White Paper

Public Key Encryption and Digital Signature: How do they work?

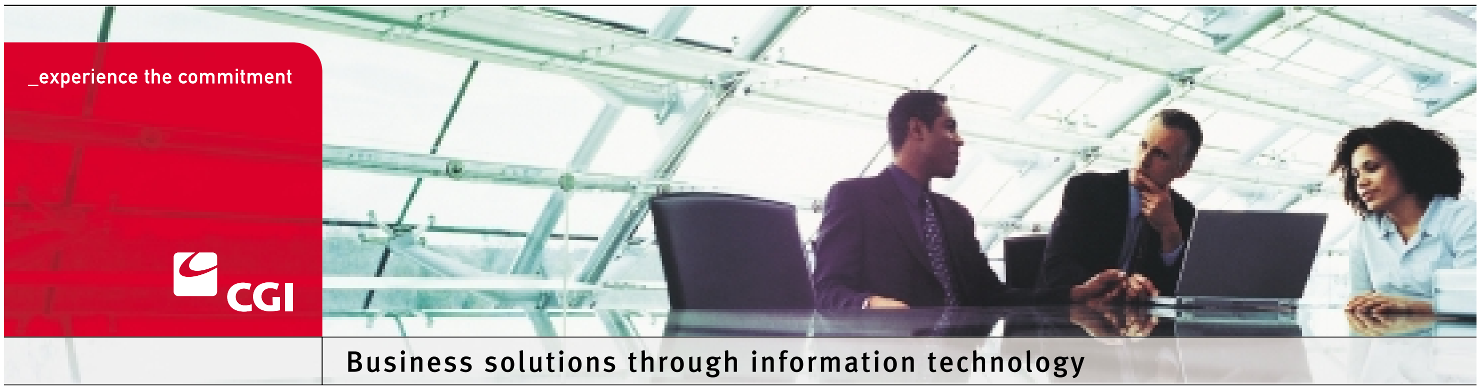

_experience the commitment
CGI
Business solutions through information technology

TABLE OF CONTENTS

Introduction

One of the major challenges facing consultants today is maintaining a level of knowledge of leading and emerging technologies, beyond the superficial or buzzword level. We need to develop a level of understanding that allows us to communicate effectively with both suppliers and customers. We then can demonstrate:

- Our knowledge of the business issues being addressed;
- The application of technology to providing solutions;
- The business benefits for the customer;
- The limitations that will remain and that will need to be mitigated in other ways.

Public-key has been around for some time now. A lot of interesting work has been done in different committees (for instance IETF/PKIX and PKCS[1]) to define Public-key related standards and techniques. But, do we know what is in there? Do we know how it works? Let's open the hood and take a look at the engine and at last, get into the details on how public-key encryption and digital signature really work.

This article is a starting point to get a feel for a vast area named **PKI** (Public-Key Infrastructure). This includes the mechanics described in this article as well as an ensemble of software, hardware and processes governed by rules and standards converging to the high level of Trust required and expected by the Industry.

1. Public-key, what it is

Public-key refers to a cryptographic mechanism. It has been named public-key to differentiate it from the traditional and more intuitive cryptographic mechanism known as: symmetric-key, shared secret, secret-key and also called private-key.

Symmetric-key cryptography is a mechanism by which the same key is used for both encrypting and decrypting; it is more intuitive because of its similarity with what you expect to use for locking and unlocking a door: the same key. This characteristic requires sophisticated mechanisms to securely distribute the secret-key to both parties[2].

Public-key on the other hand, introduces another concept involving key pairs: one for encrypting, the other for decrypting. This concept, as you will see below, is very clever and attractive, and provides a great deal of advantages over symmetric-key:

- Simplified key distribution
- Digital Signature
- Long-term encryption

However, it is important to note that symmetric-key still plays a major role in the implementation of a Public-key Infrastructure or *PKI*.

1.1 A definition

Public-key is commonly used to identify a cryptographic method that uses an *asymmetric-key* pair[3]: a *public-key* and a *private-key* [4]. Public-key encryption uses that *key pair* for encryption and decryption. The public-key is made public and is distributed widely and freely. The private-key is never distributed and must be kept secret.

Given a key pair, data encrypted with the public-key can only be decrypted with its private-key; conversely, data encrypted with the private-key can only be decrypted with its public-key. This characteristic is used to implement encryption and digital signature. Both encryption and digital signature principles are illustrated in Figure 1 and Figure 2.

1.2 Encryption and Decryption

Encryption is a mechanism by which a message is transformed so that only the sender and recipient can see. For instance, suppose that Alice wants to send a private message to Bob. To do so, she first needs Bob's public-key; since everybody can see his public-key, Bob can send it over the network in the clear without any concerns. Once Alice has Bob's public-key, she encrypts the message using Bob's public-key and sends it to Bob. Bob receives Alice's message and, using his private-key, decrypts it.

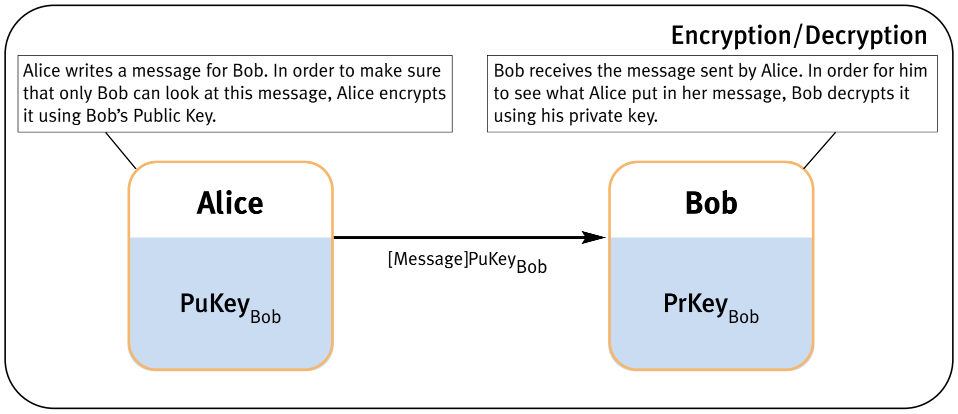

Encryption/Decryption
Alice writes a message for Bob. In order to make sure that only Bob can look at this message, Alice encrypts it using Bob's Public Key.
Bob receives the message sent by Alice. In order for him to see what Alice put in her message, Bob decrypts it using his private key.
Alice
PuKey Bob
[Message]PuKey Bob
Bob
PrKey Bob

Figure 1: Encryption/Decryption principles

1.3 Digital Signature and Verification

Digital signature is a mechanism by which a message is authenticated i.e. proving that a message is effectively coming from a given sender, much like a signature on a paper document. For instance, suppose that Alice wants to digitally sign a message to Bob. To do so, she uses her private-key to encrypt the message; she then sends the message along with her public-key (typically, the public key is attached to the signed message). Since Alice's public-key is the only key that can decrypt that message, a successful decryption constitutes a Digital Signature Verification, meaning that there is no doubt that it is Alice's private key that encrypted the message.

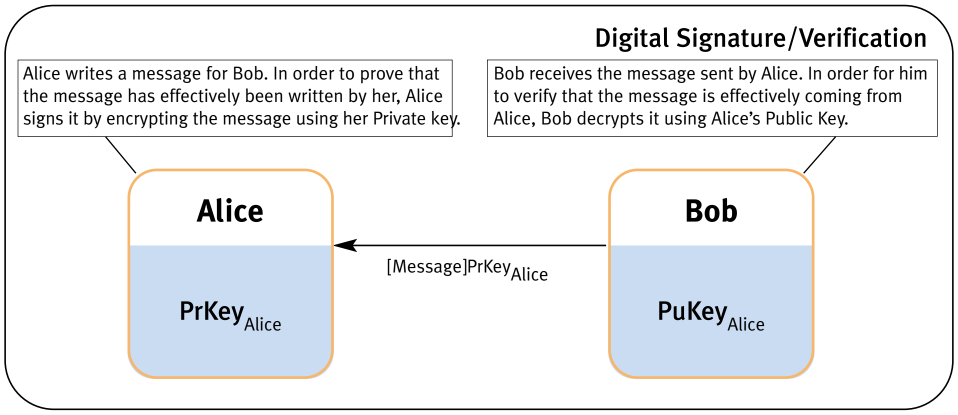

Digital Signature/Verification
Alice writes a message for Bob. In order to prove that the message has effectively been written by her, Alice signs it by encrypting the message using her Private key.
Bob receives the message sent by Alice. In order for him to verify that the message is effectively coming from Alice, Bob decrypts it using Alice's Public Key.
Alice
PrKey Alice
[Message]PrKey Alice
Bob
PuKey Alice

Figure 2: Digital Signature/Verification principles

1.4 Beyond the principles

The two previous paragraphs illustrate the encryption/decryption and signature/verification principles. Both encryption and digital signature can be combined, hence providing privacy and authentication.

As mentioned earlier, symmetric-key plays a major role in public-key encryption implementations. This is because asymmetric-key encryption algorithms[5] are somewhat slower than symmetric-key algorithms[6].

For Digital signature, another technique used is called hashing. Hashing produces a message digest that is a small and unique[7] representation (a bit like a sophisticated checksum) of the complete message. Hashing algorithms[8] are a one-way encryption, i.e. it is impossible to derive the message from the digest. The main reasons for producing a message digest are:

1 The message integrity being sent is preserved; any message alteration will immediately be detected;

2 The digital signature will be applied to the digest, which is usually considerably smaller than the message itself;

3 Hashing algorithms are much faster than any encryption algorithm (asymmetric or symmetric).

The following sections explains what really happens when encrypting and signing a message on one hand, and when decrypting a message and verifying its signature on the other hand.

1.4.1 Steps for signing and encrypting a message

Figure 3 below shows the set of operations required when Alice wants to send a signed and encrypted message to Bob.

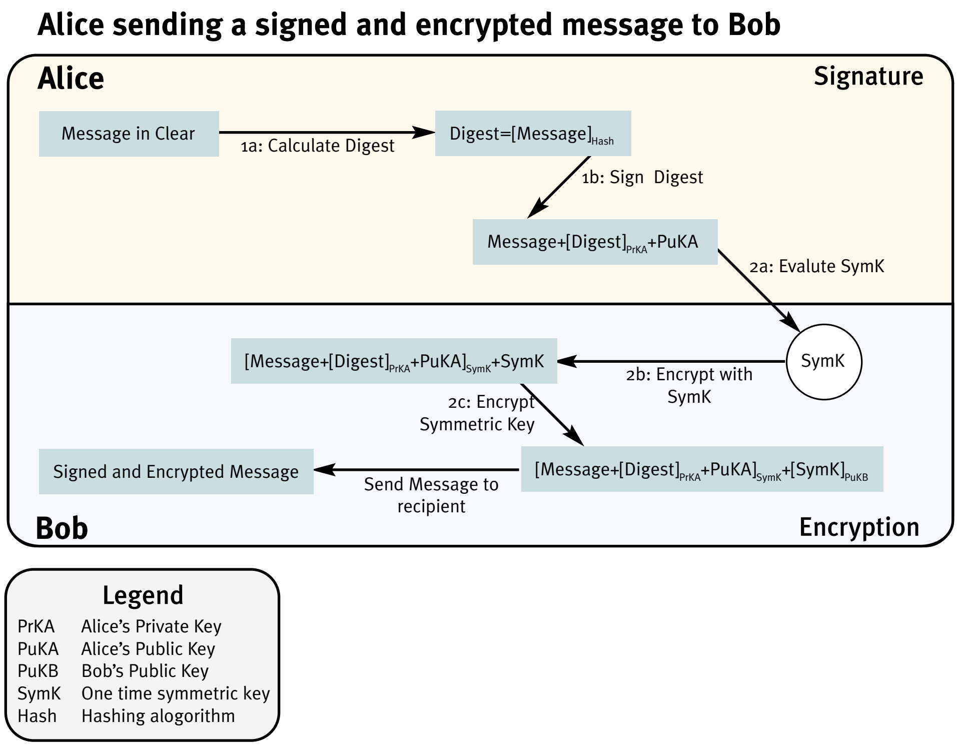

Alice sending a signed and encrypted message to Bob
Alice
Signature
Message in Clear
1a: Calculate Digest
Digest=[Message]$_{Hash}$
1b: Sign Digest
Message+[Digest]$_{PrKA}$+PuKA
2a: Evalute SymK
SymK
2b: Encrypt with SymK
[Message+[Digest]$_{PrKA}$+PuKA]$_{SymK}$+SymK
2c: Encrypt Symmetric Key
[Message+[Digest]$_{PrKA}$+PuKA]$_{SymK}$+[SymK]$_{PuKB}$
Send Message to recipient
Signed and Encrypted Message
Bob
Encryption
Legend
PrKA Alice's Private Key
PuKA Alice's Public Key
PuKB Bob's Public Key
SymK One time symmetric key
Hash Hashing alogorithm

Figure 3: Signature and Encryption details with keys

1) **Message signature.** Digital signature includes two steps:

a) *Message digest evaluation*. The main purpose for evaluating a digest is to ensure that the message is kept unaltered; this is called message integrity.

b) *Digest signature*. A signature is in fact an encryption using the issuer's (Alice in this case) private-key. Included in the signature is also the hashing algorithm name used by the issuer. The issuer's public-key is also appended to the signature. Doing so lets anyone decrypt and verify the signature using the issuer's public-key and hashing algorithm. Given the properties of public-key encryption and hashing algorithms, the recipient has proof that:
i) The issuer's private-key has encrypted the digest;
ii) The message is protected against any alteration.

2) **Message encryption.** Encryption includes the following 3 steps:

a) *Creation of a one time symmetric encryption/decryption key.* Remember that encryption and decryption algorithms using asymmetric-keys are too slow to be used for long messages; symmetric-key algorithms are very efficient and are therefore used.

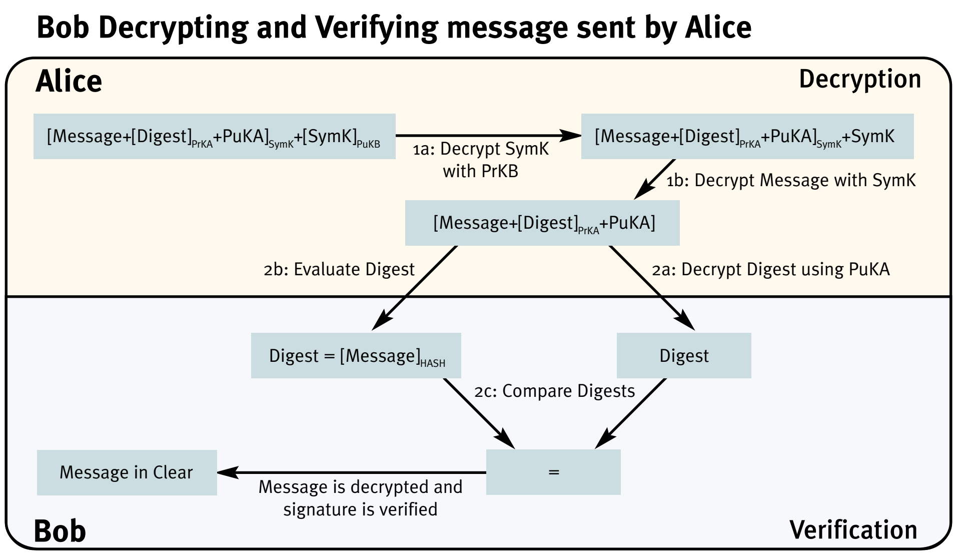
b) *Message encryption*. The whole message (the message itself and the signature) is encrypted using SymK, the symmetric-key evaluated above.

c) *Symmetric-key encryption*. SymK is also used by the recipient to decrypt the message. SymK must therefore be available to the recipient (Bob) only. The way to hide the Symk from everybody except the recipient is to encrypt it using the recipient's public-key. Since SymK is a small piece of information compared to a message (that could be very long), the performance penalty associated with the relative inefficiency of asymmetric-key algorithms is acceptable.

One interesting point to mention is that if Alice wants to send the same message to more than one recipient, say Bob and John for instance, the only additional operation to be performed is to repeat 'step 2) c)' for John. Hence, the message that both Bob and John would receive would look like:
[Message+[Digest]PrKA+PuKA]SymK+[SymK]PuKB+[SymK]PuKJ . Notice that the exact same SymK will be used by Bob and John to decrypt the message.

1.4.2 Steps for Decrypting and verifying the signature of a message

Figure 4 below shows the set of operations required when Bob wants to decrypt and verify the message sent by Alice.


Bob Decrypting and Verifying message sent by Alice
Alice
Decryption
$[Message+[Digest]_{PrKA}+PuKA]_{SymK}+[SymK]_{PuKB}$
1a: Decrypt SymK with PrKB
$[Message+[Digest]_{PrKA}+PuKA]_{SymK}+SymK$
1b: Decrypt Message with SymK
$[Message+[Digest]_{PrKA}+PuKA]$
2b: Evaluate Digest
2a: Decrypt Digest using PuKA
$Digest = [Message]_{HASH}$
Digest
2c: Compare Digests
=
Message in Clear
Message is decrypted and signature is verified
Bob
Verification

Legend

PrKA	Alice's Private Key
PuKA	Alice's Public Key
PuKB	Bob's Public Key
SymK	One time symmetric key
Hash	Hashing alogorithm

Figure 4: Decryption and verification details with keys

1) **Message decryption.** The decryption includes the following steps:

a) *Symmetric-key decryption*. The one time symmetric-key has been used to encrypt the message. This key (SymK) has been encrypted using the recipient's (Bob) public-key. Only Bob can decrypt SymK and use it to decrypt the message[9].

b) *Message decryption*. The message (which includes the message itself and the signature) is decrypted using SymK.

2) **Signature verification.** The signature verification includes the following 3 steps:

a) *Message digest decryption*. The digest has been encrypted using the issuer's (Alice) private-key. The digest is now decrypted using the issuer's public-key included in the message.

b) *Digest evaluation*. Since hashing is a one-way process i.e. the message cannot be derived from the digest itself, the recipient must re-evaluate the digest using the exact same hashing algorithm the issuer used.

c) *Digests comparison*. The digest decrypted in a) and the digest evaluated in b) are compared. If there is a match, the signature has been verified, and the recipient can accept the message as coming unaltered from the issuer. If there is a mismatch this could mean that:
i) The message has not been signed by the issuer or
ii) The message has been altered.
iii) In both cases, the message should be rejected.

1.5 Identity and keys

Until now, we have taken for granted the keys being used for encryption/decryption and digital signature/verification belong to Bob and Alice. How can we be sure that Alice is really Alice? And, how can Alice be sure that only Bob will see what she encrypted? So far, the only thing we know is that the user of a given key pair has signed and encrypted the message. But, is he really the owner? George, for instance, may have sent a message to Bob pretending that he is Alice; Bob cannot tell whether or not it is Alice or George who is sending the message. The same applies to Bob's public-key. This issue is solved by the use of certificates.

2. What is a Certificate

A certificate is a piece of information that proves the identity of a public-key's owner. Like a passport, a certificate provides recognized proof of a person's (or entity) identity. Certificates are signed and delivered securely by a trusted third party entity called a Certificate Authority (CA). As long as Bob and Alice trust this third party, the CA, they can be assured that the keys belong to the persons they claim to be.

A certificate contains among other things:

1) The CA's identity
2) The owner's identity
3) The owner's public-key
4) The certificate expiry date
5) The CA's signature of that certificate
6) Other information that is beyond the scope of this article.

With a certificate instead of a public-key, a recipient can now verify a few things about the issuer to make sure that the certificate is valid and belongs to the person claiming its ownership:

1) Compare the owner's identity
2) Verify that the certificate is still valid
3) Verify that the certificate has been signed by a trusted CA
4) Verify the issuer's certificate signature, hence making sure it has not been altered.

Bob can now verify Alice's certificate and be assured that it is Alice's private-key that has been used to sign the message. Alice must be careful with her private-key and must not divulge how to get to it; by doing so, she is enforcing one aspect of the non-repudiation feature associated with her digital signature. As will be seen in section 3.2, there is more to consider for effective non-repudiation support.

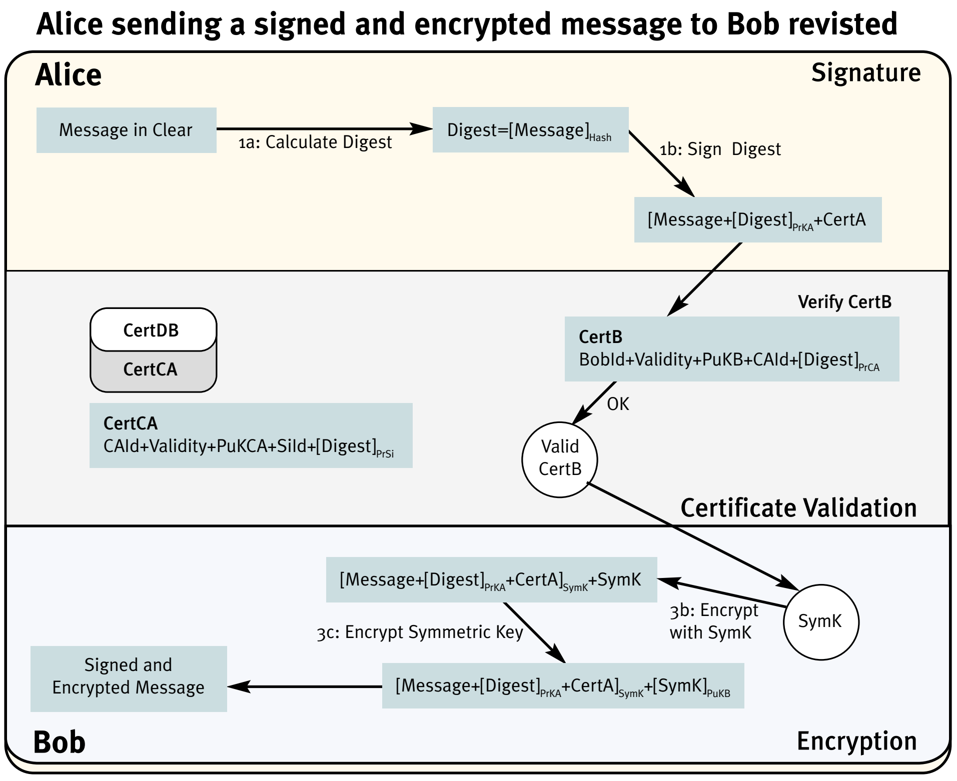
Note that certificates are signed by a CA, which means that they cannot be altered. In turn, the CA signature can be verified using that CA's certificate.

2.1 Certificate validation added to the process

When Alice encrypts a message for Bob, she uses Bob's certificate. Prior to using the public-key included in Bob's certificate, some additional steps are performed to validate Bob's certificate:

1) Validity period of Bob's certificate
2) The certificate belongs to Bob
3) Bob's certificate has not been altered
4) Bob's certificate has been signed by a trusted CA

Additional steps would be required to validate the CA's certificate in the case where Alice does not trust Bob's CA. These steps are identical to the ones requires to validate Bob's certificate. In the example below, it is assumed that both Bob and Alice trust that CA.


Alice sending a signed and encrypted message to Bob revisted
Alice
Signature
Message in Clear
1a: Calculate Digest
Digest=[Message]$_{Hash}$
1b: Sign Digest
[Message+[Digest]$_{PrKA}$+CertA
Verify CertB
CertDB
CertCA
CertB
BobId+Validity+PuKB+CAId+[Digest]$_{PrCA}$
OK
CertCA
CAId+Validity+PuKCA+SiId+[Digest]$_{PrSi}$
Valid CertB
Certificate Validation
[Message+[Digest]$_{PrKA}$+CertA]$_{SymK}$+SymK
3b: Encrypt with SymK
SymK
3c: Encrypt Symmetric Key
Signed and Encrypted Message
[Message+[Digest]$_{PrKA}$+CertA]$_{SymK}$+[SymK]$_{PuKB}$
Bob
Encryption

Legend

PrKA	Alice's Private Key	CertCA	CA's Certificate
PrCA	CA's Private Key	SymK	One time symmetric key
PrSi	CertCA Signer's Private Key	Hash	Hashing alogorithm
PuKB	Bob's Public Key	BobId	Bob's Identity
PuKCA	CA's Public Key	CAId	CA's Identity
CertA	Alice's Certificate	SiId	CertCA Signer's Id
CertB	Bob's Certificate	Validity	Certificate's Valid Period

Figure 5: Signature and Encryption details with certificates

In the Figure 5 above, a Certificate validation step is added to what is shown in Figure 3. Only the fields required for the validation of a certificate are displayed.

Alice wants to make sure that the PuKB included in CertB belongs to Bob and is still valid.

- She checks the Id field and finds BobId, which is Bob's identity. In fact, the only thing she really knows is that this certificate appears to belong to Bob.
- She then checks the validity fields and finds that the current date and time is within the validity period. So far the certificate seems to belong to Bob and to be valid.
- The ultimate verification takes place by verifying CertB's signature using the CA's public-key (PuKCA found in CertCA)[10]. If CertB signature is ok, this means that:
  a) Bob's certificate has been signed by the CA in which Alice and Bob has put all their trust.
  b) Bob's certificate integrity is proven and has not been altered in any way.
  c) Bob's identity is assured and the public-key included in the certificate is still valid and belongs to Bob. Therefore, Alice can encrypt the message and be assured that only Bob will be able to read it.

Similar steps will be performed by Bob on Alice's certificate before verifying Alice's signature.

2.2 Beyond the mechanics

So far, this article covered in some details the public-key mechanics associated with encryption and digital signature. In section 2.1 the notion of Certificate Authority has been brought up. The CA is the heart of a Public-Key Infrastructure (PKI).

3. What is a PKI

A PKI is a combination of software and procedures providing a means for managing keys and certificates, and using them efficiently. Just recall the complexity of the operations described earlier in this article for having a feel on the absolute necessity to provide users with appropriate software support for encryption and digital signature. But nothing has been said yet about management.

3.1 Key and certificate management

Key and certificate management is the set of operations required to create and maintain keys and certificates. The following is the list of the major points being addressed in a managed PKI:

1) **Key and certificate creation:** How to generate key pairs? How to issue certificates to the users? A PKI must offer software support for key pair generation as well as certificate requests. In addition, procedures must be put in place to verify the user identity prior to allowing him to request a certificate.

2) **Private-key protection:** How will the user protect his private-key against misuse by other malicious users?
Certificates are widely accessible because they are used for either encryption or signature verification. Private-keys require some reasonable level of protection because they are used either for decryption or for digital signature. A strong password mechanism must be part of the features of an effective PKI.

3) **Certificate revocation:** How to handle the situation where a user's private-key has been compromised? Similarly, how to handle the situation where an employee leaves the company? How to know whether or not a certificate has been revoked?
A PKI must provide a means by which a certificate can be revoked. Once revoked, this certificate must be included in a revocation list that is available to all users. A mechanism must be provided to verify that revocation list and refuse to use a revoked certificate.

4) **Key backup and recovery:** What happens to encrypted files when a user loses his private-key? Without key backup, all messages and files that have been encrypted with his public-key can no longer be decrypted and are lost forever. A PKI must offer private-key backup and a private-key recovery mechanism such that the user can get back his private-key to be able to get access to his files[11].

5) **Key and certificate update:** What happens when a certificate reaches or is near its expiry date? Keys and certificates have a finite lifetime. A PKI must offer a mechanism to at least update the expiry date for that certificate. Good practice though is to update the user's keys and certificates. The key and certificate update can be automatic in which case the end user gets notified that his keys have been updated, or can require that the user performs an action during or before his keys and certificates expire; if this case, the PKI must inform the user that this action is required prior the expiry time of his keys and certificates.

6) **Key history management:** After several key updates, how will a user decide which private-key to use to decrypt files?
Each key update operation generates new key pairs. Files that have been encrypted with previous public-keys can only be decrypted with their associated private-keys. Without key history management, the user would have to make decision on the key to use for decrypting files[12].

7) Certificate access: How will a user, who wants to send a message to several recipients, get their certificates?
   A PKI must offer an easy and convenient way to make these certificates available. The use of an LDAP directory is commonly used for that purpose.

3.2 Support for non repudiation of digital signature

One important point that has to be clarified is non-repudiation of digital signature. This notion refers to the fact that a user cannot deny having signed a given message. This implies that the user who signed the message is the only one who has access to the private-key used for signing. However, as we have seen above, in a managed PKI, private-keys are kept by the CA for key recovery purposes. Therefore, both the user and the CA know that private-key, which means that both can (in theory) use that key for signing a message. A user can then deny having signed that message.

In order to avoid this situation and provide non-repudiation support, there must be a second key pair used for signature/verification purposes only. No backup is made for the signing private-keys, and only the user has access to it. In the case where the user loses his password, he loses his signing key as well. At key recovery time, the encryption/decryption key pair is given back to the user and a new signature/verification key pair is generated. This causes no problem since each time a user signs a document, the associated verification certificate is appended to it, so the document signature can always be verified at any time.

4. Conclusion

Public-key is a very appealing and exciting technology, embedding both encryption and digital signature. This constitutes a breakthrough over symmetric-key cryptosystems.

Beyond the mechanics, there is a need for an infrastructure, a PKI, which includes the tools to effectively manage and use keys and certificates. Section 3 gives a feel for how a well managed PKI addresses main issues related to key/certificate usage and management. This should be covered in more details in another article.

Finally, beyond the infrastructure, there is a need for preparing, documenting and maintaining the infrastructure. From a technical point of view, implementing a PKI is a matter of days. From a business point of view, implementing a PKI is really another story, and yet another article.

About CGI

Founded 1976, CGI has worked with clients in a wide range of industries to help them leverage the strengths of information technology (IT) to optimize their business performance and
produce value-driven results. We also offer a comprehensive array of
business process outsourcing (BPO) services, enabling us to help manage and improve our clients' day-to-day business processes while freeing them up to focus more on strategic decision making. Our consulting, systems
integration and outsourcing services provide a total solution package designed to meet our clients' complete business and technology needs. We approach every engagement with one objective in mind—to help our client win and grow. CGI provides services to clients worldwide from offices in Canada, the United States, Europe, as well as centers of excellence in India and Canada.

To explore this topic and how we can help, contact your CGI account manager or visit http://www.cgi.com/web/en/head_office.htm for the location of the CGI office nearest you. Other information about CGI can be found at www.cgi.com.

References

Curry, Ian, Entrust Technologies, “Getting Acquainted With Entrust/Solo and Public-key Cryptography”, version 1.0, July 1997

Netscape, “Introduction to Public-Key Cryptography”, http://developer.netscape.com/docs/manuals/security/pkin/contents.htm

Curry, Ian, Entrust Technologies, “Version 3 X.509 Certificates”, July 1996, version 1.0

Branchaud, Marc, “A Survey of Public-key Infrastructures”, Department of Computer Science, McGill University, Montreal, 1997

Curry, Ian, Entrust Technologies, “Key Update and the Complete story on the Need for Two Key Pairs”, version 1.2, August 2000

RSA, “Intro to PKCS Standards”, http://www.rsasecurity.com/solutions/developers/whitepapers/IntroToPKCSstandards.pdf

IETF/PKIX web site, http://www.ietf.org/html.charters/pkix-charter.html

Footnotes:

1 IETF/PKIX stands for Internet Engineering Task Force/Public-Key Infrastructure based on X509 certificates. PKCS stands for Public Key Cryptography Standards; these standards have been developed by RSA in conjunction with vendors like MicroSoft, Apple, Sun, etc.)

2 Infrastructures like Kerberos provide symmetric-key distribution and management.

3 Asymmetric-key vs symmetric-key is yet another differentiation between the two mechanisms.

4 Be careful not to confuse between the private-key mechanism and a private-key. In order to avoid any confusion, Symmetric-key will be used along this article when talking about the mechanism.

5 Some asymmetric-key algorithms: RSA, DSA and ECDSA.

6 Some symmetric-key algorithms: RC2, RC4, DES and triple-DES.

7 In fact, the message digest is most probably unique in the sense that it is almost impossible to find 2 meaningful messages and, at the same time, producing the same digest. Hence the probability that a tampered message produces the same digest as the original is almost zero.

8 Some hashing algorithms: SHA-1 and MD5.

9 In fact, all recipients would be able to decrypt their own copy of the SymK. All operations that follow can therefore be performed by every recipient.

10 As a reminder, a signature is the digest of a message (in this case, the message is CertB) encrypted using the issuer’s private-key (in this case the CA’s private-key). The CA’s signature verification process is identical to what is illustrated in Figure 4 where Bob verified Alice’s signature.

11 Private-key backup is also used for key escrow i.e. a way for the company or a legal third party to get access to the users files which may contain invaluable information that would otherwise be lost for ever.

12 For instance, suppose that at her last key update, Alice received PuKAn/PrKAn pair, and that Alice kept a message that has been encrypted with PuKA1, her first public-key. Then, she needs to use PrKA1 to decrypt the message.