Introduction to the PKCS Standards

By Mohan Atreya (matreya@rsasecurity.com)

Summary

This is the second article in this series of articles, which aim to give the reader a bottoms up introduction to the basics of e-security. This article will introduce the reader to the Public Key Cryptography Standards (PKCS). The emphasis will be on what is standardized in the PKCS standards and the issues that are left open to the discretion of the developer.

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 Article-1 of this series before proceeding further with this tutorial. Alternatively, for a very interactive web seminar on e-security from RSA Security Inc., please follow the following link (http://www.rsasecurity.com/experience/esecurity/)

The wide acceptance of Public Key Cryptography requires applications developed by different vendors to interoperate seamlessly. Interoperability between applications is not always assured unless the vendors adhere to the standard formats strictly. The PKCS standards are specifications that were developed by RSA Security in conjunction with system developers worldwide (such as Microsoft, Apple, Sun etc.) for the purpose of accelerating the deployment of public key cryptography. The goal is to facilitate early adoption of these standards by vendors.

In the crypto world, PKCS is ubiquitous. These standards are used everywhere in the e-security realm. Any application developer choosing to implement security into his/her application would stumble upon these standards at some point of time. Applications ranging from web browsers to secure email clients depend on the PKCS standards to interoperate with one another. Consider the following example: a website administrator might need to migrate his server's digital credentials (server certificate & private key) to a web server supplied by a different organization. As long as the digital credentials are in a standardized format such as PKCS, the migration would be a fairly simple task.

PKCS is defined for both Binary and ASCII data types. PKCS only describes the syntax for messages in an abstract manner giving complete details about the algorithms. PKCS does not specify the representation format for the messages although BER (Basic Encoding Rules) encoded messages are a preferred format.

The following table lists the currently active PKCS standards.

Standard	Description
PKCS # 1	The RSA encryption standard. This standard defines mechanisms for encrypting and signing data using the RSA public key system.
PKCS # 3	The Diffie-Hellman key-agreement standard. This defines the Diffie-Hellman key agreement protocol.
PKCS # 5	The password-based encryption standard (PBE). This describes a method to generate a Secret Key based on a password.
PKCS # 6	The extended-certificate syntax standard. This is currently being phased out in favor of X509 v3.
PKCS # 7	The cryptographic message syntax standard. This defines a generic syntax for messages which have cryptography applied to it.
PKCS # 8	The private-key information syntax standard. This defines a method to store Private Key Information.
PKCS # 9	This defines selected attribute types for use in other PKCS standards.
PKCS # 10	The certification request syntax standard. This describes a syntax for certification requests.
PKCS # 11	The cryptographic token interface standard. This defines a technology independent programming interface for cryptographic devices such as smartcards.
PKCS # 12	The personal information exchange syntax standard. This describes a portable format for storage and transportation of user private keys, certificates etc.
PKCS # 13	The elliptic curve cryptography standard. This describes mechanisms to encrypt and sign data using elliptic curve cryptography.
PKCS # 14	This covers pseudo random number generation (PRNG). This is currently under active development.
PKCS # 15	The cryptographic token information format standard. This describes a standard for the format of cryptographic credentials stored on cryptographic tokens.

Note: PKCS #2 and #4 do not exist anymore because they have been incorporated into PKCS #1.

What has been standardized?

The two things that are standardized in PKCS are "**Message Syntax**" and "**Specific Algorithms**". These two can also be viewed as different levels of abstraction, which would be quite independent of each other.

Public key cryptography is typically used for the following purposes:

Digital Signatures: The "**signer**" signs a "**message**" such that anyone can "**verify**" that the message was signed only by the "**signer**" and thus not modified by anyone else. This can be implemented using a message digest algorithm and a public key algorithm to encrypt the message digest.

What is standardized?	
Specific message digest algorithms.	PKCS# 1
Specific public key algorithms.	PKCS# 1, 3, 13
Algorithm independent syntax for the digitally signed message.	PKCS# 7
Syntax for private keys.	PKCS# 1, 8
Syntax for encrypted private keys.	PKCS# 8
Method for deriving secret keys from passwords.	PKCS# 5

Digital Envelopes: The "**sender**" seals the "**message**" such that only the "**receiver**" can open the sealed message. The message is encrypted with a secret key and the secret key is encrypted using the receiver's public key.

What is standardized?	
Algorithm independent syntax for the digitally enveloped message.	PKCS# 7
Syntax for private keys.	PKCS# 1, 8
Syntax for encrypted private keys.	PKCS# 8
Method for deriving secret keys from passwords.	PKCS# 5

Digital Certificates: A "**Certification Authority**" signs a "**special message**" which contains the name of a user and the user's public key in such a way that "**anyone**" can verify that the "special message" was signed only by the "Certification Authority" and as a result trust the user's public key. This "special message" is termed as a certificate request and it is digitally signed using a "signature algorithm".

What is standardized?	
Algorithm independent syntax for certification requests.	PKCS# 10
Syntax for public keys.	PKCS# 1
Specific signature algorithms.	PKCS# 1

Key Agreement: Two "**communicating parties**" agree upon a secret key by exchanging messages without any prior agreements. Typically this consists of a two-phase key agreement algorithm. One party initiates the key agreement and this triggers the "**first phase**" of the key agreement after which both parties exchange the results of the first phase. After this, both parties initiate the "**second phase**" of the key agreement and as a result both parties arrive at the same secret key.

What is standardized?	
Algorithm independent syntax for key agreement messages.	PKCS# 3
Specific key agreement algorithms.	PKCS# 3

Open Issues

The PKCS developers have left some room open so that the users of PKCS would enjoy a fair amount of flexibility. Some of the issues currently left open are listed below:

S. No.	Open Issue
1.	Range of lengths of the RSA modulus in PKCS #1.
2.	Range of lengths of the Diffie-Hellman modulus in PKCS #3.
3.	Range of the length of the password used in PKCS #5.
4.	Structural requirements of the password used in PKCS #5 (i.e. The developer has to force the application to use a strong un-guessable password).
5.	Sources of pseudo-random bits used in all algorithm standards.
6.	Maximum length of certificate chain, etc.

Conclusion

In this article, we saw why the PKCS standards were developed. We understood why certain aspects had to be standardized to let applications using cryptography interoperate seamlessly. Certain aspects are left to the choice of the developer. It should be noted that inappropriate choices for the open issues could compromise the application's security.

In the next installment of this series, we will discuss about the inner workings of **Password Based Encryption (PBE)**, which builds on PKCS #5 and PKCS #12. This is a mechanism, which uses a password to generate a Secret Key, which in turn can be used to encrypt and decrypt your private keys and files.

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.

References

1. A treatise on PKCS Standards from RSA Security, Inc.
   http://www.rsasecurity.com/rsalabs/pkcs/
2. The RSA Laboratories FAQ about today's cryptography.
   http://www.rsasecurity.com/rsalabs/faq/index.html/

A Message to Developers
The RSA 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)