Example 170. Bob's public ElGamal key is $(p, g, h) = (19, 10, 6)$. Determine Bob's secret key.

Solution. We need to solve $10^x \equiv 6 \pmod{19}$. Since we haven't learned a better method, we try $x = 1, 2, 3, \ldots$ until we find the right one: $10^2 \equiv 5$, $10^3 \equiv 12$, $10^4 \equiv 6 \pmod{19}$. Hence, $x = 4$.

What's wrong with the following alternative? Encrypting the message $m = 5$ ("randomly" choosing $y = 2$), we get the corresponding ciphertext $(c_1, c_2) = (5, 9)$. Hence, $m = c_1^{-x} c_2 \pmod{p}$, that is, $5 \equiv 5^{-x} \cdot 9 \pmod{19}$, which simplifies to $5^{x+1} \equiv 9 \pmod{19}$. If we try $x = 1, 2, \ldots$, we again find the correct secret key $x = 4$. However, $x = 13$ is another solution and is not the secret key we need to decrypt other messages. What is going wrong here?

Solution. The issue is that the base $c_1 = 5$ can be literally anything ($c_1 = g^y$ where y is random and g is Bob's primitive root). In our case, 5 has order 9 modulo 19 (check that!), so that it is not a primitive root (these have order 18). Hence, $5^9 \equiv 1 \pmod{19}$, which explains why both $5^{4+1} \equiv 9 \pmod{19}$ and $5^{13+1} \equiv 9 \pmod{19}$. The situation would be worse if c_1 had even smaller multiplicative order (think of the extreme case $c_1 = 1$).

Example 171. Bob's public RSA key is $N = 33$, $e = 13$. As you determined in the midterm, Bob's secret private key is $d = 13^{-1} \pmod{20} = 17$.

(a) Explain how the decryption of, say, $c = 26$ can be sped up using the CRT.

(b) Bob's choice of $e = 13$ is actually functionally equivalent to $e = 3$ and, similarly, d can be obtained as $e^{-1} \pmod{10}$. Can you explain and generalize these claims?

(c) An RSA user is shocked by the previous part and exclaims "RSA is only half as secure as I thought...!" How shocked should we be?

Solution.

(a) To decrypt, Bob needs to compute $m = c^d \pmod{N}$. Knowing that $N = pq = 3 \cdot 11$, we instead compute $c^d \pmod{p}$ and $c^d \pmod{q}$ [which is less work] and then use the CRT to recover $m \pmod{N}$.

Here, $26^{17} \equiv (-1)^{17} \equiv 2 \pmod{3}$ and $26^{17} \equiv 4^{17} \equiv 4^7 \equiv 4 \cdot 4^2 \cdot 4^4 \equiv 4 \cdot 5 \cdot 3 \equiv 5 \pmod{11}$.

Hence, $m = 26^{17} \pmod{33} \equiv 2 \cdot 11 \cdot (11)^{-1}_{\text{mod } 3} + 5 \cdot 3 \cdot (3)^{-1}_{\text{mod } 11} \equiv 22 \cdot (-1) + 15 \cdot 4 \equiv 5 \pmod{33}$ (as we knew from the midterm problem).

Comment. In practice, using the CRT leads to about a 4-fold speed up.

(b) If you look back at our proof of Theorem 153, you'll see that (again using the CRT) we only need $de \equiv 1 \pmod{(p-1)}$ and $de \equiv 1 \pmod{(q-1)}$ in order that $m^{de} \equiv m \pmod{pq}$.

So, instead of $d \equiv e^{-1} \pmod{(p-1)(q-1)}$, it is enough that $d \equiv e^{-1} \pmod{\operatorname{lcm}(p-1, q-1)}$.

Here, $\operatorname{lcm}(2, 10) = 10$, so that we only need $d = e^{-1} \pmod{10}$. Clearly, $13^{-1} \equiv 3^{-1} \equiv 7 \pmod{10}$.

Similarly. Bob's choice of $e = 13$ is actually functionally equivalent to $e = 3$, its value modulo 10.

(c) It is definitely misleading that RSA is "half" as secure. It is indeed the case though, that the key space for the secret key d is only half (or even less) as big as that RSA user initially thought.

However, that means that, for instance, if N is 2048 bit, then the secret key is one bit (possibly more) less than what the shocked RSA user expected. That hardly qualifies as "half as secure".

Comment. However, if $\operatorname{lcm}(p-1, q-1)$ is "too small", that is, $\gcd(p-1, q-1)$ is "too big" (so that we are loosing considerably more than 1 bit for the key size), then p, q should be discarded. If $\gcd(p-1, q-1) \approx 2^e$, then we are loosing about e bits for the key size.

Example 172. (homework) A prime p is called a **safe prime** if $(p-1)/2$ is a prime, too. Is there any advantage for RSA if p is a safe prime? What might be potential issues?

Solution. If p is a safe prime, then $\gcd(p-1, q-1) = 2$. Why?!

Hence, the key space is as large as possible.

On the other hand, we need to think about whether we are weakening the security in case we might severely limit the number of possible p's to choose from. We'll discuss the frequency of primes among large numbers soon.

Another issue is that generating random safe primes will be considerably more work. On the other hand, Bob usually does not generate a public key frequently, so that this might not be much of an issue.

Example 173. What is your feeling? Can we make RSA even more secure by allowing N to factor into more than 2, say, 3 primes?

Solution. That doesn't seem like a good idea. Namely, observe that the security of RSA relies on adversaries being unable to factor N. Allowing more factors of N (while keeping the size of N fixed) makes that task easier, because more factors means that the factors are necessarily smaller.

Example 174. RSA has proven to be secure so far. However, it is easy to implement RSA in such a way that it is insecure. One important but occasionally messed up part of RSA is that ***p* and *q* must be unpredictable**, and the only way to achieve that is to choose p, q completely randomly in some huge interval $[M_1, M_2]$.

- For instance, if $N = pq$ has m digits and we know the first (or last) $m/4$ digits of p, then we can efficiently factor N.

  An adversary might know many digits of p if, for instance, we make the mistake of generating the random prime p by considering candidates of the form $10^{100} + k$ for small (random) values of k (10^{100} has no special significance; it can be replaced with any large number).

- Also, we must use a cryptographically secure PRG to generate p and q.

- In that direction, is the security of public key cryptosystems like RSA in any way compromised when used by tens of millions of users?

  For instance, in a study of Lenstra et. al. millions of public keys were collected and compared. Among the RSA moduli, about 0.2% shared a common prime factor with another one. That's terrible: if (different) public keys N and N' share a prime factor p, then everybody can determine $p = \gcd(N, N')$ and break both public keys.

  http://eprint.iacr.org/2012/064.pdf

  One source of this unsettling state of affairs is the use of "bad" PRGs (including seeds with too little entropy).

  Similarly, for Diffie–Hellman and ElGamal, it is common to use fixed primes p. While fine in principle, this may be an issue if used by millions of users faced against an adversary Eve with vast resources. See, for instance: https://threatpost.com/prime-diffie-hellman-weakness-may-be-key-to-breaking-crypto/