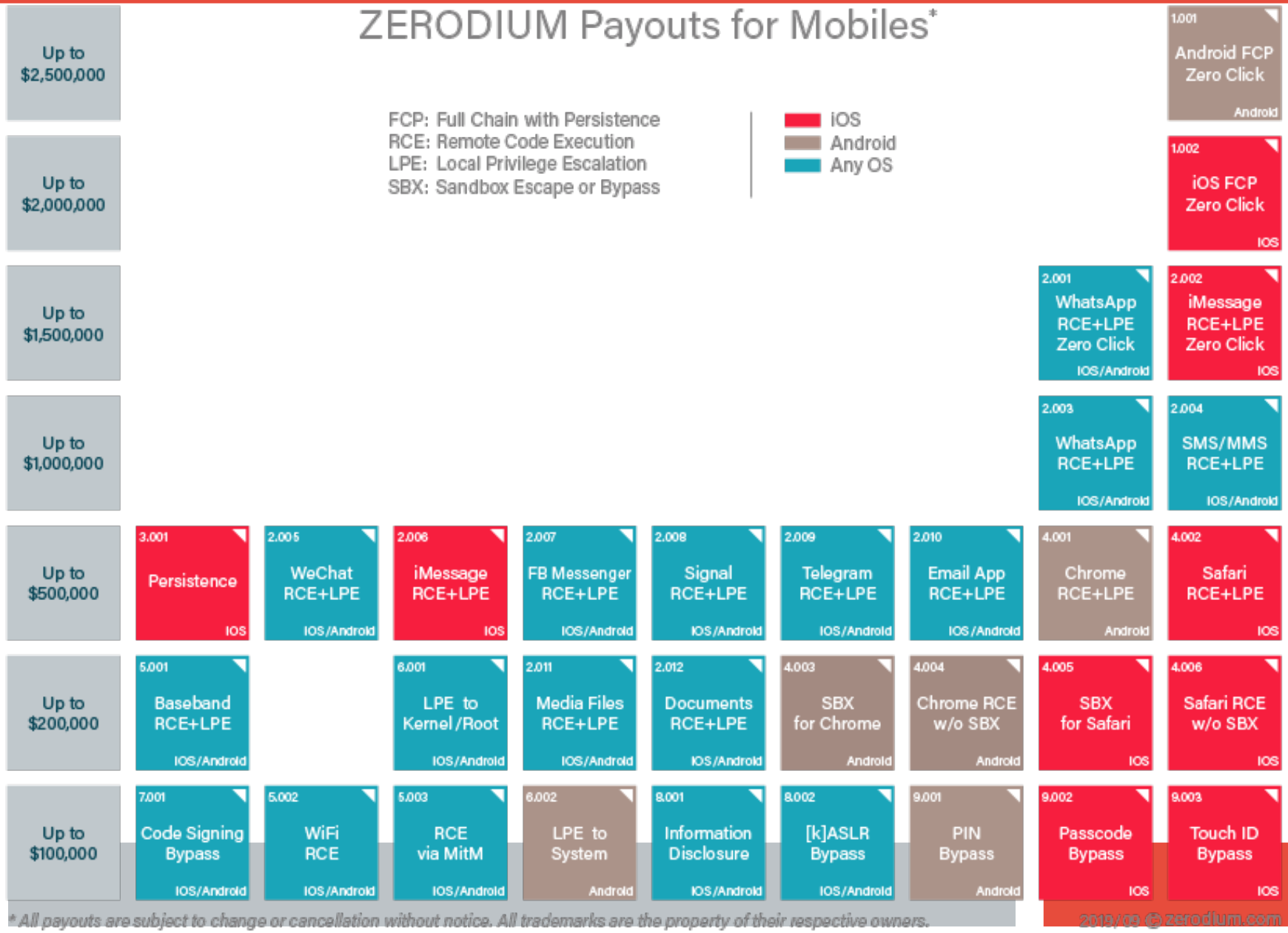
https://www.counterpointresearch.com/global-smartphone-share/

Global Mobile OS Distribution

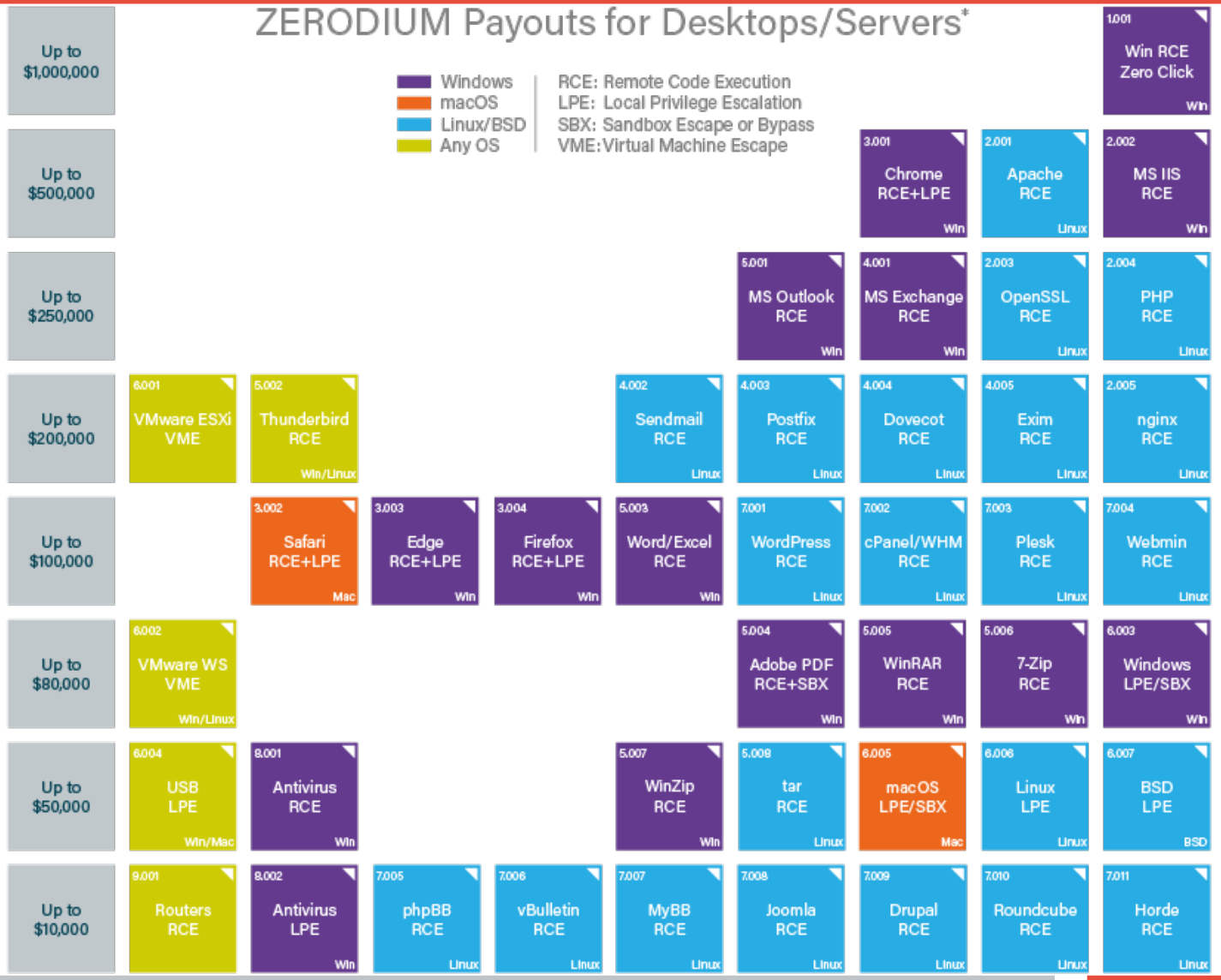


https://deviceatlas.com/blog/android-v-ios-market-share

All of this means $$$



Even more than desktop attacks



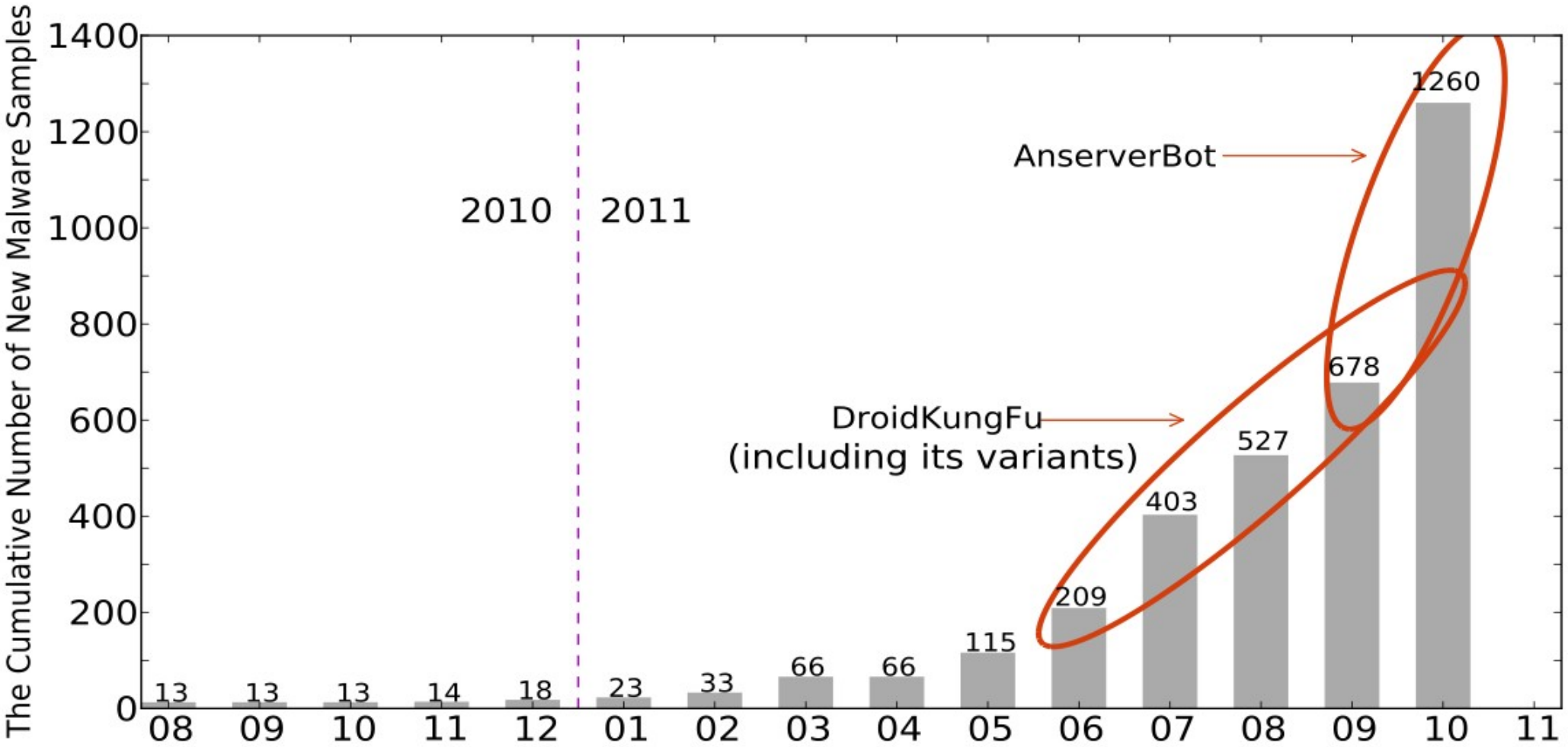
Questions: Mobile Malware

- **How might malware authors get malware onto phones?**
- **What are some goals that mobile device malware authors might have?**

Why Attack Phones?

- **Mobile Targets**
  - Sensitive data: location, health
  - Record/spoof calls (spam and phishing)
  - Steal/spoof SMS (including 2-factor SMS)
- **General Targets**
  - Valuable data: financial, passwords, contacts
  - Malware / ransomware
  - Botnet expansion

Malware in the Wild



Bring Your Own Device (BYOD)

- **Corporate Policy**
  - Allow employees to use their personal device
  - Holds company data and communications
- **Compare to fleet-based devices**
  - Not centrally managed
  - Little or no oversight
  - User has large amount of discretion
- **How do companies handle this?**
  - Secure enclave
  - Remote access tools (to wipe devices)

New Threat Model Concepts

- **Powered-off devices under complete physical control of an adversary**
  - e.g. well-resourced nation states (Border Control, Intelligence Agencies)
- **Screen locked devices under physical control of adversary**
  - e.g. thieves
- **Unlocked devices under control of different user**
  - e.g. intimate partner abuse
- **Devices in physical proximity to an adversary**
  - Capability to control radio channels, including cellular, WiFi, Bluetooth, GPS, NFC

Network Threat Model

- **Network communication under complete control of an adversary**
  - Assume first hop (e.g., router) is also malicious.
- **Passive eavesdropping and traffic analysis**
  - Tracking devices within or across networks (e.g. based on MAC address or other device network identifiers.)
- **Active manipulation of network traffic**
  - e.g. MITM on TLS.

Untrusted Code

- **Installing untrusted code**
  - Android intentionally allows installation of application code from arbitrary sources
  - Apple does not allow unsigned code to run
- **Spyware / Malware**
  - Abuses APIs supported by the OS (location, clipboard, screen reader)
  - Exploits bugs in the OS, e.g. kernel, drivers, or system services
- **UI tricks**
  - Mimic system or other app user interfaces to confuse users
  - Draw over other apps to steal input
  - Read content from system or other application user interfaces (screen-scraping)
  - Injecting input events into system or other app user interfaces

Unlocking a Device

- **Traditional**
  - PIN or pattern
  - Some apps have a secondary PIN
- **Bio-metric**
  - FaceID and TouchID

PIN and Code Issues

- **Remember Lecture 3**
- **Smudge attacks [Aviv et al., 2010]**
  - Entering pattern leaves smudge that can be detected with proper lighting
  - Smudge survives incidental contact with clothing
- **Another problem: entropy**
  - People choose simple patterns – few strokes
- **So how does a 4-6 digit PIN become secure?**
  - Use modern hashing: bcrypt, scrypt, or pbkdf2

iPhone Password Hashing

- **Password hashing where 4-6 digits takes a very long time to crack**
  - Even if the device is physically compromised...
- **Additional Constraints:**
  - Lots of computation uses up battery (limited resource)!
  - Physical access allows copying secret off and cracking remotely

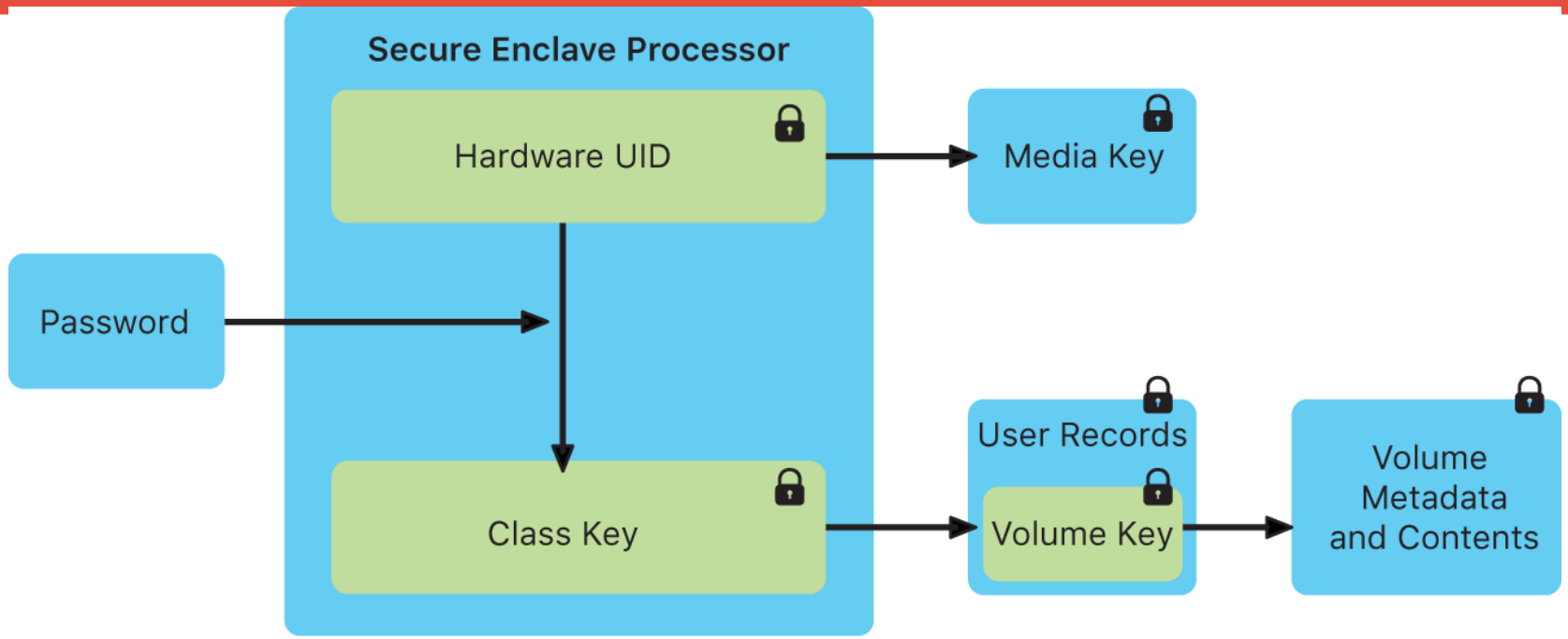
Secure Enclave

- **Additional secure processor known as the secure enclave**
  - Memory is inaccessible to normal OS
  - Utilizes a secure boot process that ensures its software is signed
- **Each secure enclave has an AES key**
  - Burned in at manufacture
- **Processor has instructions that allow encrypting and decrypting content using that key**
  - Key itself is never accessible (even via JTAG or debugging)

iPhone Unlocking

- **User passcode is mixed with AES key fused into secure enclave (known as UID).**
  - key = Encrypt(UID, passcode).
- **Key to decrypt the device can only be derived on the secure enclave on a specific phone**
  - Not possible to take offline and brute force

iPhone Unlocking Key



- **Preventing repeated passcode guesses**
  - Mix passcode with the device's UID many times (multi-round system)
  - approximately 80ms per password guess
  - Encrypt(UID, Encrypt(UID, Encrypt(UID, passcode)...))

iPhone Unlock Time Estimate

- **At 80ms per password check…**
  - 5.5 years to try all 6 digits pins
- **5 failed attempts**
  - 1min delay
- **9 failures**
  - 1 hour delay
- **>10 failed attempts**
  - erase phone
- **All enforced by firmware (ring -1) on the secure enclave itself cannot be changed by any malware that controls OS**

FBI–Apple Encryption Dispute

- **After the San Bernardino shooting in 2016, FBI tried to compel Apple to “unlock” iPhone**
- **What were they specifically requesting?**
  - Not possible to make password guessing any faster
  - Dependent on performance of burned-in AES key

FBI–Apple Encryption Dispute

- **Remember…**
  - 5 failed attempts, 1min delay; 9 failures, 1 hour delay; >10 failed attempts, erase phone
- **Security policy is software**
- **Which can be updated by Apple, not managed in hardware**

Technical Details

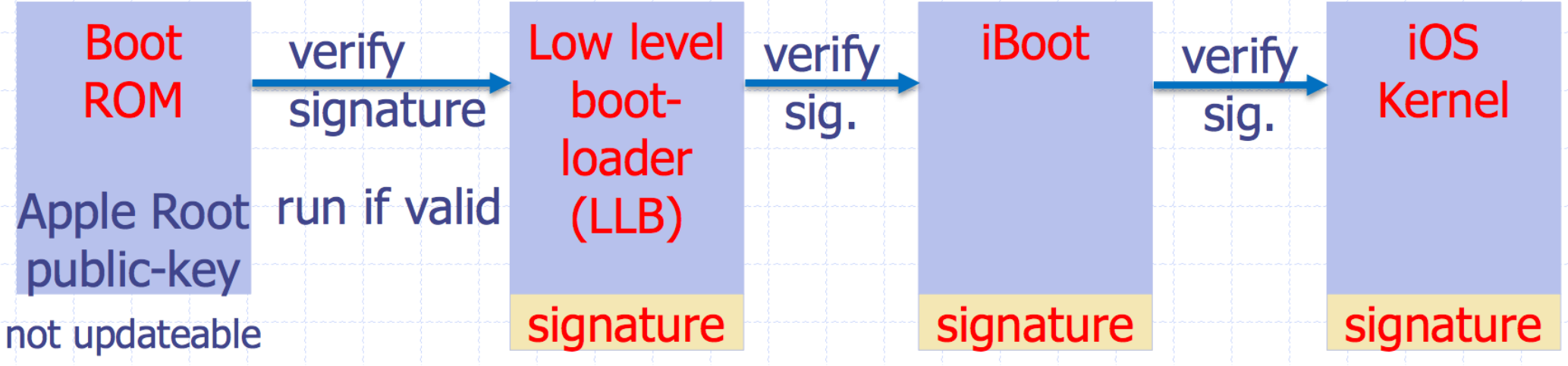
- **The court order wanted a custom version of a secure enclave firmware that would…**
  - "it will bypass or disable the auto-erase function whether or not it has been enabled" (this user-configurable feature of iOS 8 automatically deletes keys needed to read encrypted data after ten consecutive incorrect attempts)
  - "it will enable the FBI to submit passcodes to the SUBJECT DEVICE for testing electronically via the physical device port, Bluetooth, Wi-Fi, or other protocol"
  - "it will ensure that when the FBI submits passcodes to the SUBJECT DEVICE, software running on the device will not purposefully introduce any additional delay between passcode attempts beyond what is incurred by Apple hardware”

What happened?

- **Apple planned to fight the order**
  - “The United States government has demanded that Apple take an unprecedented step which threatens the security of our customers. We oppose this order, which has implications far beyond the legal case at hand. This moment calls for public discussion, and we want our customers and people around the country to understand what is at stake.”
- **One day before hearing, FBI dropped the request**
  - A third party had demonstrated a possible way to unlock the iPhone in question.
- **No precedent set re all writs act**

iOS Secure Boot Chain

- **Why couldn't the FBI just upload their own firmware onto the secure enclave?**
  - At boot, iPhone executes code from read-only memory (Boot ROM)
  - This immutable ROM code is the **hardware root of trust**
    - Laid down during chip fabrication, and implicitly trusted
  - The Boot ROM code contains the Apple Root CA public key
    - Used to verify that the bootloader is signed by Apple
  - Chain of trust where each step ensures that the next is signed

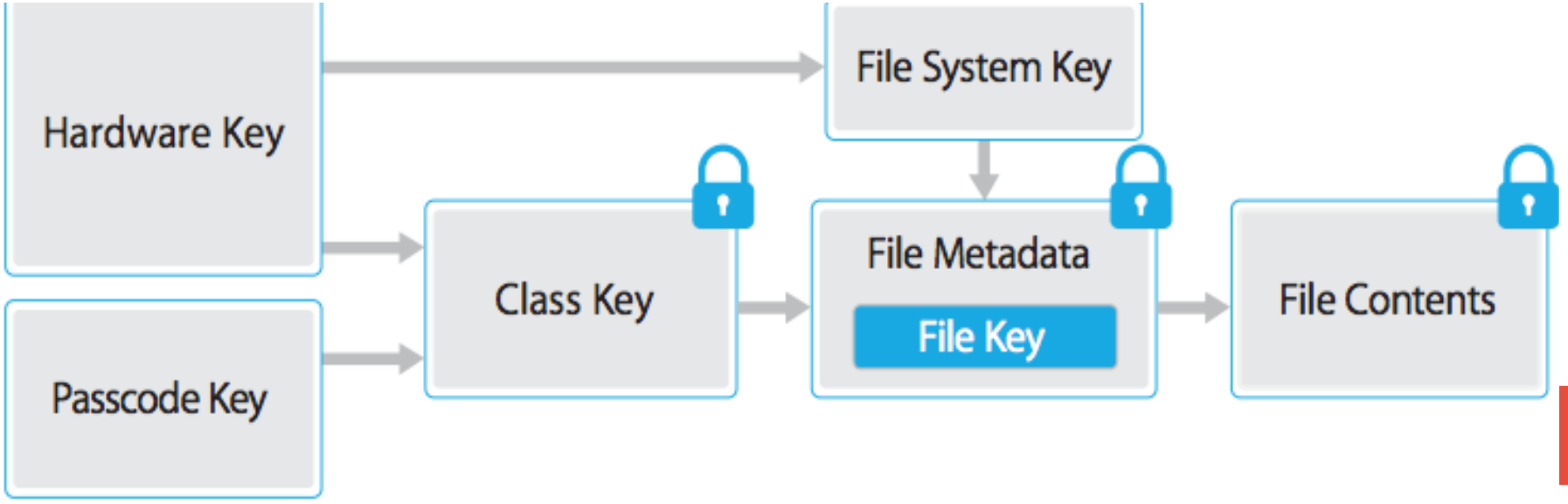


Software Update

- **Prevent devices from being downgraded to older versions that lack the security updates**
  - iOS uses System Software Authorization
- **Device connects to Apple with cryptographic descriptors of each component update**
  - e.g., boot loader, kernel, and OS image, current versions, a random nonce, and device specific Exclusive Chip ID (ECID).
- **Apple signs device-personalized message allowing update, which boot loader verifies.**

FaceID and TouchID

- **Hierarchy of encryption keys**
- **Application files written to Flash are encrypted:**
  - Per-file key: encrypts all file contents (AES-XTS)
  - Class key: encrypts per-file key (ciphertext stored in metadata)
  - File-system key: encrypts file metadata



FaceID and TouchID

- **Files are encrypted through a hierarchy of encryption keys**
- **No TouchID or FaceID**
  - Whenever the device is locked or powered off class encryption keys are erased from memory
- **When TouchID/FaceID is enabled**
  - Class keys are kept and hardware sensor sends fingerprint image to secure enclave. All ML/analysis is performed within the secure enclave.

More Information

- **iOS Security Guide**
  - https://support.apple.com/guide/security/welcome/web
  - Notes on Hardware, OS and Services Security

OS Fragmentation

- **Many different variants of Android (unlike iOS)**
  - Motorola, HTC, Samsung, ...
- **Less secure ecosystem**
  - Inconsistent or incorrect implementations
  - Slow to propagate kernel (if at all)
  - Updates to new versions only
  - Vendors must patch hardware-specific vulns
- **Where's Android 11?**
  - Google ended support for version dashboard in April 2020

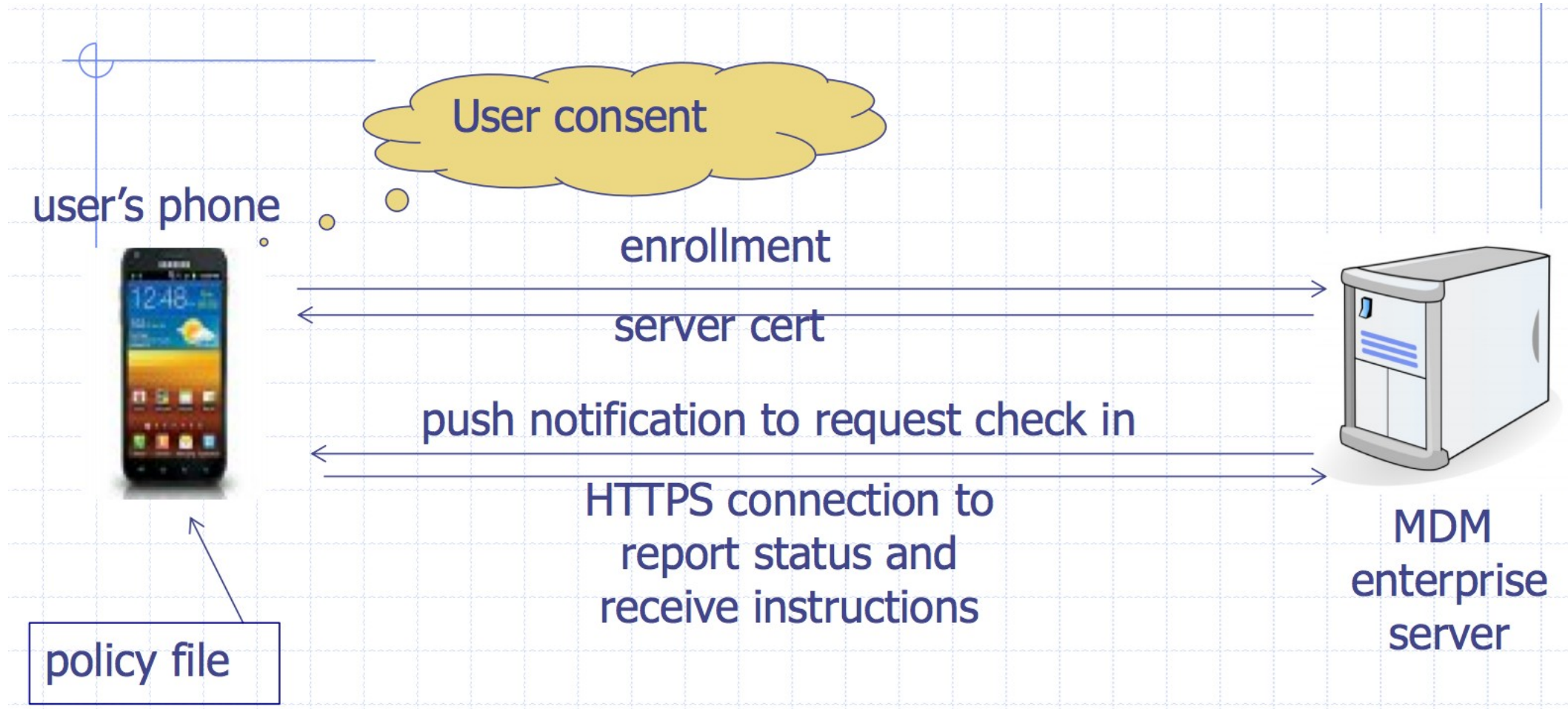
VERSION		API LEVEL	DI
4.0	Ice Cream Sandwich	15	
4.1	Jelly Bean	16	
4.2	Jelly Bean	17	
4.3	Jelly Bean	18	
4.4	KitKat	19	
5.0	Lollipop	21	
5.1	Lollipop	22	
6.0	Marshmallow	23	
7.0	Nougat	24	
7.1	Nougat	25	
8.0	Oreo	26	
8.1	Oreo	27	
9.0	Pie	28	
10.	Android 10	29	

https://www.androidauthority.com/android-distribution-chart-1105405/

Mobile Device Management

- **Manage mobile devices across organization**
- **Central server and client-side software.**
  - Now integrated into Mobile OS
- **Functions:**
  - Diagnostics, repair, and update
  - Backup and restore
  - Policy enforcement (e.g. only allowed apps)
  - Remote lock and wipe
  - GPS Tracking

Sample MDM Enrollment



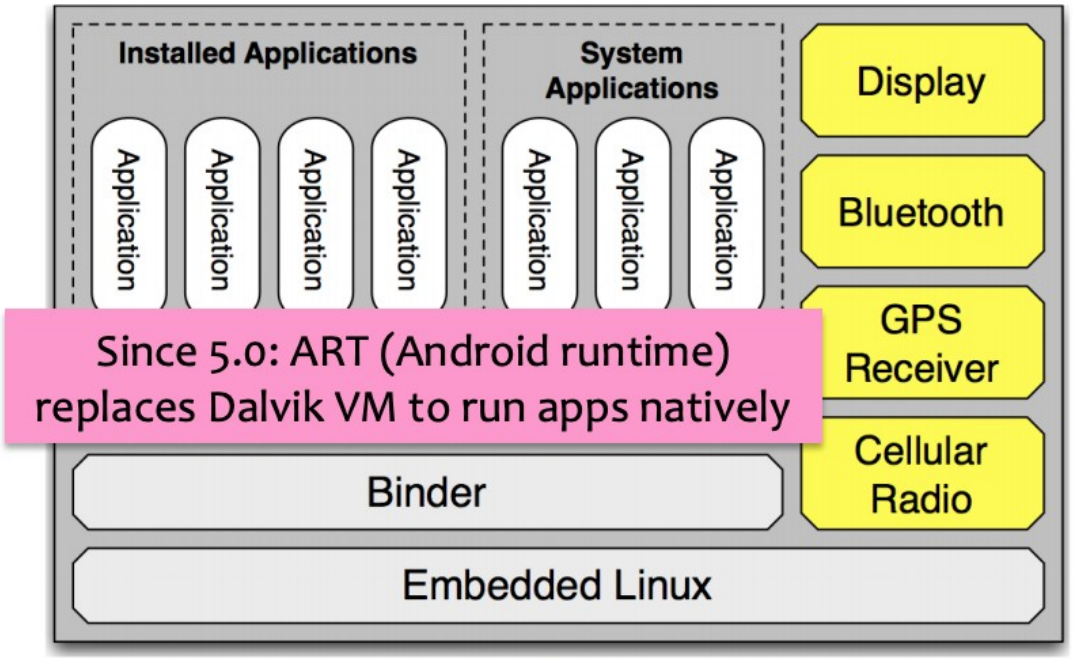
MOBILE MALWARE

Different Landscape

- **Applications are isolated**
  - Each runs in a separate execution context
  - No default access to file system, devices, etc.
  - Different than traditional OSes where multiple applications run with the same user permissions!
- **Applications are installed via App Store (and malware spreads)**
  - Market is vendor controlled (Android allows side-loading)
  - User approval of permissions

Android Isolation

- **Based on Linux with sandboxes (SE Linux)**
  - Apps run as separate UIDs, in separate processes.
  - Memory corruption errors only lead to arbitrary code execution in application,not complete system compromise!
- **Can still escape sandbox, but must compromise Linux kernel**



What is “Rooting”?

- **Allows user to run applications with root privileges**
  - e.g., modify/delete system files and app, CPU, network management
- **Done by exploiting vulnerability in firmware to install a custom OS or firmware image**
- **Double-edged sword… lots of malware only affects rooted devices**

Examples of Malware

- **DroidDream (Android vulenrability)**
  - Over 58 apps uploaded to Google app market
  - Conducts data theft; send credentials to attackers
- **Zitmo (Malicious application)**
  - Poses as mobile banking application
  - Captures info from SMS – steal banking 2FA codes
  - Works with Zeus botnet
- **Ikee (Attacks rooted iPhones)**
  - Worm capabilities (targeted default ssh password)
  - Worked only on jailbroken phones with ssh installed

Legitimate Apps may abuse permissions

Top Mobile Apps Overwhelmingly Leak Private Data: Study

By Robert Lemos | Posted 2013-07-31 Email Print

Hornyack et al.: 43 of 110 Android applications sent location or phone ID to third-party advertising/analytics servers.

paid apps
pplication-
isk more often
more likely to
applications,
ons

Android flashlight app tracks users via GPS, FTC says hold on

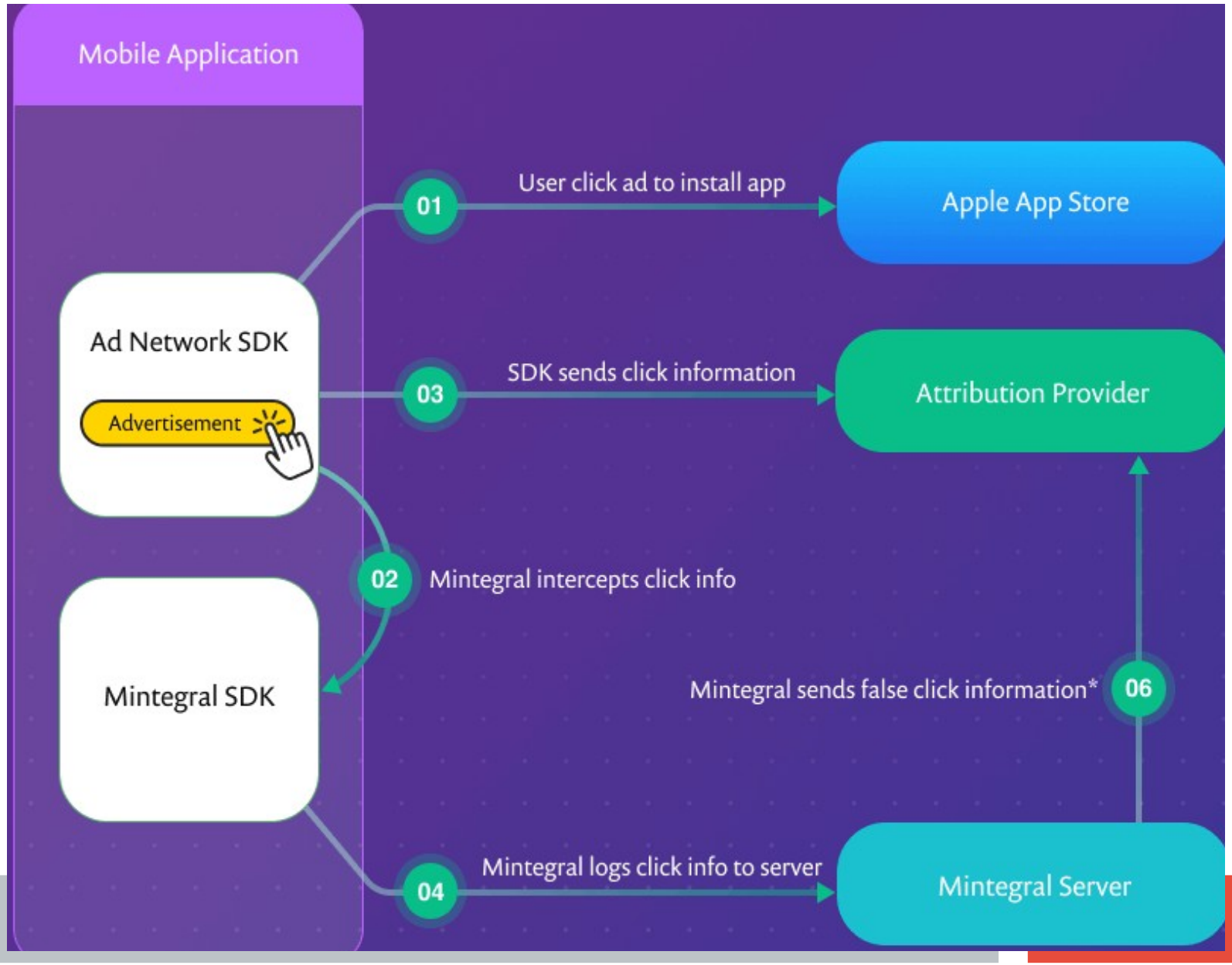
By Michael Kassner in IT Security, December 11, 2013, 9:49 PM PST

Malicious SDKs

- **Mintegral SDK**
  - Tool for app developers to build monetized ad-based marketing
- **SDK injects code into standard iOS functions**
  - Execute when the application opens a URL, including app store links, from within the app.
- **SDK Contains anti-debug protections**
  - Attempts to determine if the phone was rooted
  - Or if any type of debugger or proxy tools are in use
  - If it finds evidence that it is being watched, the SDK modifies its behavior
- **Masks malicious behaviors**
  - Help the SDK pass through Apple’s app review process without being detected

https://snyk.io/blog/sourmint-malicious-code-ad-fraud-and-data-leak-in-ios/

How Mintegral Works



Ad Fraud (1)

- **Hijacks user clicks on ads within the app**
- **Advertisements presented by ad networks**
- **Advertisers pay the ad networks to display their ads**
  - Charged on the performance of the ad (cost per click)
  - Developers get a portion of the profits the ad network makes from the advertisers
- **Mintegral intercepts all of the ad clicks (and other URL clicks as well) within the application.**
- **Uses this to forge click notifications**

Ad Fraud (2)

- **Recall the privacy lecture**
  - Advertisers need to attribute the ad sales
- **Forged notifications make it appear that the ad click came through Mintegral**
  - Even though a competing ad network served the ad
- **Mintegral steals ad revenue from competing ad networks**
  - Claiming attribution for clicks that did not occur on a Mintegral presented ad
  - Mintegral seems to perform better than the other ad networks
  - Bias toward selecting Mintegral over competing ad networks in the future

Challenges with Isolated Apps

- **So mobile platforms isolate applications for security, but....**
  - Permissions: How can applications access sensitive resources?
  - Communication: How can applications communicate with each other?

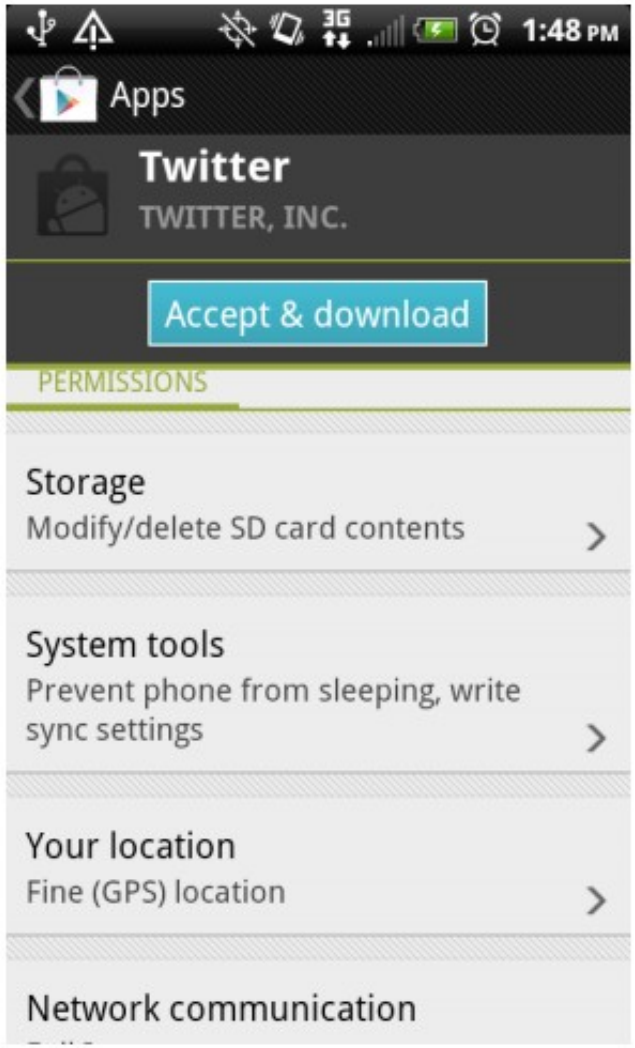
Permission Granting Problem

- **Smartphones (and other modern OSes) try to prevent such attacks by limiting applications' default access to:**
  - System Resources (clipboard, file system)
  - Devices (e.g., camera, GPS, phone, ...)
- **How should operating system grant permissions to applications?**
  - Standard approach: Ask the user.

State of the Art

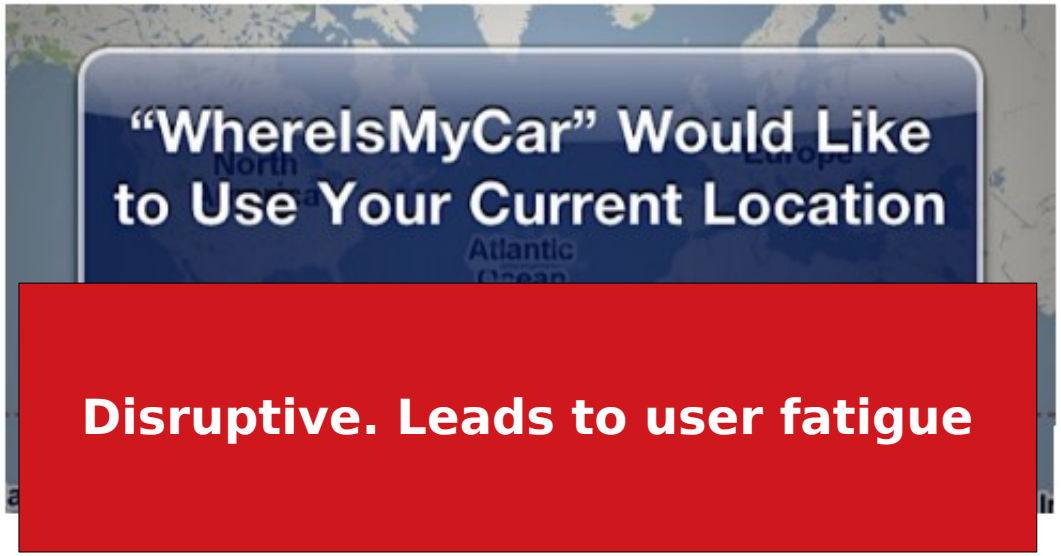
Prompts (time-of-use)

Manifests (install-time)



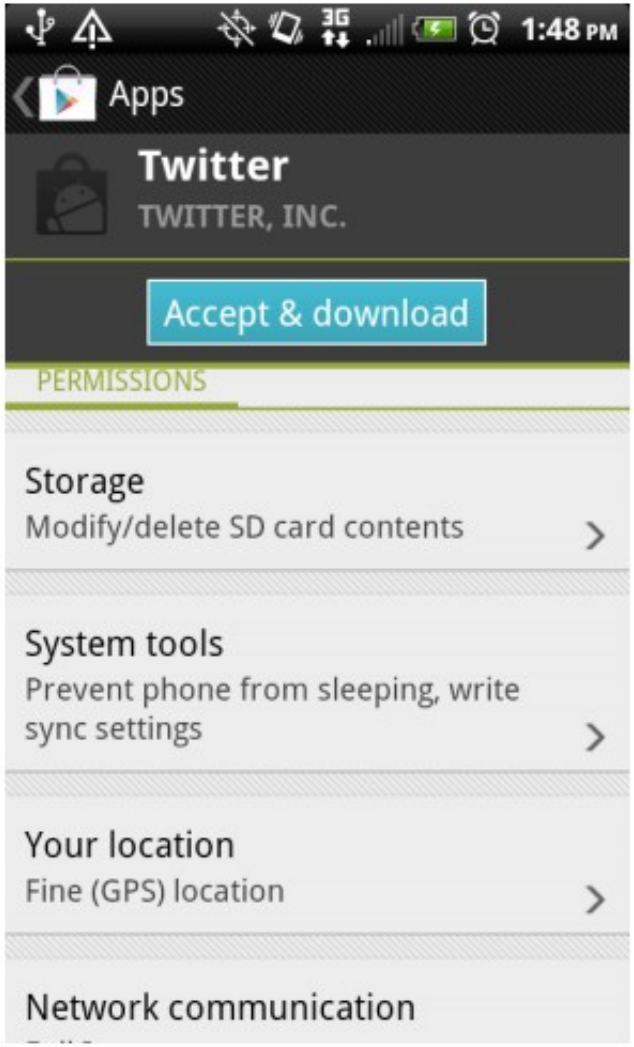
State of the Art

Prompts (time-of-use)



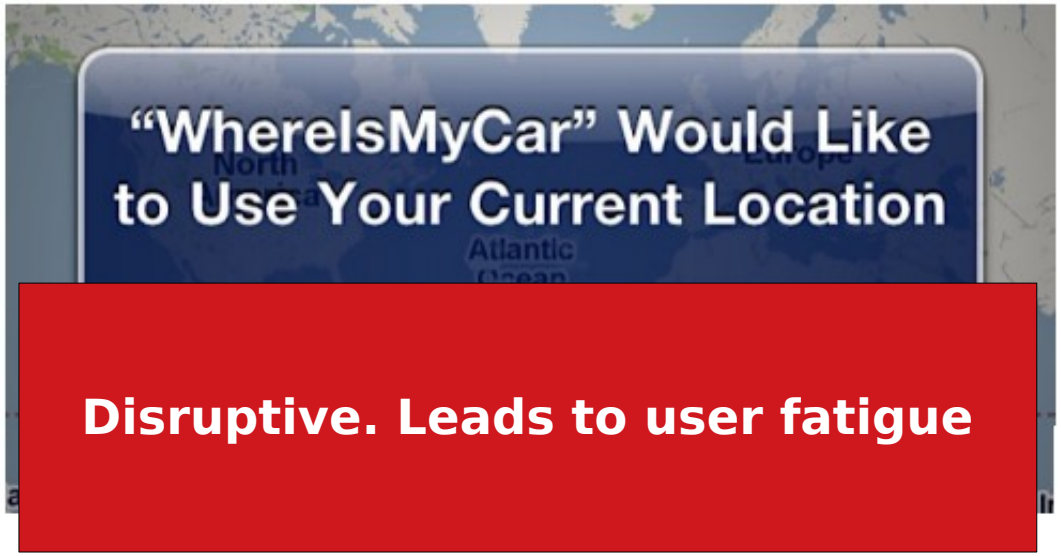
Disruptive. Leads to user fatigue

Manifests (install-time)



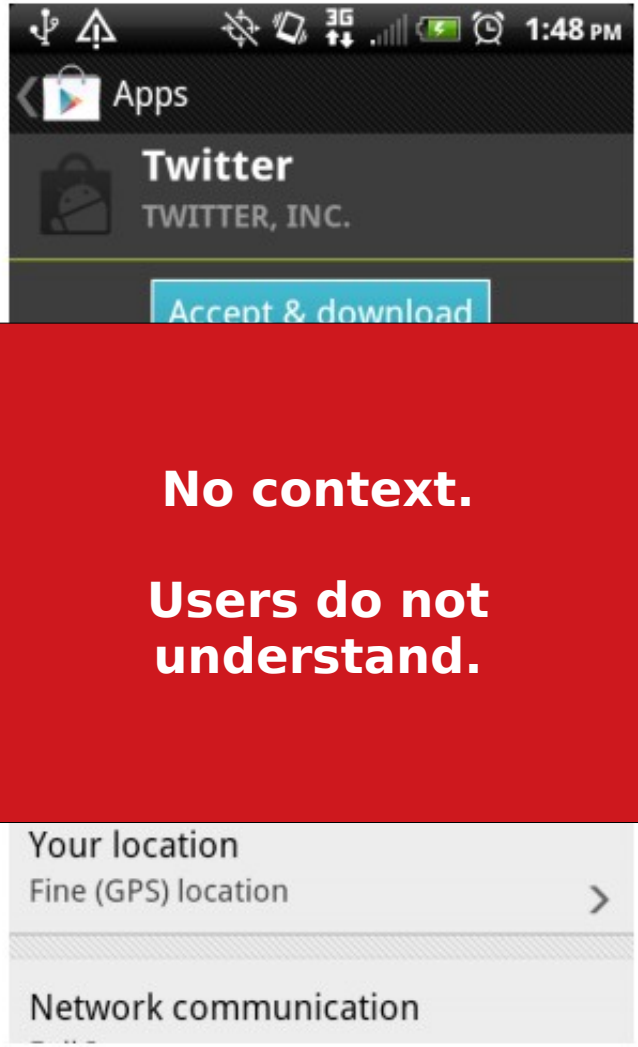
State of the Art

Prompts (time-of-use)



Disruptive. Leads to user fatigue

Manifests (install-time)

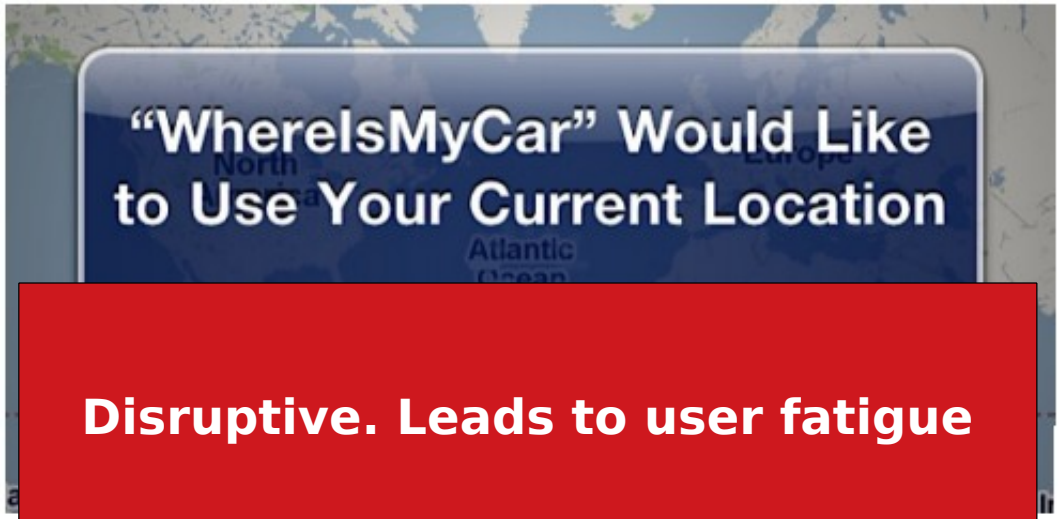


No context.

Users do not understand.

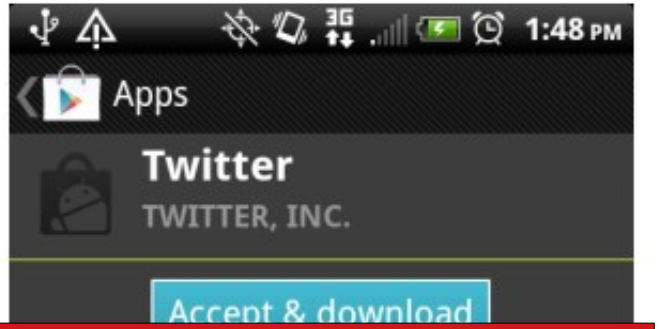
State of the Art

Prompts (time-of-use)



Disruptive. Leads to user fatigue

Manifests (install-time)

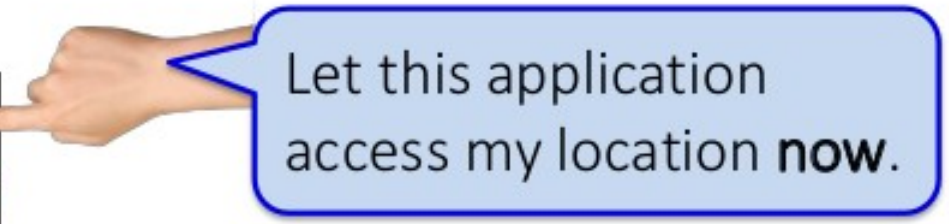


No context.

Users do not understand.

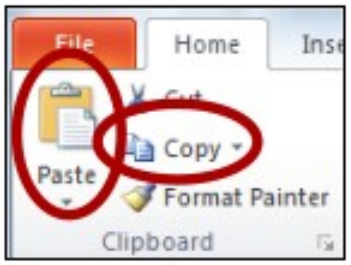
Overly permissive:
Once granted, apps can misuse or abuse

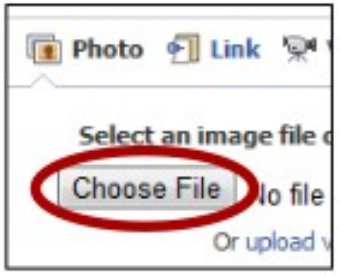
User-Driven Access Control



Insight:

A user's natural UI actions within an application implicitly carry permission-granting semantics.

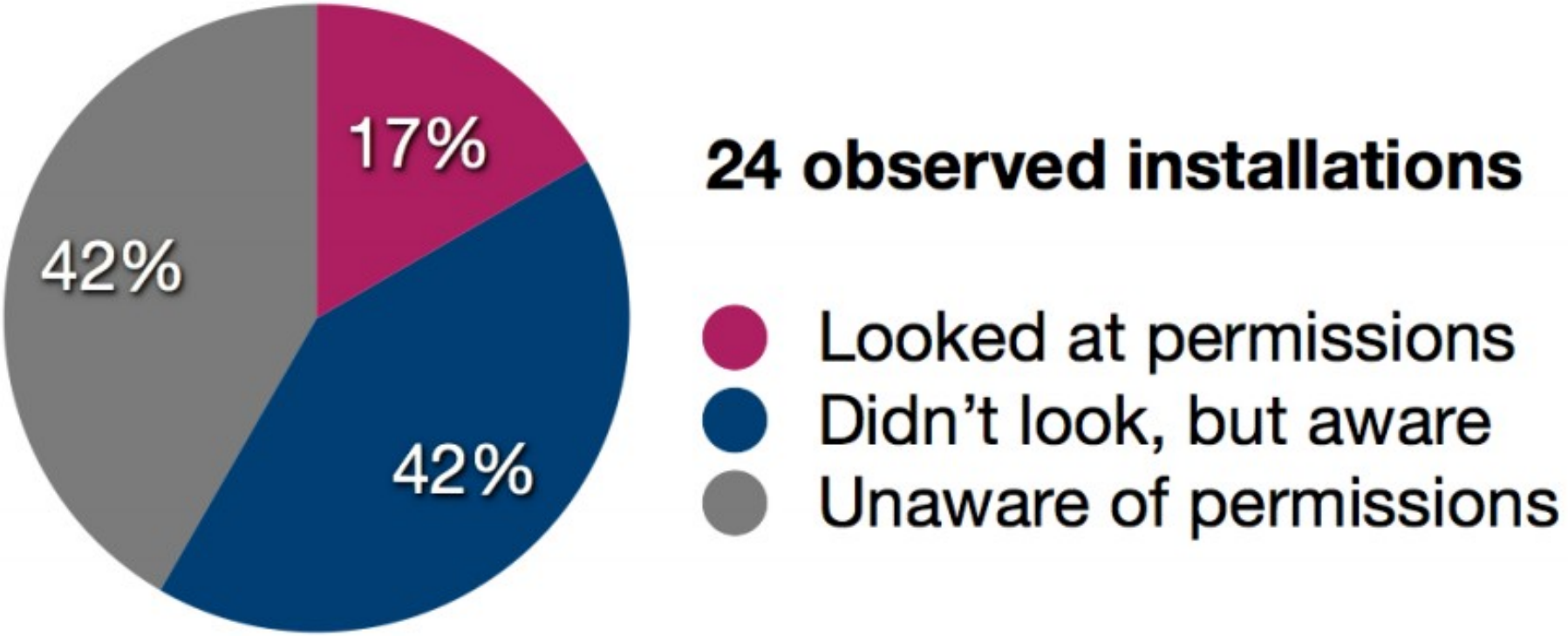




[Roesner et al]

Are Permission Manifests Usable? (Felt et al. 2011)

Do users pay attention to permissions?



... but 88% of users looked at reviews.

https://people.eecs.berkeley.edu/~dawnsong/papers/2011%20Android%20permissions%20demystified.pdf

Are Permission Manifests Usable?

Do users understand the warnings?

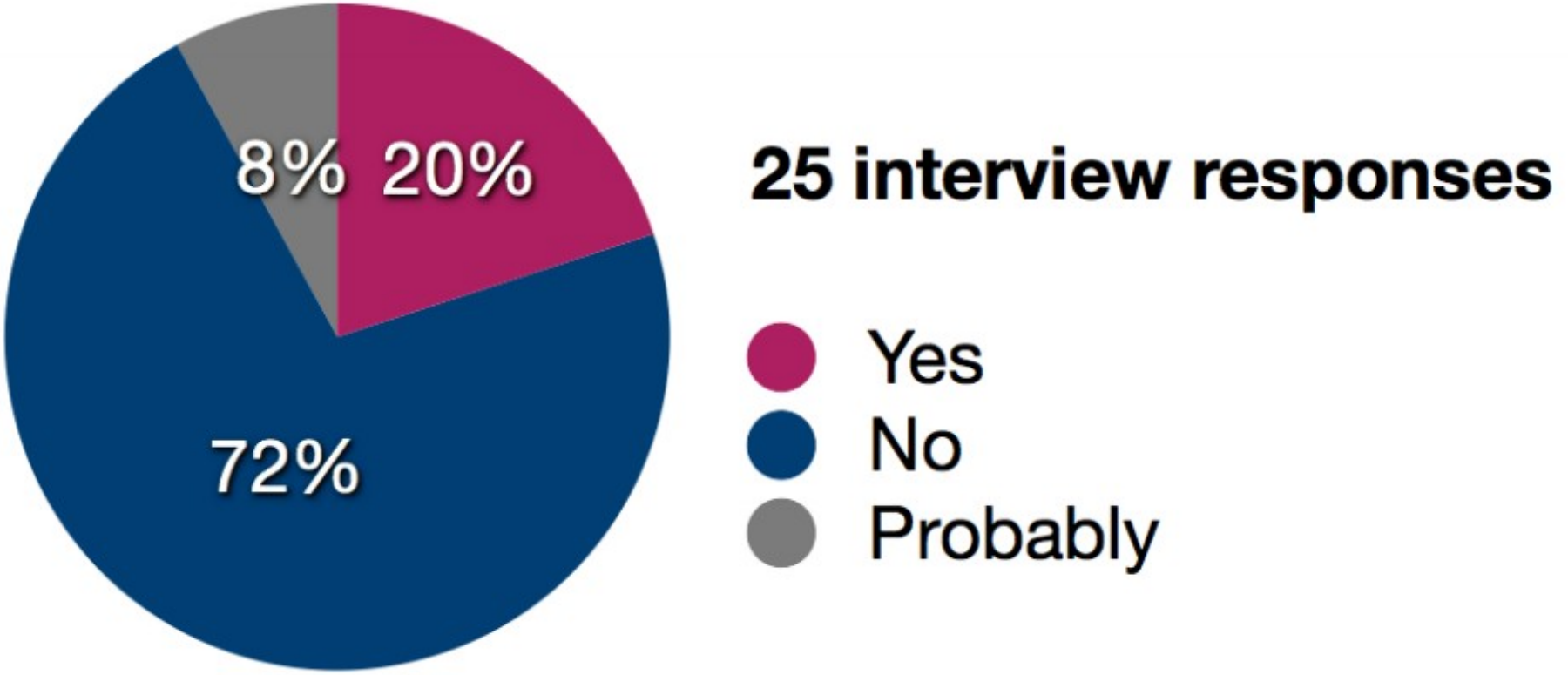
	Permission	n	Correct Answers	
1 Choice	READ_CALENDAR	101	46	45.5%
	CHANGE_NETWORK_STATE	66	26	39.4%
	READ_SMS$_1$	77	24	31.2%
	CALL_PHONE	83	16	19.3%
2 Choices	WAKE_LOCK	81	27	33.3%
	WRITE_EXTERNAL_STORAGE	92	14	15.2%
	READ_CONTACTS	86	11	12.8%
	INTERNET	109	12	11.0%
	READ_PHONE_STATE	85	4	4.7%
	READ_SMS$_2$	54	12	22.2%
4	CAMERA	72	7	9.7%

Table 4: The number of people who correctly answered a question. Questions are grouped by the number of correct choices. n is the number of respondents. (Internet Survey, $n = 302$)

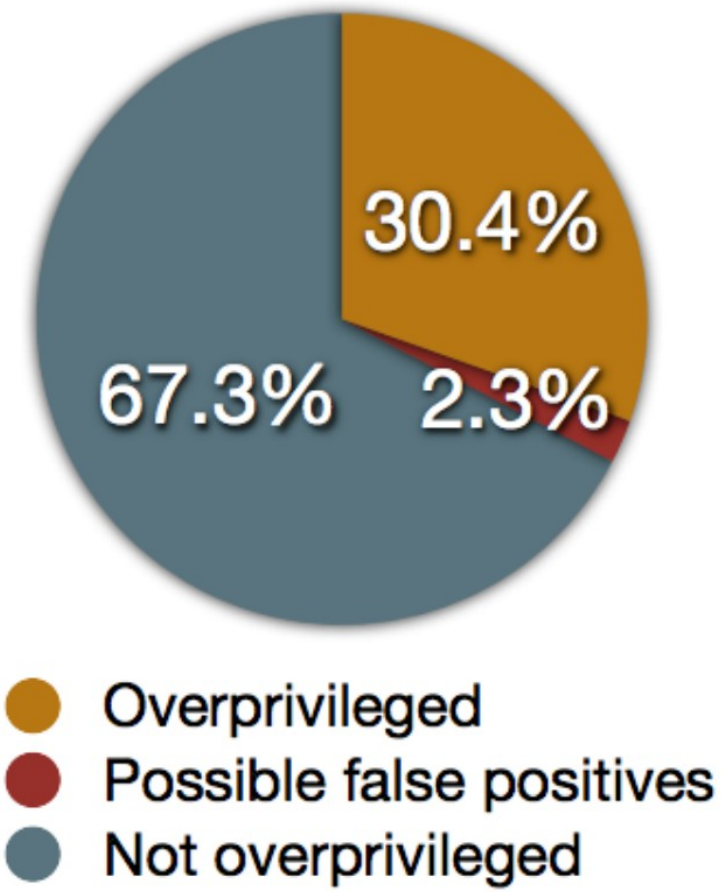
Are Permission Manifests Usable?

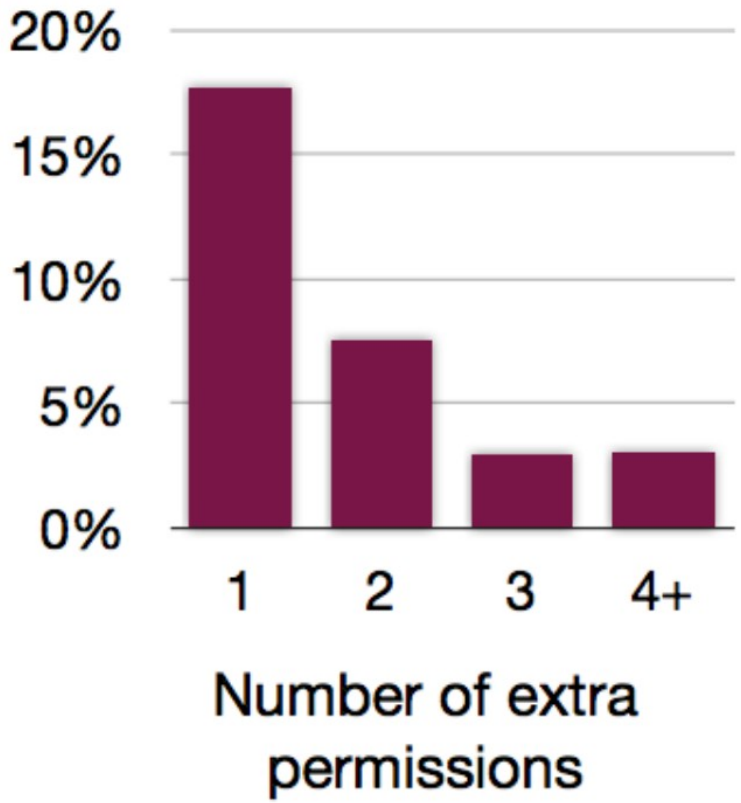
Do users act on permission information?

"Have you ever not installed an app because of permissions?"

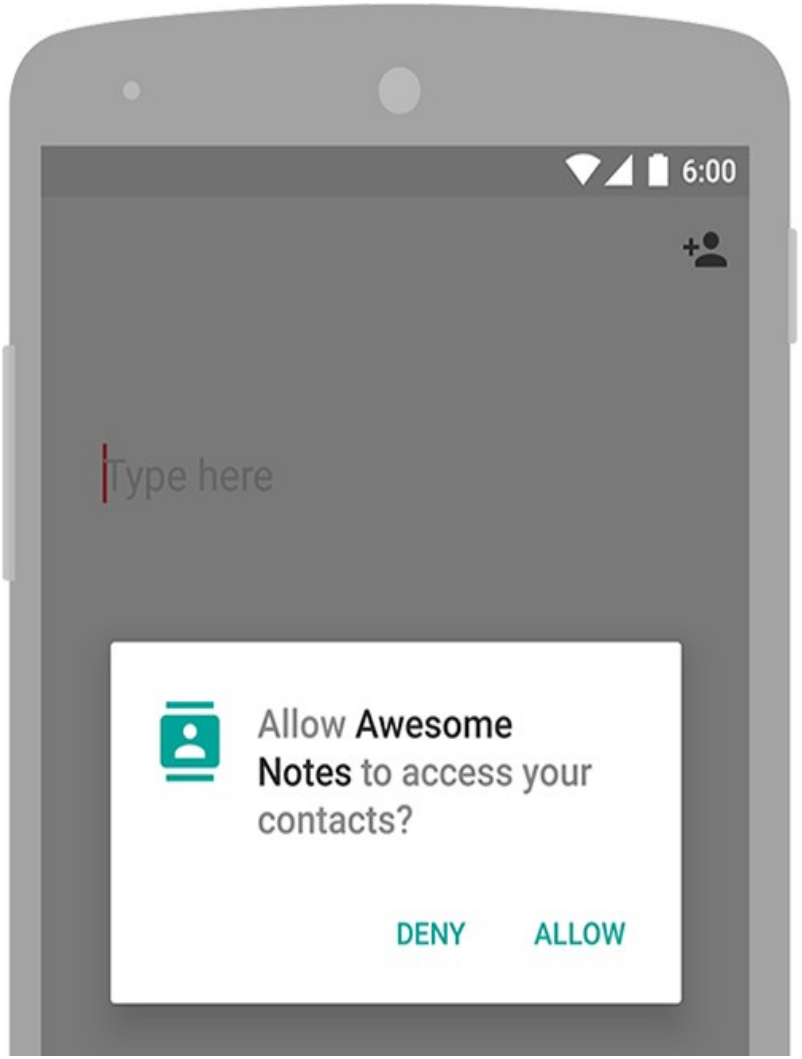


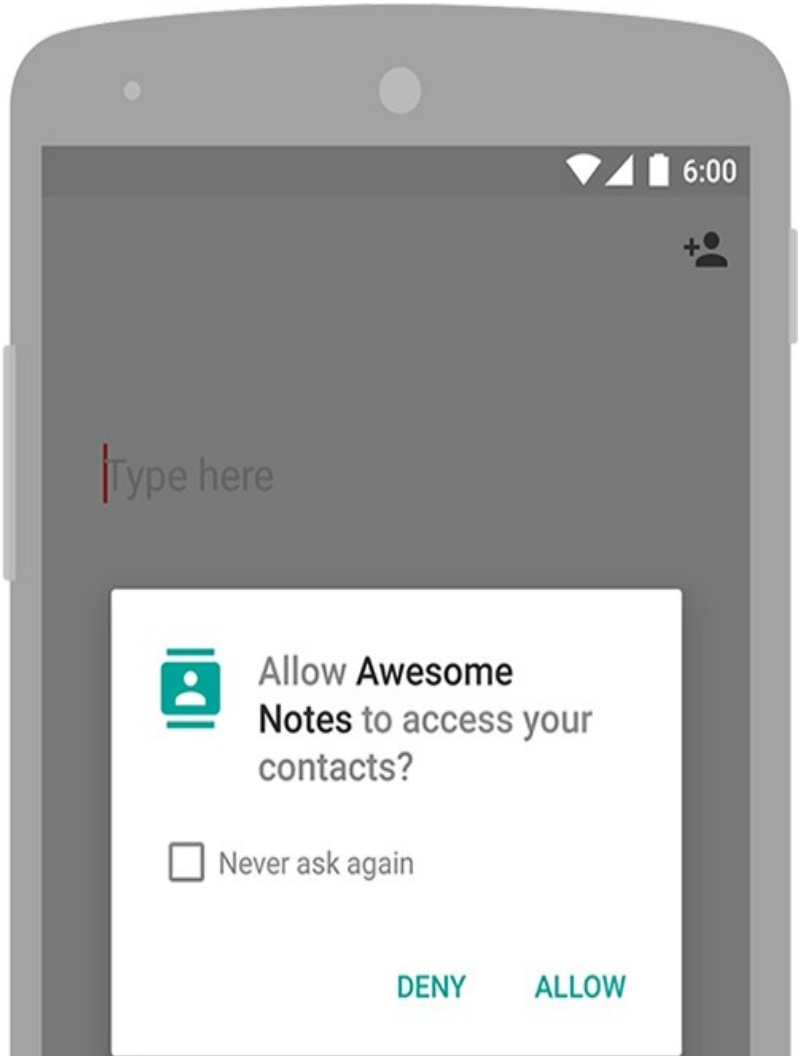
Developers Don’t know the Permissions They Need





Android Now Asks at Runtime





Manifests

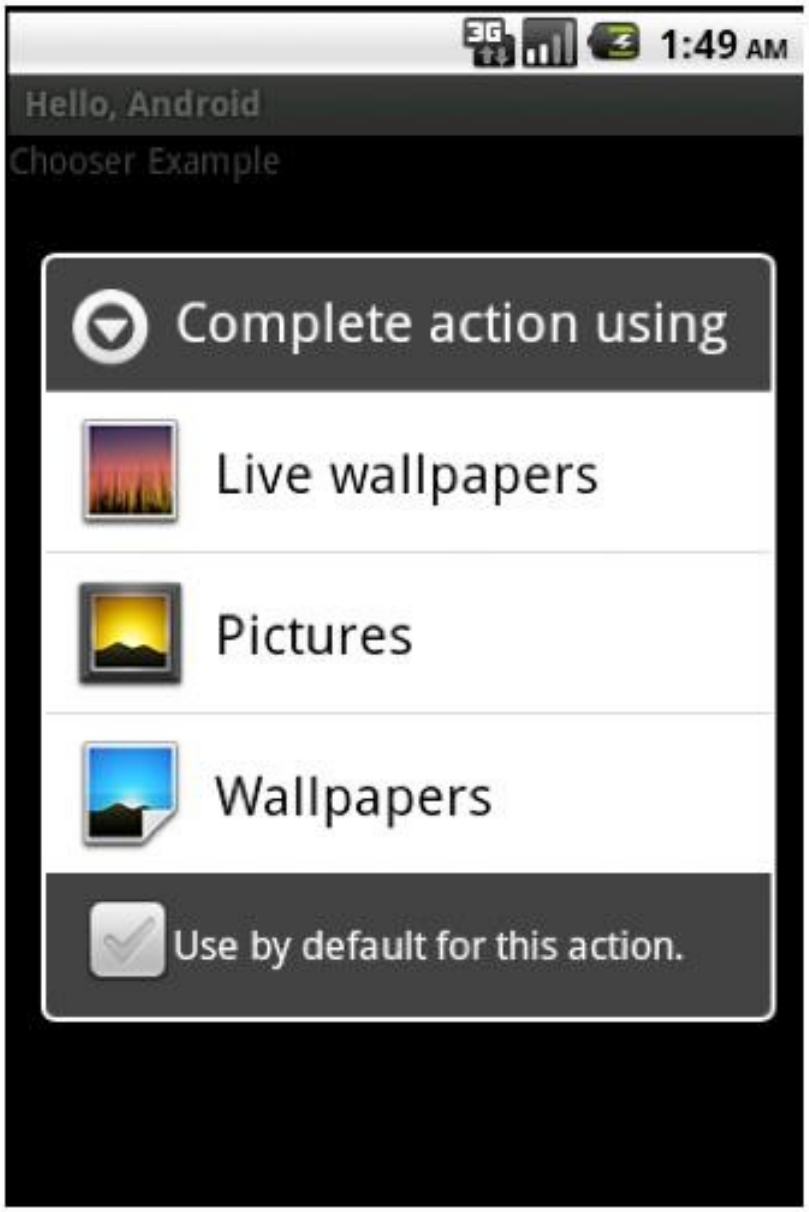
- **Android app needs to request permission in its manifest**
  - Operating System then asks the user (at runtime or install)
- **OS can also just grant any right that doesn't seem dangerous**
- **Manifest also defines what endpoints *other* apps can access.**
- **Whole class of malware that takes advantage of this of mis-configuration.**

Inter-Process Communication

- **Intents**
  - Primary mechanism for IPC in Android
- **Explicit**
  - specify name: e.g. com.example.testApp.MainActivity
- **Implicit**
  - specify action (e.g., ACTION_VIEW) and/or data (URI & MIME type)

Inter-Process Communication

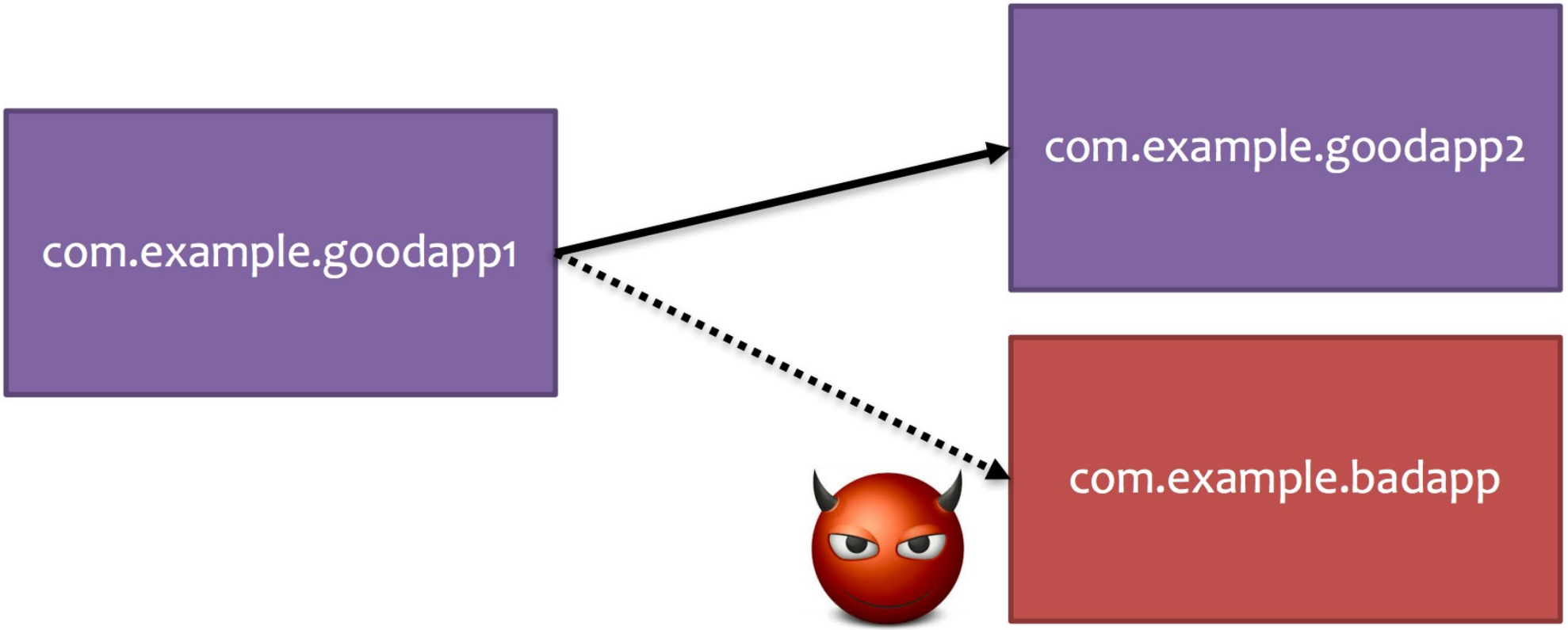
- **Implicit intents specify an action that can invoke any app on the device able to perform the action**
- **Useful when your app cannot perform the action**
  - Other apps probably can and you'd like the user to pick which app to use.



Intent Eavesdropping

Implicit intents make intra-app messages public

Attack #1: Eavesdropping / Broadcast Theft



Unauthorized Intent Receipt

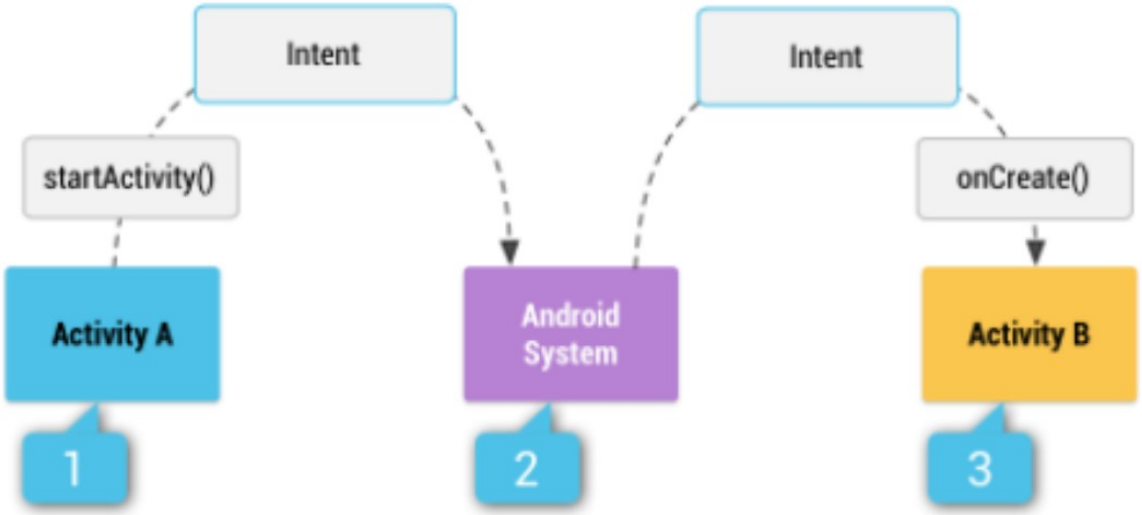
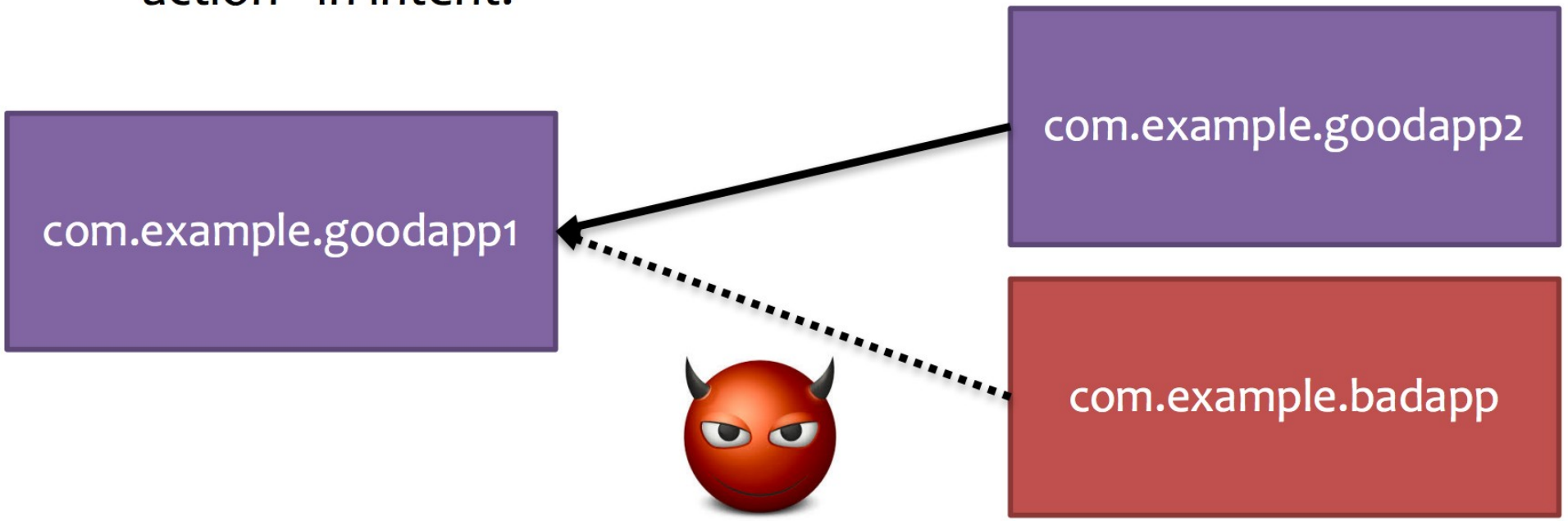


Figure 1. How an implicit intent is delivered through the system to start another activity: **[1]** *Activity A* creates an `Intent` with an action description and passes it to `startActivity()`. **[2]** The Android System searches all apps for an intent filter that matches the intent. When a match is found, **[3]** the system starts the matching activity (*Activity B*) by invoking its `onCreate()` method and passing it the `Intent`.

"**Caution:** To ensure that your app is secure, always use an explicit intent when starting a Servier. Using an implicit intent to start a service is a security hazard because you can't be certain what service will respond to the intent, and the user can't see which service starts."

Intent Spoofing

- Attack #1: General intent spoofing
  - Receiving implicit intents makes component public.
  - Allows data injection.
- Attack #2: System intent spoofing
  - Can't directly spoof, but victim apps often don't check specific "action" in intent.



Intent + Malware

- **Malware often times takes advantage of improperly filtered intents to gain access to the permissions in other applications**

MOBILE NETWORKS

GSM & UMTS (2G and 3G) History

- **1982 Study group Groupe Spéciale Mobile (GSM)**
  - Conference of European Posts and Telegraphs
- **Goals**
  - Good voice quality
  - Cheap end systems, low running costs
  - International roaming
  - Handheld mobile devices, new services (e.g. SMS)
- **Phase I of GSM specification published 1990**

Security Goals

- **Protect against interception of voice traffic on the radio channel:**
  - Encryption of voice traffic.
- **Protect signaling data on the radio channel:**
  - Maintain channel security and configuration
  - Encryption of signaling data.
- **Protections against unauthorized use (charging fraud):**
  - Subscriber authentication (IMSI, TMSI).
- **Theft of end device:**
  - Identification of MS (IMEI), not always implemented.

Components

- **MS (Mobile Station) = ME (Mobile Equipment) + SIM(Subscriber Identity Module)**
  - SIM gives personal mobility (independent of ME)
- **BSS (Base Station Subsystem) = BTS (Base Transceiver Station) + BSC (Base Station Controller)**
- **Network Subsystem = MSC (Mobile Switching Center,**
- **central network component) + VLR, HLR, AUC, ...**
- **Manage Call Routing & Roaming Information**
  - HLR (Home Location Register)
  - VLR (Visitor Location Register)
- **Manages security relevant information**
  - AUC (Authentication Center)

SIM: Subscriber Identity Module

- **Smart card (processor chip card) in MS:**
  - Current encryption key Kc (64 bits)
  - Secret subscriber key Ki (128 bits)
- **Algorithms A3 and A8**
- **IMSI (International Mobile Subscriber Identity)**
  - Unique 64-bit value/id
- **TMSI (Temporary Mobile Subscriber Identity)**
  - Temporary value used by a particular location/base station
- **PIN or PUK (PIN Unlock Code)**
- **Contacts (stored on SIM)**
- **SIM Application Toolkit (SIM-AT) platform**

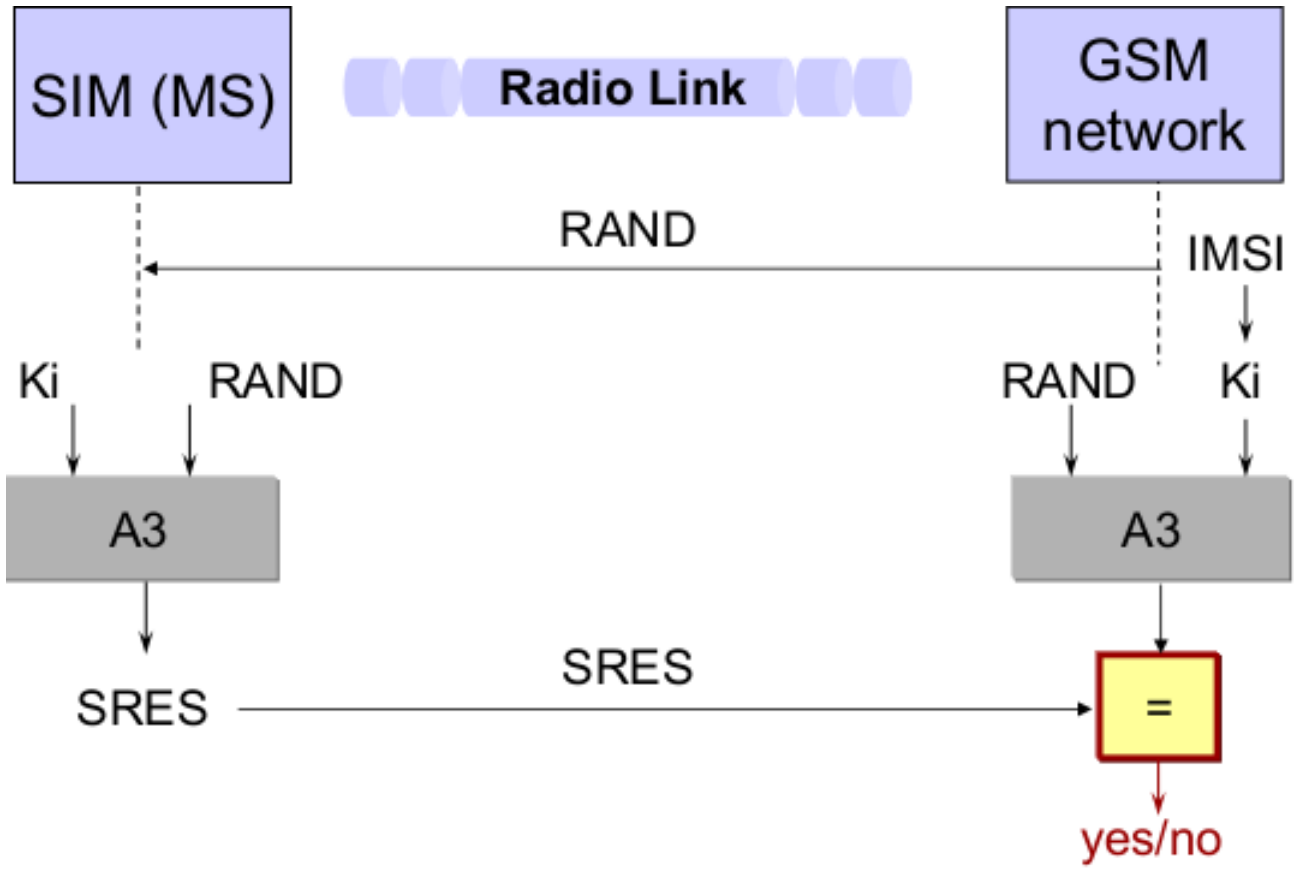
Cryptography in GSM

- **Secret Algorithms**
  - A3 authentication algorithm
  - A5 data encryption algorithm
  - A8 ciphering key generating algorithm
- **All are symmetric key ciphers**
  - Public key cryptography was considered at the time (1980s) but not considered mature enough

Authentication in ME

- **Fixed subsystem transmits a non-predictable number RAND (128 bits) to the MS.**
  - RAND chosen from an array of values corresponding to MS.
- **MS computes SRES, the ‘signature’ of RAND, using algorithm A3 and the secret : Individual Subscriber Authentication Key Ki.**
- **MS transmits SRES to the fixed subsystem.**
- **The fixed subsystem tests SRES for validity.**
- **Computations in ME performed in the SIM.**
- **Location update within the same VLR area follows the same pattern.**

GSM Subscriber Authentication



TMSI and ISMI

- When a MS makes initial contact with the GSM network, an unencrypted subscriber identifier (IMSI) has to be transmitted.
- The IMSI is sent only once, then a temporary mobile subscriber identity (TMSI) is assigned (encrypted) and used in the entire range of the MSC.
- When the MS moves into the range of another MSC a new TMSI is assigned.
- Range determined by provider, location, and signal strength (opportunity to create false BS)

Stream Cipher: A5

- **Why a stream cipher, not a block cipher?**
- **Radio links are relatively noisy.**
  - Block cipher: a single bit error in the cipher text affects an entire clear text frame
  - Stream cipher: a single bit error in the cipher text affects a single clear text bit (on average, depending on implementation)

GSM Fraud

- **Attacks the revenue flow rather than the data flow**
  - Does not break the underlying technology.
- **Roaming fraud: subscriptions taken out with a home network; SIM shipped abroad and used in visited network.**
  - Fraudster never pays for the calls
  - Home network has to pay the visited network for the services used by the fraudster
  - Scope for fraudsters and rogue network operators to collude.
- **Premium rate fraud: customers lured into calling back to premium rate numbers owned by the attacker.**
  - GSM charging system misused to get the victim's money.

GSM Fraud

- **Business model attack: Criminals open a premium rate service, call their own number to generate revenue, collect their share of the revenue from the network operator, and disappear at the time the network operator realizes the fraud.**
- **Countermeasures:**
  - Human level: exercise caution before answering a call back request.
  - Legal system: clarify how user consent has to be sought for subscribers to be liable for charges to their account.
  - Business models of network operators.
- **GSM operators have taken a lead in using advanced fraud detection techniques, based e.g. on neural networks, to detect fraud early and limit their losses.**

GSM – Summary

- **Voice traffic encrypted over the radio link (A5) but calls are transmitted in the clear after the base station.**
- **Optional encryption of signaling data**
  - but ME can be asked to switch off encryption.
- **Subscriber identity separated from equipment identity.**
- **Some protection of location privacy (TMSI).**
- **Security concerns with GSM:**
  - No authentication of network: IMSI catcher pretend to be BS and request IMSI
  - Undisclosed crypto algorithms.

UMTS (3G) Security

- **UMTS security architecture similar to GSM.**
- **Crypto algorithms are published.**
- **Some network authentication, but who is ‘the network’?**
  - The entire UMTS network? Authentication stops rogue base station from inserting bogus instructions to mobile stations; traditional viewpoint of telecom companies.
  - The visited (serving) network? More likely to be subscriber’s expectation
- **Standard crypto integrity checks of limited usefulness; noise on radio channel would invalidate authenticators.**
- **Instead, UMTS adds integrity and freshness checks on signaling data from network to MS.**
  - Checks SQN and RAND number similar to GSM
  - No actual crypto-based authentication

Base Station Attacks

- **Why bother with 2G/3G?**
  - Isn’t everyone using 4G (LTE) and 5G now?
- **Downgrade attacks!**
  - https://arxiv.org/abs/1510.07563v1

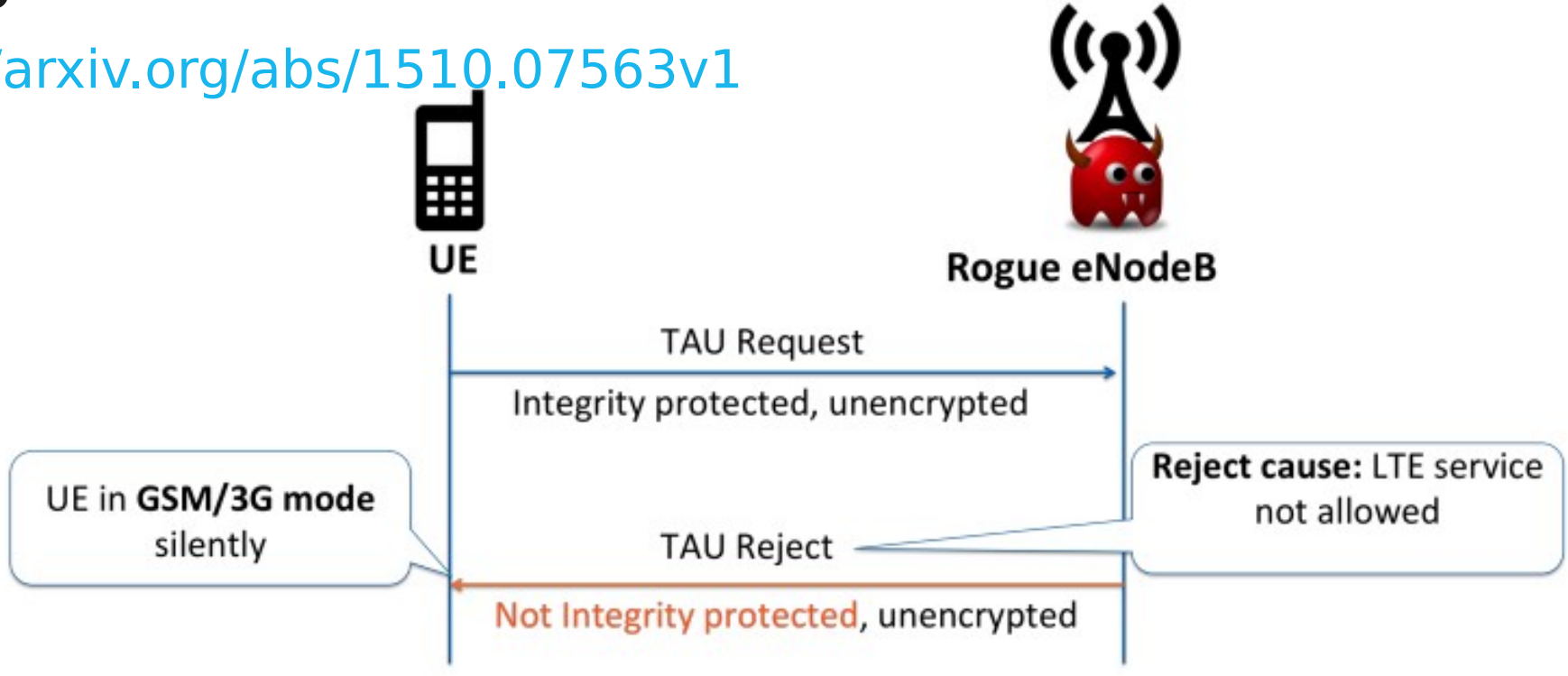


Fig. 8. DoS attack - denying LTE network services (D1)

WLAN SECURITY

WLAN (Background)

- **Wireless LAN (WLAN) specified in the IEEE 802.11 standards.**
- **Can be operated in two modes:**
  - Infrastructure mode: mobile terminals connect to a local network via access points.
  - Ad-hoc mode: mobile terminals communicate directly.
- **An open WLAN does not restrict who may connect to an access point.**
- **Public access points are known as hot spots.**

SSID & MAC

- **Each access point has a Service Set Identifier (SSID).**
- **Access points can be configured not to broadcast their SSIDs**
  - SSID included in signaling messages and can be intercepted
- **Access points can use MAC filtering**
  - White-list known MAC addresses
  - Attacker can learn valid MAC address by listening to connections
  - Then connect with spoofed MAC address.
- **Do not base access control on information the network needs to manage connections**

WEP

- **Wireless Equivalent Privacy**
  - First standard for protecting WLAN traffic (1997).
  - Unfortunately, a case study in getting cryptographic protection seriously wrong.
- **Like GSM/UMTS, uses stream cipher for secrecy.**
- **Unlike GSM/UMTS, WEP also tries to provide integrity protection of wireless traffic.**

WEP Cryptography

- **Authentication based on a shared secret: pre-shared secrets**
  - Suitable for small installations like home networks; most LANs use the same key for all terminals.
- **Sender, receiver share secret 40-bit or 104-bit key K.**
- **Transmitting a message m:**
  - sender computes 32-bit checksum CRC-32(m)
  - prepends 24-bit IV to key
  - And generates a key stream with the 64-bit (128-bit) key
- **K’ = IV||K using RC4; IV sent in the clear.**
- **Ciphertext c = (m||CRC-32(m)) ⊕ RC4(K’).**
- **Receiver computes c ⊕ RC4(K’) = (m||CRC-32(m)) and verifies checksum.**

Problems with WEP

- **CRC-32 is a linear function!**
  - An attacker who only has a ciphertext, but neither key nor plaintext, can modify the plaintext by a chosen difference Δ.
- **Compute δ = CRC-32(Δ) and add (Δ||δ) to c; this is a valid encryption of the plaintext m⊕Δ:**
  - (m||CRC-32(m)) ⊕ RC4(K') ⊕ (Δ||δ) = (m⊕Δ||CRC-32(m) ⊕ δ) ⊕ RC4(K') = (m ⊕ Δ ||CRC-32(m ⊕ Δ)) ⊕ RC4(K').
- **Second problem: IV is too small or re-used**
- **Third problem: cryptanalytic attacks on RC4, e.g. exploiting weak keys; typically require attacker to collect a sufficient amount of encrypted packets.**
-

WPA – Wi-Fi Protected Access

- **Challenge: devices already deployed in the field but you have got the standard wrong.**
  - Can’t ask users to throw away their devices; you must find a fix that works with current equipment.
  - Only software upgrades are feasible.
  - Changes to encryption must work with existing hardware architectures.
- **Challenge: quick fix while new standard is being drafted that will be forward compatible.**

WPA – Restrictions & Remedies

- **Processor load:**
  - 3DES needs about 180 instructions per byte.
  - 802.11b data throughput: 7 Mbit/s, i.e. 875000 bytes/s.
  - Processor must execute 157.5M instructions per second; way beyond a typical AP.
- **First remedy:**
  - Replace CRC with cryptographic integrity check function.
  - Plus better key management, longer IV.

Michael (Replaces CRC-32)

- **CRC replaced by “Michael” Message Integrity Code**
- **Constructed from shift, add, XOR operations (fast)**
  - 3.5 cycles/byte on ARM, 5.5 cycles/byte on i486.
- **‘Medium’ security:**
  - Target security level equivalent to guessing 2^20 messages
- **Brute force countermeasures:**
  - Base station switches off for a minute (opportunity for DoS attack) when receiving two bad packets within a second.

TKIP (Temporal Key Integrity Protocol)

- Combines encryption & integrity verification.
- Different ‘temporal’ key for each frame.
- Based on RC4 with 128-bit keys (for backward compatibility with WEP)
- 48-bit IV used as sequence number
- MIC appended to data before encryption
- Key update after 2^16 IVs have been used

TKIP – Key Hierarchy

- **Pairwise Master Key (PMK):**
  - Derived from password (pre-shared key)
- **Pairwise Transient Key (PTK):**
  - Derived from **PMK**, MAC addresses of station and AP, nonces from station and AP; split into
    - Key Confirmation Key (**KCK**): for key authentication
    - Key Encryption Key (**KEK**): for distributing group keys
    - Temporal Key (**TK**): basis for data encryption
- **Temporal session key: derived from TK and MAC address of AP.**
- **WEP key and IV (per packet): derived from temporal session key and sequence number.**

Problem: Password Guessing

- **WPA-PSK (pre-shared key) vulnerable to password guessing attacks.**
  - Attacker records traffic as victim connects to WLAN.
  - Attacker guesses a passphrase, computes master key PMK’ for the guess and the known (intercepted) values SSID and SSID length.
- **Transient key PTK’ is derived from PMK’ and the intercepted MAC addresses and nonces.**
- **Recorded encrypted messages are decrypted with candidate key PTK’.**
- **If result is meaningful plaintext, the guess of the passphrase is correct with high probability.**

WPA2 – Current Standard

- **WPA2 from WiFi Alliance, based on IEEE 802.11i.**
  - IEEE 802.11i and WPA2 overlap and are sometimes used as synonyms; however, this is not completely correct.
- **Two modes:**
  - TSN (Transitional Security Network)
    - Backwards compatible with WEP
  - RSN (Robust Security Network)
    - Adds support for AES-128
    - Encryption for ad-hoc modes

WPA2 – Cryptography

- **Authentication:**
  - For large networks with EAP (Extensible Authentication Protocol) e.g. RADIUS
  - For smaller networks with TKIP (pre-shared key)
- **Encryption: 128-bit AES (key & block size) in CCM mode: Counter mode CBC MAC Protocol (CCMP).**
  - Counter with CBC-MAC (CCM) defined in RFC 3610.
  - 64-bit MIC derived from CBC-MAC.
- **Not compatible with older hardware.**
- **Transitional Security Network (TSN) allows RSN and WEP to coexist on the same WLAN.**
  - Devices using WEP can be a security risk.

WPA3 – Future Standard

- **Developing as of 2018**
  - No word on deployment
- **Uses modern cryptography algorithms and PKI**
  - AES-256, SHA-384
  - Uses DH key exchange, plus authentication challenge

Dragonfly Key Exchange

- **Alice (the client)**
  - Generates two random values (a,A)
  - Computes a scalar value (a+A) mod q, where q is a large prime number
  - Passes is the PE (Password Equivalent, e.g. hash of the password) raised to the power of -A.
- **Bob (the access point)**
  - does the same
- **Exchange values**
  - They will have the same shared commit key (PE^{ab}).
- **Bob and Alice will only have the shared shared commit value if they both have the same password.**
- **The intruder cannot determine either the original password or the final shared value.**

Dragonfly Key Exchange

COMMIT

Random: a, A

sA = (a+A) mod q

elementA = PE^{-A}

Random: b, B

sB = (b+B) mod q

elementB = PE^{-B}

sA,PE^{-A}

sB,PE^{-B}

ss = $(PE^{sB} \times PE^{-B})^{a}$

= $(PE^{b+B-B})^{a}$

= PE^{ab}

ss = $(PE^{sA} \times PE^{-A})^{b}$

= $(PE^{a+A-A})^{b}$

= PE^{ab}

https://medium.com/asecuritysite-when-bob-met-alice/hello-to-wpa-3-ae8b9c365