# Digital Signatures using RSA

2013, Kenneth Levasseur
Mathematical Sciences
UMass Lowell
Kenneth_Levasseur@uml.edu

I assume the reader is familiar how one can use the RSA encryption system to encrypt a message with an individual's public key so that only that individual can decrypt the message in a reasonable amount of time.

## Initial Assumptions

Assume Alice and Bob are friends and that Alice wants to send Bob a message. In addition, assume that a third individual, Claude, want to disrupt the communications. We assume all three have public keys: $(e_A, n_A)$ for Alice, $(e_B, n_B)$ for Bob, and although we won't use it, $(e_C, n_C)$ for Claude. These three pairs are made public for everyone. In addition, the three individuals know their private keys $d_A$, $d_B$, and $d_C$, respectively. They are not made public.

## Motivation

Let's assume that Alice wants to send Bob a stock tip: "Buy hog futures asap! - Alice" She can encode the message by converting the characters of the message to list of character codes and combine blocks of the list into large numbers - in this case it might be just one number. Then using Bob's public key Alice would take every number, $m$, and send $m^{e_B} \bmod n_B$. Bob can then raise each number he receives to the $d_B \bmod n_B$ to recover the message first as numbers and then use the agreed upon encoding to get the original characters. This last step is some universal encoding that doesn't involve encryption at all.

The problem with this scheme is that Claude could compose the message "Never mind, sell your hog futures. - Alice" and follow the same steps as Alice in sending it to Bob. Although we might assume that encrypted messages sent by email identify the sender, there are ways to make an email message anonymous or seem to come from somewhere else - everyone has seen "spoof" messages that will fool some of the people some of the time. So how can Bob be sure that Alice is sending him either message? The answer is that Alice can sign her message using her private key.

## How Signing Works

For any RSA public/private key triple, $(e, d, n)$, the key mathematical fact is that the encryption and decryption functions are inverses of one another. That is, if

$f(m) = m^e \bmod n$ is the encryption function (which is public) and

$g(m) = m^d \bmod n$ is the the decryption function (which is private),

then

$f(g(m)) = m$ and $g(f(m)) = m$

The idea behind a digital signature using RSA is that $f$ is a function that is known to everyone, but only you know your decryption function. In order for Alice to sign a message, $m$, sends $g_A(m)$ together with an indication that the message is from Alice. When Bob gets it, he sees that the message is from Alice and applies her public encryption function, $f_A$, to $g_A(m)$ and will get $m$. If Claude tries to send a spoofed message, $m'$, purportedly from Alice to Bob, he has no way of being successful since he doesn't know $g_A$.

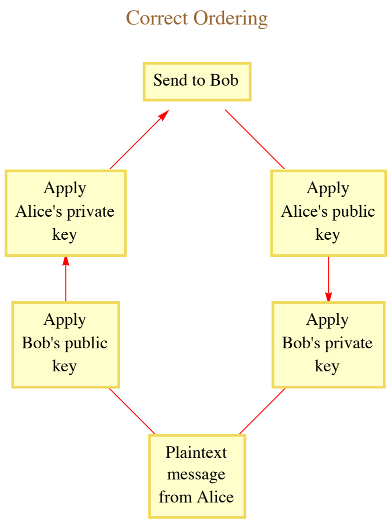
You can sign a message without encrypting it. In the scheme described in this section, anyone can intercept Alice's signed message and read it because her public key is known. Applying encryption in addition to signing a message is quite simple and is described in the next section.

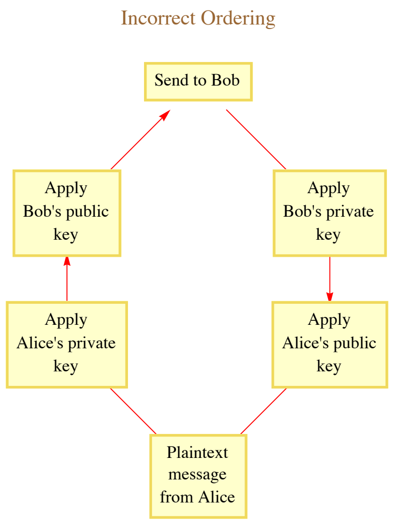
## Simultaneous Encryption and Signing

If Alice wants to sign and encrypt a message, she can follow the diagram on the bottom left, starting at the bottom of the diagram with a plaintext message and traveling clockwise. If the message is $m$, then the results coming out of the first two boxes are $f_B(m)$ and $g_A(f_B(m))$, respectively. The latter of the two numbers is what is sent to Bob. When Bob applies Alice's public key to what is received, the result is

$$\begin{aligned} f_A(g_A(f_B(m))) &= (f_A \circ g_A)(f_B(m)) && \text{by the definition of function composition} \\ &= f_B(m) && \text{because } f_A \circ g_A \text{ is the identity function.} \end{aligned}$$

Then when Bob applies his private key, he sees $g_B(f_B(m)) = (g_B \circ f_B)(m) = m$, since $g_B \circ f_B$ is the identity function.





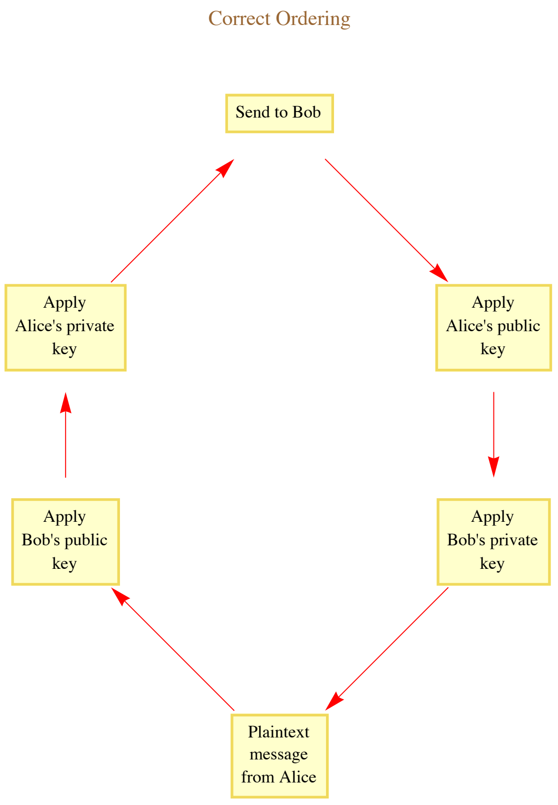
Although the mathematics works out the same if Alice follows the path on the right by first signing and then encrypting, there are some security issues that make that ordering problematic.

To recap, the order of operations for simultaneous encrypting and signing is Encrypt-Sign-Send-Authenticate-Decrypt (ESSAD).

Donald T. Davis, "Defective Sign & Encrypt in S/MIME, PKCS#7, MOSS, PEM, PGP, and XML.", Proc. Usenix Tech. Conf. 2001 (Boston, Mass., June 25-30, 2001), pp. 65-78.(180 Kbytes) (PDF, 200 Kbytes) (HTML, 80 Kbytes) Also, a shortened version of this paper appeared in Dr. Dobb's: Don Davis, "Defective Sign-and-Encrypt," Dr. Dobb's Journal #330, v.26(11) (Nov. 2001), pp. 30-36.

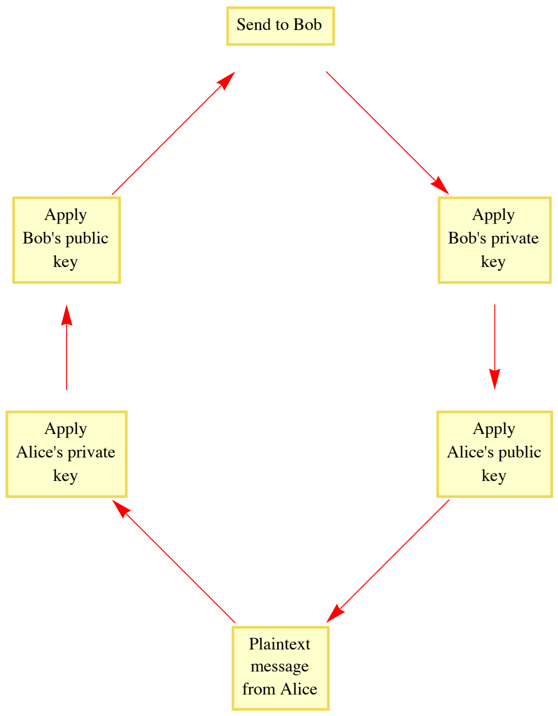
## Code

```
right = GraphPlot[{"Apply\nBob's public\nkey" → "Apply\nAlice's private\nkey",
    "Apply\nAlice's private\nkey" → "Send to Bob",
    "Send to Bob" → "Apply\nAlice's public\nkey",
    "Apply\nAlice's public\nkey" → "Apply\nBob's private\nkey",
    "Apply\nBob's private\nkey" -> "Plaintext\nmessage\nfrom Alice",
    "Plaintext\nmessage\nfrom Alice" → "Apply\nBob's public\nkey"},
  VertexLabeling → True, EdgeRenderingFunction → ({Red, Arrow[#1, 0.3]} &),
  VertexCoordinateRules → {{-1, -1}, {-1, 0}, {0, 1}, {1, 0}, {1, -1}, {0, -2}},
  PlotLabel → "Correct Ordering", LabelStyle → {Large, Brown}]
```



```
wrong = GraphPlot[{"Apply\nAlice's private\nkey" → "Apply\nBob's public\nkey",
    "Apply\nBob's public\nkey" → "Send to Bob",
    "Send to Bob" → "Apply\nBob's private\nkey",
    "Apply\nBob's private\nkey" → "Apply\nAlice's public\nkey",
    "Apply\nAlice's public\nkey" -> "Plaintext\nmessage\nfrom Alice",
    "Plaintext\nmessage\nfrom Alice" → "Apply\nAlice's private\nkey"},
  VertexLabeling → True, EdgeRenderingFunction → ({Red, Arrow[#1, 0.3]} &),
  VertexCoordinateRules → {{-1, -1}, {-1, 0}, {0, 1}, {1, 0}, {1, -1}, {0, -2}},
  PlotLabel → "Incorrect Ordering", LabelStyle → {Large, Brown}]
```

Incorrect Ordering


Send to Bob
Apply Bob's public key
Apply Bob's private key
Apply Alice's private key
Apply Alice's public key
Plaintext message from Alice