Benchmarking of post-quantum cryptography

Tanja Lange
tanja@hyperelliptic.org

Technische Universiteit Eindhoven

27 September 2013
Live demo on bench.cr.yp.to
Live demo on bench.cr.yp.to

Some cycle counts on h9ivy (Intel Core i5-3210M, Ivy Bridge):

- ronald1024 encrypt (RSA-1024, \(\approx 2^{80}\)) 46940
- mceliece encrypt (2008 Biswas–Sendrier, \(\approx 2^{80}\)) 61440
- gls254 DH (binary elliptic curve; CHES 2013) 77468
- kumfp127g DH (hyperelliptic curve; Eurocrypt 2013) 116944
- curve25519 DH (conservative elliptic curve) 182632
- ntruees787ep1 encrypt (from NTRU Inc., \(\approx 2^{256}\)) 398912
- ntruees787ep1 decrypt 700512
- mceliece decrypt 1219344
- ronald1024 decrypt 1340040
Efficient public-key encryption

- Batch operations are not yet in benchmarking framework: handle multiple encryptions or decryptions together. This is very useful for busy Internet nodes or cell towers.

- The McBits cryptosystem handles a batch of 256 decryptions together (CHES 2013 Bernstein–Chou–Schwabe):

<table>
<thead>
<tr>
<th>Public Key Size</th>
<th>Code Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.07MB</td>
<td>26544</td>
</tr>
<tr>
<td>0.21MB</td>
<td>60493</td>
</tr>
<tr>
<td>1MB</td>
<td>306102</td>
</tr>
</tbody>
</table>

- Speeds are per decryption for a batch of 256 decryptions.
- Decoding only; cipher time not included.

Tanja Lange “Benchmarking of post-quantum cryptography” http://bench.cr.yp.to/
Efficient public-key encryption

- Batch operations are not yet in benchmarking framework: handle multiple encryptions or decryptions together. This is very useful for busy Internet nodes or cell towers.
- The **McBits** cryptosystem handles a batch of 256 decryptions together (CHES 2013 Bernstein–Chou–Schwabe):

<table>
<thead>
<tr>
<th>Public Key Size (in MB)</th>
<th>Speed (per decryption)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.07MB</td>
<td>26544</td>
</tr>
<tr>
<td>0.21MB</td>
<td>60493</td>
</tr>
<tr>
<td>1.00MB</td>
<td>306102</td>
</tr>
</tbody>
</table>

- Speeds are per decryption for a batch of 256 decryptions.
- Decoding only; cipher time not included.
- Fully protected against software side-channel attacks, i.e. attacker can have account on same computer and not get any information on the secrets.

Tanja Lange “Benchmarking of post-quantum cryptography”
http://bench.cr.yp.to/
Here only consider binary codes, i.e. codes over $\mathbb{F}_2$.

- Basics of coding theory: Transmission channel is not perfect, so $x \in \mathbb{F}_2^n$ will have some bits flipped.

- Syndrome decoding: compute $Hx = s$ for big $(n - k) \times n$ matrix $H$.

\[
\begin{pmatrix}
1 & 0 & 1 & 1 & \ldots & 0 \\
0 & 0 & 0 & 1 & \ldots & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 1 & 0 & 0 & \ldots & 1
\end{pmatrix}
\begin{pmatrix}
1 \\
0 \\
0 \\
\vdots \\
1
\end{pmatrix}
= 
\begin{pmatrix}
0 \\
0 \\
1 \\
\vdots \\
1
\end{pmatrix}
\]

- Reconstruct error vector $e$ and thereby get originally sent codeword $x + e$.

- Works if not too many errors, i.e. number of 1s in $e$ is small. This number is called the weight.
Basics of coding theory

Here only consider binary codes, i.e. codes over $\mathbb{F}_2$.

- Basics of coding theory: Transmission channel is not perfect, so $x \in \mathbb{F}_2^n$ will have some bits flipped.

- Syndrome decoding: compute $Hx = s$ for big $(n - k) \times n$ matrix $H$.

\[
\begin{pmatrix}
1 & 0 & 1 & 1 & \ldots & 0 \\
0 & 0 & 0 & 1 & \ldots & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 1 & 0 & 0 & \ldots & 1
\end{pmatrix}
\begin{pmatrix}
1 \\
0 \\
0 \\
\vdots \\
1
\end{pmatrix}
= 
\begin{pmatrix}
0 \\
0 \\
1 \\
\vdots \\
1
\end{pmatrix}
= s
\]

- Reconstruct error vector $e$ and thereby get originally sent codeword $x + e$.

- Works if not too many errors, i.e. number of 1s in $e$ is small. This number is called the weight.

Tanja Lange “Benchmarking of post-quantum cryptography”
http://bench.cr.yp.to/
Basics of coding theory

Here only consider binary codes, i.e. codes over $\mathbb{F}_2$.

- Basics of coding theory: Transmission channel is not perfect, so $x \in \mathbb{F}_2^n$ will have some bits flipped.
- Syndrome decoding: compute $Hx = s$ for big $(n - k) \times n$ matrix $H$.

$$
\begin{pmatrix}
1 & 0 & 1 & 1 & \ldots & 0 \\
0 & 0 & 0 & 1 & \ldots & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 1 & 0 & 0 & \ldots & 1
\end{pmatrix}
\begin{pmatrix}
1 \\
0 \\
0 \\
\vdots \\
1
\end{pmatrix}
= 
\begin{pmatrix}
0 \\
1 \\
\vdots \\
1
\end{pmatrix}
= s
$$

- Reconstruct error vector $e$ and thereby get originally sent codeword $x + e$.
- Works if not too many errors, i.e. number of 1s in $e$ is small. This number is called the weight.

Tanja Lange “Benchmarking of post-quantum cryptography”
http://bench.cr.yp.to/
Basics of coding theory

Here only consider binary codes, i.e. codes over $\mathbb{F}_2$.

- Basics of coding theory: Transmission channel is not perfect, so $x \in \mathbb{F}_2^n$ will have some bits flipped.
- Syndrome decoding: compute $Hx = s$ for big $(n - k) \times n$ matrix $H$.

\[
\begin{pmatrix}
1 & 0 & 1 & 1 & \ldots & 0 \\
0 & 0 & 0 & 1 & \ldots & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 1 & 0 & 0 & \ldots & 1
\end{pmatrix}
\begin{pmatrix}
1 \\
0 \\
0 \\
1 \\
\vdots \\
1
\end{pmatrix}
= \begin{pmatrix}
0 \\
1 \\
\vdots \\
1
\end{pmatrix} = s
\]

- Reconstruct error vector $e$ and thereby get originally sent codeword $x + e$.
- Works if not too many errors, i.e. number of 1s in $e$ is small. This number is called the weight.

Tanja Lange “Benchmarking of post-quantum cryptography”

http://bench.cr.yp.to/
Basics of coding theory

Here only consider binary codes, i.e. codes over $\mathbb{F}_2$.

- Basics of coding theory: Transmission channel is not perfect, so $x \in \mathbb{F}_2^n$ will have some bits flipped.

- Syndrome decoding: compute $Hx = s$ for big $(n - k) \times n$ matrix $H$.

\[
\begin{pmatrix}
1 & 0 & 1 & 1 & \ldots & 0 \\
0 & 0 & 0 & 1 & \ldots & 0 \\
\vdots & \vdots & \vdots & \vdots & \ldots & \vdots \\
0 & 1 & 0 & 0 & \ldots & 1
\end{pmatrix}
\begin{pmatrix}
1 \\
0 \\
0 \\
1 \\
1
\end{pmatrix}
= 
\begin{pmatrix}
0 \\
1 \\
\vdots \\
1
\end{pmatrix} = s
\]

- Reconstruct error vector $e$ and thereby get originally sent codeword $x + e$.

- Works if not too many errors, i.e. number of 1s in $e$ is small. This number is called the weight.
Code-based cryptography

- Basics of coding theory: Transmission channel is not perfect, so \( x \in \mathbb{F}_2^n \) will have some bits flipped.

- Syndrome decoding: compute \( Hx = s \) for big \((n - k) \times n\) matrix \(H\). Reconstruct error vector \(e\) and thereby get originally sent codeword \(x + e\).

- Works if not too many errors, i.e. number of 1s in \(e\) is small. This number is called the weight.

- Code-based crypto uses \(e\) to transport key for symmetric encryption. Take \(e \in \mathbb{F}_q^n\) to have exactly weight \(t\).

- Users know how to derive keys for symmetric encryption (AES, Salsa20, ...) \(k(e)\) and key for message authentication \(r(e)\).

- To encrypt \(m\) to Bob, Alice looks up Bob’s matrix \(H\), computes \(s = He\), \(c = Enc_{k(e)}(m)\) and \(a = MAC_{r(e)}(c)\) and sends \(s, c, a\) to Bob.

Tanja Lange “Benchmarking of post-quantum cryptography”

http://bench.cr.yp.to/
How can this be secure?

\[
\begin{pmatrix}
1 & 0 & 1 & 1 & \cdots & 0 \\
0 & 0 & 0 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 1 & 0 & 0 & \cdots & 1
\end{pmatrix}
\begin{pmatrix}
1 \\
0 \\
0 \\
1 \\
\vdots \\
1
\end{pmatrix}
=
\begin{pmatrix}
0 \\
1 \\
\vdots \\
1
\end{pmatrix}
\]

- Code-based crypto uses two different views of the same code – one for the public parameter \( H \) which resembles a generic code and one for the secret key which is efficiently decodable.

- Classical decoding problem: find the closest codeword \( c \in C \) to a given \( x \in \mathbb{F}_2^n \), assuming that there is a unique closest codeword.

- In particular: Decoding a generic binary code of length \( n \) and without knowing anything about its structure requires about \( 2^{(0.5+o(1))n/\log_2(n)} \) binary operations (assuming a rate \( \approx 1/2 \))

- Coding theory deals with efficiently decodable codes, e.g. **Goppa codes** are efficiently decodable and lead to random looking public matrices \( H \).
Good security history

- Original parameters by McEliece in 1978 $n = 1024$, $k = 524$, $t = 50$, i.e. 50 errors in a $[1024, 524]$ code.
- In 2008 we wrote attack software against these original parameters. Attack on a single computer with a 2.4GHz Intel Core 2 Quad Q6600 CPU would need, on average, 1400 days ($2^{58}$ CPU cycles) to complete the attack.
- Parameters used in McBits offer much more security ($2^{80}$, $2^{128}$, and $2^{256}$ respectively), size of public key is $k(n - k)$ bits.
- Move from $2^{128}$ to $2^{256}$ to protect against attacks using quantum computers.

Good efficiency

- Encrypting is efficient – simple matrix-vector product.
- McBits shows that Goppa codes can be decoded efficiently and in constant time.

Tanja Lange “Benchmarking of post-quantum cryptography”

http://bench.cr.yp.to/