haute école
neuchâtel berne jura

arc

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts
Western Switzerland

Bachelor Spring Project

HE-ARC 2016

Overclouds

Author:
Romain Claret

Supervisor:
Marc Schaefer

April 26, 2016

Abstract

Overclouds is a project whose goal is to create an anonymous and decentralized internet data sharing service right through the browser.

1 Description

1.1 English

The initiative behind the project is to create a new generation of internet data sharing tools, suited for today's paranoia for privacy on the internet and the preservation of knowledge for the next humanity generations.

The idea is to give the ability to the user to not rely on corporate servers, or farms of servers anymore. On Over Clouds, everybody and everything are now anonymous nodes, and they connect one to another freely and anonymously.

The network is a democratic mesh of nodes. The data is moving from a node to another across the network via other nodes and is ruled by the consensus of users.

We are aiming that users only need to have a standard Internet connection and a browser with JavaScript capabilities to use the service.

1.2 French

Must to the translation when the English part is validated

Contents

2 Preface

2.1 Introduction

TODO

2.2 The Big Picture

TODO

2.3 Objectives

TODO

2.4 Specifications

TODO

2.5 Management

TODO

2.6 State of the Art

TODO

2.6.1 Similar products (Existing Networks)

TODO

2.6.2 Transfer Protocols

TODO

2.6.3 Protection

TODO

2.6.4 Cryptography

TODO

2.6.5 Hardware

TODO

2.6.6 Block-Chains

TODO

2.6.7 Decentralized applications

TODO

2.6.8 Reputation Management

TODO

2.6.9 Operating Systems

TODO

2.6.10 Technologies

TODO

3 Analyses

3.1 Communication

TODO

3.2 Cryptography

As privacy is being an integral part of Overclouds. A research has begun to find the best type of cryptography done on the client-side, from the browser as the highest priority. We were looking for a fair middle between performance and security.

3.2.1 From Scratch VS Libraries

1st Question Is it possible and does it exists right from the browser?

We started looking at what is done in JavaScript and we found an interesting list of *premium* libraries (maintained by prestigious organizations) such as Stanford Javascript Crypto Library[11], MDN[7], W3C[10], Google Closure[2], or msrCrypto[8].

Then we looked at other crypto libraries such as forge[1], jsHashes[6], crypto-browserify[12], etc...

2nd Question The natural question that followed was: is it worth make it ourselves?

The answer came pretty quick while navigating across numerous forums. It's a pretty bad idea if we don't have a team dedicated to it and a pretty strong community to test it out. Even big companies such as Microsoft or Google are struggling a bit on the last part.

However, for the fun of it, we looked at solutions to start a homemade cryptography library. We found two interesting potential starting points to make it work with the browser, a Symmetric Encryption sample [5], or a procedure for Digital Signatures[4].

Decision Shortly after the second question, it was pretty clear that it won't be possible to create our own cryptography library in the time given. So we decided to find the *best* library out there for our project.

3.2.2 Comparison of some JavaScript Cryptography Libraries

Based on the following tables we can notice that **sjcl**[11], **crypto browserify**[12], and **forge**[1] algorithms have been optimized for defined objectives.

talk about the tables and graphs

See Table 1 Related Figures 1, 2, 3, 4

See Table 2 Related Figures 5, 6, 7, 8
See Table 3 Related Figures 9, 10

3.2.3 A killer

After taking time doing research about the above algorithms, we came across a pretty amazing algorithm: **BLAKE2**[3, 9].

BLAKE2 outperforms MD5, SHA-1, SHA-2, and SHA-3 on recent Intel CPUs and it has **no known** security issues. Plus SHA-1, MD5, and SHA-512 are susceptible to length-extension.

It is a *new* algorithm designed specifically for **performance** and is multifaceted **BLAKE2s** (optimized for 8to32-bit) and **BLAKE2b** (optimized for 64-bit)

3.2.4 Now what

If we look at the graphic 11, BLAKE2 is dominating the two best above. **rusha** is close behind it, and forge's *sha256*. Plus we can note that the curves display a nearly completely linear performance.

Also on at the table 4 with the figure 12, we notice that BLAKE2 is in its own category.

3.2.5 What about OverClouds

We can note that we are not really interested in SHA1, because of potential security flaws. SHA256 is much better for us. However, BLACK2 is pretty amazing. We will try to make it work in the following implementation.

Now, if it doesn't work for whatever reason, we will certainly go with forge, crypto-browserify, or sjcl. The problem with forge and crypto-browserify is that we must trust a company or an individual. With sjcl however, we trust an institution.

4 Implementations

4.1 Communication

TODO

4.2 Cryptography

TODO

5 Evaluation

5.1 Tests

TODO

5.2 Results

TODO

5.3 Technologies Recommendations

TODO

6 Conclusion

TODO

7 Bibliography

References

[1] Inc. Digital Bazaar. forge, 2016.

[2] Google. Closure Library, 2015.

[3] Jian Guo, Pierre Karman, Ivica Nikolić, Lei Wang, and Shuang Wu. Analysis of BLAKE2. *Springer International Publishing Switzerland 2014*, 8366(8366):402–423, 2014.

[4] Inc. Info Tech. Digital Signature in the Browser, 2014.

[5] Inc. Info Tech. Symmetric Encryption Sample, 2014.

[6] Paul Johnston. jsHashes, 2015.

[7] MDN. MDN Web API Crypto, 2015.

[8] Microsoft. MSR JavaScript Cryptography Library, 2015.

[9] Ed. Saarinen, M-J. and Jean Philippe Aumasson. The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC), 2015.

[10] Ryan Sleevi and Mark Watson. Web Cryptography API, 2014.

[11] Emily Stark, Michael Hamburg, and Dan Boneh. Symmetric Cryptography in Javascript, 2012.

[12] Dominic Tarr. Crypto-Browserify, 2013.

[13] Dominic Tarr. Performance of Hashing in Javascript Crypto Libraries., 2014.

8 Annexes

8.1 JS Cryptography Library Graphs

Add other crypto libraries using **Dominic Tarr**'s benchmark set [13].

The graphs from the following figures have been made by **Dominic Tarr** [13]

8.2 JS Cryptography Library Tables

Add other crypto libraries using **Dominic Tarr**'s benchmark set [13].

The following tables have been made based on **Dominic Tarr**'s[13] benchmarks.

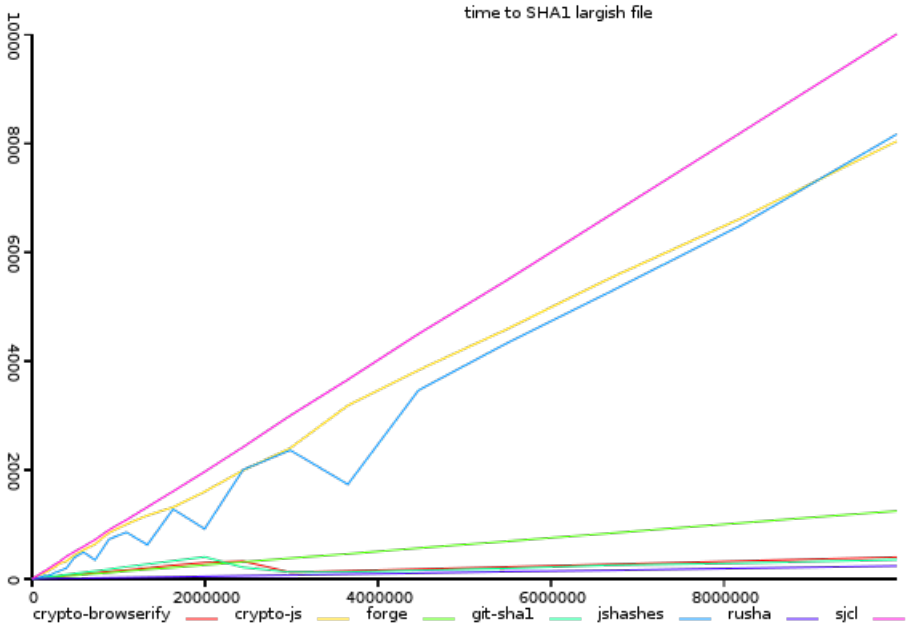

Figure 1: *y-axis shows total time taken, lower is better*

Table 1: Hashing 0-10MB Files /milliseconds based on [13]

Libraries	Sha1 (size)	Sha1 (hash)	Sha256 (size)	Sha256 (hash)
sjcl	- - - -	- - - -	- - - -	- - - -
crypto-js	- - -	- -	-	- - -
forge	+	+	+ + + +	+ + + +
crypto-browserify	+ +	+ +	+ + +	+ + +
crypto-mx	null	null	+	- -
git-sha1	+ + +	+ + +	null	null
jshashes	-	-	- -	- - -
rusha	+ + + +	+ + + +	null	null

Table 2: Key Derivation (pbkdf2) based on [13]

Libraries	Sha1 (time)	Sha1 (size)	Sha256 (time)	Sha256 (size)
sjcl	+ + + +	+ + + +	+ + + +	+ + + +
crypto-js	- - - -	- - - -	- - - -	- - - -
forge	+ + + +	+ +	+ + + +	+
crypto-browserify	+ + + +	+ +	+ + +	- -

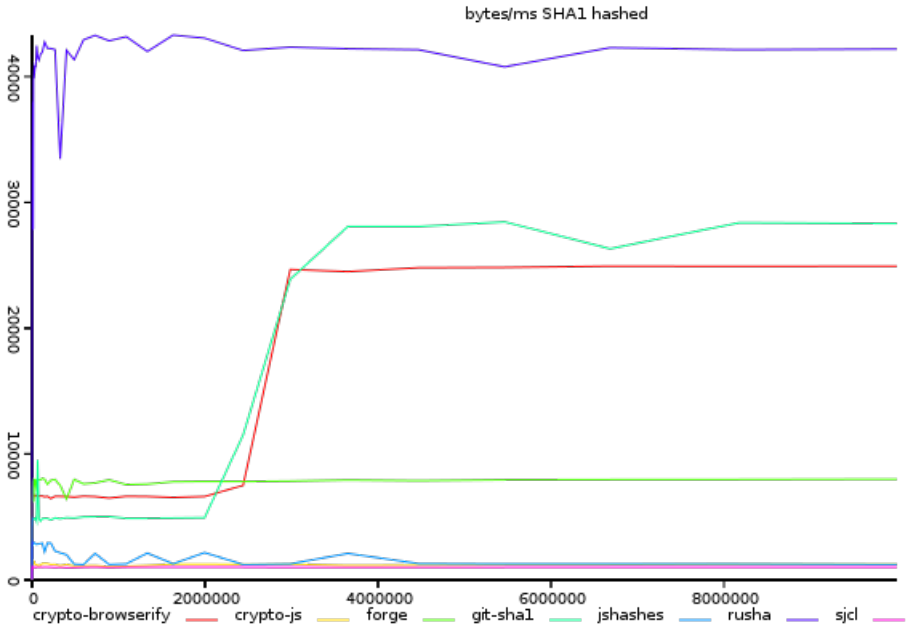

Figure 2: *y-axis shows size/time, higher is better*

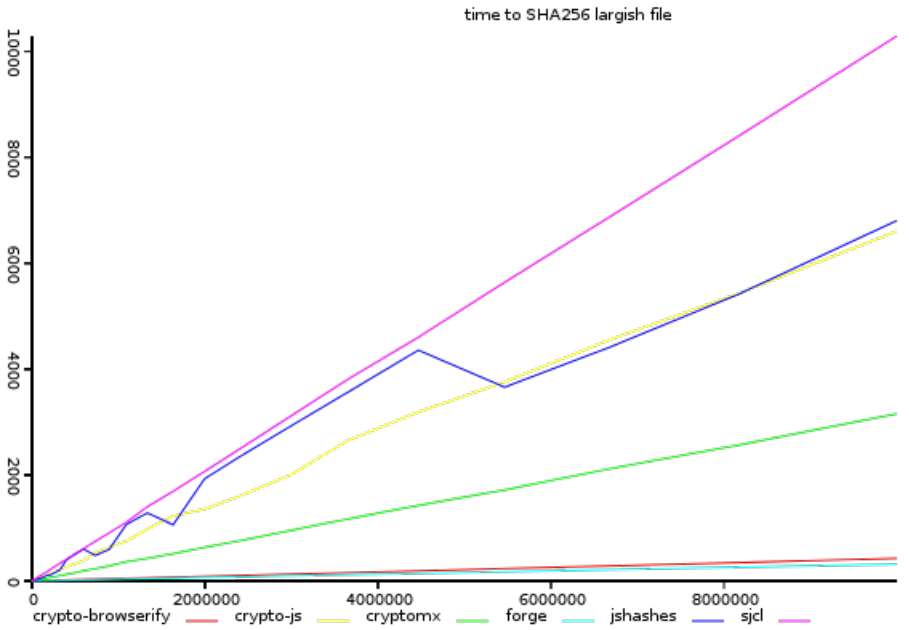

Figure 3: *y-axis shows total time taken, lower is better*

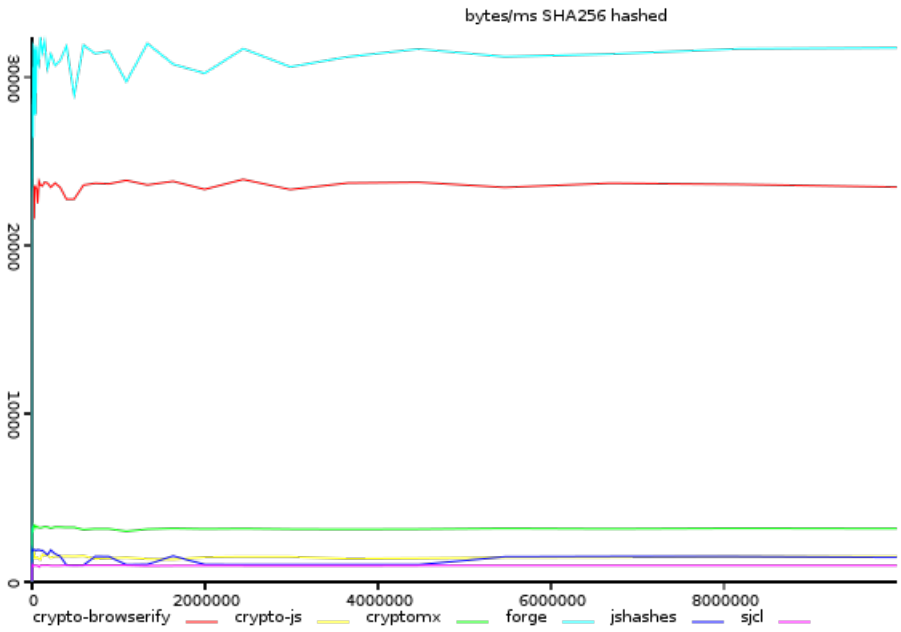

Figure 4: *y-axis shows size/time, higher is better*

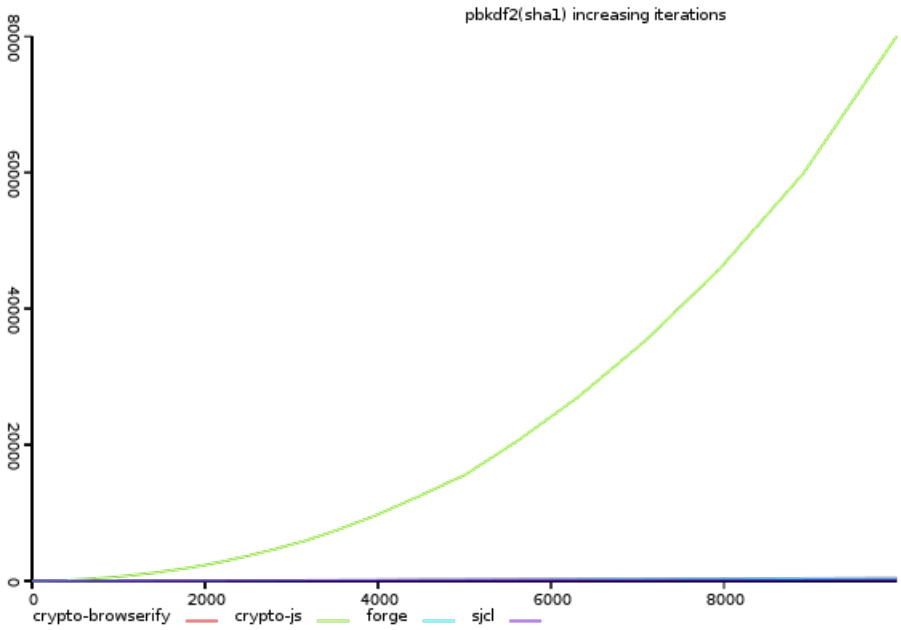

Figure 5: *y-axis shows total time taken, lower is better*

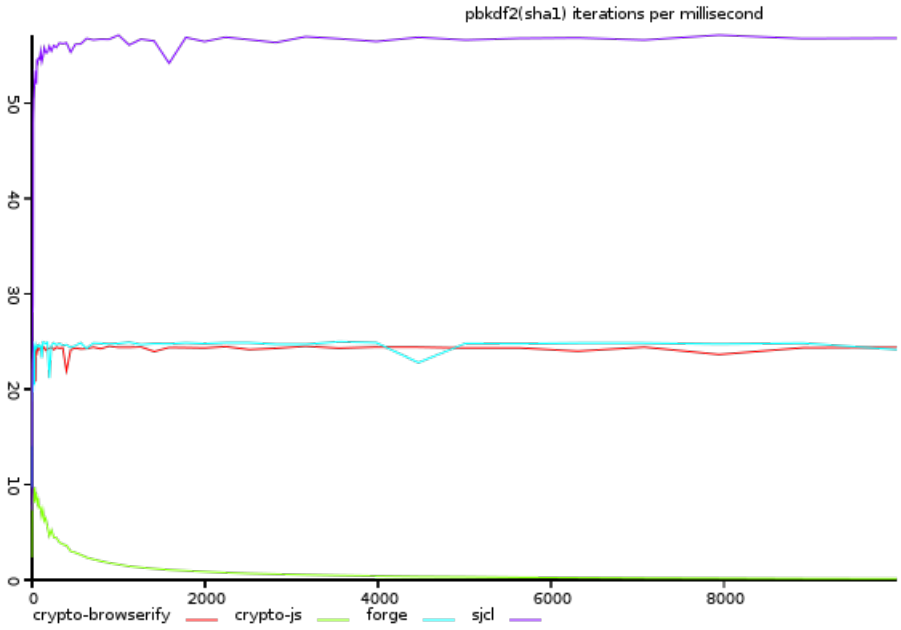

Figure 6: *y-axis shows size/time, higher is better*

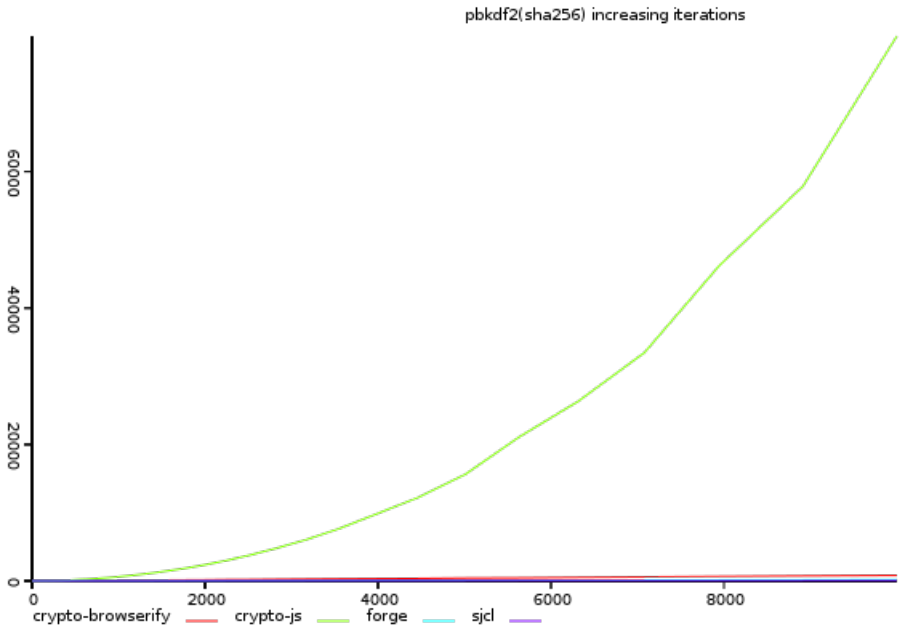

Figure 7: *y-axis shows total time taken, lower is better*

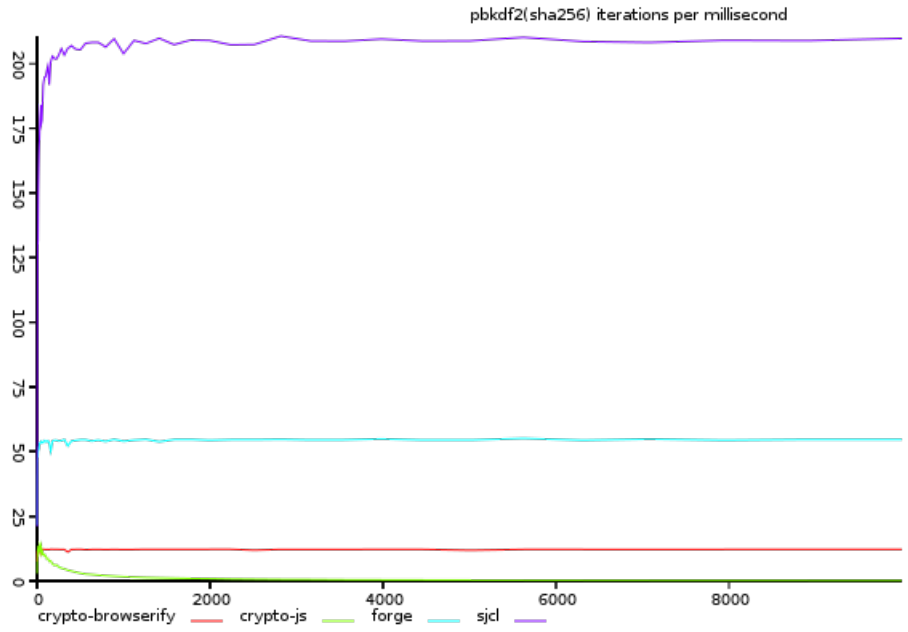

Figure 8: *y-axis shows size/time, higher is better*

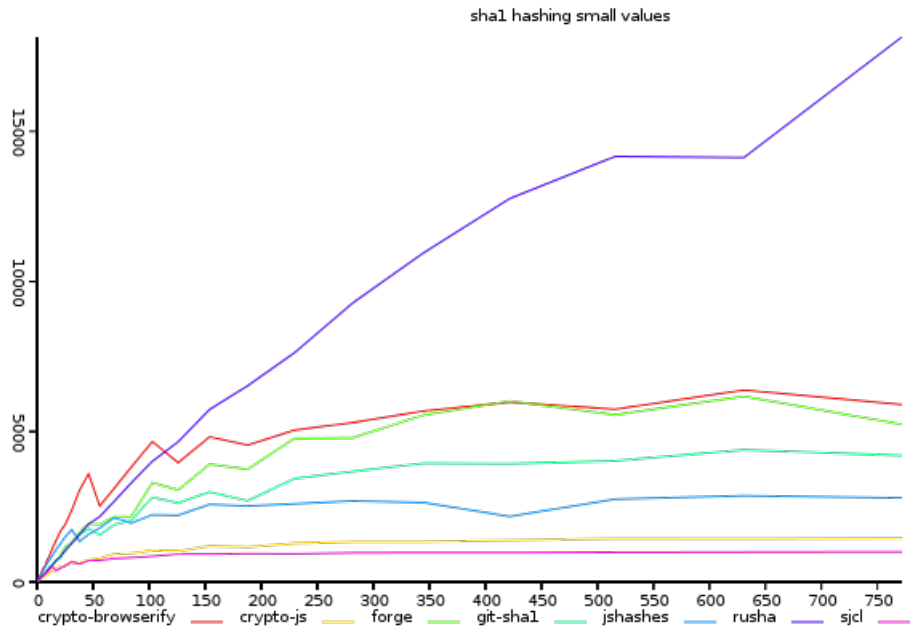

Figure 9: *y-axis shows size/time, higher is better*

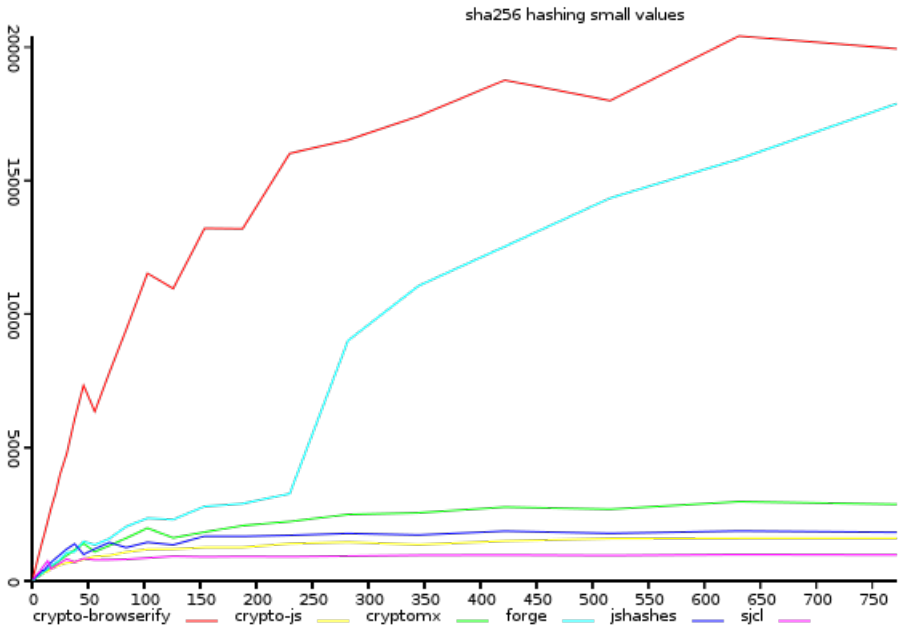

Figure 10: *y-axis shows size/time, higher is better*

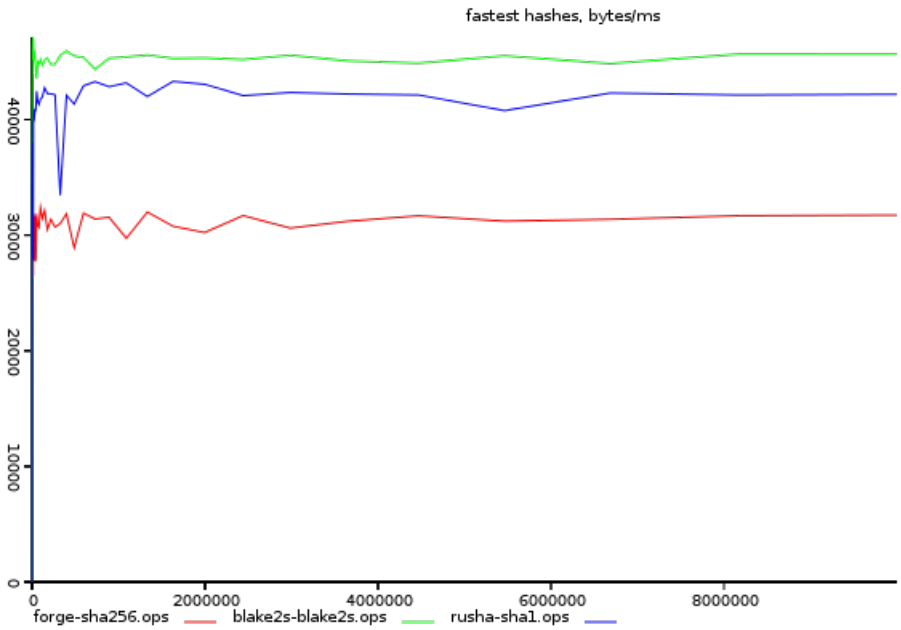

Figure 11: *y-axis size/time, higher is better*

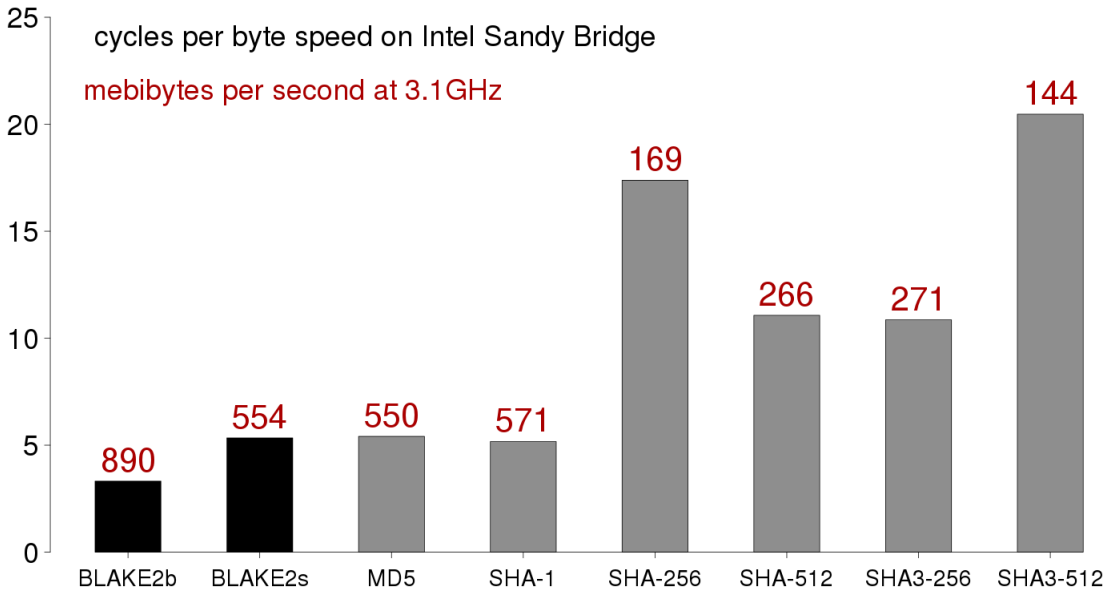

Figure 12: *lower is better*

Table 3: Hashing Small Files /milliseconds based on [13]

Libraries	Sha1 (size)	Sha256 (size)
sjcl	- - -	- - -
crypto-js	- -	- -
forge	+ +	+ + +
crypto-browserify	+ + +	+ + + +
crypto-mx	null	+
git-sha1	+	null
jshashes	-	-
rusha	+ + + +	null

Table 4: Fastest Hashes /milliseconds based on [13]

Libraries	Sha1 (size)	Sha256 (size)	blake2s (size)
rusha	+ + + +	null	null
forge	null	+ + + +	null
blake2s	null	null	+ + + +