- **CERTIFICATI e PKI**

  Certificato: consiste in una chiave pubblica e in un identificatore del suo proprietario, il tutto firmato da una terza parte fidata.

  Formato X.509: è uno standard ITU-T per le infrastrutture a chiave pubblica (PKI). La struttura di un certificato digitale X.509: versione, numero seriale, ID dell'algoritmo di firma, nome di chi ha emesso il certificato, periodo di validità, nome dell'utente, informazioni sulla chiave pubblica dell'utente, identificatore (unico) dell'emittente, identificatore (unico) dell'utente, estensioni, firma dei campi precedenti.

  CA: I certificati sono emessi da un'autorità di certificazione (CA), una qualsiasi amministrazione centrale fidata, che garantisce per l'identità di coloro per cui emette i certificati e a cui associa una data chiave garantendone l'autenticità. I certificati hanno le seguenti caratteristiche: 1. Ogni utente con accesso alla chiave pubblica dell'autorità di certificazione può verificare la chiave pubblica dell'utente che è stata certificata. 2. Nessuno, ad eccezione della chiave di certificazione, può modificare il certificato senza essere rilevato.

  Revoca dei certificati: Ogni certificato ha un periodo di validità. In genere viene emesso un nuovo certificato appena prima della scadenza del precedente. Si può revocare un certificato prima della scadenza per: 1. la chiave privata dell'utente è stata violata. 2. l'utente non è più certificato da questa autorità di certificazione. 3. il certificato dell'autorità di certificazione è stato violato.

- **CIFRARI A SOSTITUZIONE**

  L'idea base dei cifrari a sostituzione è quella di sostituire un simbolo di testo in chiaro con un simbolo di testo cifrato secondo uno schema regolare.

  Esistono diversi tipi di cifrari a sostituzione: cifrari che operano su singole lettere, **cifrari a sostituzione semplice**; operano su gruppi di lettere più grandi è detto **cifrari poligrafici**. Un **cifrario monoalfabetico** utilizza sostituzioni fisse sull'intero messaggio mentre un **cifrario polialfabetico** usa invece differenti schemi di sostituzione ed ogni lettera del testo in chiaro è trattata con un differente alfabeto secondo uno schema stabilito dalla chiave segreta. **CIFRARIO DI CESARE**: La cifratura più semplice è quella utilizzata da Giulio Cesare che prevede la sostituzione di ciascuna lettera con la lettera che si trova a tre posizioni di distanza nell'alfabeto. **CIFRARIO DI HILL**: cifrario a sostituzione polialfabetica basato sull'algebra lineare. Un blocco n di lettere è quindi considerato come uno spazio vettoriale di dimensione n, e moltiplicato per una matrice n x n, modulo 26. **Cifrario di Vigenère**. In questo schema, l'insieme di regole di sostituzione monoalfabetica è costituito da 26 cifrature di Cesare con scorrimenti da 0 a 25. Per comprendere e utilizzare lo schema si può costruire una matrice chiamata tabella di Vigenére. Il processo di crittografia è semplice: data la chiave x e la lettera in chiaro y, la lettera cifrata si trova all'intersezione della riga x e della colonna y.

  TIPI DI ATTACCHI : **CRITTOANALISI: Known Ciphertext Attack:** l'avversario conosce solo il testo cifrato.**Known Plaintext Attack:** conosce il testo cifrato e una o più coppie di testo in chiaro/testo cifrato. **Chosen Plaintext Attack:** conosce il testo cifrato e un messaggio in chiaro scelto da chi esegue l'analisi crittografica, insieme al corrispondente testo cifrato. **Chosen Ciphertext Attack:** conosce il testo cifrato e un testo cifrato scelto da chi esegue l'analisi crittografica insieme al corrispondente testo in chiaro.

  Numero di chiavi generica: il numero di chiavi è molto alto $|K| = 26! \approx 4 \cdot 1026$, ovvero si hanno 26 permutazioni dell'alfabeto composto di 26 lettere.

  Sostituzione con alfabeto shiftato: 25. Sostituzione con parole chiave lunga n caratteri (tutti diversi): n!

- **DES**

  Il DES è un algoritmo di cifratura scelto come standard dal FIPS 1976 e in seguito diventato di utilizzo internazionale. Si basa su un algoritmo a chiave simmetrica con chiave a 64, 56 utili e 8 di controllo.

  Cifratura, decifratura: il testo in chiaro viene diviso in blocchi di 64 bit, ognuno dei quali viene cifrato generando 64 bit di testo cifrato. Innanzitutto il testo in chiaro di 64 bit viene sottoposto ad una permutazione iniziale IP che dispone i bit per produrre un input permutato. Questa operazione è seguita da 16 ripetizioni della stessa funzione di permutazione e sostituzione. Per la decodifica in pratica vengono eseguite in senso inverso usando la stessa chiave. **Espansione chiave**: Come input dell'algoritmo viene utilizzata una chiave di 64 bit. I bit della chiave sono numerati da 1 a 64; un bit ogni 8 viene ignorato. La chiave viene innanzitutto permutata. La chiave risultante di 56 bit viene poi trattata come 2 quantità di 28 bit. PC1: elimina i bit di parità e compie la permutazione sui bit restanti della chiave a 56 bit. PC2:

comprime i 56 bit in input con l'eliminazione di 8 bit dando in output 48 bit dopo aver effettuato un'altra permutazione.

EFFETTO VALANGA: DES ha un forte effetto valanga, ovvero una piccola variazione nel testo in chiaro o nella chiave produce una grande variazione nel testo cifrato.

SICUREZZA DEL DES: Con chiavi di 56 bit esistono 2^{56} possibili chiavi, ovvero circa $7{,}2x10^{16}$ chiavi. Pertanto, date queste dimensioni, un attacco a forza bruta sembra poco pratico. L'algoritmo DES si è dimostrato insicuro nel 1998 quando la EFF annunciò di averlo violato e da allora è stato sostituito.

DES doppio

prevede due stadi di crittografia con due chiavi. Dato un testo in chiaro P e due chiavi di crittografia K1 e K2, il testo cifrato viene generato nel seguente modo: C=E(K2, E(K1,P)) e la decrittografia P=D(K1,D(K2,C)). Per DES questo schema prevede una chiave della lunghezza di 56x2=112 bit con un notevole incremento della potenza crittografica. Il DES Doppio è più resistente agli attacchi di forza bruta, fornendo uno spazio delle chiavi maggiore rispetto al DES singolo, e cifra in messaggio in chiaro due volte con due chiavi diverse.

KPA: Il DES Doppio non è resistente a un attacco di tipo known plaintext, chiamato **Meet-in-the-middle**. L'algoritmo si basa sulla operazione: C=E(K2, E(K1,P)) allora X=E(K1,P)=D(K2,C). Data una coppia nota (P,C), l'attacco procede nel modo seguente. Innanzitutto si esegue la crittografia di P per tutti i 2^{56} valori di K1, si memorizzano i risultati in una tabella e la si ordina per i valori di X. Si esegue poi la decrittografia di C utilizzando tutti i 2^{56} possibili valori di K2. A mano a mano che vengono prodotti i risultati si confrontano con la tabella alla ricerca di una corrispondenza. Se le due chiavi producono il testo cifrato corretto, si accettano le chiavi come corrette.

Complessità spazio: • 2^{56} righe nella tabella.Complessità tempo: • 2^{56} cifrature per x (costruzione tabella).

- **DES TRIPLO**

  Quando si scoprì che la chiave a 56 bit del DES non era abbastanza lunga da garantire la sicurezza contro attacchi a forza bruta, il TDES fu scelto come modo semplice per aumentare la lunghezza della chiave senza bisogno di cambiare algoritmo. L'uso di tre passaggi è essenziale per prevenire attacchi di tipo meet-in-the-middle che funzionano contro la doppia crittazione DES.

  Il DES viene ripetuto tre volte. In input abbiamo un blocco di testo di 64 bit, la chiave è di 112 bit (DES Triplo a 2 chiavi) oppure di 168 bit (DES Triplo a 3 chiavi), e le fasi sono 48 fasi (essendo il DES formato da 16 fasi).

  - **KPA**: Si costruisce una tabella di dimensione 2^{112} la quale contiene coppie del tipo $((k_1, k_2), DES_{k1}(DES_{k2}(x)))$ che rappresentano l'insieme di tutte le possibili cifrature di x mediante due chiavi.
  - Successivamente si effettuano tutte le 2^{56} possibili decifrature di y con la chiave k_3, e, per ognuna di esse, si ricerca il valore ottenuto nella tabella.
  - Quando la ricerca ha esito positivo si ottiene la chiave cercata che è (k_1, k_2, k_3).

  Complessità spazio: 2^{112} righe nella tabella. **Complessità tempo:** 2^{112} cifrature per x (costruzione tabella),2^{56} decifrature per y; 2^{56} ricerche in tabella.

- **DIFFIE-HELLMAN**

  L'algoritmo Diffie-Hellman risale al 1976 ed è quindi uno dei più antichi algoritmi a chiave pubblica. È particolarmente adatto alla generazione di una chiave segreta tra due corrispondenti che comunicano attraverso un canale non sicuro (pubblico). La sua sicurezza si basa sulla complessità computazionale del calcolo del **logaritmo discreto**.

  Algoritmo: Aldo genera e comunica pubblicamente un numero primo N molto elevato e un generatore g. Dopo Aldo genera un numero casuale a < N e calcola A = g^a mod N. Il numero A viene comunicato pubblicamente a Bruno. In modo del tutto analogo Bruno genera due numeri b e B = g^b mod N e invia B ad Aldo. Aldo calcola il numero k = B^a mod N. Bruno calcola k = A^b mod N. Entrambe le chiavi valgono g^ab mod N.

  Sicurezza: La cosa importante per la sicurezza è che un terzo che intercettasse i quattro numeri N, g, A, B non sarebbe in grado di ottenere k = g^ab non conoscendo nè a nè b. In effetti $a = \log_g(A)$ e $b = \log_g(B)$ ma essendo il calcolo del logaritmo discreto computazionalmente proibitivo come una fattorizzazione è pressochè impossibile calcolare questi logaritmi per numeri di 1024 bit. Non è invece sicuro per gli attacchi **man-in-the-middle**, ovvero quando una terza persona riuscisse a intercettare il numero A e

fingesse di essere la seconda persona. Per garantire l'identità dei corrispondenti, occorre l'**autenticazione**, o un meccanismo **di firma digitale.**

Non è possibile utilizzare g = 2 e p = 17 siccome 2 non è un generatore di Z^*_{17}

- **FUNZIONI HASH**

  Le funzioni hash hanno assunto un ruolo fondamentale per la garanzia dell'integrità dei dati e l'autenticazione dei messaggi. Una funzione hash prende in input un messaggio e restituisce un output denominato **codice hash**. Lo scopo di una funzione hash è quello di produrre una sorta di impronta digitale di un file, messaggio o altro blocco di dati.

  Attacco a compleanno: Si supponga che venga utilizzato un codice hash a 64 bit. Si potrebbe pensare che questa tecnica sia sufficientemente sicura. Per esempio, se un codice hash crittografato C venisse trasmesso con il corrispondente messaggio non crittografato M, un estraneo dovrebbe trovare un messaggio M^I tale che $H(M^I)= H(M)$ per sostituire il messaggio originale e ingannare il destinatario. In media, l'estraneo dovrebbe provare circa 2^{63} messaggi per trovarne uno che produca lo stesso codice hash del messaggio intercettato.

  1. il mittente A si prepara a firmare un messaggio aggiungendo il codice hash di m bit appropriato e crittografando tale codice hash con la propria chiave privata.

  2. l'estraneo genera $2^{m/2}$ varianti del messaggio, ognuna delle quali ha fondamentalmente lo stesso significato.

  3. i due insiemi di messaggi vengono confrontati per trovare una coppia di messaggi che producono lo stesso codice hash.

  4. l'estraneo offre la variante valida ad A per la firma. Questa firma può essere associata alla variante fraudolenta e trasmessa al destinatario. Poiché le due varianti hanno lo stesso codice hash, produrranno la stessa firma;

  Pertanto, se viene utilizzato un codice hash di 64 bit, l'impegno richiesto è solo dell'ordine di 2^{32}.

- **MERKLE - SCHEMA**

  Fu uno dei primi crittosistemi a chiave pubblica creato nel 1978. Non è basato su alcuna assunzione computazionale. Alice genera n chiavi distinte e nasconde ogni chiave in un **puzzle** che contiene le informazioni per il calcolo della chiave. Calcolare la soluzione di 1 puzzle richiede un tempo ragionevole, calcolare quella di più puzzle richiede un tempo elevato.

  Cifratura: Puzzle (x, ID, S){ Scegli una chiave k di 56 bit; Computa y ← CBC-DESk(x, ID, S); return (y, primi 20 bit di k) } **ID**=identificatore password, x=soluzione puzzle, S=valore noto per garantire l'unicità di un puzzle.

  Decifratura: Bob risolve i puzzle e ottiene l'ID e x. Rispedisce ad Alice l'ID.

  Computazione attaccante: Dopo aver visto IDj, cifra DESk(IDj) con tutte le possibili chiavi k e correla le cifrature con tutti i Puzzle visti. Trova così il Puzzle con IDj e decifra il corrispondente DESk(x). Per rendere più sicuro il tutto in fase di **Computazione** di cifratura è consigliato effettuare una cifratura piu complessa.

  TEMPO: Cifratura Alice: O(n), risoluzione 1 puzzle Bob: O(t), Risoluzione di n/2 puzzle Attaccante in media tempo $\theta(t \cdot n)$. Se però $n= \theta(t)$, la cifratura: O(n), decifratura O(n), computazione attaccante $\theta(n^2)$.

- **PASSWORD E AUTENTICAZIONE**

  Password: una sequenza di caratteri alfanumerici utilizzata per accedere in modo esclusivo a una risorsa informatica o per effettuare operazioni di cifratura. Si parla più propriamente di passphrase se la chiave è costituita da una frase o da una sequenza sufficientemente lunga di caratteri. Una password è solitamente associata a un username univoco, al fine di ottenere un'identificazione da parte del sistema a cui si chiede l'accesso.

  Cifratura e Hash: La password può essere salvata sul server e cifrata oppure data in input ad una funzione hash unidirezionale, che la converte in una sequenza di byte. Con questo metodo il sistema memorizza nel file di password una coppia <userid, h(password)>, dove h(password) è una funzione hash della password.

  Two-factor authentication: per riconoscere un utente, di solito si utilizza uno dei tre fattori: qualcosa che l'utente POSSIEDE, qualcosa che l'utente CONOSCE, o qualcosa che l'utente E'. Con questo metodo, servono almeno due dei tre fattori per verificare l'identità (ad esempio carta bancomat e codice pin).

  Challenge-response: In un ambiente di rete il problema principale è la trasmissione della password in modo sicuro al server. Per evitare di rivelare la password sulla rete, sono stati sviluppati i sistemi challenge-response. Nella forma più semplice, il server invia un challenge all'utente, tipicamente una stringa casuale

di byte. L'utente calcola una risposta (response), di solito una funzione che prende in input il challenge ed una password. In tal modo, anche se un intruso cattura una coppia challenge-response valida, non può violare il sistema poiché i challenge successivi sono diversi, quindi anche le corrispondenti risposte.

Sistemi biometrici: riconosce l'utente per una sua proprietà biologica. Esempi: impronta digitale, scansione retina, geometria della mano, dinamica della firma, riconoscimento del volto, riconoscimento della voce.
Vantaggi: Le caratteristiche biometriche non possono essere perse, prestate, rubate o dimenticate. Risulta molto difficile per un individuo falsificare le caratteristiche fisiche di qualcun altro.
Svantaggi: Non garantiscono un'accuratezza del 100%. Le caratteristiche possono mutare nel tempo. I dispositivi biometrici, possono non essere affidabili.

Attacchi alle password: Per scoprire password segrete si devono usare programmi appositi che sviluppano degli attacchi il cui fine è quello di scoprire più password segrete possibili. Molti programmi utilizzano un attacco, chiamato attacco del dizionario, il cui scopo è quello di confrontare tutte le password di un loro dizionario interno con quelle degli utenti contenute in un determinato file, fino a quando non coincidono. Altri programmi, invece, chiamati sniffer cercano di intercettare le password che viaggiano nella rete. Per rendere sicura la password, è preferibile usare caratteri alfanumerici, usare sostituzioni oppure caratteri non alfanumerici.

- **SCHEMA DI FIRME RSA**.

RSA: Lo schema RSA è una cifratura a blocchi in cui il testo e il testo cifrato sono interi compresi fra 0 e n-1 per un dato valore di n. normalmente n è pari a 1024 bit ovvero 309 cifre decimali. Il testo in chiaro viene crittografato a blocchi, dove ciascun blocco è un valore binario minore di un certo numero n. La crittografia e la decrittografia di un determinato blocco di testo in chiaro M e del corrispondente blocco di testo cifrato C hanno la seguente forma:

Cifratura: $C = P_B(M) = M^e \bmod n$; **Decifratura**: $M = S_B(C) = C^d \bmod n$

- **Generazione della chiave**: A genera due numeri primi grandi p e q; A calcola $n = p \times q$ e $\Phi(n) = (p - 1)(q - 1)$; A sceglie un numero $1 < e < \Phi(n)$ tale che $\gcd(e, \Phi(n)) = 1$; A calcola $d = e^{-1} \bmod \Phi(n)$ usando l'algoritmo di Euclide Esteso; A pubblica n ed e come sua chiave pubblica $P_A = (e, n)$.
  A conserva n e d come sua chiave privata $S_A = (d, n)$.
- **Generazione di una firma**: $F = sig_{(n,d)}(M) = M^d \bmod n$; **Procedura di verifica**: $ver_{(n,e)}(M,F)$ = Vero se $M = F^e \bmod n$; Falso altrimenti.

Correttezza: verifichiamo ora che la fase di cifratura e quella di decifratura sono una l'inversa dell'altra. Poiché $ed \equiv 1 \bmod \Phi(n)$ si ha: $ed = t \times \Phi(n) + 1$ con t intero e $t >= 1$. Supposto che $M \in Z_n^*$, si ha:
$C^d = (M^e)^d = M^{ed} = M^{t \times \Phi(n)+1} \bmod n = M$; in quanto per il teorema di Eulero si ha che: $x^{\Phi(n)} = 1 \bmod n$.

Sicurezza dell'RSA: La sicurezza dell'RSA si basa sul fatto che la funzione di cifratura $x^e \bmod n$ è una funzione "one-way" che è computazionalmente difficile da invertire per un nemico che volesse decifrare un messaggio. Solo conoscendo la fattorizzazione di n è possibile trovare il valore delle chiavi. Infatti conoscendo la fattorizzazione di n è possibile calcolare $\Phi(n)=(p-1)(q-1)$ e
calcolare $d=e^1 \bmod \Phi(n)$ usando l'algoritmo esteso di Euclide. Da ciò si deduce che la sicurezza dell'RSA dipende dal problema di fattorizzare grandi numeri.

ES0-ESEMPI CON CIFRARI

- **Cifrario a sostituzione:** non resiste a Known Ciphertext Attack.
  Assumiamo che il testo in chiaro sia in lingua inglese, senza "spazi" e punteggiatura.
  Esempio di testo cifrato: UZQSOVUOHXMOPVGPOZPEVSG analizzo le frequenze delle lettere sapendo che il testo cifrato è in inglese, e man mano sostituisco e ottengo: IT WAS DISCLOSED YESTERDAY
- **Cifrario di Vigenere:** non resiste a Known Ciphertext Attack
  Testo in chiaro: CODICE MOLTO SICURO Chiave: REBUS
  CODIC EMOLT OSICU RO testo in chiaro
  REBUS REBUS REBUS RE chiave
  TSECU VQPFL FWJWM IS testo cifrato

Considerato inviolabile per molto tempo ha un numero possibili chiavi = 26^t

- **Cifrario di Hill:** non resiste a Known Plaintext Attack
  Sia PQCFKU la cifratura Hill di FRIDAY per m=2
  FR=(5,17) -> PQ=(15,16)
  ID=(8,3) -> CF=(2,5) Y = K*Xmod26
  Si ha: = K * mod 26 X^(-1) = K = Y* =

ES1-RSA

Sia (n=35, e=11) la chiave pubblica di Alice. 1) Calcolare la chiave privata di Alice mediante l'algoritmo esteso di
Euclide. 2) Calcolare la firma di Alice sul messaggio m=8, mediante l'algoritmo left-to-right o right-to-left.
Risoluzione:

- Key pubblica = (11, 35) e Key privata = (d, 35). 35 = 7*5 -> p=7 q=5

  d = e^(-1)%Φ(n) -> Φ(n) = Φ(35) = Φ(p)Φ(q) = (p-1)(q-1) = 6*4 = 24

  quindi: d = 11^(-1)%24 = 11 e ottengo che la **key privata** è d = (11,35)

- LEFT-to-RIGHT m=8 -> c = m^(d)%n = 8^(11)%35 = 22 e ottengo x=8, d = 11, z=35

  Converto d in binario d = 11 = 1011 e eseguo l'algoritmo:

a=1
for 3 to 0:
a = (1*1)%35
if(y3 = = 1) //yes
then a = (1*8)%35
return a = 8;
	a=8
	for 3 to 0:
	a = (8*8)%35
	if(y2 = = 1) //no
	
	return a = 29;
		a=29
		for 3 to 0:
		a = (29*29)%35 = 1
		if(y1 = = 1) //yes
		then a = (1*8)%35
		return a = 8;
			a=8
			for 3 to 0:
			a = (8*8)%35
			if(y0 = = 1) //yes
			then a = (29*8)%35
			return a = 22;

ES2-RSA

Sia (n=35, e=5) la chiave pubblica di Alice. 1) Calcolare la chiave privata di Alice, illustrando le computazioni.2) Calcolare la firma di Alice sul messaggio m=10, illustrando le computazioni.

- Key pubblica = (5, 35) e Sa = (d, 35). 35 = 7*5 -> p=7 q=5

  d = e^(-1)%Φ(n) -> Φ(n) = Φ(35) = Φ(p)Φ(q) = (p-1)(q-1) = 6*4 = 24

  quindi: d = 5^(-1)%24 = 5 e ottengo che la **key privata** è d = (5,35)

- LEFT-to-RIGHT m=10 -> c = m^(d)%n = 10^(5)%35 = 5 e ottengo x=10, d = 5, z=35

  Converto d in binario d = 5 = 101 e eseguo l'algoritmo:

a=1
for 2 to 0:
a = (1*1)%35
if(y2 = = 1) //yes
then a = (1*10)%35
return a = 10;
	a=10
	for 2 to 0:
	a = (10*10)%35
	if(y1 = = 1) //no
	
	return a = 30;
		a=30
		for 2 to 0:
		a = (30*30)%35=25
		if(y0 = = 1) //yes
		then a = (25*10)%35
		return a = 5;
			Chiave di Alice = 5
			
			Firma = 101 = y

ES3-Cifratura Simmetrica

E(a,b)(x)=ax+b mod 26 con chiave k = (a,b), dove b∈ Z26, e x ∈ Z26.

- Si determini sotto quali condizioni sul parametro a della chiave, la funzione di cifratura E risulta univocamente decifrabile.
  Risoluzione: se y = ax+b%26 -> a^(-1)(y-b)=x -> x=a^(-1)(y-b)%26
  La funzione risulta univocamente decifrabile per i valori di a e sono invertibili in MOD 26.
- In conseguenza di quanto stabilito al punto i), si determini la dimensione dello spazio delle chiavi
  Risoluzione: 26 = 13*2 -> p=13 q=2 $\Phi(16) = (p-1)(q-1) = 12$.
  Abbiamo a=12 valori, b non ha nessun vincolo ed è 26 valori, **chiavi** = 26*12
- Sia k=(a,b)=(7,3). Si determini la cifratura del messaggio CIAO

  Risoluzione: a = 7, b = 3 Cifratura: C = 2, I = 8, A = 0, O = 14 Testo cifrato = RHDX

  Operazione: chiaro*cifrato%26 -> sostituisco in ax+b e calcolo

OPENSSL

CIFRARI SIMMETRICI

enc -in “file_input” -out “file_output” -e -pass pass: “password” -cipher_name

RSA

<u>Generare chiavi</u>

Genrsa -out “chiavi” -passout pass: “password” -cipher_name 1024/2048…

<u>Estrarre chiave pubblica</u>

Rsa -in “chiavi” -passout pass: “password” -pubout -out “chiave_pub”

<u>Cifrare con RSA</u>

Rsautl -encrypt -pubin -inkey “chiave_pub” -in “file_in” -out “file_out”

<u>Decifrare con RSA</u>

Rsautl -decrypt -inkey “private_key” -in “file_input” -out “file_output”

DH

<u>Generare parametri pub</u>

dhparam -out “param_pub” 1024/2048

<u>Generare coppia di chiavi</u>

Genkey -paramfile “param_pub” -out “couple_key”

<u>Estrarre chiave pubblica</u>

Pkey -in “couple_key” -pubout -out “pub_key”

Estrarre chiave condivisa

Pkeyutl -derive -inkey “couple_key” -peerkey “chiave_pub_altrro_utente” -out –“sharedKey”

FIRME DIGITALI

Rsautl -sign -inkey “kprv” -in “file” -out “file”

Rsautl -verify -pubin -inkey “kpub” -in …. -out ….

Sha1 -sign “kprv” -out “file_out” “file_in”

Sha1 verify “kpub” -signature “fileout” filein

Generare parametri DSA

Dsaparam -out “file_par” 1024/2048

Generare coppia di chiavi

gendsa -out “file_prv” -des3 “file_par”

estrarre kpub

dsa -in “file_pvr” -pubout -out “pub_key”

firma dsa

dgst -dss1 -sign –“file_pvr” -out “signed_file” “inFile”

verfica dsa

dgst -dss1 -verify “pub_key” -signature “signed_file” “inFile”

FUNZIONI HASH

Calcolo hash

Dgst -sha512 -out “file_has” file.txt

Hmac

Dgst -sha512 -out “hmacoutput” -hmac “stringa” file.txt