Algoritmo di cifratura simmetrico a blocchi Blowfish

Per il corso di "Sicurezza Informatica"

del Prof. Alessandro Bianchi

Fabrizio Buonomo, Walter Manfredi

Febbraio 2018

Indice

Abstract

In questo elaborato trattiamo dell'algoritmo di cifratura Blowfish. Per poterlo inquadrare a pieno esponiamo i concetti di base per la crittografia (cap. 1), distinguendone le due principali tipologie: crittografia simmetrica e asimmetrica (cap. 2), e le diverse modalità di cifratura simmetrica (cap. 3) fino ad individuare le caratteristiche che rendono sicuro un algoritmo di cifratura (cap. 4). Blowfish viene introdotto a partire dal documento di Bruce Schneier in cui è stato presentato, illustrandone le caratteristiche che lo contraddistinguono ed i passaggi eseguiti per la cifratura e decifratura dei messaggi. Parliamo dei più noti tentativi di crittanalisi effettuati su Blowfish (cap. 6) inquadrando le caratteristiche che lo rendono ad oggi un algoritmo ancora inviolato, da cui le sue diverse applicazioni (cap. 7). Nelle conclusioni riportiamo alcune considerazioni sui casi in cui Blowfish è preferibile rispetto ad altri algoritmi simmetrici di cifratura, indicando gli eventuali sostituti.

1 Introduzione alla crittografia

Il termine crittografia (o criptografia), le cui radice e desinenza vengono dal greco *kryptòs* che vuol dire nascosto, segreto, e *graphía* scrittura, indica l'area della criptologia che concerne la progettazione e l'utilizzo di algoritmi, protocolli, e sistemi che vengono utilizzati per rendere un'informazione segreta, inintellegibile se non da chi conosce la chiave usata nella composizione.

L'altra area della criptologia è la crittanalisi, la scienza e l'arte di risolvere i crittosistemi, trovando dei modi per interpretare i messaggi cifrati riportandoli in chiaro, anche senza autorizzazione.

Un sistema crittografico, o cifrario, o crittosistema, è definito come "l'insieme di algoritmi di crittografia e dei processi di gestione delle chiavi che prevede l'uso degli algoritmi in diversi contesti". [IETb]

La crittografia è realizzata tramite due fasi: cifratura e decifratura.
La fase di cifratura consiste nel nascondere le informazioni prima della loro trasmissione. Questo avviene trasformando le informazioni dalla forma originale (plaintext, o messaggio in chiaro) ad un'altra non comprensibile (ciphertext, o messaggio cifrato, crittogramma) con un determinato algoritmo di cifratura dipendente da una chiave, o password.
Al contrario, la decifratura consiste nella trasformazione dal crittogramma al messaggio originale per mezzo dello stesso algoritmo e della stessa chiave.

L'algoritmo di cifratura consiste di un certo numero di operazioni che mettono in atto il processo di cifratura, e la chiave è rappresentata da una stringa, solitamente molto lunga, che viene utilizzata dall'algoritmo in alcuni passaggi.

La crittografia fornisce un livello di sicurezza fondamentale in situazioni in cui il canale di comunicazione è suscettibile ad intercettazioni [KB10], costituendo la prima linea di difesa nella protezione dei dati da intrusi.

I passi coinvolti nel modello crittografico tradizionale, come riportati in figura 1, prevedono: [TK11]

- Un mittente che desidera inviare un messaggio ad un destinatario.
- Il messaggio originale, plaintext, è convertito in una forma disordinata, ciphertext. L'algoritmo produce un ciphertext diverso al variare del plaintext e della chiave utilizzata.
- Il messaggio cifrato viene trasmesso attraverso un canale di comunicazione.
- Raggiunto il destinatario, l'algoritmo, utilizzando la stessa chiave di cifratura, converte il messaggio cifrato nella forma originale.

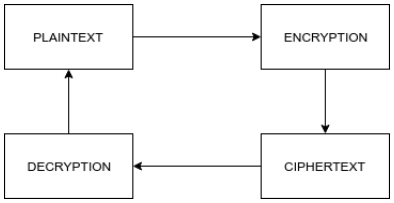

Fig. 1: Modello di comunicazione tradizionale

La sicurezza delle informazioni è di fondamentale importanza in una società che si affida alla tecnologia per lo svolgimento di diverse operazioni nella vita di tutti i giorni.
La crittografia è l'approccio più comunemente utilizzato per garantire la sicurezza nei sistemi digitali, per tenere al sicuro dati sensibili come password, dati relativi a carte di credito, conti bancari, comunicazioni militari, comunicazioni satellitari. [Mou05]
Può essere implementata con diversi livelli di sicurezza a seconda delle esigenze per ogni applicazione. [Sil+16]
Nell'attuare misure di protezione delle informazioni all'interno di un sistema, protocollo, o servizio, un progettista deve tenere conto di vari elementi: [PHS03]

- una specifica dettagliata dell'ambiente in cui il sistema (protocollo o servizio) andrà ad inserirsi, inclusa una lista di obiettivi di sicurezza da rispettare;
- una lista di minacce con la descrizione dei punti del sistema in cui potrebbe avvenire un'eventuale manomissione del flusso di informazioni;
- il livello di protezione richiesto o le risorse computazionali che ci si aspetta da un eventuale attaccante;
- il tempo di vita ipotizzato per il sistema

1.1 Servizi di sicurezza

Nell'ambito della sicurezza informatica, l'ITU-T in X.800 [ITU], così come l'IET in RFC 4949 [IETa], definisce alcuni obiettivi della sicurezza, detti servizi.
I servizi di sicurezza che l'uso della crittografia permette di soddisfare sono:

- Autenticazione delle entità: il processo di verifica dell'identità del mittente e del destinatario prima dell'invio e della ricezione di un messaggio. Si considera autenticato un qualsiasi individuo in possesso della chiave di cifratura per una comunicazione, perciò è fondamentale definire in che modo la chiave viene comunicata tra i partecipanti ad una conversazione e come viene custodita.
- Privacy/Confidenzialità: garantisce che nessuno sia in grado di leggere il messaggio se non il destinatario previsto. Solitamente è questa la caratteristica secondo cui la maggior parte delle persone identifica un sistema come sicuro. Tuttavia, nell'ambito di un sistema crittografico, questo obiettivo implica unicamente che le persone che hanno ricevuto autenticazione, nello specifico coloro in possesso della chiave di cifratura, sono abilitate a interpretare il contenuto di un messaggio.

- Integrità dell'informazione: con integrità si intende che il contenuto dei dati trasmessi è sicuro e non subirà alcun tipo di modifica nel percorso tra i nodi finali (mittente e destinatario).

1.2 Caratterizzazione dei sistemi crittografici

I sistemi crittografici vengono caratterizzati su tre dimensioni indipendenti: dal tipo di operazioni effettuate per la trasformazione da testo in chiaro a crittogramma, dal numero di chiavi utilizzate, dal modo in cui il testo in chiaro è processato.[Sta17]

Tutti gli algoritmi di cifratura si basano su due proprietà generali, che Claude Shannon ha definito in uno degli articoli fondamentali sui fondamenti teorici della teoria dell'informazione [Sha49], per annullare gli effetti di un'analisi statistica: diffusione e confusione.
Nel documento di Shannon viene definita la confusione come il basso grado di correlazione tra ciphertext e la chiave, in modo tale che non si possa risalire ad essa a partire dal ciphertext. La confusione viene raggiunta con il principio di sostituzione, per cui ogni elemento del plaintext (bit, lettere, gruppi di bit o di lettere) viene mappato in un altro elemento.
La diffusione denota la capacità dell'algoritmo di annullare l'uso di informazioni statistiche relative al linguaggio adoperato nel plaintext. Per ottenere diffusione gli elementi del plaintext vengono trasposti in maniera casuale. In generale un algoritmo può prevedere qualunque tipo di operazione, purché non perda traccia dell'informazione, e che sia quindi reversibile. I sistemi più comuni prevedono diversi stadi di sostituzioni e trasposizioni.

I sistemi in cui mittente e destinatario usano una stessa chiave per cifrare e decifrare il messaggio sono detti simmetrici, e sono considerati come sistemi crittografici tradizionali, a chiave singola, o segreta.
Nei casi in cui le chiavi usate nel processo di cifratura e decifratura siano diverse si parla di sistemi crittografici asimmetrici, sistemi a due chiavi, o di cifratura con chiave pubblica.
In seguito esporremo più in dettaglio riguardo le differenze tra i due tipi di crittografia, esponendo le debolezze a cui sono esposti.

Le modalità con cui un messaggio può essere processato da un cifrario sono la cifratura a blocchi e la cifratura a flusso.
Un cifrario a blocchi è un sistema che processa il messaggio di input un blocco di bit per volta, producendo un blocco di output della stessa dimensione, e l'operazione viene ripetuta fino al termine del messaggio. Un cifrario a flusso, o a caratteri o a stati, cifra in modo indipendente i simboli (generalmente singoli bit o byte) che codificano il testo in chiaro e la trasformazione di ogni simbolo successivo varia al procedere della cifratura secondo un certo schema.
Nella nostra trattazione ci soffermeremo sui cifrari a blocchi, categoria in cui rientra l'algoritmo Blowfish.

1.3 Crittoanalisi e attacchi a forza bruta

Agli occhi di un individuo che si trova inconsapevolmente davanti ad un crittogramma, questo può sembrare estremamente complicato da decifrare o venire catalogato come testo

scritto in una lingua sconosciuta, da cui la decisione di ignorare il messaggio. Tuttavia i sistemi crittografici possono essere suscettibili ad attacchi da parte di malintenzionati o curiosi, in letteratura anche noti come *Trudy* da *intruders*, da qui in poi attaccanti, che sono interessati a vanificare gli scopi della cifratura.

Tipicamente, l'obiettivo di un attacco ad un sistema crittografico è quello di recuperare la chiave utilizzata nella comunicazione piuttosto che il messaggio in chiaro relativo ad un singolo crittogramma. Infatti, in questo modo, è possibile interpretare ogni crittogramma ottenuto a partire dalla stessa chiave. Per farlo vengono seguiti due approcci generali: crittoanalisi e attacchi a forza bruta.[Sta17]

Gli attacchi basati sulla crittoanalisi sfruttano la natura delle operazioni dell'algoritmo e la conoscenza di alcune caratteristiche del plaintext, come ad esempio l'argomento trattato in una conversazione, o la formattazione del testo all'interno del messaggio. È possibile, infatti, che all'interno di diversi messaggi la parte relativa all'intestazione e quella conclusiva siano formattate sempre allo stesso modo, permettendo di identificare trame ben precise nel crittogramma e metterle in corrispondenza col testo originale.

L'approccio degli attacchi a forza bruta, invece che utilizzare proprietà matematiche dell'algoritmo o fare deduzioni dalle conoscenze a disposizione, consiste nell'andare a tentativi e cercare di indovinare la chiave utilizzata per la cifratura tra tutte quelle presenti nello spazio delle chiavi (per una chiave composta da n bit lo spazio delle chiavi ha cardinalità 2^n), fino ad ottenere una traduzione intelligibile del crittogramma.
In media, con un attacco a forza bruta, è possibile trovare la chiave desiderata dopo aver provato metà delle combinazioni possibili, ma nulla assicura che non sia possibile trovarla più rapidamente.

Che si adotti il primo o il secondo approccio, le conseguenze di un attacco portato a termine con successo sono disastrose: tutti i messaggi, futuri e passati, cifrati con una chiave non segreta sono compromessi.

È comune che la chiave di cifratura per una comunicazione cambi con una certa frequenza per rendere il sistema meno vulnerabile ad attacchi a forza bruta.

2 Cifrari simmetrici e asimmetrici

2.1 Cifratura a chiave simmetrica

La cifratura a chiave simmetrica, nota anche come crittografia a chiave singola, o tradizionale, utilizza la stessa chiave (k) sia per cifrare che per decifrare i messaggi (P) per cui si ha $P = D_k(E_k(P))$.

Per rendere sicuro l'utilizzo della crittografia tradizionale è desiderabile che un estraneo, pur conoscendo l'algoritmo di cifratura, ed avendo accesso ad uno o più testi cifrati, non sia in grado di comprenderne il significato né di scoprire informazioni sulla chiave utilizzata. Inoltre, è richiesto che sia il mittente che il destinatario vengano in possesso della chiave in maniera sicura e la custodiscano segretamente. Queste caratteristiche rendono la crittografia simmetrica adatta agli usi più disparati, limitando i problemi di sicurezza principali allo scambio e alla custodia delle chiavi. [Sta17]
La chiave solitamente viene distribuita prima della comunicazione tra le due entità, scambiata di persona, con un corriere fidato, o tramite un canale cifrato. Quest'ultima soluzione sposta però i problemi di sicurezza sullo scambio delle chiavi relative al canale, mentre le altre possono risultare poco pratiche. [Wikb]
La scelta delle chiavi gioca un ruolo fondamentale per la sicurezza di un algoritmo, in quanto una chiave debole vanifica lo scopo della cifratura, esponendolo ad attacchi a forza bruta.

2.2 Cifratura a chiave asimmetrica

La cifratura a chiave asimmetrica, detta anche crittografia a chiave pubblica, rappresenta forse la più importante rivoluzione nella storia della crittografia [KMS11], e viene usata per risolvere il problema della distribuzione della chiave attraverso servizi specifici.

I server di distribuzione delle chiavi generano, per ogni individuo, due chiavi: una pubblica e una privata. La chiave pubblica serve per cifrare, ed è resa nota dall'individuo a chiunque voglia inviargli un messaggio. La chiave privata è nota soltanto all'individuo, che la custodisce, ed è usata per decifrare il messaggio [Wikb]. Gli algoritmi basati sulla cifratura asimmetrica sfruttano una relazione matematica che lega chiave pubblica e privata piuttosto che basarsi sui principi di sostituzione e permutazione [KMS11]. I dati cifrati con una chiave pubblica k_e possono essere decifrati solamente dalla corrispondente chiave privata k_d: $P = D_{k_d}(E_{k_e}(P))$.

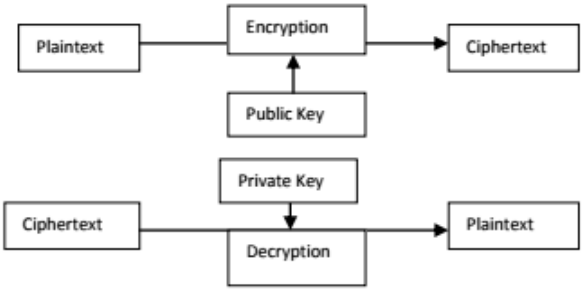

Fig. 2: Modello crittografico a chiave asimmetrica
[TK11]

Gli algoritmi a chiave asimmetrica sono più lenti in fase di esecuzione ma forniscono un livello di sicurezza più alto, rispetto a quelli a chiave simmetrica, per via della dimensione delle chiavi. Infatti, per ottenere una discreta sicurezza vengono utilizzate chiavi fino a 4096 bit, che rendono la fattorizzazione di interi molto costosa.
Nel caso dell'algoritmo *RSA* è consigliabile utilizzare chiavi di almeno 2048 bit, perchè come mostrato da Shamir nel 2003 [ST03] è possibile fattorizzare un intero di 1024 bit, con un computer dedicato dal costo di un milione di dollari, in un solo anno.

3 Cifrari a blocchi

Un cifrario a blocchi può presentarsi soltanto nel caso in cui i due utenti condividano una chiave di cifratura simmetrica. Il plaintext viene suddiviso in una serie di blocchi a dimensione prestabilita da cui si ricava il ciphertext. Negli algoritmi ad oggi più diffusi, un blocco è di lunghezza pari a 64 o 128 bit.
In generale, la cifratura a blocchi sembra applicabile a un più ampio ventaglio di applicazioni rispetto alla cifratura a flusso. La maggior parte delle applicazioni crittografiche simmetriche basate su rete fa uso di cifrari a blocchi [Sta17].

3.1 Modalità di cifratura

Di seguito riportiamo le quattro modalità standard definite dall' ISO/IEC [10117] per i cifrari a blocchi.

3.1.1 ECB (Electronic Code Book)

In questa modalità ciascun blocco di dati è cifrato indipendentemente tramite la funzione di cifratura E_k. Si suppone inoltre che la lunghezza del messaggio sia multiplo della dimensione dei singoli blocchi.
Denotato con P il plaintext, questo è spezzato in moduli più piccoli $P = [P_1, P_2, \ldots, P_n]$ e il testo cifrato C sarà così costituito da $C = [C_1, C_2, \ldots, C_n]$, dove $C_j = E_k(P_j)$ è la cifratura di P_j utilizzando la chiave k [TW02].
Questo implica che quando i dati verranno trasmessi su una rete o una linea telefonica, gli errori di trasmissione riguarderanno soltanto i blocchi in cui sono contenuti. Inoltre, i blocchi possono essere riorganizzati ed inviati indipendentemente, purché siano contrassegnati. ECB è la più debole tra le varie modalità perché non sono presenti misure di sicurezza aggiuntive, tuttavia è la più veloce e la più facile da implementare [TK11].

La modalità ECB è deterministica, se un testo in chiaro viene cifrato due volte con la stessa chiave il risultato sarà lo stesso [Mao04].

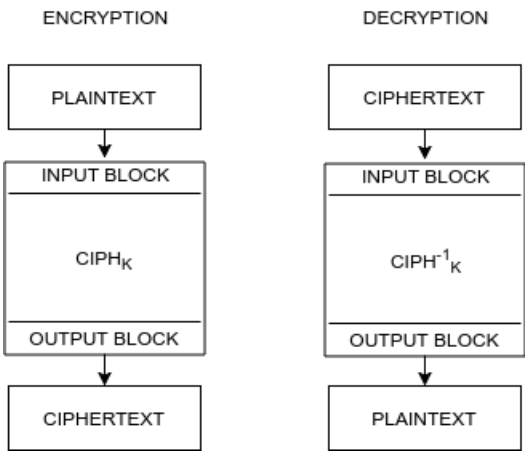

Fig. 3: Modalità ECB

3.1.2 CBC (Cipher Block Chaining)

In questa modalità di operazioni, ciascun blocco del plaintext cifrato viene sommato utilizzando l'operatore logico XOR con il blocco del prossimo plaintext che dovrà essere cifrato, perciò tutti i blocchi dipendono dai blocchi precedenti. Questo significa che per ricavare il testo in chiaro di un particolare blocco, bisogna conoscere il blocco di testo cifrato corrispondente, la chiave, e il testo cifrato del blocco precedente. Il primo blocco, non avendo un precedente testo cifrato, viene sommato mediante l'operatore logico XOR con un numero chiamato vettore inizializzato (IV, "initial vector"). Se si verifica un errore di trasmissione su di un blocco, si avranno conseguenze su tutti blocchi successivi. Se invece alcuni bit di un blocco vengono modificati da terzi durante il percorso è facile intuire la manomissione dato che la modifica sarà riscontrata soltanto sul singolo blocco. L'operatore XOR in ogni step serve quindi ad aggiungere un ulteriore livello di sicurezza [TK11].

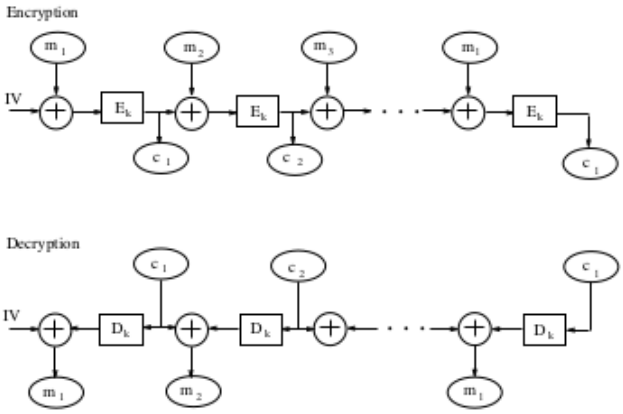

Fig. 4: Modalità CBC
[PHS03]

3.1.3 CFB (Cipher Feedback)

In questa modalità, è possibile cifrare un messaggio anche se la sua lunghezza (L_P) non è un multiplo della lunghezza dei blocchi (M): $L_P = k \cdot M + m$, con $M, m, k \in \mathbb{N}_{>0}$, rispettivamente dimensione in bit di un blocco, bit in eccesso rispetto all'ultimo blocco utilizzato, numero di blocchi utilizzati, e con $m < M$

Il testo in chiaro non è processato dall'algoritmo, bensì sommato tramite l'operatore XOR con un blocco preso in output dall'algoritmo stesso: un blocco detto "Shift Register", inizializzato con valori arbitrari, viene utilizzato come plaintext in input all'algoritmo di cifratura. Il ciphertext ottenuto passa poi attraverso una componente extra chiamata "M-Box", la quale seleziona gli M bit più a sinistra del testo cifrato. Questo valore viene sommato in XOR con il messaggio originale in chiaro, e il risultato costituisce il ciphertext finale. Infine, il testo cifrato viene reimmesso nello Shift Register, e usato come seme del testo in chiaro per il blocco da cifrare successivo.

Come nella modalità CBC, un errore in un blocco durante la trasmissione dei dati colpisce tutti i blocchi successivi. Questo modo di operare è più lento dell'EBC [TK11].

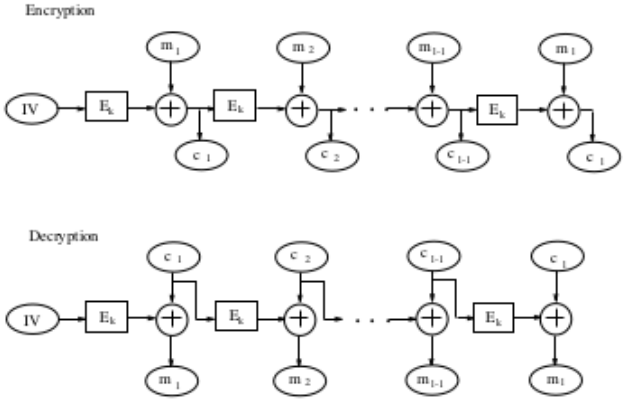

Fig. 5: Modalità CFB
[PHS03]

3.1.4 OFB (Output Feedback)

Questa modalità è simile alla CFB, tranne che il ciphertext ottenuto dall'algoritmo viene reimmesso nello Shift Register piuttosto che essere il risultato finale. Questo valore viene quindi sommato in XOR col plaintext originale per ottenere il ciphertext finale. A differenza di CFB e CBC, un errore di trasmissione in un blocco non avrà effetto sui blocchi successivi perchè, una volta che il ricevente sarà in possesso del valore iniziale dello Shift Register, potrà continuare a generare i successivi input. Tuttavia, questa modalità è meno sicura rispetto alla CFB perchè sono necessari soltanto il ciphertext iniziale e il ciphertext in uscita dall'algoritmo per trovare il plaintext dell'ultimo blocco. Non è richiesta conoscenza della chiave [TK11].

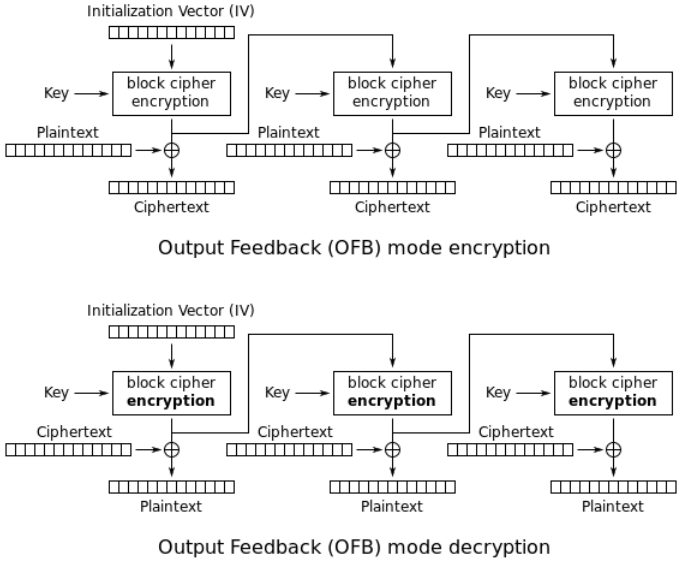

Fig. 6: Modalità OFB

4 Sicurezza di un algoritmo di cifratura

L'obiettivo nella progettazione di un crittosistema è quello di rendere veloci le operazioni di cifratura e decifratura, assicurando allo stesso tempo che la crittanalisi sia complessa. Un sistema che è sicuro dal punto di vista del costo computazionale necessario alla crittanalisi, ma che cede ad un attacco con potenza di calcolo illimitato, è chiamato computazionalmente sicuro. Mentre un sistema che può resistere alla crittanalisi, non importa quanta potenza computazionale viene utilizzata, è chiamato incondizionatamente sicuro [Sha49].

Non esiste un algoritmo di cifratura incondizionatamente sicuro, perciò, tutti gli utenti possono battersi per avere un algoritmo che rispetta uno o entrambi dei seguenti criteri: [Sta17]

- il costo di rottura del cifrario supera il valore dell'informazione cifrata
- il tempo richiesto per violare il cifrario supera il tempo di vita dell'informazione

Uno schema di cifratura è considerato sicuro dal punto di vista computazionale se uno dei due criteri precedenti viene soddisfatto [Sta17]. La forza dell'algoritmo è perfetta se non esiste un modo migliore per rompere il crittosistema che provare ogni possibile chiave in un attacco a forza bruta [Sch95].

I parametri che rendono sicuri un algoritmo riguardano la chiave utilizzata, e le operazioni di cui è composto.
La chiave utilizzata deve essere talmente grande (lunghezza n) che un attacco a forza bruta, applicabile su ogni algoritmo di cifratura, risulti impraticabile, ovvero il tempo necessario ad esaminare lo spazio delle chiavi (2^n) sarebbe troppo elevato per un calcolatore.

Una proprietà auspicabile di un algoritmo di cifratura è la presenza del cosiddetto "effetto valanga" [Sta17]: applicando un cambiamento seppur minimo all'interno del plaintext o della chiave si produce un ciphertext significativamente diverso. Se un cifrario a blocchi non mostra l'effetto valanga in maniera significativa, un crittoanalista è in grado di fare delle predizioni sull'input esaminando l'output. Questo basta per violare l'algoritmo parzialmente o completamente. Perciò costruire un cifrario che mostri un considerevole effetto valanga è uno dei principali obiettivi in fase di progettazione [Sch94].

Un altro parametro di sicurezza per un algoritmo è dato dal *coefficiente di correlazione*, una misura numerica, su come due variabili incidono l'una sull'altra [AAM13]. Il valore della coefficiente di correlazione serve a quantificare il livello di confusione presente nel ciphertext di un cifrario a blocchi ed è rappresentato da un numero reale compreso tra -1 e +1. I valori al limite indicano una forte correlazione lineare tra le variabili, mentre un valore nullo rende più difficile i tentativi di eseguire una crittoanalisi lineare o differenziale, che tratteremo in seguito in questo capitolo.

Gli attacchi a cui possono essere soggetti gli algoritmi crittografici a chiave privata vengono classificati in base alle conoscenze possedute dal crittoanalista, e sono [Sta17], come indicati in figura 7:

- Attacco con testo cifrato noto: il crittoanalista conosce solamente alcuni crittogrammi $E_k(m_1), E_k(m_2), ..., E_k(m_l)$ ed è intento a scoprire sia la chiave k o uno o più messaggi m_i per qualunque $i = 1, \ldots, l$. Questo attacco si può verificare se il crittoanalista riesce a intercettare il messaggio in transito sul canale di comunicazione, pratica nota anche come "attacco man-in-the-middle".

- Attacco con testo in chiaro noto: l'attaccante ha accesso ad una collezione di coppie $(m_i, E_k(m_i))|i = 1, \ldots, l$ e intende determinare la chiave k o decifrare i crittogrammi successivi $E_k(m_{l+1})$ non inclusi nella collezione. L'antagonista, anche per questo attacco, dovrebbe essere in grado di intercettare il canale di comunicazione, ma potrebbe dover accedere soltanto ad una parte del testo in chiaro.

- Attacco con testo in chiaro scelto: questo è un attacco in cui all'attaccante è noto un preciso testo in chiaro per cui il crittoanalista è in grado di leggere i crittogrammi corrispondenti. Questo scenario si può verificare quando gli strumenti di cifratura sono sprovvisti di supervisione per un certo lasso di tempo.

- Attacco con testo cifrato scelto: il crittoanalista è in grado di selezionare alcuni crittogrammi e osservare i corrispettivi messaggi in chiaro. Lo scopo in questo tipo di attacco è quello di trovare la chiave segreta oppure cifrare un nuovo messaggio in un crittogramma valido. Similmente agli attacchi con testo in chiaro, questo attacco può verificarsi se l'attrezzatura per decifrare viene lasciata incustodita.

Type of Attack	Known to Cryptanalyst
Ciphertext Only	■ Encryption algorithm ■ Ciphertext
Known Plaintext	■ Encryption algorithm ■ Ciphertext ■ One or more plaintext–ciphertext pairs formed with the secret key
Chosen Plaintext	■ Encryption algorithm ■ Ciphertext ■ Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key
Chosen Ciphertext	■ Encryption algorithm ■ Ciphertext ■ Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key

Fig. 7: Tipi di attacchi
[Sta17]

Di seguito illustriamo alcuni dei metodi crittoanalitici più diffusi per i cifrari a chiave simmetrica.

Crittanalisi differenziale Biham e Shamir [BS11] hanno diffuso la crittoanalisi differenziale nel 1990, anche se pare che fosse già nota ai progettisti di IBM e NSA che lavorarono alla creazione del DES nel 1974 [BL16].

L'analisi differenziale può essere applicata a crittosistemi basati su reti di Feistel con r round in cui si conoscono gli input della funzione di round F e si possono dedurre, o prevedere con alta probabilità, gli output corrispondenti.

Assumiamo di avere una S-box che trasforma le stringhe in input secondo la funzione di round $F : \Sigma^n \to \Sigma^m$
Data una coppia di stringhe in input (s_1, s_2), la S-box fornisce in output $s_1^* = F(s_1)$ e $s_2^* = F(s_2)$. Ogni coppia di tuple input/output $(s_1, s_1^*), (s_2, s_2^*)$ è caratterizzata dalle differenze: $\delta = s_1 \oplus s_2$ e $\Delta = s_1^* \oplus s_2^*$, da cui il nome differenziale. Per cui è possibile definire l'insieme $S_\Delta^\delta = \{(s_1, s_2; s_1^*, s_2^*) | s_1 \oplus s_2 = \delta \wedge s_1^* \oplus s_2^* = \Delta\}$ costituito dalle tuple con differenze fissate.

L'idea è quella di confrontare le differenze per coppie adeguatamente scelte di plaintext e ciphertext, per dedurre informazioni sulla chiave. A questo scopo vengono costruiti dei cosiddetti "profili XOR" di una S-box. [TW02]

Il profilo XOR di una S-box, relativo alla funzione F, è una tabella i cui indici di riga sono le 2^n possibili differenze δ e gli indici di colonna le 2^m differenze Δ. Ogni elemento corrisponde alla cardinalità dell'insieme S_Δ^δ, che è sempre un numero pari, e le cui somme su ogni riga ammonteranno a 2^n. Una differenza δ in input può provocare una differenza in output Δ con probabilità $P = \alpha/2^n$, in cui α indica il numero di coppie in input che determinano la differenza Δ in output.
L'attacco differenziale si basa sulla proprietà dello XOR per cui $(s_1 \oplus k) \oplus (s_2 \oplus k) = s_1 \oplus s_2$, che permette di ignorare la chiave k utilizzata.

Supponendo di poter costruire una tabella completa per il profilo XOR, dalla conoscenza dei valori δ e Δ è possibile risalire ai valori di input (s_1, s_2).
Se invece sono conosciuti s_1, s_2 e Δ, i valori $(s_1 \oplus k)$ e $(s_2 \oplus k)$ compaiono in S_Δ^δ, con $\delta = s_1 \oplus s_2$. Sia $\Xi = \{s_{i_1}, s_{i_2}, ..., s_{i_j}\}$ con $j = |S_\Delta^\delta|$ l'insieme dei possibili input per ogni blocco.
Allora la chiave k dovrà essere contenuta in $K = \Xi \oplus s_1 = \Xi \oplus s_2 = \{s_{i_1} \oplus s_1, ..., s_{i_j} \oplus s_1\} = \{s_{i_1} \oplus s_2, ..., s_{i_j} \oplus s_2\}$ [PHS03].

Crittanalisi lineare Nel 1993 Matsui ha presentato una classe di attacchi che sfrutta una scarsa nonlinearità delle S-Box. L'attacco, menzionato come crittoanalisi lineare, è un attacco con known plaintext. La crittoanalisi può anche funzionare come un attacco a known ciphertext [PHS03].

Supporre che sia possibile trovare una relazione probabilistica di tipo lineare, tra un sottoinsieme dei bit di un plaintext ed il loro stato prima delle operazioni di sostituzione nell'ultimo round della rete Feistel, equivale a supporre l'esistenza di un sottoinsieme di bit per cui l'ultima operazione di XOR produce un certo risultato con probabilità vicina ai valori 0 o 1.

Assumendo che il crittoanalista abbia a disposizione numerose coppie di plaintext-ciphertext, tutte cifrate con la stessa chiave k, è possibile cominciare a decifrare il ciphertext provando tutte le chiavi possibili per l'ultima iterazione della rete di Feistel. Per ogni chiave si confrontano i valori dei bit coinvolti in una possibile relazione lineare e si determina se questa è effettivamente presente. Tutte le volte che questo succede viene incrementato un indice corrispondente alla chiave esaminata. Al termine del procedimento, è possibile verificare

le assunzioni fatte sulla presenza di una relazione lineare tra chiave e ciphertext controllando che l'indice relativo alla chiave esaminata abbia una frequenza superiore alla metà del numero di sequenze di bit esaminate. [Sti05]

Attacco a collisione L'attacco del compleanno, *birthday attack*, è un attacco di tipo known ciphertext, così chiamato perché sfrutta i princìpi matematici alla base del paradosso del compleanno nella teoria delle probabilità.
Nel paradosso del compleanno si vuole determinare la probabilità $P(n)$ che in un gruppo di n persone, almeno due individui siano nati nello stesso giorno, non considerando gli anni bisestili.
Si ricava la probabilità per l'evento complementare $\bar{P}$, cioè che tutti gli n compleanni si festeggino in giorni distinti: $\bar{P} = \prod_{i=1}^{n}(1 - \frac{i-1}{365})$
Si ricava $P = 1 - \bar{P}$, il cui valore è di circa 0.5 per $n = 23$.

L'attacco del compleanno generalmente è applicato alle funzioni di hashing, ma può essere esteso anche ai cifrari a blocchi in modalità CBC [BL16]:
sia associata all'algoritmo di cifratura con chiave k una funzione f, lo scopo dell'attacco è quello di trovare due blocchi x_i e x_j, con $i \neq j$, tali che $f(x_i) = f(x_j)$, cioè per cui f non è iniettiva. Quando questo si verifica si ha una collisione.
Per trovare una collisione bisognerebbe confrontare i crittogrammi ottenuti per diversi blocchi di n bit in input all'algoritmo. Sotto l'ipotesi che la funzione di cifratura restituisca 2^n valori equiprobabili, trovare una collisione dopo aver valutato la funzione per un numero di blocchi pari a $1.25 \cdot \sqrt{2^n}$ ha probabilità pari a 0.5 [Sta17]. Nel caso $n = 64$ bit questo numero è vicino a 2^{32}, motivo per cui in generale è ritenuto poco sicuro utilizzare la stessa chiave per cifrare più di $2^{\frac{n}{2}}$ blocchi, appartenenti ad uno o più messaggi. [BL16]

Avendo trovato una collisione è possibile, anche per Blowfish, cifrando in modalità CBC [Figura 4], applicando la cifratura (E_k) ad ogni blocco m_i, ottenere un Initial Value c_i che viene usato per la cifratura del blocco m_{i+1}:
$E_k(m_i \oplus c_{i-1}) = c_i$
Dato che E_k consiste di operazioni di permutazione e sostituzione, ottenere una collisione in output implica che gli input siano gli stessi $(m_i \oplus c_{i-1}) = (m_j \oplus c_{j-1})$, da cui lo XOR tra due blocchi del plaintext: $(m_i \oplus m_j) = (c_{i-1} \oplus c_{j-1})$.
In molti contesti, risalire soltanto allo XOR tra due blocchi di plaintext non è sufficiente per un attacco efficace. Tuttavia, secondo Bhargavan e Leurent [BL16], è realizzabile un attacco di successo quando si verifichino le seguenti condizioni:

- la stessa informazione viene inviata più volte
- una parte del plaintext è nota

In questo caso, c'è una possibilità che una collisione porti allo XOR tra il blocco relativo all'informazione e il plaintext conosciuto, svelando l'informazione segreta. Questo attacco ha successo con alta probabilità dopo aver cifrato 2^s copie dell'informazione e 2^t blocchi conosciuti sono cifrati, con $s + t \geq n$.

5 L'algoritmo Blowfish

L'algoritmo di cifratura Blowfish è stato progettato da Bruce Schneier nel 1993, si tratta di un cifrario simmetrico a blocchi nato dalla necessità di un nuovo standard a livello mondiale a seguito delle vulnerabilità riscontrate per il DES, in circolazione da più di 15 anni. [BS11] [Mat94] [Wie93]

Molti degli algoritmi privi di crittoanalisi nota, erano coperti da brevetti o le specifiche non erano completamente note, per cui non erano utilizzabili liberamente. Inoltre, il governo Statunitense si stava indirizzando verso l'utilizzo di algoritmi secretati.[ST93]

Nella visione di Schneier [Sch94] un algoritmo di cifratura standard deve essere adeguato per varie applicazioni:

- Crittografia di massa: l'algoritmo deve essere efficiente per cifrare files o continui flussi di dati.
- Generazione casuale di bit: l'algoritmo deve essere efficiente nella produzione di serie casuali di bit.
- Cifrare pacchetti: l'algoritmo deve essere efficiente nel cifrare i dati a misura di pacchetto, tale da essere implementabile in casi in cui i diversi pacchetti debbano essere cifrati o decifrati con chiavi diverse.
- Hashing: L'algoritmo deve essere efficiente per essere convertito in una funzione di hash unidirezionale.

Blowfish si basa sulla rete di Feistel, un cifrario costituito di più iterazioni, ed una funzione interna detta "round function" [MCV01] come mostrato in figura 8.

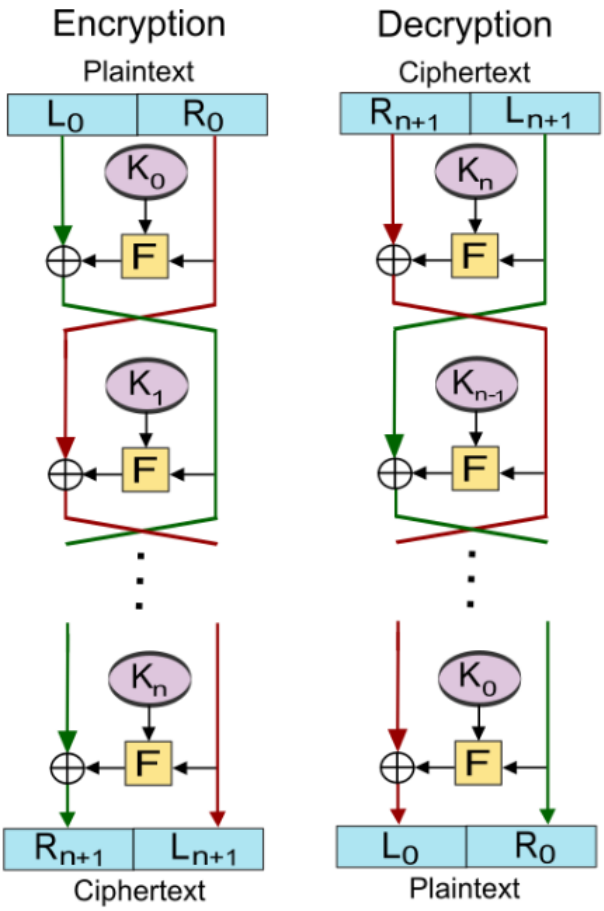

Fig. 8: Rete di Feistel
[Wika]

La chiave di cifratura può avere lunghezza variabile da 64 bit fino a un massimo di 448 bit, con valori multipli di 64, questo rende Blowfish adatto per applicazioni in cui la chiave non cambia frequentemente, come ad esempio canali di comunicazione o cifratori automatici di file [Sch94].

5.1 Descrizione dell'algoritmo

L'algoritmo consiste in due parti: espansione della chiave e cifratura dei dati. L'espansione della chiave converte la chiave della cifratura, che ha al massimo 448 bit, in numerosi array di sottochiavi per un totale di 4168 byte (33344 bit). La cifratura dei dati avviene tramite 16 iterazioni della rete di Feistel.

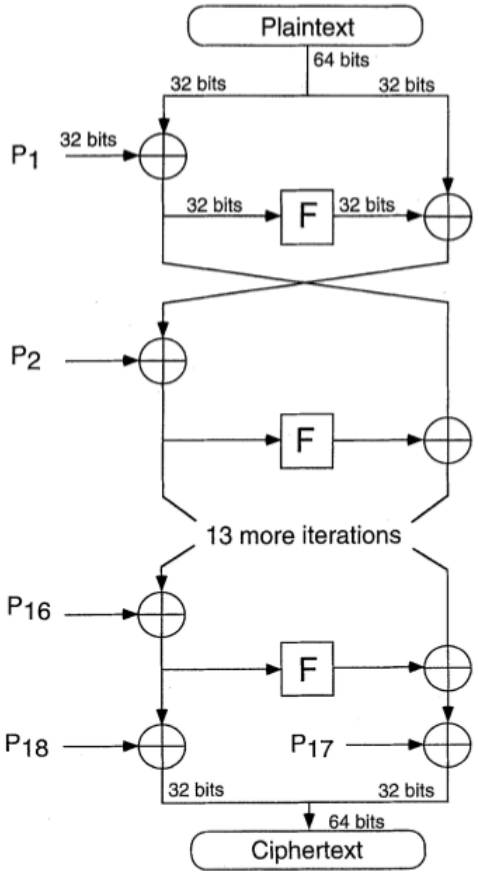

Fig. 9: Diagramma a blocco di Blowfish
[Sch94]

Ciascun round consiste in una permutazione dipendente dalla chiave, una sottochiave di round e una sostituzione dipendente dai dati. Le operazioni matematiche effettuate sono XOR e somme di word da 32 bit. Le uniche operazioni aggiuntive consistono in quattro ricerche su array (S-box) per ogni round e uno scambio finale; tutte queste operazioni sono efficienti sui microprocessori, e permettono di utilizzare chiavi precalcolate e conservate in memoria cache (per un totale di 4 kB) per aumentare la velocità di esecuzione dell'algoritmo. [Sch94]

La scelta di adoperare questo tipo di cifrario risiede nel fatto che le reti di Feistel sono semplici da comprendere e facilitano l'analisi dell'algoritmo, aumentando la fiducia nello stesso. [Smi71]

Le S-box sono variabili, dipendenti dalla chiave, questo rende l'algoritmo più difficile da rompere tramite crittoanalisi lineare e differenziale.

5.2 Sottochiavi

Blowfish usa numerose sottochiavi da 32 bit che vengono inizializzate prima di cominciare a decifrare o cifrare un messaggio, dopo l'espansione della chiave iniziale, e vengono poi usate

ad ogni round. Le sottochiavi sono contenute in:

- un "P-array", da *permutation array*, consiste di 18 sottochiavi da 32 bit: P_1, P_2, ..., P_{18}
- quattro "S-box", da *substitution box*, da 256 elementi da 32 bit ciascuna:
  $S_{1,0}$, $S_{1,1}$, ..., $S_{1,255}$
  $S_{2,0}$, $S_{2,1}$, ..., $S_{2,255}$
  $S_{2,0}$, $S_{2,1}$, ..., $S_{2,255}$
  $S_{3,0}$, $S_{3,1}$, ..., $S_{3,255}$

5.3 Cifratura

L'input di un blocco, da 64 bit, verrà di seguito indicato con x, mentre con F verrà indicata la round function. L'algoritmo per la cifratura è il seguente:

Dividere x in due parti da 32 bit: x_L, x_R
For i = 1 to 16
$x_L = x_L \oplus P_i$
$x_R = F(x_L) \oplus x_R$
scambiare x_L e x_R
scambiare x_L e x_R
$x_R = x_R \oplus P_{17}$
$x_L = x_L \oplus P_{18}$
Unire nuovamente x_L e x_R

La funzione F divide x_L in quattro parti da 8 bit: a, b, c, d: [Figura 10]
$F(x_L) = ((((S_{1,a} + S_{2,b}) mod 2^{32}) \oplus S_{3,c}) + S_{4,d}) mod 2^{32}$

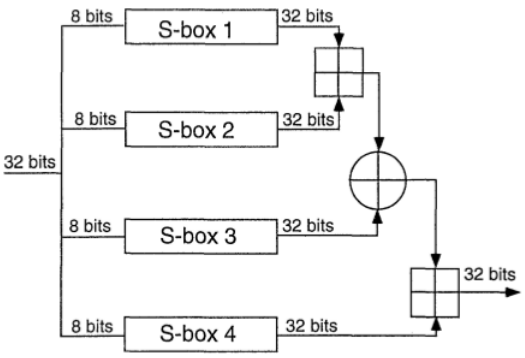

Fig. 10: Funzione F di Blowfish
[Sch94]

5.4 Decifratura

Il processo di decifratura avviene percorrendo gli stessi passi dell'algoritmo di cifratura, ma invertendo l'ordine delle sottochiavi P_1, P_2, ..., P_{18}

5.5 Generazione delle sottochiavi

Le sottochiavi vengono generate usando l'algoritmo di cifratura stesso, come indicato:

1. Gli elementi del P-array e delle quattro S-box sono inizializzati con una stringa prestabilita, consistente della rappresentazione esadecimale delle cifre decimali di π.
2. Eseguire XOR tra i P_i e 32 bit della chiave per volta.
3. Cifrare una stringa "nulla" con le sottochiavi ottenute nei passaggi (1) e (2).
4. Sostituire P_1 e P_2 con l'output del passo (3).
5. Cifrare l'output del passo (3) usando l'algoritmo Blowfish con le sottochiavi modificate.
6. Sostituire P_3 e P_4 con l'output del passo (5).
7. Continuare il processo, fino a modificare tutti gli elementi del P-array e delle S-box in ordine.

In totale, nel processo di generazione delle sottochiavi, sono necessarie 521 iterazioni. Le sottochiavi generate possono essere memorizzate in una memoria cache piuttosto che derivate dall'inizio al riutilizzo dell'algoritmo.

6 Criptanalisi di Blowfish

Di seguito descriviamo le caratteristiche che rendono Blowfish un buon algoritmo di cifratura ed alcuni tentativi di crittanalisi effettuati nel corso degli anni da diversi studiosi. Premettiamo che, ad oggi, non è nota l'esistenza di una crittoanalisi per la versione completa dell'algoritmo, con rete di Feistel a 16 round. Per questo motivo Blowfish viene identificato come computazionalmente sicuro. In fase di progettazione, Schneier aveva considerato la possibilità di modificare il numero di round dell'algoritmo e di utilizzare chiavi piccole per arrivare a rendere possibile l'implementazione in sistemi in cui fosse stato necessario fare un compromesso tra velocità di esecuzione e sicurezza. Aveva inoltre identificato nelle S-box di grandi dimensioni, e dipendenti dalla chiave, un punto di forza contro la crittanalisi lineare e differenziale. [Sch95]

Al momento della pubblicazione sulla rivista *"Dr. Dobb's Journal"* venne indetto un concorso per incentivare gli utenti ad effettuare test dell'algoritmo e la scoperta di eventuali vulnerabilità [Joh03], avvantaggiandosi della sua natura *open source*. In risposta ci furono diverse applicazioni.

John Kelsey riuscì a sviluppare un attacco per rompere una versione a 3 round di Blowfish [KB10], ma non fu in grado di estenderlo alla versione completa. Il suo attacco sfrutta la funzione F ed il fatto che l'addizione bit a bit modulo 2^{32} e lo XOR utilizzati nell'algoritmo Blowfish, non sono commutativi reciprocamente. [Joh03]

Vikramijt Chhabra tentò di trovare un modo per implementare una ricerca a forza bruta efficace della chiave. [Joh03]

Nel 1996 Serge Vaudeney analizzò una versione modificata di Blowfish in cui le S-box erano conosciute e non dipendenti dalla chiave. Per questa variante aveva mostrato [Vau96] che era possibile recuperare il P-array con un attacco differenziale a partire da $2^{8 \cdot r+1}$ plaintext scelti (in cui r indica il numero di round).
Questo tipo di attacco è impossibile per una versione con più di 8 round, in quanto richiederebbe più plaintext di quelli che è possibile generare con un cifrario a blocchi di 64 bit. Lo stesso Vaudeney notò che per alcune chiavi deboli, cioè chiavi che causano una collisione nella generazione delle S-box (la probabilità che questo accada con una scelta casuale delle chiavi è di 1 su 214), lo stesso attacco richiederebbe soltanto $2^{4 \cdot r+1}$ plaintext scelti per recuperare il P-array, sempre supponendo che siano note le S-box generate. Nel caso in cui le S-box non siano conosciute, questo attacco può servire a verificare se la chiave utilizzata per la cifratura è debole, ma non è possibile determinarla, né è possibile determinare le S-box o il P-array. L'attacco proposto è inefficace per versioni complete a 16 round.
Da questi risultati, la ricerca di vulnerabilità di Blowfish si è spostata verso la ricerca delle chiavi deboli, per poterle identificare e scartare.

Vincent Rijmen, nella sua tesi di dottorato di ricerca [Rij97], ha introdotto un attacco differenziale di secondo ordine che può rompere una versione modificata di Blowfish fino a quattro round, non riuscendo ad estenderlo.

Questi risultati non fornirono una crittoanalisi promettente per violare l'algoritmo, quindi l'autore ritenne che Blowfish fosse piuttosto promettente in termini di sicurezza.
In seguito ALabaichi, Ahmad e Mahmod [Ala13], ripercorrendo le ricerche di altri studiosi, hanno misurato il livello di sicurezza di Blowfish con un'analisi dell'effetto a valanga e del coefficiente di correlazione, certificando che fossero soddisfatte le proprietà di diffusione e confusione definite da Shannon.

7 Applicazioni d'utilizzo

Nel corso degli anni l'algoritmo di cifratura Blowfish, è stato utilizzato per vari scopi, come desiderato nella sua fase di progettazione, per via della velocità di esecuzione elevata anche in contesti con potenza di calcolo ridotta. A favorirne la diffusione sono stati anche dei risultati come quelli ottenuti, tra gli altri, da Allam Mousa [Mou05], Masram e i componenti del suo team [Mas+14], che hanno implementato a livello software vari algoritmi di cifratura simmetrici tra cui Blowfish per cifrare file in formati diversi (audio, immagini, video, testo) e analizzarne il tempo di cifratura e decifratura. Questo risulta non variare in base al formato dei dati ma di dipendere solamente dal numero di byte in un file.

Kumar e Baskaran [KB10] hanno implementato Blowfish su circuiti integrati "ASIC" (application specific integrated circuit), pensati per la risoluzione di un unico problema, in questo caso lo sviluppo di un sistema crittografico, ma che consentono di raggiungere delle prestazioni in termini di velocità di processazione e consumo elettrico difficilmente ottenibili con l'uso di soluzioni più generiche. L'implementazione dell'algoritmo direttamente a livello hardware ha comportato diversi vantaggi: velocità di esecuzione superiore rispetto ad una implementazione software e maggiore sicurezza legata alla difficoltà di accedere fisicamente alla chiave.
Inoltre Kumar e Baskaran hanno mostrato che l'implementazione da loro proposta comporta un risparmio energetico di circa l'80% rispetto ad altre soluzioni note. [KB10]

Un altro utilizzo di Blowfish è nelle FPGA (Field Programmable Gate Array). Le FPGA sono caratterizzati da un'elevata scalabilità: permettono di combinare la flessibilità, la facilità di modifica che caratterizza le implementazioni software, le prestazioni, la sicurezza delle soluzioni realizzate in hardware. Infatti, come suggerito dal nome, le funzionalità sono programmabili via software, permettendo così l'implementazione di funzioni logiche anche molto complesse.
Per questi motivi Ahmad e Ismail [AMI16] hanno progettato un miglioramento di Blowfish per un dispositivo Zynq-7000 xc7z020, proponendo di rimpiazzare AES come algoritmo di cifratura standard per la protezione delle comunicazioni senza fili (WiFi e Bluetooth). Gli autori fanno notare che AES causa un dispendio energetico superiore rispetto alla loro soluzione, che mostra inoltre un throughput superiore del 29%.
La loro proposta è quella di ridurre al minimo i requisiti hardware per rendere il design più piccolo possibile, allo stesso tempo l'utilizzo di blocchi da 128 bit, garantisce che l'algoritmo, cifrando due blocchi da 64 bit in parallelo, abbia un throughput più elevato che nella versione tradizionale.

Con lo sviluppo della tecnologia ci ritroviamo ad avere a che fare ogni giorno con oggetti intelligenti, connessi in rete e che comunicano tra loro scambiandosi informazioni sul loro stato ma anche relative ai loro utilizzatori. Da questo nasce l'esigenza di proteggere i dati comunicati tra gli oggetti utilizzando forti algoritmi di cifratura. Seguono questo obiettivo gli studi di Suresh [SN16] che si è occupato di sicurezza nel campo applicativo *Internet of Things*. Suresh ha consigliato l'utilizzo di cifrari a blocchi piuttosto che a flusso, perché più sicuri in quanto più difficili da analizzare.

Nel suo articolo, da un confronto delle prestazioni per vari algoritmi di cifratura simmetrici, si evince che l'algoritmo Blowfish è secondo soltanto ad AES in quanto a velocità di cifratura e decifratura, ma è nettamente più efficiente se si considerano l'utilizzo delle risorse di memoria, effetto valanga, e adattabilità a diversi sistemi fornita dalla dimensione variabile della chiave. Questi fattori rendono Blowfish l'algoritmo più adatto alla trasmissione sicura dei dati nel campo IoT.

8 Conclusioni

L'algoritmo di cifratura da utilizzare va scelto in base alle specifiche situazioni: è sempre necessario fare un compromesso tra velocità e sicurezza. Un algoritmo con una chiave più lunga, o con più iterazioni, è più sicuro dal punto di vista della resistenza alla crittoanalisi, ma richiede maggiori risorse per essere eseguito.

Dall'analisi condotta da Patil e dal suo team [Pat+16], Blowfish risulta veloce sia per la cifratura che per la decifratura delle informazioni, nonostante una fase di generazione delle sottochiavi, che porta alla memorizzazione di circa 4 kB di dati. Risulta più lento rispetto al DES, che tuttavia è stato violato e dimostrato insicuro, ma più veloce di AES, scelto dal National Institute of Standards and Technology (NIST) come algoritmo per la cifratura di dati, riservati e non, da parte del Governo degli Stati Uniti [PT01].
A differenza di Blowfish, AES ammette chiavi a 128, 192 o 256 bit e blocchi da 128 bit, con un numero di iterazioni di 10, 12, 14, a seconda della dimensione della chiave scelta.

Recentemente sta perdendo credito come algoritmo di cifratura generale, per via dei calcolatori sempre più performanti, per cui non è più un problema sacrificare qualche millisecondo in fase di cifratura a scapito della sicurezza, mentre in altri casi l'implementazione software di AES risulta maggiormente ottimizzata su macchine con processori di ultima generazione, annullando i vantaggi di Blowfish [EKK14] [Pan16] [Mas+14]. Da qui la diffusione e l'affermazione di AES come algoritmo crittografico standard moderno.
Minaam [MAH10] così come Allam Mousa [Mou05] hanno confrontato il consumo energetico richiesto da differenti algoritmi simmetrici tra cui Blowfish e AES nei sistemi integrati e nei Personal Computer, concludendo che l'algoritmo Blowfish supera in prestazioni gli altri algoritmi riguardo potenza di consumo, ed anche per tempo di elaborazione quando implementato su sistemi integrati [Sil+16].

Blowfish rimane quindi l'algoritmo di riferimento quando si tratta di effettuare un' implementazione a livello hardware grazie ai consumi ridotti, alla sua resistenza alla crittoanalisi (non è ad oggi ancora nota una crittoanalisi sull'algoritmo completo), la modificabilità, e la velocità ottenibile [Pat+16].
La sua implementazione a livello software è consigliata nei casi in cui la priorità sia la velocità di cifratura piuttosto che la sicurezza dell'algoritmo, che è possibile compensare cambiando frequentemente la chiave di cifratura.

A fini di completezza facciamo presente che Bruce Schneier, il progettista di Blowfish, consiglia l'utilizzo di Twofish piuttosto che Blowfish, un altro algoritmo progettato da lui nel 1998 [Sch98], che è stato poi scelto tra i cinque finalisti nella gara indetta dal NIST[ST97] e che ha portato all'affermazione di AES.
Come Blowfish, anche Twofish è un algoritmo di natura *open source* e si basa su reti di Feistel con 16 iterazioni. Come AES, la dimensione della chiave è variabile a 128, 192 o 256 bit, e la dimensione dei blocchi è di 128 bit. Rispetto al suo predecessore però, vanta l'assenza di chiavi deboli, più numerosi tentativi di crittoanalisi che ne hanno provato la robustezza, e maggiore velocità nella cifratura, con risultati che migliorano ulteriormente con l'incremento della memoria RAM a disposizione [Ran16].

Bibliografia

[10117] ISO/IEC 10116. *Information technology, Security techniques, Modes of operation for an n bit block cipher*. 2017.

[AMI16] Rafidah Ahmad, Asrulnizam Abd. Manaf, and Widad Ismail. "Development of an improved power-throughput Blowfish algorithm on FPGA". In: *2016 IEEE 12th International Colloquium on Signal Processing & Its Applications (CSPA)* (2016). DOI: 10.1109/cspa.2016.7515838.

[AAM13] Ashwak Alabaichi, Faudziah Ahmad, and Ramlan Mahmod. "Security analysis of blowfish algorithm". In: *2013 Second International Conference on Informatics Applications (ICIA)* (2013). DOI: 10.1109/icoia.2013.6650222.

[Ala13] Ashwaq Alabaichi. "Randomness Analysis of 128 bits Blowfish Block Cipher on ECB and CBC Modes". In: 15 (Jan. 2013).

[BL16] Karthikeyan Bhargavan and Gaëtan Leurent. "On the Practical (In-)Security of 64-bit Block Ciphers: Collision Attacks on HTTP over TLS and OpenVPN". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS '16. Vienna, Austria: ACM, 2016, pp. 456–467. ISBN: 978-1-4503-4139-4. DOI: 10.1145/2976749.2978423. URL: http://doi.acm.org/10.1145/2976749.2978423.

[BS11] Eli Biham and Adi Shamir. *Differential cryptanalysis of the data encryption standard*. Springer-Verlag, 2011.

[EKK14] Mansoor Ebrahim, Shujaat Khan, and UmerBin Khalid. "Symmetric Algorithm Survey: A Comparative Analysis". In: *CoRR* abs/1405.0398 (2014). arXiv: 1405.0398. URL: http://arxiv.org/abs/1405.0398.

[IETa] IETF. *Internet Security Glossary, Version 2*. URL: https://tools.ietf.org/html/rfc4949.

[IETb] IETF. *RFC 2828, Internet Security Glossary*. URL: https://www.ietf.org/rfc/rfc2828.txt.

[ITU] ITU-T. *X.800 : Security architecture for Open Systems Interconnection for CCITT applications*. URL: https://www.itu.int/rec/T-REC-X.800-199103-I/en.

[Joh03] William M. Hancock John W. Rittinghouse. *Cybersecurity Operations Handbook*. Digital Press, 2003.

[KB10] P. Karthigai Kumar and K. Baskaran. "An ASIC implementation of low power and high throughput blowfish crypto algorithm". In: *Microelectronics Journal* 41.6 (2010), pp. 347–355. DOI: 10.1016/j.mejo.2010.04.004.

[KMS11] Yogesh Kumar, Rajiv Munjal, and Harsh Sharma. "Comparison of Symmetric and Asymmetric Cryptography with Existing Vulnerabilities and Countermeasures". In: 11 (Oct. 2011).

[Mao04] W. Mao. *Modern Cryptography: Theory and Practice*. HP Professional Series. Prentice Hall PTR, 2004. ISBN: 9780130669438. URL: https://books.google.it/books?id=H42WQgAACAAJ.

[Mas+14] Ranjeet Masram et al. “Dynamic Selection of Symmetric Key Cryptographic Algorithms for Securing Data Based on Various Parameters”. In: *CoRR* abs/1406.6221 (2014). arXiv: 1406.6221. URL: http://arxiv.org/abs/1406.6221.

[Mat94] Mitsuru Matsui. “Linear Cryptanalysis Method for DES Cipher”. In: *Advances in Cryptology — EUROCRYPT '93*. Ed. by Tor Helleseth. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 386–397. ISBN: 978-3-540-48285-7.

[MCV01] A. J. Menezes, Van Oorschot Paul C., and Scott A. Vanstone. *Handbook of applied cryptography*. CRC, 2001.

[MAH10] Diaa Salama Abdul Minaam, Hatem Mohamed Abdual-Kader, and Mohiy Mohamed Hadhoud. “Evaluating the Effects of Symmetric Cryptography Algorithms on Power Consumption for Different Data Types”. In: *I. J. Network Security* 11.2 (2010), pp. 78–87. URL: http://ijns.femto.com.tw/contents/ijns-v11-n2/ijns-2010-v11-n2-p78-87.pdf.

[Mou05] A. Mousa. “Data encryption performance based on Blowfish”. In: *47th International Symposium ELMAR, 2005.* June 2005, pp. 131–134. DOI: 10.1109/ELMAR.2005.193660.

[Pan16] Madhumita Panda. “Performance analysis of encryption algorithms for security”. In: *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)* (2016). DOI: 10.1109/scopes.2016.7955835.

[Pat+16] Priyadarshini Patil et al. “A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish”. In: *Procedia Computer Science* 78 (2016), pp. 617–624. DOI: 10.1016/j.procs.2016.02.108.

[PHS03] Josef Pieprzyk, Thomas Hardjono, and Jennifer Seberry. *Fundamentals of Computer Security*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. ISBN: 978-3-662-07324-7. DOI: 10.1007/978-3-662-07324-7_1. URL: https://doi.org/10.1007/978-3-662-07324-7_1.

[PT01] Federal Information Processing and Announcing The. *Announcing the ADVANCED ENCRYPTION STANDARD (AES)*. 2001.

[Ran16] Deepali D. Rane. “Superiority of Twofish over Blowfish”. In: *International Journal of scientific research and management* (Jan. 2016). DOI: 10.18535/ijsrm/v4i11.01.

[Rij97] Vincent Rijmen. *Cryptanalysis and Design of Iterated Block Ciphers*. 1997.

[Sch94] Bruce Schneier. “Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish)”. In: *Fast Software Encryption, Cambridge Security Workshop*. London, UK, UK: Springer-Verlag, 1994, pp. 191–204. ISBN: 3-540-58108-1. URL: http://dl.acm.org/citation.cfm?id=647930.740558.

[Sch95] Bruce Schneier. *Applied Cryptography (2Nd Ed.): Protocols, Algorithms, and Source Code in C*. New York, NY, USA: John Wiley & Sons, Inc., 1995. ISBN: 0-471-11709-9.

[Sch98] Bruce Schneier. *The Twofish Encryption Algorithm*. 1998. URL: https://www.schneier.com/academic/archives/1998/12/the_twofish_encrypti.html.

[ST03] Adi Shamir and Eran Tromer. "On the Cost of Factoring RSA-1024". In: *RSA CryptoBytes* (2003).

[Sha49] C. Shannon. "Communication Theory of Secrecy Systems". In: *Bell System Technical Journal, Vol 28, pp. 656–715* (Oct. 1949).

[Sil+16] Natassya B.F. Silva et al. "Case Studies of Performance Evaluation of Cryptographic Algorithms for an Embedded System and a General Purpose Computer". In: *J. Netw. Comput. Appl.* 60.C (Jan. 2016), pp. 130–143. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2015.10.007. URL: http://dx.doi.org/10.1016/j.jnca.2015.10.007.

[Smi71] J. L. Smith. *The Design of Lucifer, A Cryptographic Device for Data Communication*. Yorktown Heights, New York, 1971.

[Sta17] William Stallings. *Cryptography and network security: principles and practice*. Pearson Prentice Hall, 2017.

[ST93] National Institute of Standards and Technology. "Clipper Chip Technology". In: (Apr. 1993).

[ST97] National Institute of Standards and Technology. *Advanced Encryption Standard (AES) Competition*. 1997. URL: https://competitions.cr.yp.to/aes.html.

[Sti05] Douglas R. Stinson. *Cryptography: Theory and Practice*. 3rd. Discrete Mathematics and Its Applications. Chapman and Hall/CRC, 2005.

[SN16] Manju Suresh and M. Neema. "Hardware Implementation of Blowfish Algorithm for the Secure Data Transmission in Internet of Things". In: *Procedia Technology* 25 (2016), pp. 248–255. DOI: 10.1016/j.protcy.2016.08.104.

[TK11] J Thakur and Nagesh Kumar. "DES, AES and Blowfish: Symmetric Key Cryptography Algorithms Simulation Based Performance Analysis". In: 1 (Jan. 2011), pp. 6–12.

[TW02] W. Trappe and L.C. Washington. *Introduction to Cryptography: With Coding Theory*. Prentice Hall, 2002. ISBN: 9780130618146. URL: https://books.google.it/books?id=kVU%5C_AQAAIAAJ.

[Vau96] Serge Vaudenay. "On the weak keys of blowfish". In: *Fast Software Encryption*. Ed. by Dieter Gollmann. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 27–32. ISBN: 978-3-540-49652-6.

[Wie93] Michael J. Wiener. *Efficient DES Key Search*. Tech. rep. Ottawa, Ontario 4H7, Canada: Bell-Northern Research, Aug. 1993. URL: http://birr.mirrorservice.org/sites/ftp.wiretapped.net/pub/security/cryptography/literature/crypto-papers/des-key-search.pdf.

[Wika] Wikipedia. *Feistel Cipher*. URL: https://en.wikipedia.org/wiki/Feistel_cipher.

[Wikb] Wikipedia. *Key distribution*. URL: https://en.wikipedia.org/wiki/Key_distribution.