在线支付实战

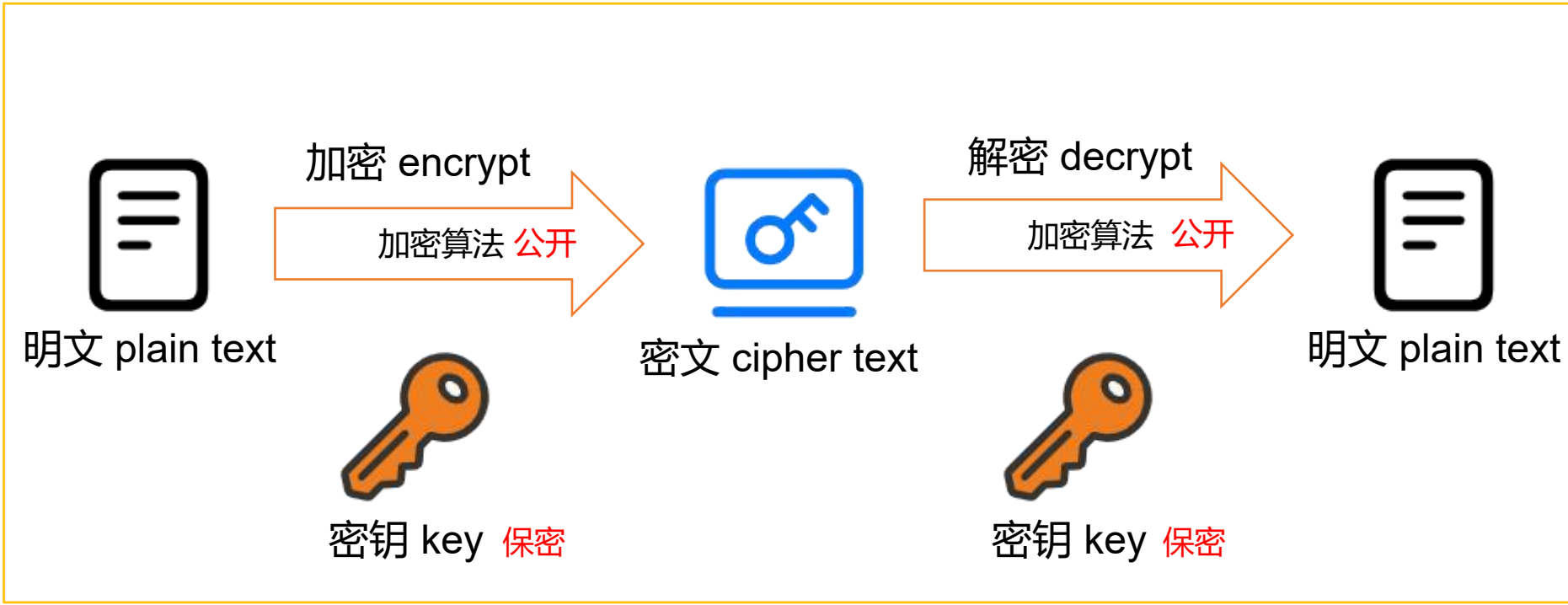
微信支付

讲师：环环

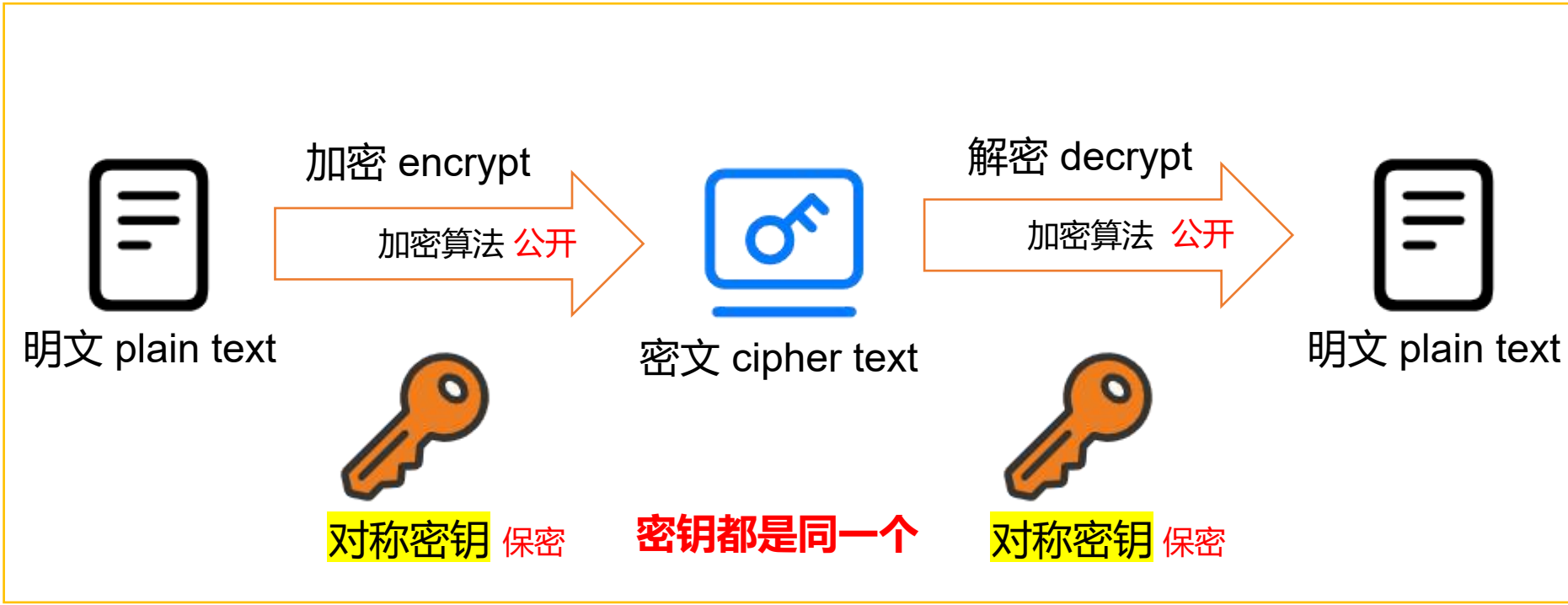
目录

1. 微信支付介绍和接入指引
2. 证书/秘钥/签名
3. 案例项目的创建
   - SpringBoot+Vue
   - Java、MyBatis-Plus、MySQL
   - HTML、JavaScript、Vue
4. 基础支付API V3
5. 基础支付API V2

1. 完整的视频、课件、笔记、源码
2. 一套教程从基础理论到应用实践，支付小白轻松入门
3. 案例专门针对支付业务开发，包含完整的API接入流程
4. 基于流行的SpringBoot+Vue架构，可直接接入常见项目
5. 涵盖API V3 和 V2，满足更多企业应用场景

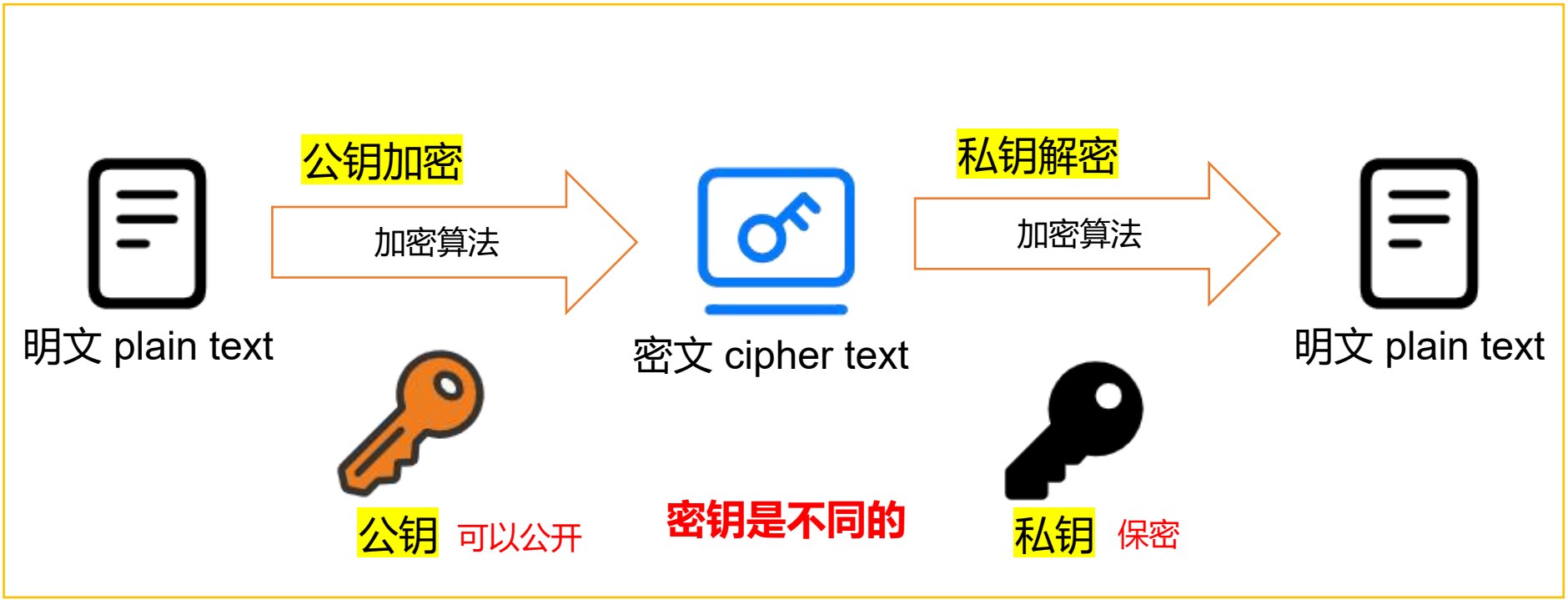


- 密钥的度量单位是位 bit，如，秘钥长度128，就是16字节的二进制串
- 按照密钥的使用方式，加密可以分为两大类：对称加密和非对称加密



- AES加密算法，密钥长度128、192或256，安全强度很高，性能很好
- 加密分组模式：将明文分组加密，微信支付中使用 AEAD_AES_256_GCM

加密 - 非对称加密



- 使用公钥加密后只能用私钥解密，反过来，私钥加密后也只能用公钥解密
- RSA加密算法：最著名的非对称加密算法

对称加密

- 优点：运算速度快
- 缺点：秘钥需要信息交换的双方共享，一旦被窃取，消息会被破解

非对称加密

- 优点：私钥严格保密，公钥任意分发，黑客获取公钥无法破解密文
- 缺点：运算速度非常慢

Bob

(Bob's public key)

(Bob's private key)

Pat Doug Susan

任何人都可以得到Bob的公钥，但是只有Bob自己拥有私钥

Susan

写信

"Hey Bob, how about lunch at Taco Bell. I hear they have free refills!"

(Bob's public key)

Encrypt with Public Key

HNFmsEm6Un
BejhhyCGKOK
JUxhiygSBCEiC
0QYIh/Hn3xgiK
BcyLK1UcYiY
lxx2ICFHDC/A

Bob

收信

HNFmsEm6Un
BejhhyCGKOK
JUxhiygSBCEiC
0QYIh/Hn3xgiK
BcyLK1UcYiY
lxx2ICFHDC/A

(Bob's private key)

Decrypt with Private Key

"Hey Bob, how about lunch at Taco Bell. I hear they have free refills!"

Bob

回信

"Hey Susan, See you at noon!"

(Susan's public key)

Encrypt with Public Key

yEGDuoj/0F4
MoF8PyYN5Me
jcHXF2NTPzF
0A3+PfqKlQ=

Susan

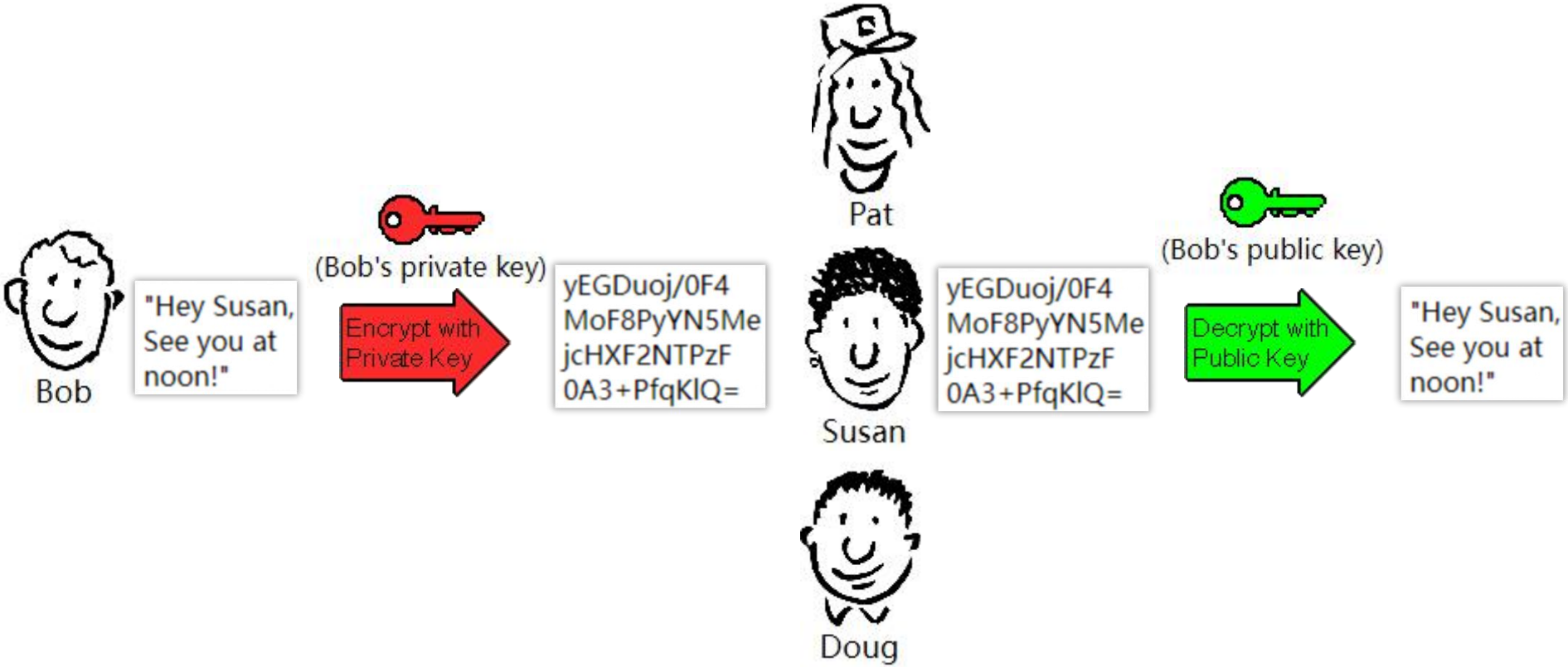
收信

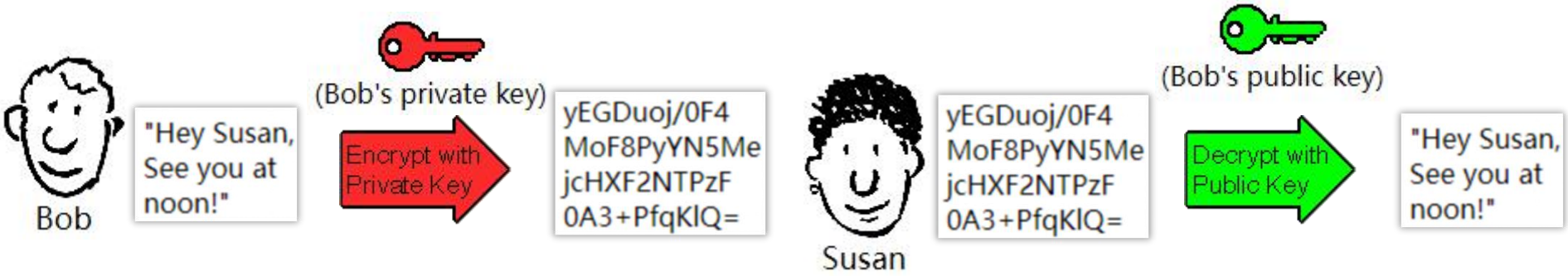
yEGDuoj/0F4
MoF8PyYN5Me
jcHXF2NTPzF
0A3+PfqKlQ=

(Susan's private key)

Decrypt with Private Key

"Hey Susan, See you at noon!"


Pat
(Bob's private key)
(Bob's public key)
Bob
"Hey Susan, See you at noon!"
Encrypt with Private Key
yEGDuoj/0F4
MoF8PyYN5Me
jcHXF2NTPzF
0A3+PfqKlQ=
Susan
yEGDuoj/0F4
MoF8PyYN5Me
jcHXF2NTPzF
0A3+PfqKlQ=
Decrypt with Public Key
"Hey Susan, See you at noon!"
Doug

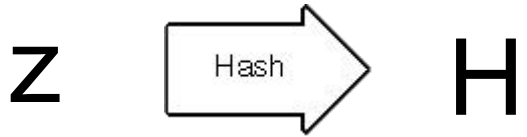

Bob
"Hey Susan, See you at noon!"
(Bob's private key)
Encrypt with Private Key
yEGDuoj/0F4
MoF8PyYN5Me
jcHXF2NTPzF
0A3+PfqKlQ=
Susan
yEGDuoj/0F4
MoF8PyYN5Me
jcHXF2NTPzF
0A3+PfqKlQ=
(Bob's public key)
Decrypt with Public Key
"Hey Susan, See you at noon!"

摘要算法和数据的完整性

如何保证信件不被篡改，即信息的完整性？

实现完整性的手段主要是摘要算法（Digest Algorithm）

也就是常说的散列函数、哈希函数（Hash Function）



任意长度

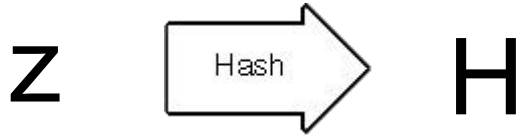
固定长度

“a” => 0cc175b9c0f1b6a831c399e269772661

“aaa” => 47bce5c74f589f4867dbd57e9ca9f808

数据指纹、摘要（Digest）

摘要算法和数据的完整性



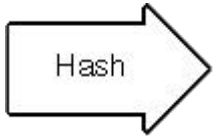
任意长度　　固定长度

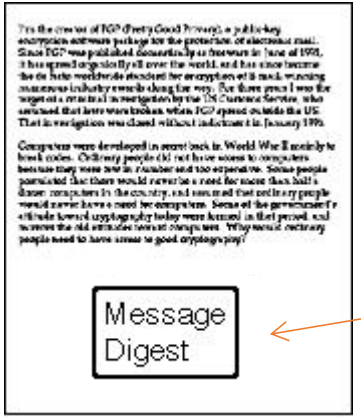
数据指纹、摘要（Digest）

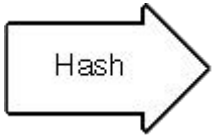
1. 不可逆：只有算法，没有秘钥，只能加密，不能解密
2. 难题友好性：想要破解，只能暴力枚举
3. 发散性：只要对原文进行一点点改动，摘要就会发生剧烈变化
4. 抗碰撞性：原文不同，计算后的摘要也要不同

常见摘要算法：MD5、SHA1、SHA2（SHA224、SHA256、SHA384）

摘要算法和数据的完整性

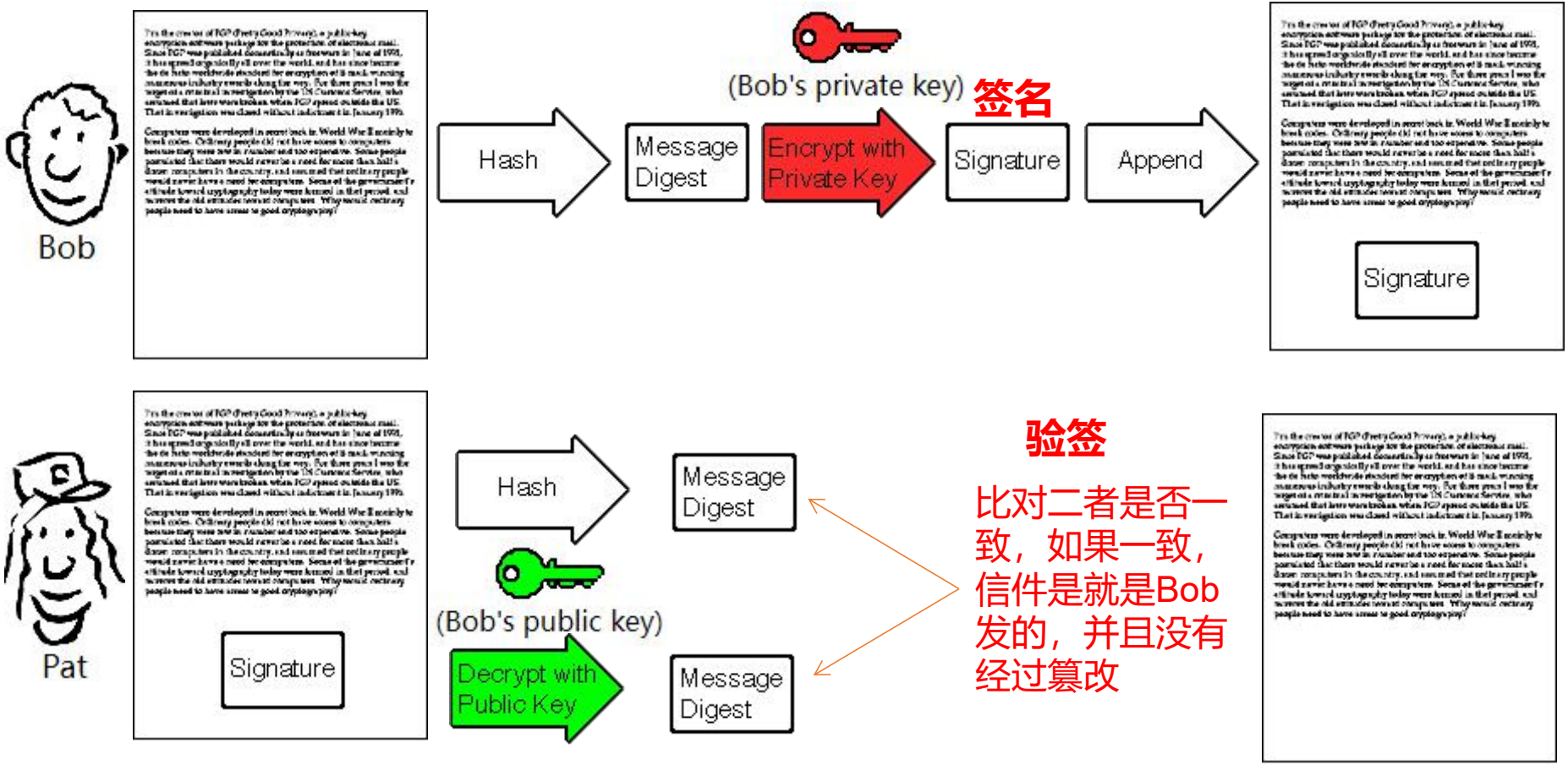




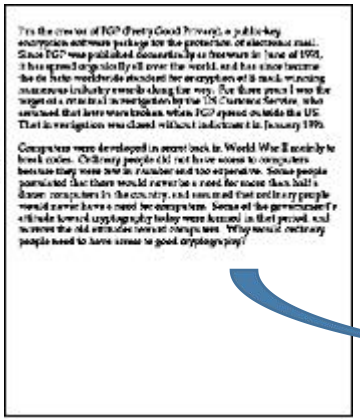


比对二者是否一致，如果一致，信件是就是Bob发的，并且没有经过篡改

数字签名


Bob
(Bob's private key)
签名
Hash
Message Digest
Encrypt with Private Key
Signature
Append
Signature
Pat
Hash
Message Digest
(Bob's public key)
Signature
Decrypt with Public Key
Message Digest
验签
比对二者是否一致，如果一致，信件是就是Bob发的，并且没有经过篡改

数字签名 + 对称加密消息原文



(Bob's private key)

签名

Hash → Message Digest → Encrypt with Private Key → Signature → Append

加密

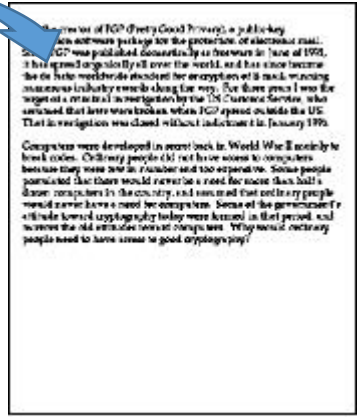
对称加密

解密

Hash → Message Digest

(Bob's public key)

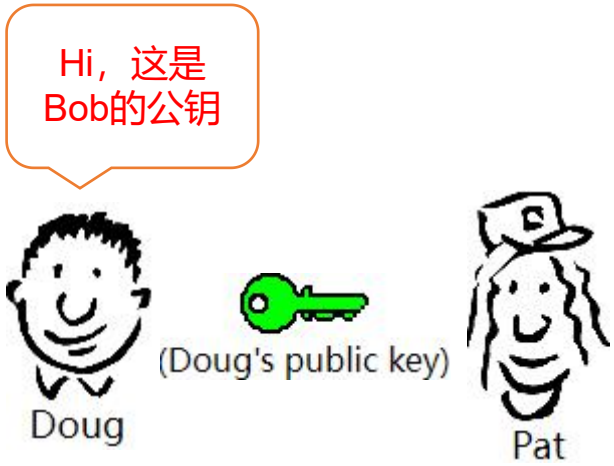
Decrypt with Public Key → Message Digest

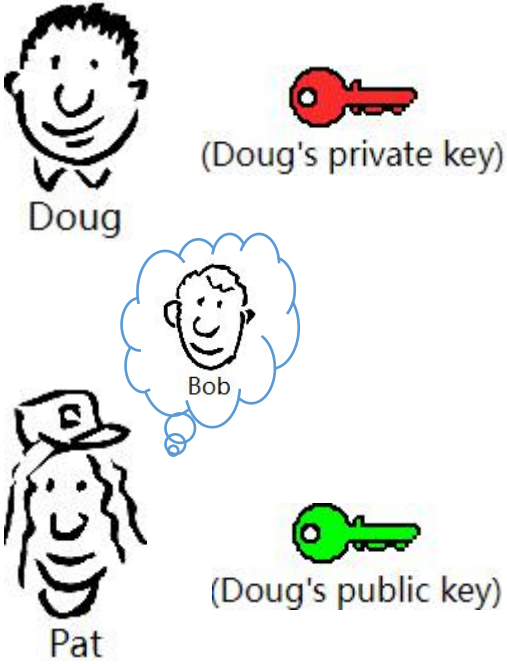


验签

比对二者是否一致，如果一致，信件是就是Bob发的，并且没有经过篡改

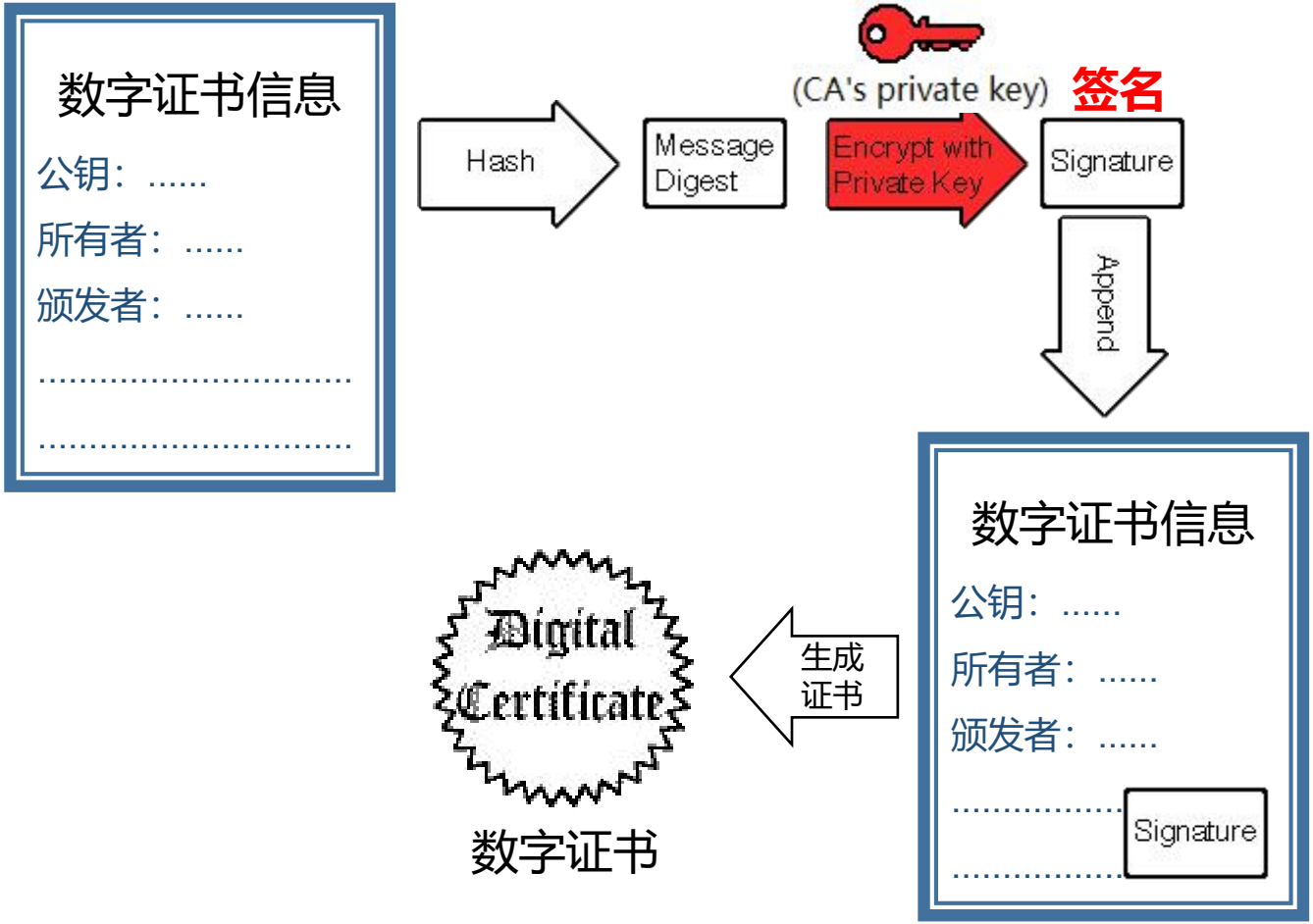
数字证书





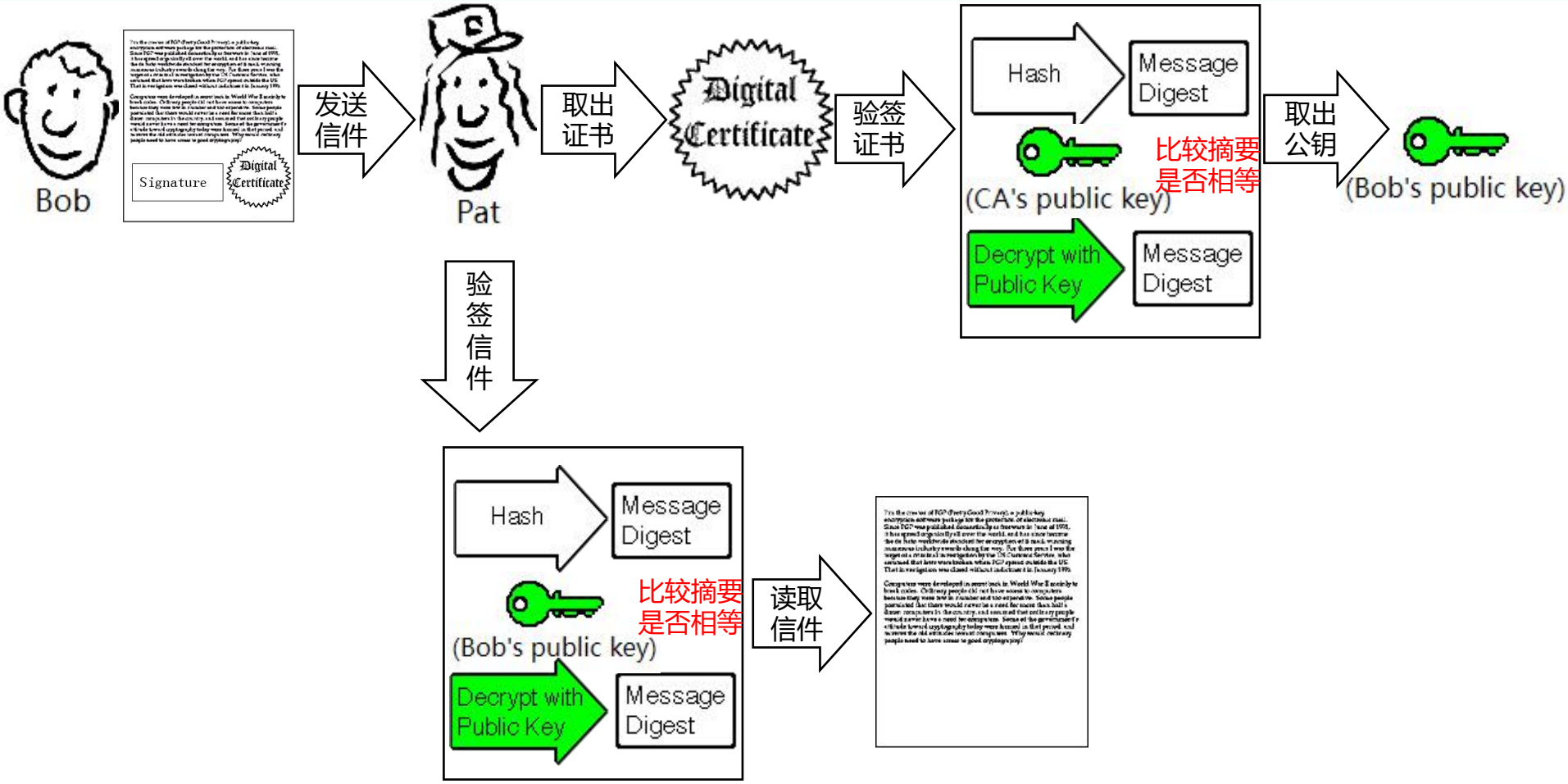
公钥的信任：黑客可以伪造公钥，怎么判断公钥是真实的？

- 公钥：Bob的公钥
- 所有者：Bob
- 颁发者：CA（Certificate Authority，证书认证机构）
- 有效期：证书的使用期限
- 签名哈希算法：指定摘要算法，用来计算证书的摘要
- 指纹：证书的摘要，保证证书的完整性
- 签名算法：用于生成签名，确保证书是由CA签发
- 序列号：证书的唯一标识


数字证书信息
公钥：.....
所有者：.....
颁发者：.....
.....
.....
Hash
Message Digest
(CA's private key)
签名
Encrypt with Private Key
Signature
Append
数字证书信息
公钥：.....
所有者：.....
颁发者：.....
.....
Signature
.....
生成证书
Digital Certificate
数字证书

数字证书

数字证书


Bob
Signature
Digital Certificate
发送信件
Pat
取出证书
Digital Certificate
验签证书
Hash
Message Digest
(CA's public key)
比较摘要是否相等
Decrypt with Public Key
Message Digest
取出公钥
(Bob's public key)
验签信件
Hash
Message Digest
(Bob's public key)
比较摘要是否相等
Decrypt with Public Key
Message Digest
读取信件

数字证书 - https协议

发送加密请求

申请数字证书

CA

响应加密网页
和数字证书

颁发数字证书：

CA用私钥加密数字证书并签名

数字证书中存有网站的公钥

获取网站公钥：

使用CA的公钥解密数字证书并验签

CA的公钥默认安装在操作系统中

判断网址与证书中的网址是否一致

是不是权威机构颁发的数字证书

网站公钥

与服务器交换加密信息

步骤：

1. 创建SpringBoot项目（Java、SpringBoot、SpringMVC、RESTful、json）
2. 引入Swagger（接口文档和测试页面生成工具）
1. 定义统一结果（让前后端数据通信更规范）
3. 创建和连接数据库（MySQL、IDEA内置的数据库管理工具）
4. 集成MyBatis-Plus（MyBatis）
5. 搭建前端环境（了解HTML和CSS、熟悉JavaScript、了解Vue）
6. 认识 Vue.js

内容：

1. 引入支付参数
2. 加载商户私钥
3. 获取平台证书和验签器
4. 获取HttpClient对象
5. API字典和接口规则
6. 内网穿透
7. API v3

内容：

1. 整合APIv2
2. APIv2和APIv3的区别
3. Native下单
4. 支付结果通知

谢谢观看