Neues Verfahren zur PIN-Berechnung und PIN-Prüfung für ec-Karten für den BdB

Version 5.0

Einleitung

Der nachfolgend ausgearbeitete Vorschlag geht von folgenden Prämissen aus:

1. Das neue Verfahren sieht - anders als bisher - eine strikte Trennung von PIN-Generierung und PIN-Verifizierung vor, wobei es nicht möglich sein darf, aus Kenntnis des PIN-Verifikationsverfahren auf das Generierungsverfahren zu schließen. Damit wird die Geheimhaltung des PIN-Generierungsverfahrens ermöglicht.
2. Das neue Verfahren soll mittelfristig umgesetzt werden; daher sind Hardware-Änderungen an den betroffenen Sicherheitsboxen zu vermeiden.
3. Verfahrensseitig wird die Option einer vom Kunden frei wählbaren PIN vorgesehen. Die für die Umsetzung dieser Variante notwendigen organisatorischen und DV-technischen Bedingungen werden jedoch nicht ausgeführt.
4. Der NOV ist so stabil, daß bei Autorisierungsanfragen stets die Kopfstelle des Kartenherausgebers erreicht wird.

Der Begriff Geheimhaltung in Prämisse 1 bedeutet, daß - anders als beim gegenwärtigen Verfahren, welches sogar veröffentlicht wurde - nur die kartenproduzierenden Verlage das Verfahren kennen müssen. Dieses Verfahren kann auch verbandsweise unterschiedlich gewählt werden.

Die in jüngster Vergangenheit diskutierten Systemangriffe auf den DES bzw. die derzeitigen PIN-Berechnungs- und PIN-Verifikations-Verfahren setzen bekannte Klartexte voraus (Known-Plaintext Attacken). Um einen Angriff erfolgreich durchzuführen, genügen nur wenige Paare von Klartexten und zugehörigen Chiffretexten, um im derzeitig eingesetzten PIN-Generierungsverfahren Instituts- bzw, Pool-Key zu berechnen.

Das Problem wird gelöst, wenn pro Karte ein **kartenspezifischer** Schlüssel für die PIN-Generierung und PIN-Verifizierung verwendet würde. Die o.g. Systemangriffe sind dann sinnlos, weil hierfür schon die Kenntnis der PIN vorausgesetzt wird und ein möglicherweise **kompromittierter** Schlüssel nur für eine Karte Gültigkeit hat. Known-Plaintext Attacken werden dadurch zwar nicht verhindert, jedoch so weit erschwert, daß ihre Durchführung nicht realisitisch erscheint.

Da im Rahmen der Einführung der ec-Karte mit Chip ein Verfahren zur dynamischen Schlüsselgenerierung spezifiziert wurde, bietet es sich an, dieses zu verwenden. Die hierfür benötigten Eingangswerte müssen im folgenden festgelegt werden.

Dynamische Schlüsselgenerierung (DKG)

e*KK(.) bezeichne die Triple-DES Verschlüsselung mit dem Schlüssel KK, d*KK(.) die zugehörige Entschlüsselung.

Zur Ableitung eines kartenindividuellen Schlüssels aus Kartenidentifikationsdaten (CID) und einem Masterkey (Key Generating Key KGK) wird der folgende Algorithmus verwendet:

Ein kartenindividueller Schlüssel KK von 16 Byte Länge wird aus

- KGK (16 Byte),
- CID mit '00' auf das nächste Vielfache von 8 Byte Länge gepaddet und
- dem öffentlich bekannten Initialwert

  I = '52 52 52 52 52 52 52 52 25 25 25 25 25 25 25 25' (16 Byte)

zu

KK = P(d*KGK(H(I,CID))).

berechnet. Hierbei bezeichnen

- P die Funktion "Parity Adjustment", die wie folgt definiert ist:

  Sei $b_1,...,b_8$ die Darstellung eines Byte als Folge von 8 Bit. Dann setzt P das niedrigstwertige Bit b_8 jedes Byte auf ungerade Parität, d. h. b_8 wird in jedem Byte so gesetzt, daß es eine ungerade Anzahl von 1 enthält.

- **d*KGK** die Triple-DES Entschlüsselung mit dem Schlüssel KGK, wobei die beiden 8 Byte langen Blöcke H_1 und H_2 von $H(I,CID) = H_1|H_2$ einzeln entschlüsselt werden (ECB-Mode):

  $$d^*KGK(H(I,CID)) = d^*KGK(H_1) \mid d^*KGK(H_2).$$

- H die in ISO 10118-2 definierte Hash-Funktion, die Werte X mit einer Länge, die ein Vielfaches von 8 Byte ist, mittels des Startwertes I (gemäß Anhang A von ISO 10118-2) auf einen Wert von 16 Byte Länge abbildet. Es werden die um das Parity Adjustment P erweiterten Transformationen u (Ad10) und u' (Ad 01) aus Anhang A von ISO 10118-2 verwendet. H ist rekursiv definiert:

  - Sei $X = x_1|...|x_n$ die Zerlegung des Wertes X in 8 Byte lange Blöcke und $L_0|R_0$ die Zerlegung des vorgegebenen Startwerts I in zwei 8 Byte Blöcke.
  - eK(X) ist die DES-Verschlüsselung eines 8 Byte Wertes mit einem 8 Byte Schlüssel.
  - $\ominus$ sei die bitweise Addition modulo 2 (XOR).

- Die Transformationen Ad10 und Ad01 transformieren 8 Byte Werte K wie folgt:

  Sei $K = k_1,...,k_{64}$ die Darstellung von K als Folge von 64 Bit. Dann ist

  $Ad10(K) = [P](k_1,1,0,k_4,...,k_{64})$

  $Ad01(K) = [P](k_1,0,1,k_4,...,k_{64})$

  [P]: Das Parity Adjustment in Ad10 und Ad01 ist optional. Wenn vor der DES-Verschlüsselung eK(X) keine Paritätsprüfung des Schlüssels K erfolgt, kann P entfallen.

- Dann errechnet sich $L_i|R_i$ aus $L_{i-1}|R_{i-1}$ und x_i wie folgt:

  L_{i-1} bestehe aus den Bits $l_1,...,l_{64}$ und R_{i-1} bestehe aus den Bits $r_1,...,r_{64}$.

  Schritt 1: $L'_i := Ad10(L_{i-1}) = [P](l_1,1,0,l_4,...,l_{64})$

  $R'_i := Ad01(R_{i-1}) = [P](r_1,0,1,r_4,...,r_{64})$

  Schritt 2: $A_i = A_{i[links]}|A_{i[rechts]} = eL'_i(x_i) \oplus x_i$.

  $B_i = B_{i[links]}|B_{i[rechts]} = eR'_i(x_i) \oplus x_i$.

  Schritt 3: $L_i = A_{i[links]}|B_{i[rechts]}$

  $R_i = B_{i[links]}|A_{i[rechts]}$

- $L_n|R_n$ ist dann der Hash-Wert von X unter H:

  $H(I,X) = L_n|R_n$

Soll ein kartenindividueller Schlüssel K von 8 Byte Länge aus KGK, CID und I berechnet werden, wird zuerst der 16 Byte lange Schlüssel KK bestimmt und die linke Hälfte von KK als K verwendet.

Notation

Wir verwenden folgende Notation

$$KK = DKG\ (CID,KGK)$$

für die Ableitung eines kartenindividuellen Schlüssels KK aus den Identifikationsdaten CID unter Verwendung eines Masterkeys KGK.

PIN-Berechnung

Bei strikter Einhaltung der Prämissen ist es nicht notwendig, daß die PIN-Berechnung verbandsübergreifend geregelt werden muß. Die nachfolgende Beschreibung soll daher als ein Realisierungsvorschlag verstanden werden. An dieser Stelle kann auch ganz auf eine PIN-Generierung verzichtet werden und eine vom Kunden selbst gewählte PIN in den Kartenproduktionsprozeß eingebracht werden. Hierfür ist jedoch eine geeignete Infrastruktur zu schaffen.

Zunächst wird durch einen Zufallsprozeß ein Institutsschlüssel KGK_{PINGEN_INST} gebildet. Zur Ableitung eines kartenindividuellen Schlüssels werden die folgenden Daten der Spur 3 verwendet:

Byte	Länge (in Byte)	Wert	Erläuterung
1-5	5	'nn..nn'	Kundenkontonummer
6-7	2	'00 0n'	Kartenfolgenummer der Spur 3
8-11	4	'nn..nn'	Bankleitzahl kontoführendes Institut
12	1	'0n'	Freizügigkeitsschlüssel
13	1	'nn'	Kontoart und Benutzungseinschränkung
14-16	3	'42 64 42'	Filler (ASCII-Codierung für BdB)

Im folgenden werden die Feldinhalte näher erläutert:

Byte 1-5

Byte 1-5 enthalten die 10-stellige BCD-kodierte Kundenkontonummer.

Byte 6-7

Byte 6-7 enthalten rechtsbündig die BCD-kodierte Kartenfolgenummer ggfs. mit vorangestellten '0'.

Byte 8-11

Byte 8-11 enthalten die 8-stellige BCD-kodierte Bankleitzahl des kontoführenden Instituts.

Byte 12

Byte 12 enthält rechtsbündig den BCD-kodierten Freizügigkeitsschlüssel aus Spur 3 mit einer vorangestellten '0'.

Byte 13

Das linke Halbbyte von Byte 13 enthält BCD-kodiert die Kontoart aus Spur 3.

Das rechte Halbbyte von Byte 13 enthält BCD-kodiert die Benutzungseinschränkung aus Spur 3.

Byte 14-16

Filler. Es wird z. Z. die ASCII-Codierung für BdB eingetragen.

Bezeichnen wir diese Daten mit SPUR3, so ergibt sich

$$KK_{PINGEN} = DKG\ (SPUR3,\ KGK_{PINGEN\_INST}).$$

Die PIN wird nun wie im folgenden skizziert berechnet:

Sei X die Konkatenation der folgenden 16 hexadezimalen Werte:

1. - 10. Stelle:	Kontonummer
11. Stelle:	Kartenfolgenummer
12. -15.Stelle:	Routing-Nummer des kartenausgebenden Instituts
16. Stelle:	PIN Modification Value

Die letzte Ziffer kann verwendet werden, um bei Vergabe **aller** Kartenfolgenummern zu einer Kontoverbindung, dennoch eine **neue** PIN errechnen zu können.

X wird Triple-DES verschlüsselt und das Ergebnis sei ein 8 Byte langer String C:

$$C = e^{*}KK_{PINGEN}\ (X)$$

Es werden zwei Alternativen zur Auswahl der PIN aus C vorgeschlagen:

Alternative 1 (ZKA):

Sei $C = (C_1, C_2, \ldots, C_{16})$ mit $C_i \in \mathbf{F}_2^4$ das Ergebnis der Triple-DES Operation mit X. Die Ziffern der PIN[i], $i \in \{1,2,3,4\}$ werden durch folgenden Algorithmus bestimmt:

```
i = 1; wähle f ∈ {6,7,8,9} beliebig aber fest;
FOR j=1 TO 16;
     IF Cj ∈ {0,...,9} THEN {
          PIN[i] = Cj ;
          i=i+1};
     IF i == 5 THEN pin_ok ()
NEXT j ;
i = 1;
FOR j=1 to 16;
     IF Cj ∈ {A,B,C,D,E,F} THEN {
          PIN[i] = Cj - 10;
          i=i+1};
     IF i == 5 THEN pin_ok ()
NEXT j

procedure pin_ok()
     {IF PIN[1] == 0 THEN PIN[1] = f;
     PRINT PIN[1..4];
     STOP;}
```

Die Dezimalisierung wird erreicht, indem die 16 hexadezimalen Ziffern von C von links nach rechts (beginnend beim most significant Byte) nach hexadezimalen Ziffern mit den Werten von X'0' bis X'9' durchsucht wird. Werden bei diesem Vorgang nicht 4 Dezimalziffern gefunden, so wird der Vorgang wiederholt, diesmal aber alle hexadezimalen Ziffern <= X'9' übersprungen und von den verbleibenden 10 (dezimal = X'A') subtrahiert.

Die PIN besteht aus der Konkatenation der 4 gefundenen Ziffern. Ist die führende Ziffer '0', dann wird sie auf '1' gesetzt.

Alternative 2:

Die Dezimalisierung wird durch folgendes Verfahren errreicht:

Es sei $C = (C_1, C_2, \ldots, C_8)$ mit $C_i \in \mathbb{F}_2^8$ das Ergebnis der Triple - DES-Operation mit X. Zur Bestimmung der PIN aus diesem Kryptogramm bilde man die Dezimalzahl

$$C' = C_1 \cdot 256^7 + C_2 \cdot 256^6 + \ldots + C_7 \cdot 256 + C_8$$

und berechne

$$D = C' \text{ MOD } 9000$$

Die PIN - dargestellt als vierstellige Ziffer - wird dann bestimmt zu

$$PIN = D + 1000.$$

Dieses PIN-Berechnungsverfahren liefert (immer noch) eine Ungleichverteilung im Bereich der vierstelligen Ziffern zwischen 1000 und 9999. Die Ziffern zwischen 1000 und $1000 + 2^{64}$ MOD 9000 (=4316) kommen einmal mehr vor als die übrigen. Ihre Auftretenswahrscheinlichkeit beträgt

$$2/12616 \approx 0{,}0001585$$

Insgesamt wird durch diese Vorgehensweise erreicht, daß die 'Schieflage' bei der Verteilung der heutigen PINs weitgehend ausgeräumt wird.

Alternative 3:

Die PINs werden durch einen Zufallszahlengenerator erzeugt, der bei Bedarf eine 8 Byte lange, binär kodierte Zufallszahl bereitstellt. Von Zufallszahlen wird gesprochen, wenn eine Folge $z_1, z_2, \ldots$ von generierten Werten gute Zufallseigenschaften hat.

Als (Pseudo-)Zufallszahlengenerator wird der DES im CBC-Mode mit 8 Byte langem Schlüssel K und ICV = '00..00' eingesetzt werden.

Als Schlüssel K kann hierbei ein 8 Byte langer, geheimer Wert verwendet werden. Als 8 Byte langer Startwert z_0 der Zufallsfolge kann ein Wert verwendet werden, der bei der Initialisierung der Sicherheitsbox eingegeben wird.

Benötigt man eine Zufallszahl, so bildet man

$$z_1 = eK('00..00', z_0)$$

als Zufallszahl und überschreibt z_0 mit z_1.

Danach wird immer wenn eine Zufallszahl benötigt wird,sukzessive die Folge der (Pseudo-) Zufallszahlen $z_2, z_3, \ldots$

$$z_i = eK('00..00', z_{i-1}),\ i > 1$$

gebildet und jeweils die zuvor verwendete Zufallszahl z_{i-1} mit der neu erzeugten Zufallszahl z_i überschrieben. Die PIN wird aus dem Zufallswert $z_i = (z_{i,1}, z_{i,2}, \ldots, z_{i,16})$ mit $z_{i,j} \in F_2^4$ wie folgt ermittelt:

```
k = 1;
FOR j=1 TO 16;
     IF k == 1 THEN {
          IF z_i,j ∈ {1,...,9} THEN {
               PIN[k] = z_i,j ;
               k = k + 1;
               }
          }
     ELSE IF z_i,j ∈ {0,...,9} THEN {
          PIN[k] = z_i,j ;
          k = k + 1;
          }
     IF k == 5 THEN pin_ok ()
NEXT j ;
```

PIN-Verifikation

Die PIN-Verifikation erfolgt mit Hilfe zweier nationaler PIN Verification Values PVN1 und PVN2, die auf dem Magnetstreifen der ec-Karte anstelle von Offset1 und Offset2 aufgebracht werden. Es ist auch möglich ohne die auf den Magnetstreifen geschriebenen PVNs zu verifizieren, sofern diese in einer **Positiv-Datei** des Autorisierungssystems gespeichert sind.

Jeder PVN wird ebenfalls mit Hilfe eines kartenindividuellen Schlüssels und Kartendaten berechnet. Die kartenindividuellen Schlüssel werden mittels des DKG-Algorithmus aus den Institutsschlüsseln $KGK_{PVNGEN1\_INST}$ und $KGK_{PVNGEN2\_INST}$ abgeleitet. Diese Schlüssel können abhängig vom Verfallsjahr geändert werden, so daß eine Kompromittierung dieser Schlüssels zeitlich (für ein Jahr) und räumlich (bezogen auf **ein** Institut) beschränkt bleibt. Zur Ermittlung eines kartenindividuellen Schlüssels werden die oben beschriebenen Daten aus SPUR3 verwendet.

Bezeichnen wir diese Kartenwerte mit SPUR3 so ermittelt sich die kartenindividuellen Schlüssel $KK_{PVNGEN1}$ und $KK_{PVNGEN2}$ aus

$$KK_{PVNGENj} = DKG(SPUR3, KGK_{PVNGENj\_INST}) \text{ für } j = 1,2$$

Alternative 1 (ZKA):

Für die PVN Berechnung wird der folgende aus 16 hexadezimalen Ziffern bestehende Wert X gebildet:

1. - 4. Stelle:	Verfallmonat und -jahr codiert als MMJJ
5. - 11.Stelle:	letzten 7 Stellen der Kontonummer
12. Stelle:	'1'
13. - 16. Stelle:	Klartext-PIN

Alternative 2:

Zur Berücksichtigung von PINs mit mehr als 4 Stellen (z. B. Selbstwahl-PIN), kann X alternativ wie folgt gebildet werden:

1. Stelle:	Länge L der PIN, 11 > L > 3
2. - 5. Stelle:	Verfallmonat und -jahr codiert als MMJJ
6. - (16-L).Stelle:	letzten 11-L Stellen der Kontonummer
(17-L). - 16. Stelle:	L-stellige Klartext-PIN

X wird Triple-DES verschlüsselt und das Ergebnis sei ein 8 Byte langer String C:

$$C_j = e^* KK_{PVNGENj}(X) \text{ für } j = 1,2$$

Eine Dezimalisierung wird nach o.g. Algorithmus mit PVNj[i] anstelle von PIN[i] erreicht, jedoch ohne eine Korrektur der ersten Stelle. Das Ergebnis dieses Vorgangs bildet den PVNj, der an Stelle des Offset j auf die Spur 3 geschrieben werden kann (j = 1,2).

Alternative 3:

Bei Verzicht auf Ersatzautorisierung können beim Bank-Verlag die vollständigen Kryptogramme C_j in der **Positiv-Datei** gespeichert und im Falle der Autorisierung geprüft werden. Hierzu wird der Wert X wie in Alternative 2 gebildet.

Eintragungen auf der Spur 3 des Magnetstreifens (Bereiche der alten Offsets) zum Zwecke der Autorisierung sind dann nicht mehr erforderlich.