MATH 587/CSCE 557 - SUMMARY OF CLASS, 3/20/07

Suppose we have a bit stream created by a LFSR, say with n cells and the rule (computed mod 2):

$$x_{i+n} = c_0 x_i + c_1 x_{i+1} + ... + c_{n-1} x_{i+n-1} \quad (i = 0, 1, 2, ...)$$

For cryptanalysis (e.g. if we know the start of the plaintext), we find the first few $x_0, x_1, x_2, ...$ (by subtracting the plaintext from the ciphertext one bit at a time) and want to solve for $c_0, c_1, ..., c_{n-1}$. How can we do this?

In fact, knowing the first $2n$ bits of the stream, $x_0, x_1, ..., x_{2n-1}$, should be enough to solve for the rule. The point is that, once we plug in the values of $x_0, x_1, ..., x_{2n-1}$, we have n equations in n variables $c_0, c_1, ..., c_{n-1}$:

$$x_n = c_0 x_0 + c_1 x_1 + ... + c_{n-1} x_{n-1}$$

$$x_{n+1} = c_0 x_1 + c_1 x_2 + ... + c_{n-1} x_n$$

$$x_{n+2} = c_0 x_2 + c_1 x_3 + ... + c_{n-1} x_{n+1}$$

$$...$$

$$x_{2n-1} = c_0 x_{n-1} + c_1 x_n + ... + c_{n-1} x_{2n-2}$$

We then solve these equations for $c_0, c_1, ..., c_{n-1}$. There is a quicker method, the Berlekamp-Massey algorithm. So even though we can produce pseudorandom bit streams with period as large as $2^n - 1$, it only takes the first $2n$ bits to crack the key.

In traditional cryptography, someone who knows the encryption key can easily find the decryption key. In public-key cryptography this is not the case. I described the history of the topic and then introduced the RSA system.

Suppose Alice wants to send a message to Bob. Bob picks two large primes, p and q. He keeps these secret but publicizes $N = pq$. He also finds two integers d, e such that $de - 1$ is divisible by $(p-1)(q-1)$. He keeps d private but publicizes e. If Alice wants to send Bob a message, say an integer x between 0 and $N - 1$, then she encrypts it as $y = x^e \pmod N$. Bob in turn computes $y^d \pmod N$, which we claim is equal to x. The point is that given e and N, it is very hard to find d, so an eavesdropper will be hard-pressed to decrypt the message. One method would be to factor N but we assume that factorization of N is hard. We will next discuss issues that implementation and security of RSA raise.