The RSA Algorithm

Evgeny Milanov

3 June 2009

In 1978, Ron **R**ivest, Adi **S**hamir, and Leonard **A**dleman introduced a cryptographic algorithm, which was essentially to replace the less secure National Bureau of Standards (NBS) algorithm. Most importantly, RSA implements a public-key cryptosystem, as well as digital signatures. RSA is motivated by the published works of Diffie and Hellman from several years before, who described the idea of such an algorithm, but never truly developed it.

Introduced at the time when the era of electronic email was expected to soon arise, RSA implemented two important ideas:

1. Public-key encryption. This idea omits the need for a “courier” to deliver keys to recipients over another secure channel before transmitting the originally-intended message. In RSA, encryption keys are public, while the decryption keys are not, so only the person with the correct decryption key can decipher an encrypted message. Everyone has their own encryption and decryption keys. The keys must be made in such a way that the decryption key may not be easily deduced from the public encryption key.

2. Digital signatures. The receiver may need to verify that a transmitted message actually originated from the sender (signature), and didn’t just come from there (authentication). This is done using the sender’s decryption key, and the signature can later be verified by anyone, using the corresponding public encryption key. Signatures therefore cannot be forged. Also, no signer can later deny having signed the message.

This is not only useful for electronic mail, but for other electronic transactions and transmissions, such as fund transfers. The security of the RSA algorithm has so far been validated, since no known attempts to break it have yet been successful, mostly due to the difficulty of factoring large numbers $n = pq$, where p and q are large prime numbers.

1 Public-key cryptosystems.

Each user has their own encryption and decryption procedures, E and D, with the former in the public file and the latter kept secret. These procedures are related to the keys, which, in RSA specifically, are sets of two special numbers. We of course start out with the message itself, symbolized by M, which is to be “encrypted”. There are four procedures that are specific and essential to a public-key cryptosystem:

a) Deciphering an enciphered message gives you the original message, specifically

$$D(E(M)) = M \ . \tag{1}$$

b) Reversing the procedures still returns M:

$$E(D(M)) = M \ . \tag{2}$$

c) E and D are easy to compute.

d) The publicity of E does not compromise the secrecy of D, meaning you cannot easily figure out D from E.

With a given E, we are still not given an efficient way of computing D. If $C = E(M)$ is the ciphertext, then trying to figure out D by trying to satisfy an M in $E(M) = C$ is unreasonably difficult: the number of messages to test would be impractically large.

An E that satisfies (a), (c), and (d) is called a "trap-door one-way function" and is also a "trap-door one-way permutation". It is a trap door because since it's inverse D is easy to compute if certain "trap-door" information is available, but otherwise hard. It is one-way because it is easy to compute in one direction, but hard in the other. It is a permutation because it satisfies (b), meaning every ciphertext is a potential message, and every message is a ciphertext of some other message. Statement (b) is in fact just needed to provide "signatures".

Now we turn to specific keys, and imagine users A and B (Alice and Bob) on a two-user public-key cryptosystem, with their keys: E_A, E_B, D_A, D_B.

2 Privacy.

Encryption, which is now a ubiquitous way of assuring a message is delivered privately, makes it so no intruder can bypass the ciphertext, which is essentially white noise. Without property (d), however, an encryption process is still not public-key, such as the NBS standard. It requires keys to be delivered privately through another secure "courier", which is an extra process that would deem NBS, for example, as slow, inefficient, and possibly expensive. Thus, RSA is a great answer to this problem. The NBS standard could provide useful only if it was a faster algorithm than RSA, where RSA would only be used to securely transmit the keys only. Thus, an efficient computing method of D must be found, so as to make RSA completely stand-alone and reliable. For it to be reliable, it would have to use simple arithmetic, which is easier to compute (a requirement of property (c)) on a general-purpose computer than are bit manipulations, where better hardware is used, where perhaps NBS would be better.

Now, Bob wants to send a private message to Alice. He will retrieve E_A from the public file, encode M, getting $C = E_A(M)$, whereafter Alice decodes it with her own D_A, which only she can do, due to property (d). She could also reply to Bob, using E_B. So all that's needed is both user's agreement to be part of the cryptosystem by placing their encryption data into a public file. No beforehand communication is needed, private or not. Also, due to property (d), no eavesdropper can deduce D from listening in on E.

3 Signatures.

For complete assurance that the message originated form a sender, and was not just sent through him by a third party who may have used the same encryption key (that of the receiver), we need a digital signature to come with the message. This has obvious implications of importance in real-life applications.

Bob wants to send a private message to Alice. To sign the document, we pull a clever little trick, all assuming that the RSA algorithm is quick and reliable, mostly due to property (c). We decrypt a message with Bob's key, allowed by properties (a) and (b), which assert that every message is the ciphertext of another message, and that every ciphertext can be interpreted as a message. Formally,

$$D_B(M) = S \; . \tag{3}$$

Then we encrypt S with Alice's encryption key.

$$E_A(S) = E_A(D_B(M)) \tag{4}$$

This way, we can assure only she can decrypt the document. When she does, she gets the signature by $D_A(E_A(D_B(M)) = S$. She now knows the message came from Bob, since only his decryption key could compute the signature. The message need not be sent separately, since Alice can deduce it from the signature itself by using Bob's publicly available encryption key, formally $E_B(S) = E_B(D_B(M)) = M$. Since S depends on M, and the encrypted transmission Bob sent depends on S, we have a transmission that depends on both the message and the signature, so both can be deduced from the transmitted document.

This brilliantly assures the message could not be modified (if needed to be presented to, say, a "judge"), since a modified M in the form of M' would have to generate a signature $S' = D_B(M')$ as well, which is impossible, since she does not known D_B by property (d).

So not only does Alice possess proof that Bob signed the message and indeed sent it, but she also cannot modify M nor forge a signature for any other message.

Now, say an "intruder" attempted to lie and tell you he was from the public file? This is not a problem in RSA, since "signatures' are used. A signature just needs to assure it came form the public file (PF) itself. Every time a user joins a network, everybody gets a securely sent copy of the most recently updated PF, which is stored on their system, and they never have to look it up. Anyone trying to send a message pretending to be in the public file would not have the appropriate signature, and would be singled out as an "intruder". He would also never receive the PF, since he never joined it.

4 Applications, predictions, hardware implementation.

This has applications to electronic fund transmissions as well. Financial information needs to be secure, and checks can be electronically signed with RSA. Further measures would have to be taken, such as implementing unique check numbers that allow a check with this certain number transmittable/cashable only once.

In fact, such a system can be applied to *any* electronic system that needs to have a cryptosystem implemented. In their 1978 RSA paper, the authors of RSA predicted a secure email world to evolve and for RSA to be used to encrypt a live telephone conversation. Now, these things are indeed a part of more than just daily life because of RSA.

The encryption device must not be the direct buffer between a terminal and the communications channel. Instead, it should be a hardware subroutine that can be executed as needed, since it may need to be encrypted/decrypted with several different sequences of keys, so as to assure more privacy and/or more signatures.

5 The math of the method.

So far, we expect to make E and D easy to compute through simple arithmetic. We must now represent the message numerically, so that we can perform these arithmetic algorithms on it. Now lets represent M by an integer between 0 and $n-1$. If the message is too long, sparse it up and encrypt separately. Let e, d, n be positive integers, with (e, n) as the encryption key, (d, n) the decryption key, $n = pq$.

Now, we encrypt the message by raising it to the eth power modulo n to obtain C, the ciphertext. We then decrypt C by raising it to the dth power modulo n to obtain M again. Formally, we obtain these encryption and decryption algorithms for E and D:

$$\begin{aligned} C &\equiv E(M) &&\equiv M^e \pmod{n} \\ M &\equiv D(C) &&\equiv C^d \pmod{n} \ . \end{aligned} \tag{5}$$

Note that we are preserving the same information size, since M and C are integers between 0 and $n-1$, and because of the modular congruence. Also note the simplicity of the fact that the encryption/decryption keys are both just pairs of integers, (e, n) and (d, n). These are different for every user, and should generally be subscripted, but we'll consider just the general case here.

Now comes the question of creating the encryption key itself. First, choosing two "random" large primes p and q, we multiply and produce $n = pq$. Although n is public, it will not reveal p and q since it is essentially impossible to factor them form n, and therefore will assure that d is practically impossible to derive from e.

Now we want to obtain the appropriate e and d. We pick d to be a random large integer, which must be coprime to $(p-1) \cdot (q-1)$, meaning the following equation has to be satisfied:

$$\gcd(d, (p-1) \cdot (q-1)) = 1 \ . \tag{6}$$

"gcd" means greatest common divisor.

The reason we want d to be coprime to $(p-1) \cdot (q-1)$ is peculiar. I will not show the "direct motivation" behind it; rather, it will become clear why that statement is important when l show towards the end of this section that it guarantees (1) and (2).

We will want to compute e from d, p, and q, where e is the multiplicative inverse of d. That means we need to satisfy

$$e \cdot d = 1 \pmod{\phi(n)} \ . \tag{7}$$

Here, we introduce the Euler totient function $\phi(n)$, whose output is the number of positive integers less than n which are coprime to n. For primes p, this clearly becomes $\phi(p) = p - 1$. For n, we obtain, by elementary properties of the totient function, that

$$\begin{aligned} \phi(n) &= \phi(p) \cdot \phi(q) \\ &= (p-1) \cdot (q-1) \\ &= n - (p+q) + 1 \ . \end{aligned} \tag{8}$$

From this equation, we can substitute $\phi(n)$ into equation (7) and obtain

$$e \cdot d \equiv 1 \pmod{\phi(n)}$$

which is equivalent to

$$e \cdot d = k \cdot \phi(n) + 1$$

for some integer k.

By the laws of modular arithmetic, the multiplicative inverse of a modulo m exists if and only if a and m are coprime. Indeed, since d and $\phi(n)$ are coprime, d has a multiplicative inverse e in the ring of integers modulo $\phi(n)$.

So far, we can safely assured the following:

$$\begin{aligned} D(E(M)) \equiv (E(M))^d \equiv (M^e)^d \pmod{n} = M^{e \cdot d} \pmod{n} \\ E(D(M)) \equiv (D(M))^e \equiv (M^d)^e \pmod{n} = M^{e \cdot d} \pmod{n} \end{aligned}$$

Also, since $e \cdot d = k \cdot \phi(n) + 1$, we can substitute into the above equations and obtain

$$M^{e \cdot d} \equiv M^{k \cdot \phi(n)+1} \pmod{n} \ .$$

Clearly, we want that to equal M. To prove this, will need an important identity due to Euler and Fermat: for any integer M coprime to n, we have

$$M^{\phi(n)} \equiv 1 \pmod{n} \ . \tag{9}$$

Since we previously specified that $0 \leq M < n$, we know that M would *not* be coprime to n if and only if M was either p or q, of the integers in that interval. Therefore, the chances of M happening to be p or