

ECE 7670

Lecture 10 – A short look at cryptography

As may be appreciated, there are several useful connections between cryptography and error control coding. Some of the algebraic techniques used in error correction codes end up getting employed in cryptographic techniques. Other methods are entirely new, but share an analytical kinship. In any event, it is arguable that cryptography is a very important facet in the technological future, providing the difference between useful network commerce and dismal failure.

We will focus on public-key algorithms. These have a public key part and a private key part. (Describe mailing a signed letter, for example). Public key algorithms rely on the fact that some problems may be extremely difficult to solve unless a key number is known. If this number is known, then it is straightforward. The easy approach (when the key is known) is known as a “trapdoor,” and such problems with known trapdoors are called trapdoor functions.

1 RSA

Named after its inventors, Ron Rivest, Adi Shamir, and Leonard Adleman, this method gets its security from the difficulty of factoring *large* numbers.

Choose two random prime numbers p and q (of roughly equal length for best security) and compute

$$n = pq$$

Then randomly choose an encryption key e such that

$$(e, (p-1)(q-1)) = 1$$

(they are relatively prime). By the extended Euclidean algorithm, there are numbers d and f such that

$$de + f(p-1)(q-1) = 1,$$

or

$$ed \equiv 1 \pmod{(p-1)(q-1)}.$$

That is, $d = e^{-1} \pmod{(p-1)(q-1)}$. The numbers e and n are the public key; d is the private key. The factors p and q should be discarded and never revealed.

Encryption: break the message into blocks of length less than the length of n . For each block m_i , compute the encryption by

$$c_i = m_i^e \pmod{n}.$$

To decrypt we compute

$$m_i = c_i^d \pmod{n} = c_i^{ed} = c_i^{k(p-1)(q-1)+1} = m_i m_i^{k(p-1)(q-1)} \pmod{n}$$

for some integer k .

Now we need a result from number theory. Recall the $\phi(m)$ function, the number of integers less than or equal to m that are relatively prime to m . Euler's theorem:

$$a^{\phi(m)} \equiv 1 \pmod{m}$$

if $(a, m) = 1$.

Assuming that $(m_i, n) = 1$, we therefore get in our decoding

$$c_i^d = m_i,$$

as desired.

To crack this, a person knowing n and e would have to factor n to find d . (At least, this is what is believed is necessary.)

Example 1 Let $p = 47$ and $q = 71$. $n = 3337$.

$$(p - 1)(e - 1) = 3220.$$

The encryption key must be relatively prime to 3220; take $e = 79$. Then we find by the Euclidean algorithm that $d = 1019$.

To encode $m_1 = 688$, we compute

$$c_1 = (688)^{79} \pmod{3337} = 1570.$$

If we exponentiate

$$(1570)^{1019} \pmod{3337} = 688.$$

□

2 On the availability of primes

The following observations have been made by Schneier:

There are approximately 10^{151} primes 512 bits in length or less. With approximately 10^{77} atoms in the universe, and if every atom in the universe needed a billion new primes every microsecond from the beginning of time until now, you would need only 10^{109} primes. So the odds of two people picking the same prime at random is, for all practical purposes, zero. “The odds of that happening are significantly less than the odds of your computer spontaneously combusting at the exact moment you win the lottery.”

Nor can all the primes of interest be prestored. If you could store one gigabyte of data per gram, the list of the 512-bit primes “would weigh so much that it would exceed the Chandrasar [sic] limit and collapse into a black hole.”

Of course, the problem is finding long primes. This is done not by factoring, but by testing: pick a number and determine if it is prime by a primality test.

3 McEliece

(Who is a Caltech/JPL) came up with a method based on a family of error correcting codes known as Goppa codes that can correct t errors. **The private key:** an $k \times n$ generator G' for the code, a permutation matrix P and a nonsingular matrix S .

The public key: The $k \times n$ matrix $G = SG'P$.

Encryption: $c = mG + z$, where z is a random sequence of weight $\leq t$.

Decryption:

1. Compute $c' = cP^{-1}$
2. Decode to find the nearest sequence m' .
3. Compute $m = M'S^{-1}$.

This has not been subjected to successful attack. However, the key information is large. For example, a key might have $n = 1024$, $k = 524$ and $t = 50$.