学堂在线
xuetangx.com

以太坊源码解析 - Account

郑嘉文

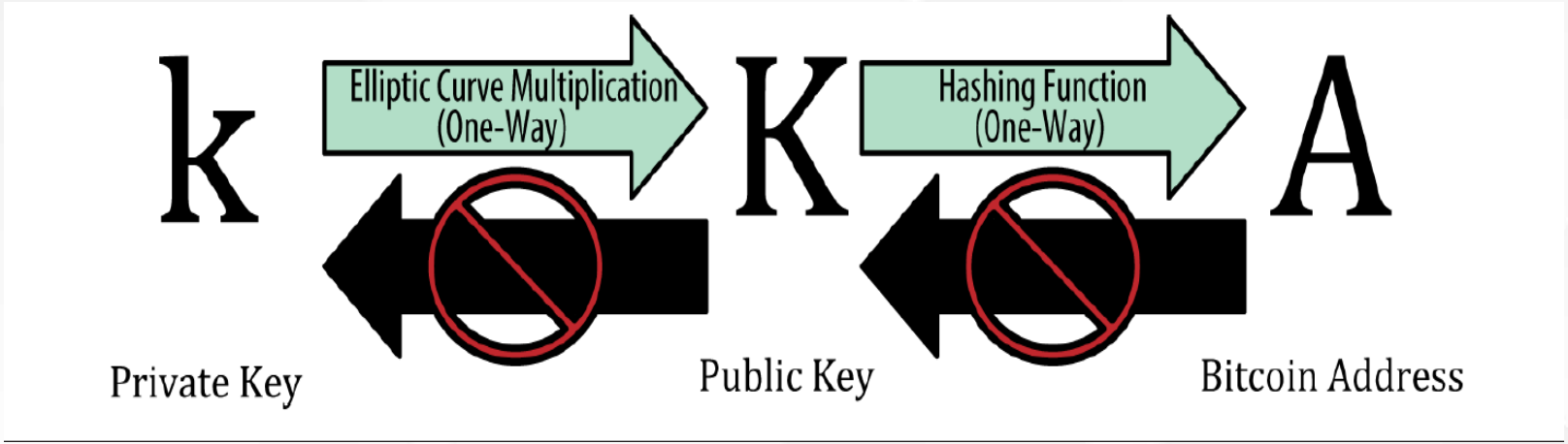
2018/03/31

目录/Contents

01

预备知识

Key/Address


k
Elliptic Curve Multiplication
(One-Way)
K
Hashing Function
(One-Way)
A
Private Key
Public Key
Bitcoin Address

私钥(Private Key)

范围

[1, n – 1] n = 1.158 * 10^77

注意

不要自己生成私钥。要使用cryptographically secure pseudo-random number generator (CSPRNG)

例子

1E99423A4ED27608A15A2616A2B0E9E52CED330AC530EDCC32C8FFC6A526AEDD

256 bit

$1.158 * 10^{77}$

10^{80} atoms

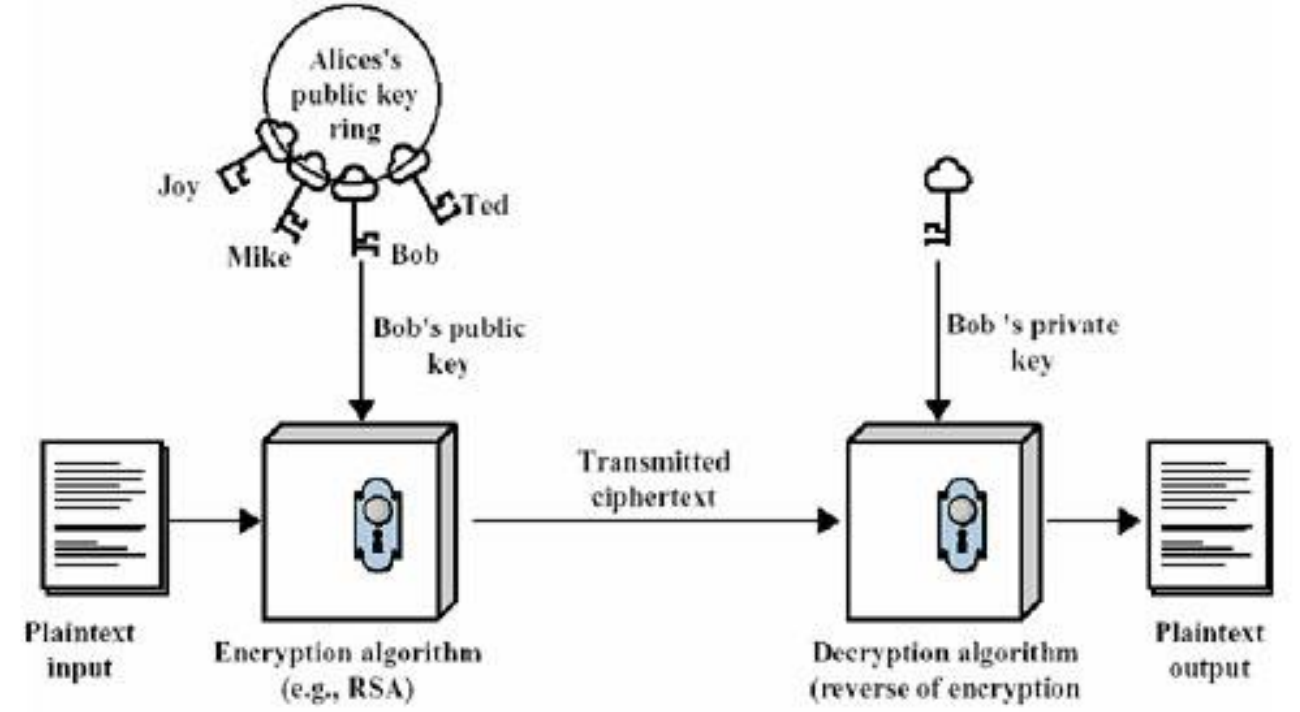
Bitcoin uses a specific elliptic curve and set of mathematical constants, as defined in a standard called secp256k1, established by the National Institute of Standards and Technology (NIST).

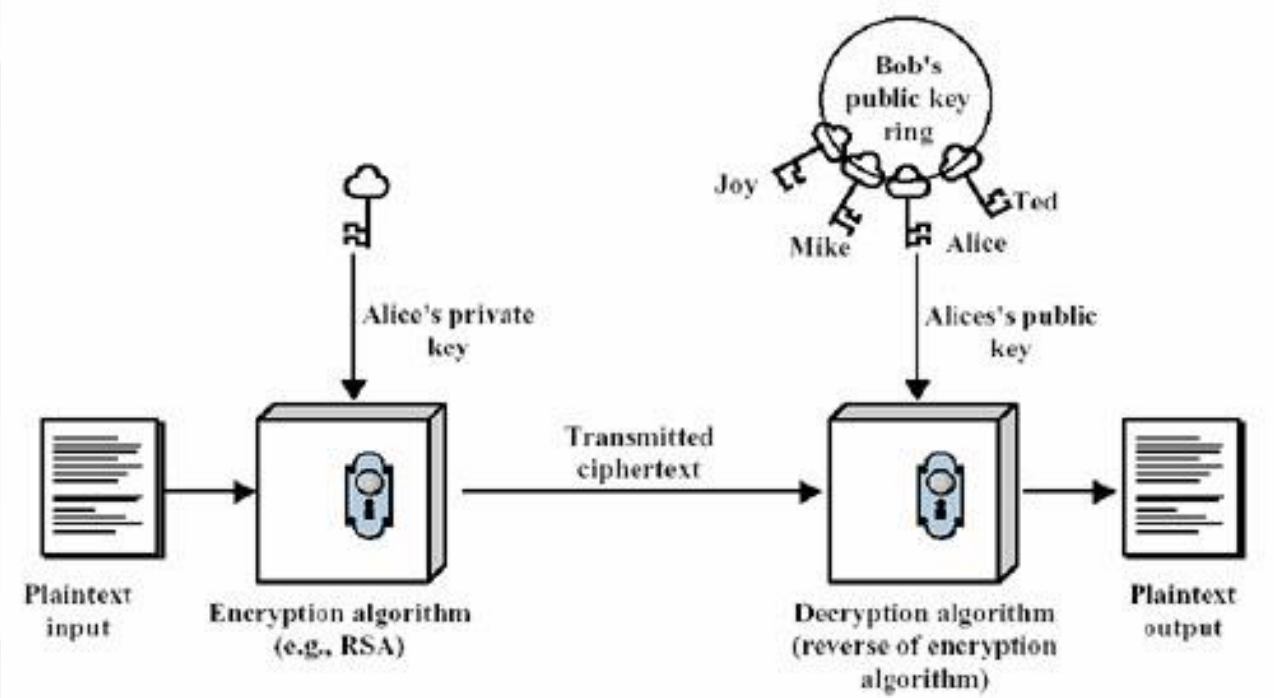
The secp256k1 curve is defined by the following function, which produces an elliptic curve:

$y^2 = x^3 + 7$ over p
or
$Y^2 \bmod p = x^3 + 7 \bmod p$

p ： prime number

基于公开密钥的加密


Alices's public key ring
Joy
Ted
Mike
Bob
Bob's public key
Bob 's private key
Transmitted ciphertext
Plaintext input
Encryption algorithm (e.g., RSA)
Decryption algorithm (reverse of encryption
Plaintext output


Bob's
public key
ring
Joy
Ted
Mike
Alice
Alice's private
key
Alices's public
key
Transmitted
ciphertext
Plaintext
input
Encryption algorithm
(e.g., RSA)
Decryption algorithm
(reverse of encryption
algorithm)
Plaintext
output

地址(Address)

公式

$A = RIPEMD160(SHA256(K))$

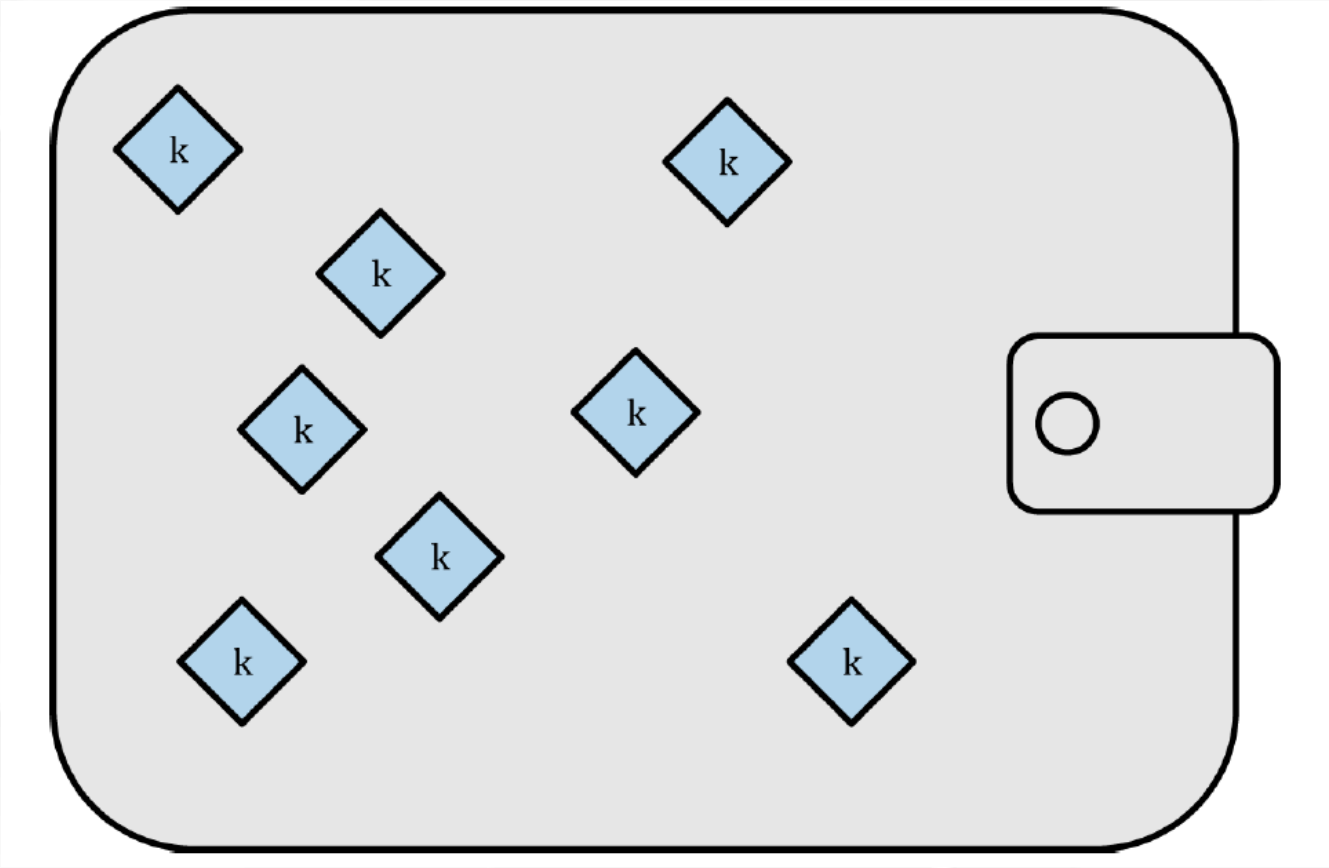
长度 160bit = 20 Byte

例子

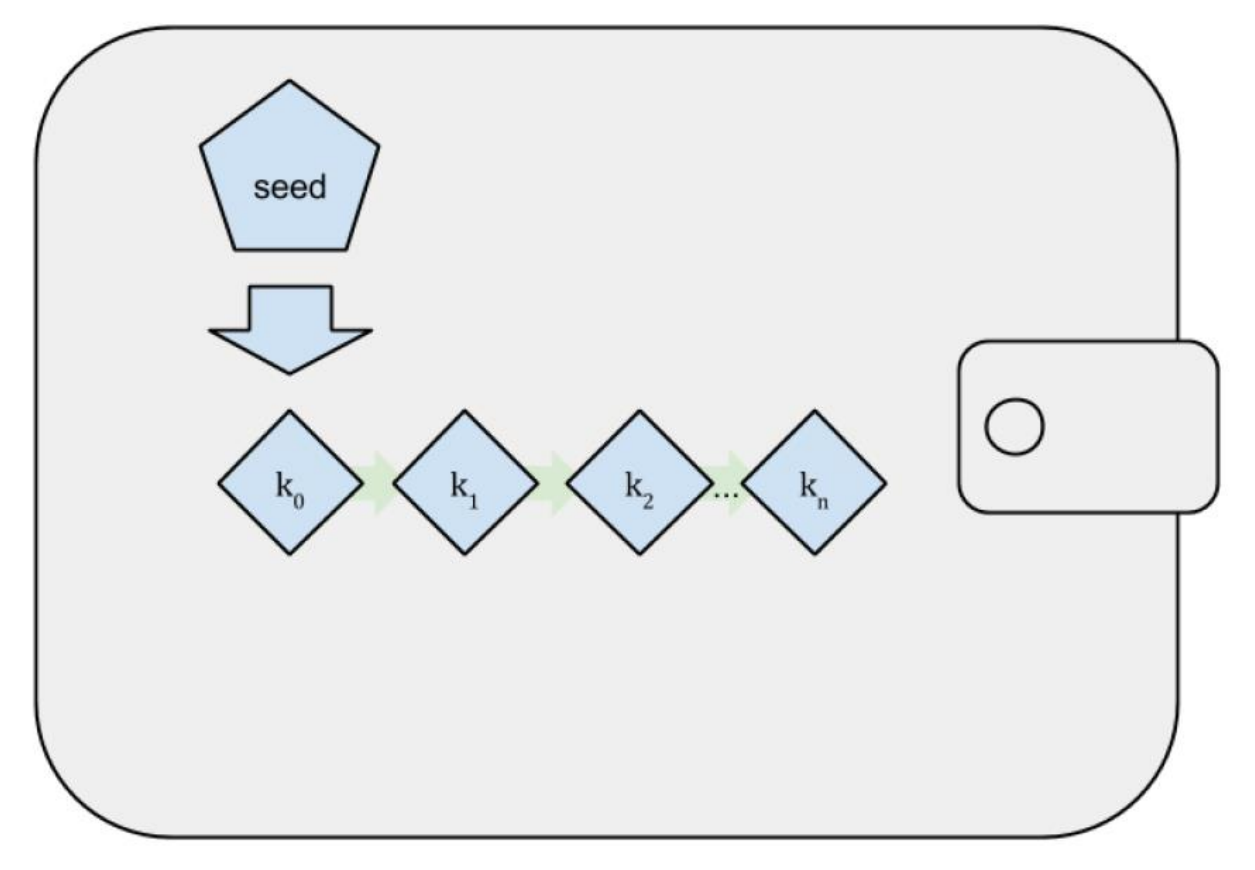
1J7mdg5rbQyUHENYdx39WVWK7fsLpEoXZy

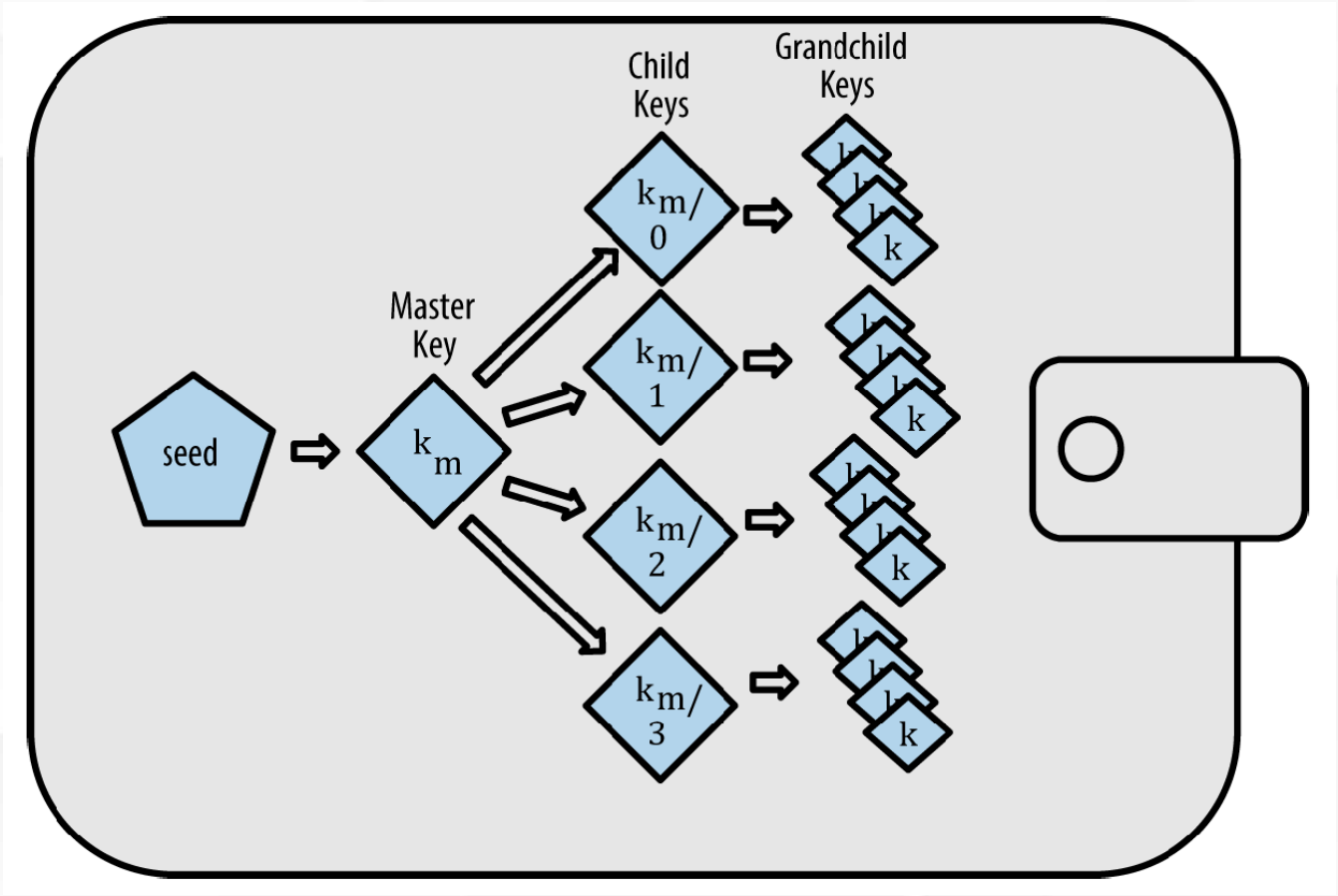
使用地址来接收和发送资金，地址采用Base58Check编码

BIP-38 proposes a common standard for encrypting private keys with a passphrase and encoding them with Base58Check so that they can be stored securely on backup media, transported securely between wallets, or kept in any other conditions where the key might be exposed.


k
k
k
k
k
k
k
k
k

Deterministic (Seeded) Wallets




Child Keys
Grandchild Keys
Master Key
seed
k_m
$k_m/0$
$k_m/1$
$k_m/2$
$k_m/3$
k

例子

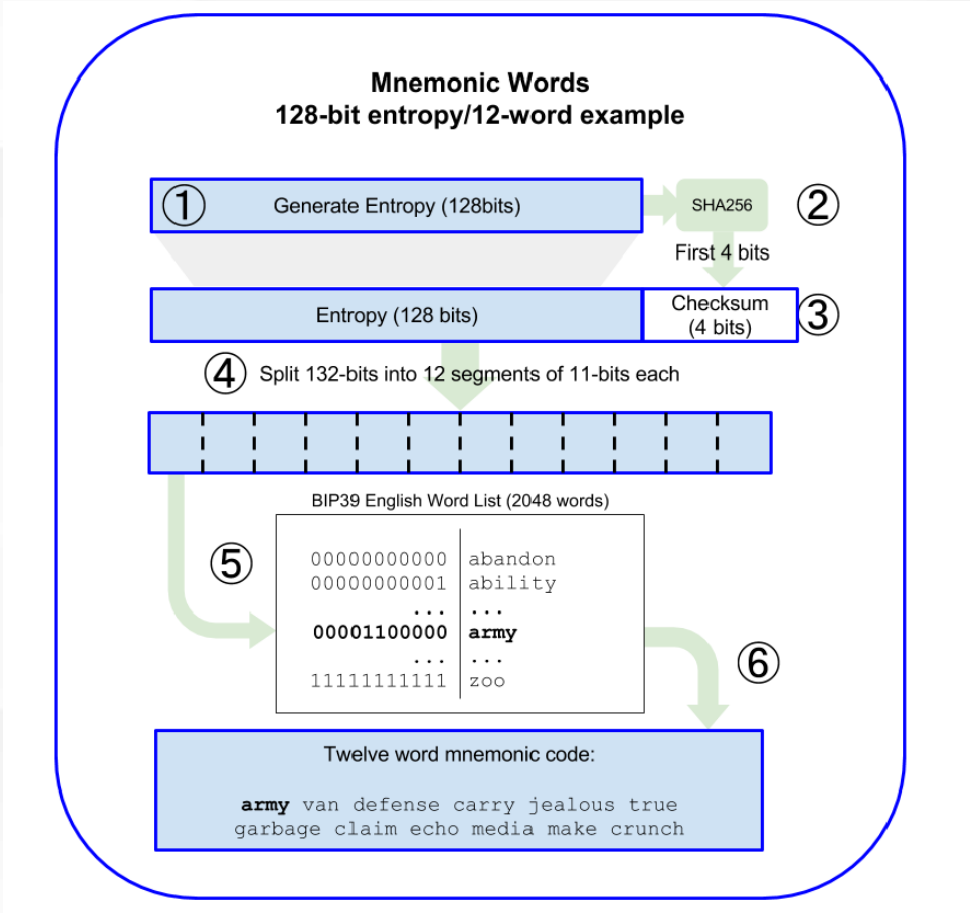
0C1E24E5917779D297E14D45F14E1A1A

army van defense carry jealous true
garbage claim echo media make crunch

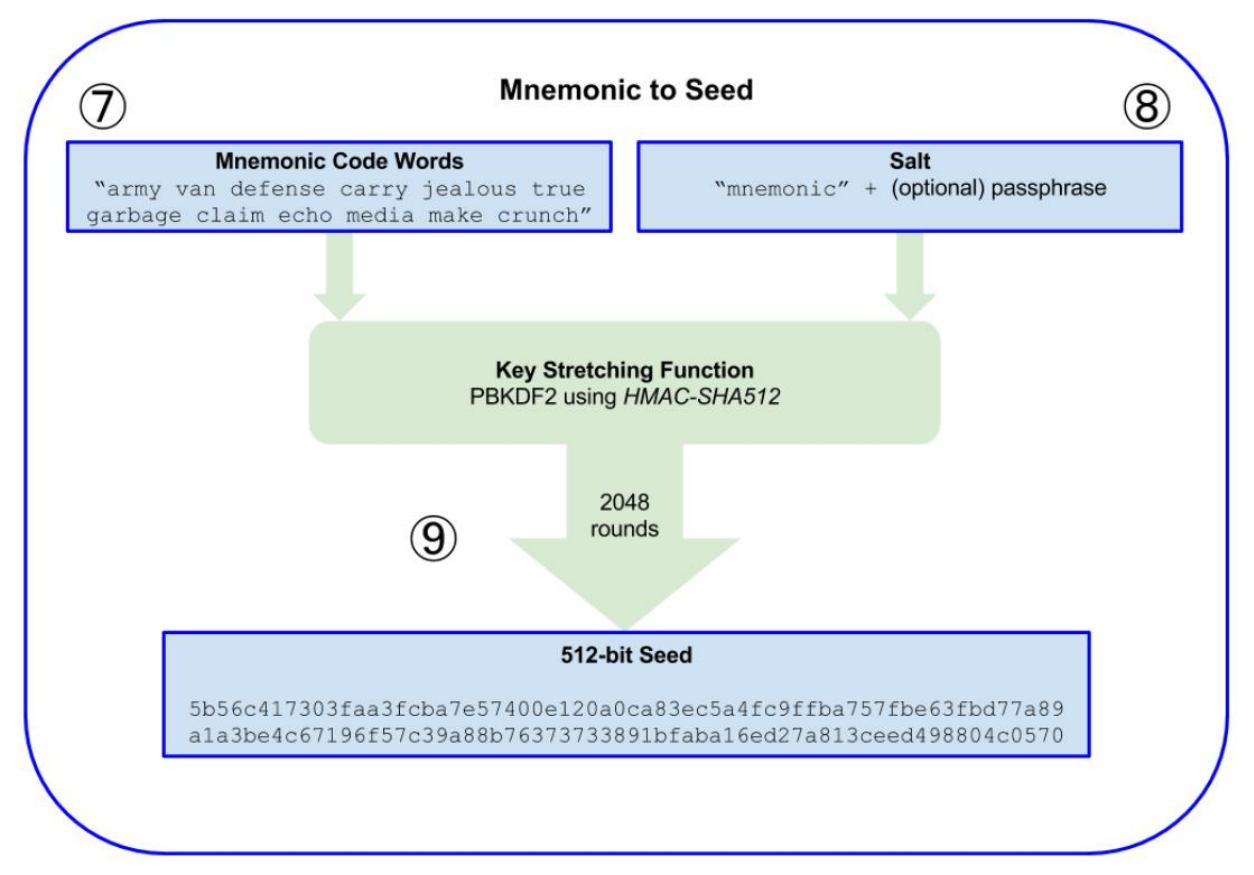
Wallet Best Practice

- Mnemonic code words, based on BIP-39
- HD wallets, based on BIP-32
- Multipurpose HD wallet structure, based on BIP-43
- Multicurrency and multiaccount wallets, based on BIP-44

Seeds and Mnemonic Codes (BIP-39)



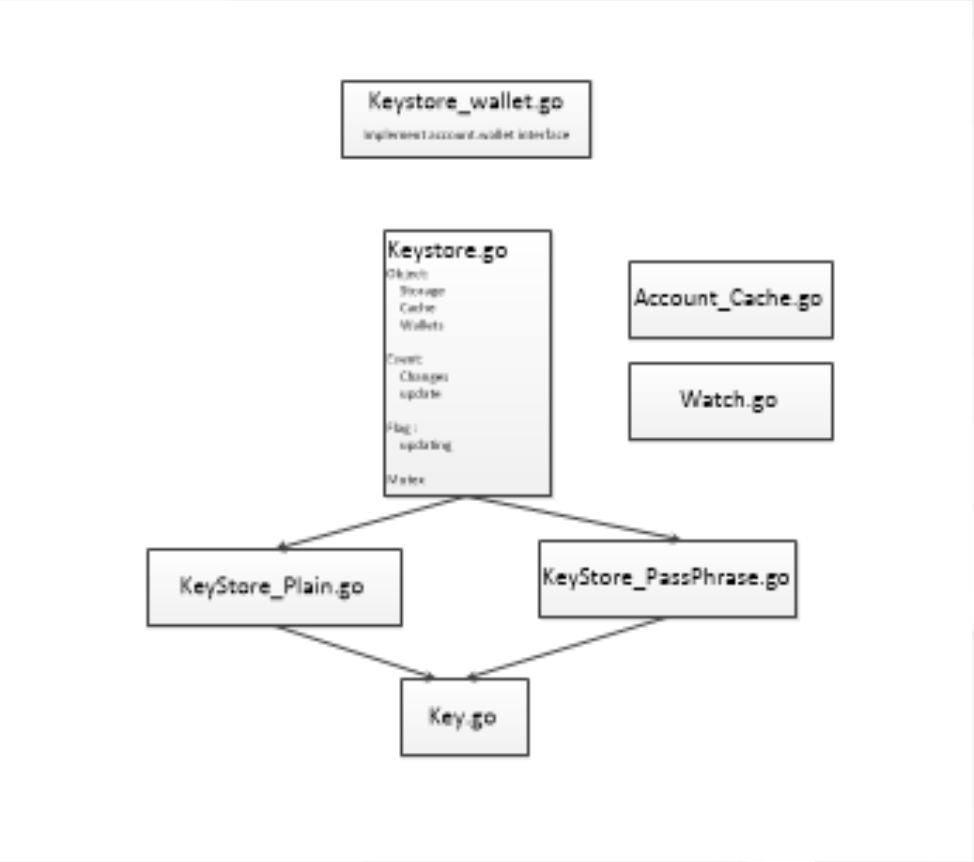
From mnemonic to seed



02

KeyStore

程序结构


Keystore_wallet.go
Keystore.go
Account_Cache.go
Watch.go
KeyStore_Plain.go
KeyStore_PassPhrase.go
Key.go

文件名	主要功能	备注
Key.go	所有Key的读入，输出	
Keystore_plain.go	无加密的Keystore	
Keystore_passphrase.go	以PassPhrase加密的Keystore	
KeyStore.go	KeyStore Interface storage keyStore cache *accountCache wallets []accounts.Wallet …	Generic
KeyStore_wallet.go	Keystore的Wrapper	Wrapper
Account_Cache.go	KeyStore文件内容Cache管理	Helper
Watcher.go	用来监视Keystore目录下文件的变化	Helper

Key File

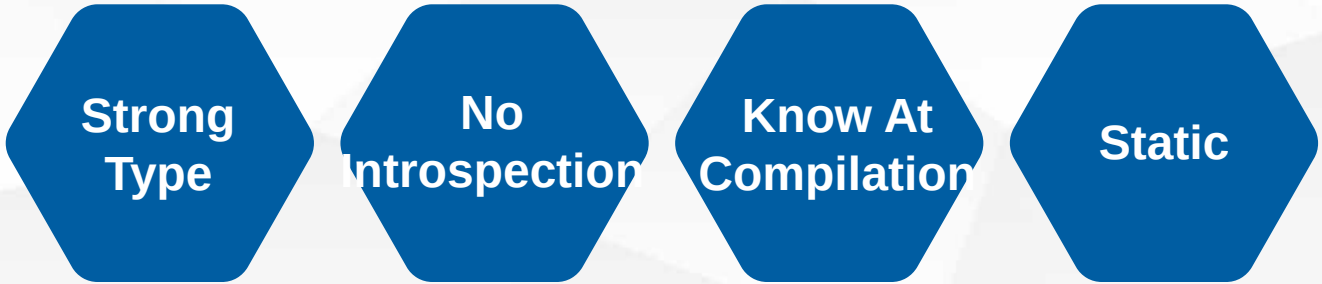

```
{
"address":"22e776db03d422ecf42cf54bff93a2babdd59871",
"crypto":{
    "cipher":"aes-128ctr",
    "ciphertext":"d65b36646477256cc1c37a4e2e569ced59277e4bfda26748ae608325efbdf600",
    "cipherparams":{
        "iv":"48e74ea69dc0a3edbb584f9bbf38d11f"
    },
    "kdf":"scrypt",
    "kdfparams":{
         "dklen":32,
         "n":262144,
         "p":1,
         "r":8,
         "salt":"c61cc7ffff2033d4487199fcf3b359c31b229657194a383eb1cf5e83ba3bc4f8"
     },
     "mac":"2458b177808c349a5ba3b799aefd904b8f064477af3fe234d17396e1b2ba0a90"
 },
"id":"318f90d1-8669-4fb1-9f69-2728e393f403",
"version":3
}
```

03

ABI（Application Binary Interface）

When using the JavaScript dapp API, calling a contract via an abstraction layer such as the eth.contract() function will send back an object with all the functions that contract can run when called in JavaScript. To standardize this introspective functionality, the Ethereum protocol comes with something called the application binary interface, otherwise known as the Contract ABI. The ABI behaves like an API, creating a standard syntax for contracts to be called by applications. The ABI dictates that the contract will send back an array that delineates the proper call signature and the available contract functions.

-- https://github.com/ethereum/wiki/wiki/Ethereum-Contract-ABI#functions



Function Selector

The first four bytes of the call data for a function call specifies the function to be called. It is the first (left, high-order in big-endian) four bytes of the Keccak (SHA-3) hash of the signature of the function.

- Strong Type
- No Introspection
- Know At Compilation
- Static

```
pragma solidity ^0.4.16;
contract Foo {
        function bar(bytes3[2]) public pure {}
        function baz(uint32 x, bool y) public pure returns (bool r) { r = x > 32 || y; }
        function sam(bytes, bool, uint[]) public pure {}
}
```

如果以参数69 and true 调用 baz(), 我们会传送 68 bytes：

0xcdcd77c000000000000000000000000000000000000000000000000000000000000045000000000000000000000000000000000000000000000000000000000000001

值（Value）	详细（Detail）
0xcdcd77c0	the Method ID This is derived as the first 4 bytes of the Keccak hash of the ASCII form of the signature baz(uint32,bool)
0x0000000000000000000000000000000000000000000000000000000000000045	the first parameter, a uint32 value 69 padded to 32 bytes
0x0000000000000000000000000000000000000000000000000000000000000001	the second parameter - boolean true, padded to 32 bytes

Function Selector – Dynamic Type

A call to a function with the signature f(uint,uint32[],bytes10,bytes) with values (0x123, [0x456, 0x789], "1234567890", "Hello, world!")

编码结果：

0x8be65246 0000000000000000000000000000000000000000000000000000000000000123
0000000000000000000000000000000000000000000000000000000000000080
3132333435363738393000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000e0
0000000000000000000000000000000000000000000000000000000000000002
0000000000000000000000000000000000000000000000000000000000000456
0000000000000000000000000000000000000000000000000000000000000789
000000000000000000000000000000000000000000000000000000000000000d
48656c6c6f2c20776f726c6421000000000000000000000000000000000000000

源代码

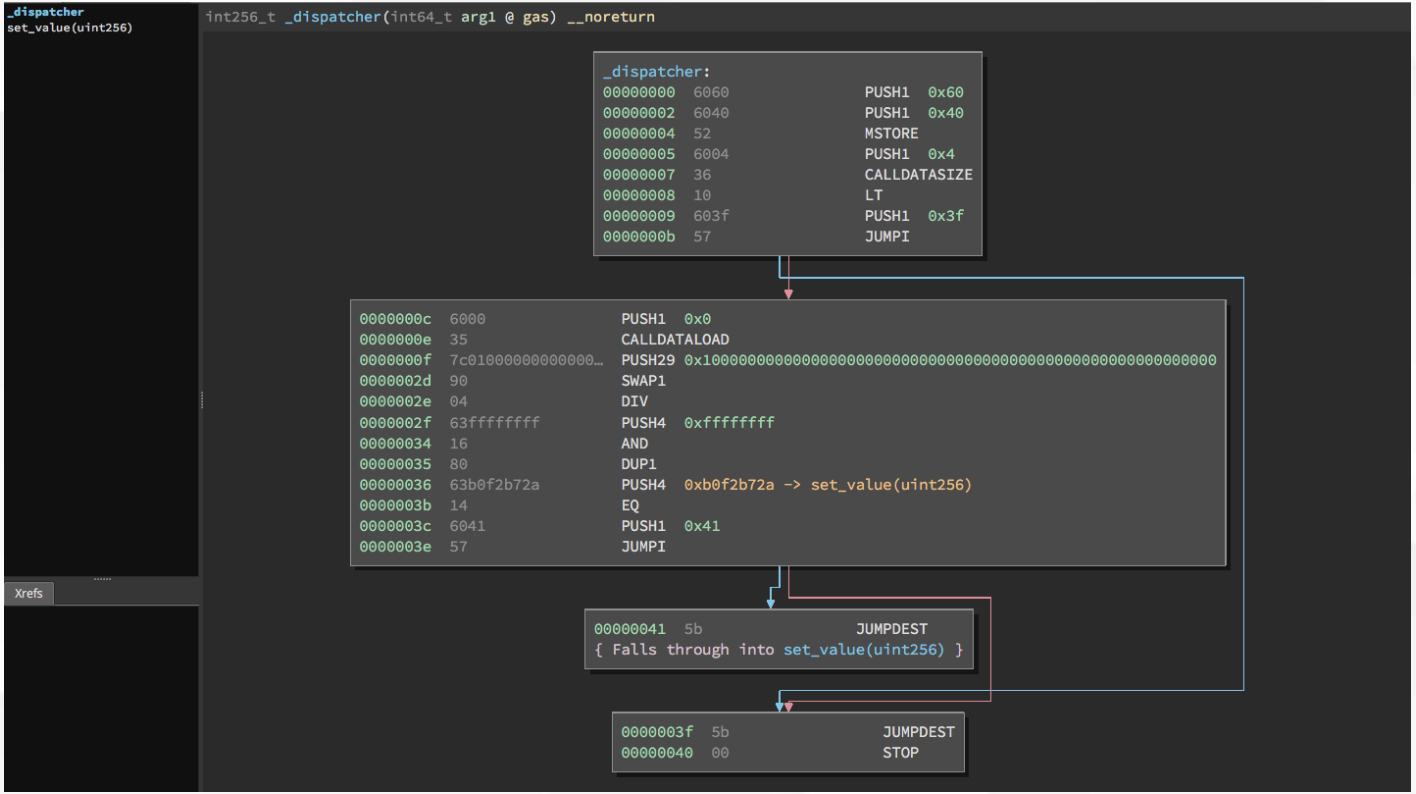
```
// c1.sol
pragma solidity ^0.4.11;

contract C {
    uint256 a;
    function C() {
        a = 1;
    }
}
```

二进制代码

```
60606040523415600e57600080fd5b5b60016000819055505b5b6036806026600039600f30
0606060405525b600080fd00a165627a7a72305820af3193f6fd31031a0e0d2de1ad2c27352b1
ce081b4f3c92b5650ca4dd542bb770029
```

```
tag_2:
// 60 01
0x1

// 60 00
0x0

// 81
dup2

// 90
swap1

// 55
sstore

// 50
pop
```

编译执行


_dispatcher
set_value(uint256)
int256_t _dispatcher(int64_t arg1 @ gas) __noreturn
_dispatcher:
00000000 6060 PUSH1 0x60
00000002 6040 PUSH1 0x40
00000004 52 MSTORE
00000005 6004 PUSH1 0x4
00000007 36 CALLDATASIZE
00000008 10 LT
00000009 603f PUSH1 0x3f
0000000b 57 JUMPI
0000000c 6000 PUSH1 0x0
0000000e 35 CALLDATALOAD
0000000f 7c0100000000000000… PUSH29 0x1000000000000000000000000000000000000000000000000000000000
0000002d 90 SWAP1
0000002e 04 DIV
0000002f 63ffffffff PUSH4 0xffffffff
00000034 16 AND
00000035 80 DUP1
00000036 63b0f2b72a PUSH4 0xb0f2b72a -> set_value(uint256)
0000003b 14 EQ
0000003c 6041 PUSH1 0x41
0000003e 57 JUMPI
Xrefs
00000041 5b JUMPDEST
{ Falls through into set_value(uint256) }
0000003f 5b JUMPDEST
00000040 00 STOP

Type Definition

Event.go：生成Event Signature

Method.go：合约Method相关

Numbers.go： 各种类型数的操作：Pack…

Type.go： geth专有的Type处理

Helper

Reflect.go ： Reflection 功能

Packing.go： 压缩功能

Abi.go：接口功能

Type Definition

Backend.go：各种Interface定义

Auth.go： 给交易签名

Base.go： 调用合约里的方法

Bind.go：将Solidity Contract语言Bind到不同的语言

Util.go： WaitforMined， WaitforDeployed

04

Resource

ABI

ABI

http://eth.guide/#resources-abi

https://github.com/ethereum/wiki/wiki/Ethereum-Contract-ABI#functions

Solidity

https://solidity.readthedocs.io/en/develop/abi-spec.html#abi-json

学堂在线
xuetangx.com

谢谢聆听

郑嘉文

2018/03