Alma Mater Studiorum · Università di Bologna

**FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI**

**Corso di Laurea in Matematica**

# AUTENTICAZIONE E FIRMA DIGITALE

**Tesi di Laurea in Algoritmi della Teoria dei Numeri e Crittografia**

**Relatore:**
**Chiar.mo Prof.**
**Davide Aliffi**

**Presentata da:**
**Elisa Bragaglia**

**II Sessione**

**Anno Accademico 2011-2012**

*A chi ha creduto in me,*
*all'infaticabile Professoressa Serra*
*e al 'lavoro più bello del mondo'.*

# Introduzione

Tipica conversazione pre-laurea:
'Ehi Eli, alla fine su cosa la fai la tesi?'
'Sulla crittografia!'
'Ah... cioè?'

Oggi, nell'era del computer e di Internet, non si conosce ancora quale ruolo chiave giochi la crittografia nella vita di tutti i giorni.
Eppure, inconsciamente, ce ne serviamo tutti; dai pagamenti on-line, alla posta elettronica, dai prelievi Bancomat, alle telefonate: questa è crittografia.
Questa scienza non serve solo a cifrare e decifrare messaggi, ma entra in gioco, nella nostra vita quotidiana, per risolvere i problemi che richiedono la sicurezza e l'integrità delle nostre informazioni, ovunque noi le lasciamo (nel computer, nel telefono...).
Quattro sono i suoi principali obiettivi:

**Riservatezza:** un osservatore esterno alla comunicazione tra due entità non deve essere in grado di leggerne i messaggi.
Qui entrano in gioco gli algoritmi di cifratura e decifrazione.

**Integrità dei dati:** colui che riceve il messaggio deve essere sicuro che questo non sia stato alterato nella trasmissione dei dati o da un antagonista.
Di questo si occupano, per esempio, le funzioni hash.

**Autenticazione:** colui che riceve un messaggio da parte di un entità, deve essere sicuro che il mittente sia realmente chi dice di essere, e non qualcun altro che cerca di impersonarlo.

Questo problema si divide in autenticazione dell'entità (provare l'identità dei soggetti coinvolti) e autenticazione dell'origine dei dati (provare che l'origine dei dati è proprio quella dichiarata).
Utili a questo scopo sono i protocolli con password e i sistemi di identificazione.

**Non ripudiabilità:** chi invia un messaggio non può negare di averlo fatto, per esempio nel caso di ordini di acquisto on-line.

In questo elaborato verranno toccati tutti questi argomenti: attraverso un breve quadro storico si presenterà l'evoluzione della crittografia a chiave pubblica, le sue problematiche e la necessità di introdurre una **firma digitale**. Verranno presentate le diverse tipologie con cui questa può essere creata, a seconda del sistema crittografico presente, la sua funzione e le applicazioni nella vita quotidiana. Infine verrà studiato il caso specifico innovativo del sistema di Feige-Fiat-Shamir.

# Indice

# Premesse

Poichè il problema principale dell crittografia è la comunicazione sicura tra entità, noi, per convenzione, chiameremo Alice e Bob gli interlocutori (possono essere due persone, ma anche due computer, due sistemi...) e Eva una terza parte che ne vuole intercettare i messaggi.
Per cifrare un messaggio è necessaria una *chiave di cifratura* $k$ e una *funzione di cifratura*:

$$E_k : M \longrightarrow C$$

con $M$ spazio dei messaggi in chiaro e $C$ spazio dei messaggi cifrati.
Bob, che riceve il testo cifrato, deve trasformarlo nel testo in chiaro originale utilizzando la *chive di decifrazione*, attraverso la *funzione di decifrazione*:

$$D_k : C \longrightarrow M \quad tale \ \ che \quad D_k(E_k(m)) = m \quad \forall \ m \in M$$

Scopo di Eva potrebbe essere:

1. Leggere il messaggio
2. Trovare la chiave e leggere tutti i messaggi cifrati con quella chiave
3. Alterare il messaggio di Alice, in modo che Bob riceva un messaggio modificato
4. Fingersi Alice e comunicare con Bob, mentre quest'ultimo pensa di comunicare con Alice

In questo elaborato verranno ampliati e discussi soprattutto i problemi di autenticazione e integrità del messaggio, corrispondenti ai punti 3. e 4.

Si suppone sempre che Eva conosca il sistema di cifratura usato da Alice, ma sia soggetta alle limitazioni effettive di calcolo, date dalla potenza non infinita dei computer a sua disposizione.
In generale, la sicurezza dei sistemi crittografici dipende dalla lunghezza della chiave; diremo che un sistema è *computazionalmente sicuro* se resiste agli attacchi computazionalmente possibili (ovvero con tempo di esecuzione al più polinomiale rispetto alla lunghezza della chiave), ma soccombe in caso di attacchi con potenza di calcolo illimitata (ovvero con tempo di esecuzione esponenziale rispetto alla lunghezza della chiave), diremo, invece, che è *incondizionalmente sicuro* se resiste a ogni tipo di attacco.

# Capitolo 1

# Dalla Crittografia a Chiave Simmetrica alla Crittografia a Chiave Pubblica

***"We stand today on the brink of a revolution in cryptography"***

*(Ci troviamo oggi sull'orlo di una rivoluzione in crittografia)*

Così si apre l'articolo di Whitfield Diffie e Martin E.Hellman pubblicato nel novenbre 1976 "*New Directions in Cryptography*"[1]; qui viene proposta per la prima volta l'innovativa idea di crittografia a chiave pubblica.
Per capire di che peso sia la novità introdotta occurre fare un passo indietro, e fare il punto della situazione storica di questa scienza.

## 1.1 Crittografia a Chiave Simmetrica

Fin dall'antichità l'uomo ha cercato un modo sicuro per proteggere i propri segreti e cammuffare i messaggi per renderli irriconoscibili agli occhi degli avversari.
Ne dà testimonianza il 'Cifrario di Cesare' (I secolo a.C.), uno dei primi esempi di

[1] [5]

crittosistema a scorrimento, seguito poi dal cifrario affine, e, in epoca più recente, dal cifrario di Vigenére (XVI secolo), dal cifrario a blocchi e dal cifrario di Hill (1929).
Nel 1928 Vernam introduce il cifrario perfetto, ma la necessità di avere chiavi lunghe come il testo e non riutilizzabili, lo rendono inapplicabile nella pratica.
L'ultima tappa, prima della così detta crittografia moderna, è costituita dalla macchina elettromeccanica tedesca ENIGMA, usata durante la Seconda Guerra Mondiale.
La crittografia moderna utilizza ancora operazioni di sostituzione e trasposizioni, ma combinate tra loro in algoritmi complessi; ricordiamo il cifrario DES (1974) e AES (2001).
I sistemi di cifratura sopra citati sono detti a **'chiave simmetrica'**, in quanto la chiave di cifratura e quella di decifrazione, che in genere si ricavano facilmente l'una dall'altra, sono note agli interlocutori.
La sicurezza di questi sistemi si basa quasi interamente sulla segretezza della chiave: è dunque necessario un canale sicuro attraverso il quale Alice possa comunicarla a Bob.
Questo è il problema principale dei sistemi a chiave simmetrica: lo scambio della chiave e la necessità di un canale sicuro nel quale questo possa avvenire senza interferenze.
Una soluzione a questo non facile problema risiede nella crittografia a **'chiave pubblica'**.

## 1.2 Nascita della Crittografia a Chiave Pubblica

L'idea di crittografia a chiave pubblica nasce negli anni '70; Whitfield Diffie e Martin E.Hellman propongono nel 1976 un nuovo sistema, che risolve il problema del canale sicuro eliminando lo scambio della chiave segreta. In questa nuova ottica Alice e Bob possono comunicare in sicurezza senza dover condividere preventivamente nessun segreto.
Il sistema a chiave pubblica ideato da Diffie e Hellman consiste in due famiglie di algoritmi $\{E_k\}_{k\in K}$ (trasformazioni di cifratura) e $\{D_k\}_{k\in K}$ (trasformazioni di decifrazione), che rappresentano le trasformazioni invertibili:

$$E_k : M \to M \qquad D_k : M \to M$$

su uno spazio finito dei messaggi $M$ con $K$ spazio finito delle chiavi, tale che:

1. $\forall\, k \in K$, $\exists\, k' \in K$ tale che $D_{k'}$ è l'inversa di $E_k$

2. $\forall\, k \in K$ e $\forall\, m \in M$, $E_k(m) = c$ e $D_{k'}(c) = m$ sono facilemente calcolabili

3. per quasi tutti i $k \in K$, non è computazionalmente possibile trovare $D_{k'}$, data $E_k$.

Si osservi che la proprietà 3. permette che la chiave di cifratura $k$ possa essere resa pubblica senza compromettere la segretezza della chiave di decifrazione $k'$.
In questo modo, si può generare la chiave $k$ casualmente e da lì generare $E_k$ e $D_{k'}$; $E_k$ può essere resa pubblica, mentre $D_{k'}$ deve rimanere segreta.
Ora, chiunque può cifrare un testo utilizzando la chiave pubblica, ma solo il legittimo destinatario, con la chiave segreta, può decifrare e leggere il messaggio.

Il problema del canale sicuro per lo scambio delle chiavi è stato eliminato.

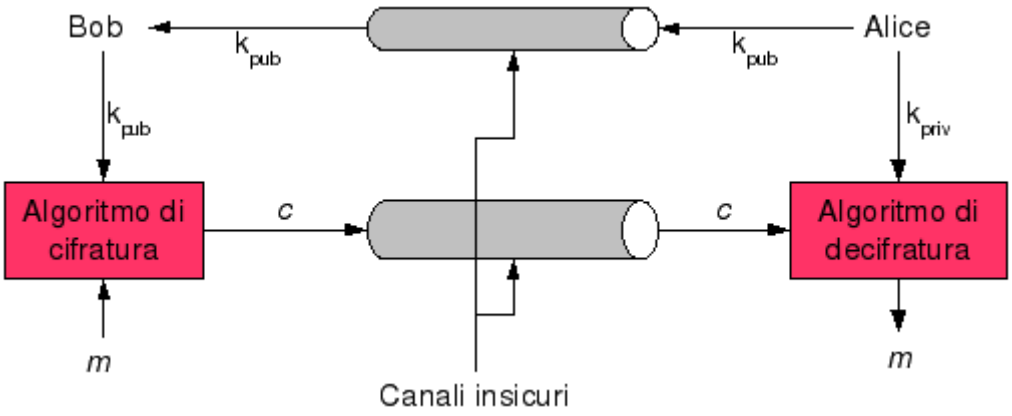



**Figura 1.1:** Bob manda un messaggio ad Alice, cifrandolo con un sistema a chiave pubblica

Persistono altre problematiche:

1. Come creare, in pratica, un sistema di questo tipo?

2. Esiste un modo efficace per attaccare questo sistema?

### 1.2.1 Scambio della chiave di Diffie - Hellman

Rigurdo la prima problematica, Diffie e Hellman hanno creato un algoritmo per lo scambio di una chiave segreta, che elimina la necessità di un canale sicuro, noto come:

**Protocollo di scambio della chiave di Diffie - Helmann**.
In questo sistema Alice e Bob si scambiano una chiave in un canale non sicuro, senza mai doverla realmente inviare, sfruttando il problema matematico del logaritmo discreto: vediamolo nel dettaglio.

**Definizione 1.1.** Dato $p$ primo, data $g \in \mathbb{Z}_p{}^*$ radice primitiva modulo $p$, allora: $\forall\, b \in \mathbb{Z}_p{}^*\ \exists!\, x \in \{0, 1, .., p-2\}$ tale che $g^x = b \ \ mod\, p$.
$x$ si chiama: ***logaritmo discreto di b rispetto alla base g***.

**Protocollo di scambio della chiave di Diffie - Helmann:**

1. Alice e Bob scelgono un numero primo $p$ grande[2] e una radice primitiva $\alpha$ modulo $p$; entrambi possono essere resi pubblici.
2. Alice e Bob scelgono a caso, rispettivamente un $x$ e un $y\ \in \{1, ..., p-2\}$ segreti.
3. Alice manda (su un canale insicuro) a Bob $\alpha^x\ \ (mod\, p)$ e Bob manda ad Alice $\alpha^y\ \ (mod\, p)$.
4. Entrambi calcolano la chiave $k = \alpha^{xy}\ \ (mod\, p)$, e possono usarla per comunicare con un cifrario a chiave simmetrica come DES o AES.

La sicurezza di questo sistema dipende crucialmente dalla difficoltà matematica di risolvere il logaritmo discreto: $x = log_\alpha\, b\ \ (mod\, p)$ , ricavando così $x$ e calcolando la chiave.
Si ha:

**risolvere il problema del logaritmo discreto $\Rightarrow$ violare il protocollo**

Vale anche il viceversa? Questo è ancora un problema aperto.
In realtà, per trovare la chiave basta risolvere il problema, noto come ***Problema computazionale di Diffie-Hellman***, di trovare $\alpha^{xy}\ \ (mod\, p)$, noti $\alpha^x\ \ (mod\, p)$ e $\alpha^y\ \ (mod\, p)$, con $p$ primo e $\alpha$ radice primitiva modulo $p$.

[2] *in crittografia, oggi, 'grande' vuol dire dell'ordine di almeno 300 cifre dicimali*

Non è noto se questo problema sia più facile del calcolo del logaritmo discreto, certamente non è più difficile.

### 1.2.2 Autenticazione unidirezionale

**Attacco: man in the middle**

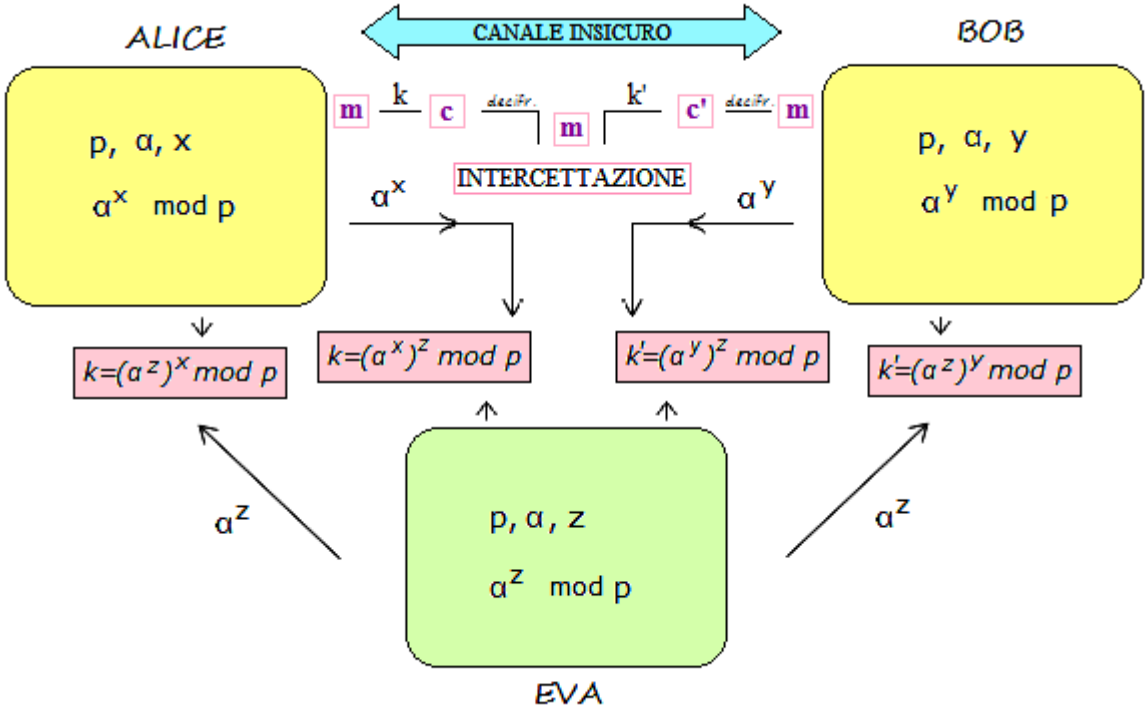
Riguardo la seconda problematica, esiste un attacco, chiamato *man in the middle* (*uomo nel mezzo*), che può compromettere la sicurezza di un sistema a chiave pubblica di questo tipo.
Eva, infatti, può accedere alla comunicazione tra Alice e Bob in questo modo:
Supponiamo che Alice voglia mandare un messaggio a Bob; per cifrarlo chiede a Bob la sua chiave pubblica; Eva intercetta la chiave di Bob e manda ad Alice la propria chiave pubblica, di cui possiede la decifrazione. Alice, che non sospetta nulla, cifra il messaggio e lo manda ad Eva, convinta di averlo mandato a Bob. Eva, decifra, legge ed eventualmente modifica il messaggio, e lo invia a Bob. Bob riceve il messaggio di Alice, senza sapere che è passato nelle mani di Eva.
In questo modo, Eva può leggere e gestire a suo piacimento tutta la conversazione tra Alice e Bob, senza che i due interlocutori lo sappiano.

***Esempio*: Caso del Protocollo di Diffie-Hellman**
Nel caso già trattato del Protocollo di Diffie-Hellman, l'attacco può essere eseguito in questo modo:
Dati $p$ e $\alpha$ pubblici , $x$ e $y$ segreti, scelti come precedentemente descitto;

1. Eva sceglie un esponente $z \in \{1, ..., p-2\}$.
2. Eva intercetta $\alpha^x$ e $\alpha^y$.
3. Eva invia $\alpha^z$ ad Alice e Bob, mentre Alice crede di ricevere $\alpha^y$ (chiave pubblica di Bob) e Bob $\alpha^x$ (chiave pubblica di Alice).
4. Eva calcola $k = (\alpha^x)^z \ mod(p)$ e $k' = (\alpha^y)^z \ mod(p)$. Alice e Bob, ignorando l'esistenza di Eva, calcolano rispettivamente $k = (\alpha^z)^x$ e $k' = (\alpha^z)^y$ .

5. Ora, quando Alice manda un messaggio a Bob, cifrato con la chiave $k$, Eva lo intercetta, lo decifra, e lo rimanda cifrato a Bob usando la chiave $k'$. Eva, di fatto, controlla tutta la comunicazione.

Nella figura sottostante è mostrato lo schema dell'attacco in questo caso particolare:




Per rendere un sistema immune da questo tipo di attacco è necessario un metodo che attesti l'autenticità della persona o dell'entità con la quale si vuole comunicare.
Sulla carta si potrebbe usare la firma personale; essa infatti ci identifica in modo unico, è difficile da falsificare, e ci vincola a ció che firmiamo.
Diffie e Hellman lavorano su un sistema di identificazione a senso unico, ovvero un algoritmo che permetta l'autenticazione di un'entità rispetto a un'altra.
Ogni sistema di questo genere è chiamato: ***one-way authentication*** (identificazione unidirezionale).
Consideriamo il seguente esempio di 'login' in un computer con più utenti:

La prima volta, per accedere al computer, un utente dovrà creare una password, che lo identificherà nelle volte sucessive, e che verrà salvata in una cartella.
In ogni accesso gli verrà richiesto di inserire la password; il computer verificherà che la password inserita dall'utente sia uguale a quella salvata precedentemente; se così avviene, l'identificazione viene ritenuta valida e l'utente può accedere al computer.
Ovviamente, per mantenere segrete le password, nessuno può avere accesso a quella cartella, ma se qualcuno avesse una legittima ragione per accedere a quell'archivio?
Questa limitazione porta a chiedersi se esiste un'alternativa a questo sistema di salvataggio password.
Una valida opzione è data dall'uso delle funzioni unidirezionali:

**Definizione 1.2.** Una funzione $f$ si definisce **unidirezionale** se:

- per ogni $x$ del dominio di $f$, $f(x)$ è facilmente calcolabile
- per quasi ogni $y$ dell'immagine di $f$, è computazionalmente impossibile calcolare una preimmagine $x$, tale che $f(x) = y$.

Ora, supponiamo che $f$ sia una funzione unidirezionale e $X$ sia la password dell'utente. Modifichiamo il sitema precedente salvando in una cartella, non più le password, ma i valori $f(x)$, con $x$ password. In questo modo l'utente, per identificarsi nell'accesso, inserisce la sua password $X$, il computer calcola $f(X)$ e lo confronta col dato salvato, se risultano uguali, il login è effettuato.
Con questa modifica, chiunque può avere accesso alla cartella dei dati salvati, senza pregiudicare la sicurezza delle password; infatti dato un $f(x)$, non è computazionalmente possibile risalire a x, poichè $f$ è unidirezionale.
È necessario, ora, trovare delle funzioni con queste caratteristiche; Diffie e Hellman propongono di usare i crittosistemi. Essi affermano, infatti, che ogni crittosistema sicuro contro un attacco di tipo 'testo in chiaro noto'[3] può essere usato per produrre una funzione unidirezionale. In particolare può essere utilizzato per questo scopo un sistema a chiave pubblica.

[3]In questo attacco Eva ha a disposizione una copia del testo in chiaro e una del corrispondente testo cifrato

Data una chiave $k$, dato un testo in chiaro fissato $m$, allora la funzione:

$$f : K \longrightarrow C \text{ tale che } f(k) = \text{cifratura } k(m) = E_k(m)$$

è unidirezionale.
Infatti, trovare $k$, dato *f(k)* è equivalente al problema di trovare la chiave, dato un testo in chiaro e il corrispondente testo cifrato; ma questo è possibile solo se il nostro crittosistema è sensibile a un attacco 'testo in chiaro noto', eventualità esclusa per ipotesi.

Rimane un'ultima problematica di questo sistema da risolvere: la possibilità, previo permessi, di poter risalire alla password di un utente.
Questo dilemma ha una soluzione: le **funzioni 'trap-door one-way'** (unidirezionali con scappatoia).
Esse sono algoritmi di cui non esiste un inverso computazionalmente calcolabile, ma che, se viene svelata una certa informazione segreta (trap-door), diventano facilente invertibili.
Un esempio di funzione unidirezionale trap-door è il 'Protocollo di Diffie-Hellman': apparentemente non c'è modo di trovare la chiave $k$, dati $\alpha^x \ (mod \ p)$ e $\alpha^y \ (mod \ p)$, ma, se viene rivelata una certa informazione, per esempio i valori casuali $x$ e $y$, allora, non è più impossibile risalire alla chiave $\alpha^{xy} \ (mod \ p)$.

L'autenticazione 'one-way' sopra proposta è il precursore della vera e propria '**firma digitale**'.
Diffie e Helmann in realtà, nonstante tutte le loro idee innovative, non trovarono, in pratica, un sistema crittografico completo e valido che corrispondesse a tutte le loro teorie. Per questo bisogna aspettare il 1977, che vede protagonista la nascita del sistema **RSA**, basato sul problema matematico della fattorizzazione in primi.

# Capitolo 2

# Algoritmo RSA e Firme Digitali

## 2.1 Rivest-Shamir-Adleman: RSA

In seguito all'articolo di Diffie-Hellman del 1976 vennero proposti vari esempi di crittosistemi a chiave pubblica. Quello di maggior successo fu **RSA**, ideato da Rivest, Shamir e Adleman nel 1977[1].
L'algoritmo si basa sulla difficoltà matematica di fattorizzare gli interi in primi, e opera come segue.
Supponiamo che Bob voglia mandare un messaggio ad Alice:

1. Alice sceglie due primi grandi $p$ e $q$ e calcola $n = pq$.
2. Alice sceglie un esponente di cifratura $e$ tale che $MCD(e, (p-1)(q-1)) = 1$
3. Alice calcola $d$ in modo che $de \equiv 1 \ mod((p-1)(q-1))$
4. Alice rende pubblici $(n, e)$ (chiave pubblica), e tiene segreti $p, q$ e $d$ (chiave segreta).
5. Bob, per comunicare con Alice, cifra il messaggio $m$ come $c \equiv m^e \ \ (mod\ n)$ e invia $c$ ad Alice.
6. Alice decifra il messaggio calcolando $c^d \equiv (m^e)^d \equiv m \ \ (mod\, n)$.

[1] [6]

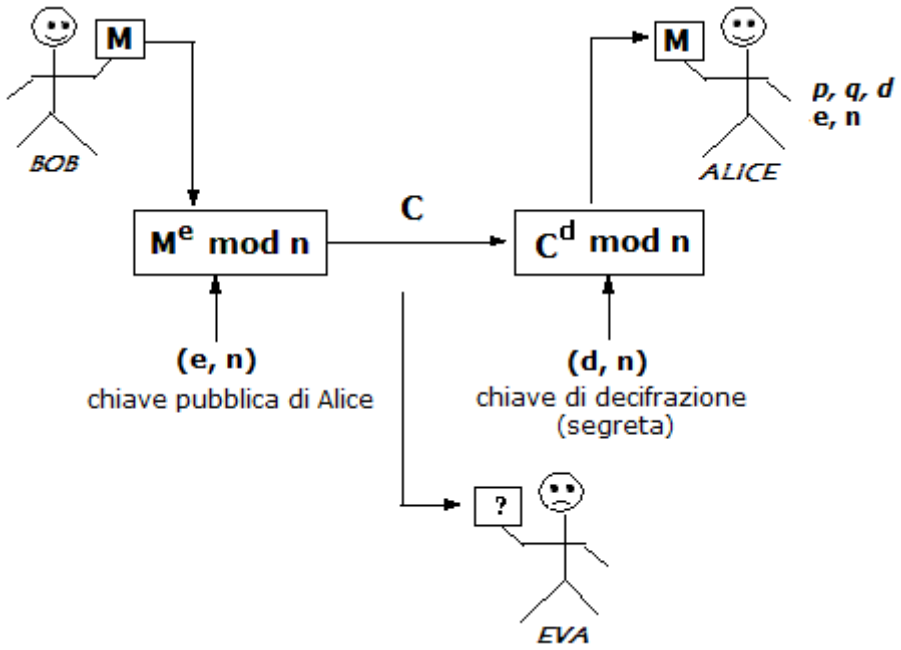
Dimostriamo, ora, che $m \equiv c^d \pmod{p}$.

*Dimostrazione.* Supponiamo $MCD(m, n) = 1$[2].
Per il Teorema di Eulero, vale: $a^{\varphi(n)} \equiv 1 \pmod{n}$.
Nel nostro caso: $\varphi(n) = \varphi(pq) = (p-1)(q-1)$.
Poichè $de \equiv 1 \pmod{(\varphi(n))}$, si può scrivere $de \equiv 1 + k\varphi(n)$ con $k$ intero opportuno.
Si ha:

$$c^d \equiv (m^e)^d \equiv m^{1+k\varphi(n)} \equiv m(m^{\varphi(n)})^k \equiv m1^k \equiv m \pmod{n}$$

□

La figura sottostante riporta lo schema del metodo:




[2]Poichè $p$ e $q$ sono primi molto grandi, è molto probabile che $m$ non possieda nessuno dei due come fattore

**Esempio:**[3]

Alice sceglie $p = 885320963$ e $q = 238855417$ primi e calcola $n = 211463707796206571$ ed $e = 9007$.

Alice manda a Bob $(n, e)$.

Il messaggio da cifrare è 'cat'.

Per convenzione, numeriamo le lettere a partire da $a = 01$ fino a $z = 26$.

Il messaggio risulta dunque: $m = 30120$.

Bob calcola

$$c \equiv m^e \equiv 30120^{9007} \equiv 113535859035722866 \ (mod\, n)$$

e invia $c$ ad Alice.

Alice, conoscendo $p$ e $q$, calcola $d$ in modo che

$$de \equiv 1 \ (mod(p-1)(q-1)),$$

ottenendo

$$d = 116402471153538991$$

.

Infine Alice calcola

$$c^d \equiv 113535859035722866^{116402471153538991} \equiv 30120 \ (mod\, p),$$

ottenendo il messaggio originale.

La sicurezza dell'algoritmo si basa sul fatto che, ad oggi, non si conoscono metodi di fattorizzazione di un intero $n$ grande computazionalmente efficienti.
Si dimostra che il problema di trovare la chiave di decifrazione $d$, conoscendo $(e, n)$ è essenzialente difficile quanto fattorizzare $n$.
Questo non esclude che ci sia un modo semplice di calcolarla, ma al momento non lo si conosce.

[3] [1] pp. 148-149

## 2.2 Firma Digitale

Fin dalla nascita della crittografia a chiave pubblica, si intuì il problema di associare l'identità delle persone ai documenti, ovvero di istituire un equivalente elettronico della firma cartacea per attestare l'autenticità dei messaggi, ed evitare così un attacco di tipo '*man in the middle*'.
Digitalizzare la nostra firma manuale non è una buona soluzione, perchè, se sulla carta è difficile falsificarla, è molto facile copiarne la digitalizzazione da un documento elettronico già firmato e incollarla su un altro, risultando la copia indistinguibile originale.
Abbiamo bisogno, perciò, di firme digitali che non siano legate solo al firmatario, ma anche al messaggio da firmare.
Inoltre, una firma digitale, poichè attesta l'autenticità di un'entità, deve essere facilmente verificabile dagli altri soggetti.
Lo schema di firma si divide dunque in: **processo di firma** e **processo di verifica**.
Tre sono le funzioni base offerte da questa tecnica:

- **Integrità:** la firma digitale deve essere propria di un documento.
  Se un antagonista modifica il documento, la firma originale non deve più risultare valida.
- **Autenticità del mittente:** la firma digitale non può essere falsificata e il mittente non può essere contraffatto.
- **Non ripudiabilità:** chi firma digitalmente un documento non può negare di averlo fatto.

Un crittosistema a chiave pubblica può facilmente generare una firma che rispetti queste condizioni.

### 2.2.1 Schema di firma RSA

Alice, per firmare un documento $m$ con la firma RSA, deve compiere i seguenti passi:

1. Alice genera $p$ e $q$, primi grandi e calcola $n = pq$.
2. Sceglie $e_A$ tale che $1 < e_A < \varphi(n)$ e $MCD(e_A, \varphi(n)) = 1$, calcola $d_A$ tale che $e_A d_A \equiv 1 \pmod{(\varphi(n))}$.
3. Alice pubblica $(e_A, n)$ e tiene $d_A, p, q$ privati.
4. La firma di Alice è :
$$y \equiv m^{d_A} \pmod{n}$$
La coppia $(m, y)$ è resa pubblica.

Bob, ricevendo il messaggio $m$, verifica l'autenticità della firma in questo modo:

1. Ottiene i parametri $(e_A, n)$ di Alice.
2. Calcola $z \equiv y^{e_A} \pmod{n}$.
Se $z = m$, la firma è valida, altrimenti la firma non è autentica.

**Esempio:**[4]

Alice genera $p = 7927$, $q = 6997$ e calcola

$$n = pq = 55465219 \qquad \varphi(n) = 7926 \text{ x } 6996 = 55450296$$

.

Alice sceglie $e_A = 5$ e risolve $e_A d_A \equiv 5d \equiv 1 \pmod{\varphi(n)}$. Risulta $d_A = 44360237$.
Chiave pubblica di Alice: $(n, e_A) = (55465219, 5)$.
Chiave privata: $d_A = 44360237$.
Sia $m = 31229978$ il messaggio da firmare, Alice calcola la firma:

$$y \equiv m^{d_A} \equiv 31229978^{44360237} \equiv 30729435 \pmod{n}$$

Bob per verificare l'autenticità della firma calcola:

$$z \equiv y^{e_A} \equiv 30729435^5 \equiv 31229978 \pmod{n}$$

e verifica $z = m$.

[4][3] pp. 434-435

Poichè la firma $y$ è propria del documento $m$, un antagonista non può incollare $y$ su un altro messaggio $m_1$ e usare la coppia $(m_1, y)$, in quanto ${y^{e_A}} \not\equiv m_1 \pmod{n}$.
Eva necessita di un $y_1$ tale che ${y_1}^{e_A} \equiv m_1 \pmod{n}$, ma questo è equivalente alla decifrazione del messaggio $m_1$, cifrato con l'algoritmo RSA, in modo da ottenere il testo in chiaro $y_1$, problema che si ritiene difficile.
Eva, allora, potrebbe scegliere prima $y_1$ e, in seguito, il messaggio $m_1 \equiv {y_1}^{e_A} \pmod{n}$; in questo modo la firma risulterebbe originale, tuttavia, poiché Eva non può controllarne il contenuto, è assai improbabile che $m_1$ risulti un messaggio di senso compiuto.
In ogni modo, per escludere quest'ultimo caso, e per una maggior efficienza nell'uso della firma digitale, si introducono le **funzioni hash crittografiche**.

**Definizione 2.1.** Sia $h : \{0,1\}^* \longrightarrow \{0,1\}^n$, una funzione definita dall'insieme delle stringhe di lunghezza arbitraria * all'insieme delle stringhe di lunghezza fissata $2^n$.
Si dice che $h$ è una funzione hash crittografica se verifica le seguenti proprietà:

1. Dato un messaggio $m$, è facile e rapido calcolare $h(m)$.
2. Dato un $y \in \{0,1\}^n$ è computazionalmente impossibile trovare un $m'$ tale che $h(m') = y$. (*Resistenza debole alle collisioni*)
3. È computazionalmente impossibile trovare due messaggi $m_1$ e $m_2$ tali che $h(m_1) = h(m_2)$. (*Resistenza forte alle collisioni*)

*Osservazione* 1. $h$ verifica la seconda proprietà $\Leftrightarrow$ $h$ unidirezionale.

*Osservazione* 2. La resistenza forte alle collisioni (3) $\Rightarrow$ la resistenza debole alle collisioni (2).
Si dimostra che non vale il viceversa.

Le funzioni hash sono particolarmente utili nelle firme digitali.
Poichè una firma digitale $y$ è spesso lunga quanto il documento che deve essere firmato $m$, risulta più conveniente firmare $h(m)$, con $h$ funzione hash crittografica. Infatti è più efficiente firmare una stringa di lunghezza fissata pari a $n$, piuttosto che un lungo documento.

Inoltre, firmando $h(m)$, si evita la possibilità che Eva trovi un testo $m_1$ corrispondente alla firma $y_1$; infatti, in questo caso, $m_1$ dovrebbe soddisfare: $y_1 = (h(m_1))^{d_A} \pmod n$. Eva dovrebbe, dato $y_1$, trovare un messaggio $m_1$ tale che $h(m_1) = {y_1}^{e_A} \pmod n$, ovvero trovare la controimmagine di $h(m_1)$, ma questo è impossibile poichè $h$ é unidirezionale per la proprietà (2).

### 2.2.2 Attacco del compleanno alle firme digitali

Consideriamo, ora, un attacco alle firme digitali: l'***'attacco del compleanno'***.
Il nome deriva dal noto '*Paradosso del compleanno*'[5]:

> Se in una stanza ci sono 23 persone, la probabilità che due di esse festeggino il compleanno nello stesso giorno è di poco superiore al 50%.
> Se ce ne sono 30, la probabilità è circa il 70%.
> Questo, che può sembrare incredibile, è tuttavia un fatto dimostrato.
> Generalizzando il problema, si ha:
> Dati N oggetti e due liste di lunghezza $r$, se ogni persona di ciascuna lista sceglie un oggetto, la probabilità che lo stesso oggetto sia scelto da due persone in liste diverse è: $1 - e^{-\lambda}$, con $\lambda = \frac{r^2}{N}$, mentre la probabilità che ci siano esattamente $x$ collisioni è circa $\frac{(\lambda^x e^{-\lambda})}{x!}$.

Nel caso delle firme digitali la situazione è la seguente:
Alice deve firmare un documento $m$ usando uno schema nel quale si firma un hash del documento, $h(m)$. Supponiamo che l'immagine della funzione hash sia $\{0,1\}^{50}$.
Eva vuole ingannare Alice e farle firmare l'hash di un altro documento $h(m_1)$. La probabilità che $h(m_1) = h(m)$ è una su $2^{50}$ (circa una su $10^{15}$).
Eva agisce in questo modo:
trova 30 punti nel documento $m$ dove può apportare piccole modifiche (aggiungere uno spazio, riformulare una frase,...) senza modificare il significato. In ognuno di questi punti

[5] [1] pp. 204-207

Eva può decidere se mantenere la versione originale o introdurre la modifica. Eva, quindi, ha a disposizione $2^{30}$ documenti sostanzialmente equivalenti al documento originale. Calcola ora la funzione hash di ognuno di questi documenti e conserva i valori corrispondenti. Analogamente Eva formula $2^{30}$ varianti del *suo* documento $m_1$, e ne calcola i corrispondenti hash.
Consideriamo ora il problema del compleanno generalizzato con $N = 2^{50}$ e $r = 2^{30}$.
Si ha: $\lambda = \frac{r^2}{N} = 2^{10} = 1024$. Pertanto, con probabilità circa $1 - e^{-1024} \sim 1$ una versione del documento $m$ e una versione del documento $m_1$ hanno lo stesso hash.
A questo punto, a Eva non rimane altro che far firmare ad Alice questa versione del documento $m$; poichè entrambi i documenti hanno lo stesso valore di $h$, anche la firma risulterà la medesima: Eva dispone perciò di una firma autentica del contratto fraudolento.

Per prevenire questo tipo di attacco, occorre lavorare con $n$ molto più grandi di 50, così da rendere sempre più alto il numero di variazioni da apportare ai documenti, per ottenere probabilità significative di successo; oppure, basterebbe che Alice, appena prima di firmare il contratto, decida di inserire lei stessa una modifica (anche insignificante) al testo; in questo modo Eva dovrebbe trovare una versione del documento fraudolento che abbia lo stesso hash del documento (unico) firmato da Alice, il che è essenzialmente impossibile.

# Capitolo 3

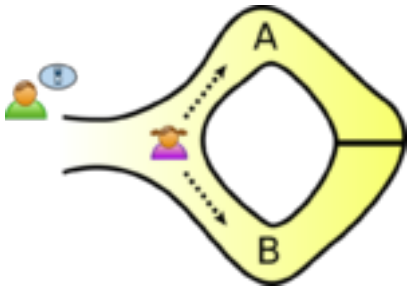
# Tecniche a Conoscenza Zero

Si consideri il problema dell'autenticazione: un server ha bisogno che Alice fornisca una prova della sua identità prima di concedere l'accesso a certe informazioni private. Questo problema viene spesso risolto richiedendo ad Alice un PIN, nomi utenti e password, . . .
Se il canale utilizzato non è sicuro chiunque può intercettare e scoprire le informazioni private di Alice e sfruttarle poi per i suoi fini. Anche nell'ipotesi in cui il canale è stato reso sicuro in qualche modo, resta un problema: ci si può fidare del server che richiede l'autenticazione? Fornendo le proprie credenziali (password o altro) si stanno fornendo al server delle informazioni sensibili, che potrebbero essere utilizzate contro l'utente in un secondo momento.
Per risolvere questa problematica è necessario un metodo che permetta di utilizzare numeri segreti senza tuttavia rivelare informazioni che potrebbero essere utilizzate da un avversario.
Nascono così le tecniche a conoscenza zero (*zero-knowledge*).

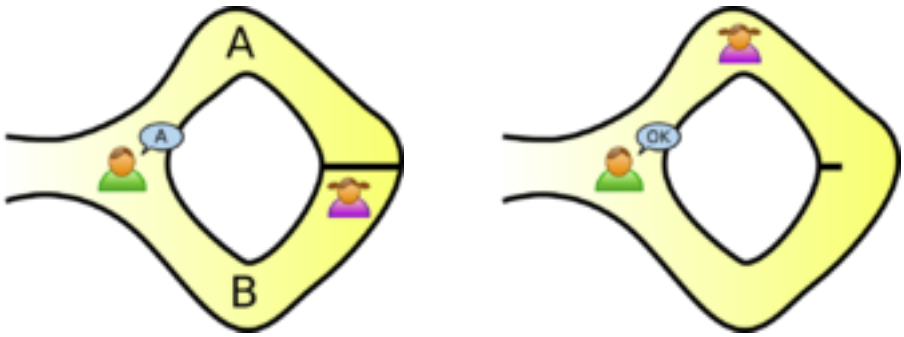
Vediamo un esempio esplicativo che permette di capire come questo sia possibile:

Supponiamo che Alice conosca una parola magica che permetta l'apertura di una porta, posta all'interno di una galleria, che impedisce il passaggio da una parte all'altra.

Bob vuole comprare il segreto, ma prima necessita di una prova che lei ne sia davvero al corrente.
Alice mette in atto un sistema, grazie al quale può provare di conoscere la parola, senza rivelarla a Bob.
Inizialmente Bob aspetta fuori dalla galleria, mentre Alice entra. Siano A e B i due sentieri percorribili, sinistro e destro, come in figura.




Alice sceglie a caso uno dei due sentieri, quindi Bob entra nella caverna e grida 'A', oppure 'B', indicando il sentiero che Alice dovrà percorrere per tornare indietro.
Se Alice conosce veramente la parola magica, è facile: apre la porta, se necessario, e ritorna attraverso il sentiero richiesto.




(a) (b)

Supponiamo, invece, che Alice abbia mentito e non conosca la parola segreta, poichè deve scegliere la direzione da prendere prima che Bob faccia la sua richiesta, ha solo il 50% di probabilità di ingannare il compagno.
Pertanto, se Alice esce dalla parte giusta 10 volte di seguito esiste una sola possibilità su $2^{10} = 1024$, che non sappia effettivamente aprire la porta.

Inoltre, supponiamo che Bob voglia ingannare Alice e riprenda tutta la scena con una telecamera; dalla registrazione non è in grado di acquisire alcuna informazione su come aprire la porta, nè su come convincere qualcuno di saperlo fare, dopotutto, non è neanche sicuro che Alice vi sia passata attraverso!
Bob, quindi, non ottiene nessuna informazione utile che possa essere trasmessa ad altri.

*Osservazione* 3. Bob non ottiene una prova in senso matematico che Alice sia in grado di aprire la porta, ma, tramite una serie di sfide e risposte, ottiene una significativa evidenza sperimentale.

Alice ha dunque provato a Bob di essere a conoscenza di un segreto, con probabilità alta a piacere (basta solo aumentare il numero delle prove), senza tuttavia fornirgli alcuna informazione utile per scoprire il segreto.
Questa si chiama anche '*dimostrazione a conoscenza zero*'.

Una dimostrazione a conoscenza zero deve soddisfare tre proprietà:

1. **Completezza:** data un'affermazione vera, un onesto dimostratore può convincere un onesto verificatore della sua veridicità.

2. **Correttezza:** data un'affermazione falsa, la probabilità che un dimostratore imbroglione ha di convincere un verificatore onesto che essa sia vera può essere resa bassa a piacere.

3. **Conoscenza zero:** data un affermazione vera, nessun verificatore imbroglione potrà sapere altro che tale informazione.

Studiamo ora il caso concreto del '***Protocollo di Feige-Fiat-Shamir***'.

## 3.1 Protocollo di Fiat-Shamir

Il protocollo di Fiat-Shamir è il primo esempio noto di protocollo a conoscenza zero. Nasce nel 1988 da un'idea di Amos Fiat e Uriel Shamir[1].
Fiat-Shamir è un protocollo che consente l'autenticazione sicura di un utente senza fornire, come ogni protocollo zero-knowledge, nessuna informazione oltre quella strettamente richiesta, ossia oltre l'identità dell'entità che deve essere autenticata.
La sicurezza di questo algoritmo si basa sulla difficoltà computazionale del calcolo della radice quadrata di un intero modulo n, con fattorizzazione ignota.

### Problema delle radici quadrate modulo n

**Definizione 3.1.** Sia $n = pq$ con $p$ e $q$ primi dispari diversi. Per *Problema dei Residui Quadratici modulo n* si intende il problema di stabilire se, dato $a \in \mathbb{Z}_n$ , l'equazione $x^2 \equiv a \pmod{n}$ ha soluzione.
Si definisce *Problema delle Radici Quadrate modulo n* il problema di calcolarne le eventuali soluzioni.

Si dimostra che: **risolvere il problema delle radici quadrate modulo $n$ è equivalente a trovare la fattorizzazione di $n$.**

Il protocollo vede coinvolte tre parti: Alice che vuole provare la sua identità, l'utente (o più in generale il server) Bob che vuole accertare l'identità di Alice e una terza parte fidata, al di sopra della parti, che chiameremo Testimone *(Trusted Third Party)*, che svolge il ruolo di garante.

L'algoritmo di Fiat-Shamir si svolge come segue:

1. Il Testimone genera in maniera casuale due primi $p$ e $q$, che mantiene segreti, e pubblica il loro prodotto $n = pq$.

[1] [7]

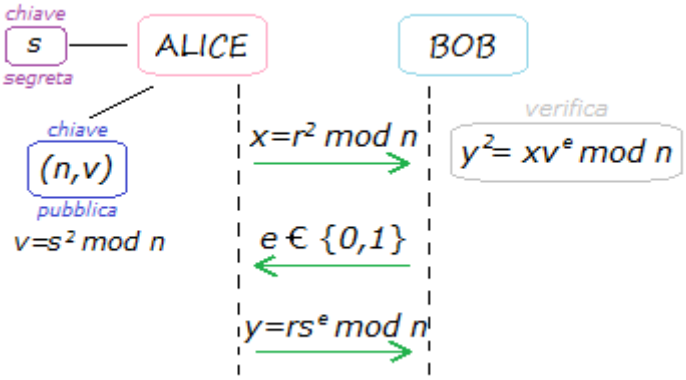
2. Alice sceglie un numero intero $1 \leq s \leq n-1$, primo con $n$. Questa è la sua chiave segreta.
3. Alice calcola $v = s^2 \pmod{n}$
4. Il Testimone registra $(n, v)$ come chiave pubblica di Alice.

**Protocollo:**

1. Alice sceglie un numero casuale $r \in \mathbb{Z}_n$ e invia a Bob $x = r^2 \pmod{n}$
2. Bob invia ad Alice un *numero di sfida* $e \in \{0, 1\}$ (*challenge)*
3. Alice risponde alla sfida calcolando e inviando a Bob $y = rs^e \pmod{n}$, ovvero:
   - Se $e = 0$ Alice invia $y = r \pmod{n}$
   - Se $e = 1$ Alice invia $y = rs \pmod{n}$

   Solo Alice che conosce la chiave segreta $s$ può calcolare correttamente $y$.
4. Bob verifica la correttezza della risposta utilizzando la chiave pubblica di Alice $(n, v)$ attraverso l'uguaglianza $y^2 \equiv xv^e \pmod{n}$
   - Se $e = 0$ Bob trova $y^2 \equiv x \equiv r^2 \pmod{n}$
   - Se $e = 1$ Bob trova $y^2 \equiv xv \equiv r^2 s^2 \pmod{n}$

La figura sottostante ci aiuta a capire questo meccanismo:
In questo modo, Bob può, iterando il procedimento fin quando non si sente sicuro con probabilità $\left(1 - \frac{1}{2^k}\right)$, convincersi che se Alice continua a rispondere correttamente alla sua sfida, è perché conosce l'informazione segreta $s$, e quindi che la sua identità è autentica.
Ovviamente **una sola iterazione non basta ad assicurare Bob dell'onestà di Alice**, in questo caso, infatti, Eva può tentare di impersonare Alice pur non conoscendo la chiave $s$.
Per poterlo fare Eva deve saper rispondere correttamente nei due casi:




- $e = 0$
- $e = 1$

Nel primo caso Eva può generare casualmente $r$ e inviare $x = r^2 \pmod{n}$ a Bob. Seguendo il protocollo, Bob invia un bit di sfida $e$; supponiamo $e = 0$.
In questo caso Eva può rispondere alla sfida: infatti non ha bisogno della chiave segreta di Alice per calcolare $y = rs^e = r \pmod{n}$.
Bob, una volta ricevuto $y$, verifica correttamente che $y^2 = x \pmod{n}$, ed Eva riesce efficacemente ad imbrogliare Bob.
Tuttavia se $e = 1$, Eva non riesce a calcolare $y = rs^e \pmod{n}$ poichè non possiede il segreto $s$ e quindi non può rispondere correttamente alla sfida di Bob.
Per superare la sfida nel caso $e = 1$, Eva può generare casualmente $r$ e inviare a Bob $x = \frac{r^2}{v}$. Bob invia un bit di sfida $e$.
In questo caso Eva risponde correttamente solo se $e = 1$: infatti, invia $y = r \pmod{n}$ (non conosce la chiave segreta). Bob verifica che $y^2 = \frac{r^2}{v} v \pmod{n} = r^2 \pmod{n}$.
Se invece $e = 0$, Eva non riesce a imbrogliare Bob, in quanto, in questo caso,
$y^2 = r^2 \neq x = \frac{r^2}{v} \pmod{n}$. Per superare correttamente la verifica, Eva dovrebbe calcolare $y = \sqrt{\frac{r^2}{v}} \pmod{n}$, ma questa è un'operazione computazionalmente difficile.
Dunque, in entrambi i casi, Eva riuscirebbe a superare solo una delle due sfide, ma mai entrambe contemporaneamente, perchè non conosce in anticipo il valore di $e$.
In conclusione, se il protocollo viene eseguito una sola volta, la probabilità che un utente fraudolento riesca ad identificarsi al posto di Alice è $\frac{1}{2}$. Una sola iterazione quindi non

basta a garantire la sicurezza, che aumenta con il numero di esecuzioni; alla $k - esima$ iterazione la probabilità che Eva riesca ad identificarsi al posto di Alice sarà $P = \frac{1}{2^k}$.

**Lemma 3.1.1.** *Per $k > 1$ arbitrario il protocollo di Fiat-Shamir è una dimostrazione a conoscenza zero.*

*Dimostrazione:* Si mostra che: se Bob simula il protocollo, senza la partecipazione di Alice, ottiene la stessa distribuzione dei messaggi, come se avesse eseguito il protocollo con Alice.
Questo prova che Bob non ottiene nulla di nuovo dallo scambio dei messaggi con Alice, perchè ne può produrre la distribuzione egli stesso, senza nessuna informazione segreta aggiunta.
Analizziamo ora la distribuzione dei messaggi:

- $x$ è radice di un numero casuale in $\{1, 2, \cdots, n-1\}$
- $e \in \{0, 1\}$ è un bit casuale
- la risposta $y = rs^e \pmod{n}$ è un numero casuale in $\{1, \cdots, n-1\}$, perché $s$ è primo con $n$

Bob può simulare il protocollo come segue:

1. Sceglie un numero casuale $y \in \{1, \cdots, n-1\}$ ed $e \in \{0, 1\}$.
2. Calcola $x = y^2 v^{-e} \pmod{n}$
3. Calcola $y^2 = xv^e \pmod{n}$, come nel protocollo originale

Bob ha dunque generato:

- $x$, radice di un numero casuale $\{1, \cdots, n-1\}$
- $e$, numero casuale $\in \{0, 1\}$
- $y$, risposta casuale in $\{1, \cdots, n-1\}$

Si ha: **la simulazione del protocollo genera la stessa distribuzione dei messaggi del protocollo originale**. □

***Esempio:***[2]

Siano: $p = 17$, $q = 23$, $n = pq = 391$.
Alice sceglie $s = 123$ e calcola $v = s^2 1 \pmod{n} = 271$
Chiave segreta di Alice: $s = 123$.
Chiave pubblica di Alice: $(271, 391)$
Protocollo:
Alice sceglie $r = 271$, calcola $x = r^2 \pmod{n} = 324$, e lo manda a Bob.
Bob invia ad Alice $e = 1$.
Alice calcola e manda a Bob: $y = rs \pmod{n} = 98$.
Bob calcola $y^2 \pmod{n} = 220$ e accetta l'autenticazione di Alice poichè
$y^2 \pmod{n} = 220 = xv \pmod{n}$.

Vediamo ora, come Bob può simulare questo scambio di messaggi:
Bob non conosce una radice quadrata di 271 (mod 391), quindi parte scegliendo la risposta $y = 271$.
Sceglie $e = 0$, e calcola $x = y^2 \pmod{n} = 324$.
Verifica: $xv^e \pmod{n} = x \pmod{n} = 324 = y^2 \pmod{n}$

## 3.2 Schema di firma di Fiat-Shamir

Rispetto alla firma RSA l'algoritmo di Fiat-Shamir consente di ridurre notevolmente (circa un quarto) il numero di moltiplicazioni modulari richieste.
Il procedimento è analogo a quello dello schema di identificazione:
Viene scelto un $n$ grande prodotto di due primi $p, q$, viene generata la chiave privata $(s_1, .., s_k)$ e la chiave pubblica $(n; v_1, .., v_k)$ con $v_i = \pm\frac{1}{s_i{}^2} \pmod{n}$.

[2] [4] pp. 245-247

1. Alice sceglie in modo casuale $l$ numeri interi, compresi tra 1 e $n$, $r_1, .., r_l$ e calcola $x_1, .., x_l$ tramite la relazione $x_i = {r_i}^2 \pmod{n}$.

2. Alice mette in relazione tramite la funzione hash $H$ il messaggio $m$ e la stringa di $x$: $H(m; x_1, .., x_l)$ generando una sequenza di bit $b_{ij}$ con $1 \leq i \leq k, 1 \leq j \leq l$.

3. Alice calcola $y_1, .., y_l$ con $y_i = r_i({s_1}^{b_{i1}} \cdots {s_k}^{b_{ik}}) \pmod{n}$.

4. Alice manda a Bob il messaggio $m$ e la firma: la stringa $b_{ij}$, i valori $y_i$ e gli $x_i$.

5. Bob calcola $z_1, .., z_l$ con $z_i = {y_i}^2({v_1}^{b_{i1}} \cdots {v_k}^{b_{ik}}) \pmod{n}$ (si noti che $z_i$ può essere uguale a $x_i$ se $b_{i,1} = \cdots = b_{i,k} = 0$).

6. Bob verifica che i bit della funzione $H(m; x_1, .., x_l)$ corrispondano a quelli ricevuti da Alice.

Come per l'algoritmo d'identificazione, la sicurezza di questo sistema è proporzionale a $1/2^{kl}$ e dipende dalla difficoltá di fattorizzazione di $n$.
Fiat e Shamir esservarono che falsificare una firma risulta essere possibile se la complessità della fattorizzazione di $n$ risulta essere inferiore a $2^{72}$. Per questo motivo suggerirono $k = 9$ e $l = 8$.

## 3.3 Protocollo di Feige-Fiat-Shamir

Il vantaggio principale del protocollo di Feige-Fiat-Shamir, o ***FFS***, che è una generalizzzione di quello precedente, riguarda la presenza di un ridotto numero di operazioni algebriche risultando così piuttosto veloce nell'implementazione.
L'***FFS*** è un protocollo di sola identificazione e, diversamente dall'RSA, non può essere impiegato nella codifica di dati.
L'algoritmo si svolge come segue:

1. Il Testimone genera in maniera casuale due primi segreti $p$ e $q$ tali che $n = pq$ sia computazionalmente impossibile da fattorizzare, e pubblica $n$.

2. Alice sceglie casualmente $k$ numeri interi compresi tra 1 e $n-1$ $s_1, .., s_k$, tali che $MCD(n, s_i)\ \forall\ i = 1, .., k$, e $b_1, .., b_k$ bits casuali.
3. Alice calcola $\forall\, i = 1, .., k\ v_i = (-1)^{b_i}(s_i{}^2)^{-1} \pmod n$
4. Il Testimone registra $(n, v_1, .., v_k)$ come chiave pubblica di Alice, mentre la stringa $(s_1, .., s_k)$ sarà la sua chiave segreta.

**Protocollo:**

1. Alice sceglie un intero casuale $r$ con $1 \leq r \leq n-1$ e un bit $b$, calcola e invia a Bob $x = (-1)^b r^2 \pmod n$
2. Bob manda ad Alice una stringa binaria lunga $k$: $e_1, ..., e_k$.
3. Alice calcola $y = r(s_1{}^{e_1} \cdots s_k{}^{e_k}) \pmod n$ e la manda a Bob.
4. Bob calcola $z = y^2(v_1{}^{e_1} \cdots v_k{}^{e_k}) \pmod n$ e verifica che $z = \pm x$ e $z \neq 0$.

Il procedimento verrà iterato finchè Bob non si convincerà che Alice conosca $s_1, .., s_k$. All'iterazione $t-esima$ la possibilità che Alice avrà di ingannare Bob sarà $\frac{1}{2^{kt}}$ (per questo algoritmo è stato consigliato k=5 e t=4).

**Lemma 3.3.1.** *Per k fissato e $t > 1$ arbitrario il protocollo di Feige-Fiat-Shamir è una dimostrazione a conoscenza zero.*

La ragione intuitiva, ma non rigorosa, per cui il protocollo non rivela alcuna informazione riguardo $s_j$ è che $x_i$ sono radici casuali, e ogni $y_i$ ne contiene un numero indipendente e casuale, che maschera il valore di $s_j$. Tutti i messaggi inviati da Alice a Bob sono così numeri casuali, con distribuzione di probabilità uniforme.

***Esempio:***[3]

Siano: $p = 683$, $q = 811$, $n = pq = 553913$, $k = 3$, $t = 1$.

[3] [3] pp. 411

Alice seleziona casualmente 3 interi: $s_1 = 157$, $s_2 = 43215$, $s_3 = 4646$, e 3 bits $b_1 = 1$, $b_2 = 0$, $b_3 = 1$.
Calcola: $v_1 = 441845$, $v_2 = 338402$, $v_3 = 124423$. Chiave pubblica di Alice: (553913;441845,338402,124423)
Chiave segreta di Alice: (157,43215,4646).
Protocollo:
Alice sceglie $r = 1279$, $b = 1$, calcola $x = 25898$, e lo manda a Bob.
Bob invia ad Alice un vettore di 3 bits: (0,0,1).
Alice calcola e manda a Bob: $y = rs_3 \pmod{n} = 403104$.
Bob calcola $z = y^2 v_3 \pmod{n} = 25898$ e accetta l'autenticazione di Alice poichè $z = +x$ e $z \neq 0$.

### 3.3.1 Sicurezza del protocollo di Feige-Fiat-Shamir

- *Probabilità di falsificazione:* Purchè la fattorizzazione di $n$ sia difficile, il miglior attacco di impersonificazione ha probabilità $2^{-kt}$ di successo.

- *Presupposto rischiesto per la sicurezza:* La sicurezza dell'algoritmo si basa sulla difficoltà di estrarre radici quadrate modulo un grande numero intero composto $n$, con fattorizzazione ignota. Come già visto, questo è equivalente alla fattorizzazione di $n$.

- *Parametri:* Sceglire $k, t$ tali che $kt = 20$ permette 1 possibilità di successo su un milione di impersonificazione da parte di un attaccante. Specifici parametri potrebbero essere:

  - per sicurezza $2^{-20}$: $k = 5$ e $t = 4$
  - per sicurezza $2^{-30}$: $k = 6$ e $t = 5$

**Attenzione:** si può vedere che nel caso in cui $t = 1$, ossia con una sola iterazione, anche se incrementiamo $k$ affinchè il prodotto $kt$ rimanga costante, il protocollo non è più un sistema a conoscenza zero!

# Bibliografia


[1] **W.Trappe, L.C.Washington**, '*Crittografia con elementi di teoria dei codici*', Pearson-Pentice Hall, 2009

[2] **Baldoni, Ciliberto, Piacentini Cattaneo**, '*Aritmetica, Crittografia e Codici*', Springer-Verlag, Milano, 2006

[3] **A.Menezes, P.C. van Oorschot, S.A. Vanstone**, '*Handbook of Applied Cryptography*', CRC Press, 1997

[4] **J.Buchmann**, '*Introduction to Cryptography*', Springer, 2001

[5] **W. Diffie, M.E. Hellman**, '*New Directions in Cryptography*', Transactions on Information Theory Vol IT22 No 6, pp 644-654 1976

[6] **R.L. Rivest, A. Shamir, L. Adleman**, '*A Method for Obtaining Digital Signature and Public-Key Cryptosystems*', Communication of the ACM Vol 21 No 2 pp 120-126, 1978

[7] **A. Fiat and A. Shamir**, *How to prove yourself: Practical solutions to identification and signature problems*, Advances in Cryptology: CRYPTO'86 (A. M. Odlyzko ed.), LCNS 263, Springer-Verlag, pp. 181-187, 1987

# Ringraziamenti

*Desidero ringraziare il professor Aliffi, che mi ha seguito con attenzione e interesse nella stesura nella tesi, per la pazienza e il prezioso aiuto.*
*Ringrazio i miei genitori perchè mi hanno dato la possibilità di arrivare fin qui e tutta la mia famiglia perchè mi è stata accanto e mi ha sopportato durante il mio percorso.*
*Un grazie speciale a Giacomo che cammina con me da diversi anni e che mi spinge sempre ad andare avanti.*
*A 'Quelle della Terra di Mezzo', a Pazzaglia e a tutti i miei amici, l'abbraccio più grande per essere semplicemente fantastici!*