“I’ve finally finished my Skein implementation! Hmm, how fast is it?”

D. J. Bernstein
University of Illinois at Chicago

Joint work with:
Tanja Lange
Technische Universiteit Eindhoven
“I’ve finally finished my Skein implementation! Hmmm, how fast is it?”

Traditional answer:
“I’ll write a timing tool! I’ll check the clock, $10000 \times$ hash 256 bytes, check the clock again, subtract, divide by 10000.”
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

“I’ve finally finished my Skein implementation! Hmmm, how fast is it?”

Traditional answer:
“I’ll write a timing tool! I’ll check the clock, $10000 \times$ hash 256 bytes, check the clock again, subtract, divide by 10000.”

Maybe more measurements:
“Oops, lots of overhead in hashing 256 bytes. I’ll try 4096 bytes.”
“I’ve finally finished my Skein implementation! Hmm, how fast is it?”

Traditional answer:
“I’ll write a timing tool! I’ll check the clock, 10000× hash 256 bytes, check the clock again, subtract, divide by 10000.”

Maybe more measurements:
“Oops, lots of overhead in hashing 256 bytes. I’ll try 4096 bytes.”

Implementor runs tool on his favorite hardware (or emulator).
Adds “Results” section to implementation paper repeating what the tool says.

Summary:
Cryptographic implementor is the benchmark implementor and the benchmark operator.
Implementor runs tool on his favorite hardware (or emulator).
Adds “Results” section to implementation paper repeating what the tool says.

Summary:
Cryptographic implementor is the benchmark implementor and the benchmark operator.
“I’ve finally finished my Skein implementation! Hmmm, how fast is it?”

Traditional answer:
“I’ll write a timing tool! I’ll check the clock, $10000 \times$ hash 256 bytes, check the clock again, subtract, divide by 10000.”

Maybe more measurements:
“Oops, lots of overhead in hashing 256 bytes. I’ll try 4096 bytes.”

Implementor runs tool on his favorite hardware (or emulator).
Adds “Results” section to implementation paper repeating what the tool says.

Summary:
Cryptographic implementor is the benchmark implementor and the benchmark operator.
“I’ve finally finished my Skein implementation! Hmmm, how fast is it?”

Traditional answer:
“I’ll write a timing tool! I’ll check the clock, $10000 \times$ hash 256 bytes, check the clock again, subtract, divide by 10000.”

Maybe more measurements:
“Oops, lots of overhead in hashing 256 bytes. I’ll try 4096 bytes.”

Implementor runs tool on his favorite hardware (or emulator). Adds “Results” section to implementation paper repeating what the tool says.

Summary:
Cryptographic implementor is the benchmark implementor and the benchmark operator.
“I’ve finally finished my Skein implementation! Hmmm, how fast is it?”

Traditional answer:
“I’ll write a timing tool! I’ll check the clock, $10000 \times$ hash 256 bytes, check the clock again, subtract, divide by 10000.”

Maybe more measurements:
“Oops, lots of overhead in hashing 256 bytes. I’ll try 4096 bytes.”

Implementor runs tool on his favorite hardware (or emulator). Adds “Results” section to implementation paper repeating what the tool says.

Summary:
Cryptographic implementor is the benchmark implementor and the benchmark operator.

This pattern repeats for every cryptographic implementor. Hundreds (thousands?) of separate ad-hoc timing tools run on various hardware.
I've finally finished my Skein implementation! Hmmm, how fast is it?

Traditional answer:
Write a timing tool! Check the clock, hash 256 bytes, check the clock again, subtract, divide by 10000.

Maybe more measurements:
Oops, lots of overhead in hashing 256 bytes. I'll try 4096 bytes.

Implementor runs tool on his favorite hardware (or emulator). Adds “Results” section to implementation paper repeating what the tool says.

Summary:
Cryptographic implementor is the benchmark implementor and the benchmark operator.

This pattern repeats for every cryptographic implementor. Hundreds (thousands?) of separate ad-hoc timing tools run on various hardware.


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.
Implementor runs tool on his favorite hardware (or emulator). Adds “Results” section to implementation paper repeating what the tool says.

Summary:
Cryptographic implementor is the benchmark implementor and the benchmark operator.

This pattern repeats for every cryptographic implementor. Hundreds (thousands?) of separate ad-hoc timing tools run on various hardware.

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 for 128-bit AES encryption.
Implementor runs tool on his favorite hardware (or emulator). Adds “Results” section to implementation paper repeating what the tool says.

Summary:
Cryptographic implementor is the benchmark implementor and the benchmark operator.

This pattern repeats for every cryptographic implementor.
Hundreds (thousands?) of separate ad-hoc timing tools run on various hardware.


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.
Implementor runs tool on his favorite hardware (or emulator). Adds “Results” section to implementation paper repeating what the tool says.

Summary:
Cryptographic implementor is the benchmark implementor and the benchmark operator.

This pattern repeats for every cryptographic implementor. Hundreds (thousands?) of separate ad-hoc timing tools run on various hardware.

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.
Implementor runs tool on his favorite hardware (or emulator). Adds “Results” section to implementation paper repeating what the tool says.

Summary:
Cryptographic implementor is the benchmark implementor and the benchmark operator. This pattern repeats for every cryptographic implementor.
Hundreds (thousands?) of separate ad-hoc timing tools run on various hardware.

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 Canniere 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.
Implementor runs tool on his favorite hardware (or emulator). Adds "Results" section on paper repeating what the tool says.

Cryptographic implementor is the benchmark implementor and the benchmark operator. This pattern repeats for every cryptographic implementor. Hundreds (thousands?) of separate ad-hoc timing tools run on various hardware.


Many specific performance results: e.g., 24 cycles/byte on P4 for 128-bit AES encryption.

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, published, often tuned; benchmarked on many computers. e.g. 18 cycles/byte on P4 for third-party asm AES.

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.

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.

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.

NESSIE's performance evaluators tuned C implementations by cryptographic systems, 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.

Project is continuing.

Has written, collected, published 55 public-key implementations matching the benchmarking API. Benchmarked on many computers.

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.


Project is continuing. Has written, collected, published 55 public-key implementations matching the benchmarking API. Benchmarked on many computers.

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.

Project is continuing.
Has written, collected, published 55 public-key implementations matching the benchmarking API.

Benchmarked on many computers.
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.

Project is continuing.
Has written, collected, published 55 public-key implementations matching the benchmarking API. Benchmarked on many computers.
ECRYPT I had five “virtual labs.” STVL, symmetric-techniques lab, included four working groups. STVL WG 1, stream-cipher group, started 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.

Project is continuing. Has written, collected, published 55 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.
five “virtual labs.”

Symmetric-techniques lab, included four working groups.


Published

Benchmarking toolkit.

Implementations using benchmarking API by designers, often tuned;

Benchmarked on many computers.

Published a new toolkit.

Project is continuing.

Has written, collected, published 55 public-key implementations matching the benchmarking API.

Benchmarked on many computers.

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.

Project is continuing.

Has written, collected, published 55 public-key implementations matching the benchmarking API.

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.

Project is continuing.

Has written, collected, published 55 public-key implementations matching the benchmarking API.

Benchmarked on many computers.

New toolkit, API; coordinated with CACE library (NaCl).

AES now 14 cycles/byte on P4.

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.

Project is continuing.

Has written, collected, published 55 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.
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.

Project is continuing. Has written, collected, published 55 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.
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.

Project is continuing.

Written, collected, published 55 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.

Many advantages of eBACS over ad-hoc benchmarking:
- >1000 compiler options.
- >100 machine-ABI pairs.
- Many message lengths.
- Very high reliability.
- Public verifiability.
- Real API, not only timing.
- Easy for implementor!

Today have 431 implementations.

Biggest disadvantage:
Report latency is high; hard to use during development.

... but we're working on this.
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. Project is continuing. Has written, collected, published 55 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”).


Many advantages of eBACS over ad-hoc benchmarking:
- >1000 compiler options.
- >100 machine-ABI pairs.
- Many message lengths.
- Very high reliability.
- Public verifiability.
- Real API, not only timing.
- Easy for implementor!

Today have 431 implementations.

Biggest disadvantage:
Report latency is high; hard to use during development. … but we’re working on this.
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. Project is continuing. Has written, collected, published 55 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.

Many advantages of eBACS over ad-hoc benchmarking:
- >1000 compiler options.
- >100 machine-ABI pairs.
- Many message lengths.
- Very high reliability.
- Public verifiability.
- Real API, not only timing.
- Easy for implementor!

Today have 431 implementations.

Biggest disadvantage: Report latency is high; hard to use during development. ... but we’re working on this.
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.

Many advantages of eBACS over ad-hoc benchmarking:
- >1000 compiler options.
- >100 machine-ABI pairs.
- Many message lengths.
- Very high reliability.
- Public verifiability.
- Real API, not only timing.
- Easy for implementor!

Today have 431 implementations.

Biggest disadvantage:
Report latency is high; hard to use during development.
... but we’re working on this.