The RSA system. The basics.

- Def. Let N,f ≥ 1 be integers. The RSA function associated to N,f is the function $RSA_{N,f} : \mathbf{Z}_N^* \to \mathbf{Z}_N^*$ defined by

  $RSA_{N,f}(w) = w^f \bmod N$ for all $w \in \mathbf{Z}_N^*$.
- Claim. Let N ≥ 2 and $e,d \in \mathbf{Z}_{\phi(N)}^*$ be integers such that $ed \equiv 1 \pmod{\phi(N)}$. Then the RSA functions $RSA_{N,e}$ and $RSA_{N,d}$ are
  - both permutations on $\mathbf{Z}_N^*$ and
  - inverses of each other, ie. $RSA_{N,e}^{-1} = RSA_{N,d}$ and $RSA_{N,d}^{-1} = RSA_{N,e}$.
- Proof. For any $x \in \mathbf{Z}_N^*$, modulo N:
  - $RSA_{N,d}(RSA_{N,e}(x)) \equiv (x^e)^d \equiv x^{ed} \equiv x^{ed \bmod \phi(N)} \equiv x^1 \equiv x$
  - Similarly, $RSA_{N,e}(RSA_{N,d}(y)) \equiv y$

- The RSA function associated to N,f can be efficiently computed using MOD-EXP(·,f,N) algorithm.
  - Hence, $RSA_{N,e}(\cdot)$ is efficiently computable given N,e
  - $RSA_{N,e}^{-1}(\cdot) = RSA_{N,d}(\cdot)$ is efficiently computable given N,d
  - But $RSA_{N,e}^{-1}(\cdot) = RSA_{N,d}(\cdot)$ is believed hard (without d) for a proper choice of parameters (good for crypto).
- Let's build algorithms that generate RSA parameters.
- Claim. There is an $O(k^2)$ time algorithm that on inputs $\phi(N)$, e where $e \in \mathbf{Z}_{\phi(N)}^*$ and $N < 2^k$, returns $d \in \mathbf{Z}_{\phi(N)}^*$ satisfying $ed \equiv 1 \pmod{\phi(N)}$.

- The RSA modulus generator:

Algorithm $\mathcal{K}_{\text{mod}}^{\$}(k)$
  $\ell_1 \leftarrow \lfloor k/2 \rfloor$; $\ell_2 \leftarrow \lceil k/2 \rceil$
  Repeat
    $p \xleftarrow{\$} \{2^{\ell_1 - 1}, \ldots, 2^{\ell_1} - 1\}$; $q \xleftarrow{\$} \{2^{\ell_2 - 1}, \ldots, 2^{\ell_2} - 1\}$
  Until the following conditions are all true:
  – TEST-PRIME$(p) = 1$ and TEST-PRIME$(q) = 1$
  – $p \neq q$
  – $2^{k-1} \leq pq$
  $N \leftarrow pq$
  Return (N, p, q)

- The random-exponent RSA generator:

Algorithm $\mathcal{K}_{\text{rsa}}^{\$}(k)$
- $(N, p, q) \xleftarrow{\$} \mathcal{K}_{\text{mod}}^{\$}$
- $M \leftarrow (p-1)(q-1)$
- $e \xleftarrow{\$} \mathbf{Z}_M^*$
- $d \leftarrow$ MOD-INV(e, M)
- Return $((N, e), (N, p, q, d))$
-

- Often for efficiency we want *e* to be small, e.g. 3. Then

Algorithm $\mathcal{K}_{\text{rsa}}^{e}(k)$
  Repeat
    $(N, p, q) \xleftarrow{\$} \mathcal{K}_{\text{mod}}^{\$}(k)$
  Until
  – $e < (p-1)$ and $e < (q-1)$
  – $\gcd(e, (p-1)) = \gcd(e, (q-1)) = 1$
  $M \leftarrow (p-1)(q-1)$
  $d \leftarrow$ MOD-INV(e, M)
  Return $((N, e), (N, p, q, d))$

One-wayness problems

- Def [ow-kea] For an adversary A consider an experiment:

Experiment $\mathbf{Exp}^{\text{ow-kea}}_{\mathcal{K}_{\text{rsa}}}(A)$
$((N,e),(N,p,q,d)) \xleftarrow{\$} \mathcal{K}_{\text{rsa}}(k)$
$x \xleftarrow{\$} \mathbf{Z}^*_N$; $y \leftarrow x^e \bmod N$
$x' \xleftarrow{\$} A(N,e,y)$
If $x' = x$ then return 1 else return 0

The *ow-kea* - advantage of A is defined as

$$\mathbf{Adv}^{\text{ow-kea}}_{\mathcal{K}_{\text{rsa}}}(A) = \Pr\left[\mathbf{Exp}^{\text{ow-kea}}_{\mathcal{K}_{\text{rsa}}}(A) = 1\right]$$

One-wayness problems

- Def [ow-cea] For an adversary A consider an experiment:

Experiment $\mathbf{Exp}^{\text{ow-cea}}_{\mathcal{K}_{\text{rsa}}}(A)$
$(N,p,q) \xleftarrow{\$} \mathcal{K}_{\text{mod}}(k)$
$y \xleftarrow{\$} \mathbf{Z}^*_N$
$(x,e) \xleftarrow{\$} A(N,y)$
If $x^e \equiv y \pmod N$ and $e > 1$
then return 1 else return 0.

The *ow-cea* - advantage of A is defined as

$$\mathbf{Adv}^{\text{ow-cea}}_{\mathcal{K}_{\text{mod}}}(A) = \Pr\left[\mathbf{Exp}^{\text{ow-cea}}_{\mathcal{K}_{\text{mod}}}(A) = 1\right]$$

Conjecture. The RSA function is believed to be ow-kea and ow-cea secure, i.e. the corresponding advantages of any polynomial-time (in k) adversaries are small.

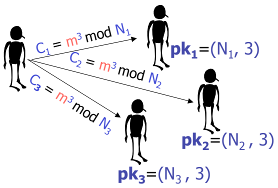
x —easy→ x^e mod N
x ←easy with d / hard without d— x^e mod N

- Let's study several known attacks that "break" RSA, i.e. compute an inverse of the RSA function on random inputs without knowing the trapdoor.

Known attacks on RSA function

1. Factoring the RSA modulus.
   - If one can factor N, i.e. compute p,q, s.t. N=pq then one can compute $d=e^{-1}$ mod (p-1)(q-1)
   - The best known algorithm to factor is GNFS.
2. Theorem [RSA with low secret exponent]. Let N=pq, where q<p<2q and p,q are prime. Let $d<1/3 \cdot N^{1/4}$. Then given (N,e) one can efficiently compute d.

3. Håstad's broadcast attack for RSA with low public exponent.



$C_1, C_2, C_3, pk_1, pk_2, pk_3$

If N_1,N_2,N_3 are relatively prime then by Chinese Remainder Theorem can combine

$$\begin{cases} C_1 = m^3 \bmod N_1 \\ C_2 = m^3 \bmod N_2 \\ C_3 = m^3 \bmod N_3 \end{cases}$$

to find $C = m^3 \bmod N_1 N_2 N_3$

Since $m^3 < N_1 N_2 N_3$ then

$m \leftarrow \sqrt[3]{C}$

9

A fix? Let's apply different polynomials to message prior to applying the RSA function.

4. Theorem [broadcast attack on padded RSA with low public exponents].
Let $N_1,\dots N_n$ be pairwise relatively prime integers and set $N_{min}=\min_i(N_i)$. Let g_i be n polynomials of maximum degree e. Suppose there exists a unique $M<N_{min}$ satisfying
$g_i(M)=0 \bmod N_i$ for all i=1,…n.
If n>e, then one can efficiently find M given all (N_i, g_i) for i=1,…,n.

5. Theorem [Related-message attack on RSA with low public exponent].
Set e=3 and let N be and RSA modulus. Let $M_1 \neq M_2 \in \mathbf{Z}_N^*$ satisfy $M_1=f(M_2) \bmod N$ for some linear polynomial f=ax+b with b≠0.
Then, given $(N,e,C_1=M_1^e \bmod N, C_2=M_2^e \bmod N)$, one can recover M_1,M_2 in time quadratic in k=|N|.

10

6. Theorem. [Coppersmith's short pad attack].
Let N,e be RSA modulus and public exponent, where |N|=k. Set $m=k/e^2$. Let $M \in \mathbf{Z}_N^*$ be a message of length at most k-m bits.
Define $M_1=2^m M+r_1$ and $M_2=2^m M+r_2$, where $0 \le r_1,r_2 \le 2^m$. Then given N,e,C_1,C_2, one can efficiently recover M.

- When e=3 the attack works as long as the pad's length is less than 1/9 of the message.

11

7. Theorem. Let N=pq be a k-bit RSA modulus. Then given k/4 least or most significant bits of p, one can efficiently factor N.

8. Theorem. Let N be a k-bit RSA modulus and let d be an RSA secret exponent. Then given the k/4 least significant bits of d, one can efficiently recover all bits of d.

Reference: http://crypto.stanford.edu/~dabo/abstracts/RSAattack-survey.html

12