Schnittstellenspezifikation

für die Absicherung von Dateien mit

FLAM®

Hybrides Verfahren mit einem klaren zufälligen Schlüssel pro FLAMFILE® und einem statischen symmetrischen Transportschlüssel (AES) geschützt durch ein Hardware-Sicherheits-Modul (HSM)

Version 1.00

06.05.2015

Inhalt

Abbildungen

Tabellen

Versionsführung

Version	Status	Datum	Durch	Änderung
00.90	In Arbeit	04.10.2011	F. Reichbott	- Erstellung des Dokuments (Übernahme der DES Spec)
00.92	In Arbeit	27.10.2011	F. Reichbott	Anpassung: ENC-Zero KTV nur 4 Bytes, wegen ICSF!
00.93	In Arbeit	02.11.2011	F. Reichbott	Neuer Funktionscode für den Verfahrenswechsel (RENW)
00.94	In Arbeit	15.01.2015	F. Reichbott	Reihenfolge ICV Bildung für Verschlüsselung korrigiert, Beschreibungen optimiert
01.00	Freigegeben	06.05.2015	F. Reichbott T. Eckert	Review und Englische Übersetzung

1 Einleitung

In diesem Dokument wird beschrieben, wie über den FLAM® Key Management EXIT (FKME) die kryptographische Absicherung von Dateien zur Speicherung bei einer Instanz (Archiv, Logs oder ähnliches) und zum Austausch zwischen verschiedenen Kommunikationspartnern (File Transfer) geschehen soll. Hierzu wird ein hybrides Verfahren spezifiziert, welches es ermöglicht, über verschiedene HSM (Hardware-Sicherheits-Module) einen zufälligen Schlüssel (FKEY) pro FLAMFILE® zu erzeugen und diesen Einmalschlüssel über einen FLAM®-Master-Key (FMKY) des HSM sicher auszutauschen. Für die sichere Übermittlung des FLAM® Keys (FKEY) wird der FLAM® Key Management CONTEXT (FKMC) im User-Header einer jeden FLAMFILE® benutzt. Für die Bereitstellung des FMKY werden die etablierten Methoden zum Schlüsselmanagement herangezogen. Das folgende Bild veranschaulicht die Zusammenhänge.

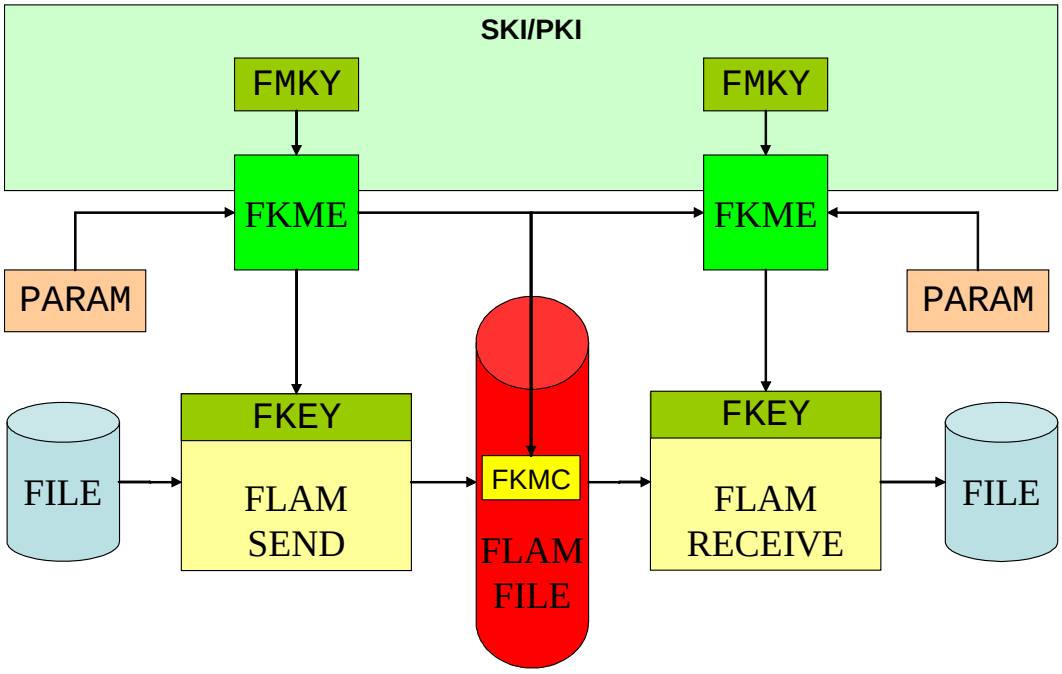

Abbildung 1 Überblick

Diese Spezifikation soll es unter anderem ermöglichen, den Anforderungen des „Payment Card Industry Data Security Standard“ (PCI DSS) und anderen Anforderungskatalogen mit FLAM® gerecht zu werden. Hierbei spielt die sichere Speicherung und Archivierung von Dateien, der Austausch dieser Dateien zwischen unterschiedlichen Parteien, als auch die Absicherung von Logs, Dumps, Datenbankbackups und anderen Dateien im Transaktionsbetrieb eine Rolle. Hinzu kommen mögliche andere Anwendungen, in denen die Vertraulichkeit und Integrität von Dateien sichergestellt werden muss.

2 Systemmodell

FLAM® mit AES (ab Version 4.0) realisiert den Integritätsschutz und die Vertraulichkeit über eine 64 Bytes lange Passphrase, die zur Ableitung von jeweils 4 128 Bit AES Schlüsseln verwendet wird. Diese 4 Schlüssel werden über alle drei Ebenen der FLAMFILE® mit Hilfe einer Einweghashfunktion abgeleitet und dienen der Verschlüsselung der Nettodaten (Segment) und der Bildung von Message Authentication Codes (MAC) in jeder Ebene über die jeweiligen Bestandteile (Segment, Member, File) der FLAMFILE®.

Damit der klare Schlüssel für FLAM® (FKEY) innerhalb einer HSM-basierten kryptographischen Infrastruktur sicher verwendet werden kann, müssen zwei Anforderungen erfüllt werden:

1. Der FKEY darf nur dort kurzzeitig klar im Arbeitsspeicher (RAM) des Rechners existieren, wo auch die Daten klar vorkommen müssen.

2. Der FKEY darf nur einmalig pro FLAMFILE® verwendet werden, damit ein mögliches Ausspähen des Wertes im Speicher des Rechners für eine andere FLAMFILE® keine Rolle spielt.

Damit existiert der FKEY nur dort, wo die klaren Daten ausgespäht werden könnten, und an mehr als diese Daten kann ein Angreifer durch den jeweiligen FKEY nicht gelangen. Die Bereitstellung des FKEY erfolgt über den FLAM® Key Management EXIT (FKME), welcher **ab Version 4.1 von FLAM®** für die Anbindung von verschiedenen kryptographischen Infrastrukturen unterstützt wird.

Damit muss der FKEY aus 64 Bytes nicht vorhersehbaren Zufall bestehen, welcher von dem jeweiligen HSM (oder Softtoken) in der entsprechenden Qualität generiert werden muss. Am Anschluss stellt das HSM (oder Softtoken) eine entsprechend durch die Hardware geschützte Struktur (FLAM Key Management CONTEXT (FKMC)) für den User-Header der FLAMFILE® bereit. Dies gestattet es, einem entsprechend personalisierten HSM (oder Softtoken), den FKEY beim Lesen in seiner klaren Form wieder herzustellen.

Zum Schutz des FKMC wird in dieser Spezifikation ein statischer symmetrischer Master Key für FLAM® (FMKY) verwendet, welcher auf der Seite des Senders zum Verschlüsseln und beim Empfänger zum Entschlüsseln des FKEY genutzt wird. Damit durch dieses hybride Verfahren die Sicherheit der HSMs nicht gefährdet wird, entspricht der klare FKEY aus der Sicht des HSM 64 Bytes normaler Daten.

Über die verschiedenen Verwendungszwecke (ENCIPHER und DECIPHER) können über die etablierten statischen Key-Management-Systeme (SKMS), welche alle Anforderungen des PCI DSS gerecht werden, gerichtete Schlüsselbeziehungen aufgebaut werden. Diese ermöglichen es, die Urheberschaft für eine FLAMFILE® eindeutig nachzuweisen. Über die verschiedenen Verwendungszwecke für das Umschlüsseln von Daten (IDATXLAT und ODATXLAT) können Vermittler, Kopfstellen und andere Besitzstandsübergänge sicher abgebildet werden. Hierbei wird nur der FKEY vom HSM umgeschlüsselt und der FKMC im

User-Header der FLAMFILE® ausgetauscht. Die eigentlichen Daten bleiben bei diesem Vorgang unter dem jeweiligen FKEY verschlüsselt. Der FKEY selbst bleibt hierbei durch das HSM durchgehend geschützt. Diese Möglichkeit zum sicheren Umschlüsseln einer FLAMFILE® stellt sicher, dass die Daten nur bei der Erzeugung und bei ihrer Verarbeitung klar existieren müssen. Dieser Sachverhalt wird durch das folgende Beispiel zwischen Generierungsstelle (GS) für die Kartenproduktion und dem Personalisierungssystem (PS) beim Kartenproduzenten deutlich gemacht:

- Sicheres Speichern der generierten Daten innerhalb der Generierungsstelle
  `FLAM(COMP, FMKY.GS2GS.ENCIPHER.ggvv)`
- Transferieren der Kartendaten zum Produzenten
  `FLAM(CHNG, FMKY.GS2GS.IDATXLAT.ggvv, FMKY.GS2PS.ODATXLAT.ggvv)`
- Empfang und sichere Speicherung beim Produzenten
  `FLAM(CHNG, FMKY.GS2PS.IDATXLAT.ggvv, FMKY.PS2PS.ODATXLAT.ggvv)`
- Entschlüsselung der gespeicherten Daten zur Produktion
  `FLAM(DECO, FMKY.PS2PS.DECIPHER.ggvv)`

Des Weiteren ist es über die Umschlüsselung möglich, von einer Generation und Version auf eine andere des gleichen oder eines anderen FMKY überzugehen. Hinzu kommt noch, wenn dieser Migrationspfad von der jeweiligen Implementierung des EXIT unterstützt wird, der Wechsel zwischen verschiedenen Cipher Suites. Für letzteres ist ein eigener Funktionscode (RENW) definiert. Dies dient dazu, die reine Umschlüsselung (Änderung der Zugriffsrechte) von der Migration der Verfahren zu trennen.

Die Generierung und der Austausch der FMKY zwischen den verschiedenen Entitäten sollten nach den etablierten und anerkannten Verfahren zum statischen Key Management (mindestens zweistufig (2 oder 3 klare Schlüsselkomponenten + FMKY als Kryptogramm)) erfolgen. Dies ist nicht Bestandteil dieser Schnittstellenspezifikation. Hier müssen die geltenden Anforderungen zur statischen Schlüsselverwaltung im jeweiligen Umfeld (VISA, ZKA, BSI, …) eingehalten werden. In diesem Dokument wird davon ausgegangen, dass die statischen FMKY für ihren entsprechenden Verwendungszweck über das HSM zur Verfügung stehen. Die Schlüssel sollten, wie alle statischen Schlüssel, regelmäßig gewechselt und adhoc ausgetauscht werden können. Hierfür sind für jeden der Schlüssel eine Generation (GG) und eine Version (VV) vorgesehen.

3 Spezifikation FLAM Key Management CONTEXT (FKMC)

3.1 Vorgaben durch FLAM®

Das Kontextfeld (FKMC) darf über alle Plattformen hinweg derzeit nicht länger als 512 Bytes werden und muss für die erfolgreiche Umschlüsselung von FLAMFILES® immer eine konstante Länge haben. Für den Inhalt werden keine Vorgaben gemacht, außer dass die ersten 50 Bytes als Information protokolliert werden und die letzten 4 Bytes einen Testwert für den FKEY, der nicht von der jeweiligen Verschlüsselung abhängen darf, beinhalten müssen.

Hinweis: Wenn mehr als 512 Bytes benötigt werden, muss hierfür eine neue Version von FLAM mit größeren Puffern bereitgestellt werden.

3.2 Aufbau der Datenstruktur in der Version 002

Die folgende Tabelle gibt eine Übersicht über die Datenstruktur. Hierbei bedeutet:

Lg = Länge in Bytes

CHR = Zeichen im jeweiligen Zeichensatz der erzeugenden Maschine
POV = BCD-kodiert, rechtsbündig mit führenden 0
BIN = Binär

Feld	Bezeichnung	Inhalt	Lg	Format
1	Infodaten	`FKMC V002 L144 AES3 KL32 EZ04 ICBC SHA2`	50	CHR
2	Generation und Version des FMKY	`GGVV`	2	BIN
3	KTV für den FMKY	`RRRRRRRR`	4	BIN
4*	Zeitpunkt (GMT) der Erzeugung der FLAMFILE®	`JJJJMMTTHHMMSSss`	8	POV
5*	Zufallszahl zur Dynamisierung (ggf. Wiedereinreichungskontrolle)	`RRRRRRRRRRRRRRR`	8	BIN
6**	Verschlüsselter FKEY	`RRR…RRR`	64	BIN
7*	HASH über 4 und 5 sowie den klaren FKEY	`RRRRRRRRRRRRRRR`	8	BIN

Tabelle 1 FLAM® Key Management CONTEXT (FKMC)

*) Die Felder werden beim Erzeugen der FLAMFILE(*) belegt und dann nicht mehr verändert.

**) Der klare FKEY wird ebenfalls nur bei der Erzeugung generiert und dann beibehalten, so dass sich nur sein Kryptogramm in Abhängigkeit vom FMKY ändern kann.

3.2.1 Erläuterung der Datenfelder

3.2.1.1 Feld 1: Infodaten

Die Infodaten nutzen die ersten 50 Bytes des Kontextfeldes zur Selbstauskunft über den EXIT und legen gleichzeitig dessen Implementierung fest. Hierbei müssen folgende Angaben in Großbuchstaben und getrennt durch ein Leerzeichen gemacht werden.

1. **Kennzeichen für den FKMC** = FKMC
   4 Bytes lange Konstante für die Prüfung und Erkennung des Zeichensatzes (EBCDIC oder ASCII).

2. **Version des EXIT** = V002
   Die Version des EXIT wird mit V002 konstant festgelegt. Dieses Kennzeichen wird zur Unterscheidung anderer Implementierungen herangezogen.

3. **Länge des EXIT** = L144
   Die Länge für das Kontextfeld dieses EXIT beträgt 144 Bytes. Die Länge kann in Abhängigkeit von der Version variieren und dient der Plausibilitätsprüfung.

4. **Algorithmus für den FMKY** = AES3
   Über das vierte Feld wird der Algorithmus für die Verschlüsselung des FKEY mit dem FMKY festgelegt. In der Version 002 sollte dies in der Regel ein 256 Bit Schlüssel sein. Bei der Verwendung von HSM kann es vorkommen, dass die Nutzende Applikation (der FKME selbst) die Länge des Masterkeys nicht kennt, in diesem Fall sollte auch AES3 eingetragen werden.

   a. 256 Bit ist AES3

   b. 196 Bit ist AES2

   c. 128 Bit ist AES1

5. **Schlüssellänge des FMKY** = KL32
   Über das fünfte Feld wird die Länge des FMKY festgelegt, die wie schon erwähnt 256 Bit betragen sollte.

   a. 256 Bit ist KL32

   b. 196 Bit ist KL24

   c. 128 Bit ist KL16

6. **Verifikationsverfahren für den FMKY** = EZ04
   Über das sechste Feld wird das Verfahren zur Berechnung des Key-Test-Values

(KTV) für den FMKY festgelegt, welches über die Methode Enc-Zeros realisiert wird. Hierbei werden aber nur die höherwertigen 4 Bytes gespeichert, was auch dem gängigen VISA Key-Test-Verfahren entspricht.

7. **Ciphermode für den FKEY** = ICBC
   Das siebente Feld legt den Modus für die Verschlüsselung des FKEY fest. ICBC steht hierbei für den Cipher-Block-Chaining Mode mit Verwendung eines Initialisierungsvektors.

8. **Verfahren für die Einweghashfunktion** = SHA2
   Als Verfahren für den Hash über den Zeitstempel, die Zufallszahl und den klaren FLAM®- Schlüssel wird SHA-256 als Algorithmus festgelegt.

9. **Weitere Informationen**
   Weitere Informationen (zum Beispiel die Implementierung des EXIT (IBMCCA, IBMDKMS, PKCS11, ATALLA, THALES, …)) sind der jeweiligen Implementierung freigestellt. Wenn keine Informationen vorliegen, werden die restlichen 10 Bytes mit Leerzeichen aufgefüllt.

Zum Check der Implementierung des EXIT gegen die Infodaten werden die ersten 40 Bytes in der Version 002 gegen die folgende Konstante verglichen (Achtung: Leerstelle am Ende):

„FKMC V002 L144 AES3 KL32 EZ04 ICBC SHA2 “

Hierbei ist es wichtig, dass der Zeichensatz für den Vergleich keine Rolle spielen darf. Die Ermittlung des Zeichensatzes sollte über das erste Byte der Infodaten erfolgen, welches in EBCDIC ‚C6’hex und in ASCII ‚46’hex für ein großes ‚F’ betragen muss. Da die ersten 50 Bytes von FLAM als Kommentar der FLAMFILE protokolliert werden, muss hier beim Schreiben der systemseitige Zeichensatz genommen werden. Beim Lesen erkennt FLAM den Zeichensatz (ASCII oder EBCDIC) für die Protokollierung. Die FKME-Implementierung muss diesen an dieser Stelle auch erkennen, was anhand des ersten Zeichens, wie oben beschrieben, auf einfache Art und Weise möglich ist.

3.2.1.2 Feld 2: Generation und Version des FMKY

In diesem Feld wird die Generation (gg) und die Version (vv) des FMKY beim Senden eingestellt. Sie dient beim Empfangen der FLAMFILE® dazu, den richtigen FLAM-Master-Key (FMKY) zu verwenden. Die Ermittlung der Generation und Version beim Senden und ihre Verwendung beim Empfangen hängen von der Implementierung des EXIT ab. In der Regel sollte dem EXIT beim Senden ein voll qualifiziertes Label für den Schlüssel zusammen mit der dazugehörigen Generation und Version übergeben werden. Beim Empfangen ist es dann ausreichend, wenn in dem Label Platzhalter für die Generation und Version enthalten sind.

3.2.1.3 Feld 3: KTV für den FMKY

Das Key-Test-Value (KTV) für den FMKY ist das Ergebnis der AES Verschlüsselung von 16 Bytes Nullen mit dem FMKY. Hierbei werden aber nur die höherwertigen (linken) 4 Bytes in dem Feld gespeichert. Das Verfahren wurde gewählt, da es von den meisten HSMs direkt unterstützt wird, den gestellten Sicherheitsanforderungen gerecht wird, auf der Seite des Senders auch über die Verschlüsselung von 16 Bytes Daten erzeugt werden kann, was bei Verzicht auf die gerichteten Schlüsselbeziehungen auch beim Empfänger möglich wird, und weil es mit Hilfe von AES als Basisalgorithmus abbildbar ist.

3.2.1.4 Feld 4: Zeitpunkt der Erzeugung der FLAMFILE®

Der Zeitpunkt der Erzeugung der FLAMFILE® muss auf Basis der Anforderungen kryptographisch sicher protokolliert werden. Dies geschieht zum einen durch die Einbeziehung in die HASH-Berechnung und zum anderen in die Bildung des IV für die CBC-Mode-Verschlüsselung. Das Format besteht, wie oben in der Tabelle angedeutet, aus:

- JJJJ - Jahr
- MM - Monat
- TT - Tag
- HH - Stunde
- MM - Minute
- SS - Sekunde
- ss - Millisekunde

Es ist wünschenswert, einen verlässlichen Zeitgeber für die Abbildung der 8 Bytes zu nutzen. Für den Zeitpunkt wird die Greenwich Mean Time (GMT) festgelegt, damit ein weltweiter Vergleich möglich wird.

3.2.1.5 Feld 5: Zufallszahl zur Dynamisierung

Die 8 Bytes Zufallszahl zur Dynamisierung dient als Kennzeichen für eine FLAMFILE® und wird in die HASH- Berechnung sowie die IV-Bildung mit einbezogen.

3.2.1.6 Feld 6: Verschlüsselter FKEY

Der 64 Bytes lange zufällige FKEY (4 Blöcke) wird in der Version 002 im CBC- Mode mit dem FMKY AES verschlüsselt. Der IV ergibt sich aus der Konkatenierung der Felder 5 und 4. (Achtung: die Reihenfolge ist hier anders als bei der Hashwertberechnung, der ICV muss hier explizit zusammengebaut werden, man kann nicht die Adresse von Feld 4 direkt als ICV

verwenden, wenn das Feld 5 dahinter folgt.) Das Ergebnis ist ein Kryptogramm für den FKEY von 64 Bytes, welches vom EXIT in binärer Form in die Kontextstruktur eingestellt wird.

3.2.1.7 Feld 7: Hash über Feld 4 und Feld 5 sowie den klaren FKEY

Die Berechnung des Hashwertes erfolgt nach dem SHA-256 Verfahren über die folgenden 80 Bytes:

- 8 Bytes Feld 4: Zeitpunkt der Erzeugung der FLAMFILE®
- 8 Bytes Feld 5: Zufallszahl zur Dynamisierung
- 64 Bytes klarer zufälliger FKEY.

Es werden nur die höherwertigen (linken) 8 Bytes des Hashwertes in der Kontextstruktur hinterlegt. Ihre Berechnung erfolgt bei der Generierung des Schlüssels, die Verifikation dieses Wertes sollte aber nur bei der Verwendung des Schlüssels erfolgen. Das heißt, dass die Verifikation nur vorgenommen wird, wenn der klare Schlüsselwert gebraucht wird. Dies ist bei der Umschlüsselung einer FLAMFILE® nicht der Fall.

4 Spezifikation FLAM® Key Management EXIT (FKME)

Der FLAM® Key Management EXIT (FKME) ist eine Funktion, welche von FLAM® aufgerufen wird, wenn entsprechende Parameter (PARAM) für einen EXIT an FLAM übergeben wurden. Er dient zur Anbindung beliebiger kryptographischer Infrastrukturen, um Session-Key-Verfahren mit FLAM® abbilden zu können.

4.1 Aufbau der Funktionsschnittstelle

Die folgende Tabelle gibt eine abstrakte Übersicht über die Parameter der Schnittstelle. Hierbei bedeutet:

INT = 32 Bit (4 Bytes) im Zweierkomplement
STR = Array von Bytes

Para	Bez.	Kommentar	Richtung	Typ	Länge
1	Fuco	Funktionscode	INPUT	INT	4 Bytes
2	RetCo	Returncode	OUTPUT	INT	4 Bytes
3	ParLen	Länge der Inputparameter	INPUT	INT	4 Bytes
4	Param	Inputparameter	INPUT	STR	variabel
5	DatLen	Länge der Daten (FKMC)	COMP:INPUT/OUTPUT DECO:INPUT CHNG:INPUT=OUTPUT RENW:INPUT	INT	4 Bytes
6	Data	Daten (FKMC)	COMP:OUTPUT DECO:INPUT CHNG:INPUT/OUTPUT RENW:INPUT	STR	variabel
7	KeyLen	COMP/DECO: Länge des Schlüssels (FKEY) RENEW: Länge des neuen Kontextfeldes (FKMC)	COMP:INPUT/OUTPUT DECO:INPUT/OUTPUT CHNG: not used RENW:INPUT/OUTPUT	INT	4 Bytes
8	Key	COMP/DECO: Schlüssel (FKEY) RENEW: Neues Kontextfeld (FKMC)	COMP:OUTPUT DECO:OUTPUT CHNG: not used RENW:OUTPUT	STR	variabel
9	MsgLen	Länge der Nachricht	INPUT/OUTPUT	INT	4 Bytes

Para	Bez.	Kommentar	Richtung	Typ	Länge
10	Message	Nachricht	OUTPUT	STR	variabel

Tabelle 2 FLAM® Key Management EXIT (FKME)

Alle Parameter werden als Zeiger (Call by Reference) übergeben. Der FKME wird als Modul mit einer Funktion zur Laufzeit von FLAM® dynamisch geladen. Hierbei sind der Name des Moduls und ggf. der Name der Funktion über FLAM® parametrisierbar. Die Inputlängen geben immer die Länge des jeweils über den nachfolgenden Parameter übergebenen Daten an. Die Outputlängen sind nach dem Ausführen der Funktion auf die Menge des benötigten Speichers gesetzt.

4.1.1 Erläuterung der Parameter

4.1.1.1 Para 1: Funktionscode

Über den Funktionscode wird die Methode für den EXIT ausgewählt. Folgende Verfahren sind definiert:

- **0 – Decompression/Entschlüsselung**
  In diesem Fall werden die Inputparameter zusammen mit dem Kontextfeld (FKMC) aus dem Userheader an den EXIT übergeben, welcher daraufhin den klaren Schlüssel (FKEY) an FLAM zurückgibt.

- **1 – Compression/Verschlüsselung**
  In diesem Fall werden die Inputparameter an den EXIT übergeben, welcher daraufhin 64 Bytes Zufall (FKEY) generiert und diesen zusammen mit dem Kontextfeld (FKMC) zurückgibt. FLAM schreibt daraufhin das Kontextfeld in den Userheader und nutzt den klaren zufälligen Schlüssel für die Erzeugung der FLAMFILE®.

- **2 – Change/Umschlüsselung**
  In diesem Fall werden die Inputparameter zusammen mit dem Kontextfeld (FKMC) aus dem Userheader an den EXIT übergeben, welcher daraufhin das neue Kontextfeld (FKMC) berechnet und an das Programm zurückgibt. Dieses Programm erstellt nun eine neue FLAMFILE® mit angepasstem Userheader. Hierbei muss der Speicherbereich für das neu erzeugte Kontextfeld groß genug sein.

- **3 - Renew/Verfahrenswechsel**
  Mit diesem Funktionscode kann ein Verfahrenswechsel (Übergang zwischen verschiedenen Arten der Verschlüsselung) vorgenommen werden, wenn dies von dem jeweiligen EXIT unterstützt wird.

- **0xFFFFFFFF – Information**
  Mit dieser Methode wird der EXIT aufgefordert, eine Selbstauskunft in das Messagefeld einzustellen.

4.1.1.2 Para 2: Returncode

Der Returncode dient der Programmsteuerung von FLAM. Ist dieser nach dem Aufruf des EXIT 0, war alles in Ordnung, und FLAM arbeitet, wie erwartet, weiter. Wenn der Returncode 4 beträgt, wurde beim Aufruf des EXIT zu wenig Speicher bereitgestellt. Dies wird auf Plattformen mit dynamischer Speicherverwaltung zu einem erneuten Aufruf mit genügend Speicher führen. Auf den anderen Plattformen führt dieser Fehler wie alle anderen Fehler > 0 zu einem Abbruch von FLAM. Die Ursachen für einen solchen Abbruch können vom EXIT nur über das Messagefeld nach außen mitgeteilt werden. Dieses wird immer ausgegeben, wenn die Länge der Nachricht ungleich 0 ist. Damit ist es dem EXIT auch möglich, Warnungen bei einem Returncode = 0 nach außen zu geben. Wenn alles in Ordnung war, dann wird dies zusammen mit der Funktion (COMP, CHNG, DECO, RENW), dem Zeitstempel und der Zufallszahl entsprechend über das Messagefeld protokolliert.

4.1.1.3 Para 3: Länge der Inputparameter

Die Länge der Inputparameter ist abhängig von der Methode und der Implementierung des EXITS. Sie wird von FLAM® beim Aufruf bestimmt und an den EXIT weitergegeben.

4.1.1.4 Para 4: Inputparameter in der Version 002

Die Inputparameter sind von der Methode und der Implementierung des EXITS abhängig. Grundsätzlich sollten sie in Abhängigkeit der Methode folgende Informationen beinhalten:

- **VERSCHLÜSSELN**
    - Ggf. das Verfahren, wenn mehrere unterstützt werden
    - Ggf. Identifikationsdaten und Authentifikationsdaten für das HSM
    - Referenzierungsdaten für den FLAM-Master-Key (FMKY)
        - Key Label und Generation + Version oder
        - Key Label und Key Label Template
- **ENTSCHLÜSSELN**
    - Ggf. Identifikationsdaten und Authentifikationsdaten für das HSM
    - Referenzierungsdaten für den FLAM-Master-Key (FMKY)
        - Key Label Template
- **UMSCHLÜSSELN**

- Ggf. Identifikationsdaten und Authentifikationsdaten für das HSM
- Referenzierungsdaten für den FLAM-Master-Key (FMKY) reingehend
  - Key Label Template
- Referenzierungsdaten für den FLAM-Master-Key (FMKY) rausgehend
  - Key Label und Generation + Version oder
  - Key Label und Key Label Template

- **Verfahrenswechsel**
  - Eine neue Version für den Output-FKMC, um von einer alten Methode (TDES) der Verschlüsselung auf eine neue Form (AES3) übergehen zu können.
  - Ggf. Identifikationsdaten und Authentifikationsdaten für das HSM
  - Referenzierungsdaten für den FLAM-Master-Key (FMKY) reingehend
    - Key Label Template
  - Referenzierungsdaten für den FLAM-Master-Key (FMKY) rausgehend
    - Key Label und Generation + Version oder
    - Key Label und Key Label Template

Für die Inputparameter beim Aufruf von FLAM steht derzeit maximal ein Puffer von 256 Bytes zur Verfügung.

4.1.1.5 Para 5: Länge der Kontextdaten (FKMC) in der Version 002

Die Länge der Kontextdaten (FKMC) beträgt 144 Bytes. Diese Größe des Puffers unterschreitet die 512 Bytes, welche von FLAM zur Verfügung gestellt werden. Damit sollte für das Kontextfeld immer genügend Speicher bereit stehen. Dieser Parameter ist ausgangsseitig vom EXIT entsprechend auf 144 zu setzen, und es muss kontrolliert werden, wenn von FLAM® ein Kontextfeld übergeben wird (ENTSCHLÜSSELUNG oder UMSCHLÜSSELUNG), dass dieser Wert auf 144 steht.

4.1.1.6 Para 6: Kontextdaten (FKMC) in der Version 002

Über diesen Parameter wird das Kontextfeld (FKMC) vom EXIT bereitgestellt oder entgegengenommen. Seine Spezifikation ist dem Punkt 3.2 zu entnehmen.

4.1.1.7 Para 7: Länge des Schlüssels (FKEY) in der Version 002

Die Länge des klaren zufälligen Schlüssels beträgt, wenn dieser vom EXIT zurückgegeben wird (VER- oder ENTSCHLÜSSELN), immer 64 Bytes, welche auch von FLAM® hierfür eingangsseitig bereitgestellt werden.

Beim RENW wird dieser Parameter genutzt, um die zur Verfügung stehende Länge für die neue Kontextstruktur anzugeben, wobei der EXIT die richtige Länge zurück gibt.

4.1.1.8 Para 8: Schlüssel (FKEY) in der Version 002

Über diesen Parameter wird FLAM® vom EXIT der 64 Bytes lange Schlüssel (Zufallszahl) zur Ver- oder Entschlüsselung zur Verfügung gestellt.

Beim RENW wird dieser Parameter genutzt, um die neue Kontextstruktur zurückzugeben.

4.1.1.9 Para 9: Länge der Message

Für die Message hält FLAM® derzeit einen maximalen Puffer von 128 Bytes bereit. Dieser kann vom EXIT genutzt werden, um Fehlerursachen oder andere Informationen von FLAM protokollieren zu lassen.

4.1.1.10 Para 10: Message

Beinhaltet die Message, welche bei Länge ungleich 0 ausgegeben wird. Wenn der Funktionscode ‚FFFFFFFFFF'hex ist, dann gibt der EXIT seine Infodaten zurück.

4.2 Abläufe in der Version 002

4.2.1 Verschlüsseln einer FLAMFILE®

FLAM® ruft den EXIT mit der folgenden Belegung der Parameter auf, und der EXIT gibt die darauf folgenden Werte zurück:

Para	Inputwerte	Outputwerte
Fuco	1	=
RetCo	0	0
ParLen	>0	=
Param	Parameter für den EXIT	=
DatLen	512	144
Data	Undefiniert	FKMC laut 3.2
KeyLen	64	64
Key	Undefiniert	64 Bytes generierter Zufall
MsgLen	128	Länge der Gutmeldung
Message	Undefiniert	Gutmeldung

Tabelle 3 Parameter beim Verschlüsseln

An dieser Stelle wird der Ablauf bei der Verschlüsselung (Fuco=1) in Stichpunkten dargelegt.

- Kontrolle der Pufferlängen
- Bestimmung der Identifikations- und Authentifikationsdaten für das HSM aus dem Parameterfeld
- Bestimmung des Labels und der Generation und Version für den Schlüssel aus dem Parameterfeld
- Initialisierung der Kontextstruktur mit den Infodaten
- Eintragen der Generation und Version in das Kontextfeld
- Berechnung des KTV für den FMKY auf Basis des Labels und Eintragung ins Kontextfeld
- Bestimmung des Zeitstempels und Eintragung ins Kontextfeld
- Generierung der Zufallszahl und Eintragung ins Kontextfeld
- Bildung des IV aus Zeitstempel und Zufallszahl
- Generierung des 64 Bytes langen Schlüsselwertes als Rückgabewert an FLAM®
- Berechnung des SHA-256 über den Zeitstempel, die Zufallszahl und den klaren Schlüsselwert und Eintragung in das Kontextfeld
- CBC Verschlüsselung des Schlüsselwertes mit dem IV und dem FMKY und Eintragung des Ergebnisses in das Kontextfeld
- Setzen der Pufferlängen, der Gutmeldung und des Returncodes

- Rücksprung aus dem EXIT

4.2.2 Umschlüsseln einer FLAMFILE®

Das Programm ruft den EXIT mit der folgenden Belegung der Parameter auf, und der EXIT gibt, wenn alles erfolgreich verläuft, die darauf folgenden Werte zurück:

Para	Inputwerte	Outputwerte
Fuco	2	=
RetCo	0	0
ParLen	>0	=
Param	Parameter für den EXIT	=
DatLen	144	144
Data	FKMC laut 3.2	FKMC laut 3.2
KeyLen	Undefiniert	=
Key	Undefiniert	=
MsgLen	128	Länge der Gutmeldung
Message	Undefiniert	Gutmeldung

Tabelle 4 Parameter beim Umschlüsseln

An dieser Stelle wird der Ablauf bei der Umschlüsselung (Fuco=2) in Stichpunkten dargelegt.

- Kontrolle der Pufferlängen
- Bestimmung der Identifikations- und Authentifikationsdaten für das HSM aus dem Parameterfeld
- Bestimmung des Inputtemplates für den Inputschlüssel aus dem Parameterfeld
- Bestimmung des Outputlabels und der Outputgeneration und -version für den Outputschlüssel aus dem Parameterfeld
- Bestimmung der Inputversion und –generation aus dem Kontextfeld
- Bestimmung des voll qualifizierten Inputlabels aus Template, Generation und Version
- Verifikation des KTV über das Inputlabel
- Eintragen der Outputgeneration und -version in das Kontextfeld
- Berechnung des KTV für den Output-FMKY auf Basis des Outputlabels und Eintragung ins Kontextfeld
- Bildung des IV aus Zeitstempel und Zufallszahl

- Umschlüsselung des 64 Bytes langen Schlüsselwertes vom Input- FMKY zum Output- FMKY auf Basis des IV und Ersetzung des Kryptogramms im Kontextfeld
- Setzen der Pufferlängen, der Gutmeldung und des Returncodes
- Rücksprung aus dem EXIT

4.2.3 Entschlüsseln einer FLAMFILE®

FLAM® ruft den EXIT mit der folgenden Belegung der Parameter auf, und der EXIT gibt die darauf folgenden Werte zurück:

Para	Inputwerte	Outputwerte
Fuco	0	=
RetCo	0	0
ParLen	>0	=
Param	Parameter für den EXIT	=
DatLen	144	=
Data	FKMC laut 3.2	=
KeyLen	64	64
Key	Undefiniert	64 Bytes entschlüsselter Zufall aus FKMC
MsgLen	128	Länge der Gutmeldung
Message	Undefiniert	Gutmeldung

Tabelle 5 Parameter beim Entschlüsseln

An dieser Stelle wird der Ablauf bei der Entschlüsselung (Fuco=0) in Stichpunkten dargelegt.

- Kontrolle der Pufferlängen
- Bestimmung der Identifikations- und Authentifikationsdaten für das HSM aus dem Parameterfeld
- Bestimmung des Templates für den statischen Schlüssel aus dem Parameterfeld
- Bestimmung der Version und Generation aus dem Kontextfeld
- Bestimmung des voll qualifizierten Labels aus Template, Generation und Version
- Verifikation des KTV über das Label
- Bildung des IV aus Zeitstempel und Zufallszahl
- CBC Entschlüsselung des 64 Bytes langen Schlüsselwertes mit Hilfe des FMKY und des IV und Bereitstellung als Rückgabewert an FLAM®

- Berechnung des SHA-256 über den Zeitstempel, die Zufallszahl und den klaren Schlüsselwert und Vergleich mit dem Wert im Kontextfeld
- Setzten der Pufferlängen, der Gutmeldung und des Returncodes
- Rücksprung aus dem EXIT

4.2.4 Verfahrenswechsel bei einer FLAMFILE®

FLAM® ruft den EXIT mit der folgenden Belegung der Parameter auf, und der EXIT gibt die darauf folgenden Werte zurück:

Para	**Inputwerte**	**Outputwerte**
Fuco	3	=
RetCo	0	0
ParLen	>0	=
Param	Parameter für den EXIT	=
DatLen	144	=
Data	FKMC laut 3.2	=
KeyLen	512	???
Key	undefiniert	FKMC in der neuen Form
MsgLen	128	Länge der Gutmeldung
Message	Undefiniert	Gutmeldung

Tabelle 6 Parameter beim Entschlüsseln

An dieser Stelle wird der Ablauf nicht explizit beschrieben, da der neuartige FKMC sowie das damit verbunden neue Verfahren nicht festgelegt sind. Wenn ein Exit den RENW unterstützt, dann muss er für jeden Verfahrensübergang eine eigene Funktion implementieren, welche über den ersten Parameter für den EXIT zu kennzeichnen ist.

5 Anhang

5.1 FLAM Implementierungsempfehlung

Damit die Implementierungen der verschiedenen EXITs sich von der Verwendung her ähnlich gestalten, werden im Folgenden einige Empfehlungen für die Implementierung gegeben, welche soweit es die jeweilige HSM Architektur zulässt, eingehalten werden sollten.

5.1.1 Behandlung von Generation und Version

Die Generation und Version wird beim Senden über ein Template aus dem Keylabel ermittelt. Hierfür sind die folgenden Ersetzungszeichen für das Template definiert:

- Generation ,++'
- Version ,**'

Alle anderen Zeichen müssen mit dem korrespondierenden Label übereinstimmen. Oder Sie können durch ein ,%' ersetzt werden, damit die Position der Generation und Version im Label vom EXIT ermittelt werden kann.

Beispiel: ,TFMKY.%%%%%%%%.%%%%%%%%.DAT0++**' beim Senden.

Das Template fürs Empfangen darf kein ‚%' beinhalten, damit der Name für den Schlüssel vervollständigt werden kann.

Beispiel: ,TFMKY.BV000000.GUD00000.DAT0++**' beim Empfangen.

Beim Erzeugen einer FLAMFILE® wird somit immer ein vollständiges Label und ein Template für den EXIT an FLAM® übergeben. Wenn später auf eine FLAMFILE® zugegriffen werden muss, wird nur ein Template übergeben, in dem ausschließlich die Generation und Version offen bleibt.

5.1.2 Übergabe der Input-Parameter über FLAM bzw. FLAMUP an den EXIT

Die Inputparameter für den EXIT werden als ein Parameter in der Parameterliste für FLAM® bzw. FLAMUP übergeben. Hierbei gilt, dass die Parameter zu einem String zusammengefasst werden müssen. Sind runde Klammern oder einfache Anführungszeichen enthalten, müssen sie durch Verdopplung escapet werden. Hierbei sollte zuerst das ggf. erforderliche KeyLabel gefolgt von den KeyTemplates angegeben werden. Danach können optional die Identifikationsdaten und Authentifikationsdaten folgen. Alle Werte werden einfach durch ein Freizeichen getrennt angegeben. Es folgt nun in Abhängigkeit des Anwendungsfalls jeweils ein Beispiel für den Parameterstring ohne Anmeldeinformationen für das HSM:

- **VERSCHLÜSSELN**
  TFMKY.BV000000.GUD00000.DAT00601
  TFMKY.%%%%%%%%.%%%%%%%%.DAT0++**

- **UMSCHLÜSSELN**
  TFMKY.BV000000.GUD00000.DAT0++**
  TFMKY.GUD00000.GUD00000.DAT00601
  TFMKY.%%%%%%%%.%%%%%%%%.DAT0++**

- **ENTSCHLÜSSELN**
  TFMKY.GUD00000.GUD00000.DAT0++**

- **VERFAHRENSWECHSEL**
  TDESAES3
  TFMKY.GUD00000.GUD00000.DAT0++**
  TFMKY.GUD00AES.GUD00AES.DAT00901
  TFMKY.%%%%%%%%.%%%%%%%%.DAT0++**

5.1.3 Die letzten 10 Bytes im Info-Feld des FKMC

Die ersten 40 Bytes der 50 Bytes des Infofeldes sind unter 3.2 spezifiziert worden. Die restlichen 10 Bytes stehen dem EXIT frei zur Verfügung. Hier ist es empfehlenswert, wenn der EXIT ein Kennzeichen für seine Implementierung hinterlegt. Folgende Kennzeichen werden festgelegt:

- LIMESSWM02
  Standardimplementierung in Software der Limes Datentechnik
  „FKMC V002 L144 AES3 KL32 EZ04 ICBC SHA2 LIMESSWM02“

- IBMCCCAxx
  IBM Implementierung gegen die SAPI der CCA
  „FKMC V002 L144 AES3 KL32 EZ04 ICBC SHA2 IBMCCCA02 “

- IBMDKMSxx
  IBM Implementierung gegen das DKMS General Purpose API
  „FKMC V002 L144 AES3 KL16 EZ04 ICBC SHA2 IBMDKMS02 “

- GUDPKCSxx
  PKCS11 Implementierung von G+D
  „FKMC V002 L144 AES3 KL16 EZ04 ICBC SHA2 GUDPKCS02 “

- BVUTIMAxx
  UTIMACO Implementierung des Bankverlages
  „FKMC V002 L144 AES3 KL16 EZ04 ICBC SHA2 BVUTIMA02 “

xx – Steht hierbei für die Version der Implementierung und sollte, wie in den Beispielen, 02 sein.

Die Limes Datentechnik wird als Implementierungen für alle Verfahren die folgenden Kennzeichner für sich reservieren:

- `LIMESSWMxx - Standardimplementierung in Software`
- `LIMESCCAxx - Implementierung für IBM47xx bzw ICSF`
- `LIMESP11xx - Implementierung für PKCS#11`

5.1.4 EBCDIC- und ASCII-Wandlung

Vom Zeichensatz ist nur das Infofeld abhängig. Beim Erzeugen der FLAMFILE® wird das Infofeld in Abhängigkeit der Plattform des Senders gefüllt. Damit muss der Empfänger prüfen, in welchen Zeichensatz die Infodaten eingestellt wurden, und ggf. eine entsprechende Wandlung vornehmen.

5.1.5 Nutzung von Messages

5.1.5.1 Fehlermeldung des Exits

Die folgenden Meldungen sollten beim Auftreten des entsprechenden Fehlers herangezogen werden.

- FKME – The function code is not supported + additional error info
- FKME – The input parameter length is not correct + additional error info
- FMKE – The input parameter is not formatted correctly + additional error info
- FKME – The length of the data field is too short + additional error info
- FKME – The length of the data field is not correct + additional error info
- FKME – The data field is not formatted correctly + additional error info
- FKME – The length of the key field is too short + additional error info
- FKME – The length of the key field is not correct + additional error info
- FKME – The key field is not formatted correctly + additional error info
- FKME – The authentication failed + additional error info

- FKME – The cipher suite is not supported (The FKMC info field is not correct) + additional error info
- FKME – The determination of generation and version failed + additional error info
- FKME – The generation and version is not formatted correctly + additional error info
- FKME – The determination of the label for the FMKY failed + additional error info
- FKME – FMKY not found + additional error info
- FKME – The calculation of the key test pattern for FMKY failed + additional error info
- FKME – The verification of the key test pattern for FMKY failed + additional error info
- FKME – The determination of the time stamp failed + additional error info
- FKME – The verification of the time stamp failed + additional error info
- FKME – The generation of the random numbers failed + additional error info
- FKME – The calculation of the hash value (FKEY) failed + additional error info
- FKME – The verification of the hash value (FKEY) failed + additional error info
- FKME – The encryption of FKEY failed + additional error info
- FKME – The decryption of FKEY failed + additional error info
- FKME – The translate of FKEY failed + additional error info

Alle weiteren Fehler müssen mit ‘FKME - ‘ beginnen und können andere Meldungen in englischer Sprache gefolgt von entsprechenden Fehlerinformationen der jeweiligen Subsysteme oder des EXITS beinhalten.

5.1.5.2 Gutmeldung des Exits

Wenn kein Fehler bei der Ausführung des jeweiligen Funktionscodes aufgetreten ist, wird immer eine so genannte Gutmeldung erzeugt, welche der Protokollierung dient. Sie setzt sich aus dem Funktionscode, dem Zeitstempel und der Zufallszahl zusammen.

„FKME – COMP successful + TSP(JJJJMMTTHHMMSSss) RND(RRRRRRRRRRRRRRRR)“

„FKME – CHNG successful + TSP(JJJJMMTTHHMMSSss) RND(RRRRRRRRRRRRRRRR)“

„FKME – DECO successful + TSP(JJJJMMTTHHMMSSss) RND(RRRRRRRRRRRRRRRR)“

„FKME – RENW successful + TSP(JJJJMMTTHHMMSSss) RND(RRRRRRRRRRRRRRRR)“

Durch die Protokollierung des Zeitstempels und der Zufallszahl werden neben den entsprechend möglichen Kontrollen und Quittierungen auch die Vorgaben von VISA und MasterCard erfüllt.

5.1.5.3 Selbstauskunft des Exits

Die ersten 50 Bytes sollten dem Infofeld des FKMC entsprechen. Danach sollten weitere nützliche Informationen über die jeweilige Implementierung des EXIT folgen.

„FKME - FKMC V002 L144 AES3 KL32 EZ04 ICBC SHA2 LIMESSWM02 + additional info“

FKME – Standardimplementierung mit festem Schlüssel (FLAMFIX02)

Für eine getrennte Übertragung des FKMC vom FLAMFILE® (neues Feature von FLAM® in der Version 4.?) muss ein Sonderfall für diesen EXIT spezifiziert werden. Hierbei wird als FMKY ein im EXIT hart codierter Wert (wird nicht veröffentlicht) genommen, damit jede FLAM® Installation das dazugehörige FLAMFILE® lesen kann, obwohl es an sich verschlüsselt wurde. Da der FMKY nicht geheim ist, handelt es sich hier genau genommen nur um eine Verschleierung. Wenn man aber den FKMC getrennt vom FLAMFILE® über einen zweiten sicheren Kanal (SSL/TLS) überträgt, kann man für das große FLAMFILE® einfach FTP nehmen und hat organisatorisch auch einen Übertragungsschutz realisiert. Für diese spezielle Implementierung werden die folgenden Festlegungen getroffen:

- Schlüsselwert für den FMKY: ist fest und wird nicht veröffentlicht
- Generation: 00
- Version: 00
- KTV: ist demzufolge auch fest
- Länge der Inputparameter: 0
- Inputparameter: keine

Diese Version des Exits ist für spezielle Anwendungen gedacht, für die ein sicherer Kanal für kleine Datenmengen und ein unsicherer Kanal für große Datenmengen zur Verfügung steht. Hierfür wird FLAM einen SPLIT und einen MERGE als Funktion unterstützen. Der SPLIT liest ein vollständiges FLAMFILE® mit FKMC und erzeugt ein neues FLAMFILE® ohne den FKMC und schreibt diesen in eine 2. Datei weg. Der MERGE baut ein vollständiges FLAMFILE® aus FKMC und dem FLAMFILE® ohne FKMC zusammen. In einer unvollständigen FLAMFILE® verbleiben im Userheader die letzten 4 Bytes der FKMC, damit man nicht ausversehen einen FKMC mit einem FLAMFILE® zusammenführt, die nicht zueinander passen.

6 Abkürzungsverzeichnis

AES = Advanced Encryption Standard

BIN = Binär

CBC = Cipher Block Chaining

CHNG = Change

CHR = Character

COMP = Compression

DAT = Daten

DECO = Decompression

DED = Decryption Encryption Decryption

DES = Data Encryption Standard

EDE = Encryption Decryption Encryption

EZ = Encrypted Zeros

FKEY = FLAM® Key

FKMC = FLAM® Key Management CONTEXT

FKME = FLAM® Key Management EXIT

FLAM® = Frankenstein Limes Access Method

FMKY = FLAM® Master Key

FUCO = Funktionscode

GG = Generation

GS = Generierungsstelle

HSM = Hardware Security Module

ICBC = CBC mit IV

INT = Integer

IV = Initialisierungsvektor

KL = Key Length

KTV = Key Test Value

LG = Länge

MAC = Message Authentication Code

MDC = Message Digest Cipher

MSG = Message

PAR = Parameter

PARAM = Parameter

PCI DSS = Payment Card Industry Data Security Standard

PIN = Personal Identification number

POV = BCD Kodierung

PS = Personalisierungsstelle

RAM = Random Access Memory

RETCO = Returncode

SKMS = Static Key Management System

STR = String

TDES = Triple DES

VV = Version

ZKA = Zentraler Kredit Ausschuss