Public-Key Cryptography

plaintext → encryption → ciphertext → decryption → plaintext

public key

private key
Cryptanalysis of Public-Key Cryptography

Popular Public-Key Cryptography

- RSA
- (EC)DSA / ElGamal

Mathematical problem

- Integer factorization
  
  \[ n = p \times q \]
  
  Find \( p \) or \( q \), given \( n \)

- (EC) Discrete Logarithm
  
  \[ g = b^k \in G \]
  
  Find \( \log_b(g) = k \), given \( b, g \) and \( G \)
Sanity check I

There have been many sanity checks of certificates and PKI

- **Analyzing RSA Standards**
  

- **Analyzing X.509**
  


- **Debian OpenSSL vulnerability**
  
There have been many sanity checks of certificates and PKI

- **Analyzing RSA Standards**
  

  The entropy of the output distribution [of standardized RSA key generation] is always almost maximal, ... and the outputs are hard to factor if factoring in general is hard.

- **Analyzing X.509**
  


- **Debian OpenSSL vulnerability**
  
Sanity check II

We look at things from a computational crypto point of view...

Our work

At the same time...
We look at things from a computational crypto point of view...

Our work

At the same time...
Aug. '10: Download all publicly-visible SSL certificates on the IPv4 Internet

- 6,185,372 X.509 certificates
- 5,481,332 PGP keys
- 11,666,704 public keys
Aug. ’10: Download all publicly-visible SSL certificates on the IPv4 Internet

- 6 185 372 X.509 certificates
- 5 481 332 PGP keys
- 11 666 704 public keys

X.509

- 6 185 230 RSA
- 141 DSA
- 1 ECDSA

47.6%: expiration date > 2011
77.7%: use ≥ SHA-1
33.4%: satisfy both requirements

PGP keys

- 2 546 752 ElGamal
- 2 536 959 DSA
- 397 621 RSA
RSA is the most widely used approach to achieve public-key cryptography.

**Keys**
- Secret information: exponent $d$, prime factors $p$, $q$
- Public information: modulus $n$ and the exponent $e$

\[
 n = p \times q \text{ with } p \approx q
\]
\[
 \gcd(e, (p - 1)(q - 1)) = 1 \text{ and } d \equiv e^{-1} \mod (p - 1)(q - 1)
\]

- Encryption: $c = m^e \mod n$
- Decryption: $m = c^d \mod n$
Check the public exponent

<table>
<thead>
<tr>
<th>X.509</th>
<th>%</th>
<th>PGP</th>
<th>%</th>
<th>Combined</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>e</td>
<td>65537</td>
<td>98.4921</td>
<td>e</td>
<td>65537</td>
<td>48.8501</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>0.7633</td>
<td>17</td>
<td>17</td>
<td>3.1035</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.3772</td>
<td>41</td>
<td>41</td>
<td>0.4574</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>0.1410</td>
<td>19</td>
<td>35</td>
<td>0.1339</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.1176</td>
<td>257</td>
<td>19</td>
<td>0.1506</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.0631</td>
<td>23</td>
<td>35</td>
<td>0.1339</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>0.0220</td>
<td>11</td>
<td>5</td>
<td>0.1111</td>
</tr>
<tr>
<td></td>
<td>47</td>
<td>0.0101</td>
<td>3</td>
<td>7</td>
<td>0.0596</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>0.0042</td>
<td>21</td>
<td>11</td>
<td>0.0313</td>
</tr>
<tr>
<td>65535</td>
<td>0.0011</td>
<td>$2^{127} + 3$</td>
<td>0.0248</td>
<td>257</td>
<td>0.0241</td>
</tr>
<tr>
<td>other</td>
<td>0.0083</td>
<td>other</td>
<td>0.6807</td>
<td>other</td>
<td>0.0774</td>
</tr>
</tbody>
</table>

Note: 8 times $e = 1$ was used!
Check moduli sizes

<table>
<thead>
<tr>
<th>%</th>
<th>bits</th>
<th>%</th>
<th>bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>384</td>
<td>0.04</td>
<td>3072</td>
</tr>
<tr>
<td>1.6</td>
<td>512</td>
<td>1.5</td>
<td>4096</td>
</tr>
<tr>
<td>0.8</td>
<td>768</td>
<td>0.01</td>
<td>8192</td>
</tr>
<tr>
<td>73.9</td>
<td>1024</td>
<td>0.003</td>
<td>16384</td>
</tr>
<tr>
<td>21.7</td>
<td>2048</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Primality and small factors
- 2 moduli are prime
- 171 have a factor $< 2^{24}$
- (68 are even)

These RSA keys were discarded.

Debian moduli
- 30,097 (21,459 distinct)
  blacklisted keys
Identical keys $1 \implies n_1 = n_2$

**Implications**

User 1 can decrypt all messages from user 2 (and vice versa)

- Most of the time harmless: renewal of key
- Possible explanation: Low-entropy when generating keys

```plaintext
seed(initial_randomness);
do { p=random(); } while( isprime(p) != true );
do { q=random(); } while( isprime(q) != true );
n = p*q;
```
Identical keys II

Cluster: certs/keys with the same modulus

Note: One cluster of size 16489  
4.3% of the RSA moduli are shared
Moduli with shared factors

\[ K_1 : a \times b \quad K_2 : c \times d \]

- User 1 and user 2 have secure keys
Moduli with shared factors

\[ K_1 : a \times b \quad K_2 : c \times d \quad K_3 : b \times c \]

- User 1 and user 3 share a factor and User 2 and 3 share a factor
Moduli with shared factors

\[
\begin{align*}
K_1 & : a \times b \\
K_2 & : c \times d \\
K_3 & : b \times c
\end{align*}
\]

- User 1 and user 3 share a factor and User 2 and 3 share a factor
- Greatest common divisor: everyone can break these keys!
Moduli with shared factors

Given two RSA moduli \( n_1 \) and \( n_2 \),

\[ n_1 \neq n_2 \land \gcd(n_1, n_2) \neq 1, \]

results in a complete loss of security for these moduli.

Checking all RSA keys for shared factors

- Straight-forward approach: \( \approx \) ten core-years
- Smarter approach: \( \approx \) ten core-hours
Moduli with shared factors

Given two RSA moduli \( n_1 \) and \( n_2 \),

\[
   n_1 \neq n_2 \land gcd(n_1, n_2) \neq 1,
\]

results in a complete loss of security for these moduli.

Checking all RSA keys for shared factors

- Straight-forward approach: \( \approx \) ten core-years
- Smarter approach: \( \approx \) ten core-hours

5321 X.509 certificates, 4627 RSA moduli
RSA keys VI

Affected keys

We found 14,901 distinct primes factoring 12,934 distinct moduli. 21,419 X.509 certs and PGP keys are affected. None of these are blacklisted.

### Primes

<table>
<thead>
<tr>
<th>#</th>
<th>bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>307</td>
<td>256</td>
</tr>
<tr>
<td>2</td>
<td>257</td>
</tr>
<tr>
<td>14,592</td>
<td>512</td>
</tr>
</tbody>
</table>

### Moduli

<table>
<thead>
<tr>
<th>#</th>
<th>bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>214</td>
<td>512</td>
</tr>
<tr>
<td>12,720</td>
<td>1024</td>
</tr>
</tbody>
</table>

- **3201** 1024-bit RSA moduli occur in 5,250 certificates which are not-expired and use SHA-1.
RSA keys VII - Discussion

RSA requires generating two random prime numbers
These primes must not be selected by anyone else before

NIST recommends: size(random seed) = 2 × size(security level)

Possible explanations:

- Poor random initial seeding → duplicate keys
- Using local entropy after each guess
  - “poor initial guess” $p_1$, with prob $1/\log(p_1)$ this is prime
  - next guesses use the local entropy

seed(initial_randomness);
do { p=random(); } while( isprime(p) != true );
do { q=random(); } while( isprime(q) != true );
n = p*q;
February 2012, new scan by EFF

7.2M distinct X.509 certs (up from 6.2M)
February 2012, new scan by EFF

7.2M distinct X.509 certs (up from 6.2M)

**RSA-1024**
- 4.7M → 3.7M keys
- > 5000 affected keys are no longer present
- 13,019 new keys affected

New: 10 RSA-2048 keys are affected, two have not expired and use SHA-1
Multi-secret systems (RSA) vs. single-secret systems (ElGamal, (EC)DSA)
Conclusions

Multi-secret systems (RSA) vs. single-secret systems (ElGamal, (EC)DSA)

Possible remedy

<table>
<thead>
<tr>
<th>Moduli $pq$ for $k$-bit primes $p$ chosen such that $q = \left[ \frac{2^{2k-1} + p - (2^{2k-1} \mod p)}{p} \right]$ is prime</th>
</tr>
</thead>
</table>

Conclusions

Multi-secret systems (RSA) vs. single-secret systems (ElGamal, (EC)DSA)

Possible remedy

Moduli $pq$ for $k$-bit primes $p$ chosen such that

$$q = \left[ \frac{2^{2k-1} + p - (2^{2k-1} \mod p)}{p} \right]$$

is prime


Misinterpretations in the Media

- “This is simply the Debian PRNG bug”
  All our results exclude the blacklisted Debian keys.
- “RSA is insecure”
  When properly generating random primes then RSA is still secure.
- “Only embedded devices are affected”
  We have multiple examples of affected keys between users.