ArmoredSoftware: Trust in the cloud

Annual Demonstration

Dr. Perry Alexander, Dr. Andrew Gill, Dr. Prasad Kulkarni,
Adam Petz, Paul Kline, Justin Dawson, Jason Gevargizian,
Leon Searl, Edward Komp

Information and Telecommunication Technology Center
Electrical Engineering and Computer Science
The University of Kansas
palexand@ku.edu,andygill@ku.edu,prasadk@ku.edu

January 15, 2015

Outline

Program Goals

Trust in the Cloud

Provide new capabilities that establish and maintain trustworthy cloud-based application deployment

- Establish trust among cloud components
  - trust among cohorts of processes
  - trust among processes and environment
- Promote informed decision making
  - data confidentiality can be confirmed
  - execution and data integrity can be confirmed
- Autonomous run-time response and reconfiguration
  - responds to attack, failure, reconfiguration, and repair
  - response varies based on measurement

Delivery Platform

Open source, standards compliant

- Lightweight integration with existing cloud infrastructure
  - OpenStack cloud infrastructure
  - Xen+XSM VM infrastructure
  - Fedora, HotSpot JVM, GHC
- Trusted Computing Group standards compliant
  - Trusted Platform Module 1.2
  - TCG vTPM (in principle)
  - Trusted OS infrastructure
- Standard communication mechanisms
  - JSON structures for all exchanged data
  - *vchan* for on-platform communication
  - TCP/IP for off-platform communication

New Technologies

- Trustworthy protocol execution
  - executable protocol representation
  - protocol execution generates evidence of trustworthiness
  - highly focused protocols
  - strand space formal semantics
- Application specific measurement
  - managed and traditional execution environments
  - compile-time assistance for measurer synthesis
  - specialized measurement bundled with applications
- Attestation driven cloud application and data management
  - health monitoring
  - problem mitigation
  - application migration
  - access control

Research & Development Plan

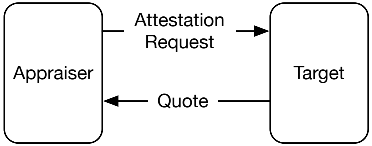
- Development and integrate measurement capabilities
  - hosted languages (Java)
  - traditional compiled languages (C, C++)
  - integrate with environment measurers (Xen,OpenStack,OS)
- Develop attestation capabilities
  - flexible, user configurable protocol representation
  - measured protocol execution
  - protocol execution appraisal
- Develop infrastructure trust argument
  - develop lightweight vTPM infrastructure supporting mobility
  - launch from known roots of trust
  - maintain trust evidence at run time
  - maintain trust over migration

Research & Development Plan

- Automated synthesis and verification
  - measurer synthesis at application compile time
  - automated evidence appraisal from protocols
  - formal trust argument
- Demonstrations
  - initial simple infrastructure demonstrations
  - cloud-based “big data” environment demonstration
  - federated trust demonstration
  - *demonstrations as discovered/directed*
- Scale up and roll out
  - integration with Xen, OpenStack, Linux
  - installation management and packaging
  - effective web presence

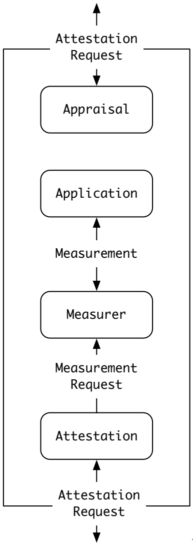
Semantic Remote Attestation

- Appraiser requests a quote
  - specifies needed information
  - provides a nonce
- Target gathers evidence
  - measures application
  - gathers evidence of trust
- Target generates quote
  - measurements and evidence
  - original nonce
  - cryptographic signature
- Appraiser assesses quote
  - good application behavior
  - infrastructure trustworthiness

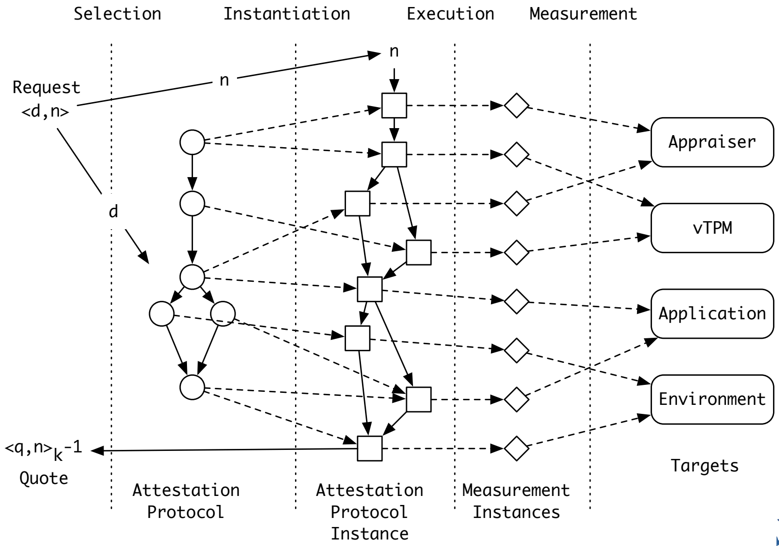


Armored Application Architecture

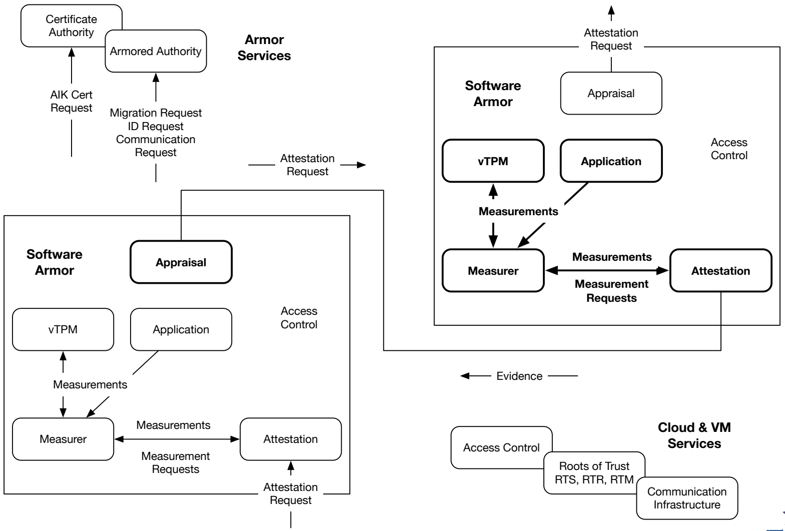
- Focus is user-space applications
- Assesses the cloud infrastructure and environment
- Attests to the state of its application
- High-assurance, lightweight infrastructure
- Influenced by the *Trusted Research Platform* and *Principles of Remote Attestation*



Measurement and Attestation



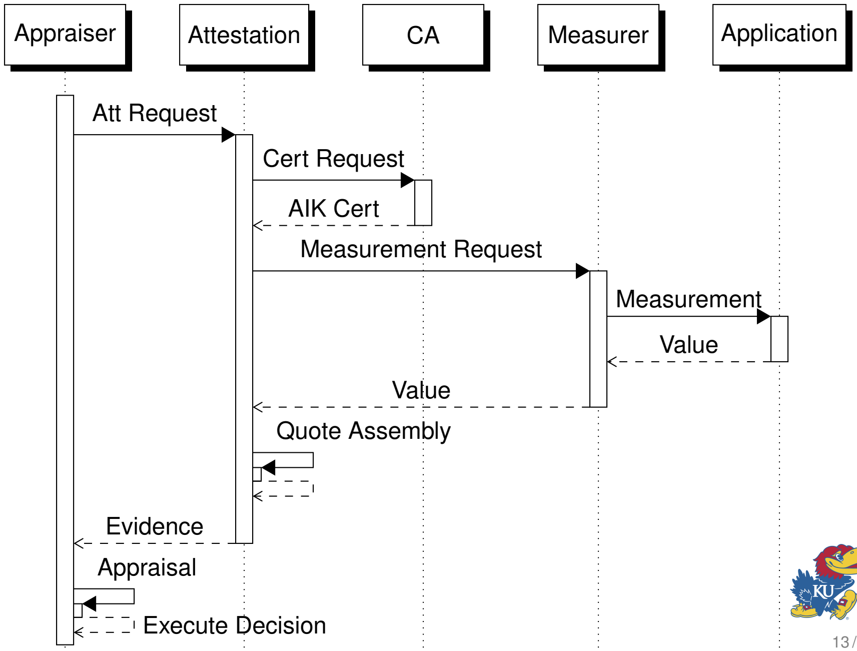
System-Level Architecture



What We Are Demonstrating

- Execution of a CA-based Attestation Protocol
  - Attestation request
  - Protocol execution
  - Evidence appraisal
- Major architectural subsystems
  - Appraiser
  - Attestation Manager
  - Measurer
  - Instrumented JVM
  - vTPM and Certificate Authority
- Anomaly Detection
  - Bad signatures and PCRs
  - Bad CA certificates
  - Bad quotes and AIKs
  - Bad measurements

Abstract CA-Based Attestation Protocol



Message List Representation

$App \rightarrow Att : d, N_{\mathrm{App}}, PCR_{\mathrm{m}}$ on C_{AppAtt}

$Att \rightarrow TPM : make\_and\_load\_identity$ on C_{AttTPM}

$TPM \rightarrow Att : AIK^{+}, AIK_{h}$ on C_{TPMAtt}

$Att \rightarrow CA : Att, AIK^{+}$ on C_{AttCA}

$CA \rightarrow Att : \{K, |AIK|\}_{EK^{+}}, \{[AIK^{+}]_{CA^{-}}\}_{K^{+}}$ on C_{CAAtt}

$Att \rightarrow TPM : activate\_identity(AIK_{h}, |AIK|)$ on C_{AttTPM}

$TPM \rightarrow Att : K$ on C_{TPMAtt}

$Att \rightarrow Meas : d$ on $C_{AttMeas}$

$Meas \rightarrow Att : e$ on $C_{MeasAtt}$

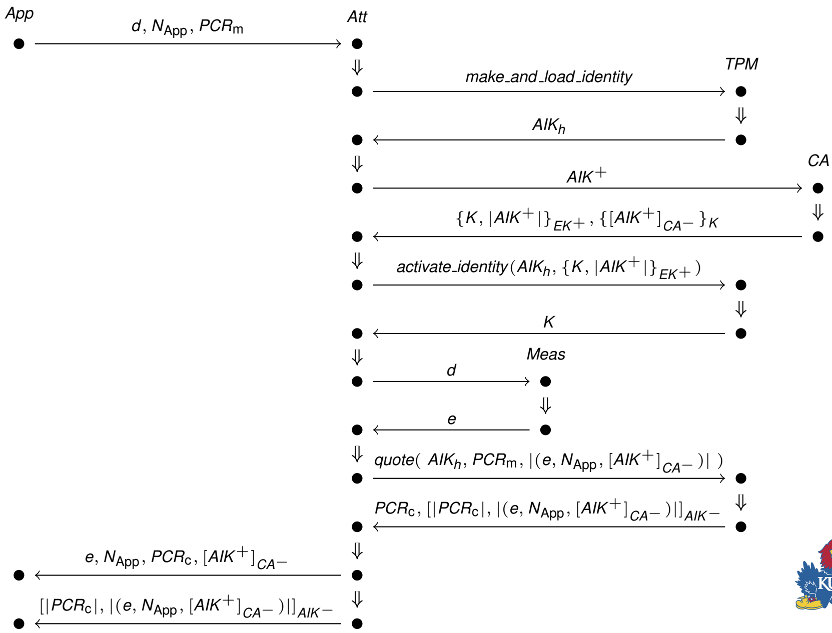
$Att \rightarrow TPM : quote(\ AIK_{\mathrm{h}}, PCR_{\mathrm{m}}, |(e, N_{\mathrm{App}}, [AIK^{+}]_{CA^{-}})|\ )$ on C_{AttTPM}

$TPM \rightarrow Att : PCR_{\mathrm{c}}, [|PCR_{\mathrm{c}}|, |(e, N_{\mathrm{App}}, [AIK^{+}]_{CA^{-}})|]_{AIK^{-}}$ on C_{TPMAtt}

$Att \rightarrow App : e, N_{\mathrm{App}}, PCR_{\mathrm{c}}, [AIK^{+}]_{CA^{-}}$ on C_{AttApp}

$Att \rightarrow App : [|PCR_{\mathrm{c}}|, |(e, N_{\mathrm{App}}, [AIK^{+}]_{CA^{-}})|]_{AIK^{-}}$ on C_{AttApp}

Strand Space Diagram Representation



App
Att
$d, N_{\text{App}}, PCR_{\text{m}}$
TPM
make_and_load_identity
AIK_h
CA
AIK^+
$\{K, |AIK^+|\}_{EK+}, \{[AIK^+]_{CA-}\}_K$
$activate\_identity(AIK_h, \{K, |AIK^+|\}_{EK+})$
K
Meas
d
e
$quote(\ AIK_h, PCR_{\text{m}}, |(e, N_{\text{App}}, [AIK^+]_{CA-})|\ )$
$PCR_{\text{c}}, [|PCR_{\text{c}}|, |(e, N_{\text{App}}, [AIK^+]_{CA-})|]_{AIK-}$
$e, N_{\text{App}}, PCR_{\text{c}}, [AIK^+]_{CA-}$
$[|PCR_{\text{c}}|, |(e, N_{\text{App}}, [AIK^+]_{CA-})|]_{AIK-}$

Attestation Request

$$\begin{tikzcd} \mathit{App} \arrow[r, "{d, N_{\mathrm{App}}, \mathit{PCR}_{\mathrm{m}}}"] & \mathit{Att} \end{tikzcd}$$

- Initiate with an attestation request
  - d abstractly defines desired evidence
  - N_{App} is the appraiser’s nonce
  - $\mathit{PCR}_{\mathrm{m}}$ selects PCRs
- Attestation agent selects and executes protocol based on request

Generating and Certifying an AIK

- Request a new *AIK* from TPM (optional)
- Receive *AIK* handle
- Request AIK^+ signed by CA (*AIK* cert)
- Receive *AIK* cert encrypted with session key *K*
- Receive *K* encrypted with public *EK*

$$\begin{tikzcd}
Att & & \\
\bullet \arrow[d, Rightarrow] & & \\
\bullet \arrow[r, "make\_and\_load\_identity"] & \overset{TPM}{\bullet} \arrow[d, Rightarrow] & \\
\bullet \arrow[d, Rightarrow] & \bullet \arrow[l, "AIK_h"] & \\
\bullet \arrow[rr, "AIK^+"] & & \overset{CA}{\bullet} \arrow[d, Rightarrow] \\
\bullet & & \bullet \arrow[ll, "\{K, |AIK^+|\}_{EK^+}, \{[AIK^+]_{CA^-}\}_K"]
\end{tikzcd}$$

Activating the AIK

$$\begin{tikzcd}
Att & & TPM \\
\bullet \arrow[rr, "activate\_identity(AIK_h, \{K, |AIK^+|\}_{EK^+})"] & & \bullet \arrow[d, Rightarrow] \\
\bullet & & \bullet \arrow[ll, "K"']
\end{tikzcd}$$

- Request TPM decryption of the *AIK* cert
- Receive *K* used to decrypt signed public *AIK*
- Only TPM can gain access to *K*
- Only TPM can obtain signed, public *AIK*
- Oddly, No manipulation of the *AIK* in this “activation” process

Measurement

$$\begin{tikzcd}
\overset{Att}{\bullet} \arrow[rr, "d"] & & \overset{Meas}{\bullet} \arrow[d, Rightarrow] \\
\bullet & & \bullet \arrow[ll, "e"']
\end{tikzcd}$$

- Request information from measurer
- Receive evidence *e* from measurer
- *d* is abstract allowing protocol reuse
- Most protocols make many requests of the measurer

Generating a Quote

$$\begin{tikzcd}
\overset{\mathit{Att}}{\bullet} \arrow[r, "quote(\ AIK_h, PCR_\mathrm{m}, |(e, N_\mathrm{App}, [AIK^+]_{CA^-})|\ )"] & \overset{\mathit{TPM}}{\bullet} \arrow[d, Rightarrow] \\
\bullet & \bullet \arrow[l, "PCR_\mathrm{c}, [|PCR_\mathrm{c}|, |(e, N_\mathrm{App}, [AIK^+]_{CA^-})|]_{AIK^-}"']
\end{tikzcd}$$

- Request a quote from the TPM
  - AIK identifies the signing AIK
  - PCR_m identifies desired PCRs
  - $|(e, N_\mathrm{App}, [AIK^+]_{CA^-})|$ guarantees integrity of returned evidence
- Receive quote from TPM
  - PCR_c is PCR composite built from requested PCRs
  - $[|PCR_\mathrm{c}|, |(e, N_\mathrm{App}, [AIK^+]_{CA^-})|]_{AIK^-}$ is the signed quote

Appraisal

$$\begin{tikzcd}
\mathit{App} & & \mathit{Att} \\
\bullet & & \bullet \arrow[ll, "{e, N_{\mathrm{App}}, PCR_{\mathrm{c}}, [AIK^{+}]_{CA^{-}}}"'] \arrow[d, Rightarrow] \\
\bullet & & \bullet \arrow[ll, "{[|PCR_{\mathrm{c}}|, |(e, N_{\mathrm{App}}, [AIK^{+}]_{CA^{-}})|]_{AIK^{-}}}"']
\end{tikzcd}$$

- Receive evidence from the attestation manager
  - evidence
  - original nonce
  - PCR composite
  - signed AIK^{+}
- Receive TPM quote from the attestation manager
  - hash of all evidence
  - PCR composite
  - signed by AIK^{-}
- Evaluate evidence and quote

Evidence Appraisal

Demonstration detects failure of all aspects of attestation

$$e, N_{\mathrm{App}}, PCR_{\mathrm{c}}, [AIK^{+}]_{CA^{-}}$$

- e – evidence gathered from running application
- N_{App} – prevents replay
- PCR_{c} – evidence in the form or PCR data from the vTPM
- $[AIK^{+}]_{CA^{-}}$ – ensures validity of AIK^{+}

$$[|PCR_{\mathrm{c}}|, |(e, N_{\mathrm{App}}, [AIK^{+}]_{CA^{-}})|]_{AIK^{-}}$$

- PCR_{c} – hash ensures integrity of PCR data
- $|(e, N_{\mathrm{App}}, [AIK^{+}]_{CA^{-}})|$ – hash ensures integrity of evidence, nonce, and signed AIK^{+}

Implementation

```
  Appraiser Main

main = do
  putStrLn "START main of Appraiser\n"
  (pcrSelect, nonce) ←
mkTPMRequest ([0..23]::[Word8])
  let mReq = mkMeasureReq [0..2]
      req = (Request mReq pcrSelect nonce)
  putStrLn $ show req
  putStrLn $ "Press enter to send Request"
  getChar
  chan ← sendRequest req
  putStrLn "\nSENT REQUEST TO ATTESTATION A..."
  putStrLn "\nRECEIVING RESPONSE...\n"
  result ← receiveResponse chan
  case (result) of
        (Left err) →
          putStrLn "Error getting response..."
        (Right response) → do
          putStrLn $ "Received: "
          putStrLn $ show response ++ "\n"
          putStrLn "Evaluating Response: "
          result ← evaluate req response
          showDemoEvalResult result

  putStrLn "END main of Appraiser"
```

```
  Attestation Agent Main

main = do
  putStrLn "START main of Attestation"

  apprChan ← server_init appId
  measChan ← client_init meaId

  publicEK ← tpm_key_pubek tpm
  tkShn ← tpm_session_oiap tpm
  tpm_takeownership tpm tkShn publicEK ownerPas
  putStrLn "\nTPM OWNERSHIP TAKEN\n"

  req ← receiveRequest apprChan
  resp ← mkResponse req
  sendResponse apprChan resp

   where
     ownerPass = tpm_digest_pass oPass
     srkPass = tpm_digest_pass sPass
```

Implementation

```
mkResponse :: Either String Request → Att Response
mkResponse (Right Request desiredE pcrSelect nonce) = do
  enterP "request AIK from TPM"
  x@(iKeyHandle, iSig) ← liftIO $ createAndLoadIdentKey
  liftIO $ putStrLn "AIK CREATED AND LOADED. "
  liftIO $ putStrLn $ "Handle: " ++ show iKeyHandle ++ "\n"

  caChan ← getPriChan
  pubAIK ← liftIO $ attGetPubKey iKeyHandle aikPass
  let caRequest = mkCARequest aikPass pubAIK iSig
  liftIO $ putStrLn $ show caRequest
  enterP "send CA Request:"
  r@(CAResponse caCertBytes actIdInput) ← liftIO $
                                          converseWithScottyCA caRequest
  liftIO $ putStrLn $ "SENT CA REQUEST"
  liftIO $ putStrLn "\nRECEIVING CA RESPONSE... \n"
  liftIO $ putStrLn $ "Received: " ++ show r

  iShn ← liftIO $ tpm_session_oiap tpm
  oShn ← liftIO $ tpm_session_oiap tpm
  enterP $ "release session key K by calling tpm_activate_identity" ++ ...
  sessionKey ← liftIO $ tpm_activateidentity tpm iShn oShn iKeyHandle
                        aikPass ownerPass actIdInput
  liftIO $ putStrLn $ "Released K: " ++ ...
  let keyBytes = tpmSymmetricData sessionKey
      decryptedCertBytes = decryptCTR aes ctr (toStrict caCertBytes)
      caCert = (decode decryptedCertBytes) :: CACertificate

  meaChan ← getMeaChan
  evidenceList ← liftIO $ mapM (getEvidencePiece meaChan) desiredE

  let evBlob = ePack evidenceList qNonce caCert
      evBlobSha1 = bytestringDigest $ sha1 evBlob
  enterP $ "call tpm_quote with arguments:\n" ++ ...
  quote ← liftIO $ mkQuote iKeyHandle aikPass pcrSelect evBlobSha1
  evPack = (EvidencePackage evidenceList qNonce eSig)
  liftIO $ tpm_flushspecific tpm qKeyHandle tpm_rt_key    Evict Loaded key

  return (Response evPack caCert quote)

 where
   aikPass = tpm_digest_pass "i"
   ownerPass = tpm_digest_pass oPass
```

Implementation

```
tpm_loadkey2 :: TPM tpm => tpm → Session → TPM_KEY_HANDLE → TPM_KEY →
                           TPM_DIGEST → IO TPM_KEY_HANDLE
tpm_loadkey2 tpm (OIAP ah en) parent key pass = do
    on ← nonce_create
    (rtag,size,resl,dat) ← tpm_transmit tpm 45 tag cod (dat on)
    let (handle,dat') = splitAt 4 dat
    return $ decode handle
    where tag = tpm_tag_rqu_auth1_command
          cod = tpm_ord_loadkey2
          dat on = concat [ encode parent, encode key, ah, encode on
                          , encode False, encode (ath on) ]
          ath on = tpm_auth_hmac pass en on 0 $ concat [encode cod, encode key]
```

Implementation

```
data TPM_KEY = TPM_KEY {
      tpmKeyUsage    :: TPM_KEY_USAGE
    , tpmKeyFlags    :: TPM_KEY_FLAGS
    , tpmKeyAuth     :: TPM_AUTH_DATA_USAGE
    , tpmKeyParams   :: TPM_KEY_PARMS
    , tpmKeyPcrInfo  :: ByteString
    , tpmKeyPublic   :: TPM_STORE_PUBKEY
    , tpmKeyEncData  :: ByteString
    } deriving (Eq)

data Session =
  OIAP
  { authHandle :: ByteString
  , nonceEven  :: TPM_NONCE }

  | OSAP
    { authHandle    :: ByteString
    , nonceOddOSAP  :: TPM_NONCE
    , nonceEven     :: TPM_NONCE
    , nonceEvenOSAP :: TPM_NONCE
    , secret        :: TPM_DIGEST
    } deriving (Eq)
```

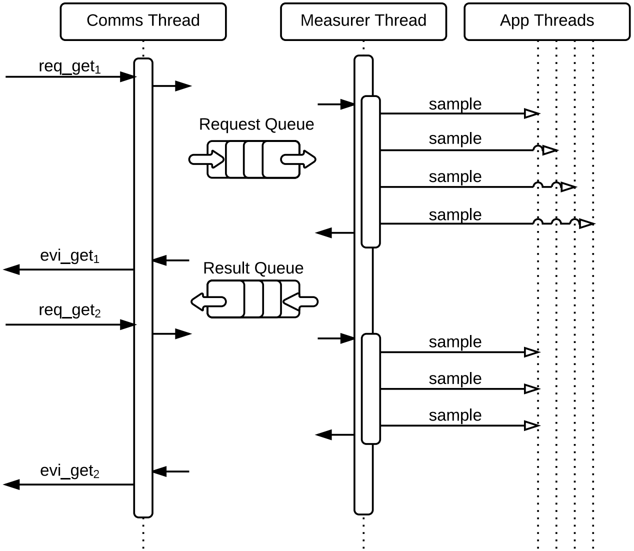
```
data TPM_QUOTE_INFO =
  TPM_QUOTE_INFO
  { tpmQuoteInfoCompHash :: TPM_COMPOSITE_HASH
   ,tpmQuoteInfoExData   :: TPM_NONCE
    } deriving (Show, Eq)

instance Binary TPM_QUOTE_INFO where
    put (TPM_QUOTE_INFO c d) = do
        put tpm_struct_ver_default
        put tpm_quote_info_fixed
        put c
        put d
    get = do
        t  ←  (get :: Get TPM_STRUCT_VER)
        f  ←  getLazyByteString (fromIntegral
        c ← get
        d ← get
        return $ TPM_QUOTE_INFO c d
```

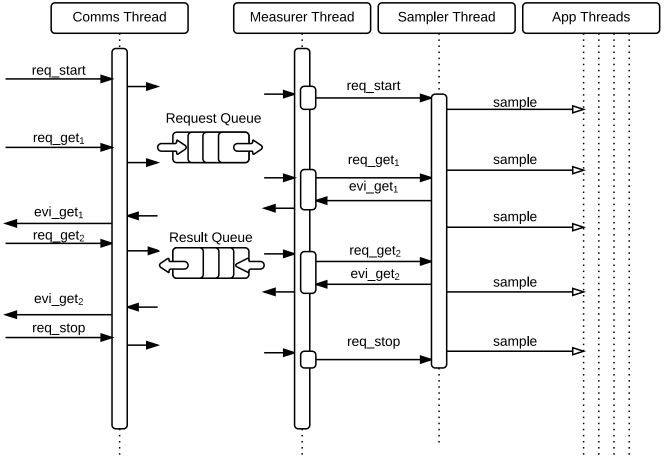
Measurement Basics

- Measurement Mechanisms
  - Sampling: Measurer spawns asynchronous sampler thread to take measurements in parallel with application threads
  - Instrumentation: Measurer inserts/toggles code instrumentation in the application threads to take measurements
- Measurement Components
  - Measurer Thread services individual measurements, dispatches sampler threads, and toggles measurement instrumentation
  - Communications Thread communicates between the Attester (Vchan) and the Measurer Thread (within run-time)
- Measurement Request Types
  - individual measurements: get evidence
  - continuous measurements: initiate, terminate, get evidence

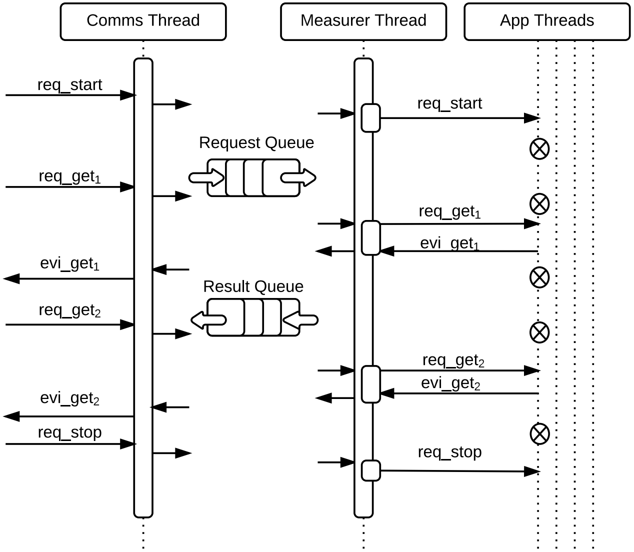
Measurer - Individual Sample



Measurer - Continuous Sampling



Measurer - Continuous Instrumentation



Measurement Status

- Current Demo
  - Individual measurement of user variable, specified externally
  - Performed by the measurer thread by monitoring the stack or heap contents
- Static Analysis to define Application Expectations
  - Use static analysis to generate ”golden” expectations for applications. Run-time measurements should fall within these defined expectations.
  - E.g. Generate a static callgraph and provide this an the ”golden” bounds to the appraiser. Track the dynamic callgraph during execution. Send the dynamic callgraph to the appraiser when requested to ensure the program never breaks the static expectation.

2-3 Slides on Communication Mechanisms

CA communication

Shared notion of AIKCertRequest, AIKCert, and CAResponse JSON structures.

Attester

- creates an AIKCertRequest (containing attester ID, AIK^+) and converts to JSON
- JSON sent as POST request to CA running as web server

Certificate Authority

- POST body bytes $\rightarrow$ UTF8 $\rightarrow$ JSON $\rightarrow$ AIKCertRequest
- looks up EK^+ associated with ID in sql database
- AIKCert $=$ AIK^+ signed with CA^-
- generates key K and encrypts with EK^+
- AIKCert encrypted with K
- both wrapped in a CAResponse, converted to JSON and sent as response.

CA communication continued

Properties

- CA only responds to receiving an $AIKCertRequest_{JSON}$
- The CACert can *only* be decrypted by knowing K (and therefore EK^{-})

Appraiser Knowledge after receiving Cert:

- signature on AIK ensures it was CA who generated signature

  +
- only an entity knowing EK^{-} could decrypt and send the CACert

  =
- **Attester is using a registered TPM**

Current Status

Completed four demonstrations culminating in running an attestation protocol in response to an attestation request.

- Attestation and Appraisal development
  - CA-Based attestation protocol execution example
  - integration with Berlios TPM 1.2 emulator
  - simple dynamic appraisal of attestation results
- Measurement development
  - on demand Java program measurement
  - HotSpot-based Java VM run time measurements
  - standard mechanism for extending measurement capabilities
- Communication infrastructure
  - vchan, TCP/IP and socket communication infrastructure
  - language-based interface with TPM 1.2
  - JSON-based data exchange formats
  - initial certificate authority API

Goals and Milestones for 2015

- Push to the cloud
  - integration with OpenStack
  - migration across Xen instances
  - vTPM function migration
- Establish roots-of-trust and trust argument
  - measured launch and remeasurement of ArmoredSoftware
  - establish trust in the Xen/OpenStack infrastructure
- Executable protocol representation and protocol semantics
  - richer protocol collection
  - evidence of proper execution
  - protocol-centered appraisal
- Operational, integrated vTPM prototype
  - integration with TPM 1.2
  - find and integrate, not build (we hope)

Goals and Milestones for 2015

- More robust communication and system services
  - Armor Authority prototype
  - Certificate Authority integration
  - communications management
- More capable measurement
  - compiler directed measurement
  - continuous measurement of trends
- More interesting download-able demonstration
  - sponsor-defined problem
  - more realistic attacker model

Questions and Guidance

- What should we be watching and integrating with?
  - operational vTPM infrastructure
  - infrastructure measured boot
- What demonstration problems are relevant?
  - federated trust
  - trust in the infrastructure
  - trust among application collections
- What would convince you to work a problem with us?

References