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):

0.07MB public key ($\approx 2^{80}$)	26544
0.21MB public key ($\approx 2^{128}$)	60493
1MB public key ($\approx 2^{256}$)	306102

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

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):

0.07MB public key ($\approx 2^{80}$)	26544
0.21MB public key ($\approx 2^{128}$)	60493
1MB public key ($\approx 2^{256}$)	306102

- 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.

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 $\mathbf{x} \in \mathbb{F}_2^n$ will have some bits flipped.
- Syndrome decoding: compute $H\mathbf{x} = \mathbf{s}$ for big $(n-k) \times n$ matrix H.

$$\begin{pmatrix} 1 & 0 & 1 & 1 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 1 & 0 & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

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 $\mathbf{x} \in \mathbb{F}_2^n$ will have some bits flipped.
- Syndrome decoding: compute $H\mathbf{x} = \mathbf{s}$ for big $(n-k) \times n$ matrix H.

$$\begin{pmatrix} 1 & 0 & 1 & 1 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 1 & 0 & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ \\ \\ \\ \end{pmatrix} = \mathbf{s}$$

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 $\mathbf{x} \in \mathbb{F}_2^n$ will have some bits flipped.
- Syndrome decoding: compute $H\mathbf{x} = \mathbf{s}$ for big $(n-k) \times n$ matrix H.

$$\begin{pmatrix} 1 & 0 & 1 & 1 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 1 & 0 & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ \end{pmatrix} = \mathbf{s}$$

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 $\mathbf{x} \in \mathbb{F}_2^n$ will have some bits flipped.
- Syndrome decoding: compute $H\mathbf{x} = \mathbf{s}$ for big $(n-k) \times n$ matrix H.

$$\begin{pmatrix} 1 & 0 & 1 & 1 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 1 & 0 & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 1 \end{pmatrix} = \mathbf{s}$$

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 $\mathbf{x} \in \mathbb{F}_2^n$ will have some bits flipped.
- Syndrome decoding: compute $H\mathbf{x} = \mathbf{s}$ for big $(n-k) \times n$ matrix H.

$$\begin{pmatrix} 1 & 0 & 1 & 1 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 1 & 0 & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 1 \end{pmatrix} = \mathbf{s}$$

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

Code-based cryptography

- Basics of coding theory: Transmission channel is not perfect, so $\mathbf{x} \in \mathbb{F}_2^n$ will have some bits flipped.
- Syndrome decoding: compute $H\mathbf{x} = \mathbf{s}$ for big $(n-k) \times n$ matrix H. Reconstruct error vector $\mathbf{e}$ and thereby get originally sent codeword $\mathbf{x} + \mathbf{e}$.
- Works if not too many errors, i.e. number of 1s in $\mathbf{e}$ is small. This number is called the weight.
- Code-based crypto uses $\mathbf{e}$ to transport key for symmetric encryption. Take $\mathbf{e} \in \mathbb{F}_q^n$ to have exactly weight t.
- Users know how to derive keys for symmetric encryption (AES, Salsa20, ...) $k(\mathbf{e})$ and key for message authentication $r(\mathbf{e})$..
- To encrypt m to Bob, Alice looks up Bob's matrix H, computes $\mathbf{s} = H\mathbf{e}$, $c = \mathrm{Enc}_{k(\mathbf{e})}(m)$ and $a = \mathrm{MAC}_{r(\mathbf{e})}(c)$ and sends $\mathbf{s}, c, a$ to Bob.

How can this be secure?

$$\begin{pmatrix} 1 & 0 & 1 & 1 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 1 & 0 & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

- 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 $\mathbf{c} \in C$ to a given $\mathbf{x} \in \mathbb{F}_2^n$, 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.