An Introduction to Public-Key Cryptosystems

by
James V. Boone

Preface

Public-key cryptosystems are in wide use today and many varied projections exist regarding their future popularity. Many individuals find that the logic behind the fundamental source of security in public-key cryptosystems systems is unclear. This short paper is intended to give the interested reader a "feel" for how such systems work and provide some references for further, more detailed, study . The treatment in this paper is neither rigorous or complete, however the terminology used is a subset of that in use by professionals in the field.

It is hoped that the explanation provided here, along with an associated numerical example, will both augment the readers intuition and stimulate their curiosity. It should at least provide the nomenclature necessary for further study.

An Introduction to "Public-Key" Cryptosystems

First, it is important to realize that there is nothing about public-key cryptosystems which changes the fact that **the basic security of any cryptographic scheme depends primarily on the length of the key and the computational difficulty of breaking the cipher.** From the aspect of the security of the cipher stream alone, there is no basic inherent advantage of either conventional (shared secret key) or "public-key" systems. The difficulties of distributing keys in a conventional system, and probably the issues surrounding the control of such keys, prompted early work[1] in what has come to be called "public-key" cryptography. Fundamentally, these systems (which are over twenty years old in concept) **use one key for encryption and a different (yet related) key for decryption.** It is worth noting at the outset that what conventional cryptologists would call **"secrecy" is still a requirement within a "public key" system.** In concept, the required secrecy is easier to manage, enforce, and monitor in a public-key system.

Stallings [2] points out that public-key algorithms have one characteristic in common, that it is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key. He also points out that some algorithms (including the popular RSA* algorithm) have the interesting system characteristic that either of the two related keys can be used for encryption with the other used for decryption. Since this characteristic has some attractive system features, this is the type used in the following material.

like what ?

---

[1] See for example, Diffie, W., and Hellman, M. "Multi-user Cryptographic Techniques." Proceeding of the AFIPS National Computer Conference, June 1976, and "New Directions in Cryptography." IEEE Transactions on Information Theory , November 1976 (same authors)

[2] Stallings, William, "Network and Internetwork Security." Prentice-Hall, Inc. and IEEE Press, 1995

* RSA = open Rivest Shamir & Adleman

Figure One represents a simplistic model of a public-key system. In this model, "A" wishes to send "B" a message. To do so, "A" must know "B's" **public key** and another **public piece of information,** called the **modulus.** The encryption algorithm uses this combination and creates the cipher text which is delivered to the decryption algorithm where "B's" **private key** and the **modulus** is used to again produce plain text. The reverse process is required for two-way communication. The major distinction between this type of system and a conventional (shared secret key) system is that **nothing "secret" or "private" must be shared between the two corespondents before the communication takes place.** "How can this work?", is the next question to be examined by using the RSA technique (developed in 1977 by Rivest, Shamir, and Adleman at MIT and published in 1978[3]).

It is worth noting that the RSA technique is a block cipher which represents plain text as integers between 0 and (n-1) for some n and the string of input bits is of fixed length.

Since some of the operations depend on the use of prime numbers and some reference is made to modular arithmetic, a very quick review of some fundamentals may be in order here.

First, some definitions. An integer, p, greater than one, is a **prime** if its **only** divisors are 1 (either plus or minus) or p (again either plus or minus). A fact which makes some of the following process actually work is that **any** positive integer, a, (greater than one) can be factored in a unique way as:

$$a = p_1^{\alpha 1} \times p_2^{\alpha 2} \times \ldots p_t^{\alpha t}$$

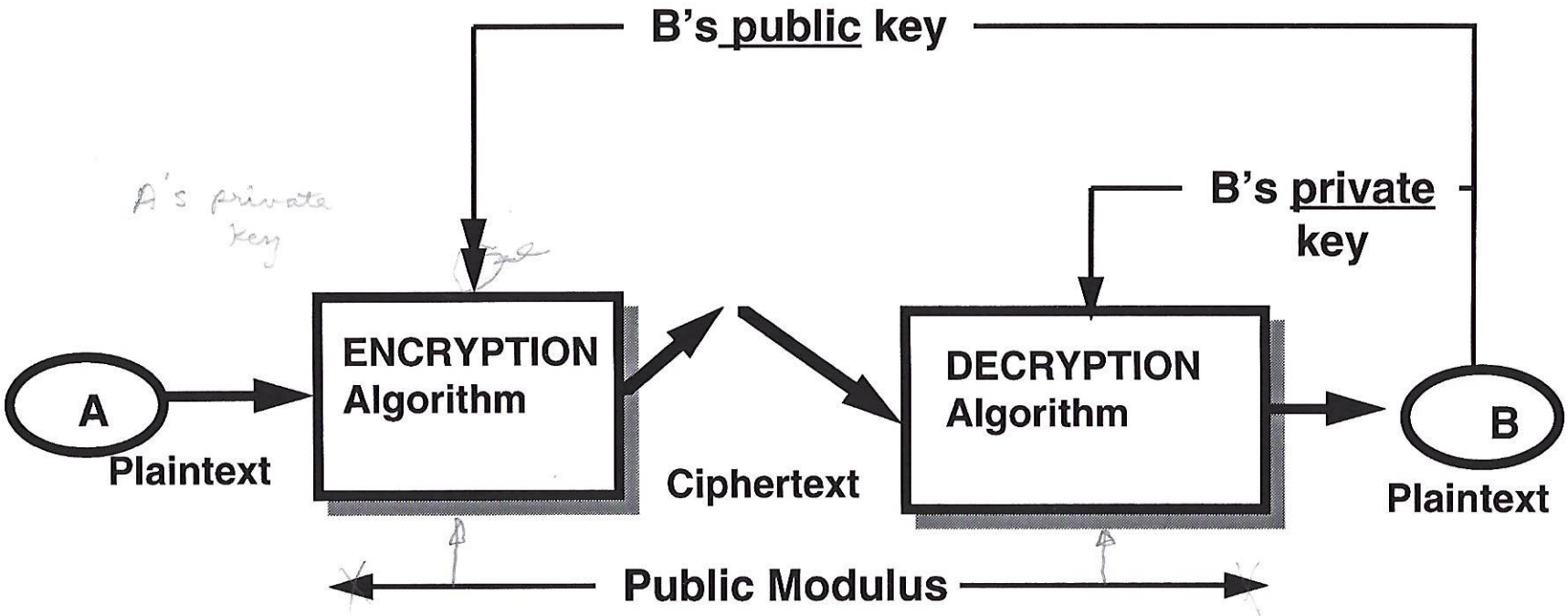
each p in this expression is a prime number, each larger than the previous one, and each exponent is greater than zero. Another useful thing to remember is that two integers, say a and b, are said to be **relatively prime** if they have **no** prime factors **in common.** In other words, their only common factor is 1.

---

[3] Rivest, R. ; Shamir, A.; and Adleman, L. "A Method for Obtaining Digital Signatures and Public Key Cryptosystems." Communications of the ACM, February 1978.

Diagram not clear

Figure 1. Model of Public-Key System



Note that each end of the system must have a pair of keys, one public, the other "private", if two-way communications are desired

Now, just a bit about modular arithmetic. It is important because it is used to help set up the exponential nature of the RSA scheme in a manner which provides the security feature. Take any positive integer, n , and any integer, a . We can divide a by n and get a **remainer** (or residue) which we will call r , and a quotent which we call q. It is always possible to satisify the relationship,

$$a = qn + r$$

Using these same definitions, we can simply define a **mod** n as the **remainder** when a is divided by n. Believe it or not, this turns out to be useful because modular arithmetic is based on this definition.

Without any detailed proof at this point, the process involved in the RSA technique is outlined as follows (rather than keep creating new representations, we will re-use some letters, i.e., the "q " below and the "r " below are **not** the same as we used above).

Rivest, Shamir , and Adleman noted that it was possible, by using the characteristics of modular arithmetic and the properties of prime numbers to create expressions as follows:

$$P = C^{K_{priv}} \bmod (r) \; = \; P^{K_{priv}K_{pub}} \bmod (r)$$

Now we are going to establish the nomenclature by,

calling the **public key,** K_{pub} and, the **private key,** K_{priv} .

Define a plain text block as, **P**, and the corresponding cipher text block as, **C**. The inventors noted that if they could implement this set of expressions, then , because of the great difficulty in factoring products prime numbers, they would have a process like the one we have illustrated in Figure One (of course they could implement the expressions or we would not be discussing this topic).

For illustrative purposes, the sequence of the process is something like the following:

- two "secret" prime numbers p and q are selected "randomly"

- the **public** modulus, $r = pq$, is calculated

- the "secret" Euler "totient" function is calculated,

$$\varphi(r) = (p-1)(q-1)$$

- **K** is selected, which is relatively prime to $\varphi(r)$, **K** is defined as **either** the private key **or** the public key

- the multiplicative inverse of [**K** mod $\varphi(r)$] is calculated using Euclid's algorithm, and this is defined to be **either** the private key [K_{priv}] or the public key[K_{pub}] depending on which choice was made in the previous step. If all you know about modular arithmetic does not spring instantly to mind , do not be concerned at this point. Also, if you cannot remember even what Euclid's algorithm is about, even though it is 2300 years old, do not worry. Note, however, that if **K** is prime relative to $\varphi(r)$, then [**K** mod$\varphi(r)$] will **also** be prime relative to $\varphi(r)$. **This is necessary** to keep both Euclid and Euler happy. It is also necessiary because it is essential that this is true in order for equations (1), (2) and (3) below to be correct. Fermat also contributes his first theorem to the process, but again, as long as we use the rules of modular arithmetic and the features of prime numbers, everyone is happy.

There is a wide variety of nomenclature in use in the literature. In an attempt at clarity (and in some disregard for mathematical convention) we will use our own descriptive nomenclature. Remember we called the **public key,** K_{pub} and, the **private key,** K_{priv} .

The relationship between the two has been described above and is

$$K_{pub} = K_{priv}^{-1} \bmod \varphi(r) \qquad (1)$$

this is what we called the "multiplicitative inverse of [**K** mod $\varphi(r)$] ".

The RSA technique now involves raising numbers to powers (or, if you like, uses expressions with exponents).

Consider our plain text block, **P**, and remember that it has some bound on its length, or value. It has a relationship to the cipher text block, **C**, which in the RSA system is:

for encryption

$$C = P^{K_{pub}} \bmod (r) \ , \qquad (2)$$

and, for decryption

$$P = C^{K_{priv}} \bmod (r) \ . \qquad (3)$$

Look back at Figure 1 and follow the process of A sending B a message.

- First, everyone knows the public modulus , **r.**

- Next, B has published the public key it is using, and A wants to send B the plain text, **P.**

- A calculates the cipher text using expression (2) above for **C** and B's **public key.**

- A transmits the cipher text to B (perhaps worrying how it is transmitted, perhaps not!)

- Upon receipt, B decrypts the message by using expression (3) above.

It is useful to review what we have here in a summary way. We have two relatively simple equations,

$$C = P^{K_{pub}} \bmod (r) \quad \text{and} \quad P = C^{K_{priv}} \bmod (r)$$

and by arithmetic processes we can also state;

$P = C^{K_{priv}} \bmod (r) = P^{K_{priv}K_{pub}} \bmod (r)$ which is where Rivest, Shamir and Adleman started. We have used some attributes of prime numbers and modular arithmetic to get to where we can make those statements. It is "relatively" easy to calculate the values required as long as P is less than r. Most find that easy to believe. And most important, it is "**infeasible**" to determine K_{priv} given K_{pub}. We have not shown how difficult or "infeasible" it is, but most would believe it if a lot of serious mathematicians said so. They do. There is a relationship between these public and private keys, but it is difficult to compute because it is currently difficult to factor r, the public modulus, into its two prime factors, p and q. The same can be said for the other factoring possibilities. An example adds crediability to this brief explanation for many people therefore, an example of the process using "small" numbers is included as an appendix to this note.

This is still an intuitively difficult situation for most of us. Why cannot someone other than B receive and decrypt the message since A did not even know B's private key? You only use your private key to decrypt. That sure helps keep it private! Our intuition tells us that A must have something that everyone else does not have! What does A know that all other people do not know? Answer, the plain text! A does then have a "secret" on that end of the process. It is the secret which all listeners would like to know, but it is nevertheless a secret. This is not very mathematical reasoning, but it makes most of us feel better. However, this does bring up the use of RSA in authentication processes, but that works too, although the process is a little different.

What if you want to read that traffic anyway? What options do you have? Well, you could:

1. Try all possible private keys. For our simple numerical problem, this is not too hard, but for larger keys and longer blocks, it becomes impractical (all other things ignored) . As in shared secret-key systems the larger the key, the more secure the system is (and probably the slower it will run with a given technological implementation).

2. Another direct approach would be to focus on being able to factor large numbers into primes. There would be several places to use this feature in the system; calculation of individual private keys, based

on public key information is obviously one. There is much energy being spent in this area! The definition of a "breakthrough" depends on what one believed in the first place. 129 digits were thought to be "totally safe" a decade ago, and now estimates range up to 250 or so! **As usual, the discriminator is "what is the underlying information worth, and to whom?"**

3. Finding other fundamental interrelationships which are weaknesses is always a good analytic approach. Several weaknesses are known in public. Meyer and Matyas[4] note that the inventors suggest that p and q should differ in length by only a few bits which will make it even more difficult for some factoring algorithms and there are some other suggestions. They also note "However, one cannot conclude that the problem of factoring in the cryptographic problem is hard merely on the basis that the classical problem of factoring is known to be hard." They suggested, in 1982, that NSA* "certify the algorithm's strength" (NSA didn't, but one can imagine several rational reasons for it to stay uninvolved).

Others (see for example Neuwirth[5])have treated problems like recovery from key loss and other operational factors. Neuwirth pointed out the fact that known weaknesses include rules for changing the public modulus without changing all parts of the system and also the hazards of using RSA in a large network where all members share a common public modulus (I probably do not do these arguments justice with this summary) . There are other known limitations.

It is interesting to note that the company creating RSA-based products is, itself, sponsoring public efforts to improve the state of the art in factoring (and some have produced good results). There are several good reasons to do this, but this view of life is almost as revolutionary as the public-key concept itself.

In summary, remember that RSA is only one of several public-key systems in use today. Each depends on the known computational difficulty of some problem. RSA uses prime numbers in its development, others (such as SEEK) use random numbers. All share

---

[4]Meyer, Carl H.; Matyas, Stephen M. "Cryptography", 1982, John Wiley and Sons, Inc. ISBN 0-471-04892-5. p45-48

[5]Neuwirth, Lee. "A Comparison of Four Key Distribution Methods", 1986, "Telecommunications"

* NSA National

one or more elements in "public", yet keep some aspect "private" (some of us may still wish to say "secret" , but that word is seldom used for various reasons).

The systems work. They are easy to use. They are cheap to use. They are secure. How secure and for how long? As we said before, it depends on the judgment of the user regards the value of the underlying information to the intruder and the technological sophistication of the intruder. A classic risk assessment problem!

James V. Boone
May 1997

A "Simple" Numerical Example To Illustrate The Process

On B's end of the path

1. Select two primes, $p=7$ *and* $q=17$ (we will see later that there are actually some "rules" for doing this, but we don't care in this simple example). **These must be kept private.**

2. Calculate the public modulus, $r= 7 \times 17 = \boldsymbol{119}$

3. Calculate $\varphi(r)= (p - 1)\ (q- 1) =6 \times 16 = \boldsymbol{96}$

4. Select $\boldsymbol{K_{pub}}$ such that it is relatively prime to $\varphi(r)= 96$ and less than 96. In this case , use **5** (5 and 96 have no common factors which is fairly easy to check, but if we were using larger numbers we would need some tools...they exist).

5. Calculate $\boldsymbol{K_{priv}}$. So now we are looking for a number which will work in an equation like

$$\boldsymbol{K_{priv}5 = 1 \bmod 96}$$

Here we use our old friend Euler again and he tells us that it is OK to use our previously known relationships and create our private key by using the expression

$$\boldsymbol{K_{priv}K_{pub} = k\,\varphi(r) + 1}\text{, where } \boldsymbol{k} \text{ is an integer.}$$

So now in our example we are looking for a number which will work in an equation like

$$\boldsymbol{K_{priv}5 = k\,96 + 1},$$

We have some constraints, but if $k = 4$, then $\boldsymbol{K_{priv}} = 77$. Check it out,

$$77 \times 5 = 4\ (96) +1 = 385.$$

Now let us see if we can use our keys to encrypt and decrypt a message. Remember that we have a **public key (5), a private key (77)** and a **public modulus (119)**. We are going to have to use all three in the following.

On A's end of the path

First, we need a "plain text", let's make it **19.**

To encrypt, raise 19 to the fifth power and get 2476099.
Next, divide by 119 and get 20807 (which you are not very interested in) and a remainder of **66** (which you want very much). Relating that to our equations (2) and (3),

$$19^5 = 66 \bmod 119$$

and **C**, the cipher text, is **66**.

In our example, user A sends **66**.

On B's end of the path again

User B decrypts using the expression,

$$66^{77} = P \bmod 119 .$$

This example , based on one by Stallings[1] is very informative because even in this simple example we already see that we have to deal with very large numbers. 66 to the 77th power is something like 1.27 times ten to the 140th power. Divide this whopper by 119 and we get another big number plus the thing we are looking for , **P = 19** as the remainder (our plain text!). **Obviously one needs to have techniques for working efficiently with exponentiation and large numbers.** Such techniques exist.

Some people also wonder about how many prime numbers there are to select from since there are some combinations which should be avoided. In theory, there is an infinite number of them, but knowing that is not very helpful. One can get a "feel" for what "how many"

---

[1] previously cited, p 121-125.

may mean by looking at the distribution of prime numbers. There are ways to calculate this, and the following table may provide some scale to the problem.

Interval	**Number of prime numers in the interval**
2-1000	168
2-10,000	1,229
2-1,000,000	78,489
2-100,000,000	5,761,455
2-1,000,000,000	50,847,478

It is pretty obvious that if we are going to select from a large population of prime numbers, that the number of digits involved will be large. This contributes to the complexity of the situation in a practical way.

Can you make any statements regarding the length of encrypted message compared to the plain text message?