前言

- SHA（Secure Hash Algorithm，安全散列算法）是数据签名等密码学应用中的重要工具，被广发应用于信息安全领域，它是一种不可逆的加密算法，理想情况下是可以通过穷举法破解该算法，但是其破译难度和成本相当高，远高于我们熟知的MD5，它相对MD5更加安全，现已成为公认的最安全散列算法之一。其优点如下：
  - 强抗碰撞：已知原数据和其SHA值，想找到一个具有相同SHA值的数据（伪造数据）是非常困难的；
  - 容易计算：从原数据计算出SHA值比较容易；
  - 抗修改性：对原数据进行任何改动，所得到的SHA值都有很大区别；
- 公开密钥算法（Public-key cryptography）的意义在于使用了一对密钥（公钥和私钥），并且从公钥推出私钥是比较困难的。以RSA算法为例：

  它的安全性基于大数分解的难度，其公钥和私钥是一对大素数的函数。从一个公钥和密文中恢复出明文的难度等价于分解两个大素数之积。

  公开密钥 n: 两个素数p和q乘积（p和q必须加密）

  e: 与（p-1) (q-1）互素

  私人密钥 d= e^-1 (mod(p-1) (q-1))

  加密 c= m^e mod n

  解密 m= c^d mod n

- 数字签名是为了保证数据的完整性与真实性，一般分为两个步骤：
  - 第一步是产生需签名的数据的哈希值；
  - 第二步是把这个哈希值用我们的私钥进行加密，然后把这个被加密起来的哈希值添加到数据后，保护哈希结果的完整性；
  - 举个例子：

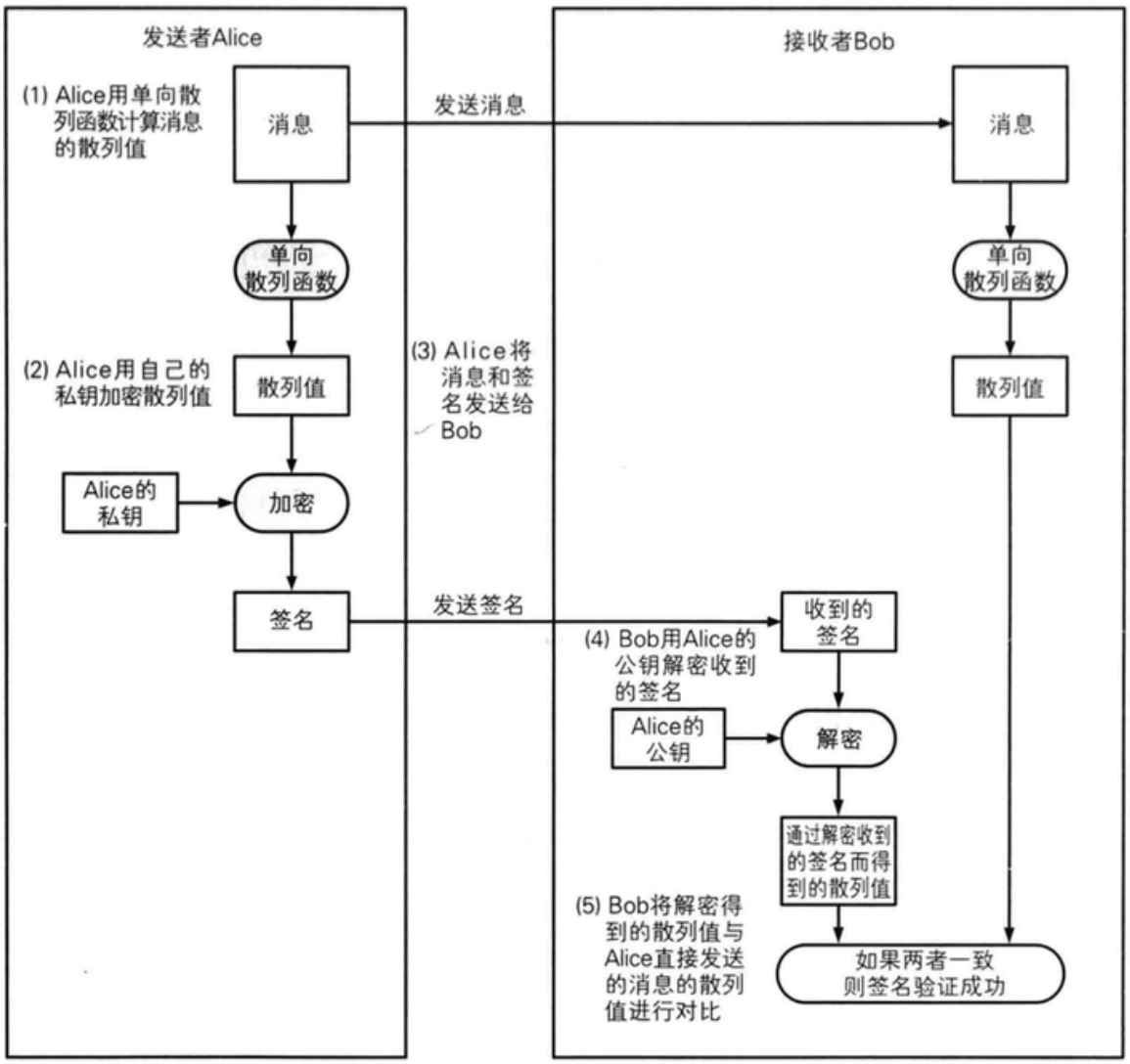

图 9-6　Alice 对消息的散列值签名，Bob 验证签名

- Ed25519第三方组件（Crypt模块没有提供该算法实现）是被认为目前最安全的签名与认证算法之一，该组件使用简单，签名认证快速，如书上所说：该组件封装了数字签名算法，签名过程不依赖随机数生成器，因此不存在时间通道攻击的问题，签名和公钥都很小，签名和验证的性能很高。

Ebookcoin中的加密和解密、签名与认证

- 首先亿书会将用户post过来的密码（secret）进行一系列处理，生成特定的密钥对：

```
privated.openAccount = function (secret, cb) {
    var hash = crypto.createHash('sha256').update(secret, 'utf8').digest();
    //计算用户所设定密码的哈希值，并以二进制的形式输出到 hash 中
    var keypair = ed.MakeKeypair(hash);
    //使用Ed25519模块对hash进行处理，生成特定的密钥对
    self.setAccountAndGet({publicKey: keypair.publicKey.toString('hex')}, cb);
};
```

- 之后亿书会对生成的公钥进行处理，以便生成用户的特定ID;

```
Accounts.prototype.generateAddressByPublicKey = function (publicKey) {
    var publicKeyHash = crypto.createHash('sha256').update(publicKey, 'hex').digest();
    //计算所生成公钥的哈希值，并以二进制的形式输出到 publicKeyHash 中
    var temp = new Buffer(8);
    //实例化一个Buffer对象，该实例代表了V8引擎分配的一段内存，是一个类似数组的对象
    for (var i = 0; i < 8; i++) {
        temp[i] = publicKeyHash[7 - i];
    }

    var address = bignum.fromBuffer(temp).toString() + 'L';
    //生成ID，该ID长度通常为20字节，加上末尾的‘L’后缀，总长度21字节
    if (!address) {
        throw Error("wrong publicKey " + publicKey);
    }
    return address;
};
```

- Nodejs中的加密和解密、签名与认证示意图：