

## **Cryptography in a very small nutshell**

### **Introduction**

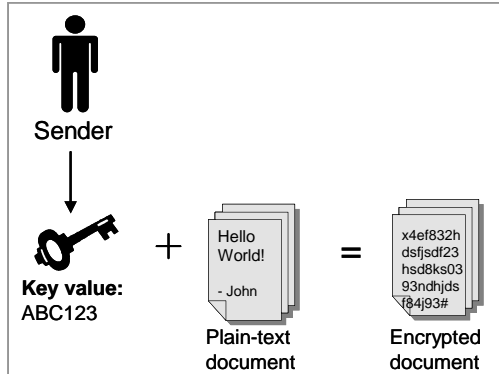
Cryptography is perhaps the single most important “enabler” of Information Security and Privacy; it is also perhaps the least understood and misused. The main source of confusion likely stems from the many encryption methods and options which are available. Another contributing factor may be that the teams selecting and implementing solutions which require encryption feel that they need to fully understand the deep inner-workings of the cryptosystem. In general, most cryptographic systems can be treated as a “black box” within project scenarios.

Similarly, an exhaustive discussion of the myriad encryption methods, protocols, and options is outside the scope of this paper. Instead, this paper will outline the three main types of cryptography: *symmetric*, *asymmetric*, and *one-way hashing*.

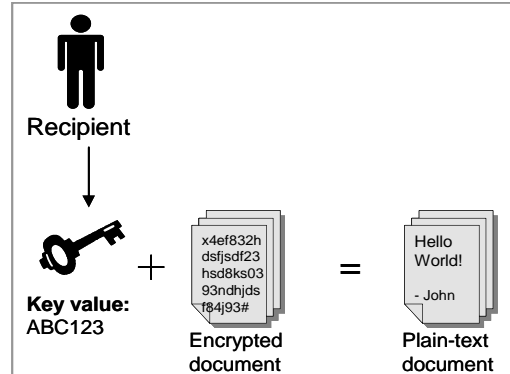
### ***Symmetric Cryptography (“Secret-key”)***

Symmetric cryptography, also referred to as “secret-key” cryptography, is a form of cryptography in which only one encryption key is used; in other words, the sender and receiver of encrypted data use two instances of the same key for both encryption and decryption. While this method does indeed encrypt the data, it is not as secure as asymmetric encryption (covered in the next section) due to the fact that the same key must be shared between two or more parties; therefore, true verification of authenticity, and the goal of non-repudiation, cannot be achieved. Further, there are issues with secure distribution of the “secret-key.” In general, if one can find a secure manner in which to communicate the “secret-key,” then encryption is not necessary – the parties could use that very channel for communication.

**Figure 1** demonstrates the process utilized to send an encrypted message using symmetric (or “secret-key”) encryption. Note that key “ABC123” is required by both the sender and the recipient (see **Figure 2**). Thus the same key is used in both the encryption process and the decryption process.



**Figure 1:** Encrypting and sending a message using Symmetric cryptography



**Figure 2:** Receiving and decrypting a message using Symmetric cryptography

### **Asymmetric Cryptography (“Public-key”)**

Asymmetric cryptography (a.k.a. “Public-key” cryptography) utilizes a key-pair, generally termed “public” and “private.” These two keys are mathematically related such that if data is encrypted using a “private” key, it can only be decrypted using the “public” key – and vice versa. With asymmetric cryptography, it is impossible to encrypt and decrypt using the same key. The true poser of this model is in that it overcomes the weaknesses of symmetric cryptography with regard to authenticity and non-repudiation.<sup>1</sup> The data sender retains complete control of his/her “private” key, while having the ability to publish the “public” key to any trusted group of users. Asymmetric encryption is extremely powerful when it comes to such areas as digital signatures. This model can also be used to provide secure key transmittal for large amounts of data encrypted using the secret-key method.<sup>2</sup>

### **Hashing Algorithms (One-way hashes)**

Hashing algorithms are not encryption, per se. Instead they are algorithms which may be used to process data such that a unique “fingerprint” or “digest” is created. The algorithms are reliable and will always produce the same result for any given data-set. This is extremely powerful for password storage because the original data cannot be derived by attempting to reverse the hash<sup>3</sup>; instead, the

<sup>1</sup> Assuming that the “private” key is not lost or compromised.

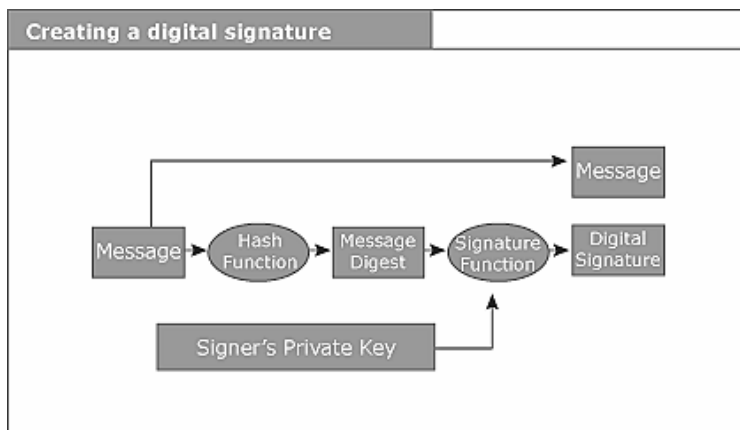
<sup>2</sup> In this scenario, the actual data is encrypted using symmetric/secret-key encryption; but a “session-key” is encrypted, by the sender, using the recipient’s public key and is included in the transmission and can be decrypted using the recipients private key.

<sup>3</sup> Instead, an ‘attacker’ must employ either “brute force” – trying all possible combinations until successfully “guessing” the correct password, or he/she can turn to a “rainbow table” – which is a pre-computed list of hashed values. This is essentially an indexed/searchable pre-computed brute force attack.

system can tell if a user is entering a string which matches the stored password by running the same hashing algorithm against the string that the user entered and then comparing the results. If the results are the same, then the string that the user entered is the same string used to create the stored hash.

Hashing algorithms are also used for document or application “fingerprinting.” This is beneficial for determining if a computer file has changed. The function is similar to the password hash comparisons mentioned above; but, generally, this type of “file fingerprinting” utilizes an MD-5 hash. Applications run the MD-5 algorithm against the file in question to obtain a “fingerprint.” Once that fingerprint is created, it can be posted to a location which maintains an index of document fingerprints; then, if someone needs to check to see if the document in question has been modified, they generate a new MD-5 hash of their document and compare it against the official/indexed version. In content management systems, this is a critical element to establishing the integrity of cataloged files.

Hashing algorithms can be combined with asymmetric cryptography in order to create methods for message authentication. See figures 3 and 4.<sup>4</sup>



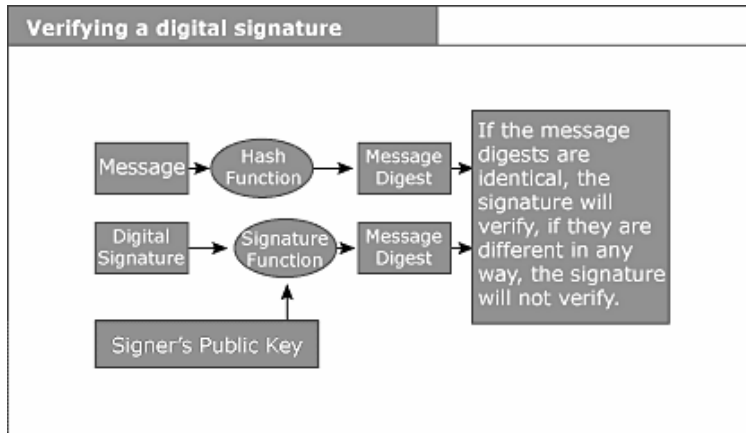
**Figure 3:** Using hash functions and asymmetric cryptography to create a digital signature.

Source: MSIA Seminar 1, week 9 lecture materials. Stephen Cobb

---

So, if the attacker can get the SHA-1 password, he/she can simply do a “SELECT” within the rainbow table and retrieve the clear-text value (assuming that it is in the table).

<sup>4</sup> The diagrams for figures 6 and 7 are taken from Stephen Cobb’s MSIA Seminar 1, week 9 lecture materials.



**Figure 4:** Using hash functions and asymmetric cryptography to validate a digital signature.

Source: MSIA Seminar 1, week 9 lecture materials. Stephen Cob

### Public-Key Infrastructures (PKI)

Message authentication, described above, is one of the main uses/examples of PKI (Public-Key Infrastructure). A Public-Key Infrastructure is needed to generate the key pairs used for secure communications. PKI, however, is not limited to digital signatures and message authentication. The potential uses for Public-Key Infrastructures are many and varied: including smart card Identity Management through certificates; secure B2B communications through mutual SSL paths; secure HTTPS secured web-pages for eCommerce or data collection; and so on. Therefore, companies should view the use or implementation of a Public-Key Infrastructure very strategically – weighing several factors.

One such factor is *build* (in-house development and deployment) versus *buy* (outsourced management/co-management). Building the solution in-house is generally a 6 – 12 month process. In addition, companies must train (or otherwise acquire) the in-house expertise required to install, configure, and maintain the system. Companies opting for the *buy* option generally see implementation timelines of 4 – 6 weeks.

One case study available on the Internet details the *University of Wisconsin – Madison's* PKI initiative; this study outlines many of the specifics related to buying versus building. This case is indicative of what most companies discover; that is, a co-managed / outsourced PKI implementation is extremely cost-effective, and the benefits of the system can be realized in a fraction of the time required to build an in-house solution. The case study data is presented in the cost and timeline comparison chart below:

PKI Implementation Cost Comparison (UW – Madison)			
Build “in-house”		Outsource / co-manage	
Implementation timeframe – 9 months –		Implementation timeframe – 4 weeks –	
Year 1 Costs		Year 1 Costs	
Implementation for 5,000 users	\$200,000	Contracted support for 5,000 users	\$43,000
1 FTE (salary and benefits)	\$100,000	1 FTE (salary and benefits)	\$100,000
<b>Year 1 Total</b>	<b>\$300,000</b>	<b>Year 1 Total</b>	<b>\$143,000</b>
Year 2 and beyond		Year 2 and beyond	
Annual License Maintenance	\$ 40,000	Annual contract	\$43,000
1 FTE (salary and benefits)	\$100,000	1 FTE (salary and benefits)	\$100,000
System upgrades/maintenance	\$ 5,000		
<b>Annual Total</b>	<b>\$145,000</b>	<b>Annual Total</b>	<b>\$143,000</b>
<b>Total cost @ 5 years</b>	<b>\$880,000</b>	<b>Total cost @ 5 years</b>	<b>\$715,000</b>
<b>Total cost @ 10 years</b>	<b>\$1,605,000</b>	<b>Total cost @ 10 years</b>	<b>\$1,430,000</b>

**Figure 5:** Implementation cost and timeline comparisons for University of Madison – Wisconsin’s PKI initiative. Details are available at the following URL. <http://www.educause.edu/ir/library/powerpoint/LIVE0611.pps>.

**Cryptography and PKI: Recommendations**

- *Employ full-disk encryption on all laptops and on “high-risk” desktops.*
- *When providing secure web-pages for customers, ensure that certificates are valid, trustworthy (e.g. not self-signed), and kept up-to-date.*
- *Evaluate the potential need for cryptographic solutions within various corporate business units which have a need to access and email sensitive employee or customer information.*

Departments such as Human Resources, Legal, the Enterprise Security Office, Finance, and Benefits regularly access and are required to email or otherwise “trade in” potentially sensitive information. For these groups, cryptographic solutions such as digital signatures, encrypted file storage, encrypted file transfer, and secure messaging.

- *When considering PKI solutions, evaluate the options with regarding co-managed / outsourced services.*

Co-managed services may allow the company a swift implementation timeline while simultaneously reducing the project cost. In addition, key management may be more secure.<sup>5</sup>

<sup>5</sup> Key management could be more secure via choosing a co-managed service. The complexities of an in-house solution, lack of adequate training, and/or configuration issues could lead to less-secure key management.