The Million-Key Question

Investigating the Origins of RSA Public Keys

Petr Švenda, Matúš Nemec, Peter Sekan, Rudolf Kvašňovský, David Formánek, David Komárek and Vashek Matyáš

svenda@fi.muni.cz

Faculty of Informatics, Masaryk University, Czech Republic
The Million-Key Question, USENIX Security 2016.08.12

RSA public key

\[ N = 9782D7123C330444C88E279BF321EE84AC39524F1D84026327B04F32E1E930FC81588010178DC75FCBF8258A068071317245D08817988813C4173495A922A41DA429A964F738020076EFFE7ED5811088873C6E58EEF1CDC9005966697BE72368B51A821FC699E9C3FD66B377E2DF2485DC401DD99CC125890E5D969A6AC8B \]
\[ e = 10001 \]
7 implementation choices observable in public keys
(biased bits of public modulus, “mask”)
Heatmap of primes’ most significant byte

\[ P \times Q = N \]
Factors of $\frac{P-1}{Q-1}$ (and its impact on modulus N)

- For RSA512b, length of prime is 256bits $\Rightarrow \frac{P-1}{Q-1}$ can be factorized
- We factorized 10k primes for every source with YAFU and…
- Small factors avoided
  - Significant bias on lower bits of N
  - Used by I. Mironov (OpenSSL)
- FIPS primes (specific range)
  - Not observable in modulus N

MIRONOV, I. Factoring RSA Moduli II.
https://windowsontheory.org/2012/05/17/factoring-rsa-modulipart-ii/
7 implementation choices observable in public key

1. Direct manipulation of the primes’ highest bits
2. Avoidance of small factors in $P-1$ and $Q-1$
3. Requirement for moduli to be Blum integers
4. Restriction of the primes’ bit length
5. Specific method to construct strong or provable primes
6. Use of another non-traditional algorithm – functionally unknown, but statistically observable
7. Type of action after candidate prime rejection

Significance
Building classification matrix

Harvest keys from known sources (learning set)

- Apply mask to learning set
- Count mask frequency
- Group sources with very similar frequencies
- Normalize mask vectors of groups

Identification of biased modulus bits (mask, 9bits)

- 2nd-7th MSb | 2nd LSB | modulus mod 3 | len(modulus) mod 2

Classification matrix

<table>
<thead>
<tr>
<th>Mask value</th>
<th>Group I</th>
<th>Group II</th>
<th>...</th>
<th>Group XII</th>
<th>Group XIII</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000000000</td>
<td>0.124</td>
<td>0.347</td>
<td></td>
<td>0.105</td>
<td>0.012</td>
</tr>
<tr>
<td>0000000001</td>
<td>0.004</td>
<td>0.038</td>
<td></td>
<td>0.236</td>
<td>0.454</td>
</tr>
<tr>
<td>0000000011</td>
<td>0.046</td>
<td>0.002</td>
<td></td>
<td>0.447</td>
<td>0.112</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111111110</td>
<td>0.394</td>
<td>0.044</td>
<td>0.320</td>
<td>0.002</td>
<td></td>
</tr>
<tr>
<td>1111111111</td>
<td>0.046</td>
<td>0.347</td>
<td></td>
<td>0.015</td>
<td>0.312</td>
</tr>
</tbody>
</table>

E.g.,

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2nd MSb</td>
<td>2nd LSB</td>
<td>modulus mod 3</td>
<td>len(modulus) mod 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>101101110</td>
<td>111111121</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Tree splits can be attributed to particular implementation choice(s)

13 classification groups

38 different sources
Input key

-----BEGIN CERTIFICATE-----
MIIG9zCCBd+gAwIBAgIIJOR2wFUwc20wDQYJKoZIhvcNAQELBQAwSTELMAkGA1UEBhMCVVMxJTAjBgNVBAMTHEdvb2dsZSBJbmMxJTA5BgNVBAoTCkdvb2dsZSBJbmMxKQYwOTI4MDgwMzA2Nz0xHhcNMTYwNzA2MDEwMzUwWhcNMTYwNzA2MDEwMzUwWQYDVQQIEwNDb29yZ2xlMjAwFw0xNjAyMDgyODE5NjQxX5cMGA1MwUwIwYDVQQDDAEzYXR0cmlnaHQxIjAwMDQwMTg4MDc4WjAXBgNVBAMTCkdvb2dsZSBJbmN0aW9uLFN1c3RMb2NzdGF0aW9uMRUwMDA5MDUwMDQ4WjASBgNVBAcTJnlQaNUhBIIB/nQsCBAoIBvAAMCQGCSqGSIb3DQEBAQUAA4GNADCBiQKBgQ08uaXejqTndPB+X1wtqWznW3T2B0OJLh5yse0iQSMZyKly9xQ17+KjQ83Mmu2K9h3x5qZrQI6rCr7w75AI6pemEXTx60d3z8u2LHnOcedlJn1DP6NKc60N1G2Jw2+WcM/6vDkmgW1K2vKuyvYzQ==
-----END CERTIFICATE-----

The Million-Key Question, USENIX Security 2016.08.12

Precomputed matrix

<table>
<thead>
<tr>
<th>Mask value</th>
<th>Group I</th>
<th>Group II</th>
<th>...</th>
<th>Group XII</th>
<th>Group XIII</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000000000</td>
<td>0.124</td>
<td>0.347</td>
<td></td>
<td>0.105</td>
<td>0.012</td>
</tr>
<tr>
<td>0000000001</td>
<td>0.004</td>
<td>0.038</td>
<td></td>
<td>0.236</td>
<td>0.454</td>
</tr>
<tr>
<td>000000011</td>
<td>0.046</td>
<td>0.002</td>
<td></td>
<td>0.447</td>
<td>0.112</td>
</tr>
<tr>
<td>111111110</td>
<td>0.394</td>
<td>0.044</td>
<td></td>
<td>0.320</td>
<td>0.002</td>
</tr>
<tr>
<td>111111111</td>
<td>0.046</td>
<td>0.347</td>
<td></td>
<td>0.015</td>
<td>0.312</td>
</tr>
</tbody>
</table>

Classification

44% OpenSSL’s group
11% POLAR SSL’s group
9% PGP®’s group...
We think that your separate key(s) were generated by (sorted from the most probable)

<table>
<thead>
<tr>
<th>Key identification (first few characters of in ascii armor/web domain):</th>
<th>Key length</th>
<th>Exponent</th>
</tr>
</thead>
<tbody>
<tr>
<td>MiGMA0GCSqGSib3DQEBAQUAA4GNaGjXpL5NYJ2qt6TG</td>
<td>1024</td>
<td>65537</td>
</tr>
<tr>
<td>imvJvM0q5GhZc7h6K18CRKX81LmFgATiKuJ2v7mB95Q</td>
<td>2046</td>
<td>65537</td>
</tr>
<tr>
<td>P1mZ7t4kwv7V+e+y=wiDAQAB</td>
<td>2046</td>
<td>65537</td>
</tr>
</tbody>
</table>

This key is hardest to attribute to a particular source library. Pick this one if you like to use the most anonymous key.

Result for same source (all inserted keys are assumed to be generated by the same source)

- You provided 3 keys. If these keys are all generated by the same source library then there is about 93% probability that correct source is identified within the first three most probable groups.

Please give us feedback: click on the source group by which your key(s) were generated and then submit feedback form.
Classification accuracy

Mask value

Probability

0 0.01 0.02 0.03

OpenSSL 1.0.2g
OpenJDK 1.8
Gemalto GXP E64

Try online app at http://crcs.cz/rsapp/
## Classification accuracy (test set, 10k keys/source)

<table>
<thead>
<tr>
<th># keys in batch</th>
<th>Top 1 match</th>
<th>Top 2 match</th>
<th>Top 3 match</th>
<th>Top 5 match</th>
<th>Top 10 match</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Group I</td>
<td>95.39%</td>
<td>98.42%</td>
<td>99.38%</td>
<td>99.75%</td>
<td></td>
</tr>
<tr>
<td>Group II</td>
<td>17.75%</td>
<td>32.50%</td>
<td>58.00%</td>
<td>69.50%</td>
<td></td>
</tr>
<tr>
<td>Group III</td>
<td>45.36%</td>
<td>72.28%</td>
<td>93.17%</td>
<td>98.55%</td>
<td></td>
</tr>
<tr>
<td>Group IV</td>
<td>90.14%</td>
<td>97.58%</td>
<td>99.80%</td>
<td>100.00%</td>
<td></td>
</tr>
<tr>
<td>Group V</td>
<td>63.38%</td>
<td>81.94%</td>
<td>97.50%</td>
<td>99.60%</td>
<td></td>
</tr>
<tr>
<td>Group VI</td>
<td>54.68%</td>
<td>69.22%</td>
<td>88.45%</td>
<td>94.60%</td>
<td></td>
</tr>
<tr>
<td>Group VII</td>
<td>7.58%</td>
<td>31.69%</td>
<td>64.21%</td>
<td>82.35%</td>
<td></td>
</tr>
<tr>
<td>Group VIII</td>
<td>15.65%</td>
<td>40.30%</td>
<td>68.46%</td>
<td>76.60%</td>
<td></td>
</tr>
<tr>
<td>Group IX</td>
<td>22.22%</td>
<td>45.12%</td>
<td>76.35%</td>
<td>83.00%</td>
<td></td>
</tr>
<tr>
<td>Group X</td>
<td>0.63%</td>
<td>6.33%</td>
<td>27.42%</td>
<td>42.74%</td>
<td></td>
</tr>
<tr>
<td>Group XI</td>
<td>11.77%</td>
<td>28.40%</td>
<td>55.56%</td>
<td>65.28%</td>
<td></td>
</tr>
<tr>
<td>Group XII</td>
<td>60.36%</td>
<td>79.56%</td>
<td>97.20%</td>
<td>99.40%</td>
<td></td>
</tr>
<tr>
<td>Group XIII</td>
<td>39.56%</td>
<td>70.32%</td>
<td>96.20%</td>
<td>99.70%</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>40.34%</td>
<td>57.90%</td>
<td>78.59%</td>
<td>85.47%</td>
<td></td>
</tr>
</tbody>
</table>

1 key

- Top 1: avg. **40.34%**, min. 0.63%, max. 95.36%
- Top 3: avg. **73.09%**, min. 39.32%, max. 98.41%

5 keys

- Top 1: avg. **78.59%**, min. 27.42%, max. 99.38%
- Top 3: avg. **97.48%**, min. 91.45%, max. 100.00%

10 keys

- Top 1: avg. **85.47%**, min. 42.74%, max. 100.00%
- Top 3: avg. **99.27%**, min. 95.00%, max. 100.00%
Sanity check with real world keys: IPv4 TLS dataset

- Datasets: IPv4 TLS scan (10M), PGP (1.4M), Cert. Transparency (13M)...
  - Problem: keys in these datasets are not annotated with source library
- Web servers market share => OpenSSL (~86%), Microsoft (~12%)

**Expected**
- Microsoft 12%
- OpenSSL 86%

**Classified (10-99 keys with same subject and issue date)**
- Microsoft 10.18%
- OpenSSL 82.84%
- Botan 5.61%
- Nettle 5.61%
- ARM mbed 1.09%
- OpenJDK 1.09%
Sanity check: keys which *cannot* be from OpenSSL

- Keys with mask value never generated by OpenSSL
- Advantage: all keys from dataset can be used

<table>
<thead>
<tr>
<th>Dataset</th>
<th>OpenSSL rare in PGP</th>
<th>Leaves ~81% for OpenSSL</th>
<th>OpenSSL is default client</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cert. Transparency</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PGP keyset</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLS IPv4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Let’s Encrypt</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>!OpenSSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cert. Transparency</td>
<td>11.80%</td>
</tr>
<tr>
<td>PGP keyset</td>
<td>47.35%</td>
</tr>
<tr>
<td>TLS IPv4</td>
<td>18.91%</td>
</tr>
<tr>
<td>Let’s Encrypt</td>
<td>1.83%</td>
</tr>
</tbody>
</table>
Impact (of the possibility) of public key classification

- Information leakage vulnerability
- Quick search for other keys from vulnerable library
- Linking related Tor hidden services operators
- Verify Crypto-as-a-Service use of secure hardware
How to defend against public key classification?

1. Developers of libraries - unify RSA key generation
   - Unlikely to happen soon, changes in critical part of code, legacy binaries…

2. Users of libraries – select from multiple keys
   - Generate multiple keys, pick the most anonymous one
   - Only about 5 keys required on average
   - [http://crcs.cz/rsapp](http://crcs.cz/rsapp)
What else?

• More in the paper and technical report [http://crcs.cz/rsa](http://crcs.cz/rsa)
  – Summary RSA generation techniques used by libraries and cards
  – Analysis of random data streams from smart cards (bias detected)
  – Systematic defect responsible for generating weak RSA keys
  – Time and power analysis of key generation on smart cards

• Download datasets and tools at [http://crcs.cz/rsa](http://crcs.cz/rsa)
Limitations of the current work

1. Lower accuracy with single key only (40% on avg.)
2. Can’t distinguish all libraries mutually (groups)
3. Some sources missing (HSMs…)
   - Will be misclassified at the moment
Conclusions

• RSA keypair generation observably bias public keys
  – Different libraries use different implementation choices
• Source library can be probabilistically estimated from RSA public key
  – Accuracy more than 85 % with 10 keys (>99 % within top three matches)
  – For some sources, even a single key is enough
• Information disclosure vulnerability
  – Forensics, de-anonymization, vulnerability scans, compliancy testing…
• Not easy to fix, will stay for longer time

Questions
