Ethereum Address

Normal Account Address

- 本质上和比特币类似
- 私钥 -> 公钥 -> Hash -> 取 20 字节

```
func PubkeyToAddress(p ecdsa.PublicKey) common.Address {
    pubBytes := FromECDSAPub(&p)
    return common.BytesToAddress(Keccak256(pubBytes[1:])[12:])
}


func FromECDSAPub(pub *ecdsa.PublicKey) []byte {
    if pub == nil || pub.X == nil || pub.Y == nil {
        return nil
    }
    return elliptic.Marshal(S256(), pub.X, pub.Y)
}
```

1. 生成 65 字节的二进制
    - 0x04 + pub.x + pub.y
2. 计算 Keccak256
    - input: 64 bytes
    - output: 32 bytes
3. 取 Hash 结果的后 20 字节，即为 Account Address

Contract Address

```
// Creates an ethereum address given the bytes and the nonce
func CreateAddress(b common.Address, nonce uint64) common.Address {
    data, _ := rlp.EncodeToBytes([]interface{}{b, nonce})
    return common.BytesToAddress(Keccak256(data)[12:])
}
```

1. 智能合约地址，在智能合约部署 Transaction 执行时生成
2. Keccak256 输入为 Transaction 发起方的 Address 和 Transaction 的 nonce
    - 虽然智能合约地址是由发起方的 Address 生成的，但是发起方的 Account 并没有因此获得对智能合约的控制权，控制权是写在智能合约代码中的
3. 取 Hash 结果的后 20 字节，即为 Contract Address