第 33 卷 第 11 期 计 算 机 工 程 2007 年 6 月
Vol.33 No.11 Computer Engineering June 2007
·安全技术· 文章编号：1000—3428(2007)11—0141—03 文献标识码：A 中图分类号：TP393.08

基于.NET 的双重签名演示实验的设计与实现

杨 艳，周 靖，王 鲁
(解放军信息工程大学电子技术学院，郑州 450004)

摘 要：双重签名是 SET 协议中对数字签名的新应用，解决了电子商务交易中消费者、商家及银行三方的安全通信问题。文章分析了 SET 协议中双重签名的工作原理，给出了一个双重签名演示实验的设计方案，并利用 Microsoft.NET 框架中提供的加密模型，使用 C#语言实现了双重签名的生成及商家和银行的验证过程。

关键词：SET；双重签名；.NET 框架；C#

Design and Implementation of Dual Signature Presentation Experiment Based on .NET

YANG Yan, ZHOU Jing, WANG Lu
(Institute of Electronic Technology, PLA Information Engineering University, Zhengzhou 450004)

【Abstract】 The dual signature is a new application of digital signature in SET protocol. It resolves security correspondence of the consumer, business and bank in electronic transaction. This paper analyzes the dual signature technology, puts forward a design project of dual signature presentation experiment, and realizes it by using the developing tool C# based on cryptography model of .NET framework.

【Key words】 SET; Dual signature; .NET Framework; C#

《网络安全通信协议》是我院信息安全专业学生的必修课程，按照 TCP/IP 的安全架构，讲述了目前常用的一些经典安全通信协议，安全电子交易(secure electronic transaction, SET)协议是其中重要的一个协议。SET 协议专门针对电子商务交易的安全标准，通过采用一系列的安全机制，保证了电子商务交易中数据的机密性、完整性、身份鉴别及不可抵赖性。其采用的核心技术包括 X.509 电子证书标准、数字签名、报文摘要、数字信封以及双重签名等技术。其中双重签名是数字签名的新应用，也是 SET 中引入的重要创新。

由于 SET 协议的原理相对比较复杂，而且由于实际环境限制，学生不能接触到实际的应用环境，因此理解起来相对比较困难。双重签名是 SET 协议的安全措施中非常重要的一部分，因此根据双重签名的基本原理，给出了一种双重签名演示实验的设计及实现，通过演示，加深学生理解 SET 协议的基本原理。

1 SET协议中双重签名的工作原理[1]

1.1 双重签名的生成

在 SET 协议中，交易时，客户需要发送订购信息(order information, OI)给商家，发送支付信息(payment information, PI)给银行。而客户发往银行的支付指令是通过商家转发的，为了避免在交易的过程中商家窃取客户的信用卡信息，以及避免银行跟踪客户的行为，侵犯消费者隐私，商家不需要知道客户的信用卡信息，银行也不需要知道客户定单的细节。但同时又不能影响商家和银行对客户所发信息的合理的验证，因此 SET 协议采用双重签名来解决这一问题。

图 1 显示了双重签名的生成过程。客户生成支付信息消息摘要(用SHA1)(payment information message digest, PIMD)和订购信息消息摘要(order information message digest, OIMD)，将PIMD和OIMD连接后再哈希，生成支付订购消息摘要(payment order message digest, POMD)。最后，客户用他的签名私钥加密最终的哈希，生成双重签名(dual signature, DS)。操作可以总结如下：$DS=E_{KRc}[H(H(PI)\|H(OI))]$，这里$KR_C$是客户的签名私钥。

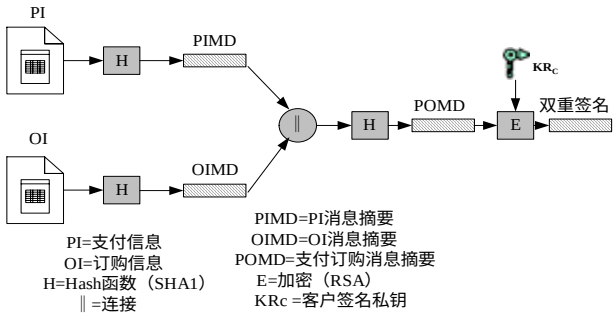

图 1 双重签名的生成

1.2 双重签名的发送

SET 协议中双重签名包含在客户向商家发送的购买请求消息中，购买请求消息如图 2 所示。在购买请求消息中，客户发送给商家双重签名 DS、PIMD、OI 及客户证书。同样，客户要传给银行 DS、PI、OIMD 及客户证书。

由于支付的相关信息是由商家转发给银行的，因此客户需要生成一个临时对称密钥K_S，加密DS、PI、OIMD，并用支付网关(收款银行所有，用来完成SET协议和现有银行交易系统之间的信息格式转换)的加密公钥加密K_S形成数字信封。由于商家不能得到K_S的值，因此商家不能读出任何与支付相关的信息，只能对消费者的订购信息解密，而银行只能对支

作者简介：杨 艳(1973－)，女，讲师、硕士，主研方向：网络安全；周 靖，硕士生；王 鲁，副教授

收稿日期：2006-09-10 **E-mail：**yangyan302@sohu.com

付信息解密，充分保证消费者的支付信息和定购信息的安全。

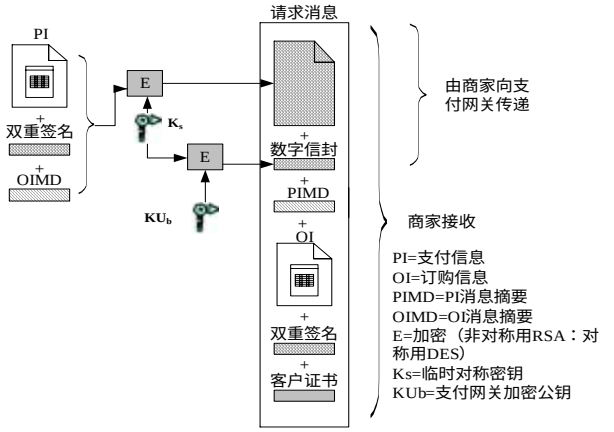

图 2 客户发送购买请求消息过程

1.3 双重签名的验证

(1)商家验证

商家收到购买请求消息后，从客户的证书中获得客户的签名公钥，进行验证，验证过程如图 3 所示。商家验证并计算如下数量：

H(PIMD ‖ H(OI))和D_{KUc}[DS]

其中KU_C是客户的签名公钥，若两个数量相等，商家就可以确认定单在传输过程中没有损害，并用客户的签名私钥签署。

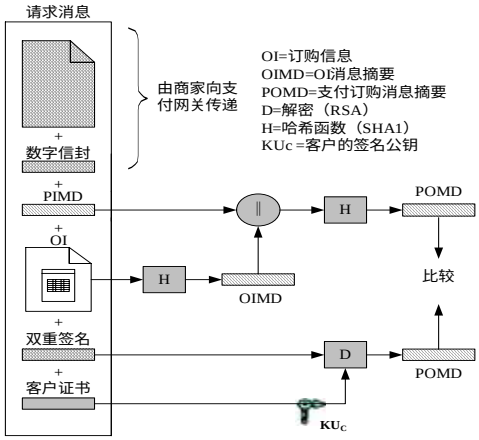

图 3 商家验证过程

(2)银行验证

银行用自己的加密私钥解密数字信封，得到对称密钥，解密支付相关信息，得到 PI、OIMD 和 DS，则银行可以计算：

H(H(PI) ‖ OIMD)和D_{KUc}[DS]

其中KU_C是客户的签名公钥，如果两个数量相等，那么银行也可以确认签名。

双重签名总结如下：

1)商家接收 OI 并验证签名。

2)银行接收 PI 并验证签名。

3)客户连接 OI 和 PI，并确保连接。

例如，假设商家为了其利益想在这次交易中用另一个 OI 代替，就必须发现另一个哈希与存在的 OIMD 匹配的 OI，用 SHA1 一般认为是不可行的。对于银行也是同样的。

2 .NET 概述

Microsoft.NET(以下简称.NET)是微软推出的下一代基于互联网平台的软件开发构想，.NET给开发人员带来了一种全新的开发框架——.NET框架(.NET Framework)。.NET 框架主要用来产生一个可以用来快速开发、部署网站服务及应用程序的开发平台[2]。

2.1 .NET 框架中的加密模型

.NET 框架的System.Security.Cryptography命名空间提供加密服务，包括安全的数据编码和解码，以及许多其它操作，如散列法、随机数字生成和消息身份验证。.NET 框架提供多种标准密码算法的实现，表 1 显示了.NET 框架中提供的一些实现相应算法的类[3]。

表 1 .NET 框架中一些实现相应算法的类

加密算法分类	具体加密算法	.NET 框架中提供的实现类
对称算法	DES	DESCryptoServiceProvider
	TripleDES	TripleDESCryptoServiceProvider
	Rijndael	RijndaelManaged
	RC2	RC2CryptoServiceProvider
不对称算法	DSA	DSACryptoServiceProvider
	RSA	RSACryptoServiceProvider
哈希算法	HMAC SHA1	HMACSHA1
	SHA1, SHA256, SHA384, SHA512	SHA1Managed SHA256Managed SHA384Managed SHA512Managed
	MD5	MD5CryptoServiceProvider
	MAC Triple DES	MACTripleDES

因为.NET 中所提供的算法具有良好定义的继承层次结构，所以对这些算法可轻松扩展，方便地将加密安全解决方案整合到 .NET 应用程序之中。

2.2 面向.NET 的全新开发工具——C#

虽然.NET 开发框架支持多种语言，但微软公司仍创造了 Visual C#.NET(简称 C#，读作 C Sharp)作为.NET 框架的主力开发语言，这是由 C#的特点决定的。C#来源于 C 和 C++，具有现代、简单、新型、面向对象而且类型安全等特点。由于继承了 C++的大量语法，C#具有 C++的灵活性及强大的底层控制能力。C#运行于.NET 平台，同时，.NET 平台的大部分类库都是用 C#来进行开发的。

3 基于.NET 的双重签名演示实验的设计

根据双重签名的基本原理，我们利用.NET 框架的 System.Security.Cryptography 命名空间中提供的 RSA 非对称密钥算法以及 SHA1 哈希算法的实现来模拟双重签名的生成与验证过程。演示实验主要包括以下 4 个模块：

(1)公私钥对生成模块

在 SET 支付时，该模块用于生成公私钥对，既可以生成用于签名的公私钥对，又可以生成用于加密的公私钥对。由于只是模拟，因此公私钥对生成后，以文件的形式传给收方。

(2)双重签名生成模块

该模块用于消费者填写订购信息与支付信息，并生成订购信息及支付信息的摘要，将两个摘要连接后再哈希，用公私钥对生成模块中生成的客户签名私钥进行签名，最终生成双重签名。同时生成传送给商家和银行的信息。

(3)商家验证模块

该模块用于商家验证订购信息与支付信息的关联性。通过收到的客户的签名公钥、支付信息的摘要与订购信息，重新计算双重签名，并与收到的双重签名进行比较，来进行验证。

(4)支付网关验证模块

该模块用于支付网关验证订购信息与支付信息的关联性。收到商家转发过来的信息，用支付网关的加密私钥解密数字信封，得到对称密钥，解密支付相关信息，得到支付信

息、订购信息的摘要及双重签名，然后进行验证。

4 基于.NET 的双重签名演示实验的实现

使用 C#语言基于 Microsoft .NET Framework1.1 来实现上述 4 个模块。

4.1 公私钥对生成

密钥的生成采用System.Security.Cryptography命名空间中的RSACryptoSericeProvider类来生成随机的密钥，密钥存储采用XML方式[4]。在本设计方案中需要生成消费者用于签名的公私钥对，以及支付网关用于加密的公私钥对。部分代码如下：

```
RSACryptoServiceProvider rsaProvider = new
RSACryptoServiceProvider();
XmlDocument xmlDoc = new XmlDocument();
xmlDoc.LoadXml( rsaProvider.ToXmlString(true) );
//生成公私钥对
xmlDoc.Save( PublicAndPrivateKeyPath );
//将公私钥对保存到指定路径下的文件
xmlDoc.LoadXml( rsaProvider.ToXmlString(false) );//生成公钥
xmlDoc.Save( PublicKeyPath );
//将公钥保存到指定路径下的文件
```

上面的代码生成两个 XML 格式的文件，公私钥信息存储到公私钥文件中，该文件中保存有完整的公私钥信息；公钥信息则存储至公钥文件，该文件只包含公钥信息。

4.2 双重签名生成

双重签名的生成共分为 5 步：

(1)订购信息和支付信息编码

在 SET 支付时，主要传递的数据为支付信息和订购信息。因此实验时首先填写订购信息和支付信息，订购信息包括商品名称、单价、数量，支付信息中用户只需填写银行账号，支付总金额自动计算。在双重签名生成前，必须进行编码。程序采用 Unicode 编码方式，首先将用户填写的订购信息采用“||”连接，形成一个订购信息字符串，形式为“商品名称||单价||数量”，然后采用 System.Text 命名空间中的 Unicode 类对其进行编码。同样，支付信息形式为“银行账号||支付金额”，也使用 Unicode 进行编码。部分代码如下：

```
string OIStr = this.TextBoxName.Text + "||" +
this.TextBoxPrice.Text + "||" + this.TextBoxNum.Text; //订购信息
string PIStr = this.TextBoxBankNum.Text + "||" +
this.TextBoxSum.Text; //支付信息
byte[] OIBytes = Encoding.Unicode.GetBytes( OIStr );
//订购信息编码
byte[] PIBytes = Encoding.Unicode.GetBytes( PIStr );
//支付信息编码
```

(2)生成哈希

使用 System.Security.Cryptography 中的 SHA1CryptoServiceProvider 类对(1)中生成的两个编码分别进行散列，生成的两个哈希为订购信息的哈希和支付信息的哈希，将两个哈希值采用“&&”的 Unicode 编码连接。部分代码如下：

```
SHA1CryptoServiceProvider sha1 = new
SHA1CryptoServiceProvider();
this.TextBoxLog.Text += ("生成订购信息哈希" +
Environment.NewLine );
byte[] OIHashBytes = sha1.ComputeHash( OIBytes );
//生成订购信息哈希
this.TextBoxLog.Text += ("生成支付信息哈希" +
Environment.NewLine );
byte[] PIHashBytes = sha1.ComputeHash( PIBytes );
//生成支付信息哈希
```

(3)生成双重签名

将订购信息的哈希和支付信息的哈希连接，再进行散列，得到订购信息与支付信息的哈希。然后消费者从“公私钥对生成”程序中生成的公私钥文件中读取签名私钥，用私钥对连接后生成的哈希进行数字签名，最终形成双重签名。将双重签名存储在双重签名文件中。部分代码如下：

```
this.TextBoxLog.Text += ("生成哈希并进行数字签名" +
Environment.NewLine );
byte[] SignedPOI = rsaProvider.SignData( POIBytes , sha1 );
//生成双重签名
this.TextBoxLog.Text += ("生成双重签名完毕." +
Environment.NewLine);
```

(4)向商家传递的信息

将支付信息的哈希与订购信息采用“&&”的 Unicode 编码连接，保存在订购信息文件中，生成的连接是发送给商家的信息；向商家传递双重签名、支付信息的哈希与订购信息、消费者的签名公钥。

(5)向支付网关传递的信息

将订购信息的哈希、支付信息及双重签名采用“&&”的 Unicode 编码连接，保存在支付信息文件中；用产生的临时对称密钥加密，消费者使用支付网关的加密公钥将对称密钥加密后形成数字信封，数字信封存储在数字信封文件中。消费者向支付网关传递的信息为数字信封、加密的支付信息文件、消费者的签名公钥。部分代码如下：

```
//生成数字信封
XmlTextReader xmlReader = new XmlTextReader
( openFileDialog.FileName );
xmlReader.WhitespaceHandling = WhitespaceHandling.None;
xmlReader.Read();
rsaProviderForPayGate = new RSACryptoServiceProvider();
rsaProviderForPayGate.FromXmlString
( xmlReader.ReadOuterXml() ); //读取支付网关公钥
byte[] DigitalEnvelopBytes =
rsaProviderForPayGate.Encrypt( SessionKeyBytes , false );
DigitalEnvelopFileWriter.Write( DigitalEnvelopBytes );
this.TextBoxLog.Text += ("数字信封保存至"+ DigitalEnvelop
+Environment.NewLine);
```

4.3 商家的验证

商家要想验证订购信息与支付信息的关联性需要 4 种信息：消费者的签名公钥，支付信息的摘要，订购信息以及双重签名。验证步骤如下：

(1)从订购信息文件中通过“&&”分隔符得到支付信息的哈希与订购信息字符串，然后通过识别“||”获取订单信息的具体内容，重现订单。

(2)商家使用 SHA1 算法对订购信息进行哈希，并将其与(1)中取得的支付信息的哈希连接在一起，再次哈希，得到本地计算出的支付信息哈希与订购信息哈希连接的哈希。

(3)使用消费者的签名公钥对双重签名进行解签名，然后和(2)中得到的哈希进行比较，如果相同，则验证成功，并显示订购信息，否则验证失败。

（下转第 150 页）