High-speed high-security cryptography on ARMs

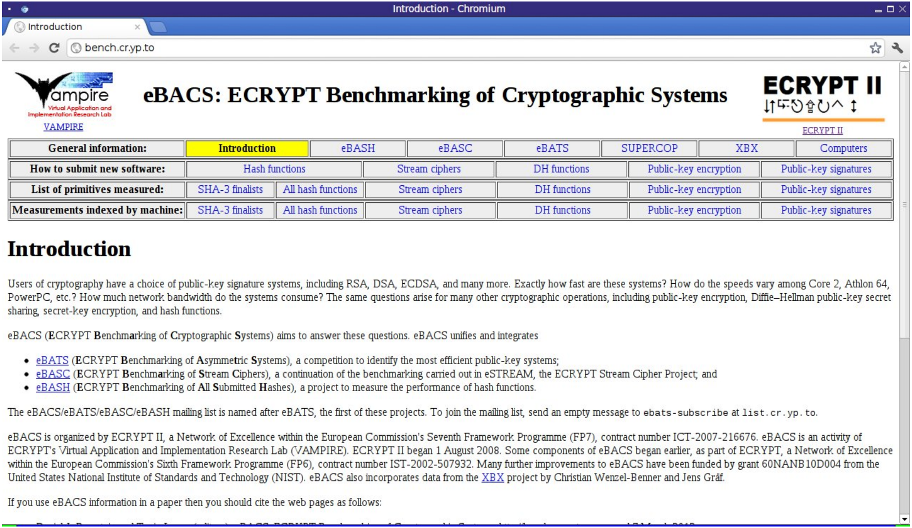
Daniel J. Bernstein
Research Professor, University of Illinois at Chicago
Professor, Cryptographic Implementations, Technische Universiteit Eindhoven

Tanja Lange
Professor, Coding and Cryptology, Technische Universiteit Eindhoven

joint work with: Peter Schwabe, Academia Sinica

"Don't roll your own crypto. Leave it to us."

Speed of cryptography: `http://bench.cr.yp.to`



Introduction - Chromium

Introduction

bench.cr.yp.to

VAMPIRE

eBACS: ECRYPT Benchmarking of Cryptographic Systems

ECRYPT II

General information:	**Introduction**	eBASH	eBASC	eBATS	SUPERCOP	XBX	Computers

How to submit new software:	Hash functions		Stream ciphers	DH functions	Public-key encryption	Public-key signatures
List of primitives measured:	SHA-3 finalists	All hash functions	Stream ciphers	DH functions	Public-key encryption	Public-key signatures
Measurements indexed by machine:	SHA-3 finalists	All hash functions	Stream ciphers	DH functions	Public-key encryption	Public-key signatures

Introduction

Users of cryptography have a choice of public-key signature systems, including RSA, DSA, ECDSA, and many more. Exactly how fast are these systems? How do the speeds vary among Core 2, Athlon 64, PowerPC, etc.? How much network bandwidth do the systems consume? The same questions arise for many other cryptographic operations, including public-key encryption, Diffie–Hellman public-key secret sharing, secret-key encryption, and hash functions.

eBACS (**E**CRYPT **B**enchm**a**rking of **C**ryptographic **S**ystems) aims to answer these questions. eBACS unifies and integrates

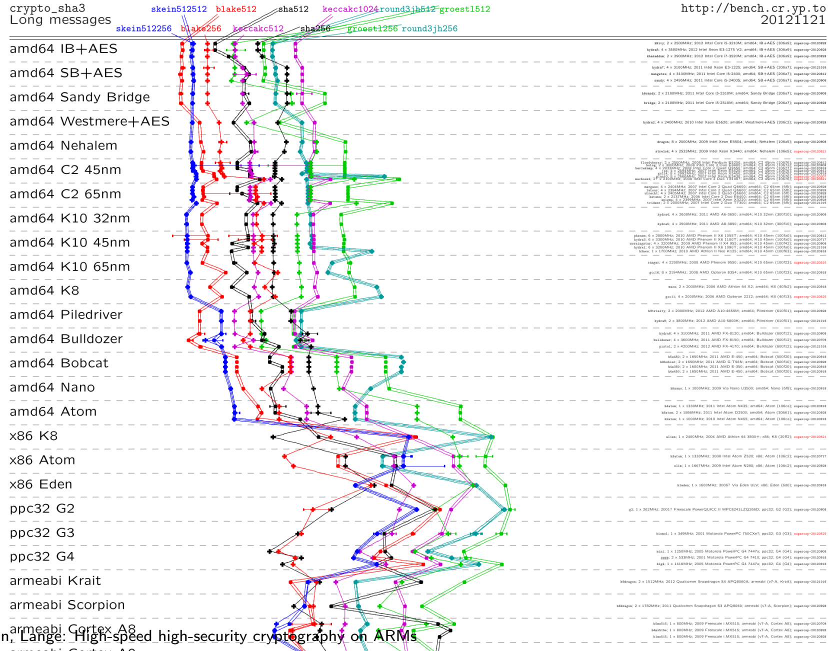
- eBATS (**E**CRYPT **B**enchmarking of **A**symme**t**ric **S**ystems), a competition to identify the most efficient public-key systems;
- eBASC (**E**CRYPT **B**enchm**a**rking of **S**tream **C**iphers), a continuation of the benchmarking carried out in eSTREAM, the ECRYPT Stream Cipher Project; and
- eBASH (**E**CRYPT **B**enchm**a**rking of **A**ll **S**ubmitted **H**ashes), a project to measure the performance of hash functions.

The eBACS/eBATS/eBASC/eBASH mailing list is named after eBATS, the first of these projects. To join the mailing list, send an empty message to `ebats-subscribe` at `list.cr.yp.to`.

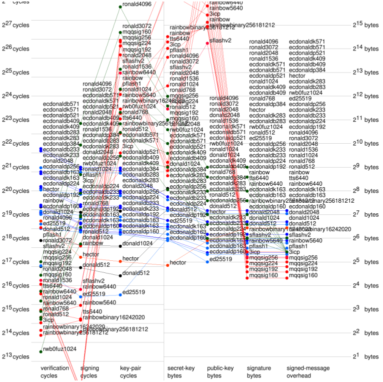
eBACS is organized by ECRYPT II, a Network of Excellence within the European Commission's Seventh Framework Programme (FP7), contract number ICT-2007-216676. eBACS is an activity of ECRYPT's Virtual Application and Implementation Research Lab (VAMPIRE). ECRYPT II began 1 August 2008. Some components of eBACS began earlier, as part of ECRYPT, a Network of Excellence within the European Commission's Sixth Framework Programme (FP6), contract number IST-2002-507932. Many further improvements to eBACS have been funded by grant 60NANB10D004 from the United States National Institute of Standards and Technology (NIST). eBACS also incorporates data from the XBX project by Christian Wenzel-Benner and Jens Gräf.

If you use eBACS information in a paper then you should cite the web pages as follows:

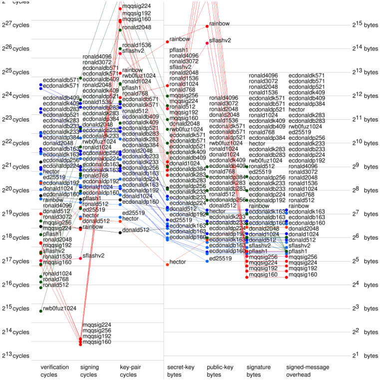
Speed comparison of the SHA-3 finalists



Speed comparison of signature systems



64-bit 2400MHz Intel Xeon E5620



32-bit 1900MHz Pentium 4

Benchmarking of cryptographic systems

- We benchmark all submitted primitives on more than 100 different CPUs. So far 1112 implementations submitted.
- Cooperating project for smaller CPUs: `xbx.das-labor.org`.
- Benchmarking framework and all implementations are public. Anybody can run benchmark on own computer (we're happy to post your data!).
- Clear speed differences between security levels.
- Clear speed differences between devices — we count cycles to eliminate influence of clock speed but CPUs do different # operations in one clock cycle.
- Clear speed differences between libraries (e.g. OpenSSL much slower than NaCl)
- Clear speed differences between choices within one family, e.g. elliptic-curve speed depends on the representation, the coordinate system, the windowing method, . . .

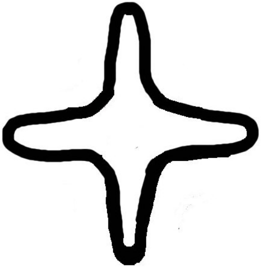
Networking and Cryptography library (NaCl)

nacl.cr.yp.to

- All cryptography has at least 128-bit security (takes at least 2^{128} operations to break).
- Covers
  - authenticated encryption (incl. DH key exchange)
  - signatures
  - both based on elliptic curves for public-key part
- Easy user interface.
- Very good speed.
- Software side-channel attack resistant: all operations take constant time, no key-dependent branches, no key-dependent addresses.
- Implementations are public domain, no known patent issues.
- Biggest early adopter: DNSCrypt from OpenDNS.

Public-key authenticated encryption

- User interface: `crypto_box`, takes public key of recipient, secret key of sender, and a nonce (number used only once)
- Diffie-Hellman key exchange using long-term keys.
- Stream cipher with MAC for bulk encryption.
- Security-optimized speed-optimized choice of cryptographic primitives, more specifically
  - Elliptic curve in twisted Edwards form, conforming to IEEE-P1363, plus additional security properties.
  - Salsa20 stream cipher (from final eSTREAM portfolio).
  - Poly1305 (information-theoretic security).



$$ax^2+y^2=1+dx^2y^2$$

Car security as a crypto performance challenge

Car security as a crypto performance challenge

Car security as a crypto performance challenge

PRESERVE deliverable 1.1, "Security Requirements of VSA":

> *The different driving scenarios we looked into indicate that in most driving situations (SUL, MUL, and SHL) the packet rates do not exceed 750 packets per second. Only the maximum highway scenario (MHL) goes well beyond this value (2,265 packets per second). . . .*
>
> *Processing 1,000 packets per second and processing each in 1 ms can hardly be met by current hardware. As discussed in [32], a Pentium D 3.4 GHz processor needs about 5 times as long for a verification (which is the most time-consuming operation in cryptographic processing overhead) and a typical OBU even 26 times as long. This is a good indication that a dedicated cryptographic co-processor is likely to be necessary.*

Performance measurement on a large processor

NaCl measurements on typical desktop CPU
(4-core 2400MHz Intel Xeon E5620, $390 last year):

- 4.99 cycles/byte for secret-key encryption (Salsa20).
- 2.66 cycles/byte for secret-key authentication (Poly1305).
- 227348 cycles for public-key session (ECDH: Curve25519).
- 272592 cycles to verify a signature (EdDSA: Ed25519).
- 133593 cycles to verify a signature inside a *batch*.
- 87336 cycles to sign a message.

Performance measurement on a large processor

NaCl measurements on typical desktop CPU
(4-core 2400MHz Intel Xeon E5620, $390 last year):

- 4.99 cycles/byte for secret-key encryption (Salsa20).
- 2.66 cycles/byte for secret-key authentication (Poly1305).
- 227348 cycles for public-key session (ECDH: Curve25519).
- 272592 cycles to verify a signature (EdDSA: Ed25519).
- 133593 cycles to verify a signature inside a *batch*.
- 87336 cycles to sign a message.

Let's focus on ECDH: **42000** operations/second.

Performance measurement on a large processor

NaCl measurements on typical desktop CPU
(4-core 2400MHz Intel Xeon E5620, $390 last year):

- 4.99 cycles/byte for secret-key encryption (Salsa20).
- 2.66 cycles/byte for secret-key authentication (Poly1305).
- 227348 cycles for public-key session (ECDH: Curve25519).
- 272592 cycles to verify a signature (EdDSA: Ed25519).
- 133593 cycles to verify a signature inside a *batch*.
- 87336 cycles to sign a message.

Let's focus on ECDH: **42000** operations/second.

90000 operations/second on another typical desktop CPU
(3300MHz 6-core AMD Phenom II X6 1100T CPU, $190 last year).

Performance measurement on a small processor

BeagleBone development board revision A6: 720MHz TI Sitara AM3359 CPU; ARM Cortex A8 core.

(Picture credits: `beagleboard.org`)

Our public-key speed:

Performance measurement on a small processor

BeagleBone development board revision A6: 720MHz TI Sitara AM3359 CPU; ARM Cortex A8 core.

(Picture credits: `beagleboard.org`)

Our public-key speed: **1447** ECDH/second. Fully protected against software side-channel attacks.

Performance extrapolation

Faster Cortex A8 cores are widely used in mobile devices:

- 1000MHz Apple A4 in iPad 1, iPhone 4 (2010);
- 1000MHz Samsung Exynos 3110 in Samsung Galaxy S (2010);
- 1000MHz TI OMAP3630 in Motorola Droid X (2010);
- 800MHz Freescale i.MX50 in Amazon Kindle 4 (2011);
- etc.

Our tests on various Cortex A8 cores demonstrate that performance is almost exactly linear in clock speed.

Performance extrapolation

Faster Cortex A8 cores are widely used in mobile devices:

- 1000MHz Apple A4 in iPad 1, iPhone 4 (2010);
- 1000MHz Samsung Exynos 3110 in Samsung Galaxy S (2010);
- 1000MHz TI OMAP3630 in Motorola Droid X (2010);
- 800MHz Freescale i.MX50 in Amazon Kindle 4 (2011);
- etc.

Our tests on various Cortex A8 cores demonstrate that performance is almost exactly linear in clock speed.

Reasonably safe ECDH extrapolations to other Cortex A8 cores:

- **1200**/second: 600MHz TI AM3352ZCZ60, **13.5€** (DigiKey).
- **2400**/second: 1200MHz Allwinner A10, reportedly **$7**.

But always better to measure directly. We're working on it!