Software speed for secret-key cryptography

D. J. Bernstein
University of Illinois at Chicago

Inadequate speed often
stops cryptographic deployment
. . . or compromises security.

Speed plays a critical role in
selection and standardization
of primitives, platforms,
tools, and implementations.

Two interacting goals:

- "Benchmarking":
  Measure speed that
  users see from crypto.
- "Optimization":
  Improve speed that
  users see from crypto.

Joint work with Tanja Lange: eBACS (ECRYPT Benchmarking of Cryptographic Systems).

http://bench.cr.yp.to

Main focus of this talk:
Future of software benchmarking.

Joint work with Tanja Lange: eBACS (ECRYPT Benchmarking of Cryptographic Systems).

http://bench.cr.yp.to

Main focus of this talk: Future of software benchmarking.

What about hardware? Collaborating with Kris Gaj and Patrick Schaumont: "Environment for Fair and Comprehensive Performance Evaluation of Cryptographic Hardware and Software."

Performance evaluator has to

- implement various designs,
- optimize for various platforms,
- write benchmarking tool,
- collect many measurements.

Too hard? Evaluator cuts corners.

Performance evaluator has to

- implement various designs,
- optimize for various platforms,
- write benchmarking tool,
- collect many measurements.

Too hard? Evaluator cuts corners.

eBACS *distributes* these tasks:

- J.-P. implements BLAKE.
- Peter, Samuel speed it up.
- Dan writes the tool.
- Søren runs tool on Pentium M.

Performance evaluator has to

- implement various designs,
- optimize for various platforms,
- write benchmarking tool,
- collect many measurements.

Too hard? Evaluator cuts corners.

eBACS *distributes* these tasks:

- J.-P. implements BLAKE.
- Peter, Samuel speed it up.
- Dan writes the tool.
- Søren runs tool on Pentium M.

eBACS does *not* distinguish "benchmarkable" software from "real" software. Uses a real API!

More operations

Secret-key operations
measured in eBACS today:

- Hash functions.
- Stream ciphers.

Plan to measure more operations:

- Authenticators.
- One-time authenticators.
- Authenticated encryption.

API co-developed with NaCl:
http://nacl.cace-project.eu
Plan to extend precomputation.

More CPUs

Plan to buy a zoo of CPUs
that are not yet covered
by the benchmarks.
Emphasis on low-end CPUs.

Extremely small CPUs need
cross-compilation structure,
separating library ABI from
benchmark-management ABI.

News: Christian Wenzel-Benner
and Jens Gräf have
collected some benchmarks
on an Atmel Atmega1281.

More communication costs

Cryptographic software competes
with other networking tools
for instruction-cache space.

Current benchmarks
don't see this.

Plan to systematically measure
varying levels of cache contention.

Also plan to measure
costs of many active keys etc.

Also plan to measure
performance of batch operations.

More parallelism

Current benchmarks are
limited to single-core
computations.

Good for high-throughput servers
that have many concurrent tasks
and that keep all CPU cores
busy with separate tasks.

But some applications need
minimum *latency* for one task.

Multiple cores save time.
Plan to measure this.
(Multiple machines can
save time too; lower priority.)

More security

"Stop using 160-bit hashes!"
. . . Users can easily find
speed of 256-bit hash software,
512-bit hash software, etc.

More security

"Stop using 160-bit hashes!"
... Users can easily find
speed of 256-bit hash software,
512-bit hash software, etc.

"Stop side-channel attacks!"
... Can users find speed of
constant-time hash software?

Plan to separately report
speed of software
declared to be constant time.
(Maybe CACE-verified?)

More automation

Implementor finishes software.
Easily sends in for benchmarking.

Software is *manually*
included in benchmark toolkit.
Toolkit is run *manually*.
Manual steps add latency:
often weeks or months.

Plan to have machines
automatically run new software
in resource-limited sandbox.

Much lower latency.
Fast feedback to implementor.

More tables and graphs

Current results online:
each primitive on Core 2 Duo;
each primitive on Opteron; etc.
http://bench.cr.yp.to

Plan to add transposed results:
BLAKE across machines;
BMW across machines; etc.

Plan to allow dynamic
superposition of results:
e.g., SHA-512 vs. SIMD vs. Skein.