Asymmetric encryption

CS642: Computer Security

Professor Ristenpart

http://www.cs.wisc.edu/~rist/
rist at cs dot wisc dot edu
Announcements

• Please turn in project proposals today
• Any time is fine
• Email them directly to me
  – subject line should include CS642 project proposal
  – I will reply to your email so you know I got it
  – (If you don’t hear back by Thursday let me know)
Asymmetric encryption

Basic setting

The RSA algorithm

PKCS #1 encryption

Digital signing & public-key infrastructure

Hybrid encryption
TLS handshake for RSA transport

Bank customer

Pick random Nc

Check CERT using CA public verification key

Pick random PMS

C <- E(pk,PMS)

Bracket notation means contents encrypted

ClientHello, MaxVer, Nc, Ciphers/CompMethods

ServerHello, Ver, Ns, SessionID, Cipher/CompMethod

CERT = (pk of bank, signature over it)

CERT <- D(sk,C)

Pick random Ns

ChangeCipherSpec,
{ Finished, PRF(MS, “Client finished” || H(transcript)) }

ChangeCipherSpec,
{ Finished, PRF(MS, “Server finished” || H(transcript’)) }

MS <- PRF(PS, “master secret” || Nc || Ns )
Trapdoor functions help us build PKE

$X$ is easy given $pk$

$f_{pk}(X)$ is hard given $pk$

$X$ is easy given $sk$

$X$ is hard given $pk$
The RSA trapdoor function

- Rivest, Shamir, Adleman 1978
- Garnered them a Turing award
RSA math

p and q be large prime numbers
N = pq
N is called the modulus

p = 7, q = 13, gives N = 91
p = 17, q = 53, gives N = 901
RSA math

p and q be large prime numbers
N = pq
N is called the modulus

\[ \mathbb{Z}_N = \{0,1,2,3,\ldots, N-1\} \]

\[ \mathbb{Z}_N^* = \{ i \mid \gcd(i,N) = 1 \} \]

The size of a set S is denoted by \(|S|\)

\[ \gcd(X,Y) = 1 \text{ if greatest common divisor of } X,Y \text{ is } 1 \]
RSA math

\[ Z_N^* = \{ i \mid \gcd(i, N) = 1 \} \]

\[
\begin{align*}
N = 13 & \quad Z_{13}^* = \{ 1,2,3,4,5,6,7,8,9,10,11,12 \} \\
N = 15 & \quad Z_{15}^* = \{ 1,2,4,7,8,11,13,14 \}
\end{align*}
\]

Def. \( \phi(N) = |Z_N^*| \) (This is Euler’s totient function)

\[
\begin{align*}
\phi(13) &= 12 \\
\phi(15) &= 8 \\
Z_{\phi(15)}^* &= Z_8^* = \{ 1,3,5,7 \}
\end{align*}
\]
RSA math

\[ Z_N^* = \{ i \mid \text{gcd}(i,N) = 1 \} \]

\( Z_N^* \) is a group under **modular multiplication**

**Fact.** For any \( a,N \) with \( N > 0 \), there exists unique \( q,r \) such that

\[ a = Nq + r \quad \text{and} \quad 0 \leq r < N \]

17 mod 15 = 2

105 mod 15 = 0

**Def.** \( a \mod N = r \in Z_N \)

**Def.** \( a \equiv b \pmod{N} \) iff \( (a \mod N) = (b \mod N) \)
RSA math

\[ \mathbb{Z}_N^* = \{ i \mid \gcd(i,N) = 1 \} \]

\( \mathbb{Z}_N^* \) is a group under modular multiplication

\[ \mathbb{Z}_{15}^* = \{ 1,2,4,7,8,11,13,14 \} \]

\[ 2 \cdot 7 \equiv 14 \pmod{15} \]

\[ 4 \cdot 8 \equiv 2 \pmod{15} \]

Closure: for any \( a,b \in \mathbb{Z}_N \), \( a \cdot b \mod N \in \mathbb{Z}_N \)

Def. \[ a^i \mod N = a \cdot a \cdot a \cdot \ldots \cdot a \mod N \]

\[ i \text{ times} \]
\[ \mathbb{Z}_N^* = \{ i \mid \gcd(i,N) = 1 \} \]

Claim: Suppose \( e, d \in \mathbb{Z}_{\phi(N)}^* \) satisfying \( ed \mod \phi(N) = 1 \),
then for any \( x \in \mathbb{Z}_N^* \) we have that
\[ (x^e)^d \mod N = x \]

First equality is by Euler’s Theorem
\[
\begin{align*}
(x^e)^d \mod N &= x^{(ed \mod \phi(N))} \mod N \\
&= x^1 \mod N \\
&= x \mod N
\end{align*}
\]
RSA math

\[ Z_N^* = \{ i \mid \gcd(i,N) = 1 \} \]

Claim: Suppose \( e, d \in Z_{\phi(N)}^* \) satisfying \( ed \mod \phi(N) = 1 \) then for any \( x \in Z_N^* \) we have that

\[ (x^e)^d \mod N = x \]

\[ Z_{15}^* = \{ 1, 2, 4, 7, 8, 11, 13, 14 \} \quad Z_{\phi(15)}^* = \{ 1, 3, 5, 7 \} \]

\( e = 3, \ d = 3 \) gives \( ed \mod 8 = 1 \)

<table>
<thead>
<tr>
<th>x</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>7</th>
<th>8</th>
<th>11</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x^3 \mod 15 )</td>
<td>1</td>
<td>8</td>
<td>4</td>
<td>13</td>
<td>2</td>
<td>11</td>
<td>7</td>
<td>14</td>
</tr>
<tr>
<td>( y^3 \mod 15 )</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>7</td>
<td>8</td>
<td>11</td>
<td>13</td>
<td>14</td>
</tr>
</tbody>
</table>
RSA admits a trapdoor permutation

\[ pk = (N, e) \quad sk = (N, d) \quad \text{with} \quad ed \mod \phi(N) = 1 \]

\[ f_{N,e}(x) = x^e \mod N \]
\[ g_{N,d}(y) = y^d \mod N \]
RSA admits a trapdoor permutation

\[ pk = (N,e) \quad sk = (N,d) \quad \text{with} \quad ed \mod \phi(N) = 1 \]

\[ f_{N,e}(x) = x^e \mod N \quad g_{N,d}(y) = y^d \mod N \]

But how do we find suitable \( N,e,d \)?

If \( p,q \) distinct primes and \( N = pq \) then \( \phi(N) = (p-1)(q-1) \)

Why?

\[ \phi(N) = |\{1,...,N-1\}| - |\{ip : 1 \leq i \leq q-1\}| - |\{iq : 1 \leq i \leq p-1\}| = N-1 - (q-1) - (p-1) = pq - p - q + 1 = (p-1)(q-1) \]
RSA admits a trapdoor permutation

\[ pk = (N,e) \quad sk = (N,d) \quad \text{with} \quad ed \mod \phi(N) = 1 \]

\[ f_{N,e}(x) = x^e \mod N \quad g_{N,d}(y) = y^d \mod N \]

But how do we find suitable \( N,e,d \)?

If \( p,q \) distinct primes and \( N = pq \) then \( \phi(N) = (p-1)(q-1) \)

Given \( \phi(N) \), choose \( e \in \mathbb{Z}_{\phi(15)} \) and calculate \( d = e^{-1} \mod \phi(N) \)
Public-key encryption

Correctness: \( D(sk, E(pk, M, R)) = M \) with probability 1 over randomness used
PKCS #1 RSA encryption

Kg outputs \((N,e),(N,d)\) where \(|N|_8 = n\)

Let \(B = \{0,1\}^8 / \{00\}\) be set of all bytes except 00

Want to encrypt messages of length \(|M|_8 = m\)

**Enc**((N,e), M, R)

\[
\text{pad} = \text{first } n - m - 2 \text{ bytes from } R \text{ that are in } B
\]

\[
X = 00 || 02 || \text{pad} || 00 || M
\]

Return \(X^e \text{ mod } N\)

**Dec**((N,d), C)

\[
X = C^d \text{ mod } N \ ; \ aa || bb || w = X
\]

If (aa \(\neq\) 00) or (bb \(\neq\) 02) or (00 \(\notin\) w)

Return error

\[
\text{pad} || 00 || M = w
\]

Return M
Hybrid encryption

Kg outputs (pk, sk)

\[\text{Enc}(pk, M, R)\]
\[K \| |R1| |R2 = R\]
\[C1 = \text{Enc}(pk, K, R1)\]
\[C2 = \text{Enc}(K, M, R2)\]
Return \((C1, C2)\)

\[\text{Dec}(sk, (C1, C2))\]
\[K = \text{Dec}(sk, C1)\]
\[M = \text{Dec}(K, C2)\]
Return M
TLS handshake for RSA transport

Bank customer

Pick random Nc

Check CERT using CA public verification key

Pick random PMS

C <- E(pk, PMS)

Bracket notation means contents encrypted

ClientHello, MaxVer, Nc, Ciphers/CompMethods

ServerHello, Ver, Ns, SessionID, Cipher/CompMethod

CERT = (pk of bank, signature over it)

C

ChangeCipherSpec,
{ Finished, PRF(MS, “Client finished” || H(transcript)) }

ChangeCipherSpec,
{ Finished, PRF(MS, “Server finished” || H(transcript’)) }

MS <- PRF(PS, “master secret” || Nc || Ns )

PMS <- D(sk, C)

Bank
Security of RSA PKCS#1

• Passive adversary sees \((N,e), C\)
• Attacker would like to invert \(C\)
• Possible attacks?
Inverting RSA: given $N, e, y$ find $x$ such that $x^e \equiv y \pmod{N}$

- EASY because $f^{-1}(y) = y^d \pmod{N}$
- EASY because $d = e^{-1} \pmod{\varphi(N)}$
- EASY because $\varphi(N) = (p - 1)(q - 1)$

Learning $p, q$ from $N$ is the factoring problem

We don’t know if inverse is true, whether inverting RSA implies ability to factor
Factoring composites

• What is \( p, q \) for \( N = 901 \)?

\[
\text{Factor}(N):
\text{for } i = 2, \ldots, \sqrt{N} \text{ do }
\begin{align*}
\text{if } N \mod i &= 0 \text{ then } \\
p &= i \\
q &= N / p \\
\text{Return } (p, q)
\end{align*}
\]

Woops… we can always factor

But not always efficiently:
Run time is \( \sqrt{N} \)

\[
O(\sqrt{N}) = O(e^{0.5 \ln(N)})
\]
# Factoring composites

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time to factor N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve</td>
<td>$O(e^{0.5 \ln(N)})$</td>
</tr>
<tr>
<td>Quadratic sieve (QS)</td>
<td>$O(e^c)$</td>
</tr>
<tr>
<td></td>
<td>$c = d \cdot (\ln N)^{1/2} \cdot (\ln \ln N)^{1/2}$</td>
</tr>
<tr>
<td>Number Field Sieve (NFS)</td>
<td>$O(e^c)$</td>
</tr>
<tr>
<td></td>
<td>$c = 1.92 \cdot (\ln N)^{1/3} \cdot (\ln \ln N)^{2/3}$</td>
</tr>
</tbody>
</table>
## Factoring records

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Year</th>
<th>Algorithm</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSA-400</td>
<td>1993</td>
<td>QS</td>
<td>830 MIPS years</td>
</tr>
<tr>
<td>RSA-478</td>
<td>1994</td>
<td>QS</td>
<td>5000 MIPS years</td>
</tr>
<tr>
<td>RSA-515</td>
<td>1999</td>
<td>NFS</td>
<td>8000 MIPS years</td>
</tr>
<tr>
<td>RSA-768</td>
<td>2009</td>
<td>NFS</td>
<td>~2.5 years</td>
</tr>
</tbody>
</table>

RSA-x is an RSA challenge modulus of size about x bits
Security of RSA PKCS#1

• Passive adversary sees (N,e),C
• Attacker would like to invert C
• Possible attacks?
  – Pick |N| > 1024 and factoring will fail
  – Active attacks?
Bleichenbacher attack

We can take a target \( C \) and decrypt it using a sequence of chosen ciphertexts \( C_1, \ldots, C_q \) where \( q \sim 1 \text{ million} \)

I’ve just learned some information about \( C_1^d \mod N \)

\[
\text{Dec}((N,d), C) = X = C^d \mod N\]

\[
\text{If } (aa \neq 00) \text{ or } (bb \neq 02) \text{ or } (00 \ w) \]

Return error

\[
pad \ || \ 00 \ || \ M = w\]

Return M

**C_1**

padding error?

**C_2**

padding error?

...
Response to this attack

• Ad-hoc fix: Don’t leak whether padding was wrong or not
  – This is harder than it looks (timing attacks)

• Better:
  – use chosen-ciphertext secure encryption
  – OAEP is common choice
Security of RSA PKCS#1

• Passive adversary sees \((N,e),C\)
• Attacker would like to invert \(C\)
• Possible attacks?
  – Pick \(|N| > 1024\) and factoring will fail
  – Active attacks?
    • Some implementations seem ok
  – Man-in-the-middle: replace \((N,e)\) with our own key
TLS handshake for RSA transport

Bank customer

Bank

Pick random Nc

Check CERT using CA public verification key

Pick random PMS

C <- E(pk, PMS)

Bracket notation means contents encrypted

ClientHello, MaxVer, Nc, Ciphers/CompMethods

ServerHello, Ver, Ns, SessionID, Cipher/CompMethod

CERT = (pk of bank, signature over it)

C

ChangeCipherSpec,
{ Finished, PRF(MS, “Client finished” || H(transcript)) }

ChangeCipherSpec,
{ Finished, PRF(MS, “Server finished” || H(transcript’)) }

MS <- PRF(PS, “master secret” || Nc || Ns )

Pick random Ns

PMS <- D(sk, C)
Digital signatures

Anyone with public key can verify a signature
Only holder of secret key should be able to generate a signature
Full Domain Hash RSA

Kg outputs pk = (N,e) , sk = (N,d)

H is a hash function

**Sign**((N,d), M )

X = 00 || H(1||M) || ... || H(k||M)

S = X^d mod N

Return S

**Ver**((N,e), M, S )

X = S^e mod N

X' = 00 || H(1||M) || ... || H(k||M)

If X = X' then

    Return 1

Return 0
Certificate Authorities and Public-key Infrastructure

\[(pk, sk)\]

Give me a certificate for \( pk' \), please

\( M = (pk', data) \)

\( S = \text{Sign}(sk, M) \)

\( S \)

http://amazon.com

\( pk' \), data, \( S \)

If \( \text{Ver}(pk, M, S) \) then trust \( pk' \)

This prevents man-in-the-middle (MitM) attacks