# SmartKey
# Manuale Utente v12.0

## Copyright e marchi registrati

# CONTRATTO DI LICENZA CON L'UTENTE FINALE PER IL PRODOTTO SMARTKEY E SMARTPICO

**INFORMAZIONE IMPORTANTE –** LEGGERE CON ATTENZIONE IL PRESENTE CONTRATTO PRIMA DI UTILIZZARE I CONTENUTI DEL PACCHETTO E/O PRIMA DI SCARICARE O INSTALLARE IL PRODOTTO SOFTWARE. TUTTI GLI ORDINI E L'UTILIZZO DEI PRODOTTI SMARTKEY E SMARTPICO (DI SEGUITO, IL "**PRODOTTO**") FORNITI DA **EUTRONSEC S.P.A.**, SOCIETÀ INTERAMENTE POSSEDUTA DA ALADDIN KNOWLEDGE SYSTEMS LTD, (DI SEGUITO "**ALADDIN**") SONO E SARANNO SOGGETTI AI TERMINI E ALLE CONDIZIONI INDICATE NEL PRESENTE CONTRATTO.

CON L'APERTURA DEL PACCHETTO CONTENENTE I PRODOTTI E/O SCARICANDO IL SOFTWARE (COME DEFINITO PIÙ SOTTO) E/O CON L'INSTALLAZIONE DEL SOFTWARE SUL SUO COMPUTER E/O UTILIZZANDO IL PRODOTTO, LEI ACCETTA IL PRESENTE CONTRATTO E DI ESSERE VINCOLATO DAI SUOI TERMINI E CONDIZIONI.

**QUALORA LEI NON ACCETTI IL PRESENTE CONTRATTO O NON VOGLIA ESSERE VINCOLATO DAL MEDESIMO, NON APRA IL PACCHETTO E/O NON SCARICHI E/O INSTALLI IL SOFTWARE E RESTITUISCA TEMPESTIVAMENTE (AL MASSIMO ENTRO 10 GIORNI DALLA DATA IN CUI HA RICEVUTO QUESTO PACCHETTO) I PRODOTTI AD ALADDIN, CANCELLI IL SOFTWARE, ED OGNI PARTE DI ESSO, DAL SUO COMPUTER E NON LO UTILIZZI IN NESSUNA MANIERA.**

Il presente Contratto si compone di 3 parti:

**Parte I** si applica se Lei scarica o utilizza il Prodotto gratuitamente solo per provarlo.

**Parte II** si applica se Lei ha acquistato o comunque avuto da Aladdin una licenza per utilizzare il Prodotto.

**Parte III** si applica a tutte le concessioni di licenza.

## PARTE I -- NORME APPLICABILI ALLA CONCESSIONE DI UNA LICENZA DI PROVA

**Concessione di Licenza**. Con il presente contratto Aladdin Le garantisce, e Lei accetta, una licenza non esclusiva di utilizzare il Prodotto in linguaggio macchina, solo in forma di codice oggetto, gratuitamente, al fine di valutare se acquistare una licenza per continuare ad utilizzare il Prodotto e solo nella misura in cui è autorizzato dal presente Contratto di Licenza. Il periodo di prova è limitato ad una durata massima di giorni specificata nel pacchetto di prova a Lei applicabile. Lei può utilizzare il Prodotto, durante il periodo di prova, nel modo descritto nella Parte III, paragrafo rubricato "Estensione della concessione".

**ESCLUSIONE DI GARANZIA**. Il Prodotto è fornito "COSI' COM'E'", SENZA GARANZIA DI ALCUN TIPO. NON SI APPLICHERANNO GARANZIE IMPLICITE DI IDONEITA' PER UNO SCOPO SPECIFICO, SODDISFAZIONE E COMMERCIABILITA'. ALCUNE GIURISDIZIONI NON PERMETTONO LE ESCLUSIONI DI GARANZIA IMPLICITA, E DUNQUE LE SUDDETTE ESCLUSIONI DI GARANZIA POTREBBERO NON APPLICARSI A LEI E LEI POTREBBE AVERE ALTRI DIRITTI CHE VARIANO IN RELAZIONE ALLA GIURISDIZIONE. L'intero rischio legato alla qualità e alla prestazione del Prodotto è a Suo carico. La presente esclusione di garanzia costituisce un parte essenziale del contratto.

Qualora Lei abbia inizialmente acquistato una copia del Prodotto senza acquistare una licenza e desidera acquistare una licenza, contatti Aladdin o un qualunque rappresentante Aladdin.

**PARTE II – NORME APPLICABILI QUANDO VIENE CONCESSA UNA LICENZA**

**Concessione di Licenza.** A seguito del pagamento da parte Sua dei canoni di licenza applicabili al tipo ed alla quantità di licenze da Lei acquistate ed indicate nel Suo ordine di acquisto, Aladdin con il presente contratto Le concede, e Lei accetta, una licenza personale limitata, non esclusiva e interamente revocabile, per utilizzare il Software (secondo la definizione del termine contenuta nella Parte III di cui sotto, nel paragrafo "Proprietà Intellettuale"), nella sola forma eseguibile, come descritta nella documentazione di accompagnamento del Software per l'utente e solo secondo i termini del presente Contratto: (i) Lei potrà installare il Software ed utilizzarlo su computer ubicati nel suo posto di lavoro, come descritto nella relativa documentazione di Aladdin; (ii) Lei potrà integrare e collegare il Software con i programmi del Suo computer per il solo scopo descritto nella Guida di SMARTKEY e SMARTPICO; comunque, qualunque parte del Software confluita in un altro programma verrà considerata opera derivata e continuerà ad essere soggetta ai termini del presente Contratto; e (iii) Lei potrà fare un ragionevole numero di copie del Software solamente a fini di back-up. Il Software non dovrà essere utilizzato per nessun altro scopo.

**Sub-Licenza**. Dopo aver integrato il Software al/ai programma/i del Suo computer in conformità all'articolo Concessione di Licenza di cui sopra, Lei potrà concedere in sub-licenza, secondo i termini del presente Contratto, il Software così integrato e rivendere le componenti hardware del Prodotto, che Lei ha acquistato da Aladdin, laddove applicabile, ai distributori e/o agli utenti. Prima di vendere e concedere in sub-licenza, Lei dovrà assicurarsi che i Suoi contratti con tutti i Suoi distributori e/o utenti finali (e che i loro contratti con i loro clienti) conterranno garanzie, esclusioni di garanzie, limitazioni di responsabilità e condizioni di licenza non meno protettive dei diritti di Aladdin rispetto alle disposizioni qui contenute. Inoltre, Lei dovrà chiarire bene ai Suoi distributori e/o agli utenti finali, che Aladdin non è e non sarà, in qualsivoglia circostanza, coinvolta o responsabile in nessun modo del software, e delle relative licenze, contenuto nei Suoi programmi che Lei integra con il Software Aladdin e che distribuirà ai Suoi distributori e/o agli utenti finali, inclusa, solo in via esemplificativa, l'estensione dei termini di licenza e la fornitura di manutenzione per qualunque elemento software e/o programma che non sia il Software Aladdin. Aladdin esclude espressamente qualsivoglia coinvolgimento e responsabilità in relazione a programmi, elementi software, e/o elementi hardware che non sono e non fanno parte del prodotto Aladdin.

**Limitazione di Garanzia**. Aladdin garantisce, solo a Suo vantaggio, che (i) il Software, quando consegnatole, e per un periodo di tre (3) mesi dopo la data di consegna, funzionerà rispettando sostanzialmente la Guida di SMARTKEY e SMARTPICO, a condizione che esso venga utilizzato sull'hardware e con il sistema operativo per cui è stato progettato; e (ii) che SMARTKEY e SMARTPICO sarà sostanzialmente privo di difetti di materiale e di lavorazione significativi per un periodo di dodici (12) mesi dopo la data di consegna. Per i consumatori italiani (come definiti dal D.lgs. 6 settembre 2005, n. 206) i suddetti periodi di garanzia sono pari a 24 mesi.

**Esclusione di Garanzia**. ALADDIN NON GARANTISCE CHE IL PROPRIO PRODOTTO SODDISFERA' LE SUE ESIGENZE O CHE IL FUNZIONAMENTO SARA' PRIVO DI INTERRUZIONI O ERRORI. NEI LIMITI CONSENTITI DALLA LEGGE, ALADDIN ESCLUDE ESPRESSAMENTE TUTTE LE GARANZIE QUI NON INDICATE E TUTTE LE GARANZIE IMPLICITE, INCLUSE, A MERO TITOLO ESEMPLIFICATIVO, LE GARANZIE IMPLICITE DI COMMERCIABILITA' ED IDONEITA' PER UNO SCOPO SPECIFICO. NESSUN CONCESSIONARIO, DISTRIBUTORE, RIVENDITORE, AGENTE O DIPENDENTE DI ALADDIN E' AUTORIZZATO A FARE MODIFICHE, AMPLIAMENTI O AGGIUNTE ALLA PRESENTE GARANZIA. (i) Qualora Lei apporti modifiche al Software o ad una qualunque altra parte del Prodotto durante il periodo di garanzia, (ii) qualora i supporti e SMARTKEY e SMARTPICO siano oggetto di incidente, abuso, od uso improprio, (iii) o qualora Lei violi i termini del presente Contratto, allora la Garanzia di cui alla Parte 2 – paragrafo 3 di cui sopra, cesserà immediatamente. Non si applicherà la garanzia qualora il Software venga utilizzato insieme ad un hardware o ad un

programma diverso dalla versione dell'hardware e del programma con cui il Software dovrebbe essere utilizzato come descritto nella guida di SMARTKEY e SMARTPICO.

**Limitazione di tutela**. In caso di violazione della suddetta garanzia, il solo obbligo di Aladdin, e la Sua unica tutela sarà, a sola discrezione di Aladdin: (i) sostituire o riparare gratuitamente il Prodotto, o il suo componente, che non soddisfa la precedente garanzia limitata; o (ii) restituire il prezzo pagato per il Prodotto, o per il suo componente. Ogni sostituzione o componente riparato verrà garantito per il restante periodo di garanzia originale o per 30 giorni, indipendentemente da quale dei due periodi sia più lungo. Reclami relativi alla garanzia devono essere inoltrati per iscritto, entro sette (7) giorni dalla scoperta del difetto ed accompagnati da elementi di prova adeguati. Tutti i Prodotti dovrebbero essere restituiti al distributore da cui sono stati acquistati (qualora non siano stati acquistati direttamente da Aladdin) e dovranno essere spediti dalla parte che effettua la restituzione con spedizione ed assicurazione pagata. Il Prodotto o il suo componente deve essere restituito con una copia della sua ricevuta. I consumatori italiani (come definiti dal D.lgs. 6 settembre 2005, n. 206) hanno due mesi dalla scoperta del difetto per segnalarlo a Aladdin, e non devono pagare alcun costo per la restituzione del Prodotto a Aladdin.

## PARTE III -- NORME APPLICABILI A TUTTE LE CONCESSIONI DI LICENZA

**Estensione della Concessione ed Usi Proibiti**. Fatta eccezione per quanto espressamente permesso negli Articoli 2.1 e 2.2 di cui sopra, Lei accetta (i) di non utilizzare il Prodotto in qualunque modo che vada oltre lo scopo della licenza da Lei acquistata in conformità al Suo ordine di acquisto; (ii) di non utilizzare, modificare, integrare o concedere in sub-licenza il Software o qualunque altro Prodotto di Aladdin se non espressamente autorizzato dal presente Contratto e dalla Guida di SMARTKEY e SMARTPICO; (iii) di non vendere, concedere in licenza (o in sub-licenza), concedere a noleggio, assegnare, trasferire, concedere in pegno, o condividere con qualcun altro i Suoi diritti derivanti da questa licenza; (iv) di non modificare, disassemblare, decompilare, sottoporre a *reverse engineering*, aggiornare o migliorare il Software o tentare di scoprire il codice sorgente del Software; (v) di non installare il Software in un server accessibile attraverso una rete aperta al pubblico; (vi) di non utilizzare copie di back-up o d'archivio del Software (o permettere che qualcun altro utilizzi tali copie) per scopi diversi dalla sostituzione di una copia originale qualora venga distrutta o diventi difettosa. Se Lei è un membro dell'Unione Europea, il presente contratto non modifica i Suoi diritti derivanti dalla legislazione adottata con la Direttiva del Consiglio sulla Protezione Giuridica dei Programmi per Elaboratori. Qualora Lei cerchi informazioni relative al significato di questa Direttiva, dovrebbe contattare Aladdin.

**Proprietà Intellettuale**. QUESTO E' UN CONTRATTO DI LICENZA E NON UN CONTRATTO DI VENDITA. Il componente software del Prodotto SMARTKEY e SMARTPICO di Aladdin, incluse le revisioni, le correzioni, le modifiche, i miglioramenti, gli aggiornamenti e/o gli upgrade del medesimo (d'ora in avanti, l'intero insieme o ogni parte dello stesso verrà definito come: "**Software**"), e la relativa documentazione, NON SONO IN VENDITA e sono e rimarranno di proprietà di Aladdin. Tutti i diritti di proprietà intellettuale (inclusi, a mero titolo esemplificativo, diritti d'autore, brevetti, segreti commerciali, marchi, ecc.) manifestati attraverso il Prodotto, o incorporati nel Prodotto, e/o allegati/connessi/relativi al Prodotto, (inclusi, solo in via esemplificativa, il codice Software ed il prodotto realizzato in conformità alla Parte II di cui sopra) sono e saranno di esclusiva proprietà di Aladdin. Il presente Contratto di Licenza non Le concede la proprietà del Software ma solo un diritto limitato di uso revocabile in conformità con i termini del presente Contratto di Licenza. Nessuna disposizione del presente Contratto costituisce una rinuncia da parte di Aladdin ai diritti di proprietà intellettuale in base a qualsivoglia legge.

**Registrazioni & Verifiche.** Nel corso del presente Contratto di Licenza con l'Utente Finale, Lei si impegna a conservare le registrazioni e lo schedario relativi a tutte le Attivazioni del Prodotto e delle *floating network seats* utilizzate attraverso i Suoi servizi ("**Registrazioni**"). Aladdin avrà diritto di

controllare le sue registrazioni in ogni momento previa comunicazione scritta. Ognuna di queste verifiche verrà effettuata durante l'ordinario orario lavorativo.

**Risoluzione**. Senza recare pregiudizio a qualsiasi altro diritto, Aladdin potrà risolvere la presente licenza in caso di violazione da parte Sua di qualsiasi termine del presente contratto. In caso di risoluzione da parte di Aladdin, Lei accetta di distruggere, o di restituire ad Aladdin, il Prodotto e la Documentazione e tutte le copie e le parti del medesimo.

**Limitazione di Responsabilità**. La responsabilità complessiva di Aladdin nei Suoi confronti o nei confronti di qualunque altra parte per qualsivoglia perdita o per danni derivanti da pretese, richieste o azioni relative al presente Contratto e/o per le controversie giudiziarie aventi ad oggetto il Prodotto, non eccederà il canone di licenza pagato ad Aladdin per l'utilizzo del Prodotto che ha dato origine all'azione o pretesa, e qualora tale criterio non sia applicabile, allora la responsabilità di Aladdin non eccederà l'importo dei canoni di licenza da Lei pagati ad Aladdin in base al presente contratto durante il periodo di dodici (12) mesi che precede l'evento. IN NESSUNA CIRCOSTANZA E SECONDO NESSUNA TEORIA LEGALE, RESPONSABILITA' EXTRA-CONTRATTUALE, CONTRATTUALE, O ALTRO, ALADDIN O I FORNITORI, RIVENDITORI O AGENTI DI ALADDIN SARANNO RESPONSABILI NEI SUOI CONFRONTI O NEI CONFRONTI DI UNA QUALUNQUE ALTRA PERSONA PER QUALUNQUE DANNO DI QUALUNQUE TIPO, INDIRETTO, SPECIALE, INCIDENTALE O CONSEQUENZIALE INCLUSI, SOLO IN VIA ESEMPLIFICATIVA, DANNI PER PERDITA DI AVVIAMENTO, INTERRUZIONE DELL'ATTIVITA', GUASTO O MALFUNZIONAMENTO DEL COMPUTER, PERDITA DI PROFITTI COMMERCIALI, PERDITA DI INFORMAZIONI COMMERCIALI, DANNI PER LESIONI PERSONALI O TUTTI GLI ALTRI DANNI COMMERCIALI O PERDITE, ANCHE QUALORA ALADDIN FOSSE STATA' INFORMATA DELLA POSSIBILITA' DI TALI DANNI, O PER QUALUNQUE ALTRA PRETESA PROVENIENTE DA QUALSIASI ALTRA PARTE. ALCUNE GIURISDIZIONI NON PERMETTONO L'ESCLUSIONE O LA LIMITAZIONE DI DANNI ACCIDENTALI O CONSEQUENZIALI O PER LESIONI PERSONALI O IN CASO DI DOLO O COLPA GRAVE DI ALADDIN, PERTANTO QUESTA LIMITAZIONE ED ESCLUSIONE POTREBBE NON APPLICARSI A LEI.

**NESSUN'ALTRA GARANZIA**. Fatta eccezione e nei limiti di quanto qui previsto specificamente, Aladdin non presta nessuna garanzia, nè espressa né implicita, relativa ai suoi Prodotti, inclusa la garanzia sulla loro qualità, rendimento, commerciabilità o idoneità per uno scopo specifico.

**Controlli all'esportazione**. Lei riconosce che il Prodotto è soggetto a determinate leggi, norme, regolamenti in materia di controllo sulle esportazioni incluse, solo in via esemplificativa, le leggi, le norme, ed i regolamenti degli Stati Uniti e/o di Israele sul controllo delle esportazioni, e Lei pertanto accetta che il Prodotto non verrà consegnato, trasferito, o esportato in qualunque altro paese o utilizzato in qualunque maniera proibita dalla legge applicabile.

**Legge Applicabile & Giurisdizione**. Il presente Contratto verrà interpretato e regolato secondo le leggi italiane (fatta eccezione per le norme sui conflitti di legge) e solamente i tribunali italiani avranno giurisdizione su ogni conflitto o controversia derivante dal presente Contratto. E' espressamente esclusa l'applicazione della Convenzione delle Nazioni Unite sui Contratti di Vendita Internazionale di Beni. Il mancato esercizio, da parte di ognuna delle parti, dei diritti concessi dal presente contratto non verrà considerata una rinuncia di quella parte al successivo esercizio di diritti o azioni in caso inadempimenti futuri.

**Software di terzi**. Qualora il Prodotto contenga un qualsiasi software fornito da terzi, tale software di terzi verrà fornito "Così come è" e sarà soggetto alle disposizioni e condizioni indicate nei contratti contenuti/allegati a tale software. Nel caso in cui tali contratti non siano disponibili, tali software di terzi verranno forniti "Così come sono" senza garanzia di alcun tipo ed il presente Contratto verrà applicato a tali terzi fornitori di software e ai software di terzi come se fossero rispettivamente Aladdin ed il Prodotto.

**Varie**. Il presente Contratto rappresenta l'intero accordo relativo alla presente licenza e può essere modificato da entrambe le parti solo per iscritto. L'ACCETTAZIONE DI UN QUALUNQUE ORDINE DI ACQUISTO PRESENTATO DA LEI E' ESPRESSAMENTE CONDIZIONATO ALLA SUA ACCETTAZIONE DEI TERMINI QUI INDICATI NONCHÉ DELLO SCOPO E DEI TERMINI EVENTUALMENTE INDICATI NEL SUO ORDINE DI ACQUISTO. Qualora una disposizione del presente Contratto venisse ritenuta non applicabile, tale disposizione sarà modificata solo per quanto necessario al fine di renderla applicabile. Il mancato esercizio, da parte di ognuna delle parti, dei diritti concessi dal presente contratto, o il mancato esercizio di un'azione contro l'altra parte in caso di violazione del presente contratto non verrà considerata una rinuncia di quella parte al successivo esercizio dei diritti o delle azioni successive in caso di future violazioni.

# Certificazioni

## CE Compliance

La chiave elettronica SmartKey è risultata in conformità alle seguenti norme:
**CEI EN 61000-4-2; CEI EN 61000-4-3; CEI EN 55022**
come richiesto da:
**CEI EN 61000-6-1, CEI EN 61000-6-2, CEI EN 61000-6-3, CEI EN 61000-6-4**
che sono specifiche per i seguenti test:
"Test di immunità ESD"
"Test di immunità ai campi EM a radio-frequenza irradiati"
"Test di Emissione Irradiata"
In conformità ai "Requisiti Essenziali" delle Direttive Europee EMC 89/336/CE e 2004/108/CE

CONFORMITA' ALLA CEI EN 60529 (IP67)
Questa chiave elettronica è conforme con I requisiti essenziali definiti nella CEI EN 60529 (IP67) concernenti la sicurezza (EN 60529:1991-10 + EN 60529 corr:1993-05 + EN 60529/A1:2000-02) così come richiesti dalla direttiva BT.

## FCC Compliance

FCC ID: TFC-AAI

Eutronsec Spa
SmartKey
Supply: 5V DC
Absorption: 30 mA

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

NOTE: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio
frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception,
which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

Caution: changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

IMPORTANT REMARKS
Due to the limited space on the product shell, all FCC certification references are on this technical manual.

# Indice

# 1 Introduzione

## *1.1 Obiettivi del manuale*

Lo scopo del presente manuale è di fornire una panoramica completa degli ambiti applicativi di SmartKey e delle potenzialità operative del prodotto in generale.

Gli argomenti trattati in questo manuale sono:

- l'importanza di proteggere il software.
- le modalità d'uso di SmartKey per quanto riguarda la protezione manuale.
- le modalità d'uso per quanto riguarda la protezione automatica: il presente manuale tratta in modo esaustivo le tecniche e gli strumenti per la protezione automatica del software e dei dati.
- le diverse situazioni applicative e modalità d'implementazione della protezione nell'ambito di reti locali.

## *1.2 A chi è rivolto il manuale*

La consultazione del presente manuale è utile a coloro che:

- si avvicinano per la prima volta alle tecniche di protezione del software e vogliono una panoramica completa sullo stato dell'arte e indicazioni pratiche ed efficaci.
- vogliono avere a disposizione le tecnologie tradizionali di protezione manuale, ma allo stesso tempo strumenti sicuri e versatili per la protezione automatica del software e dei dati.
- hanno l'esigenza di controllare l'autorizzazione all'esecuzione del software: con poche operazioni, la tecnologia SmartKey consente, infatti, utilizzando gli strumenti forniti con il *Developer Kit*, di realizzare sofisticati meccanismi di controllo licenze e parametri d'esecuzioni, con il semplice utilizzo di una chiave *SmartKey NET*.

## *1.3 Come contattare Aladdin*

Il modo migliore per contattare Aladdin riguardo SmartKey è spedire un'email all'helpdesk helpdesk@eutronsec.it.

Si può anche telefonare al Servizio di Assistenza al numero 035697055 od inviare un fax al numero 035697092.

Per contatti di tipo commerciale è possibile telefonare allo 035697080 o inviare un'email all'indirizzo info@eutronsec.it.

# 2 Perché proteggere il software?

La duplicazione illegale di un programma è una pratica molto diffusa: essa è molto semplice, poco costosa e non richiede apparecchiature complesse e dispendiose. I vari metodi usati per impedire o, per lo meno, rendere difficile la copia non si sono dimostrati efficaci, perché eludibili nel giro di pochi mesi, se non pochi giorni. Con SmartKey si affronta il problema in modo differente: non si impedisce la copia, ma l'uso del programma da parte di persone non aventi diritto.

## *2.1 La minaccia della pirateria aumenta la necessità di protezione*

In un mondo dominato dalla tecnologia, il software è diventato molto importante. Lo sviluppo del software richiede enormi risorse in termini di tempo, lavoro, soldi. Per questo motivo, il software è diventato una proprietà intellettuale dello sviluppatore o della azienda che lo produce. Ma questo diritto degli sviluppatori è spesso violato causando, quasi sempre, gravi perdite economiche. Da qui la necessità di proteggere il software.

Le aziende hanno la necessità di controllare le loro proprietà intellettuali per proteggersi dai pirati informatici che alterano e distribuiscono il software ad utenti senza licenza e senza registrazione. D'altra parte, i mancati guadagni dovuti alla pirateria hanno un effetto a catena con il risultato di minori ritorni economici per gli sviluppatori, i dipendenti e tutti coloro che lavorano nel settore dell'informatica. Questo mette un punto di domanda sulla disponibilità d'ulteriori fondi per la ricerca e lo sviluppo, per il pagamento di sviluppatori qualificati e, perfino, per il marketing di nuovi prodotti.

Un'altra ragione per cui la pirateria è diventata così diffusa è che la falsificazione di questo tipo di proprietà intellettuale permette la produzione di copie perfette e funzionali. A questo si aggiunge la possibilità di grandi profitti dalla vendita illecita di software.

Mentre i singoli paesi hanno differenti livelli di pirateria, la Business Software Alliance (BSA) ha reso noto che nel 2005 circa il 40% del software in uso nel mondo era pirata. Questo ha portato, in quello stesso anno, alla perdita di beni per oltre 10 miliardi di dollari. Negli Stati Uniti BSA stima che nel 2005 il 25% del software fosse pirata. Leggi lassiste e l'assenza di volontà politica nei paesi in via di sviluppo di prevenire la pirateria hanno solo peggiorato il problema.

## *2.2 Ragioni commerciali per proteggere il software*

La pirateria ha messo gli sviluppatori di software sulla difensiva. I dirigenti considerano nei loro business plan i mancati guadagni dovuti alla pirateria. Per le allarmanti enormi perdite, le ragioni commerciali per realizzare una soluzione per la protezione del software hanno guadagnato l'attenzione di un numero maggiore d'addetti al settore.

Tra la comunità degli sviluppatori si sta realizzando l'idea che ci potrebbe essere un buon equilibrio tra i costi per implementare la protezione software e i benefici da essa derivanti. In aggiunta, per convincere i dirigenti dell'economicità della protezione del software, gli sviluppatori devono considerare i seguenti fattori:

- La percentuale di software piratato nella propria nicchia di mercato, se si opera in una nicchia di mercato. (Se la percentuale è molto alta, si corre anche il rischio di perdere quella nicchia).
- La percentuale di software piratato nelle nazioni in cui si intende vendere il software.
- Considerati i primi due fattori, è necessario e utile aggiungere la protezione alla propria applicazione?

## *2.3 Pirateria? Una rovina per l'industria del software*

La pirateria informatica ha messo in pericolo il concetto stesso di proprietà intellettuale. Semplicemente, la pirateria informatica è la pratica di copiare e usare un prodotto software senza il permesso del proprietario o dello sviluppatore. Solo da poco si sta diffondendo tra gli utenti l'idea che l'uso e la duplicazione del software proprietario sia illegale, sebbene molti ancora mostrano un generale disinteresse nel trattare il software come proprietà intellettuale di valore.

La pirateria informatica si manifesta in diversi modi. Di seguito, sono spiegati alcuni casi:

- Furto del software: significa comprare una singola licenza e caricare il software su più computer, contrariamente alle clausole del contratto.
- Upload e download: rendere disponibili copie non autorizzate del software ad utenti collegati con modem ad un provider o ad Internet.
- Contraffazione del software: duplicare illegalmente il software e venderlo facendolo apparire come originale.
- OEM unbundling: vendere software originariamente "standalone", facendolo passare come parte integrante del proprio hardware.
- Hard disk loading: installazione illegale su hard disk dei personal computer, spesso, come incentivo per gli utenti finali a comprare l'hardware da un particolare rivenditore di hardware.
- Noleggio: vendita non autorizzata di software per uso temporaneo, come si noleggiasse una videocassetta.

Esistono anche diverse tipologie di pirati informatici:

- Rivenditori che vendono hardware con software illegale preinstallato.
- Aziende che usano copie non autorizzate del software per uso interno.
- Persone che guadagnano contraffacendo il software
- Una qualsiasi persona che fa una copia illegale di un programma di un'altra persona.

### *2.4 Storia dei sistemi di protezione del software*

Durante gli anni settanta, la maggior parte del software era di proprietà di uno specifico ambiente commerciale e funzionava su sistemi mainframe. Quindi, la pirateria non esisteva. Non appena i computer passarono da un'architettura mainframe ad una client/server, l'installazione del software sulla propria workstation divenne la norma e nacque il concetto di distribuzione e licenza.

Gli anni 80 videro la rapida diffusione dei personal computer e la conseguente nascita di una rete pirata underground a larga scala per la distribuzione illegale di software, rendendo necessaria la protezione del software. Gli anni 90 videro la pirateria fare un ulteriore salto in avanti grazie ad Internet, che aprì nuovi canali per la distribuzione illegale. Le nuove tecnologie, come i masterizzatori di CD, resero la duplicazione e la distribuzione di software illegale ancor più facile, contribuendo alla crescita ulteriore della pirateria.

La battaglia per sconfiggere la pirateria iniziò negli anni 80 usando in ambiente DOS meccanismi di protezione software come speciali formattazioni del disco ottenute usando funzioni non proprie del sistema operativo (tracce addizionali, tracce non formattate, scambio di settori del disco, modifica della velocità di rotazione del disco, etc.) o alterando il supporto magnetico.

Si diffusero anche altri meccanismi di protezione, come inibire la copia con i comandi standard di DOS (e, quindi, anche le copie di riserva). Quando le tecnologie per la protezione del software migliorarono s'incominciò ad integrare sistemi hardware di sicurezza con applicativi per la gestione delle licenze.

Con la crescente necessità di protezione si sono sviluppate tecniche per rendere il software e gli applicativi sempre più sicuri. Nonostante questo, i sostenitori della pirateria ancora sopravvivono, e crescono causando ingenti danni economici. Il sempre crescente sviluppo tecnologico è uno dei principali fattori che rende possibile la pirateria, i possibili guadagni illegali ne sono invece il fattore motivante. La pirateria non sembra destinata a terminare nel breve periodo. Tutto questo aumenta la necessità di una soluzione per la protezione del software.

### *2.5 SmartKey introduce una nuova era nella protezione del software*

Aladdin ha creato una tecnologia innovativa che si differenzia dal tradizionale paradigma di protezione. La soluzione SmartKey resiste efficacemente alla pirateria e ai metodi di cracking universalmente diffusi. L'attenzione della strategia di protezione del software si è spostata dalla "protezione della copia" ad un più realistico "controllo dell'esecuzione del software". La prevenzione dalla pirateria informatica va oltre la mera inibizione delle copia abusiva. Si è evoluta nel limitare l'uso del software ad una copia per volta.

La chiave hardware per la protezione del software SmartKey è progettata principalmente per le esigenze di questo tipo di sicurezza. Usando i più avanzati meccanismi anti-hacking, SmartKey è il sistema di protezione hardware che offre la migliore soluzione per i venditori di software.

SmartKey è un piccolo dispositivo elettronico, che può essere inserito nel connettore parallelo o USB di ogni computer. Ciascuna chiave è caratterizzata da un'unica *firma* digitale personalizzata che può essere riconosciuta dal software che deve essere protetto.

La crittografia è il cuore di questa tecnologia per la protezione software. Usando la combinazione di algoritmi e chiavi di crittografia, SmartKey resiste ai tentativi più avanzati di violazione da parte dei pirati. SmartKey inoltre supera gli svantaggi di una normale protezione contro la copia, perché permette all'utente di fare copie di riserva del proprio programma protetto.

# 3 Introduzione a SmartKey

Immaginate di poter limitare l'esecuzione di un qualunque programma applicativo per PC, in modo che siate voi a decidere su quali e quanti calcolatori possa essere eseguito il vostro software. Ebbene, SmartKey è una chiave di protezione del software, cioè un dispositivo hardware che svolge proprio questa funzione e ha lo scopo di impedire la diffusione illegale del software. Il programma protetto, tramite una chiamata a sistema, controlla se SmartKey è presente sul computer stesso o sul computer server SmartKey, nel caso di una configurazione "rete". Se la chiave non è presente, il programma si blocca. La chiamata a sistema, di solito, avviene poco dopo che il programma è stato messo in esecuzione, ma può avvenire più volte e quando il programmatore lo ritiene più opportuno.

La SmartKey è disponibile in due versioni:

- *SmartKey Parallela*: la SmartKey si collega al computer tramite la porta parallela.
- *SmartKey USB*: la SmartKey è collegata al computer tramite la porta USB.

Entrambe i modelli sono piccoli e occupano poco spazio, meno di una scatola di fiammiferi. L'uso di *SmartKey Parallela* non pregiudica l'uso della porta parallela per la stampante. Infatti, alla *SmartKey Parallela* si possono collegare altre *SmartKey Parallele* e il cavo della stampante o di un qualsiasi altro dispositivo che usa la porta parallela. Più *SmartKey USB* possono essere collegate al sistema usando hub USB. L'uso di SmartKey non rallenta il sistema né crea conflitti hardware o software con altri dispositivi e applicativi. *SmartKey Parallela* deve essere inserita nella porta parallela e collegata agli altri dispositivi prima dell'accensione del computer. *SmartKey USB*, invece, può essere inserita o tolta anche durante il funzionamento del computer.

A livello d'applicativi l'uso dei due tipi di SmartKey è uguale: un programma protetto con una *SmartKey Parallela* può essere usato con una *SmartKey USB* e viceversa senza fare alcuna modifica al software. La gestione hardware dei due tipi di SmartKey è completamente delegata ai driver forniti.

I dispositivi SmartKey sono disponibili in cinque modelli differenti che si distinguono per le funzionalità offerte:

- *SmartKey FX (Fixed)* Con algoritmi e codici di sicurezza fissi.
- *SmartKey PR (Programmable)* A codici di sicurezza programmabili e memoria interna.
- *SmartKey EP (Extended Protection)* Come PR ma con prestazioni di sicurezza estese.
- *SmartKey SP (Super Protection)* Come EP ma con più memoria disponibile e algoritmi di sicurezza programmabili dall'utente.
- *SmartKey XM (Extented Memory)* Come SP ma con una elevata capienza di memoria.
- *SmartKey NET (Network)* Per applicazioni in rete.

*SmartKey SP* e *XM* sono quelle che offrono il maggiore grado di sicurezza. *SmartKey NET* permette, inoltre, di proteggere programmi installati su più computer collegati in rete. È la più costosa, ma con un'unica chiave si proteggono i programmi di una rete di computer.

SmartKey è compatibile con Linux, Mac OS X e tutti i sistemi operativi della Microsoft Windows. Sono forniti sia i driver che le librerie per questi sistemi operativi. *SmartKey Parallela*, però, non è supportata sotto Mac OS X e *SmartKey USB* non può essere usata con quei sistemi operativi che non supportano la porta USB. La *SmartKey USB* è anche disponibile in versione *Driver Less* (*DL*), che non richiede l'installazione di alcun driver aggiuntivo nel sistema operativo per il corretto funzionamento.

Le librerie fornite permettono di scrivere programmi protetti in Windows, Linux e Mac OS X.

Per i sistemi operativi Windows di Microsoft, inoltre, sono disponibili programmi con interfaccia grafica che agevolano l'installazione, la configurazione e l'uso di SmartKey. Con l'ausilio di *Global Security System (GSS)* è possibile proteggere automaticamente un programma senza avere particolari conoscenze informatiche e senza avere a disposizione il codice sorgente del programma.

*GSS* non è disponibile sotto Linux e Mac OS X e, quindi, in questi due sistemi operativi, è possibile solo la protezione manuale.

Le seguenti tabelle riassumono il supporto fornito per i sistemi operativi Windows, Linux, Mac OS X.

| **Sistema Operativo** | **Utility di supporto** | **Protezione Automatica** | **SmartKey 2 Parallele** | **SmartKey 3 USB** | **SmartKey 3 USB DL (Driver Less)** |
|---|---|---|---|---|---|
| Windows i386 | ✓ | ✓ | ✓ | ✓ | ✓ |
| Windows x64 | ✓ | | ✓ | ✓ | ✓ |

| | | | | | |
|---|---|---|---|---|---|
| Linux i386/amd64 | | | ✓ | ✓ | ✓ |
| Mac OS X Intel/PowerPC | | | | ✓ | ✓ |

**Tabella 1** Software disponibile.

### *3.1 A chi serve SmartKey?*

SmartKey è destinata principalmente alle *software-house*, perché esse hanno la necessità di proteggere il proprio software dalla diffusione e dalla copia illegale.

Tuttavia SmartKey non serve solo agli sviluppatori di software: anche utenti finali come responsabili aziendali e dei sistemi informativi, responsabili delle vendite possono trarre grandi benefici dalla sua tecnologia. Tramite la tecnologia *GSS*, I programmi possono essere protetti senza la conoscenza dei codici sorgente.

- **Le software-house** usano SmartKey per impedire la pirateria del software e la diffusione illegale dei propri applicativi. Solo gli utilizzatori in possesso della chiave possono far eseguire applicazioni protette.
- **I responsabili delle vendite** usano SmartKey per consegnare ai potenziali clienti copie dimostrative pienamente funzionanti dei programmi, senza però il rischio che siano effettuate copie operative. Mediante SmartKey, essi possono addirittura monitorare il numero d'esecuzioni effettuate dei programmi.
- **I responsabili delle aziende** e **del sistema informativo** usano SmartKey per evitare le responsabilità civili e penali per il furto di software da parte dei dipendenti. Con SmartKey gli accordi di *site-license* del software acquistato sono posti al riparo dalle tentazioni degli utilizzatori.
- **I responsabili dei laboratori informatici** di scuole e università utilizzano SmartKey per evitare la responsabilità della copia abusiva del software da parte degli studenti. Il software ad uso scolastico è infatti acquistato a condizioni particolari e pesanti sanzioni intervengono quando il loro uso è effettuato al di fuori degli ambienti scolastici.

Gli utenti finali non devono preoccuparsi di come utilizzare SmartKey: essi devono semplicemente inserirla nella porta parallela o nella porta USB e dimenticarsela. SmartKey protegge le applicazioni restando completamente trasparente all'utilizzatore.

### *3.2 Quali sono le caratteristiche di SmartKey?*

SmartKey si basa sull'utilizzo di chip elettronici, microprocessori dedicati e algoritmi che implementano funzioni di sicurezza.

- **Elevata sicurezza**: è impossibile una clonazione via hardware grazie alla implementazione di algoritmi per la interrogazione della chiave (scrambling).
- **Installazione su porta parallela**: *SmartKey Parallela* è installata sulla porta parallela di un qualunque personal computer.
- **Installazione su porta USB**: *SmartKey USB* è installata sulla porta USB di un qualunque personal computer con almeno una porta USB.
- **Interrogazione algoritmica:** SmartKey è dotata di meccanismi d'interrogazione di tipo algoritmico utilizzabili sia per proteggere il software sia per codificare dati riservati**.**
- **Personalizzazione dei codici:** ogni SmartKey è singolarmente personalizzata con un codice interno predisposto in fabbrica e diverso per ogni utilizzatore.
- **Programmabilità dei codici:** SmartKey è dotata di codici supplementari programmabili dall'utilizzatore. Non sono necessari dispositivi di programmazione particolari, ma solo le utility software in dotazione.
- **Memoria interna:** fino a 8192 byte di memoria non volatile in lettura e scrittura oltre a 16+16 byte di codici d'accesso sono disponibili all'interno di SmartKey.
- **Utilizzo standalone e in rete:** sono disponibili modelli adatti sia per la protezione di software stand-alone e sia per applicazioni in rete locale.
- **Protezione di programmi eseguibili**: la tecnologia *Global Security System (GSS)* consente di proteggere programmi in formato eseguibile anche senza la disponibilità dei codici sorgente.
- **Interfacciamento al software:** SmartKey è utilizzabile con i principali ambienti di sviluppo e sistemi operativi. Tra cui Windows98, Windows Me, Windows NT, Windows 2000, Windows XP, Windows 2003, Windows Vista, Linux, Mac OS X, AutoCAD.
- **Autoalimentazione:** non usa batterie interne né necessita di un'alimentazione esterna.
- **Dimensioni ridotte**: Le dimensioni esterne sono molto contenute e adatte anche per applicazioni con notebook e laptop.

- **Impilabilità** (solo per *SmartKey Parallele*): Più dispositivi *SmartKey Parallele* possono essere impilate sulla medesima porta parallela, con una connessione in cascata (*daisy-chain*). Questa caratteristica non è stata implementata nella *SmartKey USB*, perché il protocollo USB stesso permette di collegare in cascata più dispositivi tramite HUB USB.
- **Trasparenza** (solo per *SmartKey Parallela*): *SmartKey Parallela* non sottrae l'uso della porta parallela, perché è passante. In cascata alla chiave può essere connessa la stampante o la maggior parte delle numerose periferiche per porta parallela (adattatore di rete, adattatore SCSI, harddisk portatile, altre chiavi di protezione, etc.). L'unico vincolo è che *SmartKey Parallela* e i dispositivi ad essa connessi vanno installati prima di accendere il computer.

### *3.3 Come funziona SmartKey?*

Il principio d'utilizzo di SmartKey consiste nella sua inserzione nella porta parallela o nella porta USB del PC e nel controllo della sua presenza da parte del software protetto. Ogni chiave è caratterizzata da "credenziali digitali" uniche personalizzate per ogni utilizzatore che vengono riconosciute dal software e ne permettono il corretto funzionamento.

Immediatamente dopo la partenza del programma, o in altri punti strategici, il software verifica se SmartKey è presente sulla porta del PC. Se l'esito ha risultato negativo il programma arresta la sua esecuzione. Nel caso invece la chiave sia presente, il programma continua correttamene la propria esecuzione, eventualmente effettuando ulteriori controlli sui parametri della chiave.

Una memoria non volatile interna permette di implementare criteri di protezione selettivi o personalizzare singolarmente le chiavi installate, ad esempio in modo che vi sia una corrispondenza tra numero di serie del software da proteggere e contenuto della chiave. SmartKey può essere utilizzata per particolari strategie di marketing quali ad esempio la concessione di pacchetti demo, affitto di software a tempo, misura dell'utilizzo del software, controllo delle successive versioni, etc.

SmartKey è una chiave di protezione del software sicura e flessibile, predisposta per essere inserita in modo trasparente sulla porta parallela o sulla porta USB del computer. Ogni chiave SmartKey ha codici unici e personalizzabili, così che ogni software-house può implementare propri sistemi di protezione originali e sicuri.

### *3.4 Quanto tempo serve per proteggere un'applicazione?*

La flessibilità di SmartKey consente la definizione di diversi livelli di protezione. Essi usano le risorse della chiave in modo diverso e richiedono un maggiore o minore tempo d'implementazione:

- **Pochi minuti**: grazie al programma *Global Security System (GSS)*, un programma in dotazione si può proteggere direttamente il vostro file eseguibile in breve tempo, senza intervenire su sorgenti. Di fatto è possibile realizzare l'intero processo di protezione (cioè predisporre i codici personalizzati, programmarli nella chiave, selezionare l'applicazione e proteggerla) in pochi minuti.
- **Alcune ore**: è il caso più comune. Non avete molto tempo a disposizione, ma non preoccupatevene: poche ore sono più che sufficienti per implementare le principali funzioni di sicurezza della chiave per uno schema di protezione personale, utilizzando i software driver in dotazione e ottenendo un programma con un elevatissimo livello di protezione.
- **Qualche giorno**: in pochi giorni si può scrivere un software quasi impossibile da violare sfruttando appieno le caratteristiche di SmartKey e adattandole alle proprie esigenze di sicurezza.

### *3.5 Qual'è il grado di sicurezza di SmartKey?*

SmartKey rappresenta lo stato dell'arte in termini di sicurezza del software. L'uso di microprocessori e di sofisticati algoritmi di crittografia rende virtualmente impossibile la clonazione della chiave.

Sia che si decida di proteggere le applicazioni tramite l'utilizzo delle funzioni in dotazione o tramite l'utility automatica *GSS*, SmartKey fornisce un elevato grado di sicurezza.

Per quanto riguarda *GSS*, questo programma non si limita semplicemente a proteggere le applicazioni, ma effettua un'operazione di crittografia del software usando i valori dei registri memorizzati in SmartKey come *chiave di codifica*. Quando è mandata in esecuzione, l'applicazione protetta con *GSS* si autodecodifica istantaneamente utilizzando i registri presenti in SmartKey. Senza SmartKey, l'applicazione non sarà mai decodificata.

### *3.6 Applicativi per SmartKey*

SmartKey è dotata del seguente software di supporto:

- GSS, *Global Sicurity System*, utility per la protezione rapida e diretta dei file eseguibili senza nessuna modifica ai sorgenti dell'applicazione.
- SPC, *SmartKey Programming Central*, utility per la programmazione e personalizzazione delle chiavi con i propri codici di sicurezza.

- SDI, *Smart Driver Installation*, utility per l'installazione manuale ed automatica dei driver.
- SCC, *SmartKey Programming Central*, utility per la configurazione dei server e dei client di una rete SmartKey.
- Driver d'interfacciamento per i vari linguaggi di programmazione e per i vari sistemi operativi.

Oltre a questi tool, sono forniti alcuni programmi scritti per i principali ambienti di sviluppo per apprendere come usare le funzioni delle librerie fornite.

Gli applicativi descritti in questo paragrafo sono disponibili solo per l'ambiente Windows.

## *3.7 SmartKey per Linux e Mac OS X.*

SmartKey può essere usata con i sistemi operativi Linux e Mac OS X, anche se con alcune limitazioni rispetto ai sistemi operativi Windows.

In ambiente Linux, SmartKey può essere usata solo per la protezione manuale standalone del software. In ambiente Mac OS X si può usare SmartKey per la protezione manuale standalone e multilan.

L'installazione di SmartKey sotto Linux e Mac OS X è spiegata nel capitolo 13.

In ambiente Linux sono supportate le *SmartKey Parallele,* le *SmartKey 3 USB* e le *SmartKey 3 USB DL*. In ambiente Mac OS X sono supportate le *SmartKey 3 USB* e le *SmartKey 3 USB DL*.

## *3.8 Getting Started*

La semplicità d'uso di SmartKey e il suo kit di sviluppo permettono di iniziare ad usarla in breve tempo.

### 3.8.1 Installazione

L'installazione dei programmi per l'uso di SmartKey e i suoi driver avvengono in due fasi.

- Installazione dei programmi SmartKey Control Central (SCC), SmartKey Driver Installation (SDI), SmartKey Global Security System (GSS), SmartKey Programming Central (SPC), SmartKey Remote Update (SRU) e installazione del kit di sviluppo.
- Installazione dei driver per *SmartKey USB*, per *SmartKey Parallela* e per *Global Security System.* (solo il programma *Global Security System* ha bisogno di driver propri.)

Inserendo nel lettore il cdrom d'installazione, parte automaticamente la procedura d'installazione dei programmi. Per portarla a termine occorre solo specificare la directory dove installare i programmi se non si vuole usare quella di default. Installati i programmi, bisogna lanciare il programma *SmartKey Driver Installation* (*SDI*) e selezionare il driver da installare.Per proteggere automaticamente un file eseguibile occorre installare il driver della SmartKey (USB o Parallela) e il driver del *GSS.* L'installazione di *SmartKey Parallela* richiede che la SmartKey stessa sia collegata al computer e all'eventuale stampante prima dell'accensione del computer. L'installazione di *SmartKey USB* richiede solo che sia inserita dopo l'installazione dei driver. La verifica che i driver siano stati installati correttamente si può fare con il programma *SmartKey Program Central.* Se la SmartKey è inserita e se i driver sono stati installati correttamente, questa deve apparire nell'elenco delle SmartKey presenti sul computer.

# 4 I modelli di SmartKey

Le esigenze di protezione del software comprendono sia applicazioni semplici per pacchetti a basso costo sia applicazioni sofisticate e costose dove è richiesta massima sicurezza e flessibilità. Per ogni situazione esiste un modello di SmartKey con un adeguato rapporto prezzo/prestazioni.

Tutti i modelli sono stati implementati per garantire la compatibilità dall'alto al basso: se un programma funziona con una SmartKey, sicuramente funzionerà anche con una più complessa. Ad esempio, un programma scritto per *SmartKey FX* funzionerà con tutte le SmartKey.

In questo capitolo sono spiegate dettagliatamente le caratteristiche dei modelli di SmartKey e i criteri di scelta per usare la chiave che meglio si adatta alle proprie esigenze.

### *4.1 FX*

*SmartKey FX* è il modello più semplice e a basso costo; usa un meccanismo di protezione basato sull'assegnazione di un codice interno d'identificazione univoco e personale: il Codice d'Identificazione o *Id-Code*.

La protezione è di tipo algoritmico (quindi non a risposta fissa) ed utilizza operazioni di codifica criptata, che fanno riferimento al codice interno. L'*Id-Code* è utilizzato come parametro principale per la codifica: è inviato un insieme di dati che è restituito opportunamente crittografato in modo diverso in funzione dell'*Id-Code*.

La verifica della presenza della chiave può pertanto essere effettuata confrontando il dato restituito con quello atteso: se i due dati coincidono l'esecuzione del programma continua regolarmente, mentre in caso contrario l'esecuzione può essere arrestata.

- Protezione algoritmica basata su un codice univoco e personale (*Id-Code*)
- Protezione algoritmica con 20 chiavi personalizzabili dall'utente usando l'algoritmo di crittografia AES per il modello *SmartKey USB 3*

### *4.2 PR*

*SmartKey PR* è il modello per le applicazioni più versatili ove è necessario procedere alla personalizzazione d'ogni singolo pacchetto, come ad esempio nelle operazioni di serializzazione dei programmi.

*SmartKey PR*, infatti, oltre a mantenere lo stesso meccanismo di protezione algoritmica della chiave SmartKey FX, è di tipo programmabile: all'interno della chiave è disponibile un registro di memoria di 64/128 byte in lettura e scrittura, a cui è possibile accedere soltanto inviando due codici d'accesso denominati *Label* e *Password* (ognuno a 16 byte). I dati memorizzati nella chiave sono definiti *Secure Data*, perché solo il conoscitore della password può leggerli o scriverli.

I codici *Label* e *Password* stessi, con lunghezza 16 byte, sono programmabili da software, senza l'ausilio di dispositivi di programmazione esterni. È pertanto possibile variare la programmazione della chiave per ogni diverso programma da proteggere. I dati memorizzati possono essere variati dinamicamente dal programma protetto, permettendo applicazioni estremamente sofisticate quali ad esempio l'utilizzo dei dati stessi come contatore d'accessi.

*SmartKey PR* utilizza un meccanismo di protezione a doppia serratura: la sicurezza è garantita sia dall'univocità dell'*Id-Code* fissato in fabbrica e sia dalla password e dai dati programmati, noti alla sola software-house.

- Protezione algoritmica basata su un codice univoco e personale (*Id-Code*)
- Protezione algoritmica con 20 chiavi personalizzabili dall'utente usando l'algoritmo di crittografia AES per il modello *SmartKey USB 3*
- Codici di sicurezza addizionali programmabili a 16 byte (*Label* e *Password*)
- 64 byte di memoria interna programmabile (*Secure Data*). 128 byte per il modello *SmartKey 3 USB DL* (*Driver Less*)
- Limitazione opzionale del numero d'esecuzioni del programma da proteggere

### *4.3 EP*

*SmartKey EP* espande le caratteristiche di sicurezza dei modelli precedenti ed è il modello adatto alle applicazioni d'alta sicurezza, quali il controllo d'accessi a banche dati, a programmi riservati, etc.

Oltre a mantenere gli stessi meccanismi di protezione della chiave *SmartKey PR*, *SmartKey EP* permette la rilevazione dei tentativi d'accesso con password errata. Un apposito contatore interno (*Fail Counter*) restituisce il numero dei tentativi d'effrazione per permettere al legale utilizzatore di intraprendere via software qualunque azione volta alla difesa dei dati o dei programmi.

La seconda caratteristica peculiare di *SmartKey EP* è l'opzione di congelamento di password e dati una volta programmati: è cioè possibile fissare in modo irreversibile i codici ed i dati programmati nella chiave stessa dalla

software-house. È impedita un'eventuale successiva manipolazione della chiave a scopo fraudolento, ad esempio per modificare i propri diritti d'accesso a banche dati o i limiti di funzionamento del software in dotazione.

- Protezione algoritmica basata su un codice univoco e personale (*Id-Code*)
- Protezione algoritmica con 20 chiavi personalizzabili dall'utente usando l'algoritmo di crittografia AES per il modello *SmartKey USB 3*
- Codici di sicurezza addizionali programmabili a 16 byte (*Label* e *Password*)
- 64 byte di memoria interna programmabile (*Secure Data*). 128 byte per il modello *SmartKey 3 USB DL* (*Driver Less*)
- Contatore dei tentativi d'accesso fraudolento (*Fail Counter*)
- Congelamento dei dati programmati
- Limitazione opzionale del numero d'esecuzioni del programma da proteggere

### *4.4 SP e XM*

*SmartKey SP* e *XM* sono i modelli più sofisticato della famiglia SmartKey. Sono adatte alle applicazioni di top-security, per programmi costosi e per ambienti a rischio ove le probabilità di diffusione di copie illegali è molto alta.

*SmartKey SP* è un'evoluzione di *SmartKey EP* e ne estende la memoria interna (Secure Data) da 64 byte a 416/896 byte. *SmartKey XM* aggiunge una elevata capienza di memoria di 8192 byte.

- Protezione algoritmica basata su un codice univoco e personale (*Id-Code*)
- Protezione algoritmica con 20 chiavi personalizzabili dall'utente usando l'algoritmo di crittografia AES per il modello *SmartKey USB 3*
- Codici di sicurezza addizionali programmabili a 16 byte (*Label* e *Password*)
- 416 byte di memoria interna programmabile. 896 per *SmartKey USB 3*, 8192 per *SmartKey XM* (*Secure Data*)
- Contatore dei tentativi di accesso fraudolento (*Fail Counter*)
- Congelamento dei dati programmati
- Limitazione opzionale del numero di esecuzioni del programma da proteggere

### *4.5 NET*

*SmartKey NET* è il modello per le applicazioni in rete locale. *SmartKey NET* ha lo scopo di

- Proteggere il software scritto per reti locali dalla copia abusiva
- Permettere il controllo del numero di utenti che contemporaneamente possono utilizzare l'applicativo protetto

La protezione richiede un'unica chiave *SmartKey NET* installata su un qualunque calcolatore della rete.

*SmartKey NET* è in grado di gestire diversi utenti in rete, il cui numero massimo è programmabile nella chiave stessa. È possibile definire il numero d'utenze abilitate all'utilizzo dello stesso pacchetto software.

- Protezione algoritmica basata su un codice univoco e personale (*Id-Code*)
- Protezione algoritmica con 20 chiavi personalizzabili dall'utente usando l'algoritmo di crittografia AES per il modello *SmartKey USB 3*
- Codici di sicurezza addizionali programmabili a 16 byte (*Label* e *Password*)
- 416 byte di memoria interna programmabile (*Secure Data*)
- Contatore dei tentativi d'accesso fraudolento (*Fail Counter*)
- Congelamento dei dati programmati
- Protezione d'applicazioni in rete locale tramite una sola chiave di protezione
- Programmabilità del numero di utenti contemporanei abilitati ad utilizzare l'applicazione protetta e del numero d'esecuzioni.

### *4.6 Comparazione dei modelli SmartKey*

Il paragone tra SmartKey ed una cassaforte aiuta a meglio comprendere la differenza tra le varie chiavi.

Il modello *FX* è analogo ad una cassaforte che richiede una particolare chiave fisica per essere aperta: casseforti di differenti proprietari hanno chiavi diverse, così chiavi *FX* di diversi utenti hanno *Id-Code* differenti.

Il modello *PR*, invece, corrisponde ad una cassaforte più sofisticata: per aprirla è sì necessario possedere una chiave fisica, ma è anche necessario impostare una particolare combinazione su un'apposita manopola (due *giri a sinistra, tre a destra, etc.*). Analogamente, la chiave *PR* è caratterizzata da un *Id-Code* (*la chiave fisica*) e da una Password (*la*

*combinazione*). Sia la cassaforte sia il modulo *PR* hanno una combinazione programmabile; solo conoscendo la Password è possibile verificarne i Secure Data (*cioè accedere al contenuto della cassaforte*).

Il modello *EP* è una variante della chiave *PR* e nella nostra analogia corrisponde ad una cassaforte con serratura (*Id-Code*) e combinazione (*Password*); in più un dispositivo interno conta il numero di combinazioni errate impostate (*Fail Counter*); inoltre un meccanismo attivabile su richiesta impedisce che qualcuno possa fraudolentemente variare la combinazione impostata.

Il modello *SP* è dotato di 416 byte (896 per il modello USB 3) anziché 64 byte (*è una cassaforte più capiente*). Il modello *XM* dispone di 8192 byte di memoria.

Infine la chiave *NET* possiede gli stessi meccanismi di sicurezza della chiave *SP*, però è adatta alle applicazioni in rete di personal computer.

Ecco una tabella riassuntiva delle caratteristiche dei modelli di SmartKey:

| SmartKey | Rete | IdCode/ AES | Password | Memoria | Fail Counter |
|---|---|---|---|---|---|
| **FX** | | ✓ | | | |
| **PR** | | ✓ | ✓ | 64/128 byte | |
| **EP** | | ✓ | ✓ | 64/128 byte | ✓ |
| **SP** | | ✓ | ✓ | 416/896 byte | ✓ |
| **XM** | | ✓ | ✓ | 8192 byte | ✓ |
| **NET** | ✓ | ✓ | ✓ | 416 byte | ✓ |

**Tabella 2** Tabella riassuntiva dei modelli di SmartKey.

### *4.7 Quale SmartKey adottare?*

Non è facile dare una risposta a questa domanda, perché vi sono molteplici ragioni, sia tecniche sia economiche, che concorrono ad orientare la scelta verso un modello piuttosto che verso un altro. Soltanto un esame caso per caso permette di definire il problema, tenendo conto di variabili quali:

- Il costo del pacchetto
- L'ambiente in cui il software protetto si colloca
- L'area geografica di diffusione del prodotto
- Il tempo a disposizione per implementare la protezione

A questo proposito è bene ricordare che proteggere il software comporta problematiche e scelte simili a quelle dei contratti d'assicurazione contro il furto ("Per quale importo devo assicurare?", "Quali garanzie accessorie devo includere nel pacchetto assicurativo?", "Qual è la probabilità di furto?").

Ecco alcune considerazioni generali che possono fornire indicazioni per la scelta di SmartKey.

La chiave *SmartKey FX* è semplice, veloce da implementare ed economica. È adatta alla protezione di programmi a basso costo o di tipo pacchettizzato, ove non è necessario distinguere tra loro i vari applicativi o le varie versioni del software. Tenete presente che in questo caso il meccanismo di protezione di tipo algoritmico si basa su un codice univoco e personalizzato assegnato in fabbrica (*Id-Code*) e non più modificabile. Pertanto tutte le chiavi *FX* di un singolo utilizzatore posseggono il medesimo codice.

La chiave *SmartKey PR* è invece il modello con il miglior rapporto prezzo/prestazioni, perché, oltre all'*Id-Code* univoco, possiede una memoria interna a cui è possibile accedere solo tramite codici riservati e programmabili. Pertanto ogni chiave è singolarmente programmabile dalla software-house in funzione delle specifiche esigenze. È quindi possibile predisporre ogni chiave per la protezione di un particolare programma o di un insieme definito di moduli dello stesso programma. È possibile scrivere nella chiave un numero di serie, il nome o il codice del cliente, una data o qualunque altra informazione che possa essere utile al meccanismo di protezione implementato. È adatta per la protezione della maggior parte del software in una fascia media di costo, perché offre un'elevata sicurezza ad un costo contenuto.

La chiave *SmartKey EP* è consigliabile nella protezione di programmi costosi, ove si vogliano scoraggiare non solo i tentativi di copiatura del software, ma anche i tentativi d'effrazione del meccanismo di protezione stesso, cioè l'effrazione o l'alterazione del contenuto della chiave.

La chiave *SmartKey SP* e *XM* sono necessarie se il proprio schema di protezione richiede un'elevata capacità di memorizzazione di dati all'interno della chiave.

La chiave *SmartKey NET* è la scelta obbligata per applicazioni in rete locale, ove non si desideri installare una chiave per ogni singolo utente.

# 5 Proteggere un’applicazione con SmartKey

Proteggere un'applicazione con SmartKey significa implementare l'*execution control*, cioè modificare l'applicazione in modo che la sua esecuzione sia vincolata alla presenza di una chiave di protezione del software.

SmartKey possiede due metodi per l'implementazione della protezione:

- **Protezione manuale** tramite intervento sui sorgenti del programma originale e l'utilizzo di software driver
- **Protezione automatica** tramite intervento diretto sul file eseguibile del programma originale

## *5.1 Protezione manuale*

Con *protezione manuale* s’intende l'intervento da parte del programmatore sul sorgente dell'applicativo da proteggere per inserire le funzioni fornite che permettono l’interfaccia tra il programma e la SmartKey attraverso i suoi driver (nel seguito si utilizzerà il termine “Application Programming Interface” o API per far riferimento alla collezione di tutte queste funzioni).

Si tratta della modalità canonica di protezione del software, che consente di definire autonomamente una propria personale strategia di protezione (quante chiamate effettuare per verificare la presenza della chiave, in quali punti del programma e in quali istanti effettuarle, quali azioni intraprendere in caso d’assenza della chiave, etc.).

Benché richieda uno sforzo da parte del programmatore, se siete in possesso dei sorgenti dei programmi da proteggere, la protezione manuale è il meccanismo che consente la massima flessibilità e sicurezza.

La spiegazione delle funzionalità atomiche delle API è contenuta nel capitolo 9. Un utilizzo naif delle API, tuttavia, non è sufficiente a garantire che programmi protetti con SmartKey realizzino un adeguato livello di sicurezza. Occorre avvalersi anche di tecniche di protezione,descritte nel capitolo 10, che suggeriscono strategie di protezione robuste. La lettura del capitolo 10 è fortemente consigliata: anche una sola apparentemente innocua strutturazione del codice può vanificare tutto il lavoro di protezione se essa espone elementi critici per la sicurezza . Un semplice esempio: bisogna evitare che la password sia memorizzata su hard disk in chiaro o che sia trasmessa in chiaro tra server e client.

Le API sono disponibili per Linux, Mac OS X e Windows e la loro sintassi è identica per tutti e tre i sistemi operativi. Questo rende veloce e semplice il porting delle porzioni di codice relative alla protezione del proprio programma da un sistema operativo all’altro.

## *5.2 Protezione automatica*

Con *protezione automatica* s’intende la possibilità di automatizzare completamente la procedura di protezione di un file eseguibile senza dover intervenire manualmente nella struttura del programma originario, sollevando il programmatore da un lavoro talvolta impegnativo.

È utilizzata a questo scopo la tecnologia proprietaria *Global Security System (GSS)* che implementa la protezione automatica trasformando un programma in modo che questo non possa funzionare se non alla presenza dell'opportuna chiave di protezione.

Tramite il software *GSS* in dotazione non è necessario preoccuparsi di modificare il programma da proteggere, né possederne il sorgente: partendo dal file originario in formato eseguibile, è generato un secondo file eseguibile che espleta le stesse funzioni del file originale, purché sia inserita nel sistema la chiave SmartKey corretta.

Il funzionamento di *GSS* è estremamente sofisticato, in quanto non si limita ad aggiungere al programma da proteggere la chiamata a SmartKey; esso inoltre opera una vera e propria crittografia del programma originario, decodificabile se il programma è eseguito alla presenza della chiave SmartKey corretta.

Quando è mandata in esecuzione l'applicazione trattata con *GSS*, essa istantaneamente si autodecodifica. Senza SmartKey, l'applicazione non potrà essere decodificata. L'operazione di codifica non rallenta l'esecuzione del programma protetto.

L'analisi di un file codificato da *GSS* è virtualmente impossibile, perché il *reverse engineering* del software risulta privo di significato fin tanto che non è decodificato run-time. Anche tutti i messaggi in formato testuale nel file eseguibile originale (contenenti ad esempio il nome della software-house, quello del cliente, il numero di serie, i valori di alcune costanti) sono tramutati in una sequenza di caratteri indecifrabile, impedendone quindi l'alterazione con le utility che agiscono direttamente sui settori dell'hard-disk.

La protezione automatica effettuata dal *GSS* si avvale inoltre di una serie di meccanismi opzionali che consentono di risolvere le esigenze specifiche di ogni situazione applicativa, quali ad esempio il controllo periodico della presenza della chiave.

La descrizione dettagliata di come proteggere automaticamente un programma tramite l’uso di GSS è contenuta nel capitolo 8.

Esiste solo la versione di GSS per Windows, quindi è possibile fare la protezione automatica solo dei programmi per Windows.

### *5.3 Utilizzare la protezione manuale o la protezione automatica?*

Prima di iniziare l'operazione di protezione del software, è necessario definire la tecnica da utilizzare. Cosa differenzia la protezione automatica da quella personalizzata? La risposta è sintetizzata nella tabella che segue:

| **Tipo di protezione** | **Tempo necessario** | **Necessarie conoscenze informatiche?** | **Necessari i codici sorgenti del programma?** | **Sicurezza** |
|---|---|---|---|---|
| Manuale | Ore | Sì | Sì | Molto elevato |
| Automatica | Minuti | No | No | Elevato |

**Tabella 3** Tipi di protezione da usare.

La **protezione manuale** è preferibile quando si hanno i sorgenti, perché grazie alla flessibilità essa consente di introdurre un livello di sicurezza molto elevato. È necessario un piccolo sforzo iniziale d'implementazione che tuttavia consente di implementare strategie personalizzate di protezione. La **protezione automatica** è una soluzione sicura e veloce. Quando si deve proteggere un applicativo Windows questa tecnica permette di risolvere anche situazione quali:

- Indisponibilità dei sorgenti; è il caso tipico dei distributori di software non protetto da chiavi nel Paese d'origine.
- Tempo limitato per l'implementazione della protezione.
- Programmi scritti in linguaggi di programmazione non comuni, e quindi senza i relativi software driver per la protezione manuale

# 6 Protezione in rete locale

Quando più calcolatori sono collegati in rete locale, è possibile proteggere il software di rete in uno dei due modi seguenti:

- Inserire una chiave di protezione di tipo standalone (*FX*, *PR*, *EP*, *SP*, *XM*) su ognuno dei calcolatori abilitati all'esecuzione dell'applicativo. In tal caso non è necessario effettuare alcuna modifica al software già protetto per la modalità standalone.
- Inserire una sola chiave di protezione di tipo *NET* con i relativi software server SmartKey.

*SmartKey NET* è un'estensione del modello *SP*, di cui possiede tutte le caratteristiche principali, oltre ad alcune caratteristiche addizionali, che consentono di effettuare la protezione utilizzando un'**unica chiave** installata su un qualunque calcolatore della rete o su un server non dedicato.

Il software fornito con la chiave consente ad ogni calcolatore della rete di interrogare l'unica *SmartKey NET*. Non si ha pertanto la necessità di utilizzare tante chiavi quante sono le postazioni di lavoro della rete.

Tramite la tecnologia proprietaria **Map** – Multi Application Protection, *SmartKey NET* consente anche di:

- Proteggere **più applicativi diversi** operanti in rete (fino a 116),
- Limitare per ogni applicativo protetto **il numero massimo di utenti** che contemporaneamente possono utilizzare l'applicativo, in modo da porre sotto controllo le licenze d'utilizzo. Ad esempio PROG1 può essere abilitato per 12 licenze, PROG2 per 27, PROG3 per un numero di licenze illimitato, etc.
- Limitare il **numero di esecuzioni** di ciascuno dei programmi protetti. Questa caratteristica può rivelarsi utile qualora si debbano creare versioni demo del software o si desideri adottare la politica del software a noleggio, consentendo all'utente un numero d'esecuzioni prestabilito.

### *6.1 La protezione automatica in rete locale*

Oltre agli applicativi funzionanti in modalità standalone, *GSS* è in grado di proteggere anche gli applicativi di rete.

L'utilizzo del programma protetto è assolutamente trasparente all'utente finale, che può utilizzare sia una chiave locale sia una di rete: *GSS* inizialmente ricercherà la chiave sulle porte locali e, qualora la ricerca dovesse fallire, continuerà tentando la comunicazione con una chiave di rete.

### *6.2 La protezione manuale in rete locale*

La gran diffusione delle reti locali richiede un approccio semplice ed intuitivo per l'interfacciamento verso la chiave. SmartKey adotta la tecnologia ***MultiLan*** che consente a chi sviluppa software di proteggere gli applicativi con un unico driver indipendentemente dall'ambiente di funzionamento, sia standalone sia in rete.

- MultiLan è driver unico, sia per applicazioni standalone sia in rete
- MultiLan individua automaticamente il tipo di rete

All'utente finale è solo richiesto di installare *SmartKey NET* su un qualunque PC nella rete. Se l'applicativo è eseguito localmente il driver cerca automaticamente la chiave (*SmartKey FX*, *PR*, *EP*, *SP*, *XM*) sulla porta parallela del PC locale.

### *6.3 Protezione di più applicativi con SmartKey*

In ambiente Lan è possibile utilizzare un'unica *SmartKey NET* per la protezione di più applicativi software. La tecnologia utilizzata è denominata **Map** – **Multi Application Protection** e consente di:

- Proteggere più di un applicativo in ambiente standalone o rete. Nel caso di rete locale è anche possibile definire per ciascun applicativo protetto un differente numero di licenze abilitate.
- Limitare il numero d'esecuzioni di ciascuno dei programmi protetti. Questa caratteristica può rivelarsi utile qualora si debbano creare versioni *demo* del software o si desideri adottare la politica del software a *noleggio*. Alla scadenza del numero d'esecuzioni preimpostato in un contatore (decrementato ad ogni avvio del programma), non è più permessa la partenza del programma.

# 7 La struttura interna di SmartKey

La struttura delle chiavi SmartKey prevede l'utilizzo di alcuni registri interni, ognuno con una particolare funzione di protezione:

- Id-Code register
- Label register (16 byte)
- Password register (16 byte)
- Secure Data register (64 / 128 / 416 / 896 / 8192 byte)
- Fail Counter register (2 byte)

### *7.1 Id-Code register: il codice personale*

L'*Id-Code* è un registro programmato in fabbrica durante il test di ogni singola chiave e non più modificabile. Ogni utilizzatore di SmartKey ha un codice di identificazione diverso ed il numero totale di possibili codici è $2^{32}$ (pari a circa 4.000.000.000).

L'*Id-Code* è presente in tutti i modelli SmartKey e garantisce che chiavi di diversi utilizzatori siano sicuramente diverse tra loro. Di fatto, assegna un codice personale ed unico ad ogni possessore di chiavi di protezione.

Per ragioni di sicurezza, il codice di identificazione riportato nell'*Id-Code* register non è leggibile direttamente, ma il suo valore condiziona il risultato dell'interrogazione algoritmica della chiave. Utenti diversi hanno differenti *Id-Code* e quindi le relative chiavi forniscono risposte differenti all'interrogazione algoritmica da parte del software protetto.

### *7.2 Label register: l'etichetta di identificazione e accesso*

Il registro *Label* contiene uno dei due codici di accesso alla chiave ed ha la funzione di identificazione della chiave corretta per la particolare applicazione in esecuzione. Di fatto, la Label costituisce un'etichetta elettronica di identificazione interna alla chiave.

La funzione della *Label* è particolarmente importante quando più chiavi *SmartKey Parallele* sono impilate in cascata sulla medesima porta parallela. Infatti, la *Label* è in questo caso una sorta di indirizzo, che permette al software protetto di interrogare la chiave giusta. Per verificare la presenza della chiave ricercata, il software protetto invia sulla porta parallela il valore della *Label*: solo la chiave con la *Label* coincidente restituirà una risposta.

È importante pertanto assegnare una Label differente per ognuno dei propri programmi applicativi, in modo che più chiavi SmartKey possano essere contemporaneamente installate.

Il registro *Label* ha una dimensione di 16 byte ($2^{128}$ combinazioni, pari a $3 \bullet 10^{38}$); siccome il numero di combinazioni è enorme è in sostanza impossibile che due diverse software-house decidano di assegnare la medesima *Label* ai propri applicativi.

Con le chiavi di tipo programmabile (*PR*, *EP*, *SP*, *XM* e *NET*), la *Label* può essere programmata off-line tramite l'utility SPC, selezionando la modalità di programmazione, oppure on-line tramite i software driver in dotazione.

Nel caso delle chiavi *FX* il registro *Label* è fisso e coincide con l'*Id-Code*.

| | | |
|---|---|---|
| **FX** | *Label* non programmabile | Id-Code |
| **PR EP SP XM NET** | *Label* programmabile | 16 byte |

**Tabella 4** Tabelle delle SmartKey con *Label* non programmabile e *Label* programmabile.

### *7.3 Password register: la chiave d'accesso ai dati*

Il registro *Password* ha estrema importanza nel meccanismo di protezione in quanto solo conoscendo com'è stato programmato è possibile accedere ai dati contenuti nella memoria non volatile della chiave. Analogamente alla combinazione di una cassaforte, la conoscenza della *Password* corretta consente l'apertura della chiave e quindi l'accesso al contenuto.

Il registro *Password*, che ha una dimensione di 16 byte, può essere programmato tramite l'utility SPC selezionando la modalità di programmazione.

Non è mai possibile rileggere direttamente il contenuto della *Password* impostata nella chiave: l'accesso alla *Password* avviene solo in scrittura durante la fase di programmazione.

È possibile anche riprogrammare la *Password* pur non conoscendo quella precedente; in tal caso però il contenuto della memoria dati (Secure Data) è automaticamente azzerato.

### *7.4 Secure Data register: i dati della memoria non volatile*

Il registro *Secure Data* è dotato di una memoria non volatile interna alla chiave, a cui è possibile accedere solo con la preventiva conoscenza della Password. Controllando il contenuto del registro è possibile evidenziare se si è alla presenza di un tentativo d'effrazione oppure di un'installazione legittima del software.

| **Modello** | **Memoria** |
|---|---|
| FX Parallela/USB | 0 |
| PR, EP Parallela/USB | 64 |
| PR, EP USB Driver Less | 128 |
| SP Parallela | 416 |
| SP USB | 896 |
| XM USB | 8192 |
| NET Parallela/USB | 416 |

**Tabella 5** Dimensione della memoria di SmartKey.

L'intervento dei *Secure Data* nel meccanismo di protezione avviene in modi diversi, ma principalmente tramite un'operazione di comparazione tra il contenuto atteso e quello effettivamente letto dalla chiave: l'esito del confronto permette di decidere se continuare o no l'esecuzione del programma.

Nel caso delle chiavi *SP, XM* e *NET* il registro è anche usato per la memorizzazione dell'algoritmo di sicurezza programmabile da utente. Infine, con la chiave di rete *NET,* alcuni byte sono utilizzati per definire il numero massimo d'utenti contemporanei dell'applicazione protetta e l'eventuale limitazione al numero totale d'esecuzioni. Il servizio di gestione delle licenze richiede 2 byte per l'abilitazione del servizio stesso e 3 byte per ogni applicazione protetta.

I *Secure Data* possono essere letti o scritti da software purché preventivamente sia stata trasferita la corretta *Password*.

In caso di *Password* errata le operazioni di lettura non restituiscono il contenuto dei *Secure Data*, bensì una serie pseudocasuale di bit. Le operazioni di scrittura con *Password* errata non hanno parimenti alcun effetto, al fine di non alterare il valido contenuto dei dati programmati.

### *7.5 Fail Counter register: l'allarme sugli accessi errati*

Il registro *Fail Counter*, disponibile solo nelle chiavi *EP, SP, XM* e *NET*, permette il conteggio automatico del numero di tentativi di accessi errati alla chiave.

Ogni volta che è tentato un accesso in lettura o in scrittura con *Password* errata, il contenuto del registro è automaticamente incrementato di uno.

Si tratta quindi di un contatore autoincrementale a sola lettura, che permette di evidenziare tentativi di effrazione con ricerca della *Password*. Per ragioni di sicurezza il contatore non è azzerabile da nessuna delle funzioni di scrittura o programmazione della chiave.

Il conteggio è compreso tra 0 e 10000. Il registro è letto durante una fase di lettura (READING MODE), previa conoscenza della Password. Se la *Password* è errata, il numero restituito è generato in modo casuale.

Il software da proteggere può utilizzare il registro per la verifica di eventuali tentativi di accesso indebito; ove ad esempio la chiave sia usata per permettere l'accesso a banche dati o a dati riservati, il software può disabilitarne permanentemente il funzionamento alterando il contenuto dei Secure Data dopo un numero programmato di accessi errati anche non consecutivi.

# 8 Protezione automatica

Il programma *Global Security System (GSS)* permette di proteggere il programma in modo automatico senza dover scrivere alcuna riga di codice e senza disporre dei file sorgenti del programma. *GSS* partendo dal file eseguibile del programma da proteggere genera un programma protetto. Il programma protetto così ottenuto ha le stesse funzionalità dell'originale, ma può funzionare solamente con la SmartKey per cui è stato generato e con quelle che hanno la sua stessa configurazione.

*GSS* offre un ulteriore grado di sicurezza: la crittografia dell'eseguibile del nuovo programma. Grazie a sofisticati algoritmi di crittografia, è estremamente difficile ricavare il programma originale partendo da quello protetto. *GSS* offre, quindi, due meccanismi indipendenti di protezione:

- blocco del programma se non è presente SmartKey
- crittografia dei dati contenuti nel nuovo file eseguibile

*GSS* permette, inoltre, di crittografare tutti i file gestiti dal software applicativo. Questo garantisce un ulteriore livello di sicurezza.

Grazie a *GSS*, SmartKey è in grado di offrire altri vantaggi. Può essere utilizzata per limitare in modo efficace il numero di esecuzioni quando i programmi sono distribuiti a scopo dimostrativo. Una volta superato il limite impostato, il software si disattiva. Inoltre, proteggere diverse applicazioni in ambiente Lan è diventato semplice grazie a SmartKey. È possibile anche impostare il numero massimo di licenze per l'applicazione protetta.

Tutti i modelli di SmartKey sono compatibili con *GSS*. Nel caso di *SmartKey FX*, la protezione si basa solo sul *Codice Identificativo*. Nel caso degli altri modelli la protezione si basa anche su altri elementi come *Password* e *Secure Data*.

La figura 1 mostra il pannello di *GSS* che serve a inserire tutti i dati necessari per creare il programma protetto. Tutti i campi da inserire sono illustrati nei paragrafi di questo capitolo.


**Figura 1** Interfaccia di GSS.

### *8.1 Protezione automatica con GSS*

La protezione offerta da SmartKey si avvale della tecnologia *GSS*, che è fornita in un pacchetto d'utilità GSS.EXE fornito insieme al Kit SmartKey. Alcuni dei vantaggi che questa utility offre sono riassunti nel seguente elenco:

- Protezione automatica dei file eseguibili
- Crittografia opzionale dei file dati associati ai programmi protetti
- Protezione basata su *Label*, *Password* e *Memoria*
- Controllo periodico della presenza della SmartKey
- Scelta dei messaggi da visualizzare

## *8.2 Protezione su piattaforme Windows con GSS*

*Global Security System* permette di proteggere tutti i file eseguibili realizzati per le piattaforme Windows 9x, Windows Me, Windows NT, Windows 2000, Windows XP, Windows 2003, Windows Vista e superiori.

Per avviare programmi Windows protetti è necessario che siano presenti nella stessa directory dell'eseguibile i moduli di run-time *modw9x.exe*, *gssvx12.vxd* e *modwnt.exe*. Questi files sono creati automaticamente da GSS nella cartella di destinazione durante la fase di protezione dell'applicativo. Con Windows NT, Windows 2000, Windows XP, Windows 2003, e Windows Vista è inoltre necessario installare il driver del dispositivo *GSS* specifico per questi sistemi operativi. Per installare correttamente il driver del dispositivo, l'utente deve possedere i diritti d'amministrazione.

L'installazione dei driver di SmartKey e del dispositivo *GSS* può essere eseguita manualmente con l'utility SDI oppure in automatico, integrando la libreria SDI alla vostra procedura d'installazione.

## *8.3 GSS: le opzioni comuni*

Le varie versioni di *GSS* possiedono un'interfaccia utente unica. Questo permette di proteggere l'integrità del codice oggetto delle applicazioni protette. È particolarmente utile quando un hacker, nel tentativo di alterare una parte del codice o dei parametri numerici o delle stringhe di testo modifica in modo fraudolento il codice oggetto.

Persino il più piccolo dei cambiamenti, anche se si tratta di un solo bit, è rilevato da *GSS*, che visualizza un messaggio d'avvertimento all'avvio e arresta immediatamente il programma. *GSS* fornisce anche la possibilità di inserire un testo che sarà visualizzato nel caso la chiave sia assente, o il file di programma sia corrotto.

### 8.3.1 Controllo della presenza della chiave

Gli utenti di SmartKey possono inviare spesso dei comandi per mantenere un tabulato della presenza della chiave grazie all'opzione "Abilita il controllo periodico della presenza della chiave SmartKey". Questa garantisce agli utenti che la chiave sia presente durante l'intera esecuzione del programma protetto.

L'opzione di controllo periodico impedisce anche agli utenti di rimuovere la chiave dopo l'avvio dell'applicazione. In sua assenza, l'applicazione protetta continuerebbe a lavorare, poiché non sarebbe inoltrata nessuna richiesta di controllare la presenza della chiave.

### 8.3.2 Programmazione dei Messaggi di Errore

SmartKey consente agli utenti di personalizzare i messaggi di errore che sono visualizzati in particolari condizioni. Utilizzando questa funzione è possibile definire i messaggi d'errore di proprio gradimento. Le condizioni per le quali è possibile programmare i messaggi d'errore sono le seguenti:

- La chiave SmartKey non è presente
- Il codice di programma è stato alterato

L'elenco dei messaggi che si possono programmare dipende sia dal modello di SmartKey sia dal carattere di Configurazione della Protezione. Anche l'utility *GSS* suggerisce alcuni messaggi di default per ciascuna situazione d'errore elencata sopra.

### 8.3.3 Crittografia del codice eseguibile

Di default, *GSS* crittografa completamente il file originale, in modo tale che il nuovo file eseguibile sia completamente crittografato. Questo difende il programma dagli attacchi degli hacker, perché è virtualmente impossibile disassemblare l'applicazione originale usando il file eseguibile generato da *GSS*.

Qualsiasi strategia di protezione priva delle tecniche di crittografia offre un livello di protezione piuttosto ridotto. La crittografia è reversibile solo se la SmartKey usata per la crittografia è collegata al computer. Se manca la SmartKey, la crittografia è irreversibile.

Nel caso specifico dei programmi basati su Windows, la crittografia permette di evitare l'esportazione e la copia illecita delle proprie risorse. Questo si rende necessario poiché esiste un'ampia disponibilità di programmi basati su Windows che consentono di estrarre le risorse (icone, cursori, dialoghi, menu, bitmap, barre degli strumenti, ecc.) e anche di copiarle e riutilizzarle.

### 8.3.4 Protezione basata su parametri

Gli utenti di SmartKey possono utilizzare una protezione basata su molteplici parametri oltre a quella offerta dalla crittografia. Questo garantisce che il programma protetto lavori solo quando la chiave utilizzata è la stessa impiegata originariamente per la creazione del file. Ciò perché ciascuna chiave possiede un insieme di parametri unico, che comprende *Id-Code*, *Label*, *Password* e *Memoria*.

Tuttavia, la protezione basata su parametri è un optional che può essere facilmente disattivato, eccetto per la *Label*. L'uso della *Label* è obbligatorio. Quando lavora un programma protetto e i parametri optional della chiave sono disattivati, è controllata solo la *Label*, tralasciando gli altri parametri. Con una buona combinazione solo di Label, il programma è eseguito con successo.

### 8.3.5 Messaggio visualizzato in assenza della chiave

Se il programma protetto è attivato senza la presenza della relativa SmartKey, il programma si arresta ed è visualizzato il seguente messaggio di default: **Chiave Mancante.**

Questo messaggio non è fisso. Così come per i messaggi associati ad altre funzioni, l'utility GSS.EXE permette anche di sostituire il messaggio standard con uno personalizzato.

### 8.3.6 Limitazione del numero di esecuzioni e licenze

L'utilizzo di SmartKey non è limitato soltanto alle applicazioni protette da accessi non autorizzati. Essa consente anche di impostare il numero di esecuzioni e di licenze permesse per l'applicazione protetta. Questa funzione è particolarmente utile per il rilascio di versioni demo dell'applicazione. Questo fornisce agli utenti la possibilità di consentire un uso limitato dell'applicazione. Al raggiungimento del numero prestabilito di esecuzioni, l'applicazione cessa di funzionare.

Dall'altro lato, la possibilità di limitare il numero di licenze garantisce un utilizzo dell'applicazione conforme alla politica sulle licenze. Questa funzione è disponibile con la versione *SmartKey NET*.

### 8.3.7 Crittografia automatica dei file di dati

Per mettere un'applicazione completamente al sicuro da accessi non autorizzati, non solo è necessario proteggere l'applicazione eseguibile, ma anche tutti i relativi dati/database (.DBF, .DAT, ecc.) ad essa associati. Grazie alla tecnologia *GSS*, SmartKey ora vi permette di fare tutto questo.

Quando una stringa di dati è inviata alla chiave, l'algoritmo pre-programmato è utilizzato per crittografarla. L'utility *GSS* è utilizzata principalmente per questo scopo. Al momento dell'avvio, l'applicazione crittografata automaticamente provvede a decrittografare i file crittografati. Quando non è in uso, questi file sono nuovamente crittografati.

### 8.3.8 Protezione delle applicazioni sul network

Oltre alla protezione delle applicazioni che lavorano in modalità standalone, SmartKey mette in sicurezza anche applicazioni su Local Area Network. Poiché l'uso di un'applicazione protetta è assolutamente trasparente per l'utente finale, è possibile utilizzare sia una chiave locale sia una chiave network. La tecnologia *GSS* cerca dapprima il dispositivo sulle porte locali. Nel caso d'insuccesso, inizia a comunicare con un dispositivo Lan.

*SmartKey NET* è stata sviluppata in particolare con questo scopo. Il software fornito insieme alla chiave permette ai computer in rete di inviare la richiesta ad una singola *SmartKey NET*. Di conseguenza, non è più necessario usare tante chiavi quante le stazioni di lavoro collegate in rete.

### 8.3.9 Protezione dei file eseguibili in serie

Talvolta può essere necessario proteggere automaticamente diversi file eseguibili in una sola operazione. In queste situazioni *gssline.exe*, che è la versione di riga di comando di GSS, si rivela molto utile.

La sintassi per l'esecuzione di GSSLINE è:

```
gssline CFG_FILE EXE_FILE [DATA_FILE...] DESTINATION_DIR
```

| | |
|---|---|
| CFG_FILE | È il file di configurazione |
| EXE_FILE or DATA_FILE | È il nome dell' .EXE o file di dati da proteggere |
| DESTINATION_DIR | È il percorso dove si trova il file .EXE finale protetto. |

**Tabella 6**

Nel digitare la riga di comando, occorre tenere in considerazione quanto segue:

- Il percorso del file protetto (DESTINATION_DIR) che deve essere prodotto da GSSLINE deve essere diverso dal percorso in cui si trova il file originale (EXE_FILE|DATA_FILE).
- Indicare chiaramente l'estensione di ciascun file (.CFG e .EXE).

## *8.4 Implementazione rapida della protezione dell'applicazione*

Fino ad ora in questo capitolo ci siamo occupati in dettaglio della protezione automatica del software. A seguire potrete trovare un insieme di passi chiave necessari per implementare rapidamente la protezione della vostra applicazione.

- Inserire *Label* e *Password* della vostra SmartKey ed attivare la protezione basata sulla *Memoria*
- Nel caso di *SmartKey NET*, è possibile attivare la protezione Map per controllare le licenze e le esecuzioni della vostra applicazione
- Scegliere il file da proteggere nel pannello Applicazione
- Scegliere un file Icona, se esistente
- Scegliere gli eventuali file di dati da proteggere relativi alla vostra applicazione

- Selezionare la cartella di destinazione dove desiderate salvare l'applicazione protetta; è consigliabile cambiare cartella per evitare di sovra-scrivere il file originale non protetto
- Cambiare il Messaggio di Errore (se lo desiderate) in funzione della situazione d’errore

# 9 Protezione manuale

La protezione manuale si basa sull'utilizzo delle funzioni della libreria del kit di sviluppo fornito. Le funzioni per SmartKey o API, sono implementate sia come librerie statiche sia come librerie dinamiche (DLL, nel caso specifico di Windows). Da un punto di vista funzionale non c'è differenza tra i due tipi di libreria. Quelle dinamiche offrono un grado di sicurezza minore perché un hacker esperto potrebbe capire quando il programma protetto usa la libreria dinamica. Questo pericolo è ridotto con le librerie statiche perché il link avviene nel momento della generazione del file eseguibile.

L'uso delle API e l'implementazione di tecniche di protezione robuste permettono di proteggere il proprio lavoro anche dalle minacce di pirati informatici esperti e dotati di raffinati strumenti d'analisi. È molto importante conoscere queste tecniche, perché un hacker potrebbe eludere i sistemi di sicurezza grazie ad un banale punto debole del codice. Ad esempio, bisogna assolutamente evitare che da un'analisi del file eseguibile si possa ricavare *Label* e *Password*. Il capitolo 10 successivo illustra ed esemplifica alcune di queste tecniche di protezione e costituisce una integrazione essenziale al presente capitolo.

I software driver consentono di attivare una serie di comandi, ognuno dei quali implementa una tra le seguenti modalità operative di sicurezza:

- Locating mode: individua se e su quale porta parallela o USB è presente la chiave.
- Scrambling mode: verifica per via algoritmica se l'*Id-Code* è corretto.
- Reading mode: legge i Secure Data.
- Block Reading mode: legge i Secure Data a blocchi.
- Writing mode: scrive i Secure Data.
- Block Writing mode: scrive i Secure Data a blocchi.
- Fixing mode: fissa il contenuto della chiave in modo che non possa essere più modificato.
- Programming mode: riprogramma il contenuto della chiave.
- AES mode: autentica per via algoritmica la SmartKey usando l'algoritmo AES.

Il kit di sviluppo di SmartKey contiene il programma *smartdem* che usa alcuni dei comandi spiegati nei paragrafi successivi. Il programma è di tipo consolle, è stato scritto in C e può essere compilato con un qualsiasi compilatore C in ambiente Linux, Mac OS X e Windows.

Il codice sorgente contenuto in *smartdem.c* è uguale per tutti i sistemi operativi, ma è differente la modalità di compilazione.

### *9.1 Modalità d'esecuzione dei comandi SmartKey*

La modalità d'esecuzione dei comandi SmartKey avviene attraverso lo scambio di un campo dati tra il programma ed il driver SmartKey. Il campo dati ha un formato fisso e contiene tutte le informazioni necessarie all'esecuzione del comando e l'eventuale risultato. È definito come una struttura (o record) con i seguenti campi:

```
struct smartkey {
      word lpt;
      word command;
      byte label[16];
      byte password[16];
      byte data[64];
      word fail_counter;
      word status;
      byte ext_data[352];
}
```

L'uso d'ogni campo può variare secondo il comando eseguito, ma generalmente è il seguente.

| | |
|---|---|
| `lpt` | Identificatore della porta parallela o USB dove si trova la SmartKey. |
| `command` | Codice del comando da eseguire. |
| `label` | *Label* della SmartKey. La label è necessaria per tutti i comandi. |
| `password` | *Password* della SmartKey. La password è necessaria per tutti i comandi che richiedono |

| | |
|---|---|
| | l'accesso alla memoria di SmartKey. |
| `data` | Contenuto della memoria della SmartKey e generico buffer per le operazioni che richiedono uno scambio di dati. |
| `fail_counter` | Contatore degli accessi falliti alla SmartKey. |
| `status` | Risultato dell'esecuzione del comando. Il valore 0 indica che il comando è stato eseguito correttamente. |
| `ext_data` | Contenuto della memoria estesa della SmartKey. |

È da considerare, che benché alcuni campi abbiano il medesimo nome dei registri fisici della chiave, questi sono entità diverse. Ad esempio, il contenuto della memoria della SmartKey è effettivamente presente nel campo *data* della struttura solo durante le operazioni *Reading* e *Writing*. Ad esempio, durante l'operazione di *Scrambling* il campo *data* contiene i dati per lo scrambling tra PC e SmartKey. La funzione *Scrambling* usa il campo *data* solo come variabile d'appoggio per lo svolgimento delle proprie operazioni, e non modifica il contenuto della memoria della SmartKey.

Per l'esecuzione di un comando occorre:

- Dichiarare una variabile di tipo struttura con i campi SmartKey.
- Riempire i campi della variabile con i valori richiesti dal comando.
  In particolare deve essere impostato il campo *command* con comando da eseguire e ogni altro campo necessario all'esecuzione del comando stesso.
- Chiamare la funzione definita nel driver SmartKey passando come argomento la variabile struttura. Il nome della funzione e la modalità di passaggio della struttura dipendono dall'ambiente di sviluppo utilizzato. Generalmente la funzione è chiamata *msclink()* e la variabile è passata per indirizzo e non per valore.
- Leggere dal campo *status* il risultato del comando e ogni altro valore di output

Nell' SDK SmartKey sono presenti esempi per i principali ambienti di sviluppo. Riferirsi al file *leggimi.txt* di ogni esempio per maggiori dettagli su come utilizzare il driver SmartKey in quello specifico ambiente.

Nel caso si voglia utilizzare un ambiente di sviluppo diverso da quelli esplicitamente supportati, è comunque possibile utilizzare direttamente le librerie disponibili se il linguaggio utilizzato è in grado di importare librerie esterne statiche (in formato .OBJ/.LIB) o dinamiche (in formato .DLL).

### *9.2 Locating*

Il comando *Locating* effettua la ricerca di SmartKey **con una Label prefissata su tutte le porte del sistema sia parallele sia USB** e consente al software protetto di individuare su quale di queste è posta la chiave stessa.

Il risultato dell'operazione di *Locating* è l'identificatore della porta su cui è installata la chiave. Questo identificatore deve essere utilizzato per tutte le successive operazioni sulla chiave. Non si possono fare assunzioni sul valore di questo identificatore, in quanto dipende dal modello di SmartKey utilizzato, dalla versione dei driver, dal sistema operativo installato e dalla configurazione del PC. Il valore dell'identificatore deve solamente essere letto dal campo *lpt* dopo il comando *Locating* ed utilizzato in tutti gli altri comandi fino al termine dell'applicazione.

L'utilizzo della funzione *Locating* consente di rendere il software protetto indipendente dalla porta parallela/USB sulla quale l'utilizzatore installerà la chiave.

Dopo il comando *Locating* il campo *lpt* port è impostato con l'identificatore della porta sulla quale è presente la chiave.

Ricordiamo che con le chiavi FX la *Label* non è programmabile e coincide con l'*Id-Code*, mentre con le chiavi programmabili la *Label* può essere programmata con una sequenza qualunque di 16 byte. In entrambi i casi la modalità *Locating* è pienamente operativa.

Lo scambio di informazioni è così organizzato:

| Models | ALL | |
|---|---|---|
| Input | COMMAND | 'L' |
| | LABEL | Label |
| Output | LPT | Porta |
| | STATUS | Stato<br>==0 SmartKey trovata<br>!=0 SmartKey non trovata |

**Tabella 7** Parametri per il comando *Locating*.

### 9.2.1 Passaggio parametri

Parametri da trasferire per eseguire un'operazione di *Locating*, con ricerca su tutte le porte parallele ed USB, di una chiave avente come *Label* "SMARTKEY".

| | | |
|---|---|---|
| COMMAND | `4C 00` | Locating (`"L"`) |
| LABEL | `53 4D 41 52 54 4B 45 59`<br>`00 00 00 00 00 00 00 00` | Label (`"SMARTKEY"`) |

**Tabella 8** Scambio di informazioni per il comando *Locating*.

Se è presente una chiave contenente la Label passata su una porta, il campo *lpt* ne conterrà l'identificatore.

## *9.3 Scrambling*

La modalità di *Scrambling*, supportata da tutti i modelli di chiavi SmartKey, si basa sulla singola personalizzazione per ogni cliente dell' *Id-Code* register. **Ha la funzione di individuare per via algoritmica se l'Id-Code è corretto**.

Il codice di identificazione contenuto nel registro *Id-Code* è utilizzato come parametro fondamentale di una funzione matematica di codifica: un insieme di dati è inviato, elaborato e restituito opportunamente crittografato in modo unico per ogni *Id-Code*. La verifica sulla presenza della chiave può pertanto essere effettuata confrontando il dato elaborato con quello atteso.

Chiavi con differenti *Id-Code* utilizzano codifiche differenti e quindi a parità di dati in ingresso, i dati restituiti saranno diversi. Ad ogni diverso *Id-Code* è pertanto associabile una tabella di corrispondenza tra dati originali e dati scramblati.

L'algoritmo di *Scrambling* utilizzato è ad alta sicurezza e di tipo non lineare.

Lo scambio di informazioni è così organizzato:

| | | |
|---|---|---|
| Models | ALL | |
| Input | COMMAND | 'S' |
| | LPT | Porta |
| | LABEL | Label |
| | DATA[0..7] | Dati originali (8 byte) |
| Output | DATA[0..7] | Dati scramblati (8 byte) |
| | STATUS | Stato<br>==0 Successo<br>!=0 Errore |

**Tabella 9** Parametri per il comando *Scrambling*.

Se è stata effettuata precedentemente una operazione di *Locating*, al campo Lpt viene automaticamente assegnato il valore corretto; non è quindi necessario che il programmatore si preoccupi di assegnarne un valore.

La funzione di *Scrambling* non altera il contenuto della memoria di SmartKey, ma utilizza il campo *data* come variabile di appoggio per i dati.

### 9.3.1 Passaggio parametri

Parametri da trasferire per eseguire un'operazione di *Scrambling*, con ricerca della chiave sulla porta parallela LPT1:

| | | |
|---|---|---|
| LPT | `01 00` | Porta |
| COMMAND | `53 00` | Scrambling (`"S"`) |
| LABEL | `53 4D 41 52 54 4B 45 59`<br>`00 00 00 00 00 00 00 00` | Label (`"SMARTKEY"`) |
| DATA | `XX XX XX XX XX XX XX XX` | Dati da scramblare (8 byte) |

**Tabella 10** Scambio di informazioni per il comando *Scrambling*.

Al termine dell'operazione i primi 8 byte del campo *data* sono sostituiti con i dati scramblati che dipendono sia dai dati originali sia dall'*Id-Code* della chiave.

In questo caso si rende necessaria un'inizializzazione del campo LPT in funzione della porta interessata. Tale operazione può essere evitata utilizzando il comando *Locating* per la ricerca automatica su tutte le porte inserite sul calcolatore.

## *9.4 Reading*

I modelli programmabili della famiglia SmartKey sono dotati di un sistema di protezione basato sull'accesso selettivo in lettura e scrittura al registro *Secure Data,* con password programmabile da software.

Ogni software-house può pertanto codificare personalmente le chiavi in proprio possesso, con il solo ausilio delle utility software in dotazione e senza necessità di dispositivi di programmazione esterni. L'amministrazione dei codici d'accesso e del contenuto della memoria non volatile è gestita direttamente dalla software-house, che diviene l'unica depositaria dei codici di personalizzazione.

Per quanto riguarda la lettura, la funzione di *Reading* permette di accedere ai campi *data* e *ext_data* per verificarne il contenuto e confrontarlo con quello atteso. È necessaria la conoscenza di *Label* e *Password*.

Il comando *Reading* permette l'accesso ai primi 416 byte di memoria della chiave. Se la chiave ha più di 416 byte di memoria, è necessario utilizzare il comando *BlockReading* per poterci accedere completamente.

Nel caso di modelli *EP, SP, XM* e *NET* è disponibile in lettura anche il valore del registro *FailCounter*.

Lo scambio di informazioni è così organizzato:

| Models | PR, EP, SP, XM, NET | |
|---|---|---|
| Input | COMMAND | 'R' |
| | LPT | Porta |
| | LABEL | Label |
| | PASSWORD | Password |
| Output | DATA | Dati letti |
| | EXT_DATA | Dati estesi letti (solo per modelli con più di 64 bytes di memoria) |
| | FAIL_CTR | Fail Counter (solo per modelli EP, SP, XM e NET) |
| | STATUS | Stato<br>==0 Successo<br>!=0 Errore |

**Tabella 11** Parametri per il comando *Reading*.

Se la chiave è trovata sulla porta indicata e con valori di *Label* e *Password* corretti il campo *data* conterrà i primi 64 byte dei *Secure Data* letti e nel caso di chiavi *SP, XM* e *NET* il campo *ext_data* conterrà i restanti 352 byte.

Se la *Password* passata alla chiave non è corretta, **i Secure Data ed il Fail Counter sono generati in modo pseudocasuale**.

### 9.4.1 Passaggio parametri

Parametri da trasferire per eseguire un'operazione di *Reading* su una chiave presente sulla porta parallela LPT1:

| LPT | `01 00` | Porta |
|---|---|---|
| COMMAND | `52 00` | Reading (`"R"`) |
| LABEL | `53 4D 41 52 54 4B 45 59`<br>`00 00 00 00 00 00 00 00` | Label (`"SMARTKEY"`) |
| PASSWORD | `45 55 54 52 4F 4E 00 00`<br>`00 00 00 00 00 00 00 00` | Password (`"EUTRON"`) |

**Tabella 12** Scambio di informazioni per il comando *Reading*.

## *9.5 Writing mode*

Conoscendo la Password, la scrittura dei *Secure Data* può avvenire con le stesse modalità previste per la lettura. La funzione di *Writing* modifica in linea il contenuto dei *Secure Data*.

Il comando *Writing* permette l'accesso ai primi 416 byte di memoria della chiave. Se la chiave ha più di 416 byte di memoria, è necessario utilizzare il comando *BlockWriting* per poterci accedere completamente.

Lo scambio di informazioni tra PC e SmartKey è così organizzato:

| Models | PR, EP, SP, XM, NET | |
|---|---|---|
| Input | COMMAND | 'W' |

| | | |
|---|---|---|
| | LPT | Porta |
| | LABEL | Label |
| | PASSWORD | Password |
| | DATA | Dati da scrivere |
| | EXT_DATA | Dati estesi da scrivere (solo per modelli SP, XM e NET) |
| Output | STATUS | Stato<br>==0 Successo<br>!=0 Errore |

**Tabella 13** Parametri per il comando *Writing*.

Se la chiave è trovata sulla porta indicata e con valori di *Label* e *Password* corretti, il campo *data* (ed eventualmente il campo *ext_data* se la chiave ha sufficiente memoria) sarà trasferito nei *Secure Data* della chiave.

Se la password passata alla chiave non è corretta, **i dati presenti nel Secure Data register non sono alterati**.

### 9.5.1 Passaggio parametri

Parametri da trasferire per eseguire un'operazione di *Writing* su una chiave inserita nella porta parallela LPT1:

| | | |
|---|---|---|
| LPT | `01 00` | Porta |
| COMMAND | `57 00` | Reading (`"W"`) |
| LABEL | `53 4D 41 52 54 4B 45 59`<br>`00 00 00 00 00 00 00 00` | Label (`"SMARTKEY"`) |
| PASSWORD | `45 55 54 52 4F 4E 00 00`<br>`00 00 00 00 00 00 00 00` | Password (`"EUTRON"`) |
| DATA | `XX XX XX XX XX XX XX XX`<br>… | Data |
| EXT_DATA | `XX XX XX XX XX XX XX XX`<br>… | Ext Data |

**Tabella 14** Scambio di informazioni per il comando *Writing*.

## *9.6 Block Reading*

La funzione *BlockReading* consente di leggere porzioni dei *Secure Data*, ad esempio una, due o poche word piuttosto che l'intero campo. Ciò consente di risparmiare qualche frazione di secondo rispetto alla lettura completa tramite *Reading*.

Il comando *BlockReading* è l'unico comando di lettura che permette l'accesso a tutta la memoria della chiave.

Lo scambio di informazioni è così organizzato:

| Models | PR, EP, SP, XM, NET | |
|---|---|---|
| Input | COMMAND | 'BR' |
| | LPT | Porta |
| | LABEL | Label |
| | PASSWORD | Password |
| | DATA[0,1] | Puntatore alla prima word da leggere (da 0 a 31 per modelli con 64 bytes di memoria, da 0 a 63 per 128 bytes di memoria, da 0 a 207 per 416 byte di memoria, da 0 a 447 per 896 bytes di memoria) (2 byte) |
| | DATA[2,3] | Numero di word da leggere (da 1 a 16) (2 byte) |
| Output | DATA[4,…] | Valori da leggere nell'area indicata dai due parametri precedenti. (2 - 32 byte) |
| | STATUS | Stato<br>==0 Successo<br>!=0 Errore |

**Tabella 15** Parametri per il comando *BlockReading*.

Se la *Password* passata alla chiave non è corretta, **i Secure Data ed il Fail Counter sono generati in modo pseudocasuale**

### 9.6.1 Passaggio parametri

Parametri da trasferire per eseguire un'operazione di *BlockReading* su una chiave presente sulla porta parallela LPT1; è richiesta la lettura di 15 word (000F hex = a 30 byte) a partire dalla dodicesima word (000B hex = indirizzo 11).

Si ricorda che nella struttura di passaggio parametri i primi due byte del campo *data* sono riservati all'indirizzo della prima word da leggere, i successivi due byte contengono il numero di word da leggere, dal quinto byte in poi sarà contenuto, al termine dell'operazione, il blocco dati letto.

| | | |
|---|---|---|
| LPT | `01 00` | Porta |
| COMMAND | `52 42` | Block Reading (`"BR"`) |
| LABEL | `53 4D 41 52 54 4B 45 59`<br>`00 00 00 00 00 00 00 00` | Label (`"SMARTKEY"`) |
| PASSWORD | `45 55 54 52 4F 4E 00 00`<br>`00 00 00 00 00 00 00 00` | Password (`"EUTRON"`) |
| DATA | `0B 00 0F 00` | Address and Number of Words |

**Tabella 16** Scambio di informazioni per il comando *BlockReading*.

## *9.7 **Block Writing***

La funzione *BlockWriting* consente di scrivere porzioni dei *Secure Data*, ad esempio una, due o poche word piuttosto che l'intero campo. Ciò consente di risparmiare qualche frazione di secondo rispetto alla scrittura completa tramite *Writing*.

Il comando *BlockWriting* è l'unico comando di scrittura che permette l'accesso a tutta la memoria della chiave.

Lo scambio di informazioni è così organizzato:

| | | |
|---|---|---|
| Models | PR, EP, SP, XM, NET | |
| Input | COMMAND | 'BW' |
| | LPT | Porta |
| | LABEL | Label |
| | PASSWORD | Password |
| | DATA[0,1] | Puntatore alla prima word da scrivere (da 0 a 31 per modelli con 64 bytes di memoria, da 0 a 63 per 128 bytes di memoria, da 0 a 207 per 416 byte di memoria, da 0 a 447 per 896 bytes di memoria) (2 byte) |
| | DATA[2,3] | Numero di word da scrivere (da 1 a 16) (2 byte) |
| | DATA[4,…] | Valori da scrivere nell'area indicata dai due parametri precedenti. (2 - 32 byte) |
| Output | STATUS | Stato<br>==0 Successo<br>!=0 Errore |

**Tabella 17** Parametri per il comando *BlockWriting*.

Se la *Password* passata alla chiave non è corretta, i dati presenti nel *Secure Data* register non sono alterati.

### 9.7.1 Passaggio parametri

Parametri da trasferire per eseguire un'operazione di *BlockWriting* su una chiave presente sulla porta parallela LPT1; è richiesta la scrittura di 10 word (000A hex = 20 byte) a partire dalla prima word (0000 hex = indirizzo 0).

Si ricorda che nella struttura di passaggio parametri i primi due byte del campo *data* sono riservati all'indirizzo della prima word da scrivere, i successivi due byte contengono il numero di word, dal quinto byte in poi sono contenuti i byte da scrivere.

| | | |
|---|---|---|
| LPT | `01 00` | Porta |
| COMMAND | `46 00` | Fixing Mode (`"F"`) |

| | | |
|---|---|---|
| LABEL | 53 4D 41 52 54 4B 45 59<br>00 00 00 00 00 00 00 00 | Label ("SMARTKEY") |
| PASSWORD | 45 55 54 52 4F 4E 00 00<br>00 00 00 00 00 00 00 00 | Password("EUTRON") |
| DATA | 0B 00 0F 00 XX XX XX XX<br>XX XX XX XX XX XX XX XX<br>XX XX XX XX XX XX XX XX | Address, Number of Words, and Data |

**Tabella 18** Scambio di informazioni per il comando *BlockWriting*.

### *9.8 Fixing*

Fissare i dati significa rendere i registri *Label*, *Password* e *Secure Data* non più riprogrammabili: **è possibile, in pratica, congelare i dati preventivamente programmati**. Dopo aver eseguito la modalità di *Fixing* non è più possibile cambiare in alcun modo il contenuto di SmartKey. La modalità di fissaggio dei dati è disponibile, unitamente al *Fail Counter*, solo sui modelli *EP*, *SP*, *XM* e *NET***.**

La possibilità di fissare i dati, permette alla software-house di generare chiavi totalmente personalizzate e non più alterabili; sono pertanto dissuasi i tentativi di alterazione del contenuto al fine di variare ad esempio priorità di accessi a banche dati oppure di abilitare moduli software non previsti.

È opportuno che l'operazione *Fixing* non sia effettuata durante la fase di test, perché altrimenti non potrete più programmare la chiave.

Consigliamo quindi che solo nella fase finale di implementazione della protezione sia valutata l'opportunità o meno di fissare i contenuti programmati nella chiave. È importante tenere conto che le funzionalità aggiuntive che coinvolgono la scrittura della memoria non saranno più disponibili.

Questo comando non può essere eseguito in rete, ma solo con la chiave usata localmente.

Lo scambio di informazioni è così organizzato:

| | | |
|---|---|---|
| Models | EP, SP, XM, NET | |
| Input | COMMAND | 'F' |
| | LPT | Porta |
| | LABEL | Label |
| | PASSWORD | Password |
| | DATA | Dati contenuti sulla chiave |
| | EXP_DATA | Dati estesi contenuti sulla chiave (solo per modelli SP, XM e NET) |
| Output | STATUS | Stato<br>==0 Successo<br>!=0 Errore |

**Tabella 19** Parametri per il comando *Fixing*.

Il *Fixing* della memoria è eseguito **soltanto se i parametri inviati Label, Password e Secure Data coincidono con il contenuto dei rispettivi registri**.

#### 9.8.1 Passaggio parametri

Parametri da trasferire per eseguire un'operazione di *Fixing* con chiave sulla porta parallela LPT1, con invio dei campi *label*, *password* e *data* e confronto con il contenuto atteso. Se i parametri coincidono, essi sono fissati.

| | | |
|---|---|---|
| LPT | 01 00 | Porta |
| COMMAND | 46 00 | Fixing Mode("F") |
| LABEL | 53 4D 41 52 54 4B 45 59<br>00 00 00 00 00 00 00 00 | Label ("SMARTKEY") |
| PASSWORD | 45 55 54 52 4F 4E 00 00<br>00 00 00 00 00 00 00 00 | Password ("EUTRON") |
| DATA | 53 45 43 55 52 49 54 59 | Data ("SECURITY DATA") |

| | | |
|---|---|---|
| | 20 44 41 54 41 00 00 00<br>00 00 00 00 00 00 00 00<br>00 00 00 00 00 00 00 00<br>00 00 00 00 00 00 00 00<br>00 00 00 00 00 00 00 00<br>00 00 00 00 00 00 00 00 | |
| EXP_DATA | 00 00 00 00 00 00 00 00<br>… | Exp Data |

**Tabella 20** Scambio di informazioni per il comando *Fixing*.

### *9.9 Programming*

La funzione di *Programming* permette di riprogrammare completamente la chiave ed in particolar modo i registri *Label* e *Password*.

La disponibilità della funzione *Programming* consente di mettere a punto un proprio programma dedicato alla preconfigurazione delle chiavi stesse, normalmente collegato ad un database, al fine di associare il contenuto di ogni chiave con un elenco clienti e/o prodotti.

La programmazione di nuove *Label* e *Password* non richiede la conoscenza della *Password* precedente (mentre è necessario conoscere la *Label* corrente), perché l'operazione azzera automaticamente il registro *Secure Data* impostato (è riempito con 0). Per ovvie ragioni legate alla sicurezza, non è invece azzerato il *Fail Counter*.

Se durante le prove vi foste dimenticati il valore della *Label* corrente, potete riportarvi nella situazione di default tramite l'utility SPC.

La modalità *Programming* è necessaria per la programmazione off-line della chiave. Consigliamo però di non utilizzarla on-line nel software da proteggere, sia per ragioni di sicurezza e sia per evitare che un errore di programmazione faccia agire la funzione su una diversa SmartKey presente nel sistema. Inoltre, normalmente è sufficiente agire sulla chiave tramite le funzioni *Reading* e *Writing* che limitano il proprio intervento al regiostro *Secure Data*.

Questo comando non può essere eseguito in rete, ma solo con la chiave usata localmente.

Lo scambio di informazioni è così organizzato:

| Models | PR, EP, SP, XM, NET | |
|---|---|---|
| Input | COMMAND | ‘P’ |
| | LPT | Porta |
| | LABEL | Nuova Label |
| | PASSWORD | Nuova Password |
| | DATA[0..15] | Label corrente (16 byte) |
| Output | STATUS | Stato<br>==0 Successo<br>!=0 Errore |

**Tabella 21** Parametri per il comando *Programming.*

#### 9.9.1 Passaggio parametri

Parametri da trasferire per eseguire un'operazione di *Programming*, con accesso alla chiave sulla porta parallela LPT2, programmando *Label* e *Password*. Si ipotizza che la *Label* memorizzata prima dell'operazione sia "LABELOLD".

| | | |
|---|---|---|
| LPT | 01 00 | Porta |
| COMMAND | 50 00 | Programming("P") |
| LABEL | 53 4D 41 52 54 4B 45 59<br>00 00 00 00 00 00 00 00 | Label ("SMARTKEY") |
| PASSWORD | 45 55 54 52 4F 4E 00 00<br>00 00 00 00 00 00 00 00 | Password ("EUTRON") |
| DATA | 4C 41 42 45 4C 4F 4C 44 | Current Label (“LABELOLD”) |

| | | |
|---|---|---|
| | 00 00 00 00 00 00 00 00 | |

**Tabella 22** Scambio di informazioni per il comando *Programming*.

Se la chiave è trovata sulla porta indicata, i valori di *Label* e *Password* sono trasferiti alla chiave, mentre il registro *Secure Data* nella chiave sono azzerati automaticamente. La *Label* corrente utilizza i primi 16 byte del campo *data*, normalmente utilizzati per il trasferimento dei *Secure Data*.

### *9.10 Comparing*

Per rendere più semplice l'utilizzo delle chiavi a chi per la prima volta si avvicina alle tecniche di protezione del software, è stato introdotto il *Comparing* mode.

Si tratta del meccanismo più semplice per utilizzare le chiavi SmartKey di tipo programmabile stand-alone, cioè i modelli *PR, EP, SP* e *XM*. Esso consente di effettuare la verifica automatica su tutte le porte del sistema della presenza di una chiave di protezione con *Label*, *Password* e *Secure Data* assegnati, ottenendone un'informazione che indica se la chiave è presente e su quale porta.

Il *Comparing* è pertanto un'estensione della modalità di *Locating* valida solo per le chiavi programmabili. Può essere utile nelle applicazioni ove è sufficiente una verifica della presenza della chiave, senza la necessità di aggiornarne il contenuto tramite operazioni di riscrittura dei *Secure Data*.

Lo scambio di informazioni è così organizzato:

| | | |
|---|---|---|
| Models | PR, EP, SP, XM | |
| Input | COMMAND | 'C' |
| | LABEL | Label |
| | PASSWORD | Password |
| | DATA | Data |
| Output | FAIL_CTR | Fail Counter |
| | STATUS | Stato<br>>=0 Successo, numero della porta su cui si trova la chiave<br><0 Errore |

**Tabella 23** Parametri per il comando *Comparing*.

Lo campo *status* indica se l'operazione è andata a buon fine ritornando il numero di porta o un valore minore di 0 in caso d'errore.

#### 9.10.1 Passaggio parametri

Parametri da trasferire per eseguire un'operazione di *Comparing* con ricerca della chiave su tutte le porte parallele presenti:

| | | |
|---|---|---|
| COMMAND | 50 00 | Comparing ("C") |
| LABEL | 53 4D 41 52 54 4B 45 59<br>00 00 00 00 00 00 00 00 | Label ("SMARTKEY") |
| PASSWORD | 45 55 54 52 4F 4E 00 00<br>00 00 00 00 00 00 00 00 | Password ("EUTRON") |
| DATA | 53 45 43 55 52 49 54 59<br>20 44 41 54 41 00 00 00<br>00 00 00 00 00 00 00 00<br>00 00 00 00 00 00 00 00<br>00 00 00 00 00 00 00 00<br>00 00 00 00 00 00 00 00<br>00 00 00 00 00 00 00 00<br>00 00 00 00 00 00 00 00 | Data("SECURITY DATA") |

**Tabella 24** Scambio di informazioni per il comando *Comparing*.

Se la chiave è trovata su una qualunque delle porte presenti nel sistema e con i valori di *Label, Password* e *Secure Data* corretti, la variabile di status assumerà al termine dell'operazione il valore 1, 2 o 3 in funzione della LPT.

### *9.11 Model Reading*

La funzione *ModelReading* permette di identificare il modello della SmartKey inserita. Si tratta di una funzione accessoria, che può essere utilizzata ad esempio per attivare un comportamento diverso per pacchetti software funzionanti sia in versione standalone sia in rete.

Lo scambio di informazioni è così organizzato:

| | | |
|---|---|---|
| Models | ALL | |
| Input | COMMAND | 'M' |
| | LPT | Porta |
| | LABEL | Label |
| Output | DATA[0] | SmartKey model<br>= '1', FX<br>= '2', PR<br>= '3', EP<br>= '9', SP<br>= 'A', NET<br>= 'D', XM |
| | DATA[1] | Memoria disponibile sulla chiave<br>= '0', 0 byte<br>= '1', 64 byte<br>= '2', 128 byte<br>= '3', 416 byte<br>= '4', 896 byte<br>= '8', 8192 byte |
| | STATUS | Stato<br>==0 Successo<br>!=0 Errore |

**Tabella 25** Parametri per il comando *ModelReading*.

#### 9.11.1 Passaggio parametri

Parametri da trasferire per eseguire un'operazione di *ModelReading*, con accesso della chiave sulla porta parallela LPT1.

| | | |
|---|---|---|
| LPT | `01 00` | Porta |
| COMMAND | `4D 00` | Model Reading (`"M"`) |
| LABEL | `53 4D 41 52 54 4B 45 59`<br>`00 00 00 00 00 00 00 00` | Label (`"SMARTKEY"`) |

**Tabella 26** Scambio informazioni per il *ModelReading*.

Se la chiave è trovata sulla porta indicata, il modello della chiave è disponibile nel primo byte del campo *data.*

### *9.12 Serial Number Reading*

Questo comando legge il *Serial Number* della SmartKey. Il *Serial Number* è un numero di 32-bit unico per ogni SmartKey.

Lo scambio di informazioni è così organizzato:

| | | |
|---|---|---|
| Models | ALL | |
| Input | COMMAND | 'N' |
| | LPT | Porta |
| | LABEL | Label |
| Output | DATA[0,1,2,3] | Numero seriale |

| | | |
|---|---|---|
| | STATUS | Stato<br>==0 Successo<br>!=0 Errore |

**Tabella 27** Parametri per il comando *SerialNumberReading*.

### *9.13 Ext Model Reading*

Questo comando legge le informazioni estese sul modello di SmartKey.

Lo scambio di informazioni è così organizzato:

| | | |
|---|---|---|
| Models | ALL | |
| Input | COMMAND | 'H' |
| | LPT | Porta |
| | LABEL | Label |
| Output | DATA[0] | Modello<br>= '1', FX<br>= '2', PR<br>= '3', EP<br>= '9', SP<br>= 'A', NET<br>= 'D', XM |
| | DATA[1] | Memoria disponibile sulla chiave<br>= '0', 0 byte<br>= '1', 64 byte<br>= '2', 128 byte<br>= '3', 416 byte<br>= '4', 896 byte<br>= '8', 8192 byte |
| | DATA[2] | Modello hardware<br>= 2, Parallel<br>= 3, USB<br>= 4, USB DL (Driver Less) |
| | DATA[3] | Funzionalità (organizzata come una maschera di bit)<br>• **bit 0 (valore 1)** - comandi AES_SET_MODE e AES_SCRAMBLE_MODE supportati<br>• **bit 1 (valore 2)** - protezione anti-condivisione ed anti-emulazione attivo<br>• **bit 3 (valore 8)** - le chiavi AES sono impostate tramite il comando AES_SET_MODE |
| | STATUS | Stato<br>==0 Successo<br>!=0 Errore |

**Tabella 28** Parametri per il comando *ExtModelReading.*

### *9.14 Fix Reading*

Questo comando legge il valore del registro di *Fix*.

Lo scambio di informazioni è così organizzato:

| | | |
|---|---|---|
| Models | EP, SP, XM, NET | |
| Input | COMMAND | 'X' |

| | | |
|---|---|---|
| | LPT | Porta |
| | LABEL | Label |
| | PASSWORD | Password |
| Output | DATA[0] | = 1, La SmartKey è Fixed<br>= 0, La SmartKey non è Fixed |
| | STATUS | Stato<br>==0 Successo<br>!=0 Errore |

**Tabella 29** Parametri per il comando *FixReading*.

### *9.15 Fail Counter Reading*

Questo comando legge il valore del registro *Fail Counter*. Questo è lo stesso valore che si ottiene con il comando *Reading*. Con questo comando è possibile ottenere il valore del registro senza leggere tutta la memoria.

Lo scambio di informazioni è così organizzato:

| Models | EP, SP, XM, NET | |
|---|---|---|
| Input | COMMAND | 'A' |
| | LPT | Porta |
| | LABEL | Label |
| | PASSWORD | Password |
| Output | FAIL_COUNTER | Valore del Fail Counter |
| | STATUS | Stato<br>==0 Successo<br>!=0 Errore |

**Tabella 30** Parametri per il comando *FailCounterReading*.

### *9.16 Autenticazione AES*

L'autenticazione AES, supportata da tutti modelli SmartKey 3 USB, è basata sulla personalizzazione da parte dell'utente di una ventina di codici di sicurezza. **Ha la funzione di individuare per via algoritmica la presenza della SmartKey**.

Questi comandi permettono un nuovo tipo di autenticazione basata sull'algoritmo AES a 128 bit, alternativo al comando *Scrambling* senza la necessità di usare una grossa tabella di coppie input/output di valori di scrambling.

E' possibile verificare se la SmartKey supporta l'autenticazione AES tramite il comando *ExtModelReading* e verificando il corrispondente bit di funzionalità.

#### 9.16.1 Autenticazione

Per identificare la chiave SmartKey, l'applicazione invia un valore casuale alla chiave. La chiave risponde con la crittografia del numero seriale precedentemente memorizzato dopo aver effettuato l'operazione XOR con il valore inviato dall'applicazione.

```
RESULT = AES_ENCRYPT( RAND XOR SERIAL )
```

L'applicazione può quindi decrittografare il risultato e ottenere il seriale rifacendo l'operazione XOR con il valore casuale.

```
SERIAL = AES_DECRYPT( RESULT ) XOR RAND
```

Se il numero seriale è valido si può assumere che una SmartKey sia presente.

- L'utilizzo del valore casuale garantisce che la risposta della chiave sia sempre diversa e quindi non riutilizzabile.
- L'utilizzo della crittografia permette solo all'applicazione di interpretare correttamente il numero seriale.
- L'applicazione può verificare la correttezza del seriale impostando a priori parte di tale numero ad un valore fisso. Ad esempio, con il seriale di 16 byte, 8 di tali byte possono essere impostati sempre a 0 per tutti i seriali. Se dopo la decrittografia questi byte sono correttamente a zero, l'applicazione è sicura che la riposta che ha

ottenuto dalla chiave è valida. Tale verifica garantisce che il programma stia comunicando con una vera SmartKey e non con una SmartKey emulata.

### 9.16.2 Utilizzo

Per l'utilizzo dei nuovi comandi, l'applicazione da proteggere deve utilizzare i driver standalone e multilan (solo con il protocollo LOCAL) che fornisce anche i nuovi comandi d'autenticazione, oltre alla normale interfaccia SmartKey.

Per com'è strutturato il protocollo, l'applicazione dovrà contenere un'implementazione dell'algoritmo AES a 128 bit e un generatore di numeri casuali per realizzare la comunicazione con la chiave SmartKey. È da sottolineare che tale implementazione deve essere necessariamente inclusa nell'applicazione e non nel driver SmartKey, in quanto se si spostasse la verifica d'autenticità nel driver SmartKey, l'applicazione potrebbe essere ingannata con un finto driver.

Per rendere efficace la protezione, l'applicazione dovrà anche contenere tutta una serie d'artifizi di programmazione per nascondere la chiave di crittografia necessaria al processo d'autenticazione.

## *9.17 AES Set*

Questo comando imposta 20 differenti chiavi di crittografia di 16 byte ciascuna per l'algoritmo AES ed il numero seriale di 16 byte. Le chiavi di crittografia ed il seriale una volta scritti non possono essere mai estratti e nemmeno sovrascritti ripetendo il comando. In altre parole, questo comando può essere eseguito solo una volta.

**ATTENZIONE! le chiavi AES possono essere impostate SOLO UNA VOLTA!**

Questo comando non può essere eseguito in rete, ma solo con la chiave usata localmente.

Lo scambio di informazioni è così organizzato:

| Modello | Solo SmartKey 3 USB | |
|---|---|---|
| Input | COMMAND | 'G' |
| | LPT | SmartKey port |
| | LABEL | SmartKey Label |
| | EXTDATA[0..15] | Numero seriale da impostare |
| | EXTDATA[16..31] | Prima chiave AES |
| | … | |
| | EXTDATA[320..335] | Ventesima chiave AES |
| Output | | Niente |

**Tabella 31** Parametri per il comando *AESSet*.

## *9.18 AES Scramble*

Questo comando effettua l'operazione di autenticazione AES. La SmartKey opera su un valore casuale scelto dall'applicazione e fornisce un valore di risposta che permette all'applicazione di verificare la presenza della chiave stessa secondo il protocollo precedentemente descritto.

Prima di utilizzare questo comando è necessario avere impostato le chiavi AES tramite il comando *AESSet*.

Lo scambio di informazioni è così organizzato:

| Modello | Solo SmartKey 3 USB | |
|---|---|---|
| Input | COMMAND | 'O' |
| | LPT | SmartKey port |
| | LABEL | SmartKey Label |
| | DATA[0..15] | Dati casuali da utilizzare |
| | DATA[16] | Chiave AES da utilizzare. La prima chiave ha indice 0. L'ultima indice 19. |
| Output | DATA[0..15] | Risultato del protocollo. |

**Tabella 32** Parametri per il comando *AESScramble*.

## *9.19 Errori*

Dopo l'esecuzione di un comando, nel campo *status* della struttura di comunicazione sarà presente uno dei seguenti valori:

| Nome | Valore | Descrizione |
|---|---|---|
| ST_OK | 0 | Operazione completata con successo. |
| ST_NONE_KEY | -1 | Dispositivo non trovato. Questo errore può essere causato da:<br>• SmartKey non inserita correttamente nella porta Parallela o USB.<br>• Driver non installati correttamente.<br>• LABEL utilizzata errata. |
| ST_SYNT_ERR | -2 | Errore di sintassi nel comando. Questo errore può essere causato da:<br>• Uno degli argomenti utilizzati nel comando è errato.<br>• Il comando non è supportato dal modello di SmartKey. Ad esempio la lettura della memoria su una SmartKey senza memoria.<br>• Il comando è impedito dallo stato corrente della SmartKey. Ad esempio la scrittura della memoria in una SmartKey con attivato il flag di *Fixing* della memoria. |
| ST_LABEL_FAILED | -3 | La LABEL utilizzata è errata. A seconda del comando o del modello di SmartKey può essere ritornato questo errore o l'errore ST_NONE_KEY. |
| ST_PW_DATA_FAILED | -4 | La PASSWORD utilizzata è errata. |
| ST_HW_FAILURE | -20 | Un controllo di integrità sul dispositivo hardware è fallito. |

### *9.20 Alcuni suggerimenti per l'utilizzo delle funzioni di SmartKey*

Un utilizzo combinato delle funzioni elencate consente la messa a punto di criteri di protezione di elevata sicurezza e flessibilità. Riteniamo comunque opportuno indicare alcuni suggerimenti *minimi* per un'implementazione sicura della protezione.

- Utilizzare sempre la modalità *Locating*, per individuare la presenza della chiave: ciò vi consentirà di rendere l'applicativo indipendente dalla porta parallela sulla quale è installata SmartKey.
- Utilizzare in diversi punti del programma la modalità *Scrambling* con valori differenti o ancora meglio la modalità *AESScrambling*. Se l'esito dei confronti con sarà positivo sarete certi che nel PC è installata una chiave col vostro *Id-Code*.
- Se utilizzate chiavi programmabili, effettuate in diversi punti del programma un'operazione di *Reading*: potrete confrontare i valori letti con quelli attesi ed assicurarvi che il software che sta funzionando è effettivamente autorizzato grazie alla particolare programmazione della chiave SmartKey presente.

Il capitolo 10 introduce suggerimenti ulteriori per un'implementazione sicura della protezione.

# 10 Tecniche di protezione dei programmi ed esempi

Utilizzando le chiavi hardware di protezione SmartKey avete introdotto un fortissimo deterrente contro i tentativi di duplicazione abusiva del software; tuttavia è bene ricordare che anche nella lotta per la protezione del software vale un principio tipico di tutti i sistemi di sicurezza:

**Un sistema di sicurezza ha lo stesso grado di vulnerabilità del suo componente più debole**

È pertanto necessario porre molta attenzione non solo al tipo di protezione, ma anche ai metodi di implementazione software della protezione stessa.

In altre parole, utilizzare una chiave di protezione hardware senza una sua accurata implementazione nel software significa arrestare gli hacker mediamente smaliziati (gli hobbisti, gli utilizzatori occasionali, ecc.), ma non organizzazioni criminal-commerciali, magari con tempo, risorse economiche e competenza a disposizione per copiare il software o impossessarsi di dati riservati.

Abbiamo pertanto ritenuto opportuno elencare nel seguito alcune tecniche e suggerimenti utili. La scelta delle tecniche da utilizzare dipende dalle singole applicazioni, dal costo e dal livello di riservatezza del software e/o dei dati protetti.

In generale per rendere la vita difficile ai potenziali pirati del software è opportuno tener presente i seguenti suggerimenti e considerazioni:

- Utilizzare più di una delle tecniche di protezione indicate nel seguito.
- Distribuire le misure di protezione lungo tutto il programma.
- Se è eliminata una delle misure di protezione, le rimanenti dovrebbero fare in modo che il programma sembri funzionare correttamente per un certo periodo, dopo di che con tempi e modi casuali si arresti.

Ricordate infine che l'aspetto psicologico è importantissimo: l'aggressore non potrà mai essere sicuro di aver disinnescato tutti i meccanismi di protezione; se ogni volta che crede di averne trovato uno, il software protetto fa emergere a distanza di tempo (anche ore o giorni!) un altro problema, probabilmente la frustrazione sarà tale da far rinunciare all'attacco.

Nel proteggere la vostra applicazione occorre considerare gli attacchi tipici e cercare di contrastare almeno questi casi più comuni. I casi sono:

- Reverse engineering dell'applicazione e rimozione di qualsiasi chiamata a SmartKey API
- Utilizzo di un emulatore High Level (User Level) in grado di intercettare e registrare qualsiasi chiamata SmartKey API e di simulare il comportamento di SmartKey API. Questi emulatori potenzialmente conoscono la semantica delle chiamate API e sono in grado di emulare SmartKey, ma con l'eccezione dell'operazioni di *Scrambling* e *AES.*
- Utilizzo di un emulatore Low Level (Kernel o Hardware) in grado di intercettare e registrare le comunicazioni fisiche su una porta Parallela o USB e di simulare il comportamento fisico di SmartKey. Generalmente questi emulatori ignorano totalmente la semantica delle comunicazioni.

Le seguenti linee di guida potranno aumentare notevolmente la protezione da questi attacchi.

## *10.1 Linee di guida generali*

Le seguenti linee di guida sono valide per tutti i modelli di SmartKey.

### 10.1.1 Controllare la chiave in diversi punti della vostra applicazione

L'applicazione non dovrebbe controllare la presenza di SmartKey soltanto in un punto dell'esecuzione. Il controllo di SmartKey dovrebbe essere duplicato in vari punti del programma. È possibile eseguire l'operazione *Locate*/*Open* soltanto all'avvio, ma le altre operazione vanno eseguite in molti altri punti.

### 10.1.2 Uso estensivo dell'operazione AES Scrambling

Se il vostro modello di SmartKey la supporta, l'operazione *AESScramble* dovrebbe essere utilizzata per controllare la presenza di una SmartKey reale.

E' necessario nascondere nella vostra applicazione una delle venti chiavi AES impostate tramite il comando *AESSet* e verificare periodicamente ed in differenti punti dell'applicazione la presenza di una SmartKey tramite l'operazione *AESScramble*.

### 10.1.3 Uso estensivo dell'operazione Scrambling

L'operazione *Scrambling* va utilizzata per controllare la presenza di una reale SmartKey.

Il controllo deve essere eseguito calcolando in precedenza un set di coppie di input e di stringhe di output dell'operazione *Scrambling.* Quando l'operazione è avviata, alcune coppie devono essere comparate con il risultato della stessa operazione su SmartKey attuale.

Durante la fase esecutiva occorre scegliere la coppia da controllare utilizzando una combinazione sia di elementi casuali sia deterministici. Per esempio, la scelta dovrebbe dipendere da: un valore casuale, il tempo attuale del sistema, il punto di esecuzione del vostro programma e qualsiasi altra variabile che possa essere diversa da un'applicazione all'altra.

**Esempio**

Supponiamo che voi abbiate identificato tre punti importanti dell'esecuzione, nei quali desiderate controllare la presenza di SmartKey: l'avvio del programma, la funzione di salvataggio e la funzione di stampa. Inoltre, desiderate che il controllo sia eseguito su base mensile, in modo da avere 12 diversi inserimenti di tempo. Infine, desiderate 100 diversi controlli casuali.

È quindi necessaria una tabella di 3 x 12 x 100 = 3600 coppie.

### 10.1.4 Nascondere Label e Password

Le stringhe *Label* e *Password* non dovrebbero essere conservate come semplice testo nella vostra applicazione. Altrimenti, una semplice analisi dei binari risultanti potrebbe rivelare queste informazioni.

Un buon approccio potrebbe essere generare una stringa casuale e calcolare lo XOR dell'informazione originaria e il suo valore casuale. Quest'informazione originaria potrà essere ricostruita in fase di esecuzione con un altro XOR con la stringa casuale originaria e il risultato precedente.

Queste informazioni possono essere conservate in molti punti del vostro programma ed essere comparate durante l'esecuzione del programma stesso.

**Esempio**

Con questi step di pre-calcolo:

```
LABEL = "SMARTKEY"
RANDOM = "01234567"
CRYPT = LABEL XOR RANDOM
```

Nella vostra sorgente potete fare:

```
CRYPT = "????????" (previously computed)
RANDOM = "01234567"
LABEL = CRYPT XOR RANDOM
SmartKeyCheckWithLabel(LABEL)
```

### 10.1.5 Utilizzare la versione .OBJ dei driver

Se è disponibile, è sempre meglio preferire la versione del driver.OBJ rispetto a quella DLL.

Il driver .DLL espone un punto d'inserimento semplice e noto. Con questo punto d'inserimento, monitorare e filtrare tutte le chiamate del driver eseguite dalla vostra applicazione diventa semplice.

### 10.1.6 CheckSum dei vostri eseguibili e delle DLL

È possibile calcolare e controllare il CRC/CheckSum della vostra applicazione e delle DLL. Si tratta di una fase molto importante se volete utilizzare uno dei driver .DLL di SmartKey. In questo modo, voi avete la certezza che il vostro programma sta utilizzando la DLL originale e non una versione fasulla.

Non bisogna utilizzare un semplice algoritmo CRC. Per esempio, il CRC32 di un file può essere modificato con un valore arbitrario cambiando solo 3/4 dei byte del file. Un funzione crittografica di hash come MD5 è sicuramente meglio.

### 10.1.7 Non arrestare immediatamente l'esecuzione se non è trovata la chiave

Se il controllo di SmartKey è negativo, il comportamento corretto è non arrestare improvvisamente l'esecuzione del programma, ma ritardare il termine su un'altra regione del vostro codice. Questo evita che il punto di controllo della chiave sia esposto.

**Esempio**

Questo esempio utilizza la variabile *KeyPresent* per conservare il risultato del controllo di SmartKey. È importante notare che la variabile è accessibile solo quando è rilevata la presenza della chiave. Questo evita in parte che sia utilizzata l'opzione debugging in grado di controllare qualsiasi accesso ad una variabile.

```
variable KeyPresent = False;

DoSomething();

if (SmartKeyPresent())
      KeyPresent = True;
```

```
	DoSomethingOther();

	If (KeyPresent == False)
		Abort();
```

Tutti gli accessi alla variabile *KeyPresent* devono essere eseguiti a diversi livelli delle chiamate di funzione.

### *10.2 Esempi d'implementazione*

Questo capitolo contiene alcuni esempi d'implementazione C delle linee di guida descritte in precedenza. Inoltre, essi si possono trovare nell'archivio *SmartKeyProtectionGuidelinesExample.zip.*

Tutti gli esempi suppongono che si lavori con una SmartKey Demo con *Label* "SMARTKEY" e *Password* "EUTRON" di default.

#### 10.2.1 Esempio 1 – Uso Base

Questo esempio mostra l'uso base di SmartKey. Lo scopo del programma è inizializzare le variabili per SmartKey e controllare che la chiave sia effettivamente presente. L'esempio NON DEVE ESSERE UTILIZZATO IN UN PROGRAMMA REALE, perché la *label* e la *password* sono inserite nel codice senza usare nessuna tecnica di protezione e dall'analisi del file eseguibile si potrebbe risalire ad esse.

```
#include "skeylink.h"

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {
	KEY_NET k;

	memset(&k,0,sizeof(k));

	strncpy(k.label,"SMARTKEY",LABEL_LENGTH);
	strncpy(k.password,"EUTRON",PASSWORD_LENGTH);

	/* Open */
	k.net_command = NET_KEY_OPEN;

	smartlink(&k); /* Chiamata a sistema */
	if (k.status != ST_OK) {
		printf("Error in NET_KEY_OPEN\n");
		exit(EXIT_FAILURE);
	}

	/* Close */
	k.net_command = NET_KEY_CLOSE;
	smartlink(&k);
	if (k.status != ST_OK) {
		printf("Error in NET_KEY_CLOSE\n");
		exit(EXIT_FAILURE);
	}

	return EXIT_SUCCESS;
}
```

### 10.2.2 Esempio 2 – Uso Base Scrambling

Questo esempio mostra l'uso base dell'operazione scrambling. Anche questo esempio NON DEVE ESSERE USATO IN UN PROGRAMMA REALE, perché i valori sono inseriti nel codice senza protezione.

```
#include "skeylink.h"

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* Scrambling input/output */
unsigned char scrambling_in[SCRAMBLE_LENGTH] = { 0x45, 0x34, 0x67, 0x23, 0xa5,
0x8f, 0x2c, 0x6d };
unsigned char scrambling_out[SCRAMBLE_LENGTH] = { 0x98, 0xab, 0x22, 0x24, 0xbb,
0xe6, 0x61, 0x8f };

int main() {
      KEY_NET k;

      /* Scrambling */
      k.net_command = NET_KEY_ACCESS;
      k.command = SCRAMBLING_MODE;
      memcpy(k.data,scrambling_in,SCRAMBLE_LENGTH);
      smartlink(&k);
      if (k.status != ST_OK) {
            printf("Error in SCRAMBLING_MODE\n");
            exit(EXIT_FAILURE);
      }

      if (memcmp(k.data,scrambling_out,SCRAMBLE_LENGTH)!=0) {
            printf("Wrong SCRAMBLING\n");
            exit(EXIT_FAILURE);
      }
      printf("Scramble ok\n");
}
```

### 10.2.3 Esempio 3/4 – Memorizzare e utilizzare una funzione C nella memoria di SmartKey

Questo esempio mostra come memorizzare e utilizzare un codice binario di una funzione C nella memoria di SmartKey. Vi sono alcune limitazioni:

- Le dimensioni della funzione devono essere inferiori o uguali a quelle della memoria di SmartKey.
- Non è possibile chiamare direttamente delle funzioni esterne, ma esse possono essere chiamate indirettamente passando come argomento un puntatore a funzione.
- Non è possibile usare direttamente delle variabili esterne, ma esse possono essere usate indirettamente passando come argomento un puntatore.
- Il vostro progetto deve essere collegato all'opzione /FIXED per evitare una nuova collocazione nel vostro codice.

#### Memorizzare la funzione

Questo esempio memorizza la funzione *my_func()* nella memoria di SmartKey.

```
#include "skeylink.h"

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
/* Function type */
typedef int my_func_t(int m, int n);

/* Buffer used to store the function */
char my_func_data[DATA_LENGTH + EXTENDED_DATA_LENGTH];

/* Function */
static int my_func(int m, int n) {
      return m * n;
}

/* Marker of the end of the function */
static int my_func_end(void) {
      return 0;
}

int main() {
      KEY_NET k;
      unsigned size;

      /* Uso preventivo delle funzioni per prevenire effetti collaterali
      dell'ottimizazione del compilatore.*/
      my_func(1,1);
      my_func_end();

      /* Compute the function size */
      size = (char*)my_func_end - (char*)my_func;

      printf("Function size %d\n", size);
      if (size > DATA_LENGTH + EXTENDED_DATA_LENGTH) {
            printf("Function size %d too big\n", size);
            exit(EXIT_FAILURE);
      }

      /* Copia della funzione sulla chiave */
      if (size > DATA_LENGTH) {
            memcpy(k.data,((char*)my_func),DATA_LENGTH);
            memcpy(k.ext_data,((char*)my_func) + DATA_LENGTH, size -
DATA_LENGTH);
      } else {
            memcpy(k.data,((char*)my_func),size);
      }

      /* Scrivi sulla SmartKey */
      k.net_command = NET_KEY_ACCESS;
      k.command = WRITING_MODE;
      smartlink(&k);
      if (k.status != ST_OK) {
            printf("Error in WRITING_MODE\n");
```

```
            exit(EXIT_FAILURE);
      }

      printf("Function written on the key\n");

      /* Close */
      k.net_command = NET_KEY_CLOSE;
      smartlink(&k);
      if (k.status != ST_OK) {
            printf("Error in NET_KEY_CLOSE\n");
            exit(EXIT_FAILURE);
      }

      return EXIT_SUCCESS;
}
```

**Uso della funzione**

Questo esempio legge la funzione *my_func()* dalla memoria di SmartKey e la esegue.

```
#include "skeylink.h"

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* Function type */
typedef int my_func_t(int m, int n);

/* Buffer usato per memorizzato la funzione */
char my_func_data[DATA_LENGTH + EXTENDED_DATA_LENGTH];

int main() {
      KEY_NET k;

      /* Leggi la funzione */
      k.net_command = NET_KEY_ACCESS;
      k.command = READING_MODE;
      smartlink(&k);
      if (k.status != ST_OK) {
            printf("Error in READING_MODE\n");
            exit(EXIT_FAILURE);
      }

      /* Copi i dati nel buffer */
      memcpy(my_func_data,k.data,DATA_LENGTH);
      memcpy(my_func_data + DATA_LENGTH,k.ext_data,EXTENDED_DATA_LENGTH);

      /* Set il puntatore a funzione */
      my_func_ptr = (my_func_t*)my_func_data;

      /* Chiama la funzione */
      result = my_func_ptr(2,3);
```

```
	if (result != 6) {
		printf("Error in function result\n");
		exit(EXIT_FAILURE);
	}

	printf("Result of the stored function %d\n",result);

	/* Close */
	k.net_command = NET_KEY_CLOSE;
	smartlink(&k);
	if (k.status != ST_OK) {
		printf("Error in NET_KEY_CLOSE\n");
		exit(EXIT_FAILURE);
	}

	return EXIT_SUCCESS;
}
```

### 10.2.4 Esempio 5 – Controllo del checksum DLL

Questo esempio mostra come calcolare e controllare un semplice checksum del file *smartlink.dll*.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {
	FILE* f;
	int c;
	unsigned checksum;

	/* Initialize the checksum */
	checksum = 0;

	/* Compute the checksum of the DLL */
	f = fopen("skeylink.dll","rb");
	if (!f) {
		printf("Error opening the DLL\n");
		exit(EXIT_FAILURE);
	}
	c = fgetc(f);
	while (c != EOF) {
		checksum += c;
		c = fgetc(f);
	}
	fclose(f);

	printf("DLL checksum %08X\n",checksum);

	if (checksum != 0x007ffcf1) {
		printf("Error invalid checksum\n");
		exit(EXIT_FAILURE);
	}
```

```
	return EXIT_SUCCESS;
}
```

### 10.2.5 Esempio 6 – Nascondere le informazioni Label e Password

Questo esempio mostra come nascondere le informazioni *Label* e *Password* utilizzando un semplice algoritmo di mascheramento. Questo algoritmo deve essere usato per evitare che dal file eseguibile si possa risalire a *Label* e *Password* rendendo vani tutte le protezioni di SmartKey.

```
#include "skeylink.h"

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* Hidden values */
static unsigned char hidden_label[LABEL_LENGTH] = {
	0x09, 0x2f, 0x2d, 0x2a, 0xd2, 0xdd, 0xed, 0xe5,
	0xd2, 0xea, 0x04, 0x20, 0x3e, 0x5e, 0x80, 0xa4

};

static unsigned char hidden_password[PASSWORD_LENGTH] = {
	0xe0, 0x91, 0xb1, 0x5a, 0x62, 0x1a, 0x7d, 0xa8,
	0xd5, 0x04, 0x35, 0x68, 0x9d, 0xd4, 0x0d, 0x48
};

int main() {
	KEY_NET k;
	unsigned i;

	memset(&k,0,sizeof(k));

	/* Calcola la label e la password corrette */
	for(i=0;i<LABEL_LENGTH;++i)
		k.label[i] = hidden_label[i] ^ (i*(i+0x7)+0x5a);
	for(i=0;i<PASSWORD_LENGTH;++i)
		k.password[i] = hidden_password[i] ^ (i*(i+0x1e)+0xa5);

	/* Open */
	k.net_command = NET_KEY_OPEN;
	smartlink(&k);
	if (k.status != ST_OK) {
		printf("Error in NET_KEY_OPEN\n");
		exit(EXIT_FAILURE);
	}

	printf("Net password %d\n",k.net_password);

	/* Close */
	k.net_command = NET_KEY_CLOSE;
	smartlink(&k);
```

```
	if (k.status != ST_OK) {
		printf("Error in NET_KEY_CLOSE\n");
		exit(EXIT_FAILURE);
	}

	return EXIT_SUCCESS;
}
```

### 10.2.6 Esempio 7 – Scrambling di dati riservati

Questo esempio mostra come nascondere dati riservati con l'operazione *Scrambling*.

Nell'esempio il valore di pi greco è memorizzato nel modo seguente.

```
#include "skeylink.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

/* Scrambled data */
unsigned char scrambled_data[SCRAMBLE_LENGTH] = {
	0x0c, 0xd8, 0xb3, 0xf6, 0x57, 0x6f, 0x4d, 0xe5
};

/* Scrambled input */
unsigned char scrambling_in[SCRAMBLE_LENGTH] = {
	0x45, 0x34, 0x67, 0x23, 0xa5, 0x8f, 0x2c, 0x6d
};

int main() {
	KEY_NET k;
	unsigned i;
	double pi;

	/* Scrambling */
	k.net_command = NET_KEY_ACCESS;
	k.command = SCRAMBLING_MODE;
	memcpy(k.data,scrambling_in,SCRAMBLE_LENGTH);
	smartlink(&k);
	if (k.status != ST_OK) {
		printf("Error in SCRAMBLING_MODE\n");
		exit(EXIT_FAILURE);
	}

	for(i=0;i<sizeof(pi);++i)
		((unsigned char*)&pi)[i] = k.data[i] ^ scrambled_data[i];

	printf("Pi greco is %g\n",pi);

	/* Close */
	k.net_command = NET_KEY_CLOSE;
	smartlink(&k);
	if (k.status != ST_OK) {
```

```
            printf("Error in NET_KEY_CLOSE\n");
            exit(EXIT_FAILURE);
      }
      return EXIT_SUCCESS;
}
```

### 10.2.7 Esempio 8/9–Generazione e utilizzo di una grande tabella Scrambling

Questo esempio mostra come generare e utilizzare una grande tabella scrambling. I valori di input dell'operazione scrambling sono calcolati in fase d'esecuzione utilizzando l'indice come inizializzazione di una semplice funzione casuale.

**Generazione del file "table.h"**

Questo esempio genera il file `table.h`

```
#include "skeylink.h"

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#define SCRAMBLE_MAX 1024

void scramble_in(unsigned char* dst, unsigned src) {
      unsigned i;
      for(i=0;i<SCRAMBLE_LENGTH;++i)
            dst[i] = (((src + 0x5a) >> i) * (i + 0x13)) ^ 0x3e;
}

int main() {
      KEY_NET k;
      unsigned i;
      FILE* f;

      srand(time(0));

      f = fopen("table.h","wt");
      if (!f) {
            printf("Error opening the file table.h\n");
            exit(EXIT_FAILURE);
      }

      fprintf(f,"void scramble_in(unsigned char* dst, unsigned src) {\n");
      fprintf(f,"\tunsigned i;\n");
      fprintf(f,"\tfor(i=0;i<SCRAMBLE_LENGTH;++i)\n");
      fprintf(f,"\t\tdst[i] = (((src + 0x5a) >> i) * (i + 0x13)) ^ 0x3e;\n");
      fprintf(f,"}\n\n");

      fprintf(f,"#define SCRAMBLE_MAX %d\n\n",SCRAMBLE_MAX);
      fprintf(f,"unsigned char SCRAMBLE[SCRAMBLE_MAX][SCRAMBLE_LENGTH] = {\n");

      for(i=0;i<SCRAMBLE_MAX;++i) {
            unsigned j;
```

```
            k.net_command = NET_KEY_ACCESS;
            k.command = SCRAMBLING_MODE;

            scramble_in(k.data,i);

            smartlink(&k);
            if (k.status != ST_OK) {
                  printf("Error in SCRAMBLING_MODE\n");
                  exit(EXIT_FAILURE);
            }

            fprintf(f,"{ ");
            for(j=0;j<SCRAMBLE_LENGTH;++j) {
                  unsigned v = k.data[j];
                  if (j)
                        fprintf(f,", ");
                  fprintf(f,"0x%02x",v);
            }
            fprintf(f," }");
            if (i+1!=SCRAMBLE_MAX)
                  fprintf(f,",");
            fprintf(f,"\n");
      }
      fprintf(f,"};\n");
      fclose(f);

      printf("Scrambling table written\n");

      /* Close */
      k.net_command = NET_KEY_CLOSE;
      smartlink(&k);
      if (k.status != ST_OK) {
            printf("Error in NET_KEY_CLOSE\n");
            exit(EXIT_FAILURE);
      }

      return EXIT_SUCCESS;
}
```

**Uso della tabella Scrambling**

Questo esempio usa il file generato per controllare la presenza della chiave SmartKey.

```
#include "skeylink.h"

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#include "table.h"

/* Return a random index in the table */
```

```
unsigned get_scrambling_index(void) {
      time_t t;
      struct tm* ptm;
      unsigned i;

      time(&t);
      ptm = localtime(&t);

      i = (rand() % (SCRAMBLE_MAX / 31)) * 31;
      i += ptm->tm_mday;
      i = i % SCRAMBLE_MAX;

      return i;
}

int main() {
      KEY_NET k;
      unsigned i;

      /* Initialized the random number generator */
      srand(time(0));

      /* Get the random index in the table */
      i = get_scrambling_index();
      printf("Scramble index %d\n",i);

      /* Do the scrambling */
      k.net_command = NET_KEY_ACCESS;
      k.command = SCRAMBLING_MODE;
      scramble_in(k.data,i);
      smartlink(&k);
      if (k.status != ST_OK) {
            printf("Error in SCRAMBLING_MODE\n");
            exit(EXIT_FAILURE);
      }

      /* Check the scramble */
      if (memcmp(k.data,SCRAMBLE[i],SCRAMBLE_LENGTH)!=0) {
            printf("Wrong SCRAMBLING\n");
            exit(EXIT_FAILURE);
      }
      printf("Scramble ok\n");

      /* Close */
      k.net_command = NET_KEY_CLOSE;
      smartlink(&k);
      if (k.status != ST_OK) {
            printf("Error in NET_KEY_CLOSE\n");
            exit(EXIT_FAILURE);
      }
```

```
	return EXIT_SUCCESS;
}
```

**Il file generato "table.h"**

Questo è il file generato `table.h`

```
void scramble_in(unsigned char* dst, unsigned src) {
	unsigned i;
	for(i=0;i<SCRAMBLE_LENGTH;++i)
		dst[i] = (((src + 0x5a) >> i) * (i + 0x13)) ^ 0x3e;
}

#define SCRAMBLE_MAX 1024

unsigned char SCRAMBLE[SCRAMBLE_MAX][SCRAMBLE_LENGTH] = {
{ 0xa2, 0x43, 0x2d, 0xdc, 0xf0, 0x49, 0x4b, 0x5c },
{ 0x00, 0x24, 0x9f, 0x6e, 0x51, 0x10, 0x9c, 0x1a },
…stripped…
{ 0x10, 0x0f, 0x5e, 0x8e, 0x5b, 0x44, 0x67, 0x11 },
{ 0xcd, 0x0a, 0x74, 0xed, 0x78, 0xc0, 0x0a, 0x97 }
};
```

### 10.2.8 Esempio 10 – Crittografare il codice

Inoltre, è possibile utilizzare SmartKey per crittografare il codice dei vostri file eseguibili o le librerie dinamiche (DLL), se questi sono sviluppati in Visual C. Un esempio completo ed estensivo si trova nell'archivio *SmartKeyEncryptionGuidelinesExample.zip*. Per ulteriori dettagli si rimanda al *leggimi.txt*.

### 10.2.9 Esempio 11 – Autenticazione AES

I seguenti sono due esempi per impostare le chiavi AES e per utilizzarle.

**Setkey**

Questo esempio imposta le chiavi AES. **Attenzione le chiavi AES possono essere impostate SOLO UNA VOLTA!**

```
#include "skeydrv.h"

#include <stdlib.h>
#include <stdio.h>
#include <windows.h>

/* Serial */
unsigned char serial[16] = {
0x39, 0x9d, 0x81, 0xd0, 0x6f, 0x78, 0x94, 0x41, 0xec, 0xfe, 0x71, 0xa1, 0x21,
0xd4, 0xe1, 0x6d,
};

/* 20 AES keys */
unsigned char aeskey[16*20] = {
0x27, 0xe1, 0x4e, 0xf4, 0x82, 0x3a, 0x1d, 0xa2, 0xbd, 0xee, 0xc7, 0xd2, 0x50,
0xe1, 0x37, 0x66,
…stripped…
0x57, 0x4b, 0x21, 0x94, 0x82, 0x38, 0x68, 0xf8, 0xf8, 0x54, 0x38, 0xa1, 0x6d,
0x05, 0x70, 0x39,
};

int main() {
```

```
	SKEY_DATA key;

	printf("SmartKey AES example\n");

	memset(&key, 0, sizeof(key));

	strncpy(key.label, "SMARTKEY", LABEL_LENGTH);
	key.command = LOCATING_MODE;
	msclink(&key);
	if (key.status != 0) {
		MessageBox(NULL, "SmartKey not found", "Error", MB_ICONERROR |
MB_SYSTEMMODAL);
		exit(1);
	}

	key.command = EXT_MODEL_READING_MODE;
	msclink(&key);
	if (key.status != 0) {
		MessageBox(NULL, "SmartKey not found", "Error", MB_ICONERROR |
MB_SYSTEMMODAL);
		exit(1);
	}
	if ((key.data[3] & 0x1) == 0) {
		MessageBox(NULL, "This SmartKey model doesn't support AES commands",
"Error", MB_ICONERROR | MB_SYSTEMMODAL);
		exit(1);
	}

	if (MessageBox(NULL, "Set the AES keys ? The operation cannot be undone
!", "Warning", MB_YESNO | MB_ICONWARNING | MB_SYSTEMMODAL) != IDYES) {
		exit(1);
	}

	memcpy(key.ext_data, serial, 16);
	memcpy(key.ext_data + 16, aeskey, 16*20);

	key.command = AES_SET_MODE;
	msclink(&key);
	if (key.status != 0) {
		MessageBox(NULL, "Error setting the AES keys and serial.\n\rPlease
note that you can set them ONLY ONE TIME!", "Error", MB_ICONERROR |
MB_SYSTEMMODAL);
		exit(1);
	}

	MessageBox(NULL, "AES keys setup correctly", "Information",
MB_ICONINFORMATION | MB_SYSTEMMODAL);

	return 0;
}
```

**Usekey**

Questo esempio utilizza le chiavi AES.

```
#include "skeydrv.h"
#include "aes.h"

#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <windows.h>

/* Serial */
unsigned char serial[16] = {
0x39, 0x9d, 0x81, 0xd0, 0x6f, 0x78, 0x94, 0x41, 0xec, 0xfe, 0x71, 0xa1, 0x21,
0xd4, 0xe1, 0x6d,
};

/* 20 AES keys */
unsigned char aeskey[16*20] = {
0x27, 0xe1, 0x4e, 0xf4, 0x82, 0x3a, 0x1d, 0xa2, 0xbd, 0xee, 0xc7, 0xd2, 0x50,
0xe1, 0x37, 0x66,
…stripped…
0x57, 0x4b, 0x21, 0x94, 0x82, 0x38, 0x68, 0xf8, 0xf8, 0x54, 0x38, 0xa1, 0x6d,
0x05, 0x70, 0x39,
};

int main() {
      SKEY_DATA key;
      aes_context aes;
      unsigned key_index;
      unsigned char result_buffer[16];
      unsigned char random_buffer[16];
      unsigned i;

      printf("SmartKey AES example\n");

      /* initialize the random number generator */
      srand(time(0));

      memset(&key, 0, sizeof(key));

      strncpy(key.label, "SMARTKEY", LABEL_LENGTH);
      key.command = LOCATING_MODE;
      msclink(&key);
      if (key.status != 0) {
            MessageBox(NULL, "SmartKey not found", "Error", MB_ICONERROR |
MB_SYSTEMMODAL);
            exit(1);
      }

      key.command = EXT_MODEL_READING_MODE;
```

```
	msclink(&key);
	if (key.status != 0) {
		MessageBox(NULL, "SmartKey not found", "Error", MB_ICONERROR |
MB_SYSTEMMODAL);
		exit(1);
	}
	if ((key.data[3] & 0x1) == 0) {
		MessageBox(NULL, "This SmartKey model doesn't support AES commands",
"Error", MB_ICONERROR | MB_SYSTEMMODAL);
		exit(1);
	}

	/* set the random value */
	for(i=0;i<16;++i)
		random_buffer[i] = rand() % 256;
	/* set the key number */
	key_index = rand() % 20;

	memcpy(key.data, random_buffer, 16);
	key.data[16] = key_index;
	key.command = AES_SCRAMBLE_MODE;
	msclink(&key);
	if (key.status != 0) {
		MessageBox(NULL, "Error using the AES key", "Error", MB_ICONERROR |
MB_SYSTEMMODAL);
		exit(1);
	}

	aes_set_key(aes, aeskey + key_index*16, 16);
	aes_decrypt(aes, key.data, result_buffer);
	for(i=0;i<16;++i)
		result_buffer[i] ^= random_buffer[i];

	if (memcmp(result_buffer, serial, 16) != 0) {
		MessageBox(NULL, "Wrong serial", "Error", MB_ICONERROR |
MB_SYSTEMMODAL);
		exit(1);
	}

	MessageBox(NULL, "Correct serial", "Information", MB_ICONINFORMATION |
MB_SYSTEMMODAL);

	return 0;
}
```

# 11 Protezione manuale in rete

*SmartKey NET* supporta i comandi standard del modello SP e un set di comandi asserviti alla rete: Open, ACCESS, CLOSE e USER NUMBER.

Tutte le funzioni spiegate nei paragrafi successivi sono state usate nel programma per Windows *smartdem.c* che si trova nelle directory *Sdk\Manual_Protection\Client_Windows_Libraries_And_Examples\GenericWin32Dll* e *Sdk\Manual_Protection\Client_Windows_Libraries_And_Examples\GenericWin32Obj*. Il programma può essere compilato con un qualsiasi compilatore C.

## *11.1 Open Mode*

La modalità Open è utilizzata dall'utente per attivare la comunicazione con SmartKey. L'operazione di Open deve essere effettuata prima d'ogni operazione di accesso alla memoria della chiave.

Il comando Open genera una speciale password, denominata *Net Password*, che deve essere utilizzata in tutti i successivi comandi.

Nel primo byte del campo data è riportato il tipo di protocollo usato nella connessione. Se si vuole prevenire l'utilizzo del protocollo LOCAL è sufficiente controllare questo byte. Generalmente questo è utile per forzare l'utilizzo del Server SmartKey per la sua gestione delle licenze.

Per utilizzare questo comando con la protezione Map è necessario specificare il codice di applicazione Map nel campo Data come descritto nel capitolo riguardante il Map.

Lo scambio di informazioni è così organizzato:

| Models | NET | |
|---|---|---|
| Input | NET_COMMAND | 'O' |
| | LABEL | Label |
| | PASSWORD | Password |
| Output | NET_PASS | Net Password |
| | DATA[0] | Tipo di protocollo di rete utilizzato nella connessione:<br>= 0, LOCAL<br>= 2, ANP<br>= 3, TCPIP |
| | STATUS | Status<br>==0 Successo<br>!=0 Error |

**Tabella 33** Scambio di informazioni per *Open Mode*.

## *11.2 Access mode*

La modalità ACCESS permette il vero e proprio accesso alla chiave, selezionata tramite la Label che dovrà *sempre* essere passata prima di effettuare ogni operazione standard di interrogazione della chiave. Questa modalità richiede la *NET-Password* che identifica il Client richiedente l'accesso e la Label che identifica la chiave a cui si vuole accedere.

Lo scambio di informazioni è così organizzato:

| Models | NET | |
|---|---|---|
| Input | NET_COMMAND | 'A' |
| | NET_PASS | Net Password |
| | COMMAND | SmartKey Command |
| | … | |
| Output | … | |
| | STATUS | Status<br>==0 Successo<br>!=0 Error |

**Tabella 34** Scambio di informazioni per *Access Mode*.

Per il resto si seguono le medesime modalità già analizzate nel caso di protezione manuale con applicazioni standalone.

### *11.3 User number mode*

Il comando consente di ottenere il numero di utenti connessi al server che utilizzano la chiave specificata con il campo Label.

Questo comando funziona solo se si sta utilizzando un protocollo di rete e non il procollo LOCAL. Se usato con il protocollo LOCAL è ritornato l'errore –2 (SYNT_ERR).

Per utilizzare questo comando con la protezione Map è necessario specificare il codice di applicazione Map nel campo DATA come descritto nel capitolo riguardante il Map.

Lo scambio di informazioni è così organizzato:

| | | |
|---|---|---|
| Models | NET | |
| Input | NET_COMMAND | 'A' |
| | NET_PASS | Net Password |
| | COMMAND | 'U' |
| Output | STATUS | Status<br>>=0 Number di utenti connessi<br><0 Error |

**Tabella 35** Scambio di informazioni per *User Number Mode*.

### *11.4 Close mode*

La modalità CLOSE permette la "chiusura" (richiesta di LOGOUT) della comunicazione con SmartKey da parte del programma connesso con una precedente chiamata di Open.

Se a causa di un errore nel software applicativo, il programma dovesse terminare senza aver prima eseguito una CLOSE della chiave aperta, possono verificarsi problemi connessi alla gestione del timeout del Client, che a volte richiedono un boot della macchina. È fondamentale seguire con attenzione la procedura di apertura e chiusura, per non precludere inutilmente l'utilizzo della chiave ad altri utenti.

Lo scambio di informazioni è così organizzato:

| | | |
|---|---|---|
| Models | NET | |
| Input | NET_COMMAND | 'C' |
| | NET_PASS | Net Password |
| Output | STATUS | Status<br>==0 Successo<br>!=0 Error |

**Tabella 36** Scambio di informazioni per *Close Mode*.

### *11.5 Close mode su timeout*

Per evitare di tener impegnate delle licenze in caso di terminazione anomala del programma (ad esempio perché è stato spento il PC), è implementata una funzione trasparente di refresh per **timeout**, ovvero la disconnessione automatica degli utenti che non hanno utilizzato la modalità CLOSE in modo corretto. La funzione di controllo del timeout è completamente trasparente all'utilizzatore ed allo sviluppatore ed è gestita dai software driver della chiave; non è quindi necessario effettuare periodiche operazioni d'accesso alla chiave per garantirsi il refresh del timeout.

### *11.6 Errori*

In aggiunta agli errori del driver standalone, il campo *status* può assumere uno dei seguenti valori:

| **Nome** | **Valore** | **Descrizione** |
|---|---|---|
| ST_NET_ERROR | -5 | Errore genrico nella comunicazione di rete. |
| ST_USER_ERROR | -8 | Massimo numero di utenti e licenze raggiunto. |
| ST_EXEC_ERROR | -16 | Massimo numero di esecuzioni raggiunto. |

| | | |
|---|---|---|
| ST_NET_PWD_ERR | -9 | La *netpassword* specificata non corrisponde a nessuna connessione aperta. |
| ST_INIT_ERROR | -11 | Errore generico durante l'inizializzazione della libreria. |
| ST_TOO_MANY_OPEN_KEY | -14 | Si è raggiunto il limite massimo di connessioni aperte. |
| ST_NET_CONF_ERROR | -21 | Errore generico nella configurazione della libreria. |
| ST_NET_ANP_INIT_ERROR | -22 | Errore nella inizializzazione del protocollo ANP. |
| ST_NET_TCPIP_INIT_ERROR | -23 | Errore nella inizializzazione del protocollo TCPIP. |
| ST_NET_LOCAL_INIT_ERROR | -25 | Errore nella inizializzazione del protocollo Locale. |
| ST_NET_KEY_NOT_MAP | -26 | Tentativo di apertura di una connessione MAP senza l'utilizzo di un dispositivo configurato MAP. |

### *11.7 Driver Standalone o Multilan?*

È stato illustrato nel capitolo sulla protezione manuale l'utilizzo dei driver relativi ad applicativi standalone. Nel capitolo attuale invece è stata discussa la tecnologia Multilan, che permette di utilizzare driver sia per ambiente standalone che di rete. Ecco alcuni consigli su quali dei due tipi di driver è opportuno utilizzare:

- **Applicativi sicuramente solo standalone**: utilizzare un driver di tipo standalone, perché la dimensione è inferiore (circa la metà dell'analogo MultiLan, perché non include il supporto di rete).
- **Applicativi standalone e rete**: utilizzare un driver di tipo MultiLan, perché rende indipendente l'applicativo dalla modalità di funzionamento (standalone o rete).

# 12 Protezione di più applicativi con SmartKey

In ambiente Lan è possibile utilizzare un'unica *SmartKey NET* per la protezione di più applicativi software. La tecnologia utilizzata è denominata **Map** – **Multi Application Protection** e consente di:

- Proteggere **più di un applicativo** in ambiente standalone o rete. Nel caso di rete locale è anche possibile definire per ciascun applicativo protetto un differente numero di licenze abilitate.
- Limitare il **numero d'esecuzioni** di ciascuno dei programmi protetti. Questa caratteristica può rivelarsi utile qualora si debbano creare versioni *demo* del software o si desideri adottare la politica del software a *noleggio*. Alla scadenza del numero d'esecuzioni preimpostato in un contatore (decrementato ad ogni startup del programma), non è più permessa la partenza del programma.

### *12.1 Le modalità operative*

L'implementazione di Map prevede solamente alcune lievi differenze rispetto alla metodologia operativa già descritta e cioè:

- La limitazione opzionale del numero d'esecuzioni e, nel caso di *SmartKey NET*, la programmazione del numero di licenze per ogni singolo applicativo
- Una differente modalità di chiamata al driver di gestione per quanto riguarda le operazioni di Open Mode e USER NUMBER mode

### *12.2 Programmazione del numero di licenze e del numero di esecuzioni*

Il limite massimo di applicativi proteggibili con la medesima chiave è 116. Il numero massimo di esecuzioni relativamente a ciascun applicativo è compreso tra 1 e 65.535; il valore -1 inattiva tale controllo e quindi non vi è limitazione nel numero di esecuzioni.

Il numero di licenze e di esecuzioni per ciascun programma è impostabile scrivendo il valore nei primi byte del campo Secure Data ed eventualmente nel campo Extended Data della chiave, seguendo il seguente schema:

| **Posizione** | **Valore** | **Significato** |
|---|---|---|
| 0 | 'M' (4D hex) | Codice fisso di identificazione Map |
| 1 | 'A' (41 hex) | Codice fisso di identificazione Map |
| 2 | 0 - 50 | Numero massimo di utenti per l'applicativo 1 |
| 3-4 | 0 - 65535 | Numero massimo di esecuzioni per l'applicativo 1 |
| 5 | 0 - 50 | Numero massimo di utenti per l'applicativo 2 |
| 6-7 | 0 - 65535 | Numero massimo di esecuzioni per l'applicativo 2 |
| 8 | 0 - 50 | Numero massimo di utenti per l'applicativo 3 |
| 9-10 | 0 - 65535 | Numero massimo di esecuzioni per l'applicativo 3 |
| … | … | |
| 62 | 0 - 50 | Numero massimo di utenti per l'applicativo 20 |
| 63 -0 (ext_data) | 0 - 65535 | Numero massimo di esecuzioni per l'applicativo 20 |
| 1 (ext_data) | 0 - 50 | Numero massimo di utenti per l'applicativo 21 |
| 2-3 (ext_data) | 0 - 65535 | Numero massimo di esecuzioni per l'applicativo 21 |
| … | … | |
| 285 (ext_data) | 0 - 50 | Numero massimo di utenti per l'applicativo 116 |
| 287-288 (ext_data) | 0 - 65535 | Numero massimo di esecuzioni per l'applicativo 116 |
| … | … | |

**Tabella 37** Impostazioni per gestione licenze.

Ad esempio, qualora si volessero proteggere 3 applicativi differenti con i seguenti numeri di licenze e numero d'esecuzioni:

- Primo programma: 23 licenze Esecuzioni illimitate.
- Secondo programma: 4 licenze 4000 esecuzioni.
- Terzo programma: 12 licenze 100 esecuzioni.

È necessario impostare la memoria della chiave come segue:

| **Posizione** | **Valore** | **Significato** |
|---|---|---|
| 0 | 'M' (4D hex) | Codice fisso di identificazione Map |
| 1 | 'A' (41 hex) | Codice fisso di identificazione Map |
| 2 | 23 (17 hex) | Massimo 23 utenti per l'applicativo 1 |
| 3-4 | 65535 (FFFF hex) | Nessuna limitazione al n. di esecuzioni del programma |
| 5 | 4 (04 hex) | Massimo 4 utenti per l'applicativo 2 |
| 6-7 | 4000 (0FA0 hex) | Limitazione attiva per 4000 esecuzioni |
| 8 | 12 (0C hex) | Massimo 12 utenti per l'applicativo 3 |
| 9-10 | 100 (64 hex) | Limitazione attiva per 100 esecuzioni |
| … | … | |

**Tabella 38** Impostazione per protezione applicativi.

### *12.3 La protezione automatica Map*

Se la protezione automatica tramite l'utility *GSS* è effettuata con una chiave di rete, un apposito pulsante consente di associare ad ogni applicativo un codice, che sarà utilizzato per risalire al numero di licenze per il quale il programma è abilitato nonché l'eventuale limitazione del numero d'esecuzioni del programma (si veda il capitolo 8).

### *12.4 La protezione manuale Map*

Siccome si ha la possibilità di proteggere più applicativi, limitatamente ai comandi Open e User Number è necessario specificare a quale programma si fa riferimento (identificato da un numero compreso tra 1 e 116). A questo scopo nel passaggio dei parametri relativo ad ognuno dei due comandi, il campo Data della relativa struttura dati deve essere inizializzato come segue:

| **Offset** | **Valore** | **Significato** |
|---|---|---|
| 0 | 'M' (4D hex) | Codice fisso di identificazione Map |
| 1 | 'A' (41 hex) | Codice fisso di identificazione Map |
| 2 | 1 - 116 | Numero di riferimento dell'applicativo |

**Tabella 39** Impostazione codice applicativo per comandi *Open/User Number Mode*.

Per il comando Open lo scambio di informazione è organizzato come segue:

| Models | NET | |
|---|---|---|
| Input | NET_COMMAND | 'O' |
| | LABEL | Label |
| | PASSWORD | Password |
| | DATA[0] | 'M' (4D hex) |
| | DATA[1] | 'A' (41 hex) |
| | DATA[2] | 1 – 116, Codice identificativo dell'applicativo della licenza da utilizzare |
| Output | NET_PASS | Net Password |
| | DATA[0] | Tipo di protocollo di rete utilizzato nella connessione:<br>= 0, LOCAL<br>= 2, ANP |

| | | |
|---|---|---|
| | | = 3, TCPIP |
| | STATUS | Status<br>==0 Successo<br>!=0 Error |

**Tabella 40** Scambio di informazioni per Open Mode con Map.

Nel primo byte del campo Data è riportato il tipo di protocollo usato nella connessione. Se si vuole prevenire l'utilizzo del protocollo Local è sufficiente controllare questo byte. Generalmente questo è utile per forzare l'utilizzo del Server SmartKey per la sua gestione delle licenze.

Se il comando Open è effettuato senza il protocollo Map, la chiave SmartKey è comunque aperta, ma senza il controllo di licenze permettendo l'esecuzione di un numero illimitato di applicazioni.

Per il comando User Number lo scambio di informazione è organizzato come segue:

| Models | NET | |
|---|---|---|
| Input | NET_COMMAND | 'A' |
| | NET_PASS | Net Password |
| | COMMAND | 'U' |
| | DATA[0] | 'M' (4D hex) |
| | DATA[1] | 'A' (41 hex) |
| | DATA[2] | 1 – 116, Codice identificativo dell'applicativo per il quale si vuole sapere il numero di licenze utilizzate |
| Output | STATUS | Status<br>>=0 Numero di licenze in uso<br><0 Error |

**Tabella 41** Scambio di informazioni per *User Number Mode*.

### 12.4.1 Open Mode Map: un esempio

Parametri da trasferire per eseguire un'operazione di Open utilizzando la protezione multipla Map su una chiave di rete; è richiesta l'apertura dell'applicativo 2. Se la chiave è stata programmata come nella tabella di programmazione precedente, non oltre 4 utenti contemporanei potranno utilizzare l'applicativo.

| | | |
|---|---|---|
| NET_COMMAND | `4F 00` | Open (`"O"`) |
| LABEL | `53 4D 41 52 54 4B 45 59`<br>`00 00 00 00 00 00 00 00` | Label (`"SMARTKEY"`) |
| PASSWORD | `45 55 54 52 4F 4E 00 00`<br>`00 00 00 00 00 00 00 00` | Password ("`EUTRON`") |
| DATA | `4D 41 02` | Map, 2° applicazione |

**Tabella 42** Esempio impostazione *Open Mode Map*.

# 13 Installazione di SmartKey

L'installazione dei driver SmartKey è eseguita dall'applicazione *SmartKey Driver Installer* (SDI). *SDI* permette di installare e disinstallare tutti i driver necessari al corretto funzionamento di SmartKey: i driver per *SmartKey Parallela***,** quelli per *SmartKey USB* e quelli per *Global Security System* (GSS), il programma per la protezione automatica.

Si ricorda che se si sta utilizzando una *SmartKey 3 USB DL* (Driver Less) non è necessario installare nessun driver.

Inoltre, è possibile integrare nel pacchetto d'installazione del proprio programma le procedure d'installazione di SmartKey grazie alle librerie fornite nel kit di sviluppo fornito. Le funzioni possono essere integrate con i principali software che generano pacchetti di distribuzione come *InstallShield.*

## *13.1 Avvertenze per l'installazione*

*SmartKey USB* può essere inserita e tolta mentre il computer è funzionante (come previsto dallo standard USB). *SmartKey Parallela* deve essere inserita prima di accendere il computer e la stampante collegata in cascata. In caso contrario, sia *SmartKey Parallela* sia la stampante potrebbero non funzionare correttamente.

## *13.2 Opzioni di SmartKey Driver Installer (SDI)*

SDI mette a disposizione tre opzioni per i tre tipi d'installazione e disinstallazione, *SmartKey Parallela***,** *SmartKey USB* e *Global Security System*. La figura 2 mostra l'interfaccia grafica per la *SmartKey USB* che è uguale a quella per *SmartKey Parallela* e *Global Security System*.

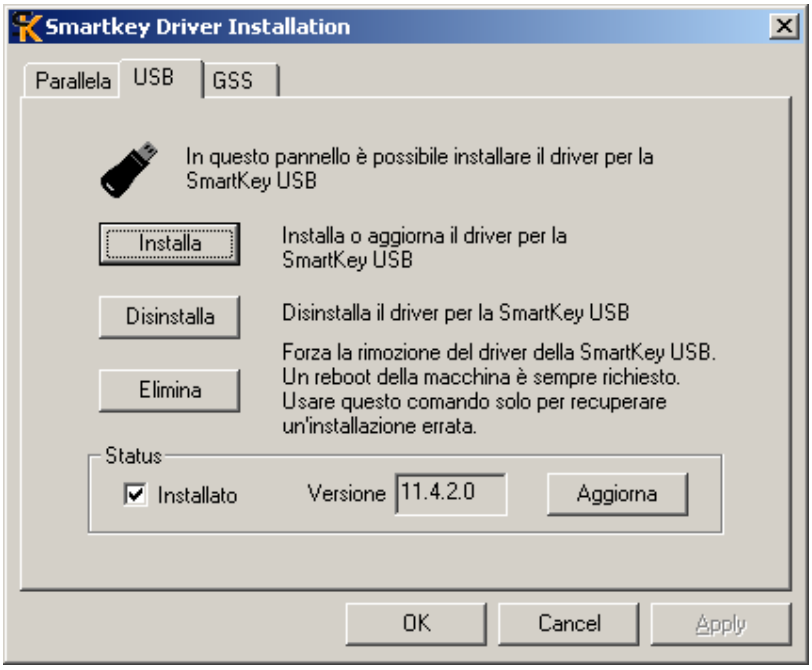


**Figura 2** L'interfaccia grafica di SDI per la *SmartKey USB.*

L'interfaccia grafica di SDI mette a disposizione tre opzioni

- *Installa*: installa i driver del tipo selezionato
- *Disintalla*: disinstalla i driver del tipo selezionato
- *Elimina*: disinstalla i driver senza controllare eventuali dipendenze.

Se l'installazione non va a buon fine, *SDI* apre un pannello in cui mostra dettagliatamente le operazioni che sono fallite e il numero dell'errore del sistema operativo. La figura 3 mostra un esempio di "pop up" dovuto al fallimento di un'installazione.

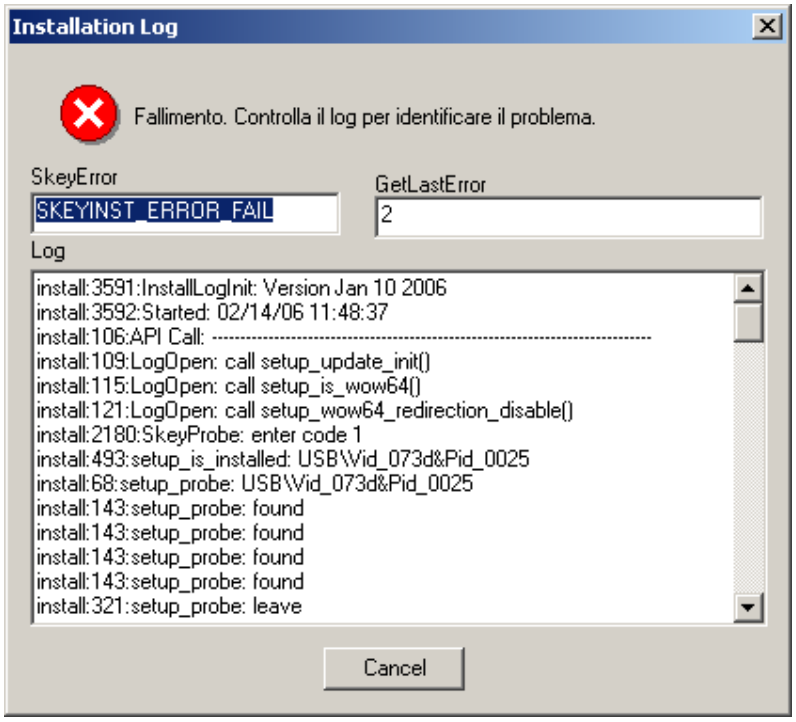


**Figura 3** Pop up con log e numero di errore del sistema operativo.

Normalmente le funzioni da utilizzare sono solo *Installa* e *Disinstalla*. La funzione *Elimina* deve essere usata solo se si presentano problemi nell'esecuzione delle altre funzioni. La funzione *Elimina* rimuove ogni riferimento dei driver SmartKey dal sistema, rendendo possibile il recupero di tutte quelle condizioni di errore che possono presentarsi durante l'installazione di un driver. L'uso di *Elimina* richiede sempre il reboot del sistema.

### *13.3 La libreria SDI*

Le funzioni usate per il programma SDI sono contenute nella libreria *skeyinst.dll*, messa a disposizione nel kit di sviluppo fornito. Più precisamente, i prototipi delle funzioni sono contenuti in *skeyinst.h* e le funzioni in *skeyinst.dll*. La libreria può essere usata sia per scrivere programmi sia per scrivere script di setup delle installazioni.

Le funzioni della libreria sono:

| | |
|---|---|
| *SkeyInstallUSB* | Installa i driver per *SmartKey USB* |
| *SkeyInstallPar* | Installa i driver per *SmartKey Parallela* |
| *SkeyInstallGSS2* | Installa i driver per *Global Security System* |
| *SkeyUnInstallUSB* | Disinstalla i driver per *SmartKey USB* |
| *SkeyUnInstallPar* | Disinstalla i driver per *SmartKey Parallela* |
| *SkeyUnInstallGSS2* | Disinstalla i driver per *Global Security System* |
| *SkeyForceUnInstallUSB* | Forza la rimozione dei driver per *SmartKey USB* |
| *SkeyForceUnInstallPar* | Forza la rimozione dei driver per *SmartKey Parallela* |
| *SkeyForceUnInstallGSS2* | Forza la rimozione dei driver per *Global Security System* |
| *SkeyLogFile* | Attiva la scrittura di un file di log di tutte le operazioni effettuate. |
| *SkeyGetLastError* | Ritorna l'ultimo codice d'errore. |

Le funzioni d'installazione *SkeyInstallUSB*, *SkeyInstallPar* e *SkeyInstallGSS2* controllano automaticamente lo stato dei driver prima di eseguire l'installazione.

Per la creazione di script d'installazione e disinstallazione è vivamente consigliato non usare le funzioni *SkeyForceUnInstallUSB*, *SkeyForceUnInstallPar* e *SkeyForceUnInstallGSS2*. Queste funzioni sono state implementate solo per risolvere situazioni anomale, e non per l'utilizzo normale.

È consigliato usare la funzione *SkeyLogFile*, perché grazie al file di log si può capire dove lo script d'installazione o il programma si è bloccato.

Le funzioni possono generare i seguenti codici d'errore:

| | |
|---|---|
| SKEYINST_OK | L'operazione si è conclusa correttamente. |
| SKEYINST_ERROR_WAIT | L'operazione non è stata eseguita, perché in quel momento il sistema operativo stava installando un altro componente. In questo caso è necessario chiedere all'utente di completare ogni altro processo d'installazione in corso e quindi ritentare l'operazione. |
| SKEYINST_WARNING_MUST_REBOOT | L'operazione si è conclusa correttamente ma occorre un riavvio del sistema per completare l'operazione. |
| SKEYINST_WARNING_MUST_INSERT | L'operazione si è conclusa correttamente ma occorre inserire il dispositivo USB per completare l'operazione. |
| SKEYINST_ERROR_FAIL | L'operazione è fallita a causa di un errore del sistema operativo. In questo caso si possono ottenere maggiori informazioni dal file di log. |

## *13.4 Installazione di SmartKey sotto Linux*

Per Linux sono disponibili due package per l'installazione e l'utilizzo delle chiavi SmartKey:

- user level usb – package *smartkey-linux-user-usb.tar.gz*
- user level lpt – package *smartkey-linux-user-lpt.tar.gz*

I package sono indipendenti dal kernel e dalla distribuzione utilizzata.

### 13.4.1 Linux user level usb

Il package user level usb non richiede l'installazione di nessun driver per l'accesso alla chiave ma dispone del file oggetto da compilare staticamente con l'applicazione per comunicare con il dispositivo. Questo package supporta solo la *SmartKey 3 USB* e la *SmartKey 3 USB DL*.

### 13.4.2 Linux user level lpt

Il package user level lpt non richiede l'installazione di nessun driver per l'accesso alla chiave ma dispone del file oggetto da compilare staticamente con l'applicazione per comunicare con il dispositivo. Questo package supporta solo la *SmartKey Parallela* ed è necessario avere i privilegi di root per l'accesso al dispositivo.

### 13.4.3 Utilizzo delle API per Linux

I prototipi delle API si trovano in *clink.h.* A seconda del package utilizzato è disponibile un file oggetto da compilare staticamente con la propria applicazione. Non è possibile fare il link dinamico. Consultare il file *LEGGIMI* allegato al package per maggiori informazioni.

## *13.5 Installazione di SmartKey sotto Mac OS X*

In Mac OS X sono supportate le chiavi *SmartKey 3 USB* e *SmartKey 3 USB DL*.

Non è richiesta l'installazione di nessun driver per l'accesso al dispositivo SmartKey; è disponibile una libreria statica e una dinamica per l'accesso al dispositivo SmartKey. Per l'installazione della libreria dinamica è disponibile un package per l'installazione automatica. Sono disponibili versioni delle librerie sia per applicazioni per Mac OS X PPC che per applicazioni Universal (Mac OS X PPC e Mac OS X Intel).

Il package da utilizzare è la Mac OS X disk image *smartkey-sdk-macosx.dmg* presente nella directory *nix/*.

### 13.5.1 Utilizzo delle API per Mac OS X

I prototipi delle API si trovano nel file *clink.h* per l'interfaccia standalone e *skeylink.h* per l'interfaccia multilan. E' disponibile una libreria statica per il link statico e una libreria dinamica distribuita come Framework per il link dinamico. Consultare il file LEGGIMI allegato al package per maggiori informazioni.

# 14 Installazione di SmartKey in rete

*SmartKey NET* può essere utilizzata con qualsiasi tipo di rete locale, grazie al software di supporto in dotazione. È inoltre possibile scegliere fra i seguenti protocolli di comunicazione:

- TCPIP: Protocollo standard per la trasmissione dati tra computer.
- ANP: Protocollo di trasmissioni dati tramite file condivisi. Si consiglia di usare questo protocollo solamente quando non si può usare il TCPIP. ANP è ancora disponibile solo per garantire la compatibilità con i vecchi programmi DOS.

A seconda del protocollo scelto la chiave sarà utilizzabile via remoto su PC con i seguenti sistemi operativi secondo lo schema:

| Client SmartKey | TCPIP | ANP |
|---|---|---|
| **Windows 9x/NT/2000/XP/2003/Vista** | ✓ | ✓ |
| **Mac OS X** | ✓ | |

**Tabella 43** Protocolli supportati da SmartKey installata su computer cliente.

La chiave sarà invece fisicamente installata su un PC che sarà denominato KeyServer. Il KeyServer potrà essere eventualmente il server di rete o un generico PC client. Il server può utilizzare uno dei seguenti sistemi operativi a seconda del protocollo utilizzato:

| Server SmartKey | TCPIP | ANP |
|---|---|---|
| **Windows 9x/NT/2000/XP/2003/Vista** | ✓ | ✓ |
| **Linux** | ✓ | |
| **Mac OS X** | ✓ | |

**Tabella 44** Tabella protocolli supportati da SmartKey installata su computer server.

### *14.1 Protocollo TCPIP*

Il protocollo TCP/IP può essere utilizzato su una qualsiasi rete che lo supporti. Il protocollo non pone vincoli sulla scelta del server SmartKey che può essere indifferentemente il server di rete od un generico client.

Per l'utilizzo del protocollo occorre:

- Individuare l'indirizzo di rete del PC su cui saranno installati la chiave e il server SmartKey.
- Verificare che il PC server SmartKey sia raggiungibile da tutti gli altri PC attraverso la rete.
- Identificare una porta TCP/UDP libera che sarà utilizzata per la comunicazione. Il numero della porta deve essere compreso tra 1024 e 49151 per evitare conflitti con altri protocolli. Per esempio 13527.
- Installare e configurare correttamente sia il server sia i client SmartKey per l'utilizzo del protocollo TCPIP specificando l'indirizzo del server SmartKey e la porta TCPIP da utilizzare.

Se possibile conviene sempre utilizzare il protocollo TCPIP rispetto agli altri protocolli.

### *14.2 Protocollo ANP*

Il protocollo ANP (Algorithmic Network Protection) sfrutta la presenza di filesystem condivisi per la comunicazione tra client e server. L'unica condizione per il suo funzionamento è il supporto del locking dei file da parte del filesystem condiviso.

Il protocollo ANP risulta quindi quello più generale data la comune disponibilità di filesystem condivisi in tutte le tipologie di rete e sistemi operativi. Per contro, l'utilizzo di file rende il protocollo inefficiente rispetto al TCPIP. Si consiglia di usare questo protocollo solamente quando non si può usare il TCPIP. ANP è ancora disponibile solo per garantire la compatibilità con i vecchi programmi DOS.

Il protocollo non pone vincoli sulla scelta del server SmartKey che può essere indifferentemente il server di rete od un client qualsiasi.

Per l'utilizzo del protocollo occorre:

- Individuare un disco di rete condiviso da tutti i PC client (può essere quello su cui gira normalmente l'applicativo di rete), ad esempio *N:*

- Sul disco di rete individuato creare una directory di lavoro per il programma server SmartKey. Il nome della directory è arbitrario, ad esempio:

  ```
  MD N:\ANP
  ```

- Rendere questa directory accessibile in lettura e in scrittura a tutti i computer della rete che vorranno utilizzare il programma protetto. È fondamentale che tutti i computer possano creare e modificare files all'interno di questa directory. Se non si è certi di questa possibilità, prima di procedere con i passi successivi, provare a copiare alcuni file all'interno di questa directory da tutti i computer client che dovranno utilizzare il programma protetto.
- Installare e configurare correttamente sia il server sia i client SmartKey per l'utilizzo del protocollo ANP e della directory condivisa creata.

Su alcuni sistemi operativi di rete, il locking a livello di file non è abilitato in modo automatico al momento dell'installazione: occorre pertanto fare riferimento alla documentazione per i dettagli sulla disponibilità ed attivazione del file locking.

Può accadere che il nome completo della directory condivisa che serve per il protocollo ANP possa assumere valori diversi tra il KeyServer e i vari PC client. Il parametro che è passato al caricamento del programma server identifica la directory condivisa, così com'è vista dal KeyServer che non necessariamente è vista dai PC client allo stesso modo e con lo stesso nome. Infatti il KeyServer potrebbe vedere la directory condivisa come *C:\MYDIR* mentre i PC client potrebbero vedere la stessa directory come *F:\MYDIR*. In questo caso la configurazione per il server SmartKey deve essere *C:\MYDIR* mentre la configurazione per il client SmartKey deve essere *F:\MYDIR*.

### *14.3 Installazione per Windows*

Il modo migliore per installare e configurare il server ed i client in Windows è utilizzare l'applicazione *SmartKey Configuration Central* descritta nel capitolo 14.4.

Per installare e configurare il server è anche possibile utilizzare l'utility a riga di comandi *askeyadd* disponibile nella directory *Sdk\Manual_Protection\Server_Programs\Service_Windows* per il servizio e *Sdk\Manual_Protection\Server_Programs\Executable_Windows* per la versione applicativo del server.

Per configurare i client è anche possibile utilizzare l'utility a riga di commando *cskeycfg* disponibile nella directory *Sdk\Manual_Protection\Client_Windows_Libraries_And_Examples\CSkeyCfg*. Per maggiori dettagli riferirsi alla documentazione inclusa nelle stesse directory.

### *14.4 Installazione su Linux e Mac OS X*

Per l'installazione del server su Linux e Mac OS X è sufficiente decomprimere il package corrispondente e con diritti di root digitare il comando *./skinstall* presente nel package.

Per la disinstallazione è sufficiente digitare il comando *./skuninstall* nel package del server.

Consultare il file *LEGGIMI* allegato al package del server per i dettagli sull'installazione e sulla configurazione del server.

| **Package** | **Descrizione** |
|---|---|
| `smartkey-server-linux.tar.gz` | Server SmartKey per Linux |
| `smartkey-server-macosx.tar.gz` | Server SmartKey per Mac OS X PowerPC/Intel |

**Tabella 45** SmartKey Server per Linux e Mac OS X.

# 15 SmartKey Configuration Central (SCC)

*SmartKey Configuration Central* (SCC) è un programma con interfaccia grafica che agevola la configurazione del server dove si trova la *SmartKey NET* e il client dove si trova l'applicativo da proteggere. SCC lavora in ambiente Windows e permette la configurazione del client e del server su reti con protocollo ANP e TCPIP.

## *15.1 Configurazione del server*

Configurare il server SmartKey significa configurare un programma che in background scambia automaticamente dati con l'applicativo protetto che si trova su un client. Lo scambio di dati client\server può avvenire attraverso il protocollo TCPIP o attraverso lo scambio di file che si trovano in una directory comune (protocollo ANP). Il programma server di Smartkey è disponibile in due versioni: *service* o *eseguibile*. La versione *service* si avvia automaticamente ad ogni boot del sistema operativo ed è installabile su sistemi con Windows NT, Windows 2000, Windows XP, Windows 2003 e Windows Vista.

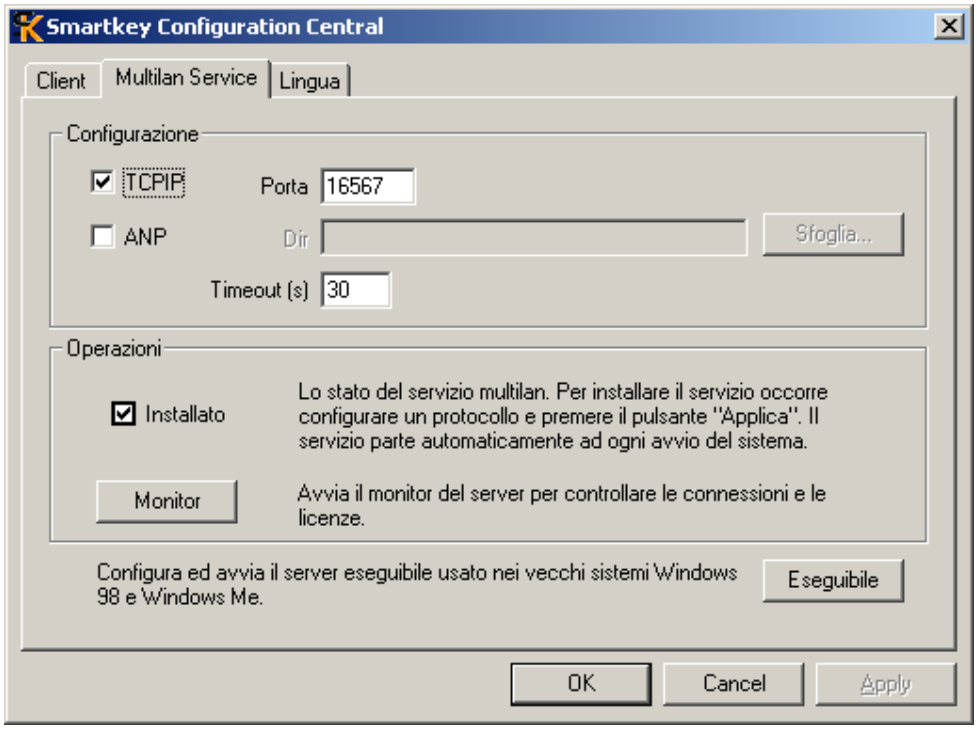


**Figura 4** *SmartKey Configuration Central* configurazione del service.

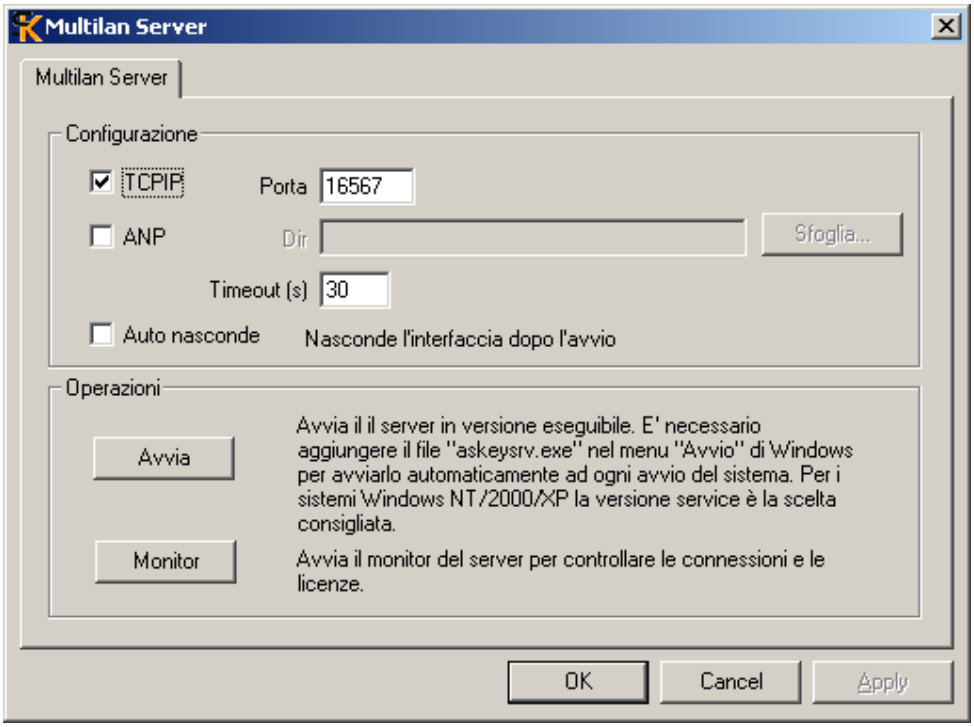


**Figura 5** *SmartKey Configuration Central* configurazione del server.

La figura 5 mostra i pannelli di SCC utilizzati per configurare ed avviare il programma server di Smartkey. Per il server "Service", configurare il protocollo desiderato e premere il pulsante *Applica*. Il *service* di Smartkey sarà automaticamente installato ed avviato. L'avvio del servizio avverrà automaticamente anche ad ogni riavvio della macchina.

Per la versione *eseguibile*, configurare il protocollo desiderato e premere il pulsante *Applica*. Premere poi *Avvia* per avviare il server eseguibile. Se si desidera fare eseguire automaticamente il programma server ad ogni riavvio della macchina, aggiungere il file *askeysrv.exe* nel menu *Programmi-Avvio* di Windows

Le opzioni disponibili per la configurazione sono:

- *TCPIP*: abilita il protocollo TCPIP.
- *TCPIP Porta*: numero della porta (TCP) usata per il protocollo TCPIP.
- *ANP*: abilita il protocollo ANP.
- *ANP Dir*: directory utilizzata dal protocollo ANP per lo scambio di file.
- *Timeout*: timeout in secondi prima che un client che non risponde sia disconnesso e la sua licenza venga rilasciata. Utile soprattutto nel caso si dovessero verificare problemi di rete.
- *Auto nasconde*: chiude automaticamente la finestra del server dopo la partenza.
- *Monitor:* visualizza il Monitor del server Smartkey per la verifica delle licenze disponibili e le connessioni dei client.

### *15.2 Configurazione del client*

La configurazione del computer su cui gira l'applicativo può essere fatta attraverso la finestra *Multilan Client* di SCC. Nella tabella centrale sono elencati tutti i protocolli con cui l'applicativo protetto deve cercare la SmartKey. La figura 6 mostra un esempio in cui il client è stato configurato in modo tale che gli applicativi Windows cerchino la SmartKey sul computer *187.11.10.74*, porta *16567* e tra quelle installate sulle porte locali.

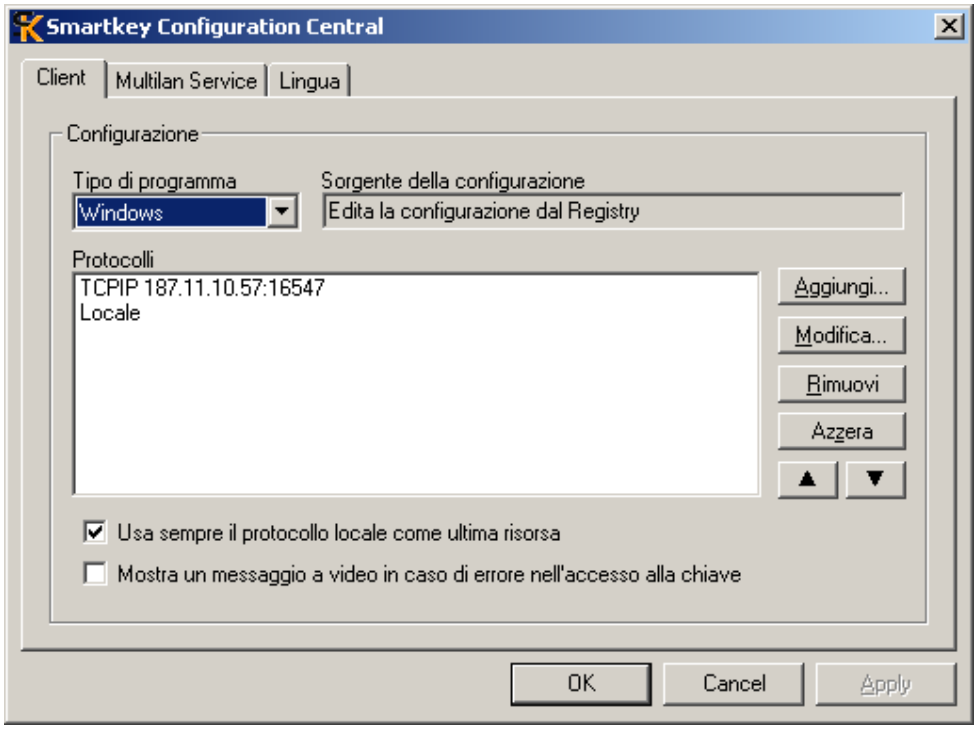


**Figura 6** *SmartKey Configuration Central*: modalità client.

La finestra *Multilan Client* mette a disposizione le seguenti opzioni:

- *Tipo di programma*: permette di selezionare che tipo d'applicazione proteggere: Windows, Windows 3.1 (16 bit) e DOS (16 bit).
- *Usa sempre il protocollo locale…*: se abilitato cerca sempre la SmartKey in locale anche se tale protocollo non è stato esplicitamente inserito.
- *Mostra un messaggio a video…*: se attivata visualizza in una finestra tutti gli errori che avvengono durante la comunicazione con la SmartKey. Questa opzione è molto utile per identificare problemi di comunicazione.

### 15.2.1 Fasi per la selezione e la configurazione

Quando si preme il pulsante *Aggiungi* è visualizzata una finestra come quella della figura 7 in cui è possibile scegliere uno tra i tre tipi di protocolli disponibili.


**Figura 7** Pannello per selezione protocollo.

Se si sceglie il protocollo Local non occorre inserire ulteriori parametri. Se si sceglie il protocollo ANP, si apre la finestra *Protocollo ANP*, come mostrato nella figura 8 e bisogna inserire il nome della directory condivisa da utilizzare per la comunicazione tra server e client. Se si sceglie il protocollo TCPIP, si apre la finestra *Protocollo TCPIP*, come mostrato nella figura 9 in cui bisogna inserire il nome simbolico o numerico del server e la sua porta.

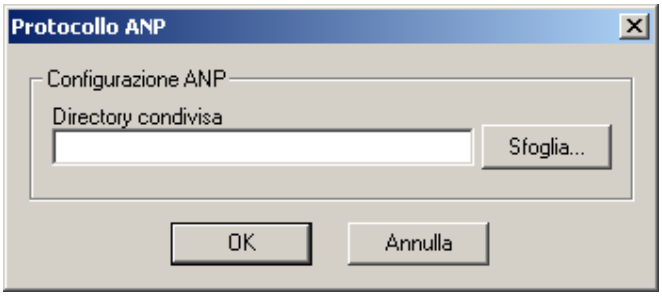


**Figura 8** Pannello per la configurazione del protocollo ANP.


**Figura 9** Pannello per la configurazione del protocollo TCP/IP.

# 16 SmartKey Programming Central (SPC)

*SmartKey Programming Central* (*SPC*) è in grado di programmare la SmartKey, cioè inserire i dati che determinano la *configurazione* della SmartKey. Secondo il differente modello di SmartKey utilizzato ci possono essere più o meno campi da impostare.

*SPC* permette anche di leggere la configurazione della SmartKey, modificarla, salvarla su file, ripristinarla da file e, infine, scriverla su SmartKey. La finestra di SPC ha due parti, come mostrato dalla figura 10. Nella parte sinistra si seleziona dall'elenco la SmartKey che si vuole configurare. Nella parte destra si seleziona uno dei 10 pannelli per la configurazione della SmartKey.

Se non si dovesse visualizzare la propria SmartKey, si può aggiornare l'elenco con il pulsante *Aggiorna*. Se non si dovesse ancora visualizzare la SmartKey significa che il sistema non la ha riconosciuta e bisogna controllare l'installazione dei driver.

Se la SmartKey selezionata è *Fixed*, e quindi non scrivibile, non sono visualizzati i pannelli *Programmazione*, *Fixing*, *Contenuto* e *Ripristina Default*. Il primo pannello da usare è *Identificazione* che permette l'identificazione della SmartKey e l'abilitazione all'accesso agli altri pannelli. Una volta fatta l'identificazione, l'accesso ai pannelli non è soggetto ad alcun vincolo.

## *16.1 Pannello identificazione*

L'identificazione della SmartKey avviene selezionando il pannello *Identificazione* ed inserendo i valori di *Label* e *Password*. I valori possono essere scritti sia in formato testo sia in formato esadecimale. Le due opzioni possono essere selezionate con i pulsanti *Ascii* e *Hex*. Se si usa una SmartKey che ha ancora i valori di default, si può evitare di inserire tali valori selezionando l'opzione *Utilizza i valori di default*. I valori di default sono *Label* = SMARTKEY e *Password* = EUTRON.

La figura 10 mostra l'esempio di un computer in cui è presente una SmartKey di tipo NET, i cui driver sono stati installati correttamente, ed è stato selezionato il pannello *Identificazione*.

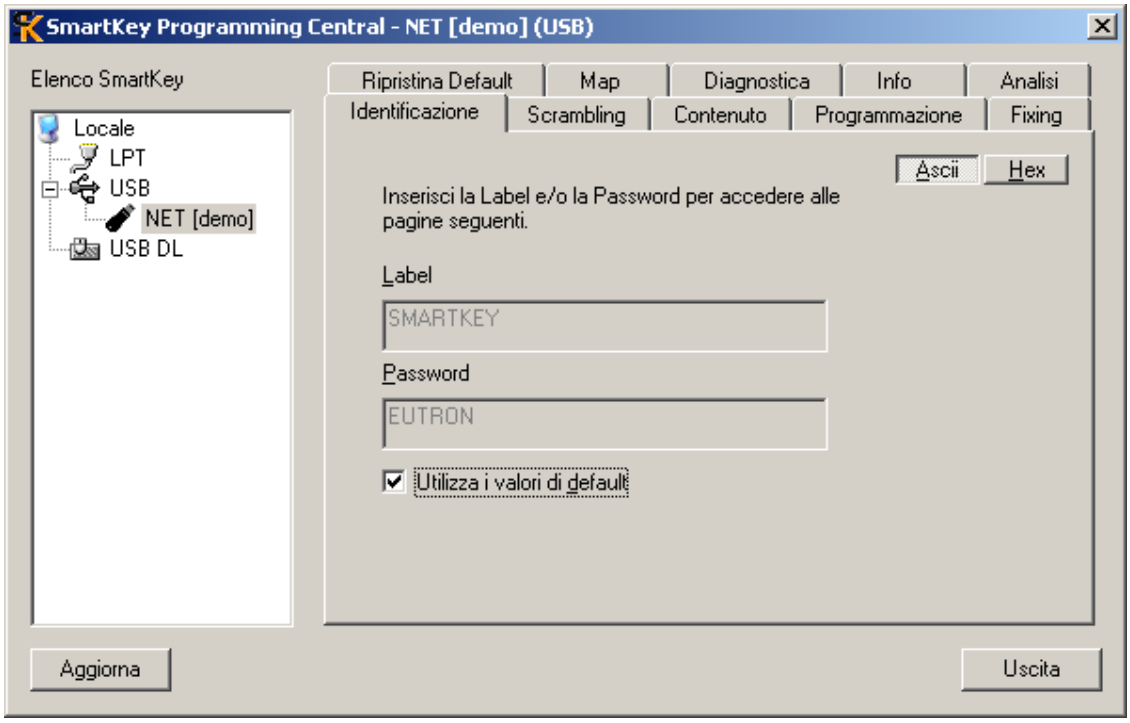


**Figura 10** Pannello per l'identificazione della SmartKey.

## *16.2 Pannello info*

Selezionando il pannello *Info* sono visualizzate alcune informazioni sulla SmartKey.

I valori visibili sono:

- Modello: tipo di SmartKey
- Dispositivo: tipo di dispositivo che SmartKey usa (USB o parallela).

- Memoria: la grandezza della memoria programmabile della SmartKey
- Seriale (dec): numero seriale in formato decimale
- Seriale (Hex): numero seriale in formato esadecimale
- Fail counter: il numero di volte che qualcuno ha inserito o la *label* o la *password* sbagliata.
- Fix: indica se la configurazione può essere modificata.
  - *Fixed*: non può essere modificata
  - *Not Fixed*: può essere modificata
- AES: indica quale livello di supporto AES la chiave possiede
  - *No:* la chiave non supporta le funzioni AES
  - *Basic:* la chiave supporta le funzioni AES_SET e AES_SCRAMBLE
- AES Keys: indica se le chiavi AES sono state impostate
  - *Yes*: le chiavi sono impostate (e non possono più essere modificate)
  - *No*: le chiavi non sono impostate
- Scrambling: mostra un esempio di scrambling: *In* è il valore d'ingresso e *Out* il valore d'uscita. I due valori (*In* e *Out*) identificano la chiave in modo univoco, dato che lo scrambling dipende dall'*id-Code* della chiave.

La figura 11 mostra un pannello di tipo info relativo ad una *SmartKey USB DL* di tipo *NET*.

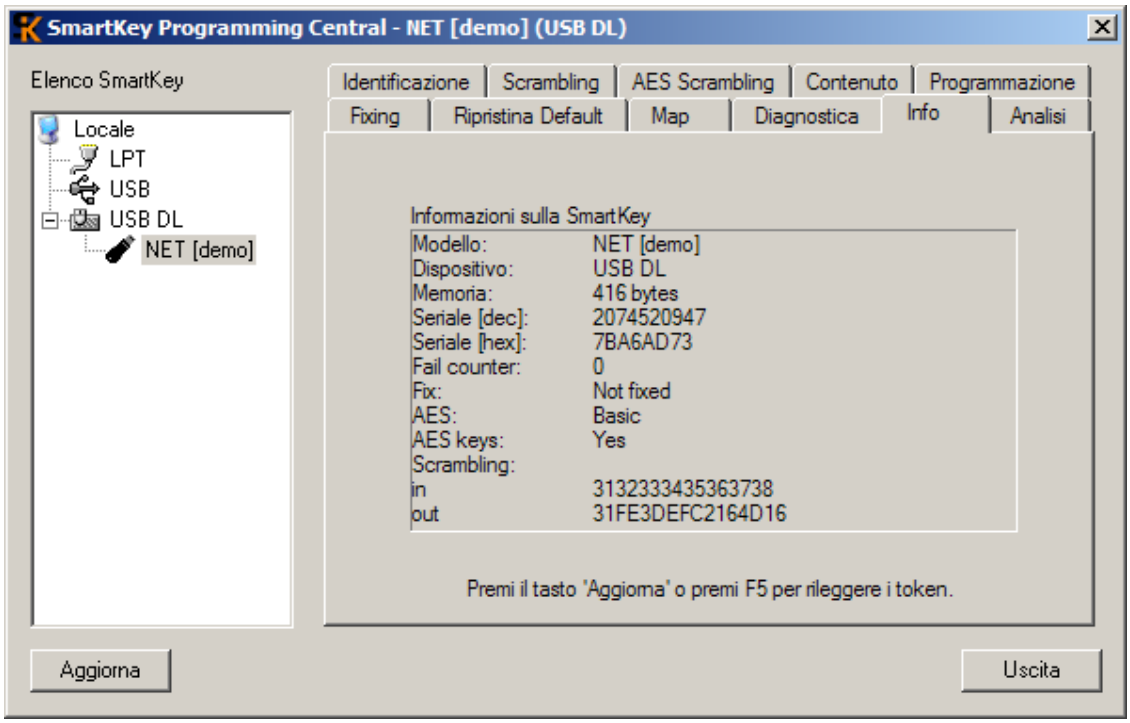


**Figura 11** Pannello Info della SmartKey.

### *16.3 Pannello ripristina default*

Il pannello *Ripristino Default* permette di ripristinare i valori di default e resettare la memoria di SmartKey. Per fare questo basta premere il pulsante *Ripristina Default. Ripristina Default*, però, funziona solo se SmartKey non è *Fixed*, cioè se è ancora riscrivibile. I valori di default sono:

- *Label*: SMARTKEY
- *Password*: EUTRON
- *Contenuto della memoria*: tutte le celle contengono il valore "\0" (00 Hex).
- *Flag di fissaggio SmartKey:* non Fixed

La figura 12 mostra il pannello *Ripristina Default* di una *SmartKey USB* di tipo *NET*.

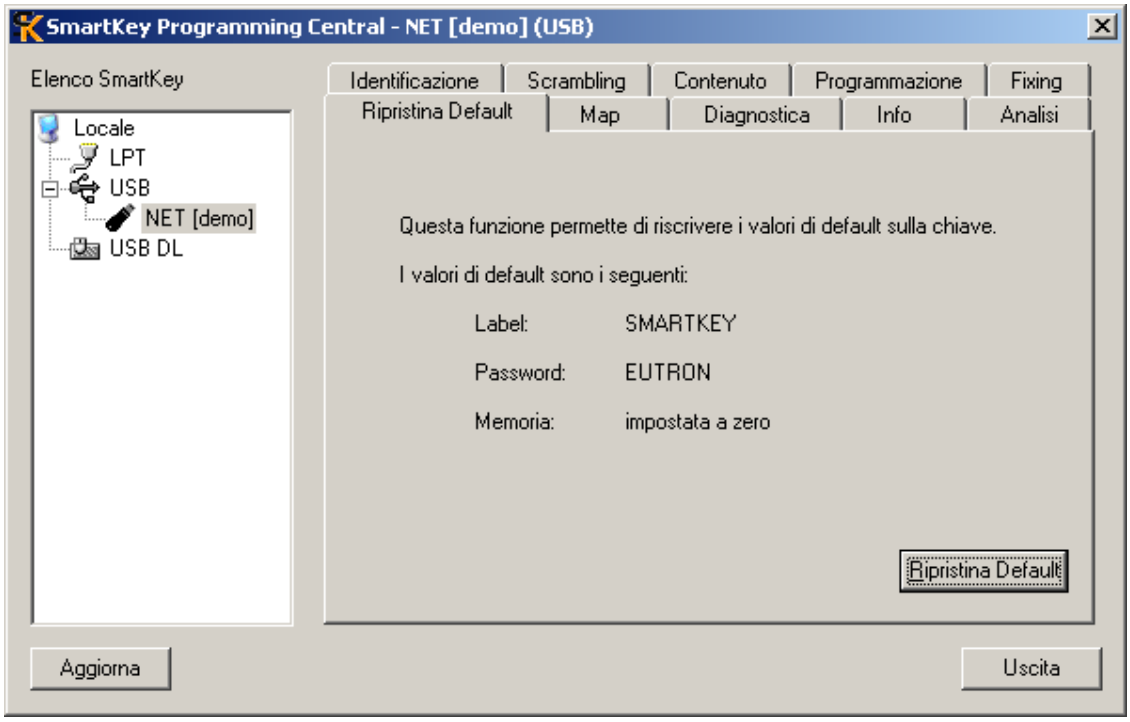


**Figura 12** Pannello *Ripristina Default* della SmartKey.

### *16.4 Pannello Map*

Il pannello *Map* permette di associare ad ogni applicazione il numero di esecuzioni possibili ed il numero di licenze.

La figura 13 mostra l'esempio di un pannello *Map* usato per la configurazione di una SmartKey di tipo *NET*.

Il pannello ha una tabella con tre colonne:

- *N. di applicazione*, numero identificativo dell'applicazione,
- *N. di esecuzioni*, il numero massimo di esecuzioni
- *N. di licenze*, il numero di licenze massime, ovvero il numero massimo di utenti che possono usare contemporaneamente il programma.

Se la SmartKey non è di tipo *NET*, il numero di colonne è limitato a due, *N.applicazione* e *N.esecuzione*. Con il pannello *Map* si possono fare due tipi di configurazione: quella per la gestione del numero di esecuzioni e quella per la gestione del numero di licenze, se si usa una *SmartKey NET*. Di seguito sono mostrate le operazioni da compiere per le due configurazioni:

**Configurazione SmartKey per la gestione del numero di esecuzioni**

- Inserire nella colonna *N. di esecuzione* il numero massimo di esecuzioni sulla riga corrispondente all'applicazione. Il valore può variare tra "-1" e "65.535". Il valore "-1" è interpretato da SmartKey come "Infinite esecuzioni".
- Non mettere nessun valore nella colonna *N. di licenze*. Questa colonna è visualizzata solo se si sta configurando una *SmartKey NET***.**
- Sistemate tutte le applicazioni, premere il pulsante *Scrivi* per scrivere su SmartKey i valori della configurazione

**Configurazione SmartKey per la gestione del numero di licenze. (Solo per *SmartKey NET*)**

- Inserire nella colonna *N. di esecuzione* il valore "-1" sulla riga corrispondente all'applicazione
- Inserire nella colonna *N. di licenze* numero di licenze sulla riga corrispondente. Il numero può variare tra "0" e "50".
- Sistemate tutte le applicazioni, premere il pulsante *Scrivi* per scrivere su SmartKey i valori della configurazione

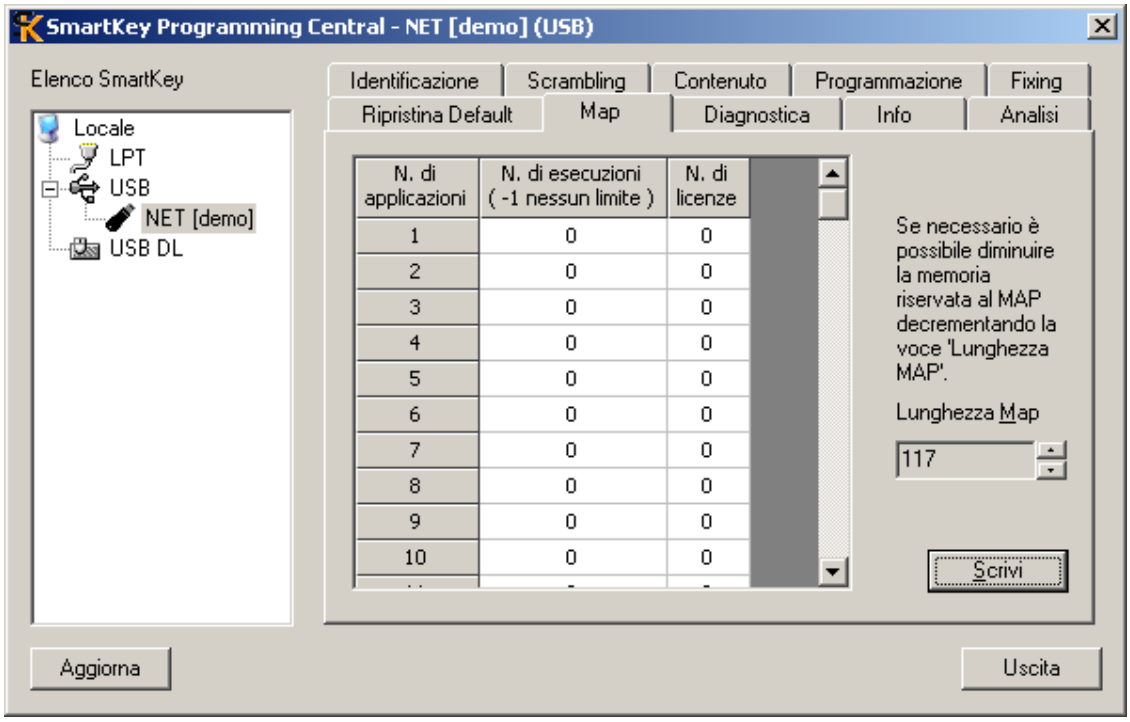


**Figura 13** Pannello *Map* di una *SmartKey USB* di tipo *NET*.

I dati necessari per la configurazione fatta con *Map*, sono inseriti nella memoria di SmartKey e, quindi il numero di applicazioni che si possono proteggere dipende dalla capienza della memoria. Occorrono due byte per abilitare il servizio *Map* e tre byte per ogni applicazione da proteggere. Quando si usa la protezione *Map* è meglio evitare di scrivere la memoria di SmartKey usando altri applicativi, perché si corre il rischio di sovrascrivere le celle di memoria destinate a *Map.* Le celle di memoria usate sono sequenziali a partire dalla cella 00. Se si volessero usare le celle di memoria libera, occorre tenere presente che la prima cella utile è la numero (2+(3 * numero delle applicazioni protette)).

Due semplici esempi sono:

- *Si vuole fare eseguire l'applicazione 1 solo 5 volte*:
  N.Applicazione: 1
  N.di esecuzione: 5
  N.di licenze: nessun valore (Lasciare pure il valore 0 di default)
- *Si vuole fare eseguire l'applicazione 1 al massimo da 10 persone contemporaneamente*:
  N.Applicazione: 1
  N.di esecuzione: -1
  N. di licenze: 10

(L'ultimo esempio vale solo per *SmartKey NET*)

### *16.5 Pannello scrambling*

Il pannello *Scrambling* permette di vedere l'output dell'algoritmo di scrambling dati valori d'ingresso conosciuti. I dati, sia quelli di ingresso sia quelli d'uscita, possono essere visualizzati in formato ascii o formato esadecimale. La selezione avviene con i pulsanti *Ascii* e *Hex*. Questa funzione è utile per i programmatori che vogliono inserire nel proprio codice le funzioni per lo scrambling.

La figura 14 mostra il pannello di *Scrambling* con i dati visualizzati in formato esadecimale.

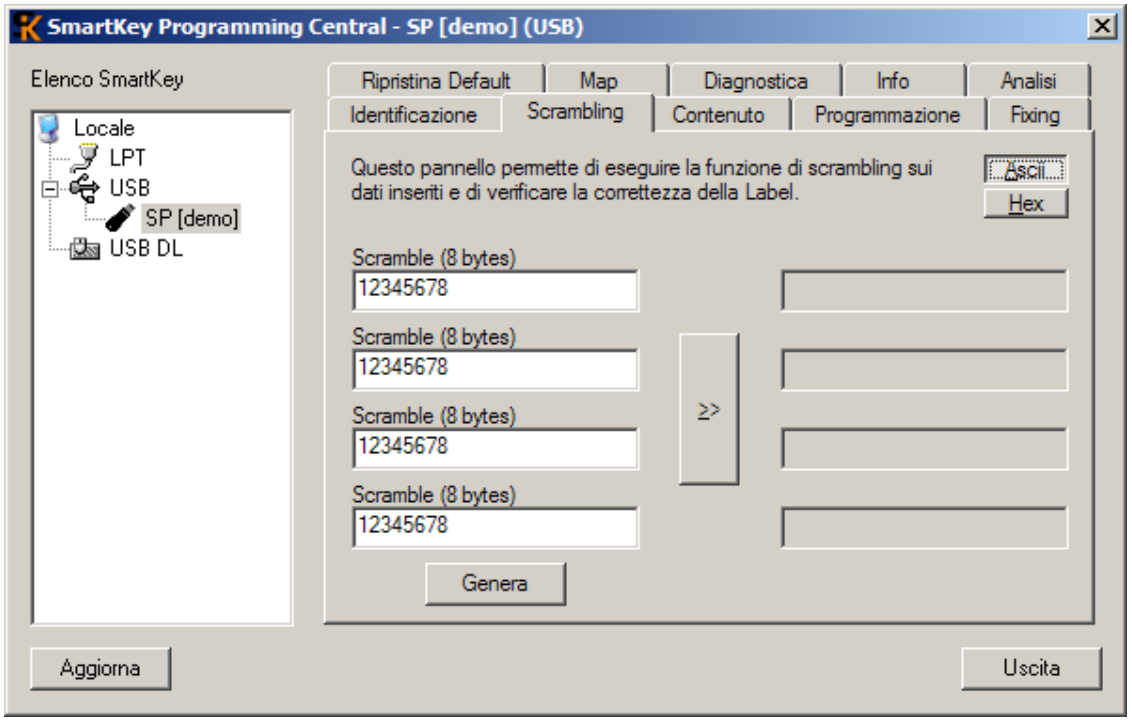


**Figura 14** Pannello *Scrambling.*

### *16.6 Pannello contenuto*

Il pannello *Contenuto* permette di leggere e scrivere la memoria interna della SmartKey**.** I valori possono essere visualizzati in formato decimale, selezionando il tasto *Dec*, in formato Ascii, selezionando il tasto *Ascii*, oppure in formato esadecimale, selezionando il tasto *Hex*. I valori da inserire vanno scritti direttamente nella tabella che si trova al centro. Ogni cella della tabella corrisponde ad una cella di memoria della SmartKey. La scrittura dei valori nella SmartKey avviene dopo che si preme il tasto *Scrivi.* Il pannello permette di visualizzare, inoltre, il numero seriale della SmartKey e il numero di accessi falliti. La figura 15 mostra un pannello *Contenuto* con i valori visualizzati in formato testo. Il numero della cella è determinato dalla somma del numero che si trova sulla colonna a sinistra e il numero che si trova in alto. Ad esempio, il valore “E” si trova nella cella 9 (8 + 1), il valore “U” nella cella 10 (8 +2)

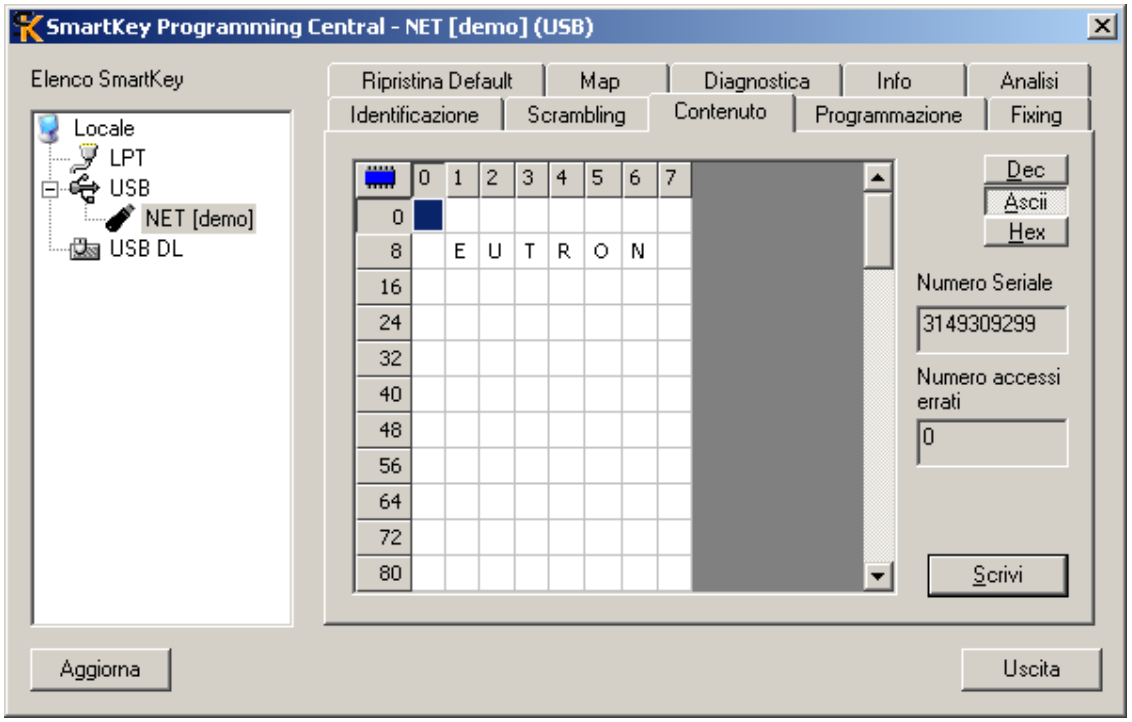

**Figura 15** Pannello *Contenuto* di una *SmartKey USB* di tipo *NET*.

ATTENZIONE: L'accesso alla memoria della SmartKey avviene in modo diretto e senza filtro. Si consiglia di non inserire dati nella memoria se si vuole usare SmartKey per limitare il numero di licenze multi utente o il numero di esecuzioni massime di un programma, perché si corre il rischio di scrivere su celle di memoria necessarie per questi due tipi di servizi.

### *16.7 Pannello fixing*

Il pannello *Fixing* permette di rendere la SmartKey non più riscrivibile. Premendo il pulsante *Fissa*, i registri *Label* e *Password* e la *memoria dati* diventano non-modificabili. L'operazione *Fissa* è irreversibile. La figura 16 mostra un esempio di pannello *Fixing*.

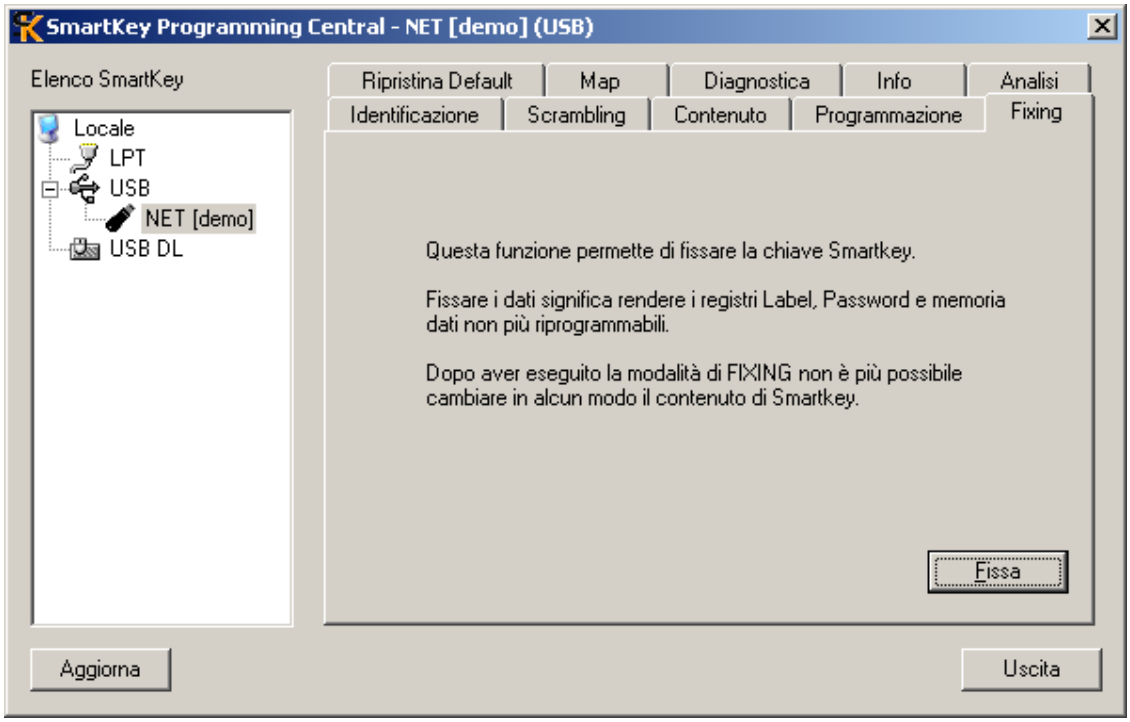


**Figura 16** Pannello *Fixing* di una *SmartKey USB* di tipo *NET*.

### *16.8 Pannello programmazione*

Il pannello *Programmazione* permette di gestire i file che contengono la configurazione della SmartKey. *Programmazione* permette di creare una nuova configurazione e salvarla su file, ripristinarne una salvata su file , modificarla e salvarla nuovamente su file e, infine, scrivere la configurazione selezionata su SmartKey. La catalogazione delle varie *configurazioni* avviene in base al nome del cliente a cui SmartKey è destinata. Infatti, ogni configurazione salvata deve essere associata al cliente che userà la SmartKey. La figura 17 mostra l'esempio di una *SmartKey USB* di tipo *NET* in cui è stato selezionato la configurazione destinata al cliente "JQSAED". (La selezione del file è fatta usando il mouse e cliccando sul file). La tabella centrale di *Programmazione* mostra le possibili configurazioni che posso scrivere in SmartKey**.** Ogni configurazione corrisponde ad un file aperto usando il tasto *Apri*. I pulsanti del pannello hanno i seguenti compiti:

- *Apri:*Apre un file contenente la configurazione e la aggiunge nella lista della tabella centrale. (Nella figura 17 sono presenti tre configurazioni ed è stata selezionata la configurazione per il cliente INFO)
- *Salva:* Salva la configurazione selezionata in un file.
- *Nuovo:* Crea una nuova configurazione. Selezionando il pulsante si apre il pannello *Dati Clienti* (figura 18) con tutti i campi vuoti.
- *Modifica:* Modifica la configurazione selezionata. Selezionando il pulsante si apre il pannello *Dati Clienti* (figura 18) contenenti tutti i valori della configurazione.
- *Cancella:* elimina dalla tabella la configurazione selezionata.
- *Scrivi:* scrive la configurazione selezionata nella SmartKey. (Nel caso della figura 17 è scritto la configurazione per il cliente JQSAED). Se il valore FIX della configurazione è pari a 1, la scrittura è

irreversibile (In caso d'errore, SmartKey diventa inutilizzabile). Se uguale a 0, i valori di SmartKey possono essere modificati.(L'irreversibilità di scrittura aumenta il grado di sicurezza.)

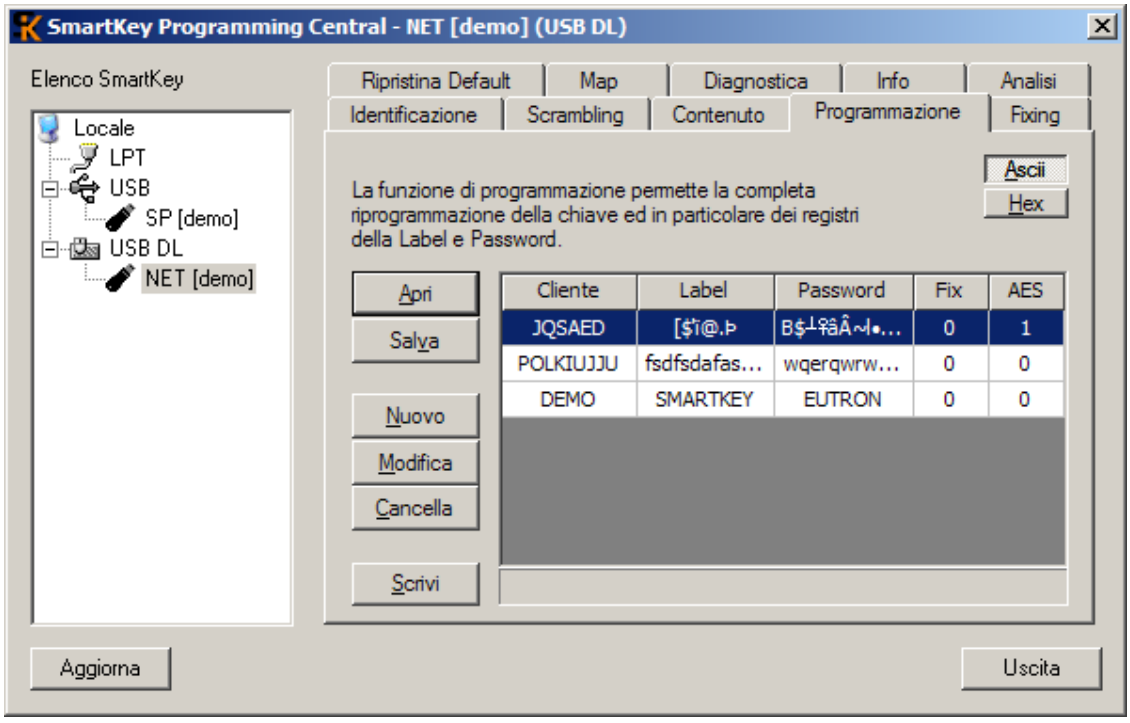


**Figura 17** Pannello *Programmazione* di una *SmartKey USB* di tipo *NET*.

La figura 18 mostra il pannello *Dati Cliente*. Il pannello permette di scrivere i dati della configurazione.

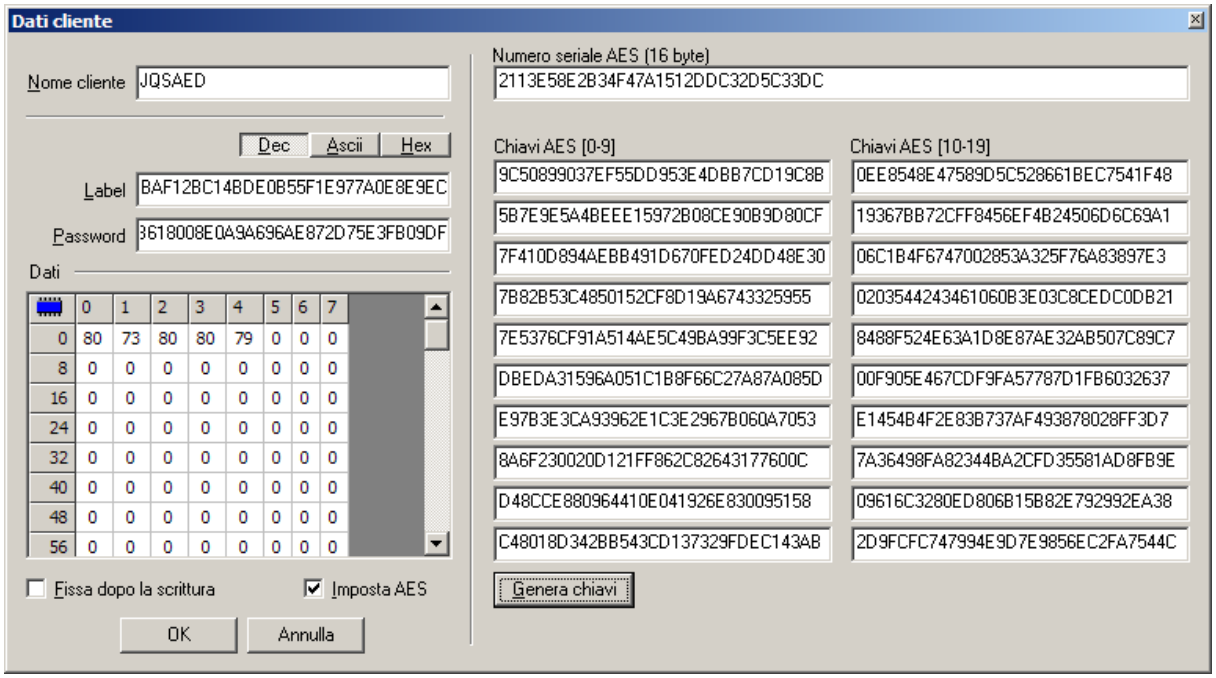


**Figura 18** Pannello *Dati Clienti*.Il pannello per l'impostazione della configurazione.

I campi di *configurazione* sono i seguenti:

- *Nome cliente:* il nome del cliente a cui la SmartKey è destinata.
- *Label:* Label della SmartKey
- *Password:* Password della chiave
- *Dati:* Nella *tabella dati* sono visualizzate le celle di memoria.

- *Fissa dopo la scrittura:* Se si seleziona l'opzione, la scrittura della configurazione su SmartKey sarà indelebile. La scelta dell'opzione è visibile anche dalla tabella centrale del pannello configurazione. Se FIX è uguale a 1 significa che l'opzione *Fissa dopo la scrittura* è stata selezionata, se uguale a 0 significa che l'opzione non è stata selezionata.
- *Imposta AES:* selezionare questa opzione significa scrivere le chiavi AES e il seriale AES all'interno della SmartKey. Le chiavi e il seriale AES sono utilizzati per l'autenticazione AES (vedi capitolo 9.16) e una volta impostati non sono più modificabili. La scelta dell'opzione è visibile anche dalla tabella centrale del pannello configurazione. Se AES è uguale a 1 significa che l'opzione *Imposta AES* è stata selezionata, se uguale a 0 significa che l'opzione non è stata selezionata.

I valori di *Nome Cliente*, *Label* e *Password* possono essere visualizzati in formato ascii o esadecimale, selezionando i pulsanti *Ascii* e *Hex*. I valori di *Dati* possono essere visualizzati in formato decimale o ascii o esadecimale, selezionando i pulsanti *Dec*, *Ascii* e *Hex*.

### *16.9 Pannello AES Scrambling*

Il pannello *AES Scrambling* permette di autenticare la chiave utilizzando l'algoritmo AES a 128 bit, anziché il comando *Scrambling*. Questo pannello è visualizzato solo se le chiavi AES sulla SmartKey sono state impostate.

Per eseguire l'autenticazione è necessario utilizzare il comando AES_SCRAMBLE e la decrittografia AES. Il comando AES_SCRAMBLE necessita in ingresso di una stringa di 16 byte casuali e dell'indice della chiave AES presente sulla SmartKey che si vuol utilizzare per l'autenticazione. L'applicativo deve conoscere il valore della chiave AES perché deve decrittografare il risultato ottenuto dalla AES_SCRAMBLE. Dopodichè, i dati decrittografati devono eseguire l'operazione di XOR con i dati casuali già utilizzati in ingresso al comando AES_SCRAMBLE. Se il valore ottenuto è uguale al numero seriale AES, l'autenticazione della chiave SmartKey si è conclusa con successo. Maggiori dettagli sull'autenticazione AES sono illustrati nel capito 9.16.

La figura 19 visualizza i dati e l'esito di un'autenticazione AES.

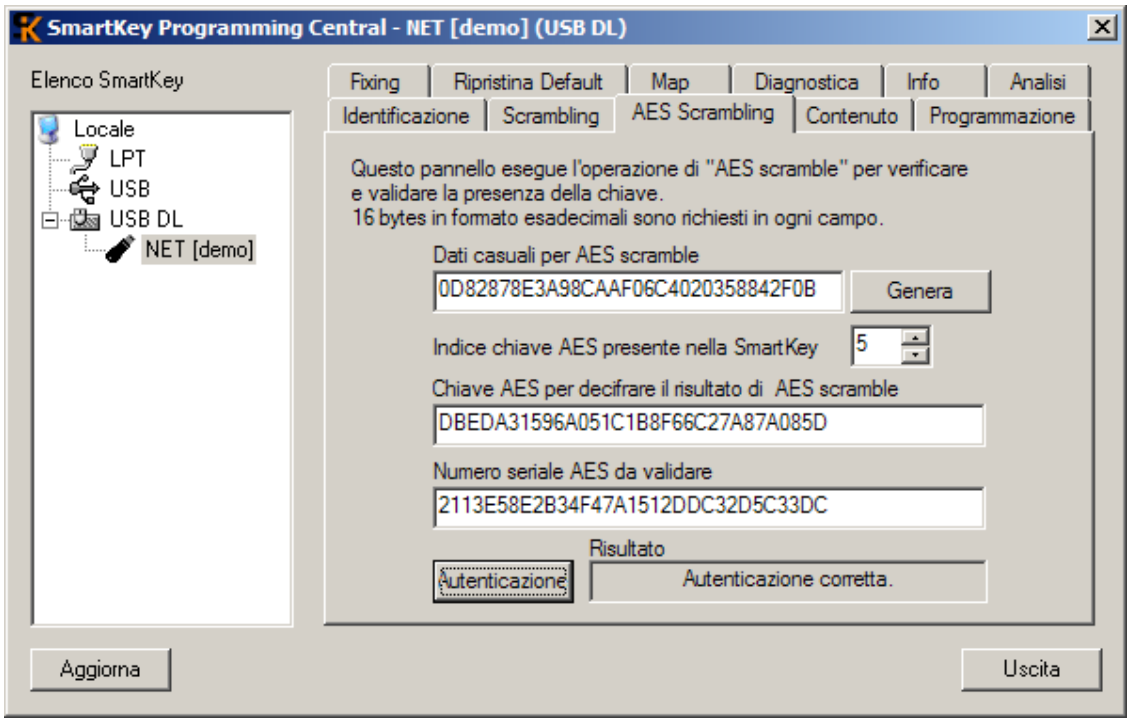


**Figura 19** Pannello *AES Scrambling*.

### *16.10 Pannello diagnostica*

SPC permette di analizzare l'intera memoria della SmartKey eseguendo cicli di lettura e scrittura in modo di diagnosticare se SmartKey è installata e funzionante correttamente. La figura 20 mostra il pannello *Diagnostica* per una *SmartKey NET***.** In cui sono stati selezionati 3 cicli di scrittura e lettura. La lettura e la scrittura sono selezionate tramite le opzioni *Test di scrittura* e *Test di lettura*. Il numero di cicli è impostato tramite *Numeri di cicli*. Il tasto *Esegui* fa iniziare il processo di diagnostica e il tasto *Interrompi* termina anzitempo il processo.

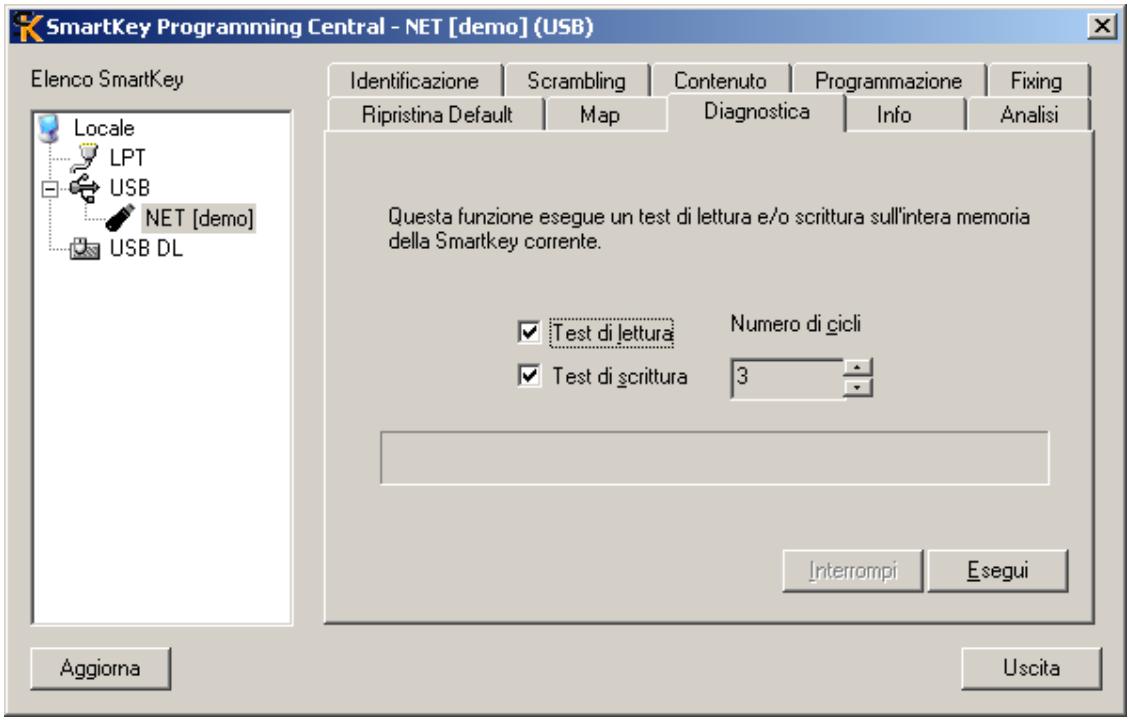


**Figura 20** Pannello *Diagnostica.*Pannello per eseguire la diagnostica di SmartKey.

### *16.11 Pannello analisi*

Il pannello *Analisi* permette di eseguire test riguardanti i driver di SmartKey, i dispositivi, il sistema e la SmartKey stessa e genera un file di report. Il file di report, un normale file di testo, può essere poi spedito all'assistenza clienti per avere una spiegazione dettagliata sulle cause del problema e informazioni su come risolverlo. Per generare un report più dettagliato si può allegare un elenco di programmi che usano la S*martKey*. Il tasto *Aggiungi* permette di aggiungere all'elenco i nomi dei programmi. Il tasto *Genera report* fa partire le procedure d'analisi e di scrittura sul file del report. La figura 21 mostra il pannello *Analisi* per una *SmartKey NET* in cui non sono stati inseriti programmi nella lista delle applicazioni.

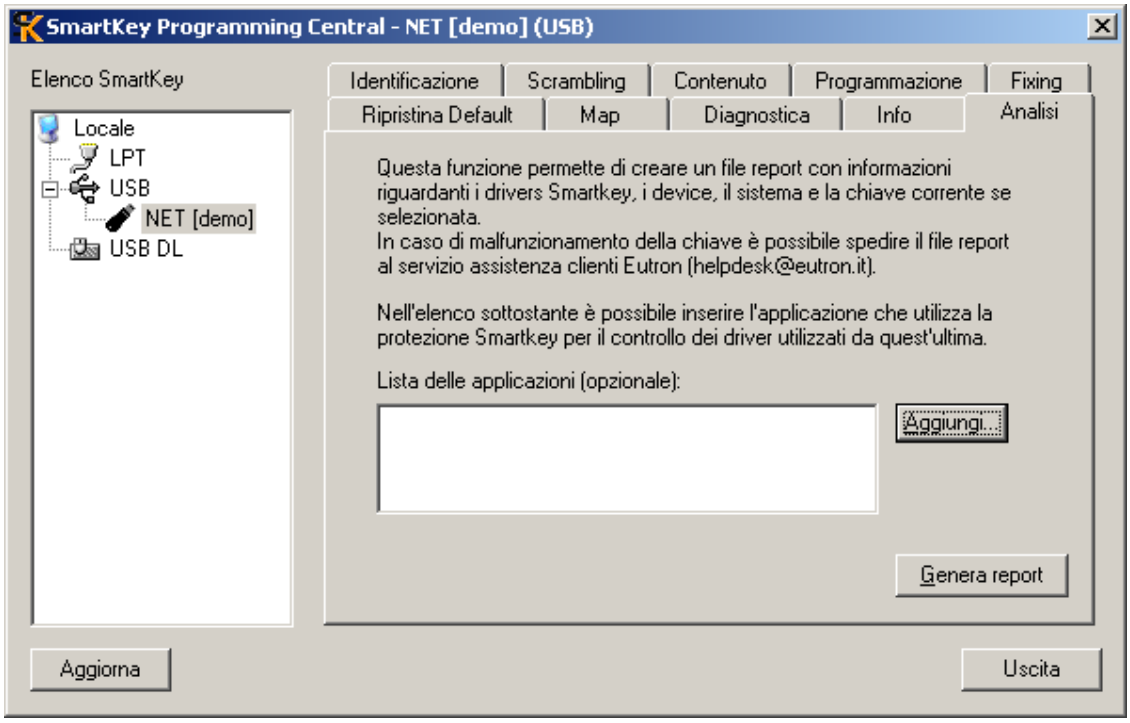


**Figura 21** Pannello *Analisi*. Il pannello genera file di report.

# 17 Specifiche Tecniche

### *17.1 Avvertenze*

- Inserire *SmartKey Parallela* tra il PC e la stampante quando entrambi sono spenti.
- SmartKey è sensibile alle cariche elettrostatiche. Evitare di toccare i pin dei connettori di SmartKey.
- Non sottoporre la SmartKey a temperature elevate o ad elevati sbalzi di temperature.
- Eventuali guasti elettrici del computer o di sue periferiche possono danneggiare irreversibilmente la SmartKey.
- Non inserire SmartKey nella porta seriale a 25 poli: le tensioni presenti possono alterare il contenuto della SmartKey e danneggiarla.

### *17.2 Funzionalità*

Meccanismo di protezione: tramite password e algoritmico
Codici di accesso: fissi o programmabili 16+16 byte
Dati memorizzabili: 64/128/416/896/8192 byte
Rivelazioni tentativi di accesso
Possibilità di *congelare* i dati memorizzati

### *17.3 SmartKey 2 Parallela*

Numero di scritture: 100.000 (tipico)
Memorizzazione dei dati: 10 anni (tipico)
Dimensioni della chiave: 48 x 52 x 15 mm
Interconnessione: porta parallela standard Centronics
Connettore lato computer: D-type 25 poli maschio
Connettore lato stampante: D-type 25 poli femmina
Alimentazione: autoalimentata da porta parallela
Expected MTBF: 2.800.000 ore
Temperatura di funzionamento: 0 - +70 ºC (32 - 158 ºF)
Umidità: 20-80 % umidità relativa

### *17.4 SmartKey 3 USB / SmartKey 3 USB DL*

Numero di scritture: 100.000 (tipico)
Memorizzazione dei dati: 10 anni (tipico)
Interconnessione: USB 2.0 LowSpeed
Connettore: USB Male Type A
Alimentazione: autoalimentata da porta USB
Expected MTBF: 3.000.000 ore
Temperatura: -20 - +80°C
Umidità: 20-95% umidità relativa