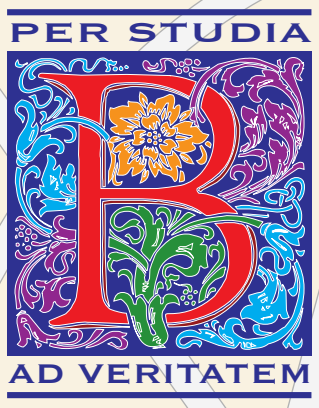


SYNTHESIS SERVICES GmbH

Capitalia Informatica
Relazione finale di audit, *fase 1*
Algoritmi QPK
Progetto di integrazione PBY

Davide Butti
Centro Studi Bononia
Synthesis Services GmbH

Relazione finale dell'Audit PBY-QPK, *fase 1*

Davide Butti

Release 2
18 dicembre 2006

Indice

Capitolo 1

Introduzione

In questa relazione vengono esposti i risultati dell'attività di audit svolta, per conto di Capitalia Informatica, nei riguardi di una proposta nuova piattaforma per la protezione delle transazioni via Web. Tutti i risultati contenuti in questo documento rappresentano le conclusioni cui è giunto l'autore, basandosi sul meglio delle proprie conoscenze, delle informazioni raccolte alla data odierna e sui principi metodologici esposti più oltre (1.2). Una parte delle informazioni confluite nella presente relazione è stata resa disponibile dai rispettivi aventi diritto sotto il quadro normativo di un *non disclosure agreement*; le stesse informazioni sono pertanto divulgate a Capitalia Informatica in forma strettamente confidenziale, e nel rispetto di quanto previsto dai predetti accordi.

Le interviste e gli scambi di informazioni necessari per il perfezionamento dell'audit si sono svolti secondo il seguente calendario:

- **26-27.10.2006** presso la sede di Capitalia Informatica a Roma.
- **17.11.2006** presso il Centro Studi «Vito Volterra» nell'Università di Roma Tor Vergata.
- **05.12.2006** presso la sede Synthesis Services a Lugano.

Nel corso del presente capitolo verrà descritto l'oggetto dell'audit (*target of evaluation*, TOE), con considerazione agli scopi che esso si prefigge, gli strumenti a tale fine adoperati, e le potenzialità di sviluppo future. Sarà inoltre presentata la metodologia seguita durante le varie fasi dell'attività.

Target of evaluation

1.1 Oggetto dell'audit

Il TOE è una piattaforma per la protezione di transazioni informatiche attraverso canali di comunicazione non sicuri. Alla forma attuale, nella quale è stato valutato, consiste nel progetto di applicazione di una nuova classe di algoritmi crittografici alla protezione di transazioni Web (`http`). Tali algoritmi formano l'oggetto di studio di una équipe di ricercatori dell'Università di Roma Tor Vergata (Accardi, Regoli, Susa), i quali ne detengono la proprietà intellettuale. Il progetto di applicazione viene proposto dalla società PBY S.r.l., che ha sviluppato semplici softwares dimostrativi che implementano i predetti algoritmi, e condotto lavori di analisi preliminare per l'applicazione di questi ai sistemi di Capitalia Informatica tramite una specifica architettura e tecnologia.

La valutazione del TOE – di conseguenza – è stata condotta su due livelli differenti e paralleli. Nei confronti dei fondamenti teorici si è proceduto all'analisi della letteratura, degli studi e delle implementazioni pilota disponibili; nei confronti del software applicativo di cui è stata proposta la realizzazione si è svolta un'indagine tesa a identificarne la compatibilità con gli obiettivi assunti in partenza, e descritti in (1.1.1). Tale duplice approccio, nei confronti di un TOE che non è attualmente specificato e realizzato per intero, caratterizza l'audit descritto in questa relazione, e viene definito *fase 1*.

Si considera imprescindibile, al fine di garantire la qualità e la consistenza del TOE così come verrà realizzato e completato, prevedere ulteriori attività di audit indipendente, dedite alla revisione di tutti i processi di analisi, progettazione, sviluppo, test e documentazione del software applicativo (per l'architettura di quest'ultimo si veda 2.1).

1.1.1 Scopo del TOE

I proponenti del TOE assumono come dato fondamentale la fragilità degli odierni sistemi di protezione delle transazioni Web. Tale fragilità viene ricondotta ad un insieme di fattori che possono essere riepilogati come segue:

1. Comportamenti che si riconducono a tecniche di *social engineering*; questo termine indica azioni tramite le quali il comportamento degli individui viene influenzato su larga scala, generalmente ingannando la buona fede delle vittime. Ricadono in questa categoria due tipi di attacchi ampiamente praticati dagli hackers:

   Phishing È un tipo di attacco nel quale, sfruttando accorgimenti di variabile sottigliezza [2], l'utente di un servizio Web viene indotto a rivelare informazioni riservate ad un soggetto che poi ne fa un uso illegittimo.

   Pharming È un attacco simile al precedente, ma maggiormente basato sullo sfruttamento di vulnerabilità tecnologiche anziché comportamentali. In questa fattispecie non è l'utente ad essere ingannato in merito all'identità dell'interlocutore, ma direttamente il software da questi utilizzato.

   Va notato che, sebbene casi semplici di phishing e pharming possano essere evitati con accorgimenti di immediata attuazione, esistono esem-

pi molto più sottili per i quali la soluzione non è banale. Inoltre, questi attacchi comportano la massima pericolosità perché fanno leva su debolezze che il fornitore del servizio non è in grado di controllare ed eliminare.

2. Comportamenti basati sulla difficoltà di mantenere la segretezza; se da un punto di vista teorico i sistemi *one time password* basati per esempio sulle liste di stralcio offrono la massima sicurezza possibile – in quanto equivalenti al Cifrario di Vernam [4] – purtroppo la loro natura materiale li rende facilmente duplicabili e, di conseguenza, inefficaci. In certe applicazioni, inoltre, è criticabile il fatto che l'identificazione dell'utente avvenga solamente all'inizio di una sessione di lavoro.

liste di stralcio e Cifrari di Vernam

3. Vulnerabilità degli algoritmi di cifratura dei dati. Sebbene questo fattore presenti una minore probabilità rispetto agli altri di influire sulla sicurezza di un sistema nel medio e breve termine, è pure vero che gli algoritmi correntemente utilizzati (*e.g.* DES, 3-DES, AES) non sono – sotto determinati profili – ottimali e pienamente soddisfacenti[1] (si veda 3.3 e 4.1).

Rispetto a tali assunti, il TOE viene presentato come rimedio efficace ed efficiente alle fragilità ivi descritte, e proposto come strumento per garantire un livello di sicurezza alle transazioni Web, maggiore di quello offerto da soluzioni alternative. Viene inoltre presentata una modalità di *deployment* in grado di adattare il TOE ad applicazioni già esistenti, con diversi gradi di integrazione (si veda 2.4 e 6.1.1).

1.1.2 Strumenti del TOE

Gli strumenti impiegati dal TOE per la realizzazione degli scopi identificati in (1.1.1) possono essere distinti in due classi:

Algoritmici Viene proposto l'impiego di nuovi algoritmi per l'identificazione dell'utente (QPK-OTP) e per la cifratura dei dati riservati (QPK-DYN-1Apq), al fine di garantire un livello di sicurezza maggiore.

Tecnologici Viene proposto l'utilizzo della tecnologia "U3 Smart Drive", al duplice scopo di fornire all'utente un ambiente di esecuzione software *trusted*, e di mettere a disposizione uno *storage* sicuro per custodire gli elementi segreti necessari agli algoritmi QPK.

Si noti che l'unione di questi due differenti ordini di mezzi è elemento essenziale per la realizzazione degli scopi del TOE (si veda 2.2). Tali mezzi devono essere implementati sia dal lato dell'utilizzatore del servizio (dispositivo U3) sia dal lato del fornitore del servizio (software di integrazione). Nel proseguimento della trattazione i termini *fornitore del servizio* e *utilizzatore del servizio* verranno utilizzati per identificare, rispettivamente, la Banca e il cliente remoto.

1.1.3 Ambito di applicazione del TOE

Allo stato attuale di specifica, il TOE viene proposto come strato di autentica-

strato di autenticazione e protezione

zione e protezione dei contenuti HTML sul canale `http`. Tramite il dispositivo U3 si mette a disposizione dell'utilizzatore un browser Web personalizzato, in grado di gestire gli specifici protocolli QPK; dal lato del fornitore gli stessi protocolli e le connesse funzioni di gestione sono realizzate da uno strato di software *anteposto* alle strutture esistenti. Una integrazione più profonda con il software applicativo già esistente non viene esclusa ma demandata ad una eventuale fase ulteriore di *deployment*.

browser personalizzato

Nella valutazione complessiva del TOE si tiene anche conto della portata di alcune previste future applicazioni dello stesso, in ambiti che possono completare e rafforzare la sicurezza del sistema così come è proposto attualmente. Le eventuali future applicazioni sono indicate come segue:

- Protezione di un canale e-mail; partendo dalla considerazione che la maggior parte degli attacchi di phishing viene condotta attraverso tecniche di *spamming*[2], la possibilità di prevedere un sistema di comunicazione con la Banca separato dai normali sistemi di posta elettronica riveste un certo interesse.
- *Storage* riservato (*crypto-filesystem*); con piccole varianti, l'infrastruttura proposta dal TOE potrebbe essere applicata alla creazione di aree di archiviazione di dati riservati nella LAN aziendale.

1.2 Metodologia dell'audit

Lo stato del TOE nel momento in cui viene valutato in questa *fase 1*, come descritto in (1.1), rende necessario un approccio variato, nei confronti dell'applicazione informatica proposta e degli algoritmi crittografici su cui si basa. Nel seguito della presente sezione sono illustrati gli strumenti metodologici adottati per l'una e per gli altri.

1.2.1 Audit degli algoritmi crittografici

Gli algoritmi crittografici proposti dal TOE, descritti in dettaglio in (3) e (4), sono il risultato di una serie di ricerche condotte al Centro Studi «Vito Volterra» sotto la guida del Prof. Accardi. I problemi generali che sottostanno a tali ricerche, relativi soprattutto alla generazione di successioni pseudocasuali, sono discussi in articoli pubblicati nella letteratura scientifica; le applicazioni strettamente legate alla crittografia, così come sono impiegate nel TOE, non sono invece pubblicate.

Nei confronti di questa materia si è seguito un approccio ispirato al principio della *peer review* utilizzato nella comunità scientifica; più in dettaglio, riguardo agli algoritmi in esame, si è proceduto a:

peer review

- Accedere ai fondamenti teorici dichiarati dai proponenti degli algoritmi, verificando la legittimità della loro applicazione al caso specifico.
- Accedere ai risultati pubblicati dall'équipe di studio, verificandone il *follow-up* e gli eventuali sviluppi o critiche che ne sono derivati.
- Raccogliere direttamente dagli studiosi interessati, per mezzo di interviste dedicate ai vari singoli aspetti, la descrizione dei contenuti che non sono pubblicati.

- Raccogliere tutti gli elementi necessari alla ripetizione degli esperimenti svolti durante l'elaborazione degli algoritmi, che nel caso specifico rivestono particolare importanza.
- Valutare la legittimità e l'esattezza degli esperimenti descritti al punto (1.2.1), riproducendoli in maniera indipendente e verificando la congruenza dei risultati.
- Verificare che le descrizioni procedurali degli algoritmi proposti gli esperimenti svolti non configurino violazioni di brevetti, marchi industriali o copyright nell'Unione Europea, Stati Uniti e Giappone.
- Valutare il grado di rispondenza fra le caratteristiche degli algoritmi – così come confermate dagli esperimenti riprodotti – e gli attributi complessivi del TOE, ammettendo che la realizzazione informatica di quest'ultimo sia svolta secondo le evidenze raccolte come descritto in (1.2.2).

1.2.2 Audit dell'applicazione informatica

L'analisi dell'insieme delle applicazioni informatiche e delle tecnologie proposte dal TOE ha richiesto un approccio maggiormente sfumato, come descritto in (1.1); la valutazione è stata svolta unicamente nei confronti delle analisi preliminari e della progettazione architetturale svolte, che alla data rappresentano le sole realizzazioni pratiche del TOE attuate. Il valore dei risultati ottenuti in questa parte dell'audit assume pertanto una duplice valenza; da un lato *convalida* delle scelte proposte, dall'altro *orientamento* metodologico ed ingegneristico rivolto alla futura realizzazione del TOE.

ISO Common Criteria

Il quadro di riferimento assunto come guida in questa parte dell'audit risiede nel documento ISO 15408, noto anche come *Common Criteria*, che mette a disposizione una metodologia ampiamente accettata e collaudata per la valutazione della sicurezza dei sistemi IT. La metodologia descritta in tale standard individua in modo strutturato ed esaustivo gli aspetti di un TOE che sono in grado di influenzarne le caratteristiche di sicurezza; per tutti questi aspetti, poi, sono previsti vari livelli crescenti di "garanzia" (*assurance*). Il livello di garanzia da adottare per ogni componente può essere individuato tramite una semplice matrice, una volta che si sia prescelto il livello di garanzia (*Assurance level*) che si vuole ottenere[3]. Preliminarmente, e senza che tale scelta venga assunta come principio assoluto, si è deciso – sull'analogia dell'esperienza realizzata con TOE analoghi – di assumere come obiettivo l'*Evaluation assurance level* EAL4. Tale livello, che conduce ad un TOE *metodicamente progettato, testato e sottoposto a revisione*, è normalmente assunto come riferimento nei casi di applicazioni informatiche che trattano dati riservati.

Rinunciando, per la attuale *fase 1* dell'audit, ad un'applicazione rigorosa e completa di ISO 15408, dalla stessa si è ritenuto opportuno ritenere ed adottare i seguenti punti:

- A fronte della descrizione architetturale del TOE fornita dai proponenti, si è proceduto alla formazione di una "checklist" basata sull'elenco degli *Assurance components* individuati da ISO 15408. Questa checklist è stata utilizzata internamente, durante l'audit, per dirigere la raccolta

delle informazioni[4], attuata mediante una serie di interviste e discussioni con i proponenti del TOE. Le informazioni raccolte sono state in seguito verificate e convalidate mediante l'accesso alla documentazione rilevante, nonché tramite l'applicazione delle norme comunemente accettate di ingegneria del software[5].

- Facendo riferimento ai livelli di garanzia previsti da EAL4 per i vari componenti, si è valutata l'adeguatezza delle soluzioni proposte nel TOE, intervenendo talora mediante l'individuazione di specifiche *raccomandazioni* raccolte in una specifica sezione di questa relazione (7).

- I contenuti della metodologia ISO 15408 sono stati condivisi con i proponenti del TOE, raccomandando di assumerne i principali contenuti come guida del futuro lavoro di analisi, sviluppo, test, implementazione e documentazione. In particolare, è stata posta particolare enfasi sugli aspetti che devono essere presi in considerazione fin dall'inizio delle attività[6].

L'esito finale della metodologia sopra esposta può quindi assumere solo molto parzialmente la forma di un'indicazione "go/no-go" nei confronti di questa parte del TOE; più realisticamente, esso si è configurato come delineazione delle condizioni *sine quibus non* necessarie ad ottenere la rispondenza del TOE agli scopi prefissati. Per l'esposizione di queste valutazioni si veda (2).

Note

[1]La sicurezza delle transazioni Web è normalmente affidata all'utilizzo dello standard SSL/-TLS, che risulta nel protocollo `https`. Tale standard si basa su certificati digitali X.509 per verificare un determinato tipo di "corrispondenza" fra l'identità reale e quella attesa della controparte della comunicazione. Il largo livello di automazione presente nel meccanismo di *handshake* iniziale di SSL lascia un certo spazio di manovra per l'allestimento di attacchi di phishing alquanto insidiosi (si veda ad esempio [6]); risulta quindi accoglibile la tesi per cui sia possibile raggiungere un maggiore livello di sicurezza adottando algoritmi e protocolli proprietari, di portata meno ampia. Cfr. anche (2.2).

[2]Nei casi più comuni, l'utente riceve messaggi e-mail che indicano fraudolentemente come mittente la Banca stessa; in questi messaggi sono poi contenuti link più o meno "well-behaved" che inducono l'utente ad aprire connessioni con applicazioni Web costruite a perfetta imitazione di quelle delle Banca. Se l'utente non rileva l'inganno ed inserisce nell'applicazione le proprie credenziali, l'attaccante è in grado di riutilizzare le stesse per effettuare una connessione alla reale applicazione della Banca, potendo poi operare a proprio piacere. È evidente da questa descrizione che il miglior rimedio contro questo pericolo risiede, come delineato in (3.1), nell'allestimento di uno schema *one time password* dove l'identificazione sia svolta ad ogni singola transazione `http`.

[3]Nella terminologia ISO 15408, *Evaluation assurance level* indica una scala fatta di 7 livelli di crescente garanzia complessiva della sicurezza, denominati da `EAL1` a `EAL7`, per la cui definizione dettagliata si rimanda a [14]. Tale quadro ha un riferimento solo indicativo e non normativo, nel senso che è anche possibile scegliere livelli di garanzia per i singoli componenti, diversi da quelli previsti dallo specifico `EAL` in uso. Naturalmente, tali scelte dovranno essere motivate in maniera dettagliata, per evitare che il procedimento nel suo complesso perda di valore.

[4]Dal momento che questa parte del TOE è scarsamente descritta in formato documentale, ma gran parte delle informazioni relative sono presenti sotto forma di conoscenze dei proponenti – specificamente di quelli fra loro appartenenti alla struttura PBY – il risultato di questa raccolta di informazioni riveste una grande importanza nei confronti dello sviluppo successivo, in quanto forma il nucleo di una prima (e incompleta) sistemazione organica di tutte le informazioni *security-related* attinenti al TOE.

[5]A dipendenza delle varie tecnologie di sviluppo indicate dai proponenti del TOE, tali norme sono da ricercare nelle pratiche di (a) analisi e sviluppo *Object Oriented* (cfr. [18]); (b) applicazione dei *Patterns* (cfr. [16]); (c) programmazione strutturata e strutture dati fondamentali (cfr. [17]).

[6]Per esempio la necessità di poter ricostruire la storia di tutte le varianti apportate al codice sorgente, e l'utilità di adottare e seguire uno standard di documentazione dettagliata dello stesso.

Capitolo 2

Descrizione e valutazione di alto livello

2.1 Architettura del TOE

2.1.1 Profilo di utilizzo

Nell'architettura proposta, l'utilizzatore dei servizi via Web offerti dalla Banca non si serve più del normale browser HTML per accedere agli stessi. Nel caso provi a farlo, per esempio digitandone manualmente l'indirizzo in un browser tradizionale, il servizio riconosce opportunamente la circostanza e non autorizza l'ingresso nell'applicazione[1].

Per accedere ai servizi l'utilizzatore dovrà servirsi unicamente di un particolare browser, che gli viene fornito dalla Banca su un dispositivo U3. È questo un tipo particolare di *flash drive* USB, in grado di permettere l'utilizzo di software senza che questo venga installato sul PC[2]. Lo speciale browser installato su questo dispositivo U3, e utilizzabile direttamente inserendo lo stesso nella porta USB di un PC, ha la particolare caratteristica di contenere un'implementazione degli algoritmi QPK necessari a identificarsi correttamente ai servizi della Banca e scambiare con questi gli opportuni messaggi cifrati[3].

Le particolari caratteristiche dell'algoritmo utilizzato per l'identificazione dell'utente (vedi 3.1) fanno sì che, per l'accesso al servizio Web, l'utilizzatore debba essere in possesso del dispositivo U3 (che è personalizzato in maniera differente per ogni utente) e contemporaneamente essere a conoscenza di una password ad esso associata (PIN). Tale PIN può essere cambiato dall'utente in maniera autonoma in qualsiasi momento. A partire dal PIN e dai dati riservati contenuti nel dispositivo U3, il protocollo di autenticazione genera una stringa di autenticazione diversa per ogni interazione con il servizio (sostanzialmente una *one time password* automatica)[4].

Nel primo livello previsto di integrazione con le applicazioni della Banca, questa identificazione tramite PIN e dispositivo U3 risulta *aggiuntiva* rispetto ad ogni meccanismo di autenticazione già esistente[5]; se quindi l'applicazione prevede, ad esempio, una pagina di login, questa verrà visualizzata subito dopo l'immissione del PIN (se esatto). La transizione ad un secondo, più profondo livello di integrazione con le applicazioni permetterà in seguito di

eliminare il login dell'applicazione, ad esempio effettuando una mappatura fra l'identificazione fornita dal TOE e le credenziali *legacy*.

Non appena il dispositivo U3 viene estratto dalla porta USB, il browser viene chiuso e tutti i dati temporanei eventualmente generati sono rimossi dal PC; in ogni caso, anche se la chisura e la rimozione non dovessero avvenire[6] o fossero incomplete, appena il dispositivo U3 viene a mancare non è più possibile il funzionamento dell'algoritmo di identificazione (che si basa su alcuni parametri che sono memorizzati in un'area protetta del dispositivo U3 e vengono letti di volta in volta), e di conseguenza l'applicazione Web non risulta più accessibile.

2.1.2 Infrastruttura per la fornitura del servizio

Sui Web server impiegati dalla Banca per il supporto delle applicazioni viene installato un apposito software di filtraggio, che riconosce in base alla URL il traffico diretto da e verso le applicazioni protette dal TOE. Tale traffico viene trasferito, su canale sicuro, ad una batteria di server facenti parte del TOE, i quali utilizzano gli algoritmi QPK per svolgere le funzioni di identificazione dell'utente e cifratura del traffico. Quando tali funzioni hanno successo, il flusso di dati elaborato viene ritornato – attraverso la stesso canale utilizzato in senso inverso – ai Web server, che ne proseguono l'elaborazione in maniera normale.

L'attività del TOE viene tenuta sotto controllo tramite un apposito software di monitoraggio[7] che raccoglie e visualizza parametri e statistiche di funzionamento.

distribuzione e gestione dei dispositivi U3

La Banca provvede alla distribuzione agli utilizzatori del dispositivo U3; al momento in cui il singolo dispositivo viene associato ad uno specifico utente, alcune informazioni prese dal dispositivo stesso devono essere inserite nel database del TOE per permettere il funzionamento di quello. È possibile, in questo ambito, la realizzazione anche contemporanea di scenari diversi:

A. La Banca riceve i dispositivi U3 in grandi quantità, con il software client del TOE preinstallato, e provvede a distribuirli nelle proprie Agenzie. Nel momento in cui un dispositivo viene affidato ad un utente, il corrispondente legame viene inserito in tempo reale nel database, ad esempio tramite un'applicazione disponibile via intranet ed accessibile mediante il dispositivo stesso[8]. Il cliente può anche, contestualmente, scegliere ed immettere nell'applicazione il proprio PIN, che viene assegnato al dispositivo.

B. La Banca affida la gestione della consegna iniziale dei dispositivi U3 ad un ente esterno. In questo caso, l'ente esterno riceve dalla Banca o direttamente dal cliente la richiesta del rilascio di un nuovo dispositivo, e provvede alla sua personalizzazione (per esempio tramite accesso tipo VPN ai sistemi della Banca) e spedizione o consegna. In questo caso è possibile fare ricorso ai tradizionali schemi di attivazione tramite canale alternativo (per esempio tramite telefono) per evitare che un dispositivo smarrito o sottratto durante la consegna possa essere utilizzato da un soggetto non legittimato.

È sempre possibile per la Banca o per il soggetto descritto al punto (B), in tempo reale, disattivare in maniera temporanea o definitiva ogni singolo di-

spositivo, mantenendo in tal modo un controllo completo sul funzionamento del TOE.

Va inoltre rimarcato che l'introduzione del TOE a protezione di specifiche applicazioni Web può essere effettuato in maniera progressiva, senza che l'accesso tradizionale sia disattivato da subito (si vedano le considerazioni in 2.4.2); si ritiene comunque opportuno considerare transitoria e non definitiva l'eventuale coesistenza dell'applicazione "standard" con la versione protetta dal TOE, per non correre il rischio di annullare i benefici da quest'ultimo ricavati.

2.2 Valutazione dei vantaggi

L'introduzione di una corretta implementazione del TOE permette di realizzare importanti benefici, identificabili con una serie di vantaggi *operativi* (connessi cioé ad una maggiore robustezza e facilità di utilizzo) ed una serie di vantaggi *strategici* (relativi alla possibilità di elevare la qualità o l'efficienza del servizio offerto, attraverso miglioramenti non necessariamente percepibili dall'utilizzatore).

2.2.1 Vantaggi operativi

Una serie di vantaggi immediatamente conseguente all'introduzione del TOE, e direttamente sperimentabile dagli utilizzatori, può essere riepilogata come segue:

Maggiore convenienza di utilizzo: l'utente non deve più custodire o ricordare credenziali, passwords, o liste di stralcio. Per utilizzare le applicazioni è necessario solamente essere in possesso del dispositivo U3 e conoscere il PIN (che può essere scelto e variato a piacere).

Immunità agli attacchi *phishing*: questi attacchi non sono possibili nei confronti del TOE, per due ordini di motivi. In primo luogo, gli utilizzatori sono al corrente che l'unico strumento utilizzabile per accedere alle applicazioni della Banca è il browser contenuto nel dispositivo U3. Di conseguenza, diviene per loro facile rendersi conto se un hyperlink malevolo, veicolato ad esempio da un messaggio e-mail, tenta di indurli ad inserire proprie credenziali o informazioni riservate in un'applicazione non legittimata. In secondo luogo, anche ammettendo che un utilizzatore trascuri questa norma operativa e fornisca più o meno inconsciamente il proprio PIN al sito attaccante, quest'ultimo non è in grado di utilizzarlo in alcun modo, dal momento che non possiede gli algoritmi necessari (contenuti nel dispositivo U3)[9].

Immunità agli attacchi *pharming*: anche questi attacchi non sono possibili nei confronti del TOE; anche se il PC che si utilizza per accedere all'applicazione sia gravemente compromesso per esempio da virus *trojan*, o da tecniche di *DNS spoofing*, il meglio che l'attaccante riesce ad ottenere è di farsi inviare uno stream `http` cifrato con un algoritmo di cui non possiede i parametri, e che non riesce a decifrare[10].

2.2.2 Vantaggi strategici

Una classe di vantaggi di natura differente rispetto a quelli descritti in (2.2.1) è descritta dai seguenti punti:

Maggiore protezione dei dati: le caratteristiche dell'algoritmo QPK-DYN, utilizzato per la cifratura dei dati riservati trasmessi dall'applicazione, garantiscono una sicurezza più alta rispetto agli algoritmi utilizzati dai sistemi SSL/TLS; inoltre, dal momento che non vengono utilizzati certificati digitali[11], il TOE ha migliori caratteristiche di sicurezza anche perchè tutti i parametri necessari al suo funzionamento possono essere prodotti internamente dalla Banca[12] (si veda anche nota 20, capitolo 4).

Indipendenza dal browser: dal momento che l'accesso alle applicazioni avviene unicamente attraverso il browser personalizzato distribuito con il dispositivo U3, le nuove applicazioni eventualmente scritte appositamente per supportare il TOE non dovranno più necessariamente prevedere una compatibilità con diversi tipi di browser. Per lo stesso motivo, le applicazioni non dovranno essere sottoposte a verifica (o a modifiche) in occasione del rilascio di nuove versioni di browser.

Controllo centralizzato sulle applicazioni: in uno scenario in cui il TOE venga applicato alla protezione di molteplici applicazioni, il monitoraggio e le statistiche permesse dal sistema di gestione del TOE permette la raccolta e correlazione di informazioni che precedentemente non erano disponibili in forma unificata. A livello di gruppo bancario, è anche possibile prevedere un unico sistema di gestione della sicurezza ad uso di più banche.

Compatibilità con tecnologie diverse: anche se nella descrizione attuale del TOE la struttura proposta viene applicata alla protezione di applicazioni Web (si veda 1.1.1), il medesimo approccio può essere attuato per la protezione di applicazioni di natura differente, come e-mail, applicazioni *rich client* ed altre.

Dalla considerazione di questi vantaggi emerge l'indicazione di valutare il *return on investment* connesso all'introduzione del TOE non solo sulla base degli immediati benefici che esso è in grado di produrre, ma anche nel quadro di uno sviluppo più organico ed esteso all'interno della struttura IT.

2.3 Valutazione dei rischi

Lo sviluppo e l'introduzione del TOE comportano l'assunzione di una serie di rischi che si è provveduto a stimare, procedendo anche in questo caso ad un approccio diversificato nei confronti dei fondamenti teorici e degli strumenti realizzativi impiegati.

2.3.1 Rischi connessi all'introduzione dei nuovi algoritmi crittografici

È un principio intuitivo abbastanza diffuso nel campo della crittografia il fatto che la robustezza di un protocollo o algoritmo di protezione delle informa-

zioni possa essere stimato in base al tempo trascorso dall'introduzione dello stesso, senza che venga identificato un modo per "romperlo"[13]. In base a questa considerazione sarebbe ovvio concludere che l'introduzione di nuovi algoritmi, come propone il TOE, comporti un elevato rischio connesso con l'incertezza sulla robustezza di questi.

Occorre notare che questo principio, se pure contiene un fondamento di realtà che non deve essere sottovalutato[14], nasconde innanzitutto un atteggiamento di fondamentale *diffidenza* e mancanza di comprensione approfondita di una disciplina, quale è la crittografia, estremamente complessa e sofisticata. Tale atteggiamento è anche motivato dal fatto che un sistema crittografico è generalmente composto da molte parti diverse, e gli strumenti ingegneristici adottati nella sua costruzione non sono sempre adeguati a garantire il raggiungimento di un determinato standard di sicurezza in tutte quelle parti[15].

consolidate *vs.* più sicure

Rinunciando quindi alla posizione assolutamente conservativa che si limita all'adozione passiva di tecniche "consolidate" – termine che non implica il concetto di "più sicure" – si ritiene più opportuno valutare con strumenti analitici il più possibile raffinati le caratteristiche proposte dal TOE, in modo da confrontarne *con ragionevole grado di certezza* la robustezza con quella dei sistemi di impiego ormai comune.

Occorre allora rilevare come l'analisi approfondita degli algoritmi QPK-OTP e QPK-DYN, come dettagliata nei capitoli 3 e 4 – e nell'ipotesi che vengano attuate le raccomandazioni ivi contenute – permetta di concludere che il rischio connesso con il loro utilizzo da parte del TOE *non è maggiore rispetto all'utilizzo di algoritmi già noti in letteratura e nell'industria*, ed è estremamente sensato attendersi anzi che tale rischio sia sensibilmente più basso. Si noti, infine, che presupponendo anche solo che gli algoritmi QPK *eguaglino* in robustezza quelli standard, il risultato complessivo sarebbe comunque quello di un TOE in grado di raggiungere gli scopi prefissati.

2.3.2 Rischi connessi all'implementazione

Rispetto all'architettura di una normale applicazione Web-based, il TOE introduce un certo numero di elementi aggiuntivi, identificabili con:

- Il dispositivo U3 fornito all'utilizzatore.
- Il browser personalizzato installato sul dispositivo U3.
- Il server che implementa le funzioni di identificazione e cifratura presso il fornitore del servizio[16].

Per una descrizione dettagliata delle caratteristiche di questi elementi si rimanda ai capitoli 5 e 6. I risultati della valutazione ivi svolta permettono di affermare che – nell'ipotesi in cui le premesse e le raccomandazioni descritte siano soddisfatte – i benefici derivanti dall'introduzione di una complessità aggiuntiva siano più che proporzionalmente compensati dalla maggior robustezza conseguita dal TOE. Occore notare infatti come una buona parte della vulnerabilità delle applicazioni Web – ed al cui rimedio il TOE è per definizione diretto (vedi 1.1.1) – sia dovuta al fatto che in esse uno strumento di utilizzo generico quale è il browser sia spinto a svolgere funzionalità per le

quali non è stato progettato sin dal principio[17]. È quindi legittimo attendersi che una soluzione basata su un software diretto ad uno scopo più circoscritto, come avviene nel caso del TOE, possa essere meglio specificata e verificata, e presentare un rischio minore.

Una valutazione più cauta è necessaria riguardo ai rischi derivanti dalla distribuzione e utilizzo di software scritto appositamente per il TOE; allo stato attuale si ritiene che – nei confronti di tutte le potenziali fonti di rischio – siano opportunamente previsti strumenti in grado di controllare e limitare quest'ultimo. La corretta realizzazione e implementazione di tali strumenti potrà però essere studiata e confermata soltanto da interventi di audit successivi, condotti su un livello di completamento del TOE maggiore.

2.4 Valutazione dell'impatto

L'introduzione del TOE, oltre a comportare una serie di benefici ed un miglioramento della sicurezza nell'utilizzo delle applicazioni Web, è destinata a modificare il modello di utilizzo delle stesse, nonchè la struttura necessaria ad erogarle; nel seguito di questa sezione viene valutata la portata di tali modifiche.

2.4.1 Impatto per l'utilizzatore del servizio

Il maggior vantaggio *percepito* di una applicazione Web è la possibilità di accedervi non avendo a disposizione altri strumenti che una qualsiasi piattaforma elaborativa dotata di capacità di *browsing*[18]. Da questo punto di vista, l'introduzione del TOE a protezione delle applicazioni comporta una forte restrizione, perchè l'accesso alle stesse diviene possibile unicamente tramite piattaforma di tipo PC, compatibile con il dispositivo U3[19] e con il software proprietario distribuito con quello. Questo punto è stato considerato dai proponenti del TOE, che lo risolvono con la seguente duplice argomentazione:

1. Di fronte all'alternativa se offrire prestazioni di sicurezza e robustezza maggiori o conservare la compatibilità con una classe di di metodi di accesso, si considera che – trattandosi di ambito bancario – la sicurezza debba avere la precedenza.

2. Nell'ottica di perseguire un'introduzione graduale del TOE (si veda anche 2.4.2), si intende rivolgere la prima implementazione dello stesso alla protezione di applicazioni utilizzate da una classe ristretta di utenti[20]; per questo motivo si ritiene che le limitazioni sopra citate non appaiano significative.

Si richiama l'attenzione sulla necessità di seguire da vicino l'evoluzione della tecnologia U3, in modo da poter eventualmente valutare in futuro l'opportunità di modificare il TOE così da supportare sistemi operativi differenti. Occorre ricordare che le basi teoriche contenute negli algoritmi QPK si prestano, per loro natura, all'implementazione su dispositivi dalle capacità elaborativa molto limitate; per questo motivo è possibile immaginare che, qualora lo si desideri, sia possibile in futuro progettare varianti del TOE in grado di funzionare – con livelli di robustezza paragonabili – in congiunzione con utilizzatori dotati di dispositivi diversi dal Personal Computer[21].

Dal punto di vista degli utilizzatori che accedono alle applicazioni tramite la piattaforma Microsoft Windows™ supportata fin dall'inizio, è infine evidente che il TOE comporta un impatto positivo. Da un lato infatti puramente operativo, come descritto in (2.2.1), l'accesso risulta facilitato; altrettanto importante, da un punto di vista *soggettivo*, è il fatto che l'utilizzatore sia in grado di identificare fisicamente il possesso del proprio dispositivo U3 con la garanzia che la sua sicurezza è tutelata[22].

2.4.2 Impatto per il fornitore del servizio

Lo schema proposto dal TOE per l'integrazione con le applicazioni già esistenti è strutturato in modo da permettere, almeno nella fase iniziale (si veda 2.1.1), completa trasparenza e una complessità di implementazione minima[23].

Molte scelte e valutazioni di tipo organizzativo, invece, devono essere svolte per quanto riguarda la gestione logistica e il supporto dei dispositivi U3; come descritto in (2.1.2), esiste innanzitutto l'alternativa *make or buy* di collocare tale gestione[24] all'interno o all'esterno della Banca; successivamente, si pone il problema di stabilire e certificare procedure efficienti per la gestione stessa. Queste tematiche ricadono al di fuori dell'oggetto del presente audit, in quanto si valuta che esse non siano in grado di influire sulla qualità del TOE o sulla sua rispondenza al proprio scopo; si raccomanda in ogni caso di svolgere il vaglio di queste alternative in una fase precoce di implementazione del TOE, in modo da poter tenere conto – nella specifica e realizzazione dello stesso – delle decisioni prese.

2.5 Valutazione complessiva

validità complessiva del TOE

Si ritiene che il TOE, valutato nel suo complesso, costituisca una valida proposta nei confronti degli scopi da esso dichiarati, e sintetizzati in (1.1.1); esso contiene inoltre molteplici elementi innovativi, che lo mettono in grado di garantire un livello di robustezza e sicurezza per molti aspetti superiore a quello di sistemi con scopo similare.

Viene inoltre considerato in maniera positiva il fatto che le proprietà esibite dal TOE non dipendano da un singolo elemento caratterizzante, ma siano determinate dall'accostamento di una molteplicità di contenuti e di tecniche innovative in grado di rendere il sistema robusto e flessibile.

Per contro, la grande complessità dei contenuti proposti dal TOE, ed i molteplici strumenti necessari alla sua realizzazione, fanno ritenere che una corretta implementazione, tenendo anche conto delle raccomandazioni esposte nelle varie parti del presente lavoro, costituisca un compito particolarmente impegnativo. Anche sulla scorta delle indicazioni contenute in ISO 15408, si raccomanda pertanto di adottare una metodologia di sviluppo e verifica che preveda una costante revisione indipendente.

Note

[1]È da rilevare che questo controllo può essere fatto in maniera sicura e non attaccabile con tecniche di phishing o pharming (si veda 1.1.1), grazie all'utilizzo di protocolli di identificazione e cifratura proprietari.

[2]Per una descrizione dettagliata della piattaforma U3, si rimanda a (5).

[3]A differenza del normale scenario delle applicazioni protette con SSL/TLS, nelle quali pressochè tutto il traffico HTML è cifrato, l'architettura del TOE raggiunge un'efficienza maggiore, permettendo di decidere *applicativamente* quali contenuti proteggere.

[4]Proseguendo questa analogia, si rimanda alla trattazione dell'algoritmo QPK-OTP nel capitolo 3; si noti che tale algoritmo viene parametrizzato in modo da permettere la creazione di *un numero finito* di autenticazioni, trascorse le quali il dispositivo U3 deve essere sostituito. Benché non sia un problema rendere tale numero di autenticazioni "ragionevolmente" grande, occorre svolgere una valutazione preliminare in grado di stimarne correttamente il valore.

[5]Questa strada viene scelta in quanto ha il vantaggio di non richiedere alcun intervento sulle applicazioni esistenti. Per una valutazione dell'impatto di eventuali interventi quando si decida di evitare la doppia identificazione, si veda (2.4).

[6]per esempio a causa di un PC *pesantemente* compromesso, con intento malevolo o senza.

[7]che al momento si suppone *web based*.

[8]Questo approccio ha il duplice vantaggio di eliminare la necessità di un riporto manuale dei dati del dispositivo nel database e di realizzare un test di funzionamento dell'oggetto che viene rilasciato al cliente.

[9]Più propriamente, anche se conosce i dettagli di tali algoritmi, l'hacker non ne ha a disposizione i *parametri*.

[10]Si ricordi inoltre che l'identificazione dell'utilizzatore è ripetuta *ad ogni transazione*, utilizzando le stringhe sempre diverse prodotte dall'algoritmo QPK-OTP. Quindi l'attacco *pharming* non è neppure in grado di causare inconvenienti tramite tecniche di replaying delle informazioni ricevute (ad esempio molteplice esecuzione di ordini dispositivi).

[11]i quali si basano sull'algoritmo di cifratura asimmetrico RSA.

[12]Non sono necessari, in particolare, *grandi* numeri primi o convalide di certificati X.509 da parte di Autorità di Certificazione.

[13]Si veda ad esempio [19]; è evidente che tale principio intuitivo è basato su un paradosso logico, dal momento che se esso venisse applicato alla lettera, *nessun* algoritmo di cifratura sarebbe mai utilizzato nella realtà. Si è ritenuto in ogni caso opportuno prendere in considerazione questa diffusa opinione per meglio illustrare i principi di valutazione del rischio che ad essa sottostanno. Si noti, inoltre, che in tale opinione risiede anche un assunto palesemente errato, e cioè che, nel caso un sistema di crittografia venga violato, la cosa si venga (rapidamente) a sapere; è evidente che, se un'eventualità del genere si verifica, chi riesce a violare il sistema ha tutto l'interesse – salvo rare eccezioni – a mantenere segreta la propria scoperta, in modo da trarne il massimo vantaggio.

[14]Dietro al principio citato si può individuare il presupposto che è alla base, per esempio, dell'interesse per le realizzazioni *open source*, ossia la considerazione che quando un oggetto (software, algoritmo, pubblicazione) è a disposizione del libero utilizzo e della critica di tutti, si ha la maggior probabilità che ne vengano rapidamente messi in luce i difetti.

[15]È questo uno degli scopi che la metodologia ISO 15408 si prefigge di raggiungere, tramite un approccio organico a tutte le parti del sistema in esame.

[16]Comprendendo nel termine *server* tutto l'insieme dei componenti necessari, quali Database Server, Application Server, Clustering Management, Console di gestione.

[17]Questo ragionamento può essere esteso a larga parte degli elementi che costituiscono la struttura che viene complessivamente denominata *Internet*; in tutti i componenti che di essa permettono il funzionamento, l'attenzione alla sicurezza, riservatezza e protezione dei dati non sono mai state al centro di una progettazione organica, bensì di volta in volta aggiunte a posteriori.

[18]Con questo termine si intende la combinazione di: (a) connettività Internet; (b) software client `http`; (c) capacità di rendering HTML; (d) compatibilità con SSL/TLS.

[19]Attualmente la tecnologia U3 supporta unicamente sistemi operativi Microsoft Windows™; non è noto se una versione compatibile con *Linux* e *MacOS™* sarà resa disponibile e quando.

[20]Nella fattispecie, si tratta della clientela dei servizi di remote banking *corporate*, contrapposta alla clientela ben più vasta e variegata dei servizi *consumer*.

[21]La classe di dispositivi che meglio si può adattare all'implementazione di una simile variante del TOE è quella dei telefoni cellulari con funzioni di "palmare"; in questi apparecchi è infatti presente, sotto forma di *SIM*, un dispositivo Smartcard che può utilmente impiegato per l'implementazione dei protocolli QPK.

[22]Questa affermazione non può essere ripetuta per tutti gli altri sistemi di protezione delle applicazioni Web, in quanto non si riesce mai ad avere nozione – fino a fatti avvenuti – di essere sottoposti ad un attacco, sia esso riuscito o fallimentare. Si pensi alla facilità con cui per esempio una lista di stralcio può essere fotocopiata, o in altro modo duplicata, senza che il legittimo proprietario ne venga a conoscenza.

[23]Nessuna modifica viene prevista per il codice delle applicazioni, rendendosi necessario solamente un intervento sulla configurazione dei Web server tramite l'installazione di un filtro. Si vedano in dettagli nel capitolo 6.

[24]Il termine è inteso qui nel senso più ampio, includendo quindi il supporto tecnico, l'help-desk, la logistica, l'approvvigionamento.

Capitolo 3

Algoritmo OTP

Introdurre un protocollo efficiente per l'identificazione dell'utilizzatore, evitando tutte le vulnerabilità connesse con l'utilizzo di una canale di comunicazione non sicuro, è obiettivo primario del TOE (si veda 1.1.1). A tale scopo viene proposto l'utilizzo di un algoritmo proprietario denominato QPK-OTP[1], nel quale risultati già noti e consolidati in campo matematico vengono applicati alla crittografia in modo innovativo. Nel presente capitolo, diversamente che nel resto della trattazione, viene utilizzato il termine *password* anziché *PIN*, per omogeneità con la terminologia utilizzata nei documenti forniti dai proponenti del TOE.

3.1 Scopo funzionale

Scopo di QPK-OTP è di permettere al fornitore del servizio di stabilire con certezza l'identità dell'utilizzatore. Considerando che il compito di accertare tale identità è reso non banale dalla natura non affidabile e non sicura del mezzo di trasmissione dei dati (rete pubblica), l'algoritmo deve essere in grado di garantire le seguenti proprietà:

1. L'utilizzatore deve essere in grado di dimostrare la propria identità combinando un elemento *conosciuto* (password) con un elemento *posseduto* (dispositivo U3)[2].

2. La password e le informazioni contenute nel dispositivo U3 devono rimanere private, conosciute solamente dall'utilizzatore e dal fornitore del servizio, in modo da non poter essere sottratte durante la loro trasmissione su canali non sicuri.

3. I messaggi che vengono generati e inviati dall'utilizzatore al fornitore del servizio per provare la propria identità (informazioni *pubbliche*) devono:

   (a) essere completamente scorrelati dalle informazioni private che vengono utilizzate per produrli[3];

   (b) essere completamente differenti[4] ogni volta che l'identificazione viene ripetuta, indipendentemente dal fatto che la password e le altre informazioni private siano cambiate o rimangano costanti. Da questa

proprietà l'algoritmo prende il proprio nome OTP (*one time password*, "password utilizzata una volta sola"), e si noti che in questo contesto il termine *password* viene usato in senso lato, ad indicare *il messaggio generato a partire dalla password [e dalle altre informazioni private]*;

(c) essere completamente differenti, ad ogni ripetizione dell'identificazione, per ogni ammissibile password che l'utilizzatore possa utilizzare[5];

(d) essere generati in modo diverso per ogni utilizzatore del servizio, in modo che i messaggi generati da utilizzatori diversi che casualmente hanno scelto password uguali, siano in ogni caso differenti. In caso contrario, un utilizzatore legittimo del sistema potrebbe impersonarne un altro semplicemente conoscendone la password, ciò che viola la proprietà descritta al punto 1.

4. L'utilizzatore deve essere in grado, in qualsiasi momento, di cambiare la propria password.

3.2 Basi matematiche

3.2.1 Formalizzazione del problema dell'identificazione

Rispettando le proprietà enunciate nella descrizione degli scopi dell'algoritmo di identificazione (si veda 3.1), è possibile definire in termini più formali il funzionamento di uno schema OTP.

Per iniziare, si consideri lo scenario in cui un utilizzatore A debba dimostrare la propria identità al fornitore del servizio, denominato B. Naturalmente A conosce la propria password, denominata x_A; dal momento però che tale password non deve essere trasmessa sul canale non sicuro, ci si dovrà accontentare di trasmettere in sua vece il risultato dell'applicazione ad x_A di una certa funzione algoritmica i, contenuta nel dispositivo U3. Si può denominare *documento* il risultato di tale applicazione, e denotarlo con $\mathcal{D}_A$; allora

$$\mathcal{D}_A = i(x_A)$$

Dalla proprietà (3d) è però noto che che la funzione i che l'utilizzatore impiega per generare il proprio documento deve essere specifica a lui solo fra tutti gli utilizzatori; si potrà allora definire tale funzione i_A. Ma poiché infine il documento prodotto dovrà essere diverso ogni volta, sarà necessario che la funzione i_A possa tenere conto di un parametro, detto *contatore*, che viene incrementato ogni volta che si calcola un nuovo documento; tale parametro, identificabile con il *tempo*, può essere denotato t. L'espressione giunge allora alla sua formulazione definitiva:

$$\mathcal{D}_A = i_A(x_A, t)$$

Il documento $\mathcal{D}_A$ così generato viene inviato – insieme al proprio identificativo utente e al valore del contatore t – al fornitore del servizio. Quest'ultimo, per verificare se l'identificazione è legittima (cioé se la password x_A e la funzione i_A utilizzate sono quelle giuste per l'utente A), reperisce dal proprio database i "veri" elementi x_A e i_A, e li utilizza insieme a t per calcolare la "vera" versione del documento, indicata $\mathcal{D}'_A$. Se $\mathcal{D}'_A = \mathcal{D}_A$ l'identificazione è valida, altrimenti non lo è.

3.2.2 Generatori OTP classici

È evidente dalla descrizione formale del procedimento che il punto cruciale di ogni schema OTP risiede nella funzione i. Essa deve possedere, oltre alla già citata proprietà di *iniettività*, una forte caratteristica "*one-way*". Con questo termine si indica una funzione che sia calcolabile in tempo polinomiale, mentre la sua inversa non lo sia; formalmente, si ricerca i tale che[6]:

$$i(\cdot, t) \in \mathcal{P} \quad , \quad i^{-1}(\cdot, t) \in \mathcal{NP} \qquad \forall t \ \in \mathbb{N}$$

Le funzioni i maggiormente utilizzate come generatori OTP sono tutte caratterizzate dal fatto di lavorare su anelli del tipo $\mathbb{F}_p$, con p numero primo (quindi in arimtetica modulare); esse sono raggruppate nelle seguenti categorie[7]:

Generatori congruenti lineari Le funzioni comprese in questa classe generano ogni elemento della successione compiendo una trasformazione lineare sull'elemento precedente, secondo lo schema:

$$x_t = \alpha x_{t-1} + \beta$$

(In questa espressione come nelle seguenti, per semplicità, la struttura dei generatori è esemplificata in dimensione 1).

Generatori congruenti lineari di ordine *k*, detti anche *a scorrimento di registro*. Le funzioni comprese in questa classe prevedono formule di generazione più complesse, nelle quali per la produzione di un nuovo elemento della sequenza vengono utilizzati molti degli elementi precedenti – o addirittura tutti – secondo lo schema:

$$x_{t+k} = \sum_{i=1}^{k-1} \alpha_i x_{t+i}$$

Generatori congruenti esponenziali Le funzioni di questa classe, considerate quelle più sicure[8], calcolano dapprima una successione di valori (si denotino y_n) con uno dei due metodi esposti sopra; poi, fissata una g radice primitiva del campo $\mathbb{F}_p$ utilizzato, definiscono il proprio output come:

$$x_n = g^{y_n}$$

Allo stato attuale delle conoscenze la classe di metodi con le migliori proprietà, come ricordato in nota, è rappresentato dai *congruenti esponenziali*. Purtroppo, però, la loro applicazione diretta è resa difficile dal fatto che la funzione esponenziale presente nella loro definizione obbliga a gestire numeri molto grandi, circostanza che può essere problematica soprattutto se si considera la possibilità di implementare un sistema OTP su un dispositivo limitato quale Smartcard o Token.

3.2.3 Applicazione della teoria dei polinomi primitivi al problema OTP

I proponenti del TOE hanno sviluppato un approccio originale che, combinando risultati della Teoria dei campi di Galois e proprietà dei polinomi primitivi, permette di generalizzare l'approccio del *generatore congruente esponenziale*, conseguendo al tempo stesso due importanti risultati:

1. La complessità di calcolo viene fortemente ridotta, eliminando la necessità di gestire operazioni con numeri molto grandi e mantenendo sotto controllo la dipendenza dalla grandezza del parametro t[9];
2. È possibile calcolare il grado del *polinomio interpolatore di Newton* associato alla funzione i che viene in questo modo creata (ne esistono infinite). Tale proprietà risulta estremamente importante perchè permette di svolgere in maniera forte l'analisi degli attacchi che possono essere rivolti all'algoritmo QPK-OTP.

L'approccio sopra descritto permette quindi di generare infinite funzioni i, specifiche per ogni utilizzatore del TOE; tali funzioni sono l'*informazione privata* che viene distribuita tramite il dispositivo U3. Se la password è una stringa di caratteri ASCII[10] di lunghezza r, allora la generica funzione i è del tipo:

$$i : (\mathbb{Z}_{2^8})^r \times \mathbb{N} \rightarrow \mathbb{F}_p \times \mathbb{F}_p$$

ossia, una funzione che associa ad ogni valore della password e del contatore t una coppia di numeri compresi fra 1 e p. In una prima versione di QPK-OTP, tale coppia di numeri costituisce, insieme all'identificativo utente ed al valore del contatore t, il *documento* $\mathcal{D}_A$ che l'utilizzatore A invia al fornitore di servizio per essere identificato; la cardinalità dell'insieme $\mathbb{F}_p \times \mathbb{F}_p$ rappresenta quindi il numero totale di documenti $\mathcal{D}$ possibili. Un potenziale attaccante – non conoscendo nulla della struttura che governa i valori generati dalla funzione i (vedi 3.2.4) – per tentare di essere identificato illegittimamente può solo inviare dei valori a caso, sperando di "indovinare" quello giusto; le probabilità allora che l'attacco abbia successo sono

$$P_a = |\mathbb{F}_p \times \mathbb{F}_p|^{-1} = \frac{1}{p^2}$$

Per rendere piccolo il valore della probabilità P_a è in prima battuta possibile, evidentemente, scegliere grandi valori di p. Sapendo che il TOE sarà progettato in modo da non permettere più di un ristretto numero di tentativi di identificazione falliti[11], la significanza statistica di simili attacchi viene resa praticamente nulla[12].

È comunque sempre possibile, qualora tale approccio non sia ritenuto sufficiente, ridurre ancora più rapidamente il valore di P_a decidendo di sfruttare, per ogni identificazione, più di un ciclo di contatore; dal momento che ogni ciclo produrrà una nuova coppia di numeri *indipendente dalla precedente*, si vede rapidamente che, notando con w il numero di cicli di contatore utilizzati per ogni identificazione, la probabilità di riuscita dell'attacco diviene:

$$P_a = |\mathbb{F}_p^{2w}|^{-1} = \frac{1}{p^{2w}}$$

e può quindi essere controllata agevolmente.

3.2.4 Analisi degli attacchi

Come si è descritto in (3.2.3), il merito principale della teoria sviluppata per l'algoritmo QPK-OTP è permettere di conoscere il grado del *Polinomio interpolatore di Newton* associato ad ogni funzione i che si è in grado di creare, e

viceversa, con p noto, avere un metodo in grado di costruire tutte le infinite funzioni i aventi polinomio interpolatore di grado uguale a p.

L'importanza di tali risultati risiede nel fatto che il grado del polinomio interpolatore associato alla funzione i indica *quanti valori della funzione i è necessario osservare per poter ricostruire integralmente la struttura della funzione stessa*; più esattamente, se il polinomio interpolatore ha grado p, solo conoscendo $p + 1$ valori distinti della funzione i, insieme coi relativi valori t, è possibile ricostruire la funzione stessa. Poichè ricostruire la funzione è l'unico modo in cui un attacco può essere portato a termine con successo, per dimostrare che l'algoritmo non è violabile deterministicamente nè statisticamente basta garantire che l'algoritmo QPK-OTP non viene mai utilizzato più di un numero m di volte, con $m \ll p$[13].

teorema di interpolazione

3.3 Specificità

Si evidenziano alcune importanti differenze negli scopi e presupposti dell'algoritmo QPK-OTP rispetto ad alcune note implementazioni di protocolli di autenticazione, basati per esempio su certificati digitali ed algoritmi di cifratura asimmetrica:

- Si presume che la password dell'utilizzatore sia resa nota al fornitore del servizio[14]. Tale presupposto potrebbe essere inaccettabile in contesti dove il rapporto utilizzatore–fornitore non fosse univoco per tutte le coppie di soggetti che hanno necessità di identificarsi reciprocamente (ed è infatti proprio in tali contesti che trovano applicazione le strutture *PKI*). Nel caso del TOE la richiesta appare invece sensata, considerando che l'algoritmo è sviluppato esplicitamente per assolvere alle esigenze di una coppia specifica di ruoli, e mai al di fuori di essa.

- L'algoritmo QPK-OTP presenta una "vita utile" nota a priori, sotto forma di numero massimo di utilizzi della funzione di identificazione. La determinazione analitica del valore da assegnare a questo numero massimo, sulle base di parametri quantitativi ben definiti e fondati, costituisce un fattore specifico di rilievo notevole, nei confronti degli algoritmi dove tale grandezza può solo essere stimata con metodi statistici *a posteriori*.

3.4 Punti di attenzione e raccomandazioni

Si valuta che il lavoro teorico alla base dell'algoritmo QPK-OTP sia adeguatamente in grado di garantire le proprietà richieste dallo scopo funzionale dello stesso. Vengono di seguito esposte alcune considerazioni che dovranno essere tenute presenti nelle fasi successive di realizzazione del sistema proposto, per poter garantire il mantenimento dello standard adottato e il raggiungimento degli scopi elencati in (1.1.1).

- Una parte della documentazione fornita dai proponenti del TOE fa riferimento alla possibilità di costruire sistemi OTP caratterizzati dall'utilizzo di una funzione i *comune* a molteplici utilizzatori, in opposizione a sistemi OTP del tipo trattato in (3.2). Dal momento che le proprietà postulate

per il TOE escludono che una simile realizzazione possa rivestire una qualche utilità, si raccomanda di chiarire meglio nella documentazione la portata di tale scelta.

- Nella documentazione fornita a descrizione dell'algoritmo QPK-OTP vengono descritte le proprietà che il polinomio $h(t) \in \mathbb{F}_p[t]$, utilizzato nella definizione della funzione i, deve necessariamente possedere. Nella stessa documentazione viene poi fatta una menzione molto generica del fatto che certe proprietà aggiuntive del polinomio, che non vengono descritte, influenzino il comportamento della funzione i. In considerazione dell'importanza del punto in questione, si raccomanda di mettere a disposizione maggiori informazioni circa il procedimento da seguire per la determinazione di $h(t)$, possibilmente anche allegando le pubblicazioni di Mauer e Shparlinski a cui si fa riferimento nella documentazione.

- Come noto, la legislazione di Stati Uniti e Giappone permette il rilascio di brevetti a copertura di realizzazioni software, mentre tale possibilità è esclusa dal diritto dei paesi dell'Unione Europea[15] [20],[21]. Non è stato possibile, durante la *fase 1* dell'audit, determinare se i risultati su cui si basa QPK-OTP si distacchino in maniera sufficiente dal corpus di conoscenze attualmente già di pubblico dominio; per questo motivo, non si è in grado di valutare se l'algoritmo possa formare oggetto di una richiesta di brevetto. Si raccomanda di svolgere una ricerca in tale senso poiché, se il contenuto di QPK-OTP dovesse risultare brevettabile, e un brevetto dai contenuti simili non fosse ancora stato rilasciato, esisterebbe un rischio concreto che – dopo la diffusione del TOE – qualche persona o organizzazione ne ricostruisse i dettagli tramite *reverse engineering*, e richiesse di brevettarli a proprio nome sotto forma di software. Questa eventualità avrebbe la grave conseguenza di impedire l'esportazione del TOE – o di parti dello stesso contenenti l'algoritmo QPK-OTP – verso i paesi citati in precedenza; il rischio descritto potrebbe essere annullato valutando l'opportunità di depositare richiesta di brevetto per un'implementazione software dell'algoritmo, o in alternativa rilasciandolo nel pubblico dominio tramite pubblicazione.

Note

[1]Come nel caso del protocollo QPK-DYN, in senso stretto con la sigla QPK-OTP si indica una *classe* di algoritmi, all'interno della quale la scelta di opportuni parametri individua singoli specifici algoritmi.

[2]I due elementi di natura diversa garantiscono proprietà complementari: la password, se pure può essere sottratto senza che il proprietario se ne accorga, può per contro essere cambiata; il dispositivo U3, se pure conserva costanti le proprie caratteristiche nel tempo, non può essere sottratta senza che il fatto sia notato. È cruciale rilevare il fatto che il dispositivo U3 non possa essere *clonato* (altrimenti ricadrebbe nella stessa categoria della password). Si noti infine che benché nel presente capitolo si faccia costantemente riferimento al "dispositivo U3", tutti i risultati esposti sono validi anche considerando, in suo luogo, un qualsiasi dispositivo di *storage* sicuro come ad esempio una smartcard; è per altri aspetti del TOE, non per l'algoritmo QPK-OTP, che la tecnologia U3 assume una rilevanza speciale. (Si veda 1.1.2).

[3]Ovviamente, essendo i messaggi prodotti a partire dalle informazioni private in maniera deterministica, una qualche forma di correlazione deve necessariamente esistere; più esattamente, messaggi successivi non devono essere correlati *in un modo ricostruibile dall'esterno*. Questa proprietà non può essere assegnata in termini assoluti in maniera analitica, ma solo verificata *a posteriori* con metodi statistici; si vedano le considerazioni in (4.3).

[4]Intendendo con ciò che i messaggi debbano *ricoprire* nel grado maggiore possibile tutto lo spazio del propri possibili valori.

[5]In termini formali, la funzione utilizzata per produrre tali messaggi deve essere *(computazionalmente) iniettiva*; si veda più oltre, (3.2.1).

[6]Si noti che la congettura che $\mathcal{P} \neq \mathcal{NP}$, oltre a non essere provata, risulta essere fuori dalla portata dimostrativa di vaste classi di metodi della logica matematica.

[7]Le categorie qui rappresentate coincidono con alcune di quelle descritte in (4.2); questo perchè il problema della generazione OTP è un caso specifico del problema generale di generazione di successione pseudocasuali, anche se rispetto a quest'ultimo si danno esigenze meno stringenti nei confronti delle proprietà statistiche che si richiedono alle sequenze prodotte dalla funzione i al variare di t.

[8]Il problema di calcolare l'inversa della funzione "elevamento a potenza" in aritmetica modulare è noto come *logaritmo discreto*; sulla natura "one-way" di tale problema si basano numerose applicazioni crittografiche, come lo scambio di chiavi Diffie-Hellman e l'algoritmo di firma digitale DSA introdotto dal NIST.

[9]Questo significa che non si corre il rischio che il calcolo della funzione i diventi via via più lento *in maniera eccessiva*, al crescere di t. In termini più rigorosi, la complessità computazionale della funzione presenta una dipendenza *polinomiale* da t.

[10]Cioè una sequenza di r numeri compresi fra 1 e 2^8.

[11]Avendo cura di prevedere meccanismi per i quali questo comportamento, che per esempio disabilita un identificativo utente dopo 3 tentativi falliti, non finisca per *penalizzare* l'utente legittimo, che si potrebbe veder negare l'accesso al servizio a causa dei tentativi di attacco perpetrati da un utente illegittimo o attaccante. L'attaccante anzi potrebbe decidere di compiere consapevolmente tentativi errati di identificazione, con lo scopo di impedire l'accesso ad altri utenti (*denial of service*); si vedano le indicazioni in (3.4).

[12]Del resto, la natura organicamente concepita del TOE fa sì che, anche se un attaccante dovesse riuscire a indovinare l'informazione $\mathcal{D}_A$ che gli permette di essere identificato con successo con un certo valore t, questo non gli permetterebbe affatto di essere in una posizione migliore per indovinare il successivo valore di $\mathcal{D}_A$ (al tempo $t+1$). Inoltre, per riuscire a inviare un messaggio facendosi passare per A, l'attaccante dovrebbe ancora riuscire, per la singola ipotetica transazione in cui sia riuscito ad aggirare l'OTP, a violare la cifratura QPK-DYN.

[13]Una volta che il contatore raggiunge il valore m, la specifica funzione i contenuta nel dispositivo U3 non è più utilizzabile. Si potrà quindi, ad esempio, decidere di adottare uno schema nel quale, approssimandosi a questo punto, il TOE fornisce un avvertimento e all'utente viene rilasciata un nuovo dispositivo U3 con una nuova funzione i. In alternativa, senza procedere ad una sostituzione fisica, è anche possibile istituire un protocollo in base al quale, all'approssimarsi del valore m, una nuova funzione i viene automaticamente generata e scaricata sul dispositivo U3, utilizzando il canale di comunicazione che è ancora sicuro. Per una discussione di queste alternative di implementazione, si veda (5.1.2).

[14]E del resto non può che essere così, data la proprietà di iniettività della funzione generatrice di OTP.

[15]Con la possibile eccezione (parziale) del Regno Unito, che come in molti altri casi distacca la propria prassi da quella dei paesi dell'Europa continentale, in favore di un approccio mag-

giormente simile a quello statunitense. Si veda `http://en.wikipedia.org/wiki/Software_-patents_under_United_Kingdom_patent_law`.

Capitolo 4

Algoritmo DYN

L'algoritmo proposto dal TOE per la cifratura delle informazioni riservate su canale insicuro è una variante della famiglia QPK-DYN; tale variante prende il nome QPK-DYN-1Apq dal nome dei dati $\{A, p, q\}$ che ne formano i parametri principali[1]. Nel corso di questo capitolo, a meno di possibili ambiguità, l'algoritmo sarà sempre designato con QPK-DYN.

4.1 Scopo funzionale

Scopo di QPK-DYN è garantire la riservatezza dei dati trasmessi, realizzandone una cifratura dotata delle seguenti caratteristiche:

1. Immunità agli attacchi di tipo statistico. Partendo dal presupposto che un possibile attaccante sia a conoscenza di alcune informazioni generali del messaggio in chiaro[2], occorre che le informazioni relative alla distribuzione e frequenza dei valori di ogni byte trasmesso siano opportunamente "annegate" entro valori *normali* di quelle quantità.

2. Resistenza agli attacchi basati sulla ripetizione. Potendosi attendere che spesso il testo in chiaro del messaggio che viene scambiato sia lo stesso (cfr. nota 2), si desidera che il testo cifrato appaia invece totalmente diverso di volta in volta. Questa proprietà determina poi anche quella, più debole, per cui a testi in chiaro "quasi" uguali (e cioé con minime differenze) debbano corrispondere testi cifrati "molto" diversi. Il senso esatto dei termini *quasi* e *molto* in questo capoverso sarà stabilito più oltre, vedi (4.2.1), punto 3.

3. Robustezza nei confronti degli attacchi basati sull'accumulo di informazioni. Dal momento che il TOE prevede uno scenario dove i parametri $\{A, p, q\}$, considerati chiave privata, abbiano una durata operativa di diversi anni e quindi di migliaia di transazioni, si ricerca una soluzione nella quale l'eventuale osservazione di un numero elevato di campioni di testo cifrato (conoscendo il corrispondente testo in chiaro) non renda maggiormente probabile l'individuazione della chiave privata[3].

Uno dei risultati più noti della crittografia risiede nella definizione delle caratteristiche dell'algoritmo di cifratura noto come *Cifrario di Vernam*. Tale

algoritmo prevede che due soggetti, prima di iniziare lo scambio di messaggi cifrati, siano *già* in possesso di una determinata chiave di cifratura di lunghezza pari a quella dei messaggi da scambiare; tale chiave di cifratura deve consistere in una sequenza casuale[4] di simboli (lettere, bytes), e deve essere tenuta assolutamente segreta. Con questi presupposti, è sufficiente che i due soggetti utilizzino sul messaggio in chiaro e la chiave una funzione binaria *idempotente se applicata 2 volte*, ossia tale che si abbia

$$f(\mathbb{F}_2) \to \mathbb{F} \quad : \quad f(f(a_1, a_2), a_2) = a_1 \quad \forall (a_1, a_2) \in \mathbb{F}_2$$

Inoltre, è di vitale importanza che la parti di chiave già utilizzate non vengano mai impiegate una seconda volta.

È stato dimostrato da E.C. Shannon che il Cifrario di Vernam offre la maggiore sicurezza possibile (ossia non è attaccabile dalla crittoanalisi), e che ogni sistema di cifratura inattaccabile deve essere equivalente al Cifrario di Vernam (si veda [4]). Tale sistema non è agilmente utilizzabile, ed è normalmente impiegato solo in applicazioni particolari, a causa delle caratteristiche della chiave privata; quest'ultima, infatti, deve essere creata e poi distribuita in maniera sicura a tutti i soggetti che dovranno comunicare. Qualsiasi indebolimento della riservatezza della chiave durante tutto il procedimento rende il protocollo del tutto privo di valore; il problema è poi reso peggiore dal fatto che la chiave, come si è visto, presenta un'estensione pari a quella dei messaggi che dovranno essere trasmessi.

Il risultato di Shannon permette però di rivolgere l'attenzione verso un approccio particolarmente interessante al problema di trovare un algoritmo di cifratura "quasi-Vernam": se si è in grado di produrre, a partire da determinati dati iniziali e utilizzando parametri *piccoli*[5] che sono tenuti segreti, sequenze arbitrariamente lunghe dotate di buone proprietà di casualità[6], allora si potrà realizzare un protocollo di cifratura che si approssima alla massima robustezza possibile (che, come si è visto, coincide con la inattaccabilità alla crittoanalisi). L'algoritmo QPK-DYN svolge esattamente questa funzione; applicando gli strumenti matematici descritti in (4.2), esso assolve il compito di generare una sequenza pseudocasuale della stessa lunghezza ℓ del messaggio che deve essere cifrato.

nozione di test del carattere di casualità

4.2 Basi matematiche

Le tecniche tradizionalmente utilizzate nel calcolo numerico per la generazione di numeri pseudocasuali possono essere distinte in due classi:

Generatori congruenti lineari (LCG) Negli algoritmi compresi in questa classe ogni elemento della successione pseudocasuale è prodotto compiendo una trasformazione lineare sull'elemento precedente, del tipo:

$$x_n = \alpha x_{n-1} + \beta$$

Operando su un campo come $\mathbb{Z}_p$ (ossia in aritmetica modulare), con p numero primo, si ottiene una non linearità che rende difficile l'individuazione dei fattori α e β. Algoritmi di questo tipo sono ampiamente utilizzati, per esempio, nell'implementazione della funzione "*random*" in numerosi sistemi operativi.

Generatori a scorrimento di registro (FSR) Gli algoritmi compresi in questa classe prevedono formule di generazione più complesse, nelle quali per la generazione di un nuovo elemento nella sequenza pseudocasuale vengono utilizzati molti degli elementi precedenti – o addirittura tutti – secondo lo schema:

$$x_{n+k} = \sum_{i=1}^{k-1} \alpha_i x_{n+i}$$

Anche gli algoritmi FSR lavorano in aritmetica modulare, e sono ampiamente studiati. I migliori di questi algoritmi sono in grado di produrre successioni con proprietà pseudocasuali molto buone, ma purtroppo presentano lo svantaggio di diventare via via più lenti in modo molto rapido al crescere della lunghezza della sequenza da generare (si veda [8]).

In tempi più recenti, diversi studiosi – fra cui il Prof. Accardi, che compare fra gli autori del TOE – hanno inaugurato una nuova direzione nello studio della generazione di successioni pseudocasuali, basato sulla teoria dei sistemi dinamici caotici[7], e più in particolare sulla *teoria ergodica*. I lavori svolti in questo campo puntano sostanzialmente nella direzione di una nuova generalizzazione dei concetti della teoria ergodica, sì da poterli applicare al calcolo numerico. La classe di algoritmi QPK-DYN consiste nell'applicazione di alcuni risultati di questi studi al problema della generazione di successioni pseudocasuali, considerate tali in base alle proprietà necessarie per un utilizzo in crittografia.

4.2.1 Generalizzazione della teoria ergodica

I lavori della teoria ergodica classica contengono già molti risultati che permettono di valutare in maniera precisa la "caoticità" delle orbite[8] prodotte da un sistema dinamico nella sua evoluzione. Tali risultati – fra i quali la *gerarchia ergodica*, che permette di classificare il grado di caoticità di un sistema – non sono però applicabili direttamente ai problemi di generazione di successioni pseudocasuali, a causa della finitezza degli insiemi con cui si opera. Per questi motivi ci si è per molto tempo arrestati di fronte al risultato della teoria ergodica classica, per cui i sistemi operanti con insiemi finiti sono in grado di produrre solamente orbite di tipo periodico, prive quindi di interesse nel caso specifico.

gerarchia ergodica

Lo studio numerico delle orbite di sistemi dinamici instabili è però continuato nonostante le limitazioni sopra descritte[9], portando alla constatazione *a posteriori* che le orbite prodotte da certi sistemi, benché periodiche, possiedono molteplici fra le proprietà di casualità più utilizzate. Fra questi sistemi, particolare interesse è stato riservato [9][8] agli *endomorfismi iperbolici del toro n-dimensionale*, noti anche – nel caso 2-dimensionale – con il termine anglosassone di "cat maps" o "Sinai-Arnold maps"; si tratta di sistemi la cui azione è associata ad una matrice A dotata di peculiarità specifiche.

I lavori di Accardi *et al.*, che a quanto consta attualmente sono i soli ad affrontare lo studio di questi sistemi in una prospettiva applicata alla crittografia, stabiliscono alcuni risultati iniziali nella direzione di una "teoria ergodica delle orbite periodiche". Più in dettaglio, questi risultati assicurano:

teoria ergodica delle orbite periodiche

1. Che è possibile costruire matrici A e scegliere il vettore iniziale x_0 (punto iniziale dell'orbita) in modo da ottenere orbite periodiche di lunghezza arbitraria[10].
2. Che l'orbita ottenuta utilizzando i parametri $\{A, x_0\}$ avrà una distribuzione tanto più uniforme quanto più lungo sarà il periodo dell'orbita stessa.
3. Che minime variazioni nel valore del vettore iniziale x_0 producono orbite completamente differenti (questa proprietà è sovente considerata la "firma inequivocabile" del comportamento caotico).
4. Che – poiché per ogni orbita caotica "vera" (ossia non periodica) è possibile costruire un'orbita periodica che la approssima con precisione arbitraria – è ragionevole attendersi che le orbite periodiche esibiscano proprietà statistiche di casualità che tendono a raggiungere quelle delle orbite non periodiche. Benché, allo stato attuale delle conoscenze, questa ipotesi non disponga di una dimostrazione formale, essa è supportata dai risultati dei test statistici più noti (Si veda 4.3).

4.2.2 La variante QPK-DYN-1Apq

Sebbene la trattazione esposta in (4.2.1) dimostri che un'ampia indagine teorica sia a fondamento delle buone proprietà statistiche di casualità prodotte dai sistemi dinamici di tipo $\{A, x_0\}$, un'analisi degli attacchi condotta ipotizzando le peggiori condizioni possibili[11] porta a considerare la necessità di un ulteriore raffinamento dell'algoritmo, dal momento che in questa situazione particolare il sistema proposto non sarebbe significativamente più sicuro di altri algoritmi già noti, permettendo un attacco di "forza bruta" di tipo probabilistico. La variante effettivamente proposta nel TOE, di conseguenza, contiene effettivamente *due* sistemi dinamici del tipo sopra descritto, caratterizzati dal fatto di operare ciascuno su campo numerico differente. Questa variante è anche nota come "versione a due matrici", anche se in senso leggermente improprio; non vengono infatti utilizzate due matrici diverse, ma la stessa matrice A viene utilizzata due volte, per operazioni su campi numerici diversi.

Formalmente, quindi, si dirà che l'algoritmo QPK-DYN-1Apq è caratterizzato dalla quaterna $\{A, x_0, p, q\}$, dove p e q sono reciprocamente primi; l'output del sistema dinamico sarà ottenuto combinando tramite XOR le orbite del sistema $\{A, x_0\}$ su $\mathbb{Z}_p$ e quelle del sistema $\{A, x_0\}$ su $\mathbb{Z}_q$. Questo accorgimento, che non altera le caratteristiche statistiche già note per la versione "a matrice singola", ha l'effetto di distruggere la possibilità (basata su metodi puramente probabilistici) di attacco sopra descritta, dal momento che l'attaccante si troverebbe a dover ipotizzare come ugualmente probabili *tutte* le possibili successioni pseudocasuali prodotte dal sistema, e non sarebbe quindi in grado di ricavare alcuna informazione utile dal proprio attacco[12].

4.3 Tests statistici

Dallo stato attuale delle conoscenze teoriche, come descritto nella sezione (4.2), risulta evidente l'importanza rivestita dai tests statistici nella valuta-

zione del grado di "(*pseudo*)casualità" di una successione. Sebbene, nel caso specifico dell'algoritmo QPK-DYN, ipotesi *fortemente probabili* siano alla base dei procedimenti proposti, una valutazione quantitiva tramite metodi statistici *a posteriori* permette di ottenere, se non una conferma delle ipotesi stesse[13], un raffronto dettagliato con le proprietà esibite da successioni le cui caratteristiche di casualità sono note. In questo senso, l'applicazione dei tests permette di stabilire una classificazione delle proprietà di casualità delle successioni che si hanno a disposizione; ripetendo i tests per un numero statisticamente significativo di volte, le proprietà misurate possono essere ricondotte (ed assegnate) ai meccanismi – o algoritmi – che le hanno generate.

Pur senza avere conoscenza dettagliata dei meccanismi generativi sottostanti, è allora possibile confrontare le successioni prodotte dall'algoritmo allo studio con quelle prodotte, per esempio, da fenomeni naturali[14] o da altri algoritmi già studiati.

I tests agiscono sulla base di un meccanismo di *verifica di ipotesi*; si parte dall'ipotesi che la successione in esame sia stata prodotta da un processo casuale in senso markoviano, e si derivano in maniera analitica i risultati numerici che il test, applicato alla successione, dovrebbe dare (risultato atteso). Dal raffronto fra il risultato atteso e i risultati calcolati su molti campioni di successione si ottiene – con metodi statistici – una misura numerica che può essere utilizzata per la classificazione dell'algoritmo che genera le successioni.

4.3.1 Fonti dei tests

Tests statistici per l'analisi delle successioni casuali sono stati proposti da diverse fonti, in diversi formati. A partire dal lavoro fondamentale di D. Knuth [1], che descrive una serie di metodi e di algoritmi, sono poi stati sviluppati e implementati numerosi tests, raccolti in *suites* spesso citate ed utilizzate. Le più note di queste sono:

Diehard Sviluppata da G. Marsaglia, questa raccolta contiene tests concepiti per misurare proprietà di casualità piuttosto complesse; sebbene molto estensiva e completa, la suite *Diehard* non è direttamente indirizzata all'analisi delle proprietà di casualità che sono rilevanti ai fini della crittografia.

NIST STS Sviluppata dal *National Institute of Standards and Technology* con il fine principale di integrare le debolezze di *Diehard*, questa raccolta contiene un numero maggiore di tests a basso livello, che mirano a individuare quelle strutture – all'interno delle successioni pseudocasuali – che ne indeboliscono l'utilizzo ai fini crittografici.

Appare chiaro dagli studi condotti (si veda ad esempio [12]) che i contenuti di queste due raccolte di tests sono complementari; questa considerazione, unitamente ad altre di carattere strutturale[15], è alla base dell'introduzione di una più recente suite denominata *Dieharder*. Questa, che nasce fin dall'inizio sotto la licenza GNU che ne garantisce la disponibilità e modificabilità in formato sorgente, presenta una serie di importanti innovazioni. Da un lato essa contiene tutti i tests previsti dalla suite *Diehard* e si pone come obiettivo quello di integrare via via i contenuti della NIST STS; dall'altro, la sua natura modulare ed estensibile permette a chiunque ne abbia necessità di aggiungere

alla raccolta nuovi tipi di test, rimanendo costante l'impianto strutturale e statistico di base (ciò che garantisce la confrontabilità dei risultati).

Inoltre *Dieharder* prevede come modalità operativa standard che l'algoritmo di generazione in esame sia *direttamente integrato* nel software di test, e da questo opportunamente richiamato e pilotato. Questo approccio mostra come la suite sia maggiormente diretta alla verifica delle proprietà *dell'algoritmo in sé*, più che di singole successioni[16]. Sulla base di queste considerazioni, la valutazione delle proprietà del TOE è stata svolta applicando i tests di *Dieharder*.

4.3.2 Risultati dei tests applicati alle successioni prodotte dall'algoritmo QPK-DYN

Per valutare i risultati dell'applicazione dei tests *Dieharder* al TOE si è innazitutto anzitutto raccolto l'output di un certo numero di esecuzioni della suite completa; si è poi proceduto, avendo a disposizione una implementazione in codice sorgente dell'algoritmo QPK-DYN, a ripetere separatamente i tests, rilevando una sostanziale omogeneità nei risultati ottenuti[17].

Tali risultati possono essere classificati come *generalmente buoni*, innanzitutto perché la maggior parte dei tests viene passata di larga misura, ossia con valori di p lontani dalle soglie di attenzione. Più nel dettaglio, concentrando la propria attenzione sui risultati dei singoli tests, è possibile osservare che l'algoritmo QPK-DYN ottiene risultati leggermente migliori di quelli riportati da R.G. Brown stesso in [12] come tipici di un *buon* algoritmo (`mt19937_1999`).

Evitando di soffermarsi solo sui risultati generali dei tests, ma analizzando anche i valori dei vari parametri che comportano brusche variazioni nelle statistiche osservate, è possibile trarre alcune conclusioni più circostanziate, che si auspica possano indirizzare ulteriori raffinamenti di QPK-DYN.

Occorre innanzitutto rilevare la circostanza che sia `mt19937_1999` sia QPK-DYN falliscono in maniera chiarissima il test "*Bit Distribution*" per n-uple di lunghezza pari o maggiore di 6, mentre ottengono buoni risultati per tutte le lunghezze fino a 5. Questa circostanza significa che se si considera la sequenza pseudocasuale prodotta come costituita da una successione di valori a 5 bit, allora i valori che questa assume sono uniformemente distribuiti su $\mathbb{Z}_{2^5}$, mentre se si riuniscono i bit in gruppi di 6 o più, la distribuzione assunta per esempio in $\mathbb{Z}_{2^6}$ non è più uniforme. Si suggerisce che, tenendo conto di questa caratteristica delle successioni prodotte, sia possibile costruire delle regole in grado di ottenere distribuzioni uniformi sui campi di caratteristica desiderata[18].

Un altro test che viene fallito in maniera sostanzialmente identica dai due algoritmi è *OPERM5*; in termini intuitivi, ciò indica che benchè – sul campione di 1000000 valori generati – nella successione siano rappresentate con la "giusta"[19] proporzione tutte le permutazioni di 5 interi successivi, la *regolarità* con cui esse alternano intervalli di prevedibilità ad altri di imprevedibilità è indice dell'esistenza di una correlazione all'interno della successione generata.

In conclusione, si valuta che i risultati dei tests analizzati siano compatibili con le caratteristiche provvisoriamente assegnate all'algoritmo QPK-DYN sulla base di ipotesi teoriche, e di conseguenza non vengono sollevate obiezioni al suo impiego nel TOE; inoltre, assume particolare rilievo il fatto che l'algoritmo

proposto dal TOE ottenga risultati *sicuramente non peggiori* di quelli riportati da un algoritmo ben analizzato come `mt19937_1999`. Stimando in ogni caso che lo strumento *Dieharder* permetta indagini estremamente sottili di comportamenti anche molto specifici degli algoritmi di generazione, e ritenendo quasi certo che la novità dell'algoritmo QPK-DYN lasci ancora spazio per attuarne ulteriori raffinamenti, si raccomanda di approfondire lo studio nella direzione degli elementi raccolti nella presente sezione; da questo approfondimento ci si aspetta possa scaturire, semplicemente, una migliore comprensione dei meccanismi sottostanti a QPK-DYN, oppure anche un suo potenziamento.

4.4 Specificità

L'approccio utilizzato dall'algoritmo QPK-DYN raggiunge gli scopi descritti in (4.1), al pari di altri algoritmi esistenti nella letteratura e nell'industria (si veda ad esempio [5]); contiene però, rispetto ai sistemi maggiormente noti, alcune importanti specificità:

- L'algoritmo utilizza solamente operazioni aritmetiche elementari svolte utilizzando numeri interi; sono quindi possibili implementazioni molto efficienti su qualsiasi microprocessore *general purpose*, anche dotato di potenze di calcolo modeste.
- La creazione delle chiavi private $\{A, p, q\}$ può essere fatta mediante un'elaborazione dalla complessità modesta, alla portata di normali Personal Computer; in particolare, i numeri primi p e q possono essere, ad esempio, $< 2^{31}$. Questo punto è molto importante perchè permette di non dipendere da enti esterni (in grado di avere accesso a supercalcolatori) per poter garantire la robustezza dell'algoritmo[20].
- L'algoritmo QPK-DYN viene combinato, nel TOE, con l'algoritmo QPK-OTP che presiede all'identificazione della controparte prima di ogni scambio di messaggi (si veda 3.1); il fatto che i due algoritmi siano sviluppati nel medesimo ambito permette di fare in modo che le caratteristiche degli stessi siano reciprocamente adattate, ottenendo una robustezza ottimale. In modo analogo, la conoscenza *a priori* dell'ambiente tecnologico nel quale l'algoritmo sarà impiegato ha permesso di progettare alcuni dettagli di quest'ultimo in maniera confacente al primo.

4.4.1 Controllo dell'integrità

L'algoritmo QPK-DYN non comprende alcuna funzione per il controllo dell'integrità dei messaggi che, dopo essere ricevuti, vengono decifrati. Poichè nessun altro elemento nel TOE è predisposto ad asservire a questo scopo, risulta indispensabile prevedere l'aggiunta di un meccanismo di tipo MAC (*Message Authentication Code*) a copertura del rischio di alterazione dei messaggi durante la trasmissione[21].

Occorre notare che la particolare costruzione dell'algoritmo QPK-DYN fa sì che questo esibisca un'elevata resistenza al rumore sul canale trasmissivo; con questa proprietà si intende indicare che, se del testo cifrato vengono alterati n bytes, anche dopo la decifrazione i bytes che compariranno alterati[22]

sono n. Tale caratteristica, se pure desiderabile in senso generale, introduce paradossalmente una vulnerabilità più importante in un'applicazione come quella proposta dal TOE; se si immagina infatti che un attaccante sia in grado di sapere esattamente in quale esatto punto del messaggio compaia un byte che egli desidera alterare[23], questi potrà svolgere la propria azione – modificando in maniera casuale il singolo byte – senza che nel resto del messaggio si producano anomalie che potrebbero (meglio) evidenziare che lo stesso è stato compromesso.

Di questa particolarità si raccomanda di tenere particolarmente conto al momento dell'integrazione dell'algoritmo QPK-DYN all'interno dello struttura complessiva del TOE.

4.5 Punti di attenzione e raccomandazioni

Si ritiene che le analisi e i tests condotti dai proponenti del TOE forniscano una dimostrazione sufficiente del fatto che l'algoritmo QPK-DYN esibisca le proprietà necessarie. Vengono di seguito esposte alcune considerazioni che dovranno essere tenute presenti nelle fasi successive di realizzazione del sistema proposto, per poter garantire il mantenimento dello standard adottato e il raggiungimento degli scopi elencati in (1.1.1).

- I primi risultati ottenuti dall'applicazione della teoria ergodica alle orbite periodiche lasciano spazio a molteplici futuri approfondimenti e precisazioni; se da un lato i risultati più importati su cui fonda QPK-DYN sono legati a teoremi già dimostrati – e non ci si attende quindi che siano posti in discussione – dall'altro è pur sempre possibile che avanzamenti teorici in questo campo rendano in futuro attuabili tipi di attacco all'algoritmo che oggi non sono ipotizzabili o attuabili[24]. Si raccomanda quindi di svolgere un'attività di periodica revisione dei progressi teorici raggiunti, in modo da poter individuare tempestivamente eventuali correzioni o integrazioni da apportare al TOE per mantenerne valide le proprietà oggi osservate.

- La letteratura esaminata e la documentazione messa a disposizione lasciano sostanzialmente indeterminato l'approccio concretamente usato, all'inizio dell'algoritmo QPK-DYN, per la scelta del vettore iniziale x_0; i risultati forniti in [11], infatti, illustrano le proprietà che tale vettore iniziale deve possedere per garantire le caratteristiche dell'orbita desiderate. In questa fase dell'audit questa informazione è sufficiente; si raccomanda però di specificare questo punto in maniera algoritmica al momento di intraprendere l'implementazione del TOE, di modo che di quest'ultima possa essere più agevolmente verificata l'esattezza.

- Dal momento che le orbite prodotte da un sistema $\{A, x_0, p\}$ sono periodiche, a partire dal momento in cui un vettore x_n della successione prodotta coincide con x_0, la successione inizia a ripetersi. Questa circostanza deve chiaramente essere evitata, perchè introdurrebbe nella successione pseudocasuale una evidente autocorrelazione che ne distruggerebbe molte proprietà statistiche[25]. Il documento [13] prevede, al punto 5, un esplicito controllo dell'eventuale ripetizione dell'orbita; si ritiene che tale punto debba essere meglio specificato relativamente ai seguenti aspetti:

1. Dal momento che il documento, che contiene la descrizione algoritmica di QPK-DYN, non indica metodi di condizionamento del vettore iniziale x_0, non è chiaro se si preveda, tramite un simile condizionamento da aggiungere in un secondo tempo, di evitare del tutto la necessità del controllo di ripetizione (ad esempio conoscendo in anticipo il periodo massimo di orbita necessario per la cifratura dello specifico messaggio che si sta elaborando).
2. Nel caso si decida di mantenere operante il controllo descritto più sopra al punto (1)[26], l'operazione di "random kick" rappresentata da r_k viene descritta come *una delle possibili*, senza specificare se la scelta di questa o di altre possibili funzioni analoghe sia indifferente o possa invece influire sul comportamento successivo dell'orbita (in analogia all'influsso operato dalla opportuna scelta del vettore iniziale x_0).

- Il ruolo e l'importanza rivestiti dai tests statistici è ampiamente illustrato in (4.3); si ritiene che la scelta effettuata di servirsi della suite *Dieharder* sia ben motivata e corrispondente agli scopi del TOE. Peraltro, riconoscendo la natura di "work in progress" di tale raccolta di tests, e il fatto che al momento essa si fondi principalmente sulla suite di Marsaglia – non concepita specificamente per la verifica di proprietà rilevanti per la crittografia – si raccomanda di sottoporre periodicamente l'algoritmo ai tests della suite, via via che nuovi moduli verranno resi disponibili, con particolare interesse nei confronti di ulteriori tests NIST. Si vedano inoltre le raccomandazioni descritte in (4.3.2).

- Come noto, la legislazione di Stati Uniti e Giappone permette il rilascio di brevetti a copertura di realizzazioni software, mentre tale possibilità è esclusa dal diritto dei paesi dell'Unione Europea[27] [20],[21]. Dal momento che i dettagli dell'algoritmo QPK-DYN non sono disponibili nel pubblico dominio, e poichè non risulta che un brevetto dai contenuti simili sia mai stato rilasciato, esiste un rischio concreto che – dopo la diffusione del TOE – qualche persona o organizzazione ne ricostruisca i dettagli tramite *reverse engineering*, e richieda di brevettarli a proprio nome sotto forma di software. Questa eventualità avrebbe la grave conseguenza di impedire l'esportazione del TOE – o di parti dello stesso contenenti l'algoritmo QPK-DYN – verso i paesi citati in precedenza; si raccomanda, a copertura di questo rischio, di valutare l'opportunità di depositare richiesta di brevetto per un'implementazione software dell'algoritmo, o in alternativa di rilasciarlo nel pubblico dominio tramite pubblicazione.

Note

[1]Questi parametri formano la *chiave privata* del protocollo; esiste la possibilità di aumentare ulteriormente la sicurezza dell'algoritmo mantenendone privati altri elementi. Per esempio, è possibile fissare il vettore iniziale x_0 basandosi anche sul PIN utente, sul numero di serie del dispositivo U3, o più in generale su un output dell'algoritmo QPK-OTP utilizzato per l'identificazione subito prima dello scambio di dati.

[2]La natura sottostante a molti dei messaggi scambiati nell'ambito di un'applicazione Web è infatti nota o facilmente inferibile; in molti casi, infatti, i messaggi aderiscono alla sintassi di linguaggi quali HTML o XML, i quali esibiscono un'elevata ridondanza nel senso di Shannon (si veda [5]). Vi è poi da notare come un possibile attaccante sia in grado di studiare in grande dettaglio le caratteristiche dei messaggi scambiati in un sistema come il TOE, semplicemente divenendo utilizzatore legittimo dello stesso ed attuando un minimo di *reverse engineering*.

[3]Tale risultato non è realizzabile *alla lettera*, dal momento che, essendo l'algoritmo e i dati su cui opera finiti, per definizione anche l'output prodotto esibisce un contenuto finito di informazione; di conseguenza, a patto che siano noti i dettagli dell'algoritmo ed il testo in chiaro di tutti i messaggi cifrati intercettati, un numero finito di osservazioni permette sempre di ricostruire la chiave privata con la probabilità voluta, e quindi di "rompere" l'algoritmo. Un risultato è comunque accettabile quando si possa dimostrare che utilizzando il limitato numero di osservazioni permesso dall'applicazione, la probabilità con cui si riesce a costruire la chiave privata non è migliore che se si tentasse di farlo in maniera casuale. Vedi (4.2.2).

[4]È facile rendersi conto, intuitivamente, che la sequenza deve essere casuale perchè in caso contrario un attaccante potrebbe facilmente ricostruirla e, con essa, decifrare ogni messaggio ed essere in grado di inviarne lui stesso. Occorre però notare che, a differenza di altri ambiti matematici dove determinate proprietà statistiche delle sequenze pseudo-casuali sono necessarie per motivi teorici, nelle sequenze utilizzate con i cifrari di Vernam la cosa importante è che il possibile attaccante *non sia in grado di riconoscere, congetturare o ricostruire* alcuna struttura nella sequenza stessa. In questo senso, quindi, una sequenza "casuale" formata da una stringa di valori '1' ripetuti senza variazione, sarebbe altrettanto valida delle sequenze più complesse che si possano produrre; l'unico fattore discriminante è che si suppone (correttamente) che un avversario "intelligente" sia in grado di tentare *per prime* le sequenze dotate di una struttura altrettanto "intelligente" o "intelligibile". Vale poi la circostanza che sequenze dotate di specifiche proprietà statistiche sono in grado di vanificare gli attacchi che sono basati proprio sulle caratteristiche statistiche del testo cifrato.

[5]Con questo attributo si intende indicare, intuitivamente, parametri la cui memorizzazione richiede uno spazio molto inferiore rispetto a quello necessario per contenere i messaggi che devono essere cifrati. Questa proprietà, unitamente alla possibilità di generare chiavi molto lunghe *riutilizzando* sempre gli stessi parametri, permette di superare la maggior parte delle difficoltà associate all'utilizzo del Cifrario di Vernam. Naturalmente si opera sempre nell'assunzione che ci sia un evento iniziale (e.g. consegna fisica di un dispositivo prima che cominci lo scambio di messaggi) durante il quale i parametri vengano scambiati in maniera sicura.

[6]Non è possibile pervenire ad una definizione univoca della misura di *casualità* di una successione di numeri. I risultati esposti in [7] mostrano che per verificare il carattere di casualità di una successione di numeri sarebbe necessario verificare – tramite appositi test – l'esistenza di un numero infinito di proprietà. Ne consegue necessariamente che ogni test di casualità effettivamente realizzabile (con successioni finite e utilizzando algoritmi finiti) contiene un elemento di arbitrarietà, e la sua validità deve essere considerata in funzione delle specifiche proprietà (finite) che ogni applicazione richiede. Per gli stessi motivi, appare chiaro che nessuna successione numerica generata da un algoritmo reale (finito) può essere definita casuale *tout-court*; viene normalmente utilizzato invece il termine "pseudocasuale". Occorre sempre tenere presente che tale termine non ha una valenza assoluta (al contrario di *casuale*), ma solo in associazione ad uno specifico insieme di test di casualità.

[7]È questo un campo di ricerca che ha dato a partire dagli anni '80 enormi risultati in una serie di campi scientifici a cavallo fra la matematica, fisica, biologia, medicina, astronomia, nonché in moltissimi ambiti tecnologici. Risultati che sono noti anche al di fuori della cerchia degli studiosi direttamente interessati sono quelli ottenuti da Benoît Mandelbrot nella disciplina oggi nota come *geometria frattale*; in questa prolifica branca della matematica vengono studiate le caratteristiche di sistemi dinamici in grado di esibire, malgrado la definizione formale molto semplice, comportamenti infinitamente complessi. La nozione di *sistema dinamico*, ripresa dalla Meccanica Razionale classica e dalla Teoria dei Sistemi, indica un sistema la cui evoluzione nel tempo (output) dipende dagli stati via via assunti dal sistema stesso negli istanti passati (input). La categoria dei sistemi *caotici* contiene quei sistemi nei quali una variazione infinitesimale

nello stato iniziale porta a variazioni finite nell'evoluzione successiva del sistema stesso. Tale caratteristica di "incommensurabilità" fra input e output conferisce ai sistemi caotici – che pure sono, evidentemente, deterministici – caratteristiche di imprevedibilità che possono essere molto utili nella generazione di successioni pseudocasuali.

[8]Con il termine *orbita* si indica l'insieme degli output prodotti da un sistema dinamico a partire da un punto iniziale e prendendo ripetutamente come input per la funzione ϕ costitutiva del sistema gli output via via prodotti.

[9]Per esempio sotto la spinta dello sviluppo dei metodi di integrazione *Monte-Carlo*, che basano il proprio funzionamento proprio su successioni di valori pseudocasuali in $\mathbb{Z}_p^d$, ossia di vettori d-dimensionali interi o razionali.

[10]Naturalmente il vettore iniziale x_0 dovrà contenere una qualche dipendenza da circostanze esterne al messaggio da cifrare e dovrà essere scelto in modo da essere sempre diverso.

[11]Vale a dire, conoscenza da parte dell'attaccante di: (a) dettagli di funzionamento dell'algoritmo; (b) dati in chiaro (per intero); (c) dati cifrati (per intero).

[12]Questa conclusione ha una valenza molto forte: anche nel caso in cui l'attaccante ha potuto osservare quantità arbitrarie di testo cifrato, e conosca tutto il corrispondente testo in chiaro (ad esempio utilizzando tecniche di compromissione dell'integrità del software, che però nel TOE sono rese molto difficili dall'impiego della tecnologia U3 – si veda 5.1), non è in grado di utilizzare queste informazioni, ad esempio, per decifrare messaggi futuri o sostituirsi ad uno dei due soggetti (*impersonation*).

[13]Il campo in rapido sviluppo noto come *Matematica sperimentale*, propone una metodologia nella quale una vasta classe di risultati pratici, ottenuti tramite il calcolo automatizzato, viene analizzata ricercando un'evidenza di strutture; in questo modo, in maniera non dissimile da quanto avviene nelle scienze applicate, i risultati "sperimentali" permettono di abbozzare e raffinare schemi teorici dei quali i risultati ottenuti siano casi specifici. Si veda ad esempio la pubblicazione *Experimental Mathematics*, `http://www.expmath.org`.

[14]I cosiddetti "generatori casuali *fisici*" – nei quali viene utilizzato come sorgente dell'informazione un processo difficilmente prevedibile come il rumore termico o il decadimento radioattivo – sono spesso considerati il *non plus ultra* della generazione di successione casuali. In realtà, analisi statistiche mostrano che le successioni da essi ottenute esibiscono comunque contenuti più o meno evidenti di struttura non casuale, ad esempio sotto forma di correlazioni interne. Questo risultato, se da un lato non deve sorprendere – è infatti evidente che dietro a qualsiasi processo fisico esista un meccanismo deterministico che lo governa – mostra l'estrema sensibilità degli strumenti statistici messi a punto; esso indica anche che la pura casualità, come definita in termini matematici, non esiste nella realtà, con la possibile eccezione dei fenomeni quantistici, dove il principio di Heisenberg "nasconde" alla nostra conoscenza il principio deterministico.

[15]L'inadeguatezza maggiore riscontrata nelle *suites* di test classiche risiede nella loro mancanza di parametrizzazione. In sostanza, questi tests sono concepiti (per l'utilizzo da parte di ricercatori non necessariamente specializzati) in modo da dare un risultato del tipo SÌ/NO, senza fornire indicazioni più sfumate. In contrasto, la metodologia adottata in *Dieharder* permette di variare i parametri utilizzati nel calcolo statistico, così da fornire indicazioni maggiori sul "grado" di superamento o non superamento di ogni test. Lo strumento che ne risulta è maggiormente adatto a guidare l'indagine e il raffinamento degli algoritmi indagati.

[16]le quali potrebbero anche non avere la lunghezza adeguata, o essere prodotte in circostanze e con input non adeguati.

[17]Si ricordi che l'interpretazione del significato dei valori di p forniti dai tests statistici non è né immediata né assoluta; basti ricordare che in alcune circostanze perfino i generatori "fisici" falliscono nettamente determinati tests. Si vedano maggiori dettagli in [12].

[18]Per esempio combinando, sulla scorta ed analogia di alcuni suggerimenti contenuti nel lavoro [12], l'output di due generatori indipendenti e miscelandolo, sia usando la stessa lunghezza di raggruppamento in bit, sia usandone due diverse (≤ 5).

[19]Intendendo con ciò la proporzione che si osserverebbe ammettendo un processo puramente markoviano all'origine della successione.

[20]Situazione opposta si ha nel caso di algoritmi, come per esempio RSA, che fondano la propria robustezza sulla disponibilità di numeri primi molto grandi. Ogniqualvolta elementi necessari al funzionamento di un algoritmo di cifratura non possono essere prodotti *in-house*, si crea il potenziale per una situazione di compromissione della sicurezza stessa, o di manipolazione della domanda/offerta.

[21]È per esempio possibile immaginare di applicare a questo scopo una variante dell'algoritmo QPK-OTP, in congiunzione con una funzione di *hash* tradizionale come MD5 o SHA-1. In questo scenario, dopo aver effettuato l'hash del testo *cifrato*, il valore fornito dalla funzione di hash può essere utilizzato come input x_A di una funzione i_A (si veda 3), producendo un valore di

verifica che viene trasmesso insieme al messaggio cifrato. In questo modo il ricevente, prima di procedere alla produzione della chiave condivisa e quindi alla decifrazione, può procedere in modo identico ed essere quindi certo (a meno della probabilità di collisione della funzione di hash) dell'integrità del messaggio.

[22]Più esattamente, su un canale trasmissivo a 8 bit, per ciascuno degli n bytes la cui immagine cifrata è stata alterata si ha una probabilità pari a $\frac{255}{256}$ che lo stesso risulti alterato.

[23]Non è difficile immaginare che un attaccante, accedendo ad un sistema come il TOE da utilizzatore legittimo, sia in grado di individuare il testo non cifrato dei messaggi – ad esempio HTML – che vengono inviati al fornitore del servizio, e di conseguenza di evincere la struttura (fissa) di quelli.

[24]Questa circostanza, del resto, è vera per *qualsiasi* struttura matematica applicata alla crittografia tradizionale, con la sola possibile eccezione dei metodi della nascente crittografia quantistica. Da questo punto di vista può essere considerato come un vantaggio dell'algoritmo QPK-DYN il fatto di basarsi su un campo di indagine ancora recente e molto meno sviluppato, per esempio, della Teoria dei numeri che sottostà ai problemi di fattorizzazione dei numeri interi.

[25]È pur vero che (a) essendo impossibile che la circostanza sopra descritta si verifichi simultaneamente per i due "sottosistemi" $\{A, x_0, p\}$ e $\{A, x_0, q\}$, le conseguenze della ripetizione sono notevolmente attenuate; (b) nel caso la circostanza si avveri con bassa probabilità, l'utilizzo che un eventuale attaccante ne può fare è estremamente limitato. Cionondimeno, risulta preferibile eliminare completamente la possibilità che la ripetizione si verifichi.

[26]Tale controllo, se pure dovesse essere reso inutile dal punto di vista teorico, potrebbe rimanere in essere per una maggiore robustezza dell'algoritmo dal punto di vista del *software engineering*. Si possono infatti immaginare scenari nei quali la lunghezza del messaggio da cifrare non è nota a priori e non è nota con assoluta certezza; in queste circostanze potrebbe essere rischioso confidare nella lunghezza dell'orbita periodica senza poter garantire – tramite opportuno condizionamento di x_0 – che essa avrà sicuramente almeno la lunghezza desiderata.

[27]Con la possibile eccezione (parziale) del Regno Unito, che come in molti altri casi distacca la propria prassi da quella dei paesi dell'Europa continentale, in favore di un approccio maggiormente simile a quello statunitense. Si veda `http://en.wikipedia.org/wiki/Software_patents_under_United_Kingdom_patent_law`.

Capitolo 5

Client U3-based

Alcune delle caratteristiche che concorrono a formare le proprietà di sicurezza e robustezza del TOE sono garantite dall'utilizzo della tecnologia U3. Tale piattaforma, che specifica un nuovo tipo di dispositivo di *storage* USB denominato *smart disk*, è proposta da un consorzio guidato dalle aziende Sandisk e M-Systems e presenta – rispetto ad un normale dispositivo di memorizzazione di massa – una serie di contenuti aggiuntivi che vengono descritti nelle sezioni che seguono.

5.1 Descrizione della piattaforma U3

Lo scopo dello standard U3 è permettere all'utilizzatore di installare sul proprio smart disk una serie di applicazioni, ed essere in grado di utilizzarle su diversi PC, di volta in volta, senza dover effettuare installazioni e con la garanzia che i files eventualmente copiati sul disco fisso del PC sono rimossi non appena l'applicazione termina o lo smart disk viene rimosso[1]. Le applicazioni contenute nello smart disk, denominate applicazioni *mobili*, vengono ivi installate utilizzando tools appositamente forniti come parte dell'U3 Software Development Kit; durante l'installazione, insieme agli eseguibili ed ai files di supporto specifici dell'applicazione, vengono memorizzati sullo smart disk alcuni files di servizio denominati `manifest`, che descrivono l'ambiente necessario all'esecuzione di quella.

applicazioni mobili

La piattaforma U3 mette a disposizione:

- un software denominato *Launchpad* che viene automaticamente eseguito non appena lo smart disk è inserito in un PC e che permette di visualizzare l'elenco delle applicazioni installate sul dispositivo; il ciclo di vita di queste è controllato dal Launchpad stesso, che si fa carico delle seguenti operazioni:

  (a) Verificare, consultando gli appositi `manifest` creati al momento dell'installazione, se tutti i requisiti richiesti dall'applicazione sono soddisfatti.

  (b) Apportare eventuali modifiche alla configurazione del sistema, necessarie a stabilire un ambiente standardizzato così come l'applicazione richiede di trovarlo. Per esempio, possono essere create voci

nel registro di sistema, create unità disco virtuali, copiati files su una zona temporanea del disco del PC, installate librerie condivise. Quando l'ambiente è preparato, il Launchpad provvede ad attivare l'applicazione.

(c) Eseguire un controllo sulla presenza costante dello smart disk; se si rileva che lo stesso è stato rimosso dal PC, viene avviato un processo di "cleaning" che annulla le operazioni effettuate prima del lancio dell'applicazione. In particolare, vengono rimossi tutti i files copiati sul disco del PC.

- La possibilità di memorizzare informazioni in una parte di memoria protetta, che non è visibile dal sistema operativo come disco.
- Una piccola quantità di memoria ROM che può essere letta utilizzando le specifiche API, e che, notabilmente, contiene un numero di serie la cui unicità è garantita.

Esistono tre differenti tipi di applicazione che possono essere installati su uno smart disk U3 o utilizzati in abbinamento con questo:

U3 Launchpad Application (U3LP) È questo il tipo più semplice dei tre; si tratta di una normale applicazione Microsoft Windows™, che viene installata sullo smart disk e diviene una *mobile application*. Non è necessario modificare un'applicazione esistente per trasformarla in U3LP; è sufficiente verificare che essa non faccia uso di particolari tecnologie o *patterns* di funzionamento non (completamente) supportati[2]. Una volta installata con successo sullo smart disk, l'applicazione U3LP – il cui ciclo di vita è controllato dal Launchpad – è utilizzabile su qualsiasi PC, nello stesso modo e con lo stesso funzionamento dell'applicazione stand-alone.

U3 Launchpad+ Application (U3LP+) Questa tipologia corrisponde ad una normale applicazione U3LP, che attraverso l'interfaccia DAPI *Device API* è in grado però di fare uso di speciali funzioni a basso livello messe a disposizione dallo smart disk. Fra queste funzioni, per esempio, rientra la capacità di accedere ad uno *storage* protetto oppure di rilevare l'identificativo seriale del dispositivo. È in questa categoria che rientra il browser personalizzato proposto nell'architettura del TOE (si veda 5.2).

U3 Aware Application (U3A) Un'applicazione Microsoft Windows™ standard, che *non* viene installata sullo smart disk, ma che utilizza DAPI per accedere alle funzioni di basso livello del dispositivo U3. Per esempio una normale applicazione potrebbe essere resa U3A allo scopo di utilizzare uno smart disk come *dongle* di autorizzazione, oppure un software di backup potrebbe sfruttare lo *storage* protetto del dispositivo, archiviandovi files – eventualmente in forma cifrata per il massimo livello di protezione[3]

5.1.1 Vantaggi

In riferimento agli scopi dichiarati del TOE (si veda 1.1.1), l'utilizzo della piattaforma U3 comporta diversi ordini di vantaggi:

- Dal momento che il TOE prescrive di fornire agli utilizzatori un browser personalizzato per rimediare alle vulnerabilità di quelli tradizionali, si valuta che la piattaforma U3 sia in grado di offrire lo strumento più conveniente e meno rischioso per veicolare tale distribuzione di software. In seconda istanza, l'architettura U3 è abbastanza flessibile – ed il supporto che attualmente riceve dall'industria sufficientemente ampio – per poter accomodare variazioni o estensioni di una certa entità nella struttura interna del TOE.

- La natura di *mobilità* delle applicazioni installate su smart disk permette all'utilizzatore di lavorare spostandosi senza problemi da un PC a un altro[4], superando al tempo stesso il rischio di utilizzare – senza rendersene conto – un browser (o un componente di questo) compromesso.

- La capacità di riunire in un unico dispositivo il veicolo di distribuzione del software e, al tempo stesso, uno *storage* protetto nel quale memorizzare le informazioni private necessarie al funzionamento del TOE, viene valutata nel senso di una positiva e benefica unitarietà negli strumenti e nei metodi impiegati.

5.1.2 Caratteristiche critiche

Pur mantenendo una valutazione positiva nei confronti della tecnologia U3 e dell'impiego che il TOE ne propone, si considera tale tecnologia presenti, a fianco dei vantaggi descritti, una serie di caratteristiche critiche che devono essere tenute sotto controllo, per non compromettere le proprietà del TOE (si vedano anche le indicazioni in 7.4.1).

I punti sui quali si ritiene necessario un intervento di analisi più approfondito sono i seguenti:

- La maggior parte dei dispositivi U3 prodotti attualmente utilizza la tecnologia "flash" con porte NAND per l'implementazione dello *storage*; tale scelta è una conseguenza diretta della stretta parentela generativa che lega gli smart disks ai più tradizionali flash disks. Il fatto però che questa tecnologia comporti una vita del dispositivo limitata a circa 100'000 scritture, la rende meno che ideale per un utilizzo di tipo smart disk; in questo scenario, infatti, le scritture sono molto frequenti, in quanto il sistema *launchpad* deve spesso riportare sul dispositivo flash lo stato dei files modificati dall'applicazione. Si pensi ad esempio, nella realtà del TOE, al continuo aggiornamento del contatore dell'algoritmo QPK-OTP[5]. Il presente punto non costituisce necessariamente un problema, ma deve essere analizzato mediante stime dei carichi di utilizzo e simulazioni su dispositivi reali, in modo da confermare che il TOE come viene consegnato all'utilizzatore abbia effettivamente la vita utile che ci si attende, anche in base ad altre considerazioni.

- Il software *Launchpad* viene automaticamente installato – in maniera permanente – su ogni PC al quale venga collegato uno smart disk U3. Se questo software non è prodotto in forma personalizzata come parte del TOE, ma si decide di utilizzarlo nella versione standard fornita dal produttore[6], si ritiene che sia necessario valutare il rischio che tale software

possa essere – senza presupporre la malafede del produttore – affetto da virus o altrimenti compromesso[7].

5.2 Il browser proprietario

La scelta di sostituire allo strato di sicurezza Internet standard, basato su SSL/TLS, l'implementazione completamente proprietaria consistente nei protocolli QPK-OTP e QPK-DYN, fa sì che si debba proporre un software client per l'accesso alle applicazioni. Dal momento poi che le applicazioni che il TOE si propone di proteggere – almeno in prima battuta – sono applicazioni Web, la scelta di realizzare un browser personalizzato appare sensata e al tempo stesso necessaria.

Durante la valutazione delle analisi preliminari svolte dai proponenti del TOE è stato messo in luce che le strade praticabili per lo sviluppo del browser sono essenzialmente due[8], mutuamente esclusive:

1. È possibile utilizzare un qualsiasi linguaggio di programmazione in grado di creare applicazioni COM (*Active-X*) client, e con questo sviluppare la parte effettivamente connessa alla specificità del TOE, compresi gli algoritmi QPK e l'interfacciamento tramite DAPI al dispositivo U3 (si veda 5.1). Sarà poi sufficiente utilizzare un componente COM server che implementi le funzioni di Web browsing, e demandare a questo tutte le funzionalità standard di parsing, visualizzazione, networking.

2. In alternativa, è possibile accedere al codice sorgente di un moderno browser in licenza *open source*, e personalizzarlo con i contenuti specifici del TOE.

Una valutazione condotta anche tenendo presenti le caratteristiche dell'ambiente operativo U3 nel quale il prodotto dovrà funzionare, porta a raccomandare l'opzione 2, per molteplici motivi:

- Un browser che fosse realizzato come COM client, nei confronti di un componente in grado di svolgere le funzioni di navigazione vera e propria, si troverebbe in una delle seguenti possibili situazioni: (a) essere obbligato a portare con sé il componente COM server adatto; (b) basarsi su un componente COM come quello di Microsoft™, "normalmente" disponibile su ogni PC. Ognuna delle due possibilità presenta inconvenienti di gravità tale da scoraggiarne l'adozione. Il caso *a* è reso di difficile applicazione dal fatto che un componente COM server, prima di essere utilizzato, deve essere registrato nel sistema operativo; tale operazione non è sufficientemente rapida, veloce e robusta da potersi proporre di eseguirla ogni volta che si utilizzi il browser. Inoltre, possono sorgere problemi riguardo alla rimozione di un componente al termine dell'utilizzo del TOE. Il caso *b* è caratterizzato da una fragilità ancora maggiore, perchè il componente cercato potrebbe non esistere sul PC, oppure potrebbe recare una versione non compatibile, oppure ancora essere compromesso da codice malevolo o virus.

- Se il browser viene realizzato personalizzando un software già esistente, è possibile raggiungere un maggiore stabilità. Dal momento infatti che le

uniche applicazioni Web a cui il browser dovrà mai accedere sono quelle della Banca, le cui caratteristiche sono note e controllate, non riveste più un'importanza fondamentale la necessità di aggiornare il motore HTML molto frequentemente; in ritorno, potendosi basare su un blocco di codice che non cambia più (o cambia raramente), è possibile ottenere un software più stabile, del quale è più facile garantire e dimostrare le proprietà indicate dalla metodologia ISO 15408.

Note

[1]Naturalmente la rimozione dei files copiati sul disco fisso può essere garantita solo *in circostanze normali*; nell'eventualità, per esempio, di un *halt* improvviso del sistema ciò non può avvenire. Per questo motivo si dovrà sempre evitare di trasferire sul PC contenuti riservati come sono ad esempio le chiavi private utilizzate dal TOE.

[2]Si veda ([15]); è il caso per esempio delle tecnologie COM, Java™, .NET™, oppure ancora dei software che apportano modifiche alla propria configurazione in un tempo successivo alla propria installazione, per esempio per implementare uno schema *try and buy*.

[3]È questo un tipo di utilizzo che i proponenti del TOE prevedono fra le possibili future applicazioni dell'accoppiata U3-QPK. Va ricordato che su un singolo smart disk possono essere installati numerosi softwares, e che quindi l'utilizzatore può convenientemente impiegare un unico dispositivo per tutte le proprie esigenze di *secure computing*.

[4]Si valuta opportuno non spingere questa considerazione alle estreme conseguenze, rappresentate dalla presunzione di poter operare in maniera completamente sicura in ambienti *totalmente sconosciuti* (e.g., Internet Cafe). Non è infatti possibile escludere a priori che in un contesto simile, un PC sia allestito con software malevolo *appositamente* per approfittare della buona fede degli utilizzatori che vi si recano. In uno scenario di questo tipo anche il TOE – allo stato attuale della specifica – non è in grado di garantire la sicurezza totale (per poterlo fare, lo smart disk dovrebbe incorporare un microprocessore, e tutti gli algoritmi QPK dovrebbero essere eseguiti su quello. È ipotizzabile che in futuro venga realizzato un dispositivo simile, e allora il TOE si troverebbe in una posizione ideale per poterne trarre il maggior beneficio). È infatti per esempio possibile che il sistema operativo che governa il PC tracci l'intera massa dei dati che transitano nella memoria RAM o sul bus di sistema; in quel caso, verrebbero tracciati anche i dati privati letti (pur senza essere copiati) dallo smart disk nel momento in cui vengono usati. L'attaccante avrebbe allora un clone completo di *tutte* le informazioni private dell'utente, e potrebbe agire facendosi passare per esso; l'unico baluardo di difesa che rimane al legittimo utilizzatore è dato dalla possibilità di accorgersi, al suo prossimo tentativo di accesso al servizio, che la sicurezza dello stesso è stata violata. Ma ovviamente potrebbe essere troppo tardi per porvi rimedio.

[5]Alcuni modelli di smart disk possono essere maggiormente resistenti al problema della degradazione del dispositivo flash, perchè implementano algoritmi di *wear leveling* in grado – con carichi medi di occupazione dello *storage* – di distribuire le diverse scritture successive di uno stesso dato su differenti celle di memoria fisiche.

[6]Questo approccio può rendere sicuramente più semplice l'implementazione del TOE, eliminando un componente di software personalizzato; occorre però valutare se affidandosi alla versione standard di Launchpad di un particolare produttore, non si corra il rischio che nuove versioni di quello comportino incompatibilità impreviste con il TOE.

[7]Nell'applicazione del TOE il fornitore del servizio distribuisce gli smart disks agli utilizzatori, sotto la propria responsabilità; inoltre, anche se la responsabilità fosse esplicitamente e validamente esclusa dal punto di vista legale, vale comunque la considerazione che ogni elemento di rischio possibile, in grado di compromettere le proprietà del TOE in senso globale, deve essere opportunamente ponderato e classificato.

[8]scartando, per evidenti motivi, l'opzione di scrivere un intero browser HTML dalle fondamenta.

Capitolo 6

Application Server e integrazione con sistemi *legacy*

6.1 Architettura proposta per l'integrazione

Si è descritta in (2.1.2) l'infrastruttura necessaria all'introduzione del TOE presso il fornitore di servizio; in questo capitolo verranno riepilogate le scelte tecnologiche e strutturali proposte in questo ambito, accompagnandole a valutazioni e raccomandazioni ad esse pertinenti.

6.1.1 Deployment graduale

I proponenti hanno effettuato un'osservazione dettagliata dello schema di funzionamento delle applicazioni Web individuate come primo target per l'implementazione del TOE. Come descritto più ampiamente in (2.1.1), si propone un primo livello di introduzione a basso impatto, senza effettuare interventi sull'applicativo, e livelli successivi di integrazione maggiore, da realizzarsi gradualmente. Tale approccio a perfezionamenti successivi, senza rischio di compromettere le applicazioni già produttive, consegue dall'adozione di un'architettura adeguata alla scopo, e in grado di conservare un sufficiente grado di disaccoppiamento del TOE rispetto all'applicazione cui viene agganciato, e viceversa.

6.1.2 Interfacciamento

Il TOE propone che il traffico `http` diretto alle applicazioni da proteggere sia intercettato a livello dei Web server della Banca, prima ancora che esso sia instradato verso gli application server, da un modulo scritto secondo l'interfaccia ISAPI. Tale modulo, in base ad una configurazione presente localmente, decide di trattare o meno una transazione HTTP, sulla base di informazioni di vario tipo. Se il modulo determina che la transazione deve essere trattata, ne arresta il normale flusso e provvede a contattare il Motore di runtime QPK,

utilizzando un socket TCP come protocollo di trasporto. Per evitare di introdurre un *single point of failure* nel sistema, si raccomanda di fare riferimento alle considerazioni in (6.2).

Si valuta che la scelta di utilizzare un trasporto di basso livello come TCP sia in grado di fornire prestazioni elevate, senza incorrere nell'*overhead* di protocolli più complessi; si raccomanda di mettere a disposizione una descrizione dettagliata del meccanismo di *pooling* che si propone di implementare, e degli agenti esterni (e.g. load balancer, packet sprayer o altri) che esso presuppone.

6.1.3 Motore di runtime QPK

Con questo termine si conviene di indicare un insieme di server che, in zona di sicurezza presso il fornitore di servizio, svolgono tutte le funzioni elaborative e di gestione dati necessarie al funzionamento degli algoritmi QPK. I proponenti del TOE hanno indicato in maniera preliminare che questi server saranno implementati con un ambiente Java Enterprise, utilizzando una metodologia *object-oriented*. Più in particolare il motore:

- Riceve, autentica ed instrada nella maniera corretta le richieste di azione tramite l'interfaccia Web Service di cui si è parlato in (6.1.2).
- Implementa l'algoritmo QPK-OTP per identificare gli utilizzatori del servizio.
- Implementa l'algoritmo QPK-DYN per crittare e decrittare le parti riservate dei messaggi scambiati fra l'utilizzatore e il servizio.
- Mantiene e consulta un database dove sono archiviati i dati privati relativi agli algoritmi QPK-OTP e QPK-DYN. Si noti che, come emerge dalla descrizione degli algoritmi stessi effetttuata in 3 e 4, entro questo database sono archiviate anche informazioni private che compaiono *nella stessa forma* in cui un utilizzatore le può direttamente usare[1] (si veda 3.3).
- Mantiene e aggiorna una serie di statistiche relativamente al funzionamento dei protocolli QPK e alle loro performance, ma anche in merito all'utilizzo delle applicazioni. Si raccomanda di garantire la pseudo-anonimità[2] delle informazioni convogliate in queste statistiche.

6.1.4 Monitoraggio e gestione

Nella struttura proposta dal TOE, le operazioni di configurazione degli utenti, autorizzazione degli stessi, profilazione delle applicazioni potranno essere svolte utilizzando una Web application dedicata. Si raccomanda di prendere in considerazione la creazione di procedure software in grado di automatizzare il processo di personalizzazione del dispositivo U3 (si veda 2.1.2).

6.2 Salvaguardia della ridondanza

Si assume[3] che la struttura delle applicazioni Web su cui il TOE andrà applicato contempli, come è naturale, un utilizzo estensivo di funzionalità di *clustering* e *load balancing* che evitino l'insorgere di *single points of failure*. Occorre

di conseguenza porre fin dall'inizio fra gli obiettivi caratterizzanti[4] del TOE il mantenimento di un livello complessivo di *fault-tolerance* almeno uguale a quello esistente prima dell'introduzione del TOE.

Note

[1]La protezione di queste informazioni dall'uso illegittimo che ne potrebbe fare un collaboratore del fornitore di servizio è ottenuta tramite un sistema di cifratura "locale", ancora basato sull'algoritmo QPK-DYN. Rimane la considerazione generale che, nell'architettura proposta, non vi è perfetta simmetria di trattamento dei ruoli, in quanto il fornitore del servizio *possiede* tutti i fattori di identificazione di ogni utente, e di conseguenza potrebbe impersonarne ciascuno. Nell'applicazione proposta tale particolarità non comporta evidentemente rischi aggiuntivi, e può quindi considerarsi del tutto accettabile; si noti che così non accadrebbe nel caso fra gli attori coinvolti nell'utilizzo del TOE comparissero eventuali parti terze rispetto ad utilizzatore e fornitore.

[2]Nel senso che le informazioni non devono essere normalmente riconducibili a nomi di utenti, nomi di persone o altre informazioni prontamente riconoscibili, bensì ad identificatori *opachi* in grado solo a particolari condizioni di essere messi in relazione con i dati sopracitati.

[3]Un esame della struttura delle applicazioni Web prima dell'introduzione del TOE è già stato in parte svolto da PBY; si raccomanda di documentarne il contenuto anche al fine di poter servire da base per impostare una "analisi delle differenze" durante e dopo l'eventuale implementazione del TOE.

[4]Ossia fra gli elementi caratterizzanti "esterni", in contrapposizione a quelli interni, che coincidono con lo *scopo del TOE* delineato in (1.1.1).

Capitolo 7

Raccomandazioni e linee guida per lo sviluppo

In vista dell'organizzazione e strutturazione dei lavori concreti di realizzazione del TOE, in questo capitolo sono state raccolte le indicazioni di principio e le valutazioni di merito svolte non già su oggetti realizzati o abbozzati, ma perlopiù su schemi progettuali o di architettura; sono parimenti esposte raccomandazioni che rappresentano una diretta conseguenza delle specifiche caratteristiche degli algoritmi QPK-OTP e QPK-DYN.

7.1 Indicazioni sul metodo

Con riferimento alle considerazioni esposte in (1.2.2), dove si descrive l'approccio metodologico seguito per una parte di questo audit, e basato sull'applicazione dello standard ISO 15408, si raccomanda di orientare le future attività tecniche e organizzative connesse con lo sviluppo del TOE secondo la stessa metodologia. Da un'applicazione sistematica della stessa ci si attende il raggiungimento di una serie di vantaggi:

Future attività di audit Ammettendo che il TOE venga realizzato, nella forma descritta nel presente documento o in altra simile, con ogni probabilità altri interventi di audit indipendente saranno necessari nei suoi confronti per garantire che la sicurezza e la robustezza ottenute siano effettivamente quelle cercate; in una simile circostanza, avere già strutturato le proprie attività secondo le indicazioni di ISO 15408 permette un notevole risparmio di tempo.

Razionalizzazione dello sviluppo Muovendosi secondo le linee guida tracciate da ISO 15408 e proposte nel corso di tutta la presente relazione, sarà possibile ottenere importanti vantaggi in termini di razionalizzazione delle attività di sviluppo, con conseguente ottimizzazione delle risorse e delle energie impiegate.

Migliore adattabilità ai cambiamenti Con ogni probabilità il TOE, durante la propria vita operativa – ma anche durante il proprio ciclo di sviluppo e implementazione – andrà soggetto ad una serie

> di revisioni, modifiche, estensioni, cambiamenti di varia portata. È esperienza ripetuta ormai innumerevoli volte che i sistemi e le applicazioni *meglio specificate* e *meglio documentate* sono quelle in grado di adattarsi in maggior grado e più velocemente a esigenze o circostanze ambientale mutate.

7.2 Indicazioni sulla strutturazione

I componenti di cui il TOE è composto – almeno in via teorica in questa fase iniziale – sono abbastanza numerosi e differenziati da poter far nascere qualche preoccupazione in merito alla complessità organizzativa e strutturale necessaria per coordinare tante competenze, approcci e strumenti. Sono infatti coinvolti *almeno*:

- la metodologia ISO 15408;
- la metodologia di progettazione e sviluppo *object oriented*, eventualmente con il supporto di tools tipici quali Rational™ o Eclipse;
- ambienti di sviluppo Java™;
- ambienti di sviluppo `C/C++`;
- ambienti di modellazione *Entity-Relationship*
- lo *U3 Software Development Kit*

Si raccomanda di mantenere sotto controllo tale complessità strutturale definendo in maniera formale e documentata le responsabilità, gli *inputs* e i *deliverables* per tutte le attività che vengono pianificate o svolte. Questo strumento di razionalizzazione può essere anche utilizzato, a posteriori, per verifiche di tipo gestionale.

7.3 Indicazioni su specifiche funzionali

Come viene indicato in (4.4.1), l'algoritmo QPK-DYN – in congiunzione con l'algoritmo QPK-OTP – offre le necessarie garanzie di riservatezza dei dati e identificazione dell'interlocutore, ma lascia aperto il problema dell'*integrità* del messaggio, intesa come capacità del TOE di rilevare e segnalare circostanze di modifica illegittima dei dati mentre sono trasmessi sul canale non sicuro.

Apparendo evidente che la sicurezza di un'applicazione via Web non può prescindere dalla garanzia dell'integrità, sia per i motivi esposti alla nota 23 del capitolo 4, sia per scongiurare attacchi di tipo *Denial of service*, si raccomanda di includere nell'analisi preliminare uno schema completo di *Message Authentication Code*, considerando quest'ultimo come punto improrogabile di verifica in ogni futuro intervento di audit.

7.4 Indicazioni su analisi aggiuntive

7.4.1 Studio del comportamento dei dispositivi U3 in circostanze anomale

Si è rilevato in (5.1.1) che il fatto che le applicazioni U3LP e U3LP+ abbiano il proprio ciclo di vita gestito dal *Launchpad*, se pure garantisce che alla chiusura dell'applicazione o all'estrazione dello smart disk tutti i files temporanei lasciati da quella sul PC vengano rimossi, non può certo assicurare il rispetto di tale circostanza in maniera forte, come dimostra il semplice controesempio fornito nella nota 4 del capitolo 5.

Si raccomanda – di conseguenza – da un lato, di individuare la giusta considerazione da assegnare a questa caratteristica della piattaforma U3, senza sopravvalutarla; dall'altro, di agire in maniera preventiva, eseguendo una serie di tests comportamentali mirati a descrivere come lo smart disk si comporta in determinate circostanze anomale, quali:

- Spegnimento fisico del PC (distacco dell'alimentazione) mentre un'applicazione U3LP è in funzione, verificando poi dopo la riaccensione del computer se rimangono tracce di dati privati sull'hard disk (estendendo la ricerca anche al file di *swap*). Una versione ancora più forte di questo test si può ottenere smontando, dopo lo spegnimento, l'hard disk, rimontandolo in funzione *slave* su un PC differente, e verificando i contenuti del disco senza che nessun software automatizzato abbia possibilità di ripulirli.

- Distacco forzato (senza preavviso) del dispositivo U3, mentre un'applicazione U3LP è in funzione, verificando che i dati privati siano rimossi dall'hard disk.

Sulla scorta di alcune indicazioni raccolte in rete (si veda [22]), si suggerisce inoltre di verificare il perfetto funzionamento del *Launchpad* e delle applicazioni afferenti al TOE utilizzando profili utente dotati di privilegi minimi, soprattutto nel caso di applicazioni U3LP+ che accedono direttamente al dispositivo fisico.

Bibliografia

[1] Knuth, D.E., *The Art of Computer Programming*, Vol. 2 – Seminumerical algorithms, Addison-Wesley, Reading (1981)

[2] Wikipedia, s.v. *Phishing*, `http://en.wikipedia.org/wiki/Phishing`

[3] Wikipedia, s.v. *Pharming*, `http://en.wikipedia.org/wiki/Pharming`

[4] Baldoni, M.W., Ciliberto, C., Piacentini Cattaneo, G.M., *Aritmetica, crittografia e codici*, Springer-Verlag Italia, Milano (2006)

[5] Schneier, B., *Applied Cryptography*, Wiley press, New York (1996)

[6] Gabrilovich, E., Gontmakher, A., *The Homograph attack*, Communications of the ACM, 45(2): 128

[7] Schnorr, C.P., *Zufälligkeit und Wahrscheinlichkeit*, Springer-Verlag, Berlin (1971)

[8] Barash, L., Shchur, L.N., *Periodic orbits of the ensemble of Sinai-Arnold cat maps and pseudorandom number generation*, Physical Review E 73, 036701 (2006)

[9] Accardi, L., De Tisi, F., Di Libero, A., *Sistemi dinamici instabili e generazione di successioni pseudo-casuali*, C.N.R. Rassegna di metodi statistici ed applicazioni, Cagliari (1981)

[10] Arnold, V.I., Avez, A., *Ergodic problems of Classical Mechanics*, Benjamin, New York (1968)

[11] Abundo, M., Accardi, L., Auricchio, A., *Hyperbolic automorphisms of tori and pseudo-random sequences*, Calcolo 29 (1992)

[12] Brown, R.G., *DieHarder: A GNU public licensed random number tester*, Duke University Physics Department, Durham (2006)

[13] QPK Group, *QP-DYN with 2 primes: description and attacks*, com. priv., Roma (16.09.2006)

[14] ISO/IEC 15408, *Information Technology – Security Techniques – Evaluation Criteria for IT Security*, 3 Voll., Genève (2005)

[15] U3, *U3 Deployment Kit*, Version 1.1, Revision 1.0, Redwood City (2006)

[16] Gamma, E., Helm, R., Johnson, R., Vlissides, J., *Design Patterns. Elements of Reusable Object-Oriented Software*, Addison-Wesley, Reading (1995)

[17] Horowitz, E., Sahni, S., Anderson-Freed, S., *Fundamentals of Data Structures in C*, W.H. Freeman and Company, New York (1993)

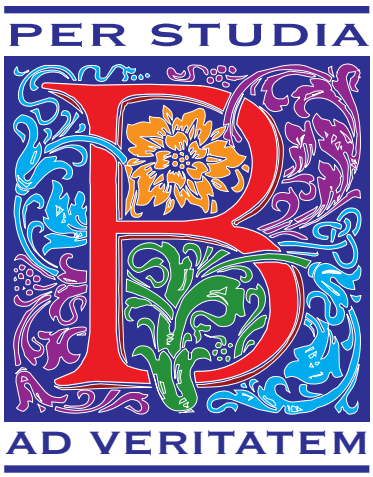
[18] Booch, G., *Object-Oriented Analysis and Design*, Addison-Wesley, Reading (1993)

[19] Menezes, A., van Oorschot, P., Vanstone, S., *Handbook of Applied Cryptography*, CRC Press, Boca Raton (1996)

[20] Klemens, B., *Math You Can't Use: Patents, Copyright, and Software*, Brookings Institution Press, Washington (2005)

[21] Wikipedia, s.v. *Software patents under United States patent law*, `http://en.wikipedia.org/wiki/Software_patents_under_United_States_patent_law`

[22] Wikipedia, s.v. *U3*, `http://en.wikipedia.org/wiki/U3`



SYNTHESIS SERVICES GmbH