WHY JOHNNY THE DEVELOPER CAN’T WORK WITH PUBLIC KEY CERTIFICATES

Martin Ukrop
Usable security researcher
Centre for Research on Cryptography and Security, Masaryk University
Red Hat research cooperation

Vashek Matyas
Head of the security laboratory
Centre for Research on Cryptography and Security, Masaryk University
Simple and Solvable Error

File does not exist.

File /etc/ssl/certs/certificate1.pem does not exist.

Clicking OK will solve everything and create a secure solution with no bugs or vulnerabilities.
Is it really that bad?

- Let’s find out!
- What is the most used tool for generating certificates?

Q: Have you ever used OpenSSL?

[xukrop@styx ~]$ openssl version
OpenSSL 1.1.0g-fips 2 Nov 2017
[xukrop@styx ~]$ [xukrop@styx ~]$ openssl x509 -help
Usage: x509 [options]
Valid options are:
  -help             Display this summary
  -inform format    Input format - default PEM (one of DER, NET or PEM)
  -in infile       Input file - default stdin
  -outform format   Output format - default PEM (one of DER, NET or PEM)
  -out outfile      Output file - default stdout
  -keyform PEM|DER  Private key format - default PEM
  -passin val       Private key password/pass-phrase source
  -serial           Print serial number value
  -subject_hash     Print subject hash value
  -issuer_hash      Print issuer hash value
  -hash             Synonym for -subject_hash
  -subject          Print subject DN
  -issuer           Print issuer DN
  -email            Print email address(es)
  -startdate        Set notBefore field
Empirical experiment in usability

- 87 participants (👨) of DEVCONF.cz
  (developer conference by Red Hat Czech)

Task:

You are a software tester.
Use command line OpenSSL (v1.0.2g)

1. ... to issue a self-signed certificate.
2. ... to validate 4 given certificates.
Conference research booth at DEVCONF.cz
“Interacting with OpenSSL voluntarily?”

“Sorry, not even for research.”
“We all know OpenSSL sucks.”

“But finally, there is someone collecting empirical evidence.”
Task success

Task 1 (certificate generation)

- Succeeded: 46% (39/87) 🧑
- Did not succeed: 10% (8/87) 🧑
- Thought that they succeeded but did not: 44% (37/87) 🧑

Task 2 (certificate validation)

- Success: 19% (14/72) 🧑
- Incorrect: 10% (7/72) 🧑
- Ignoring OS cert store: 71% (51/72) 🧑
Created certificates

- 4096-bit key: 20% (16/82)
- 2048-bit key: 42% (34/82)
- 1024-bit key: 38% (30/82)

- Organization = “Internet Widgits Pty Ltd.”: **42% certificates** (27/65)
- cca **260 000** such certificates online (Censys.io dataset 2018-02-28)
Used resources

Task 1 (generating certificates)
- Online only: 32
- Online & manual page: 47
- Local manual page only: 6
- CLI help only: 2

Task 2 (validating certificates)
- Online only: 25
- Online & manual page: 33
- Local manual page only: 10
- CLI help only: 4
Stack Exchange: 73% people (58/79 )

How to create a self-signed certificate with openssl?

I'm adding https support to an embedded linux device. I have tried to generate a self-signed certificate with these steps:

```bash
openssl req -new > cert.csr
openssl rsa -in privkey.pem -out key.pem
openssl x509 -in cert.csr -out cert.pem -req -signkey key.pem -days 1001
cat key.pem>>cert.pem
```

This works, but I get some errors with, for example, google chrome:

This is probably not the site you are looking for!
The site's security certificate is not trusted!

Am I missing something? Is this the correct way to build a self-signed certificate?
WISC knowledge base: 40% people

Verifying that a Certificate is issued by a CA

How to use OpenSSL on the command line to verify that a certificate was issued by a specific CA, given that CA's certificate:

```
$ openssl verify -verbose -CAfile cacert.pem server.crt
server.crt: OK
```

If you get any other message, the certificate was not issued by that CA.

See Also:

- How to turn a X509 Certificate in to a Certificate Signing Request
- Verifying that a Private Key Matches a Certificate
Web pages used

- Describing security implications: 23% web pages
- Explaining individual parameters: 27% web pages
- Changes after copy-paste: 9% people (8/87 😊)

CRoCS

#RSAC
Q: How to display manual page for this?

**openssl verify** `-CAfile ca.pem cert.pem``

1. man openssl
   - ?
   - (53/53

2. man openssl verify
   - 28
   - (15/53

3. man openssl.verify
   - 2
   - (1/53

4. man openssl-verify
   - 8
   - (4/53

5. man verify
   - 100
   - (53/53

---

**crocs**

RSA Conference 2018
OpenSSL manual page

NAME
openssl - OpenSSL command line tool

SYNOPSIS
openssl command [ command_opts ] [ command_args ]
openssl list [ standard-commands | digest-commands | cipher-commands | cipher-algorithms | digest-algorithms | public-key-algorithms]
openssl no-XXX [ arbitrary options ]

DESCRIPTION
OpenSSL is a cryptography toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) network protocols and related cryptography standards required by them.

The openssl program is a command line tool for using the various cryptography functions of OpenSSL's crypto library from the shell. It can be used for
Has the world moved on?

Our work

- `man openssl verify` now works
- Fixed URLs for online documentation
- Research into better error messages

OpenSSL team

- High-level `help` command
- `-help` argument for each command
- Improved defaults (key lengths, ...)
Takeaways I.

OpenSSL usability is poor.

(But better than other tools.)

Improvement is possible!
People may not know they failed if the tool does not tell them.
Takeaways III.

Documentation (manuals, tutorials, Q&A forums, ...) matters a lot.

Stack Overflow is a seriously used resource.
What should you do next?

**USING security products?**
- Ask your developers what they find unusable.
- Investigate past vulnerabilities: Caused by tool unusability?
- Report usability issues back to developers.

**DEVELOPING security products?**
- In your project, strive for good “developer experience” (DX).
- Ask users on usability feedback.
- Organize a usability lab study to improve your product.
CARE FOR YOUR DEVELOPERS. 
DEVELOPER EXPERIENCE (DX) MATTERS.

Martin Ukrop
mukrop@mail.muni.cz

Centre for Research on Cryptography and Security
Masaryk University, Brno, Czech Republic
Improved Factorization of $N = p^r q^s$

Jean-Sébastien Coron,
Rina Zeitoun

Speaker: Mehdi Tibouchi
Agenda

1. State-of-the-art / Motivations
2. Reminder on Coppersmith / BDH Methods
3. [CFRZ16] Method to factorize $N = p^r q^s$
4. An Improvement to Factorize $N = p^r q^s$
1. State-of-the-art / Motivations

2. Reminder on Coppersmith / BDH Methods

3. [CFRZ16] Method to factorize $N = p^r q^s$

4. An Improvement to Factorize $N = p^r q^s$
### RSA Cryptosystem / Signature

#### RSA Key Generation
- Generate two large primes \( p \) and \( q \)
- Compute \( N = p \times q \)
- Select \( (e, d) \) such that \( ed \equiv 1 \mod \phi(N) \)

#### Encryption/Decryption Process

\[
C \equiv m^e \mod N \quad \rightarrow \quad m \equiv C^d \mod N
\]

Factorize \( N = p \times q \) \Rightarrow Break RSA
State-of-the-art: Modulus $N = p^r q$

Decryption with Modulus $N = p^r q$  [Takagi98]

- For equivalent security, use a smaller prime $p$
  - Decryption becomes faster

State-of-the-art: Modulus $N = p^r q$

Decryption with Modulus $N = p^r q$  [Takagi98]

- For equivalent security, use a smaller prime $p$
  
  Decryption becomes faster

Vulnerability of $N = p^r q$  [BDH99]

- **BDH method**: Factoring $N = p^r q$ for Large $r$

  Condition:  $r \simeq \log q$


What About Modulus $N = p^r q^s$ ?

Decryption with Modulus $N = p^r q^s$ [LKYL00]

- Decryption is even faster
  - For an 8192-bit $N = p^2 q^3$: decryption is 15 times faster

What About Modulus $N = p^r q^s$?

Decryption with Modulus $N = p^r q^s$ [LKYL00]

- Decryption is even faster
  - For an 8192-bit $N = p^2 q^3$: decryption is 15 times faster

Problem: Factorization of $N = p^r q^s$

- Factorization of $N = p^r q^s$ in polynomial time
  - Left as an open problem in [BDH99]

Polynomial time factorization of $N = p^r q^s$ with $r > s$ if:

$$r \simeq \log^3 \max(p, q)$$

Based on Lattice Attacks

CFRZ method

BDH method

Coppersmith method

[CFRZ16] Factoring $N = p^r q^s$ for Large $r$ and $s$. Coron, Faugère, Renault, Zeitoun, CT-RSA 2016.
Polynomial time factorization of $N = p^r q^s$ with $r > s$ if:

$$r \simeq \log^3 \max(p, q)$$

Based on Lattice Attacks

[CFRZ16] Factoring $N = p^r q^s$ for Large $r$ and $s$. Coron, Faugère, Renault, Zeitoun, CT-RSA 2016.
Polynomial time factorization of $N = p^r q^s$ with $r > s$ if:

$$r \simeq \log^3 \max(p, q)$$

Based on Lattice Attacks

[CFRZ16] Factoring $N = p^r q^s$ for Large $r$ and $s$. Coron, Faugère, Renault, Zeitoun, CT-RSA 2016.
Outline

1. State-of-the-art / Motivations

2. Reminder on Coppersmith / BDH Methods

3. [CFRZ16] Method to factorize $N = p^r q^s$

4. An Improvement to Factorize $N = p^r q^s$
## Coppersmith’s Theorem

### The Problem (Univariate Modular Case)

**Input:**
- A polynomial \( f(x) = x^\delta + a_{d-1}x^{\delta-1} + \cdots + a_1x + a_0 \)
- \( N \) an integer of unknown factorization

**Find:**
- All integers \( x_0 \) such that \( f(x_0) \equiv 0 \mod N \)
### The Problem (Univariate Modular Case)

- **Input:**
  - A polynomial \( f(x) = x^\delta + a_{d-1}x^{\delta-1} + \cdots + a_1x + a_0 \)
  - \( N \) an integer of unknown factorization

- **Find:**
  - All integers \( x_0 \) such that \( f(x_0) \equiv 0 \mod N \)

### Coppersmith’s Theorem for the Univariate Modular case

- The solutions \( x_0 \) can be found in polynomial time in \( \log(N) \) if
  \[
  |x_0| < N^{1/\delta}
  \]
BDH Method to factorize \( N = p^r q \)

Extension of Coppersmith method for unknown modulus

- Write \( p = K \cdot t + x \)
- Solve \( f(x) = (K \cdot t + x)^r \equiv 0 \pmod{p^r} \)

\[
\begin{align*}
\text{Exh. Search: } p^{1/2} & \quad (r = 1) \\
\text{Exh. Search: } p^{1/3} & \quad (r = 2) \\
\text{Exh. Search: } p^{1/4} & \quad (r = 3) \\
\cdots & \quad (r \approx \log q) \\
\end{align*}
\]
1 State-of-the-art / Motivations

2 Reminder on Coppersmith/BDH Methods

3 [CFRZ16] Method to factorize $N = p^r q^s$

4 An Improvement to Factorize $N = p^r q^s$
[CFRZ16] Result

Polynomial time factorization of \( N = p^r q^s \) with \( r > s \) if:

\[
r = \Omega(\log^3 \max(p, q))
\]

Decompose \( r \) and \( s \), Apply BDH or Coppersmith

\[
N = p^r q^s = (p^\alpha q^\beta)^u p^a q^b = P^u Q
\]

[CFRZ16] Factoring \( N = p^r q^s \) for Large \( r \) and \( s \). Coron, Faugère, Renault, Zeitoun, CT-RSA 2016.
Polynomial time factorization of $N = p^r q^s$ with $r > s$ if:

\[ r = \Omega(\log^3 \max(p, q)) \]

Decompose $r$ and $s$, Apply BDH or Coppersmith

\[ N = p^r q^s = (p^\alpha q^\beta)^u p^a q^b = P^u Q \]

\[ \Rightarrow \text{Apply BDH or Coppersmith method depending on } a \text{ and } b \]

[CFRZ16] Factoring $N = p^r q^s$ for Large $r$ and $s$. Coron, Faugère, Renault, Zeitoun, CT-RSA 2016.
Polynomial time factorization of $N = p^r q^s$ with $r > s$ if:

$$r = \Omega(\log^3 \max(p, q))$$

Decompose $r$ and $s$, Apply BDH or Coppersmith

$$N = p^r q^s = (p^\alpha q^\beta)^u p^a q^b = P^u Q$$

$\Rightarrow$ Apply BDH or Coppersmith method depending on $a$ and $b$

$$u = \Omega(\log Q) \Rightarrow r = \Omega(\log^3 \max(p, q))$$

[CFRZ16] Factoring $N = p^r q^s$ for Large $r$ and $s$. Coron, Faugère, Renault, Zeitoun, CT-RSA 2016.
Method to Factorize $N = p^r q^s$

First Step: Decompose $r$ and $s$

$$\begin{align*}
    r &= u \cdot \alpha + a \\
    s &= u \cdot \beta + b
\end{align*}$$

Last Step: recover $p$ and $q$

$$P = Ru$$

$$Q = \frac{N}{Pu}$$
[CFRZ16] Method to Factorize $N = p^r q^s$

First Step: Decompose $r$ and $s$

\[
\begin{align*}
  r &= u \cdot \alpha + a \\
  s &= u \cdot \beta + b
\end{align*}
\]

Use LLL

Last Step: recover $p$ and $q$

Compute $Q = N / P$

Retrieve $p$ and $q$
[CFRZ16] Method to Factorize $N = p^r q^s$

First Step: Decompose $r$ and $s$

\[
\begin{align*}
r &= u \cdot \alpha + a \\
s &= u \cdot \beta + b
\end{align*}
\]

Use LLL

Rewrite $N$ from new decomposition

\[
N = p^r q^s = p^{u \cdot \alpha + a} q^{u \cdot \beta + b} = p^{u \cdot \alpha} q^{u \cdot \beta} p^a q^b = (p^{\alpha \beta})^u p^a q^b = P^u Q
\]
[CFRZ16] Method to Factorize $N = p^r q^s$

**First Step: Decompose $r$ and $s$**

$$
\begin{align*}
  r &= u \cdot \alpha + a \\
  s &= u \cdot \beta + b
\end{align*}
$$

Use LLL

**Rewrite $N$ from new decomposition**

$$
N = p^r q^s = p^{u \cdot \alpha + a} q^{u \cdot \beta + b} = p^u q^u (p^{\alpha} q^{\beta})^{u \cdot a b} = P^u Q
$$

**Last Step: recover $p$ and $q$**

1. **Recover $P$**
2. **Compute $Q = N / P^u$**
3. **Retrieve $p$ and $q$**
1. State-of-the-art / Motivations
2. Reminder on Coppersmith / BDH Methods
3. [CFRZ16] Method to factorize $N = p^r q^s$
4. An Improvement to Factorize $N = p^r q^s$
Our Result

Polynomial time factorization of $N = p^r q^s$ with $r > s$ if:

$$r = \Omega(\log q)$$
Our Result

Polynomial time factorization of $N = p^r q^s$ with $r > s$ if:

$$r = \Omega(\log q)$$

The idea: Decompose $N^\alpha$ instead of $N$

[CFRZ16]:
- Write $N = P^u Q$
- Apply BDH/Copp. on $N$
- Condition: $u = \Omega(\log Q)$

This paper:
- Write $N^\alpha = P^r q$
- Apply BDH on $N^\alpha$
- Condition: $r = \Omega(\log q)$
Our Improvement to Factorize $N = p^r q^s$

First Step: Find $\alpha$ and $\beta$

Since $\gcd(r, s) = 1$, there exist $\alpha, \beta \in \mathbb{N}$ such that:

$$\alpha \cdot s - \beta \cdot r = 1$$
Our Improvement to Factorize $N = p^r q^s$

<table>
<thead>
<tr>
<th>First Step: Find $\alpha$ and $\beta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since $\gcd(r, s) = 1$, there exist $\alpha, \beta \in \mathbb{N}$ such that:</td>
</tr>
<tr>
<td>$\alpha \cdot s - \beta \cdot r = 1$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rewrite $N^\alpha$ from new decomposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N^\alpha = (p^r q^s)^\alpha = p^{\alpha r} q^{\alpha s} = p^{\alpha r} q^{\beta r+1} = (p^{\alpha q^{\beta}})^r q = P^r q$</td>
</tr>
</tbody>
</table>
Our Improvement to Factorize $N = p^r q^s$

First Step: Find $\alpha$ and $\beta$

Since $\gcd(r, s) = 1$, there exist $\alpha, \beta \in \mathbb{N}$ such that:

$$\alpha \cdot s - \beta \cdot r = 1$$

Rewrite $N^\alpha$ from new decomposition

$$N^\alpha = (p^r q^s)^\alpha = p^{\alpha r} q^{\alpha s} = p^{\alpha r} q^{\beta r + 1} = (p^\alpha q^\beta)^r q = P^r q$$

Last Step: Apply BDH and recover $p$ and $q$

1. **Recover $P$**
2. **Compute** $q = N^\alpha / P^r$
3. **Compute** $p = (N/q^s)^{1/r}$
Complexity comparison:

Time complexities for factoring $N = p^r q^s$, where $\log p \approx \log q$

<table>
<thead>
<tr>
<th>Condition on $N = p^r q^s$</th>
<th>[CFRZ16]</th>
<th>New Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r = \Omega(\log q)$</td>
<td>$O(\log^{12.5} N)$</td>
<td>$O(\log^{14.25} N)$</td>
</tr>
<tr>
<td>$r = \Omega(\log^3 q)$</td>
<td>$O(\log^8 N)$</td>
<td>$O(\log^{14.25} N)$</td>
</tr>
</tbody>
</table>
Generalization for $k$ prime factors

Factoring $N = \prod_{i=1}^{k} p_i^{r_i}$ for large $r_i$

- **Condition** to find a non-trivial factor of $N$ in time polynomial (with $r = \max\{r_i\}$ and $p = \max\{p_i\}$):

  $$r = \Omega(\log^{\theta_k} p)$$

  where

  $$\theta_k = 2(k - 1) \left(1 + \sum_{i=1}^{k-2} \prod_{j=i}^{k-2} j\right) + 1$$

<table>
<thead>
<tr>
<th>$k$</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\theta_k$ in [CFRZ16]</td>
<td>3</td>
<td>17</td>
<td>61</td>
<td>257</td>
<td>1301</td>
</tr>
<tr>
<td>New $\theta_k$</td>
<td>1</td>
<td>9</td>
<td>31</td>
<td>129</td>
<td>651</td>
</tr>
</tbody>
</table>
Experiments for $N = p^r q^s$ with 128-bit primes $p$ and $q$

<table>
<thead>
<tr>
<th>$N = p^5q^3$</th>
<th>Method</th>
<th>Decomposition</th>
<th>Bits given</th>
<th>Dim.</th>
<th>LLL$_f$</th>
<th>LLL$_c$</th>
<th>Est. time</th>
</tr>
</thead>
<tbody>
<tr>
<td>[CFRZ16]</td>
<td>$N = (p^2q)^3p^{-1}$</td>
<td>57</td>
<td>52</td>
<td>17s</td>
<td>3.5s</td>
<td>1.6 · 10$^{10}$ years</td>
<td></td>
</tr>
<tr>
<td>New</td>
<td>$N^2 = (p^2q)^5q$</td>
<td>46</td>
<td>78</td>
<td>0.3h</td>
<td>29s</td>
<td>6.5 · 10$^7$ years</td>
<td></td>
</tr>
</tbody>
</table>

| $N = p^7q^4$          | [CFRZ16]          | $N = (p^2q)^4p^{-1}$ | 51         | 57   | 45s     | 2.4s    | 1.7 · 10$^8$ years |
|                       | $N^2 = (p^2q)^7q$  | 43               | 92         | 1.9h | 291s    | 8.1 · 10$^7$ years  |

| $N = p^8q^3$          | [CFRZ16]          | $N = (p^2q)^4q^{-1}$ | 51         | 61   | 86s     | 4.2s    | 3 · 10$^8$ years   |
|                       | $N^3 = (p^3q)^8q$  | 57               | 95         | 6h   | 320s    | 1.4 · 10$^{12}$ years |

| $N = p^9q^5$          | [CFRZ16]          | $N = (p^2q)^5p^{-1}$ | 48         | 61   | 113s    | 4.2s    | 3.7 · 10$^7$ years  |
|                       | $N^2 = (p^2q)^9q$  | 43               | 108        | 3.9h | 801s    | 2.2 · 10$^8$ years  |

Still unpractical but asymptotically polynomial-time for large $r$
Conclusion and Remarks

Conclusion

- Improvement over [CFRZ16] method to factorize $N = p^r q^s$ for large $r$
- Generalization for moduli $N = \prod p_i^{r_i}$

Remarks

- Much simpler than [CFRZ16]
- Coppersmith’s method is not used anymore
- Bézout instead of LLL used to find good decomposition of $r$ and $s$
- For 128-bit primes, unpractical compared to ECM factorisation
- But ECM scales exponentially while our method is polynomial
- For large $p$ and $q$, our algorithm must outpace ECM
### Conclusion

- Improvement over [CFRZ16] method to factorize $N = p^r q^s$ for large $r$
- Generalization for moduli $N = \prod p_i^{r_i}$

### Remarks

- Much simpler than [CFRZ16]
  - Coppersmith’s method is not used anymore
  - Bézout instead of LLL used to find good decomposition of $r$ and $s$
- For 128-bit primes, unpractical compared to ECM factorisation
- But ECM scales exponentially while our method is polynomial
  - For large $p$ and $q$, our algorithm must outpace ECM