

Progetto

C. 1007

Data Scadenza Inchiesta

28-02-2009

Data Pubblicazione

2009-...

Classificazione

79-...

Titolo

**Protocollo di comunicazione per il trasferimento di informazioni di sicurezza (allarmi)
Parte 1: Livelli di trasporto**

Title



IMPIANTI E SICUREZZA DI ESERCIZIO



CEI COMITATO ELETTROTECNICO ITALIANO

AEIT FEDERAZIONE ITALIANA DI ELETTROTECNICA, ELETTRONICA, AUTOMAZIONE, INFORMATICA E TELECOMUNICAZIONI

CNR CONSIGLIO NAZIONALE DELLE RICERCHE

PREMESSA

La presente Norma Sperimentale fa parte dell'insieme di norme 79/5 che definisce il "Protocollo di comunicazione per il trasferimento di informazioni di sicurezza" in Sistemi di Supervisione Centralizzati per la gestione di segnalazioni di allarme.

Questo insieme di norme è stato definito e strutturato tenendo come riferimento il modello ISO/OSI con i relativi livelli ed è così organizzato:

Norma Sperimentale 79-5.1: tratta dei livelli data-link, rete e trasporto e annulla e sostituisce la seconda edizione della Norma 79/5 "Protocollo di comunicazione per il trasferimento di informazioni di sicurezza (allarmi) – Parte 1: Livello di Trasporto" e la Norma 79-7 "Protocollo CEI 79/5 – Guida all'applicazione";

Norma Sperimentale CEI 79-5.2: tratta del livello applicativo e annulla e sostituisce la seconda edizione della Norma 79/6 "Protocollo di comunicazione per il trasferimento di informazioni di sicurezza (allarmi) – Parte 2: Livello applicativo".

Norma Sperimentale 79-5.3: introdotta in questa revisione della Norma, contiene le definizioni dei termini utilizzati nelle altre due Norme, le tabelle di valorizzazione dei campi contenuti nelle trame applicative descritte nella Norma 79/5.2 e le prescrizioni inerenti la compilazione dei profili di interoperabilità.

Questa separazione è stata adottata al fine di agevolare il recepimento delle eventuali EPT (Estensioni di Protocollo Transitorie) presentate al CEI, limitando alla sola Norma 79-5.3 gli interventi di aggiornamento conseguenti.

1 Oggetto e scopo

1.1 Oggetto

La presente Norma sperimentale si applica ai sistemi di sicurezza centralizzati (quali ad esempio, antieffrazione, antintrusione, antifurto, antirapina, antiaggressione, rivelazione incendio, controllo accessi, videosorveglianza,.....), ove è necessaria la trasmissione di allarmi, comandi ed informazioni tra un Centro di Supervisione e Controllo (CSC) e una o più Centrali di Allarme (CA).

1.2 Scopo

Lo scopo della presente Norma, è la definizione (con riferimento al modello ISO/OSI) dei livelli Data-Link, Rete e Trasporto del protocollo di comunicazione per lo scambio bidirezionale di comandi e informazioni di sicurezza tra Centrali di Allarme (CA) di costruttori diversi, ed uno o più Centri di Supervisione e Controllo remoti (CSC). Il livello applicativo è descritto nella Parte 2 (Norma CEI 79-5.2). L'architettura ed i criteri di funzionamento delle Centrali, dei Sistemi di Trasmissione e dei Sistemi di Supervisione e Controllo esulano dagli scopi della presente Norma.

2 Definizioni

Per la definizione e il significato dei termini utilizzati fare riferimento alla sezione "Definizioni" nella Norma CEI 79-5/3.

3 Requisiti e prestazioni

3.1 Architettura dell'interfaccia

L'interfaccia definita dalla presente Norma prevede una struttura stratificata di protocolli secondo le linee guida del modello Open System Interconnection (OSI) dell'ISO.

L'interfaccia prevede l'implementazione di cinque dei sette livelli di protocollo:

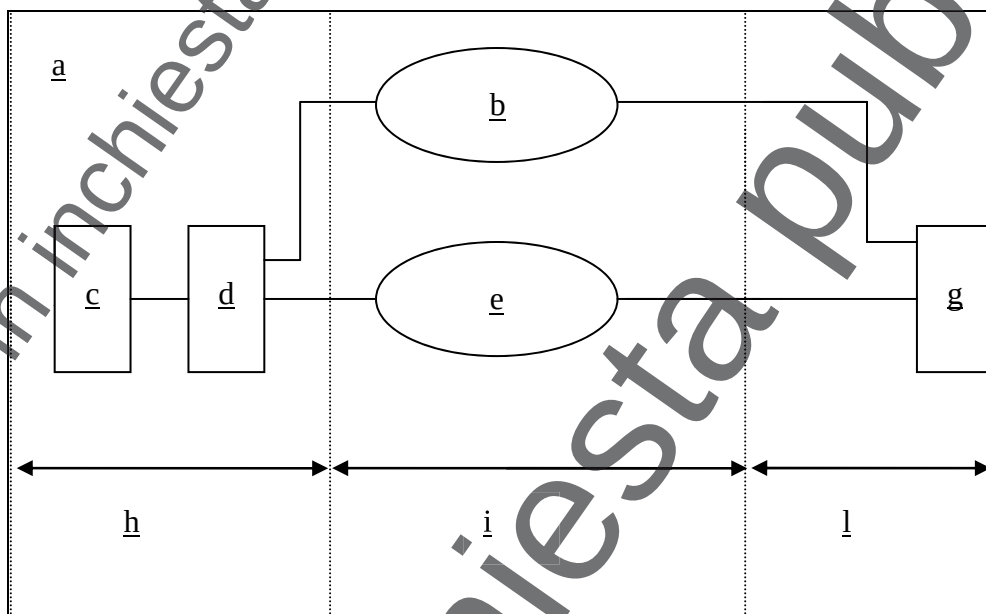
- 1) Livello fisico: definisce le modalità di interconnessione fisica, lo standard adottato e i circuiti utilizzati.
- 2) Livello di "data-link": definisce le regole logiche per assicurare il trasporto privo di errori di pacchetti di informazione dalla CA all'interfaccia diretta con il CSC e viceversa.
- 3) Livello di rete: a tale livello appartiene la gestione dell'apertura e dell'abbattimento dei collegamenti.
- 4) Livello di trasporto: definisce le regole volte ad assicurare la trasmissione corretta e protetta in modalità "end-to-end" dei pacchetti di informazione dalla CA agli organi centrali del CSC e viceversa. A tale livello appartiene la gestione dei criteri di utilizzo dei canali di collegamento, con particolare riferimento ai collegamenti di riserva. A tale livello appartengono le funzioni di identificazione e di protezione crittografica dei messaggi.
- 5) Livello di sessione: non implementato.
- 6) Livello di presentazione: non implementato.
- 7) Livello di applicazione: definisce le modalità di trasferimento delle informazioni di allarme e di stato della CA al CSC, e viceversa dei comandi nella direzione opposta.

L'interfaccia prevede due modalità di comunicazione:

- TIPO A: prevede una interrogazione ciclica tra CSC e CA di tipo master-slave in cui il CSC è il master. Ogni CA può trasmettere solo in seguito all'interrogazione
- TIPO B: prevede una comunicazione paritetica tra gli interlocutori, senza prevedere nessuna procedura di polling. Ogni interlocutore (CSC e CA) trasmette i dati quando resi disponibili dal livello superiore.

A titolo di esempio la Fig. 1 illustra possibili architetture per un sistema disupervisione e controllo.

Il protocollo descritto nella presente Norma si applica alle informazioni che transitano sulla rete di collegamento (i). Laddove la rete di comunicazione utilizzata prevede un protocollo standard è prevista la possibilità di veicolare il protocollo descritto nella presente Norma inserendolo nella parte dati di tale protocollo.



LEGENDA

- | | |
|---|--|
| <u>a</u> Schema fisico | <u>f</u> Convertitore di protocollo di comunicazione (CPC) |
| <u>b</u> Collegamento di backup | <u>g</u> Centrale di allarme |
| <u>c</u> Host computer | <u>h</u> Sistema di supervisione e di controllo |
| <u>d</u> Interfaccia di comunicazione (FEP) | <u>i</u> Rete di collegamento |
| <u>e</u> Collegamento principale | <u>l</u> CA |

In questo caso la CA comunica in modo nativo con il CSC utilizzando il protocollo CEI 79-5.

Fig. 1 - Esempi di architetture di un sistema di centralizzazione

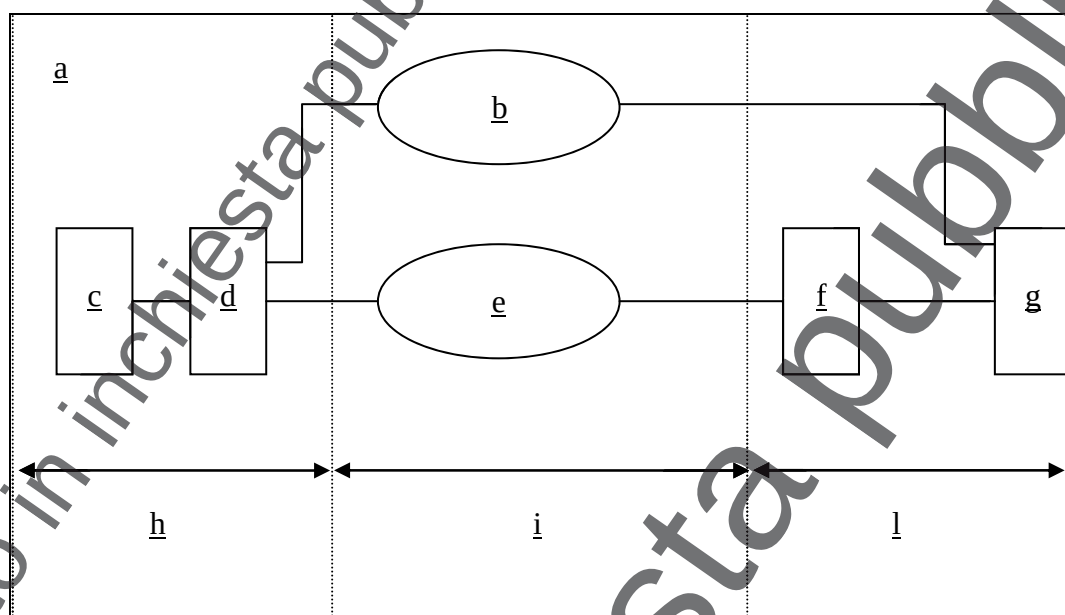


Figura 1 - In questo caso la CA è dotata di un CPC per la comunicazione primaria e di un comunicatore di backup a bordo centrale.

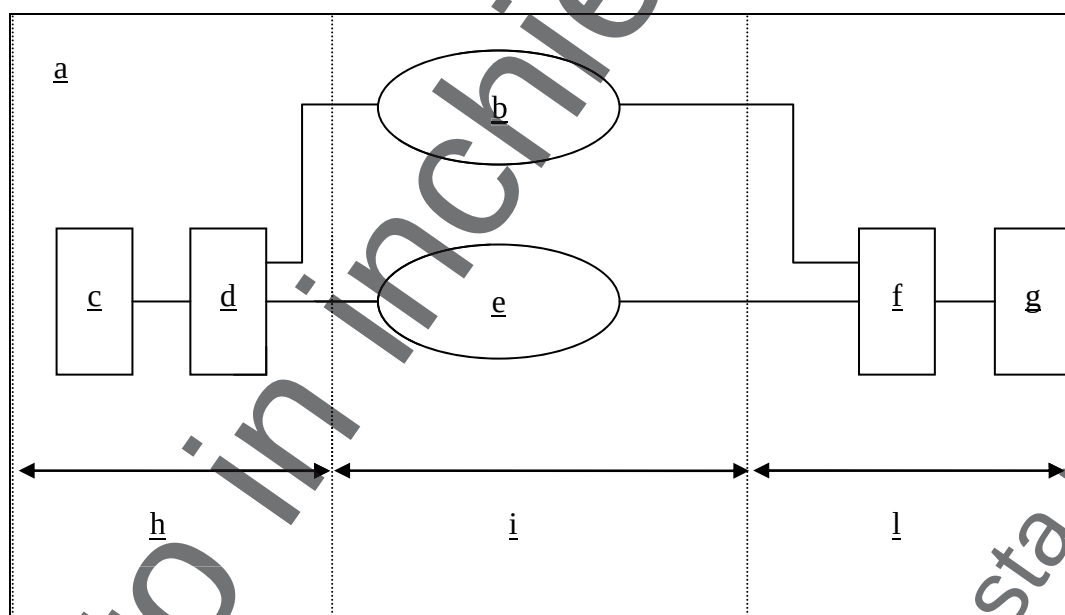


Figura 2 - In questo caso il CPC ha sia l'interfaccia per la comunicazione primaria che per quella di backup.

In collegamenti che comprendono un convertitore di protocollo CPC, il protocollo di comunicazione specificato dalla presente Norma si applica fra il CSC e l'ingresso del convertitore. La linea di collegamento centrale-CPC dovrà essere controllata dal convertitore stesso, e un suo malfunzionamento dovrà generare un allarme verso il CSC.

La linea di backup potrà essere collegata o alla centrale o, se previsto, al CPC.

3.2 Livello fisico

In questa edizione della norma non vengono date specifiche sul livello fisico in quanto si ritiene che tale caratteristica sia determinata dal mercato e dalle singole realtà applicative.

È comunque richiesto che il livello fisico implementato dal dispositivo sia specificato nel suo profilo di interoperabilità.

3.3 Livello di data link

Sono previste due possibili tipologie di comunicazione: tipo A e tipo B.

Tipo A

Prevede una interrogazione tra CSC e CA di tipo master-slave, in cui il CSC è il master. Ogni CA può trasmettere solo in seguito all'interrogazione da parte del CSC. In assenza di dati dai livelli superiori vengono trasmessi pacchetti di data-link vuoti (polling di sopravvivenza di data-link).

Il colloquio deve avere una cadenza tale da garantire il trasporto dell'evento di allarme nei tempi richiesti dai requisiti di sistema.

Vengono definiti due time-out (T1 e T2), uno per la CA ed uno per la CSC:

- T1 Time-out di attesa risposta (tempo massimo tra l'interrogazione del CSC e la risposta dalla CA, massimo 60 s)
- T2 Time-out di attesa interrogazione (tempo massimo tra due interrogazioni consecutive dal CSC verso una CA, max 300 s)

Il time-out T1 deve essere configurabile sul CSC, mentre T2 deve essere configurabile sulla CA.

I time-out sono validi solo a sessione aperta. Quando il CSC rileva lo scadere del time-out di attesa risposta può ritrasmettere o considerare chiusa la connessione; quando la CA rileva lo scadere del time-out di attesa interrogazione può considerare chiusa la connessione.

Il protocollo consente a un CSC e a più CA di interconnettersi e di operare assieme. Le comunicazioni sono possibili solo fra il CSC e le CA. Le CA non possono comunicare tra di loro.

Il protocollo garantisce l'invio corretto dei dati resi disponibili dal livello trasporto della CA e consente di controllare completamente il collegamento e rilevare disconnessioni.

La trasmissione di un pacchetto (da parte del CSC o della CA) e l'accettazione dello stesso da parte del destinatario indica solo che il dato è arrivato correttamente e non occorre ritrasmissione. Se al livello trasporto di un interlocutore sono pronti più pacchetti da trasmettere è necessario attendere il pacchetto di accettazione generato dal destinatario prima di inviare un altro pacchetto di trasporto.

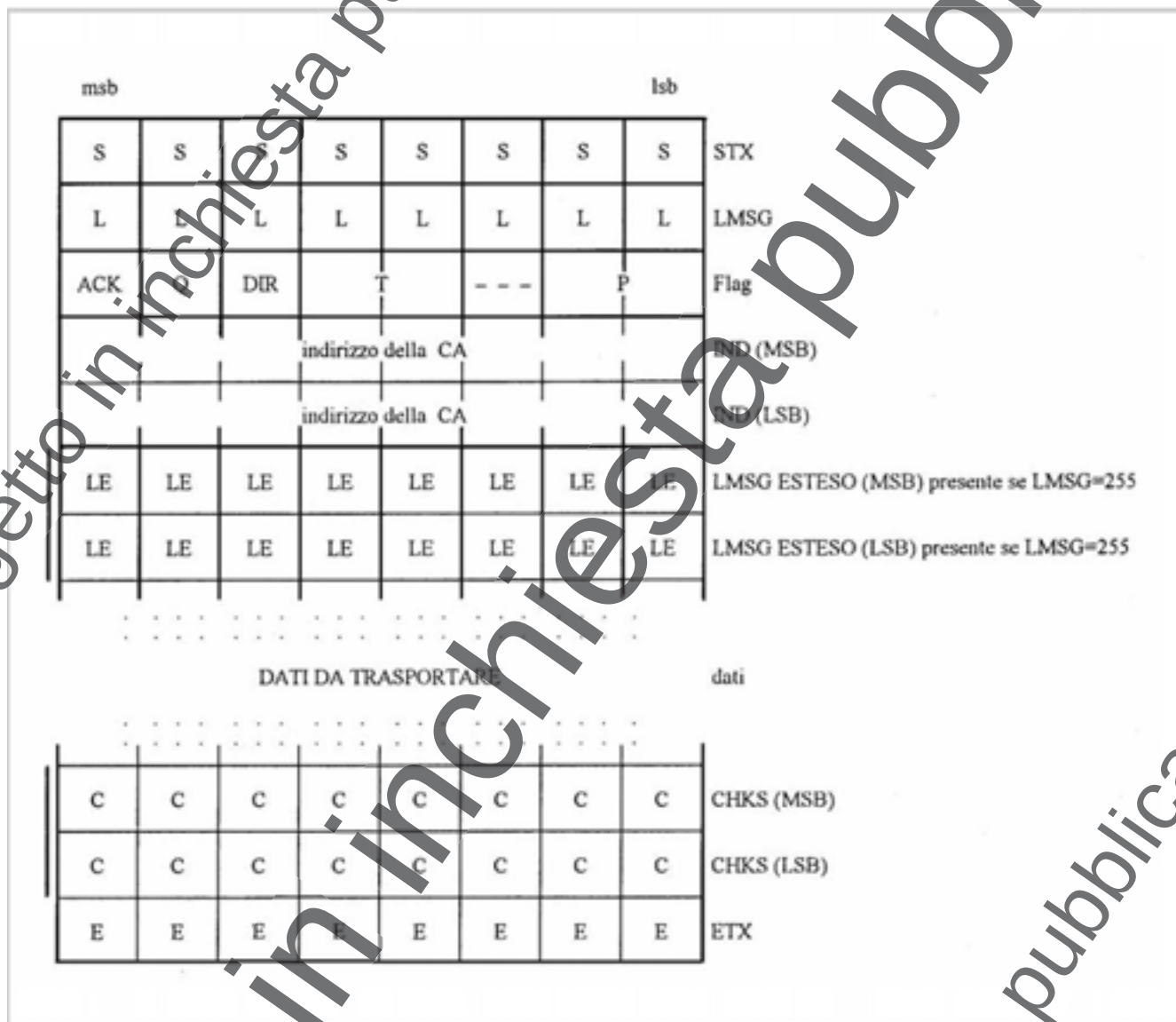
Tipo B

Lo scambio dati avviene in modo paritetico tra gli interlocutori, senza prevedere una procedura di "polling" per questo livello: i dati vengono immessi sulla rete di trasmissione da ogni interlocutore quando resi disponibili dai livelli superiori. Ne consegue che non vengono mai trasmessi pacchetti di data-link vuoti e non vengono utilizzati time-out di livello data-link.

La struttura del messaggio è uguale a quella prevista per il Tipo A, salvo che per i campi ACK e Q che vanno posti a zero: in questa modalità al livello data-link non è richiesta la conferma di ricezione dei pacchetti né la gestione delle ritrasmissioni. I pacchetti formalmente non corretti vengono scartati. Eventuali recuperi dell'errore tramite ritrasmissioni sono delegati al livello 4 (trasporto).

3.3.1 Struttura del pacchetto

La struttura del pacchetto dati al livello data link, sia per il tipo A che per il tipo B, è la seguente:



La struttura è trasparente, consente cioè il trasporto di dati in formato binario.

NOTA 1 Il primo bit trasmesso in linea è il bit meno significativo di ciascun byte (lsb).

NOTA 2 Il primo byte dei campi IND, LMSG ESTESO e CHKS (quello più vicino alla testa del pacchetto) è il più significativo e il secondo byte è il meno significativo.

Gli elementi che costituiscono il pacchetto sono:

STX	Carattere di Start of Text (esadecimale 02). Indica l'inizio del pacchetto.
LMSG	Se minore o uguale a 248, indica la lunghezza del campo dati in byte (escluso i 5 byte di controllo in testa ed i 3 in coda); Se uguale a 255, indica che la lunghezza del campo dati è definita dai byte denominati LMSG ESTESO; in questo caso la lunghezza del campo dati è oltre 248 byte e può arrivare ad un massimo di 65525 byte (escluso i 7 byte di controllo in testa ed i 3 in coda); I valori da 249 a 254 non sono utilizzabili.
LMSG ESTESO	Presente solo se LMSG è uguale a 255; in tal caso, indica la lunghezza del campo dati in byte (escluso i 7 byte di controllo in testa ed i 3 in coda); I valori minori di 249 e maggiori di 65525 non sono utilizzabili.
Flag	Flag di gestione del pacchetto composto da: - ACK questo bit è l'acknowledge dell'ultimo pacchetto da/a la CA indirizzata. Il valore "0" indica ACK positivo, il valore "1" indica ACK negativo. - Q è il bit di sequenza. Il bit è commutato ad ogni successivo pacchetto da/a una particolare CA, ma non è commutato nel caso di ritrasmissione di un pacchetto. Consente a chi riceve di determinare se si tratta di una ritrasmissione oppure di un nuovo pacchetto. - DIR è il bit di direzione. Se "0", indica che il pacchetto è originato dal CSC e, se "1", indica che il pacchetto è originato dalla CA. - T identificatore del tipo di pacchetto (2 bit) 00 non definito 01 pacchetto dati 10 non definito 11 non definito - P identificatore del tipo di protocollo (2 bit). 00 protocollo con CRC2 (Sommatoria su 16 bit di tutti i caratteri trasmessi (STX, CHKS, ETX esclusi)) 01 protocollo con CRC16 (polinomio $X^{16}+X^{15}+X^2+1$ calcolato sui caratteri trasmessi escluso STX, CHKS e ETX) 10 protocollo senza CRC, i byte CHKS vengono omessi 11 uso riservato per impieghi futuri
IND	Indirizzo fisico della CA (su due byte).
dati	Sono i dati (eventuali) dal/al livello trasporto della CA.
ETX	Carattere di End of Text (esadecimale 03). Serve per la chiusura del pacchetto.
CHKS	È il checksum di pacchetto, calcolato secondo l'algoritmo specificato nel campo "P" del carattere FLAG e trasmesso su due byte (il byte più significativo per primo). I due byte CHKS vengono omessi se il campo P è uguale a 10.

Relativamente alla struttura del messaggio, si richiede che tutti i byte ed i parametri previsti debbano essere verificati, pena lo scarto del messaggio stesso.

Con riferimento alla possibilità di avere una lunghezza messaggio LMSG di tipo esteso, ottenuta con l'aggiunta di due byte dopo l'indirizzo della CA e ponendo a 255 il byte di LMSG, al fine di avere un campo dati di lunghezza maggiore di 248 byte, si evidenzia che deve essere considerata un'opzione da concordare tra le parti. Tale opzione deve essere evidenziata nel profilo di centrale e di centro.

3.3.2 Modalità di inserimento di byte di controllo (Byte Stuffing)

La modalità di inserimento di Byte Stuffing è indipendente dal tipo di implementazione del livello data-link.

I caratteri di controllo STX (hex 02) e ETX (hex 03) vengono trasmessi come singolo carattere.

Eventuali caratteri STX (hex 02), ETX (hex 03), DLE (hex 10), presenti nel pacchetto (escluso quindi i caratteri di controllo) vengono fatti precedere da un carattere DLE con il bit 8 forzato a 1 (STX diventa hex 10 hex 82, ETX diventa hex 10 hex 83, DLE diventa hex 10 hex 90).

Il calcolo di LMSG (o LMSG ESTESO) e CHKS deve essere effettuato considerando la struttura originale del messaggio, senza tener conto dei caratteri DLE aggiuntivi. Per un approfondimento vedere l'Appendice A.

Eventuali altri caratteri di controllo, propri del mezzo trasmissivo, che fosse necessario mascherare, possono essere trattati con la stessa metodica, previo accordo tra le parti. Tale variazione dovrà essere evidenziata nel profilo di centrale e di centro.

3.3.3 Esempi di sequenze di interrogazione

Tipo A

La Norma presenta in 3.3.1 una struttura pacchetto del tipo illustrato nelle Fig. 2.1 e 2.1 a.

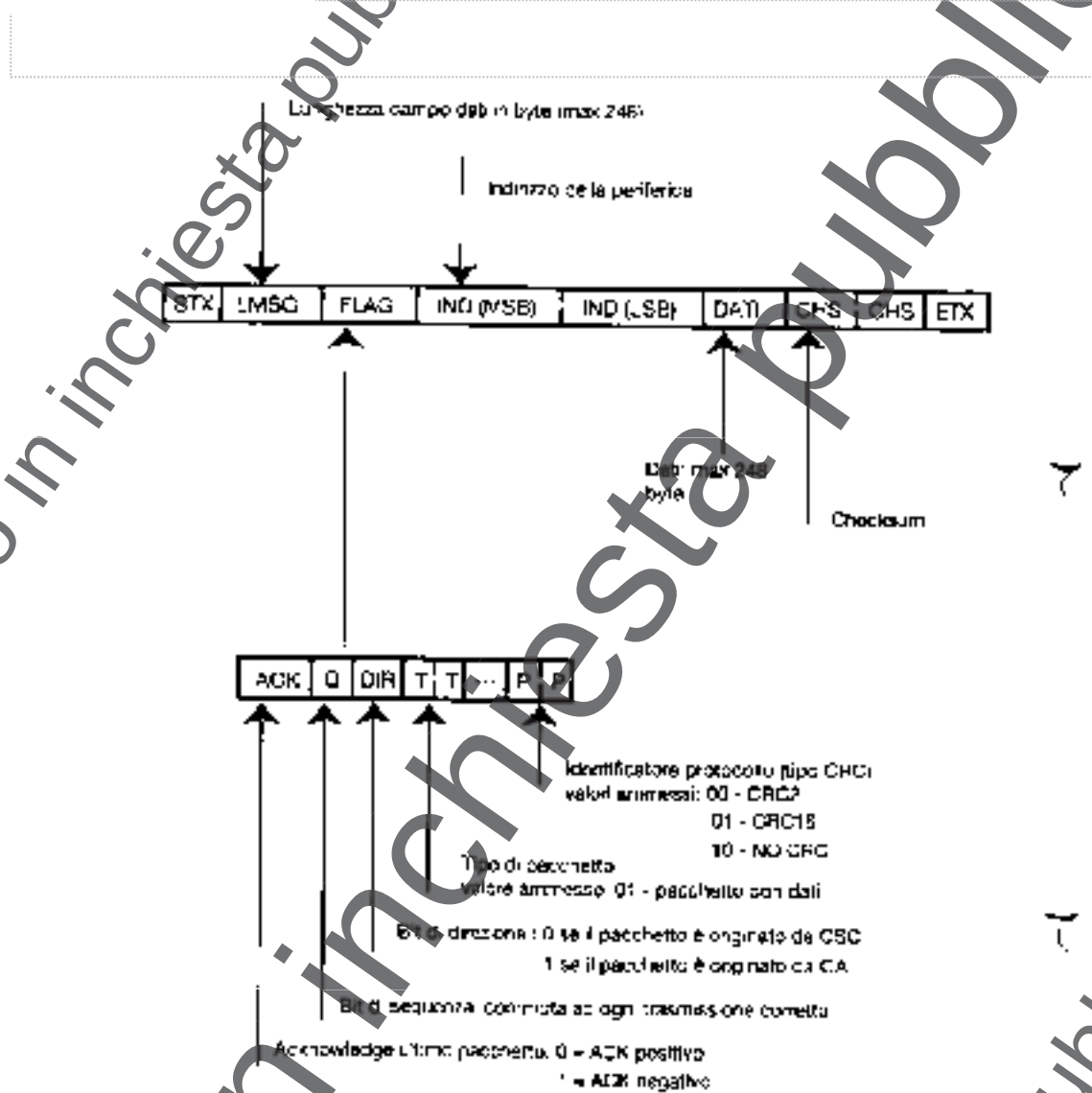


Fig. 2.1 - Struttura del pacchetto standard

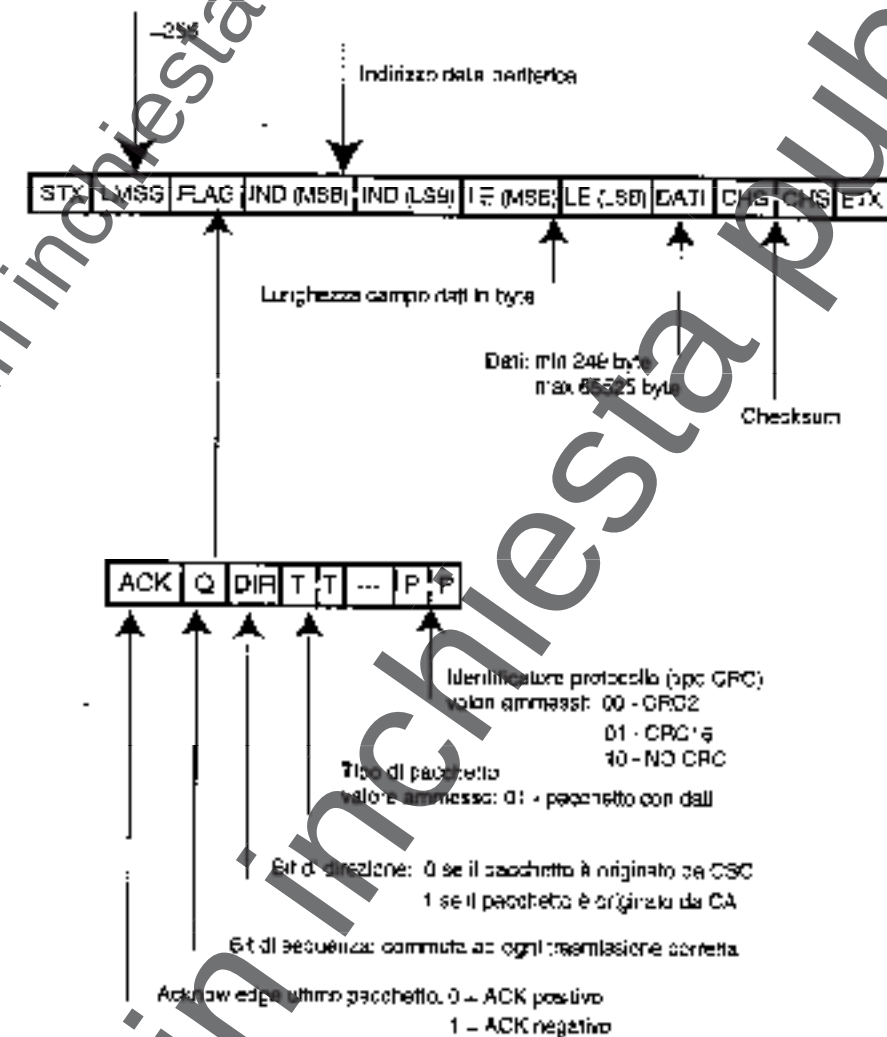


Fig. 2.43 - Struttura del pacchetto CAN

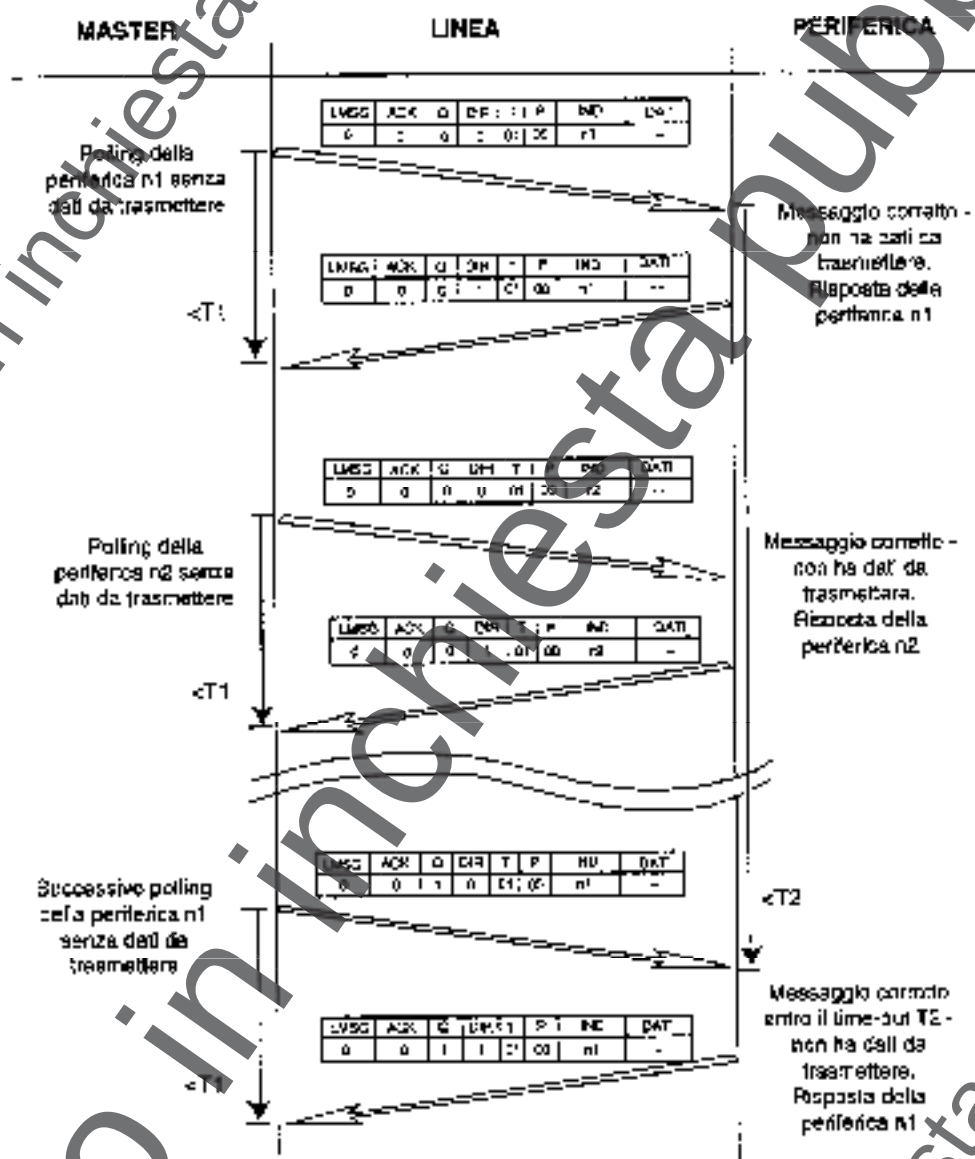
Il pacchetto descritto viene inviato alternativamente in linea da chi interroga (lato master) e da chi risponde (lato slave).

Quindi, il pacchetto costituisce al tempo stesso:

- 1) la trama di interrogazione (polling frameframe);
- 2) la trama di trasporto dati (data frameframe) in entrambe le direzioni;
- 3) la conferma alla ricezione precedente (piggy-backed acknowledgement).

La trama di interrogazione può o meno contenere dati, mentre contiene sempre l'informazione di conferma (o di mancata conferma) della ricezione precedente.

Le Fig. 2.2 e 2.3 illustrano alcune sequenze di scambio pacchetti con e senza dati vivi. Sono evidenziati i due time-out (T1 e T2).



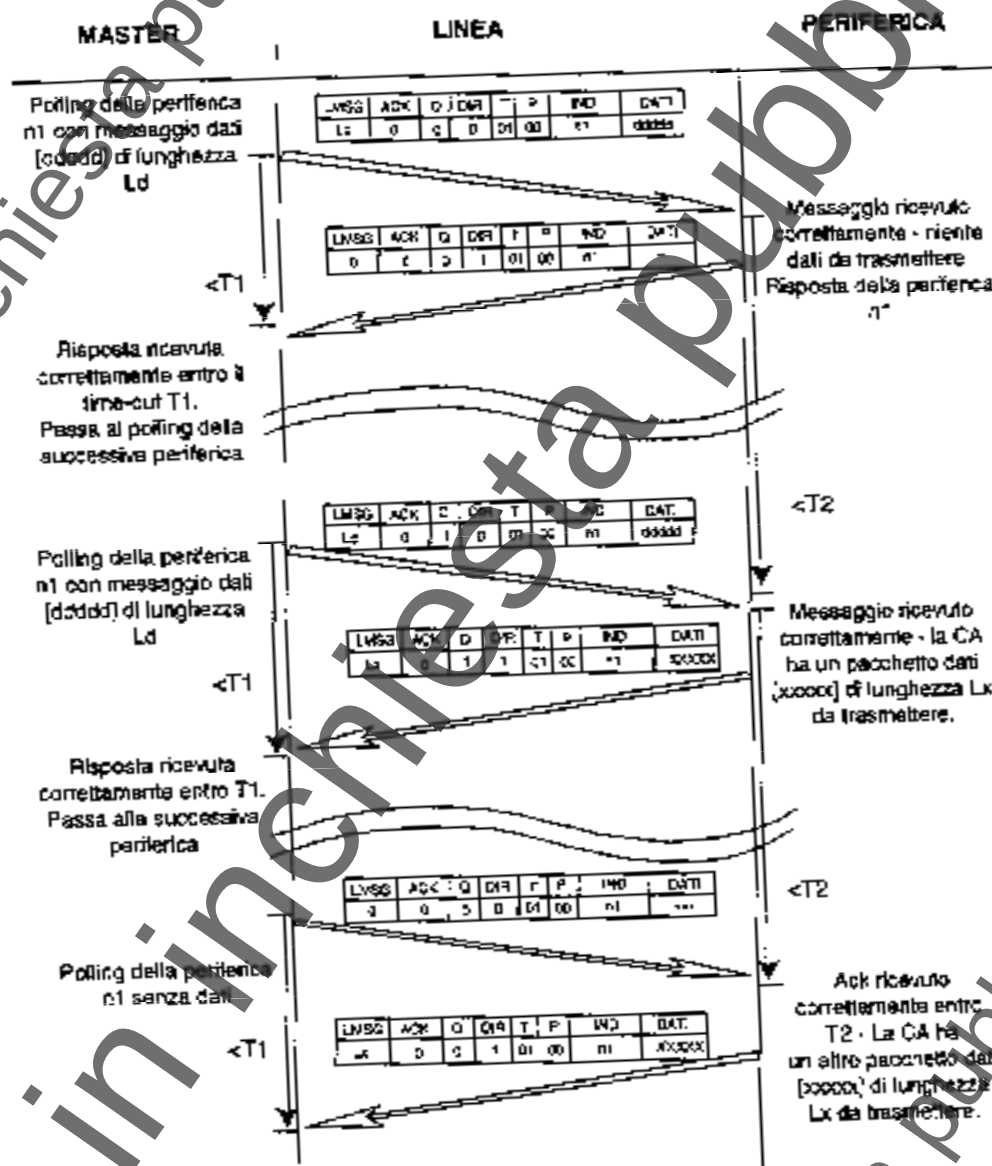
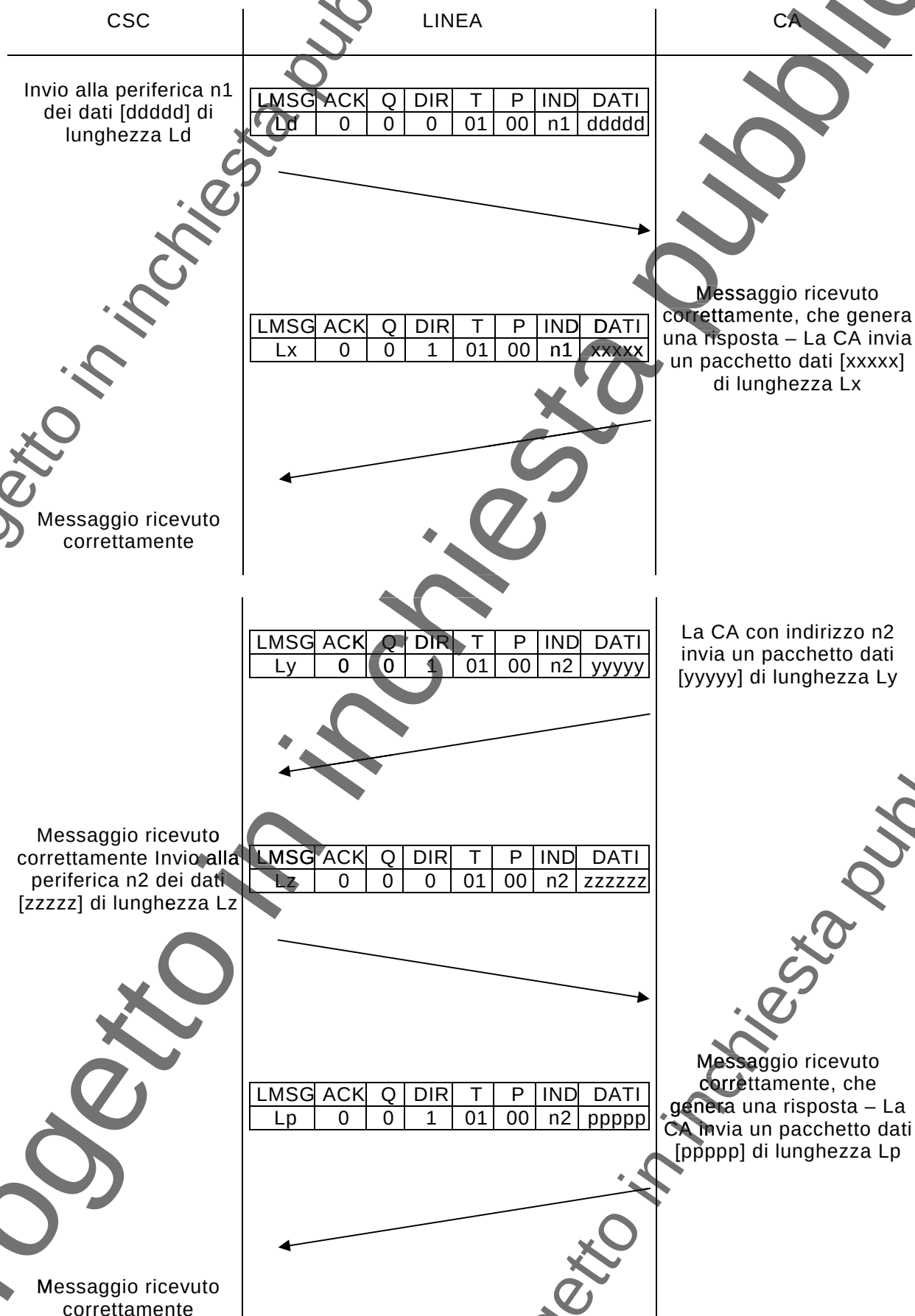


Fig. 2.3 — Sequenze di polling con scambio di dati vivi

Tipo B

Qualsiasi interlocutore ha dei dati da trasmettere li invia in rete



3.3.4 Utilizzo dei campi ACK e Q nel pacchetto

Tipo A

ACK: è il bit di conferma (acknowledgement) della buona ricezione dell'ultimo pacchetto da/a la

CA (Centrale di Allarme).

Il valore "0" indica conferma (ACK) positiva, il valore "1" indica conferma negativa.

Q è il bit di sequenza.

È commutato ad ogni successivo pacchetto da/a un particolare CA, ma non è commutato nel caso di ritrasmissione di un pacchetto. Consente a chi riceve di determinare se si tratta di una ritrasmissione oppure di un nuovo pacchetto.

Il bit Q viene commutato dal *master* di protocollo dopo aver ricevuto una risposta corretta e prima di inviare un nuovo pacchetto, lo *slave* risponde con il medesimo Q ricevuto. Il lato *master* considera risposta corretta la ricezione di un messaggio con Q uguale all'ultimo Q inviato, mentre il lato *slave* considera risposta corretta la ricezione di un messaggio con Q commutato rispetto all'ultimo stato inviato.

In questo modo Q consente di rivelare quasi tutte le situazioni di errata ricezione di un pacchetto, come schematizzato in fig. 3.1 e 3.2. C'è una situazione nella quale è necessario l'uso di ACK:

Dopo una ripresa di collegamento esiste il pericolo che il *master*, inviando nel primo messaggio un dato Q (ad esempio = 0) ed un ACK positivo, cioè =0, finisca involontariamente col confermare all'interlocutore *slave* il buon esito dell'ultimo invio dello *slave* stesso, con conseguente possibile perdita di un messaggio, nel caso in cui invece l'interruzione abbia portato alla mancata ricezione di tale messaggio.

Appare quindi opportuno che, dopo un'interruzione di collegamento, il primo messaggio da *master* a *slave*:

- 1) non contenga mai dati vivi;
- 2) abbia lo stato di ACK forzato a 1, in modo da impedire ogni possibile perdita di dati.

Tali considerazioni comportano che, per entrambi i lati del protocollo, in ogni caso in cui viene ricevuto un messaggio con Q ed ACK incoerenti tra loro, indicanti cioè l'uno una conferma positiva e l'altro una contraria, deve valere la regola per cui il dato negativo prevale ed innesca così la procedura di ripetizione per conferma negativa.

Nel primo caso, illustrato in Fig. 3.1, l'errore blocca o altera un messaggio da *master* a *slave*, in tal caso il ricevente (*slave*) non riceve affatto o scarta il pacchetto, ed il trasmittente (*master*), dopo un tempo di attesa risposta, ripete l'invio senza commutare Q. La correzione dell'errore di trasmissione comporta la ripetizione di un pacchetto.

La seconda situazione è mostrata in Fig. 3.2. Anche in questo caso un errore di linea blocca o altera un messaggio, questa volta diretto da *slave* a *master*. Il ricevente del messaggio (*master* di protocollo) non riceve il pacchetto oppure lo scarta, quindi, dopo un tempo di attesa, ripete l'invio del precedente pacchetto (già ricevuto dal suo interlocutore) senza commutare Q rispetto all'ultimo invio nella medesima direzione. In tale situazione lo *slave* di protocollo si vede pervenire un pacchetto ripetuto e con Q non commutato. Rilevata tale condizione, esso deve ripetere la trasmissione di quel pacchetto che era stato vittima dell'errore di linea. È da notare che per ripetere tale operazione lo *slave* deve forzare la commutazione di Q in modo da tornare indietro di un passo. La correzione dell'errore di trasmissione comporta la ripetizione di due pacchetti

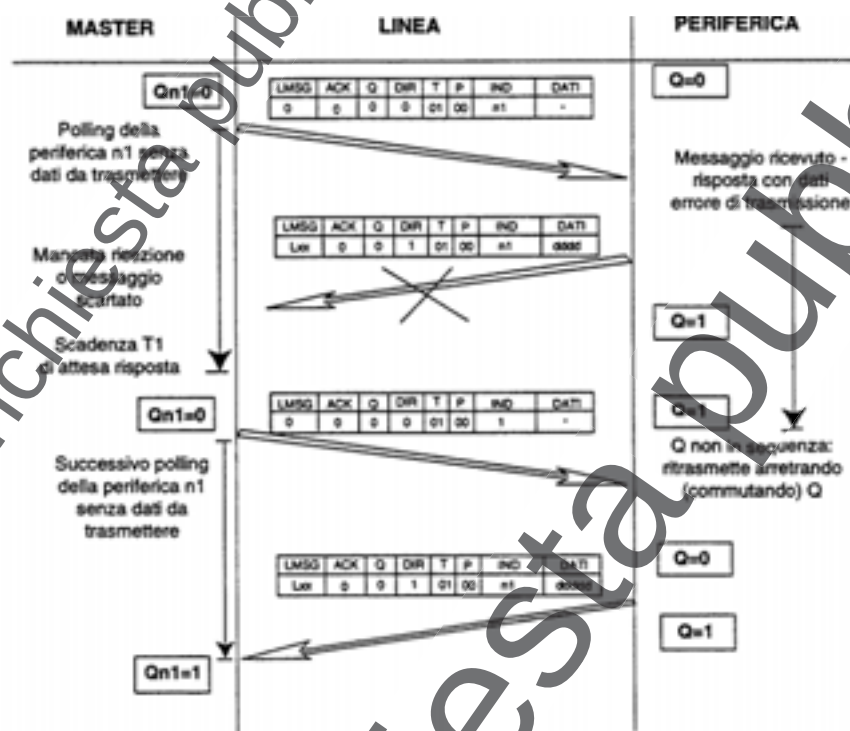


Fig. 3.2 — Sequenza di polling con errore da slave a master

La figura 3.3 illustra il caso di primo scambio dati dopo un'interruzione della connessione.

In tale situazione il master pone ACK=1 in modo da forzare lo slave alla ripetizione dell'ultimo messaggio dati eventualmente inviato prima dell'interruzione del collegamento.

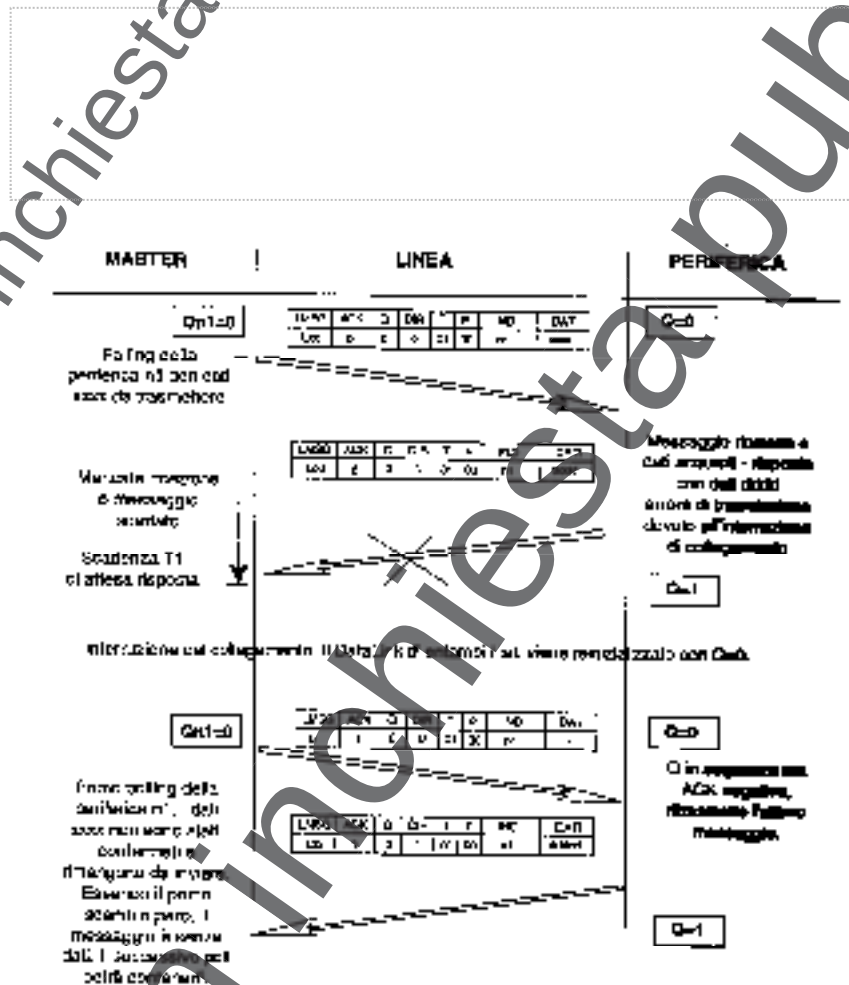


Fig. 3.3 - Prima sequenza di dati dopo interruzione di collegamento

Tipo B

Come già indicato, i campi ACK e Q vanno posti a 0.

La conferma di avvenuta ricezione dei pacchetti è affidata ai meccanismi previsti dal livello di trasporto.

3.4 Livello rete

La funzione specifica del livello di rete, situato tra il livello di data-link ed il livello di trasporto, è quello di mettere a disposizione del livello sovrastante le procedure necessarie alla gestione delle linee di comunicazione, intendendo con ciò:

- l'attivazione del colloquio CSC-CA sul portante fisico;
- il monitor della qualità del collegamento;
- l'eventuale ricorso a una linea di riserva;
- l'abbattimento della connessione fisica.

Tutte le operazioni di instaurazione/abbattimento/commutazione del collegamento, avvengono in base alle richieste del livello di trasporto; questo deve essere informato riguardo alla diagnostica di rete in modo da provvedere di conseguenza.

Una CA può essere in grado di operare su più tipi di vettori (dedicati, commutati), ovvero su un solo vettore che deve essere dichiarato dal costruttore.

Si considera che per ciascun tipo di rete esista un'unica connessione attiva per volta, e quindi non è necessario aggiungere al messaggio alcuna informazione aggiuntiva propria del livello rete (header di rete nullo).

3.4.1 Collegamento su linea di riserva

In caso di collegamento su linea di riserva attivato dalla CA, terminata la fase di apertura del collegamento, la CA si predispone per ricevere interrogazioni con indirizzo 0 (sia per l'indirizzo fisico del livello data link che per l'indirizzo logico di destinazione del livello trasporto).

Normalmente gli indirizzi ammessi per la CA partono dal valore 1: l'indirizzo 0 è riservato per l'instaurazione del collegamento con il centro su una linea di riserva con chiamata dalla CA.

Ricevuto questo messaggio di interrogazione con indirizzo 0 (entro un tempo massimo di 50 secondi dall'attivazione della linea di riserva), la CA risponde con il proprio indirizzo e da questo momento in poi la comunicazione prosegue con l'indirizzo specifico della CA.

La CA, venendo a mancare i criteri di consenso e/o il flusso di dati di trasporto per un tempo superiore ad un valore definito T2 (max 300 secondi), disconnette la linea.

In caso di collegamento con chiamata dal CSC, la CA attiva il protocollo al livello data link e si predispone a ricevere messaggi di interrogazione con il proprio indirizzo.

Gli interlocutori (CA e CSC) si possono scambiare dati, solo dopo la fase iniziale di riconoscimento, utilizzando per la CA l'indirizzo specifico (non l'indirizzo 0).

La norma non entra nel merito del tipo di linea da utilizzare sia per quanto riguarda il collegamento primario che per quello di riserva.

Azioni del livello di rete

Attivazione del colloquio CSC-CA

È la procedura con la quale vengono attivati i mezzi trasmissivi a disposizione.

Il livello di rete riceve la richiesta dal livello di trasporto, provvede ad inviare un'informazione di servizio al livello fisico e quindi avvisa il livello di data link in modo che questo attivi la sessione di comunicazione.

Monitor della qualità del collegamento

Il livello di rete riceve dal livello data link informazioni dalle quali è in grado di stimare la qualità del collegamento (retry, time-out ecc.), in modo che il livello di trasporto possa valutare l'opportunità di richiedere la commutazione sulla linea di riserva.

Ricorso alla linea di riserva

Ove si renda necessaria l'operazione di "commutazione correttiva" il livello di rete opera come segue:

- sospende il data link
- chiede l'attivazione della linea di riserva,
- riattiva il data link sulla linea di riserva.

L'instaurazione del collegamento segue le modalità indicate in 3.4.1.

Abbattimento della connessione fisica

Su richiesta dei livelli superiori può essere necessario chiudere il collegamento richiedendo di troncatura la connessione fisica.

La chiusura del collegamento può avvenire anche per mancanza di risposta da parte dell'interlocutore per un tempo che non può eccedere in ogni caso i 300 s; tale time-out deve essere innescato solo dopo che si è raggiunta la connessione logica, terminate quindi tutte le fasi di identificazione.

Una volta che la connessione è stata instaurata sulla linea di riserva, al fine di monitorare lo stato della linea principale, nel caso di interfaccia di tipo A è consentito di mantenere attivo il data link sulla linea principale, purché non avvenga scambio di dati; nel caso di interfaccia di tipo B, è consentito di mantenere attivo il trasporto sulla linea principale, purché non avvenga scambio di dati applicativi. Quando si verifica che la linea principale è di nuovo attiva la CA abbatte la connessione sulla linea di riserva, e il suo livello di rete riattiva il livello di trasporto sulla linea principale.

Livello trasporto

Sono previste due possibili tipologie di comunicazione: tipo A e tipo B.

3.5 Tipo A

Tutte le comunicazioni tra livello applicativo e di rete e viceversa devono necessariamente transitare attraverso il livello di trasporto.

I servizi offerti dal livello trasporto ai livelli superiori sono:

- apertura e abbattimento della connessione end-to-end
- controllo di flusso della connessione end-to-end e scambio dati full-duplex con finestra di 1 messaggio
- identificazione degli interlocutori
- autenticazione dei dati
- protezione crittografica dei dati nei due sensi
- gestione dei canali di connessione a disposizione (con particolare riferimento ai collegamenti di riserva) e relativa diagnostica

Al livello trasporto è demandata la funzione di apertura e abbattimento della connessione end-to-end tra CSC e CA.

Il livello trasporto accetta i pacchetti generati dall'applicazione, li passa al livello rete e si assicura che tutti i dati arrivino correttamente al livello omologo della destinazione.

A tale scopo il corretto arrivo a destinazione di ciascun pacchetto trasmesso deve essere confermato dalla ricezione del relativo pacchetto di conferma. Nessun nuovo pacchetto dati può essere inoltrato da un interlocutore se il precedente pacchetto non è stato confermato positivamente dall'altro.

Il livello trasporto, nel caso di connessioni protette da crittografia, ha il compito di gestire le chiavi di cifratura, di autenticare ed eventualmente di cifrare e decifrare i pacchetti in transito.

A tale scopo sono previste funzioni per:

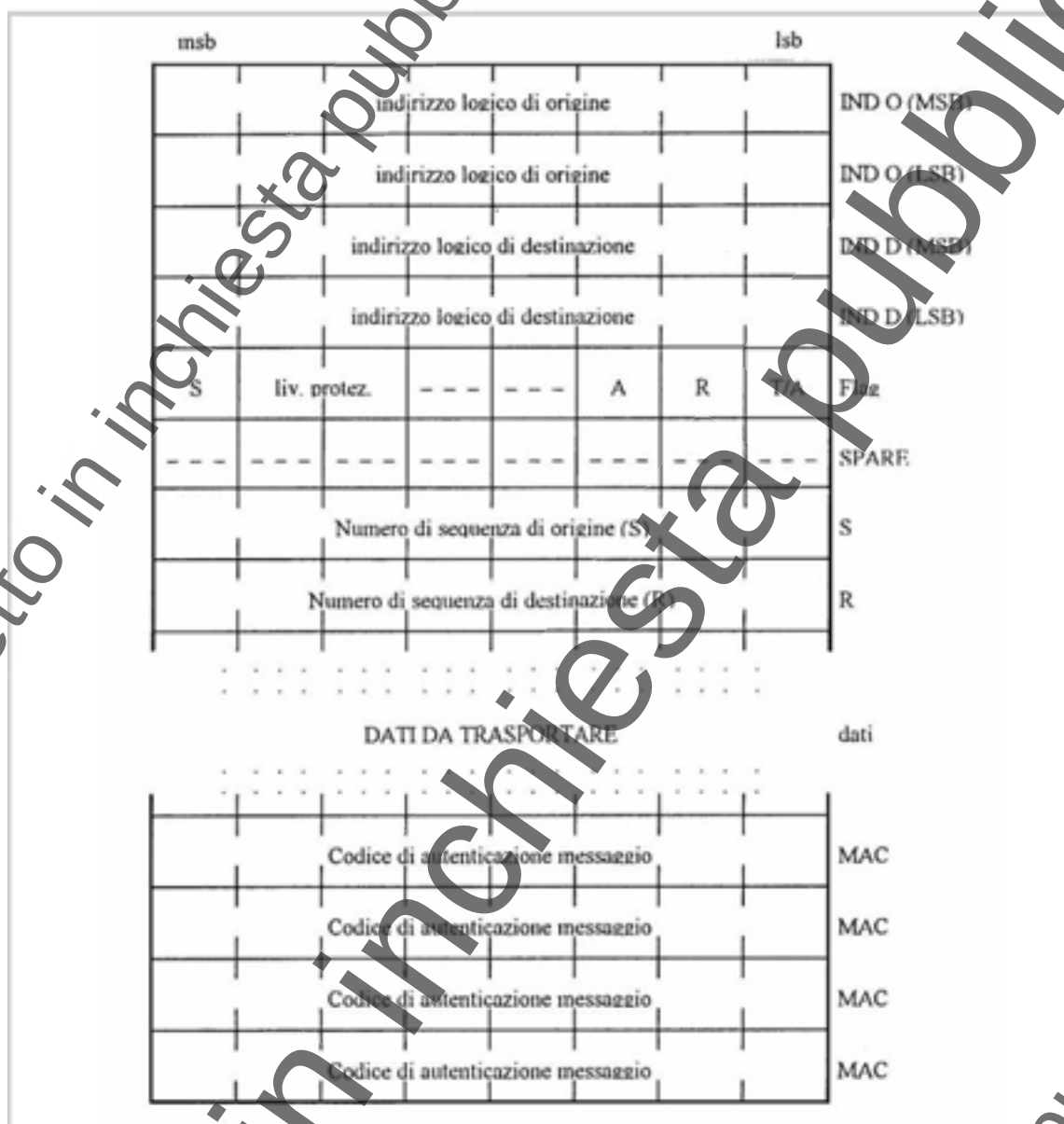
- autenticazione dei dati
- cifratura e decifratura dei dati
- generazione dinamica delle chiavi di sessione (SK) (al CSC)
- invio delle chiavi SK da CSC a CA
- riallineamento in caso di caduta totale o parziale della connessione.

Il livello trasporto riceve dal livello rete tutte le informazioni relative allo stato dei mezzi di comunicazione e gestisce di conseguenza l'utilizzo dei vettori a disposizione e le eventuali procedure di notifica al livello applicativo.

3.5.1 Struttura del pacchetto di trasporto

Per lo svolgimento delle funzioni sopracitate il livello trasporto utilizza una struttura dati composta da una testata propria del livello trasporto e da un campo dati che può contenere o le informazioni generate dall'applicazione o i dati necessari al livello per espletare le proprie funzioni (gestione chiavi, identificazione o altro).

La struttura del pacchetto di trasporto illustrata fa riferimento ai livelli di protezione 0, 1 e 2. Per il livello di protezione 3 la struttura del pacchetto sarà definita nel relativo profilo di centrale interoperabile.



IND O indirizzo logico dell'origine (2 byte)

IND D indirizzo logico della destinazione (2 byte)

Flag byte di servizio per specificare il tipo di servizio. I bit non utilizzati di questo byte devono essere posti a 0

T/A 0 = pacchetto contenente dati applicativi

1 = pacchetto di trasporto

R 1 = rimessa in passo del contatore di sequenza

Impone al ricevente di inizializzare il numero di sequenza del suo omologo

0 = il contatore prosegue nella sequenza corrente

A 1 = è richiesto messaggio di conferma esplicito (con firma)

0 = deve essere inviata conferma veloce, non ci deve essere cioè conferma esplicita (con firma)

livello di protezione dei DATI (2 bit)

0 = trasmissione in chiaro

1 = trasmissione in chiaro con autenticazione

2 = trasmissione cifrata con autenticazione

3 = trasmissione cifrata con autenticazione, con chiavi di cifratura di lunghezza maggiore di 64 bit

I valori 0, 1, 2 o 3 devono essere sempre presenti, indipendentemente dal fatto che nel pacchetto ci siano o meno dati. Tali valori indicano infatti il tipo di protezione della comunicazione.

S 0 = livello di protezione come configurato
 1 = livello di protezione degradato

L'informazione S=1 è indipendente e prioritaria rispetto all'indicazione del livello di protezione. Quest'ultima è infatti una caratteristica del sistema e non tiene conto di eventuali malfunzionamenti che comportano una temporanea degradazione indicata appunto da S. Quando il livello di protezione dei dati è 0, necessariamente anche S deve essere posto a 0.

SPARE byte a disposizione per future espansioni . . Deve essere posto a 0

BYTE S numero di sequenza del trasmittente il messaggio per controllo di flusso (3.5.1)

BYTE R numero di sequenza del ricevente il messaggio per controllo di flusso (3.5.1)

Dati Dati da trasportare (se necessario) I dati da trasportare possono essere o dati applicativi o dati di trasporto. Non è previsto l'invio di pacchetti misti. (79-7) La Norma non prescrive nulla sulla gestione dell'invio dei messaggi.

MAC Codice di autenticazione messaggio (se richiesto). È presente solo se il livello di protezione è > 0. Nel caso non sia richiesta autenticazione i byte del MAC non devono essere inseriti. È costituito da 4 byte nel caso di livello di protezione 1 e 2. Con livello di protezione 3 la lunghezza dipende dalla dimensione del MAC utilizzato.

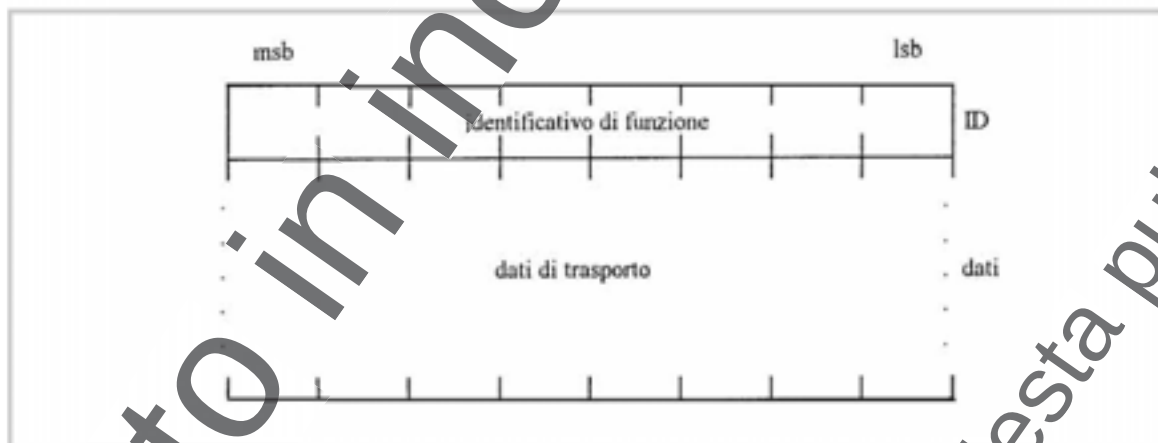
Quando $T/A = 1$ non è detto che devono essere a 0 A ed S, in quanto T/A è un'indicazione dinamica e quindi indipendente dagli altri elementi; in ricezione A ed S non vengono presi in considerazione, poiché le eventuali mascherature sono relative solo ai dati applicativi.

3.5.1.1 Struttura dei dati di trasporto

Se il pacchetto contiene dati di applicazione (bit $T/A = 0$), la struttura degli stessi è quella definita dal livello applicativo.

I dati vengono passati dal livello di trasporto a quello applicativo così come sono ricevuti.

Se il pacchetto contiene dati di trasporto (quindi con bit $T/A = 1$) la struttura dei dati è la seguente:



L'identificativo di funzione è un byte che definisce lo scopo del pacchetto dati. La lunghezza del pacchetto dati dipende dal tipo di funzione.

3.5.1.2 Codici di identificazione funzione

Si definiscono i tipi di pacchetti riportati in tabella. Il numero di byte del campo dati nei pacchetti con ID= 2, 3, 4, 5, 6, 7, 8-10 valgono nel caso di livello di protezione 0, 1 o 2. Per il livello di protezione 3 la lunghezza del campo dati dipende dalla dimensione della chiave di cifratura, del blocco dati e del MAC utilizzati.

ID	tipo di pacchetto	numero byte nel campo dati	significato dati
1	richiesta trasferimento chiave	0	
2	trasferimento chiave - fase b	8	Valore casuale R1
3	trasferimento chiave - fase c	24	R1 R2 SK cifrati con MK
4	trasferimento chiave - fase d	8	valore R2 cifrato con SK
5	trasferimento chiave - fase e	8	valore R2
6	richiesta identificazione	8	valore casuale R1
7	identificazione - fase b	16	R1 R2 cifrati con SK
8	identificazione - fase c	8	valore R2
9	sospensione a tempo	1	tempo di sospensione in secondi
10	conferma firmata	4	MAC di conferma
11	errore di decifratura dati	0	
12	richiesta di trasferimento chiavi da CA	0	
13	richiesta trasferimento chiave con attivazione chiave master	1	numero della chiave master da attivare
14	Richiesta di identificazione da CA	0	
...	...	...	...
255	pacchetto di controllo	2	ID di riferimento + codice causa

L'informazione ID deve essere espressa in binario, come pure tutte le altre codifiche.

Nel pacchetto con ID = 13 la chiave master con numero $n \neq 0$ è la stessa utilizzata nel caso di ID = 1: se viene trasmesso un pacchetto con ID = 1 oppure con ID = 13 e $n = 0$ si deve attivare la stessa chiave master.

La cifratura dei pacchetti con ID 3 e 7 si applica come indicato in Appendice C. I pacchetti con codice 255 sono utilizzati come pacchetti di controllo per segnalare eventuali errori durante le procedura del livello di trasporto.

Il pacchetto di controllo contiene al suo interno due byte, di cui il primo è l'ID del pacchetto errato, il secondo il tipo di errore; le due informazioni devono essere congruenti, in caso contrario rifiutate.

I valori R1 e R2 hanno la stessa lunghezza delle chiavi di cifratura SK e MK.

Le notazioni R1||R2 e R1||R2||SK indicano che l'operazione di cifratura deve essere eseguita sul blocco formato dai byte di R1 e R2 oppure R1, R2 e SK concatenati come nell'esempio di seguito indicato, nel caso di chiave SK a 64 bit:

R1

R1_1	R1_2	R1_3	R1_4	R1_5	R1_6	R1_7	R1_8
------	------	------	------	------	------	------	------

R2

R2_1	R2_2	R2_3	R2_4	R2_5	R2_6	R2_7	R2_8
------	------	------	------	------	------	------	------

R2

R1_1	R1_2	R1_3	R1_4	R1_5	R1_6	R1_7	R1_8	R2_1	R2_2	R2_3	R2_4	R2_5	R2_6	R2_7	R2_8
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

3.5.2 Controllo di flusso

Si assume che le reti utilizzate possano portare ad errori di sequenza dovuti ai ritardi di trasporto delle reti stesse. Il meccanismo del controllo di flusso ha lo scopo di assicurare che i dati vengano ricevuti in sequenza, nello stesso ordine con il quale sono stati trasmessi.

Di seguito vengono presentati i campi utilizzati nel pacchetto di trasporto e le fasi previste per la gestione del controllo di flusso.

3.5.2.1 Campi utilizzati

Si prevedono nella testa del pacchetto di trasporto due campi distinti che riportano i numeri di sequenza del trasmittente (campo S) e del ricevente (campo R). Se non intervengono sequenze particolari, il valore dei contatori S e R evolve da 0 a 255 (con ricircolo).

Di seguito vengono indicati alcuni campi del pacchetto di trasporto, la cui struttura sarà descritta più avanti, utilizzati per il controllo di flusso.

Flag T/A	0 = pacchetto con dati applicativi 1 = pacchetto di trasporto
Flag R	1 = rimessa in passo del contatore di sequenza (impone al ricevente di impostare il proprio contatore interno R al valore del Byte S del pacchetto) 0 = il contatore R prosegue nella sequenza corrente
Flag A	1 = è richiesto il messaggio di conferma esplicita 0 = è sufficiente conferma "veloce"
Byte S	Numero di sequenza origine S
Byte R	Numero di sequenza destinatario R

3.5.2.2 Fase di start

Ad ogni apertura di connessione devono essere inizializzate le seguenti variabili:

Flag R=0

Contatore di sequenza origine S = 0

Contatore di sequenza destinazione R = 0.

3.5.2.3 Fase di normale funzionamento

Durante questa fase, lo scambio dati tra due generiche unità avviene con una sequenza standard di pacchetti: il primo con dati (che per comodità chiameremo nel seguito Pd) a cui seguirà quello di conferma (che chiameremo nel seguito Pc), che chiuderà lo scambio.

Un pacchetto si considera "con dati" quando le dimensioni del pacchetto sono maggiori di quelle definite per il pacchetto di conferma vuoto e non è un pacchetto di conferma firmata.

Il Pc può essere sfruttato per contenere dati, ma solo se l'unità che lo trasmette non è a sua volta in attesa di conferma.

Se il ricevente non può processare il messaggio (per esempio ha le code piene verso il suo livello superiore), può inviare un messaggio di sospensione a tempo determinato. Il trasmittente del pacchetto sospeso cancella l'attesa di conferma e ritrasmette il messaggio allo scadere del tempo di sospensione.

La ritrasmissione in questo caso viene gestita esclusivamente dal livello di trasporto ed è indipendente dallo stato del livello Data-Link.

Durante il tempo di sospensione il trasmittente del pacchetto sospeso può ricevere dati che deve confermare con pacchetti di conferma semplice o firmata.

Se il trasmittente richiede una conferma firmata, il ricevente dovrà inviare un pacchetto dati di risposta contenente una firma di convalida.

In questa fase il trattamento del pacchetto ricevuto dal livello di trasporto dipende dallo stato in cui esso si trova, come descritto di seguito.

Stato di riposo

Stato di attesa di una richiesta di trasmissione dati da parte del livello applicativo, o di avviso di ricezione dati da parte del livello di rete.

A seconda dei casi lo stato successivo è quello di trasmissione o di ricezione.

Stato di trasmissione

Viene attivato uno scambio dati trasmettendo un pacchetto Pd, i cui valori delle sequenze origine e destinazione vengono valorizzati con i contatori S ed R interni.

Il trasmittente passa direttamente nello stato di attesa conferma.

Stato di attesa conferma

Si distinguono due casi:

A) È scaduto il time-out di attesa conferma.

Il time-out di attesa conferma può essere al più di 600 s.

Se non è stato superato il numero massimo di ritrasmissioni previste (al massimo 3) viene rinviato il pacchetto Pd, altrimenti viene trasmesso il pacchetto Pd con il flag R=1.

Se invece è stato già trasmesso il pacchetto Pd con il flag R=1, viene richiesta la chiusura e la riapertura della connessione end-to-end, e si torna nello stato di riposo.

B) Viene ricevuto un pacchetto Pc dal livello di rete.

Se il pacchetto Pc contiene dati, si passa direttamente nello stato di ricezione: la gestione dell'attesa conferma verrà effettuata all'uscita dallo stato di ricezione.

Viene esaminato il valore del numero di sequenza R del pacchetto e confrontato con il contatore interno S incrementato di 1 modulo 256.

Se i due sono uguali, il valore del contatore interno S viene incrementato di 1 modulo 256, lo scambio si considera andato a buon fine e si passa nello stato di riposo.

In caso contrario, il pacchetto non viene preso in considerazione e ci si comporta come se fosse scaduto il time-out di attesa conferma.

Stato di ricezione

È lo stato in cui si analizza un pacchetto Pd o un pacchetto Pc con dati, ricevuto dal livello di rete.

Si confronta il valore del numero di sequenza S del pacchetto ricevuto e del contatore interno R. Se i due sono uguali ($S_p == R_{int}$), si incrementa di 1 modulo 256 il valore assunto dal contatore interno R e viene trasmesso un pacchetto Pc, del tipo indicato:

Pacchetto di conferma firmato: se esplicitamente richiesto.

Pacchetto di sospensione: in presenza di problemi di trattamento.

Pacchetto con dati: se non è stata richiesta conferma firmata e non si è in attesa di un pacchetto di conferma.

Pacchetto di conferma vuoto: negli altri casi.

Se i due valori sono diversi, si ignora il pacchetto dati ricevuto e viene inviato un pacchetto di conferma vuoto, mantenendo inalterato il valore del contatore interno R.

In entrambi i casi, i numeri di sequenza S e R del pacchetto di conferma Pc assumono il valore dei contatori interni.

Al termine, il ricevente ritorna allo stato chiamante.

3.5.3 Sicurezza delle comunicazioni

Per la cifratura e l'autenticazione dei dati la presente Norma contempla l'utilizzo di algoritmi a chiave segreta e di tipo "block cypher".

Si prevedono quattro livelli di protezione:

livello 0 trasmissione dei dati in chiaro

livello 1 trasmissione dei dati in chiaro con autenticazione dei dati e riconoscimento reciproco, con algoritmo che utilizza chiavi a 64 bit, elaborando blocchi dati di 64 bit, e con MAC a 32 bit (ad esempio FEAL-N o DEA).

livello 2 trasmissione dei dati cifrata con autenticazione dei dati e riconoscimento reciproco, con algoritmo che utilizza chiavi a 64 bit, elaborando blocchi dati di 64 bit, e con MAC a 32 bit (ad esempio FEAL-N o DEA).

livello 3 trasmissione dei dati cifrata con autenticazione dei dati e riconoscimento reciproco, con algoritmo che utilizza chiavi o blocchi dati o MAC di lunghezze maggiori di quelle specificate per il livello 2. L'algoritmo di cifratura utilizzato deve essere necessariamente pubblicato in una norma tecnica ISO-IEC e sottoposta al CEI per il recepimento.

Il livello di protezione di ciascuna connessione end-to-end è configurata localmente in CSC e CA e definisce il livello massimo di protezione disponibile sui dati. Il livello massimo di protezione è ad arbitrio del costruttore e va dichiarato dallo stesso nel profilo di centrale interoperabile. Il livello di protezione configurato è indicato nella testata di ciascun pacchetto contenente dei dati. Se uno dei due interlocutori, per qualunque motivo, non è in grado di operare al livello di protezione configurato, può procedere in modalità degradata (corrispondente al livello 0) segnalando il suo stato all'altro interlocutore tramite il bit S nel byte FLAG previsto nella testata del pacchetto. Questa situazione, per esempio, si ha in assenza o corruzione della chiave master o per mancanza o guasto dei dispositivi di cifratura.

Per la modalità di utilizzo nella cifratura dei dati, si fa riferimento alla Norma ISO 10116 (modes of operation for a "n" bit block cypher algorithm) e in particolare viene utilizzato il modo CBC come descritto nell'Appendice B della presente Norma.

Per la modalità di utilizzo nell'autenticazione dei dati, si fa riferimento alla Norma ISO 9797-1:1999, come descritto nell'Appendice C della presente Norma.

In Allegato è descritto l'algoritmo FEAL, già incluso nella Norma 79-7 prima edizione (ottobre 1994), utilizzabile per i livelli di protezione 1 e 2.

Nel profilo di centrale interoperabile devono essere indicati dal costruttore della CA, l'algoritmo utilizzato e le eventuali opzioni implementative previste dall'algoritmo stesso.

In caso di implementazione del livello di protezione 3, nel profilo di centrale interoperabile devono essere specificate la lunghezza delle chiavi, la dimensione del blocco dati, la dimensione del MAC e deve essere allegata la documentazione dell'algoritmo utilizzato o un riferimento normativo.

Gli algoritmi di cifratura utilizzabili per il livello 3 sono specificati nella Norma 79-5.3. Per il recepimento di nuovi algoritmi di cifratura vanno seguite le stesse procedure indicate per le EPT, descritte nella Norma 79-5.2.

Per il riconoscimento reciproco degli interlocutori si fa riferimento alla Norma ISO 9798 Parte 2 e si adotta il primo meccanismo descritto con la variante dell'inserimento del cambio della chiave dinamica. La procedura di riconoscimento è descritta più avanti.

Le procedure di protezione utilizzate prevedono l'uso di una chiave master (MK) presente nella CA e nel CSC.

La generazione e l'inserimento nel sistema della chiave master MK non è contemplato dalla presente normativa e sarà realizzato con le modalità previste da ciascun costruttore e con le procedure amministrative definite a livello di progetto di sistema.

È prevista la possibilità di avere più chiavi master su CSC e CA e di attivare, durante il trasferimento della chiave dinamica, una delle chiavi master precedentemente inserite.

Riguardo la cifratura dei dati, si chiarisce che essa si applica esclusivamente alla parte indicata come "dati da trasportare" nei pacchetti con dati di "tipo applicativo" (T/A = 0).

Per i pacchetti con dati di "tipo trasporto" (T/A = 1), il byte denominato "ID" non deve essere cifrato.

3.5.4 Fasi operative del livello di trasporto

3.5.4.1 Apertura della connessione

Lo scambio dati tra CSC e CA non può avvenire se prima non è stata attivata la connessione "end-to-end" e non è stata effettuata una operazione di identificazione. L'apertura di un collegamento tra CSC e CA è segnalata al livello di trasporto dai livelli inferiori. L' "apertura del collegamento" deve essere intesa come attivazione dei livelli 1, 2 e 3.

Il livello di trasporto richiede al livello di rete la connessione, indicando il tipo di linea (primaria o di riserva). Il livello di rete effettua il tentativo di connessione sul tipo di linea indicata e può rispondere con l'informazione di connessione attiva o di connessione fallita. Nel caso di connessione fallita il livello di rete continuerà comunque ad effettuare i tentativi di connessione.

L'informazione di connessione attiva può quindi essere segnalata anche successivamente dopo una connessione fallita. Una volta ricevuta l'informazione di connessione attiva, il livello di trasporto inizializza i parametri legati alla connessione tra le due unità, di seguito indicati:

- lo stato della connessione (primaria o secondaria);
- le variabili tipo di trattamento (normale, identificazione, cambio chiave, ecc.) e fase in corso;
- numero di sequenza di origine (campo S);
- numero di sequenza di destinazione (campo R);
- il livello di protezione (0 = trasmissione in chiaro, 1 = trasmissione in chiaro con autenticazione, 2 o 3 = trasmissione cifrata con autenticazione), a seconda della configurazione.

Quindi il CSC inizia la procedura di scambio chiave o identificazione, descritte in 3.5.4.3 e 3.5.4.6.

Distinguiamo le procedure da seguire in caso di connessione su linea primaria o di riserva; in entrambi i casi faremo riferimento a due stati:

prima connessione: è la condizione in cui il CSC non ha mai identificato la CA interessata, cioè non dispone di una chiave di sessione valida;

connessione ok dopo una connessione fallita: è la condizione in cui il CSC ha già identificato in precedenza la CA interessata (indipendentemente dal tipo di connessione) e pertanto dispone di una chiave di sessione valida per quella CA.

Connessione sulla linea primaria

Il livello di trasporto verifica lo stato della connessione ed esegue le seguenti procedure in funzione dello stato corrente:

— Prima connessione

Se il livello di protezione è diverso da zero, il lato CSC esegue la procedura di scambio chiavi e dell'identificazione. Se il livello di protezione è uguale a zero il lato CSC esegue comunque la procedura di identificazione omettendo le operazioni di cifratura (identificazione in chiaro)

— Connessione ok dopo una connessione fallita

Verifica lo stato della connessione della linea di riserva e ne chiede la sconnessione se attiva o fallita. Se il livello di protezione è diverso da zero, il CSC esegue la procedura di identificazione; se il livello di protezione è uguale a zero il lato CSC esegue comunque la procedura di identificazione omettendo le operazioni di cifratura (identificazione in chiaro).

Connessione sulla linea di riserva

Il livello di trasporto verifica lo stato delle connessioni ed esegue le seguenti procedure in funzione dello stato corrente:

— Prima connessione

Se il livello di protezione è diverso da zero, il lato CSC esegue la procedura di scambio chiavi e di identificazione. Se il livello di protezione è uguale a zero il lato CSC esegue comunque la procedura di identificazione omettendo le operazioni di cifratura (identificazione in chiaro).

— Connessione OK dopo una connessione fallita

Se il livello di protezione è diverso da zero, il lato CSC esegue la procedura di identificazione; se il livello di protezione è uguale a zero il lato CSC esegue comunque la procedura di identificazione omettendo le operazioni di cifratura (identificazione in chiaro).

3.5.4.2 Modalità per il cambio linea

Il sistema quando ha a disposizione due tipi di linee di comunicazione, primaria e riserva, privilegia l'uso della linea primaria ed utilizza la linea di riserva (commutata) solo in caso di necessità. Entrambi i sistemi (CSC e CA) utilizzano la linea di riserva solo in caso di caduta della linea primaria e nel caso in cui vi siano messaggi da inviare. La linea di riserva rimane attiva solo per il tempo necessario per lo "scarico" dei messaggi da inviare.

3.5.4.3 Trasferimento della chiave dinamica

Oltre ai casi specificati in precedenza relativi all'apertura della prima connessione, la procedura di trasferimento della chiave dinamica è svolta periodicamente su iniziativa del CSC ed è sempre seguita dalla procedura di identificazione, illustrata in 3.5.4.6. Il trasferimento della chiave dinamica può essere richiesto anche dalla CA (se il suo livello di protezione è >0), che invia al CSC un pacchetto di trasporto con ID=12.

La CA effettua questa richiesta quando deve trasferire dati ma non dispone di una chiave di sessione SK valida.

Quando riceve questa richiesta il CSC inizia la stessa procedura che svolge periodicamente di sua iniziativa. Se il livello di protezione della CA è 0 viene eseguita la sola procedura di identificazione.

La procedura, attivata su iniziativa del CSC o in risposta alla richiesta effettuata dalla CA, prevede:

- il CSC genera in modo casuale una nuova chiave ed emette una richiesta di trasferimento e cambio chiave;
- la CA risponde con un valore casuale R1 (in chiaro);
- il CSC rimanda il valore R1 seguito da un valore casuale R2, seguito dalla nuova chiave SK, il tutto cifrato sotto MK;
- la CA decifra il pacchetto e se riconosce R1 accetta temporaneamente la nuova SK e rimanda R2 cifrato sotto questa SK;
- il CSC decifra il pacchetto e se riconosce R2 considera SK temporaneamente confermata e rimanda R2 in chiaro;
- la CA confronta R2 con quello ricevuto nel passo "d" e se sono uguali considera SK confermata altrimenti mantiene attiva la chiave SK precedente;
- il CSC riceve il pacchetto di controllo di flusso trasmesso dalla CA a conferma del pacchetto ricevuto in fase f e considera la chiave SK confermata. Se il pacchetto non arriva regolarmente mantiene attiva la chiave SK precedente.

Per ogni centrale allarmi (CA) esiste una coppia di chiavi MK SK definita. Tali coppie possono essere modificate con procedure diverse nel tempo.

La scelta di avere chiavi uguali per tutte le CA è una scelta di sistema da concordare con l'utente.

La norma non definisce il periodo massimo per l'aggiornamento della chiave, in quanto questo valore di sistema va concordato con l'utente.

3.5.4.4 Errori nella procedura di trasferimento chiavi

Una situazione di errore durante una sequenza di trasferimento delle chiavi è segnalata con l'invio di un pacchetto di controllo (con ID = 255) che riporta nel campo dati il codice di identificazione del pacchetto rifiutato seguito dal codice dell'errore.

I casi possibili sono i seguenti:

errori in fase b

ID segnalato = 1 e codice di errore = 4

È segnalato dalla CA se essa opera a livello di protezione = 0, oppure manca o è corrotta la chiave master oppure mancano i mezzi di cifratura. La CA ignora il comando e il CSC rilancia la richiesta e, se la condizione persiste, passa in funzionamento degradato.

ID segnalato = 13 e codice di errore = 5

È segnalato dalla CA se essa opera a livello di protezione = 0, oppure se mancano i mezzi di cifratura. La CA ignora il comando e il CSC rilancia la richiesta e, se la condizione persiste, passa in funzionamento degradato.

ID segnalato = 13 e codice di errore = 6

È segnalato dalla CA se la chiave master selezionata risulta non presente o corrotta. La CA ignora il comando e il CSC rilancia la richiesta selezionando una chiave master diversa e, se la condizione persiste, passa in funzionamento degradato.

errore di decifratura in fase d

ID segnalato = 3 e codice errore = 1 La CA abortisce la procedura e il CSC rilancia il trasferimento. Se la condizione persiste la CA e il CSC considerano la chiave MK non più valida e passano in funzionamento degradato.

errore di decifratura in fase e

ID segnalato = 4 e codice errore = 2

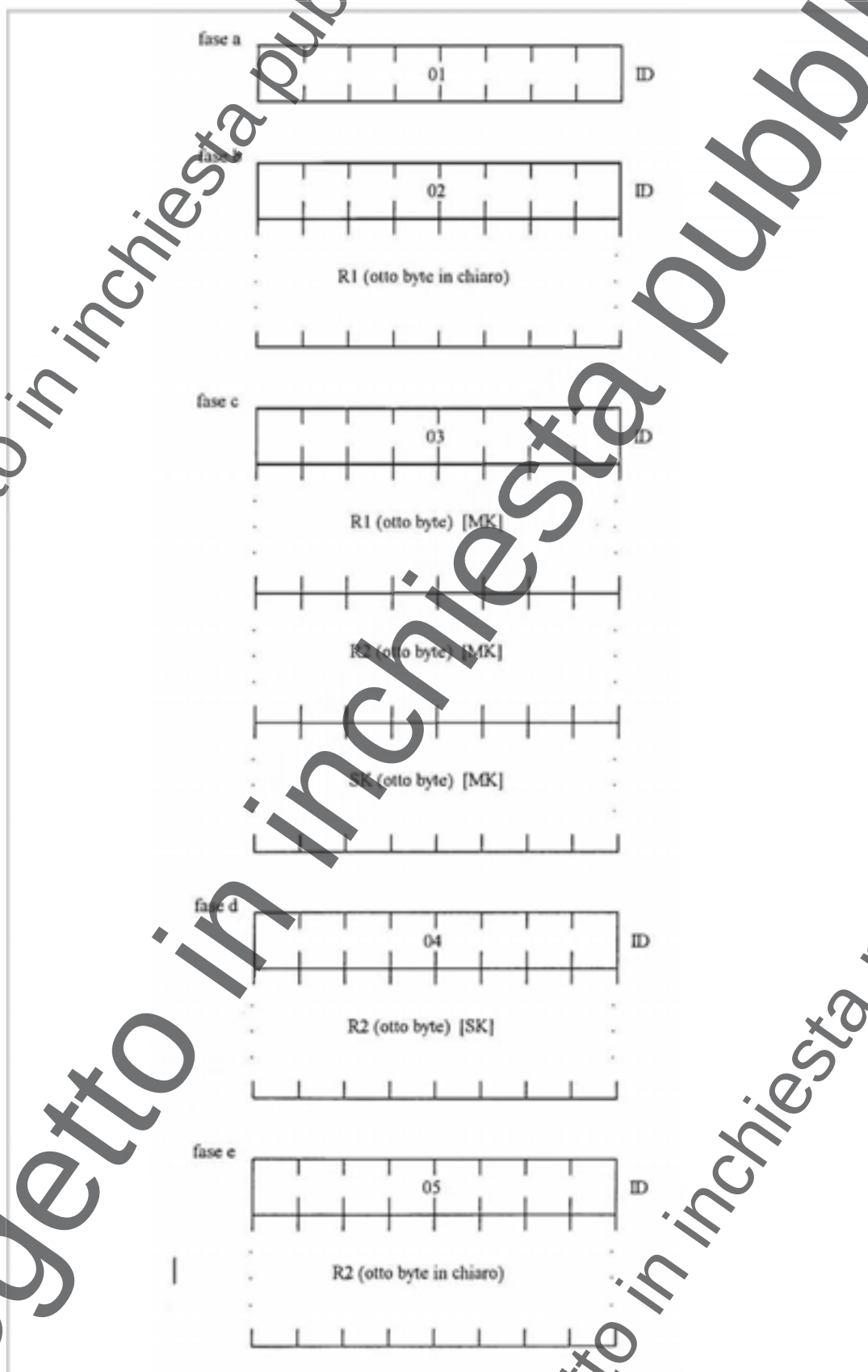
La CA abortisce la procedura e il CSC rilancia il trasferimento. Se la condizione persiste passano in funzionamento degradato.

errore di ricezione in fase f

ID segnalato = 5 e codice errore = 3

La CA abortisce la procedura e il CSC rilancia il trasferimento. Se la condizione persiste passano in funzionamento degradato.

3.5.4.5 Struttura dei pacchetti scambiati



3.5.4.6 Identificazione

La procedura di identificazione è attivata su iniziativa del CSC ad ogni apertura di connessione e, ciclicamente, a connessione attiva.

La procedura di identificazione viene inoltre attivata dal CSC a seguito di una richiesta di identificazione ricevuta da una CA (ID=14).

La CA deve effettuare questa richiesta

- quando deve trasferire nuovi dati ma, pur avendo attivi i livelli 1, 2 e 3, non si ritiene identificata dal CSC
- allo scadere del timeout di identificazione
- nella modalità di collegamento di tipo B (vedi capitolo 3.6) allo scadere del timeout previsto per la procedura di verifica di esistenza in vita

Una procedura di identificazione segue sempre quella di trasferimento della chiave dinamica quando il livello di protezione è > 0 . Se il livello di protezione è 0, la procedura di identificazione deve essere eseguita omettendo le operazioni di cifratura specificate di seguito.

La procedura prevede:

- il CSC emette una richiesta di identificazione contenente un valore casuale R1 (in chiaro);
- la CA rimanda il valore R1 seguito da un valore casuale R2, il tutto cifrato sotto SK;
- il CSC decifra il pacchetto e, se riconosce R1, considera valida l'identificazione e rimanda R2 (in chiaro);
- la CA, se riconosce R2, considera l'identificazione valida.

Nel caso in cui un interlocutore non abbia gli elementi per cifrare (il livello configurato è 0, oppure manca o è corrotta la chiave master, oppure mancano i mezzi di cifratura), non vengono eseguite le operazioni di cifratura.

Con l'esclusione del livello 0, la mancata operazione di cifratura comporta il forzamento a 1 del bit S nel byte FLAG della testa del pacchetto di trasporto.

3.5.4.7 Errori durante la procedura di identificazione

Una situazione di errore durante una sequenza di identificazione è segnalata con l'invio di un pacchetto di controllo (con ID = 255) che riporta nel campo dati il codice di identificazione del pacchetto rifiutato, seguito dal codice dell'errore.

I casi possibili sono i seguenti:

chiave SK non installata

ID segnalato = 6 e codice errore = 1

Il CSC inizia la procedura di trasferimento chiavi.

errore di decifratura in fase b

ID segnalato = 7 e codice errore = 2

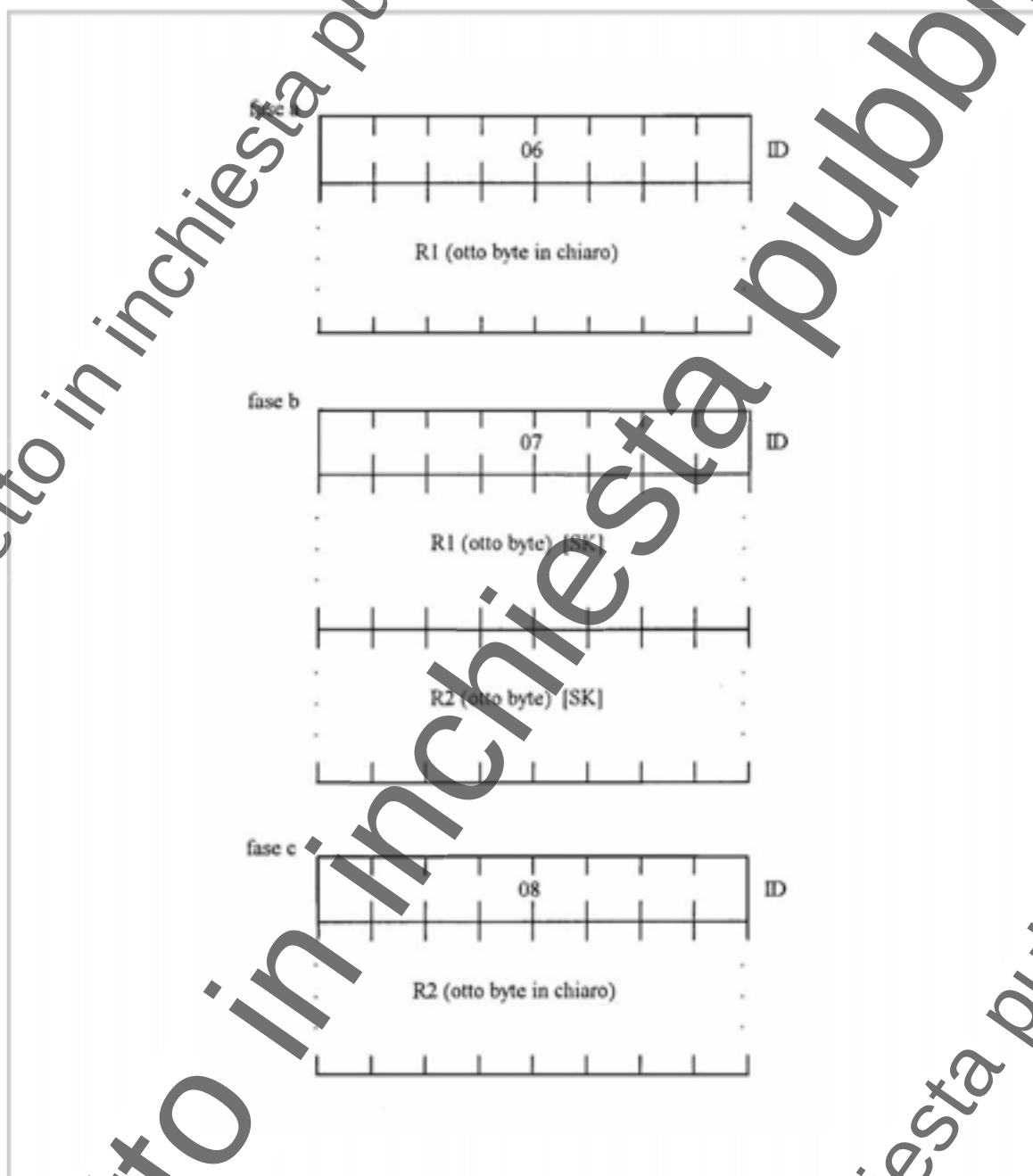
La CA abortisce la procedura e il CSC inizia la procedura di trasferimento delle chiavi.

errore di ricezione in fase c

ID segnalato = 8 e codice errore = 3

Il CSC inizia la procedura di trasferimento chiavi.

3.5.4.8 Struttura dei pacchetti scambiati



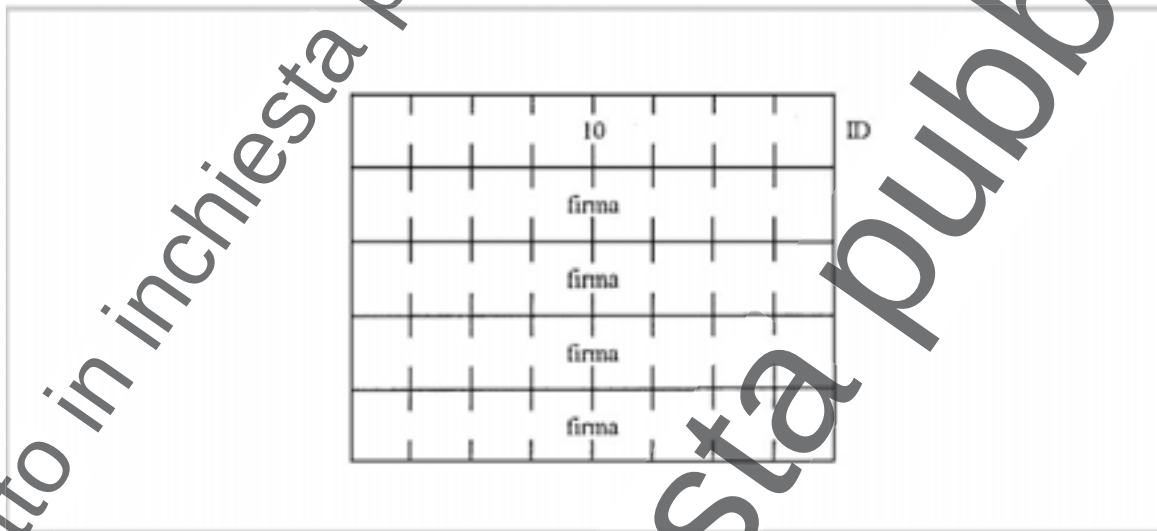
3.5.4.9 Conferma firmata

Chi trasmette un pacchetto dati con MAC (con dati cifrati o in chiaro) può richiedere al ricevente una risposta contenente una firma di convalida.

Per la costruzione di una risposta firmata si utilizza il seguente algoritmo: si prendono i 64 bit risultanti dal calcolo di verifica del MAC e si effettua una operazione di XOR con la variabile iniziale utilizzata nel calcolo del MAC (vedi Appendice C). Il risultato viene cifrato utilizzando la chiave SK e i 32 bit più significativi (quattro byte) del risultato della cifratura costituiscono la firma richiesta.

Chi ha trasmesso i dati verifica la firma di risposta e considera il pacchetto correttamente ricevuto solo se l'esito della verifica è positivo.

La struttura del pacchetto è:



3.5.4.10 Errore di decifratura di un pacchetto dati

Il ricevente di un pacchetto di dati con MAC (con dati cifrati o in chiaro) che non verifica corretto il MAC ricevuto rimanda al trasmittente un pacchetto di segnalazione (con ID = 11) per segnalare il disallineamento delle chiavi.

Il trasmittente considera il pacchetto non ricevuto.

Se il trasmittente è il CSC, esso inizia una procedura di trasferimento delle chiavi (vedi 3.5.4.2). Se il trasmittente è la CA, essa emette una richiesta di trasferimento chiave (con ID = 12) a cui il CSC risponde iniziando la procedura di trasferimento delle chiavi (vedi 3.5.4.2).

3.6 Tipo B

Il livello di trasporto specificato per il Tipo A si applica, senza alcuna variante, alle applicazioni di Tipo B su reti strutturate.

La connessione viene mantenuta aperta a livello di trasporto anche in assenza di una continua attività in linea del livello 2.

Il corretto stato della connessione, già controllato ai livelli inferiori nelle modalità specifiche alle varie soluzioni di rete, è verificato con frequenza più bassa tramite la periodica procedura di identificazione prevista dal livello 4.

Nel caso di segnalazione da parte dei livelli di rete di un abbattimento e successivo ripristino del collegamento con la CA, il CSC deve immediatamente iniziare la procedura di identificazione prevista dal livello 4.

Per quanto riguarda il controllo di flusso, nel caso di iniziativa di trasmissione contemporanea ciascun ricevente accetta i dati ricevuti e ritrasmette la conferma per gli stessi, ma non invia nuovi dati fino a quando non ha ricevuto dall'altro interlocutore la conferma al proprio precedente messaggio.

PROCEDURA DI COMUNICAZIONE CON TECNICA DI BYTE STUFFING

APPENDICE A

RICEZIONE

La Fig. A.1 mostra il diagramma a stati della procedura di ricezione dei messaggi che provvede ad eliminare i byte di controllo (DLE) inseriti in trasmissione per garantire la trasparenza del protocollo.

STATO 1 – IDLE

È lo stato di partenza (o di ripartenza) della procedura. Vengono ricevuti e scartati tutti i caratteri diversi da STX.

Se si riceve un carattere STX lo si considera il carattere di start del messaggio e si transisce nello stato 2 (RX).

STATO 2 – RX

È lo stato in cui si ricevono e si caricano i byte costituenti il messaggio.

Se si riceve il carattere ETX, lo si considera il carattere di fine messaggio e si transisce nello stato 3 (MESSAGE READY).

Se si riceve un carattere DLE, lo si scarta e si transisce nello stato 4 (CONTROL ELIMINATOR).

Se si riceve un carattere STX, questo è considerato un errore nella sequenza, lo si scarta e si transisce nello stato 5 (SEQUENCE ERROR).

Se non vengono ricevuti più caratteri per un tempo superiore ad un time-out di protezione (LONG TIME OUT, parametro configurabile compatibile con quelli di attesa risposta e attesa interrogazione), il collegamento si considera interrotto e si transisce nello stato 5 (SEQUENCE ERROR) per poter successivamente inizializzare la procedura.

STATO 3 – MESSAGE READY

È lo stato di transizione nel quale si giunge una volta terminata la ricezione di un messaggio completo e dal quale si riinizializza la procedura di ricezione.

STATO 4 – CONTROL ELIMINATOR

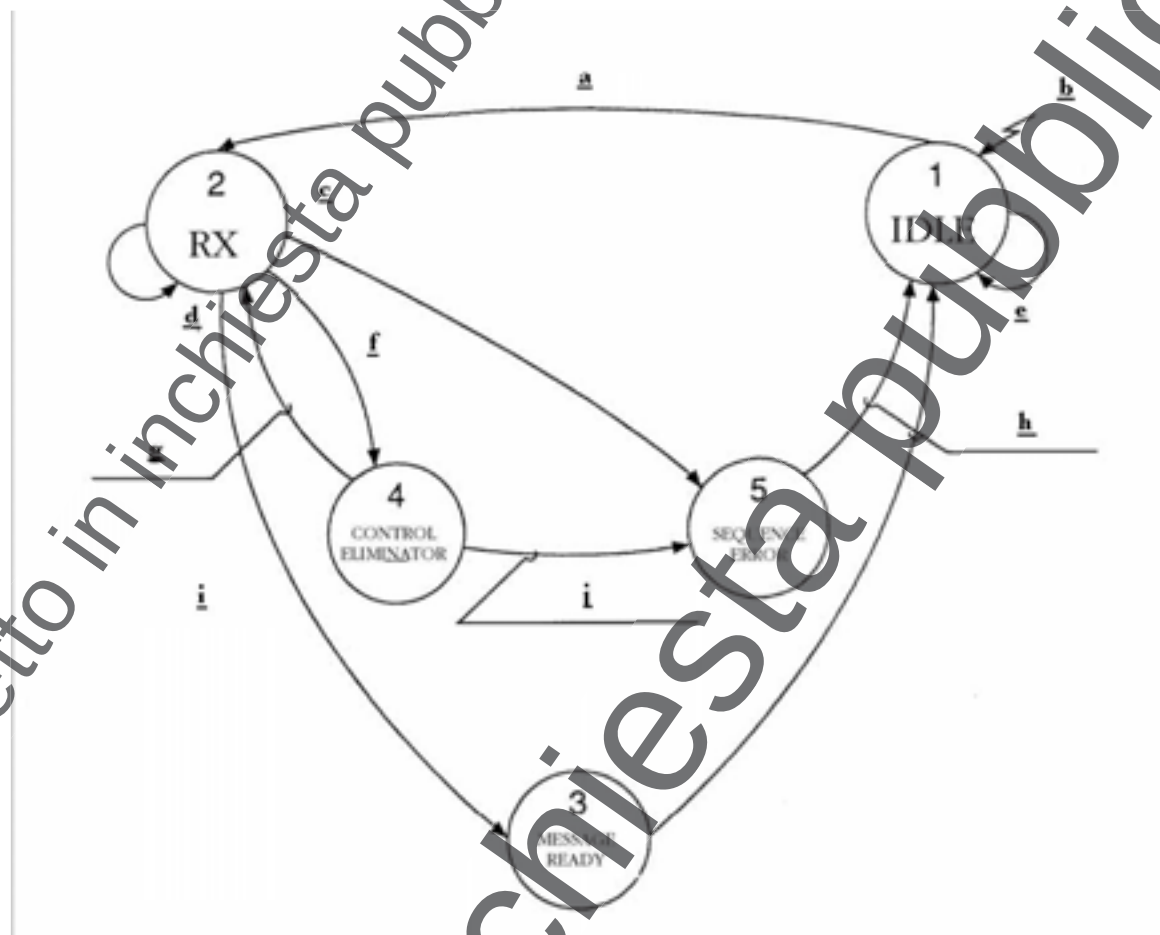
È lo stato nel quale si giunge dopo aver ricevuto un carattere di controllo (DLE) che precede i caratteri STX+hex 80, ETX+hex 80 e DLE+hex 80 situazione che si verifica quando STX, ETX e DLE fanno parte del messaggio in quanto dati da trasmettere. Pertanto in questo stato, se si ricevono i caratteri STX+hex 80 o ETX+hex 80 o DLE+hex 80, si sottrae hex 80 e si caricano nel buffer di ricezione, quindi si transisce nuovamente nello stato 2 (RX).

Se invece si riceve qualunque altro carattere, questo viene considerato un errore nella sequenza, esso viene scartato e la procedura transisce nello stato 5 (SEQUENCE ERROR).

Se non vengono ricevuti più caratteri per un tempo superiore ad un time-out di protezione (LONG TIME OUT), il collegamento si considera interrotto e si transisce nello stato 5 (SEQUENCE ERROR) per poter successivamente inizializzare la procedura.

STATO 5 - SEQUENCE ERROR

È lo stato nel quale la procedura transita se si è rilevato un errore nella sequenza di ricezione dei byte costituenti il messaggio o quando si interrompe il collegamento. La procedura di ricezione viene riinizializzata.



LEGENDA

a CH.RX = STX

b START RX

c LONG T.O. oppure CH.RX = STX

d CH.RX ≠ STX, DLE, ETX

e CH.RX ≠ STX

f CH.RX = DLE

g CH.RX = STX+hex 80, DLE+hex 80, ETX+hex 80

h RESTART RX

i CH.RX = ETX

j LONG T.O. oppure CH.RX ≠ STX+hex 80, DLE+hex 80, ETX+hex 80

Fig. A.1 Diagramma di stato della procedura di ricezione

Esempio (tutte le cifre in hex):

Si illustra, di seguito, un pacchetto di data link vuoto inviato dalla CA con Crc 16 e la sequenza di operazioni da eseguire per la corretta ricezione.

Dopo l'eliminazione del byte stuffing, dove tutte le sequenze 10 82 diventano 02, 10 83 diventano 03 e 10 90 diventano 10, si procede con le verifiche di CHKS e di tutti i campi componenti il pacchetto.

Ricevuto dal fisico

STX	LMSG	Flag	IND		CHKS			ETX
02	00	29	00	01	08	10	90	03

elimino STX ed ETX

LMSG	Flag	IND		CHKS		
00	29	00	01	08	10	90

elimino byte stuffing

LMSG	Flag	IND		CHKS		
00	29	00	01	08	10	

verifico correttezza CHKS

LMSG	Flag	IND	
00	29	00	01

e per ultimi controllo LMSG, Flag e IND.

TRASMISSIONE

Esempio di trasmissione da CSC con CRC2 (tutte le cifre in hex):

Ipotizziamo di ricevere dal livello superiore 9 byte di dati da trasmettere

Dati								
00	01	00	02	01	00	00	00	01

a questi aggiungo LMSG, Flag e IND

LMSG	Flag	IND		Dati								
09	09	00	01	00	01	00	02	01	00	00	00	01

calcolo il CHKS

LMSG	Flag	IND		Dati								CHKS	
09	09	00	01	00	01	00	02	01	00	00	01	00	18

inserisco il byte stuffing

LMSG	Flag	IND		Dati										CHKS	
09	09	00	01	00	01	00	10	82	01	00	00	00	01	00	18

aggiungo testa e coda

STX	LMSG	Flag	IND		Dati										CHKS		ETX
02	09	09	00	01	00	01	00	10	82	01	00	00	01	00	18	03	

Questo pacchetto può essere inviato sul fisico.

CONTROLLO DI FLUSSO

APPENDICE B

Di seguito viene presentato un diagramma che schematizza le possibili transizioni tra gli stati individuati per la gestione del controllo di flusso, implementato nel livello di trasporto.

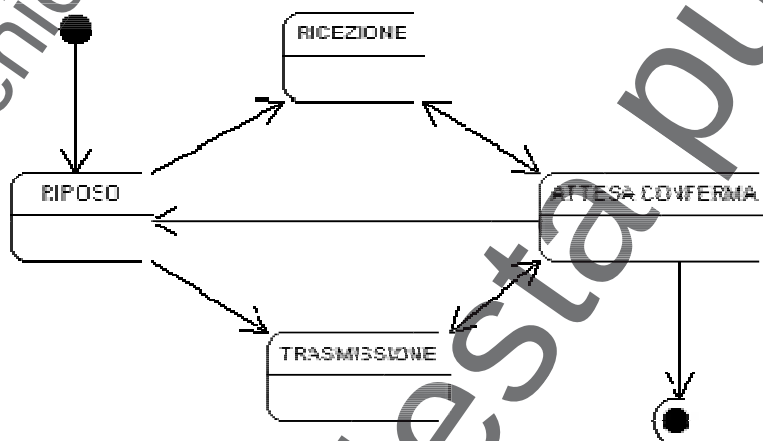
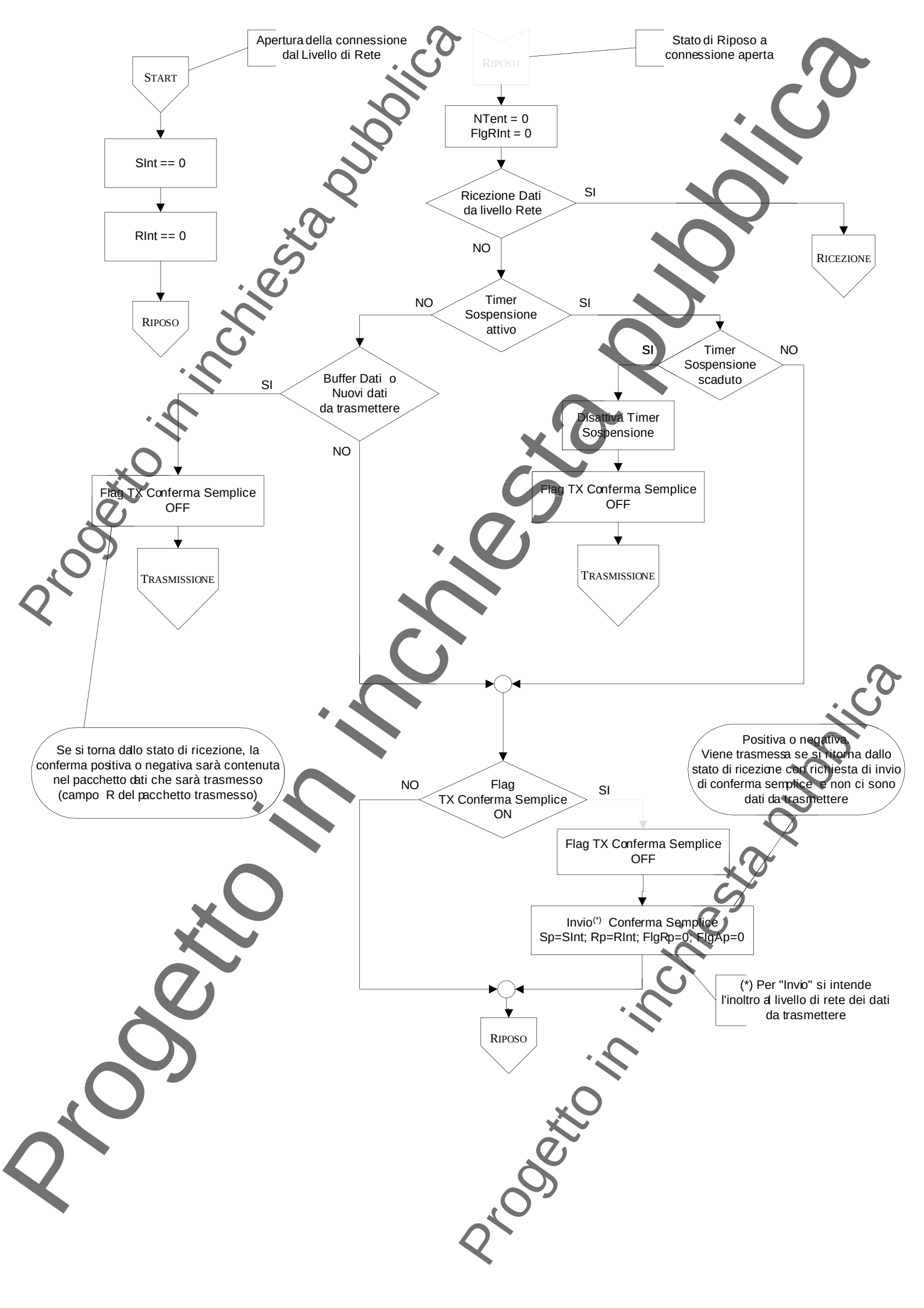


Figura 3: Diagramma di stato per la gestione del controllo di flusso.

Diagrammi logici per la gestione del controllo di flusso

I diagrammi riportati di seguito fanno riferimento agli stati del diagramma presentato sopra, descrivendone le logiche di transizione.



In questo stato vengono trattati i pacchetti con dati, applicativi o di trasporto, ricevuti nello stato di Riposo o di Attesa Conferma

RICEZIONE

Flag TX Conferma Semplice
Flag TX Conferma Firmata
Flag TX Sospensione
OFF

Richiesta di Resync
dal trasmettente ?

NO
SI

FlgRp == 1

Rint = Sp

Conferma negativa

Conferma positiva

SI
NO

Sp == Rint

Rint = Rint + 1
(MOD 256)

E' richiesta conferma firmata ?

SI

NO

FlgAp == 1

Flag TX Conferma Firmata
ON

Dati Applicativi

Dati Trasporto

Flag A/T
(Tipo Dati)

SI

NO

Coda applicativo libera

Invio Dati RX
a livello Applicativo

Flag TX Sospensione
ON

Elaborazione
Dati RX con
eventuale
preparazione dati
trasporto da
trasmettere

SI

NO

Coda Trasporto libera

Flag TX Sospensione
ON

NO
SI

Flag TX Sospensione
ON

NO
SI

Flag TX Conferma firmata
ON

Flag TX Conferma Semplice
ON

Invio(*) Conferma Firmata
Sp=SInt; Rp=RInt; FlgRp=0; FlgAp=0

Solo positiva

Invio(*) Pacchetto Sospensione
Sp=SInt; Rp=RInt; FlgRp=0; FlgAp=0

Conferma positiva
perchè pacchetto in
giusta sequenza ma
scartato per problemi
di trattamento

La conferma positiva o negativa, sarà inviata in un pacchetto vuoto o con dati se si torna allosato di riposo o con la eventuale ritrasmissione del pacchetto in attesa di conferma

(*) Per "Invio" si intende l'inoltro al livello di rete dei dati da trasmettere

Attesa conferma

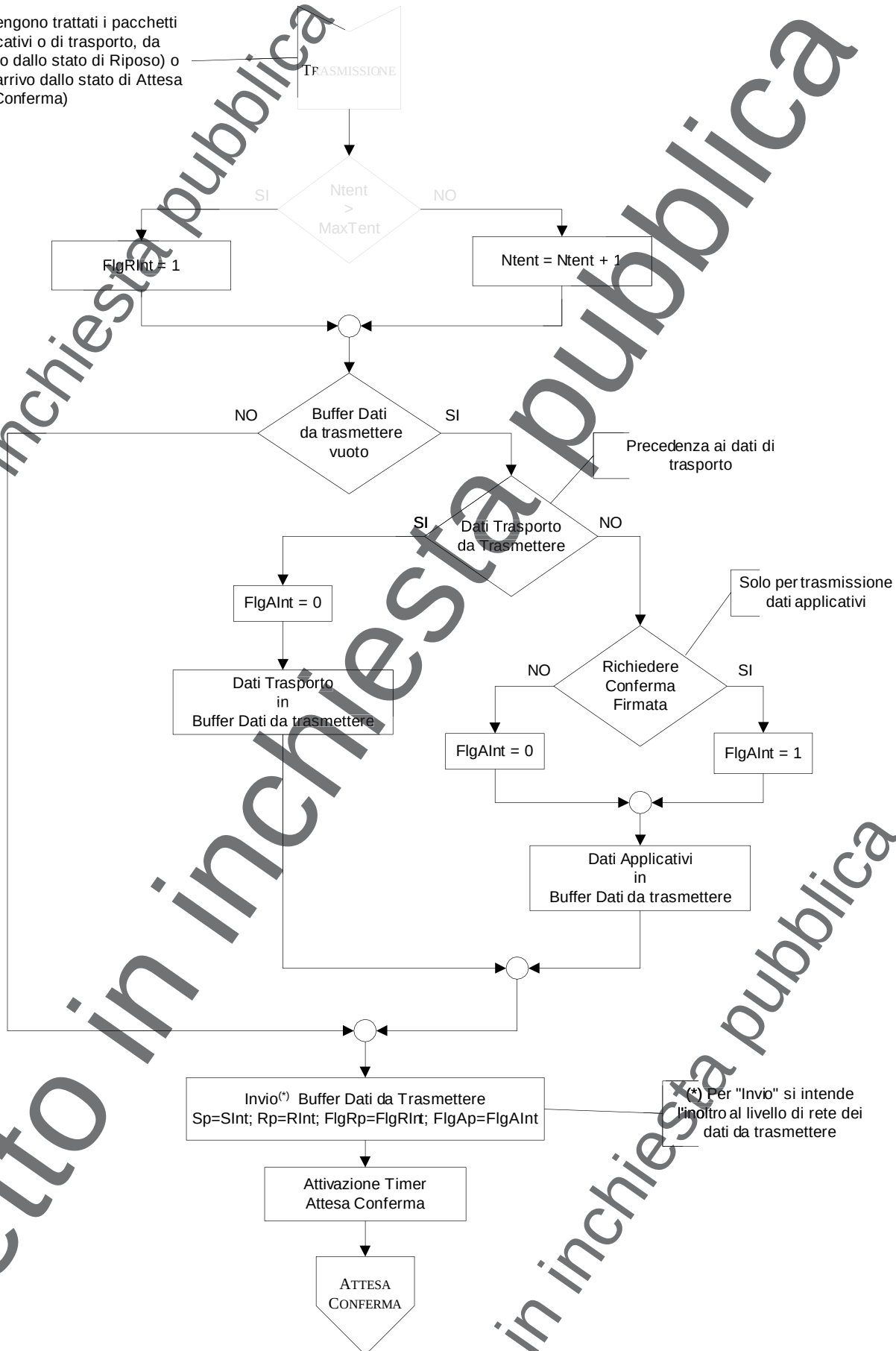
Riposo

Stato Provenienza

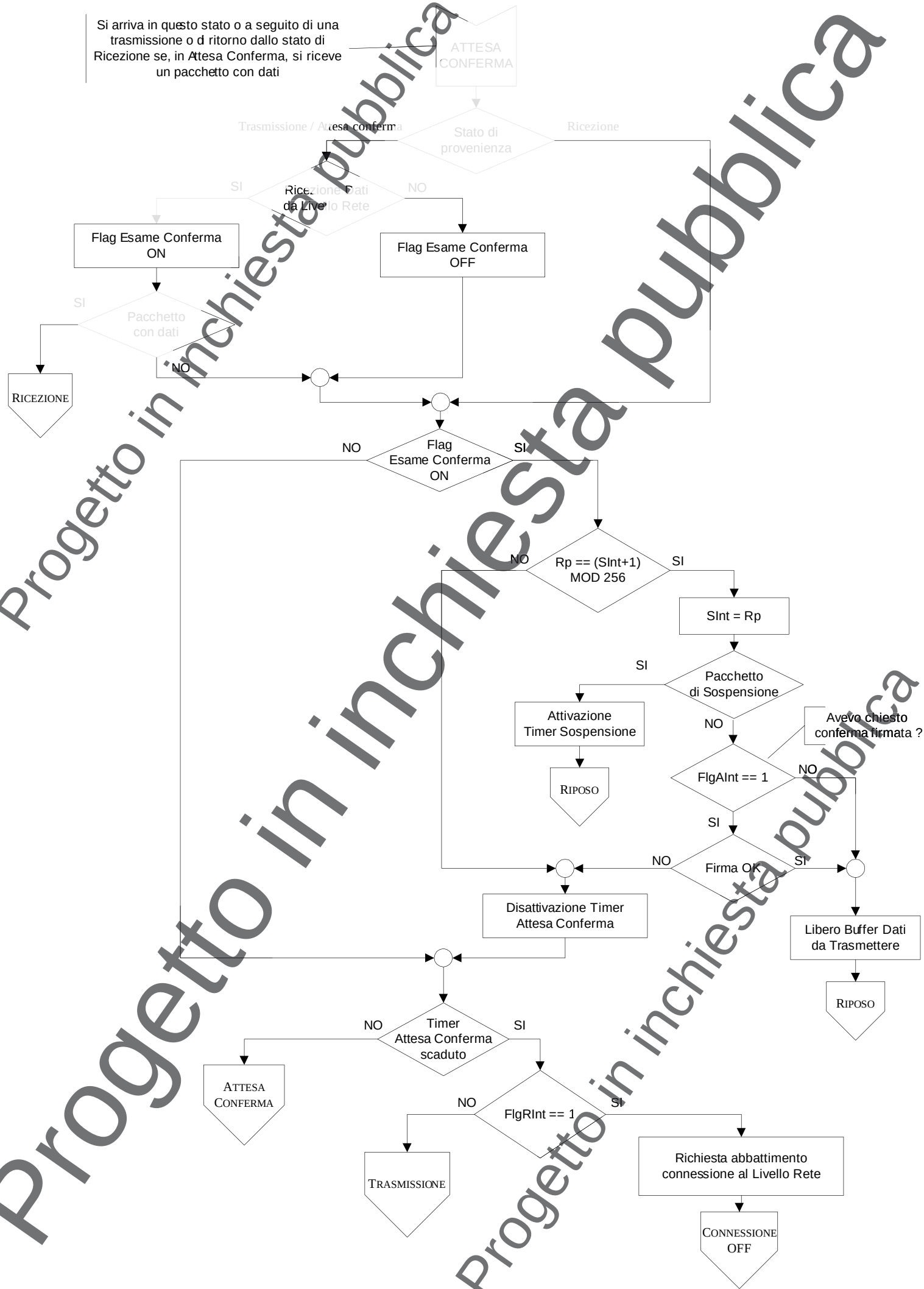
ATTESA
CONFERMA

RIPOSO

In questo stato vengono trattati i pacchetti con dati, applicativi o di trasporto, da trasmettere (arrivo dallo stato di Riposo) o da ritrasmettere (arrivo dallo stato di Attesa Conferma)



Si arriva in questo stato o a seguito di una trasmissione o d ritorno dallo stato di Ricezione se, in Atesa Conferma, si riceve un pacchetto con dati



Esempi di sequenze di trasmissione

CASO 1: Sequenza base senza errori.

Prima A invia un pacchetto dati e B invia la conferma senza dati; poi B invia un pacchetto dati e A la rispettiva conferma.

Pd: pacchetto dati

Pc: pacchetto di conferma ricezione dati

Sint-rx = Numero di sequenza Origine del Ricevente

Rint-rx = Numero di sequenza Destinazione del Ricevente

Sint-tx = Numero di sequenza Origine del Trasmittente

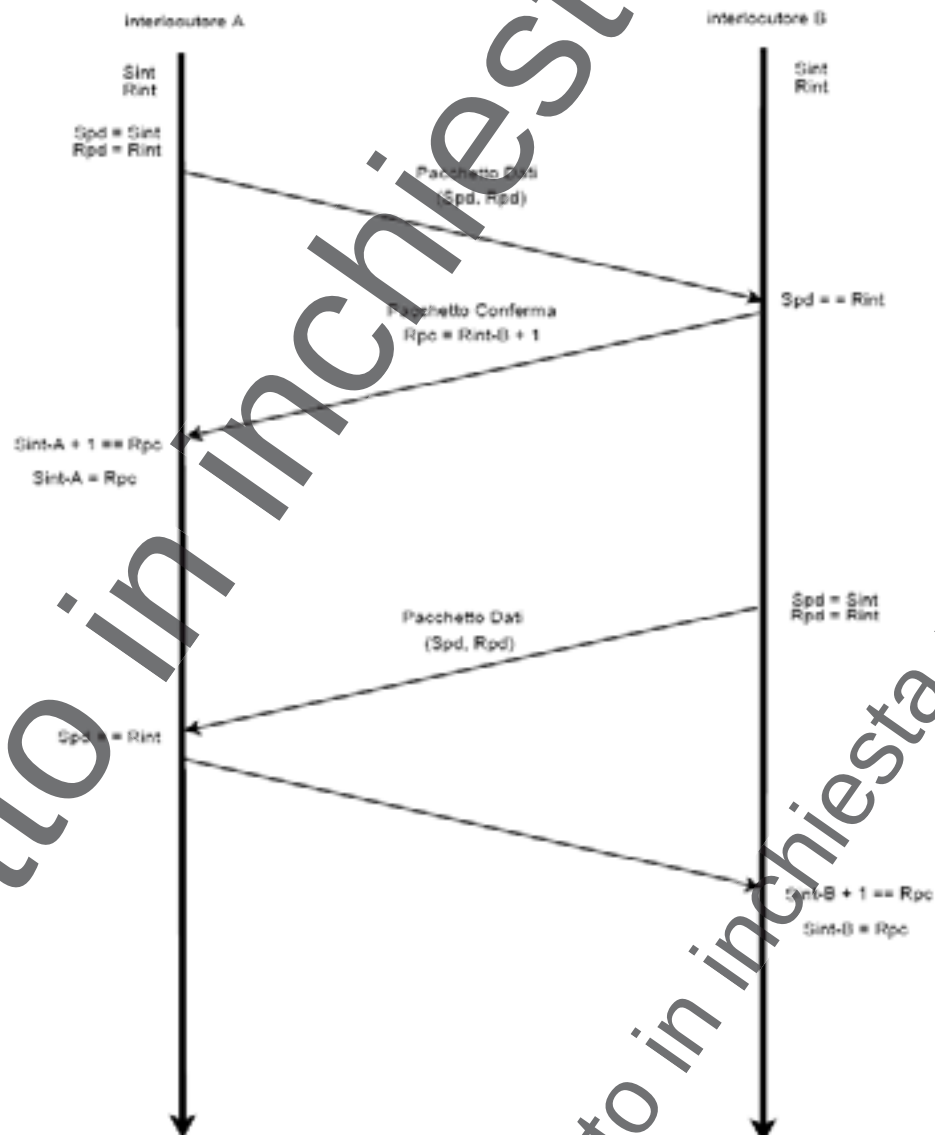
Rint-tx = Numero di sequenza Destinazione del Trasmittente

Spd = Numero di sequenza Origine pacchetto dati

Rpd = Numero di sequenza Destinazione pacchetto dati

Spc = Numero di sequenza Origine pacchetto di conferma

Rpc = Numero di sequenza Destinazione pacchetto di conferma



CASO 2: Trasmittente invia Pd e non riceve un Pc.
Allo scadere del time out di attesa del pacchetto di conferma valido il trasmittente reinvia il Pd.
Il ricevente rifiuta il Pd e ritrasmette il pacchetto di conferma precedente.

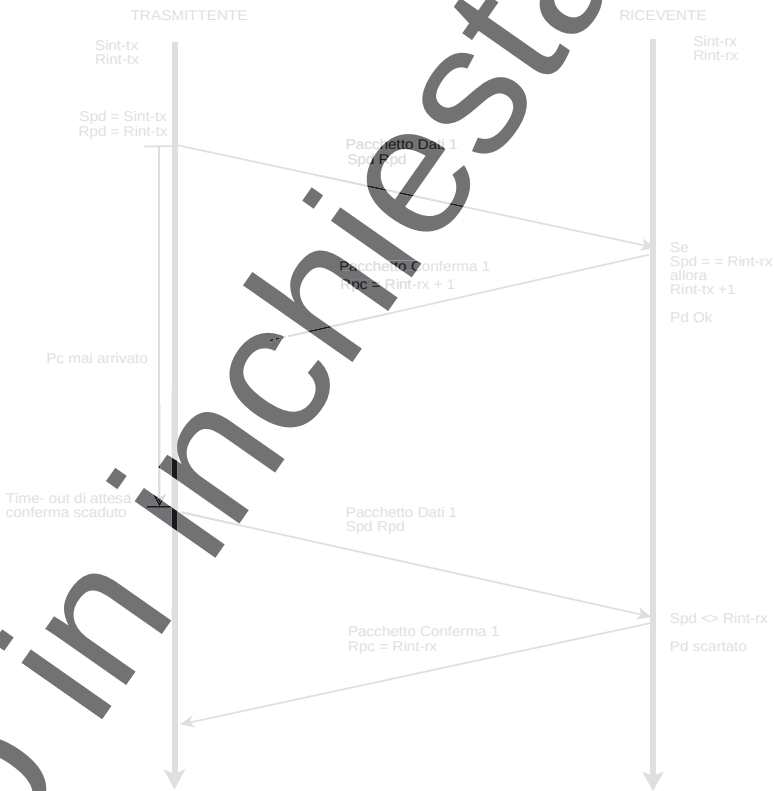
Pd: pacchetto dati
Pc: pacchetto di conferma ricezione dati

Sint-rx = Numero di sequenza Origine del Ricevente
Rint-rx = Numero di sequenza Destinazione del Ricevente

Sint-tx = Numero di sequenza Origine del Trasmittente
Rint-tx = Numero di sequenza Destinazione del Trasmittente

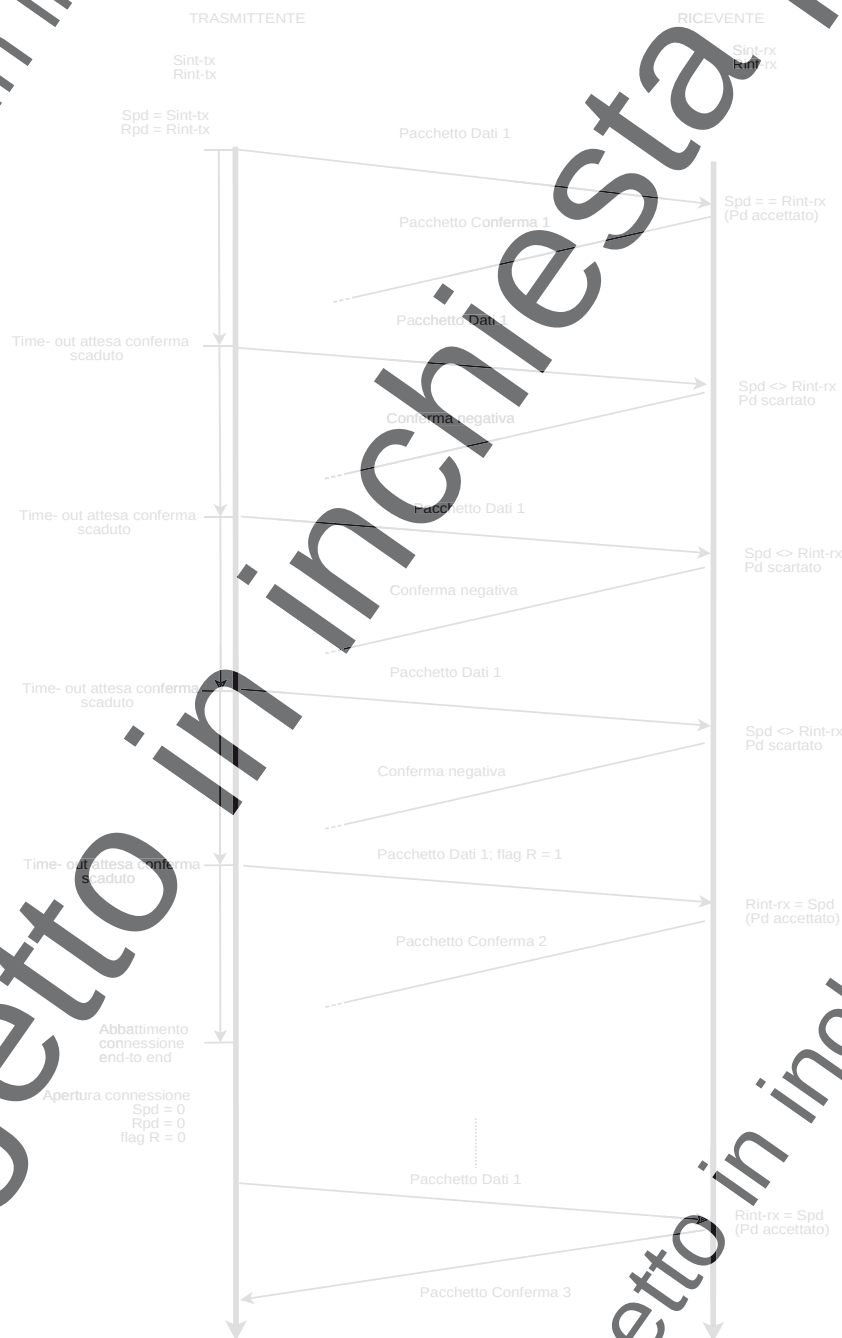
Spd = Numero di sequenza Origine pacchetto dati
Rpd = Numero di sequenza Destinazione pacchetto dati

Spc = Numero di sequenza Origine pacchetto di conferma
Rpc = Numero di sequenza Destinazione pacchetto di conferma



CASO 3: Trasmittente invia Pd. Allo scadere del time-out di attesa conferma ritrasmette il Pd.
Allo scadere del quarto time-out, il trasmittente reinvia il pacchetto dati settando il flag R a 1.
Se il messaggio di conferma non viene di nuovo ricevuto, il trasmittente abbate la connessione end-to-end, la rilancia e ripete la trasmissione di Pd.
In questo caso il ricevente riceve per 3 volte lo stesso Pd.

Pd: pacchetto dati
Pc: pacchetto di conferma ricezione dati
Sint-rx = Numero di sequenza Origine del Ricevente
Rint-rx = Numero di sequenza Destinazione del Ricevente
Sint-tx = Numero di sequenza Origine del Trasmittente
Rint-tx = Numero di sequenza Destinazione del Trasmittente
Spd = Numero di sequenza Origine pacchetto dati
Rpd = Numero di sequenza Destinazione pacchetto dati
Spc = Numero di sequenza Origine pacchetto di conferma
Rpc = Numero di sequenza Destinazione pacchetto di conferma



CASO 4: Trasmittente invia Pd e ricevente riceve un messaggio con Spd diverso dal valore del proprio contatore Rint-rx.

Il ricevente quindi invia un messaggio di conferma con Spd uguale al valore di Rint-rx e scarta il pacchetto dati ricevuto. Il trasmittente ritrasmette il Pd subito dopo la ricezione del pacchetto di conferma negativa.

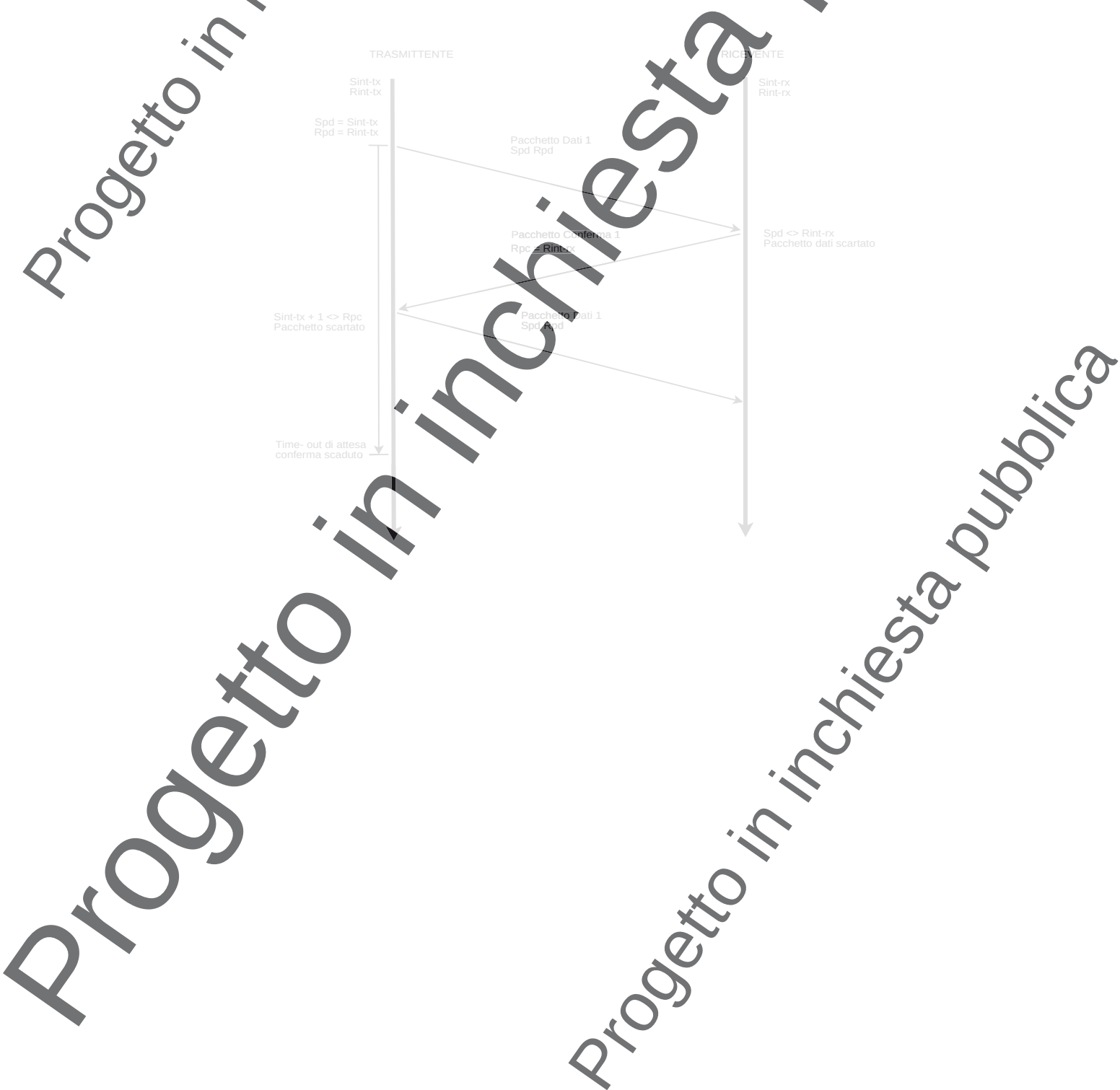
```
sequenceDiagram
    participant T as TRASMITTENTE
    participant R as RICEVENTE
    Note over T: Sint-tx  
Rint-tx
    T->>R: Spd = Sint-tx  
Rpd = Rint-tx
    Note over R: Spd R  
Rpd R
    R->>T: Spd R  
Rpd R
    Note over T: Spd = Sint-tx  
Rpd = Rint-tx
    T->>R: Spd = Sint-tx  
Rpd = Rint-tx
```

Sint-rx = Numero di sequenza Origine del Ricevente
Rint-rx = Numero di sequenza Destinazione del Ricevente

Sint-tx = Numero di sequenza Origine del Trasmittente
Rint-tx = Numero di sequenza Destinazione del Trasmittente

Spd = Numero di sequenza Origine pacchetto dati
Rpd = Numero di sequenza Destinazione pacchetto dati

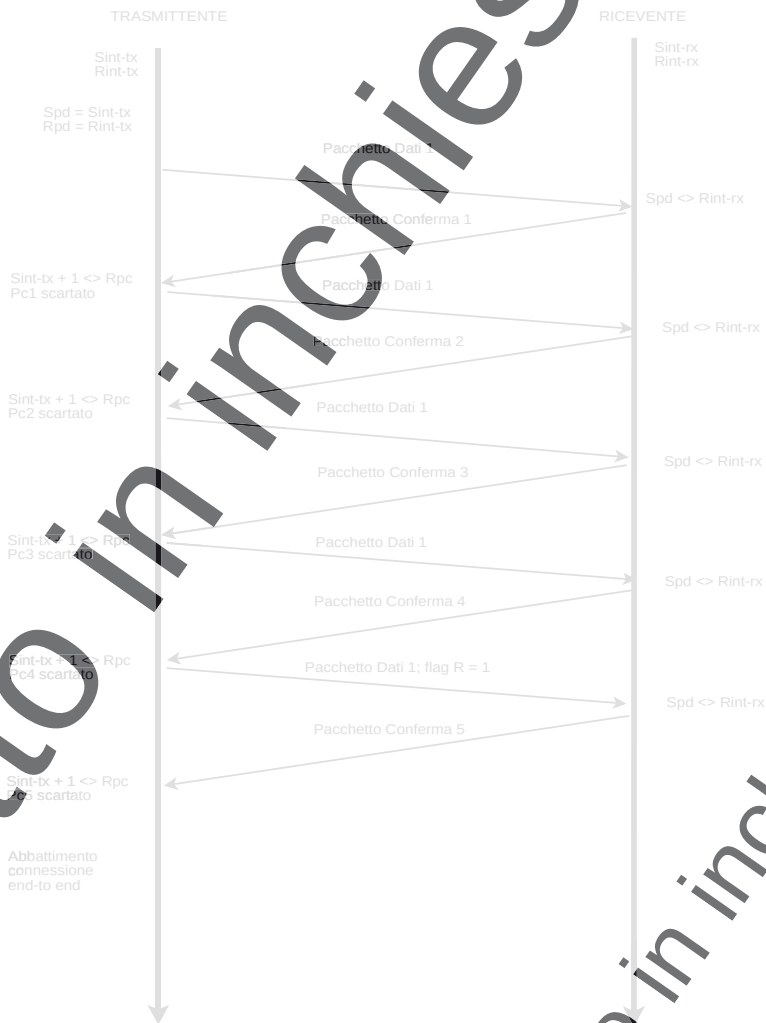
Spc = Numero di sequenza Origine pacchetto di conferma
Rpc = Numero di sequenza Destinazione pacchetto di conferma



CASO 5: Trasmittente invia Pd e il ricevente riceve un messaggio con Spd diverso dal proprio contatore Rint-rx. Il trasmittente scarta il Pc e ritrasmette subito il Pd.

Dopo il quarto messaggio di conferma errato ricevuto, il trasmittente reinvia il pacchetto dati settando il flag R a 1. Se il successivo messaggio di conferma ricevuto è ancora errato, il trasmittente abbatte la connessione end-to end e la rilancia.

Pd: pacchetto dati
Pc: pacchetto di conferma ricezione dati
Sint-rx = Numero di sequenza Origine del Ricevente
Rint-rx = Numero di sequenza Destinazione del Ricevente
Sint-tx = Numero di sequenza Origine del Trasmittente
Rint-tx = Numero di sequenza Destinazione del Trasmittente
Spd = Numero di sequenza Origine pacchetto dati
Rpd = Numero di sequenza Destinazione pacchetto dati
Spc = Numero di sequenza Origine pacchetto di conferma
Rpc = Numero di sequenza Destinazione pacchetto di conferma



CASO 6: Iniziativa di trasmissione contemporanea.
Ciascun ricevente accetta i dati ricevuti e ritrasmette la conferma per gli stessi, ma non invia nuovi dati fino a quando non ha ricevuto dall'altro interlocutore la conferma al proprio precedente messaggio.
Pc1 e Pc2 quindi non possono essere pacchetti di conferma con dati ma necessariamente vuoti.

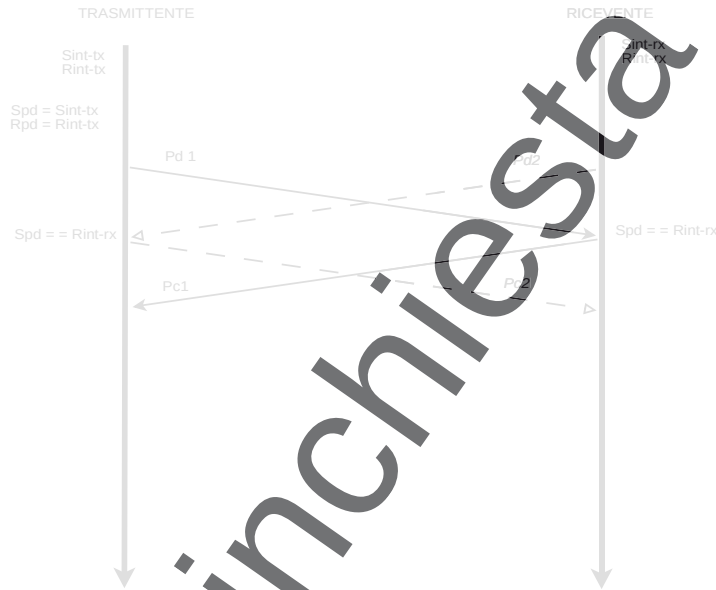
Pd: pacchetto dati
Pc: pacchetto di conferma ricezione dati

Sint-rx = Numero di sequenza Origine del Ricevente
Rint-rx = Numero di sequenza Destinazione del Ricevente

Sint-tx = Numero di sequenza Origine del Trasmittente
Rint-tx = Numero di sequenza Destinazione del Trasmittente

Spd = Numero di sequenza Origine pacchetto dati
Rpd = Numero di sequenza Destinazione pacchetto dati

Spc = Numero di sequenza Origine pacchetto di conferma
Rpc = Numero di sequenza Destinazione pacchetto di conferma



CIFRATURA DEI DATI

APPENDICE C

La modalità di cifratura dei dati fa riferimento alla Norma ISO 10116 :2006(E), "Cipher Block Chaining (CBC) mode".

Di seguito viene illustrato il procedimento, considerando la dimensione del blocco dati $n=64$ bit utilizzata per il livello di protezione 2.

I dati vengono divisi in blocchi da 64 bit. Ogni blocco di dati viene messo in XOR con i 64 bit risultanti dall'operazione di cifratura del blocco precedente (o con un vettore di inizializzazione se si tratta del primo blocco) e quindi cifrato con la chiave di cifratura (MK o SK, in funzione del contesto). Il risultato di tale operazione è il dato da inviare in linea.

L'ultimo blocco viene trattato in modo diverso: il risultato della cifratura del blocco precedente viene cifrato e il risultato della cifratura viene messo in XOR con l'ultimo blocco dati (accostando i bit a sinistra e completando con 0 a destra se i bit sono meno di 64). Vengono quindi trasmessi in linea tanti bit (a partire dai più significativi) quanti erano i bit dell'ultimo blocco.

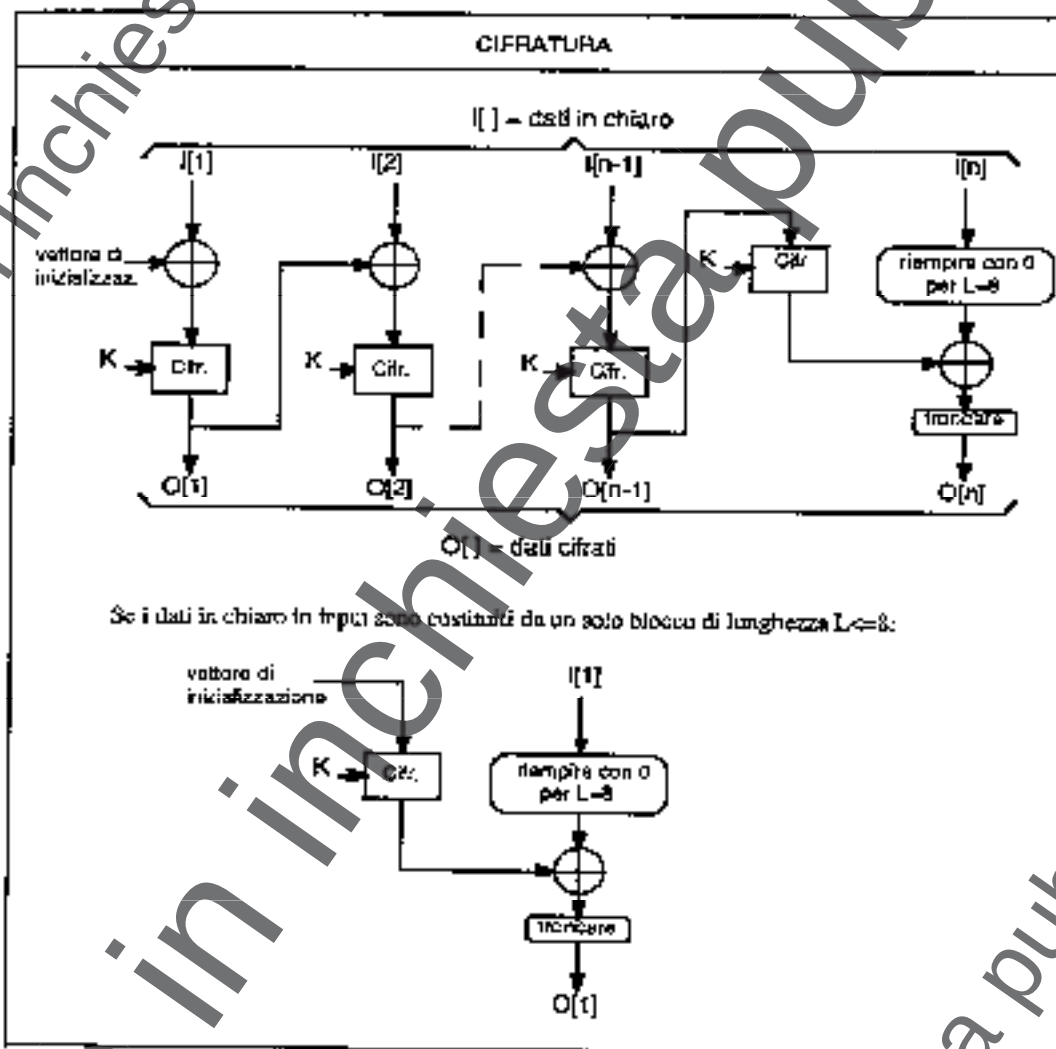
In ricezione i blocchi ricevuti vengono decifrati e il risultato viene messo in XOR con il blocco precedente (o con il vettore di inizializzazione se si tratta del primo blocco). Il risultato è il dato in chiaro.

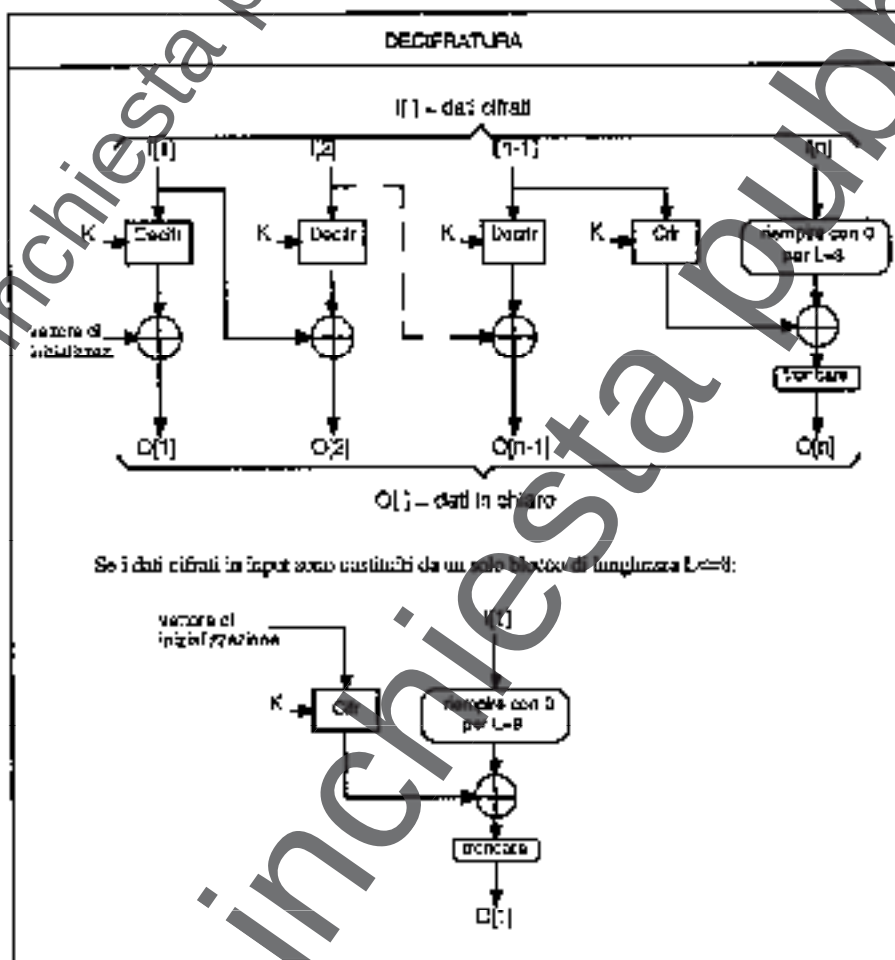
L'ultimo blocco si ricava cifrando il blocco precedente ed eseguendo lo XOR fra il risultato della cifratura e i bit ricevuti (accostati a sinistra). Vengono quindi utilizzati tanti bit (a partire dai più significativi) quanti erano i bit dell'ultimo blocco ricevuto.

Il vettore di inizializzazione è sempre costituito dal contenuto dei contatori S e R della testa del pacchetto di trasporto ripetuto 4 volte in modo da formare una stringa di 64 bit: SRSRSRSR, in modo identico a quanto descritto nell'Appendice C per il calcolo del MAC.

Cifratura dei dati

Con riferimento all'Appendice B vengono di seguito riportati gli schemi che evidenziano le operazioni di cifratura-decifratura.





CODICE DI AUTENTICAZIONE MESSAGGIO (MAC)

APPENDICE D

È utilizzato per validare il contenuto dei pacchetti dati applicativi per livelli di protezione maggiori di 0. Il processo di autenticazione prevede di utilizzare una variabile iniziale nota solo agli interlocutori. Tale variabile entra nel processo di calcolo del MAC e ne condiziona il risultato.

Tale variabile è costituita dal contenuto dei contatori S e R della testa del pacchetto di trasporto. Per i livelli di protezione 1 e 2 tale contenuto viene ripetuto per quattro volte in modo da formare una stringa da 64 bit:

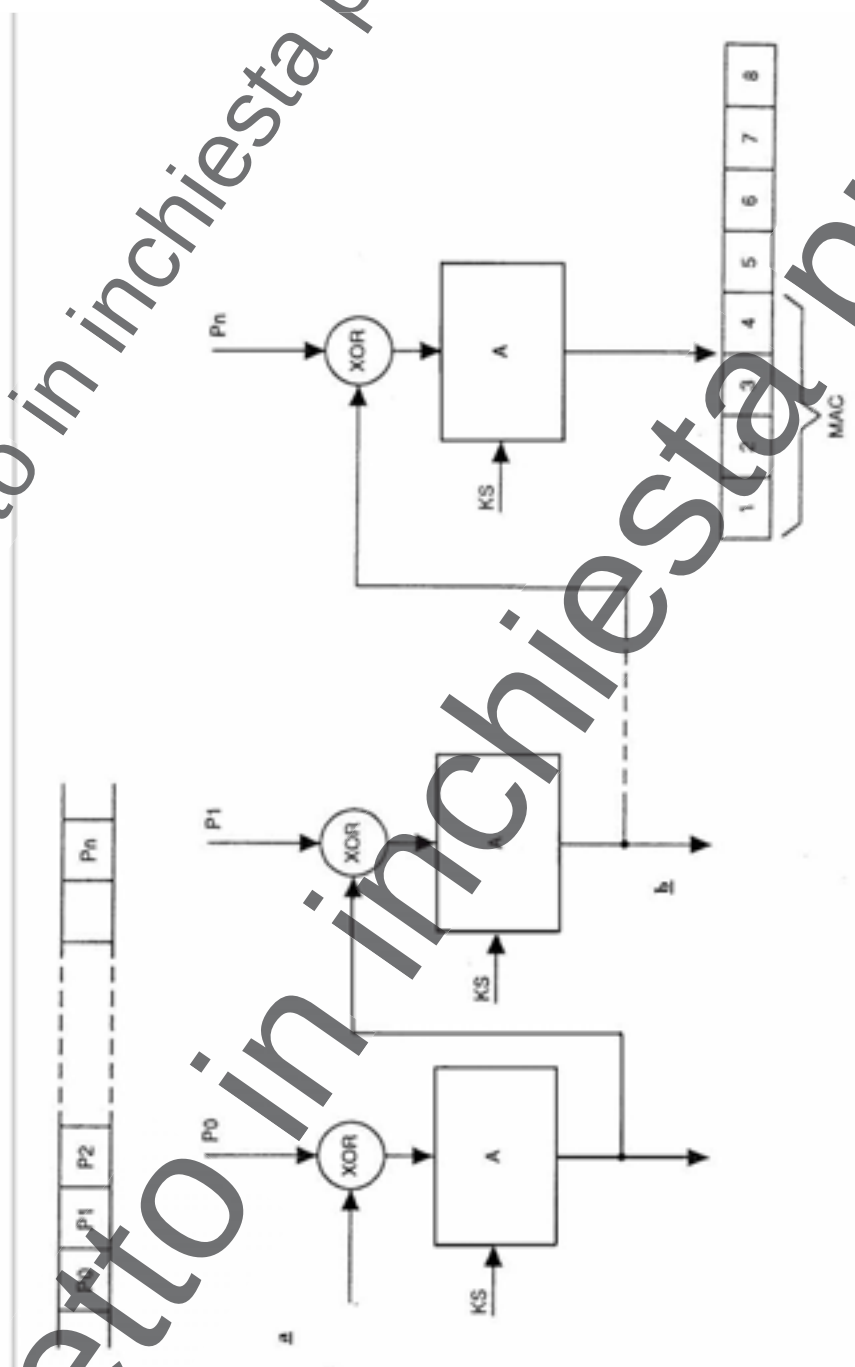
S R S R S R S R

Il procedimento di calcolo consiste nel suddividere i dati in blocchi da 64 bit. Il primo blocco viene messo in XOR con la stringa variabile e il risultato viene cifrato con la chiave SK.

Il risultato di tale operazione è una stringa da 64 bit che viene messa in XOR con il secondo blocco da 64 bit di dati e il risultato viene cifrato con la chiave SK. Il procedimento continua fino ad esaurimento dei blocchi dati. Se l'ultimo blocco è più corto di 64 bit questi verranno giustificati a sinistra e si aggiungeranno degli 0 a destra fino al completamento del blocco.

Il codice di autenticazione (MAC) è costituito dai 32 bit più significativi (quattro byte) risultanti dall'ultimo ciclo di cifratura. Al ricevimento dei dati viene svolta la stessa operazione e si verifica la congruenza del risultato con quanto ricevuto.

La chiave SK deve essere modificata dal CSC prima che il valore dei due contatori si ripresenti uguale (S = S ed R = R) ciò per evitare che si crei una maschera uguale alla precedente. Il procedimento è schematizzato in Fig. C.1.

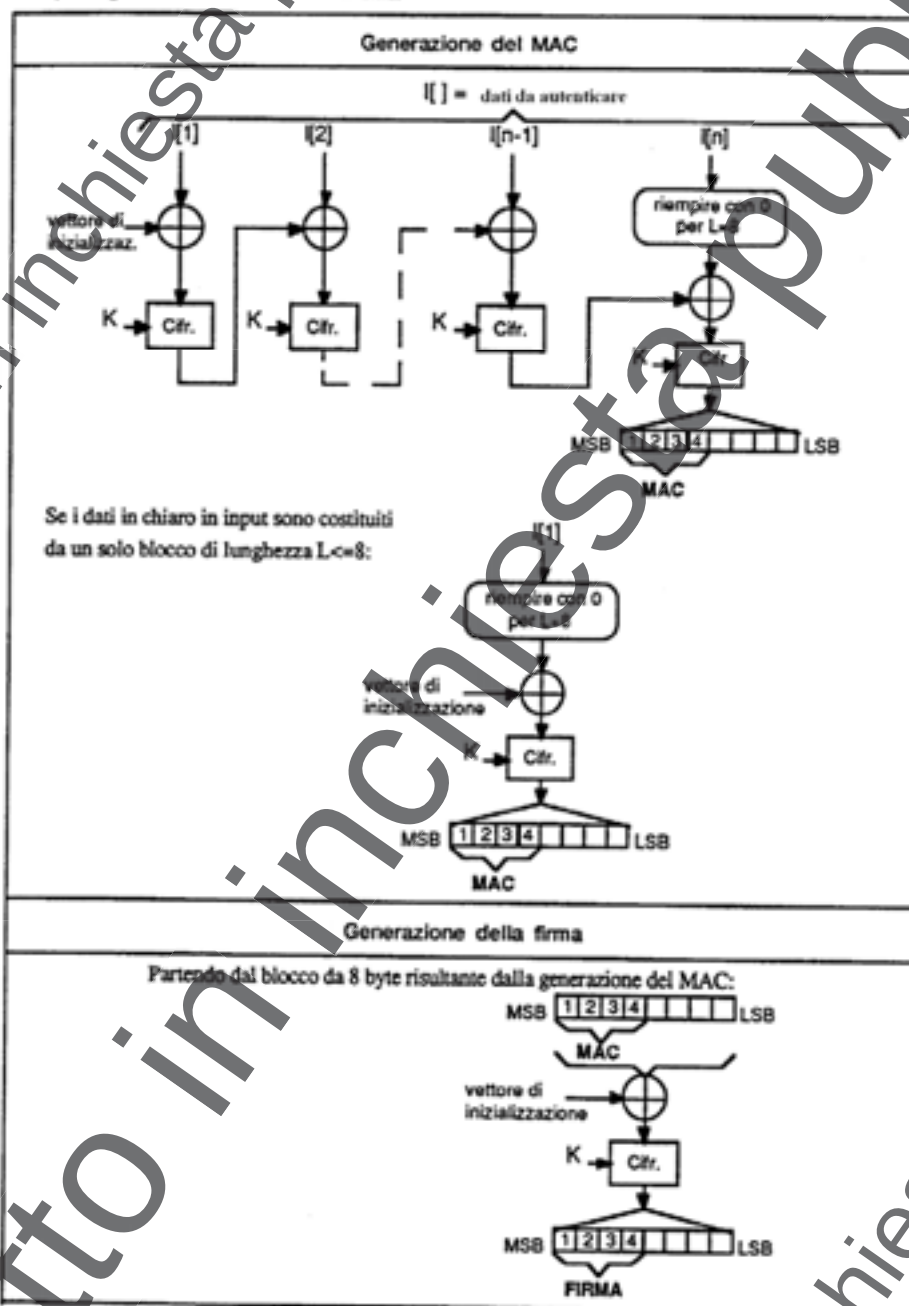


Legenda

- a Vettore di inizializzazione
- b Dati cifrati

Figura D.1 Procedura di calcolo del MAC

Con riferimento all'Appendice C viene di seguito riportato lo schema che evidenzia la procedura per la generazione del MAC e della firma.



ALGORITMO FEAL

APPENDICE E

1. Outline of FEAL-MAC

(1) Algorithm structure

The algorithm structure of FEAL-MAC is almost the same as DEA. MAC is defined as the leftmost 32-bits of the last ciphertext block. Round number of FEAL in FEAL-MAC is set at 16. A 64-bit message authentication key is used.

(2) FEAL cipher⁽¹⁾

FEAL-cipher (exactly FEAL-N) is a DES-like 64-bit block cipher algorithm. N, the number of internal processing stages of FEAL is a parameter and determined by the user. FEAL inputs a 64-bit key block. The outline of FEAL is explained in Annex 1.

(3) FEAL-MAC authentication key

FEAL-MAC uses a 64-bit authentication key. The 64-bit authentication key is the 64-bit FEAL cryptographic key.

2. Justification of proposal

2.1 FEAL-MAC goals

FEAL-MAC is designed to achieve two goals:

(1) Fast program processing

FEAL-MAC is designed to achieve high processing speed in software implementations for message authentication.

(2) Making exhaustive key search more difficult

The number of trials needed for exhaustive search is 2^{64} .

2.2 How the goals are achieved

(1) Fast program processing

FEAL-MAC internal processing is mainly performed with byte-wise (8-bits) data handling operations.

(1) FEAL: Fast Data Encipherment Algorithm.

The program processing speed of FEAL-MAC is dramatically increased by the 1-byte data handling technique. Details are explained in Annex 2.

- (2) **Making exhaustive key search more difficult**
FEAL-MAC uses a 64-bit key by using the 64-bit FEAL cryptographic key.

2.3 Additional merits

- (1) **Small program size**
Originator's experience is that FEAL-MAC program size is about 450 bytes for the 8-bit microprocessors intended for IC cards, and about 450 bytes for 16-bit microprocessors (80286) using the 1-byte data handling technique given in clause (1) of 2.2.
- (2) **Hardware Implementation**
The originator feels that FEAL-MAC is also suitable for hardware implementation. A FEAL⁽¹⁾ LSI chip using 2-micron CMOS technology was fabricated in 1988 and achieve 96 Mega bits per second in CBC mode. This means that a FEAL-MAC LSI (N = 16) chip could realistically achieve 48 Mega bits per second using the same technology. This processing speed is good compared with the DES chips currently available.
- (3) **Preparation to accept 128-bit key**
As LSI device speed will increase while costs will decrease, it is predicted that equipment for exhaustive search for all FEAL-MAC keys can be made within the near future (about one to two decades). To offset this, FEAL-MAC can accept a 128-bit authentication key by replacing FEAL-N in FEAL-MAC with FEAL-NX that accepts 128-bit key, explained in Annex 3 (in this case, FEAL-MAC will be modified to accept this replacement).

2.4 Experience in use

FEAL is currently used for many purposes by a variety of companies. Some of the uses are described, where notations are:

MA: Message authentication
CODE: Confidentiality
PA: Peer entity authentication
N: Round number, N of FEAL

- (1) An IC card security module (N = 8, CODE) has been implemented by NTT-Data Co., Ltd. Negotiations are in progress for use in medical and financial applications.
- (2) IC card security modules (N = 8, CODE) are used in a home stock trading system.
- (3) NTT will sell fax terminals integrated with cipher system (CODE, N = 8), and plans to use FEAL (MA, CODE, PA) in ISDN systems where value of N is not determined yet.
- (4) Encipherment equipment (N = 8, CODE) and modems with cipher (N = 8, CODE) have been made for communications systems used by public interest organizations.
- (5) Negotiations are under way with a company (in North America) for CD-ROM usage (CODE, N: not determined), with an Australian company for international communication (CODE, N = 16), and with a group of banks in a developing country for financial use (CODE, N = 16).

(1) FEAL with N = 8.

2.5 Patents

FEAL related patents were filed in Japan in 1985 and in France, FRG, UK and U.S.A. in 1986. The US patent on FEAL was issued in 1989, but others are in the pending stage in the each country.

3. Security consideration

Security testing of FEAL

The test report below is known to the originator. The report was published just recently and the authors are involved with research into ciphers so their conclusions are thought to be reliable. Annex 4 briefly explains the report's findings.

- (1) Helen Gustafson, Ed Dawson, Bill Caelli, "COMPARISON OF BLOCK CIPHERS", proceeding of AUSCRYPT '90, pp 153- pp165.

Annex 1 FEAL Outline

(1) **FEAL cipher family**

FEAL-4 was published in May 1987 for EUROCRYPT'87, and FEAL-8 was published in the REVIEW of ECL (Electrical Communication Labs. of NTT). Vol. 36 May 1988. FEAL-N was submitted to RIPE project in September 1989, where FEAL-N with $N = 4/8$ is identical to FEAL-4/8 respectively, i.e., FEAL-N is upwardly compatible with FEAL-4/8, and N is a user defined parameter. FEAL-NX (where X means an expansion of FEAL-N) was published at SCIS90⁽¹⁾ February 1990. FEAL-NX is also upwardly compatible with FEAL-N, and this guarantees that FEAL-NX can work as any of the above algorithms. The FEAL family is made from ciphers described above. N is even, and equal to or larger than four ($N \geq 4$, N : even).

(2) **N and 128-bit key**

Large values of N are expected to counter chosen plaintext attacks, while a 128-bit key is useful to counter powerful exhaustive key searches. The 128-bit key is ineffective against chosen plaintext attacks. Where chosen plaintext attacks are avoided⁽²⁾, small N ($N = 8$, for NTT's original applications) is useful to achieve better processing speeds, and where the attacks cannot be avoided, N should be increased.

(1) SCIS90: The 1990 Symposium on Cryptography and Information Security sponsored by Technical Group on Information Security in Institute of Electronics, Information and Communication Engineers of Japan.

(2) Examples:

(1) CBC/CFB-1,8 mode cipher communications with frequent exchanges (see ISO/DIS 10126-2, ISO 8732 and ISO 9150).

(2) Cipher users do not encipher plaintexts offered by any third party or peer entity in the ECB mode, using their secret key, and do not return the resulting ciphertexts to them etc.

Annex 2

1-byte data handling techniques for FEAL-MAC

Most of the programming steps are consumed by the f/f_K functions of FEAL in FEAL-MAC, and the S_0/S_1 -functions in f/f_K that need most programming steps in f/f_K functions can be coded very efficiently using the following techniques. Our experience is that one-stage can be coded in a maximum of about 25 steps per one data randomization stage. Either of the two following techniques can be used.

A2.1 Technique using 1-bit/2-bit left rotation instructions

If a 1-bit or 2-bit left rotation is used, S_0/S_1 functions can be coded. $S_1(X_1, X_2) = \text{Rot2}((X_1 + X_2 + 1) \bmod 256)$ is given below.

```
add  R1, R2    ; R1 ← (R1) + (R2) mod 256, where  $X_1/X_2$  is in R1/R2
inc  R1        ; R1 ← (R1) + 1
rot  R1        ; R1 ← 1-bit left rotation on (R1)
rot  R1        ; R1 ← 1-bit left rotation on (R1)
```

where R1/R2: 8-bits long registers (R1): content of R1,
add: add instruction between registers,
inc: increment 1 of (R1), rot: 1-bit left rotation

coding of S_0 function is obtained by omitting `inc` instruction from the above code. This technique is adaptable for many 8-bit/16-bit processors having 1-bit or 2-bit left rotation instructions. Our experience using 16-bit microprocessor (i80286) indicates that 21 steps for f -function and 25 steps for one stage are needed, if paired stage coding is employed.

A2.2 Table search technique

This idea is shown below for $S_1(X_1, X_2) = \text{Rot2}((X_1 + X_2 + 1) \bmod 256)$

First step: $X \leftarrow X_1 + X_2$ (add X_1 and X_2 , then sum is stored into X)
Second step: $Y \leftarrow 2\text{-bit left rotation on } (X+1)$ from table pointed X

This technique is suitable for processors that have a base register to point the table, and that do not have a 1-bit/2-bit left rotation instruction for 8-bit data. A detailed example is given below. Our experience is that programming a 32-bit CPU (i370 series) requires 24 steps for the f -function.

- (1) An example of table construction is given below where:
 - a) Addresses are expressed in bytes,
 - b) BR (base register) points to top address of the table,
 - c) At the address, $(BR)+X$, $\text{Rot2}(X)$ is stored for $0 \leq X \leq 510$,
 - d) At the address, $(BR)+512+Y$, $\text{Rot2}(Y+1)$ is stored for $0 \leq Y \leq 510$,
 where following notations are used:
 BR: base-register 16-/32-bits long
 (BR), (R1): contents of register BR, R1 respectively
 $\text{Rot2}(Z)$: the result of 2-bit left rotation on 9-bit data, Z
 X, Y: 9-bits long
- (2) Sample programs of S-box are shown where:

R1, R2: register 16- or 32-bits long
 AR: add instruction between registers
 LC: load 1-byte instruction from memory to register
 X1, X2: 8-bits non-negative integer

Coding of S_2 -function where output = $\text{Rot2}((X1 + X2) \bmod 256)$

AR R1, R2 ; $R1 \leftarrow (R1) + (R2)$, X1 is in R1, X2 is in R2.

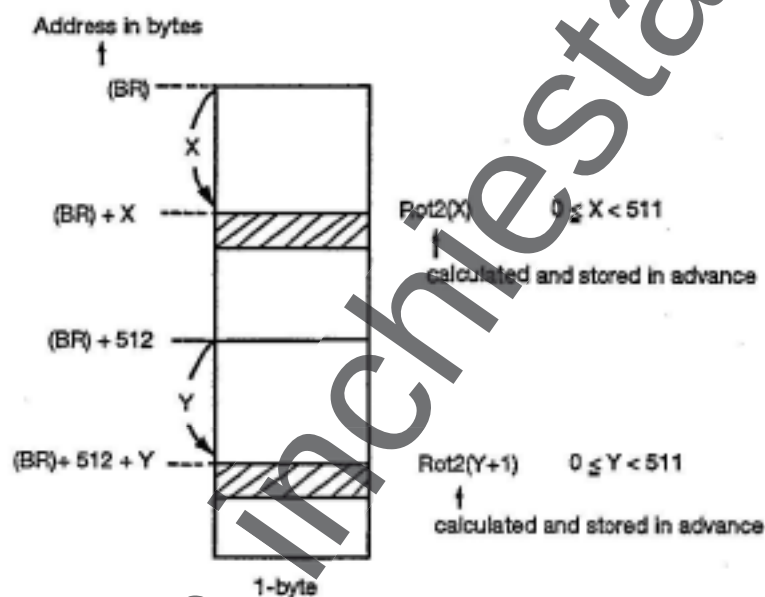
LC R1, 0(R1, BR) ; $R1 \leftarrow \text{Rot2}(X)$ from address, $(R1) + (BR)$ where $(R1) = X$.

Coding of S_1 -function where output = $\text{Rot2}((X1 + X2 + 1) \bmod 256)$

AR R1, R2 ; $R1 \leftarrow (R1) + (R2)$, X1 is in R1, X2 is in R2.

LC R1, 512(R1, BR) ; $R1 \leftarrow \text{Rot2}(Y + 1)$ from address, $(BR) + 512 + (R1)$ where $(R1) = Y$.

The table is explained in the following.



Annex 3 128-bit key cipher FEAL-NX

A3.1 Outline of FEAL-NX

FEAL-NX uses both a key schedule and data randomization that is the same as that used in FEAL-N. The FEAL-NX key schedule is made by slightly modifying the FEAL-N key schedule, so that the right half of FEAL-NXs' 128-bit key is all zeros, FEAL-NX key schedule works as FEAL-N key schedule. FEAL-NX works as FEAL-N when the right half 64-bits of FEAL-NXs 128-bit key are all zeros.

A3.2 Key schedule of FEAL-NX

The key schedule yields the extended key K_i ($i = 0, 1, 2, \dots, N+7$) from the 128-bit key.

A3.2.1 Definition of auxiliary variables K_L and K_R

Auxiliary 64-bit variables K_L and K_R are defined.

K_L = Left half of 128-bit key block

K_R = Right half of 128-bit key block

I.e., (K_L, K_R) is the inputted 128-bit key-block.

A3.2.2 Parity bit processing

A3.2.2.1 Non-use of key parity bits

There is no special processing here.

A3.2.2.2 Use of key parity bits

Bit positions, 8, 16, 24, 32, 40, 48, 56, 64, of both K_L and K_R are set to zeros, i.e., all parity bits in the key block are set to zero.

Note - Parity check is outside the scope of the FEAL-NX enciphering/deciphering algorithms.

A3.2.3 Calculation of auxiliary variable Q_r

K_R is divided into left half K_{R1} and right half K_{R2} , and Q_r is defined as:

$$Q_r = K_{R1} \oplus K_{R2} \quad \text{for } r = 1, 4, \dots, (r = 3i + 1; i = 0, 1, \dots)$$

$$Q_r = K_{R1} \quad \text{for } r = 2, 5, \dots, (r = 3i + 2; i = 0, 1, \dots)$$

$$Q_r = K_{R2} \quad \text{for } r = 3, 6, \dots, (r = 3i + 3; i = 0, 1, \dots)$$

Where $1 \leq r \leq (N/2) + 4$. ($N \geq 4$, N : even)

Note - For FEAL-N, $K_R = (\emptyset, \emptyset)$ (64 zeros) and $Q_r = (\emptyset, \emptyset)$ (64 zeros).

A3.2.4 Iterative calculation

Let A_0 be the left of auxiliary variable K_L and let B_0 be the right, i.e., $K_L = (A_0, B_0)$ and $D_0 = \emptyset$. Then calculate K_i ($i = 0$ to $N + 7$) ($i = 1$ to $(N/2) + 4$, ($N \geq 4$, N : even))

$$D_r = A_{r-1}, A_r = B_{r-1}$$

$$B_r = f_K(\alpha, \beta) \oplus f_K(A_{r-1}, (B_{r-1} \oplus D_{r-1} \oplus Q_r))$$

$$K_{2r-1} = (B_r, B_{r+1}), K_{2r-1+1} = (B_{r+2}, B_{r+3})$$

where A_r, B_r, B_{r+1} and Q_r are auxiliary variables. $B_r = (B_{r0}, B_{r1}, B_{r2}, B_{r3})$. $B_{r0}, \dots, B_{r3}$ are each 8 bits long. Function f_K is the same as that of FEAL-N.

Annex 4 Outline of Test Report

A4.1 Introduction

[GDC90] reports the analysis and comparison of four block ciphers; DES, FEAL-4, FEAL-8 and Madryga. Two kinds of tests are applied in the report. One type tests randomness while the other measures avalanche effects. This annex is an abstract of the report.

A4.2 Randomness Tests

A4.2.1 Measures of Randomness

Report [GDC90] evaluates the ciphers from the point of the randomness of finite sequences. The randomness tests can be described as follows:

1. Statistical Tests:
 - (a) Frequency Test:
The number of zeros and ones is examined.
 - (b) Serial Test:
The number of 00, 01, 10 and 11 entries is examined.
 - (c) Runs Test:
The number of blocks (runs of ones) and gaps (runs of zeros) is examined. Here, a run is a continuous sequence of a symbol (one or zero).
2. Sequence Complexity:
The complexity of a sequence is examined. The complexity of a sequence is a measure of the rate at which new patterns emerge as we move along the sequence⁽¹⁾. (See [LZ76].)
3. Binary Derivative:
A binary derivative is taken by considering each adjacent pair of digits and recording a zero if they are the same and a one if they are different⁽²⁾. The successive binary derivative is used to evaluate the difference between patterned and random sequences. (See [CR88].)

A4.2.2 Applying Randomness Tests

In [GDC90], two types of experiments were applied for all randomness tests as described above, i.e., Frequency Test, Serial Test, Runs Test, Sequence Complexity and Binary Derivative.

Experiment 1 was to generate non-random sequences as a binary plaintext. If the ciphertext is independent of the plaintext it should be random. In Experiment 1, 64 bit vectors containing 64, 63, 62 zeros or ones were used as plaintext. There are a total of 4162 such vectors. And the randomness of the resulting ciphertexts was tested.

Experiment 2 generated a large number of random plaintext vectors $p_1, \dots, p_r$. Let $c_1, \dots, c_r$ be the resultant ciphertext vectors with a fixed key.

Define $s_i = p_i \oplus c_i$, for $i = 1, \dots, r$.

The vector s_i should be random if the ciphertext is independent of the plaintext.

(1) Example: the sequence 1001101110000111 has 6 patterns: 1/0/01/50/11/100/00111.

(2) Example: the sequence 1000100011010101 is changed to 10011001011111 by binary derivative.

In Experiment 2, 10000 random plaintext vectors were used. The random plaintext vectors were generated by a stream cipher, and the randomness of vectors s_1 was tested. One result is shown below. (Full results are given in [GDC90].) Here, it can be seen that a good value is about 5%.

Table 1 — Number of Ones

% of n where $n < 24$ or $n > 40$, n : number of ones

	FEAL-4	FEAL-8	Madryga	DES
Exp 1	2,90	3,63	2,38	3,56
Exp 2	3,31	3,02	1,64	2,12

The report concludes that there is no noticeable difference between the ciphers.

A4.3 Avalanche Effect

The avalanche effect is that a "small" change in the plaintext or key gives rise to a "large" change in the ciphertext.

There are two types of avalanche effects which can be examined:

1. the plaintext-ciphertext avalanche effect;
2. the key-ciphertext avalanche effect.

A block cipher satisfies the plain-ciphertext strict avalanche effect if each ciphertext bit changes with a probability of one half whenever a single plaintext bit is changed. And a block cipher satisfies the key-ciphertext strict avalanche effect if each ciphertext bit changes with a probability of one half whenever a single key bit is changed.

In this experiment, 10000 random strings were generated by a stream cipher. Also 1000 random strings were generated for DES. ([GDC90] reports that the DES cipher was restricted to 1000 random strings due to the slowness in encryption time of the DES software.) The Kolmogorov-Smirnov Test was used to assess the resulting changes.

Here, it can be seen that a good value is 0.

Table 2 — Avalanche Effect

$$D = \max |F(x) - RCF(x)|$$

Cipher	Plaintext-Ciphertext	Key-Ciphertext
FEAL-4	0,0173	0,0083
FEAL-8	0,0116	0,0156
Madryga	0,6453	0,0184
DES*	0,0248	0,0276

The author of [GDC90] warns that quantity of test data for DES is small compared with the other ciphers.

A4.4 Conclusion

1. Randomness Tests:

As a result of the randomness tests, [GDC90] concludes that there is no noticeable difference between the ciphers, FEAL-4, FEAL-8, DES and Madryga, from the point of the randomness.

2. Avalanche Effects:

Next, regarding the avalanche effects, the report states:

Based on the experimental results in Section 3 and 4 (of the report) it is concluded that the three Feistel ciphers FEAL-4 and 8, and DES satisfy plaintext-ciphertext independence and both the strict plaintext-ciphertext avalanche effect and the strict key-ciphertext effect.

References

- [GDC90] H. Gustafson, E. Dawson and B. Caelli, "Comparison of Block Cipher", Abstracts of ASCRYPT'90, pp. 153-165 (Jan., 1990).
- [LZ76] A. Lempel and J. Ziv, "On the Complexity of Finite Sequences", IEEE Trans. on Information Theory, Vol.IT-22, pp.75-81 (Jan., 1976).
- [CR88] J.M. Carroll and L.E. Robbins, "Using Binary Derivatives to Test an Enhancement of DES", Cryptologia, Vol.XII, No.4, pp.193-208 (Oct., 1988).

Part X: FEAL-MAC

3. Source of algorithm FEAL

FEAL parameters shall be selected as:

- a) Round number $N = 16$
- b) Non-use of key parity bits

4. Algorithm specification

4.1 Authentication key

A FEAL cryptographic key consists of 64 bits. Bit positions of the key are numbered from left to right as 1 (MSB), 2, ..., 64 (LSB).

A 64-bit authentication key of FEAL-MAC is assigned to bit positions 1 to 64 of the FEAL cryptographic key.

When the FEAL cryptographic 64-bit key is written, it shall be represented by eight pairs of hexadecimal characters with each pair separated by a space as:

hh hh hh hh hh hh hh hh

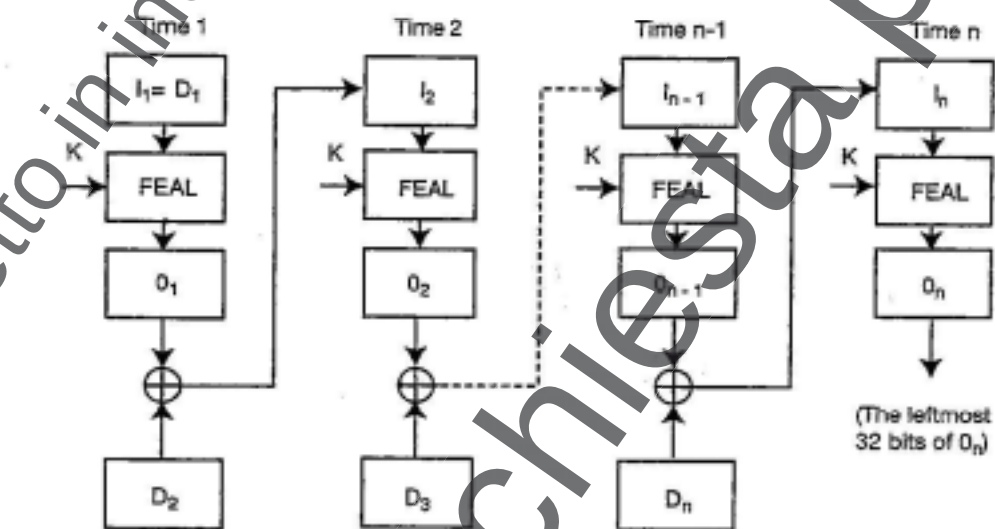
The active key shall be randomly or pseudo-randomly generated. The entire 64-bit key shall be protected from all unauthorized parties throughout its active life. Keys shall be changed from time to time.

4.2 Generating the Message Authentication Code (MAC) using FEAL

The MAC is generated as illustrated in Figure 1. FEAL shall be used in the cipher block chaining modes of operation (CBC mode) with the initialization vector set at zero as specified in ISO 8372. The input register (I_1) (see the Figure 1) is initialized with the first 64 bits of data (D_1) to be authenticated. This input is passed through the FEAL which includes an authentication key (K) as defined above to produce 64 bits in the out-put register (O_1). The second 64 bits of data (D_2) to be

authenticated are modulo-2 added with the 64 bits in the output register, the result being loaded into the input register (I_2).

This process continues until either fewer bits remain to be authenticated. These remaining bits shall be left-justified and zeros shall be appended to form a 64 bit data block (D_n). This block is then modulo-2 added with the 64 bits in the output register. The result (I_n) is passed through the FEAL producing the final 64 bit output block (O_n). The leftmost 32 bits of final output (O_n) shall be taken as the MAC. An example of the MAC computation is given in Annex B of this Standard.



Legend

- FEAL : Fast Data Encipherment Algorithm (Encryption mode)
- O : Output
- K : FEAL key
- D : Data block
- $\oplus$: modulo-2 addition

Fig. 1 — FEAL-MAC Authentication Algorithm

Annex A (normative)

FEAL SPECIFICATIONS

Note — This Annex is an integral part of the body of the Standard.

1. Introduction

1.1 Outline of the FEAL cipher

FEAL means FEAL-N (N round FEAL with 64-bit key) in this document. FEAL is a 64-bit block cipher algorithm that enciphers 64-bit plaintexts into 64-bit ciphertexts and vice versa with a 64-bit key.

FEAL has two options: round number and key parity. The round number (N) specifies the internal iteration number for data randomization, and the key parity option selects either use or non-use of the parity bits in a key block.

Where chosen plaintext attacks are not a threat⁽¹⁾, small N (ex. N = 6) is useful to achieve better processing speeds, and where these attacks become a threat, the number of N should be increased.

1.2 FEAL options

FEAL options are defined below.

- (1) Round number (N)
Determines the round number (N) for FEAL data randomization, where $N \geq 4$ and even.
- (2) Key parity
Indicates:
 - Use of key parity bits in a key-block, or
 - Non-use of key parity bits in a key-block

1.3 Definitions and explanations

1.3.1 Definitions

(1) Key-block

A 64-bit block that includes an effective key, and includes key parity bits only when the key parity option indicates key parity use.

(2) Key parity bits

Odd parity bits for 8-bit data in a key-block (see key-block).

(3) Key-block with no parity bits

A key-block that does not include parity bits.

(1) Examples:

(1) CBC/CFB-1,8 mode cipher communications with frequent key exchanges (see BSADIS 10126-2, ISO 8732 and ISO 9160).

(2) Cipher users do not encipher plaintexts offered by any third party or peer entity, in the ECB mode with their secret keys, and do not return the resulting ciphertexts to them, etc.

(4) **Key-block with parity bits**

A key-block that includes key parity bits. Parity bit positions are $8 \times i$ where $i = 1, 2, 3, \dots, 8$ etc.

(5) **Key**

Effective key information used for enciphering/deciphering.

(6) **Round number (N)**

The internal iteration number for FEAL data randomization.

(7) **Extended key**

16-bit blocks, K_i , which are a randomized and extended form of the key, are output from FEAL key schedule, where $i = 0, 1, \dots, N + 7$.

1.3.2 Conventions and Notations

- (1) $A, A_r, \dots$: blocks
- (2) $(A, B, \dots)$: concatenation in this order
- (3) $A \oplus B$: exclusive-or operation of A and B
- (4) ϕ : zero block, 32-bits long
- (5) $\Leftarrow$: Transfer from left side to right side
- (6) Bit position: 1, 2, 3, ... from the first left side bit (MSB) in a block towards the right.

2. Enciphering algorithm

2.1 Computation stages

The extended key K_i used in the enciphering procedure is generated by the key schedule described in clause 4. Function f used in the procedure is defined in clause 5. The computation stages, specified more fully in subclauses 2.2 to 2.4, shall be as follows (see also Figure 1):

- a) Pre-processing (see 2.2)
- b) Iterative calculation (see 2.3)
- c) Post-processing (see 2.4)

2.2 Pre-processing

Plaintext P is separated into L_0 and R_0 of equal lengths (32 bits), i.e. $P = (L_0, R_0)$.

First,

$$(L_0, R_0) = (L_0, R_0) \oplus (K_N, K_{N+1}, K_{N+2}, K_{N+3})$$

Next,

$$(L_0, R_0) = (L_0, R_0) \oplus (\phi, L_0)$$

2.3 Iterative calculation

Input (L_0, R_0) , and calculate the equations below for r from 1 to N iteratively.

$$R_r = L_{r-1} \oplus f(R_{r-1}, K_{r-1})$$

$$L_r = R_{r-1}$$

Output of r -th round is (L_r, R_r) .

2.4 Post-processing

Interchange the final output of the iterative calculation, (L_N, R_N) , into (R_N, L_N) .
Next calculate:

$$(R_N, L_N) = (R_N, L_N) \oplus (\phi, R_N)$$

Lastly,

$$(R_N, L_N) = (R_N, L_N) \oplus (K_{N+4}, K_{N+5}, K_{N+6}, K_{N+7})$$

Ciphertext is given as (R_N, L_N) .

3. Deciphering algorithm

3.1 Computation stages

The extended key K_i used in this deciphering procedure is generated by the key schedule described in clause 4. Function f used in the procedure is defined in clause 5. The computation stages, specified more fully in subclauses 3.2 to 3.4, shall be as follows (see also Figure 1):

- Pre-processing (see 3.2)
- Iterative calculation (see 3.3)
- Post-processing (see 3.4)

3.2 Pre-processing

Ciphertext (R_N, L_N) is separated into R_N and L_N of equal lengths.
First,

$$(R_N, L_N) = (R_N, L_N) \oplus (K_{N+4}, K_{N+5}, K_{N+6}, K_{N+7})$$

Next,

$$(R_N, L_N) = (R_N, L_N) \oplus (\phi, R_N)$$

3.3 Iterative calculation

Input (R_N, L_N) and calculate the equations below for r from N to 1 iteratively,

$$\begin{aligned} L_{r-1} &= R_r \oplus f(L_r, K_{r-1}) \\ R_{r-1} &= L_r \end{aligned}$$

Output of r -th round is (R_r, L_r) .

3.4 Post-processing

Interchange the final output of the iterative calculation, (R_0, L_0) , into (L_0, R_0) .
Next calculate:

$$(L_0, R_0) = (L_0, R_0) \oplus (\phi, L_0)$$

Lastly,

$$(L_0, R_0) = (L_0, R_0) \oplus (K_N, K_{N+1}, K_{N+2}, K_{N+3})$$

Plaintext is given as (L_0, R_0) .

4. Key schedule

The FEAL-N key schedule yields the extended key K_i ($i = 0, 1, 2, \dots, N+7$) from the 64-bit key.
Function f_K used in the operation is defined in clause 5.

4.1 Computation stages

The computation stages shall be as follows (see also Figure 2).

- Parity bit processing (see 4.2)
- Iterative calculation (see 4.3)

4.2 Parity bit processing

4.2.1 Non-use of key parity bits

There is no special processing here.

4.2.2 Use of key parity bits

Bit positions, 8, 16, 24, 32, 40, 48, 56, 64, of the 64-bit key are set to zeros.

4.3 Iterative calculation

Let A_0 be the left half of the 64-bit key and let B_0 be the right, i.e., the 64-bit key = (A_0, B_0) and $D_0 = \emptyset$.

Then calculate K_i ($i = 0$ to $N + 7$) for $r = 1$ to $(N/2) + 4$. ($N \geq 4$, N : even)

$$D_r = A_{r-1}$$

$$A_r = B_{r-1}$$

$$B_r = f_K(\alpha, \beta) = f_K(A_{r-1}, (B_{r-1} \oplus D_{r-1}))$$

$$K_{2(r-1)} = (B_{r0}, B_{r1})$$

$$K_{2(r-1)+1} = (B_{r2}, B_{r3})$$

where A_r , B_r and D_r are 32-bit auxiliary variables. $B_r = (B_{r0}, B_{r1}, B_{r2}, B_{r3})$ and $B_{r0}, \dots, B_{r3}$ are each 8 bits long.

5. Functions

This clause describes functions used in clauses 2.3 and 4.

5.1 Function f (see also Figure 3)

$f(\alpha, \beta)$ is shortened to f . α and β are divided as follows, where α_i and β_i are 8-bits long. Functions S_0 and S_1 are defined in clause 5.3.

$$\alpha = (\alpha_0, \alpha_1, \alpha_2, \alpha_3), \beta = (\beta_0, \beta_1).$$

$f = (f_0, f_1, f_2, f_3)$ are calculated in sequence.

$$f_1 = \alpha_1 \oplus \beta_0$$

$$f_2 = \alpha_2 \oplus \beta_1$$

$$f_1 = f_1 \oplus \alpha_0$$

$$f_2 = f_2 \oplus \alpha_3$$

$$f_1 = S_1(f_1, f_2)$$

$$f_2 = S_0(f_2, f_1)$$

$$f_0 = S_0(\alpha_0, f_1)$$

$$f_3 = S_1(\alpha_3, f_2)$$

Example in hex:

Inputs: $\alpha = 00\ FF\ FF\ 00$, $\beta = FF\ FF$, Output: $f = 10\ 04\ 10\ 44$

5.2 Function f_K (see also Figure 4)

Inputs of function f_K , α and β , are divided into four 8-bit blocks as:

$$\alpha = (\alpha_0, \alpha_1, \alpha_2, \alpha_3), \beta = (\beta_0, \beta_1, \beta_2, \beta_3).$$

$f_K(\alpha, \beta)$ is shortened to f .

$f_K = (f_{K0}, f_{K1}, f_{K2}, f_{K3})$ are calculated in sequence, where functions S_0 and S_1 are defined in clause 5.3.

$$\begin{aligned} f_{K1} &= \alpha_1 \oplus \alpha_0 \\ f_{K2} &= \alpha_2 \oplus \alpha_3 \\ f_{K1} &= S_1(f_{K1}, (f_{K2} \oplus \beta_0)) \\ f_{K2} &= S_0(f_{K2}, (f_{K1} \oplus \beta_1)) \\ f_{K0} &= S_0(\alpha_0, (f_{K1} \oplus \beta_2)) \\ f_{K3} &= S_1(\alpha_3, (f_{K2} \oplus \beta_3)) \end{aligned}$$

Example in hex:

Inputs: $\alpha = 00\ 00\ 00\ 00$, $\beta = 00\ 00\ 00\ 00$, $f_K = 10\ 04\ 10\ 44$

5.3 Function S

S_0 and S_1 are defined as follows:

$$S_0(X_1, X_2) = \text{Rot2}((X_1 + X_2) \bmod 256)$$

$$S_1(X_1, X_2) = \text{Rot2}((X_1 + X_2 + 1) \bmod 256)$$

where X_1 and X_2 are 8-bit blocks and $\text{Rot2}(T)$ is the result of a 2-bit left rotation operation on 8-bit block, T .

Example: Suppose $X_1 = 00010011$, $X_2 = 11110010$ then

$$T = (X_1 + X_2 + 1) \bmod 256 = 00000110$$

$$S_1(X_1, X_2) = \text{Rot}(T) = 0011000$$

6. Example of working data

Working data are shown in bit sequence and in hexadecimal (hex).

6.1 FEAL OPTIONS (see clause 1.2)

In this example following FEAL options are selected:

- (1) Round number $N = 16$ is selected.
- (2) Non-use of key parity bits is selected.

6.2 Input data

Input data are the key-block and the plaintext.

The example of the key-block K is given by the 64-bit sequence:

$K = 0000\ 0001\ 0010\ 0011\ 0100\ 0101\ 0110\ 0111$
 $1000\ 1001\ 1010\ 1011\ 1100\ 1101\ 1110\ 1111$ in bit sequence

$K = 01\ 23\ 45\ 67\ 89\ AB\ CD\ EF$ in hex

The plaintext P in the example is a 64-bit sequence:

$P = 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000$
 $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000$ in bit sequence

$P = 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00$ in hex

6.3 The key schedule (see clause 4)

Consider first the generation of the extended keys, $K_0, K_1, K_2, \dots, K_{23}$, each consisting of 16 bits generated from the key block K .

6.3.1 Parity bit processing (see 4.2)

As non-use of key parity bits is selected, there is no special processing here.

6.3.2 Iterative calculation (see 4.3)

Let A_0 be the left of K_L and let B_0 be the right half of K_L , i.e.,

$K_L = A_0 || B_0$ and $D_0 = \emptyset$. Thus:

$A_0 = 0000\ 0001\ 0010\ 0011\ 0100\ 0101\ 0110\ 0111$ in bit sequence
= 01 23 45 67 in hex

$B_0 = 1000\ 1001\ 1010\ 1011\ 1100\ 1101\ 1110\ 1111$ in bit sequence
= 89 AB CD EF in hex

$D_0 = 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000$ in bit sequence
= 00 00 00 00 in hex

Calculate D_1, A_1, B_1, K_0 and K_1 as:

$D_1 = A_0 = 0000\ 0001\ 0010\ 0011\ 0100\ 0101\ 0110\ 0111$ in bit sequence
= 01 23 45 67 in hex

$A_1 = B_0 = 1000\ 1001\ 1010\ 1011\ 1100\ 1101\ 1110\ 1111$ in bit sequence
= 89 AB CD EF in hex

$B_1 = f_K(A_0, B_0 \oplus D_0)$
= 1101 1111 0011 1011 1100 1010 0011 0110 in bit sequence
= DF 3B CA 36 in hex

$K_0 = 1101\ 1111\ 0011\ 1011$ in bit sequence
= DF 3B in hex

$K_1 = 1100\ 1010\ 0011\ 0110$ in bit sequence
= CA 36 in hex

If this procedure is continued it will be found that the extended key K_i is given in hex by:

$K_0 = \text{DF 3B}$	$K_1 = \text{CA 36}$	$K_2 = \text{F1 7C}$	$K_3 = \text{1A EC}$
$K_4 = \text{45 A5}$	$K_5 = \text{B9 C7}$	$K_6 = \text{26 EB}$	$K_7 = \text{AD 25}$
$K_8 = \text{8B 2A}$	$K_9 = \text{EC B7}$	$K_{10} = \text{AC 50}$	$K_{11} = \text{9D 4C}$
$K_{12} = \text{22 CD}$	$K_{13} = \text{47 9B}$	$K_{14} = \text{A8 D5}$	$K_{15} = \text{0C B5}$
$K_{16} = \text{53 23}$	$K_{17} = \text{0A 3E}$	$K_{18} = \text{F3 19}$	$K_{19} = \text{C2 74}$
$K_{20} = \text{82 83}$	$K_{21} = \text{0E 38}$	$K_{22} = \text{01 49}$	$K_{23} = \text{7E B7}$

6.4 The Enciphering Algorithm (see clause 2)

6.4.1 Pre-processing (see 2.2)

$P = 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000$
0000 0000 0000 0000 0000 0000 0000 0000 in bit sequence
 $P = 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00$ in hex

P is separated into L_0 and R_0 both 32-bits long.

First,

$$\begin{aligned}(L_0, R_0) &= (L_0, R_0) \oplus (K_{16}, K_{17}, K_{18}, K_{19}) \\ &= 0101\ 0011\ 0010\ 0011\ 0000\ 1010\ 0011\ 1110 \\ &\quad 1111\ 0011\ 0001\ 1001\ 1100\ 0010\ 0111\ 0100 \text{ in bit sequence} \\ &= 53\ 23\ 0A\ 3E\ F9\ 19\ C2\ 74 \text{ in hex}\end{aligned}$$

Next,

$$\begin{aligned}(L_0, R_0) &= (L_0, R_0) \oplus (\phi \oplus L_0) \\ &= 0101\ 0011\ 0010\ 0011\ 0000\ 1010\ 0011\ 1110 \\ &\quad 1010\ 0000\ 0011\ 1010\ 1100\ 1000\ 0100\ 1010 \text{ in bit sequence} \\ &= 53\ 23\ 0A\ 3E\ AD\ 3A\ C8\ 4A \text{ in hex}\end{aligned}$$

Output of this processing stage is:

$$\begin{aligned}L_0 &= 0101\ 0011\ 0010\ 0011\ 0000\ 1010\ 0011\ 1110 \text{ in bit sequence} \\ &= 53\ 23\ 0A\ 3E \text{ in hex} \\ R_0 &= 1010\ 0000\ 0011\ 1010\ 1100\ 1000\ 0100\ 1010 \text{ in bit sequence} \\ &= AD\ 3A\ C8\ 4A \text{ in hex}\end{aligned}$$

6.4.2 Iterative calculation (see 2.3)

6.4.2.1 Calculation of R_0 and L_0 at the first stage

First, calculate $f(R_0, K_0)$ as:

$$\begin{aligned}f(R_0, K_0) &= 0111\ 1110\ 1111\ 1111\ 1110\ 0010\ 1011\ 0100 \text{ in bit sequence} \\ &= 7E\ FF\ E2\ B4 \text{ in hex}\end{aligned}$$

Where details are described in clause 6.4.2.2.

$$L_0 \oplus f(R_0, K_0) = 2D\ DC\ E8\ 8A \text{ in hex}$$

Output of first stage of the iterative calculation is:

$$\begin{aligned}L_1 = R_0 &= 1010\ 0000\ 0011\ 1010\ 1100\ 1000\ 0100\ 1010 \text{ in bit sequence} \\ &= AD\ 3A\ C8\ 4A \text{ in hex} \\ R_1 &= 0010\ 1101\ 1101\ 1100\ 1110\ 1000\ 1000\ 1010 \text{ in bit sequence} \\ &= 2D\ DC\ E8\ 8A \text{ in hex}\end{aligned}$$

6.4.2.2 Calculation f of the first stage

In the calculation of $f(R_0, K_0)$, shown below, f is shortened to f , and α e β are defined as:

$$\begin{aligned}\alpha &= (\alpha_0, \alpha_1, \alpha_2, \alpha_3) = R_0 \\ &= 1010\ 0000\ 0011\ 1010\ 1100\ 1000\ 0100\ 1010 \text{ in bit sequence} \\ &= AD\ 3A\ C8\ 4A \text{ in hex}\end{aligned}$$

$$\begin{aligned}\beta &= (\beta_0, \beta_1) = K_0 \\ &= 1101\ 1111\ 0011\ 1011 \text{ in bit sequence} \\ &= DF\ 3B \text{ in hex}\end{aligned}$$

$$\alpha_0 = 1010\ 0000 = AD \text{ in hex,}$$

$$\alpha_2 = 1100\ 1000 = C8 \text{ in hex,}$$

$$\beta_0 = 1101\ 1111 = DF \text{ in hex,}$$

$$\alpha_1 = 0011\ 1010 = 3A \text{ in hex}$$

$$\alpha_3 = 0100\ 1010 = 4A \text{ in hex}$$

$$\beta_1 = 0011\ 1011 = 3B \text{ in hex}$$

$f(f_0, f_1, f_2, f_3)$ are calculated by the sequence:

$f_1 = \alpha_1 \oplus \beta_0 = 1110\ 0101 = E5$ in hex
 $f_2 = \alpha_2 \oplus \beta_1 = 1111\ 0011 = F3$ in hex
 $f_1 = f_1 \oplus \alpha_0 = 0100\ 0101 = 45$ in hex
 $f_2 = f_2 \oplus \alpha_3 = 1011\ 1001 = B9$ in hex
 $f_1 = S_1(f_1, f_2) = 1111\ 1111 = FF$ in hex
 $f_2 = S_0(f_2, f_1) = 1110\ 0010 = E2$ in hex
 $f_0 = S_0(\alpha_0, f_1) = 0111\ 1110 = 7E$ in hex
 $f_3 = S_1(\alpha_3, f_2) = 1011\ 0100 = B4$ in hex

6.4.2.3 Continued calculations

If the above calculations are continued it will be found that L_r and R_r etc. are as given in hex by Table 1.

Table 1 — The Process Stages

r	L_r	R_r	K_{r-1}	$f(R_{r-1}, K_r)$
0	53 23 0A 3E	A0 3A C8 4A		
1	A0 3A C8 4A	2D DC E8 8A	DF3B	7E FF E2 B4
2	2D DC E8 8A	1D 78 92 DD	CA36	BD 42 5A 97
3	1D 78 92 DD	2C FF B1 56	F17C	01 23 59 DC
4	2C FF B1 56	13 2F 1B 5E	1AEC	0E 57 89 83
5	13 2F 1B 5E	DD 96 94 44	45A5	F1 89 25 12
6	DD 96 94 44	07 07 E7 5B	B9C7	14 28 FC 05
7	07 07 E7 5B	DD 6F D5 22	26EB	0D F9 41 76
8	DD 6F D5 22	A6 8C D2 FA	AD25	A1 8B 35 A1
9	A6 8C D2 FA	3D FD 87 07	8B2A	E0 92 52 35
10	3D FD 87 07	9D 1D F1 56	ECB7	3B 91 23 AC
11	9D 1D F1 56	89 6D 99 D2	AC50	B4 90 1E D5
12	89 6D 99 D2	A3 1B C5 4A	9D4C	3E 06 34 1C
13	A3 1B C5 4A	E1 1A 7F 16	22CD	68 77 E6 C4
14	E1 1A 7F 16	DD A5 07 2D	479B	7E BE C2 67
15	DD A5 07 2D	61 59 76 CA	A8D5	80 43 09 DC
16	61 59 76 CA	B8 5D 03 12	0CB5	65 F8 04 3F

6.4.3 Post processing (see 2.4)

First, interchanging L_{15} and R_{15} yields:

$(R_{15}, L_{15}) = 1011 \ 1000 \ 0101 \ 1101 \ 0000 \ 0011 \ 0001 \ 0010$
0110 0001 0101 1001 0111 0110 1100 1010 in bit sequence
= BB 5D 03 12 61 59 76 CA in hex.

Next,

$(R_{15}, L_{15}) = (R_{15}, L_{15}) \oplus (K \oplus R_{15})$
 $(R_{15}, L_{15}) = 1011 \ 1000 \ 0101 \ 1101 \ 0000 \ 0011 \ 0001 \ 0010$
1101 1001 0000 0100 0111 0101 1101 1000 in bit sequence
= BB 5D 03 12 D9 04 75 D8 in hex.

Lastly,

$(R_{15}, L_{15}) = (R_{15}, L_{15}) \oplus (K_{20}, K_{21}, K_{22}, K_{23})$
= 0011 1010 1101 1110 0000 1101 0010 1010
1101 1000 0100 1101 0000 1011 0110 1111 in bit sequence
= 3A DE 0D 2A D8 4D 0B 6F in hex.

Ciphertext is given as (R_{15}, L_{15}) .

The final result (ciphertext) is:

C = 0011 1010 1101 1110 0000 1101 0010 1010
1101 1000 0100 1101 0000 1011 0110 1111 in bit sequence
= 3A DE 0D 2A D8 4D 0B 6F in hex.

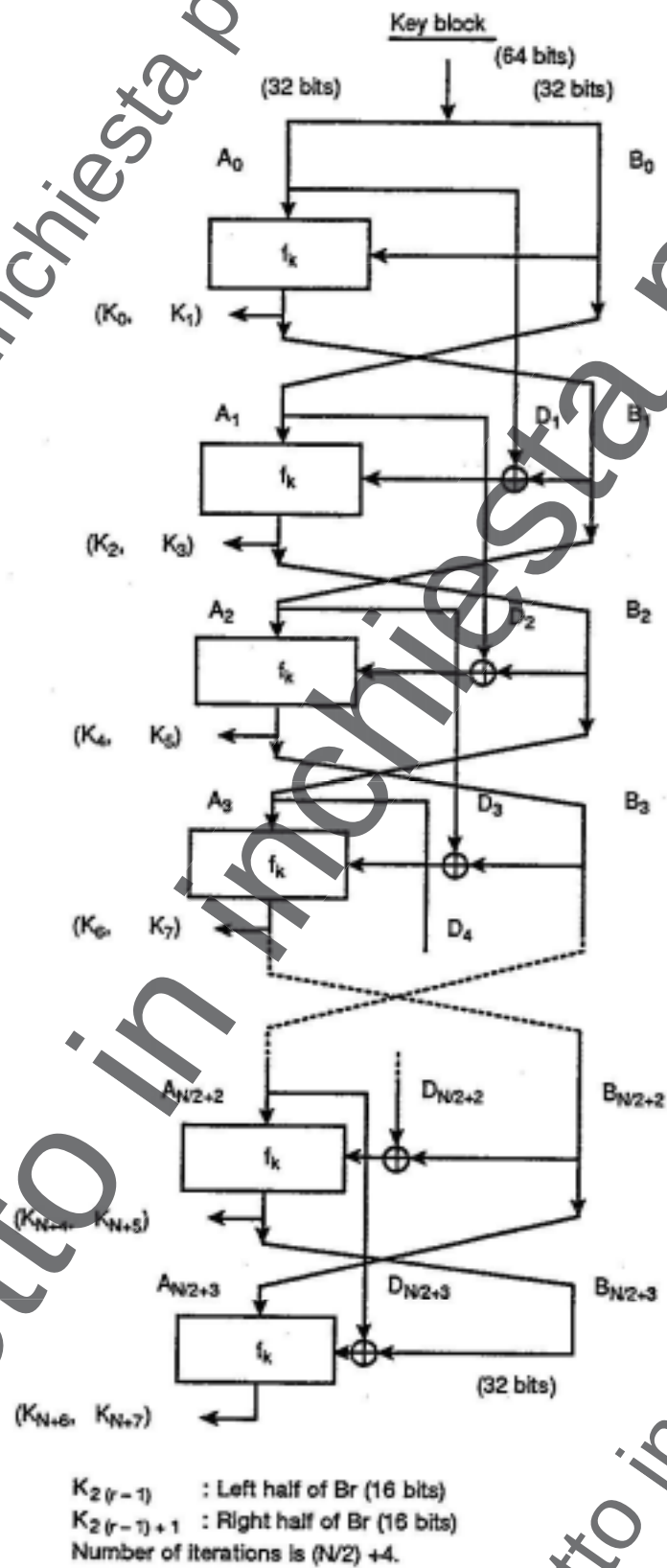
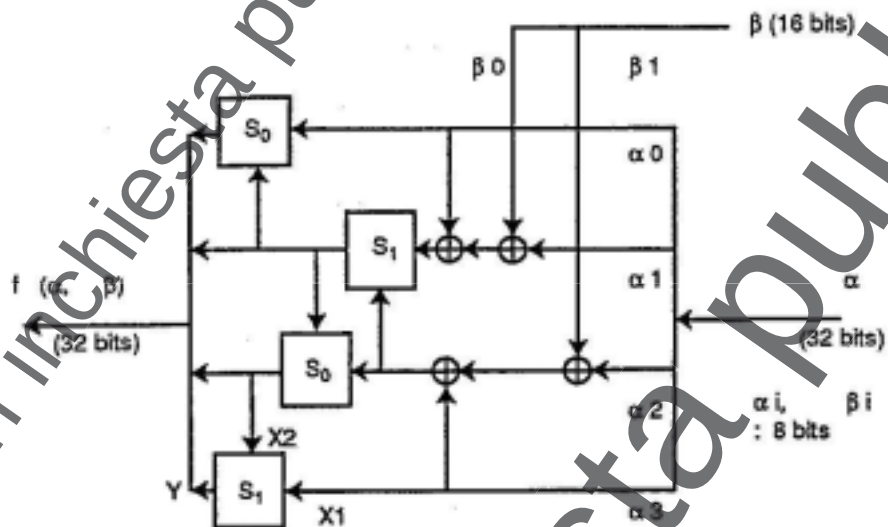


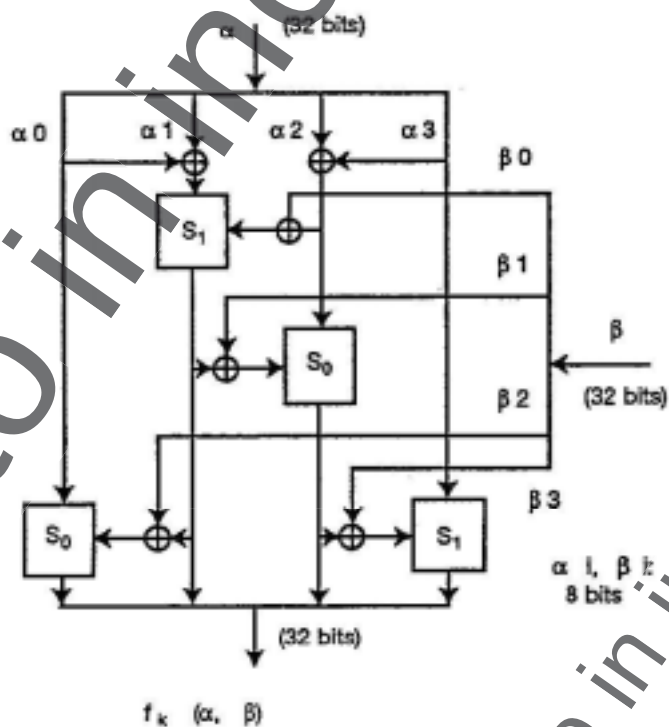
Fig. 2 — Key Schedule

Progetto in inchiesta pubblica



$Y = S_0(X_1, X_2) = \text{Rot2}((X_1 + X_2) \bmod 256)$
 $Y = S_1(X_1, X_2) = \text{Rot2}((X_1 + X_2 + 1) \bmod 256)$
 $Y = \text{output (8 bits), } X_1/X_2 \text{ (8 bits) inputs,}$
 $\text{Rot2}(Y)$: a 2-bit left rotation on 8-bit data Y

Fig. 3 — f -function



Note — S_0/S_1 are the same as S_0/S_1 in f -function.

Fig. 4 — f_k -function

Annex B

Example of Message Authentication for coded character sets: FEAL-MAC

(This Annex does not form part of this Standard)

B.1 Purpose

This Annex presents examples of the entire authentication process, including extraction of message elements, editing, and the MAC computation, for each of the four coded character set options in the Standard, using the algorithm FEAL-MAC.

B.2 Example message

The following message will be used as an example for each of the options. The MACs computed in Sections B.4 through B.7 are each substituted in place of "XXXX XXXX" in the MAC field. This demonstrates how the value of the MAC changes using the different authentication options in the standard.

```
TO YOUR BANK
FROM OUR BANK
QD-80 07 14-DQ///// 10567 QY-127-XQ
QT-

TRNSFR USD $ 1234567,89 FRM ACCNT 48020-168
      ///// TO ACCNT 40210-178

-TQ

KEEP ON QT EXPECT VISIT ON FRIDAY OF
NEW DIV VP ON PROJECT QT-QWERT-TQ BE

Careful
REGARDS

QUIRTO
QK-1357BANKATOBANKB-KQ
QMDXXXX XXXX-MQ
```

B.3 Example cryptographic key

E6 A1 2F 07 9D 15 C4 37

B.4 OPTION 2: Coded characters; entire message; no editing (see 6.5)

B.4.1 Resulting authentication element

The processing of the example message would result in the following authentication element. It is identical to the original message, except that the MAC and its delimiters are not included, and the parity bit for each character is set to zero.

The explicit delimiters are used to locate and identify particular message elements. Mismatched delimiters, intervening delimiters, or an invalid message element format would result in a failure to compute the MAC.

```

TO YOUR BANK
FROM OUR BANK
QD-80 07 14-DQ///// 1056/ QX-127-XQ
QT-

TRNSFR USD $ 1234567,89 FRM ACCNT 48020-166
/////TO ACCNT 40210-178

-TQ

KEEP ON QT EXPECT VISIT ON FRIDAY OF
NEW DIV VP ON PROJECT QT-QWERT-TO BE

Careful

REGARDS

QUIRTO
QK-1357BANKATOBANKB-KQ

```

B.4.2 MAC computation

First Data Block: 0A202020544F2059 (ISO 646 representation of linefeed, space, space, space, T, O, space, Y).

TIME	FEAL IN	REAL OUT	DATA	DATA+OUT=FEEDBACK
1	0A202020544F2059	DF43D2B8061EB1D1	4F55522042414E4B	90168098445FFF9A
2	90168098445FFF9A	D8250A5BCA703CD9	0A0A20202048524F	D22F2A7BEA366E96
3	D22F2A7BEA366E96	B3D7997A0D88C29A	4D204F5552204241	FEF7D62F5FA880DB
4	FEF7D62F5FA880DB	83F9E28D60C50D00	4E4B0A0A20202051	CDB2E88740E52D51
5	CDB2E88740E52D51	DC2C2E2E920B89A0	442D383020303720	9801161EB23BBE80
6	9801161EB23BBE80	7C6DCC7BBAE909D4	31342D4451202F2F	4D59E13FEBC926FB
7	4D59E13FEBC926FB	2B9F9367C4D8904A	2F2F2F2031303536	04B0BC47F5E8A57C
8	04B0BC47F5E8A57C	94C1A9C829D30683	2F202051582D3132	8BE1899971FE37B1

TIME	FEAL IN	FEAL OUT	DATA	DATA+OUT=FEEDBACK
9	BBE1899971FE37B1	AFB8EACC12355E18	372D58510A0A2020	9996B29D163F7E36
10	9996B29D183F7E38	AA8DB672CF3ABA60	2051542D0A0A0A20	8AECF25FC530B040
11	8AECF25FC530B040	E106F2CA5780D00D	202054524E534652	C126A69819D3965F
12	C126A69819D3965F	EF8038ED478713C2	2055534420243132	CFD56BA967A322F0
13	CFD56BA967A322F0	EEB6C5D8DA918CE3	33343536372C3839	DD82F0EEEDBDB4DA
14	DD82F0EEEDBDB4DA	2092328DD68FC7D9	2046524D20414343	00D460C0F6CEB49A
15	00D460C0F6CEB49A	E86DC7E75934839B	4E54203438303230	A639E7D36104B1AB
16	A639E7D36104B1AB	90DC23C2EAE4A58	2D3136360A202020	BDED15F4E0CE6A78
17	BDED15F4E0CE6A78	FBE24F3C352D3D8C	2020202020202020	DBC26F1C150D1DAC
18	DBC26F1C150D1DAC	811FCE984334C455	20202020202F2F2F	A13FEEB8631BEB7A
19	A13FEEB8631BEB7A	0EE66530162381A	2F2F202020202020	21C946732142183A
20	21C946732142183A	5EB61A35467C0F3F	544F04143434E54	0AF93A74053F416B
21	0AF93A74053F416B	1FD05E826EC9649C	2034303231302D31	3FE46EB05FF949AD
22	3FE46EB05FF949AD	FBE0D720C0E8ABB8	37390A0A2020202D	CCD8DD2AE0C88B95
23	CCD8DD2AE0C88B95	8D87E29F940A7D1E	54510A0A2020204B	D9D6E895B42A5D55
24	D9D6E895B42A5D55	443A4689E835E1DB	454550204F4E2051	017F16A9A77BC18A
25	017F16A9A77BC18A	5DA7C63D086A853E	5420455850454354	09878365582FC66A
26	09878365582FC66A	72E81D0273468AA8	205649534954204F	52BE54513A12AAE7
27	52BE54513A12AAE7	F7C6988EB672B054	4E20465249444159	B9E6DEDCFF36FC0D
28	B9E6DEDCFF36FC0D	F0998881903D0F55	204F460A2020204E	D0D6CE8DB01D2F1B
29	D0D6CE8DB01D2F1B	F74236F4636EB678	4557204449562056	B21516B02A38962E
30	B21516B02A38962E	A66866D10B4217A4	50204F4E2050524F	F648299F2B1245EB
31	F648299F2B1245EB	82539562B248F0B1	4A4543542051542D	C816D6369219A49C
32	C816D6369219A49C	C00058FDAA3BEBBF	51574552542D5451	91571DAFFE16BFEE
33	91571DAFFE16BFEE	BED3AFB46D62EFE0	2042450A0A202020	9E91EABE6742CFC0
34	9E91EABE6742CFC0	B462CB365A8A7FF0	4361726566756C0A	F703B9533CFF13FA
35	F703B9533CFF13FA	96DCD9AB0760E4EE	0A20202052454741	9CFCF98B5525A3AF
36	9CFCF98B5525A3AF	69798199CF16455C	5244530A0A202020	3B3DD293C536657C
37	3B3DD293C536657C	4C490FFD947F10B3	51554952544F0A20	1D1C46AFC0301A93
38	1D1C46AFC0301A93	E1B3EB32AFC99950	2020514B2D313335	C193BA7962F8AA65
39	C193BA7962F8AA65	BEBDA0F8DA320AD0	3742414E4B41544F	89FFE1B691735E9F
40	89FFE1B691735E9F	059E36563E0DE563	42414E4B422D4B51	47DF781D7C20AE32
41	47DF781D7C20AE32	382FA72AC86E393B	0A2020200A0A0000	328F870AC264393B
42	320F870AC264393B	EE780F6ED7849AE3		

MAC = EE780F6E

Note that the last four digits of DATA, time 41, represent binary zero "pad" characters (see ISO 8731 Part x).

B.5 OPTION 3: Coded characters; extracted message elements; no editing (see 6.6)

B.5.1 Resulting authentication element

The processing of the example message would result in the following authentication element. The MAC and its delimiters are not included, and the parity bit for each character is set to zero. Only the extraction rules of Section 6.6.2 are applied. All characters other than message elements and their corresponding delimiters are deleted.

The explicit delimiters are used to locate and identify particular message elements. Mismatch delimiters, intervening delimiters, or an invalid message element format would result in a failure to compute the MAC.

```
QD-80 07 14-DOQX-127-XQQT-
TRANSFR USD $ 1234567,89 FRM ACCNT 49020-156
/////TO ACCNT 40210-178
-TQQT-QWERT-TQOK-1357BANKATOBANKB-KQ
```

B.5.2 MAC computation

First Data Block: 51442D3830203037 (ISO 646 representation of Q, D, -, 8, 0, space, 0, 7).

TIME	FEAL IN	FEAL OUT	DATA	DATA+OUT=FEEDBACK
1	51442D3830203037	8B6E0D34DD412747	2031342D44515158	AB5F39199910761F
2	AB5F39199910761F	BC0B17B8C11713BB	2D3132372D585151	913A258FEC4F42EA
3	913A258FEC4F42EA	04A048E65F3823A5	542D0A0A0A202020	908D42EC55180385
4	908D42EC55180385	15A856C2DE412CF9	54524E5346522055	41FA169198130CAC
5	41FA169198130CAC	E4DF49B099645278	5344202431323334	B79B6998B856614C
6	B79B6998B856614C	3E6B1777D8B37DD1	3536372C38392046	0B5D205BE38A5D97
7	0B5D205BE38A5D97	7F0D4944756BED67	524D204143434E54	2D4069053628A333
8	2D4069053628A333	8A0289E8F516CEE2	2034383032302D31	AA36B1D8C726E3D8
9	AA36B1D8C726E3D3	91928CA0C0A920AF	36360A2020202020	A7A48680E089008F
10	A7A48680E089008F	DF676DE058A1B28F	2020202020202020	FF474DC0788192AF
11	FF474DC0788192AF	704E35B7A379774A	2020202F2F2F2F2F	506E15988C565865
12	506E15988C565865	9FDC59713B57BFA0	202020202020544F	BFFC79511B77EBEF
13	BFFC79511B77EBEF	F6D168163586380A	204143434E542034	D6902B557B02183E
14	D6902B557B02183E	5F0276D52B4C0B5F	303231302D313738	6F3047E5067D3C67
15	6F3047E5067D3C67	3D7801197A2E9E44	0A0A2020202D5451	377221395A03CA15
16	377221395A03CA15	3C5E8944A77426AD	51542D5157455254	6D0AA415F03174F9
17	6D0AA415F03174F9	46D51C1FD629B386	2D5451514B2D3133	5B814D4E9D0482B5
18	5B814D4E9D0482B5	4E8CAB6FE8E59413	353742414E4B4154	7B5BE92EA6AED547
19	7B5BE92EA6AED547	51DDE79051C8E26C	4F42414E4B422D4B	1E9FA6DE1A8ACF27
20	1E9FA6DE1A8ACF27	472B4356BA0F86E7	5100000000000000	162B4356BA0F86E7
21	162B4356BA0F86E7	D16BC169EC8D1E52		

MAC = D16BC169

Note that the last fourteen digits of DATA, time 20, represent binary zero "pad" characters.

B.6 OPTION 4: Coded characters; entire message; editing (see 6.7)

B.6.1 Resulting authentication element

The processing of the message would result in the following authentication element. The MAC and its delimiters are not included, and the parity bit for each character is set to zero. Only the editing rules of Section 6.7.2 are applied. Each carriage return and each line feed is replaced by a single space, lower case alphabets are translated to upper case, any characters other than the uppercase letters, digits, space, comma, full stop, hyphen, solidus, asterisk, and open and close parentheses are deleted, leading spaces of the message are deleted, and each sequence of consecutive space is replaced by single space.

The explicit delimiters are used to locate and identify particular message elements. Mismatched delimiters, intervening delimiters, or an invalid message element format would result in a failure to compute the MAC.

TO YOUR BANK FROM OUR BANK QD-80 07 14-DQ / / / / 1056/ QX-1
27-XQ QT- TRNSFR USD 1234567,89 FRM ACCT 48020-166 / / / /
TO ACCT 40210-178 -TQ KEEP ON QT EXPECT VISIT ON FRIDAY O
F NEW DIV VP ON PROJECT QT-QWERT-TQ BE CAREFUL REGARDS QUI
RTO QK-1357BANKATOBANKB-KQ

B.6.2 MAC computation

First Data Block: 544F20594F555220 (ISO 646 representation of T, O, space, Y, O, U, R, space).

TIME	FEAL IN	FEAL OUT	DATA	DATA+OUT=FEEDBACK
1	544F20594F555220	CBF9A9CBDA7064C6	42414E4B2046524F	89B8E780FA363689
2	89B8E780FA363689	AF32023F0390FB9	4D204F5552204241	E2124D6A84194DF8
3	E2124D6A84194DF8	A5CD5648B7110D0F	4E4B2051442D3830	EB867619F33C353F
4	EB867619F33C353F	8B623E7F6E244A2D	2030372031342D44	AB52095F5F106769
5	AB52095F5F106769	524522D691160EB7	51202F2F2F2F2F20	03650DF9BE392197
6	03650DF9BE392197	5B39648AB42236F1	313035362F205158	6A0961BC9B0267A9
7	6A0961BC9B0267A9	60327E8AF2BB8DB8	2D3132372D585120	4DB34CBDDFE3DC98
8	4DB34CBDDFE3DC98	29B72B3FFE8C0AB9	51542D2054524E53	78E3061FAADE44EA
9	78E3061FAADE44EA	F5F56919E9EECEDA	4652205553442031	B3A7494CBAAAEFE8
10	B3A7494CBAAAEFE8	82154648641407D8	3233343536372C38	B026727D52232BE9
11	B026727D52232BE9	AFBA67F35AA28480	392046524D204143	969A21A11782C5C3
12	969A21A11782C5C3	11C7412185A43E56	434E542034383032	52891501B19C0E64
13	52891501B19C0E64	865C81677B692021	302D313636202F2F	B671B0514D490F0E
14	B671B0514D490F0E	BD37F1037988CB8F	2F2F2F20544F2041	9218DE232DC7EBCE
15	9218DE232DC7EBCE	42BE79DC7BE3A8F9	43434E5420343032	01FD37885BD798CB
16	01FD37885BD798CB	EBE1499261694176	31302D313738202D	DAD164A35651615B
17	DAD164A35651615B	08E30253831A976D	5451204B45455020	5CB22218C65FC74D
18	5CB22218C65FC74D	AAE16E8EA8C3AEB0	4F4E205154204553	E5AF4EDFFCE3EBE8

TIME	FEAL IN	FEAL OUT	DATA	DATA+OUT=FEEDBACK
19	E5AF4EDFFCE3EBE8	117FCF9DE801AFBE	5045435420564953	41B58CC9C857E6ED
20	41B58CC9C857E6ED	BAEA2DF0A389E415	4954204F4E204652	F3BE0DBFEDA9A247
21	F3BE0DBFEDA9A247	9FF41D07D51FF498	49444159204F4620	D6B05C5EF550B2B8
22	D6B05C5EF550B2B8	BBFB52CD5CDB63AE	4E45572044495620	F5BE05ED1892358E
23	F5BE05ED1892358E	61DD06252542C20E	5650204F4E205052	378D268A6B62925C
24	378D268A6B62925C	8A99B5B03CD20A9F	4F4A454354205154	C5D3F0F368F25BCB
25	C5D3F0F368F25BCB	47DD1F353F934658	2D51574552542D54	6A8C48706DC76B0C
26	6A8C48706DC76B0C	E4887E5E41D43347	5120424520434152	B5A83C1B61977215
27	B5A83C1B61977215	288541A0D1B9E45C	4546554C2D524547	6DC314ECF1EBA11B
28	6DC314ECF1EBA11B	C320CD2B90671724	4152445320515549	82728978B036426D
29	82728978B036426D	02DF55B5D831F5F3	52544F20514B3D31	508B1A95897AD8C2
30	508B1A95897AD8C2	EE2A7EB7A5FF4A03	33353742414E4B41	DD1F49F5E4B10142
31	DD1F49F5E4B10142	DADA26892125CEF8	544F42414E4B422D	8E9564C86F6E8CD5
32	8E9564C86F6E8CD5	35FAA35F5AA7176D	4B51290000000000	7EAB835F5AA7176D
33	7EAB835F5AA7176D	730D4F4C1F1651BC		

MAC = 730D4F4C

Note that the last ten digits of DATA, time 32, represent binary zero "pad" characters.

B.7 OPTION 5: Coded characters; extracted message elements; editing (see 6.8)

B.7.1 Resulting authentication element

The processing of the message would result in the following authentication element. The MAC and its delimiters are not included, and the parity bit for each character is set to zero. The extraction rules of Section 6.6.2 and the editing rules of Section 6.7.2 are applied. All characters other than message elements and their corresponding delimiters are deleted. Each carriage return and each line feed is replaced by a single space, lower case alphabetic characters are translated to upper case, any characters other than the uppercase letters, digits, space, comma, full stop, hyphen, solidus, asterisk, and open and close parentheses are deleted, leading spaces of the message are deleted, and each sequence of consecutive space is replaced by single space.

The explicit delimiters are used to locate and identify particular message elements. Mismatched delimiters, intervening delimiters, or an invalid message element format would result in a failure to compute the MAC.

QD-80 07 14-DQX-127-XQQT- TRNSFR USD 1234567.89 FRM ACCNT
48020-166 / / / / / TO ACCNT 40210-178 -TQQT-QWERT-TQK-13578A
NKATOBANKB-KQ

B.7.2 MAC computation

First Data Block: 51442D3830203037 (ISO 646 representation of Q, D, -, 8, 0, space, 0, 7).

TIME	FEAL IN	FEAL OUT	DATA	DATA+OUT+FEED BACK
1	51442D3830203037	8B6E0D34DD412747	2031342D44515158	AB5F39199910761F
2	AB5F39199910761F	BC0B17B8C11713BB	2D3132372D585151	913A258FEC4F42EA
3	913A258FEC4F42EA	C4A048E65F3823A5	542D54524E534652	908D1CB4116B65F7
4	908D1CB4116B65F7	49E7A441D1BE31F0	2055534420313233	69B2F705F18F03C3
5	69B2F705F18F03C3	C58A431DD17AF60F	343536372C383920	F1BF752AFD42CF2F
6	F1BF752AFD42CF2F	61C82112EAAAE6CB	46524D204143434E	279A8C32ABE9A585
7	279A8C32ABE9A585	AE046C93792DEE8C	542034383032302D	FA2458AB491FDEA1
8	FA2458AB491FDEA1	93380FD1AB58B432	313636202F2F2F2F	A20E39F184779B1D
9	A20E39F184779B1D	FEE6A7FAFE18F209	2F20544F20414343	D1C6F3B5DE59B14A
10	D1C6F3B5DE59B14A	D4B916F7E0E9CC5A	4E54203430323130	9AED36C3D0DBFD6A
11	9AED36C3D0DBFD6A	D486DF78B5B5836F	2D313738202D5451	F9B7E8409598D73E
12	F9B7E8409598D73E	4D47BBC0F3D5DDB1	51542D5157455254	1C139691A4908FE5
13	1C139691A4908FE5	5E0CB935C8364EAA	2D5451514B2D3133	7358E864831B7F99
14	7358E864831B7F99	2AF014F59AF438FF	353742414E4B4154	1FC756B4D4BF79AB
15	1FC756B4D4BF79AB	2129C00636FFBF97	4F42414E4B422D4B	6E6B81487DBD92DC
16	6E6B81487DBD92DC	26F5E2FC94A14632	5100000000000000	77F5E2FC94A14632
17	77F5E2FC94A14632	4F9679B518E77145		

MAC = 4F9679B5

Note that the last fourteen digits of DATA, time 16, represent binary zero "pad" characters.

La presente Norma è stata compilata dal Comitato Elettrotecnico Italiano e beneficia del riconoscimento di cui alla legge 1° Marzo 1968, n. 186.

Editore CEI, Comitato Elettrotecnico Italiano, Milano – Stampa in proprio
Autorizzazione del Tribunale di Milano N. 4093 del 24 Luglio 1956

Responsabile: Ing. R. Bacci

Comitato Tecnico Elaboratore

CT 79 – Sistemi di rilevamento e segnalazione per incendio, intrusione, furto, sabotaggio e aggressione

Altre norme di possibile interesse sull'argomento

PROGETTO

€ —

