The Return of RSA

Square and Multiply

Using the RSA method, it is necessary to exponentiate modulo a number, and the exponents can be rather large. We can cut down the amount of time this takes by arranging the operations in a certain way. Let $F = Z_q$ and take α as a non-zero element of F. If $c = \alpha^m$, then c can be computed from α and m with only $\lceil 2 \log_2 q \rceil$ multiplications. The binary representation of m gives the order of the needed multiplications, which consist only of squaring and multiplying by α. For instance, if $m = 171$ then $171 = 128 + 32 + 8 + 2 + 1 = (10101011)_2$ and the computation of α^{171} is carried out by starting with 1, then, working from the most significant bit down, we square the current value and if there is a 1 in the binary representation we also multiply by α. Thus,

$$\alpha^{171} = (((((((1)^2\alpha)^2)^2\alpha)^2)^2\alpha)^2)^2\alpha)^2\alpha.$$

Protocol Failure

Dictionary Attacks

A dictionary attack on a cryptosystem occurs when it is possible to take all the components that are used to make up a plaintext and encrypt them separately (as in taking all the words in a dictionary and finding their encrypted equivalents). To decrypt a message given such a list, one only has to do a table look up to find the corresponding plaintext.

Thus, for example, if in a public key system based on factoring (such as RSA) or the discrete log problem, the plaintext message is blocked into blocks of size one (i.e., individual letters) which are then run through the encryption algorithm, the cryptanalyst has an easy method for decrypting without finding the key.

Example

The plaintext is encoded by replacing each letter with its corresponding value mod 26, i.e., A = 0, B = 1, C = 2 , etc. The RSA system was used to encipher this message using the public values n = 18721 and encryption exponent 25, and the following ciphertext was produced: **365, 0, 4845, 14930, 2608, 2608, 0**
The cryptanalyst, knowing this encoding scheme, just calculates x^{25} mod (18721) for each x in the range 0 to 25 to get the following table of values:

A = 0	F = 1759	K = 6279	P = 13444	U = 10334
B = 1	G = 18242	L = 2608	Q = 16	V = 365
C = 6400	H = 12359	M = 4644	R = 13663	W = 10789
D = 18718	I = 14930	N = 4845	S = 1437	X = 8945
E = 17173	J = 9	O = 1375	T = 2940	Y = 11373 Z =5116

365, 0, 4845, 14930, 2608, 2608, 0 → VANILLA

RSA Common Modulus Problem

Suppose that, in the RSA system, two participants have the same public modulus (but different encryption exponents). If these encryption exponents are b and c, and gcd(b,c) = 1, then the following can happen. Suppose that Alice sends the same message x to these two participants, that is Alice sends $y_1 = x^b \bmod n$ to the first and $y_2 = x^c \bmod n$ to the other. If y_1 and y_2 are intercepted by Oscar, then he can calculate:

$$d \equiv b^{-1} \bmod c,$$
$$e = (db - 1)/c,$$

and then calculate,

$y_1^d (y_2^e)^{-1} \bmod n \equiv (x^{bd})(x^{ce})^{-1} \bmod n \equiv x^{bd - ce} \bmod n \equiv x \bmod n$. Thus obtaining the message without factoring the modulus.

Other Failures

To avoid this problem, each user in the system should have a different modulus. A similar protocol failure occurs if three participants have the same encryption exponents (with different moduli), and so this should be avoided as well.