Example 158. Bob's public RSA key is $N=33$, $e=13$. His private key is $d=17$.

(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. Note that the private key is $d \equiv 13^{-1} \pmod{20} \equiv 17$.

(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, $m = 26^{17} \pmod{33} \equiv 2 \cdot 11 \cdot (11)^{-1}_{\bmod 3} + 5 \cdot 3 \cdot (3)^{-1}_{\bmod 11} \equiv 22 \cdot (-1) + 15 \cdot 4 \equiv 5 \pmod{33}$.
Comment. In practice, using the CRT leads to about a 4-fold speed up.

(b) If you look back at our proof of Theorem 141, 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 159. Is there any advantage for RSA if p is a safe prime? 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.
Another issue is that generating random safe primes is 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 160. 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 the issue 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 $c_1 = 1$).

Armin Straub
straub@southalabama.edu

Further comments on RSA and ElGamal

Semantic security

Definition 161. Bob's public key cryptosystem is **semantically secure** if Eve cannot do better than guessing in the following challenge:

- Bob determines a random public and private key. The public key is given to Eve.
- Eve selects two plaintexts m_1 and m_2.
- Alice flips a fair coin and, accordingly, using the public key encrypts m_1 or m_2 as c.
- Eve now needs to decide whether c is the encryption of m_1 or m_2.

For this definition to make precise mathematical sense, we need to assume that Eve's computing power is somehow limited (typically, she is limited to polynomial-time algorithms).

Comment. Also, many variations exist of what semantic security exactly is. All of these try to capture the idea that an attacker does not learn anything about m from knowing c. The one above is often referred to as IND-CPA (Indistinguishability under Chosen Plaintext Attack).

Important comment. Realize that semantic security is a very strong property to ask for! In particular, this is much stronger than what we usually think about in terms of security: you might call a cipher secure if it is "impossible" for an attacker to get m from c. Semantic security is requiring that an attacker gets so little information from c that she cannot even tell whether it came from (her own choices) m_1 or m_2.

Example 162. Is vanilla RSA semantically secure?

Solution. No. Eve can just encrypt both m_1 and m_2 herself, and compare with c. She then knows for sure which of the two was encrypted.

Comment. As mentioned before, in practice, RSA is never used in its vanilla (or "textbook") version. Instead, it is randomized (like ElGamal is by design) by padding the plaintext with random stuff.

Check out OAEP: https://en.wikipedia.org/wiki/Optimal_asymmetric_encryption_padding

The resulting RSA-OAEP has been proven semantically secure (under the "RSA assumption" that finding m from c is hard).