Textbook RSA
And its insecurities

Foundations of Cryptography
Computer Science Department
Wellesley College

Fall 2016

Table of contents

RSA encryption

- So far, lots of talk, but no action. We haven't seen a single example of a real, live public-key encryption scheme.
- That is about to change. Today we introduce a very well-known scheme based on the *RSA assumption* discussed several weeks ago.
- We start with a key generation algorithm that should look familiar.

GenRSA

Algorithm 11.25.
RSA key generation GenRSA

Input: Length n; parameter t
Output: N, e, d as described below

$(N, p, q) \leftarrow \text{GenModulus}(1^n)$*
$\phi(N) := (p-1)(q-1)$
choose e such that $\gcd(e, \phi(N)) = 1$
compute $d := [e^{-1} \mod \phi(N)]$**
return N, e, d

*$N = pq$ with p, q n-bit primes.

**Such an integer d exists since e is invertible modulo $\phi(N)$.

Textbook RSA

Construction 11.26.
Define a public-key encryption scheme using GenRSA as follows:

- Gen: On input 1^n run GenRSA(1^n) to obtain N, e, and d. The public key is $\langle N, e \rangle$ and the private key is $\langle N, d \rangle$.
- Enc: On input a public key $pk = \langle N, e \rangle$ and a message $m \in \mathbb{Z}_N^*$, compute the ciphertext

$$c := [m^e \mod N].$$

- Dec: On input a private key $sk = \langle N, d \rangle$ and a ciphertext $c \in \mathbb{Z}_N^*$, compute the message

$$m := [c^d \mod N].$$

Textbook RSA in action

Example 11.27.
Say that GenRSA outputs $(N, e, d) = (391, 3, 235)$.*

To encrypt the message $m = 158 \in \mathbb{Z}_{391}^*$ using the public key $(391, 3)$, we compute

$$c := [158^3 \mod 391] = 295.$$

To decrypt, the receiver computes

$$[295^{235} \mod 391] = 158.$$

Ta-da!

*Note that $391 = 17 \cdot 23$ so $\phi(391) = 16 \cdot 22 = 352$. Moreover, $3 \cdot 235 = 1 \mod 352$.

One thing's for sure

- Computing the private key is at least as hard as factoring moduli output by GenRSA.
- Unfortunately, this says nothing about whether the message can be recovered from the ciphertext using other means.
- Worst still, "textbook RSA" is *not* secure with respect to any of the definitions of security proposed so far.*

*Why?

Encoding binary strings as elements of $\mathbb{Z}_N^*$.

Remark. Let $\ell = \|N\|$. Binary strings of length $\ell - 1$ may be view as elements of $\mathbb{Z}_N$. Strings of varying length may be padded* to bring them up to correct length.

Concern. Not all encoded messages m lie in $\mathbb{Z}_N^*$ since it may be the case that $\gcd(m, N) \neq 1$. What then?

*In some unambiguous fashion. More on this in your homework.

The choice of e

Remark. For the most part, there does not appear to be any difference in the hardness of the RSA problem for different choices of the exponent e, as long as it isn't too small.

Concern. One popular choice is to set $e = 3$, since then computing eth powers modulo N requires only two multiplications. However, this choice does leave textbook RSA vulnerable to certain attacks* (more soon).

*More evidence that Construction 10.15 leaves much to be desired.

Attacks on "textbook" RSA

Remark. When e is small, for example $e = 3$ and the message m is such that $m < N^{1/3}$, then the encryption $c = [m^3 \mod N] = m^3$ doesn't involve any modular reduction. We can recover the message m by computing the cube root of c.

Remark. It gets worse, there is a more general attack for any size message when e is small and the message is sent to multiple receivers. To see how, we will first need ...

The Chinese remainder theorem

Theorem 8.24. Let $N = pq$ where p and q are relatively prime. Then

$$\mathbb{Z}_N \simeq \mathbb{Z}_p \times \mathbb{Z}_q \text{ and } \mathbb{Z}_N^* \simeq \mathbb{Z}_p^* \times \mathbb{Z}_q^*.$$

Moreover, let f be the function mapping elements $x \in \{0, \dots, N-1\}$ to pairs (x_p, x_q) with $x_p \in \{1, \dots, p-1\}$ and $x_q \in \{1, \dots, q-1\}$ defined by

$$f(x) \stackrel{\text{def}}{=} ([x \mod p], [x \mod q]).$$

Then f is an isomorphism from $\mathbb{Z}_N$ to $\mathbb{Z}_p \times \mathbb{Z}_q$ as well as an isomorphism from $\mathbb{Z}_N^*$ to $\mathbb{Z}_p^* \times \mathbb{Z}*_q$.

For example. Take $N = 15 = 5 \cdot 3$ and consider $\mathbb{Z}_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$.

Attacking textbook RSA using the Chinese remainder theorem

Example. Let $e = 3$, and say m was sent to three different parties holding public keys $pk_1 = \langle N_1, 3\rangle$, $pk_2 = \langle N_2, 3\rangle$, and $pk_3 = \langle N_3, 3\rangle$. The eavesdropper sees

$$c_1 = [m^3 \mod N_1] \text{ and } c_2 = [m^3 \mod N_2] \text{ and } c_3 = [m^3 \mod N_3].$$

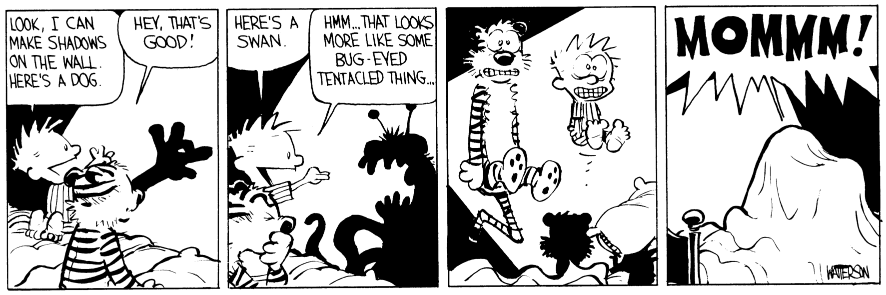
Assume $\gcd(N_i, N_j) \neq 1$ for all i, j.* Let $N^* = N_1 N_2 N_3$. An extended version of the Chinese remainder theorem says there exists a unique $\hat{c} < N^*$ such that:

$$\begin{aligned} \hat{c} &= c_1 \mod N_1 \\ \hat{c} &= c_2 \mod N_2 \\ \hat{c} &= c_3 \mod N_3 \end{aligned}$$

*If not we're done. Why?

Brute again

Since textbook RSA is deterministic, if the message m is chosen from a small list of possible values, then it is possible to determine m from the ciphertext $c = [m^e \mod N]$ by trying each value of m, $1 \leq m \leq \mathcal{L}$.



When $\mathcal{L}$ is large, as for example in the case of hybrid encryption where $\mathcal{L} = 2^{\ell}$, one might hope Brute would not be a threat. Unfortunately, ...

A quadratic improvement in recovering m

We assume that $m < 2^{\ell}$ and that the attacker knows ℓ. The value α is a constant with $\frac{1}{2} < \alpha < 1$.

Algorithm 11.28.
An attack on textbook RSA encryption

Input: Public key $\langle N, e \rangle$; ciphertext c; parameter ℓ
Output: $m < 2^{\ell}$

set $T := 2^{\alpha \ell}$
for $r = 1$ to T:
  $x_i := [c/r^e \mod N]$
sort the pairs $\{(r, x_r)\}_{r=1}^{T}$ by their second component
for $s = 1$ to T:
  if $[s^e \mod N] \stackrel{?}{=} x_r$ for some r
    return $[r \cdot s \mod N]$

Common modulus attack

The boss wants to use the same modulus N for each of its employees. Since it is not desirable for messages encrypted to one employee to be read by another other, the company issues different (e_i, d_i) to each employee.

That is, the public key of the ith employee is $pk_i = \langle N, e_i \rangle$ and the private key is $sk = \langle N, d_i \rangle$ What's wrong with this picture?

Common modulus attack: The sequel

Remark. So, suppose the employees all trust each other, and security only needs to be maintained against outsiders.

Suppose the same message m is encrypted and sent to two different employees with the public keys (N, e_1) and (N, e_2) where $\gcd(e_1, e_2) = 1$. Then an eavesdropper sees

$$c_1 = m^{e_1} \mod N \text{ and } c_2 = m^{e_2} \mod N.$$

What's the harm in that?

Padded RSA

- RSA does not possibly satisfy any of our definitions of security* and indeed is vulnerable to a number of realistic attacks.
- A simple "fix" might be to add some form of random padding to the message before encryption.

*It is deterministic.

Padded RSA: The construction

Construction 11.30.
Let ℓ be a function with $\ell(n) \leq 2n - 4$ for all n. Define a public-key encryption scheme as follows:

- Gen: On input 1^n, run GenRSA(1^n) to obtain (N, e, d). Output public key $pk = \langle N, e \rangle$, and the private key $sk = \langle N, d \rangle$.
- Enc: On input a public key $pk = \langle N, e \rangle$ and a message $m \in \{0,1\}^{\|N\| - \ell(n) - 2}$, choose a random string $r \leftarrow \{0,1\}^{\ell(n)}$ and interpret $\hat{m} := r \| m$ as an element of $\mathbb{Z}_N$. Output the ciphertext

$$c := [\hat{m}^e \mod N].$$

- Dec: On input a private key $sk = \langle N, d \rangle$ and a ciphertext $c \in \mathbb{Z}_N^*$, compute

$$\hat{m} := [c^d \mod N],$$

and output the $\| N \| - \ell(n) - 2$ low-order bits of $\hat{m}$.

How'd we do?

Warning. When ℓ is too small, so that $\ell(n) = \mathcal{O}(\log n)$, then a brute-force search through all possible values of padding r can be carried out in $2^{\mathcal{O}(\log n)}$ time.

Remark. When the padding is as large as possible and m is just a single bit, then it is possible to prove security based on the RSA assumption.

Remark. In between the situation is not so clear. For certain ranges of ℓ we cannot prove security based on the RSA assumption. But no polynomial-time attacks are known either.