1 Demo4 Diagrams

1.1 Message Sequence Diagram

$A \rightarrow B : d, N_{\mathrm{A}}, PCR_{\mathrm{m}}$ on C_{AB}

$B \rightarrow T : make\_and\_load\_identity$ on C_{BT}

$T \rightarrow B : AIK^{+}, AIK_{\mathrm{h}}$ on C_{TB}

$B \rightarrow C : B, AIK^{+}$ on C_{BC}

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

$B \rightarrow T : activate\_identity(AIK_{\mathrm{h}}, |AIK|)$ on C_{BT}

$T \rightarrow B : K$ on C_{TB}

$B \rightarrow M : d$ on C_{BM}

$M \rightarrow B : e$ on C_{MB}

$B \rightarrow T : quote(\ AIK_{\mathrm{h}}, PCR_{\mathrm{m}}, |(e, N_{\mathrm{A}}, [AIK^{+}]_{CA^{-}})|\ )$ on C_{BT}

$T \rightarrow B : PCR_{\mathrm{c}}, [|PCR_{\mathrm{c}}|, |(e, N_{\mathrm{A}}, [AIK^{+}]_{CA^{-}})|]_{AIK^{-}}$ on C_{TB}

$B \rightarrow A : e, N_{\mathrm{A}}, PCR_{\mathrm{c}}, [AIK^{+}]_{CA^{-}}$ on C_{BA}

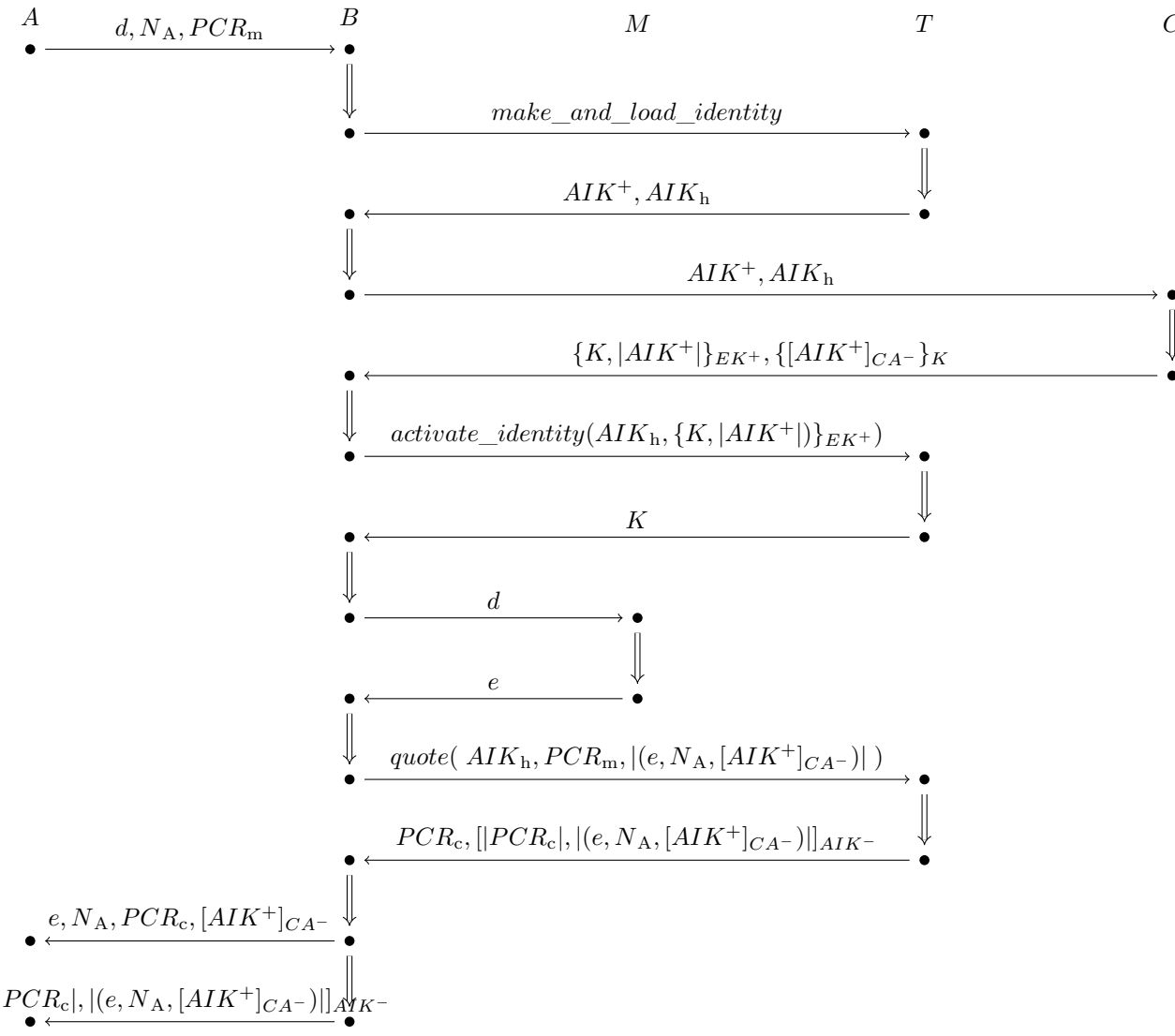
$B \rightarrow A : [|PCR_{\mathrm{c}}|, |(e, N_{\mathrm{A}}, [AIK^{+}]_{CA^{-}})|]_{AIK^{-}}$ on C_{BA}

KEY

A:	Appraiser
B:	Attestation Agent
T:	TPM
C:	Certificate Authority
M:	Measurer
d :	desired evidence
e :	gathered evidence
N_{A} :	nonce generated by A
PCR_{m}:	pcr mask indicating desired pcr registers
PCR_{c}:	pcr composite structure containing select pcr register values
AIK_{h}:	AIK key handle(used by TPM to reference loaded keys)
K:	Session key created by C

©The University of Kansas, 2013

1.2 Strand Space Diagram



Appraisal

```
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 AGENT...\n"
  putStrLn "\nRECEIVING RESPONSE...\n"
  result ← receiveResponse chan
  case (result) of
       (Left err) →
         putStrLn "Error getting response. Error was: " ++ err
       (Right response) → do
         putStrLn $ "Received: "
         putStrLn $ show response ++ "\n"
         putStrLn "Evaluating Response: "
         result ← evaluate req response
         showDemoEvalResult result

  putStrLn "END main of Appraiser"
```

Attestation

```
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 ownerPass srkPass
  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
```

```
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
```