Modern Cryptography Dec 17, 2019

Homework 11

Lecturer: Daniel Slamanig, TA: Karen Klein, Guillermo Perez Due: 23.59 CET, Jan 8, 2020

To get credit for this homework it must be submitted no later than Wednesday, December 11th via TUWEL. If you have not registered for the tutorial (192.063 Tutorial on Introduction to Modern Cryptography 2019W) on TUWEL, please do so. If you are unable to register for the course on TUWEL for some reason, submit your homework via email to via email to `michael.walter@ist.ac.at`, please use "MC19 Homework 11" as subject.

1. Textbook RSA Encryption

   - Prove the correctness of the textbook RSA encryption algorithm as introduced in the lecture, i.e., show that for all $n \in \mathbb{N}$, $((N,d),(N,e)) \leftarrow \mathsf{KeyGen}(1^n)$ any $m \in \mathbb{Z}_N$ it holds that $(m^e)^d \equiv m \pmod N$.

     Solution: We have two different cases to consider. First, if m is in $\mathbb{Z}_N^*$, then $m^{|\mathbb{Z}_N^*|} = m^{\phi(N)} \equiv 1 \pmod N$. Thus, since $e \cdot d \equiv 1 \pmod{\phi(N)}$, we have

     $$(m^e)^d = m^{e \cdot d} \equiv m^{e \cdot d \mod \phi(N)} \equiv m \pmod N$$

     .

     What if m is not in $\mathbb{Z}_N^*$? In that case, the Chinese Remainder Theorem tells us that we have an isomorphism $\mathbb{Z}_{\mathbb{N}} \cong \mathbb{Z}_p \times \mathbb{Z}_q$, where $N = p \cdot q$, both prime (this was seen in Tutorial 9). Moreover this isomorphism is given by mapping $x \in \mathbb{Z}_N$ to $(x \mod p, x \mod q) \in \mathbb{Z}_p \times \mathbb{Z}_q$. In particular, $(m^e)^d \equiv m \pmod N$ if and only if:

     $$(m^e)^d \equiv m \pmod p \qquad \wedge \qquad (m^e)^d \equiv m \pmod q$$

     Thus, we only need to check these two equalities. Now, since $m \notin \mathbb{Z}_{\mathbb{N}}^*$, $gcd(m,N) \neq 1$, which implies that one of p or q divides m (both cannot divide m, since then $m \geq N$). Assume without loss of generality that p divides m. In this case, $m \equiv 0 \pmod p$, so clearly

     $$(m^e)^d \equiv (0^e)^d \equiv 0 \equiv m \pmod p$$

     On the other hand, since $gcd(m,q) = 1$, then $m \in \mathbb{Z}_q^*$. Also, $e \cdot d \equiv 1 \pmod{\phi(N)}$ implies $e \cdot d \equiv 1 \pmod{\phi(q)}$, since $\phi(q) \mid \phi(N)$. Thus, by the same argument as above, $(m^e)^d \equiv m \pmod q$. □

   - **[11.20 in book, 2nd edition]** Fix an RSA public key (N,e) and assume we have an algorithm $\mathcal{A}$ that always correctly computes $\mathsf{lsb}(x)$ (i.e., the least significant bit of x) given $x^e \bmod N$. Write full pseudocode for an algorithm $\mathcal{A}'$ that computes x from $x^e \bmod N$.

Solution:

Lets define $\gamma = [2^{-1} \mod N]$ to be the inverse of 2 modulo N. The idea is to use γ to bit-wise right-shift x and use $\mathcal{A}$ to learn all the bits of x one-by-one, as $x^e \cdot \gamma^e = (x \cdot \gamma)^e \mod N$.

If $\texttt{lsb}(x) = 0$ then $[\gamma \cdot x] \mod N]$ is indeed just a right-shift of x (as x is divisible by 2), so to obtain the ℓ right-most bits of x, we just need to obtain the $\ell - 1$ right-most bits of $[\gamma \cdot x] \mod N]$ and append a 0 to the right. If $\texttt{lsb}(x) = 1$, however, $[\gamma \cdot x] \mod N] = [(x+N)/2 \mod N]$, so multiplying by γ does not correspond to a right-shift. We claim the following however: the ℓ right-most bits of x are the ℓ right-most bits of $2x' - N$, where x' consists of the $\ell - 1$ right-most bits of $[\gamma \cdot x \mod N]$.

To see this, lets first define the function $b(y, d)$ to be the function returning the d right-most bits of y (essentially, $b(y, d) = y \mod 2^d$, it just makes notation slightly easier). Our claim from before can be reformulated as:

$$b(x, \ell) = (2 \cdot b(\gamma \cdot x \mod N,\ \ell - 1) - N) \mod 2^\ell$$

Now observe that:

$$\begin{aligned}
2 \cdot b(\gamma \cdot x \mod N, \ell - 1) \mod 2^\ell &= b(2 \cdot \gamma \cdot x \mod N, \ell) \mod 2^\ell \\
&= (2 \cdot \gamma \cdot x \mod N) \mod 2^\ell \\
&\Longrightarrow \\
(2 \cdot b(\gamma \cdot x \mod N, \ell - 1) - N) \mod 2^\ell &= ((2 \cdot \gamma \cdot x) \mod N - N) \mod 2^\ell \\
&= (2 \cdot \gamma \cdot x - N \mod N \mod 2^\ell \\
&= (2 \cdot \gamma \cdot x \mod N) \mod 2^\ell \\
&= b(2 \cdot \gamma \cdot x \mod N, \ell) \\
&= b(x \mod N, \ell) \\
&= b(x, \ell)
\end{aligned}$$

Thus, applying the previous reasoning recursively, we can recover x. The following algorithm `GetBits` illustrates this.

Input: $\langle N, e \rangle$; $c \in \mathbb{Z}_N^*$; ℓ
Output: the ℓ least significant bits of $[c^{1/e} \mod N]$
if $\ell = 1$ **then**
  return $\mathcal{A}(N, e, c)$
else
  $x_0 := \mathcal{A}(N, e, c)$
  $\gamma := [2^{-1} \mod N]$
  $x' = \texttt{GetBits}(N, e, [c \cdot \gamma^e \mod N], \ell - 1)$
  if $x_0 = 0$ **then**
    return $x' || x_0$
  else
    return $[2x' - N \mod 2^\ell]$
  end
end

Algorithm 1: Algorithm `GetBits`

□

- A message m is encrypted using textbook RSA encryption with keys $(493, 3)$ and $(493, 5)$ yielding ciphertexts 293 and 421 respectively. Use the fact that the two public keys share the same modulus to recover m and describe how the attack works (Hint: common modulus attack).

  Solution: We know that

  $$m^3 = [293 \mod 493]$$
  $$m^5 = [421 \mod 493]$$

  Now, since 3 and 5 are coprime, using the extended euclidean algorithm, we can find integers x, y such that

  $$3x + 5y = gcd(3, 5) = 1$$

  In this particular case $x = 2$, $y = -1$ satisfy the equation. This implies that

  $$(m^3)^x \cdot (m^5)^y = (m^3)^2 \cdot (m^5)^{-1} = m \mod 493$$

  Note that if $(m^2)^{-1}$ did not exist, m^{-1} would not exist, so m and N would not be coprime - and thus we would have found a factor of N. Hence, we can assume the inverse exists, in which case it can be easily calculated using the extended Euclidean algorithm.

  $$\begin{aligned} 493 &= 1 \cdot 421 + 72 \\ 421 &= 5 \cdot 72 + 61 \\ 72 &= 1 \cdot 61 + 11 \\ 61 &= 5 \cdot 11 + 6 \\ 11 &= 1 \cdot 6 + 5 \\ 6 &= 1 \cdot 5 + 1 \end{aligned}$$

  and hence

  $$\begin{aligned} 1 &= 6 - 5 \\ 1 &= 6 - (11 - 6) = 2 \cdot 6 - 11 \\ 1 &= 2 \cdot (61 - 5 \cdot 11) - 11 = 2 \cdot 61 - 11 \cdot 11 \\ 1 &= 2 \cdot 61 - 11 \cdot (72 - 61) = 13 \cdot 61 - 11 \cdot 72 \\ 1 &= 13 \cdot (421 - 5 \cdot 72) - 11 \cdot 72 = 13 \cdot 421 - 76 \cdot 72 \\ 1 &= 13 \cdot 421 - 76 \cdot (493 - 421) = 89 \cdot 421 - 76 \cdot 493 \end{aligned}$$

  which implies

  $$89 \cdot 421 = 1 \mod 493.$$

  and so, 89 is the inverse of 421.

Thus,

$$m = 293^2 \cdot 89 = 47 \mod 493$$

Note that this works in general if e_1, e_2 in the public keys (N, e_1), (N, e_2) are coprime. If they are not, we will only be able to find $m^{gcd(e_1, e_2)}$. □

2. Insecure Public-Key Encryption

- Let us assume the following public-key encryption scheme. Choose integers $a, b, a', b' \in \mathbb{N}$, with $a > 1$, $b > 1$, and compute:

$$M = ab - 1, \quad e = a'M + a, \quad d = b'M + b, \quad n = \frac{ed - 1}{M}.$$

The public key is (n, e), the private key is d. To encrypt a plaintext m, one computes $c = em \bmod n$. Alice decrypts a ciphertext c as $m = cd \bmod n$.

– Verify that decryption recovers the message.

Solution: Observe that $ed = 1 + nM$ and therefore $ed = 1 \mod n$. It follows that

$$\begin{aligned} cd \mod n &= emd \mod n \\ &= med \mod n \text{ (Associativity)} \\ &= m \mod n \text{ (By observation above).} \end{aligned}$$

□

– Show how the Euclidean algorithm can be efficiently used to break the encryption scheme.

Solution: By the observation above that $ed = 1 \mod n$, it follows that e is invertible modulo n. Therefore its inverse $e^{-1} \mod n$ can be found out using the Extended Euclidean algorithm (as we saw in the lecture). Now we can extract the message from a ciphertext $c = em \mod n$ by simply computing

$$e^{-1}c = e^{-1}em = m \mod n.$$

□