Firme Digitali Basate Su Funzioni Hash

Tommaso Gagliardoni[1]

[1] **Kudelski Security, Switzerland**

PQCifris May 9th, 2019

Rome, Italy

Overview of This Talk

Overview of This Talk

- What is a hash function?

Overview of This Talk

- What is a hash function?
- How to build hash-based digital signatures?

Overview of This Talk

- What is a hash function?
- How to build hash-based digital signatures?
- Why is that interesting?

What is a hash function?

What is a hash function?

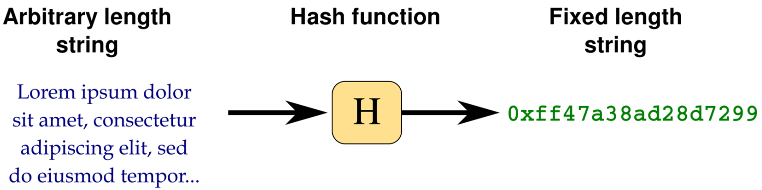
Definition

A *hash functions family* is a tuple $(X, Y, \{f_\lambda\}_\lambda)$ indexed by a parameter $\lambda \in \mathbb{N}$ such that $f_\lambda : X \to Y$ and blah blah blah...

What is a hash function?

Definition

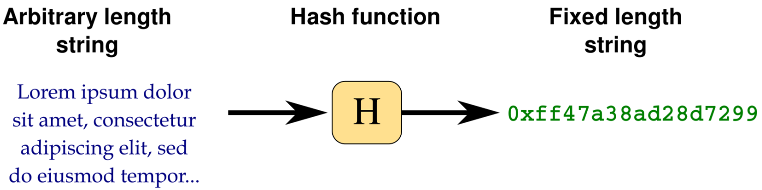
A *hash functions family* is a tuple $(X, Y, \{f_\lambda\}_\lambda)$ indexed by a parameter $\lambda \in \mathbb{N}$ such that $f_\lambda : X \to Y$ and blah blah blah...



What is a hash function?

Definition

A *hash functions family* is a tuple $(X, Y, \{f_\lambda\}_\lambda)$ indexed by a parameter $\lambda \in \mathbb{N}$ such that $f_\lambda : X \to Y$ and blah blah blah...



...with certain properties...

What is a hash function?

First preimage resistance (one-wayness)

Given y, it is hard to find x such that $H(x) = y$

Second preimage resistance

Given x, it is hard to find $x' \neq x$ such that $H(x) = H(x')$

Collision resistance

It is hard to find x and $x' \neq x$ such that $H(x) = H(x')$

What is a hash function?

First preimage resistance (one-wayness)

Given y, it is hard to find x such that $H(x) = y$

Second preimage resistance

Given x, it is hard to find $x' \neq x$ such that $H(x) = H(x')$

Collision resistance

It is hard to find x and $x' \neq x$ such that $H(x) = H(x')$

But remember: collision resistance is a much stronger assumption.

How to build hash-based digital signatures?

How to build hash-based digital signatures?

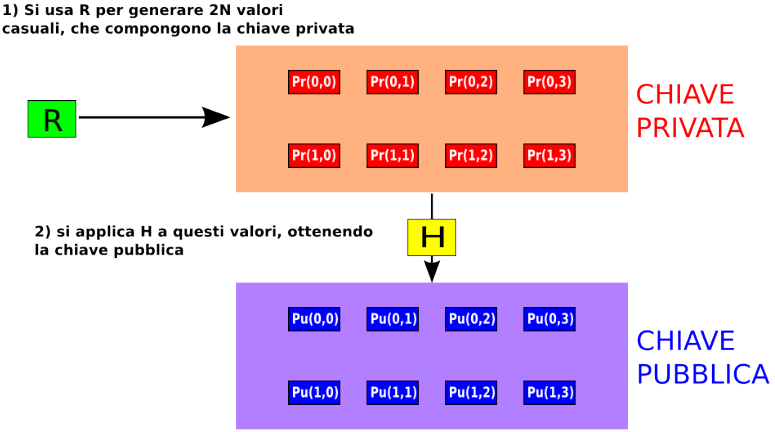
In the following we assume the use of a N-bit output hash function H (we use $H = 4$ as a didactical example).

We do not specify for now which properties we require from H.

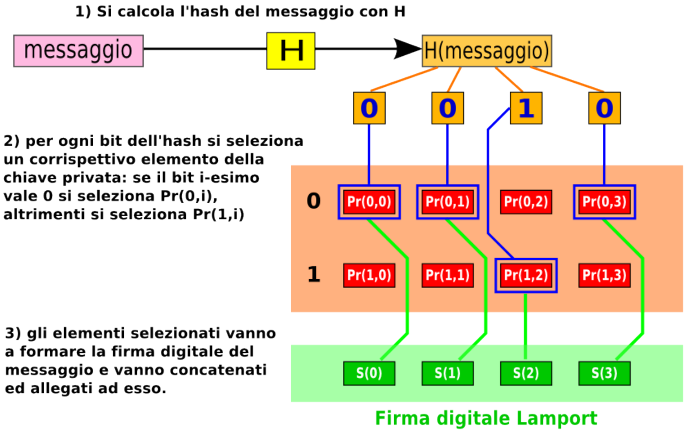
We also use a cryptographically secure source of randomness R

We now introduce the Lamport one-time signature scheme (1979)

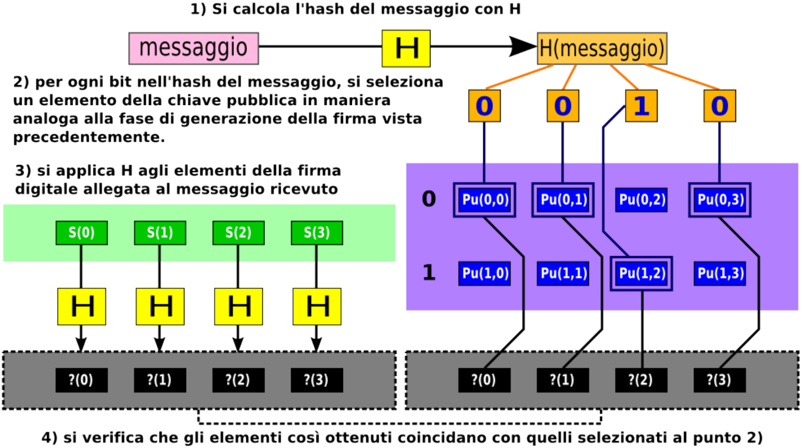
Key Generation



Signature Generation

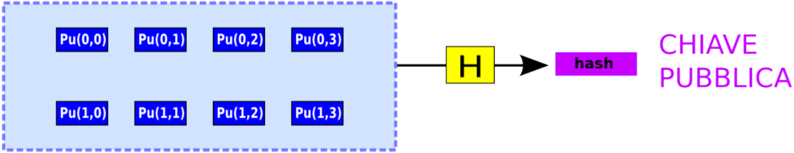


Signature Verification

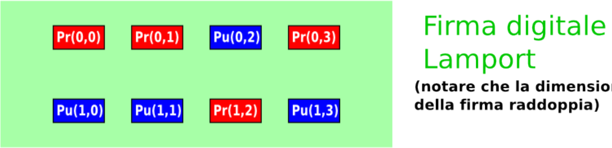


A possible optimization

Invece di pubblicare come chiave pubblica tutti i 2N valori Pu(i,j) se ne pubblica solo l'hash



A questo punto però bisogna includere nella firma Lamport, oltre ai valori Pr(i,j) selezionati col solito metodo, anche i valori Pu(i,j) non utilizzati. La firma diventa cioè del tipo:



(notare che la dimensione della firma raddoppia)

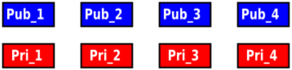
Per verificare la firma bisognerà prima trasformare i Pr(i,j) in Pu(i,j) (il verificatore può farlo tramite il solito processo di selezione basato sull'hash del messaggio) e poi verificare che l'hash di tutti i blocchi così ottenuti coincida con la chiave pubblica nota.

How to build hash-based digital signatures?

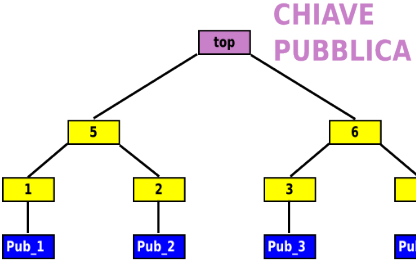
- very fast
- easy to implement
- large signatures (e.g. for SHA-256: 16 KiB vs. 256 B for RSA-2048)
- the key can only be used once

Merkle-Lamport scheme (1989)

Supponiamo di volere una chiave che possa fimare fino a 4 documenti diversi. Generiamo allora 4 diverse coppie di chiavi pubblica/privata:



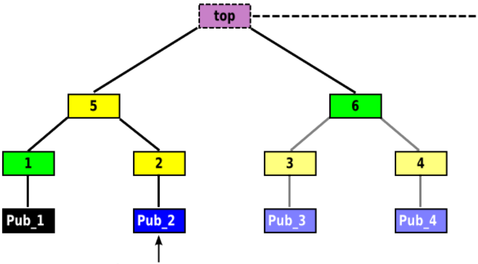
Ora costruiamo un Hash Tree (o Merkle Tree, in questo caso binario) usando come foglie gli hash delle chiavi pubbliche generate. Il "top hash" dell'albero sarà la vera chiave pubblica.



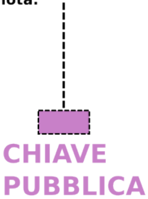
How to build hash-based digital signatures?

Questa struttura può ora essere usata per firmare fino a 4 documenti: ogni volta dovremo scegliere una tra le 4 chiavi ed usarla per generare una firma. Ognuna delle 4 chiavi è monouso. Per generare una firma dovremo allegare al messaggio, oltre agli hash dati dalla normale procedura di firma già vista, alcuni nodi intermedi dell'albero, in modo da poter fornire al verificatore le informazioni necessarie a risalire (e controllare) al top hash.

Supponiamo ad esempio di aver già "bruciato" la chiave 1, e di voler firmare un messaggio con la chiave 2. Allora oltre alla firma generata dalla chiave 2 dovremo allegare al messaggio i nodi marcati in verde:



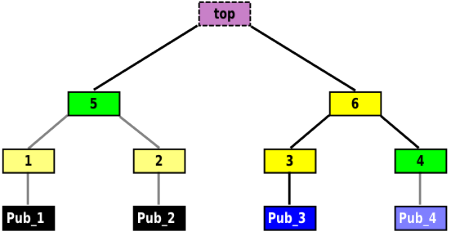
La firma è valida se il top hash ricavato coincide con la chiave pubblica nota.



(il verificatore può ricavare Pub_2 seguendo il metodo standard di verifica)

How to build hash-based digital signatures?

Altro esempio: supponiamo di aver "bruciato" le chiavi 1 e 2 e di voler usare la chiave 3:



Vantaggi: si può pregenerare una chiave con un numero arbitrariamente alto di utilizzi (ricordiamo tralaltro che sarebbe buona usanza far "scadere" le chiavi dopo un certo periodo)

Svantaggi: la dimensione della firma aumenta leggermente.

How to build hash-based digital signatures?

- at the base of every other hash-based signature variant
- other optimizations possible
- in particular: Winternitz scheme trades off signature size for speed
- state of the art: XMSS scheme
- all these approaches are stateful (troubles for: backups, multiple devices, load balancing...)
- SPHINCS+ (NIST submission, current 2nd round) eliminates the state

XMMS

C Implementation, using OpenSSL [HRS16]

	Sign (ms)	Signature (kB)	Public Key (kB)	Secret Key (kB)	Bit Security classical/ quantum	Comment
XMSS	3.24	2.8	1.3	2.2	236 / 118	h = 20, d = 1,
XMSS-T	9.48	2.8	**0.064**	2.2	**256 / 128**	h = 20, d = 1
XMSS	3.59	8.3	1.3	14.6	196 / 98	h = 60, d = 3
XMSS-T	10.54	8.3	**0.064**	14.6	**256 / 128**	h = 60, d = 3

Intel(R) Core(TM) i7 CPU @ 3.50GHz
All using SHA2-256, w = 16 and k = 2

SPHINCS+

	n	h	d	$\log(t)$	k	w	bitsec	sec level	sig bytes
SPHINCS$^+$-128s	16	64	8	15	10	16	133	**1**	8 080
SPHINCS$^+$-128f	16	60	20	9	30	16	128	**1**	16 976
SPHINCS$^+$-192s	24	64	8	16	14	16	196	**3**	17 064
SPHINCS$^+$-192f	24	66	22	8	33	16	194	**3**	35 664
SPHINCS$^+$-256s	32	64	8	14	22	16	255	**5**	29 792
SPHINCS$^+$-256f	32	68	17	10	30	16	254	**5**	49 216

https://sphincs.org

Why is that interesting?

Why is that interesting?

Quantum resistance:

Why is that interesting?

Quantum resistance:

- quantum algorithms like Shor break number-theoretic assumptions (RSA, DH, etc) with speedup from (sub)exponential to polynomial ($\approx$ degree 3)

Why is that interesting?

Quantum resistance:

- quantum algorithms like Shor break number-theoretic assumptions (RSA, DH, etc) with speedup from (sub)exponential to polynomial ($\approx$ degree 3)
- however these attacks do not apply to hash functions. The best generic attack is given by Grover search algorthm (and variants)

Why is that interesting?

Quantum resistance:

- quantum algorithms like Shor break number-theoretic assumptions (RSA, DH, etc) with speedup from (sub)exponential to polynomial ($\approx$ degree 3)
- however these attacks do not apply to hash functions. The best generic attack is given by Grover search algorthm (and variants)
- security in these cases is well understood: query complexity speedup is only quadratical (actually, only $O(N^{\frac{1}{2}})$ VS $O(N^{\frac{1}{3}})$ for finding collisions)

Why is that interesting?

Quantum resistance:

- quantum algorithms like Shor break number-theoretic assumptions (RSA, DH, etc) with speedup from (sub)exponential to polynomial ($\approx$ degree 3)
- however these attacks do not apply to hash functions. The best generic attack is given by Grover search algorthm (and variants)
- security in these cases is well understood: query complexity speedup is only quadratical (actually, only $O(N^{\frac{1}{2}})$ VS $O(N^{\frac{1}{3}})$ for finding collisions)
- these bounds are provable and hold even if $P = NP$ (caveat: but they hide many details)

Why is that interesting?

But what if H is vulnerable? (e.g.: SHA-1)

Why is that interesting?

But what if H is vulnerable? (e.g.: SHA-1)

- H can be drop-in replaced in a black-box way

Why is that interesting?

But what if H is vulnerable? (e.g.: SHA-1)

- H can be drop-in replaced in a black-box way
- (if someone breaks hash function X we can just replace it with Y, we have not such an option with tools like RSA, where we need every time to find new suitable hard mathematical problems)

Why is that interesting?

But what if H is vulnerable? (e.g.: SHA-1)

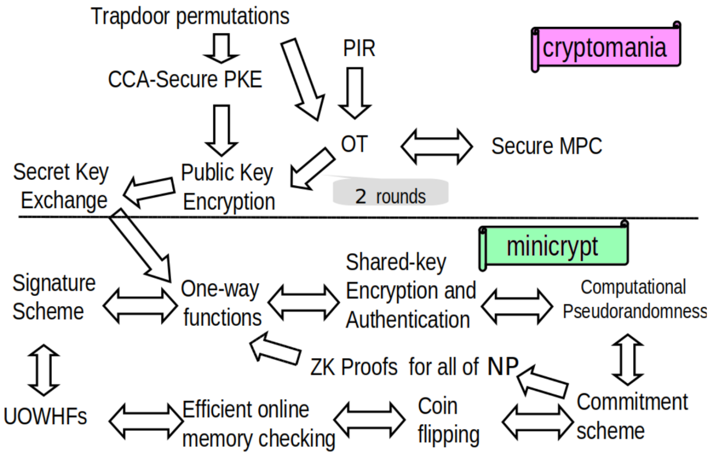
- H can be drop-in replaced in a black-box way
- (if someone breaks hash function X we can just replace it with Y, we have not such an option with tools like RSA, where we need every time to find new suitable hard mathematical problems)
- most importatly: we need hash functions anyway for any real-world cryptographic application!

BONUS QUESTION

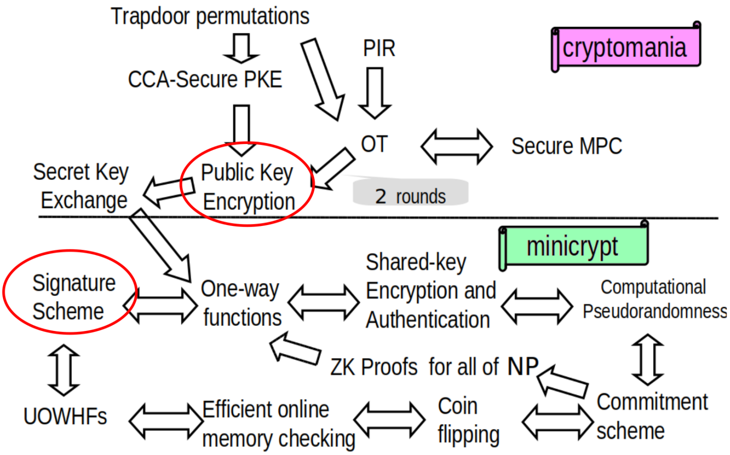
Why only signatures?

(this is a very interesting question)

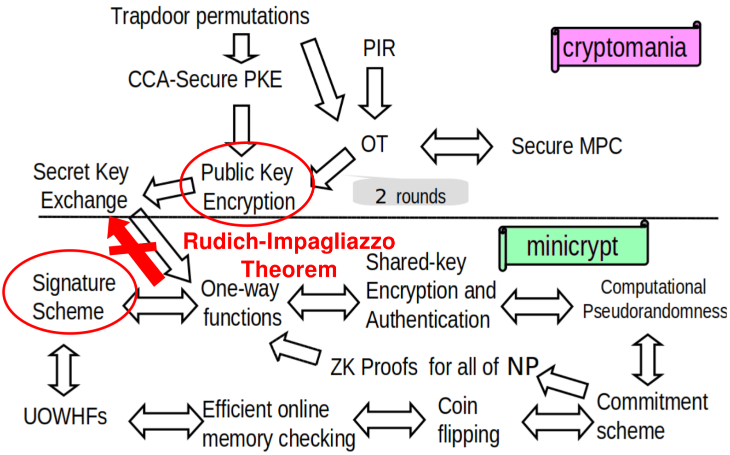
BONUS QUESTION



BONUS QUESTION



BONUS QUESTION



A Matter of Theory

- In Cryptomania much more cryptographic tasks are possible, but they require stronger assumptions

A Matter of Theory

- In Cryptomania much more cryptographic tasks are possible, but they require stronger assumptions
- The stronger the assumptions we need for our cryptography, the more likely it is that they do not hold (i.e. someone might find a way a way to solve the underlying problem)

A Matter of Theory

- In Cryptomania much more cryptographic tasks are possible, but they require stronger assumptions
- The stronger the assumptions we need for our cryptography, the more likely it is that they do not hold (i.e. someone might find a way a way to solve the underlying problem)
- In Minicrypt we can still perform a surprising amount of useful cryptographic tasks, by only using one-way functions! This is a stronger assumption than just $P \neq NP$, but still a very minimal one!

A Matter of Theory

- In Cryptomania much more cryptographic tasks are possible, but they require stronger assumptions
- The stronger the assumptions we need for our cryptography, the more likely it is that they do not hold (i.e. someone might find a way a way to solve the underlying problem)
- In Minicrypt we can still perform a surprising amount of useful cryptographic tasks, by only using one-way functions! This is a stronger assumption than just $P \neq NP$, but still a very minimal one!
- In particular, quantum computers cannot break objects in Minicrypt in any meaningful way!

A Matter of Theory

- In Cryptomania much more cryptographic tasks are possible, but they require stronger assumptions
- The stronger the assumptions we need for our cryptography, the more likely it is that they do not hold (i.e. someone might find a way a way to solve the underlying problem)
- In Minicrypt we can still perform a surprising amount of useful cryptographic tasks, by only using one-way functions! This is a stronger assumption than just $P \neq NP$, but still a very minimal one!
- In particular, quantum computers cannot break objects in Minicrypt in any meaningful way!
- Digital signatures are more akin to block ciphers than public-key encryption!!!

A Matter of Theory

- In Cryptomania much more cryptographic tasks are possible, but they require stronger assumptions
- The stronger the assumptions we need for our cryptography, the more likely it is that they do not hold (i.e. someone might find a way a way to solve the underlying problem)
- In Minicrypt we can still perform a surprising amount of useful cryptographic tasks, by only using one-way functions! This is a stronger assumption than just $P \neq NP$, but still a very minimal one!
- In particular, quantum computers cannot break objects in Minicrypt in any meaningful way!
- Digital signatures are more akin to block ciphers than public-key encryption!!!
- collision-resistant hash functions are conjectured to be in Cryptomania. However, SPHINCS+ and similar only need weaker assumptions (conjectured to hold in Minicrypt)

Conclusions

Take-home message (TL;DR)

- hash-based signatures are fast and easy to implement
- most importantly, they offer the strongest crytographic quantum security guarantees among the various families of candidates for signatures
- their drawbacks are mainly larger signatures, but this can be traded for speed
- SPHINCS+ is the current candidate at the NIST competition

End Of This Talk

Thanks for your attention!

tommaso.gagliardoni@kudelskisecurity.com

KUDELSKI SECURITY