Designing Secure Systems guest lecture

Mark D. Ryan

15 November 2017

UNIVERSITY OF BIRMINGHAM | Security and Privacy

Online services (aka cloud computing)

Communication (1995-)

- Email
- Gmail
- Whatsapp/ iMessage/ Telegram
- Facebook/ LinkedIn/ Snapchat/ Instagram

Online services (aka cloud computing)

Communication (1995-)

- Email
- Gmail
- Whatsapp/ iMessage/ Telegram
- Facebook/ LinkedIn/ Snapchat/ Instagram

Workflow (2005-)

- Canvas/ Easychair/ ServiceNow
- JustGiving/ SurveyMonkey
- Online docs
- Calendar
- Banking

Online services (aka cloud computing)

Communication (1995-)

- Email
- Gmail
- Whatsapp/ iMessage/ Telegram
- Facebook/ LinkedIn/ Snapchat/ Instagram

Workflow (2005-)

- Canvas/ Easychair/ ServiceNow
- JustGiving/ SurveyMonkey
- Online docs
- Calendar
- Banking

IoT (2015-)

- Thermostats
- Lights
- Fridge, kettle, toilet,…
- Speech-understanding speakers
- Cars, trains, …

Online services (aka cloud computing)

Communication (1995-)

- Email
- Gmail
- Whatsapp/ iMessage/ Telegram
- Facebook/ LinkedIn/ Snapchat/ Instagram

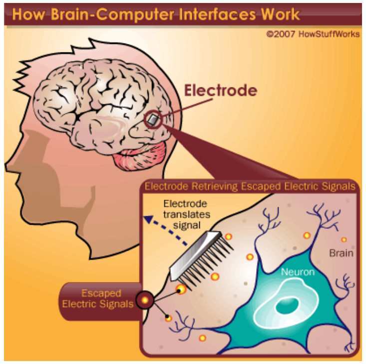
Workflow (2005-)

- Canvas/ Easychair/ ServiceNow
- JustGiving/ SurveyMonkey
- Online docs
- Calendar
- Banking

IoT (2015-)

- Thermostats
- Lights
- Fridge, kettle, toilet,…
- Speech-understanding speakers
- Cars, trains, …

Brain (2025-)



Poll 1

- In 20 years time, when Apple (or Amazon or Google) announces a brain-computer interface for $500, I will:
  - Never buy it. On the contrary: I will campaign to have it banned.
  - Resist it at first, but eventually succumb. I hate the idea, but I know that the incredible convenience will (as always) prove greater than my reservations.
  - Be an early adopter, as I was for smartphones, online-everything, speech-understanding speakers, IoT devices, driverless cars…

How should the data amassed by online services be used?

Acceptable uses (?)

- Providing the service
- Developing additional services
- Targeted advertising
- Necessary, proportionate, targeted surveillance

Unacceptable uses (?)

- Aggregation with data from other sources
- Mass surveillance
- Commercial pestering (spam, etc.)
- Blackmailing, coercion, ...
- **Algorithmic determination of eligibility for services**

Video

- https://www.youtube.com/watch?v=yzrmdXWEh20

How can we allow the acceptable uses while preventing the unacceptable ones?

Laws and regulations

- Flexible
- Hard to enforce

Clever technologies

- Inflexible
- Ideally, they are self-enforcing

An old technology: PGP encryption

PGP encryption

- Provides end-to-end encryption in mail systems
- Provides "web of trust" model to allow senders to authenticate the recipient's public key
  - You get your friends to certify your key
  - Your interlocutor can then judge whether your key is "sufficiently" certified.

Prevents

- Cloud-side analysis of messages
  - Including cloud-side search of messages

Allows (in fact, exacerbates)

- Cloud-side analysis of metadata (aka communications data)

Some recent technologies: Signal protocol

Signal protocol

- Provides end-to-end encryption in messaging systems
  - Used by Whatsapp and Signal
  - 1Bn users
  - (Compare with PGP: hardly any users.)

Prevents

- Cloud-side analysis of messages
  - Including cloud-side search of messages

Allows

- Cloud-side analysis of metadata (aka communications data)

Some recent technologies: End-to-end encrypted "Google Docs"

"Private editing using untrusted cloud services" (Yan Huang, David Evans, Virginia University, 2011.)

- Participants share a password using some trusted means (e.g., email, whatsapp, …)
- Documents are stored encrypted with key derived from the password.
- Participants can simultaneously edit the same document...
- Available as browser extension and server code.

Prevents

- Cloud-side analysis of documents
  - Including cloud-side search of documents

Allows

- Cloud-side analysis of metadata (aka communications data)

Some recent technologies: End-to-end encrypted Easychair (or Canvas or ServiceNow or ...)

"Privacy Supporting Cloud Computing: ConfiChair, a Case Study" (M. Arapinis, S. Bursuc, M. Ryan, University of Birmingham, 2012.)

- Participants share keys using trusted means
- Documents are stored encrypted with those keys.
- In-browser key translation….
- Supports more complex work flows.

Prevents

- Cloud-side analysis of documents
  - Including cloud-side search of documents
- **Cloud-side analysis of some metadata (e.g., who reviews who's paper).**

Usability hit?

- PGP: **heavy usability hit.** “Why Jonny can’t encrypt” became a famous paper.
- Signal protocol: **almost no hit!**
  - Possibly some issues arise if you change phone after a message has been sent.
- Huang/Evans private editing: **moderate hit** (copy/paste passwords).
- Confichair: **moderate hit** (copy/paste keys).

Poll 2

- In 20 years time, when everything is connected to the internet, and everything we say or do is recorded in the cloud, how ***will*** we routinely address this problem?
  - By ignoring it.
  - By having legislation aimed at curbing abuses of data.
  - By having in place clever technologies that precisely control how the data can be used.

What we want

Technology which

- Allows cloud to perform certain specified *computations* with user data, but prevents other computations being done.
  - "Reverse DRM"
- Provides a rich policy language to specify what kinds of computations are allowed.
- Provides secure evidence to clients about the policy: "attestation".

Approaches to solving the "confidentiality from the cloud provider" problem

Crypto

- Fully homomorphic encryption
- Functional encryption
- Order-preserving encryption
- Multi-party computation
- White-box crypto
- Indistinguishability obfuscation

Challenges

- Restrictions on the program P
- Use-case restrictions
- Performance

Hardware

- TPM & Intel TXT
- ARM Trustzone
- ***Intel SGX***

Challenges

- Requirement to trust HW design and implementation
- Size of TCB
- Business model
- Documentation

Binding keys to programs using Intel SGX remote attestation

Mark D. Ryan

Designing secure systems lecture

15 November 2017

UNIVERSITY OF BIRMINGHAM | Security and Privacy

Intel SGX

Intel SGX is a set of processor instructions which allow one:

- To set up an enclave (code & memory) such that the code runs in a way that it and its memory are protected from interference from the OS and other software
- To securely report the state of the enclave, locally and remotely

Present on all (major) Intel processors from Skylake (2015) onwards

Not the first *hardware security anchor*

Trusted platform module (TPM)

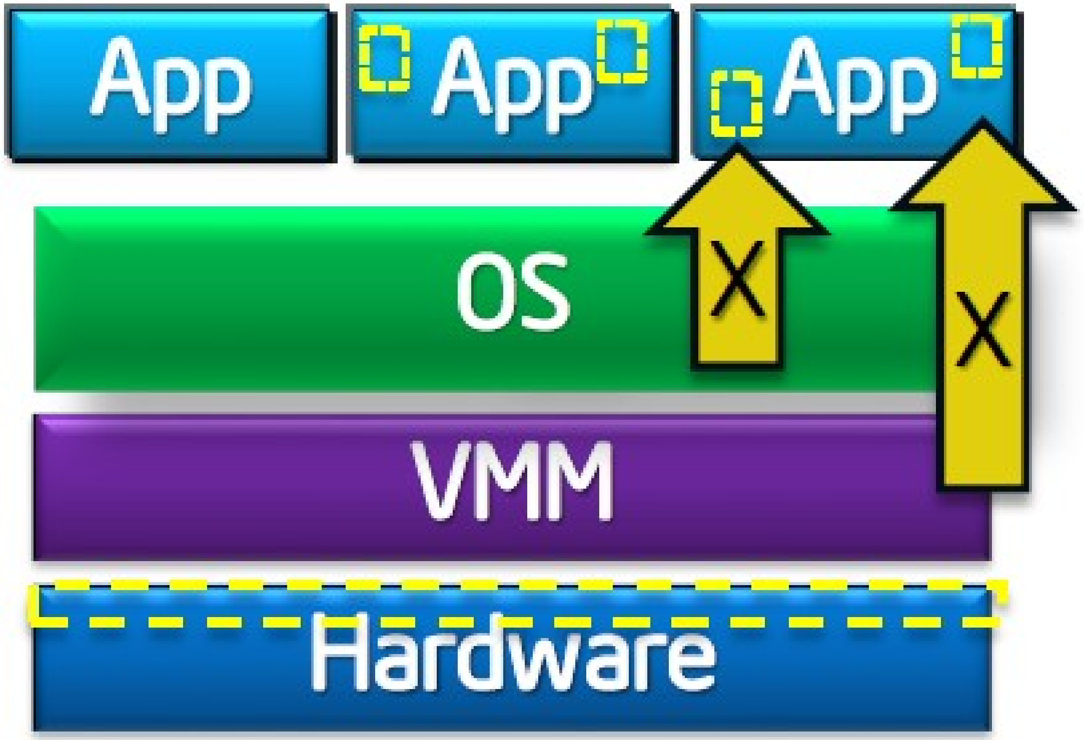
- Version 1 (2004), 1.2 (2008), 2.0 (2014-)
- Separate chip soldered to motherboard
- API that allows you to create keys whose secret part never leaves the TPM
  - A key can be locked to “authdata” (like a password to use the key)
  - And/or can be locked to PCR values, which “measure” the boot sequence

Best known use: Microsoft Bitlocker

ARM TrustZone

- ARM processors have two execution modes, with hardware-enforced access control between them:
  - “Normal world” Runs the rich OS (e.g., Android) and apps
  - “Secure world” Runs security-critical code.

Intel SGX: attacks addressed



An enclave within an app is protected from interference from other software, including the OS and VMM. Note that enclaves can only run in ring 3 (user space).

Intel SGX: attacks not addressed

- Side-channel attacks
  Cache and page access patterns
  - Extraction of RSA secret keys, under assumptions, by co-located [enclave] processes
  - Programmer is expected to mitigate this attack
- Hardware attacks
  - Chip decapsulation
  - Trojan hardware: vulnerabilities possibly introduced in the supply chain

Intel SGX

Not suited for:

- Applications that involve I/O on the platform
  - Password managers
  - Banking apps

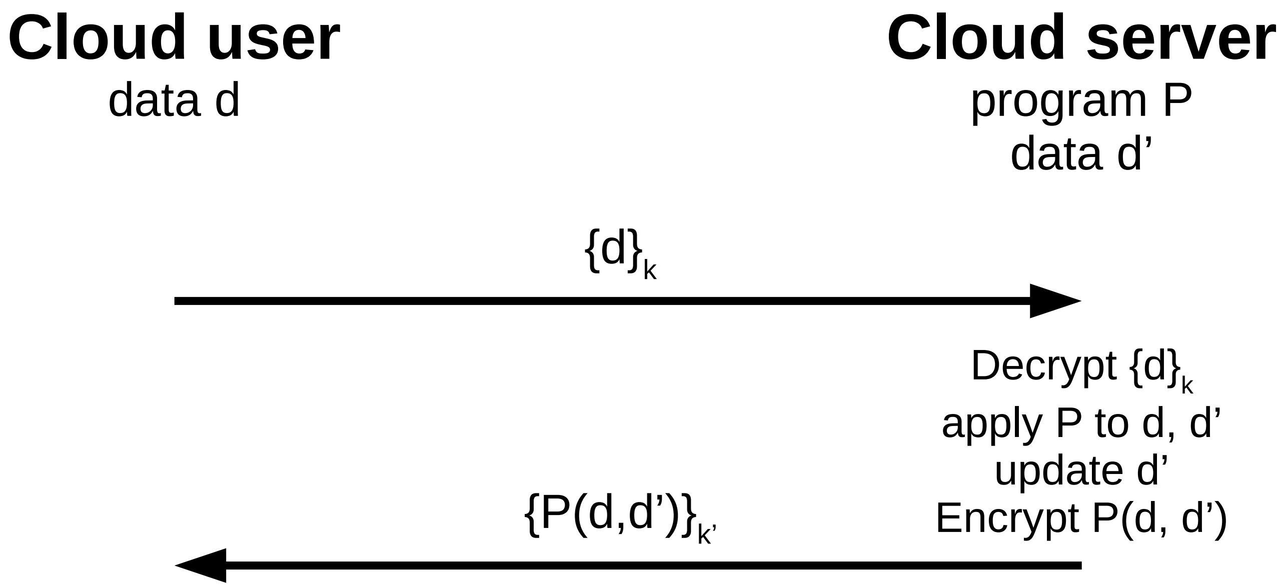
Partly suited for:

- DRM, where a server delivers content to your device, along with restrictions on how you use it

Well suited for:

- Cloud computing (“reverse DRM”), in which your device sends data to a cloud server, and you want to impose restrictions on how it is processed

Example: confidentiality from the cloud provider



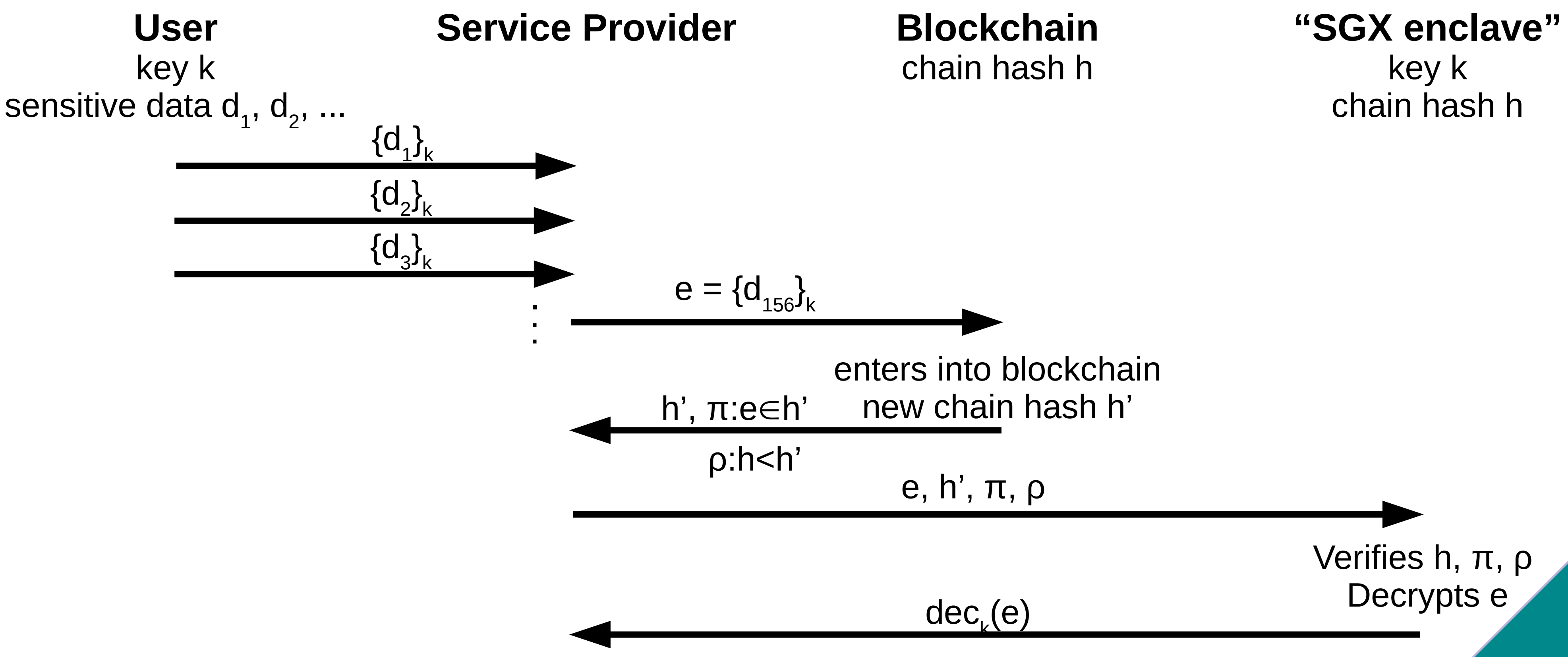
Bob cannot access d except by applying P to it and returning that to Alice.

In general, Bob does not know d, d' or k, k'
Bob does know P

Example: “accountable decryption”

Escrow with accountability: whenever it decrypts, SP creates evidence which cannot be suppressed or discarded.

Use case: user uploads her encrypted location continually; SP decrypts it only when she reports lost phone.



Intel SGX concepts

Protected memory

- Enclave Page Cache (EPC), access control, MEE

Enclave

- *“SGX enclave control structure”* (SECS)
  - Core data about the enclave, held in a dedicated EPC page.
- Life cycle of an enclave
  - Creation / loading / initialisation (aka launching) / teardown

Intel SGX concepts

Enclave measurement

- An enclave measurement (noted MRENCLAVE) is a hash of its code and initial data

Enclave identity

- MRENCLAVE: Its measurement is the strictest way to identify an enclave.
- MRSIGNER: An “enclave certificate” is a more flexible way to identify an enclave. The certificate is signed by the “independent software vendor” (ISV), and includes ISVPRODID and ISVSVN.
  - Allows data migration from old security versions to new ones.

Intel SGX concepts

As well as *processor instructions...*

- ECREATE, EADD, EEXTEND, EINIT, … : managing the enclave life cycle
- EGETKEY, EREPORT, … : managing data within an enclave.

… there are *Intel-provided enclaves*

- Launch enclave
- Provisioning enclave
- Quoting enclave

Intel SGX secret values

Some secret values are built into the platform.

Known to the processor and to Intel:

- *SGX Master derivation key*
  - Derived from *provisioning secret*

Known to the processor (but not to Intel)

- *Seal secret* (also known as SEAL_FUSES)
- OWNER_EPOCH

SKIP

Setting up an enclave

- System software uses ECREATE to set up the initial memory page allocated to the enclave, which contains the SGX Enclave Control Structure (SECS)
- It uses EADD to allocate further pages containing enclave code and initial data
- It uses EEXTEND to update the enclave's 'measurement'
- After loading the initial code and data pages into the enclave, the system uses a 'Launch Enclave' (LE) to obtain an EINIT token
  - The token is provided to the EINIT instruction to initialise the enclave
  - LE is a privileged enclave provided (e.g.) by Intel, signed by and Intel private key

SKIP

Initialising an enclave (more detail)

Untrusted system software sets up SECS and the enclave certificate SIGSTRUCT

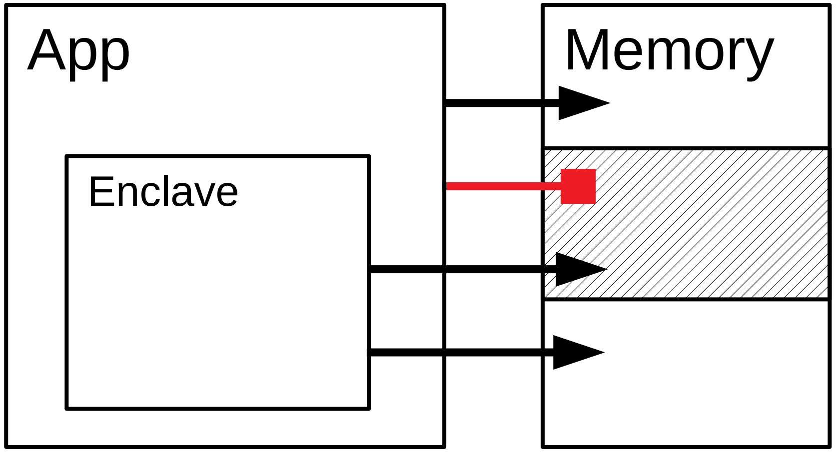
SECS	SIGSTRUCT	EINITTOKEN
MRENCLAVE	ENCLAVEHOST	MRENCLAVE
MRSIGNER	VENDOR	MRSIGNER
ATTRIBUTES	ATTRIBUTES	ATTRIBUTES
- DEBUG	ATTRIBUTEMASK	launch enclave info
- XFRM	ISVPRODID	MAC
ISVPRODID	ISVSVN	
ISVSVN	signature	

A launch enclave

- checks the enclave certificate SIGSTRUCT against SECS
- checks the “launch policy”
- produces EINITTOKEN
- Produces the EINITTOKEN MAC using a launch key obtained using EGETKEY

The processor instruction EINIT checks EINITTOKEN and initialises the enclave

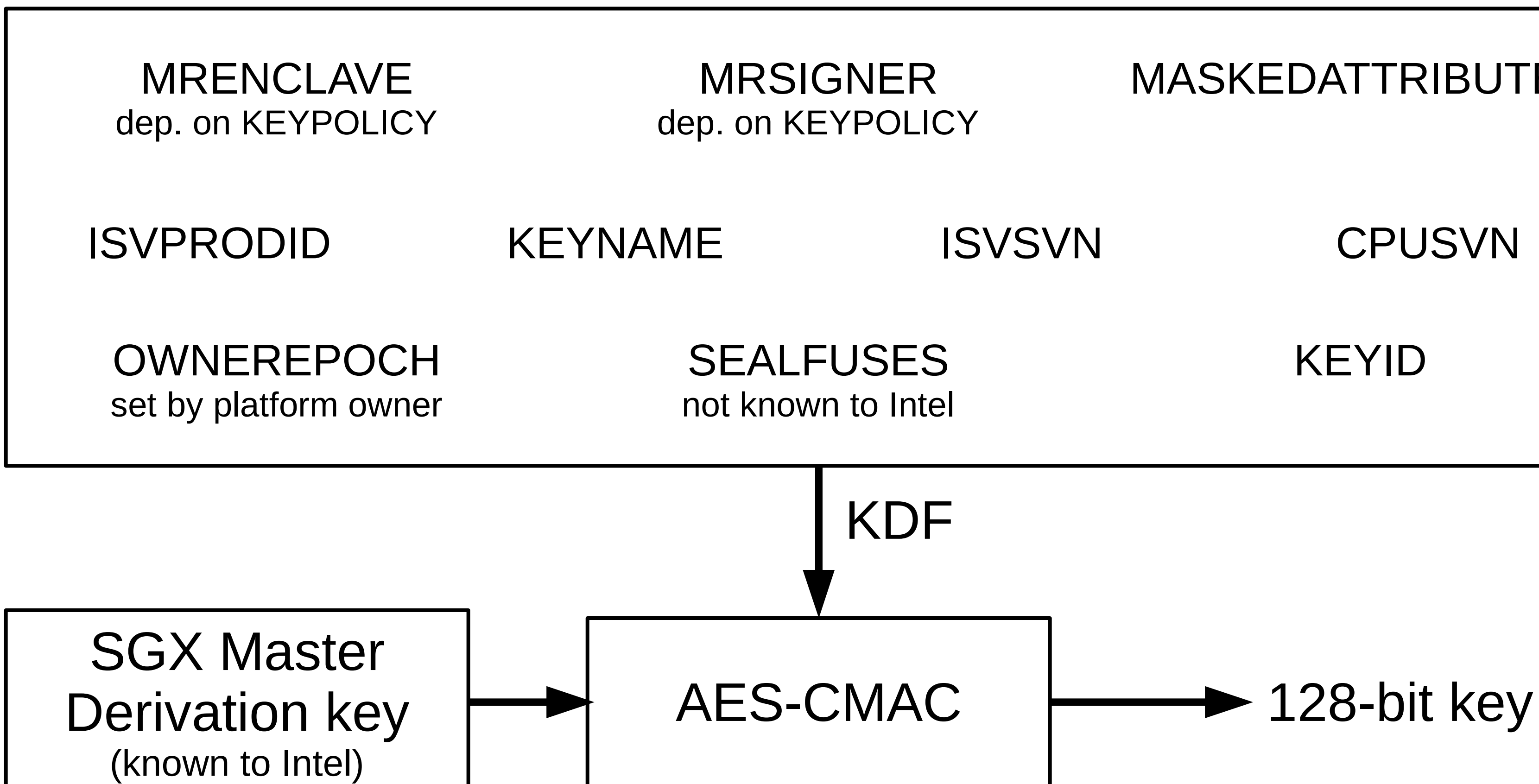
What an enclave can do



- Computations
- Access its own [encrypted] memory
- Access app memory
- Communicate with user, but insecurely
- Communicate with another party, which can be secure if the enclave shares a key with the other party
- Attest its identity (a hash of its binary and initial data) to another party
- “Seal” data, i.e. encrypt data with a key that only it can access, for persistent storage
  - Can use Platform Service Enclave (PSE) for *trusted time* and *monotonic counter*
- Teardown

Seal keys obtained using EGETKEY

Key request		
	KEYNAME	e.g. seal key, report key, provisioning key
	KEYID	
	KEYPOLICY	MRENCLAVE and/or MRSIGNER
	ATTRIBUTEMASK	
	ISVSVN	must be ≤ the caller's ISVSVN
	CPUSVN	must be ≤ the calling platform's CPUSVN



SKIP

Migrating data between enclaves

- **Same platform, same enclave (just a different instance):**
  - Sealed blob can migrate.

- **Same platform, different enclave:**
  - If it's a newer security version of the same ISVPRODID, and the KEYPOLICY is set to MRSIGNER, then the sealed blob can be migrated.
  - More generally, the EREPORT mechanism can be used to set up a secure channel between two arbitrary enclaves on the same platform
- **Different platform, same or different enclave:**
  - Need remote attestation.

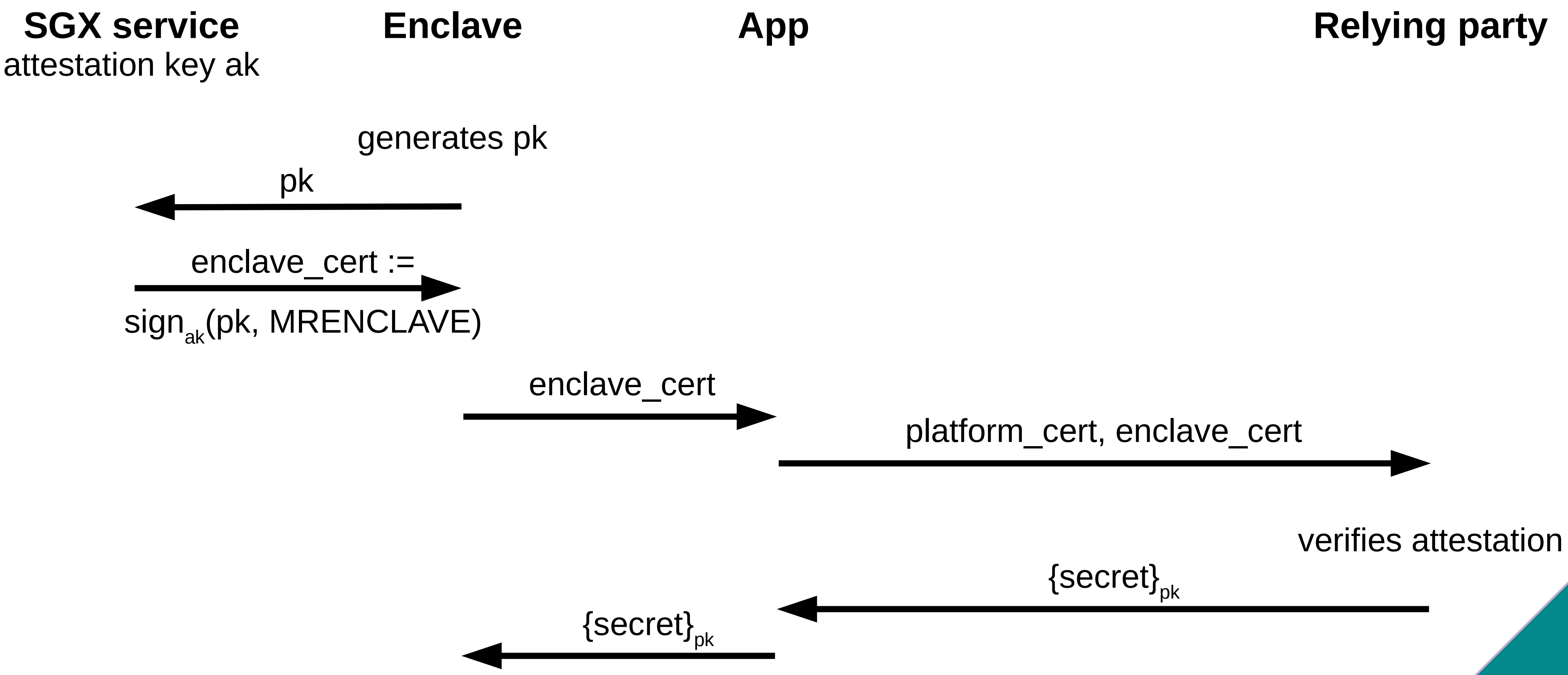
Remote attestation

- **How can a remote party know that it is talking to a given enclave?**
  - An enclave is identified by MRENCLAVE [strict] or by MRSIGNER/ISVPRODID [more flexible]

- **How can a remote party know that a given key can be used exclusively by a given enclave?**

Simple remote attestation

Platform with SGX has an “attestation” signing key ak, and
Intel has certified it : platform_cert := $sign_{Intel}$(pub(ak))

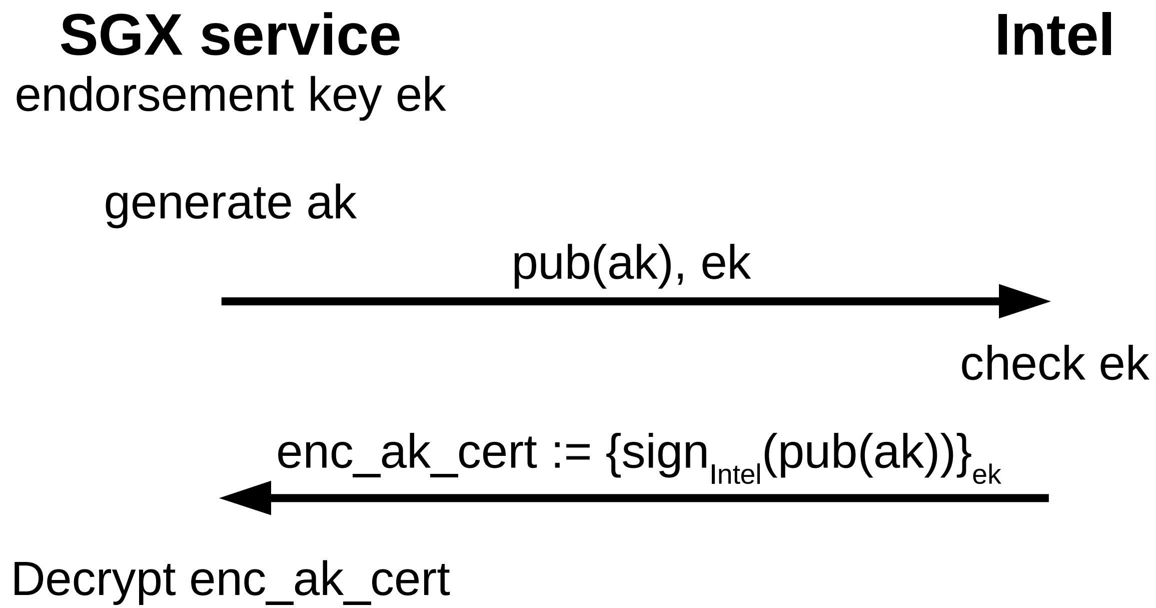


Objection 1: privacy concern

Privacy concern: not acceptable because RP can identify (using platform cert) *which* platform it is interacting with

This concern is not applicable if the attestation is that of a cloud service: cloud services do not require privacy

Solution 1: “Privacy CA” for provisioning ak



Solution 2: “Direct anonymous attestation” (DAA)

Objection 2: revocation concern

Intel would like to be able to revoke platform attestation keys if:

- Revocation based on private key:
  the private part is seen in the wild (e.g. published on the Internet), or
- Revocation based on signature:
  the key is perceived as signing erratically

Possible solutions

- Certificate revocation-list checking, or
- Short-lived certificates, that must be renewed periodically (e.g., every month)

EPID Signatures and Verification

Issuer: gpk, isk

Join: P_i obtains sk_i by interacting with issuer

Sign: $\sigma = \text{sign}_{sk_i}^{gpk,\ sigRL}(m)$; or (*if sk_i is revoked*) $\sigma = \perp$

Verify: Verify(gpk, m, PrivRL, SigRL, σ) = valid or invalid

Revoke:

- RevokePriv (gpk, ski)
  - checks sk_i, and
  - adds sk_i to PrivRL
- RevokeSig (gpk, PrivRL, m, σ)
  - verifies σ, and
  - adds σ to SigRL

Remote attestation

Provisioning the attestation key

- A ‘provisioning enclave’ uses EGETKEY to obtain a symmetric ‘provisioning key’ which Intel can also compute
- It runs the EPID join protocol with Intel (protected by the provisioning key), obtaining its attestation signing key
- It uses EGETKEY to obtain a ‘provisioning seal key’ and stores the attestation key encrypted by the provisioning seal key

Remote attestation

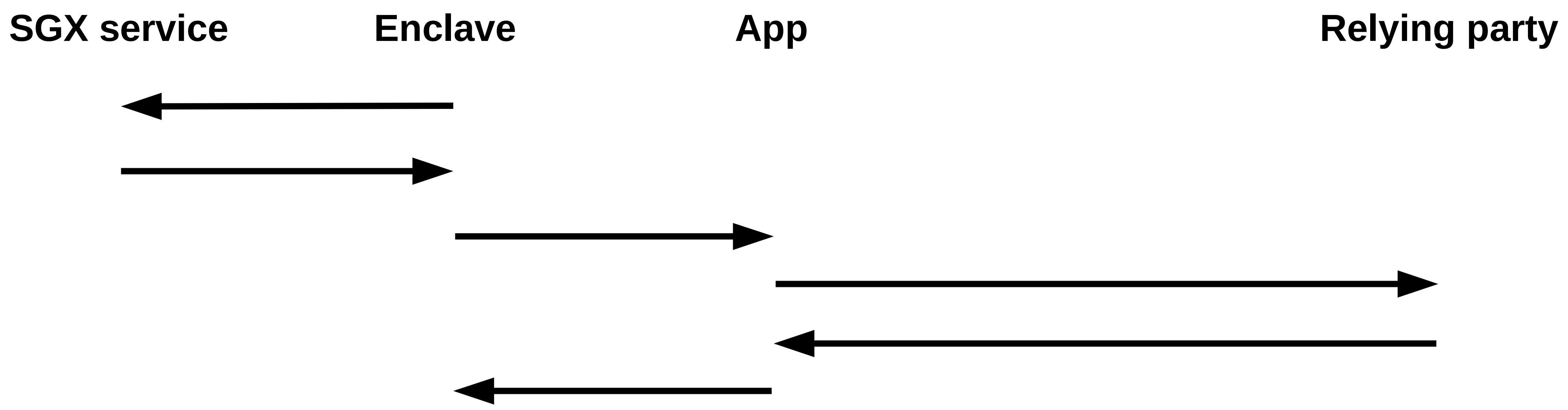
Producing a REPORT

- The attesting enclave uses EREPORT to produce a report structure, MAC’d with a report key
- The report is passed to a quoting enclave

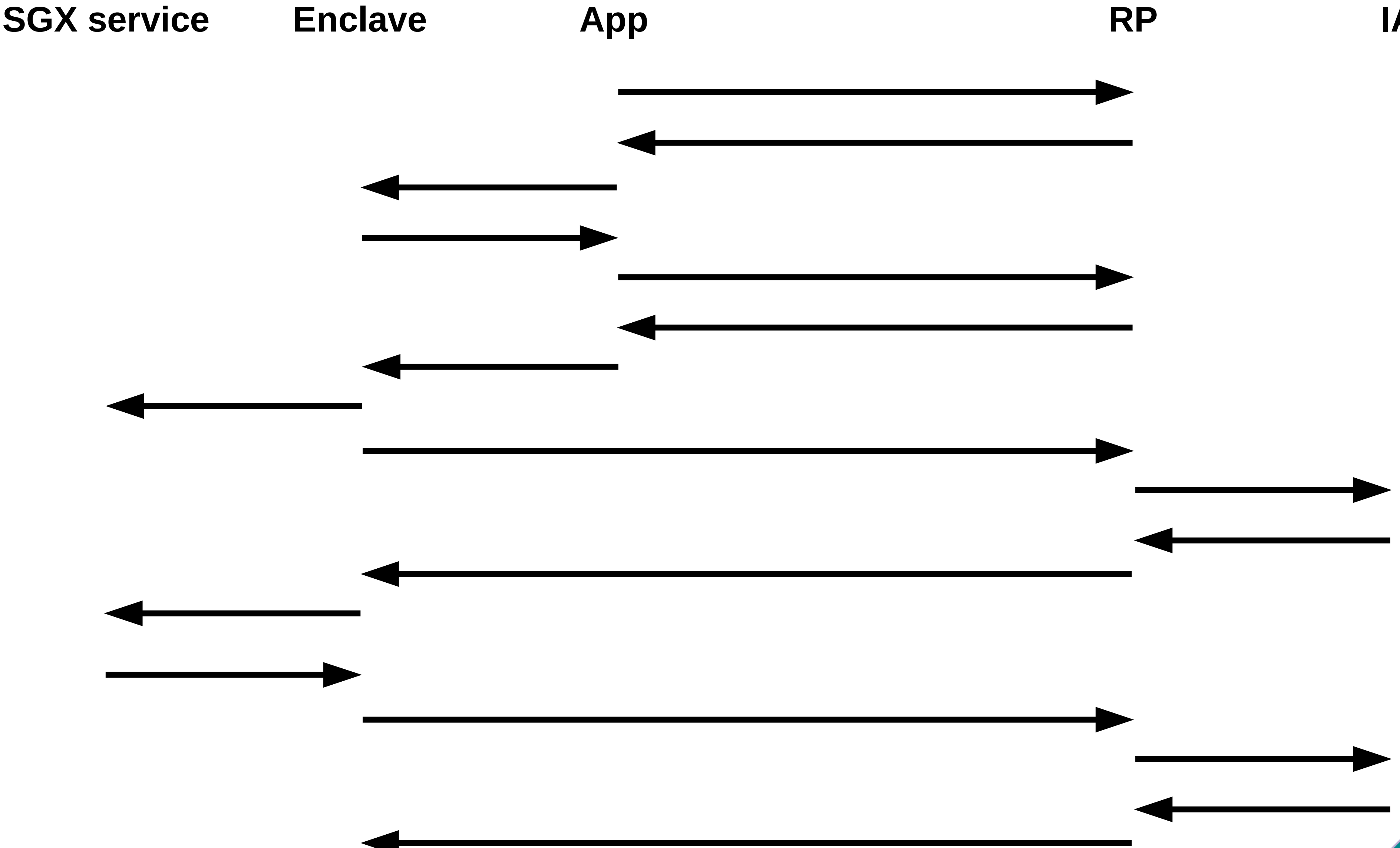
“Quoting” the report

- The quoting enclave uses EGETKEY to obtain a report key to check the report MAC
- It uses EGETKEY to obtain a provisioning seal key to decrypt the attestation key
- It uses the attestation key to sign the report (along with a received challenge)

Simple remote attestation



Intel’s remote attestation



SGX uses in research literature

S. M. Kim, J. Han, J. Ha, T. Kim, D. Han. *Enhancing Security and Privacy of Tor's Ecosystem by Using rusted Execution Environments*. USENIX NDSI, 2017.

F. Schuster, M. Costa, D. Fournet, C. Gkantsidis, M. Peinado, G. Mainar-Ruiz, M. Russinovich. *VC3: Trustworthy Data Analytics in the Cloud Using SGX*. IEEE S&P, 2015.

M. D. Ryan. *Making Decryption Accountable*. 25th Security Protocols Workshop, Springer LNCS, 2017.

> K. Severinson, M. D. Ryan, C. Johansen. *Accountable Decryption Using Intel SGX*. In preparation.
>
> - Very small, short-lived enclave (no page caching)

Conclusions

SGX: a powerful architecture for managing secret data

+ Enables processing of data that cannot be read by anyone, except for code running in the enclave

+ Minimal TCB: nothing trusted except for x86 processor

+ Not suitable for applications involving user I/O, but well suited for cloud-based applications

- Hardware and side-channel attacks

- Requires interaction with Intel at three distinct points:
   - Launch approval (by platform)
   - Join protocol to obtain attestation key (by platform)
   - Verify protocol to verify attestation (by relying party)

- Among other objections, this is privacy-invasive