eBACS:  
ECRYPT Benchmarking of Cryptographic Systems  

http://bench.cr yp.to  

D. J. Bernstein  
University of Illinois at Chicago  

Joint work with:  
Tanja Lange  
Technische Universiteit Eindhoven

NESSIE’s performance evaluators tuned C implementations of many cryptographic systems, all supporting the same API; wrote a benchmarking toolkit; ran the toolkit on 25 computers.

Many specific performance results: e.g., 24 cycles/byte on P4 for 128-bit AES encryption.
ECRYPT I had five “virtual labs.” STVL, symmetric-techniques lab, included four working groups. STVL WG 1, stream-cipher group, ran eSTREAM (2004–2008).

De Cannière published eSTREAM benchmarking toolkit. Stream-cipher implementations matching the benchmarking API were contributed by designers, published, often tuned; benchmarked on many computers. e.g. 18 cycles/byte on P4 for third-party asm AES in toolkit.
2006: VAMPIRE, “Virtual Application and Implementation Lab,” started eBATS ("ECRYPT Benchmarking of Asymmetric Systems"), measuring efficiency of public-key encryption, signatures, DH.

*Published* a new toolkit.

Have written, collected, published 46 public-key implementations matching the benchmarking API. Benchmarked on many computers.
2008: VAMPIRE started eBASC (“ECRYPT Benchmarking of Stream Ciphers”) for post-eSTREAM benchmarks.

VAMPIRE also started eBASH (“ECRYPT Benchmarking of All Submitted Hashes”).

eBACS (“ECRYPT Benchmarking of Cryptographic Systems”) includes eBATS, eBASH, eBASC. Continues under ECRYPT II.

New toolkit, API; coordinated with CACE library (NaCl). AES now 14 cycles/byte on P4.
eBASH → public

eBASH has already collected 51 implementations of 28 hash functions in 14 families.

http://bench.cr.yp.to/results-hash.html already shows measurements on 69 machines; 95 machine-ABI combinations.

Each implementation is recompiled 1201 times with various compiler options to identify best working option for implementation, machine.
e.g. 576 bytes, katana (2137MHz Core 2 Duo 6f6), 64-bit ABI:

<table>
<thead>
<tr>
<th>25%</th>
<th>50%</th>
<th>75%</th>
<th>hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.53</td>
<td>3.54</td>
<td>3.56</td>
<td>edonr512</td>
</tr>
<tr>
<td>4.89</td>
<td>4.89</td>
<td>4.90</td>
<td>bmw512</td>
</tr>
<tr>
<td>6.51</td>
<td>6.53</td>
<td>6.53</td>
<td>md5</td>
</tr>
<tr>
<td>6.68</td>
<td>6.68</td>
<td>6.69</td>
<td>edonr256</td>
</tr>
<tr>
<td>9.18</td>
<td>9.21</td>
<td>9.22</td>
<td>bmw256</td>
</tr>
<tr>
<td>9.51</td>
<td>9.53</td>
<td>9.58</td>
<td>sha1</td>
</tr>
<tr>
<td>11.94</td>
<td>11.94</td>
<td>11.97</td>
<td>keccakr1024c576</td>
</tr>
<tr>
<td>12.22</td>
<td>12.24</td>
<td>12.26</td>
<td>blake64</td>
</tr>
<tr>
<td>16.21</td>
<td>16.25</td>
<td>16.25</td>
<td>sha512</td>
</tr>
<tr>
<td>16.81</td>
<td>16.81</td>
<td>16.82</td>
<td>ripemd160</td>
</tr>
<tr>
<td>19.44</td>
<td>19.46</td>
<td>19.46</td>
<td>blake32</td>
</tr>
<tr>
<td>23.57</td>
<td>23.67</td>
<td>23.72</td>
<td>sha256</td>
</tr>
<tr>
<td>26.83</td>
<td>26.85</td>
<td>26.86</td>
<td>groestl256</td>
</tr>
</tbody>
</table>

etc.
Tables show medians, quartiles of cycles/byte to hash 8-byte message, 64-byte message, 576-byte message, 1536-byte message, 4096-byte message, (extrapolated) long message.

Actually have much more data. e.g. Reports show best options. e.g. Graphs show medians for 0-byte message, 1-byte message, 2-byte message, 3-byte message, 4-byte message, 5-byte message, ..., 2048-byte message.
Submitter → eBASH

Define output size in api.h:

```
#define CRYPTO_BYTES 64
```
Define output size in api.h:

```c
#define CRYPTO_BYTES 64
```

Define hash function in hash.c, e.g. wrapping existing NIST API:

```c
#include "crypto_hash.h"
#include "SHA3api_ref.h"

int crypto_hash(
    unsigned char *out,
    const unsigned char *in,
    unsigned long long inlen)
{
    Hash(crypto_hash_BYTES*8
        ,in,inlen*8,out);
    return 0; }
```
Send to the mailing list the URL of a tar.gz with one directory `crypto_hash/yourhash/ref` containing `hash.c` etc.

Measurements magically appear! Much easier than trying to do your own benchmarks.

More details and options: http://bench.cr.yp.to /call-hash.html

Also easy for third parties to run the benchmark suite.