RSA Assumption

For all PPT algorithms $A$, for all sufficiently large $n$,

$$\text{Prob}(A(N,e,x^e \mod N) = x) < \text{neg}(n)$$

(over $n$-bit $N=pq$, $\gcd(e, \phi(N)) = 1$, $x$ in $\mathbb{Z}_N^*$)

RSA Assumption is true ONLY IF
Factoring Assumption is true

Average Hardness of Inverting RSA vs. Worst Case Hardness

Claim: Let $S=U_{n \in N}$, $S_n$, $S_n = \{(N,e) | N \in H_n, e \in \mathbb{Z}_{\phi(n)}\}$.
If $\exists$ PPT $B$ s.t. for all $(N,e) \in S_n$ for $n \Rightarrow \infty$,

$$\text{prob}_x(B(N,e,\text{RSA}_{N,e}(x)) = x) > \text{non-neg}(n)$$
then $\exists$ PPT algorithm $A$ to invert $\text{RSA}_{N,e}(x)$ $\forall x \forall (N,e) \in S$.

Proof: Given $y = x^e \mod N$, choose random $r$ in $\mathbb{Z}_N^*$ and map $y$ to $z=y^r \mod N$. Now, run $B(z)$. If successful, i.e $B(z) = x^r \mod N$, output $x = B(z)/r \mod N$, else choose another $r$.
In expected $1/\varepsilon$ trials will be successful.

RSA and Factoring Integers

- **Fact 1:** Given $N$, $e$, $p$, and $q$, it's easy to compute $\phi(N)$ and $d=e^{-1} \mod \phi(N)$.
- **Fact 2:** Given only $N$, $e$, computing $\phi(N)$ is as hard as factoring $N$.
- **Fact 3:** Given only $N$, $e$, computing $d$ is as hard as factoring $N$.

Conclusions:
- If can factor, can invert RSA
- But, is Inverting (breaking) RSA as hard as factoring? MAJOR OPEN PROBLEM
**Trapdoor Function Equivalent to Factoring**

[Rabin]

Let \( N = pq \), \( p, q \) primes and \( e = 2 \)!!

Define \( \text{Rabin}_N(x) = x^2 \mod N \)

\( \text{Rabin}_N(x) : Z_N^* \rightarrow \mathbb{QR}_N \) where \( \mathbb{QR}_N = \text{set of Squares mod } N \)

Observation:
- \( \text{Rabin}_N \) is not a permutation but a 4-1 function

Claims:
- square roots of \( x^2 \mod N \) are \( x, N-x, y, N-y \)
- If factorization of \( N \) is known, there exists a PPT algorithm for computing square roots mod \( N \)
- If only \( N \) is known, computing square roots mod \( N \) is provably as hard as factoring.

---

**Chinese Remainder Theorem (CRT)**

Let \( p_1 \ldots p_t \) s.t. \( \gcd(p_i, p_j) = 1 \)

and \( y_1 \ldots y_t \) be integers. Then there is a unique solution \( u \mod N = \prod p_i \) s.t.

\[
\begin{align*}
    u &\equiv y_1 \mod p_1 \\
    u &\equiv y_2 \mod p_2 \\
    \vdots \\
    u &\equiv y_t \mod p_t
\end{align*}
\]

and \( u \) can be easily computable.

---

**Example CRT**

\( p = 3, \ q = 7, \ N = 21, \ y_1 = 2, \ y_2 = 5 \) \( u = ? \)

**Def:** CRT coefficients of \( p \) are \( c \) and \( d \) s.t.

\[
\begin{align*}
    c &\equiv 1 \mod p \text{ and } 0 \mod q, \ e.g. \ c = 7, \text{ as } 7 \mod 3 = 1, 7 \mod 7 = 0 \\
    d &\equiv 1 \mod q \text{ and } 0 \mod p, \ e.g. \ d = 15, \text{ as } 15 \mod 3 = 0, \ 15 \mod 7 = 1
\end{align*}
\]

Given \( c \) and \( d \), compute \( u \) as follows

\[
    u = y_1 \cdot c + y_2 \cdot d = 2 \cdot 7 + 5 \cdot 15 = 89 \mod 21 = 5 \mod 21
\]
**Computing Square Roots mod Composites As Hard As Factoring**

**Theorem:**
If \( \exists \) PPT algorithm \( A \) s.t. \( A(N,x^2 \mod N)=x \), then \( \exists \) PPT algorithm to factor \( N \).

**Proof:** On input \( N \), choose a random \( r \) in \( \mathbb{Z}_N^* \).
Compute \( x=A(N,r^2 \mod N) \).
Now \( x^2 = r^2 \mod N \), and
\( x \neq +r \mod N \) (with prob 1/2).
\( x^2 = r^2 \mod N \rightarrow (x-r)(x+r) = 0 \mod N \)
either \( p|(x-r) \) or \( q|(x-r) \).
gcd\( (x-r,N) = p \) or \( q \) \( \quad \text{QED} \)

---

**Computing Square Roots mod Composites on Average As Hard As Factoring**

**Theorem:** If \( \exists \) PPT algorithm \( A \) s.t. \( A(N,x^2 \mod N)=x \) with non-neg (n) probability, then \( \exists \) PPT algorithm to factor \( N \).

**Proof:** On input \( N \), choose a random \( r \) in \( \mathbb{Z}_N^* \).
Compute \( x=A(N,r^2 \mod N) \).
Now \( x^2 = r^2 \mod N \), and
\( x \neq +r \mod N \) (with prob 1/2).
\( x^2 = r^2 \mod N \rightarrow (x-r)(x+r) = 0 \mod N \)
So, gcd\( (x-r,N) = p \) or \( q \) \( \quad \text{QED} \)

---

**Computing Square Root mod \( N \), Given Factorization of \( N \)**

Let \( N=pq \), and \( z=x^2 \mod N \).

\[ \text{SQRT}_N(p,q,z) : \]
- Let \( z_1 = z \mod p \). Compute \( x_1 = \text{SQRT}_p(z_1) \)
- Let \( z_2 = z \mod q \). Compute \( x_2 = \text{SQRT}_q(z_2) \)
- Compute \( c_1 = 1 \mod p \) and \( 0 \mod q \) (CRT)
- Compute \( c_2 = 1 \mod q \) and \( 0 \mod p \) (CRT)
- Output \( x = x_1 c_1 + x_2 c_2 \)

Note: \( (x_1 c_1 + x_2 c_2)^2 = z_1 \mod p \)
\( (x_1 c_1 + x_2 c_2)^2 = z_2 \mod q \)
By CRT, there exists unique \( z \) s.t.
\( z = z_1 \mod p \) and \( z = z_1 \mod q \)

---

**Rabin Collection of Trapdoor Functions**

Define \( \text{Rabin} = \{ \text{Rabin}_N \mid \text{where } N=pq, p,q \text{ primes} \} \)

**Theorem:** Under Factoring-assumption,
Rabin is a collection of trapdoor functions.

**Generation:** Choose \( n \)-bit \( p,q \) and test
for primality. If primes set \( N=pq \), trapdoor\( _N = \{ p,q \} \)

**Evaluation:** Computing \( \text{Rabin}_N(x) \) takes \( O(n^2) \) time

**Hard to Invert:** by Factoring Assumption

**Trapdoorness:** Given \( N,p \) and \( q \) can compute square roots mod \( N \) in \( O(n^3) \).
Let $N=pq$, $p,q$ primes s.t. $p=q=3 \mod 4$.
Define $BW_N: \mathbb{QR}_N \rightarrow \mathbb{QR}_N$
as $BW_N(x) = x^2 \mod N$

**Claims:**

- When $p=q=3 \mod 4$, then each square has a unique square root which itself is a square.
- $BW_N$ is a permutation over the squares mod $N$.
- If factorization of $N$ is known, there exists a PPT algorithm for inverting $BW_N$.
- If factorization is not known, computing square roots mod $N$ is provably as hard as factoring $N$. 