# Important directories under mysteryhill

rpcore/

core components of RPCore services and integration with OpenStack

rpclient/

Code for the “Attestation Agent” running inside a VM, to wait for attestation challenge (e.g. from MtW), contact RPCore to retrieve a quote of the VM, and construct and send back the response to the challenge

build/

scripts for RPCore package compilation

install/

installation scripts for XEN/KVM compute node setup

docs/

mainly the setup instructions, please read mysteryhill/docs/setup/README as a starting point

ManifestTool/

The client tool for VM image preparation (encryption, manifest file creation, and encryption VM image and manifest file uploading)

Other directories can be ignored (e.g. information in Setup-Standardization is obsolete already, please refer to docs/ for the most recent setup instructions)

# Important directories under mysteryhill/rpcore/

src/

All source code.

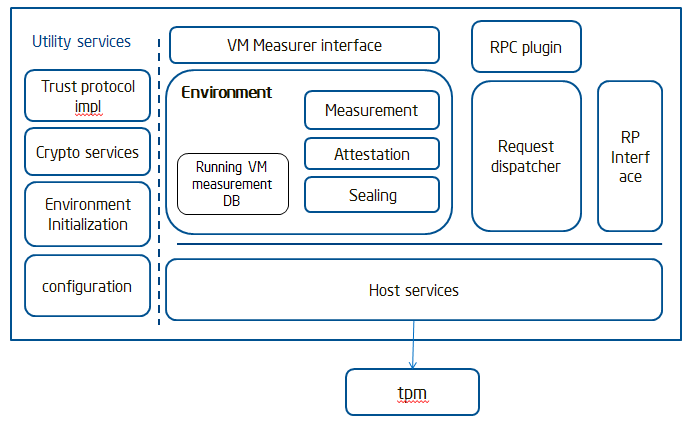
rptmp/

Configuration files for RPCore. This directory needs to be copied to /tmp/, and for each

machine, it needs to be reconfigured if tpm version of RPCore is to be used . See RPCore installation and configuration doc.

# Important directories under mysteryhill/rpcore/src

Makefile: the main makefile to compile all RPCore components.



RPCore components architecture

fileProxy/

This directory contains code from the Cloud Proxy project. Majority of code is written by John Manferdelli, some are from Open Source project. The subdirectory that is used in RPCore is fileProxy/Code. For example,

* fileProxy/Code/commonCode/: common utility functions, such as tinyXML, logging utilities, etc
* fileProxy/Code/cryptUtility: cryptography utility functions

For source code in fileProxy/ that are used by RPCore, they are defined in makefiles for RPCore. See rpcore/src/Makefile for all makefiles used in the project.

Imvm/

Code for the component that implements “verifier”, which does integrity verification of VM images. As of current implementation, verifier is an independent process running in OS, it can be extended to run inside a VM.

rpapps/

Makefiles for several RP applications, the most useful one is keyNegoServer.mak, which is to make KSS, a MtW equivalent service for RPCore enrollment.

rpchannel/

Code for supporting XmlRPC calls between RPCore and clients that request RPCore services, for encoding/decoding of XmlRPC calls, for implementing XmlRPC calls over TCP, etc.

rpcore/

The core components of RPCore services.

* modtcService.cpp: “main” function of RPCore, execution starts here
* nontpmrpcorsvc-g.mak: makefile for non-tpm version of RPCore
* tcsd-rpcoresvc-g.mak: makefile of tpm version of RPCore, where communication with TPM is through tcsd
* tao/
  + taoInit.cpp: initialization of RPCore, triggered when RPCore starts and the configuration directory /tmp/rptmp/config/TrustedOS is empty. The initialization process will create RSA key pairs that RPCore will use later for signing purpose (referred as RPCore’s policy key), and get the public key certified by MtW. As of current implementation, we use KSS, a MtW equivalent service to certify the key.  
    start kss from /tmp/rptmp/config  
    run rpcore (e.g. nontpm version): nontpmrpcore –initKeys  
    run titest from rpcore/bin/debug/  
    (To run titest and rpcore, you also need to set env:

export LD\_LIBRARY\_PATH= YOUR\_WORKDIR/mysteryhill/rpcore/lib/:$LD\_LIBRARY\_PATH)

* + taoEnvironment.cpp -- setup and run taoEnvironment, which provides API interfaces to upper stack (Apps measured launched by RPCore), to support requests from App for “quote, seal, unseal,” etc.
  + taoHostServices.cpp – middle layer for taoEnvironment to communicate with “host”, the hosting OS and platform that RPCore is running on, for instance, to retrieve host measurement, retrieve certificate, evidence, get TPM’s pcr values, request TPM to seal/unseal, etc.
  + TPMHostsupport.cpp – interface to communicate with TPM, through tcsd. (used by tcsd-rpcoresvc)
  + emuTPMHostsupport.cpp – emulate TPM functions, for nontpmrpcorsvc-g.
  + RP\_RSAKey-helper.cpp – utility functions to create openssl RSA key pairs, and convert them to populate RP’s RSA data structure. The purpose is to be able use openssl library to create standard format of CSR request for public key certification.

rpcrypto/

Basic cryptography utility functions, such as Big Number implementation, base64 encoding, decoding, etc.

rpproxy/

Implementation of RP proxy (for KVM setup) as in the design document.

rpquote/

sample code for requesting RPCore services, e.g. request for quotes of hosted programs (e.g. VM) such as GenHostedComponentQuote().

rptrust/

supporting functions for requesting RPCore services. See rpquote for how they are called.

rptest/

Some testing applications for debugging/testing RPCore services. For instance, taoinittest/ has the code of titest, which is a component emulating the role of Trust Agent in the process of RPCore keys provisioning.

rptools/

some utility tools, especially crypt utitlity tools

rptpm/

Code for interacting with /emulating TPM functions.

* TPMtcsd: code to interface with tcsd and call TPM fucntions (used for tpm version of RPCore)
* TPMDirect: code to emulate TPM functions (used for nontpmrpcore)
* rpmmio: our own simple device driver to interact directly with TPM, used by initrd during TCB-protection boot process