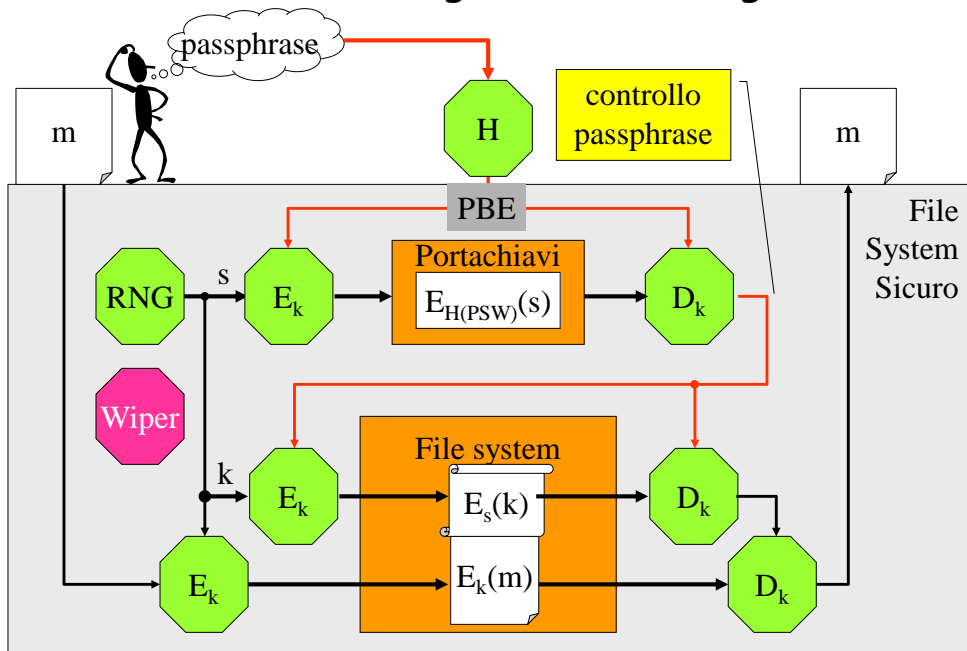
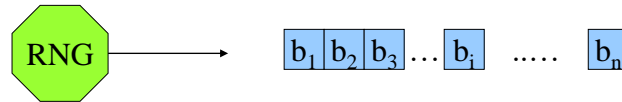


Tre livelli della gerarchia di segreti



MECCANISMI DI BASE:
Generatori di bit casuali

Random Number/Bit Generator



Proprietà di casualità dei bit:

- indipendenza statistica
- valori equiprobabili

Test statistici (FIPS 140-2): χ^2

- Monobit: *numero* di 1 e di 0
- Pocker: *sequenze* di M bit
- Run: *block* (1) e *gap* (0)
- Long Run: *block* più lungo

Altri test statistici:

- Autocor.: *differenze* dopo shift
- TdF: *distanza* pattern ripetitivi
- Compressione LZ: *lunghezza*
- ...

True Random Number Generator

Fenomeni fisici:

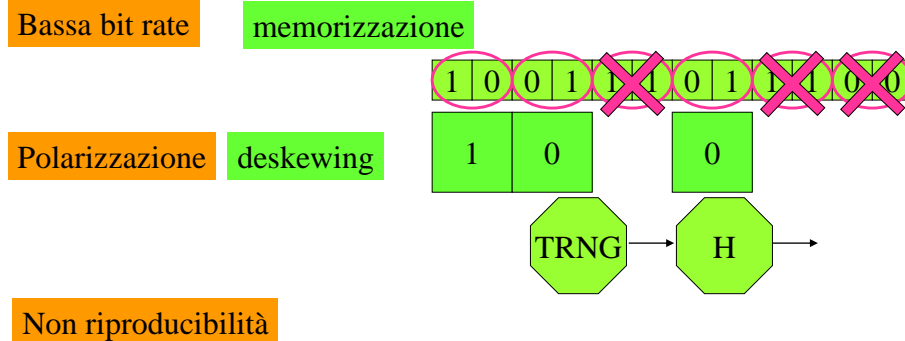
Decadimento radioattivo, rumore termico, turbolenza di un fluido

Segnali di apparati elettronici:

Microfono, telecamera, oscillatore

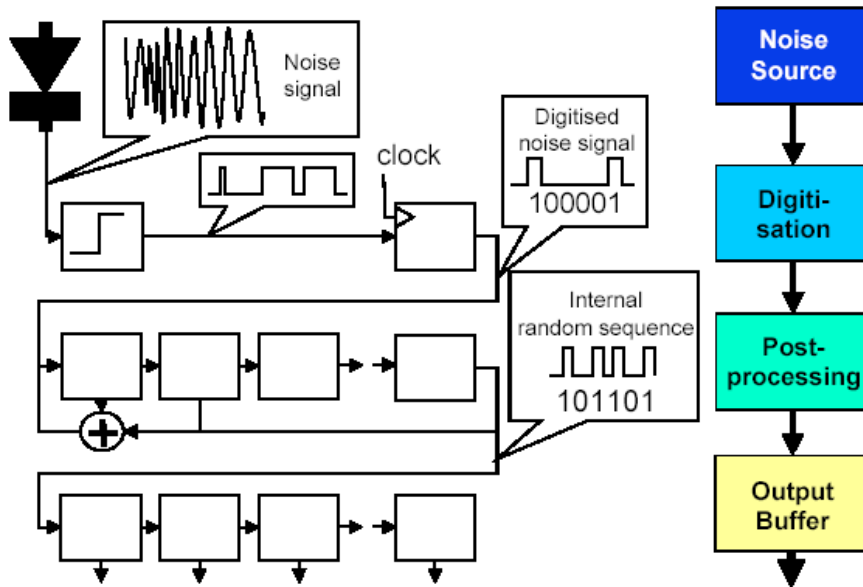
Programmi di estrazione di rumore dal funzionamento del computer

Tastiera, Mouse, n° di processi attivi, traffico di rete

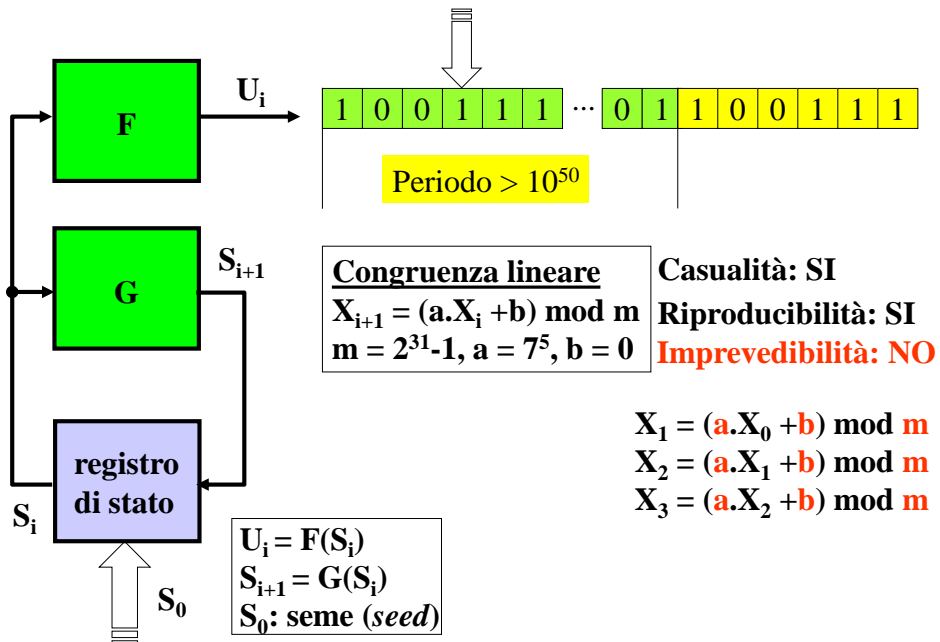


Non riproducibilità

Elaborazione del rumore



Pseudo Random Number Generator



Cryptographically Secure PseudoRandom Bit Generator

casualità dell'uscita

Test statistici

imprevedibilità dell'uscita

Next-bit test: dati L bit della stringa d'uscita non deve esistere un algoritmo polinomiale in grado di predire il bit L+1-esimo con probabilità $> 0,5$

indeducibilità del seed

One-way function

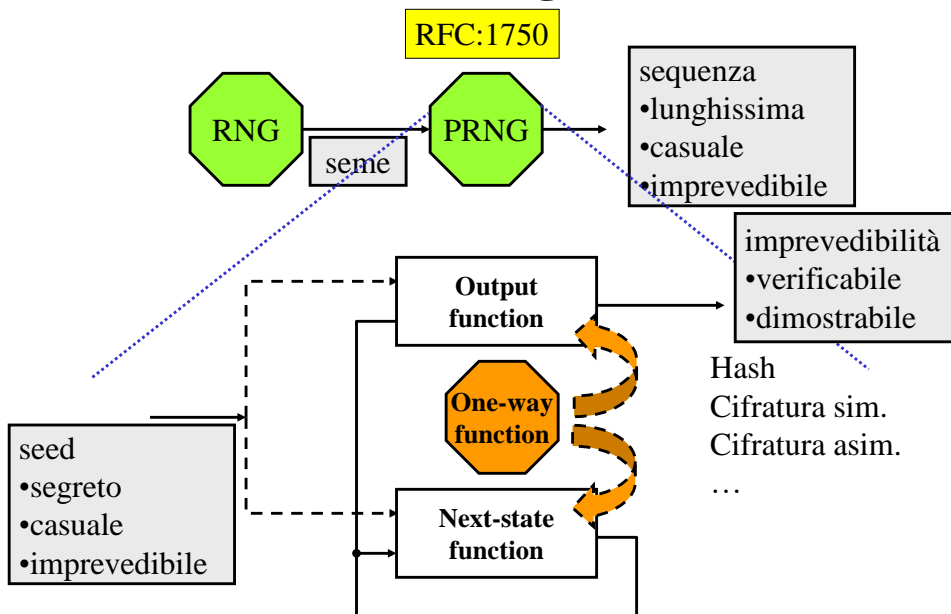
Crittografia simmetrica:

- verifica sperimentale
- alta velocità

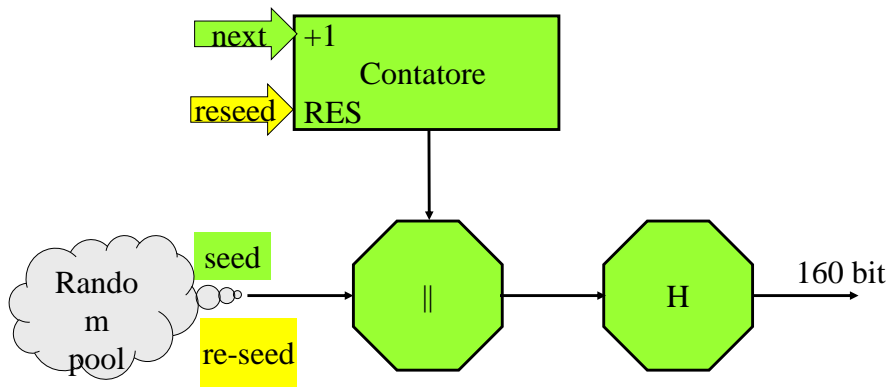
Crittografia asimmetrica:

- dimostrazione teorica
- bassa velocità

PRNG crittografico



Secure Random di Java

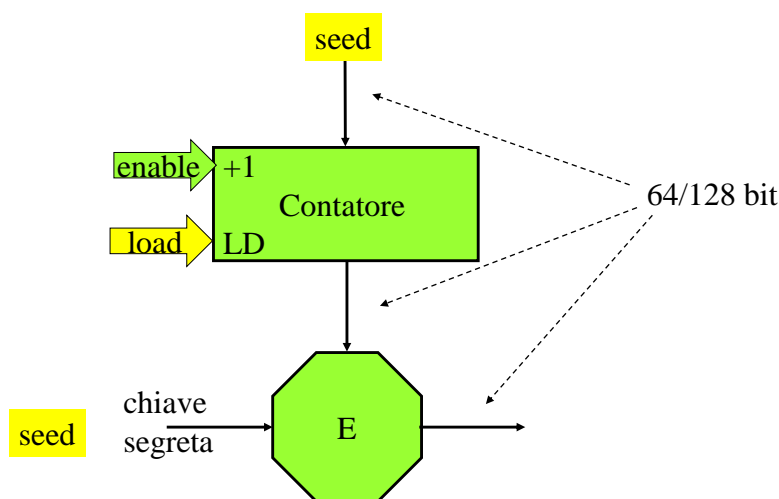


Uso Errato di PRNG

```
String GenerateReceiptURL(String baseUrl) {  
    Random ranGen = new Random();  
  
    ranGen.setSeed((new Date()).getTime());  
    return(baseUrl + Gen.nextInt(400000000) + ".html");  
}
```

```
public static int generateRandom(int maximumValue) {  
    SecureRandom ranGen = new SecureRandom();  
    return ranGen.nextInt(maximumValue);  
}
```

PRNG con cifratura di un contatore



Uso Errato di PRNG

Esempi di uso errato:

- Predictable netscape seed
- Microsoft windows 2000/XP random number generator
- Debian OpenSSL
- RSA public key factoring
- Java nonce collision
- ...

To appear in Proceedings of the 21st USENIX Security Symposium, August 2012. Initial public release, July 2, 2012. For the latest version of this paper, please visit www.usenix.org, and our online key check service, visit <http://bitcointalk.org>.

Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices

Nathia Heninger¹, Zakir Durumeric²,
¹University of California, San Diego
nathia@cs.ucsd.edu

Eric Wustrow², J. Alex Halderman²
²The University of Michigan
wustrow@umich.edu, halderman@umich.edu

Abstract

RSA and DSA are full asymptotically when used with sufficiently random number generators, but the extent to which these problems arise in practice has never been comprehensively studied at Internet scale. We perform the largest ever network survey of TLS and SSH servers and present evidence that widespread key reuse is widespread. We find that 0.19% of TLS certificates share keys due to insufficient entropy during key generation, and we suggest that almost 1% of certificates share the same RSA private keys. These keys are not unique to some previous. These keys are dangerous, as we are able to attack RSA private keys for 0.19% of TLS hosts and 0.19% of SSH hosts, breaking their public keys, shared material, and thus the confidentiality and integrity of all communications. We report and analyze the vulnerable keys, finding that the two majority appear to be broken or embedded devices. In cooperation with their software companies, we are able to produce the vulnerable and identify specific software behavior that makes them, including a host that carries keys in the Linux random number generator. Finally, we suggest actions and open issues for developers, users, and the security community.

1 Introduction and Roadmap

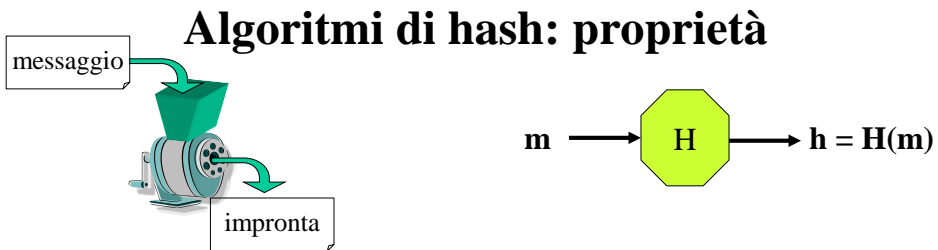
request that today's widely used operating systems and server software generate random numbers securely. In this paper, we use the perspective empirically by examining the public keys in use on the Internet.

The first component of our study is the most comprehensive Internet-wide survey to date of one of the most important cryptographic primitives, TLS and SSH certificates. To date, the largest survey of public key certificates was collected by a million unique TLS certificates from 0.19 million hosts and 0.19 million unique SSH keys from 0.19 million hosts. This is 0.19% more TLS keys than the latest released EFF TLS Observatory dataset [1].

Our techniques take less than 10 hours to scan the entire address space for Internet hosts and less than 10 hours to process keys from them. The results give us a previously unattainable perspective of the universe of keys.

Next, we build on this dataset to find evidence of several kinds of problems related to inadequate randomness. To our surprise, at least 0.19% of TLS hosts and 0.19% of SSH hosts use the same keys as other hosts in an apparently vulnerable manner (Section 4). In the case of TLS, at least 0.19% of hosts use weak private keys that have never changed by the owner, and another 0.19% use keys that are not unique to the host. We also find other kinds of problems related to random number generation. Only a handful of the vulnerable TLS certificates are signed by browser-trusted certificate authorities. From more than 100,000, we are able to extract the private keys for 94,000 (0.30%) of the TLS hosts and

MECCANISMI DI BASE: Algoritmi di hash



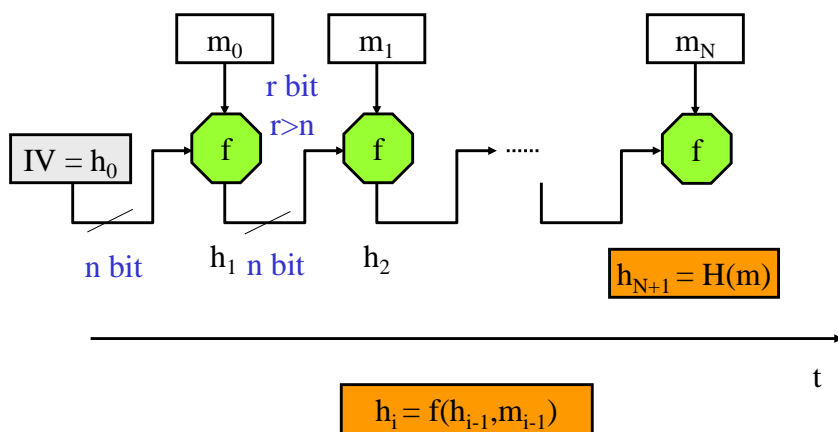
❑(efficienza): “il calcolo di $H(x)$ deve essere facile per ogni x ”

❑(robustezza debole alle collisioni): “per ogni x deve essere infattibile trovare un $y \neq x$ tale che $H(y) = H(x)$ ”

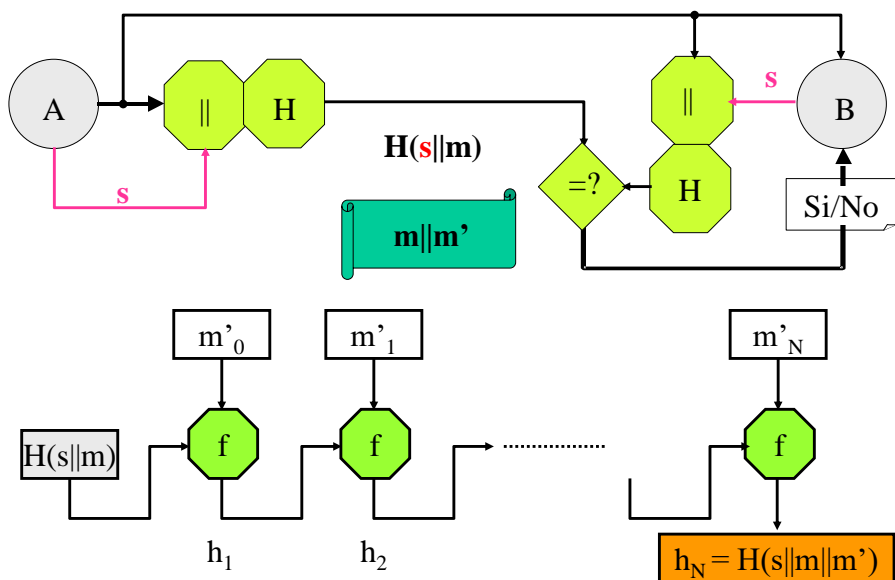
❑(robustezza forte alle collisioni): “deve essere infattibile trovare una qualsiasi coppia y, x tale che $H(y) = H(x)$ ”

❑(unidirezionalità): “per ogni h deve essere infattibile trovare un x tale che $H(x) = h$ ”

Efficienza: compressione iterata



Attacco con lenght extension

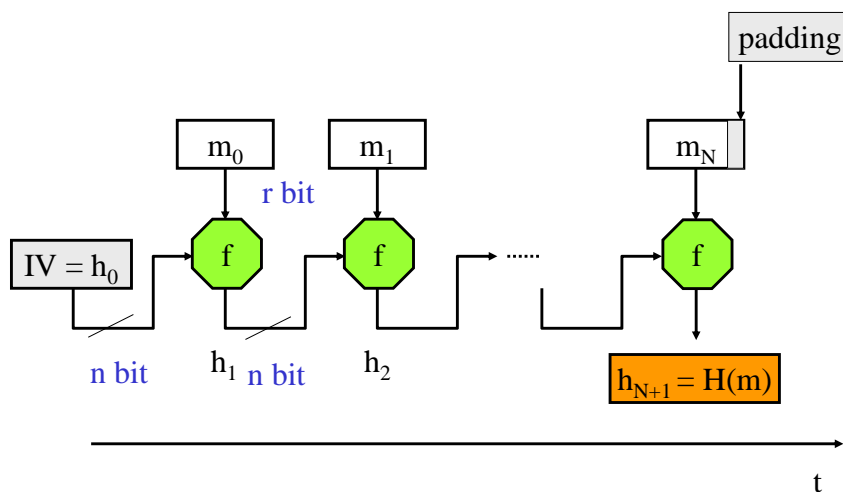


account_from=10203040&account_to=90807060&amount=100&hash=AABBCC112233
dove hash = MD5("SECRET1020304090807060100")

E se volessi trasferire 1000?
account_from=10203040&account_to=90807060&amount=1000&hash_new=?

Dove hash_new= MD5("SECRET10203040908070601000") =
= MD5(SECRET102030409080706010||0)

Efficienza: compressione iterata



account_from=10203040&account_to=90807060&amount=100&hash=AABBCC112233

dove hash = MD5("SECRET1020304090807060100")

E se volessi trasferire 1000?

account_from=10203040&account_to=90807060&amount=1000&hash=?

L'input alla funzione hash è:

"SECRET1020304090807060100" + PADDING + LENGTH + "0"
+ PADDING + LENGTH

Il **messaggio** quindi da validare sarebbe ora:

"SECRET1020304090807060100" + PADDING + LENGTH + "0"

ma quello che in origine erano metadata (+ PADDING + LENGTH)
attribuiscono un significato al nuovo messaggio o lo rendono
"insensato"?

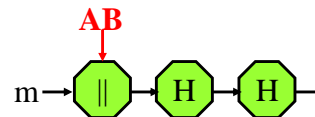
QUINDI: attacco non sempre possibile ma se possibile gravi danni

Contromisura: impronta di un'impronta

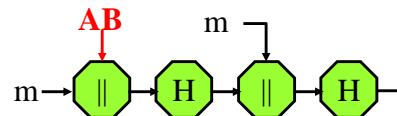
m messaggio

AB segreto condiviso da A e da B

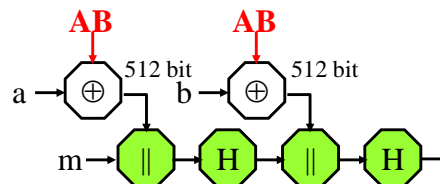
- $H(H(AB||m))$



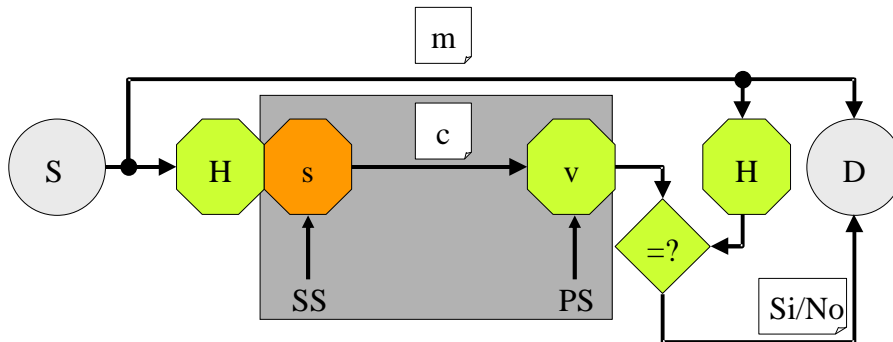
- $H(H(AB||m)||m)$



- **HMAC(AB, m)**
RFC(2104)



Resistenza alle collisioni e firma digitale

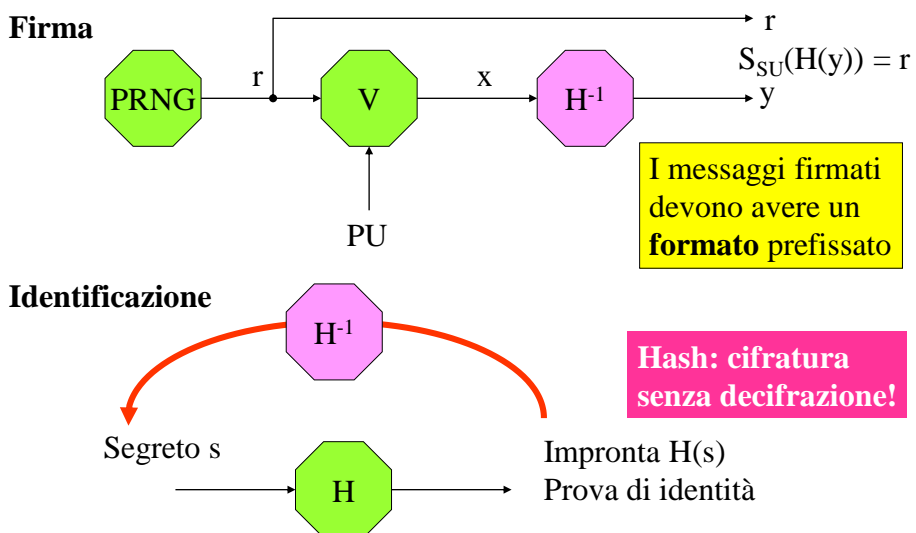


Quando valgono R19, R20 l'impronta $H(m)$ identifica m

Con la firma di $H(m)$ si ottiene:

- efficienza
- individuazione di modifiche a m e/o a c apportate dall'intruso
- S non può sostenere di aver inviato m^* e non m
- D non può sostenere di aver ricevuto da S un m^* da lui inventato

Unidirezionalità (non invertibilità)



Secure HASH functions

Algoritmi di hash più noti ed usati	MD5 1991	SHA-1 1994	RIPEMD-160 1996
Digest length	128 bits	160 bits	160 bits
Basic unit of processing	512 bits	512 bits	512 bits
Number of steps	64 (4 rounds of 16)	80 (4 rounds of 20)	160 (5 paired rounds of 16)
Maximum message size	∞	$2^{64}-1$ bits	∞
Tiger (1996) 192 bit	NIST 2002: SHA-256 SHA-384 SHA-512	Whirlpool (2002) 512 bit	SHA-3 (2015) resistente all'attacco con estensione di messaggio

Funzione Hash crittografica: dimensionamento dell'impronta

Quanti bit
deve avere
un'impronta
per essere
sicura?

Complessità del calcolo di una collisione

IPOTESI: una funzione hash sottoposta ad ingressi scelti a caso restituisce, con eguale probabilità, uno dei suoi 2^n valori d'uscita.

Problema correlato: data una funzione hash con n possibili output e un determinato $H(x)$, se H viene applicato a K input casuali, quale deve essere il valore di k tale che la probabilità che almeno un input y soddisfi $H(y)=H(x)$ sia 0,5?

un tentativo: probabilità di successo $P_1(2^n, 1) = 2^{-n}$,
probabilità di insuccesso $1 - 2^{-n}$.

k tentativi: probabilità di successo $P_1(2^n, k) = 1 - (1 - 2^{-n})^k$

Teorema binomiale: $(b+a)^n = \sum_{i=0}^n \binom{n}{i} b^i a^{n-i}$

$$(1 - 2^{-n})^k = 1 - k 2^{-n} + \frac{k(k-1)}{2!} 2^{-2n} - \frac{k(k-1)(k-2)}{3!} 2^{-3n} + \dots$$

$$P_1(2^n, k) = k \cdot 2^{-n} - \frac{k(k-1)}{2} \cdot 2^{-2n} + \frac{k(k-1)(k-2)}{6} \cdot 2^{-3n} - \dots \text{ecc.}$$

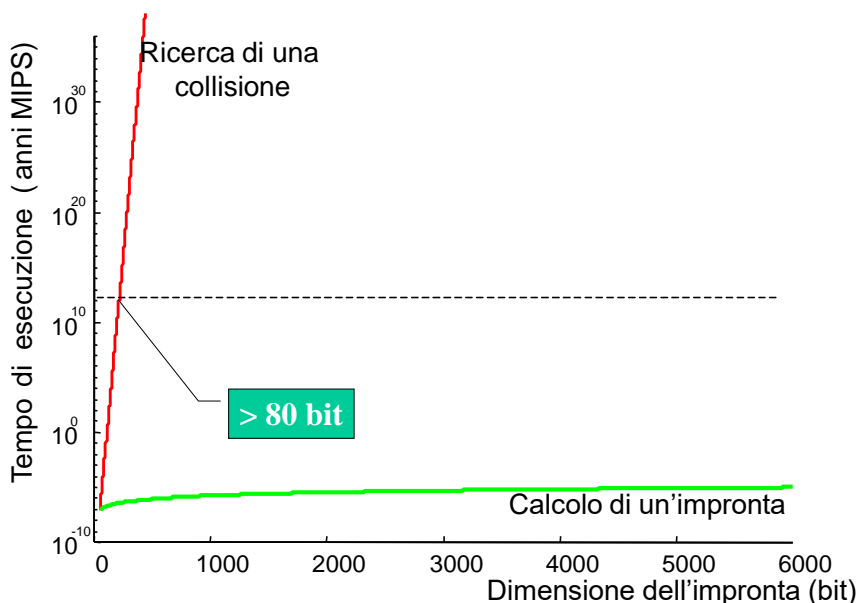
$\approx k \cdot 2^{-n}$ quando 2^{-n} è molto piccolo

S : probabilità di successo desiderata

$$S = P_1(2^n, k) \rightarrow k = S \cdot 2^n$$

$$O(\exp(n))$$

Robustezza debole alle collisioni



Un Possibile Attacco in assenza di Resistenza debole


L'intruso prepara 2 versioni di un contratto M ed M'

a) M è favorevole ad Alice

b) M' è sfavorevole ad Alice

modifica M' a caso (piccoli cambiamenti come aggiunta spazi) finchè $h(M) = h(M')$

Alice firma M  $\text{Firma}_{\text{kpriv}}(h(M))$

L'intruso ha quindi la firma di M'  $\text{Firma}_{\text{kpriv}}(h(M'))$

In media , l'intruso dovrebbe provare 2^n messaggi se n è il numero di bit dell'impronta

Un Possibile Attacco in assenza di Resistenza debole

Cara Alice,

ti {
 scrivo
 sto scrivendo

Il paradosso del giorno del compleanno

Birthday paradox

Nell'ipotesi che le date di nascita siano equiprobabili, è sufficiente scegliere a caso **253** persone per avere una probabilità $> 0,5$ che una di queste compia gli anni in un dato giorno.

Sono invece sufficienti **23** persone scelte a caso per avere una probabilità $> 0,5$ che due o più compiano gli anni nello stesso giorno.

PARADOSSO del COMPLEANNO

Qual è il valore minimo di k tale che la probabilità che almeno due persone (una qualsiasi coppia di persone) in un gruppo di k persone abbiano lo stesso compleanno sia maggiore di 0,5?

Si ignora il 29 febbraio e si ipotizza che ciascun compleanno sia equiprobabile

$P(n, k)$ = probabilità che esista almeno un duplicato in k elementi quando ciascun elemento può assumere un valore ugualmente probabile fra 1 e n

$Q(n, k)$ = probabilità che non esistono duplicati

Se $k >= 365$ è impossibile che non esistano duplicati. Consideriamo quindi il caso $k <= 365$

Sia N il numero dei vari modi in cui vi possono essere k valori senza duplicati. Si può scegliere uno qualsiasi dei 365 valori come primo elemento, uno qualsiasi dei rimanenti come secondo elemento e così via =>

$$N = 365 \times 364 \times \dots (365 - k + 1) = 365! / (365 - k)!$$

$$\text{Numero totale di possibili valori} = 365^k$$

$$Q(365, k) = (365! / (365 - k)!) / 365^k$$

$$P(365, k) = 1 - Q(365, k) = 1 - (365! / (365 - k)!) / 365^k$$

Calcolo di due input in collisioni

$P_2(2^n, k)$ probabilità di due uscite identiche con $k \leq 2^n$ ingressi scelti a caso

•sequenze d'uscita possibili: $(2^n)^k$ differenti

•sequenze con valori tutti diversi: $2^n! / (2^n - k)!$

$$\begin{aligned} P_2(2^n, k) &= 1 - \frac{\overset{\text{tutti diversi}}{2^n!} / \overset{\text{tutti}}{(2^n - k)!}}{(2^n)^k} \\ &= 1 - \frac{2^n \times (2^n - 1) \times (2^n - 2) \times \dots \times (2^n - k + 1)}{2^{nk}} \\ &= 1 - (1 - 1/2^n)(1 - 2/2^n) \dots (1 - (k-1)/2^n) \end{aligned}$$

N.B. $(1-x) \leq e^{-x}$ è valida per $x \geq 0$,

e^{-x} è una buona approssimazione di $(1-x)$ per $x < 0,3$

$$\begin{aligned} P_2(2^n, k) &\cong 1 - \exp[-2^{-n}(1+2+\dots+(k-1))] \\ &= 1 - \exp[-2^{-n}(k(k-1)/2)] \text{ e per } k \text{ grande} \\ &\cong 1 - \exp[-2^{-n}(k^2/2)] \end{aligned}$$

IPOTESI: $P_2 = 1/2$

$$1 - 1/2 = \exp[-2^{-n}(k^2/2)]$$

$$\ln 2 = 2^{-n} \times (k^2/2)$$

$$k = [2 \times (\ln 2)^{1/2} \times 2^{n/2}] = 1,18 \times 2^{n/2}$$

$$\text{Paradosso del compleanno: } k = (2 \cdot \ln(2))^{1/2} \cdot (365)^{1/2} = 22,54.$$

Robustezza forte alle collisioni

