User Manual

April 25, 2022

- **common:**
  - **cert.go**
    - **RsaPublicKeyToPemBytes(*rsa.PublicKey) ([]byte, error):**
      - Marshal RSA public key to bytes.
    - **PemBytesToRsaPublicKey([]byte) (*rsa.PublicKey, error):**
      - Unmarshal bytes to RSA public key.
    - **X509CertFromFile(string) (*x509.Certificate, error):**
      - Read x509 cert from file.
    - **LoadRSAKeyPairFromFile(string) (*rsa.PrivateKey, error):**
      - Load rsa key pair from file.
  - **crypto.go**
    - **SignStrucRSASHA256(interface, *rsa.PrivateKey) ([]byte, error):**
      - Generate a signature using SHA256 and RSA.
    - **RCSRCreateSignature(*rsa.PrivateKey, *RCSR) error:**
      - Generate a signature, and fill the signature in the RCSR(Root Certificate Signing Request).
    - **RCSRGenerateRPCSignature(*RCSR, *rsa.PrivateKey) error:**
      - Sign the RCSR using the previous RPC (Root Policy Certificate), and fill in the signature in RCSR.
    - **RCSRVerifySignature(*RCSR) error:**
      - Verify the signature of RCSR using the public key in hash. Called by CA to check whether the public key is correct.
    - **RCSRVerifyRPCSIgnature(*RCSR, rpc *RPC) error:**
      - Verify the RCSR using RPC; verify the RPC signature.
    - **RCSRGenerateRPC(rcsr *RCSR, notBefore time.Time, serialNumber int, caPrivKey *rsa.PrivateKey, caName string) (*RPC, error):**
      - Called by PCA. Sign the RCSR and generate RPC.
    - **RPCVerifyCASignature(caCert *x509.Certificate, rpc *RPC) error:**
      - Called by domain owner, check whether CA signature is correct
  - **json.go**
    - **JsonStrucToFile(struc interface, filePath string) error :**
      - Marshal structure to bytes, and store them in a file
    - **JsonStrucToBytes(struc interface) ([]byte, error):**
      - Json struc to bytes.

        - **JsonBytesToSPT(sptBytes []byte) (*SPT, error):**
            - [-] Sign Policy Certificate (SPT)
        - **JsonBytesToRPC(rpcBytes []byte) (*RPC, error)**
        - **JsonBytesToPoI(poiBytes []byte) (*trillian.Proof, error)**
        - **JsonBytesToLogRoot(logRootBytes []byte) (*types.LogRootV1, error):**
            - Bytes to Trillian log root.
        - **JsonBytesToProof(proofBytes []byte) (*trillian.Proof, error):** Bytes to Trillian proof.
        - **JsonFileToRPC(struc *RPC, filePath string) error:**
            - Read RPC in Json format from file.
        - **JsonFileToSPT(struc *SPT, filePath string) error:**
            - Read SPT in Json format from file.
    - **structure.go**
        - **RCSR** Root Certificate Signing Request
        - **RPC** Root Policy Certificate
        - **SPT** Signed Policy Timestamp
        - **SPRT** Signed Policy Revocation Timestamp

- **domainowner**

    - **NewDomainOwner(subject string) *DomainOwner**
        - Return a new domain owner for one domain.
    - **(do *DomainOwner) GenerateRCSR(domainName string, version int) (*common.RCSR, error)**
        - Generate a RCSR for RPC.

- **logverifier**

    - **NewLogVerifier(hasher merkle.LogHasher) *LogVerifier**
        - Return a new log verifier.
    - **(logVerifier *LogVerifier) HashLeaf(input []byte) []byte**
        - Hash one leaf using the hasher.
    - **((c *LogVerifier) VerifyInclusionWithPrevLogRoot(trusted *types.LogRootV1, newRoot *types.LogRootV1, consistency [][]byte, leafHash []byte, proof []*trillian.Proof) error**
        - This function verify the leaf using an old log root(tree head)
    - **(c *LogVerifier) VerifyRoot(trusted *types.LogRootV1, newRoot *types.LogRootV1, consistency [][]byte) (*types.LogRootV1, error)**
        - VerifyRoot verifies that newRoot is a valid append-only operation from trusted root. If trusted.TreeSize is zero, a consistency proof is not needed.
    - **(c *LogVerifier) VerifyInclusionByHash(trusted *types.LogRootV1, leafHash []byte, proofs []*trillian.Proof) error**
        - VerifyInclusionByHash verifies that the inclusion proof for the given Merkle leafHash matches the given trusted root.

- **policylog**

    - **client**
        - **PLGetAdminClient(configPath string) (*PLAdminClient, error)**

    - Get a admin client for the policy log. Admin client can create a tree.
  - **(client PLAdminClient) CreateNewTree() (*trillian.Tree, error)**
    - Create a new tree.
  - **SaveAdminClientConfigToFile(config *AdminClientConfig, configPath string) error**
    - Save admin client configure to file.
  - **ReadAdminClientConfigFromFile(config *AdminClientConfig, filePath string) error**
    - Read admin client configure from file.
  - **PLNewLogClient(configPath string, treeId int64) (*PLLogClient, error)**
    - Get a log client. Log client can add leave to log, or fetch proof inclusion from the log.
  - **(c *PLLogClient) SetTreeId(treeID int64)**
    - Set the treeID of the log client. One log client will interact with one specific tree identified by treeID.
  - **(c *PLLogClient) AddLeaves(ctx context.Context, data [][]byte) *AddLeavesResult**
    - Add leaves to the tree.
  - **(c *PLLogClient) FetchInclusions(ctx context.Context, leavesData [][]byte) *FetchInclusionResult**
    - Fetch inclusion proof of a leaf.
  - **(c *PLLogClient) FetchInclusions(ctx context.Context, leavesData [][]byte) *FetchInclusionResult**
    - Fetch inclusion proof of a leaf.
  - **(c *PLLogClient) GetCurrentLogRoot(ctx context.Context) (*types.LogRootV1, error)**
    - Get current log root of the target tree.
  - **(c *PLLogClient) UpdateTreeSize(ctx context.Context) error**
    - Update the tree size of the target tree.
  - **(c *PLLogClient) GetConsistencyProof(ctx context.Context, trusted *types.LogRootV1, newRoot *types.LogRootV1) ([][]byte, error)**
    - Get consistency proof between two log root.
  - **(c *PLLogClient) QueueRPCs(ctx context.Context, fileNames []string) (*QueueRPCResult, error)**
    - Queue a batch of RPC into the log, then return the inclusion proof of every newly-added leaves.
  - **(c *PLLogClient) FetchInclusions(ctx context.Context, leavesData [][]byte) *FetchInclusionResult**
    - Fetch inclusion proof of a leaf.
  - **SaveLogClientConfigToFile(config *PLLogClientConfig, configPath string) error**
    - Save log client config to file.
  - **ReadLogClientConfigFromFile(config *PLLogClientConfig, filePath string) error**
    - Load log client config from file.
- **server**
  - **PLCreateLogServer(configPath string)**
    - Get a new log server from the config file.
  - **PLCreateLogSigner(configPath string)**

        - Get a new log signer from the config file.

- **pca**
    - **NewPCA(configPath string) (*PCA, error)**
        - Return a new PCA.
    - **(pca *PCA) SignAndLogRCSR(rcsr *common.RCSR) error**
        - Sign the rcsr and generate a rpc -¿ store the rpc to the ”fileExchange” folder; policy log will fetch rpc from the folder.
    - **(pca *PCA) ReceiveSPTFromPolicyLog() error**
        - When policy log returns SPT, this func will be called this func will read the SPTs from the file, and process them.
    - **(pca *PCA) GetValidRPCByDomain(domainName string) (*common.RPC, error)**
        - Return the new RPC with SPT.