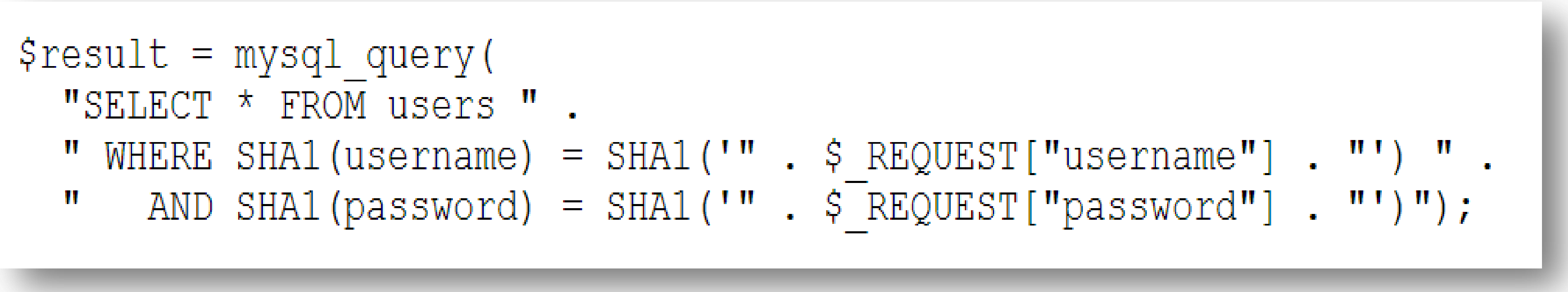
The Password Hashing Competition

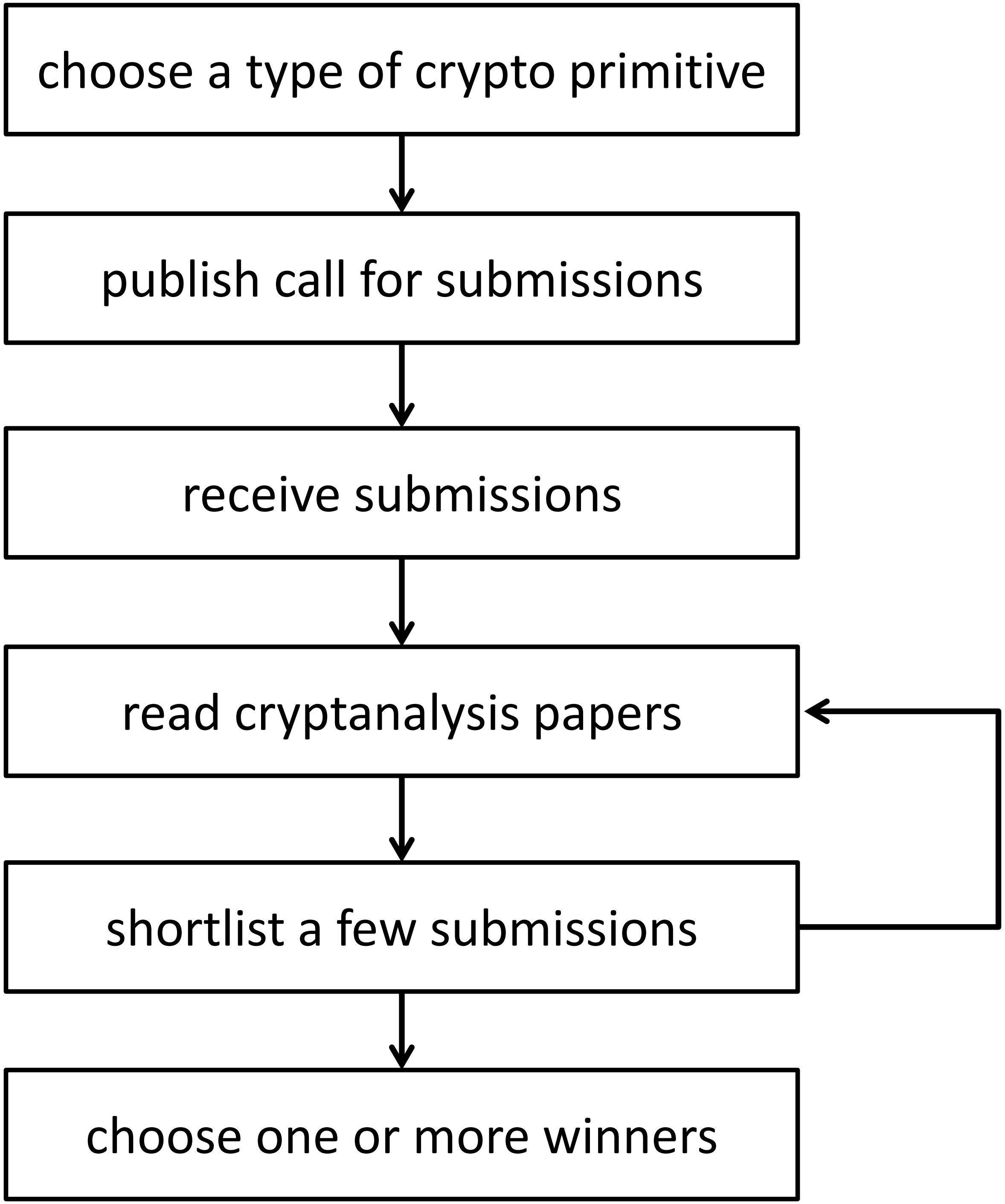


```
$result = mysql_query(
  "SELECT * FROM users " .
  " WHERE SHA1(username) = SHA1('" . $_REQUEST["username"] . "') " .
  "   AND SHA1(password) = SHA1('" . $_REQUEST["password"] . "')");
```

JP Aumasson — @veorq

PasswordsCon13 LV

A crypto competition



choose a type of crypto primitive

publish call for submissions

receive submissions

read cryptanalysis papers

shortlist a few submissions

choose one or more winners

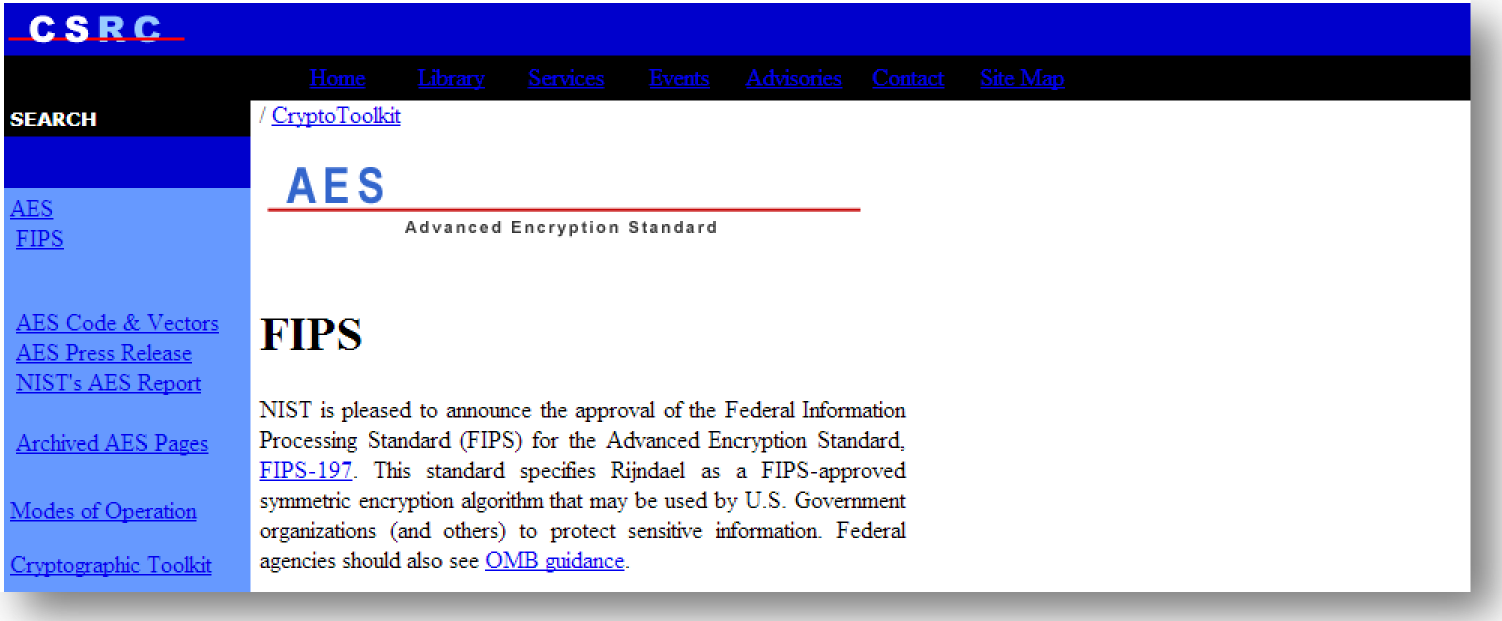
1997-2000: **AES** (NIST)

Block ciphers

15 submissions

5 'finalists'

1 winner: Rijndael



FIPS

NIST is pleased to announce the approval of the Federal Information Processing Standard (FIPS) for the Advanced Encryption Standard, FIPS-197. This standard specifies Rijndael as a FIPS-approved symmetric encryption algorithm that may be used by U.S. Government organizations (and others) to protect sensitive information. Federal agencies should also see OMB guidance.

2004-2008: **eSTREAM** (ECRYPT)

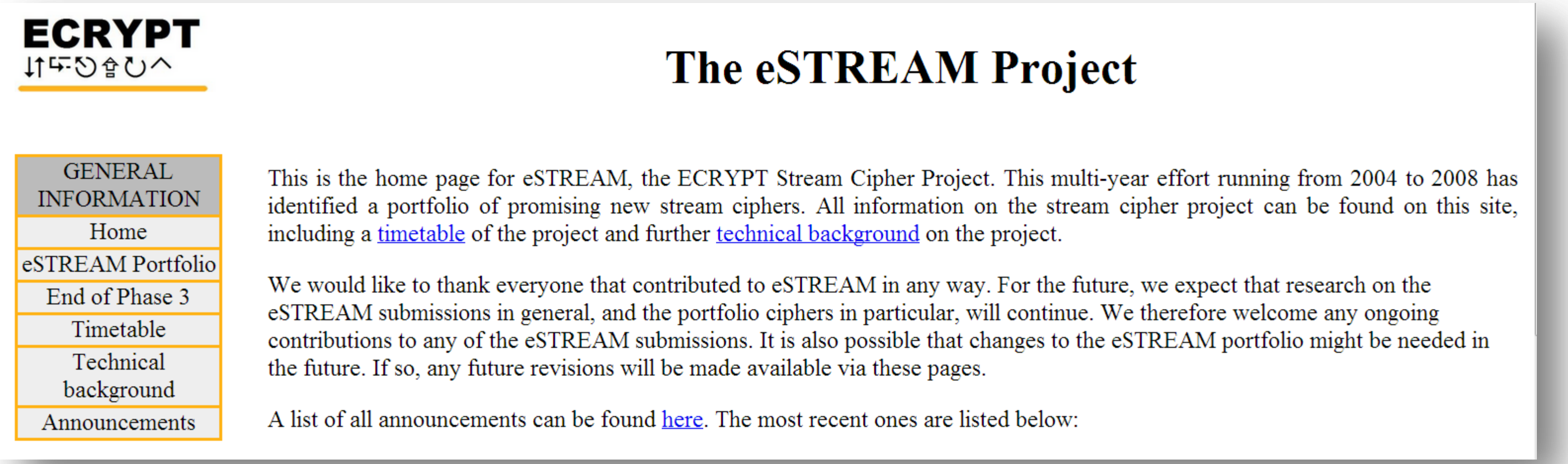
Stream ciphers

34 submissions

27 ‘second-round’ candidates

16 ‘finalists’

portfolio of 8 (-1) winners



The eSTREAM Project

This is the home page for eSTREAM, the ECRYPT Stream Cipher Project. This multi-year effort running from 2004 to 2008 has identified a portfolio of promising new stream ciphers. All information on the stream cipher project can be found on this site, including a timetable of the project and further technical background on the project.

We would like to thank everyone that contributed to eSTREAM in any way. For the future, we expect that research on the eSTREAM submissions in general, and the portfolio ciphers in particular, will continue. We therefore welcome any ongoing contributions to any of the eSTREAM submissions. It is also possible that changes to the eSTREAM portfolio might be needed in the future. If so, any future revisions will be made available via these pages.

A list of all announcements can be found here. The most recent ones are listed below:

2007-2012: **SHA-3** (NIST)

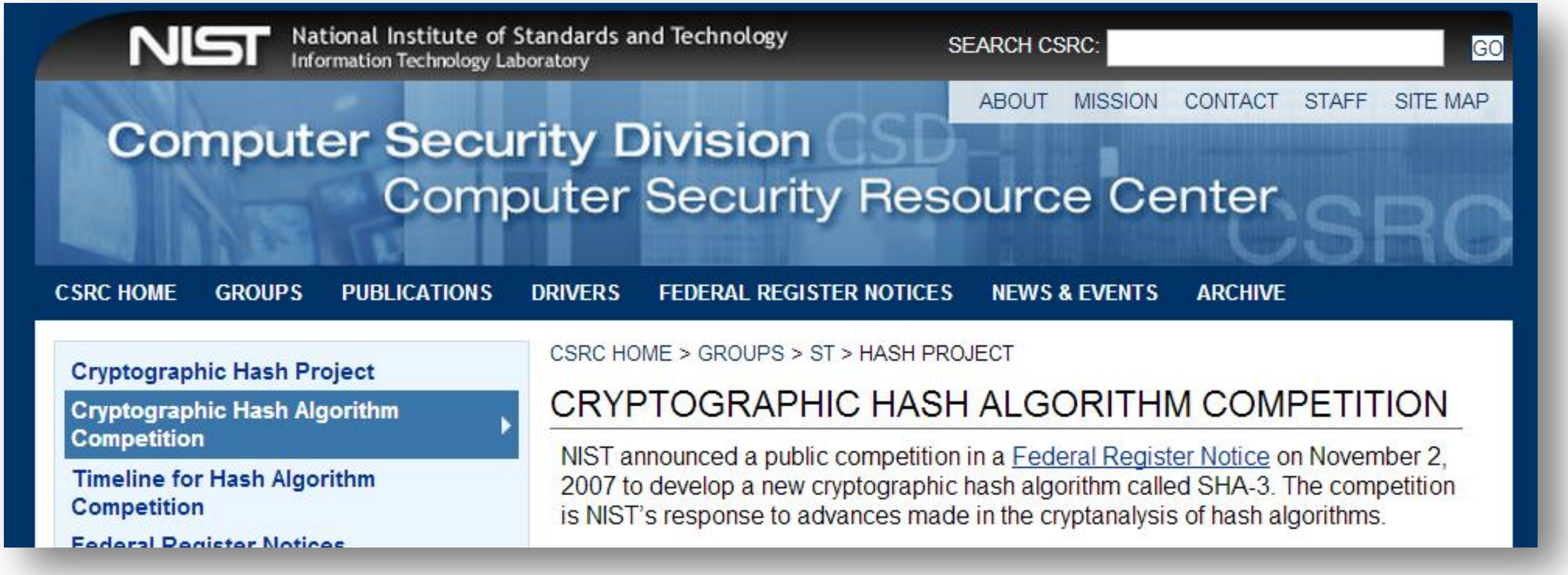
Hash functions

51 submissions

14 ‘second-round’ candidates

5 ‘finalists’

1 winner: Keccak



NIST National Institute of Standards and Technology
Information Technology Laboratory
SEARCH CSRC:
GO
ABOUT MISSION CONTACT STAFF SITE MAP
Computer Security Division CSD
Computer Security Resource Center CSRC
CSRC HOME GROUPS PUBLICATIONS DRIVERS FEDERAL REGISTER NOTICES NEWS & EVENTS ARCHIVE
Cryptographic Hash Project
Cryptographic Hash Algorithm Competition
Timeline for Hash Algorithm Competition
Federal Register Notices
CSRC HOME > GROUPS > ST > HASH PROJECT

CRYPTOGRAPHIC HASH ALGORITHM COMPETITION

NIST announced a public competition in a Federal Register Notice on November 2, 2007 to develop a new cryptographic hash algorithm called SHA-3. The competition is NIST’s response to advances made in the cryptanalysis of hash algorithms.

Crypto demolition derbies

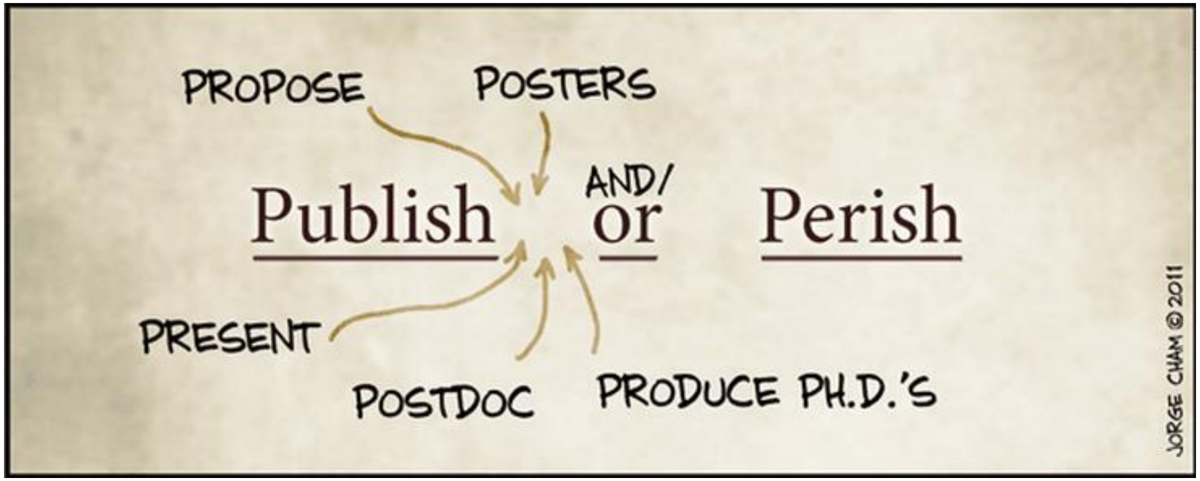
Survival of the **fittest**

Balance between

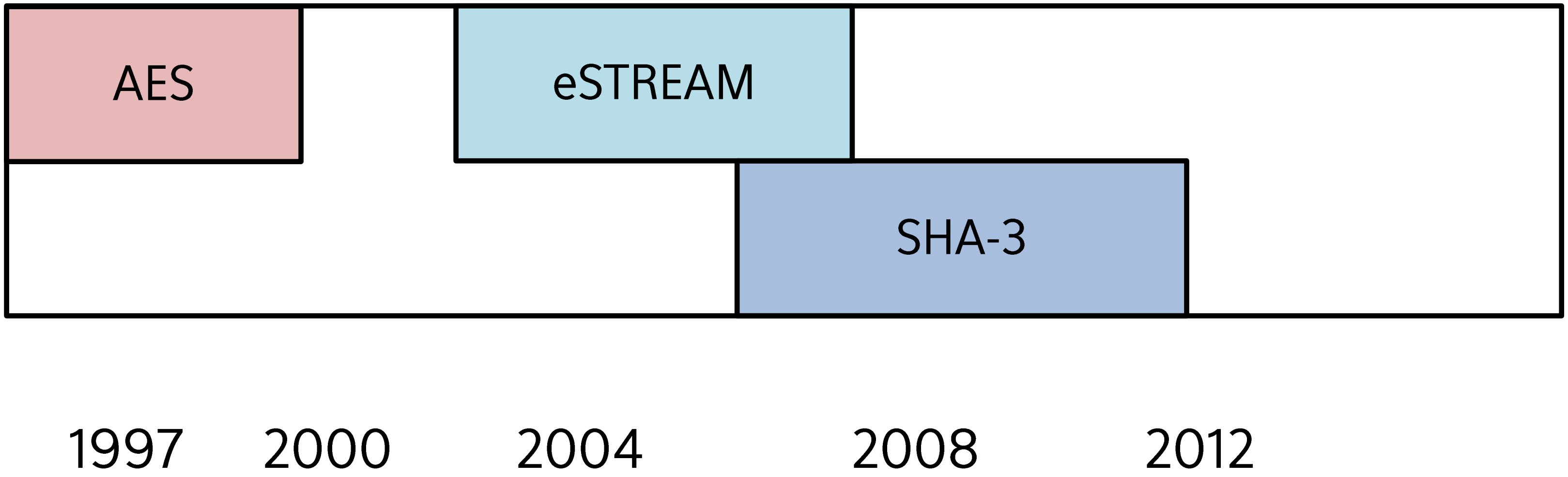
security, functionalities, efficiency, simplicity, etc.

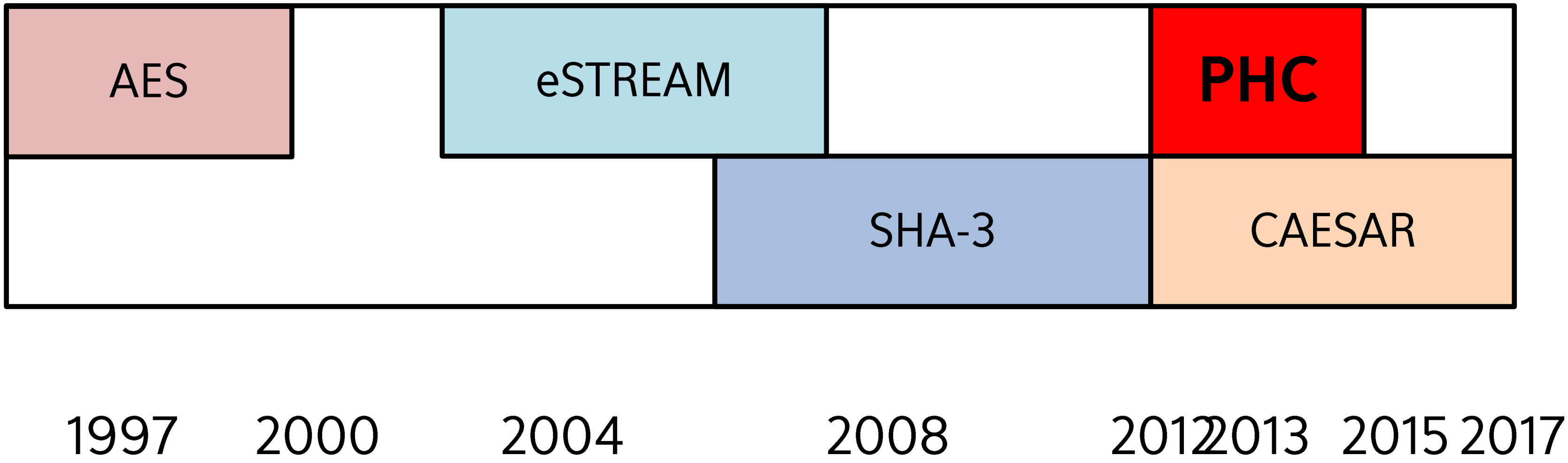
Incentive model

Design great ciphers → **reputation++**
Break candidate ciphers → **papers++**
Design|analyze|implement → **grants++**
Competition and conferences → **fun++**



→ **Free work** for the organizers


AES
eSTREAM
SHA-3
1997
2000
2004
2008
2012



AES

eSTREAM

PHC

SHA-3

CAESAR

1997 2000 2004 2008 2012 2013 2015 2017

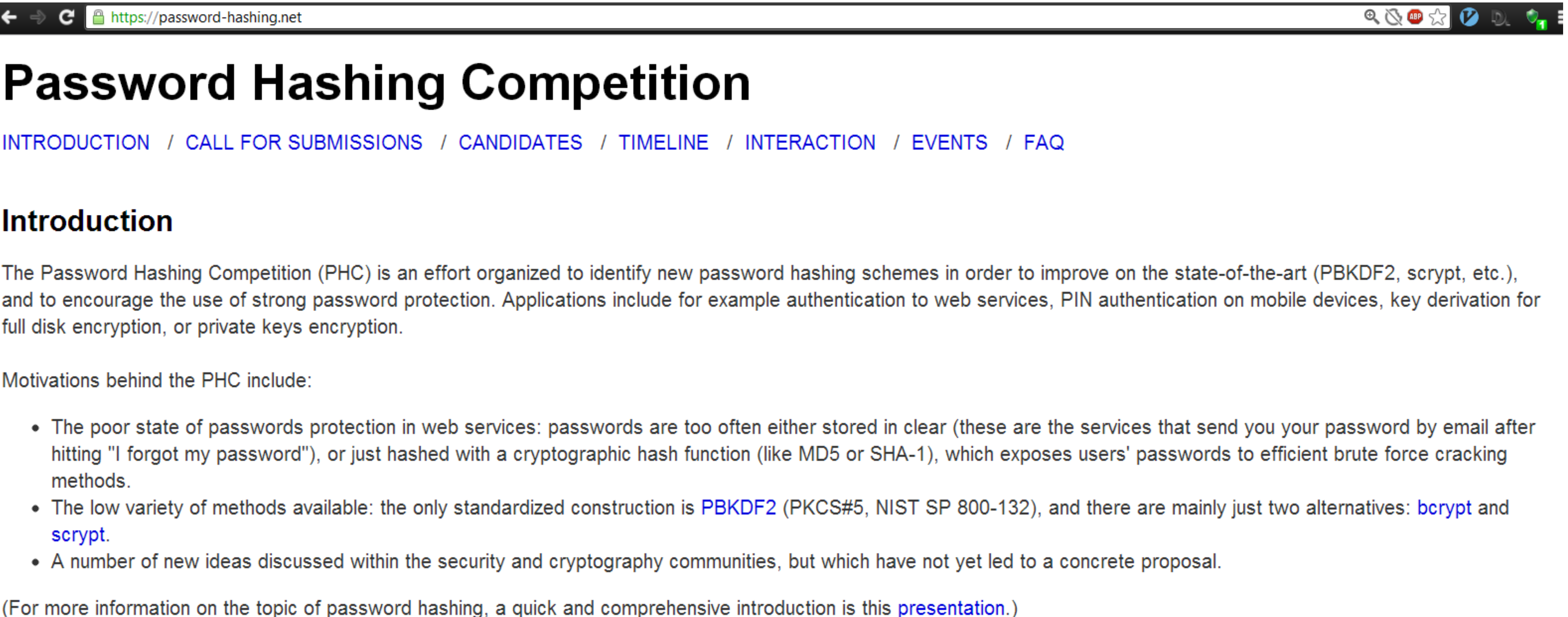
Password Hashing Competition (PHC)

2013-2015

Password hashing schemes

Organized by a group of passionate experts

Open to everyone, vendor-neutral, no sponsors


https://password-hashing.net
INTRODUCTION / CALL FOR SUBMISSIONS / CANDIDATES / TIMELINE / INTERACTION / EVENTS / FAQ

Password Hashing Competition

Introduction

The Password Hashing Competition (PHC) is an effort organized to identify new password hashing schemes in order to improve on the state-of-the-art (PBKDF2, scrypt, etc.), and to encourage the use of strong password protection. Applications include for example authentication to web services, PIN authentication on mobile devices, key derivation for full disk encryption, or private keys encryption.

Motivations behind the PHC include:

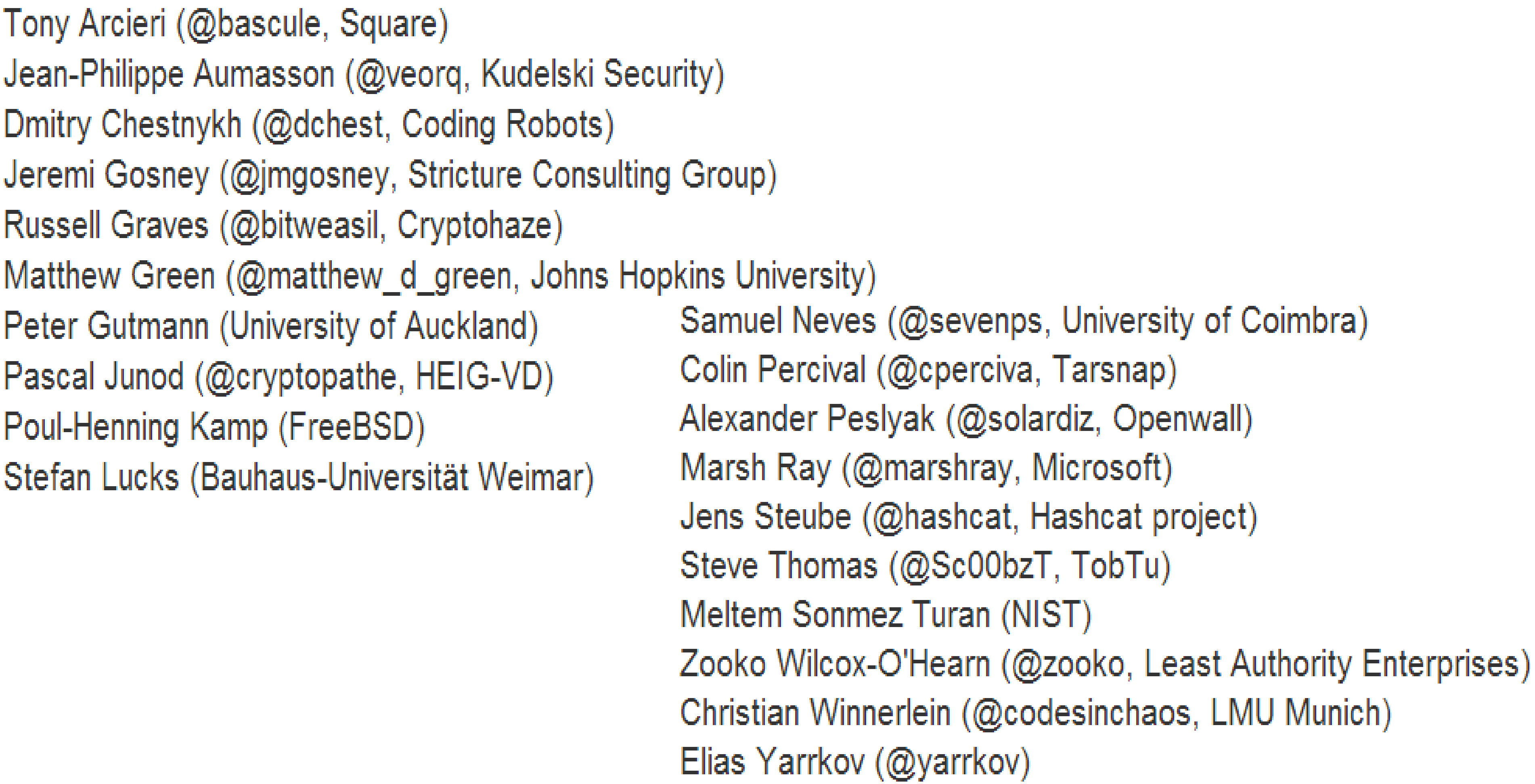
- The poor state of passwords protection in web services: passwords are too often either stored in clear (these are the services that send you your password by email after hitting "I forgot my password"), or just hashed with a cryptographic hash function (like MD5 or SHA-1), which exposes users' passwords to efficient brute force cracking methods.
- The low variety of methods available: the only standardized construction is PBKDF2 (PKCS#5, NIST SP 800-132), and there are mainly just two alternatives: bcrypt and scrypt.
- A number of new ideas discussed within the security and cryptography communities, but which have not yet led to a concrete proposal.

(For more information on the topic of password hashing, a quick and comprehensive introduction is this presentation.)

PHC panel

From industry, academia, US government

Crackers, software engineers, cryptographers…



Tony Arcieri (@bascule, Square)

Jean-Philippe Aumasson (@veorq, Kudelski Security)

Dmitry Chestnykh (@dchest, Coding Robots)

Jeremi Gosney (@jmgosney, Stricture Consulting Group)

Russell Graves (@bitweasil, Cryptohaze)

Matthew Green (@matthew_d_green, Johns Hopkins University)

Peter Gutmann (University of Auckland)

Pascal Junod (@cryptopathe, HEIG-VD)

Poul-Henning Kamp (FreeBSD)

Stefan Lucks (Bauhaus-Universität Weimar)

Samuel Neves (@sevenps, University of Coimbra)

Colin Percival (@cperciva, Tarsnap)

Alexander Peslyak (@solardiz, Openwall)

Marsh Ray (@marshray, Microsoft)

Jens Steube (@hashcat, Hashcat project)

Steve Thomas (@Sc00bzT, TobTu)

Meltem Sonmez Turan (NIST)

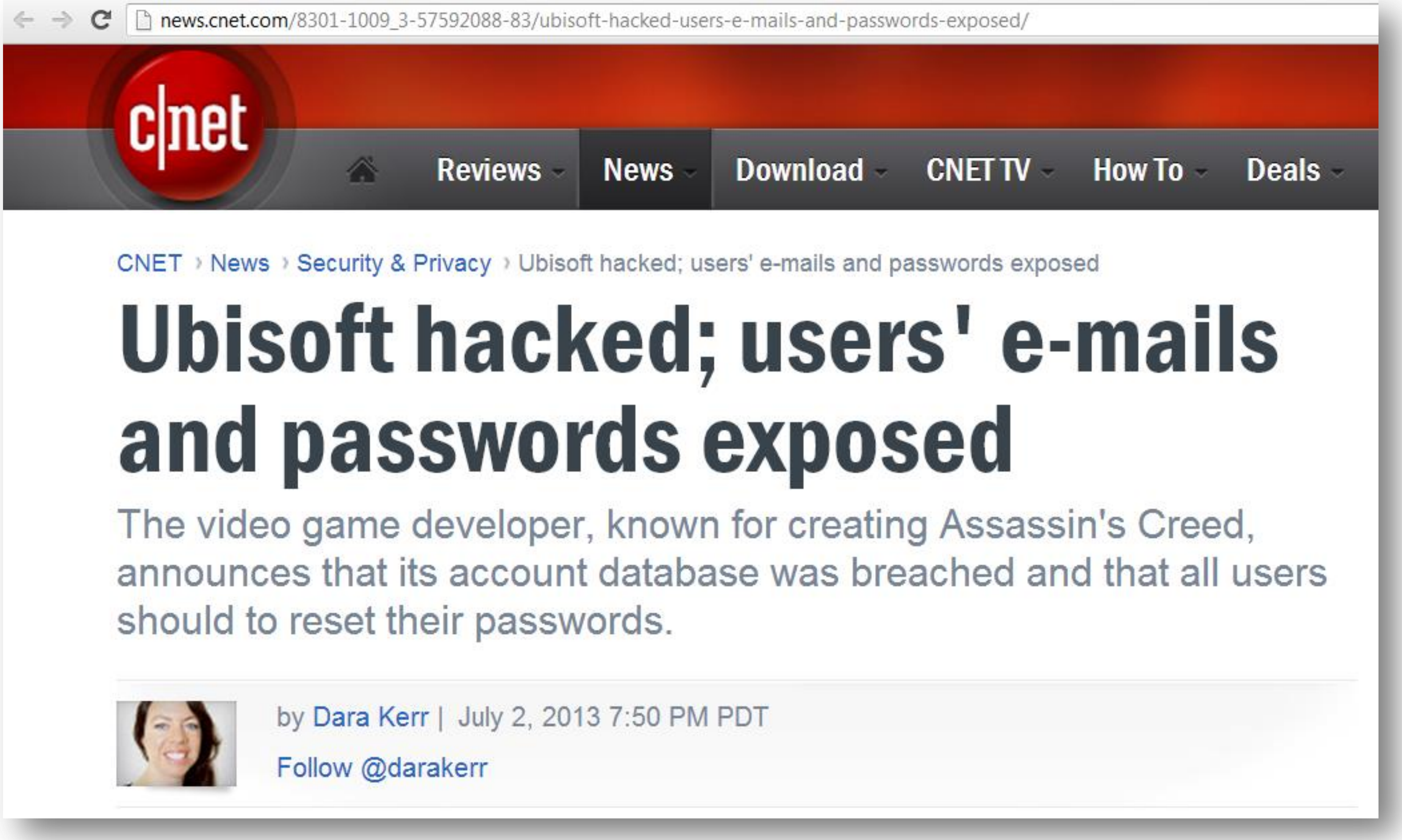
Zooko Wilcox-O'Hearn (@zooko, Least Authority Enterprises)

Christian Winnerlein (@codesinchaos, LMU Munich)

Elias Yarrkov (@yarrkov)

Motivations?

July 2, 2013



news.cnet.com/8301-1009_3-57592088-83/ubisoft-hacked-users-e-mails-and-passwords-exposed/

CNET › News › Security & Privacy › Ubisoft hacked; users' e-mails and passwords exposed

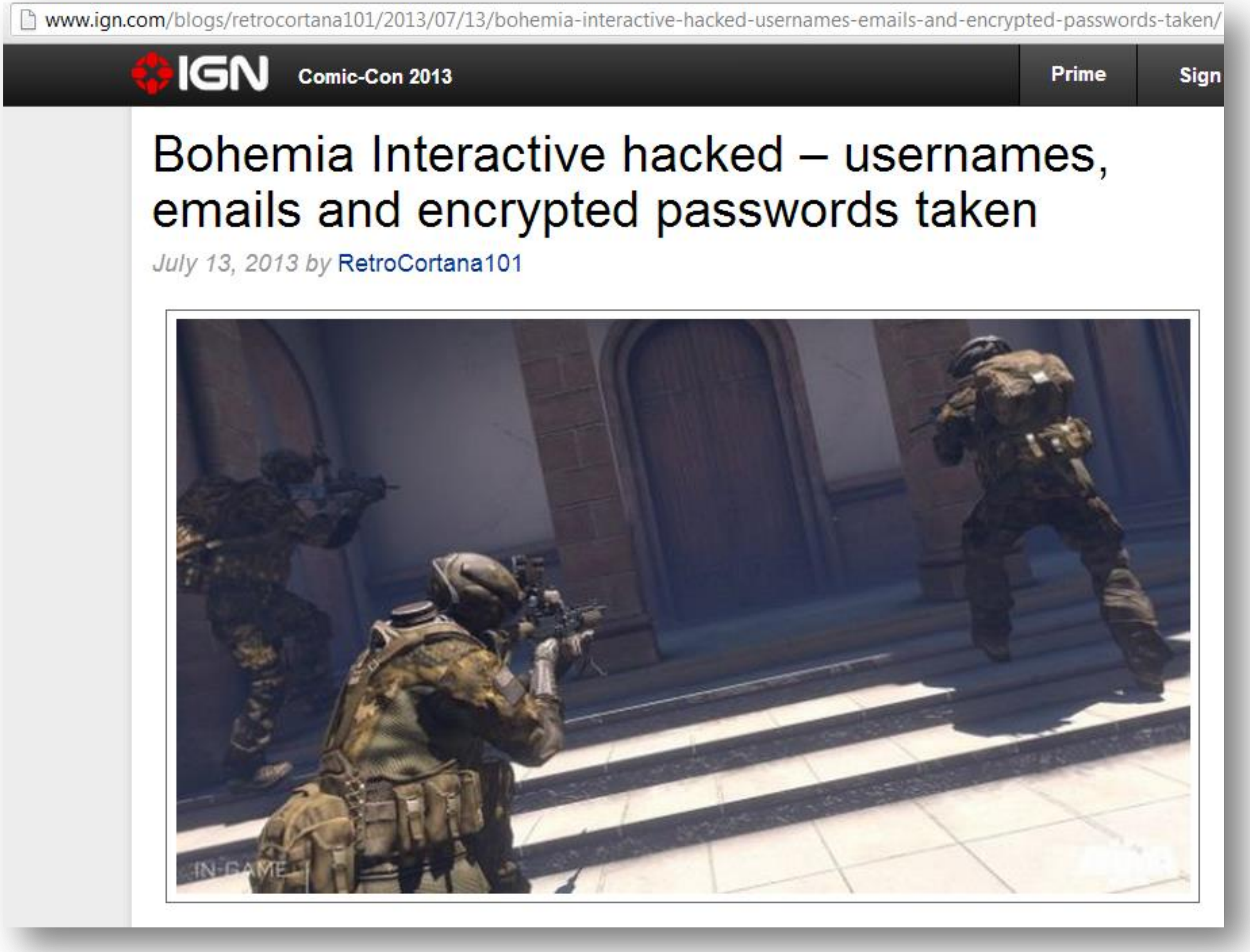
Ubisoft hacked; users' e-mails and passwords exposed

The video game developer, known for creating Assassin's Creed, announces that its account database was breached and that all users should to reset their passwords.

by Dara Kerr | July 2, 2013 7:50 PM PDT

Follow @darakerr

July 13, 2013

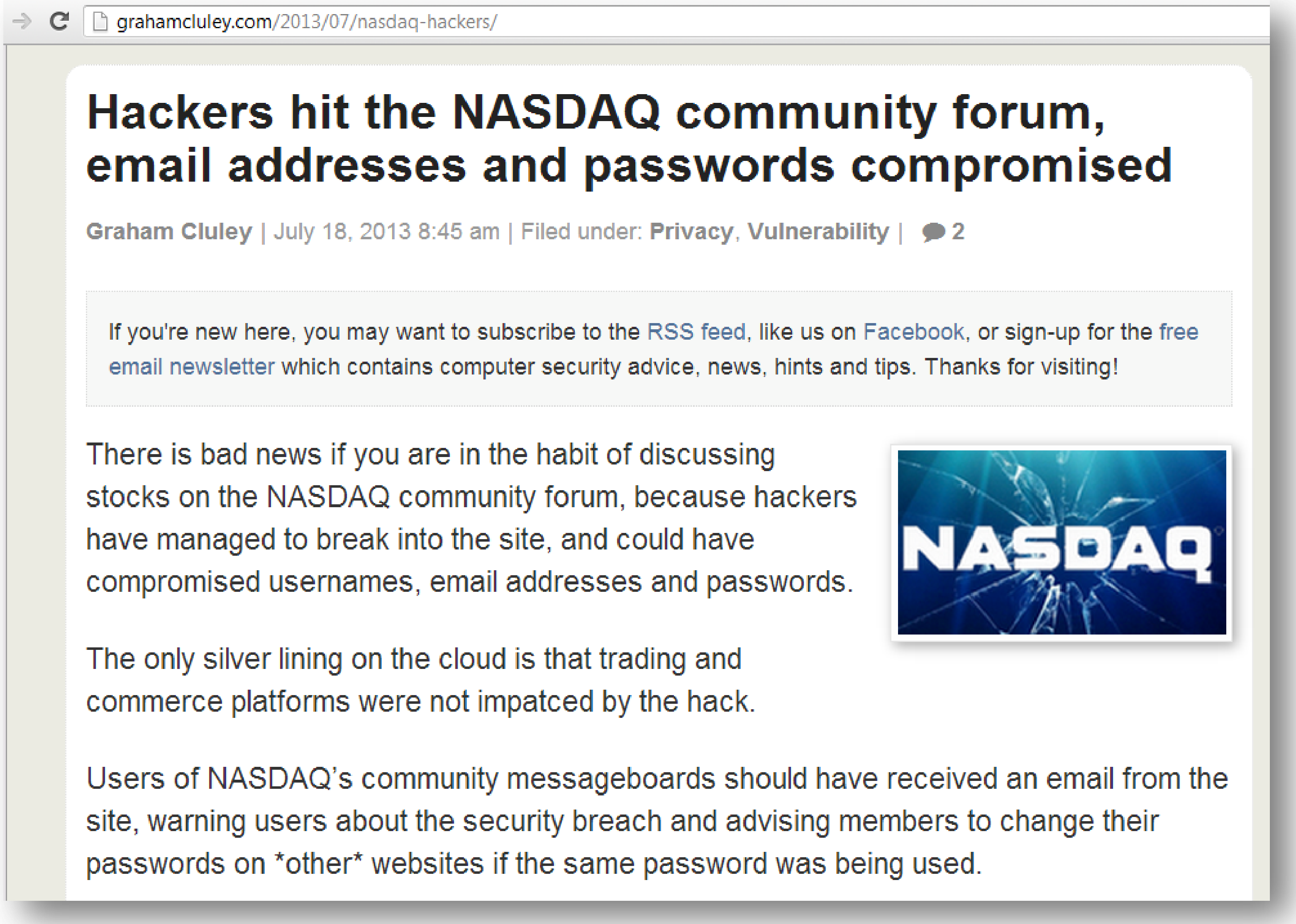


www.ign.com/blogs/retrocortana101/2013/07/13/bohemia-interactive-hacked-usernames-emails-and-encrypted-passwords-taken/

Bohemia Interactive hacked – usernames, emails and encrypted passwords taken

July 13, 2013 by RetroCortana101

July 18, 2013



grahamcluley.com/2013/07/nasdaq-hackers/

Hackers hit the NASDAQ community forum, email addresses and passwords compromised

Graham Cluley | July 18, 2013 8:45 am | Filed under: **Privacy**, **Vulnerability** | 2

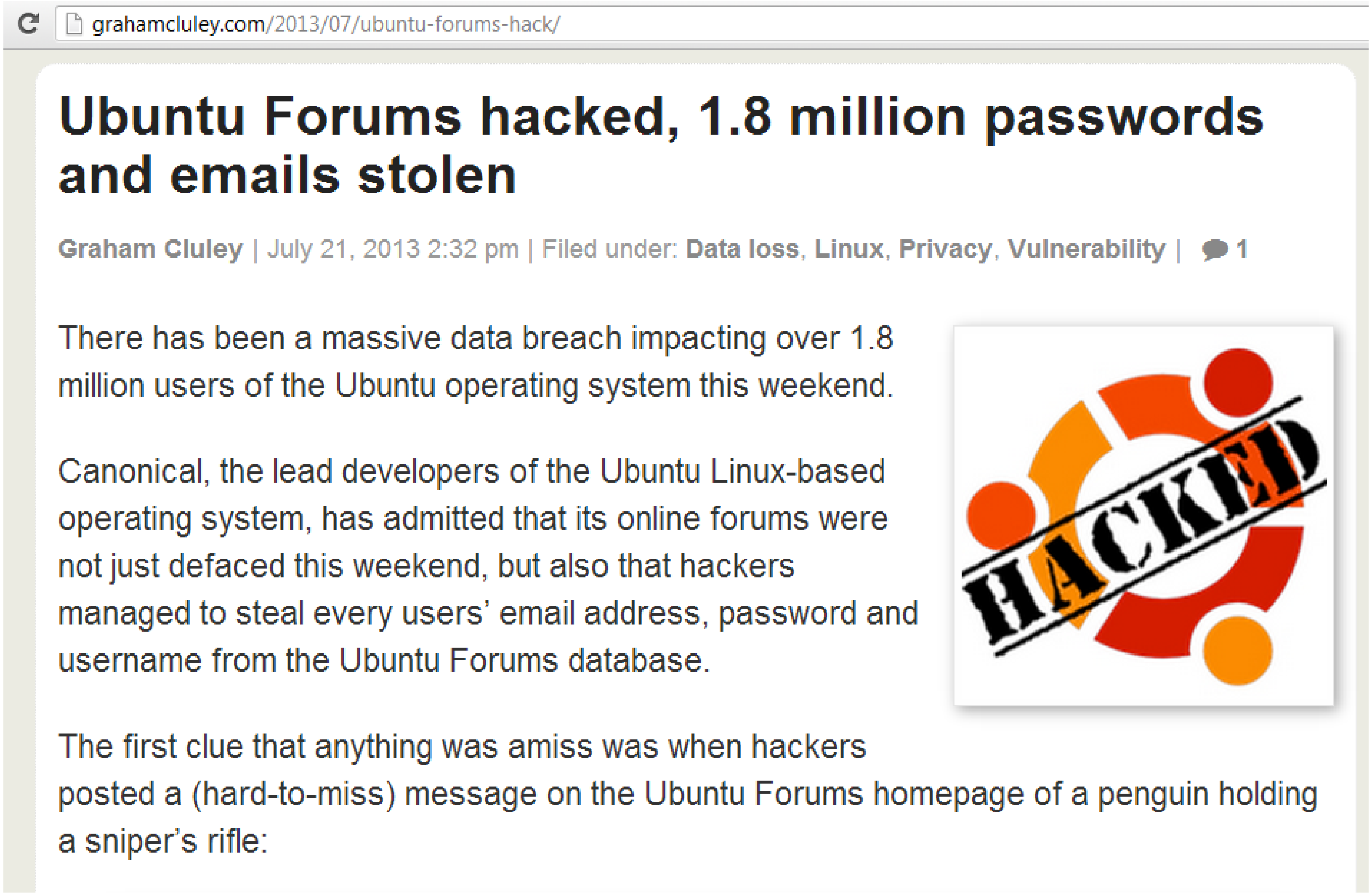
If you're new here, you may want to subscribe to the RSS feed, like us on Facebook, or sign-up for the free email newsletter which contains computer security advice, news, hints and tips. Thanks for visiting!

There is bad news if you are in the habit of discussing stocks on the NASDAQ community forum, because hackers have managed to break into the site, and could have compromised usernames, email addresses and passwords.

The only silver lining on the cloud is that trading and commerce platforms were not impatced by the hack.

Users of NASDAQ's community messageboards should have received an email from the site, warning users about the security breach and advising members to change their passwords on *other* websites if the same password was being used.

July 21, 2013



grahamcluley.com/2013/07/ubuntu-forums-hack/

Ubuntu Forums hacked, 1.8 million passwords and emails stolen

Graham Cluley | July 21, 2013 2:32 pm | Filed under: **Data loss**, **Linux**, **Privacy**, **Vulnerability** | **1**

There has been a massive data breach impacting over 1.8 million users of the Ubuntu operating system this weekend.

Canonical, the lead developers of the Ubuntu Linux-based operating system, has admitted that its online forums were not just defaced this weekend, but also that hackers managed to steal every users' email address, password and username from the Ubuntu Forums database.

The first clue that anything was amiss was when hackers posted a (hard-to-miss) message on the Ubuntu Forums homepage of a penguin holding a sniper's rifle:

Just use **scrypt**!

scrypt

1) Sequential initialization of a large array V

```
V[i] = H( V[i-1] ), i=0..N-1
```

scrypt

1) Sequential initialization of a large array V

```
V[i] = H( V[i-1] ), i=0..N-1
```

b83546b4					

scrypt

1) Sequential initialization of a large array V

```
V[i] = H( V[i-1] ), i=0..N-1
```

b83546b4	**b2e2a2f5**				

scrypt

1) Sequential initialization of a large array V

```
V[i] = H( V[i-1] ), i=0..N-1
```

b83546b4	b2e2a2f5	**10cbd82a**			

scrypt

1) Sequential initialization of a large array V

```
V[i] = H( V[i-1] ), i=0..N-1
```

b83546b4	b2e2a2f5	10cbd82a	...		

scrypt

1) Sequential initialization of a large array V

```
V[i] = H( V[i-1] ), i=0..N-1
```

b83546b4	b2e2a2f5	10cbd82a	...	**57500361**	

scrypt

1) Sequential initialization of a large array V

```
V[i] = H( V[i-1] ), i=0..N-1
```

b83546b4	b2e2a2f5	10cbd82a	...	57500361	**299c689f**

scrypt

1) Sequential initialization of a large array V

```
V[i] = H( V[i-1] ), i=0..N-1
```

b83546b4	**b2e2a2f5**	10cbd82a	...	57500361	299c689f

2) Sequential unpredictable accesses

```
X = H( X ⊕ V[ X mod N ] ), i=0..N-1
```

scrypt

1) Sequential initialization of a large array V

```
V[i] = H( V[i-1] ), i=0..N-1
```

b83546b4	b2e2a2f5	10cbd82a	...	57500361	**299c689f**

2) Sequential unpredictable accesses

```
X = H( X ⊕ V[ X mod N ] ), i=0..N-1
```

scrypt

1) Sequential initialization of a large array V

```
V[i] = H( V[i-1] ), i=0..N-1
```

b83546b4	b2e2a2f5	**10cbd82a**	...	57500361	299c689f

2) Sequential unpredictable accesses

```
X = H( X ⊕ V[ X mod N ] ), i=0..N-1
```

scrypt

1) Sequential initialization of a large array V

```
V[i] = H( V[i-1] ), i=0..N-1
```

b83546b4	b2e2a2f5	10cbd82a	...	57500361	299c689f

2) Sequential unpredictable accesses

```
X = H( X ⊕ V[ X mod N ] ), i=0..N-1
```

scrypt

1) Sequential initialization of a large array V

```
V[i] = H( V[i-1] ), i=0..N-1
```

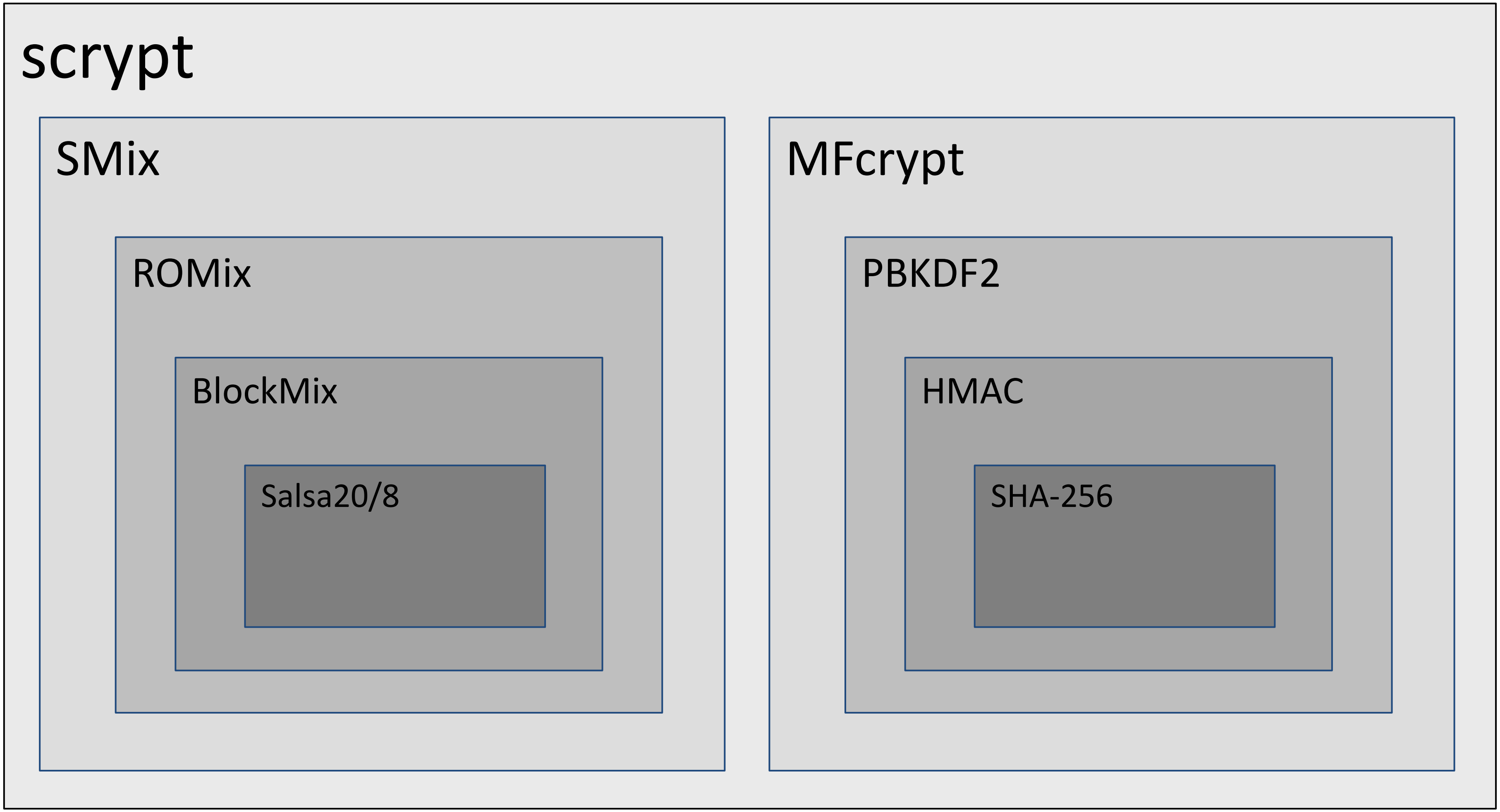
b83546b4	b2e2a2f5	10cbd82a	...	**57500361**	299c689f

2) Sequential unpredictable accesses

```
X = H( X ⊕ V[ X mod N ] ), i=0..N-1
```

is scrypt simple enough?

More core = more bugs = more tests, etc.



is scrypt user-friendly?

3 parameters:

- **N**: “Integer work metric”
- **r**: “Block size parameter”
- **p**: “Parallelization parameter” (**r** also affects parallelism)

Which parameters should one choose?

- Some recommendations in the 2009 paper, but different applications have different requirements

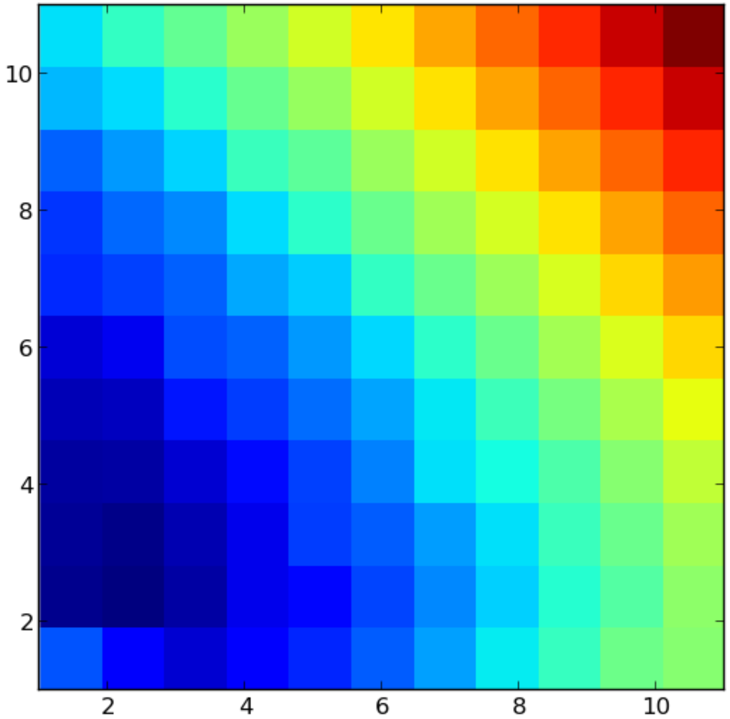
How are these *affecting scrypt performance*?

is scrypt user-friendly?

N and **r** have *similar effect* for the defender:

N×**r** basic operations

N×**r**×128 bytes of memory



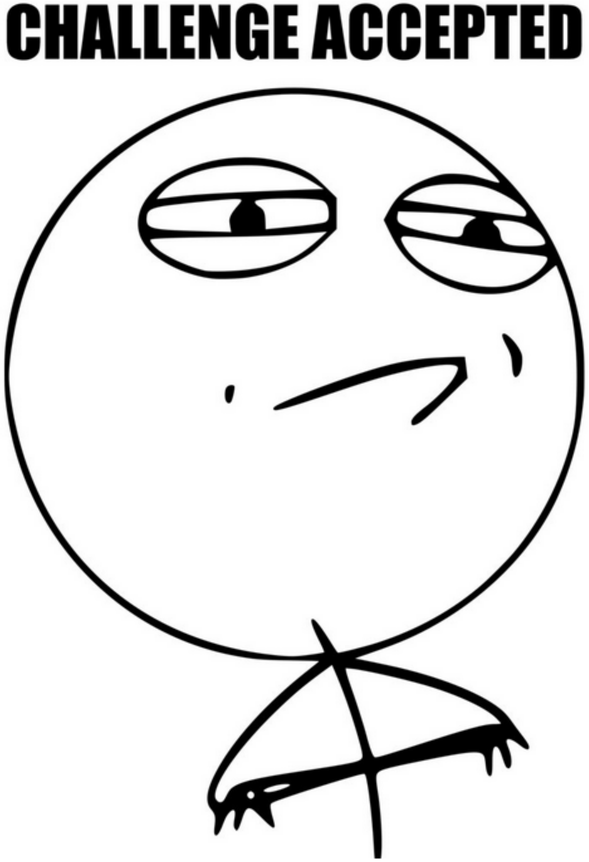
log(time) of scrypt with
X=log(**N**)
Y=log(**r**)
color range ~ 0.1 to 2000ms

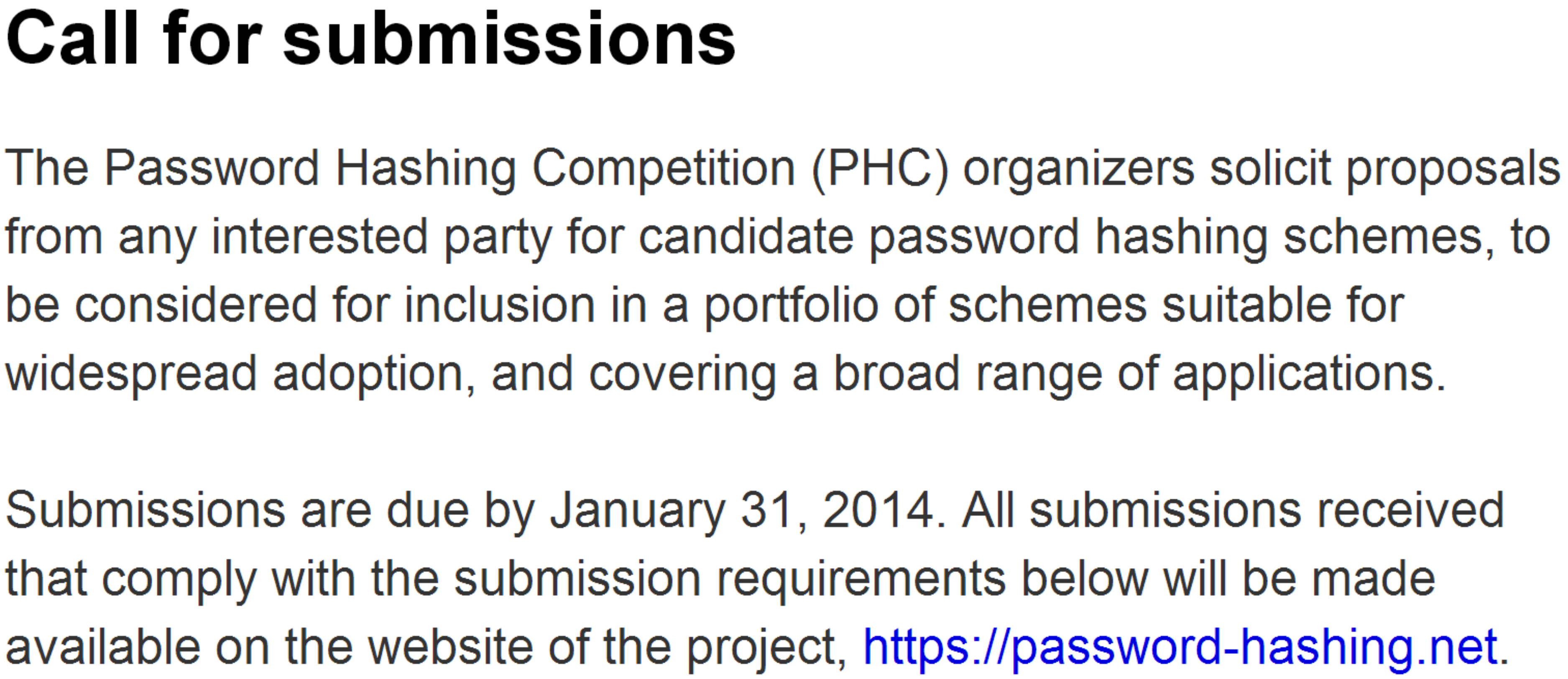
Impossible to increase only time (and not memory)

Potential problem for low-memory devices

Also impossible to increase only memory

We need something better


CHALLENGE ACCEPTED



Call for submissions

The Password Hashing Competition (PHC) organizers solicit proposals from any interested party for candidate password hashing schemes, to be considered for inclusion in a portfolio of schemes suitable for widespread adoption, and covering a broad range of applications.

Submissions are due by January 31, 2014. All submissions received that comply with the submission requirements below will be made available on the website of the project, https://password-hashing.net.

https://password-hashing.net/call.html

Minimal I/O requirements

- 0 to 128-byte password

  Encoding of characters to bytes is up to users

- 16-byte salt

  May support shorter and longer salts as well

- 1 cost parameter

  May support 2 or more (e.g. time and memory)

- 16-byte hash

  May support short and longer hashes as well

Evaluation criteria

Security and functionality

- Pseudorandom behavior
- Minimal speedup with crackers’ SW or HW
- Effectiveness of the cost parameters
- Flexibility and scalability
- Resilience to side-channel attacks

Evaluation criteria

Simplicity

- Often overlooked in “clever” schemes
- **Specs**: clarity, conciseness, number of components, prior knowledge, etc.
- **Implementation**: mapping from spec, support for existing instructions, etc.

“Complexity provides both opportunity and hiding places for attackers“ --Dan Geer

Design choices

- *Application*? (key derivation, storage…)
- *Platform*? (64-bit SW, mobile, low-end…)
- platform-*optimized vs generic*
- *Length*: do we need more than 16 bytes?
- How to implement "*memory hardness*"?
  - reads vs. writes; blocks size; predictability and order; etc.
  - prove rigorous bounds on time-memory-tradeoff?
- What degree and type of *parallelism*?

WARNING
CHALLENGES
AHEAD

Crypto research

Todos includes:

- Create generic constructions (like HMAC for MACs)
- Prove rigorous security bounds on time/memory
- Define minimal security requirements
- Dedicated hardware architectures?
- Cryptanalyze PHC candidates

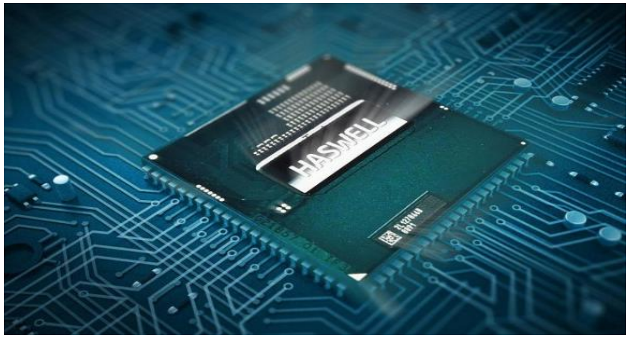
Optimization and technology-dependency

Password hashing is very *technology-dependent*

- For both defenders and attackers
- How will server chips look like in 10 years?
- What will be the most effective cracking method?

For example, hashes could be optimized for AVX2:

- 256-bit registers
- SIMD arithmetic
- Gather instructions
- VPERMD, VFM*, etc.



-› Better security for AVX2 servers, but inconsistent performance accross platforms…

Leakage resilience

Protection against the extraction of information from the *physical implementation* of a hashing scheme

Pure timing

If passwords of any length are supported, etc.

Cache timing

Password-dependent lookups in large tables, etc.

Memory leaks

Is it necessary to securely wipe the memory? etc.

Client-side hashing?

Should hashing be performed by the *clients*?

- For which application?
- Share effort between server and client?
- How to deal with diversity of client CPUs?
- Optimize a hash for JavaScript?

Addresses the *risk of DoS* on servers

Different models than store-and-compare-hashes?

Updatability

- How to update hashes to a different security level?
  - *Without requiring a new login*
  - Schemes based on a fast hash as a proxy?

- Motivations: adapt to new technology and research
  - Defenders (server CPU, cores available, etc.)
  - Attackers (hardware, techniques, etc.)

More ideas

Programmable hashes

- Algorithm = F(password)
- Defeats custom hardware
- ≈ Code generator for a custom VM
- Consistency? Interoperability?

Security through obesity (J. Spilman)

- Pollute the DB with dummy hashes
- Hide usernames from the DB
- Huge DB (e.g. 1TB) complicates download

Thank you!

See you at **Passwords13**-Norway!