Understanding Attestation: Analyzing Protocols that use Quotes

Joshua Guttman
John Ramsdell

HCSS
30 April 2019

MITRE

Protocols vs. System context

- Systems discharge protocol assumptions
- Protocols connect system parts

Protocols vs. System context

- Systems discharge protocol assumptions
- Protocols connect system parts

How can we analyze them jointly?

For instance: Building atop Intel SGX

AMD has an alternative

- SGX: security services for enclaves within user processes

  confidentiality: code, data encrypted whenever evicted
  attestation: other entities can ascertain

  - code
  - selected data esp. public key

  resident in an enclave

For instance: Building atop Intel SGX

AMD has an alternative

- SGX: security services for enclaves within user processes

  confidentiality: code, data encrypted whenever evicted
  attestation: other entities can ascertain

  - code
  - selected data esp. public key

  resident in an enclave

- This can be a big deal:

  Protect enclave secrets, allowing
  Secure channels between components running
  Known code, all
  Independent of vulnerable lower levels

  e.g. operating system unexpected hardware sysadmins

For instance: Building atop Intel SGX

AMD has an alternative

- SGX: security services for enclaves within user processes

  confidentiality: code, data encrypted whenever evicted
  attestation: other entities can ascertain

  - code
  - selected data esp. public key

  resident in an enclave

- This can be a big deal:

  Protect enclave secrets, allowing
  Secure channels between components running
  Known code, all
  Independent of vulnerable lower levels

  e.g. operating system unexpected hardware sysadmins

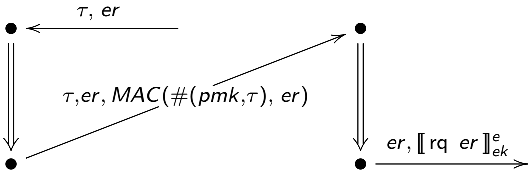
although with limitations. . .

SGX: How it provides attestation

- Enclave Record includes:
  - Enclave id
  - Hash of controlling code
  - Message, in our usage always including public key
  - Many supplementary fields
- Processor provides local enclave attestation MAC
- Quoting Enclave converts local quote to remote quote EPID
- Intel: validates EPID remote quotes online
  - ensures supply-chain origin
- Application-level enclaves prepare remote quotes via QE

SGX core roles

local-quote *epid-quote* τ



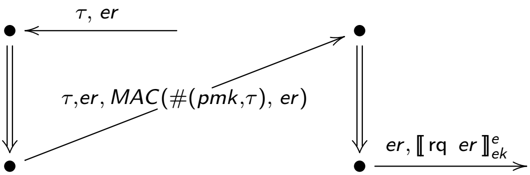
τ, er

τ,er, $MAC(\#(pmk,\tau), er)$

er, $\llbracket \mathrm{rq}\ \ er \rrbracket^{e}_{ek}$

SGX core roles

local-quote *epid-quote* τ

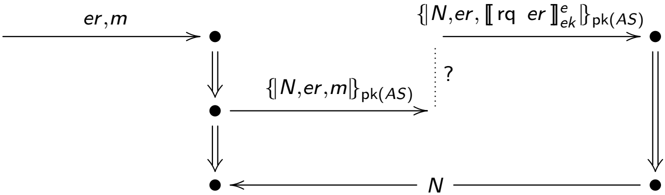


τ, er

τ,er, $MAC(\#(pmk,\tau), er)$

er, $[\![\,\mathsf{rq}\ \ er\,]\!]^{e}_{ek}$

attest-client *attest-server*



er,m

$\{\!|N,er,m|\!\}_{\mathsf{pk}(AS)}$

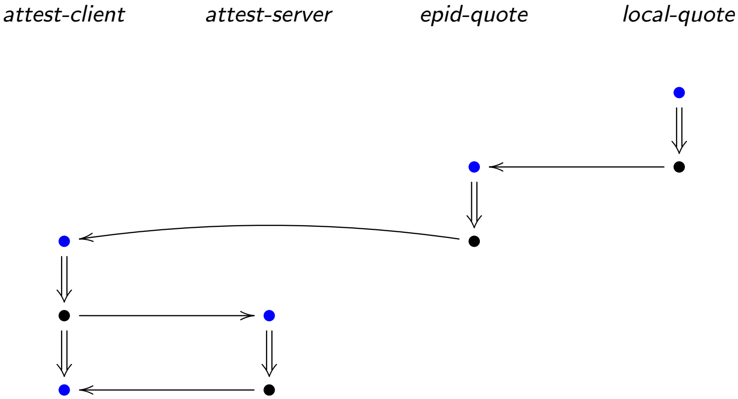
?

$\{\!|N,er,[\![\,\mathsf{rq}\ \ er\,]\!]^{e}_{ek}|\!\}_{\mathsf{pk}(AS)}$

N

SGX desired execution

If *attest-client* runs with non-compromised AS



Facts: $\mathtt{EnclCodeKey}(eid, ch, k, pmk)$ $\mathtt{ManuMadeEpid}(ek)$
Non keys: $\mathtt{Non}(\mathrm{dk}(AS))$ $\mathtt{Non}(pmk)$ $\mathtt{Non}(ek)$

CPSA: Cryptographic Protocol Shapes Analyzer

A tool for just this kind of analysis

- Explores possible executions that enrich a given scenario $\mathbb{A}_0$
  - Computes: what could have happened assuming $\mathbb{A}_0$ occurred
  - Each scenario ("skeleton") is a model i.e. structure
  - Each step $\mathbb{A} \longrightarrow \mathbb{B}$ adds information i.e. is a homomorphism
  - Search branches when different $\mathbb{B}_i$ are candidates $\mathbb{A} \longrightarrow \mathbb{B}_i$
- Enumerates models $\{\mathbb{C}_i\}_i$ that support all executions
  - If $\mathbb{D}$ is any execution such that $\mathbb{A}_0 \longrightarrow \mathbb{D}$ then $\exists i$.

$$\mathbb{A}_0 \longrightarrow \mathbb{C}_i \longrightarrow \mathbb{D}$$

- Often surprisingly few $\{\mathbb{C}_i\}_i$ needed

How CPSA works

- A reception n is realized in $\mathbb{A}$ iff for every reception n the adversary can obtain $\mathrm{msg}(n)$ from earlier transmissions
- Explores a transition relation $\mathbb{A} \rightsquigarrow \mathbb{B}$
    - $\rightsquigarrow \subseteq \mathrel{\cdot\!\!\!\longrightarrow}$
    - Each step $\mathbb{A} \rightsquigarrow \mathbb{B}$ brings some unrealized n “closer” to realized
    - A cohort

$$\begin{tikzcd} & \mathbb{B}_i \\ \mathbb{A} \arrow[ur, squiggly] \arrow[dr, squiggly] & \vdots \\ & \mathbb{B}_j \end{tikzcd}$$

covers all minimal ways to enrich some n

If $J \colon \mathbb{A} \mathrel{\cdot\!\!\!\longrightarrow} \mathbb{D}$ where $\mathbb{D}$ all realized
then J factors through some $\mathbb{A} \rightsquigarrow \mathbb{B}_i$

CPSA with rules

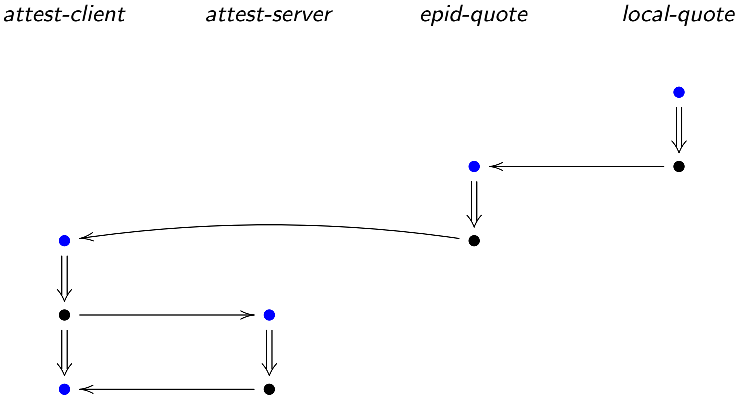
- A geometric sequent is a formula

  $$\forall\overline{x}\,.\,(\Phi \Longrightarrow \bigvee_{i} \exists\overline{y_i}\,.\,\Psi_i)$$

  where Φ, Ψ_i are conjunctions of atomic formulas
- Each geometric sequent adds persistent information
- Computes the models that are possible executions plus satisfy all the sequents

SGX desired execution

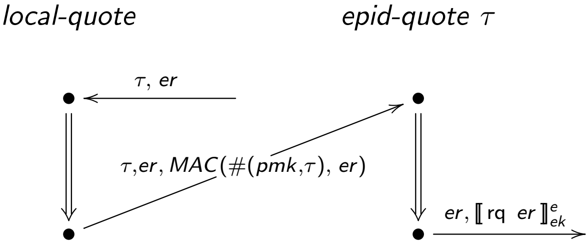
If *attest-client* runs with non-compromised AS

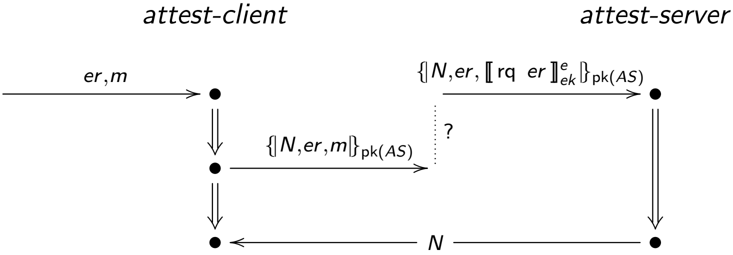


Facts: $\texttt{EnclCodeKey}(eid, ch, k, pmk)$ $\texttt{ManuMadeEpid}(ek)$

Non keys: $\texttt{Non}(\mathrm{dk}(AS))$ $\texttt{Non}(pmk)$ $\texttt{Non}(ek)$

SGX core roles


local-quote
epid-quote τ
τ, er
$\tau, er, MAC(\#(pmk,\tau), er)$
$er, [\![\, \mathrm{rq}\ \ er\,]\!]^{e}_{ek}$


attest-client
attest-server
er, m
$\{\!|N, er, m|\!\}_{\mathrm{pk}(AS)}$
$\{\!|N, er, [\![\, \mathrm{rq}\ \ er\,]\!]^{e}_{ek}|\!\}_{\mathrm{pk}(AS)}$
?
N

Rule governing local quote

Quote guarantees enclave

Rule

$$
\begin{array}{l}
\forall z\colon \textsc{strd},\quad \mathit{eid}, \mathit{ch}, \mathit{rest}\colon \textsc{mesg},\quad k : \textsc{akey},\quad \mathit{pmk}\colon \textsc{skey}\,.\\
\qquad \mathtt{LocQt}(z, 2) \quad \wedge\\
\qquad \mathtt{LocQtER}(z, \mathit{eid} :: \mathit{ch} :: k :: \mathit{rest}) \quad \wedge\\
\qquad \mathtt{LocQtPr}(z, \mathit{pmk}) \quad \wedge \quad \mathtt{Non}(\mathit{pmk})\\
\qquad \Longrightarrow\\
\qquad\qquad \mathtt{EnclCodeKey}(\mathit{eid}, \mathit{ch}, k, \mathit{pmk}).
\end{array}
$$

Three types of rules

Hardware rules summarize processor constraints

Trust rules summarize organizational standards esp. for

- delivering private keys to code
- certifying public keys

Attestation rules summarize

behavioral requirements on known code

Rule governing local quote

Quote guarantees enclave

Rule

$$
\begin{array}{l}
\forall z\colon \textsc{strd},\quad \mathit{eid},\mathit{ch},\mathit{rest}\colon \textsc{mesg},\quad k\colon \textsc{akey},\quad \mathit{pmk}\colon \textsc{skey}\,. \\
\quad \mathtt{LocQt}(z,2) \quad \wedge \\
\quad \mathtt{LocQtER}(z, \mathit{eid} :: \mathit{ch} :: k :: \mathit{rest}) \quad \wedge \\
\quad \mathtt{LocQtPr}(z, \mathit{pmk}) \quad \wedge \quad \mathtt{Non}(\mathit{pmk}) \\
\quad \Longrightarrow \\
\qquad\qquad \mathtt{EnclCodeKey}(\mathit{eid}, \mathit{ch}, k, \mathit{pmk}).
\end{array}
$$

Non-compromised keys $\texttt{Non}(K)$

A non-compromised key K has two properties

1. Only the authorized entity/ies possesses K
2. That entity uses K only in accordance with expectations
   i.e. only in accord with protocol

Non-compromised keys $\texttt{Non}(K)$

A non-compromised key K has two properties

1. Only the authorized entity/ies possesses K
2. That entity uses K only in accordance with expectations
   i.e. only in accord with protocol

(1) induces a protection requirement: hardware and upper levels

- must protect key from disclosure

(2) induces a behavioral requirement: software in control

- sends only properly prepared msgs
- sends them only in expected control flow

Rule governing attest server

AS says EPID key is manufacturer-made and non-compromised

Rule

$$
\begin{aligned}
&\forall z\colon \textsc{strd},\ K_{epid} : \textsc{akey}\,.\\
&\quad \mathtt{AttServ}(z, 2) \quad \wedge\\
&\quad \mathtt{ASQtKey}(z, K_{epid})\\
&\quad \Longrightarrow\\
&\qquad\qquad \mathtt{ManuMadeEpid}(K_{epid}) \quad \wedge \quad \mathtt{Non}(K_{epid}).
\end{aligned}
$$

Three types of rules

Hardware rules summarize processor constraints

Trust rules summarize organizational standards esp. for

- delivering private keys to code
- certifying public keys

Attestation rules summarize

behavioral requirements on known code

Attestation rule for application level code

Rule

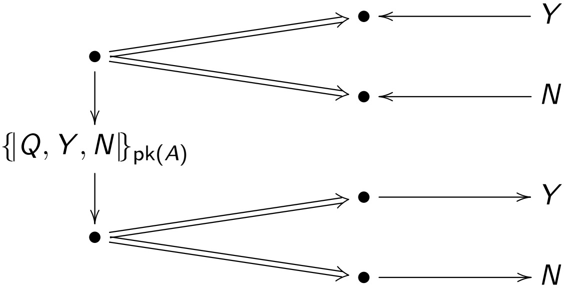
$$
\begin{array}{l}
\forall e, ch \colon \textsc{mesg},\ k \colon \textsc{akey},\ pmk \colon \textsc{skey} \,. \\
\qquad \texttt{PeerCode}(ch) \quad \wedge \\
\qquad \texttt{EnclCodeKey}(e, ch, k, pmk) \\
\Longrightarrow \\
\qquad \texttt{Non}(k^{-1})
\end{array}
$$

Induces a behavioral requirement

- Code that hashes to *ch* should:
  - Freshly generate a keypair K, K^{-1}
  - Move K enclave record
- Code that hashes to *ch* should not:
  - Disclose K^{-1}
  - Disclose computed values providing advantage on K^{-1}
- Code that hashes to *ch* should:
  - Use K^{-1} only in accordance with the protocol

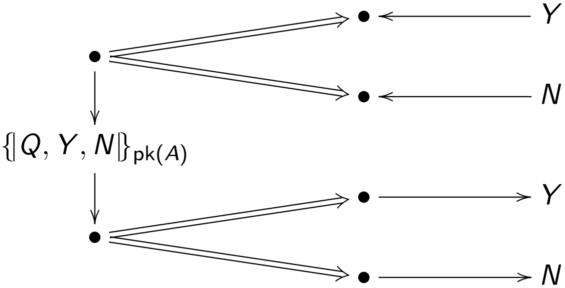
Example application protocol

Yes-or-No protocol



Example application protocol

Yes-or-No protocol

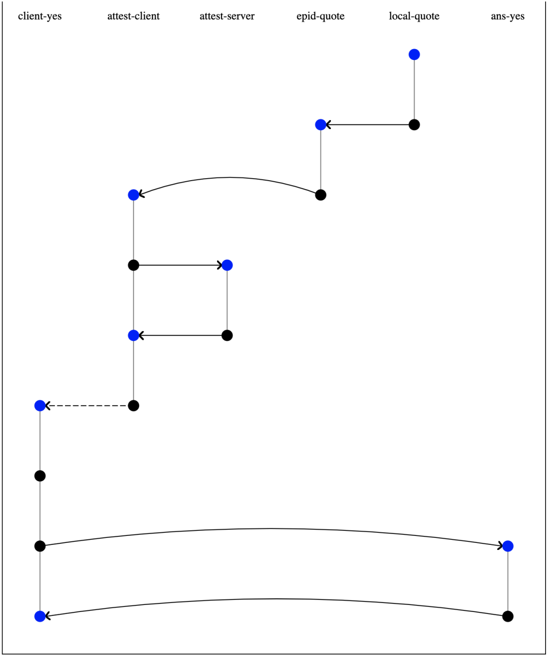


Rule

If *questioner takes step 1*

then *attest-client has run with:*

- pk(A) *in enclave record*
- *Code hash ch such that* `PeerCode`(ch)


client-yes
attest-client
attest-server
epid-quote
local-quote
ans-yes

Three types of rules

Hardware rules summarize processor constraints

Trust rules summarize organizational standards esp. for

- delivering private keys to code
- certifying public keys

Attestation rules summarize

behavioral requirements on known code

Methodology

- Protocols formalize system use of encrypted msgs
- System context: assumptions expressed as rules
  - Feed in to CPSA analysis
  - Codify requirements on components
- CPSA identifies possible executions
  - Displays runs of protocol under assumptions
  - Guides protocol/rule refinement