FeliCa カードユーザーズマニュアル (非公式版)

切敷 裕大[*1]

2026 年 7 月 24 日

[*1] アンノウン・テクノロジーズ株式会社

おことわり

内容の確かさには万全を期しておりますが、リバースエンジニアリングの性質上、不正確な結果が含まれている場合がありますので、ご了承ください。

本文書では、特筆ない限り以下が前提です。

- ビット順序は、最下位ビットをビット 0 とします。
- データ長は、オクテット単位とします。

目次

第 1 章

はじめに

FeliCa は、交通系 IC カードを筆頭に多くの SF(ストアードフェア) 型電子マネーや身分証明書等に使用されている一方、その仕様において最も重要な暗号に関する仕様は非公開とされており、広く知られていません。暗号に関するセキュリティは、公にされ広く専門家が検証してこそ安全性が担保されるものです。このため、本文書では FeliCa の隠された仕様を明らかにします。本文書は、JIS X 6319-4[4]、FeliCa カード ユーザーズマニュアル 抜粋版 (以下、U-MAN)[3]、felica-tool[2] および Proxmark3[1] を大いに参考にしています。より詳しい情報を知りたい方はこれらも併せてご確認ください。

第 2 章

FeliCa とは

FeliCa は、ソニー株式会社が開発した非接触型 IC カード技術の一つで、NFC Type-F とも呼ばれます。市場にある実装としては FeliCa Standard と FeliCa Lite-S があります。本文書では、前者を取り扱います。

第 3 章

通信プロトコル

カードとカードリーダとの間の通信プロトコルを説明します。ただし、すべてが公になっている物理層およびデータリンク層の説明は割愛し、アプリケーション層のみを取り扱います。

3.1 コマンドパケット

コマンドパケットのデータ構造を以下に示します。(表 3.1)

表 3.1: コマンドパケットのデータ構造

オフセット	長さ	項目
0x00	0x01	コマンドコード
0x01	可変	コマンドデータ

3.1.1 コマンドコード

コマンドの種類を識別するための 1 バイトの値です。

3.1.2 コマンドデータ

コマンドの処理指示を規定するデータで、コマンドごとに形式が異なります。

3.2 レスポンスパケット

レスポンスパケットのデータ構造を以下に示します。(表 3.2)

表 3.2: レスポンスパケットのデータ構造

オフセット	長さ	項目
0x00	0x01	レスポンスコード
0x01	可変	レスポンスデータ

3.2.1 レスポンスコード

レスポンスの種類を識別するための 8 ビットの値です。

3.2.2 レスポンスデータ

レスポンスの処理結果を規定するデータで、コマンドごとに形式が異なります。

3.3 コマンド一覧

各コマンドの概要ならびにコマンドコードおよびレスポンスコードを以下に示します。(表 3.3) ただし、表中ではコマンドコードを CC、レスポンスコードを RC と表記します。また、まだ知られていないコマンドが存在している可能性があります。

表 3.3: コマンド一覧

名称	CC	RC	概要
Polling	0x00	0x01	カードリーダがカードを捕捉および特定する
Request Service	0x02	0x03	エリアやサービスの存在確認と鍵バージョンを取得する
Request Response	0x04	0x05	カードの存在とモードを確認する
Read Without Encryption	0x06	0x07	サービス属性が認証不要のサービスからブロックデータを読み出す
Write Without Encryption	0x08	0x09	サービス属性が認証不要のサービスへブロックデータを書き込む
Search Service Code	0x0A	0x0B	エリアコードとサービスコードを取得する
Request System Code	0x0C	0x0D	カード内に登録されているシステムコードを取得する

次ページに続く

表 3.3: コマンド一覧 (続き)

名称	CC	RC	概要
Request Block Information	0x0E	0x0F	指定したノードに割り当てられているブロック数を取得する
Authentication1	0x10	0x11	カードリーダがカードを DES 暗号方式で認証する
Authentication2	0x12	0x13	カードがカードリーダを DES 暗号方式で認証する
Read	0x14	0x15	サービス属性が認証必要のサービスから DES 暗号方式でブロックデータを読み出す
Write	0x16	0x17	サービス属性が認証必要のサービスへ DES 暗号方式でブロックデータを書き込む
Request Code List	0x1A	0x1B	指定した親ノードに対するノードのリストを反復的に取得する
Request Block Information Ex	0x1E	0x1F	指定したノードに割り当てられているブロック数と空きブロック数を取得する
Set Parameter	0x20	0x21	カード通信パラメータを設定する (暗号化方式およびノードコードサイズ)
Get Container Issue Information	0x22	0x23	コンテナの情報 (フォーマットバージョンや携帯電話のモデルなど) を取得する
Get Area Information	0x24	0x25	指定したエリアに関する情報を取得する
Get Node Property	0x28	0x29	ノードプロパティを取得する
Get Container Property	0x2E	0x2F	コンテナプロパティを取得する
Request Service v2	0x32	0x33	エリアやサービスの存在確認と鍵バージョンを取得する (AES 暗号方式対応)
Internal Authenticate and Read	0x34	0x35	MAC つき通信有効サービスに対して内部認証を行いブロックデータを読み出す
External Authenticate and Write	0x36	0x37	MAC つき通信有効サービスに対して外部認証を行いブロックデータを書き込む

次ページに続く

表 3.3: コマンド一覧 (続き)

名称	CC	RC	概要
Get System Status	0x38	0x39	システムごとの設定状態を取得する
Request Product Information	0x3A	0x3B	製品情報を取得する
Request Specification Version	0x3C	0x3D	カード OS のバージョンを取得する
Reset Mode	0x3E	0x3F	モードを Mode0 にリセットする
Authentication1 v2	0x40	0x41	カードリーダがカードを AES 暗号方式で認証する
Authentication2 v2	0x42	0x43	カードがカードリーダを AES 暗号方式で認証する
Read v2	0x44	0x45	サービス属性が認証必要のサービスから AES 暗号方式でブロックデータを読み出す
Write v2	0x46	0x47	サービス属性が認証必要のサービスへ AES 暗号方式でブロックデータを書き込む
Delete Key	0x48	0x49	AES 鍵と DES 鍵が設定されているノードに対して、DES 鍵の使用を停止する
Update Random ID	0x4C	0x4D	乱数化 ID(IDr) を更新する
Get Container ID	0x70	0x71	モバイル FeliCa からコンテナ ID を取得する
Set Node Property	0x78	0x79	ノードプロパティを設定する
Register Issue ID	0x80	0x81	システムを初期化し発行 ID を登録する
Register Area	0x82	0x83	エリアを登録する
Register Service	0x84	0x85	サービスを登録する
Register Issue ID Ex	0x86	0x87	システムを初期化し発行 ID を登録する
Change System Block	0x8E	0x8F	発行系コマンドの結果を確定する

3.4 製造 ID および製造パラメーター

Polling コマンドのレスポンスデータとして取得できる製造 ID(IDm) および製造パラメータ (PMm) を説明します。

3.4.1 IDm

IDm は、カードリーダがカードを識別するための ID です。カード内に複数のシステムが存在する場合、システムごとに IDm が異なります。

IDm のデータ構造を以下に示します。(表 3.4) ただし、製造者コードの先頭 1 バイトの上位 4 ビットはカード内でのシステム番号を示します。

表 3.4: IDm のデータ構造

オフセット	長さ	項目
0x00	0x02	製造者コード
0x02	0x06	カード識別番号

3.4.2 PMm

PMm のデータ構造を以下に示します。(表 3.5) ただし、ROM 種別および IC 種別を合わせて IC コードといいます。

表 3.5: PMm のデータ構造

オフセット	長さ	項目
0x00	0x01	ROM 種別
0x01	0x01	IC 種別
0x02	0x06	最大応答時間パラメータ

第 4 章

ファイルシステム

FeliCa の内部は、システム・エリア・サービス・ブロックという階層構造になっています。システムはエリアを内包し、エリアは子エリアまたはサービスを内包し、サービスはブロックを内包します。いずれも 1 枚のカードに複数存在することができます。このうち、システム・エリア・サービスはディレクトリのようなメタデータで、ブロックは実際のデータを格納する領域です。これらをまとめてノードと呼びます。ノードにはそれぞれを表すコードがあります。いずれのノードの存在もブロックを消費します。

4.1 システム

システムは、論理的なカードの単位であり、階層構造の最上位にあります。システム (ノード) を表すコードは 0xFFFF です。システムには、以下のシステム定義情報が定義されています。

- システムコード
- 発行 ID 情報
- システム鍵
- システム鍵バージョン
- システムプロパティ

4.1.1 システムコード

システムコードは 2 バイトの値です。紛らわしいですが、システム (ノード) を表すコードとシステムコードは異なる概念です。システムコードには、例えば以下のようなものがあります。(表 4.1)

表 4.1: システムコードの例

システムコード	名称
0x0003	鉄道サイバネティクス領域
0x80CD	フリー領域
0xFE00	共通領域
0xFE0F	管理領域

4.1.2 発行 ID 情報

発行 ID 情報は、各 8 バイトの発行 ID(IDi) と発行パラメータ (PMi) からなります。これは、Authentication2 コマンドまたは Authentication2 v2 コマンドで相互認証を成功させるとレスポンスデータから得ることができます。IDi を特定のアルゴリズムで文字列に変換すると、鉄道サイバネティクス規格の交通系 IC カードの裏面右下に記載されている ID 番号になります。

4.1.3 システム鍵

システム鍵は、DES 暗号方式では 8 バイトの値です。AES 暗号方式の場合は 16 バイトの値です。

4.1.4 システム鍵バージョン

システム鍵バージョンは 2 バイトの値です。

4.1.5 システムプロパティ

詳細不明です。

4.1.6 システム切り替え

カードが、Polling コマンドを受信したり、現在操作されているシステムのものとは異なる IDm 宛のコマンドパケットを受信したりした場合、現在操作されているシステムが切り替わり Mode0 にリセットされます。ただし、Authentication1、Authentication1 v2 または Internal Authenticate and Read コマンドが成功し、システム切り替えが発生した場合には Mode1 になります。

4.2 エリア

エリアは、システムまたは少なくとも 1 つの親エリアに内包されます。エリアには、以下のエリア定義情報が定義されています。

- エリアコード
- エンドサービスコード
- 割り当てブロック数
- エリア鍵
- エリア鍵バージョン
- エリアプロパティ

4.2.1 エリアコード

エリアコードは 2 バイトの値です。第 6 ビットから第 15 ビットがエリア番号、第 0 ビットから第 5 ビットがエリア属性 (表 4.2) を示します。エリアコードはカード内におけるエリアの論理的な始点をも示します。

表 4.2: エリア属性

エリア属性	値
子エリア作成可能エリア	0b000000
子エリア作成不可能エリア	0b000001

4.2.2 エンドサービスコード

カード内におけるエリアの論理的な終点です。

4.2.3 割り当てブロック数

エリアに割り当てられているブロックの数です。

4.2.4 エリア鍵

エリア鍵は、DES 暗号方式では 8 バイトの値です。AES 暗号方式の場合は 16 バイトの値です。

4.2.5　エリア鍵バージョン

エリア鍵バージョンは 2 バイトの値です。

4.2.6　エリアプロパティ

詳細不明です。

4.2.7　エリア 0

システムには、必ず領域が 0x0000 から 0xFFFE までのエリア 0 が含まれます。

4.3　サービス

サービスは、少なくとも 1 つのエリアに内包されます。サービスには、以下のサービス定義情報が定義されています。

- サービスコード
- 割り当てブロック数
- サービス鍵
- サービス鍵バージョン
- サービスプロパティ

4.3.1　サービスコード

サービスコードは 2 バイトの値です。第 6 ビットから第 15 ビットがサービス番号、第 0 ビットから第 5 ビットがサービス属性 (表 4.3) を示します。サービスコードはカード内におけるサービスの論理的な位置をも示します。

表 4.3: サービス属性

サービス属性	アクセス制御	値
ランダムサービス	リード/ライトアクセス: 認証必要	0b001000
	リード/ライトアクセス: 認証不要	0b001001
	リードオンリーアクセス: 認証必要	0b001010
	リードオンリーアクセス: 認証不要	0b001011
サイクリックサービス	リード/ライトアクセス: 認証必要	0b001100
	リード/ライトアクセス: 認証不要	0b001101
	リードオンリーアクセス: 認証必要	0b001110
	リードオンリーアクセス: 認証不要	0b001111
パースサービス	ダイレクトアクセス: 認証必要	0b010000
	ダイレクトアクセス: 認証不要	0b010001
	キャッシュバック/ デクリメントアクセス: 認証必要	0b010010
	キャッシュバック/ デクリメントアクセス: 認証不要	0b010011
	デクリメントアクセス: 認証必要	0b010100
	デクリメントアクセス: 認証不要	0b010101
	リードオンリーアクセス: 認証必要	0b010110
	リードオンリーアクセス: 認証不要	0b010111

4.3.2 割り当てブロック数

サービスに割り当てられているブロックの数です。

4.3.3 サービス鍵

サービス鍵は、DES 暗号方式では 8 バイトの値です。AES 暗号方式の場合は 16 バイトの値です。

4.3.4　サービス鍵バージョン

サービス鍵バージョンは 2 バイトの値です。

4.3.5　サービスプロパティ

リミットパースサービスオプションまたは MAC つき通信オプションが搭載された製品でのみ設定されます。

4.3.6　オーバーラップサービス

複数のサービスコードで同じブロック群を管理することをオーバーラップするといい、同一システム内で同一サービス番号をもつサービスをオーバーラップしたサービスをオーバーラップサービスといいます。U-MAN[3] の「オーバーラップサービス」の節にある、オーバーラップの例を示した図が理解を助けます。

4.4　ブロック

ブロックはカード内の不揮発性メモリにおけるデータの実体を 16 バイト単位に分割したものです。メタデータを格納する以外のブロックはサービスごとに一定数が割り当てられ、サービスを内包するエリアがその割り当て総数を管理します。U-MAN[3] の「エリア」の節にある、エリアによるブロック数管理を示した図が理解を助けます。

第 5 章

モード

カードには、Mode0 から Mode3 までの状態 (モード) が存在します。カードで実行できるコマンドは、これにより制限されています。電源断の状態から電源が供給されると、Mode0 になります。Mode1 以上には DES 暗号方式、AES 暗号方式または MAC つき通信方式の区分があり、それぞれ Authentication1、Authentication1 v2 または Internal Authenticate and Read コマンドを実行すると Mode1 に遷移します。

DES 暗号方式および AES 暗号方式では、Mode1 において Authentication2 または Authentication2 v2 コマンドを実行すると該当方式の Mode2 に遷移します。Mode2 では、相互認証が必要なコマンド群 (Read, Write, Read v2, Write v2 等) を実行できます。発行系コマンドを実行すると、該当方式の Mode3 に遷移します。

一方、MAC つき通信方式には Mode2 および Mode3 が存在しません。Mode1 で実行できるのは External Authenticate and Write コマンドであり、これを実行すると Mode0 に戻ります。

Mode0 以外に遷移すると、カードは Polling コマンドを受け付けなくなります。ただし、システム切り替えのために別システムを指定した Polling コマンドは、いずれのモードでも実行できます。また、いずれのモードにおいても Reset Mode コマンドにより Mode0 に遷移します。現在のモードは、Request Response コマンドで確認できます。より詳細な情報は、U-MAN[3] の「モード」の節を参照してください。

第 6 章

コマンド

コマンドに与えるパラメータとコマンドの詳細について述べます。ただし、コマンドの詳細は、ページ数の都合で一部コマンドを省略します。

6.1 ブロックリスト及びブロックリストエレメント

ブロックリストは、アクセス対象となるサービスおよびブロック番号を特定するための、ブロックリストエレメントの集合です。ブロックリストエレメントには、長さが 2 バイトのもの (表 6.1) と 3 バイトのもの (表 6.2) があります。ただし、表中のオフセットおよび長さはビット単位です。

表 6.1: ブロックリストエレメント (2 バイト) のデータ構造

オフセット	長さ	項目
0	1	長さフラグ (0b1)
1	3	アクセスモード
4	4	サービスコードリスト順番
8	8	ブロック番号・鍵バージョン

表 6.2: ブロックリストエレメント (3 バイト) のデータ構造

オフセット	長さ	項目
0	1	長さフラグ (0b0)

次ページに続く

表 6.2: ブロックリストエレメント (3 バイト) のデータ構造 (続き)

オフセット	長さ	項目
1	3	アクセスモード
4	4	サービスコードリスト順番
8	16	ブロック番号・鍵バージョン (リトルエンディアン)

6.1.1 アクセスモード

ブロックリストエレメントが対象とするノードへのアクセス方法を指定します。(表 6.3)

表 6.3: アクセスモード

アクセスモード	値
パースサービスへのキャッシュバックアクセス以外	0b000
パースサービスへのキャッシュバックアクセス	0b001
鍵変更	0b100

6.2 Polling

Polling コマンドは、カードリーダがカードを捕捉および特定するためのコマンドです。指定したシステムコードを持つシステムの IDm および PMm が取得できます。

6.2.1 コマンドパケットのデータ構造

Polling コマンドパケットのデータ構造を以下に示します。(表 6.4)

表 6.4: Polling コマンドパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	コマンドコード	0x00
0x01	0x02	システムコード (x)	$0x0000 \leq x \leq 0xFFFF$ (ただし、各バイト 0xFF はワイルドカード。) ビッグエンディアン

次ページに続く

表 6.4: Polling コマンドパケットのデータ構造 (続き)

オフセット	長さ	項目	値または備考
0x03	0x01	リクエストコード	0x00: 要求なし、0x01: システムコード要求、0x02: 通信性能要求
0x04	0x01	応答可能な最大スロット数の指定	0x00, 0x01, 0x03, 0x07, 0x0F (詳細は U-MAN[3] の「Polling」の節にある、タイムスロット規定の表を参照)

6.2.2 レスポンスパケットのデータ構造

Polling レスポンスパケットのデータ構造を以下に示します。(表 6.5)

表 6.5: Polling レスポンスパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	レスポンスコード	0x01
0x01	0x08	IDm	-
0x09	0x08	PMm	-
0x11	0x02	リクエストデータ	コマンドにおけるリクエストコードが 0x00 以外であり、かつ製品が対応するリクエストコードが指定された場合のみ返送

対応していないリクエストコードを指定した場合、レスポンスパケットにリクエストデータは付加されません。リクエストコードを指定してもリクエストデータが返送されない場合があることを前提に実装してください。

6.3 Request Service

Request Service コマンドは、エリアやサービスの存在確認と鍵バージョンを取得するためのコマンドです。

6.3.1 コマンドパケットのデータ構造

Request Service コマンドパケットのデータ構造を以下に示します。(表 6.6)

表 6.6: Request Service コマンドパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	コマンドコード	0x02
0x01	0x08	IDm	-
0x09	0x01	ノード数 (n)	$0x01 \leq n \leq 0x20$
0x0A	2n	ノードコードリスト	リトルエンディアン

6.3.2 レスポンスパケットのデータ構造

Request Service レスポンスパケットのデータ構造を以下に示します。(表 6.7)

表 6.7: Request Service レスポンスパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	レスポンスコード	0x03
0x01	0x08	IDm	-
0x09	0x01	ノード数 (n)	$0x01 \leq n \leq 0x20$
0x0A	2n	ノード鍵バージョンリスト	ただし、ノードが存在しない場合には 0xFFFF。リトルエンディアン

6.4 Request Response

Request Response コマンドは、カードの存在とモードを確認するためのコマンドです。

6.4.1 コマンドパケットのデータ構造

Request Response コマンドパケットのデータ構造を以下に示します。(表 6.8)

表 6.8: Request Response コマンドパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	コマンドコード	0x04
0x01	0x08	IDm	-

6.4.2 レスポンスパケットのデータ構造

Request Response レスポンスパケットのデータ構造を以下に示します。(表 6.9)

表 6.9: Request Response レスポンスパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	レスポンスコード	0x05
0x01	0x08	IDm	-
0x09	0x01	モード (n)	0x00 $\leq$ n $\leq$ 0x03

6.5 Read Without Encryption

Read Without Encryption コマンドは、サービス属性が認証不要のサービスからブロックデータを読み出すためのコマンドです。

6.5.1 コマンドパケットのデータ構造

Read Without Encryption コマンドパケットのデータ構造を以下に示します。(表 6.10)

表 6.10: Read Without Encryption コマンドパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	コマンドコード	0x06
0x01	0x08	IDm	-
0x09	0x01	サービス数 (m)	0x01 $\leq$ m $\leq$ 0x10
0x0A	2m	サービスコードリスト	リトルエンディアン
-	0x01	ブロック数 (n)	0x01 $\leq$ n $\leq$ 製品の最大読み出し可能ブロック数
-	2n $\leq$ N $\leq$ 3n	ブロックリスト	-

6.5.2 レスポンスパケットのデータ構造

Read Without Encryption レスポンスパケットのデータ構造を以下に示します。(表 6.11)

表 6.11: Read Without Encryption レスポンスパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	レスポンスコード	0x07
0x01	0x08	IDm	-
0x09	0x01	ステータスフラグ 1	0x00: 正常
0x0A	0x01	ステータスフラグ 2	0x00: 正常
0x0B	0x01	ブロック数 (n)	0x01 ≤ n ≤ 製品の最大同時読み出し可能ブロック数
0x0C	16n	ブロックデータ	-

6.6 Write Without Encryption

Write Without Encryption コマンドは、サービス属性が認証不要のサービスにブロックデータを書き込むためのコマンドです。

6.6.1 コマンドパケットのデータ構造

Write Without Encryption コマンドパケットのデータ構造を以下に示します。(表 6.12)

表 6.12: Write Without Encryption コマンドパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	コマンドコード	0x08
0x01	0x08	IDm	-
0x09	0x01	サービス数 (m)	0x01 ≤ m ≤ 0x10
0x0A	2m	サービスコードリスト	リトルエンディアン
-	0x01	ブロック数 (n)	0x01 ≤ n ≤ 製品の最大同時書き込み可能ブロック数
-	2n ≤ N ≤ 3n	ブロックリスト	-
-	16n	ブロックデータ	-

6.6.2 レスポンスパケットのデータ構造

Write Without Encryption レスポンスパケットのデータ構造を以下に示します。(表 6.13)

表 6.13: Write Without Encryption レスポンスパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	レスポンスコード	0x09
0x01	0x08	IDm	-
0x09	0x01	ステータスフラグ 1	0x00: 正常
0x0A	0x01	ステータスフラグ 2	0x00: 正常

6.7 Search Service Code

Search Service Code コマンドは、エリアコードとサービスコードを取得するためのコマンドです。

6.7.1 コマンドパケットのデータ構造

Search Service Code コマンドパケットのデータ構造を以下に示します。(表 6.14)

表 6.14: Search Service Code コマンドパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	コマンドコード	0x0A
0x01	0x08	IDm	-
0x09	0x02	ノードインデックス	0x0000 $\leq$ n $\leq$ 0xFFFF リトルエンディアン

6.7.2 レスポンスパケットのデータ構造

Search Service Code レスポンスパケットのデータ構造を以下に示します。(表 6.15)

表 6.15: Search Service Code レスポンスパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	レスポンスコード	0x0B
0x01	0x08	IDm	-
0x09	0x02 または 0x04	ノードコード	2 バイト: サービスコード、4 バイト: エリアコードおよびエリアエンドサービスコード。 いずれもリトルエンディアン。0xFFFF の場合は該当するノードが存在しないことを示す

ノードインデックスを 0 から順に増やしながら本コマンドを実行し、レスポンスのノードコードが 0xFFFF となった時点を終端として、カード内のノードを走査できます。

6.8 Request System Code

Request System Code コマンドは、カード内に登録されているシステムコードを取得するためのコマンドです。

6.8.1 コマンドパケットのデータ構造

Request System Code コマンドパケットのデータ構造を以下に示します。(表 6.16)

表 6.16: Request System Code コマンドパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	コマンドコード	0x0C
0x01	0x08	IDm	-

6.8.2 レスポンスパケットのデータ構造

Request System Code レスポンスパケットのデータ構造を以下に示します。(表 6.17)

表 6.17: Request System Code レスポンスパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	レスポンスコード	0x0D
0x01	0x08	IDm	-
0x09	0x01	システムコード数 (n)	-
0x0A	2n	システムコードリスト	システム 0 から順に列挙。ビッグエンディアン。

6.9 Request Block Information

Request Block Information コマンドは、指定したノードに割り当てられているブロック数を取得するためのコマンドです。モバイル FeliCa のみ対応しています。

6.9.1 コマンドパケットのデータ構造

Request Block Information コマンドパケットのデータ構造を以下に示します。(表 6.18)

表 6.18: Request Block Information コマンドパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	コマンドコード	0x0E
0x01	0x08	IDm	-
0x09	0x01	ノードコード数 (n)	-
0x0A	2n	ノードコードリスト	リトルエンディアン

6.9.2 レスポンスパケットのデータ構造

Request Block Information レスポンスパケットのデータ構造を以下に示します。(表 6.19)

表 6.19: Request Block Information レスポンスパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	レスポンスコード	0x0F
0x01	0x08	IDm	-

次ページに続く

表 6.19: Request Block Information レスポンスパケットのデータ構造 (続き)

オフセット	長さ	項目	値または備考
0x09	0x01	ブロック情報数 (n)	-
0x0A	2n	ブロック情報リスト	各ノードに割り当てられているブロック数を示す

6.10 Authentication1

Authentication1 コマンドは、DES 方式でカード側認証チャレンジを開始するためのコマンドです。

6.10.1 コマンドパケットのデータ構造

Authentication1 コマンドパケットのデータ構造は、表 7.1 を参照してください。

6.10.2 レスポンスパケットのデータ構造

Authentication1 レスポンスパケットのデータ構造は、表 7.3 を参照してください。

6.11 Authentication2

Authentication2 コマンドは、DES 方式の相互認証を完了するためのコマンドです。

6.11.1 コマンドパケットのデータ構造

Authentication2 コマンドパケットのデータ構造は、表 7.2 を参照してください。

6.11.2 レスポンスパケットのデータ構造

Authentication2 レスポンスパケットのデータ構造は、表 7.4 を参照してください。

6.12 Read

Read コマンドは、相互認証後の DES セッションでブロックデータを読み出すためのコマンドです。

6.12.1 コマンドパケットのデータ構造

Read コマンドの暗号化前内部ペイロード構造は、表 7.12 を参照してください。

6.12.2 レスポンスパケットのデータ構造

Read レスポンスの暗号化前内部ペイロード構造は、表 7.13 を参照してください。

6.13 Write

Write コマンドは、相互認証後の DES セッションでブロックデータを書き込むためのコマンドです。

6.13.1 コマンドパケットのデータ構造

Write コマンドの暗号化前内部ペイロード構造は、表 7.14 を参照してください。アクセスモード 0b100(鍵変更) のブロックデータに格納する鍵変更パッケージは、表 7.16 および同節の生成アルゴリズムを参照してください。

6.13.2 レスポンスパケットのデータ構造

Write レスポンスの暗号化前内部ペイロード構造は、表 7.15 を参照してください。

6.14 Get Node Property

Get Node Property コマンドは、ノードプロパティを取得するためのコマンドです。リミットパースサービスオプションまたは MAC つき通信オプションのノードプロパティが取得できます。本コマンドは、一部の AES カード製品および AES/DES カード製品にのみ搭載されています。

6.14.1 コマンドパケットのデータ構造

Get Node Property コマンドパケットのデータ構造を以下に示します。(表 6.20)

表 6.20: Get Node Property コマンドパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	コマンドコード	0x28
0x01	0x08	IDm	-
0x09	0x01	取得対象	0x00: リミットパースサービス、0x01: MAC つき通信有効サービス
0x0A	0x01	ノード数 (n)	0x01 ≤ n ≤ 0x10
0x0B	2n	ノードコードリスト	リトルエンディアン

6.14.2 レスポンスパケットのデータ構造

Get Node Property レスポンスパケットのデータ構造を以下に示します。(表 6.21)

表 6.21: Get Node Property レスポンスパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	レスポンスコード	0x29
0x01	0x08	IDm	-
0x09	0x01	ステータスフラグ 1	0x00: 正常
0x0A	0x01	ステータスフラグ 2	0x00: 正常
0x0B	0x01	ノード数 (n)	ステータスフラグ 1 が 0x00 の場合のみ返送
0x0C	mn	ノードプロパティ	ステータスフラグ 1 が 0x00 の場合のみ返送。ノードコードリストの順に列挙される。 取得対象が 0x00 のとき m=10、0x01 のとき m=1

取得対象に 0x00(リミットパースサービス) を指定した場合のノードプロパティのデータ構造を以下に示します。(表 6.22)

表 6.22: リミットパースサービスのノードプロパティのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	リミットパースサービス有効フラグ	0x01: 有効、0x00: 無効

次ページに続く

表 6.22: リミットパースサービスのノードプロパティのデータ構造 (続き)

オフセット	長さ	項目	値または備考
0x01	0x04	上限値	リトルエンディアン、2 の補数表現。無効の場合は全バイトが 0xFF
0x05	0x04	下限値	リトルエンディアン、2 の補数表現。無効の場合は全バイトが 0xFF
0x09	0x01	世代番号	無効の場合は 0xFF

取得対象に 0x01(MAC つき通信有効サービス) を指定した場合のノードプロパティは、MAC つき通信有効フラグ 1 バイトのみです。値は 0x01 が有効、0x00 が無効を表します。

6.15 Request Service v2

Request Service v2 コマンドは、暗号方式識別子と鍵バージョン情報を取得するためのコマンドです。

6.15.1 コマンドパケットのデータ構造

Request Service v2 コマンドパケットのデータ構造を以下に示します。(表 6.23)

表 6.23: Request Service v2 コマンドパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	コマンドコード	0x32
0x01	0x08	IDm	-
0x09	0x01	ノード数 (n)	$0x01 \leq n \leq 0x20$
0x0A	2n	ノードコードリスト	リトルエンディアン

6.15.2 レスポンスパケットのデータ構造

Request Service v2 レスポンスパケットのデータ構造を以下に示します。(表 6.24)

表 6.24: Request Service v2 レスポンスパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	レスポンスコード	0x33
0x01	0x08	IDm	-
0x09	0x01	ステータスフラグ 1	0x00: 正常
0x0A	0x01	ステータスフラグ 2	0x00: 正常
0x0B	0x01	暗号方式識別子	ステータスフラグ 1 が 0x00 の場合のみ返送
0x0C	0x01	ノード数 (n)	ステータスフラグ 1 が 0x00 の場合のみ返送。0x01 ≤ n ≤ 0x20
0x0D	2n または 4n	鍵バージョンリスト	ステータスフラグ 1 が 0x00 の場合のみ返送。 暗号方式識別子が 0x41 または 0x43 のときは 4n バイト (前半 AES, 後半 DES)。 それ以外は 2n バイト

6.16 Get System Status

Get System Status コマンドは、システムごとの設定状態を取得するためのコマンドです。

6.16.1 コマンドパケットのデータ構造

Get System Status コマンドパケットのデータ構造を以下に示します。(表 6.25)

表 6.25: Get System Status コマンドパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	コマンドコード	0x38
0x01	0x08	IDm	-
0x09	0x02	予約領域	0x0000

6.16.2 レスポンスパケットのデータ構造

Get System Status レスポンスパケットのデータ構造を以下に示します。(表 6.26)

表 6.26: Get System Status レスポンスパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	レスポンスコード	0x39
0x01	0x08	IDm	-
0x09	0x01	ステータスフラグ 1	0x00: 正常
0x0A	0x01	ステータスフラグ 2	0x00: 正常
0x0B	0x01	フラグ	詳細不明
0x0C	0x01	データ長 (n)	-
0x0D	n	データ	詳細不明

フラグおよびデータの意味は詳細不明です。

6.17 Request Specification Version

Request Specification Version コマンドは、カード OS のバージョンを取得するためのコマンドです。

6.17.1 コマンドパケットのデータ構造

Request Specification Version コマンドパケットのデータ構造を以下に示します。(表 6.27)

表 6.27: Request Specification Version コマンドパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	コマンドコード	0x3C
0x01	0x08	IDm	-
0x09	0x02	予約領域	0x0000

6.17.2 レスポンスパケットのデータ構造

Request Specification Version レスポンスパケットのデータ構造を以下に示します。(表 6.28)

表 6.28: Request Specification Version レスポンスパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	レスポンスコード	0x3D
0x01	0x08	IDm	-
0x09	0x01	ステータスフラグ 1	0x00: 正常
0x0A	0x01	ステータスフラグ 2	0x00: 正常
0x0B	0x01	フォーマットバージョン	0x00 固定。ステータスフラグ 1 が 0x00 の場合のみ返送
0x0C	0x02	基本バージョン	ステータスフラグ 1 が 0x00 の場合のみ返送。リトルエンディアン
0x0E	0x01	オプション数 (m)	ステータスフラグ 1 が 0x00 の場合のみ返送
0x0F	2m	オプションバージョンリスト	ステータスフラグ 1 が 0x00 の場合のみ返送。リトルエンディアン

オプションバージョンリストの各要素の割り当てを以下に示します。(表 6.29)

表 6.29: オプションバージョンリストの割り当て

位置	オプション
D0-D1	DES オプション
D2-D3	0x00, 0x00 (モバイル FeliCa では固有のバージョンが設定されている場合がある)
D4-D5	拡張オーバーラップオプション
D6-D7	リミットパースサービスオプション
D8-D9	MAC つき通信オプション
D10-D11	乱数化 ID オプション

基本バージョンおよび各オプションバージョンは、いずれも 2 バイトのデータです。上位 4

ビットは 0b1000 固定であり、残りの 12 ビットが BCD 形式のバージョン値を表します。例えば、バージョン 5.0.0 は 0x500 となります。オプションの有無およびバージョンの値は製品ごとに異なります。

6.18 Reset Mode

Reset Mode コマンドは、モードを Mode0 にリセットするためのコマンドです。

6.18.1 コマンドパケットのデータ構造

Reset Mode コマンドパケットのデータ構造を以下に示します。(表 6.30)

表 6.30: Reset Mode コマンドパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	コマンドコード	0x3E
0x01	0x08	IDm	-
0x09	0x02	予約領域	0x0000

6.18.2 レスポンスパケットのデータ構造

Reset Mode レスポンスパケットのデータ構造を以下に示します。(表 6.31)

表 6.31: Reset Mode レスポンスパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	レスポンスコード	0x3F
0x01	0x08	IDm	-
0x09	0x01	ステータスフラグ 1	0x00: 正常
0x0A	0x01	ステータスフラグ 2	0x00: 正常

6.19 Authentication1 v2

Authentication1 v2 コマンドは、AES 方式でカード側認証チャレンジを開始するためのコマンドです。

6.19.1 コマンドパケットのデータ構造

Authentication1 v2 コマンドパケットのデータ構造は、表 7.6 を参照してください。

6.19.2 レスポンスパケットのデータ構造

Authentication1 v2 レスポンスパケットのデータ構造は、表 7.8 を参照してください。

6.20 Authentication2 v2

Authentication2 v2 コマンドは、AES 方式の相互認証を完了するためのコマンドです。

6.20.1 コマンドパケットのデータ構造

Authentication2 v2 コマンドパケットのデータ構造は、表 7.7 を参照してください。

6.20.2 レスポンスパケットのデータ構造

Authentication2 v2 レスポンスパケットのデータ構造は、表 7.9 を参照してください。

6.21 Read v2

Read v2 コマンドは、相互認証後の AES セッションでブロックデータを読み出すためのコマンドです。

6.21.1 コマンドパケットのデータ構造

Read v2 コマンドの暗号化前内部ペイロード構造は、表 7.12 を参照してください。

6.21.2 レスポンスパケットのデータ構造

Read v2 レスポンスの暗号化前内部ペイロード構造は、表 7.13 を参照してください。

6.22 Write v2

Write v2 コマンドは、相互認証後の AES セッションでブロックデータを書き込むためのコマンドです。

6.22.1 コマンドパケットのデータ構造

Write v2 コマンドの暗号化前内部ペイロード構造は、表 7.14 を参照してください。鍵変更 (アクセスモード 0b100) は Write v2 では有効ではありません。

6.22.2 レスポンスパケットのデータ構造

Write v2 レスポンスの暗号化前内部ペイロード構造は、表 7.15 を参照してください。

6.23 Register Issue ID

Register Issue ID コマンドは、発行 ID 関連情報を登録するための発行系セキュアコマンドです。

6.23.1 コマンドパケットのデータ構造

Register Issue ID コマンドの暗号化前内部ペイロード構造は、表 7.17 を参照してください。

6.23.2 レスポンスパケットのデータ構造

Register Issue ID レスポンスの暗号化前内部ペイロードは、ステータスフラグ 1/2 に加え、成功時のみ残りブロック数 (2 バイト) を返します。

6.24 Register Area

Register Area コマンドは、エリアを登録するための発行系セキュアコマンドです。

6.24.1 コマンドパケットのデータ構造

Register Area コマンドの暗号化前内部ペイロード構造は、表 7.18 を参照してください。

6.24.2 レスポンスパケットのデータ構造

Register Area レスポンスの暗号化前内部ペイロードは、ステータスフラグ 1/2 の 2 バイトです。

6.25 Register Service

Register Service コマンドは、サービスを登録するための発行系セキュアコマンドです。

6.25.1 コマンドパケットのデータ構造

Register Service コマンドの暗号化前内部ペイロード構造は、表 7.19 を参照してください。

6.25.2 レスポンスパケットのデータ構造

Register Service レスポンスの暗号化前内部ペイロードは、ステータスフラグ 1/2 に加え、成功時のみ残りブロック数 (2 バイト) を返します。

6.26 Change System Block

Change System Block コマンドは、発行系コマンドの結果を確定するための発行系セキュアコマンドです。

6.26.1 コマンドパケットのデータ構造

Change System Block コマンドの暗号化前内部ペイロード構造は、表 7.20 を参照してください。

6.26.2 レスポンスパケットのデータ構造

Change System Block レスポンスの暗号化前内部ペイロードは、ステータスフラグ 1/2 の 2 バイトです。

6.27 ステータスフラグ

ステータスフラグは、コマンド処理の結果を表す値で、ステータスフラグ 1 とステータスフラグ 2 からなります。

6.27.1 ステータスフラグ 1

ステータスフラグ 1 は、コマンド処理の成否やエラーが発生したブロック位置またはサービス位置を示します。(表 6.32)

表 6.32: ステータスフラグ 1 の値と意味

値	意味
0x00	正常終了
0xFF	コマンドパケットにリストを含まないコマンドでのエラーまたはリストに依存しないエラー
上記以外	エラーが発生したリスト上の位置

エラーが発生した位置の表現形式は製品によって 2 種類あり、いずれであるかはステータスフラグ 1 の値だけでは判別できません。(表 6.33) 例えば、ブロックリストの 10 番目に指定したノードでエラーが発生した場合、順番形式では 0x0A を、ビットマップ形式では 0x02 を返します。

表 6.33: ステータスフラグ 1 におけるエラー位置の表現形式

形式	内容
順番形式	エラーが発生したブロックリストまたはサービスコードリスト上の位置 (1 始まり) をそのまま設定する
ビットマップ形式	各ビットがリスト上の位置に対応し、1 がエラーあり、0 がエラーなしを表す。ビット 0 が 1 番目または 9 番目、ビット 1 が 2 番目または 10 番目、以下同様にビット 6 が 7 番目または 15 番目に対応し、ビット 7 は 8 番目に対応する

6.27.2 ステータスフラグ 2

ステータスフラグ 2 は、エラーの詳細内容を示します。(表 6.34) また、すべての製品で共通の仕様ではない「カード固有仕様」があります。(表 6.35)

表 6.34: ステータスフラグ 2 の値と意味

値	意味
0x00	正常終了
0x01	パースのデクリメント時に、計算結果がゼロ未満になるまたはキャッシュバック時に計算結果が 4 バイトを超える数になる
0x02	パースのキャッシュバック時に、指定されたデータがキャッシュバックデータの値を超えている
0x03	リミットパースサービスの書き込み時に、パースデータが上限値と下限値との間に入らない
0x70	メモリ異常 (致命的エラー)
0x71	メモリ書き換え回数が上限を超えている (警告であり、書き込み処理は行われる)

0x71 は警告であるため、書き込み処理自体は実行されます。書き換え回数の上限値は製品ごとに異なり、このときステータスフラグ 1 が 0x00 となる製品と 0xFF となる製品があります。

表 6.35: ステータスフラグ 2 の値と意味 (カード固有仕様)

値	意味
0xA1	コマンドで指定されたサービス数またはノード数が規定値の範囲外である
0xA2	コマンドで指定されたブロック数が製品規定値の範囲外である
0xA3	ブロックリストエレメントで指定されたサービスコードリスト順番が、コマンドで指定されたサービス数または相互認証時に指定したサービス数の範囲外である
0xA4	コマンドで指定されたエリアコードのエリア属性またはサービスコードのサービス属性が誤っている
0xA5	コマンドで指定されたエリアまたはサービスにアクセスできない、またはコマンドで指定されたパラメーターが成功条件を満たしていない
0xA6	ブロックリストエレメントで指定されたサービスコードリスト順番で指定されたアクセス先またはノードコードリストで指定されたノードが存在しない
0xA7	ブロックリストエレメントで指定されたアクセスモードが誤っている
0xA8	ブロックリストエレメントで指定されたブロック番号がサービスに割り当てられているブロック数を超えている

次ページに続く

表 6.35: ステータスフラグ 2 の値と意味 (カード固有仕様) (続き)

値	意味
0xA9	発行コマンドにおいて書き込みに失敗した
0xAA	鍵変更に失敗した
0xAB	発行コマンドにおいてパッケージパリティまたはパッケージ MAC が不正である
0xAC	発行コマンドにおいてパラメーターが不正である
0xAD	登録しようとしたサービスがすでに存在している
0xAE	発行コマンドにおいてシステムコードが不正である
0xAF	コマンドで指定されたサイクリックサービスへの同時書き込みブロック数がサービスに割り当てられているブロック数を超えている
0xC0	発行コマンドにおいてパッケージ識別子が不正である
0xC1	発行コマンドにおいてパッケージ内とパッケージ外のパラメータが不一致である
0xC2	発行コマンドが無効化されている
0xC3	コマンドで指定されたノードの属性が誤っている

第 7 章

相互認証およびセキュアメッセージングアルゴリズム

7.1 Authentication1 / Authentication2 (DES)

7.1.1 コマンドパケットのデータ構造

Authentication1 コマンドパケットのデータ構造を以下に示します。(表 7.1)

表 7.1: Authentication1 コマンドパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	コマンドコード	0x10
0x01	0x08	IDm	-
0x09	0x01	エリア数 (a)	-
0x0A	2a	エリアコードリスト	リトルエンディアン
-	0x01	サービス数 (s)	-
-	2s	サービスコードリスト	リトルエンディアン
-	0x08	チャレンジ 1A	8 バイト

Authentication2 コマンドパケットのデータ構造を以下に示します。(表 7.2)

表 7.2: Authentication2 コマンドパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	コマンドコード	0x12
0x01	0x08	IDm	-
0x09	0x08	チャレンジ 2B	8 バイト

7.1.2 レスポンスパケットのデータ構造

Authentication1 レスポンスパケットのデータ構造を以下に示します。(表 7.3)

表 7.3: Authentication1 レスポンスパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	レスポンスコード	0x11
0x01	0x08	IDm	-
0x09	0x08	チャレンジ 1B	8 バイト
0x11	0x08	チャレンジ 2A	8 バイト

Authentication2 レスポンスパケットのデータ構造を以下に示します。(表 7.4) 本レスポンスは IDm を含まず、レスポンスコードに続く部分の全体が暗号化されています。

表 7.4: Authentication2 レスポンスパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	レスポンスコード	0x13
0x01	0x20	暗号化ペイロード	乱数 R_2 を鍵とする DES-CBC(IV=0) で暗号化されている。復号後の構造は表 7.5 を参照

復号後のペイロードのデータ構造を以下に示します。(表 7.5)

表 7.5: Authentication2 レスポンスの復号後ペイロードのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x02	トランザクション番号 (TN)	リトルエンディアン

次ページに続く

表 7.5: Authentication2 レスポンスの復号後ペイロードのデータ構造 (続き)

オフセット	長さ	項目	値または備考
0x02	0x06	トランザクション ID (TID)	乱数 R_1 の後半 6 バイトと一致する
0x08	0x08	発行 ID (IDi)	-
0x10	0x08	発行パラメータ (PMi)	-
0x18	0x08	MAC	先行する 24 バイトに対して算出される

7.1.3 DES 相互認証アルゴリズム

DES 相互認証に用いる主要パラメータは以下で導出します。

- システム鍵を K_{sys}、エリア鍵列を $K_{\mathrm{area},i}$、サービス鍵列を $K_{\mathrm{srv},j}$ とすると、

$$G_0 = K_{\mathrm{sys}}, \quad G_i = \mathrm{DES}_{K_{\mathrm{area},i}}(G_{i-1})$$

$$K_{\mathrm{group}} = G_m, \quad U_0 = K_{\mathrm{group}}, \quad U_j = \mathrm{DES}_{K_{\mathrm{srv},j}}(U_{j-1}), \quad K_{\mathrm{user}} = U_n$$

  でグループサービス鍵 K_{group} とユーザサービス鍵 K_{user} を得ます。
- IDm を 8 バイトベクトルとし、

$$L = K_{\mathrm{group}} \oplus IDm$$

$$\alpha = \mathrm{DES}_L(K_{\mathrm{user}}), \quad \beta = \mathrm{DES}_\alpha(L)$$

  を計算します。
- 2-key 3DES(鍵順 K_1-K_2-K_1) で乱数を交換します。R_1, R_2 を乱数とすると、

$$C_{1A} = 3\mathrm{DES}_{\alpha,L}(R_1), \quad C_{1B} = 3\mathrm{DES}_{L,\beta}(R_1)$$

$$C_{2A} = 3\mathrm{DES}_{L,\beta}(R_2), \quad C_{2B} = 3\mathrm{DES}_{\alpha,L}(R_2)$$

  となります。
- Authentication2 の平文は

$$\mathrm{TN}(2) \parallel \mathrm{TID}(6) \parallel \mathrm{IDi}(8) \parallel \mathrm{PMi}(8)$$

  です。ここで TID は R_1 の後半 6 バイトを用います。発行 ID 情報として用いるデータは後半 16 バイトの IDi(8) ‖ PMi(8) です。

7.2 Authentication1 v2 / Authentication2 v2 (AES)

7.2.1 コマンドパケットのデータ構造

Authentication1 v2 コマンドパケットのデータ構造を以下に示します。(表 7.6)

表 7.6: Authentication1 v2 コマンドパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	コマンドコード	0x40
0x01	0x08	IDm	-
0x09	0x01	オペレーションパラメータ	-
0x0A	0x01	ノード数 (n)	-
0x0B	2n	ノードコードリスト	リトルエンディアン
-	0x10	チャレンジ 1A	16 バイト

Authentication2 v2 コマンドパケットのデータ構造を以下に示します。(表 7.7)

表 7.7: Authentication2 v2 コマンドパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	コマンドコード	0x42
0x01	0x08	IDm	-
0x09	0x10	チャレンジ 2B	16 バイト

7.2.2 レスポンスパケットのデータ構造

Authentication1 v2 レスポンスパケットのデータ構造を以下に示します。(表 7.8)

表 7.8: Authentication1 v2 レスポンスパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	レスポンスコード	0x41
0x01	0x08	IDm	-

次ページに続く

表 7.8: Authentication1 v2 レスポンスパケットのデータ構造 (続き)

オフセット	長さ	項目	値または備考
0x09	0x10	チャレンジ 1B	16 バイト
0x19	0x10	チャレンジ 2A	16 バイト
0x29	0x04	チャレンジ 3C	4 バイト

Authentication2 v2 レスポンスパケットのデータ構造を以下に示します。(表 7.9) 本レスポンスも IDm を含みません。トランザクション番号のみ平文で、ペイロードと MAC が暗号化されています。

表 7.9: Authentication2 v2 レスポンスパケットのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	レスポンスコード	0x43
0x01	0x02	トランザクション番号 (TN)	リトルエンディアン。暗号化されない
0x03	0x10	暗号化ペイロード	AES-128-OFB で暗号化されている。復号後の構造は表 7.10 を参照
0x13	0x08	暗号化 MAC	同じ OFB ストリームの後段で暗号化されている

復号後のペイロードのデータ構造を以下に示します。(表 7.10)

表 7.10: Authentication2 v2 レスポンスの復号後ペイロードのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x08	発行 ID (IDi)	-
0x08	0x08	発行パラメータ (PMi)	-

7.2.3 AES 相互認証アルゴリズム

AES 相互認証に用いる主要パラメータは以下で導出します。

- 16 バイト定数を $I_0 = (01\ 01\ 00 \cdots 00\ 80)$、ノード鍵列を $K_{\mathrm{node},j}$ とすると、

$$S_0 = I_0, \quad S_j = \mathrm{AES}_{K_{\mathrm{node},j}}(S_{j-1}), \quad K_{\mathrm{group}} = S_n$$

でグループ鍵 K_{group} を得ます。

- グループ鍵 K_{group}、個別化コード K_{ind} として

$$H = K_{\mathrm{group}} \oplus K_{\mathrm{ind}}$$

を計算します。

- 16 バイトコンテキストブロック $B(\mathrm{prefix}, IDm)$ を

$$B[0..1] = \mathrm{prefix},\ B[2..5] = 0,\ B[6..13] = IDm,\ B[14..15] = 0x01\ 0x00$$

で定義し、

$$\alpha = \mathrm{AES}_H(B([0x01, 0x02], IDm))$$

$$\beta = \mathrm{AES}_H(B([0x02, 0x02], IDm))$$

を得ます。

- チャレンジ 3C(4 バイト) から

$$\beta' = \beta \oplus (C_{3C} \parallel 0x00^{12})$$

を作り、

$$C_{1A} = \mathrm{AES}_\alpha(R_1),\ C_{1B} = \mathrm{AES}_{\beta'}(R_1),\ C_{2A} = \mathrm{AES}_{\beta'}(R_2),\ C_{2B} = \mathrm{AES}_\alpha(R_2)$$

で認証値を生成・検証します。

- セキュアメッセージングで使う TID は R_1 から導出し、

$$\mathrm{TID} = R_1[2..7]$$

(6 バイト、先頭を 0 として 2 バイト目から 7 バイト目) とします。

- Authentication2 v2 レスポンスの復号後ペイロードは

$$\mathrm{IDi(8)} \parallel \mathrm{PMi(8)}$$

です。

- Authentication2 v2 成功後、R_2 からセッション鍵を導出します。

$$K_{\mathrm{enc}} = \mathrm{AES}_{R_2}(01\ 03\ 00 \cdots 00\ 01\ 00)$$

$$K_{\mathrm{mac}} = \mathrm{AES}_{R_2}(02\ 03\ 00 \cdots 00\ 01\ 00)$$

7.3 セキュア Read/Write コマンドフォーマット

Read, Write, Read v2, Write v2 は、暗号化されたセキュアフレーム内部に同一形式の内部ペイロードを格納します。差分は暗号方式のみです。

表 7.11: セキュア Read/Write 4 コマンドの差分

コマンド	暗号方式	ペイロード構造
Read	DES	Read 系共通 (表 7.12 および表 7.13)
Read v2	AES	Read 系共通 (表 7.12 および表 7.13)
Write	DES	Write 系共通 (表 7.14 および表 7.15)
Write v2	AES	Write 系共通 (表 7.14 および表 7.15)

表 7.12: Read 系コマンド (Read/Read v2) 内部ペイロードのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	ブロック数 (n)	0x01 ≤ n ≤ 製品の最大同時読み出し可能ブロック数
0x01	2n〜3n	ブロックリスト	ブロックリストエレメントの 2 バイト形式または 3 バイト形式

表 7.13: Read 系レスポンス (Read/Read v2) 内部ペイロードのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	ステータスフラグ 1	0x00: 正常
0x01	0x01	ステータスフラグ 2	0x00: 正常
0x02	0x01	ブロック数 (n)	ステータスフラグ 1 が 0x00 の場合のみ返送
0x03	16n	ブロックデータ	ステータスフラグ 1 が 0x00 の場合のみ返送

表 7.14: Write 系コマンド (Write/Write v2) 内部ペイロードのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	ブロック数 (n)	0x01 ≤ n ≤ 製品の最大同時書き込み可能ブロック数
0x01	2n〜3n	ブロックリスト	ブロックリストエレメントの 2 バイト形式または 3 バイト形式
可変	16n	ブロックデータ	16 バイト ×n

表 7.15: Write 系レスポンス (Write/Write v2) 内部ペイロードのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	ステータスフラグ 1	0x00: 正常
0x01	0x01	ステータスフラグ 2	0x00: 正常

7.3.1 DES 鍵変更パッケージの生成と使用方法

Write コマンドで DES 鍵変更を行う場合、ブロックリストエレメントのアクセスモードを 0b100(鍵変更) に設定し、ブロックデータには 16 バイトの鍵変更パッケージを格納します。

表 7.16: 鍵変更パッケージ (16 バイト) のデータ構造

オフセット	長さ	項目	値または備考
0x00	0x08	Parameter1	新鍵バージョン情報を含む暗号ブロック
0x08	0x08	Parameter2	新鍵本体を含む暗号ブロック

親鍵を K_{parent}、旧鍵を K_{old}、新鍵を K_{new}、新鍵バージョンを v とします。まず、8 バイトのバージョンブロック V を

$$V = 00\ 00\ 00\ 00\ 00\ 00\ v_{\text{LE},0}\ v_{\text{LE},1}$$

で作成し、次を計算します。

$$P_1 = \text{DES}_{K_{\text{parent}}}\Big(\text{DES}_{K_{\text{old}}}\big(\text{DES}_{K_{\text{new}}}(V)\big)\Big)$$

$$P_2 = \text{DES}_{K_{\text{parent}}}\Big(\text{DES}_{K_{\text{old}}}(K_{\text{new}})\Big)$$

$$\text{KeyChangePackage} = P_1 \parallel P_2$$

使用時は、鍵変更対象ごとに次を 1 組として Write コマンドに渡します。

- ブロックリストエレメント: アクセスモード=0b100、ブロック番号・鍵バージョン=v、サービスコードリスト順番=対象サービス
- ブロックデータ (16 バイト): 上記 KeyChangePackage

複数鍵を同時に変更する場合は、ブロックリストとブロックデータの順序を一致させて並べます。失敗時はステータスフラグ 2 として、例えば 0xAA(鍵変更失敗) や 0xAB(パッケージ MAC 異常) が返ることがあります。

7.3.2 DES セキュアメッセージング

DES セッションでは、暗号化前ペイロードを

$$P = \mathrm{TN}(2, \mathrm{LE}) \parallel \mathrm{TID}(6) \parallel \mathrm{CommandPayload}$$

とし、任意の方式で 8 バイト境界へパディングした P' に MAC を付与して DES-CBC(IV=0) で暗号化します。

MAC は次で計算します。

$$M_0 = [\mathrm{Len},\ \mathrm{Code},\ 0,\ 0,\ 0,\ 0,\ 0,\ 0]$$

$$M_i = \mathrm{DES}_{B_i}(M_{i-1})$$

ここで B_i は P' の 8 バイトブロック列、$\mathrm{Len} = 2 + |P'| + 8$ です。最終値 M_n を MAC(8 バイト) として付与します。

応答は復号後に MAC 検証し、末尾 8 バイトの MAC を除去した後にパディングを除去します。さらに TID 一致と TN 単調増加を検証します。

7.3.3 AES-128 セキュアメッセージング

AES セッションでは、送信データを

$$\mathrm{TN}(2, \mathrm{LE}) \parallel \mathrm{EncPayload} \parallel \mathrm{EncMAC}(8)$$

とします。Payload そのものは OFB で暗号化し、MAC も同じ OFB ストリームの後段で暗号化します。

初期ベクトル IV(16 バイト) は次で構成します。

$$IV[0] = 0x01,\ IV[1] = \mathrm{FrameLen},\ IV[2] = \mathrm{Code},$$

$$IV[3..4] = \mathrm{TN},\ IV[5..10] = \mathrm{TID},$$

$$IV[11..13] = \mathrm{C}_{3\mathrm{C}}[1..3],\ IV[14..15] = 0$$

MAC 平文は AES-CMAC で計算し、先頭 8 バイトを使用します。

$$B_0[0] = 0x19,\ B_0[1..13] = IV[1..13],\ B_0[14..15] = |Payload|_{\mathrm{BE16}}$$

$$MAC = \mathrm{CMAC}_{K_{\mathrm{mac}}}(B_0 \parallel Payload)[0..7]$$

OFB 暗号化時、Payload 長が 16 の倍数でない場合は次の 16 バイト境界までキーストリームを消費してから MAC を暗号化します。

7.4 DES 発行系セキュアコマンドフォーマット

Register Issue ID, Register Area, Register Service, Change System Block は DES セキュアコマンドとして送受信されます。暗号化前の内部ペイロード構造を以下に示します。

表 7.17: Register Issue ID コマンド内部ペイロードのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x08	発行 ID	-
0x08	0x08	発行パラメータ	-
0x10	可変	パッケージ	DES で生成された発行パッケージ

表 7.18: Register Area コマンド内部ペイロードのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x02	エリアコード	リトルエンディアン
0x02	可変	パッケージ	DES で生成された発行パッケージ

表 7.19: Register Service コマンド内部ペイロードのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x02	サービスコード	リトルエンディアン
0x02	可変	パッケージ	DES で生成された発行パッケージ

表 7.20: Change System Block コマンド内部ペイロードのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x00	なし	ペイロードなし

レスポンス内部ペイロードのデータ構造を以下に示します。

表 7.21: Register Issue ID レスポンス内部ペイロードのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	ステータスフラグ 1	0x00: 正常
0x01	0x01	ステータスフラグ 2	0x00: 正常
0x02	0x02	残りブロック数	ステータスフラグ 1 が 0x00 の場合のみ返送。リトルエンディアン

表 7.22: Register Area レスポンス内部ペイロードのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	ステータスフラグ 1	0x00: 正常
0x01	0x01	ステータスフラグ 2	0x00: 正常

表 7.23: Register Service レスポンス内部ペイロードのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	ステータスフラグ 1	0x00: 正常
0x01	0x01	ステータスフラグ 2	0x00: 正常
0x02	0x02	残りブロック数	ステータスフラグ 1 が 0x00 の場合のみ返送。リトルエンディアン

表 7.24: Change System Block レスポンス内部ペイロードのデータ構造

オフセット	長さ	項目	値または備考
0x00	0x01	ステータスフラグ 1	0x00: 正常
0x01	0x01	ステータスフラグ 2	0x00: 正常

発行パッケージ平文 (暗号化前) は 16 バイトで、各コマンドのフォーマットは以下のとおりです。

表 7.25: Register Issue ID 用発行パッケージ平文 (16 バイト) のデータ構造

オフセット	長さ	項目	値または備考
0x00	0x02	システムコード	ビッグエンディアン
0x02	0x02	Area0 鍵バージョン	リトルエンディアン
0x04	0x08	Area0 鍵	-
0x0C	0x04	予約領域	0x00000000

表 7.26: Register Area 用発行パッケージ平文 (16 バイト) のデータ構造

オフセット	長さ	項目	値または備考
0x00	0x02	サービスコード開始	リトルエンディアン
0x02	0x02	サービスコード終端	リトルエンディアン
0x04	0x02	サイズ	リトルエンディアン
0x06	0x02	鍵バージョン	リトルエンディアン
0x08	0x08	エリア鍵	-

表 7.27: Register Service 用発行パッケージ平文 (16 バイト) のデータ構造

オフセット	長さ	項目	値または備考
0x00	0x02	サービスコード	リトルエンディアン
0x02	0x02	予約領域	0x0000

次ページに続く

表 7.27: Register Service 用発行パッケージ平文 (16 バイト) のデータ構造 (続き)

オフセット	長さ	項目	値または備考
0x04	0x02	サイズ	リトルエンディアン
0x06	0x02	鍵バージョン	リトルエンディアン
0x08	0x08	サービス鍵	-

7.4.1 発行パッケージ作成アルゴリズム

パッケージ鍵を K_{pkg}、平文パッケージを P とします ($|P|$ は 8 の倍数、かつ 0 でないことが条件です)。

- MAC 生成鍵を
$$K_{\mathrm{macpkg}} = K_{\mathrm{pkg}} \oplus 0xFF \ldots FF$$
で作成します。
- $$C = \mathrm{DES\text{-}CBC}_{IV=0,K_{\mathrm{macpkg}}}(P)$$
を計算し、最終 8 バイトを MAC_{pkg} とします。
- $$Package = \mathrm{DES\text{-}CBC}_{IV=0,K_{\mathrm{pkg}}}(P \parallel MAC_{\mathrm{pkg}})$$
を計算し、この暗号文をコマンドの「パッケージ」フィールドに格納します。

参考文献

[1] C. Herrmann et al. *Proxmark3 – Iceman repo*. https://github.com/RfidResearchGroup/proxmark3.

[2] kormax. *GitHub - kormax/felica-tool: Application for analyzing characteristics of FeliCa cards*. 2025. URL: https://github.com/kormax/felica-tool (visited on 05/28/2025).

[3] ソニー株式会社. *FeliCa* **カード ユーザーズマニュアル 抜粋版**. Version 2.31. 2026. URL: https://www.sony.co.jp/Products/felica/business/tech-support/data/card_usersmanual_2.31j.pdf (visited on 07/23/2026).

[4] 一般財団法人日本規格協会. *JIS X 6319-4:2016* **ＩＣカード実装仕様―第４部：高速処理用近接型ＩＣカード**. 2016.