Digital Signatures & Digital Envelopes

By Mohan Atreya (matreya@rsasecurity.com)

Summary

This is the fifth article in this series. In this article, we discuss about the basics of Digital Signatures and Digital Envelopes. We will also see how they can be used to provide security to application frameworks being used in the real world.

Introduction

This tutorial assumes that the reader is familiar with basic terms in cryptography such as Public Key cryptography, Secret Key cryptography and Message Digest algorithms. Please review these concepts in Articles 1, 2, 3 & 4 of this series before proceeding further with this tutorial. Alternatively, follow the link (**http://www.rsasecurity.com/rsalabs/faq/index.html/**) for a set of frequently asked questions (FAQ's) on e-security from RSA Laboratories.

Security protocols such as SET and S/MIME are based on the concept of digital signatures and digital envelopes. SET has been designed carefully to prevent the merchant from viewing the cardholder's credit card number. Cardholders who use SET put their card numbers in an envelope that can be opened only by the card-processing center, not the merchant. For further information on SET, please follow the link (**http://www.setco.org**). The S/MIME protocol uses digital envelopes and signatures to ensure the authenticity, privacy, and integrity of e-mail. Newer messaging frameworks implementing P2P, B2B and B2C also use these concepts in a big way.

Digital Signatures

A Digital Signature identifies the signer and ensures the integrity of the signed data. Figure 1 describes the step-by-step process of creating digitally signed data. To create a digital signature, the sender needs access to his private key. It should be noted that only message digest of the message is encrypted using the signer's private key. This makes sense because the actual message might be extremely large and public key operations can be extremely slow. Moreover, by signing the message digest of the message rather than the message itself, we are also ensuring the integrity of the data.

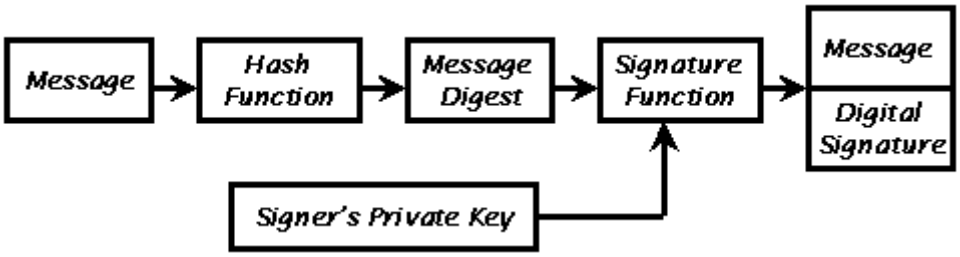

Figure 1: The process used to create a Digital Signature.

The receiver does not need access to any secret piece of information to be able to verify the digital signature. Figure 2 gives the step-by-step process of verifying a digital signature.

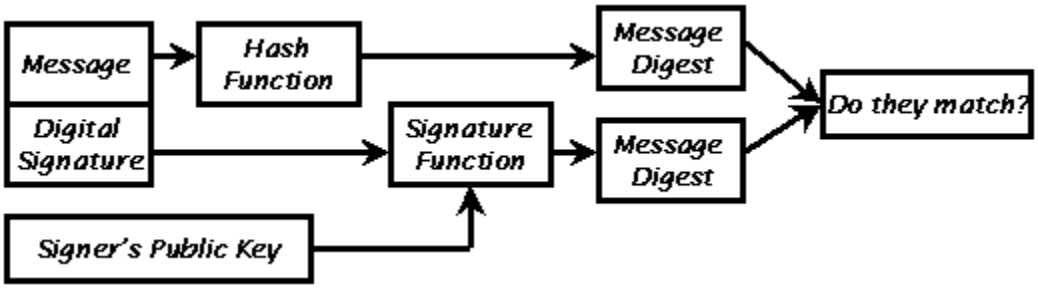

Figure 2: The process used to verify a Digital Signature.

The signature also gives us the added feature of non-repudiation. Digital Signatures are usually time-stamped before they are actually signed. This ensures that the receiver would also be able to verify when the data was actually signed. Of course, the correctness of the time-stamp would be very questionable if the sender generates it internally. Certain third party time-stamping services provide digitally signed timestamps, which can be included in a digital signature. It should be noted that some of these mechanisms are open to collusion.

When it comes to the RSA algorithm, it is important to note that due to the mathematics involved, the signature verification process is always faster than the signature generation process. With digital signatures, we can make sure that we authenticate the sender and ensure the integrity of the data as it travels over the public network. Digital signatures do not give the user data privacy. This can be addressed by using digital signatures in conjunction with digital envelopes.

Digital Envelopes

With digital signatures, the data is transmitted in the clear. This is not acceptable for certain systems, which transmit and receive extremely sensitive data. It is therefore critical that we somehow ensure that the data transmitted over the network is private (data privacy). Digital Envelopes can address this requirement by using both symmetric and asymmetric encryption algorithms.

Digital envelopes use an one-time, symmetric ***Data Encryption Key*** (DEK or a Session Key) for bulk data encryption. The symmetric DEK has to be transmitted over the wire to the receiver so that he is able to decrypt the encrypted data. The DEK is transported to the receiver after encrypting it with the Receiver's Public Key. The DEK is used only once for transmitting that particular message.

To be able to create the digital envelope, the sender only needs access to the receiver's public key. Figure 3 describes the exact procedure used to create a digital envelope. Note that any of the symmetric algorithms (DES, 3-DES, RC4 etc.) can be used to encrypt the message. However, It should be noted that the receiving party must also have access to the same symmetric algorithm.

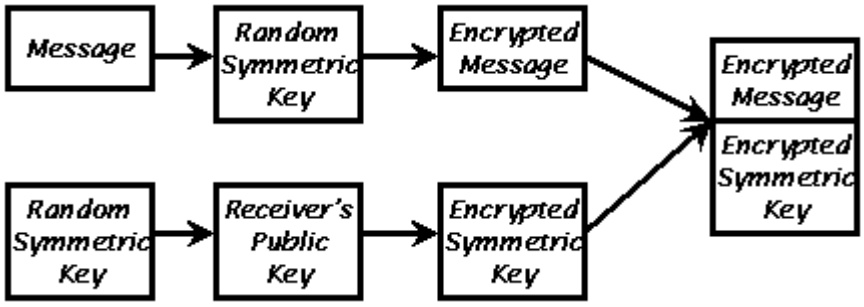

Figure 3: The process used to create a Digital Envelope.

To be able to verify/unwrap the digital envelope, the receiver needs to have access to a private piece of information- his Private Key. This ensures that only the authorized person is able to decipher the digital envelope. Figure 4 describes the step-by-step process used to decipher the digitally enveloped data.

It should be noted that this mechanism does not cater for the automatic identification of the algorithms used while verifying the digital envelope. So, the sender needs to somehow transmit this information to the user using some out-of-band methods. In-band methods cannot be used because the received data is strongly encrypted.

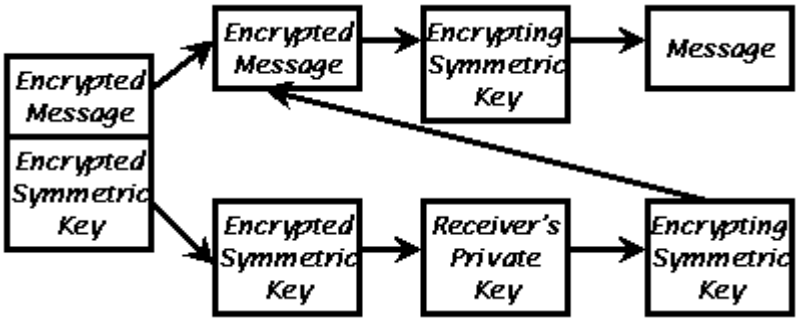

Figure 4: The process used to verify a Digital Envelope.

Simple digital envelopes ensure the privacy of data but do not unsure the authenticity, non-repudiation or the integrity of the data. This can be easily achieved by digitally signing the data before wrapping it with a digital envelope. So, in effect, the Digital Envelope carries the Digital Signature imparting data confidentiality features to a Digital Signature. Figure 5 shows the step-by-step process of creating signed & enveloped data.

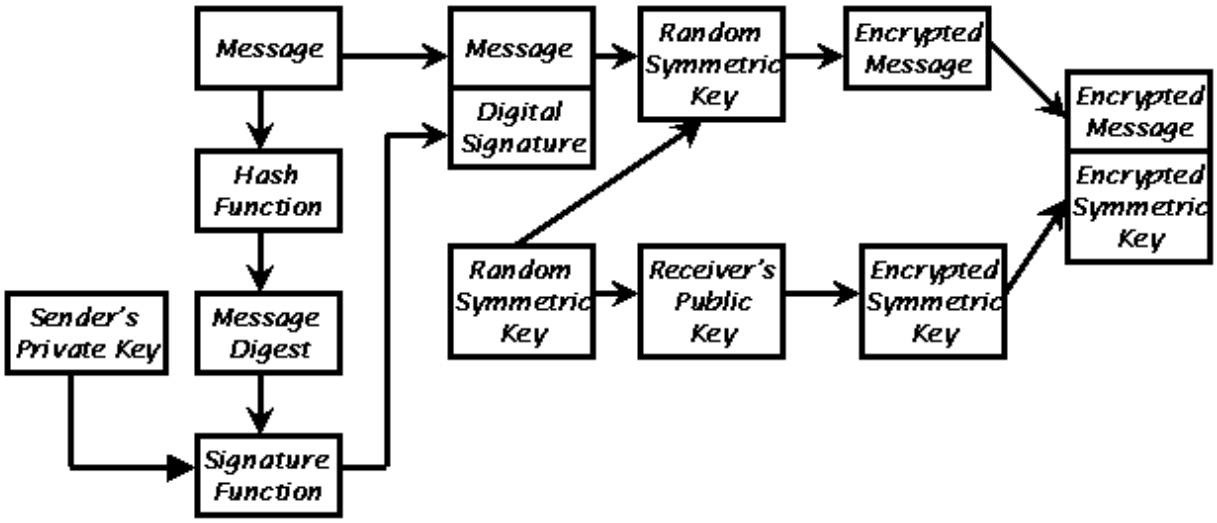

Figure 5: The process used to create a digital envelope carrying digitally signed data.

At the receiver's side, the signature can be viewed or verified only by authorized person because only he would have access to the corresponding private key to unwrap the DEK. Figure 6 describes a step-by-step procedure of the unwrapping of enveloped & signed data.

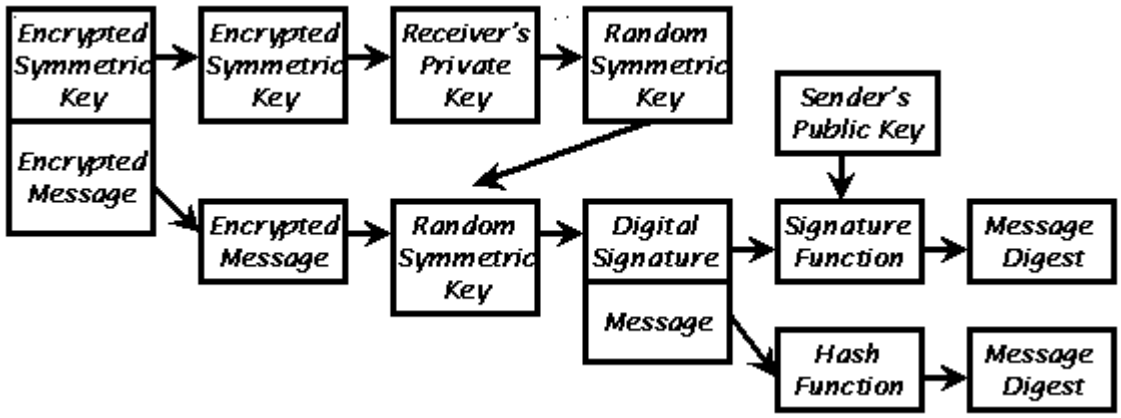

Figure 6: The process used to verify a digital envelope carrying digitally signed data.

One of the biggest advantages of using digital envelopes is that both the sender & receiver systems do not have to be online at the same time to be able to **negotiate** an online session key like in the SSL/TLS protocols. So, digital envelopes are very well suited for store-forward type of systems.

Another big advantage of digital envelopes is enhanced performance. The actual data transfer will occur only after the digital envelope has been successfully created. Therefore, cryptography does not have to be performed on the fly like in SSL. Performing cryptographic operations on the fly can be extremely CPU intensive and will necessitate the usage of high-end systems. An

application using digital envelopes can use existing data transfer protocols such as http, email or ftp to transport the data.

Some application frameworks actually use digital envelopes to securely transport a symmetric key to be used for future data encryption. After a specified amount of time, the systems would exchange a digital envelope again to replace the older symmetric key.

Digital envelopes do not come without disadvantages. It is obvious from the description that it is not exactly very well suited to multicast communication systems. Since the symmetric key is encrypted with the receiver's public key, this process has to be performed for each receiver in the multicast system. Again, this deficiency might not really cause problems because this can be performed offline before actually multicasting the data.

PKCS#7 Standards

The PKCS#7 standard defines a general syntax for messages that include cryptographic enhancements such as digital signatures and digital envelopes. This standard generates data which is BER encoded.

This standard defines six data types (1) Data, (2) Signed Data, (3) Enveloped Data, (4) Signed-and-Enveloped Data, (5) Digested Data and (6) Encrypted Data. If both the sender and receiver implement digital signatures/envelopes using the PKCS#7 standard, they would be able to exchange data unambiguously without the need for exchanging algorithm information by out-of-band methods.

Every PKCS#7 blob usually encapsulates some content (such as an encrypted message or signed message) and one or more certificates used to encrypt or sign this content. The receiver of this PKCS#7 blob can use the public key from the attached certificate to verify the digital signature.

Conclusion

In this article, we discussed how digital signatures and digital envelopes are created and verified. We also saw that PKCS#7 standards can be used to define digital signatures & digital envelopes in such a way that applications can interoperate unambiguously. We also saw how these concepts are used in the real world to secure our emails (S/MIME), credit card transactions (SET) as well as proprietary messaging frameworks.

In the next installment in this series, we will learn more about the internals of a public key infrastructure. This article will elaborate further on digital certificates, certificate authorities and the lifecycle of digital credentials. We also discuss how the concepts we have discussed so far seamlessly fit into a public key infrastructure.

About the Author
Mohan Atreya is a Technical Consultant with RSA Security. He has advanced degrees from National University of Singapore and Nanyang Technological University.

About RSA Security Inc.
RSA Security Inc., The Most Trusted Name in e-Security™, helps organizations build secure, trusted foundations for e-business through its RSA SecurID® two-factor authentication, RSA BSAFE® encryption and RSA Keon® digital certificate management systems. With more than a half billion RSA BSAFE-enabled applications in use worldwide, more than six million RSA SecurID users and almost 20 years of industry experience, RSA Security has the proven leadership and innovative technology to address the changing security needs of e-business and bring trust to the new, online economy. RSA Security can be reached at www.rsasecurity.com.

A Message to Developers:

The RSA BSAFE (**http://www.rsasecurity.com/products/bsafe/**) family of toolkits provides you with all the components you need to make your applications safe and secure. As a developer, you can save many months of development and testing, thus allowing you to focus on your application development and roll out your application with confidence. The BSAFE family comprises the following toolkits:

Core Functionality	BSAFE Toolkit Details
Core Cryptographic Toolkits	BSAFE Crypto-C & BSAFE Crypto-J
Public Key Infrastructure (PKI) Toolkits	BSAFE Cert-C & BSAFE Cert-J
Protocol Level Toolkits	BSAFE SSL-C & BSAFE SSL-J (SSL protocol for point-point security) BSAFE S/MIME-C (S/MIME Protocol for secure messaging) BSAFE WTLS-C (Wireless Transport Layer Security for WAP)