11.1 Computing with encrypted messages

ElGamal

The encoding process looks as follows:

$$Enc(pk, m) = (g^r, m \cdot Y^r)$$

For m_1 and m_2 we then get:

$$\begin{aligned} Enc(pk, m_1) &= (g^{r_1}, m_1 \cdot Y^{r_1}) \\ Enc(pk, m_2) &= (g^{r_2}, m_2 \cdot Y^{r_2}) \end{aligned}$$

For the operations $\otimes$ and $\oplus$ we then get:

$$\begin{aligned} Enc(pk, m_1) \otimes Enc(pk, m_2) &= (g^{r_1}, m_1 \cdot Y^{r_1}) \otimes (g^{r_2}, m_2 \cdot Y^{r_2}) \\ &= (g^{r_1} \otimes g^{r_2}, m_1 \cdot Y^{r_1} \otimes m_2 \cdot Y^{r_2}) \\ The \otimes\ can\ be\ replaced\ with\ a\ multiplication\ (\cdot) : \\ &= (g^{r_1} \cdot g^{r_2}, m_1 \cdot Y^{r_1} \cdot m_2 \cdot Y^{r_2}) \\ &= (g^{r_1+r_2}, \underbrace{m_1 \cdot m_2}_{m_3} \cdot Y^{r_1+r_2}) \\ &= Enc(pk, m_3) \qquad\qquad with\ m_3 = m_1 \cdot m_2 \end{aligned}$$

RSA

The encoding process looks as follows:

$$Enc(pk, m) := m^{pk} \% N$$

For m_1 and m_2 we then get:

$$\begin{aligned} Enc(pk, m_1) &= m_1^{pk} \% N \\ Enc(pk, m_2) &= m_2^{pk} \% N \end{aligned}$$

For the operations $\otimes$ and $\oplus$ we then get:

$$\begin{aligned} Enc(pk, m_1) \otimes Enc(pk, m_2) &= m_1^{pk} \% N \otimes m_2^{pk} \% N \\ &= (m_1^{pk} \otimes m_2^{pk}) \% N \\ The \otimes\ can\ be\ replaced\ with\ a\ multiplication\ (\cdot) : \\ &= (m_1^{pk} \cdot m_2^{pk}) \% N \\ &= ((\underbrace{m_1 \cdot m_2}_{m_3})^{pk}) \% N \\ &= Enc(pk, m_3) \qquad\qquad with\ m_3 = m_1 \cdot m_2 \end{aligned}$$

11.2 RSA parameters

11.2.a Why e must be odd?

Because p and q are prime (and therefore odd) the product of $(p-1)(q-1)$ is even. Because e and $\Phi(pq)$ must be **coprime**, e must be odd, otherwise these two numbers would have at least 2 as a common divisor.

11.2.b Given N and $\Phi(N)$

We have given:

$$\begin{aligned} N &= pq \\ \Phi(N) &= (p-1)\cdot(q-1) \end{aligned}$$

Therefore we can compute:

$$\left| \begin{aligned} N &= pq \\ \Phi(N) &= (p-1)\cdot(q-1) \end{aligned} \right| \Leftrightarrow \left| \begin{aligned} N &= pq \\ q &= \tfrac{\Phi(N)}{p-1}+1 \end{aligned} \right| \Rightarrow N = p\cdot\left(\tfrac{\Phi(N)}{p-1}+1\right)$$

With this we can now compute p:

$$\begin{aligned} & N &= p\cdot\left(\frac{\Phi(N)}{p-1}+1\right) \\ \Leftrightarrow\quad & N\cdot(p-1) &= p\cdot\Phi(N)+p^2-p \\ \Leftrightarrow\quad & 0 &= p^2+p\cdot(\Phi(N)-N-1)+N \\ \Leftrightarrow\quad & p_{1/2} &= -\frac{\Phi(N)-N-1}{2}\pm\sqrt{\left(\frac{\Phi(N)-N-1}{2}\right)^2-N} \end{aligned}$$

The solutions p_1 and p_2 are then the prime factorization of N.

11.3 Bad choice of prime factors

11.3.a p is "small"

We assume: $|\, p \,| = O(log\ \lambda)$.
So a possible algorithm for prime factorization can look as the following:

primeFactorization(N):

$root\ := \sqrt{N};$
$i\ := 1;$
while TRUE **do**
  if $i \mid N$
    return i and N/i;
  else
    $i++;$

Because we know that in the worst case this algorithm needs $O(log\ \lambda)$ steps, it is clearly efficient in λ.

11.3.a $\mid p - q \mid$ is "small"

We assume: $\mid p - q \mid = O(log\ \lambda)$.
Therefore we know p and q must be in range of $\{\lfloor \sqrt{N} \rfloor - \frac{|p-q|}{2},\ \lfloor \sqrt{N} \rfloor + \frac{|p-q|}{2}\}$. Furthermore we know that at least one of these must be in range $\{\lfloor \sqrt{N} \rfloor - \frac{|p-q|}{2},\ \lfloor \sqrt{N} \rfloor\}$ and the other one in $\{\lfloor \sqrt{N} \rfloor,\ \lfloor \sqrt{N} \rfloor + \frac{|p-q|}{2}\}$. If we found one prime factor, the other one is trivial and therefore we only need to search in one of these ranges. The algorithm can therfore look like the following:

primeFactorization(N):

$root\ := \sqrt{N}$;
 $step\ := 0$;
 while TRUE **do**
   $i\ := root + step$;
   if $i \mid N$
     return i and N/i;
   else
     $step++$;

Because we know that in the worst case this algorithm needs $\frac{O(log\ \lambda)}{2}$ steps, it is clearly efficient in λ.

11.4 RSA oracle

We have:

$$public\ key\ = \{N, e\}$$
$$ciphertext\ c$$

We can now choose an x and compute $c' = c \cdot x^e$. Because we know that $Enc(pk, m_1) \cdot Enc(pk, m_2) = Enc(pk, m_1 \cdot m_2)$, it follows that in the decryption Alice computes:

$$\begin{aligned} Dec(d, c') &= c'^d\ mod\ N\ = c^d \cdot x^{e^d}\ mod\ N \\ &= m \cdot x\ mod\ N \end{aligned}$$

With this we can recover the plaintext m.