

**Download it now!****PDF** (55 KB)[Free Acrobat™ Reader](#)

Introduction to cryptography, Part 3: Asymmetric cryptography

Murdoch Mactaggart

Freelance Writer

March 2001

The security afforded by asymmetric cryptosystems depends on mathematical problems that are difficult to solve, such as factoring large integers into primes. Public key systems use two keys such that one key, the public key, can be used to encrypt some text that can then only be decrypted using the securely-held private key. Alternatively, the private key can be used to encrypt some information that anyone with access to the widely-available corresponding public key can decrypt, so satisfying themselves that the message was generated by the holder of the private key.

Introduction

The need to share a secret key, which is required for both encryption and decryption, can be a major vulnerability where symmetric key systems of cryptography are used. In asymmetric, or public key, cryptography this is not an issue in the same way. Two keys, mathematically related, are used and work together in such a way that plain text encrypted with the one key can only be decrypted with the other. One of these keys will typically be kept private by one individual, so there's almost no need to share keys, thus avoiding the risk of compromising security. The second key, the so-called public key, needs to be made as widely-known as possible.

The encryption/decryption process can work in both directions. That is, I can encrypt something using your public key and so ensure that only you can read it using your private key. I can also encrypt something using my private key and although this is a pointless exercise from the point of view of information hiding -- as anyone with access to my public key can decrypt the message -- it is the basis of a system of authentication, confirming that the message could only have originated from someone with access to my private key, presumably me.

Earlier, I mentioned the seminal paper, *New Directions in Cryptography*, published in 1976 by Whitfield Diffie and Martin Hellman (then at Stanford University). However, as a report from the UK's cryptography authority shows (J. H. Ellis, *The Possibility of Secure Non-Secret Digital Encryption*, CESG Report, January 1970), a very similar mechanism may have been proposed and examined earlier but kept secret by the appropriate UK authorities. Whatever the actual origins (and such developments invariably build on work carried out earlier) the introduction of the asymmetric cryptosystem concept, and its subsequent refinement in a variety of specific systems, is an extremely important development.

e-mail it!

Contents:[Introduction](#)[Examples of asymmetric ciphers](#)[Other asymmetric cryptosystems](#)[Hash functions](#)[MD4 and MD5](#)[SHA and SHA-1](#)[Conclusion](#)[About the author](#)

Symmetric and asymmetric systems have their own strengths and weaknesses. In particular, asymmetric systems are vulnerable in different ways, such as through impersonation, and are much slower in execution than are symmetric systems. However, they have particular benefits and, importantly, can work together with symmetric systems to create cryptographic mechanisms that are elegant and efficient and can give an extremely high level of security. The complementary nature of these two general methods is discussed more fully in a later article; this article concentrates more on the technical and other aspects relevant to public key systems.

Although hash functions are not the same as public-key cryptosystems, it's convenient to consider them here as well, since they're often used for message digests and, hence, as part of the overall systems that incorporate authentication and digital signing.

Examples of asymmetric ciphers

Diffie-Hellman

The Diffie-Hellman protocol is fully described in the paper mentioned earlier. It allows two users to share a secret key over some insecure exchange mechanism without first agreeing on some secret values. It has two system parameters, each public, one of which, p , is a prime number and the other, usually known as a generator, is an integer that is less than p ; this generator is capable of generating every element from 1 to $p-1$ when multiplied by itself a certain number of times, modulo p .

In practical terms, the following procedure might take place. Firstly, each person generates a random private value, say a and b . Each then derives public values using the public parameters p and g and their particular private values, a or b , with the general formula $g^n \bmod p$ where n is a or b as appropriate. They then exchange these public values. Lastly, one person computes $k_{ab} = (g^b)^a \bmod p$ and the other computes $k_{ba} = (g^a)^b \bmod p$. $k_{ab} = k_{ba} = k$, the shared secret key.

This key exchange protocol is vulnerable to impersonation, the so-called middle-person attack. If A and B are seeking to exchange keys then a third person, C, might intervene between each exchange. A thinks the initial public value is being sent to B, but, in fact, it's intercepted by C who then transmits a different public value to B, repeating the process for the transfer which B thinks is being made direct to A. This results in A and C agreeing on one shared secret key and B and C agreeing on another. C can then sit in the middle, intercept and decrypt the messages from A to B using the A/C key, modify and forward them to B, using the B/C key, and repeat the process on the return with neither A nor B being aware of what's happening.

In order to counter this, the authenticated Diffie-Hellman key agreement protocol was developed by Diffie and others in 1992. In this protocol it's necessary to use existing private/public key pairs and associated digital certificates for the public key elements that validate the initial public values exchanged.

RSA

In 1977, a year after the publication of the Diffie-Hellman paper, three researchers at MIT developed a practical method using the suggested ideas. This became known as RSA, after the initials of the three developers -- Ron Rivest, Adi Shamir, and Leonard Adelman -- and is probably the most widely-used public key cryptosystem. It was patented in the US in 1983, duly adopted as a standard, and has always been widely available outside the US in implementations developed locally even though, until recently, its export was restricted.

As with other such systems, RSA uses large prime numbers to construct the key pairs. Each pair shares

the product of two primes, the modulus, but each also has a specific exponent. RSA Laboratories explains the working of the RSA cryptosystem as follows:

“Take two large primes, p and q , and compute their product $n = pq$; n is the modulus. Choose a number, e , less than n and relatively prime to $(p - 1)(q - 1)$, which means that e and $(p - 1)(q - 1)$ have no common factors except 1. Find another number d such that $(ed - 1)$ is divisible by $(p - 1)(q - 1)$. The values e and d are called the public and private exponents, respectively. The public key is the pair (n, e) ; the private key is (n, d) .”

Knowledge of the public key offers a route to obtaining the private key but this depends on factoring the modulus into its component primes. This is difficult and can be made essentially impossible by choosing keys of adequate length. The measurement needed is the length of the modulus; RSA Laboratories currently recommends key sizes of 1024 bits for general corporate use and double that, 2048 bits, for extremely valuable material. For ordinary use, key lengths of 768 bits are adequate as these cannot readily be broken using current techniques. As always, the cost of securing material needs to be considered in conjunction with the material's value and whether the costs of breaking the protection might be excessive. RSA Laboratories mentions a recent study of RSA key-size security based on factoring techniques available in 1995. This study suggested that a 512-bit key might be factored for less than \$1 million in eight months of effort. In fact, a particular RSA 512-bit number, known as RSA-155, was factored in seven months during 1999 as part of the regular RSA Security challenge.

It's also important to remember that the various figures given as to the extent of security offered are averages and that it might be possible on occasion to identify a particular private key much more quickly. Also, the security offered assumes that factoring prime numbers is a difficult problem. If this were to change with the discovery of new mathematical techniques that make factoring easy, then the security given by the RSA and similar algorithms could immediately become worthless.

Note, also, that there is a trade-off in the speed of encryption/decryption when key sizes are increased. Doubling the modulus will approximately quadruple the time taken for operations using the public key, and increase by a factor of eight the time required to carry out private key operations. Further, key generation would increase on average by a factor of 16 when the modulus is doubled. Given the steady and rapid increase in computing power and the fact that asymmetric cryptography is generally used with short texts, this may not matter very much in practice.

Other asymmetric cryptosystems

The ElGamal system, named for its developer, depends on the discrete logarithm problem and has encryption and signature variants, while the Digital Signature Algorithm (DSA) is based in part on ElGamal. The system appears to be as secure as RSA but is generally slower and has the effect of expanding the message by a factor of two during encryption. These limitations are less relevant for signature uses.

Other systems include the Merkle-Hellman knapsack cryptosystem, first published in 1978, and the Chor-Rivest knapsack cryptosystem, first published in 1984 and in a revised version in 1988. There's also the LUC public-key system developed in Australia and New Zealand. The McEliece public key encryption algorithm is based on algebraic coding theory and uses a class of error-correcting codes known as Goppa codes. These codes provide fast decoding, but use key sizes of around half a megabit and expand the message text considerably.

A further group of public-key cryptosystems, first proposed in the mid-1980s and currently arousing some interest, is what are known as elliptic curve cryptosystems. These are based on mathematical constructions from number theory and algebraic geometry, and are generally defined over finite fields. These appear to have the potential for offering similar security to existing systems while using smaller key sizes, something that might be particularly useful in mobile computing or in smartcard-based systems. RSA Laboratories suggest that elliptic curve cryptosystems with 160-bit key sizes might offer very roughly the same security as RSA with a 1024-bit key. However, there is some concern that elliptic curve cryptosystems might be vulnerable to specialized attacks that have not yet been fully developed.

Hash functions

Hashing involves transforming an arbitrary string of data into a fixed-length result. The original length can vary greatly but the result will always be of the same length, typically 128 or 160 bits in cryptographic use. Hashing is used widely for populating indexes that are used for rapid exact-match searching; the concept is no different in cryptographic use, although the techniques vary greatly. When hashing is used to construct index entries, a balance needs to be struck between the density of entries anticipated in a working system and the likelihood of collisions -- that is, of different entries returning the same hash value. Unless the index is made very large and sparsely filled, there will certainly be collisions, but in indexing these can easily be handled by, say, chaining to empty values and then examining the original entries that share a hash value before returning a result. However, this option is not realistic when hashing is used in cryptosystems and the relevant algorithms need to eliminate collisions as far as possible. The number of, say, 128-bit hash values is finite at 3.4×10^{38} . However, as the number of possible messages is infinite, collisions are certainly possible (and will, in fact, be infinite in number). Moreover, the chances of two different messages resulting in the same hash value are vanishingly small in any well-constructed cryptographic hash algorithm, and for all practical purposes it can be assumed that collisions will not occur.

A hash function works in one direction only and is useless for the purposes of retrieving plain text. What it does do very well, however, is provide a digital identifier that is specific to one message only and will change if the plain text message is changed in any way, even to the extent of adding or removing a space. Subject to the caveat given in the preceding paragraph, this means that a suitable hash function can be used to confirm that a given message has not been changed. The value is what's known as a message digest. Message digests, being small and virtually unique for a given message, are typically used as elements in digital signatures and in digital time stamping, each of which is discussed in a later feature.

If a collision can be generated, then it may be possible to forge a digest and send a fraudulent message. One approach to doing this is to use what's known as the "birthday attack," a type of brute force attack named for the surprising effect that the probability of two or more people in a group of 23 sharing the same birthday is greater than $1 / 2$.

Someone who wants to forge a message first creates both an instance of a fraudulent message and of an innocent message which whoever is being targeted might reasonably sign. He then uses the appropriate hashing algorithm with an arbitrary secret key to generate $2^{n/2}$ variations of the innocent message together with an equal number of variations on the fraudulent message, n being the bit-length of the message digest. Given that the slightest change will result in a different message digest, it's possible, at least theoretically, to create messages that differ only in minor detail. According to the birthday paradox, the probability that one of the variations of the innocent message will match the hash value of

one of the variations of the fraudulent message is greater than $1/2$. The forger has the unsuspecting target sign the selected innocent message and duly transfers to the version of the fraudulent message, which earlier shared the same digest value the new digest created by the signer for the innocent message. With this approach, it's not necessary to know the secret key used by the target in generating the message digest.

MD4 and MD5

MD2, MD4 and MD5 are message-digest algorithms developed by Ron Rivest for use in digital signature applications where a message is to be compressed to a digest and then encrypted by a private key. MD2 was designed for 8-bit computer systems while MD4 and MD5 were developed for 32-bit computer systems. MD4, developed in 1990, is now considered insecure.

MD5 is described by RSA Laboratories as "MD4 with safety belts," and although slower than MD4, it is considered secure. As with MD4, a plain text message is padded to ensure that its length in bits plus 448 is divisible by 512. A 64-bit binary representation of the original message length is then added and the message is processed in 512-bit blocks using an iterative compression function, each block being processed in four distinct rounds.

SHA and SHA-1

The Secure Hash Algorithm (SHA) was developed by the National Institute of Standards and Technology, a division of the US Department of Commerce, which issues standards for cryptographic routines. However, the algorithm turned out to be less secure than its name implied and a revision of the original algorithm was duly published in 1994 and named SHA-1.

In contrast to MD5, SHA-1 generates a message digest of 160 bits and is considered more secure, although slower in execution. It takes a maximum plain text message of up to 2^{64} bits in length.

Conclusion

Public key cryptography is an important part of techniques used particularly for authentication. Although a public key system could be used simply to encrypt plain text, its practical value probably lies more in its use in conjunction with other elements, including message digests and symmetric cryptosystems, to support authentication and the secure transmission of compound packets of information.

Public key systems such as RSA use large key-lengths -- a 768-bit key is now the minimum recommended. Factoring into primes is the principle method of attacking many such systems and this is very difficult to do. As computer systems increase in power, keys that earlier were secure may become vulnerable to brute force attack, so it's important to ensure that keys of adequate length are used to provide protection against future developments. Although such increases in computer power make it easier to attack legacy keys, there's actually a greater benefit gained by more easily being able to use in reasonable time longer keys that should be impossible to break in the foreseeable future.

About the author

Murdoch Mactaggart is a freelance writer and business consultant who writes on software development, the Internet and on business and management issues around these areas. Whether readers can make accurate sense of what he writes is a moot point but, flexible though he tries to be, he generally sticks to English rather than introducing languages of his own making. Contact him at IBMDev@TextBiz.com.

What do you think of this article?

Killer! (5)

Good stuff (4)

So-so; not bad (3)

Needs work (2)

Lame! (1)

Comments?

[Privacy](#) [Legal](#) [Contact](#)