-Protocolli crittografici: Definisce l'interazione tra le parti (entità coinvolte nello schema) per ottenere le proprietà di sicurezza desiderate.
-Primitive di crittografia: *Tecniche di cifratura*: -cifrari simmetrici o a chiave privata (cifrari a blocchi strema cipher); -cifrari asimmetrici o a chiave pubblica;
-Tecniche per l'autenticazione d'integrità: Funzioni HASH, MAC, e firme digitali. Scambio e condivisione di chiavi e generazione di numeri pseudo-casuali.
-Principio di Kerckhoffs: La sicurezza di un cifrario deve dipendere solo dalla segretezza della chiave, e non dalla segretezza dell'algoritmo usato.
-Cifrari asimmetrici: Usano una cassaforte con due lucchetti; con una chiave pubblica chiudiamo la cassaforte, con l'altra privata apriamo la cassaforte. Chiunque può cifrare un messaggio per Alice, ma solo Alice può cifrare un messaggio cifrato per lei. Non ci sono chiavi condivise tra Alice e Bob, ciascuno dei 2 utenti genera solo la propria coppia di chiavi e rende pubblica la chiave pubblica. Ogni utente memorizza una sola chiave,quella privata.
-Firma digitale: Equivalente alla firma convenzionale, deve poter essere facilmente prodotta dal legittimo firmatario e nessun utente deve poter riprodurre le firme si altri.Chiunque può facilmente verificare una firma.
-Algoritmi per firme digitali: RSA e DSS, HASH.
-Funzioni HASH: Il valore HASH h(M) è una rappresentazione non ambigua e non falsificabile del messaggio M. *Tipico uso:* Computo al tempo t il valore HASH del file M; conservo H=h(M) in un luogo sicuro. Per controllare se il file è stato successivamente modificato calcolo h(M') e verifico se H=h(M'); assicura se una file è stato modificato.
-MAC(Message Authentication Code):Sì da in input un segreto e un messaggio e il MAC restituisce b bit. Garantisce integrità dei dati e autenticità dei dati.
-Proprietà di sicurezza:*Confidenzialità*(informazioni sono accessibili in lettura solo da chi è autorizzato), *Autenticazione*, *Integrità*(solo chi è autorizzato può modificare l'attività di un sistema o le informazioni trasmesse), *Non ripudio*(chi invia e chi riceve non può negare la trasmissione del messaggio), *Anonimia*(protezione dell'identità o del servizio utilizzato. Internet non garantisce anonimato; dall'indirizzo IP si identifica l'utente)
-Tecniche di steganografia:Disposizione delle lettere/parole in un messaggio innocuo; contrassegna dei caratteri(ripasso a matita,fori sulle lettere..); inchiostro invisibile. *Tecniche moderne*: informazione nascoste nei bit (watermarking). *Vantaggi*: le parti possono nascondere si essere in comunicazione. *Svantaggi*: il messaggio segreto può essere scoperto. La segretezza è perduta quando il sistema viene scoperto.
-Cifrari simmetrici: Crittosistemi a chiave privata/segreta. Alice e Bob conoscono la stessa chiave k. Cifrari a blocchi agiscono su ogni blocco input.Stream cipher: messaggi cifrati continuamente e flusso di chiavi. ***Sostituzione***: ciascun elemento viene mappato su un altro elemento. ***Trasposizione***: elementi del testo in chiaro vengono scambiati di posto.
-Cifrari con shift: *Sicurezza*: dato un testo cifrato Y, deve essere difficile risalire alla chiave usata K. Una volta individuata la chiave, conoscendo D_k sarà possibile decrittografare tutti i messaggi. *Attacco a forza bruta*: si tenta ogni possibile chiave fino ad ottenere un risultato. In media bisogna provare la metà delle chiavi.Sono detti monoalfabetici: una volta scelta la chiave ogni carattere è mappato ad un unico carattere cifrato.
-Cifrari monoalfabetici: Al posto di una lettera dell'alfabeto in chiaro si mette quella corrispondente dell'alfabeto cifrante. Le chiavi possibili ,con un alfabeto di 21 lettere sono 21! (circa 50 miliardi).
-Cifrari a sostituzione: Si usa una frase chiave che sostituisce le prime lettere dell'alfabeto in chiaro senza eventuali ripetizioni. La sostituzione prosegue con le lettere conseguenti all'ultima lettera della chiave, senza ripetere quelle già presenti. Chiavi possibili: più di 26, ma meno di 26!. **-** *Crittoanalisi*: Tramite analisi crittografica basata sulla natura dell'algoritmo, sfrutta conoscenze sul testo in chiaro e sull'algoritmo (es. Frequenza occorrenza lettere). Invece con attacco a forza bruta si tenta ogni possibile chiave fino ad ottenere un risultato; in media bisogna provare la metà delle

chiavi. –***Numero chiavi possibili****:* Sostituzione generica: 26!. Sostituzione con alfabeto shiftato: 25. Sostituzione con parole chiave lunga n caratteri (tutti diversi): n!.
-Sicurezza dei cifrari: -***Incondizionatamente sicuro*** *(Unconditionally secure)*: indipendentemente dalle risorse disponibili, è impossibile decrittare il testo cifrato(esiste solo un sistema, il *One-Time pad* –*vedere dopo*-). -***Computazionalmente sicuro*** *(Computationally secure)*: il tempo richiesto per violare la cifratura è grande e superiore alla vita utile delle informazioni contenute.
-Omofoni: Molti simboli per cifrare singoli caratteri frequenti cosi da abbassare le frequenze del testo cifrato.
-Disco di Alberti: Propose di usare più alfabeti cifranti e di sostituirli durante la cifratura.
-Cifrario di Porta: Cifrario per digrammi. Le lettere dell'alfabeto sono sistemate sia sulla riga che sulla colonna di intestazione della tabella. Ogni coppia di lettere corrisponde ad un numero che parte da 0 per aa e arriva a 675 per zz.
-Cifrario di Playfair: Le lettere in chiaro sono sostituite dalla lettera sulla stessa riga e colonna (diagonale inversa), le lettere ripetute vanno separate da una lettera di riempimento (ball → balxl); le lettere sulla stessa riga vengono sostituite dalle lettere a destra (es: ar → rm); le lettere sulla stessa colonna vengono sostituite dalle lettere sottostanti (es: mu → cm). *Sicurezza di Playfair*: Migliore rispetto alla cifratura monoalfabetica, perché per esempio, mentre in quest'ultima vi sono solo 26lettere, nella playfair vi sono 26*26=676 digrammi, e dunque l'identificazone dei singoli digrammi è più difficoltosa. *Debolezze:* L'analisi è condotta in base alla frequenza dei digrammi più comuni nella lingua.
-Cifrario di Hill: m lettere in chiaro sono sostituite con m lettere cifrate secondo m equazioni lineari. Data una chiave K sottoforma di matrice, si cifra il testo in chiaro moltiplicando a 2 a 2 i caratteri convertiti in numero del testo in chiaro * la matrice. Si ottengono 2 numeri per ogni operazione di questo tipo e si fa il modulo per trovare le lettere corrispondenti cifrate. Per la decifratura si applica lo stesso procedimento utilizzando il testo cifrato e la matrice inversa.
-Cifrario di Vigenère: Cifrario a sostituzione polialfabetica. Si utilizza il quadrato di Vigenère, composto nella prima riga e nella prima colonna da tutte le lettere dell'alfabeto, proseguendo completando tutte le righe e le colonne con le lettere dell'alfabeto mancanti. Dato un testo in chiaro ed una chiave, si divide il testo in chiaro per la lunghezza della chiave, e per avere il testo crittato si utilizza il quadrato trovando le corrispondenze tra la lettera del testo in chiaro e la lettera della chiave. Quindi, per la crittazione si utilizza la seguente formula: $C=(Mi+k_{mod\ t})$mod t. Per la decrittazione invece, si fa l'operazione inversa, ovvero si sottrae ad ogni lettera del messaggio cifrato, il valore della corrispondente lettera della chiave: $Mi=(Ci-K_{mod\ t})$mod t. Numero possibile di chiavi = 26^t (dove t è il numero delle lettere dell'alfabeto). Crittoanalisi: E' possibile romperlo usando indice di coincidenza e indice mutuo di coincidenza. Si determinano le ripetizioni e da esse si stima la lunghezza della chiave. Si analizzano le frequenze delle lettere in ognuno degli alfabeti cifranti corrispondenti alle lettere della chiave. Resiste all'analisi delle frequenze. Una lettera cifrata corrisponde a più simboli in chiaro ed esiste un numero grande di chiavi.
-Indice di coincidenza:probabilità che due caratteri presi a caso in x1, x2,…xn, siano uguali.Il suo utilizzo nella crittoanalisi serve per determinare la lunghezza della chiave.
-Indice mutuo di coincidenza: probabilità che due caratteri presi a caso in x1, x2,…xn, e in y1,y2…yn siano uguali. Il suo utilizzo nella crittoanalisi serve per determinare Il valore della chiave.
-Operazioni di trasformazione: *Sostituzione*: ciascun elemento del testo viene mappato su un altro elemento. *Trasposizione*: Elementi del testo in chiaro vengono scambiati di posto. *Sicurezza della trasposizione*: per brevi messaggi poca (es: 3 lettere → 6 anagrammi); su una frase diventa grande.
-Cilindro di Thomas Jefferson: Cilindro di 15 cm e 36 dischi di legno; numero possibili ordinamenti dei dischi = 36!
-Enigma: In astratto enigma consisteva di 3 pezzi: tastiera, scambiatore, visore. L'alfabeto cifrante cambia dopo ogni lettera. Lo scambiatore ha 26 alfabeti cifranti.Con 2 rotori la cifratura torna al punto iniziale dopo 26 * 26 cifrature, cioè ho 676 alfabeti cifranti. Le chiavi di enigma aumentano

se calcoliamo che i rotori possono essere inseriti in posizioni reciproche diverse.Vi è una semplice sostituzione monoalfabetica ma con un grande numero di chiavi. Le sostituzioni non cambiano durante la cifratura; si rompe con l'analisi delle frequenze. I rotori hanno un piccolo numero di chiavi, ma l'assetto cambia continuamente e resiste all'analisi delle frequenze.
-Cifrario One-Time pad: Lunghezza chiave = lunghezza testo in chiaro e chiave composta da bit indipendenti e casuali. Per ottenere il testo cifrato si fa lo XOR bit a bit fra testo in chiaro e chiave. Impossibile decifrare il messaggio. E' *unconditionally secure*, cioè indipendentemente dal tempo e dalle risorse a disposizione è impossibile decrittografare il testo cifrato. Necessità però di un repertorio di chiavi.
-Concetto di probabilità: Il termine probabilità viene usato come misura del grado di plausibilità di una affermazione, ovvero del "verificarsi di un certo evento". Per evento si intende qualsiasi affermazione o proposizione della quale sia verificabile il contenuto di verità.
-Probabilità soggettiva: La probabilità di un evento A è la misura del grado di fiducia che un individuo coerente attribuisce, secondo le sue informazioni e opinioni, all'avverarsi di A.Per passare ad un concetto pratico di probabilità bisogna quantificare in un numero il livello di probabilità e stabilire una serie di regole che questi numeri devono soddisfare.
-Probabilità classica: La definizione classica di probabilità è: La probabilità, P(A), di un evento A è il rapporto tra il numero N di casi "favorevoli" e il numero totale M di casi possibili e mutuamente incompatibili(che non possono verificarsi simultaneamente). $P(A) = N / M$.
-Probabilità matematica: Sia S un insieme di possibili risultati (A_i) di un esperimento, e tali eventi sono mutuamente escludentesi, allora per ognuno di essi esiste una probabilità P(A) rappresentata da un numero reale soddisfacente gli assiomi di probabilità. Assiomi: *1.* $P(A_i) \geq 0$. *2.* con A_1 e A_2 mutuamente escludentesi deve valere: P(A_1 oppure A_2) = P (A_1) + P(A_2) dove P(A_1 oppure A_2) è la probabilità di avere il risultato A_1 o il risultato A_2. *3.* la sommatoria delle probabilità di tutti gli eventi mutuamente escludentesi deve risultare uguale a 1. La probabilità di non ottenere l'evento è uguale ad 1 meno la probabilità di ottenerla. La probabilità è un numero reale appartenente all'intervallo [0.1].
-Somma di probabilità:Se due eventi sono mutuamente escludentesi: $P(A \cup B) = P(A) + P(B)$; se l'accadere di uno non preclude la possibilità che si presenti anche il secondo, $P(A \cup B) = P(A) + P(B) – P(A \cap B)$.
-Prodotto di probabilità: Se 2 eventi possono verificarsi simultaneamente: $P(A \cap B) = P(A) + P(B) - P(A \cup B)$.
-Probabilità condizionata: La valutazione di probabilità dipende dallo stato di informazione. Invece di probabilità assoluta si parla di probabilità condizionata ad una certa informazione e si indica con P (A | B), letta probabilità di A dato B.
-Probabilità composte: La probabilità del prodotto di 2 eventi è uguale al prodotto della probabilità di uno degli eventi per la probabilità condizionata dell'altro calcolata a condizione che il primo abbia luogo: $P(A \cap B) = P(A)\ P(B \mid A) = P(B)\ P(A \mid B)$. Se gli eventi A e B sono mutuamente escludentesi la probabilità condizionata si annulla per definizione. Se gli eventi sono indipendenti si ha che P(B | A) = P(B) e P(A | B) = P(A); quindi la probabilità del prodotto di 2 eventi indipendenti è uguale al prodotto delle probabilità di questi eventi.
-Known Ciphertext Attack: L'avversario conosce solo il testo cifrato.
-Known Plaintext Attack: L'avversario conosce sia il testo in chiaro che il testo cifrato.
-Chosen Plaintext Attack: L'avversario può ottenere la cifratura di un testo in chiaro a sua scelta.
-Chosen Ciphertext Attack: L'avversario può ottenere la decifratura di un testo cifrato a sua scelta.
-Chosen Text Attack: L'avversario può ottenere la cifratura e la decifratura di coppie di testi chiaro/cifrato.
-Cifrari a blocchi (DES): Cifrari a chiave simmetrica, il testo in chiaro è diviso in blocchi di lunghezza fissa e viene cifrato un blocco alla volta; opera su blocchi di n bit di input per produrre blocchi di n bit di output; con blocchi n bit di testo in chiaro ho 2^n possibili input. In generale per n

bit la dimensione della chiave è $n*2^n$. *Trasformazione reversibile o non singolare*: ogni blocco di testo in chiaro deve produrre un blocco cifrato univoco. *Sicurezza DES:* Questo cifrario, per blocchi di piccola dimensione è equivalente alla cifratura a sostituzione ed è quindi vulnerabile all'analisi statistica. Per blocchi grandi si mascherano le caratteristiche statistiche ma vi sono problemi pratici per la dimensione della chiave.
-Cifratura di flussi (DES): Esegue la crittografia di un flusso digitale di dati un bit o un byte alla volte, esiste un flusso di chiavi (esempi classici Vigenère, Vernam). Utilizzando alcune modalità di funzionamento si può utilizzare la cifratura a blocchi per cifrare flussi.
-Cifratura di Feistel (DES): Molti dei cifrari a blocchi in uso si basano sulla proposta di Feistel seguendo l'idea dell'uso di cifrature in sequenza, per ottenere cifrature più complesse di ogni singola componente. Feistel alterna permutazioni e sostituzioni applicando i principi di Shannon per contrastare l'analisi statistica. Ogni cifra del testo cifrato è prodotta da più cifre del testo in chiaro. Per blocchi binari si utilizza la trasposizione; blocchi grandi migliorano la sicurezza ma riducono la velocità. Stesso principio vale per le chiavi grandi. A partire dalla chiave iniziale vengono prodotte tante sottochiavi quanti sono i round. La funzione round più è complessa, più resiste alle crittoanalisi.
-Struttura di Feistel: Il testo in chiaro viene diviso in un blocco sinistro e in un blocco destro. La parte destra diventa la parte sinistra del livello successivo. Per la parte sinistra,invece, viene eseguito uno XOR con una funzione generata con la sottochiave.
-Cifrari di Feistel: Per la cifratura basta implementare un solo round e lo stesso codice può essere usato per ogni round. La decifratura usa lo stesso algoritmo della cifratura, ma considerando le sottochiavi in ordine inverso.
-DES (Data Description Standard): Nello standard DES la chiave è lunga 64 bit, 8 byte di cui l'ottavo bit è di parità. Il bit di parità è lo XOR dei precedenti 7 bit.
-Struttura del DES: *Cifratura DES:* la funzione DES prevede due input, il testo in chiaro da crittografare e la chiave. L'elaborazione del testo in chiaro viene eseguita in tre fasi. Prima il testo in chiaro, di 64 bit , attraversa una permutazione iniziale che dispone i bit per produrre un input "permutato"; dopo, c'è una fase costituita da 16 ripetizioni della stessa funzione di permutazione e sostituzione. L'output dell'ultima fase è costituito da 64 bit che dipendono dal testo in chiaro e dalla chiave.La chiave attraversa invece prima una funzione di permutazione, poi per ognuna delle 16 ripetizioni, viene prodotta una sottochiave;la funzione di permutazione e la stessa per ciascuna ripetizione ma viene prodotta ogni volta una sottochiave differente. I risultati prodotti dall'elaborazione del testo in chiaro e dall'elaborazione della chiave vengono scambiate per produrre un "preoutput" che poi attraversa una permutazione inversa della permutazione iniziale.Ad eccezione delle permutazioni iniziale e finale, DES ha esattamente la stessa natura della cifratura di Feistel. *Decifratura DES:* come nel caso della cifratura Feistel , la decifratura utilizza lo stesso algoritmo della cifratura applicando le sottochiavi in ordine inverso. *Esempio:* Nella fase di cifratura partendo da 2 elementi L15 e R15 ,una chiave K16 ed una funzione f, abbiamo dopo la cifratura : R15=L16 ed L15=R16⊕ f(L16,K16).Nella fase di decifratura scambiando gli elementi e quindi partendo da L16 e R16 otteniamo le stesse modifiche applicate in fase di cifratura.
-Proprietà del complemento: Se l'algoritmo DES riceve in input un messaggio x e lo cifra con una chiave k da in output un messaggio y. Se riceve in input il messaggio x negato (cioè il complemento bit per bit di x) e lo cifra con la chiave k anch'essa in completo, restituisce in output il complemento bit a bit di y.
-Effetto valanga(DES):due testi che differiscono di un solo bit e sono cifrati con la stessa chiave differiscono per 34 bit.Due testi uguali cifrati con chiavi diverse per un solo bit,differiscono per 35 bit.
-Svantaggi DES:C'è lo svantaggio che con 8 iterazioni DES realizza una funzione random.DES ha una chiave di soli 56 bit, blocchi di 64 bit, criteri costruttivi non chiari ed è lento nell'implementazione software.

-Attacchi al DES: Attacco a forza bruta, Time-space trade-off .Analisi crittografica: *Crittoanalisi lineare:* recupera la chiave a partire da 2^{43} coppie di testi in chiaro noti; *Crittoanalisi differenziale:* recupera la chiave a partire da 2^{47} coppie di testi scelti.
-Modalità operative del DES: Servono a cifrare testi più lunghi di 64 bit. Cinque modalità : -*electronic codebook chaining (**ECB**)*: ciascun blocco in chiaro di 64 bit viene codificato in modo indipendente con la stessa chiave.Una sua applicazione è nella trasmissione sicura di singoli valori.Se la lunghezza del messaggio non è un multiplo di 64, se è più lungo spezzo l'input in blocchi da 64 bit, se è più corto effettuo il padding.VANTAGGI : l'ECB è il metodo più semplice e veloce ed eventuali errori non si propagano.SVANTAGGI: ha lo stesso blocco in chiarocorrisponde lo stesso blocco cifrato, quindi per messaggi lunghi non è sicuro e ci possono essere possibili attacchi di sostituzione.-*Cipher block chaining(**CBC**)* : in ogni blocco l'input si ottiene come XOR dei 64 bit di testo in chiaro e dei precedenti 64 bit di testo cifrato.Una sua applicazione è nella trasmissione di caratteri a blocchi.VANTAGGI: non ci sono ripetizioni.SVANTAGGI: è meno veloce dell'ECB, c'è la propagazione di errori e c'è dipendenza tra i blocchi.-*Cipher feedback(**CFB**)*: può operare in tempo reale,infatti cifro e trasmetto ciascun carattere;la lunghezza del testo cifrato è uguale a quella del testo in chiaro.Come nel CBC il testo cifrato è funzione del testo in chiaro precedente.Nella decrittografia si usa DES. VANTAGGI: può operare in tempo reale.SVANTAGGI: gli errori vengono propagati.-*Output feedback (**OFB**)*: la struttura è simile al CFB ma al registro viene mandato l'output di DES invece che il testo cifrato.Una possibile applicazione è tramite i flussi su canali rumorosi.VANTAGGI: non si propagano gli errori di trasmissione dei bit. SVANTAGGI:per quanto riguarda la modifica dl flusso è più vulnerabile di CFB.- *Counter(**CTR**)*: si impiega un contatore delle dimensioni del blocco in chiaro.Per ogni blocco successivo il contatore viene incrementato.VANTAGGI: Efficienza hardware e software, precalcolo dell'output DES, accesso casuale, sicurezza dimostrabile ed è semplice in quanto richiede solo l'algoritmo di crittografia.
-Rijndael (AES): non è un cifrario di Feistel, lavora in parallelo sull'intero blocco in input.E' un cifrario a blocchi iterato.La dimensione del blocco è di 128 bit, la lunghezza della chiave può essere 128, 192 o 256 bit.Ogni round è una composizione uniforme e parallela di 4 passi.
-Parametri AES*:* nel AES-128 la lunghezza della chiave è di 4 word(128 bit),la dimensione del blocco è di 4 word e il numero di rounds è 10;con 128 bit ci sono 2^{128} chiavi possibili.Nel AES-192 la lunghezza della chiave è di 6 word(192 bit),la dimensione del blocco è di 4 word e il numero di rounds è 12; con 192 bit ci sono 2^{192} chiavi possibili. Nel AES-256 la lunghezza della chiave è di 8 word(256 bit),la dimensione del blocco è di 4 word e il numero di rounds è 14; con 256 bit ci sono 2^{256} chiavi possibili.
-AES *cifratura:*le fasi di cifratura utilizzate sono 4.-*Substitute bytes:* usa una s-box per svolgere una sostituzione del blocco byte per byte. -*Shift Rows:* compie una semplice permutazione.-*Mix columns:* una sostituzione che utilizza l'aritmetica.-*Add round key:* una semplice operazione di XOR bit a bit del bloco corrente con una porzione della chiave espansa.
-AES decifratura:L'algoritmo di decifratura non è lo stesso della cifratura.Infatti usa una diversa sequenza di trasformazioni, le trasformazioni inverse. Lo svantaggio è che necessita di una doppia implementazione.Esiste anche un algoritmo di decifratura che ha la stesa struttura di quello di cifratura, infatti ha la stessa sequenza di trasformazioni, usa le trasformazioni inverse ma richiede una diversa schedulazione delle chiavi.
-DES:Cifratura multipla: Si può costruire un cifrario più sicuro a partire dal DES senza modificarne la struttura.Questo lo si fa tramite la cifratura multipla che consiste nel cifrare il messaggio varie volte con chiavi differenti, sperando che questo aumenti la sicurezza, *DES DOPPIO , DES TRIPLICATO(3-DES).*
-DES Doppio (Funzionamento e Sicurezza): Il DES doppio prevede due fasi di crittografia con 2 chiavi, utilizzate in sequenza. La fase di decifratura richiede l'applicazione delle chiavi in ordine inverso. Questo schema, apparentemente, prevede una chiave della lunghezza doppia di DES(56*2 =112 bit) con un notevole incremento della potenza crittografica, ma questo solo apparentemente, in

quanto il risultato di una doppia cifratura, potrebbe equivalere alla cifratura dello stesso messaggio con una sola chiave.

-Attacco meet in the middle (DES Doppio): Data una coppia nota(P e C) l'attacco procede nel seguente modo: innanzitutto si esegue la crittografia di P per tutti i 2^{56} valori possibili di k_1.Si memorizzano i risultati in una tabella e si ordina la tabella.Poi si esegue la decritografia di C utilizzando tutti i 2^{112} possibili valori di k_2 .Man mano che vengono prodotte le decrittografie si confrontano i risultati con la tabella e si ricerca una corrispondenza.Quando viene trovata una corrispondenza si esegue il test delle due chiavi risultanti contro una nuova coppia nota di testo in chiaro/testo cifrato.Se le due chiavi producono il testo cifrato corretto si tratta delle chiavi corrette.Per un determinato testo in chiaro P esistono 2^{64} possibili valori cifrati che potrebbero essere prodotti da una doppia applicazione dell'algoritmo 2DES.

-3-DES: Esiste il 3-DES con tre chiavi ed il 3-DES con due chiavi.Il 3-DES con tre chiavi *(DES triplicato)* prevede tre fasi di crittografia con 3 chiavi diverse.Usa in pratica una chiave di 168 bit (56+56+56).Questa tecnica è una contromisura ovvia contro l'attacco meet in the middle.Lo svantaggio di questo approccio è che si ha una chiave troppo lunga. Un'alternativa è una tripla crittografia con 2 chiavi *(DES triplo)*.Vi sono sempre 3 fasi di cifratura, ma nella terza fase si riutilizza la prima chiave.La *compatibilità* di 3-DES con 2 chiavi con il DES semplice è garantita ponendo k=k', in questo modo diventa equivalente al semplice DES. La *compatibilità* del 3-DES con tre chiavi con DES è garantita ponendo k'''=k'' oppure k'=k''.

-Attacchi a 3-DES: Gli attacchi portati a 3-DES non sono una minaccia realistica poiché nessuno di questi è riuscito a romperlo.Un attacco prevede la ricerca di valori di testo in chiaro che producono un primo valore intermedio e poi nell'utilizzo di *meet in the middle* per determinare le due chiavi.Un altro attacco è l'attacco a testo in chiaro noto che si basa sull'osservazione che si conoscano un risultato intermedio del testo cifrato ed il testo cifrato finale, riducendo quindi il problema ad un attacco a 2-DES.Naturalmente chi svolge l'attacco non può avere tutte queste informazioni, quindi l'attacco risulta di difficile implementazione.

-Blowfish: E' un cifrario simmetrico a blocchi ed ha le caratteristiche del cifrario di Feistel, ma in più è veloce, compatto e semplice da implementare e da analizzare.Espande la chiave convertendo una chiave di al più 14 word in un array di 18 sotto-chiavi a 32 bit. *Cifratura:* Utilizza due operazioni, + per la somma di word e $\oplus$ per lo XOR. *Decifratura:* Utilizza lo stesso algoritmo ma con le sottochiavi in ordine inverso.

-Blowfish vs DES: Con Blowfish sia le sottochiavi che le S-BOX dipendono dalla chiave mentre in DES le S-BOX sono fissate. Inoltre, con Blowfish in ogni round le operazioni coinvolgono tutto il blocco, in DES solo la parte destra. Blowfish è invulnerabile ad attacchi di forza bruta(con dim.chiave 14 word).

-RC5:è un altro cifrario simmetrico. *Parametri:* la dimensione dei blocchi è variabile(32,64 o 128 bit), il numero di round è variabile (da 0 a 255) e la dimensione della chiave è variabile(da 0 a 255 byte).Altre caratteristiche sono che usa operazioni comuni dei processori, che usa poca memoria, che è semplice da implementare e da analizzare. *Cifratura:* utilizza tre operazioni primitive: la soma delle word, lo XOR bit a bit e la rotazione circolare a sinistra.*Decifratura:* la decifratura deriva fondamentalmente dall'algoritmo di cifratura, solo che in questo caso allle due word di testo cifrato vengono inizialmente assegnate le due variabili di una word A e B.

-Stream Cipher:è un cifrario simmetrico moderno a blocchi,molto più veloce dei normali cifrari a blocchi, infatti ha poche linee codice e operazioni semplici .Cifra il messaggio un byte (o bit) alla volta; utilizza una sequenza (keystream) pseudo-casuale generata a partire dalla chiave; combina la keystream tramite XOR con il messaggio.Per complicare la crittoanalisi si può usare un keystream con un lungo periodo e un keystream con le stesse caratteristiche di una sequenza casuale.

-RC4:cifrario simmetrico,la dimensione della chiave è variabile (da 1 a 56 byte). Genera una keystream con periodo maggiore di 10^{100}, usa operazioni orientate ai byte ed è semplice da implementare ed analizzare.Per la crittografia si esegue l'operazione di XOR del valore k con il byte successivo di testo in chiaro.Per la decrittografia si esegue l'operazione di XOR del valore k

con il byte successivo di testo cifrato. *Attacchi:* con chiave 128 bit non ci sono attacchi noti, esiste un attacco al protocollo WEP quindi il problema non dipende da RC4.I problemi del WEP sono che la chiave non viene cambiata e lo spazio delle chiavi è piccolo.

-Cifrari simmetrici: I due utenti per condividere una chiave comune possono ho utilizzare un canale privato ad esempio, un corriere fidato o un incontro faccia a faccia in un luogo segreto, oppure possono utilizzare una terza parte privata che stabilisce la chiave di sessione e la invia ad entrambi in modo sicuro. Per quanto riguarda la gestione delle chiavi nei cifrari simmetrici, in una rete con n utenti ogni coppia di utente deve condividere una chiave, quindi ogni utente deve memorizzare n-1 chiavi arrivando così a un numero totale delle chiavi segrete dell'ordine di (n^2)/2; inoltre l'aggiunta di un nuovo utente alla rete implica la distribuzione della chiave a tutti i precedenti utenti.

-Cifrari asimmetrici: Sono caratterizzati dall'utilizzo di due chiavi una pubblica e una privata che hanno il compito una di cifrare o verificare la firma e l'altra di decifrare o creare la firma. Per quanto riguarda la cifratura del messaggio avendo due utenti A e B e l'utente B, volesse inviare un messaggio ad A, dovrà cifrarlo mediante la chiave pubblica; una volta che il messaggio cifrato arriva ad A quest'ultimo dovrà decifrarlo mediante l'utilizzo della chiave privata. Con questo metodo chiunque può cifrare un messaggio per l'utente A e solo quest'ultimo sarà in grado di decifrarlo. Nella cifratura simmetrica quindi non vi è nessuna condivisione di chiavi ma bensì ogni utente genera da solo la propria coppia di chiavi, pubblica e privata, e rende pubblica la chiave pubblica. Fanno parte di questa categoria di algoritmi gli algoritmi: RSA, curva ellittica, Diffide-Helmann e DSS.

-Cifrari ibridi: Nei sistemi di cifratura ibridi oltre alle due chiavi privata e pubblica si ha l'aggiunta di una chiave di sessione. Per quanto riguarda la cifratura del messaggio avendo due utenti A e B e l'utente A volesse inviare un messaggio a B, l'utente B dovrà inviare ad A prima la chiave di sessione k cifrata con la chiave pubblica e poi il messaggio cifrato con la chiave di sessione; una volta che A ha ricevuto entrambe le informazioni da B dovrà prima decifrare la chiave di sessione k mediante la sua chiave privata per poi decifrare il messaggio vero e proprio mediante la chiave di sessione k. I vantaggi di questo metodo di cifratura sono: che la chiave di sessione viene usata solo per uno o pochi messaggi ed inoltre è molto più veloce della sola crittografia a chiave pubblica.

-RSA: RSA è basato sul problema complesso della fattorizzazione in numeri primi. Il suo funzionamento base è: 1. si scelgono due numeri primi, p e q abbastanza grandi da garantire sicurezza; 2. si calcola il loro prodotto $n=p*q$, chiamato modulo; 3. si sceglie poi un numero e, chiamato esponente pubblico, più piccolo e coprimo di $(p-1)(q-1)$; 4. si calcola il numero d, chiamato esponente privato, tale che $e*d=1(mod(p-1)(q-1))$. La chiave pubblica è (n,e) mentre la chiave privata è (n,d); per calcolare d da e, o il contrario, non basta la conoscenza di n, ma serve il numero $(p-1)(q-1)$, ovvero $\phi(n)$. Un messaggio M viene *cifrato* attraverso l'operazione $M^e \pmod n$, mentre il messaggio C viene *decifrato* con $C^d \pmod n$. Il procedimento funziona solo se la chiave e utilizzata per cifrare e la chiave d utilizzata per decifrare sono legate tra loro dalla relazione $ed=1 \pmod n$, e quindi quando un messaggio viene cifrato con una delle due chiavi può essere decifrato solo utilizzando l'altra.

-Generazione delle chiavi (RSA): Fasi: -Selezionare p,q (entrambi primi e $p \neq q$); -Calcolare $n=p*q$; -Calcolare $\phi(n)=(p-1)(q-1)$; -Selezionare l'intero e in modo che sia primo con $\phi(n)$, cioè $\gcd(e,\phi(n))=1$, con e minore di $\phi(n)$; -Calcolare d con la formula $d=e^{-1} \bmod \phi(n)$. Quindi la chiave pubblica sarà la coppia (e,n) e la chiave privata la coppia (d,n).

-Curve ellittiche su Zp: Sono i punti (x,y) in Zp X Zp tali che $y^2=x^3+ax+b \pmod p$, con $p>3$ numero primo.

-El Gamal: *Parametri:* Messaggio M; Numero casuale k appartenente a Z_{p-1} ; Chiave pubblica dell'utente A che comprende p, il generatore g e β; α che è la chiave privata dell'utente A. *Sicurezza*: -Sicurezza generazione chiavi: conoscendo (p,g, β) e sapendo che $\beta=g^{\alpha} \bmod p$ l'attaccante vuole calcolare α e ci riuscirebbe solo se sapesse calcolare il logaritmo discreto di $\beta= g^{\alpha} \bmod p$. – Sicurezza cifratura: conoscendo (p,g, β) e sapendo che $\beta=g^{\alpha} \bmod p$ e $C=(y_1, y_2)$, dove $y_1 \leftarrow$

g^k mod p e $y_2 \leftarrow M\beta^k$ mod p l'attaccante vuole calcolare M, ciò equivale a risolvere il problema di Diffie-Hellman.

-Cifratura El Gamal: Avendo a disposizione la chiave pubblica (p,g, β) dell'utente A; con p numero primo, g generatore di Z_p e $\beta=g^{\alpha}$ mod p, con α chiave privata dell'utente A appartenente a Z_{p-1} . La cifratura di M per A risulta: $y_1 \leftarrow g^k$ mod p e $y_2 \leftarrow M\beta^k$ mod p => $C \leftarrow (y_1, y_2)$, con k numero casuale appartenente a Z_{p-1} .

-Decifratura El Gamal: Dovendo decifrare il messaggio cifrato C, avendo a disposizione la chiave privata e la chiave pubblica dell'utente A oltre naturalmente al messaggio da decifrare, svolgo la seguente formula: $M \leftarrow z^{-1} y_2$ mod p , con $z \leftarrow y_1^{\alpha}$ mod p.

-Funzioni Hash: L'idea alla base è che il valore hash h(M) è una rappresentazione non ambigua e non falsificabile del messaggio M. L'output della funzione hash è detto *fingerprint* o *digest* o *hash*. Ha le proprietà di comprimere e di essere facile da computare. Le funzioni hash sono utlizzate per firme digitali, per l'integrità dei dati, e per le certificazioni nel tempo. Una funzione hash produce un output di lunghezza fissata per ogni messaggio di lunghezza arbitraria: h: $\sum^* \rightarrow \sum^n$. Due messaggi $m_1,m_2 \in x$ collidono se: $h(m_1)=h(m_2)$. Possono esistere infinite collisioni. Un possibile **attacco** alle funzioni hash consiste nel preparare due versioni di un contratto M ed M' (M favorevole alla vittima, ed M' sfavorevole), e nel modificare M' a caso (piccoli cambiamenti come aggiunta spazi) finchè h(M)=h(M'). Quindi, non appena la vittima firma M, l'attaccante ha anche la firma di M'.

-Firme Digitali e Funzioni Hash: Il problema si pone quando si vogliono creare firme digitali per messaggi lunghi. Una soluzione è la divisione in blocchi del messaggio e la firma per ogni blocco. Sorge però il problema della sicurezza, in quanto una permutazione/composizione delle firme è una nuova firma. Una soluzione di uso corrente è firmare il valore hash del messaggio: [firma di M]=Fk(h(M)). I vantaggi principali derivano dall'integrità dei dati e dall'efficienza degli algoritmi.

-Integrità dei Dati e Funzioni Hash: Un tipico uso delle funzioni hash è il seguente: -compuo al tempo T il valore hash del file M; -conservo H=(h(M)) in un luogo sicuro; -per controllare se il file è stato successivamente modificato, calcolo h(M') e verifico se H=h(M'): *h(M)* rappresenta l'impronta digitale del file. Tutto ciò assicura se un ile è stato modificato o meno.

-Proprietà (o Sicurezza) delle Funzioni Hash: -*One way (pre-image resistant):* dato y è computazionalente difficile trovare M tale che y =h(M). –*Sicurezza debole o weak collision (2nd pre-image resistance):* dato M è computazionalmente difficile trovare un altro M' tale che h(M)=h(M'). –*Sicurezza forte o strong collision (collision resistance):* computazionalmente difficile trovare 2 diversi messaggi con lo stesso valore hash.

-One-Way Hash Function (OWHF): Verifica le proprietà pre-image e 2nd pre-image resistance e viene deta weak one-way hash function.

-Collision Resistant Hash Function (CRHF): Verifica le proprietà di collision resistance e viene detta strong one-way hash function.

-Funzioni One-Way: Una funzione f è One-Way (OWF) quando, per ogni x nel dominio di f è facile calcolare y=f(x), ma, dato y, è computazionalmente inammissibile trovare x tale che y=f(x). Le differenza con OWHF consistono nel fatto che non ci sono limitazioni sul condominio (non necessariamente comprime), e non è richiesta la sicurezza forte (2nd pre-image).

-Relazioni fra le proprietà: -*Sicurezza forte→sicurezza debole* (*collision→2nd pre-image*): Sia h collision resistant e fissa un input x_1. Se h non è anche 2nd pre-image, allora è possibile trovare x_2 tale che $h(x_1)=h(x_2)$. Quindi (x_1,x_2) è una coppia di collisioni. -*2nd pre-image !→ image:* Considera la funzione identità. –*image !→2nd pre-image:* Considera la funzione $g(x)=x^2$mod n. Dato x, -x è soluzione.

-Sicurezza forte→One-way: E' possibile dimostrare che un hash collision resistant deve essere one-way. Si procede provando per contraddizione: se esiste un algoritmo di inversione, allora esiste un algoritmo Las Vegas (cioè che o dà una risposta esatta o non risponde proprio), che trova collisioni con probabilità ≥ ½.

-Paradosso del compleanno: Può essere formulato con una domanda: qual è il valore minimo di k tale che la probabilità che almeno 2 persone in un gruppo di k persone abbiano lo stesso

compleanno, sia >0,5? Ignorando il 29febbraio, si suppone che ciascun compleanno sia ugualmente probabile. Per rispondere si può definire: P(n,k)=probabilità che esista almeno un duplicato in k elementi quando ciascun elemento può assumere un valore ugualmente probabile tra 1 e n. *Esempi*: 1)Quante persone scegliere a caso affinché, con probabilità ≥ 0,5, ci siano almeno due con lo stesso compleanno? Risposta: bastano 23 persone; in un gruppo di 23 persone ci sono 253 coppie. -2)Scegliamo a caso elementi in un insieme di cardinalità n. Quanti elementi scegliere se si vuole che la probabilità che ci siano almeno due elementi uguali sia ε? $t=\sqrt{(n \cdot 2 \ln(1/(1-\varepsilon)))}$. -3)Scegliamo a caso due elementi z_1,z_2 in un insieme di cardinalità n. Calcolare la probabilità che siano diversi: $Prob(z_2 \neq z_1)=1-Prob(z_2=z_1)=1-1/n$. Se gli elementi invece sono 3, diventa (1-2/n)(1-1/n). -4)Scegliamo a caso elementi in un insieme di cardinalità n. Quanti elementi scegliere se si vuole che la probabilità che ci siano almeno due elementi uguali sia ε? $t=\sqrt{(n \cdot 2 \ln(1/(1-\varepsilon)))}$. Se ε=0,5 allora $t=1,17\sqrt{n}$, mentre se n=365 e ε=0,5 allora t=22,3.

-Attacco del compleanno: Il paradosso del compleanno può essere utilizzato per attaccare le funzioni hash. Se il digest prodotto è di n bit: genera r1 varianti del messaggio originale e r2 varianti del finto messaggio. La probabilità di trovare una coppia di varianti (una di r1 e una di r2) che producono lo stesso digest è >1/2 quando la funzione hash è applicabile a k input casuali con: -$K=SQR(2^n)=2^{n/2}$ (se n=40bit si può avere una collisione con probabilità ½ calcolando2^{20}=1.000.000 hash casuali). Modelli di lunghezza arbitraria sono trattati utilizzando hash con input fisso: il messaggio input M viene diviso in k blocchi di lunghezza fissa ($m_1,m_2,\ldots,m_k$), che possono essere trattati in *seriale/iterato* e *parallelo*.

-Modello hash parallelo: Una funzione hash parallelo è resistente alle collisioni se lo è la funzione h (Damgard).

-Modello hash iterate: Input M. Padding ed aggiunta della lunghezza di M. Si ottiene un messaggio con blocchi di taglia uguale $X_1,X_2,\ldots,X_n$. Comutazione del valore hash: H_0 è una costante iniziale; Computazione di … $H_i=f(X_i,H_{i-1})$…; Valore hash $H_n=X_n,H_{n*1}$). Nel modello generale hash iterate, una collisione per h(M) implica una collisione di f.

-Funzioni hash a cascata: Le funzioni hash sono composte in "parallelo". Trovare una collisione per $H(M)=H1(M) \cdot H2(M)$ significa trovare una collisione sia per H1 che per H2.

-Funzioni hash basate su cifrari a blocchi: Se è disponibile una implementazione di un cifrario a blocchi, si procede in questo modo: Cifrario a blocchi $E_k(\cdot)$ per input ad n bit; Funzione g che da n bit produce una chiave; $M'_1 \ldots M'_i$ è il messaggio M con eventuale padding; H_0 è una costante predefinita; H_i il valore hash; $H_i=E_{g(Hi-1)}(M'_i)$ XOR M'_i [Matyas-Meyer-Oseas] → $H_i=E_{g(Hi-1)}(M'_i)$ XOR M'_i XOR H_{i-1} [Miyaguchi-Preneel] → $H_i=E_{M'i}(H_{i-1})$ XOR H_{i-1} [Davies-Meyer].

-Attacco Meet in the Middle: Per gli hash che hanno una struttura a catena, può essere utilizzata una variante del birthday attack (attacco del compleanno), con lo scopo di generare un falso messaggio $R=(r_1,r_2)$ diverso da $M=(m_1,m_2)$ con h(R)=h(M). L'attacco si svolge in questo modo: genera r1 varianti della prima parte e vai alla fase intermedia; genera r2 varianti della seconda parte e vai indietro alla fase intermedia; Trova una collisione (la probabilità di successo è la stessa del birthday attack). Considera lo schema di hash per $M=(m_1,m_2)$: $h_1=E(m_1,IV)$, $d=h(m)=E(m_2,h_1)$, con E cifratura e IV valore iniziale. L'attacco consiste nel trovare $M^*=(m^*_1,m^*_2)$ collida con M: -Trova $i=1,\ldots,r_1$ varianti $m^*_{1,j}$, e calcola $h^*_{1,i}=E(m^*_1,IV)$; -Trova $i=1,\ldots,r_2$ varianti $m^*_{2,j}$, e calcola $h^*_{2,i}=D(m^*_2,IV)$. Se si utilizzano più round, vale lo stesso risultato.

-Little-endian e Big-endian: Consiste nel trasformare parole di byte in parole di 32 bit. Data la sequenza di byte B1,B2,B3,B4 nella parola W: -Architetture *Little-endian*: il byte con indirizzo più basso è quello meno significativo; valore parola: $W=2^{24}B4+2^{16}B3+2^8B2+2^0B1$. –Architetture *Big-endian*: il byte con indirizzo più basso è quello più significativo; valore parola: $W:2^{24}B1+2^{16}B2+2^8B3+2^0B4$.

-Obiettivi di progettazione: -*Sicurezza forte:* deve essere computazionalmente difficile trovare 2 messaggi con lo stesso valore hash. *–Sicurezza diretta:* la sicurezza non deve essere basata su problemi teorici difficili computazionalmente. *–Velocità:* l'algoritmo deve essere adatto per

implementazioni software molto veloci. *–Semplicità e Compattezza:* deve essere semplice da descrivere e da implementare; non và fatto uso di tabelle e di complesse strutture dati.
-MD4/5: -*Padding del messaggio*:MD4 processa il messaggio in blocchi di 512bit (ogni blocco consta di 16 parole di 32bit). M è il messaggio originario di b bit: M'=M 100…0 b. M' consta di un numero di bit multiplo di 512, ovvero di un numero L blocchi di 512bit, ovvero di N parole con N multiplo di 16. *–Operazioni:* MD4/5 impiegano diverse operazioni sulle word, tutte molto veloci: (XΛY), (XVY), (X XOR Y), (¬X), (X+Y), (X<<s). *–Funzioni:* Funzioni definite su parole di 32bit: -round 1: F(X,Y,Z)=(XΛY)V((¬X)ΛZ); -round 2: G(X,Y,Z)= (XΛZ)V(YΛZ)V(XΛY); -round 3: H(X,Y,Z)=X XOR Y XOR Z; -round 4: I(X,Y,Z)=Y XOR (XV(¬Z)).
-Funzione di compressione: E' composta da quattro round, di cui ognuno prende in input un blocco corrente di 512bit=16word e un valore corrente del buffer, 4word ABCD per 128bit. Ogni round consiste di 16 operazioni [ABCD,k.s.i]; L'output dell'ultima fase viene sommato all'input della prima fase (la somma avviene word a word), mentre l'output della L-esima fase è il digest a 128bit.
-Funzione compressione MD5: Ogni round consiste di 16 operazioni [ABCD.k.s.i], in cui ognuna agisce sul buffer di 4 word ABCD: A←B+(A+W(B,C,D)+X[k]+T[i]))<<s, dove K è l'indice della parola, s indica lo shift ciclico, i è l'indice dell'iterazione, W è la funzione del round (F,G,H,I), X[k]←M'[16i+k] è la k-esima word di 32bit nell'iesimo blocco e T[i] è l'iesimo elemento della tabella di 64 valori.
-Differenze tra MD4 e MD5: -*MD5:* 4 round con 4·16 operazioni; 4 funzioni logiche; 64 costanti additive; ogni passo aggiunge il risultato del passo precedente. *–MD4:* 3 round con 3·16 operazioni; 3 funzioni logiche; 2 costanti additive; ad ogni passo NON viene aggiunto il risultato del passo precedente.
-Sicurezza di MD4: MD4 è stato oggetto di molti attacchi, ad esempio con: -crittoanalisi dei primi 2 round: è stato provato che è facile trovare collisioni con round 3 omesso. –crittoanalisi degli ultimi 2 round: sono state trovate collisioni con round 1 omesso.
-SHS (Secure Hash Standard): Standard del governo americano dal 1993, con operazioni efficienti su architetture 32bit big-endian. Sfrutta gli stessi principi di MD4 e MD5, ma è più sicuro
-SHA (Secure Hash Algorithm): *Padding del messaggio:* SHA processa il messaggio in blocchi di 512bit, di cui ognuno consta di 16 parole di 32bit. M è il messaggio originario di b bit → padding: M'=M 100…0 b; M' consta di un numero di bit multiplo di 512, ovvero di un numero di parole N multiplo di 16.
-SHA-1: L'algoritmo prende come input un messaggio con lunghezza massima di meno di 2^{64}bit e produce in output un codice digest di 160bit. L'elaborazione globale di un messaggio ha la struttura con un blocco di 512bit e un codice hash e una variabile concatenamento di 160bit. L'elaborazione è costituita dai seguenti passi: aggiunta dei bit di riempimento, aggiunta della lunghezza, inizializzazione del buffer, elaborazione del messaggio in blocchi di 512bit e l'output che è il codice digest di 160bit.
-Confronto tra SHA1 e MD4/5: -*Sicurezza forte:* è maggiore in SHA-1 con output di 32bit più lungo di MD4/5 (160 contro 128). *–Sicurezza contro l'analisi:* MD5 è soggetta ad alcuni attacchi. *–Velocità:* entrambi gli algoritmi sono molto veloci; SHA-1 ha più passi (80 contro 64) e il buffer ha 160bit rispetto ai 128bit di MD5. *–Semplicità e Compattezza:* sono entrambi semplici da descrivere e da implementare, senza uso di tabelle e di complesse strutture dati.
-SHA-256: Messaggio diviso in blocchi di 512bit e parole da 32bit. **–SHA-512:** Messaggio diviso in blocchi di 1024 bit e parole da 64bit. **–SHA-384:** Valore hash = primi 384bit di SHA-512, con costanti iniziali cambiate.
-Digital Timestamp: La *marca temporale* di un documento è qualcosa aggiunto ad esso che prova che il documento è stato "prodotto" prima, dopo oppure ad un fissato momento. E' in genere facile provare che un documento è stato prodotto dopo una data fissata, mentre è in genere difficile provare che un documento è stato prodotto prima di una certa data fissata. Per il Digital Timestamp possono verificarsi diverse problematiche, per cui si possono applicare 2 soluzioni, corrispondenti a

due famiglie di protocolli: *Protocolli distribuiti* (senza Autorità Fidata, per avere più "testimonianze" del tempo e *Protocolli con "link"* (con Autorità Fidata, per collegare tra loro le marche dei documenti).
-Problemi del Protocollo Distribuito: Ci vogliono molte persone in grado di rispondere immediatamente all'utente, e si possono inoltre verificare problemi relativi alla durata (vita) delle firme digitali, per cui la firma potrebbe non essere più valida al tempo della verifica della marca temporale (ad esempio se la chiave privata è stata compromessa o lo schema di firme è stato rotto).
-**Protocollo con "link":** Riceve tutte le richiesta in intervalli prefissati, le collega tra loro, invia ad ognuna una marca temporale, e infine vincola se stesso a "non poter predatare".
-Sicurezza del sistema: Fissato il valore hash della radice, non è possibile inserire un nuovo valore nell'albero di hash o cambiarne anche un solo valore, altrimenti si determinerebbe una collisione per la funzione hash stessa. Inoltre si potrebbe rompere lo schema colludendo solo con il TSS e creando un insieme di alberi collegati lunghi "a sufficienza". Una possibile soluzione sta nel pubblicizzare SuperHash ad intervalli regolari, ad esempio ogni giorno su Internet, sui quotidiani, ecc.
-Digital Notary: -Il cliente usa del software venduto dalla Surety; -Funzione hash con un digest di 288bit (MD5+SHA); -Il sistema usa una struttura ad albero; -L'unità di tempo corrisponde ad un secondo; -Un numero seriale è inserito nel documento; -Il SuperHash è pubblicato in posti accessibili via rete, su un CD-ROM, ed ogni settimana sul Sunday New-York Times.
-PGP Digital Timestamping Service: -Il TSS firma ogni documento che riceve; -Ogni firma ha un numero seriale; -Il TSS memorizza tutte le firme che genera; -Tutte le marche emesse possono essere esaminate; -Ogni giorno viene pubblicato il numero seriale dell'ultima firma effettuata e tutte le marche emesse nella giornata.
-MAC (Message Authentication Code): E' applicato per determinare l'autenticità del messaggio M, e l'integrità del messaggio M. MAC garantisce solo autenticazione e integrità, mentre se si vuole avere anche confidenzialità, si può cifrare prima/dopo con una diversa chiave condivisa K'. *Costruzione:* il MAC può essere costruito o basandosi su cifrari a blocchi (CBC-MAC), oppure su funzioni hash (metodo del segreto prefisso, metodo del segreto suffisso, HMAC). Le *Proprietà* del MAC sono: -*Easy computation:* dato un valore M e la chiave K, MAC(K,M) è facile da calcolare. –*Compression:* M di lunghezza finita, output di lunghezza fissata. –*Computation-resistance:* data una o più coppie MAC(K,Mi), non è possibile calcolare un nuovo MAC(K,Mj), Mi≠Mj. –*Key non-recovery:* data una o più coppie MAC(K,Mi), non è possibile calcolare K.
-Attacco al MAC: Gli attacchi al MAC hanno principalmente tre scopi: -*Total break*, per determinare la chiave K; -*Selective forgery*, in cui dato un messaggio M, si vuole determinare y tale che y=MAC(M,K); -*Existential forgery:*, per determinare una coppia (M,y) tale che y=MAC(M,K).Vi sono tre tipi di attacco al MAC: -*Known Message Attack:* l'attaccante conosce una lista di messaggi ed i relativi MAC; -*Chosen Message Attack:* l'attaccante sceglie dei messaggi e chiede alla vittima di computarne i MAC; -*Adaptive Chosen Message Attack:* è come il Chosen Message Attack, ma le scelte dipendono dalle risposte precedenti.
-MAC basati su Funzioni Hash: Presenta notevoli vantaggi: sono in genere più veloci dei cifrari a blocchi, sono in genere incluse nelle funzioni di libreria, non ci sono restrizioni sull'esportazione dagli USA. Bisogna però fare molta attenzione alla loro costruzione.
-Metodo del segreto prefisso: MAC(K,M)=H(K,M). *Existential forgery attack:* per funzioni hash iterate, considero M'=Mx_{n+1} e so calcolare f(x_{n+1}H(K,M))=H(K,Mx_{n+1})=MAC(K,M'). Una possibile soluzione è H(K,L,M), con L=lunghezza di M.
-Metodo del segreto suffisso: MAC(K,M)=H(M,K). *Existential forgery attack:* attacco del compleanno: Dato M, in $O(2^{|hash(.)|/2})$ passi calcolare M': H(M)=H(M'). Quindi, la soluzione è H(M,K)=H(M',K)=MAC(M',K).
-HMAC: HMAC è una tipologia di codice per l'autenticazione di messaggi (Message Autentication Code - MAC) basata su funzione di hash utilizzata in diverse applicazioni legate alla sicurezza informatica. Tramite HMAC è infatti possibile garantire sia l'integrità che l'autenticità di un

messaggio. HMAC utilizza infatti una combinazione del messagio originale e una chiave segreta per la generazione del codice. Il vantaggio di HMAC è il non essere legata a nessuna funzione di hash particolare, questo per rendere possibile una sostituzione della funzione nel caso fosse scoperta debole. Nonostante ciò le funzioni più utilizzate sono MD5 e SHA-1. Il messaggio viene suddiviso in blocchi di lunghezza pari a j bit. Seleziono una chiave segreta K, se queta risulta essere più lunga di j bit a questa applico la funzione H. Quello che si ottiene è detta K', la chiave di HMAC. K' = K se |K| = j bit K' = K + padding di zeri se |K| < j bit K' = H(K) se >= jbit. Una volta definita K l'eventuale padding della chiave originale la funzione HMAC calcolerà il valore nel seguente modo: $HMAC_K(M)=h((K\oplus opad)||h((K\oplus ipad)||M'))$, con M'=alla suddivisione in blocchi del messaggio. -*Sicurezza HMAC:* La sicurezza dipende dalle proprietà della funzione hash usata da HMAC. Se l'attaccante ha successo in un attacco ad HMAC allora: può computare l'output della funzione di compressione anche quando IV è casuale e/o è a lui sconosciuto, e può computare collisioni nella funzione hash anche quando IV è casuale e/o sconosciuto. *–Attacchi ad HMAC:* Il miglior attacco conosciuto è basato sul paradosso del compleanno: occorrono $2^{|hash(.)|/2}$ coppie $(M,HMAC_k(M))$. Per attaccare HMAC sono necessarie molte coppie M,HMAC: l'avversario non può calcolarle perché non conosce K, e deve quindi osservare un flusso di messaggi generati con la stessa chiave. HMAC con MD5 è ragionevolmente sicura.

-Utilizzo di Hash e MAC: Le funzioni HASH e MAC insieme all'uso di cifrari a/simmetrici garantiscono diverse proprietà durante lo scambio di messaggi, fra cui segretezza e autenticazione (sia del testo in chiaro che di quello cifrato).

-Firma digitale: Equivalente alla firma convenzionale. *Requisiti:* la firma digitale deve poter essere facilmente prodotta dal legittimi firmatario, nessun utente deve poter riprodurre la firma di altri, e chiunque può facilmente verificare una firma digitale.

-Attacco alla firma digitale: Esistono tre tipi di attacchi alle firme digitali: -*Key-only attack*, in cui l'attaccante conosce solo la chiave pubblica della vittima; -*Known Message Attack*, in cui l'attaccante conosce una lista di messaggi e le relative firme della vittima; -*Chosen Message Attack*, in cui l'attaccante sceglie dei messaggi e chiede alla vittima di firmarli. Gli *Scopi* dell'attacco alle firme digitali possono essere principalmente tre: -*Total break*, per determinare la chiave privata della vittima, e poter firmare qualsiasi messaggio; -*Selecitve forgery*, in cui dato un messaggio M, si può determinare la firma F tale che VERIFICA(F,M,kpub)=SI; -*Existential forgery,* per determinare una coppia (M,F) tale che VERIFICA (F,M,kpub)=SI.

-Firma RSA: Firma di M: $F\leftarrow M^d \bmod n$. *Verifica della firma* di M: vera se $M=F^e \bmod n$, altrimenti è falsa. *Sicurezza della firma* RSA: -Selective forgery, Key only attack: per falsificare la firma di M, bisogna calcolare $M^d \bmod n$; il che è equivalente a "rompere" il crittosistema RSA. –Existential forgery, Key only attack: per generare messaggi e firma, bisogna scegliere F a caso, e calcolare $M\leftarrow F^e \bmod n$. –Existential forgery, Known message attack: per generare messaggi e firme conoscendo le coppie (M_1,F_1) e (M_2,F_2), si possono utilizzare le proprietà di omomorfismo: $F_1=M_1^d \bmod n$, $F_2=M_2^d \bmod n$, $(F_1,F_2)^e \bmod n=F_1^eF_2^e \bmod n=M_1M_2 \bmod n$: $F_1F_2 \bmod n$ è una firma valida per $M_1M_2 \bmod n$. –Selective forgery, Chosen message attack: per falsificare la firma di M, si può scegliere M_1 e M_2 tali che $M=M_1M_2 \bmod n$, e chiedere alla vittima di firmare M_1 e M_2 ottenendo F_1 e F_2: $F_1F_2 \bmod n$ è una firma valida per M.

-Firma digitale di messaggi grandi: Se M>n, si può firmare in questo modo: $Firma(M)\leftarrow(Firma(M_1), Firma(M_2),\ldots)$.

-Firma digitale con hash: $Firma(M)\leftarrow(Firma(h(M)))$. I vantaggi scaturiscono dall'efficienza, dall'integrità, e dalla sicurezza.

-Firma RSA con hash: Firma di M: $F\leftarrow[h(M)]^d \bmod n$. *Verifica firma* RSA con hash: Verifica firma di M: vera se $h(M)=F^e \bmod n$, altrimenti è falsa. *Sicurezza firma* RSA con hash: -*Existential forgery, Key only attack:* Per generare messaggi e firme si può scegliere F a caso, e calcolare $z\leftarrow F^e \bmod n$, e $M\leftarrow h^{-1}(z)$. Per invertire "h" si procede con $M\leftarrow h^{-1}(z)$.

-Schemi di firme digitali (oltre a DSS): El-Gamal (che è una versione modificata di DSS), Schnorr, Fiat-Shamir, GQ (Guillou-Quisquater), DSA, ECDSA (DSA su curve ellittiche). Ci sono

vari schemi: One-time signatures, sono schemi validi per un solo messaggio (altrimenti la firma è compromessa) e sono: Rabin, Lamport-Merkle, Matyas_Meyer (basto su DES) e GMR (Goldwasser,MIcali,Rivest). Un altro è la Blind signatures, in cui il firmatario non vede il messaggio che firma. Nell'Arbitrated signatures si richiede una terza parte fidata (TTP) per la generazione e la verifica della firma; con l' Undeniable signatures la verifica viene fatta in collaborazione con il firmatario. Fail stop signatures una firma falsa viene sempre scoperta (il firmatario può sempre dimostrarlo). Chameleon signatures solo il destinatario della firma è in grado di provare la validità.

-Digital Signature Standard (DSS): Lo standard DSS utilizza un algoritmo progettato per fornire solo la funzionalità di firma digitale. A differenza di RSA, non può essere utilizzato per la crittografia o per lo scambio delle chiavi. Ciò nonostante si tratta di una tecnica a chiave pubblica. Le firme DSS sono sempre di 320 bit (buone per smart card) e la Sicurezza è basata sull'intrattabilità del problema del logaritmo discreto. *Parametri:* La chiave privata è (p, q, α, s), mentre quella pubblica è (p, q, α, β), dove s è un numero casuale, s<q, $\beta = \alpha^s \bmod p$ e p è il primo di L bit, $512 \leq L \leq 1024$ (L multiplo di 64), q è un numero primo di 160 bit, q|(p-1), mentre α è un numero in Zp* di ordine q ($\alpha^q = 1 \bmod p$). Per generare α bisogna usare il concetto di ordine di un elemento, cioè: Ordine di α € Zn* = è il più piccolo intero positivo r tale che $\alpha^r = 1 \bmod n$; quindi per il Teorema di Lagrange: Per ogni α € Zn*, ord(α) divide Φ(n); Se p è primo, ord(α) divide p-1.

-Firma DSA: Funzionamento dell'algoritmo: Vi sono 3 parametri pubblici che possono essere comuni a un gruppo di utenti. Viene scelto un numero primo q di 160 bit, successivamente viene scelto un numero primo p tale che q divida (p-1), e infine g. Avendo a disposizione questi numeri, ciascun utente seleziona una chiave privata e genera una chiave pubblica. La chiave privata x dovrebbe essere scelta casualmente, mentre la chiave pubblica viene calcolata dalla chiave privata come $y=g^x \bmod p$. Per *creare una firma* un utente calcola 2 quantità, r ed s, che sono funzioni dei componenti della chiave pubblica, della chiave privata dell'utente, del codice hash del messaggio e di un intero aggiuntivo k (generato casualmente). Per *verificare una firma* il destinatario genera una quantità v che è funzione delle componenti della chiave pubblica globale, della chiave pubblica del mittente e del codice hash del messaggio in arrivo. La firma è valida se la quantità v corrisponde alla componente r della firma. (((*Generazione firma*: La firma DSA è composta da (γ, δ) dove "$\gamma = (\alpha^r \bmod p) \bmod q$", "$\delta = (SHA(M)+s\ \gamma)r^{-1} \bmod q$", mentre r è numero casuale di un intervallo tra [1,q-1]. Firma $_{(p,\ q,\ \alpha,\ s)}(M,r) = (\gamma, \delta)$. Per *verificare la firma* si prendono due variabili e' ed e''; e'=SHA(M) $\delta^{-1} \bmod q$, mentre e''= $\gamma\ \delta^{-1} \bmod q$. La firma è vera se $\gamma = (\alpha^{e'}\ \beta^{e''} \bmod p) \bmod q$, altrimenti è falsa. La lunghezza della firma è di 320 bit. La sicurezza è basata sul valore privato s, i valori p,q, α possono essere gli stessi per un gruppo di utenti, quindi un'autorità sceglie p,q,α, mentre il singolo utente sceglie solo s e calcola β.)))

-Accordo su chiavi: Per mettersi d'accordo sulle chiavi ci sono due schemi: Diffie-Hellman che è basato sull'intrattabilità del problema del logaritmo discreto e quello di Puzzle di Merkle che non è basato su alcuna assunzione computazionale.

-Diffie-Hellman: si sceglie un numero primo p, generatore g di Z_p^* ; g è generatore di Z_p^* se $\{g^i | 1 \leq i \leq p-1\} = Z_p^*$. Generatori di Zn*: Zn* ha un generatore $n = 2,4,p^k,2p^k$, con p primo e $k \geq 1$. Se p è primo, allora Zp* ha un generatore; Il numero di generatori di Zn* è Φ(Φ(n)), Se p è primo, il numero di generatori di Zp* è Φ(p-1). Funzionamento: alice sceglie un x appartenente a Z_p^* e bob un y appartenente a Z_p^*, poi alice manda $g^x \bmod p$ mentre bob manda $g^y \bmod p$, entrambi sanno che $k = g^{xy} \bmod p$, alice quindi dovrà elevare il messaggio di bob a x e bob il messaggio di alice alla y, per decifrare il messaggio dell'altro. *–Sicurezza:* La sicurezza dello scambio di chiavi Diffie-Hellman, si basa sul fatto che, mentre è relativamente facile calcolare dei valori esponenziali modulo un numero primo, è molto difficile calcolare i logaritmi discreti. Per valori primi molto estesi, quest'ultima operazione è considerata impossibile.

-Logaritmo discreto: La sicurezza di molte tecniche crittografiche si basa sulla intrattabilità del logaritmo discreto: come Crittosistema ElGamal, l'Accordo su chiavi Diffie-Hellman le Firme digitali DSS. Esso dice: Dati a,n,b calcolare x tale che $a^x = b \bmod n$. Se n è primo, i migliori

algoritmi hanno complessità Ln[a,c] = $O(e^{(c+o(1))(\ln n)a(\ln\ln n)1-a})$ con c > 0 ed 0 < a < 1 , il Miglior algoritmo è il Number field sieve, il tempo medio euristico è di Ln[1/3, 1.923].
-Problema di Diffie-Hellman: in input si ha un numero primo p, un generatore g, g^x mod p e g^y mod p (dove x è la chiave privata di A e y è la chiave privata di B); in output bisogna ottenere g^{xy} mod p. Bisogna quindi trovare x; il miglior algoritmo conosciuto calcola l'algoritmo discreto $\log_{g,p}(g^x \bmod p)$, il problema è che non si sa se sono equivalenti. Per scegliere le chiavi di Diffie-Hellman si scelgono a caso 2 numeri primi p1 p2, $p = 1 + 2p_1p_2$, Se p non è primo, go to 1. g = **Scegli_generatore**$(p,(2,1,p_1,1,p_2,1))$.
-Puzzle di Merkle: *Schema:* Non basato su assunzioni computazionali; Alice genera n chiavi distinte e "nasconde" ogni chiave in un puzzle. Il puzzle contiene informazioni per il calcolo della chiave, La soluzione di un puzzle richiede un tempo ragionevole, La soluzione di tutti i puzzle richiede un tempo troppo elevato. Il puzzle è composto da (x, ID, S), dove x è la soluzione del puzzle (richiede 2^{35} operazioni in media), ID è l'identificativo del puzzle (che è unico per ciascun puzzle) ed S è un valore noto (serve per garantire l'unicità della soluzione del puzzle, ad esempio 32 bit nulli). Dato il puzzle si sceglie una chiave k, poi si computa y <- $CBC\text{-}DES_k$ (x, ID, S), in return si deve avere (y, primi 20 bit di k). Il vantaggio di questo sistema è che il tempo delle computazioni di alice per la costruzione del puzzle sarà $\theta(n)$ e quello di bob x la risoluzione del puzzle di $\theta(t)$, mentre un attaccante risolverà in media n/2 puzzle, quindi in un tempo di $\theta(tn)$; ora se n= $\theta(n)$, il tempo di alice sarà sempre $\theta(n)$, quello di bob $\theta(n)$ mentre quello dell'attaccante $\theta(n^2)$.
-Public Key-Infrastructure: Le chiavi pubbliche vengono distribuite secondo alcune tecniche come: Invio point-to-point su canale fidato, Annuncio pubblico, Directory disponibile pubblicamente, Autorità per le chiavi pubbliche, Certificati per le chiavi pubbliche. Il primo può essere fatto per esempio con scambio diretto, uso di un corriere fidato, invio su canale pubblico, autenticazione (per esempio: hash su canale fidato) e va bene per un uso non frequente e per piccoli sistemi; L'annuncio pubblico consiste nell'invio ad altri utenti o al Broadcast della chiave, ad esempio: aggiunta della chiave pubblica PGP ai messaggi inviati a forum pubblici, il problema principale è la fiducia; la directory disponibile pubblicamente è un entità fidata che gestisce la directory di chiavi pubbliche. Ogni partecipante: registra la propria chiave pubblica di persona o in modo autenticato e può aggiornare la propria chiave se è usata da troppo tempo o la chiave privata è compromessa, inoltre può accedere alla directory (è necessaria comunicazione sicura ed autenticata).L'autorità per le chiavi pubbliche gestisce directory chiavi pubbliche, ha una chiave pubblica nota a tutti gli utenti e ogni utente chiede la chiave pubblica desiderata, quindi l'autorità la invia. Gli svantaggi sono la necessità di server on-line e un effetto collo di bottiglia.
-Certificati: Mondo digitale un'autorità riconosciuta lega un nome ad una chiave pubblica, l'autorità di certificazione rappresenta una terza parte fidata la cui firma garantisce il legame tra chiave ed identità. Alcune proprietà dei certificati sono che ognuno può leggerli e determinare nome e chiave pubblica, ognuno può verificarli ed assicurarsi dell'autenticità, solo l'Autorità può crearli ed aggiornarli… Esempi di dati in un certificato: periodo di validità chiave pubblica, numero seriale o identificatore chiave, info addizionali su chiave (ad es., algoritmi ed utilizzo), info addizionali su utente, stato della chiave pubblica; formato più diffuso: definito dallo standard internazionale ITU-T X.509. U*n certificato può essere revocato* per vari motivi come Compromissione chiave privata, Info non più valide (es., cambio affiliazione), Non più utile per lo scopo prefissato, Compromissione algoritmo, Perdita o malfunzionamento (di security token, perdita di password o PIN), Cambio politiche di sicurezza (es., la CA non supporta più servizi per certificati). *I metodi per revocarli sono:* Data scadenza dentro un certificato (Certificati "a breve scadenza"), Notifica manuale (Informazione tramite canali speciali, utilizzabile solo per sistemi piccoli o chiusi), File pubblico di chiavi revocate (si usa certificate Revocation List (CRL)) e Certificato di revoca, che sostituisce certificato revocato nella directory. La Certificate Revocation List (CRL) è una lista firmata dalla CA contenente: numeri seriali dei certificati emessi revocati (ma non ancora scaduti), quando è avvenuta la revoca, altro (per es., motivi). La data della CRL indica quanto sia aggiornata.

-Standard dei certificati X.509 è il più diffuso standard per i certificati. Ci sono varie versioni dalla 1 alla 3, e sono composte da vari campi: version (che può avere valore 1 per default, 2 se presente "issuer unique identifier" oppure "subject unique identifier", 3 se ci sono estensioni), serial number (che ha valore intero, unico per ogni CA e identifica senza ambiguità il certificato), Signature algorithm ID (è l'algoritmo usato per firmare il certificato, i parametri associati, questo parametro è poco importante perché informazione ripetuta), Issuer name (è il nome X.500, sequenza di coppie nome-valore che identificano univocamente un'entità, della CA che ha creato e firmato il certificato), Validity period (ci sono due date, una per quando è stato creato il certificato e l'altra per quando scade), Subject name (è il nome utente del certificato, cioè chi conosce la chiave privata corrispondente), Subject's public key (è la chiave pubblica del soggetto ed è identificativo dell'algoritmo e dei parametri associati) qua finiscono i campi della versione 1; Issuer unique identifier (è opzionale, è una sequenza di bit utile per identificare la CA, che ha emesso il certificato nel caso in cui il nome X.500 sia stato riutilizzato) Subject unique identifier (è opzionale, è una stringa di bit utile per identificare il soggetto nel caso che il nome X.500 sia stato riutilizzato) qua finisce la versione 2; la versione 3 ha in + il campo extension (che raccoglie diversi campi di estensione). Un campo comune a tutte le versioni è la firma dei precedenti campi (firma dell'hash di tutti gli altri campi, include il signature algorithm identifier).
-Autenticazione X.509: ci sono tre procedure: Autenticazione One-way, Autenticazione Two-way, Autenticazione Three-way. Assumiamo che i due comunicanti conoscano le chiavi pubbliche, come primo messaggio cè uno scambio di certificati oppure i certificati sono ottenuti dalle directory. Nella one-way A invia a B la Firma A $\{t_A, r_A, B, sgnData, E_{PKB}(K_{AB})\}$ dove t_A è il timestamp con l'espiration time, r_A è il campo nonce e gli altri due sono campi opzionali. Un attaccante tenterà di effettuare un attacco replay, ma il campo nonce è unico fino all'espiration time in modo appunto di vanificare questo attacco (B deve conservare il nonce); si conserva integrità ed originalità del messaggio. Nel two-way B risponde al messaggio di A mandando la sua firma (integrità ed originalità messaggio di B). Nel three-way si vuole eliminare il check dei timestamp (questo è necessario in assenza di clock sincronizzato), quindi si fanno le cose precedenti e in + A invia Firma $_A$ $\{r_B\}$.
-X.509 versione 3: Colma i requisiti non soddisfatti dalla versione 2: Subject field non adeguato: nomi X.509 sono corti, e mancano dettagli identificativi che potrebbero essere utili, -Subject field non adeguato per le applicazioni che riconoscono entità dall'indirizzo email, URL, -Vi è necessità di indicare politiche di sicurezza, -Vi è necessità di limitare il danno che potrebbe fare una CA maliziosa, ponendo vincoli all'applicabilità di un particolare certificato, -E' importante distinguere chiavi diverse usate dallo stesso utente in tempi diversi. L'estensione opzionale della versione 3 la rende una soluzione flessibile, migliore dell'aggiungere altri campi fissi alla versione 2. Ogni estensione contiene: -Identificatore estensione, -Indicatore di criticità (indica se l'estensione può essere ignorata, se è TRUE e l'implementazione non riconosce l'estensione allora deve trattare il certificato come non valido) e un Valore estensione. Le estensioni si possono raggruppare in tre categorie: -***Key and Policy Information*** che raggruppa: Authority key identifier (indica quale di + chiavi pubbliche della CA usare per verificare la firma di un certificato o della CRL), Subject key identifier (identifica quale di + chiavi pubbliche viene certificata), Key usage (è una restrizione sull'uso della chiave certificata), Private-key usage period (periodo uso della chiave privata), Certificate policy (insieme di regole che indica l'applicabilità di un certificato ad una comunità e/o classi di applicazioni con requisiti di sicurezza comuni), Policy mappings (usato solo per CA da altre CA. permette ad una CA di indicare che una propria politica può essere considerata equivalente ad un'altra politica usata dalla CA soggetto). -***Certificate Subject and Issuer Attributes*** che raggruppa Subject alternative name (contiene uno o + nomi alternativi, in formati alternativi, imp per app che hanno formati propri per i nomi) Issuer alternative name (contiene uno o + nomi alternativi, in formati alternativi), Subject directory attributes (contiene attributi della directory X.500 per il soggetto del certificato). -***Certification Path Constraints*** che raggruppa: Basic constraints (indica se il soggetto può agire come CA. se si possono specificare vincoli sulla

lunghezza della certification path), Name constraints (indica uno spazio dei nomi in cui tutti i seguenti certificati in un certification path devono essere), Policy constraints (inibisce policy mappings per la parte rimanente della certification path).

-Secret Sharing (Condivisione di segreti): Un dealer vuole condividere un segreto S tra n partecipanti in modo che: - k o più partecipanti possano ricostruire S; - k-1 o meno partecipanti non hanno alcuna informazione su S. Per la condivisione dei segreti ci sono due schemi: (n,n) e (k,n). Con il primo schema il dealer sceglie un numero primo p e degli elementi in Z_p da a_1 a a_{n-1} e $a_n = S - a_1 - \ldots - a_{n-1} \bmod p$. Poi il dealer distribuisce ai vari partecipanti a_1, a_2 ecc… *Ricostruzione:* Come detto prima tutti i partecipanti insieme sanno che il $S=a_1+\ldots+a_{n-1}+a_n \bmod p$, ma basta la mancanza di un solo partecipante che tutti gli altri non hanno alcuna informazione sul segreto. *Schema (k.n):* inizializzazione: il dealer sceglie un numero primo p e degli elementi in Z_p da a_1 a a_{n-1} e $a_n = S - a_1 - \ldots - a_{k-1} \bmod p$. Calcolo share schema: il dealer calcola "$f(x)=S+a_1x+\ldots+a_{k-1}x^{k-1}$" e "for i=1 to n do y1<- f(i)". Il dealer distribuisce ai vari partecipanti y_1 fino a y_n . Per la ricostruzione: il segreto può essere ricostruito da k partecipanti anche senza il partecipante p_n . Sui k partecipanti abbiamo le seguenti informazioni: K equazioni $y_i = S+a_1i+\ldots+ a_{k-1}i^{k-1}$ per $i=i_1, i_2, \ldots i_k$ e k incognite: S, a_1, …, $a_{k-1.}$ Lo stesso schema può essere visto utilizzando la formula di Lagrange per l'interpolazione polinomiale: -Un insieme di n punti individua univocamente una curva di ordine n-1 (Es. dati due punti esiste una sola retta passante) - Per uno schema a soglia k utilizzo un polinomio di grado k-1: $y_i = S+a_1i+\ldots+ a_{k-1}i^{k-1}$ (Distribuisco a ciascun partecipante un punto, Solo k partecipanti avranno k punti e possono ricostruire la curva, Risolvendo k equazioni in k incognite). Ricostruzione del segreto: k-1 partecipanti senza p_n non riescono ad avere alcuna informazione sul segreto. Le informazioni con k-1 partecipanti sono: -k-1 equazioni: $y_i = S+a_1i+\ldots+ a_{k-1}i^{k-1}$ per $i=i_1, i_2, \ldots ik-1$; -k incognite: S, $a_1,\ldots, a_{k-1}$; -Non possono ricostruire il segreto: -Ogni segreto è equamente possibile.