CoinEx DEX 验证节点部署保护投票私钥

longcpp @ 20191105

CoinEx DEX的验证节点部署需要妥善保护用来投票的Ed25519的私钥,一旦被窃取的私钥被用来双签则根据DEX公链规则会通过扣除质押的CET代币总量的一定比例来惩罚验证节点运营方,造成可观的经济损失. 私钥的保护典型的策略是将私钥保存在更安全的计算环境中(可以是位于内网环境中的一台安全主机, 也可以更进一步用硬件安全模块(Hardware Security, 记为HSM)提供更为完善的密钥保护机制), 验证节点在需要签名投票时,向保护该私钥的计算环境请求签名服务. 下文中用远程签名服务来指代这一私钥保护策略, Tendermint项目开发了专门用于远程签名服务的tmkms工具, 而tmkms支持3种签名实现方式, `softsign` 不依赖任何HSM而是在分离的主机上通过软件实现Ed25519签名服务, `yubihsm` 则通过对YubiHSM 2来提供签名服务,而 `ledgertm` 则通过对Ledger的支持实现签名服务. 其中 `yubihsm` 以及 `ledger` 两种模式由于对HSM的采纳能够更好的保护签名私钥,而tmkms工具中对YubiHSM 2的支持更为完善. 因此本文介绍 `softsign` 和 `yubihsm` 两种模式的远程签名服务部署方式.

tmkms是用Rust语言实现的配合Tendermint应用的密钥管理服务,项目的整体目标是为签名服务提供: 1) 高可用性,通过直接访问验证者的公式签名私钥提供服务; 2) 防止双签, 通过维持最新的区块高度等信息提供预防双签的功能; 3) 支持硬件安全模块, 由HSM管理密钥的整个生命周期,在主机沦陷的情况下保证私钥不会被窃取. tmkms同时支持纯软件形式(没有HSM)的运行模式,在生产环境中不推荐是中纯软件[softsign]模式的签名服务,目前看来支持最好的应该是YubiHSM的HSM,生产环境中推荐使用 `yubihsm` 的模式.

使用tmkms提供的远程签名服务时, DEX的验证节点会打开一个特定的端口等待远程签名服务提供者通过该端口与验证节点建立连接,连接建立之后当验证节点参与共识过程需要投票签名时,将需要签名的消息通过该链接发送给远程签名服务提供者,签好之后将签名信息发送给验证节点. 在配置tmkms和cetd以实现remote signer功能时, tmkms 和 cetd 上需要配置一样的IP地址和端口号, 而tmkms节点上则需要配置相应的私钥. 另外为了方便检测remote signer的是否能够正常运行, Tendermint项目还额外提供了 `tm-signer-harness` 工具用来测试remote signer提供的功能的正确性.接下来展示 `softsign` 和 `yubihsm` 两种模式的部署和配置.

纯软件模式的远程签名服务

可以直接从源代码编译安装tmkms,步骤如下. 通过指定 `--features=softsign` 表示采用纯软件形式的密钥管理和签名服务,也可以指定 `--features=yubihsm` 表示支持YubiHSM 2的HSM,另外也可以指定 `--features=ledgertm` 表示支持Ledger的HSM.

```
→  Downloads git clone https://github.com/tendermint/kms.git && cd kms
[...] # 如果确实,需要安装依赖的软件包, Rust, libusb等
$ cargo build --release --features=softsign
```

也可以通过Rust的cargo工具进行安装,同样需要指定要的密钥管理和签名服务形式: `[softsign||ledgertm||yubihsm]`:

```
→  ~ cargo install tmkms --features=softsign
```

编译安装之后,在 `$HOME/.cargo/bin/$` 文件夹下会新增可执行文件 `tmkms` .通过

```
tmkms start
```

命令可以启动密钥签名服务.服务所需的配置文件默认是当前工作目录下的 `tmkms.toml` 文件,也可以通过 `-c` 选项指定配置文件

```
tmkms start -c /path/to/tmkms.toml
```

与Coinex Chain的DEX的测试网适配的采用纯软件模式[softsign]的签名服务的配置文件 `tmkms.toml` 内容如下:

```
[[chain]]
id = "coinexdex-test1"
key_format = { type = "bech32", account_key_prefix = "coinexpub",
consensus_key_prefix = "coinexvalconspub" }

# Validator configuration
[[validator]]
addr = "tcp://127.0.0.1:26659"
chain_id = "coinexdex-test1"
reconnect = true # true is the default
secret_key = "./secret_connection.key"

# Provider configuration
[[providers.softsign]]
chain_ids = ["coinexdex-test1"]
path = "./signing.key"
```

`[[chain]]` 小节下是所适配的区块链的信息, `id` 为 `chain_id` , `key_format` 表示链的地址格式信息, `bech32` 编码, `account_key_prefix` 和 `consensus_key_prefix` 表示两种密钥的前缀信息.

`[[validator]]` 表示验证者的相关信息, `addr` 表示验证节点的网络连接信息, `chain_id` 与 `[[chain]]` 小节中的 `id` 一致, `seckey_key` 指定了存放与验证节点建立安全连接所需密钥的文件(注意不是用于共识过程投票的密钥),这个密钥可以由tmkms工具生成:

```
➜  kms git:(master) ✗ tmkms keygen secret_connection.key
03:08:31 [INFO] Wrote random Ed25519 private key to secret_connection.key
```

`[providers.softsign]` 是关于签名服务的信息,仅采用纯软件的形式时需要提供 `[providers.softsign]` 小节的信息.如果采用YuhiHSM 2,则需要填写 `[providers.yubihsm]` . 相应的, Ledger的HSM则对应 `[providers.ledgertm]` ,Tendermint KMS中的 `tmkms.toml.example` 有模板. `chain_ids` 表示签名私钥可以用于的链的id信息,与前面两个小节相对应. `path` 则指明了存储密钥的文件, 这个是验证节点的共识私钥. 出于dex的测试的目的,执行下面的命令之后,

```
➜  dex git:(master) ✗ ./scripts/setup_single_testing_node.sh
```

会生成目录 `~/.cetd` :

```
➜  dex git:(master) ✗ tree ~/.cetd
/Users/long/.cetd
├── config
|   ├── cetd.toml
|   ├── config.toml
|   ├── genesis.json
|   ├── gentx
|   |   └── gentx-050724c2e304ee9abd1b3555144ece17d3ade604.json
|   ├── node_key.json
|   └── priv_validator_key.json
└── data
    └── priv_validator_state.json

3 directories, 7 files
```

其中 `priv_validator_key.json` 存储了用于共识过程签名的私钥,为了生成tmkms所需要的密钥文件,执行命令:

```
➜  kms git:(master) ✗ tm-signer-harness extract_key -tmhome ~/.cetd -output
./signing.key
```

其中命令行工具 `tm-signer-harndess` 是Tendermint项目的 `tools` 提供的工具,说明文档在这里,编译安装时首先进入tendermint项目的 `tendermint/tools/tm-signer-harndess` 目录下,然后执行以下命令会在 `$GOPATH$/bin` 下新增可执行文件 `tm-signer-harndess` .

```
make
make install
```

`tm-signer-harndess` 工具主要用来测试远程签名服务的功能.测试之前,首先要启动 `tmkms` 服务:

```
→  kms git:(master) ✗ tmkms start -c tmkms.toml
```

启动 tm-signer-harndess 测试命令:

```
→  ~ tm-signer-harness run -addr tcp://127.0.0.1:26657 -tmhome ~/.cetd
```

测试运行结果展示如下:

```
➜ kms git:(master) ✗ tmkms start -c tmkms.toml
05:10:03 [INFO] tmkms 0.5.0 starting up...
05:10:03 [INFO] [keyring:softsign:coinexdex-test1,coinexdex-test1] added validator key 1zcjduepqp5lhxd
jhx6n9mfvz7c5ved87wdn2gays4njeyzrqu9rk2nx3r9xsmmhvha
05:10:03 [INFO] KMS node ID: 079AE9C9881217FD9E999CFD238A8D266B0B4975
05:10:03 [ERROR] [coinexdex-test1@tcp://127.0.0.1:26657] I/O error: Connection refused (os error 61)
05:10:04 [INFO] KMS node ID: 079AE9C9881217FD9E999CFD238A8D266B0B4975
05:10:04 [ERROR] [coinexdex-test1@tcp://127.0.0.1:26657] I/O error: Connection refused (os error 61)
05:10:05 [INFO] KMS node ID: 079AE9C9881217FD9E999CFD238A8D266B0B4975
05:10:05 [WARN] [coinexdex-test1] 127.0.0.1:26657: unverified validator peer ID! (F27978E5EFB57102C340
2F6394EC1F129A79D218)
05:10:05 [INFO] [coinexdex-test1@tcp://127.0.0.1:26657] connected to validator successfully
05:10:05 [ERROR] [coinexdex-test1@tcp://127.0.0.1:26657] I/O error: failed to fill whole buffer
05:10:06 [INFO] KMS node ID: 079AE9C9881217FD9E999CFD238A8D266B0B4975
05:10:06 [ERROR] [coinexdex-test1@tcp://127.0.0.1:26657] I/O error: Connection refused (os error 61)
^C05:10:07 [INFO] Waiting for client threads to stop...
05:10:07 [INFO] [coinexdex-test1@tcp://127.0.0.1:26657] shutdown request received
➜ kms git:(master) ✗
```

```
➜ ~ tm-signer-harness run -addr tcp://127.0.0.1:26657 -tmhome ~/.cetd
I[2019-07-18|13:10:05.141] Loading private validator configuration      keyFile=/Users/long/.cetd/conf
ig/priv_validator_key.json stateFile=/Users/long/.cetd/data/priv_validator_state.json
I[2019-07-18|13:10:05.144] Loading chain ID from genesis file           genesisFile=/Users/long/.cetd/
config/genesis.json
I[2019-07-18|13:10:05.145] Loaded genesis file                          chainID=coinexdex-test1
I[2019-07-18|13:10:05.145] Listening at                                 proto=tcp addr=127.0.0.1:26657
I[2019-07-18|13:10:05.145] Resolved TCP address for listener            addr=127.0.0.1:26657
I[2019-07-18|13:10:05.146] Starting test harness
I[2019-07-18|13:10:05.146] Attempting to accept incoming connection     acceptRetries=100
I[2019-07-18|13:10:05.146] Starting SignerValidatorEndpoint             impl=SignerValidatorEndpoint
I[2019-07-18|13:10:05.689] TEST: Public key of remote signer
I[2019-07-18|13:10:05.689] Local                                        pubKey=PubKeyEd25519{0D3F73365
736A65DA582F628CCB4FE7366A47490ACE5920860E147654CD1194D}
I[2019-07-18|13:10:05.689] Remote                                       pubKey=PubKeyEd25519{0D3F73365
736A65DA582F628CCB4FE7366A47490ACE5920860E147654CD1194D}
I[2019-07-18|13:10:05.689] TEST: Signing of proposals
I[2019-07-18|13:10:05.691] Successfully validated proposal signature
I[2019-07-18|13:10:05.691] TEST: Signing of votes
I[2019-07-18|13:10:05.691] Testing vote type                            type=1
I[2019-07-18|13:10:05.693] Successfully validated vote signature        type=1
I[2019-07-18|13:10:05.693] Testing vote type                            type=2
I[2019-07-18|13:10:05.694] Successfully validated vote signature        type=2
I[2019-07-18|13:10:05.694] SUCCESS! All tests passed.
I[2019-07-18|13:10:05.694] Stopping SignerValidatorEndpoint             impl=SignerValidatorEndpoint
➜ ~
```

可以看到 `tm-singer-harness` 的所有测试均能通过.注意到在 `tmkms` 执行完成后,工作目录下会产生一个新文件 `coinexdex-test1_priv_validator_state.json` ,文件中存储的是对应的区块链的状态, `tmkms` 利用其中存储的状态信息来防止双签错误.

```
➜  kms git:(master) ✗ cat coinexdex-test1_priv_validator_state.json
{"height":101,"round":0,"step":6,"block_id":"D04B98F48E8F8BCC15C6AE5AC05080
1CD6DCFD428FB5F9E65C4E16E7807340FA"}%
```

在单节点的测试网络,可以修改 `~/.cetd/config/config.toml` 来实现远程签名服务配置:

```
# TCP or UNIX socket address for Tendermint to listen on for
# connections from an external PrivValidator process
priv_validator_laddr = "tcp://127.0.0.1:26657"
```

配置保存后,在不启动 `tmkms` 服务的条件下启动 `cetd` 服务,会发现启动失败:

```
➜ dex git:(master) ✗ ./cetd start
I[2019-07-18|14:26:07.459] Starting ABCI with Tendermint                module=main
E[2019-07-18|14:26:10.723] OnStart                                      module=privva
l err="accept tcp 127.0.0.1:26657: i/o timeout"
ERROR: Error with private validator socket client: failed to start private validator:
 accept tcp 127.0.0.1:26657: i/o timeout
➜ dex git:(master) ✗
```

失败原因是无法链接签名服务,说明remote signer配置已经开始发挥作用.接下来首先启动 `tmkms` 服务,然后再启动 `cetd` 服务

```
➜ kms git:(master) ✗ tmkms start -c tmkms.toml
06:37:51 [INFO] tmkms 0.5.0 starting up...
06:37:51 [INFO] [keyring:softsign:coinexdex-test1,coinexdex-test1] added validator key 1zcjduepqp5lh
xdjhx6n9mfvz7c5ved87wdn2gays4njeyzrqu9rk2nx3r9xsmmhvha
06:37:51 [INFO] KMS node ID: 079AE9C9881217FD9E999CFD238A8D266B0B4975
06:37:51 [ERROR] [coinexdex-test1@tcp://127.0.0.1:26657] I/O error: Connection refused (os error 61)
06:37:52 [INFO] KMS node ID: 079AE9C9881217FD9E999CFD238A8D266B0B4975
06:37:52 [ERROR] [coinexdex-test1@tcp://127.0.0.1:26657] I/O error: Connection refused (os error 61)
06:37:53 [INFO] KMS node ID: 079AE9C9881217FD9E999CFD238A8D266B0B4975
06:37:53 [WARN] [coinexdex-test1] 127.0.0.1:26657: unverified validator peer ID! (CB68E9EAF537B2E914
E7E4751495C8B0334E69B3)
06:37:53 [INFO] [coinexdex-test1@tcp://127.0.0.1:26657] connected to validator successfully
06:38:04 [ERROR] [coinexdex-test1@tcp://127.0.0.1:26657] I/O error: failed to fill whole buffer
06:38:05 [INFO] KMS node ID: 079AE9C9881217FD9E999CFD238A8D266B0B4975
06:38:05 [ERROR] [coinexdex-test1@tcp://127.0.0.1:26657] I/O error: Connection refused (os error 61)
^C06:38:05 [INFO] Waiting for client threads to stop...
06:38:06 [INFO] [coinexdex-test1@tcp://127.0.0.1:26657] shutdown request received
```

```
➜ dex git:(master) ✗ ./cetd start
I[2019-07-18|14:37:53.114] Starting ABCI with Tendermint                module=main
E[2019-07-18|14:37:53.964] Couldn't connect to any seeds                module=p2p
I[2019-07-18|14:37:56.169] Executed block                               module=state height=1 validT
xs=0 invalidTxs=0
I[2019-07-18|14:37:56.189] Committed state                              module=state height=1 txs=0
appHash=644DB063EF2E4EC5093B79F3796BBF597989E121C59DC90B50CAE48FAFC4A35E
I[2019-07-18|14:37:58.364] Executed block                               module=state height=2 validT
xs=0 invalidTxs=0
I[2019-07-18|14:37:58.383] Committed state                              module=state height=2 txs=0
appHash=3BBFFE0184F9A20CA42C85F721B6622F2F4E71F93F828009A0CDFF586891F4BD
I[2019-07-18|14:38:00.562] Executed block                               module=state height=3 validT
xs=0 invalidTxs=0
I[2019-07-18|14:38:00.578] Committed state                              module=state height=3 txs=0
appHash=A6E89F89C6C8AEFBCFBADA18F0F35C772DF8AF9D5185AE7EE7EF3A3908A30172
I[2019-07-18|14:38:02.758] Executed block                               module=state height=4 validT
xs=0 invalidTxs=0
I[2019-07-18|14:38:02.777] Committed state                              module=state height=4 txs=0
appHash=9A4C3F848893A8D980F5F6B18EA275CB3E604FCFCE3BEEF45A1FB170EC5A346D
^C                                                                      ➜ ↛↛↛
➜ dex git:(master) ✗
```

中断两个进程之后,查看文件 coinexdex-test1_priv_validator_state.json :

```
➜  kms git:(master) ✗ cat coinexdex-test1_priv_validator_state.json
{"height":4,"round":0,"step":6,"block_id":"975FD583AFA177B6AC0102A4EAEBD5F5
592E3941460E1807435152DB0E270AAE"}%
```

可以看到文件中记录了最近签署过的区块高度. 为了验证tmkms的防双签服务,重置下cetd的状态,然后再启动tmkms服务:

```
➜ kms git:(master) ✗ tmkms start -c tmkms.toml
06:43:10 [INFO] tmkms 0.5.0 starting up...
06:43:10 [INFO] [keyring:softsign:coinexdex-test1,coinexdex-test1] added validator key 1zcjduepqp5lh
xdjhx6n9mfvz7c5ved87wdn2gays4njeyzrqu9rk2nx3r9xsmmhvha
06:43:10 [INFO] KMS node ID: 079AE9C9881217FD9E999CFD238A8D266B0B4975
06:43:10 [ERROR] [coinexdex-test1@tcp://127.0.0.1:26657] I/O error: Connection refused (os error 61)
06:43:11 [INFO] KMS node ID: 079AE9C9881217FD9E999CFD238A8D266B0B4975
06:43:11 [ERROR] [coinexdex-test1@tcp://127.0.0.1:26657] I/O error: Connection refused (os error 61)
06:43:12 [INFO] KMS node ID: 079AE9C9881217FD9E999CFD238A8D266B0B4975
06:43:12 [ERROR] [coinexdex-test1@tcp://127.0.0.1:26657] I/O error: Connection refused (os error 61)
06:43:13 [INFO] KMS node ID: 079AE9C9881217FD9E999CFD238A8D266B0B4975
06:43:13 [WARN] [coinexdex-test1] 127.0.0.1:26657: unverified validator peer ID! (36FD7EF909D1341690
18BDCDFD871496C116D204)
06:43:13 [INFO] [coinexdex-test1@tcp://127.0.0.1:26657] connected to validator successfully
06:43:15 [ERROR] [coinexdex-test1@tcp://127.0.0.1:26657] attempted double sign: height regression: l
ast height:4 new height:1
06:43:16 [INFO] KMS node ID: 079AE9C9881217FD9E999CFD238A8D266B0B4975
06:43:17 [WARN] [coinexdex-test1] 127.0.0.1:26657: unverified validator peer ID! (36FD7EF909D1341690
18BDCDFD871496C116D204)
06:43:17 [INFO] [coinexdex-test1@tcp://127.0.0.1:26657] connected to validator successfully
06:43:18 [ERROR] [coinexdex-test1@tcp://127.0.0.1:26657] attempted double sign: height regression: l
ast height:4 new height:1
06:43:19 [INFO] KMS node ID: 079AE9C9881217FD9E999CFD238A8D266B0B4975
06:43:19 [ERROR] [coinexdex-test1@tcp://127.0.0.1:26657] I/O error: Connection refused (os error 61)
^C06:43:20 [INFO] Waiting for client threads to stop...
06:43:20 [INFO] [coinexdex-test1@tcp://127.0.0.1:26657] shutdown request received
➜ kms git:(master) ✗
```

```
➜  dex git:(master) ✗ ./cetd unsafe-reset-all
I[2019-07-18|14:43:00.426] Removed all blockchain history                module=main dir=/Users/long/
.cetd/data
I[2019-07-18|14:43:00.433] Reset private validator file to genesis state module=main keyFile=/Users/
long/.cetd/config/priv_validator_key.json stateFile=/Users/long/.cetd/data/priv_validator_state.json
➜  dex git:(master) ✗ ./cetd start
I[2019-07-18|14:43:12.571] Starting ABCI with Tendermint                 module=main
E[2019-07-18|14:43:13.846] Couldn't connect to any seeds                 module=p2p
E[2019-07-18|14:43:15.950] enterPropose: Error signing proposal          module=consensus height=1 ro
und=0 err=EOF
E[2019-07-18|14:43:17.791] Ping                                          module=privval err=EOF
E[2019-07-18|14:43:18.958] Error signing vote                            module=consensus height=1 ro
und=0 vote="Vote{0:1BF80B2967DB 1/00/1(Prevote) 000000000000 000000000000 @ 2019-07-18T06:43:18.9585
01Z}" err=EOF
^C
➜  dex git:(master) ✗
```

可以看到tmkms在抱怨 `attempted double sign` ,因为本地记录的 `coinexdex-tes1` 签署过的最新区块高度为4,而 `reset` 之后会从区块高度1开始请求签名服务,因此判定是双签错误.删除文件 `coinexdex-test1_priv_validator_state.json` 之后可以成功执行.

基于YubiHSM2的远程签名服务

`softsign` 模式通过分离私钥存储与验证节点,可以为私钥存储提供更为妥善的安全环境,例如托管在公司内网之中. 但是如果想要更好的保护签名私钥的安全性,则推荐采用基于YubiHSM2的 `yubihsm` 模式. 可以在链接购买YubiHSM2产品,价格为650美金可以说是成本较低的HSM方案了.值得注意的是官方网站中明确说明YubiHSM2产品无法运输到中国大陆,但是可以运送到中国香港.

YubiHSM2是一个指甲盖大小的HSM产品,需要插入到USB槽中使用.关于YubiHSM2的更多产品信息和使用手册,参见链接,推荐阅读其中的Concepts部分以及Practical Guide部分,并熟悉其中的关于 `type` , `id` 以及 `session` 的概念,密钥备份等功能在其中也有说明,值得注意的是在tmkms项目中也有关于 `yubihsm` 部分的文档,为了理解这个文档需要理解YubiHSM2中的Session, Domain和Capabilities的概念. 另外需要额外注意的是, 文档中执行完 `tmkms yubihsm setup` 的输出如下,其中对应 key 0x0001的口令是 `double section release consider diet pilot flip shell mother alone what fantasy much answer lottery crew nut reopen stereo square popular addict just animal` , 而对应authkey 0x0002的口令是 `kms-operator-password-1k02vtxh4ggxct5tngncc33rk9yy5yjhk` 而非 `1k02vtxh4ggxct5tngncc33rk9yy5yjhk` ,笔者由于默认为 `1k02vtxh4ggxct5tngncc33rk9yy5yjhk` 是对应的口令而在操作YubiHSM2时耽误了较多时间.

```
$ tmkms yubihsm setup
This process will *ERASE* the configured YubiHSM2 and reinitialize it:

- YubiHSM serial: 9876543210

Authentication keys with the following IDs and passwords will be created:

- key 0x0001: admin:
```

```
    double section release consider diet pilot flip shell mother alone what
fantasy
    much answer lottery crew nut reopen stereo square popular addict just
animal

- authkey 0x0002 [operator]:  kms-operator-password-
1k02vtxh4ggxct5tngncc33rk9yy5yjhk
- authkey 0x0003 [auditor]:   kms-auditor-password-
1s0ynq69ezavnqgq84p0rkhxvkqm54ks9
- authkey 0x0004 [validator]: kms-validator-password-
1x4anf3n8vqkzm0klrwljhcx72sankcw0
- wrapkey 0x0001 [primary]:
21a6ca8cfd5dbe9c26320b5c4935ff1e63b9ab54e2dfe24f66677aba8852be13

*** Are you SURE you want erase and reinitialize this HSM? (y/N):
```

本文中我们采用简单的方式部署YubiHSM2以便验证由tmkms提供的 `yubihsm` 模式的remote signer的功能,主要参考的资料是Node A-Team团队的博客.有了之前 `softsign` 部署的经验,与单节点的dex适配的基于'yubihsm 的`remote signer` 的主要操作包括: 重置YubiHSM2, 编译安全tmkms并向YubiHSM2内部导入签名私钥或者直接在YubiHSM2内部生成Ed25519签名私钥.

重置YubiHSM2需要用 `./yubihsm-connector -d` 建立与HSM之间的连接, 然后利用 `yubihsm-shell` 对HSM进行 `factory reset` :

```
$ ./yubihsm-shell
Using default connector URL: http://127.0.0.1:12345
yubihsm> connect
Session keepalive set up to run every 15 seconds
yubihsm> session open 1 password
Created session 0
yubihsm> reset 0
Device successfully reset
yubihsm> session open 1 password
Created session 0
yubihsm> session close 0
yubihsm>
```

安装tmkms并导入签名密钥并退出yubihsm-connector和yubihsm-shell,用 `--features=yubihsm` 选项安装 `tmkms` 将 `cetd` 生成的私钥导入到YubiHSM2中. 其中 `tmkms yubihsm keys import -t json -i 1 ~/.cetd/config/priv_validator_key.json` 的 `-t` 选项表示文件为JSON格式, `-i` 选项指定密钥的id, 必须与配置文件 `tmkms.toml` 中的一致,最后的 `~/.cetd/config/priv_validator_key.json` 是由cetd生成的验证者密钥信息的文件(我们此处是从外部向HSM导入签名私钥信息,也可以直接由HSM自身生成签名私钥,参见文档). 导入签名密钥之后, 执行 `tmkms yubihsm keys list` 可以看到HSM中的密钥信息.

```
$ cargo build --release --features=yubihsm
Finished release [optimized] target(s) in 1.13s
$ cp target/release/tmkms ~/.cargo/bin/
$ tmkms yubihsm detect
Detected YubiHSM2 USB devices:
- Serial #0009680371 (bus 20)
$ tmkms yubihsm keys list
error: no keys in this YubiHSM (#0009680371)
$ tmkms yubihsm keys import -t json -i 1
~/.cetd/config/priv_validator_key.json
Imported key 0x0001
$  tmkms yubihsm keys list
Listing keys in YubiHSM #0009680371:
- 0x0001:
coinexvalconspub1zcjduepqr6ufttg9cn3gyv2e2zumtyxgr8gncjqfex3suvjkff24dcw0yn
0qxeem09
```

为了使上述命令能够成功执行,需要配置 `tmkms` 的配置文件,由于是 `yubihsm` 模式所以需要配置 `[[providers.yubihsm]]` 小节. 其中 `auth` 中制定了密钥的ID以及访问该私钥所需的口令,此处使用了ID为1的私钥的默认口令"password".

```
[[chain]]
id = "coinexdex-test1"
key_format = { type = "bech32", account_key_prefix = "coinexpub",
consensus_key_prefix = "coinexvalconspub" }

# Validator configuration
[[validator]]
addr = "tcp://127.0.0.1:26659"
chain_id = "coinexdex-test1"
reconnect = true # true is the default
secret_key = "./secret_connection.key"

# Provider configuration
# [[providers.softsign]]
# chain_ids = ["coinexdex-test1"]
# path = "./signing.key"

# enable the `yubihsm` feature to use this backend
[[providers.yubihsm]]
adapter = { type = "usb" }
auth = { key = 1, password = "password" } # or pass raw password as
`password`
```

```
keys = [{ chain_ids = ["coinexdex-test1"], key = 1 }]
serial_number = "0009680371" # identify serial number of a specific YubiHSM
to connect to
# connector_server = { laddr = "tcp://127.0.0.1:12345", cli = { auth_key =
1 } } # run yubihsm-connector compatible server
```

接下来配置 `cetd`： `~/.cetd/config/config.toml` 注释掉其中的两行:

```
# Path to the JSON file containing the private key to use as a validator in
the consensus protocol
# priv_validator_key_file = "config/priv_validator_key.json"
```

配置 `listen remote signer` 的IP地址和端口号,与 `tmkms` 的配置文件中一致:

```
# TCP or UNIX socket address for Tendermint to listen on for
# connections from an external PrivValidator process
priv_validator_laddr = "tcp://127.0.0.1:26659"'
```

接下来利用 `tm-singer-hardness` 工具测试基于 `yubihsm` 模式的remote signer的功能的正确性与完备性,首先利用 `tmkms start -c /path/to/tmkms.toml` 启动remote signer, 然后启动 `tm-signer-hardness` 工具:

```
 long@LMBP  ~/Downloads/yubihsm2-sdk/bin  tm-signer-harness run -addr
tcp://127.0.0.1:26659 -tmhome ~/.cetd
I[2019-11-05|18:20:44.086] Loading private validator configuration
keyFile=/Users/long/.cetd/config/priv_validator_key.json
stateFile=/Users/long/.cetd/data/priv_validator_state.json
I[2019-11-05|18:20:44.087] Loading chain ID from genesis file
genesisFile=/Users/long/.cetd/config/genesis.json
I[2019-11-05|18:20:44.087] Loaded genesis file
chainID=coinexdex-test1
I[2019-11-05|18:20:44.088] Listening
proto=tcp addr=127.0.0.1:26659
I[2019-11-05|18:20:44.088] Resolved TCP address for listener
addr=127.0.0.1:26659
I[2019-11-05|18:20:44.088] Starting SignerListenerEndpoint
impl=SignerListenerEndpoint
I[2019-11-05|18:20:44.088] SignerListener: Listening for new connection
I[2019-11-05|18:20:44.088] Starting test harness
I[2019-11-05|18:20:44.088] Attempting to accept incoming connection
acceptRetries=100
```

```
I[2019-11-05|18:20:44.100] Attempting to accept incoming connection
acceptRetries=99
I[2019-11-05|18:20:44.113] Attempting to accept incoming connection
acceptRetries=98
I[2019-11-05|18:20:44.123] Attempting to accept incoming connection
acceptRetries=97
I[2019-11-05|18:20:44.136] Attempting to accept incoming connection
acceptRetries=96
I[2019-11-05|18:20:44.146] Attempting to accept incoming connection
acceptRetries=95
I[2019-11-05|18:20:44.157] Attempting to accept incoming connection
acceptRetries=94
I[2019-11-05|18:20:44.168] Attempting to accept incoming connection
acceptRetries=93
I[2019-11-05|18:20:44.180] Attempting to accept incoming connection
acceptRetries=92
I[2019-11-05|18:20:44.190] Attempting to accept incoming connection
acceptRetries=91
I[2019-11-05|18:20:44.514] SignerListener: Connected
I[2019-11-05|18:20:44.514] Accepted external connection
I[2019-11-05|18:20:44.514] TEST: Public key of remote signer
I[2019-11-05|18:20:44.515] Local
pubKey=PubKeyEd25519{1EB895AD05C4E282315950B9B590C819D13C4809C9A30E32564A55
56E1CF24DE}
I[2019-11-05|18:20:44.515] Remote
pubKey=PubKeyEd25519{1EB895AD05C4E282315950B9B590C819D13C4809C9A30E32564A55
56E1CF24DE}
I[2019-11-05|18:20:44.515] TEST: Signing of proposals
I[2019-11-05|18:20:44.663] Successfully validated proposal signature
I[2019-11-05|18:20:44.663] TEST: Signing of votes
I[2019-11-05|18:20:44.663] Testing vote type
type=1
I[2019-11-05|18:20:44.810] Successfully validated vote signature
type=1
I[2019-11-05|18:20:44.810] Testing vote type
type=2
I[2019-11-05|18:20:44.958] Successfully validated vote signature
type=2
I[2019-11-05|18:20:44.958] SUCCESS! All tests passed.
 long@LMBP  ~/Downloads/yubihsm2-sdk/bin 
```

可以看到基于'yubihsm'模式的remote signer通过了 `tm-signer-hardness` 的测试, 接下来首先删除tmkms工作目录下 `coinexdex-test1_priv_validator_state.json` 文件,然后依次启动 `tmkms` 服务和 `cetd` 服务,执行结果如下:

```
~/Go/src/github.com/tendermint/kms master ● tmkms start
10:25:37 [info] tmkms 0.6.3 starting up...
10:25:37 [info] [keyring:yubihsm] added consensus key coinexvalconspub1zcjduepqr6ufttg9cn3gyv2e2zumtyxgr8gncjqfex3suvjkff24dc
w0yn0qxeem09
10:25:37 [info] KMS node ID: E19F932F04BC03D4D96BE2A6C79DC23358565630
10:25:37 [error] [coinexdex-test1@tcp://127.0.0.1:26659] I/O error: Connection refused (os error 61)
10:25:38 [info] KMS node ID: E19F932F04BC03D4D96BE2A6C79DC23358565630
10:25:38 [error] [coinexdex-test1@tcp://127.0.0.1:26659] I/O error: Connection refused (os error 61)
10:25:39 [info] KMS node ID: E19F932F04BC03D4D96BE2A6C79DC23358565630
10:25:39 [error] [coinexdex-test1@tcp://127.0.0.1:26659] I/O error: Connection refused (os error 61)
10:25:40 [info] KMS node ID: E19F932F04BC03D4D96BE2A6C79DC23358565630
10:25:40 [error] [coinexdex-test1@tcp://127.0.0.1:26659] I/O error: Connection refused (os error 61)
10:25:41 [info] KMS node ID: E19F932F04BC03D4D96BE2A6C79DC23358565630
10:25:41 [error] [coinexdex-test1@tcp://127.0.0.1:26659] I/O error: Connection refused (os error 61)
10:25:43 [info] KMS node ID: E19F932F04BC03D4D96BE2A6C79DC23358565630
10:25:43 [info] [coinexdex-test1@tcp://127.0.0.1:26659] connected to validator successfully
10:25:43 [warn] [coinexdex-test1] tcp://127.0.0.1:26659: unverified validator peer ID! (031F580AB92E6AE0F0A2D8B7240712398491C
A16)
10:25:45 [error] [coinexdex-test1:tcp://127.0.0.1:26659] attempted double sign Proposal at h/r/s: 1/0/0 (<nil> != 128C402F88)
10:25:48 [info] [coinexdex-test1@tcp://127.0.0.1:26659] signed PreVote:<nil> at h/r/s 1/0/1 (106 ms)
10:25:48 [info] [coinexdex-test1@tcp://127.0.0.1:26659] signed PreCommit:<nil> at h/r/s 1/0/2 (106 ms)
10:25:49 [info] [coinexdex-test1@tcp://127.0.0.1:26659] signed Proposal:128C402F88 at h/r/s 1/1/0 (128 ms)
10:25:50 [info] [coinexdex-test1@tcp://127.0.0.1:26659] signed PreVote:128C402F88 at h/r/s 1/1/1 (128 ms)
10:25:50 [info] [coinexdex-test1@tcp://127.0.0.1:26659] signed PreCommit:128C402F88 at h/r/s 1/1/2 (128 ms)
10:25:53 [info] [coinexdex-test1@tcp://127.0.0.1:26659] signed Proposal:758A65A8DF at h/r/s 2/0/0 (128 ms)
10:25:53 [info] [coinexdex-test1@tcp://127.0.0.1:26659] signed PreVote:758A65A8DF at h/r/s 2/0/1 (128 ms)
10:25:53 [info] [coinexdex-test1@tcp://127.0.0.1:26659] signed PreCommit:758A65A8DF at h/r/s 2/0/2 (128 ms)
10:25:56 [info] [coinexdex-test1@tcp://127.0.0.1:26659] signed Proposal:37A20BE348 at h/r/s 3/0/0 (128 ms)
10:25:56 [info] [coinexdex-test1@tcp://127.0.0.1:26659] signed PreVote:37A20BE348 at h/r/s 3/0/1 (128 ms)
10:25:56 [info] [coinexdex-test1@tcp://127.0.0.1:26659] signed PreCommit:37A20BE348 at h/r/s 3/0/2 (128 ms)
10:25:59 [info] [coinexdex-test1@tcp://127.0.0.1:26659] signed Proposal:DFD8F3F98D at h/r/s 4/0/0 (128 ms)
10:25:59 [info] [coinexdex-test1@tcp://127.0.0.1:26659] signed PreVote:DFD8F3F98D at h/r/s 4/0/1 (128 ms)
10:25:59 [info] [coinexdex-test1@tcp://127.0.0.1:26659] signed PreCommit:DFD8F3F98D at h/r/s 4/0/2 (128 ms)
10:26:02 [info] [coinexdex-test1@tcp://127.0.0.1:26659] signed Proposal:D0BB17BA5B at h/r/s 5/0/0 (128 ms)
10:26:02 [info] [coinexdex-test1@tcp://127.0.0.1:26659] signed PreVote:D0BB17BA5B at h/r/s 5/0/1 (128 ms)
10:26:02 [info] [coinexdex-test1@tcp://127.0.0.1:26659] signed PreCommit:D0BB17BA5B at h/r/s 5/0/2 (128 ms)
^C
```

```
~/Downloads/yubihsm2-sdk/bin cetd start
I[2019-11-05|18:25:42.422] starting ABCI with Tendermint                module=main
I[2019-11-05|18:25:42.465] --minimum-gas-prices option not set!         module=main
E[2019-11-05|18:25:45.562] enterPropose: Error signing proposal         module=consensus height=1 round=0 err
="signerEndpoint returned error #4: double signing requested at height: 1"
I[2019-11-05|18:25:50.381] Executed block                               module=state height=1 validTxs=0 inva
lidTxs=0
I[2019-11-05|18:25:50.397] Committed state                              module=state height=1 txs=0 appHash=B
A35356AB72C321E0BFC0121058310BC25EEB387E66E10A16CBBA9AB49615D9E
I[2019-11-05|18:25:53.392] Executed block                               module=state height=2 validTxs=0 inva
lidTxs=0
I[2019-11-05|18:25:53.403] Committed state                              module=state height=2 txs=0 appHash=2
C94AFC5437FA09E0BD09633FF03996FC650D79293539AAD3BD868A89BFA1F1D
I[2019-11-05|18:25:56.404] Executed block                               module=state height=3 validTxs=0 inva
lidTxs=0
I[2019-11-05|18:25:56.417] Committed state                              module=state height=3 txs=0 appHash=5
7E8515C56062D53FBF2DA5988626E9951EFEF6CCBA96189F269B89024A90511
I[2019-11-05|18:25:59.413] Executed block                               module=state height=4 validTxs=0 inva
lidTxs=0
I[2019-11-05|18:25:59.431] Committed state                              module=state height=4 txs=0 appHash=9
F3AE3253CA8A6C36447D519FBDBB8CA8C8C6BB3089594A459E4DA257B05450C
I[2019-11-05|18:26:02.419] Executed block                               module=state height=5 validTxs=0 inva
lidTxs=0
I[2019-11-05|18:26:02.430] Committed state                              module=state height=5 txs=0 appHash=8
2AF110125E0DAB28CDF7B485BFB2363E7ED1491EC3A7F42B0EA799D9E5D5AA9
E[2019-11-05|18:26:04.125] SignerListener: Ping timeout                 module=privval
E[2019-11-05|18:26:07.228] SignerListener: Ping timeout                 module=privval
^CE[2019-11-05|18:26:09.698] Not stopping BlockPool -- have not been started yet module=blockchain impl=Block
Pool
```

问题记录

1. 基于 yubihsm 模式的remote signer在 cetd 和 tmkms 都是新配置的情况下总会在对 height=1 情况的处理时报告错误, 例如 tmkms 报告 10:25:45 [error] [coinexdex-test1:tcp://127.0.0.1:26659] attempted double sign Proposal at h/r/s: 1/0/0 (<nil> != 128C402F88) , 而 cetd 则报告 E[2019-11-05|18:25:45.562] enterPropose: Error signing proposal module=consensus height=1 round=0 err="signerEndpoint returned error #4:

`double signing requested at height: 1"` .中断两个服务之后再重新开始2个服务,则不会出现这种情况.猜测是tmkms在初始化时的判断逻辑有问题,留待继续跟进.

2. 在配置了remote signer的情况下,通过 `cetd tendermint show-validator` 或者 `cetd tendermint show-address` 得到的关于验证节点的信息与remote signer所提供的并不一致:

```
long@LMBP  ~  cetd tendermint show-validator
coinexvalconspub1zcjduepqr6ufttg9cn3gyv2e2zumtyxgr8gncjqfex3suvjkff24dc
w0yn0qxeem09
 long@LMBP  ~  cetd tendermint show-address
coinexvalcons1xs7x6atmqxq9h76qa89dhnw2h0lshvtkcx4ul0
 long@LMBP  ~  cetcli q tendermint-validator-set --chain-
id=coinexdex-test1
blockheight: 27
validators:
- address: coinexvalcons1xs7x6atmqxq9h76qa89dhnw2h0lshvtkcx4ul0
  pubkey:
coinexvalconspub1zcjduepqr6ufttg9cn3gyv2e2zumtyxgr8gncjqfex3suvjkff24dc
w0yn0qxeem09
  proposerpriority: 0
  votingpower: 1000000000
```

甚至在删除本地~/.cetd/config/priv_validator_key.json前后执行查询到的信息也不一致:

```
 $ cetd tendermint show-validator
coinexvalconspub1zcjduepqqdrhfdnelxwfh67ypq4wuzf0x0kva3gpkhs95npj20yhvk
rn850sxc9x4c
 $ rm ~/.cetd/config/priv_validator_key.json
 $ cetd tendermint show-validator
coinexvalconspub1zcjduepqstk776c9jdvfd38usf6f96f5dsm4zurdflgkhs3z97qhd3
4k532qd5qrym
```

通过链接 和 链接 可以发现相关实现 `ShowValidatorCmd` 总是会通过读取本地priv_validator_key.json文件来返回结果, 并且在文件不存在时会临时生成一个 `priv_validator_key.json` 文件来读取相应的值,并不关心是否配置了remote signer.

```
// ShowValidator - ported from Tendermint, show this node's validator
info
func ShowValidatorCmd(ctx *Context) *cobra.Command {
  cmd := cobra.Command{
    Use:   "show-validator",
    Short: "Show this node's tendermint validator info",
    RunE: func(cmd *cobra.Command, args []string) error {

      cfg := ctx.Config
```

```
        UpgradeOldPrivValFile(cfg)
        privValidator := pvm.LoadOrGenFilePV(
          cfg.PrivValidatorKeyFile(), cfg.PrivValidatorStateFile())
        valPubKey := privValidator.GetPubKey()

        if viper.GetString(cli.OutputFlag) == "json" {
          return printlnJSON(valPubKey)
        }

        pubkey, err := sdk.Bech32ifyConsPub(valPubKey)
        if err != nil {
          return err
        }

        fmt.Println(pubkey)
        return nil
      },
    }

    cmd.Flags().StringP(cli.OutputFlag, "o", "text", "Output format
  (text|json)")
    return &cmd
  }
```

在配置了remote signer情况下,这似乎是一个bug,但是根据下面的讨论,cosmos团队似乎只是想要一个能够转换该文件内容的工具:Implement gaiad tendermint show_address #1464:

"gaiad tendermint show_address should show the address of your validator encoded as bech32. Currently, the only way to find it is to cat priv_validator.json and then manually transform it from HEX to bech32."

对 `priv_validator_key.json` 文件的正确的处理逻辑应当是: `cetd` 启动的时候首先判定是否有remote singer有的话就调用 `getpubkey` 然后根据返回值生成json文件(此时json文件不应该包含priv_key部分的信息).如果没有remote singer就自己生成一个.只有在没有remote signer并且没有本地JSON文件的时候才随机生成一个(包含priv_key) 这样能保证JSON文件与节点保持一致.