Chapter 4
Public Key Cryptology - Part I

February 15, 2010

4

The concept of public key cryptology (PKC) emerged in the early 1970's in the British Government's communications center CESG, Cheltenham. (See J.H.Ellis "The Possibility of Secure Non-Secret Digital Encryption", CESG Report, January 1970).

The idea was that a person A would have two keys, a secret (private) key K_{AS} and a public key K_{AP}. Anyone wishing to send a message m secretly to A would encrypt it with K_{AP}. Nobody, except A, could decrypt it - with K_{AS}.

$$\begin{array}{ll} \textit{Encryption} & m \longrightarrow \mathrm{E}(\mathrm{K}_{AP};\ m) = \text{cypher-text} = c \\ \textit{Decryption} & \mathrm{D}(\mathrm{K}_{AS};\ c) = m \end{array}$$

The earliest practical implementation of this idea is due to Clifford Cocks (CESG Report, 20^{th} November 1973). It was as follows:

$A's$ public key, $K_{AP} = p.q = n$, the product of two primes. $(p,\ q-1)$ and $(q,\ p-1)$ are co-primes. The factors p and q are secret, so the secret key $K_{AS} = p\ and\ q$. Clearly this secrecy rest on the infeasibility of factorising n, assumed to be very large.

To encrypt m the sender forms $c{=}m^n\ mod\ n$. To decrypt the receiver, holder of the secret factors, forms

$$\begin{aligned} \mathrm{m}_1 &= \mathrm{c}^{p'}\ mod\ q \\ \mathrm{m}_2 &= \mathrm{c}^{q'}\ mod\ p \end{aligned}$$

where $p.p' = 1\ mod\ (q-1)$ and $q.q' = 1\ mod\ (p-1)$. Euclid's algorithm is used to find p' and q'. Clearly

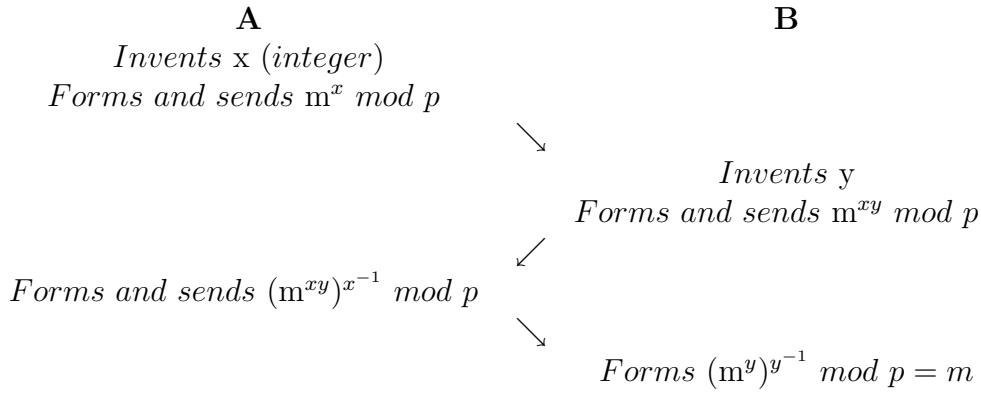
$$\begin{aligned} \mathrm{m}_1 &= \mathrm{c}^{p'} = \mathrm{m}^{pqp'} = \mathrm{m}^{q(1+t(q-1))} = \mathrm{m}^{q} = \mathrm{m} \bmod \mathrm{q} \\ \mathrm{m}_2 &= \mathrm{c}^{q'} = \mathrm{m}^{qpq'} = \mathrm{m}^{p(1+s(p-1))} = \mathrm{m}^{p} = \mathrm{m} \bmod \mathrm{p} \end{aligned}$$

(using $x^{p-1} = 1\ mod\ p$, etc. see Chapter 3, Modular Arithmetic)

so m can be found using the Chinese Remainder Theorem. It is the solution *mod* n (where $n = p.q$) to m_1 *mod* q and m_2 *mod* p.

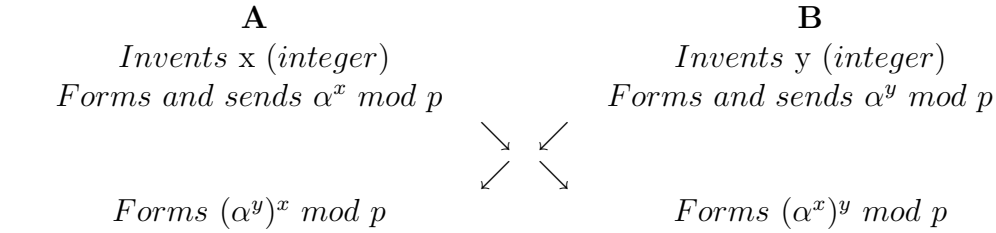
Cock's solution is none other than RSA, see below, but simpler because the public key is a single integer and with built-in CRT speed-up. (Moreover, unlike RSA, it does not rely on the Discrete Logarithm (DL) problem whereby an RSA secret key exponent might be vulnerable when c and m are both known.)

In 1974 Malcolm Williamson followed with a three-pass scheme for conveying a secret m between two persons A and B sharing a modulus p.



Note x^{-1} and y^{-1} are found *mod* $\phi(p)$. The system is that used to pass a message in a padlocked bag. A padlocks it, B adds his padlock, A removes his padlock, B removes his one. This scheme subsequently became known as Massey Omura.

Williamson then invented the system for A and B to establish a shared secret using a common modulus p, and base α.



$$Shared\ Secret\ (SS) = \alpha^{xy}\ mod\ p$$

Note that both Williamson schemes depend on the infeasibility of solving for DLs. Observing α^x, α^y and knowing α and p does not reveal x or y. This latter scheme was later 'invented' by Diffie and Hellman, and is popularly known by their names.

4.1 Certification

An obvious problem with these schemes for secret communication is:

1. How does one know that the other party A is who he claims to be?
2. How does a sender of a secret m in the Cocks scheme know that n belongs to A and not to an imposter?
3. In the Williamson schemes, how does A know that he is sharing secrets with B and not with an imposter C?

The solution is for a person's public information (eg. a public key) to be certified by a Trusted Third Party (TTP) as genuine. Certification is performed by the TTP digitally signing a block of data called a Certificate which contains

1. A user's identity in some agreed form (name, address)
2. His public information, e.g. K_{AP} his public key
3. Relevant dates: of creation, of expiry etc.
4. Scope of use, attributes etc.
5. The TTP's signature to the above items

The public, with access to a certificate, may verify that the user is genuine, or at least considered so by the TTP, and that the public information does indeed correspond to that user.

We see from the form of the certificate that it is necessary to consider *digital signatures* which are not part of the above CESG schemes. Those schemes are all as their name suggests, about non-secret encryption. The Cocks scheme relies not on a secret key but rather on secret knowledge: p, q the factors of n. The subsequent RSA scheme uses an explicit secret key, as well as a public key, which enables its use for digital signature.

Note These PKC techniques are often known as "Asymmetric Key techniques" to distinguish them from traditional "symmetric key" techniques in while two parties A, B share a common secret key for secure communication. Asymmetric techniques get over the problem of distributing the shared secret key securely by using a public key - but that of course must be genuine, (example: as verified by certification)

4.2 RSA

The RSA technique is named after its authors Rivest, Shamir and Adelman who presented it in 1977, MIT Memmo LCS!TM82 4/4/77. (Given the close collaboration between British and US Intelligence and Secerity Services, and the frequent involvement of research establishments, it is hard to believe that the authors of RSA were totally ignorant of the work at CESG over the previous years. It is notable that their original paper was called "A Method for Obtaining Digital Signatures and Public-key Crypt-systems" emphasising the signature as opposed to the encryption capabilities of a technique otherwise almost identical to that of Cocks.) RSA is the most widely used PKC technique.

For RSA a user has a key-pair and uses a modulus

$$n = p.q = \textit{the product of two secret primes.}$$

He has a public key exponent e and secret key exponent d.

$$d = e^{-1} \ mod \ \phi(n)$$

Without factorising n, $\phi(n) = (p-1)(q-1)$ cannot be found, hence d cannot be found. For encryption we have,

$$\begin{aligned} \textit{Encryption of } m &: \ \mathrm{c}=\mathrm{m}^e \ mod \ n \\ \textit{Decryption of } c &: \ \mathrm{m}=\mathrm{c}^d (= m^{ed} = m^{(1+r\phi(n))} \ mod \ n) \end{aligned}$$

(See Chapter 3)

But we can also create a digital signature to m

$$\textit{Sign } m: \ \mathrm{S}=\mathrm{Signature}=\mathrm{H(m)}^d \ mod \ n$$

Verification of signature to m: Check $H(m) = S^e \ mod \ n$. Here $H(m)$ is a hash or digest of the message and the verification consists of hashing the received m and checking it equals the "decrypted" signature. Note the $H(m)$ is usually signed rather than m to avoid exposing d should m be trivial, and also to handle very long m (larger than n). Likewise for encrypting long messages a symmetric algorithm with $key = k$ is often used and then k is encrypted and transported secretly under RSA.

$$\begin{array}{rcl} & A & B \\ \mathrm{k}' = k^e \ mod \ n & \longrightarrow & \mathrm{k}=(\mathrm{k}')^d \ mod \ n \\ \mathrm{c}=\mathrm{E(k;m)} & \longrightarrow & \mathrm{m}=\mathrm{D(k;c)} \end{array}$$

Here (d, n) is $B's$ secret key and (e, n) is $B's$ public key, to be used by all who with to send him secret messages and by all who wish to verify $B's$ signature to a message.

It is usual, however, for a user to have two RSA key-pairs: one for encryption and secret messages; one for digital signatures. This obviates a potential

fraud where F, a fraudster, gets hold of ($c = m^e \ mod \ n$) a cypher-text sent to B and presents it to B as innocent data for signing. If B then performs $c^d \ mod \ n$ he reveals m to F. This fraud is frustrated if B, as above, signs $H(c)$ rather than c.

The importance of digital signatures is due to their blocking of repudiation. A sender cannot repudiate a signed message because he alone has the secret key which signed it. This is different to the case of Message Authentication Check (MAC), where sender and receiver share a secret key. A sender can repudiate the MAC attached to a message claiming that the receiver created it himself. Certificates, signed by a TTP, with its secret key are non-repudiable - the holders of the TTP's public key cannot create the certificates.

Note the the security of RSA relies on

1. Inability to factorise n and hence find d
2. Inability to find discrete logarithms (d) either from $m = c^d \ mod \ n$ or $s = H(m)^d \ mod \ n$ knowing $n, \ m, \ c, \ H(m)$ or s.

Example RSA encryption $p = 7, \ q = 11, \ n = 77$
$(p-1) = 6, \ (q-1) = 10$
$e = 13, \ d = 37, \ (e.d = 1 \ mod \ 60)$

Accordingly we suppose A has $Public \ Key(13, \ 77)$ and $Secret \ Key \ (37, \ 77)$
B sends message $m = 2$ to A as follows:
$cyphertext = c = 2^{13} \ mod \ 77 = 30$
A deciphers it to $m = 30^{37} \ mod \ 77 = 2$

Note that decryption (or more generally operations with d the secret key exponent) can be speeded up if we use the Chinese Remainder Theorem (as did Cocks); assuming that the holder of d knows the factors of n namely $p, \ q$.

$Form \quad \mathrm{d}_p = d \ mod \ (p-1)$ and $d_q = d \ mod \ (q-1)$
$then \quad \mathrm{m}_p = c^{d_p} = (c^d \ mod \ n) \ mod \ p, \quad m_q = c^{d_q} \ mod \ q = (c^d \ mod \ n) \ mod \ q$

and we can form $m \ mod \ p.q$ by means of the Theorem. Thus using the above example

c=30 $\quad d_p = d \ mod \ (p-1) = 37 \ mod \ 6 = 1$
$d_q = d \ mod \ (q-1) = 37 \ mod \ 10 = 7$
$m_p = 30^1 \ mod \ 7 = 2$
$m_q = 30^7 \ mod \ 11 = 8^7 \ mod \ 11 = 2$
$Therefore$ m=2 mod 77

4.3 RSA Parameters

A user's parameters $p,\ q,\ e$ (and hence d) must be created somewhere. Either the user creates his own, in which case he must follow strict procedures if they are to be secure. This can be done by the use of approved software. However the result must be checked independently (eg. by a TTP) to ensure it is unique as well as valid. No two persons should have the same keys. Additionally the TTP should certify the user's public key - and probably issue it on some accessible data base.

Or the TTP creates user keys and delivers them securely (how?) to their owners. In this case the TTP must destroy it's own copy of a user's secret key after delivery to its rightful owner.

There are obvious administrative and management problems to be solved in establishing a secure RSA service supporting a large number of users.

To generate large primes - and for security against factorisation and other attacks n will be 512 bits at least, but more likely 2048 - the standard method is to start with a random number of suitable size. It is tested for primality, and if it fails the test it is slightly modified and the test repeated until success is achieved.

If p is a prime then $\alpha^{p-1} = 1 \ mod \ p$ for all α (see Chapter 3) so a test could be: try a large number of α, and if the equation holds assume p is prime. Unfortunately this is inadequate. The existence of Carmichael numbers could lead one to suppose n is as prime when it is not because $\alpha^{n-1} = 1 \ mod \ n$ for all α, if n is a Carmichael number.

(If $n = \prod p_i$, where p_i is prime, then $\phi(n) = LCM \ \phi(p_i)$,and if $\phi(n) \mid (n-1)$ we have a Carmichael number. Example $n = 1729 = 7*13*19$. $\phi(n) = LCM(6,\ 12,\ 18) = 36$. $36 \mid 1728$). An improved version of this test is due to Rabin, see appendix 7X.

But it is not sufficient to find any primes for p and q. They must satisfy special conditions, namely they must be strong primes. $(p-1)$ must have a large prime factor p' and in turn $(p'-1)$ must have a large prime factor p''. Similarly $(q-1)$ must have a large prime factor q' and $(q'-1)$ must have a large prime factor q''.

One reason for this is to obviate an attack by repeated encryption of cypher-text c, a message m encrypted under a public key exponent e.

$$c = m^e \ mod \ n$$

An attacker sees c and knows $(e,\ n)$. He forms $(((c^e)^e)^{e\cdots})^e = c \ mod \ n$. That is, he encrypts c with e until he gets back to c. This is bound to happen - we are in a finite group. How soon it happens depends on the order of c, which is outside the attacker's control and usually outside the control of the message originator.

If $c^{e^k} = c \ mod \ n$ then $c^{e^{k-1}} = m \ mod \ n$ and the attacker has decrypted c to m. To obviate this attack the probability (order of $c \ mod \ n$ is small) must be low. Similarly the probability (order of $e \ mod \ \phi(n)$ is small) must be low.

The first is ensured by the fact that the probability (order of c is divisible by $p'q'$) is high. The second is ensured by the fact that the probability (order of e is divisible by $p''q''$) is high.

We know that if p, a prime, is such that $p-1 = p'r$ with p' a large prime then the number of elements $mod \ p$ whose order is divisible by $p' \geq 2\phi(p') = 2(p'-1)$. See Chapter 8.

If the RSA modulus n=p.q	*and* Order $(\alpha) = p'r \ mod \ p$
	and Order $(\alpha) = q's \ mod \ q$
Then Order $(\alpha) \ mod \ pq =$	$\lambda(p'r, q's) = p'q'\lambda(rs) \geq 2p'q'$
There exist	$2\phi(p')$ elements $mod \ p$ whose order is divisible by p'.
	$2\phi(q')$ elements $mod \ p$ whose order is divisible by q'.

so there exist $4\phi(p')\phi(q')$ elements $mod \ pq$ whose order is divisible by $p'q'$, of which $3\phi(p')\phi(q')$ have order divisible by $2p'q'$. The proportion of elements whose order is not divisible by $p'q'$ is $P = 1 - \frac{4\phi(p')\phi(q')}{pq-1}$.

Example $p = 47,\ q = 59,\ p' = 23,\ q' = 29$
$P = 1 - \frac{4*22*28}{47*59-1} = 0.111$

Thus large p', q' as factors of $(p-1),\ (q-1)$ ensure that the probability that *order* (c) is large $mod \ pq$. Similar reasoning applies to the probability that $Order \ (e) \ mod \ (p-1)(q-1)$ provided we have large p'', q'' as factors of $(p'-1),\ (q'-1)$.

Example $p = 167,\ p-1 = 166,\ p' = 83,\ p'' = 41$
$q = 359,\ q-1 = 358,\ q' = 179,\ q'' = 89$
$p''q'' = 3649 \quad p'q' = 14857 \ pq = 59953$

$Prob(Order\ c\ is\ divisible\ by\ 14857) \geq \frac{4*82*178}{59952} \sim 97\%$
$Prob(Order\ e\ is\ divisible\ by\ 3649) \geq \frac{4*40*88}{83*179-1} \sim 95\%$

However, for very large integers (and perhaps we may consider $(p-1),\ (q-1)$ as very large "ordinary" integers) the probability that they have very large prime factors is very high and maybe it is not necessary to construct suitable $p,\ q$ in that case. But normally some algorithm for creating suitable $p,\ q$ similar to that in appendix Y is used.

For example, the probability that the largest prime factor of an integer x is less than $x^{.25}$ is almost zero, so that if RSA factors $p,\ q$ are of order 2^{2048} then we may perhaps assume that there exist $p',\ q'$ of order of 2^{500} and $p'',\ q''$ of order 2^{125} making the repeated encryption attack insensible.

Appendix Y sketches a method for generating RSA parameters (without making this assumption).

The choice of e.

In many systems a community uses a standard e, public key exponent, so that users' public keys are just their modulus n. This simplifies management, and the choice of a small e reduces the complexity of its use (exponentiation is of a cubic complexity: double the length of a number and you increase the time taken by a factor of 8).

A typical choice for e is Fermat-5, or $2^{16}+1$. Exponentiation then involves 16 squarings and a multiplication.

Note, however, that a small exponent can give problems. For example, suppose $e = 3$ were chosen, then the same secret message m sent to 3 different recipients with moduli $n_1,\ n_2,\ n_3$ would result in 3 cypher-texts.

$$c_1 = m^3\ mod\ n \quad c_2 = m^3\ mod\ n_2 \quad c_3 = m^3\ mod\ n_2$$

An eavesdropper seeing these could put them together using the Chinese Remainder Theorem to find

$$c = m^3\ mod\ (n_1 n_2 n_3)$$

and find m by extracting the cubic root of c.

Appendix X
Finding Large Primes

The test $x^{n-1} = 1 \; mod \; n$? (for any $2 \leq x \leq n-2$) will give a positive result if n is prime. But if n is not prime it can still give a misleading positive result. In particular, if n is a Carmichael Number, all $x's$ will give a positive result, giving the impression that n is prime.

Example $n = 561 = 3 * 11 * 17 \;\; n - 1 = 560$
$LCM(\phi(n_i)) = LCM(2,\ 10,\ 16) = 80 \mid 560$
$Therefore\ Order\ (x) \mid 80 \mid 560$
for all x; so $x^{n-1} = 1 \; mod \; n$ for all x. Except of course if $HCF(x,\ n) \neq 1$.

A better test is needed for the primality of n. The Rabin test is such a test. If $n - 1 = q^{2^k}$, this test looks at $(x^{q2^i} \; mod \; n)$ for $i = 0$ to k. The test operates on an arbitrary x and with increasing i. The algorithm is as follows:

1. Set $i = 0$. $x^{2^i q} = 1 \; mod \; n$? If true, exit with "$n$ probably prime".
2. $x^{2^i q} = -1 \; mod \; n$? If true, exit with "$n$ probably prime".
3. $i = i + 1 \;\; x^{2^i q} = 1 \; mod \; n$? If true exit with "$n$ not prime" (because in step 2 the square root of this was not -1 *or* 1).
4. $i = k$? If true exit with "n not prime (because $x^{n-1} \neq 1 \; mod \; n$)
5. Go to step 2.

Essentially this test tries to find the first time (for increasing i) that $x^{q2^i} = 1 \; mod \; n$, and ensure that the square roots of this quantity are ± 1. (If n is composite other square roots exist.)

The test can give misleading positive results for certain choices of x. That is, it can say "n is probably prime" when n not a prime. The test will never say "n is not prime" when n is prime.

It is shown below that the probability, p, of a misleading positive result is such that $p \leq \frac{1}{4}$. In other words the number of $x's$ which will give a misleading result. N say, is such that $\frac{N}{n-1} \leq \frac{1}{4}$. Thus, if the Rabin Test is repeated m times, i.e. with m different $x's$ the probability that n is not prime (after m "probably prime" outputs)$\leq (\frac{1}{4})^m$. That is

$$Prob(n\ is\ prime) \geq 1 - (\tfrac{1}{4})^m$$

Clearly, for large enough m, we can be virtually certain that n is prime.

Proof that $p \leq \frac{1}{4}$ is relatively long. There are four parts to it, each handling different types of composite n, and counting the number N of solutions (values of x) to $x^{q2^i} = \pm 1 \; mod \; n$, and showing $\frac{N}{(n-1)} \leq \frac{1}{4}$.

The four parts handle:

1. $n = \prod n_i^{e_i}$ with n_i primes, and not all $e_i = 1$. (In this case it is sufficient to consider only ($x^{n-1} \; mod \; n = 1$?)-which holds for all "probably prime" outputs of the algorithm)

2. $n = n_1 n_2 ... n_r$ for $r \geq 3$

3. $n = n_1 n_2$ with $n_1 - 1 = 2^{k_1} q_1$, $n_2 - 1 = 2^{k_2} q_2$ and $k_1 \neq k_2$

4. As in 3 but for $k_1' = k_2$

1. $n = \prod n_i^{e_i}$ with at least one $e_i > 1$. Consider an exponent t (later we put $t = n - 1$). How many solutions are there to $x^t = 1 \; mod \; n_i^{e_i}$ (i.e. modulo a prime component of n). There exists a primitive α_i and we can express $x = \alpha_i^{y_i} \; mod \; n_i^{e_i}$.

$$Order(\alpha_i^t) = \frac{\phi(n_i^{e_i})}{HCF(t, \; \phi(n_i^{e_i}))}$$

Solutions to $x^t = 1 \; mod \; (n_i^{e_i})$ are then $x = \alpha_i^{y_i}$ with $y_i = 0, \; s_i, \; 2s_i, \; \ldots (h_i - 1)s_i$ and $s_i = \frac{\phi(n_i^{e_i})}{h_i}$. (Because $(\alpha_i^{y_i})^t = (\alpha_i^t)^{y_i} = 1 \; mod \; (n_i^{e_i})$), and the $\alpha_i^{y_i}$ are all distinct since α is primitive). Therefore there exist h_i solutions.

Putting this result together for all $n_i^{e_i} \; i = 1 \; to \; r$ we get that there exist $\prod h_i$ solutions to $x^t = 1 \; mod \; n$ using the Chinese Remainder Theorem.

Now take $t = n - 1$
$h_i = HCF((n-1), \; n_i^{e_i - 1}) \leq (n_i - 1)$
because there is no common factor between $(n-1)$ and n_i.

$$\begin{aligned} \mathrm{N} &= \text{Number of solution to } \mathrm{x}^{n-1} = 1 \; mod \; n \\ &\leq \prod (n_i - 1) \\ Now(\mathrm{n}_i - 1) &\leq \mathrm{n}_i^{e_i} \; for \; e_i = 1 \end{aligned}$$

$and(\mathrm{n}_i) \leq \frac{2}{9} n_i^{e_i} \; for \; e_i \geq 2$
taking the smallest values $n_i = 3, \; e_i = 2$. So $N \leq \frac{2}{9} \prod n_i^{e_i} = \frac{2}{9}(n)$. So $p = \frac{N}{n-1} \leq \frac{2}{9} \quad \frac{n}{n-1} \leq \frac{1}{4}$. Provided $n \geq 9$.

2. Suppose $n = \prod n_i$ and $n_i - 1 = 2^{k_i} q_i$, $n - 1 = 2^k q$.

   Define $h_i = HCF(n-1,\ n_i - 1) = 2^{k'_i} q'_i$ with $k'_i = (k_i,\ k)$ and $q'_i = HCF(q,\ q_i)$.

   Consider $(mod\ n_i)$ the number of solutions to $x^q = 1\ mod\ n_i$. It is $q'_i = HCF(q,\ \phi(n_i)) = HCF(q,\ 2^{k_i} q_i)$ and to $x^{2^j q} = -1\ mod\ n_i$. (It is half the number of solutions to $x^{2^{j+1} q} = 1\ mod\ n_i$ (with $j + 1 \le k$)

   $$\begin{aligned} i.e.((\text{HCF})/(2))(2^{j+1} q,\ 2^{k_i} q_i) &= \text{q}'_i 2^j \text{ if } 1 \le j \le k'_i \\ ((\text{HCF})/(2))(2^{j+1} q,\ 2^{k_i} q_i) &= 0 \text{ if j+1>k}'_i \end{aligned}$$

   because if $k'_i = k_i x^{2^{k_i} q} = 1\ mod\ n_i$ implies $x^{2^{k_i} q} = 1\ mod\ n_i$).

   Using the Chinese Remainder Theorem

   $$\begin{aligned} \text{N} &= \text{Number of solution to } \text{x}^{2^j q} = 1\ mod\ n\ (with\ j = 0) \\ and\ to\quad & \text{x}^{2^j q} = -1\ mod\ n\ (with\ 0 \le j < k) \\ is\quad & \text{N} = \textstyle\prod_{i=1,r} q'_i + \Sigma_{0 \le j < smallest k'_i (= k_1)} \prod_{i=1,r} \end{aligned}$$

   (Because $k_1 \le k$, since $n = Z_q^k + 1 = \prod n_i = \prod(1 + 2^{k_i} q_i)$)
   or $2^k q = 2^{k_1} q_1 + 2^{k_2} q_2 + 2^{k_r} q_r + Product\ terms\ with\ factor\ 2^{k_1}$
   Therefore $k \ge k_1$ where $k_1 \le k_2 \le \ldots_r$)
   So $N = \prod q'_i (1 + \sum_{0 \le j < k} 2^{jr})$
   Now $\phi(n) = \prod(n_i - 1) = \prod_i q_i 2^{k_1 + k_2 + \ldots + k_r}$
   So

   $$(\text{N})/(\phi(n)) \le (1 + \textstyle\sum_{j < k_1} 2^{jr})/(2^{k_1 + k_2 + \ldots + k_r}) \qquad (0.1)$$

   $\le (1 + \sum_{j<k_1} 2^{jr})/(2^{k_1 r})$ since k_1 is smallest
   $= (1 + (2^{r k_1} - 1)/(2^r - 1))/(2^{r k_1}) \le \frac{1}{2^r} + \frac{1}{2^r - 1} - \frac{1}{(2^r - 1) 2^r}$ since $k_1 \ge 1$
   $= (1)/(2^{r-1})$

   $(N)/(n-1) \le (N)/(\phi(n)) < \frac{1}{2^{r-1}} < \frac{1}{4}$ Provided r (the number of prime factors of n) ≥ 3

3. Suppose $n = n_1 n_2$ $k_2 > k_1 \ge 1$
   Then

$$\begin{array}{rcl} \mathrm{N}_{\overline{\phi(n)}} & \leq & (1+\Sigma_{j<k_1} 2^{jr})/(2^{k_1+k_2}) \\ \textit{from equation} & ?? & \textit{with } \mathrm{r}{=}2 \\ & \leq & (1{+}1{+}2^r + 2^{2r} + \ldots + 2^{(k_1-1)r})/(2^{k_1+k_2}) \\ & = & (1{+}(2^{k_1} r - 1)/(2^r - 1))/(2^{k_1+k_2}) \\ & \leq & 1/8 + (1/3)\ ((2^{k_1})/(2^{k_2}) - 1/8) \\ & \leq & 1/3\ (1/2 + 1/4) = 1/4 \end{array}$$

(using $r = 2,\ k_1 + k_2 \geq 3$)

4. Finally, if $n = n_1 n_2$ and $k_1 = k_2$ we first show $q_1' \neq q_1$ and $q_2' \neq q_2$ simultaneously.

   For consider $(n{-}1) = n_1 n_2 {-} 1 = n_1 n_2 {-} n_1 {+} n_1 {-} 1 = n_1(n_2{-}1){+}(n_1{-}1)$. Suppose $n_2 > n_1$. Then $(n-1)$ cannot be divisible $(n_2 - 1)$. Therefore $2^{k_1} q_1$ and $2^{k_2} q_2$ cannot both divide $2^k q$. But $k_1 \leq k$ and $k_1 = k_2$ by hypothesis

$$\begin{array}{rcl} \textit{Therefore either } \mathrm{q}_1' & = & \mathrm{HCF}(\mathrm{q}_1,\ q) < q_1 \\ \textit{or } \mathrm{q}_2' & = & \mathrm{HCF}(\mathrm{q}_2,\ q) < q_2 \text{ holds} \end{array}$$

   Suppose $q_1' < q_1$ since $q_1' | q_1$ and these numbers are odd. We have $q_1'/3$

$$\begin{array}{rcl} \mathrm{N}/\phi(n) & = & ((\mathrm{q}_1' q_2')/(q_1 q_2))(1 + \Sigma_{j<k_1} 2^{rj})/2^{2k_1} \\ & & (\textit{See expression for } \mathrm{N},\ \phi(n)) \\ \textit{So } \mathrm{N}/\phi(n) & \leq & 1/3\ (1{+}\ (2^{2k_1} - 1)/3)/2^{2k_1} \\ & = & (1/3)\ ((2/3)/2^{2k_1} + 1/3) \\ & \leq & (1/3)\ (1/6 + 1/3)\ \textit{since } \mathrm{k}_1 \geq 1 \\ & = & 1/6 \\ & < & 1/4 \end{array}$$

Example Test if 561 is prime
$n = 561,\ n - 1 = 560 = q2^k = 35 * 2^4$
$x^{560} = 1\ mod\ 561\ all\ x$
Try $x = 2$

$$\begin{array}{rcl} 2^{35} & = & 263 \bmod 561 \\ 2^{70} & = & 166 \bmod 561 \\ 2^{140} & = & 67 \bmod 561 \\ 2^{280} & = & 1 \end{array}$$

So *not prime* because $\sqrt{1} = \sqrt{2^{280}} \neq -1$
(But if we chose $x = 101$, we would get $101^{35} = -1\ mod\ 561$ and 561 "probably prime")

Appendix Y

Sketch of procedure to generate RSA keys (512 bits)

1. Pick random $p^4 \sim 200\ bits$.
2. p'' prime? $\rightarrow No \rightarrow$p$'' = p'' + 2$. Go to step 2.
3. $p' = 2p'' + 1$
4. p' prime? $\rightarrow No \rightarrow$p$' = p' + 2p''$. Go to step 4.
5. $p = 2p' + 1$
6. p prime? $\rightarrow No \rightarrow$p = p + 2p$'$. Go to step 6.
7. $(p-1)$ prime to e? $p = p + 2p'$. Go to step 6.
8. Generate q'', q' as per step 1 and step 4.
9. $k = \frac{2^{500}}{pq'}$ an integer. Make k even.
10. $q = kq' + 1$
11. q prime? $\rightarrow No \rightarrow$q = q + 2q$'$. Go to step 11.
12. $(q-1)$ prime to e? $\rightarrow No \rightarrow$q = q + 2q$'$. Go to step 11.
13. $n = pq,\ d = e^{-1}\ mod\ (p-1)(q-1)$