Cryptographic Hash Functions

Christina Boura

SECRET Project Team Paris-Rocquencourt
Gemalto, France

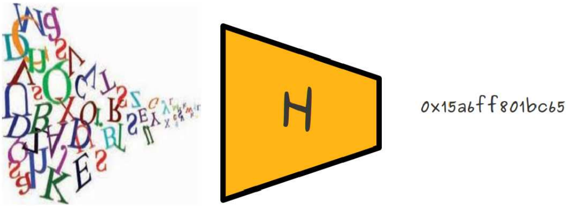
November 15, 2011

Outline

Outline

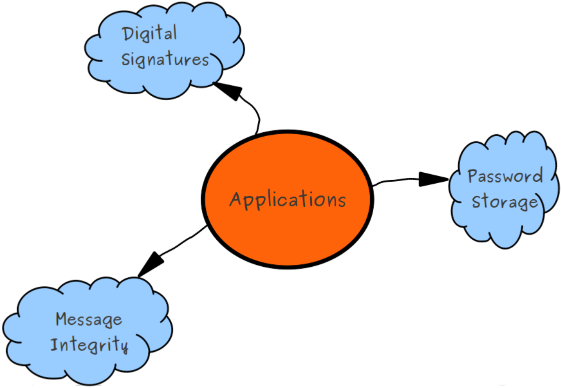
What is a hash function?

$H : \{0,1\}^* \rightarrow \{0,1\}^n$. No secret key!



The output, called **hash** or **digest** can be used as a fingerprint of the document.

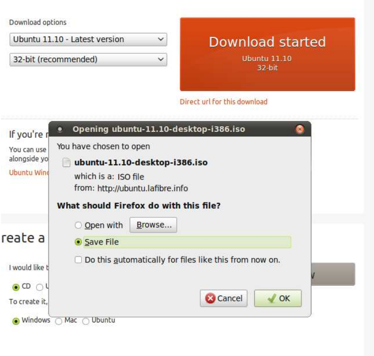
Using Hash Functions



Message Integrity

Verify the integrity of a downloaded software (no **errors during transmission**, no **corruption**).

Example: Check the integrity of a Linux distribution.



Message Integrity Example

Compute the hash (with MD5) of the downloaded distribution.

```
boura@satan: ~
File  Edit  View  Terminal  Help
boura@satan:~$ md5sum ubuntu-11.10-desktop-i386.iso
c396dd0f97bd122691bdb92d7e68fde5  ubuntu-11.10-desktop-i386.iso
boura@satan:~$
```

Message Integrity Example

Compute the hash (with MD5) of the downloaded distribution.

```
boura@satan: ~
File  Edit  View  Terminal  Help
boura@satan:~$ md5sum ubuntu-11.10-desktop-i386.iso
c396dd0f97bd122691bdb92d7e68fde5  ubuntu-11.10-desktop-i386.iso
boura@satan:~$
```

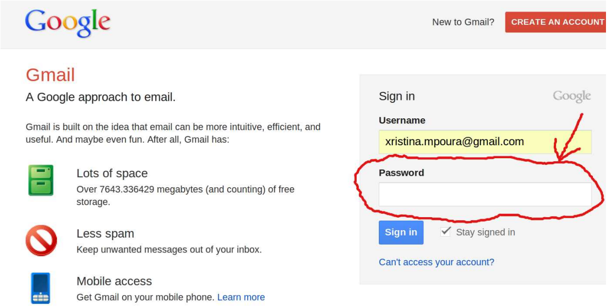
Compare this hash to the one given on the Ubuntu's web page.

11.10

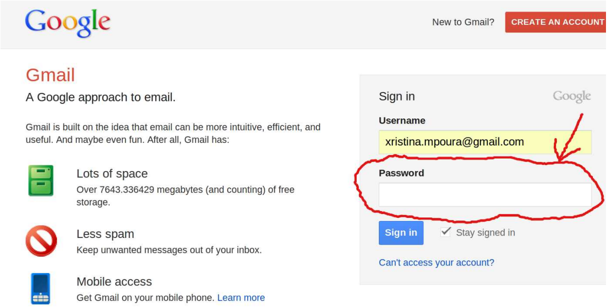
(Oneiric Ocelot: October 2011 (Supported until April 2013)

md5 Hash	*Version*
5e427f31e6b10315ada74094e8d5d483	ubuntu-11.10-alternate-amd64.iso
24da873c870d6a3dbfc17390dda52eb8	ubuntu-11.10-alternate-i386.iso
62fb5d750c30a27a26d01c5f3d8df459	ubuntu-11.10-desktop-amd64.iso
c396dd0f97bd122691bdb92d7e68fde5	ubuntu-11.10-desktop-i386.iso
f8a0112b7cb5dcd6d564dbe59f18c35f	ubuntu-11.10-server-amd64.iso
881d188cb1ca5fb18e3d9132275dceda	ubuntu-11.10-server-i386.iso
196f12bdbf60ee759e11d4986bc19ce3	ubuntu-11.10-wubi-amd64.tar.xz
7ab19159b48e2135f82d76193e1a0d55	ubuntu-11.10-wubi-i386.tar.xz

Password Storage



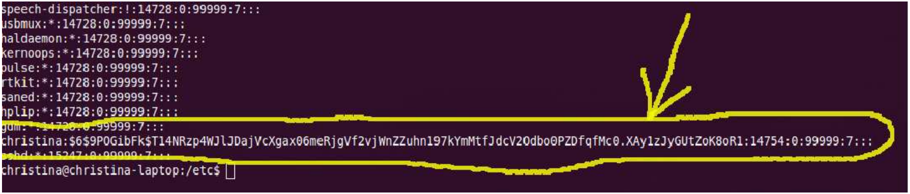
Password Storage



- The hash of the **password** is stored on the server.
- For every connection, the hash of the typed **password** is computed.
- If it corresponds to the stored one, then the connection is established.

Password Storage (Linux)

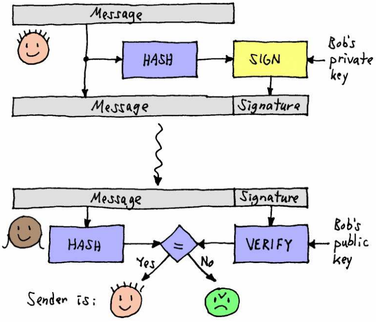
`etc/shadow`



Slow hash functions wanted for this application !

Digital Signatures

Prove the authenticity of a digital document.



Cryptographic Hash Function Properties

For the shown applications, a "simple" hash function is not enough . Additional security properties are needed. Let H be a hash function with n bits of output.

- **Preimage resistance**

  Given a hash h, it must be *computationally infeasible* to find a message m such that $H(m) = h$.

Example: Password Storage
Time complexity of the generic attack: $\mathcal{O}(2^n)$

Cryptographic Hash Functions-Properties

- **Second-preimage resistance**

  Given a message m, it must be *computationally infeasible* to find a message $m' \neq m$, such that $H(m') = H(m)$.

  Examples: Data integrity, Digital signatures
  Time complexity of the generic attack: $\mathcal{O}(2^n)$

- **Collision Resistance**

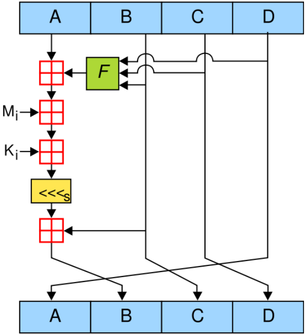
  It must be *computationally infeasible* to find two messages m, m', with $m' \neq m$ such that $H(m') = H(m)$.

  Examples: Data integrity, Digital signatures
  Time complexity of the generic attack: $\mathcal{O}(2^{\frac{n}{2}})$ (Birthday paradox)

What a hash function looks like...

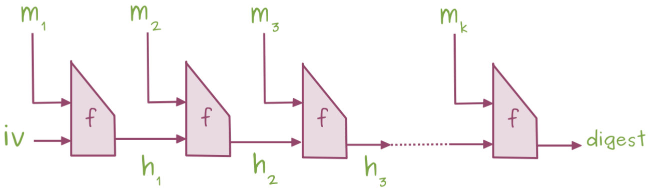
The **MD5** hash function. [Rivest91]

- $\boldsymbol{A, B, C, D}$: 32-bit registers.
- $F(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z)$
- $\wedge, \vee, \neg$ denote **AND, OR** and **NOT** operations respectively.
- Modular additions, rotations.
- Repeat **65** times.



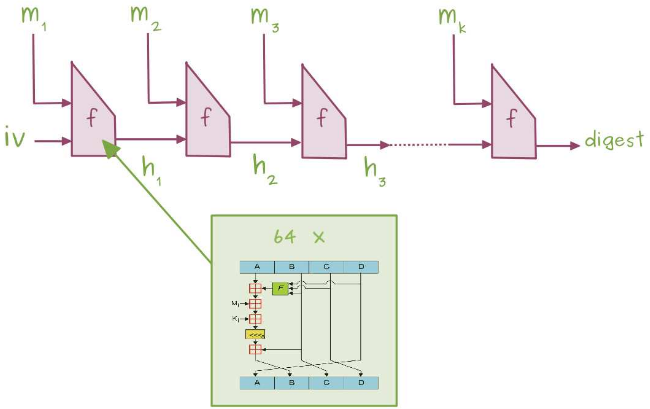
Treating large messages (MD construction)

Split the message into fixed length blocks.
Example : The Merkle-Damgård construction [Merkle-Damgård 89]



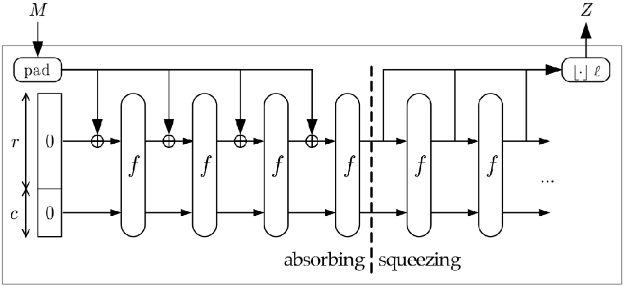
Treating large messages (MD construction)

Split the message into fixed length blocks.
Example : The Merkle-Damgård construction [Merkle-Damgård 89]

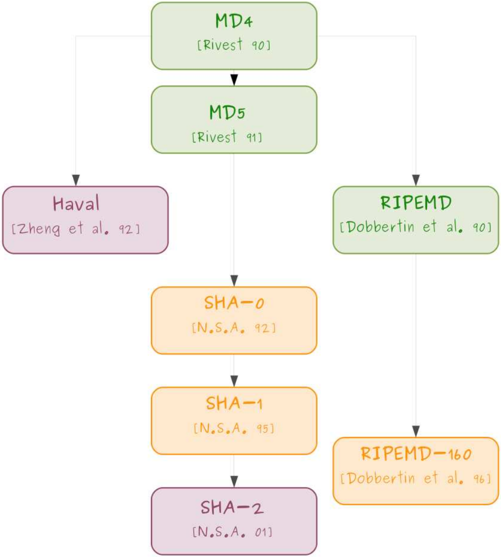


Treating large messages (the sponge construction)

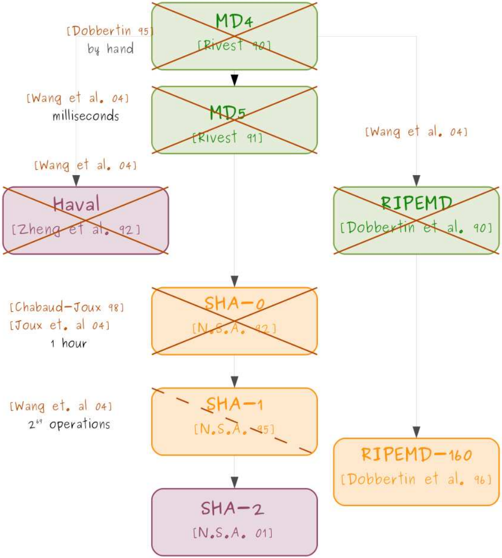
Sponge construction [Bertoni-Daemen-Peeters-Van Assche 07]



Commonly used hash functions



Commonly used hash functions



Outline

The NIST SHA-3 competition

Public Competition launched by the **National Institut of Standards and Technology (NIST)**.

SHA-3 Timeline

- October 2008: **64** candidates received.
- December 2008: **51** candidates accepted for the 1st round.
- July 2009: **14** candidates accepted for the 2nd round.
- December 2010: **5** finalists selected.
- 2012: Selection of the winner.

SHA-3 Timeline

- October 2008: **64** candidates received.
- December 2008: **51** candidates accepted for the 1st round.
- July 2009: **14** candidates accepted for the 2nd round.
- December 2010: **5** finalists selected.
- 2012: Selection of the winner.

- Three candidates submitted from INRIA : **FSB, Shabal** (SECRET project team), **SIMD** (Cascade project team).
- No French candidate among the 5 finalists...

Outline

Security Evaluation of Hash Functions

The **new hash standard** must be secure.
No weaknesses must have been found after 4 years of public evaluation.

Possible Weaknesses :

- **Attacks :** Finding preimages, second preimages or collisions. (theoretical or practical)
- **Distinguishers on the internal components.**

Security Evaluation of Hash Functions

The **new hash standard** must be secure.
No weaknesses must have been found after 4 years of public evaluation.

Possible Weaknesses :

- **Attacks :** Finding preimages, second preimages or collisions. (theoretical or practical)
- **Distinguishers on the internal components.**

A distinguisher may be the starting point for some attacks, can invalidate the security proofs...

Security Evaluation of Hash Functions

The **new hash standard** must be secure.
No weaknesses must have been found after 4 years of public evaluation.

Possible Weaknesses :

- **Attacks :** Finding preimages, second preimages or collisions. (theoretical or practical)
- **Distinguishers on the internal components.**

A distinguisher may be the starting point for some attacks, can invalidate the security proofs...
and not only this !

Distinguishers permit to gain Belgian beers !

Congratulations to the winners of the third KECCAK cryptanalysis prize

16 February 2010

We are happy to announce that **Christina Boura** and **Anne Canteaut** are the winners of the third KECCAK cryptanalysis prize for their paper entitled *A zero-sum property for the KECCAK-f permutation with 18 rounds*. We are currently arranging practical details with the winners to give them the awarded Lambic-based beers and book. *Congratulations to them!*

The algebraic degree of a hash function

The structure of many hash functions is based on **iterated** permutations. A permutation can be seen as a vectorial function over $\mathbf{F}_2^n$.

Example:

$$F(x_0, x_1, x_2) = (x_0 + x_1, x_2 x_3 + x_1, x_0 x_1 + 1).$$

The **algebraic degree** of a vectorial function, is the maximum algebraic degree of its coordinates. (*multivariate degree*)

The algebraic degree of a hash function

The structure of many hash functions is based on **iterated** permutations.
A permutation can be seen as a vectorial function over $\mathbf{F}_2^n$.

Example:

$$F(x_0, x_1, x_2) = (x_0 + x_1, x_2 x_3 + x_1, x_0 x_1 + 1).$$

The **algebraic degree** of a vectorial function, is the maximum algebraic degree of its coordinates. (*multivariate degree*)

In this example : $\deg(F) = 2$.

Our Contribution

Question : What is the algebraic degree of F after m rounds ?

Our work :

- **New bounds** on the algebraic degree of different classes of iterated permutations.
- Distinguishers for many **SHA-3** candidates.

Our Contribution

Question : What is the algebraic degree of F after m rounds ?

Our work :

- **New bounds** on the algebraic degree of different classes of iterated permutations.
- Distinguishers for many **SHA-3** candidates.

Thank you for your attention !