AVR230 : DES(ﾃﾞｰﾀ暗号化規格)ﾌﾞｰﾄﾛｰﾀﾞ

要点

- ■ ﾌﾞｰﾄﾛｰﾀﾞ能力を持つ全てのAVRﾏｲｸﾛ ｺﾝﾄﾛｰﾗに適合
- ■ ﾌﾞｰﾄﾛｰﾀﾞ能力を持つどのAVRにもｺﾝﾊﾟｲﾙされたｿﾌﾄｳｪｱまたは高い慎重性を有するﾃﾞｰﾀの安全な転送が可能
- ■ 簡単な使用と形態設定可能な応用例を包含
  - ・ 2進ﾌｧｲﾙとﾃﾞｰﾀの暗号化
  - ・ 目的対象ﾌﾞｰﾄﾛｰﾀﾞ作成
  - ・ 目的対象への暗号化ﾌｧｲﾙのﾀﾞｳﾝﾛｰﾄﾞ
- ■ ﾃﾞｰﾀ暗号化規格(DES)に従った暗号化算法
- ■ 安全性を増すための3重ﾃﾞｰﾀ暗号化規格(3DES)
- ■ 全AVRﾃﾞﾊﾞｲｽでの2Kﾊﾞｲﾄ内のDESﾌﾞｰﾄﾛｰﾀﾞ
- ■ ATmega128を除く全AVRﾃﾞﾊﾞｲｽでの2Kﾊﾞｲﾄ内の3DESﾌﾞｰﾄﾛｰﾀﾞ
- ■ ﾃﾞｰﾀの転送、解読、ﾌﾗｯｼｭ ﾒﾓﾘ書き込みを含む16Kﾊﾞｲﾄ応用の代表的な更新時間
  - ・ DES、115200bps、16MHz目的対象周波数：20秒
  - ・ 3DES、115200bps、16MHz目的対象周波数：50秒

序説

この応用記述はﾌﾞｰﾄﾛｰﾀﾞ能力を持つAVRﾏｲｸﾛ ｺﾝﾄﾛｰﾗでﾌｧｰﾑｳｪｱがどう安全に更新し得るかを記述します。その方法はﾌｧｰﾑｳｪｱを暗号化するのにﾃﾞｰﾀ暗号化規格(DES)の使用を含みます。この応用記述は3重ﾃﾞｰﾀ暗号化規格(3DES)も支援します。

ﾏｲｸﾛ ｺﾝﾄﾛｰﾗを含む電気設計は可搬型音楽再生機、ﾍｱﾄﾞﾗｲﾔ、ﾐｼﾝなどでﾌｧｰﾑｳｪｱを装備されることが常に必要です。多くの電気設計が急速に発達するため、既に出荷または販売されてしまった製品を更新することができる必要が増大しています。特に製品が既に最終顧客へ届いてしまっている場合、ﾊｰﾄﾞｳｪｱに変更を行うのは難しくなると言ってよいでしょう。しかしAVR系統のようなﾌﾗｯｼｭ ﾏｲｸﾛ ｺﾝﾄﾛｰﾗに基づく製品でﾌｧｰﾑ ｳｪｱは容易に更新することができます。

多くのAVRﾏｲｸﾛ ｺﾝﾄﾛｰﾗは要求あり次第、ﾌｧｰﾑｳｪｱ更新を受け取ってﾌﾗｯｼｭ ﾒﾓﾘを書き換えることができるﾌﾞｰﾄﾛｰﾀﾞを作ることが可能なように形態設定されます。ﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘ空間はﾌﾞｰﾄﾛｰﾀﾞ領域(BLS)と応用領域の2つの領域に分けられます。両領域はﾌﾞｰﾄﾛｰﾀﾞのｺｰﾄﾞがBLSで保護され得ると同時に未だ応用領域でｺｰﾄﾞを更新できるような読み書き保護用の専用施錠ﾋﾞｯﾄを持ちます。従って、BLSの更新方法は外側のｱｸｾｽに備えて簡単に保護することができます。

ﾌｧｰﾑｳｪｱに問題が残っており、それは代表的に、ﾌﾗｯｼｭ ﾒﾓﾘに書かれて施錠ﾋﾞｯﾄが設定される前に保護されないことです。これはﾌｧｰﾑｳｪｱが現場で更新されることが必要な場合にそれが書き込み台または製造業者の建物を出る瞬間から不正ｱｸｾｽに関して開いていることを意味します。

この応用記述はﾌﾗｯｼｭ ﾒﾓﾘとEEPROMへ転送されるべきﾃﾞｰﾀが暗号法を用いることによって何時でも如何にして保護され得るかを示します。この考えは書き込み台を去る前にﾃﾞｰﾀを暗号化して目的対象AVRにﾀﾞｳﾝﾛｰﾄﾞされた後でだけそれを解読することです。この手順はﾌｧｰﾑｳｪｱの不正な複製を防げませんが、その複製(それと勿論原型)は正しい解読鍵なしでは実質的に無効です。解読鍵は書き込み環境の外側、目的対象AVRの内側の1つの位置にだけ格納されます。施錠ﾋﾞｯﾄが設定されている場合、その鍵は目的対象AVRから読むことができません。また、この鍵は暗号化されたﾃﾞｰﾀから再生成することもできません。ﾃﾞｰﾀへのｱｸｾｽを得る唯一の方法は正しい鍵を使用することによるだけです。

8-bit AVR®
マイクロ コントローラ

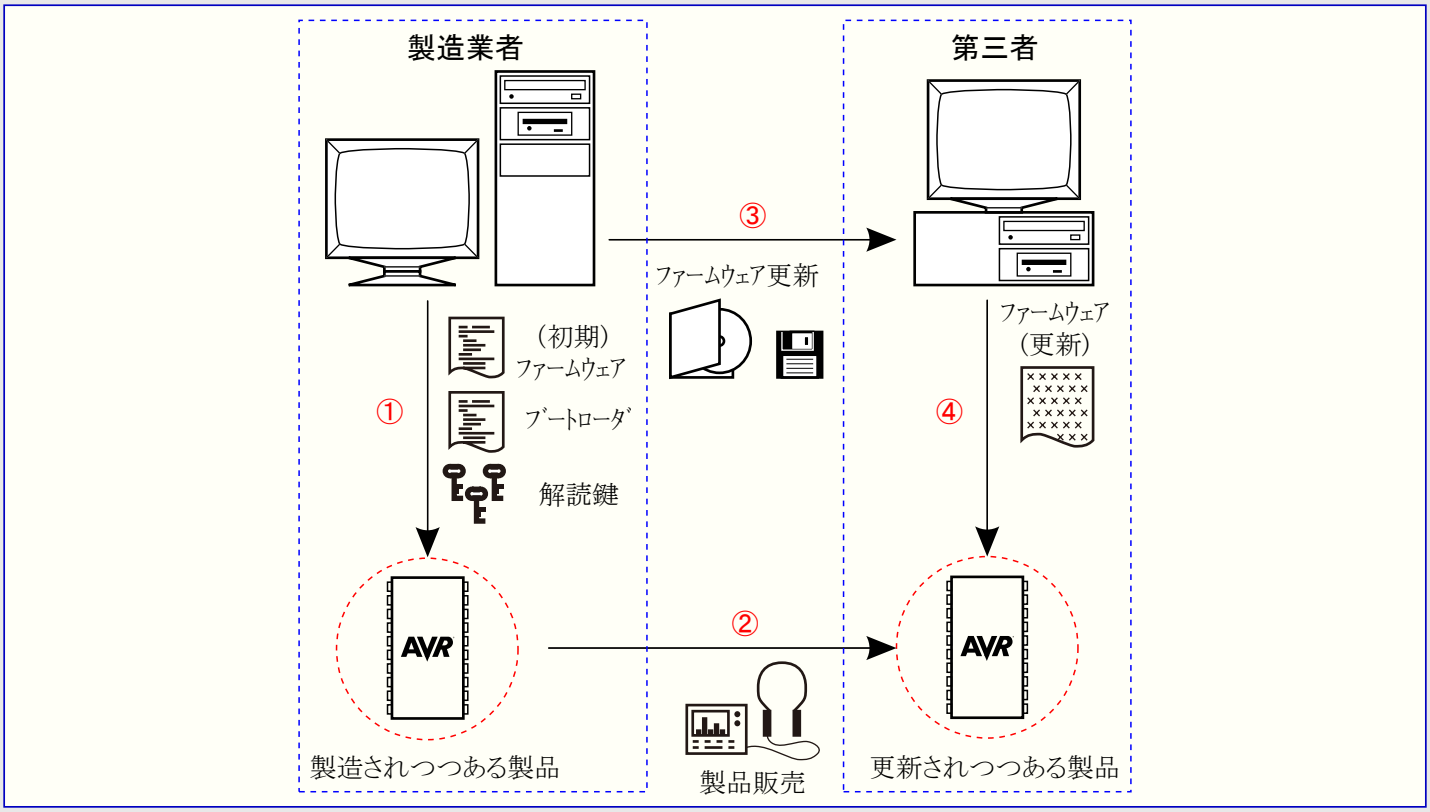
応用記述

本書は一般の方々の便宜のため有志により作成されたもので、ATMEL社とは無関係であることを御承知ください。しおりのはじめにでの内容にご注意ください。

Rev. 2541D-04/05, 2541DJ2-01/14

図1.は製品がどのように最初に製造され、初期ファームウェアを設定され、販売されて、そして後で新版のファームウェアで更新される例を示します。

図1. AVRに基づく設計の製造と更新の手順の例



①. 製造中、マイクロ コントローラは最初にブートローダ、解読鍵、応用ファームウェアを装備されます。ブートローダは実際の応用とプログラミングを受け取ってそれをフラッシュ メモリ内にプログラミングする準備をし、同時に鍵はやって来るデータの解読に必要です。施錠ビットはAVR内側のファームウェアを保護するように設定されます。

②. 製品はその後に代理店へ出荷されるか、または最終的な顧客に販売されます。施錠ビット設定はAVR内側で保護されたファームウェアを維持し続けます。

③. ファームウェアの新規公開が完了され、既に配給されてしまった製品の更新の必要があります。ファームウェアは従って暗号化されて代理店に出荷されます。暗号化されたファームウェアは解読鍵なしで用を足さず、従ってソフトウェアの局所的な複製(例えば、代理店のハードドライブ)は安全性を危険に晒しません。

④. 代理店は在庫の全ての構成単位と顧客によって返された(例えば、修理中の)それらを更新します。暗号化されたファームウェアはAVRにダウンロードされ、マイクロ コントローラの内側で一度だけ解読されます。施錠ビット設定はAVRの内側で更新されたファームウェアの保護を維持し続けます。

理屈

暗号は秘密情報維持の芸術または科学で、暗号法の隠蔽または暗号鍵の保護に基づきます。使用する方法の安全性に基づくだけの算法は歴史的興味が主で、実世界の要求には合いません。最新の算法は暗号化と解読の制御に鍵を用います。一致する鍵なしでは掻き混ぜられたメッセージやデータを平文に整列することができません。

暗号化

暗号化はそれの内容が部外者から隠されるようにメッセージやデータを符号化する方法です。その元の形式に於いて平文のメッセージやデータはマイクロ コントローラのファームウェアのように作者や配給者が秘密維持を望む情報を含むかもしれません。例えば、マイクロ コントローラが現場で更新される時に、不正な複製の試みや逆行分析に対してファームウェアを保護することが難しくなるかもしれません。ファームウェアの暗号化はそれが解読されるまでそれを使用できなくします。

解読

解読は元のメッセージやデータを取り戻す方法で、代表的に正しい鍵を知ることなく実行することができません。鍵はデバイスが暗号化したデータを受け取ってそれを解読し、フラッシュ メモリまたはEEPROMの選択した部分を書き換えることができるように、マイクロ コントローラのブートローダ内に格納することができます。解読鍵は暗号化したデータから取り戻すことができず、施錠ビットがプログラム(0)されてしまっている場合、AVRマイクロ コントローラから読むことができません。

暗号鍵算法

暗号鍵に基づく算法は対称と非対称の2つの種類に分けられます。対称算法は暗号化と解読に対して同じ鍵を用い、一方非対称算法は違う鍵を用います。最も研究され、多分最も大きく広がった対称算法はDESです。

ﾃﾞｰﾀ暗号化規格 – DES

ﾃﾞｰﾀ暗号化規格(DES:Data Encryption Standard)は元来1970代に開発され、後に米国国立標準技術研究所(NIST:US National institute of Standards)によって標準規格に転化されました。DESは56ﾋﾞｯﾄ鍵を用いる対称暗号算法です。この算法は実際問題として非常に強力と証明され、他の多くのものよりも長く残っています。

DES算法は56ﾋﾞｯﾄ暗号鍵を使用し、これは可能な鍵の組み合わせ数が2^{56}=72,057,594,037,927,936=7,206×10^{16}であることを意味します。

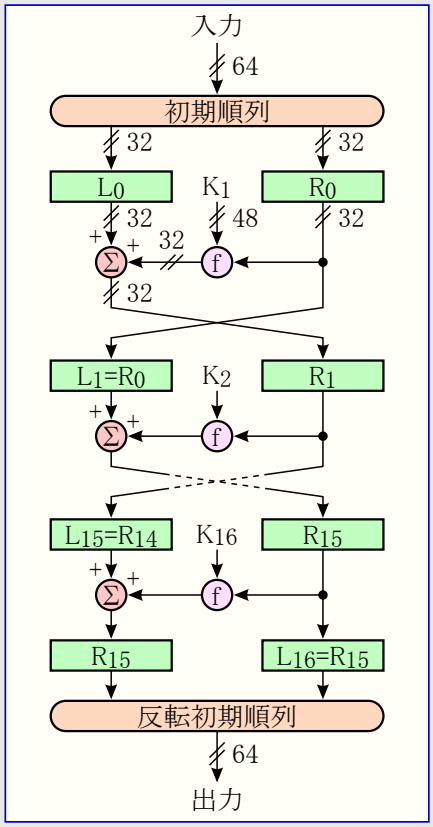
DESは64ﾋﾞｯﾄのﾃﾞｰﾀ塊で操作する塊暗号です。各入力塊は**図2.**で図解されるように処理されます。

図2.は単一ﾃﾞｰﾀ塊がどう暗号化されつつあるかを図解します。最初に、入力ﾋﾞｯﾄの順が順列関数に従って変更されます。下位32ﾋﾞｯﾄ(R_0)はその後に上位32ﾋﾞｯﾄ(L_0)から分離されて処理されます。暗号鍵の異なる補助組(Kn)を使用する各段で、16処理段階があります(**図2.**では1,2,16段階だけが図解されます)。最後にﾋﾞｯﾄ順が初期順列関数に対して逆に変更されます。

解読法は暗号化法と同じで、鍵の補助組(Kn)の順が逆にされるだけです。

DES算法はDES規格でもっと詳細に記述されます(「**参照**」章をご覧ください)。

図2. DES算法に従った暗号化の流れ



3重ﾃﾞｰﾀ暗号化規格 – 3DES

3重ﾃﾞｰﾀ暗号化規格(3DES)は3回のDES使用に基き、従って鍵長は56から168ﾋﾞｯﾄに増加します。3DESはDESよりも非常に強力ですが、いくつかのより新しい算法と比べると、実時間応用に関してむしろ遅いです。この応用記述に関してはﾀｲﾐﾝｸﾞが重要でなく、3DESが良いと考えられるかもしれません。

3DES法は3つの56ﾋﾞｯﾄ暗号鍵を使用します。従って組み合わせ数は2^{168}=3.741×10^{50}に増加します。

3DESはANSI® x9.52で定義されるように、3回のDES使用に基づきます。暗号化の流れは右で図解されます。

暗号化中、入力は初めに最初の鍵で暗号化され、その後に第2の鍵で解読され、最後に第3の鍵で暗号化されます。解読中、**図4.**で図解されるように、鍵の順と符号化/解読塊の順は逆にされます。

規格で以下の鍵操作が定義されています。

図3. 3DES法に従った暗号化の流れ

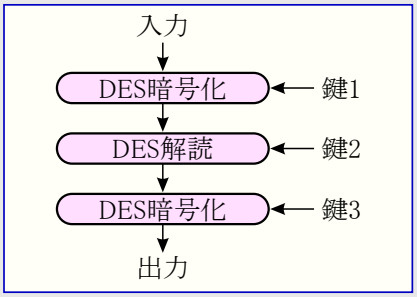


図4. 3DES法に従った解読の流れ

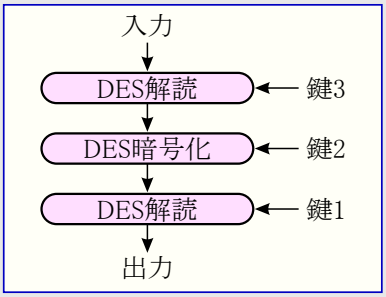


表1. 3DES鍵任意選択

任意選択	独立鍵数	連携する鍵
鍵任意選択1	3 (鍵1、鍵2、鍵3)	(なし)
鍵任意選択2	2 (鍵1、鍵2)	鍵3=鍵1
鍵任意選択3	1 (鍵1)	鍵1=鍵2=鍵3

3DESの完全な暗号長を利用する最初の鍵任意選択1だけが注目されるべきです。3つ目の鍵任意選択3は本質的に単一DESと同じです。

暗号塊連鎖 – CBC

DESと3DESは塊暗号で、この算法は固定量のﾃﾞｰﾀ塊で操作することを意味します。64ﾋﾞｯﾄの塊でﾃﾞｰﾀを暗号化するのに56ﾋﾞｯﾄ鍵が用いられます。既知の入力塊と一定(とは言え未知)の暗号鍵に関して、その出力塊は常に同じです。この情報は暗号ｼｽﾃﾑの攻撃を欲する誰かにとって有用な提供かもしれません。

同じ平文の塊を違う暗号文の塊に暗号化させるのに一般的に使用される方法があります。そのような方法は暗号塊連鎖(CBC:Cipher Block Chaining)と呼ばれます。

CBCは先行塊が全ての後行塊に影響を及ぼすような暗号塊を接続する方法です。これは最初に平文の塊と直前の暗号文の塊でXOR操作を実行し、その後に結果を暗号化することによって成し遂げられます。これは1つの暗号文ﾋﾞｯﾄが依存するところの平文ﾋﾞｯﾄ数を増します。

どれくらい安全か?

単に可能な全ての鍵を次々に調べることによって、鍵に基づくどの暗号法も破ることが理論的に可能なことを理解することが重要です。別の方法で破ることができない理想的な系を考慮すると、ﾃﾞｰﾀ保護のﾚﾍﾞﾙは暗号鍵の長さに指数関数的に比例します。

不正な解読の試みに関する一般的な方法は総当りの使用、換言すると可能な全ての鍵を通して進み、理解できる出力が現れるまで一度に1つずつそれらを試します。人間の操作者は手動でこのような手順を取れませんが、特別なｿﾌﾄｳｪｱまたはﾊｰﾄﾞｳｪｱを用いて自動化することができます。これは理解できないものから理解できる出力を区別する方法を作ることができる人間の操作者が未だ必要です。総当り攻撃では、鍵の長さで指数関数的に増加する鍵を破るにに計算力が必要とされます。平均計算力が時間に渡って指数関数的に増加するため、どの鍵も破られるまでは時間の問題なだけです。

表2.は鍵長対暗号強度の結果を説明します。これらの値が指針の意味だけで、予測の多くに於いて大きな変動があることに注意してください。この情報は「参照」章で言及される供給元から集められています。

表2. 暗号破り予測時間 対 暗号鍵長 (「参照」をご覧ください。)

鍵長	例	個人、小企業	組織犯、大企業、政府	
		標準的なPC	接続されたPC	特殊なﾊｰﾄﾞｳｪｱ
32ﾋﾞｯﾄ		数分～数時間		
40ﾋﾞｯﾄ		数日～数週間	数分～数時間	
56ﾋﾞｯﾄ	DES	実質的に破られない	数週間～数ヶ月	数日～数週間
64ﾋﾞｯﾄ			非常に長い期間	短期間
80ﾋﾞｯﾄ			実質的に破られない	長期間
128ﾋﾞｯﾄ				実質的に破られない
168ﾋﾞｯﾄ	3DES			

別の攻撃方法は暗号化された情報がないﾊｰﾄﾞｳｪｱを目標とするものです。この応用記述ではATMELのﾌﾗｯｼｭ ﾒﾓﾘに基づくAVR構造によってﾊｰﾄﾞｳｪｱが保護されます。解読鍵を含むDES/3DESﾌﾞｰﾄﾛｰﾀﾞはﾌﾗｯｼｭ ﾒﾓﾘ内に格納され、そのﾒﾓﾘはその後に施錠ﾋﾞｯﾄを用いて保護されます。ﾌｧｰﾑｳｪｱは暗号化されたﾊﾟｯﾁを用いて現場で今や更新することができますが、解読鍵はAVRから取り戻すことができません。

施錠ﾋﾞｯﾄが設定されると、全ての書き込みと外部読み込みｱｸｾｽが否定されます。ﾒﾓﾘへのｱｸｾｽを得る唯一の方法はﾃﾞﾊﾞｲｽを消去することです。それを行うと、施錠ﾋﾞｯﾄが消去される前にﾌﾞｰﾄﾛｰﾀﾞと解読鍵が破壊されます。

ｼｽﾃﾑの安全性改善方法に於ける推奨

この応用記述は外側のｱｸｾｽから設計を保護する時に使用することができる技法を提供します。例え完全に保護することができる設計がないとは言え、それは安全性を破るために必要とされる労力が可能な限り高くなるように構成することができます。基本工学的な技能の人が複製することができる安全でない設計と、少数で非常に熟練した侵入者だけが破ることができる設計の間には重大な違いがあります。最初の例では、設計が容易に複製され、そして製造業者の知的所有権に違反して逆行分析さえもされ、そしてその設計に関する潜在的な市場をも危うくします。2つ目の例では、設計を破るために必要とされる労力が大きいので、殆どの侵入者は単に彼ら自身の製品を開発することに集中します。

安全なｼｽﾃﾑ構築法には、可能な限り破るのを難しくするように設計すべきであると言う、一般的な1つの規則だけがあります。破ろうとする間に、保護を欺くのに使用することができるどの手法も試みられるでしょう。外来の状況さえ考慮されるべきです。考慮されなければならない少しの例が以下で与えられます。

ﾌｧｰﾑｳｪｱ更新中に電力が取り去られたなら、何が起きますか?。電力が戻って回復された時にﾏｲｸﾛ ｺﾝﾄﾛｰﾗの状態はどうですか?。施錠ﾋﾞｯﾄとﾘｾｯﾄ ﾍﾞｸﾀは何時も正しく設定されていますか?。

平文ﾃﾞｰﾀのように見える何かを作ることができる何れかの仮定がありますか?。DESに関して総当り法によって破られるには探す様式があるに違いありません。攻撃者は単に全ての鍵の組み合わせを試み、理解できるﾌｧｰﾑｳｪｱを探してその出力を監視する筈がありません。攻撃ｿﾌﾄｳｪｱはﾌﾟﾛｸﾞﾗﾑ ﾒﾓﾘの開始での割り込みﾍﾞｸﾀ、0または1で穴埋めされたﾒﾓﾘ領域、以下同様のような既知の様式を検索するように形態設定されなければなりません。

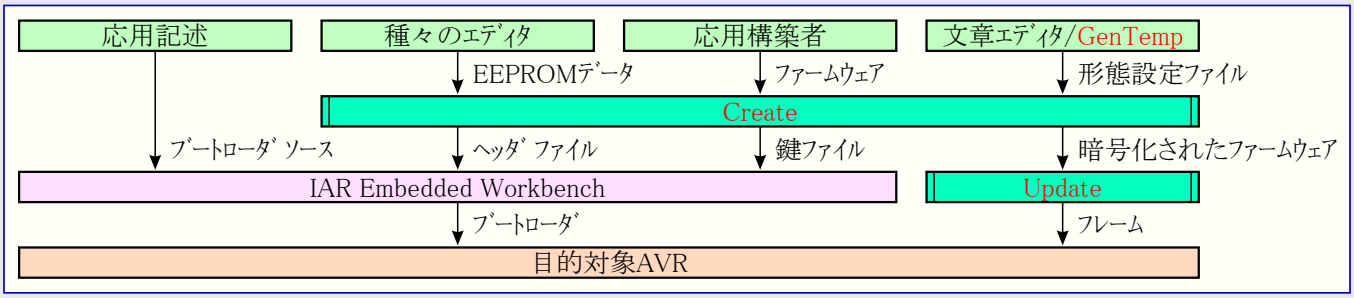
解読処理から得ることができる何れかの還元がありますか?。このようなどの還元も総当り攻撃基盤構築に於いて攻撃者を手助けし得ます。例えば、ﾌﾞｰﾄﾛｰﾀﾞ内側の解読算法が各塊処理に対して良/否形式の合図を与える場合、この信号は攻撃様式への還元として容易に道筋を定めることができます。

符号化したﾌﾚｰﾑは別の順で送られるべきですか?。ﾌﾞｰﾄﾛｰﾀﾞに送られる最初のﾌﾚｰﾑが常に暗号化したﾌｧｲﾙの最初の塊を含む場合、攻撃者はこれからいくつかの仮定を作ることができます。例えば、最初のﾌﾚｰﾑはｱﾄﾞﾚｽ0から始まるﾌﾟﾛｸﾞﾗﾑ ﾃﾞｰﾀの割付でそれは割り込みﾍﾞｸﾀ表を含むと仮定することができます。この情報は鍵検索を細かく区別する攻撃者を助けます。ｼｽﾃﾑの安全を増すため、乱順でﾌﾚｰﾑを送ってください(どうせ、解読されたﾌﾚｰﾑはそれらの正しいｱﾄﾞﾚｽに割付されます)。

実装と使い方

本章は応用の形態設定と使用方法を記述します。手順は**図5.**で図解されます。

図5. ﾌﾟﾛｼﾞｪｸﾄの流れの概要



主な段階は次の通りです。

1. 目的対象AVR用の応用を作成してください。必要ならば、独立したﾌｧｲﾙでEEPROM設計(配置)を作成してください。
2. ﾌﾟﾛｼﾞｪｸﾄ依存情報で形態設定ﾌｧｲﾙを作成してください。GenTempと呼ばれる応用がﾌｧｲﾙ ﾌﾚｰﾑ作成に使用できます。
3. Createと呼ばれる応用を走らせてください。これはﾍｯﾀﾞ ﾌｧｲﾙ、鍵ﾌｧｲﾙ、暗号化されたﾌｧｲﾙを作成します。
4. IAR EWを使用して、目的対象AVR用のﾌﾞｰﾄﾛｰﾀﾞを形態設定して構築してください。
5. 目的対象AVRにﾌﾞｰﾄﾛｰﾀﾞをﾀﾞｳﾝﾛｰﾄﾞ(ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ)して施錠ﾋﾞｯﾄとﾋｭｰｽﾞ ﾋﾞｯﾄを設定してください。
6. 今や暗号化したﾌｧｰﾑｳｪｱは何時でもAVRにﾀﾞｳﾝﾛｰﾄﾞ(書くことが)できます。

この手続きは以下でもっと詳細に検討されます。

形態設定ﾌｧｲﾙ

形態設定ﾌｧｲﾙはﾌﾟﾛｼﾞｪｸﾄを形態設定するのに使用されるﾊﾟﾗﾒｰﾀの一覧を含みます。ﾊﾟﾗﾒｰﾀは**表3.**で記述されます。

表3. 形態設定ﾌｧｲﾙ任意選択の要約

ﾊﾟﾗﾒｰﾀ	説明	既定	必須
PAGE_SIZE	AVRﾌﾗｯｼｭ ﾍﾟｰｼﾞの10進ﾊﾞｲﾄでの大きさ。この値はﾃﾞﾊﾞｲｽ依存です、ﾃﾞｰﾀｼｰﾄをご覧ください。	N/A	はい
KEY1	16進での最初の暗号鍵。奇数ﾊﾟﾘﾃｨ ﾋﾞｯﾄとして使用される全ての第8ﾋﾞｯﾄを持つ8つの乱数ﾊﾞｲﾄであるべきです。	なし: 暗号なし	いいえ、しかし強く推奨
KEY2	第2暗号鍵。最初の暗号鍵同様ですが、2鍵3DESに使用。省略されたなら、単一DES暗号が使用されます。	なし: 単一DES使用	いいえ、しかし推奨
KEY3	第3暗号鍵。最初の暗号鍵同様ですが、3鍵3DESにだけ使用。省略されたなら、単一または2鍵DES暗号が使用されます。	なし: 単一または2鍵DES	いいえ
INITIAL_VECTOR	暗号塊連鎖に使用。16進での8つの乱数ﾊﾞｲﾄであるべきです。	0000000000000000	いいえ、しかし強く推奨
SIGNATURE	ﾌﾚｰﾑ確認ﾃﾞｰﾀ。これはどの4ﾊﾞｲﾄにもできますが、乱数で選ばれた値が推奨されます。	00000000	はい
ENABLE_CRC	CRC検査許可。YesまたはNo。許可されたなら、全ての応用領域は上書きされ、応用はそれが開始を許される前にCRC検査を通らなければなりません。	なし	いいえ、しかし推奨
MEM_SIZE	目的対象AVR内の応用領域の大きさ(10進ﾊﾞｲﾄでの)。	なし	はい、CRC使用の場合

注: DES算法は56ﾋﾞｯﾄ鍵を使用しますが、形態設定ﾌｧｲﾙ内で鍵は8ﾊﾞｲﾄ(64ﾋﾞｯﾄ)幅で与えられます。これは全ての第8ﾋﾞｯﾄが奇数ﾊﾟﾘﾃｨ ﾋﾞｯﾄとして使用されるためです。

形態設定ファイルはどの有効なファイル名も与えることができます(この名前はプロジェクト ファイルを作成する応用へのパラメータとして後で与えられます)。下はATmega16用の試供形態設定ファイルです。

```
PAGE_SIZE       = 128
MEM_SIZE        = 14336
KEY1            = 0123456789ABCDEF
KEY2            = FEDCBA9876543210
KEY3            = 89ABCDEF01234567
INITIAL_VECTOR  = 0011223344556677
SIGNATURE       = 89ABCDEF
ENABLE_CRC      = YES
```

パラメータのいくつかは目的対象AVRの特定の知識なしに設定することができません。**表4.**はブートローダ機能を持つ存在するいくつかのAVRマイクロ コントローラの特性を要約します。この表に存在しないデバイスについてはデバイスのデータシートを参照してください。

表4. AVR特性要約

特性	ATmega								
	8	8515	8535	16	162	169	32	64	128
フラッシュ容量(バイト)	8K			16K			32K	64K	128K
フラッシュ ページ容量(バイト)	64			128			256		
フラッシュ ページ数	128						256		512
(最大)BLS容量(バイト)	2048						4096	8192	
BLSページ数	32			16			32		
MEM_SIZE(バイト)	6144			14336			28672	57344	122880
PAGE_SIZE(バイト)	64			128			256		

PC応用 - GenTemp

この応用記述は形態設定ファイル用の雛形を生成する小さなPC応用を含みます。この応用は乱数暗号鍵を生成し、ベクタを初期化し、(フラッシュ ページ容量のように)満たされるべき使用者用の他のパラメータをそのままにします。この手続きが真実の乱数暗号鍵を保証して人の予想を取り去ります。

この応用は次のように使用されます。

```
GenTemp FileName.Ext
```

ここでFileName.Extは作成される形態設定ファイルの名前です。ファイルが生成された後で選んだどの平文エディタを使用しても編集することができます。

PC応用 - Create

このPC応用はMicrosoft®のVisual C++ 6.0版を用いて作成されています。これは形態設定ファイルから情報を読んで、ブートローダ用の鍵とヘッダ ファイルを生成します。これはファームウェアの暗号化にも使用されます。代表的に、この応用は最低2回走行し、最初はブートローダ用の鍵とヘッダ ファイルを生成するため、2回目は新しいファームウェアが暗号化される時です。

注: プロジェクト ファイルを生成する時とファームウェアを暗号化する時に同じ暗号情報(形態設定ファイル)が使用されることが非常に重要です。さもなければ、ブートローダは正しい暗号鍵の組を持たずにデータを解読できないかもしれません。

暗号化されたファームウェアを解読するのに形態設定ファイル内の情報を用いることが可能なことにも注意すべきです。従って、形態設定ファイルは何時も安全を保たれなければならず、最初に使用された後で変更されるべきではありません。

命令行引数

この応用は以下の命令行引数を受け入れます。

表5. 命令行引数の要約

引数	説明
-c <ファイル名.ext>	形態設定ファイルへのパス
-d	設定なら、各フラッシュ ページの内容は書く前に削除されます。そうでなく、特に書かれなければ直前のデータが保たれます。
-e <ファイル名.ext>	EEPROMファイル(EEPROM内容になるデータ)へのパス
-f <ファイル名.ext>	フラッシュ ファイル(応用領域内容になるコード)へのパス
-h <ファイル名.ext>	出力ヘッダ ファイル名。このファイルは後にブートローダでインクルードされなければなりません。
-k <ファイル名.ext>	出力鍵ファイル名。このファイルは後にブートローダでインクルードされなければなりません。
-l [BLB12][BLB11][BLB02][BLB01]	設定する施錠ビット。これらの施錠ビットは全データが転送された後で、制御が更新された応用に渡される前に設定されます。
-n	不感知。暗号化されたファイルに不感知記録の乱数を追加します。不感知記録がブートローダによって無視されるため、この設定は出力ファイルの予測性だけで、応用に影響を及ぼしません。
-o <ファイル名.ext>	出力ファイル名。これは配布され得る暗号化したファイルで、更新が必要な時に目的対象へ送られます。

初回走行

最初の走行では代表的にブートローダ用の鍵とヘッダ ファイルが生成されるだけです。鍵とヘッダ ファイルの生成は命令行引数を用いて要求されます。

```
例: Create -c Config.txt -h BootLdr.h -k DESKeys.inc
```

鍵とヘッダ ファイルはブートローダ応用のプロジェクト フォルダに複写され、ブートローダ コード内にインクルードされなければなりません。

注: ブートローダ プロジェクト ファイル名が上で言及したファイル名、換言するとBootLdr.hとDESKeys.incを使用するように予めコンパイルされていることに注意してください。これらのファイル名は変更されないことが推奨されます。

後続走行

後続する走行ではファームウェアを符号化するのにこの応用が使用されます。暗号化に先立って、ソース ファイルはコンパイル、アセンブルされて1つのコード セグメント ファイルと/または1つのEEPROMセグメント ファイルにリンクされなければなりません。ファイルはIntelのHEX形式でなければなりません。

ファイル名はコマンド プロンプトで与えられ、形態設定ファイル内のデータに従って暗号化されたファイルが生成されます。

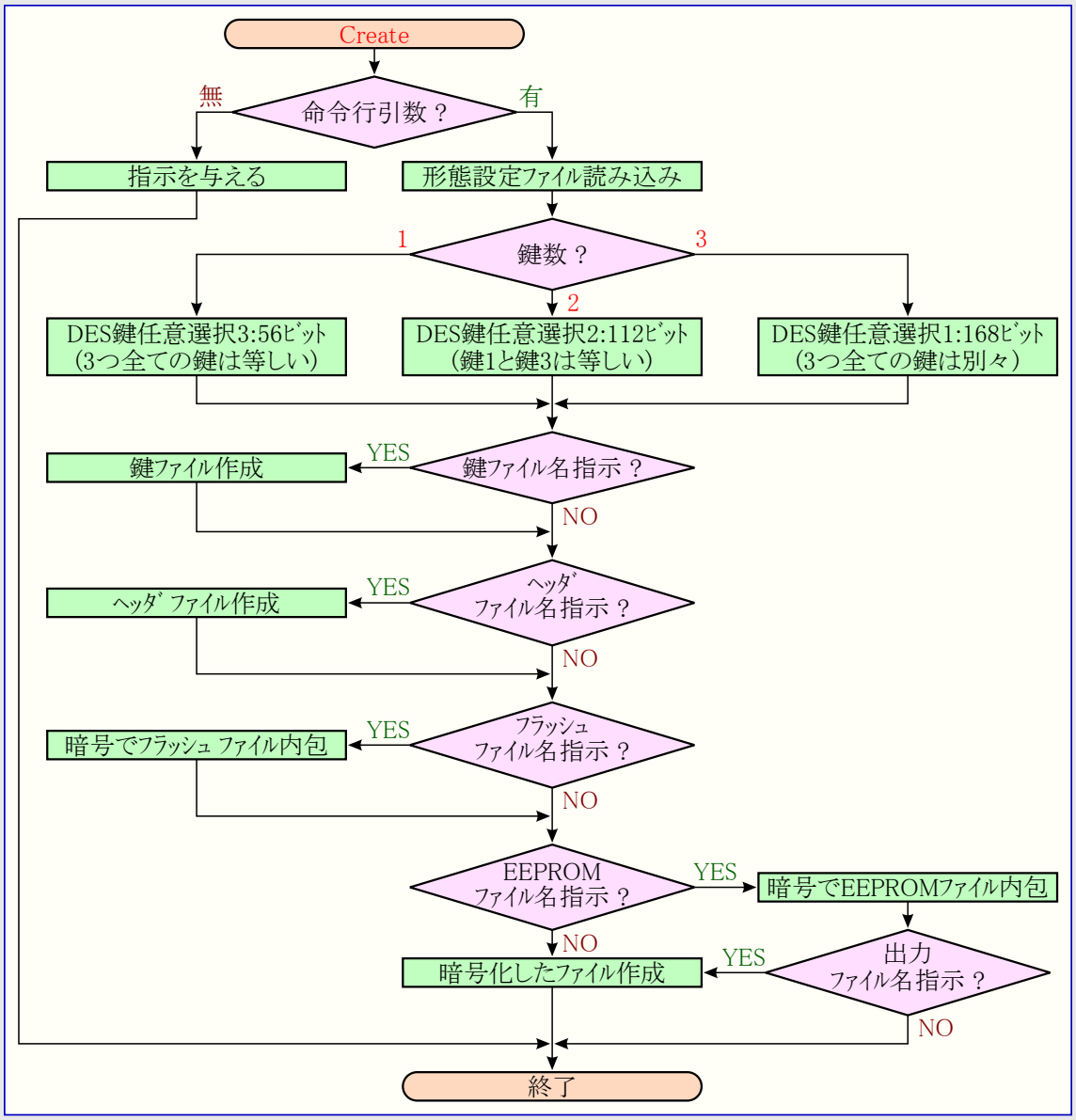
```
例: Create -c Config.txt -e EEPROM.hex -f Flash.hex -oUpdate.enc -l BLB11 BLB12
```

応用ソフトウェアとEEPROMのファイルは単一の暗号化したファイル内に結合されます。

ﾌﾟﾛｸﾞﾗﾑの流れ

ﾌﾟﾛｸﾞﾗﾑの流れが**図6.**で図解されます。

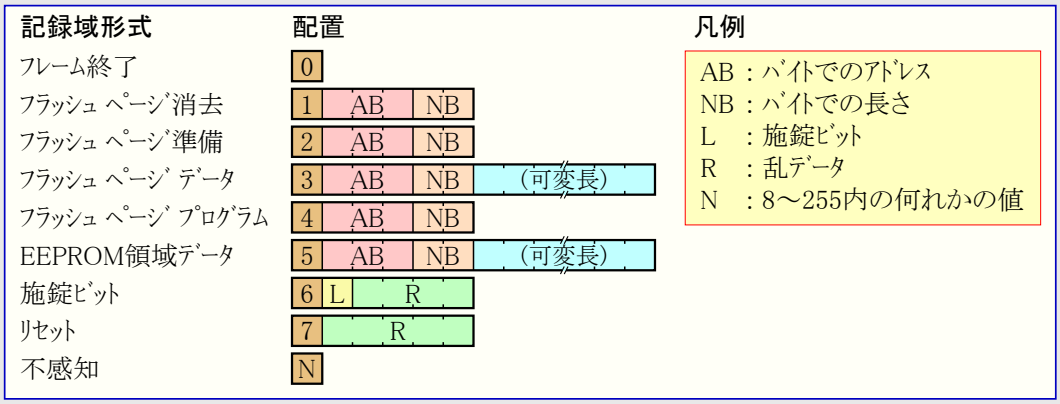
図6. PC応用のﾌﾟﾛｸﾞﾗﾑの流れ



暗号化されたﾌｧｲﾙ

ﾌﾗｯｼｭとEEPROMのﾌｧｲﾙは暗号化されて1つの目的対象ﾌｧｲﾙに格納されます。けれども、暗号化の前にﾃﾞｰﾀは記録域内に編成されます。図7.で図解されるように、7つの記録域形式があります。

図7. 記録域の形式

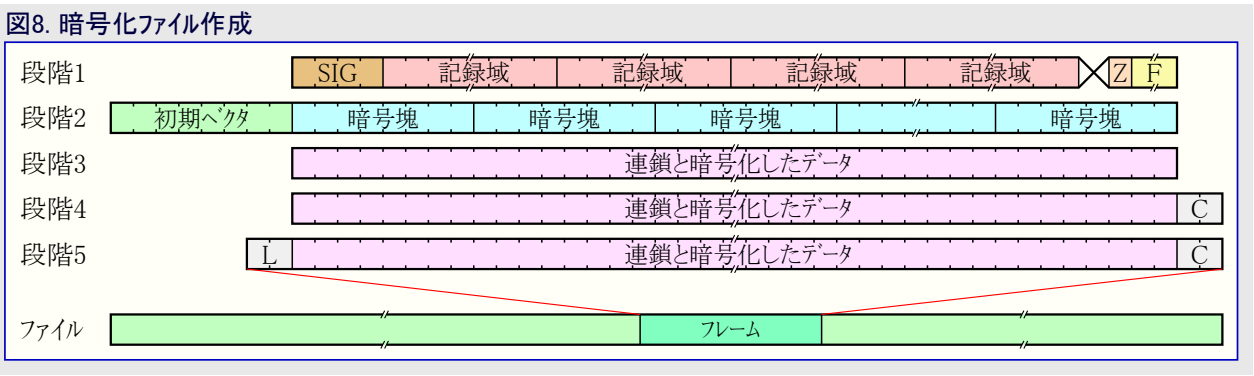


記録域形式はその記録域の先頭ﾊﾞｲﾄとして与えられます。応用ﾃﾞｰﾀは記録域形式1,2,3,4(換言すると、消去、準備、ﾌﾗｯｼｭのﾍﾟｰｼﾞ緩衝部への読みと書き)に分類されます。EEPROM領域に関するﾃﾞｰﾀは記録域形式5内に構成されます。施錠ﾋﾞｯﾄは記録域形式6で送られます。記録域形式0と7は各々ﾌﾚｰﾑと転送の終了用です。

他の全ての記録域、換言すると7を超える記録域識別子を持つそれらは不感知形式です。この任意選択が許可される(Createﾂｰﾙをご覧ください)と、不感知記録域の乱数はこのﾌｧｲﾙ内の散らされた不定位置に配置されます。

出力ﾌｧｲﾙは図8.で図解されるように作成されます。

図8. 暗号化ﾌｧｲﾙ作成



段階は以下で記述されます(番号は図8.参照)。

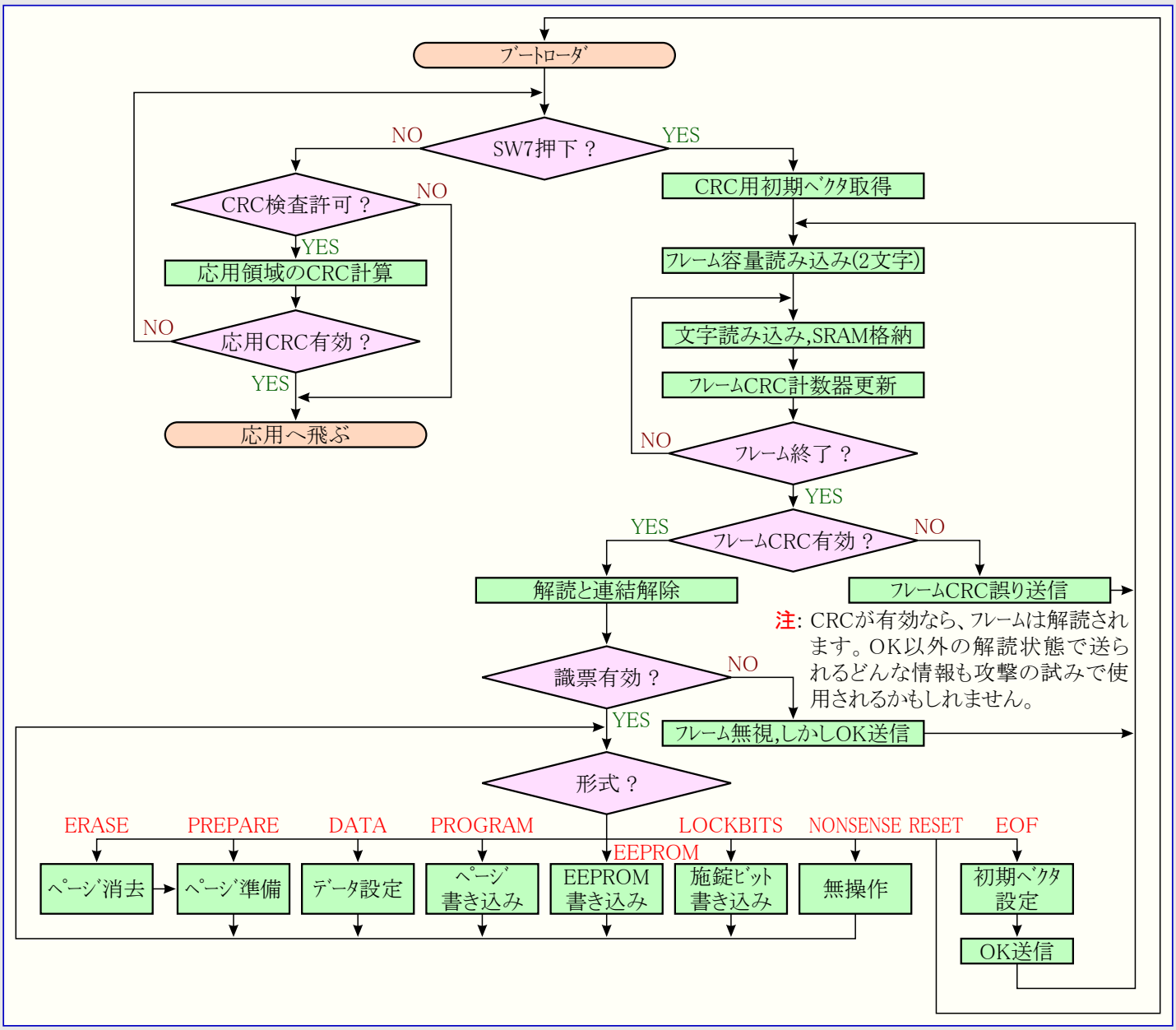
1. ﾃﾞｰﾀは記録域内に構成され、それはその後にﾌﾚｰﾑ識票(SIG)に後続して整列されます。ﾌﾚｰﾑの最後を記すために0(Z)が追加され、ﾌﾚｰﾑは一様な64ﾋﾞｯﾄ生成物を作成するために乱ﾃﾞｰﾀ(F)で穴埋めされます。
2. 初期ﾍﾞｸﾀがﾌﾚｰﾑに付着されます。先頭ﾌﾚｰﾑではそのﾍﾞｸﾀが形態設定ﾌｧｲﾙで与えられるものと等価です。後続ﾌﾚｰﾑでは初期ﾍﾞｸﾀが直前のﾌﾚｰﾑの最後の暗号塊と等価です。
3. 初期ﾍﾞｸﾀと暗号塊は連結されて暗号化されます。初期ﾍﾞｸﾀはその後にﾌﾚｰﾑから取り去られます。
4. CRC-16ﾁｪｯｸｻﾑ(C)が生成されてﾌﾚｰﾑに追加されます。
5. 長さ情報を除いたﾌﾚｰﾑの長さ(L)が計算されて、そのﾌﾚｰﾑの先頭に保存されます。

ﾌﾚｰﾑは出力ﾌｧｲﾙに書かれ、ﾃﾞｰﾀが処理されてしまうまでこの手順が繰り返されます。

ﾌﾞｰﾄﾛｰﾀﾞ

ﾌﾞｰﾄﾛｰﾀﾞはﾃﾞﾊﾞｲｽが暗号化されたﾌｧｰﾑｳｪｱで更新され得る前に目的対象AVRに存在しなければなりません。ﾌﾞｰﾄﾛｰﾀﾞはPCと通信をしてﾌﾗｯｼｭ ﾒﾓﾘの応用領域とEEPROMをﾌﾟﾛｸﾞﾗﾐﾝｸﾞする能力があります。この応用記述と共に含まれるﾌﾞｰﾄﾛｰﾀﾞはIAR Embedded Workbench 2.28a版を用いて作成されていますが、他のどんなｺﾝﾊﾟｲﾗへも移すことができます。ﾌﾟﾛｸﾞﾗﾑの流れは図9.で図解されます。

図9. AVRﾌﾞｰﾄﾛｰﾀﾞのﾌﾟﾛｸﾞﾗﾑの流れ



鍵とﾍｯﾀﾞ ﾌｧｲﾙ

ﾌﾞｰﾄﾛｰﾀﾞをｺﾝﾊﾟｲﾙし得る前に、構成設定が必要ないくつかのﾊﾟﾗﾒｰﾀがあります。始めるにはPC応用のCreateによって生成された暗号鍵とﾍｯﾀﾞ ﾌｧｲﾙがﾌﾞｰﾄﾛｰﾀﾞにｲﾝｸﾙｰﾄﾞできるようにﾌﾞｰﾄﾛｰﾀﾞのﾌｫﾙﾀﾞに複写されなければなりません。このﾌｧｲﾙはﾌﾞｰﾄﾛｰﾀﾞのｿｰｽ ｺｰﾄﾞ内部の#include擬似命令でそれらが参照される時にｲﾝｸﾙｰﾄﾞされます。

例(これはdes.cｿｰｽ ｺｰﾄﾞの先頭の複製です。):

```
        ～
#include"des.h"
#include"bootldr.h"
#include"loader.h"

#if KEY_COUNT > 0

#include"deskeys.inc"
        ～
```

ﾌﾟﾛｼﾞｪｸﾄ ﾌｧｲﾙ

この応用記述は以下のﾃﾞﾊﾞｲｽ用のﾃﾞﾊﾞｲｽ特定ﾌﾟﾛｼﾞｪｸﾄ ﾌｧｲﾙと共にやって来ます。

- ATmega8
- ATmega8515
- ATmega16
- ATmega169
- ATmega32
- ATmega64
- ATmega128

対応するAVRで予め定義されたﾌﾟﾛｼﾞｪｸﾄ ﾌｧｲﾙを使用してください。一覧にされていないAVRﾃﾞﾊﾞｲｽについては可能な限り近い目的対象ﾃﾞﾊﾞｲｽに合うﾃﾞﾊﾞｲｽのﾌﾟﾛｼﾞｪｸﾄ ﾌｧｲﾙを使用して、下で記述されるように以下の項を変更してください。

ﾘﾝｶ ﾌｧｲﾙ

ﾌﾞｰﾄﾛｰﾀﾞが上位ﾒﾓﾘ領域、換言するとﾌﾞｰﾄﾛｰﾀﾞ領域(BLS)に属すため、IARｺﾝﾊﾟｲﾗは修正されたﾘﾝｶ ﾌｧｲﾙが必要です。ﾘﾝｶ ﾌｧｲﾙは.xclの拡張子を持ち、各AVRﾃﾞﾊﾞｲｽ個別でIARｺﾝﾊﾟｲﾗと共に供給されます。この応用記述は2つのﾘﾝｶ ﾌｧｲﾙと共に来て、(割り込みﾍﾞｸﾀ表の選択構成設定に依存して)それの1つが使用されるべきです。

表6. ﾘﾝｶ ﾌｧｲﾙ選択

ﾘﾝｶ ﾌｧｲﾙ名	使用指定
bootldr.xcl	全AVRﾃﾞﾊﾞｲｽ、ｺｰﾄﾞ用に割り当てられた割り込みﾍﾞｸﾀ表(ﾒﾓﾘ節約)
bootldr_interrupts.xcl	全AVRﾃﾞﾊﾞｲｽ、RETIで穴埋めされた割り込みﾍﾞｸﾀ表

ﾘﾝｶ ﾌｧｲﾙは"Project"⇒"Options"、"XLINK"分野、"include"ﾀﾌﾞ、"XCL file name"領域下で定義されます。この応用記述と共に来るﾃﾞﾊﾞｲｽ特定ﾌﾟﾛｼﾞｪｸﾄ ﾌｧｲﾙでﾘﾝｶ ﾌｧｲﾙが既に構成設定されていることに注意してください。

その他ｺﾝﾊﾟｲﾗ設定

ProjectｳｨﾝﾄﾞｳでTargetsをReleaseに設定してください。これはﾃﾞﾊﾞｯｸﾞ ｺｰﾄﾞなしでﾌﾞｰﾄﾛｰﾀﾞの最小全体ｺｰﾄﾞ量を作成します。ｺｰﾄﾞ量は2Kﾊﾞｲﾄのﾌﾞｰﾄ領域だけを持つAVRﾃﾞﾊﾞｲｽに対して際どくなります。

注: TargetsがReleaseに設定されていない場合、ﾌﾟﾛｼﾞｪｸﾄは必然的に正しくﾘﾝｸしません。

以下の設定は"Project"⇒"Options"下で得られるﾀﾞｲｱﾛｸﾞ ｳｨﾝﾄﾞｳで定義される必要があります。ﾃﾞﾊﾞｲｽ特定ﾌﾟﾛｼﾞｪｸﾄ ﾌｧｲﾙで既に全ての設定が定義されていることに注意してください。

表7. 必要なｺﾝﾊﾟｲﾗ設定

分野	ﾀﾌﾞ	設定内容	例
General	Target	目標AVRに合うように"Processor configuration"を設定。	-cpu=m8, AT90mega8
		"Memory model"をSmallに設定。	
		"Configure system using dialogs (not in .XCL file)"のﾁｪｯｸを外す。	
	Library Configuration	"Enable bit definitions in I/O-include files"をﾁｪｯｸ。	
AAVR	Preprocessor	目標AVRに合うようにｼﾝﾎﾞﾙ"INCLUDE_FILE"を定義。	INCLUDE_FILE="iom8.h"
		目標に合うようにｼﾝﾎﾞﾙSPMREGを定義。	SPMREG=SPMCR
XLINK	Output	目標をﾌﾟﾛｸﾞﾗﾐﾝｸﾞできるように出力ﾌｧｲﾙ形式を定義。 Intel-standardに設定。	
	#define	目標ﾌﾞｰﾄﾛｰﾀﾞ領域に合うようにｼﾝﾎﾞﾙBOOT_SIZEを16進ﾊﾞｲﾄで定義。	BOOT_SIZE=800
		目標のﾌﾗｯｼｭ容量に合うようにｼﾝﾎﾞﾙFLASH_SIZEを16進ﾊﾞｲﾄで定義。	FLASH_SIZE=2000
		目標の割り込みﾍﾞｸﾀ表容量に合うようにｼﾝﾎﾞﾙIVT_SIZEを16進ﾊﾞｲﾄで定義。	IVT_SIZE=26
		目標のSRAM量に合うようにｼﾝﾎﾞﾙRAM_SIZEを16進ﾊﾞｲﾄで定義。	RAM_SIZE=400
		(I/O領域に後続する)SRAMの開始に合うようにｼﾝﾎﾞﾙRAM_BASEを16進ﾊﾞｲﾄで定義。	RAM_BASE=60
	Include	"XCL file name"項目下で、"Override default"をﾁｪｯｸ。	
		"XCL file name"項目下で、ﾎﾞｯｸｽ内にﾌｧｲﾙ名を入力。	$PROJ_DIR$¥bootldr.xcl

下表は現在支援されているAVRﾃﾞﾊﾞｲｽに関するｺﾝﾊﾟｲﾗ任意選択のいくつかを要約します。後で説明されるように、ﾌﾞｰﾄﾛｰﾀﾞ開始ｱﾄﾞﾚｽがﾋｭｰｽﾞ設定に依存することに注意してください(**表8.**と「**表9. 推奨ﾋｭｰｽﾞ ﾋﾞｯﾄ**」をご覧ください)。

表8. ｺﾝﾊﾟｲﾗ設定参照基準(推奨ﾋｭｰｽﾞ設定に於ける)

項目	ATmega8	ATmega8515	ATmega16	ATmega169	ATmega32	ATmega64	ATmega128
ﾘﾝｶ ﾌｧｲﾙ名	bootldr.xcl						
BOOT_SIZE	800					800	1000
FLASH_SIZE	2000		4000		8000	10000	20000
IVT_SIZE	26	22	54	5C	58	8C	8C
RAM_SIZE	400	200	400		800	1000	1000
RAM_BASE	60			100	60	100	100

ﾌﾞｰﾄﾛｰﾀﾞのｲﾝｽﾄｰﾙ

ﾌﾞｰﾄﾛｰﾀﾞをｺﾝﾊﾟｲﾙし、その後にAVR Studio®を用いてそれを目的対象に書き込んでください。ﾌﾞｰﾄﾛｰﾀﾞを走行する前に以下のﾋｭｰｽﾞ ﾋﾞｯﾄが次のように形態設定されなければなりません。

- ﾌﾞｰﾄﾛｰﾀﾞ領域の容量。前の方で記述されるように、この領域容量がBOOT_SIZE設定に合うようにﾋｭｰｽﾞ ﾋﾞｯﾄを設定してください。通常、BLSは語で与えられますが、BOOT_SIZEﾊﾟﾗﾒｰﾀがﾊﾞｲﾄで与えられることに注意してください。
- ﾌﾞｰﾄ ﾘｾｯﾄ ﾍﾞｸﾀ。ﾌﾞｰﾄ ﾘｾｯﾄ ﾍﾞｸﾀは許可されなければなりません。
- 発振器任意選択。発振器ﾋｭｰｽﾞ ﾋﾞｯﾄはﾃﾞﾊﾞｲｽﾞ依存です。それらは形態設定が必要かもしれません(UARTに影響を及ぼします)。

注: 正しい発振器任意選択設定に特別な注意を払ってください。例え小さな誤調整でも、おそらく通信失敗に終わるでしょう。このｿﾌﾄｳｪｱは8MHz発振周波数を使用するように設計されています。

表9.は推奨ﾋｭｰｽﾞ ﾋﾞｯﾄ設定を一覧にします。ﾃﾞﾊﾞｲｽ依存のﾋｭｰｽﾞ ﾋﾞｯﾄの詳細な説明についてはﾃﾞｰﾀｼｰﾄをご覧ください。

表9. ﾋｭｰｽﾞ ﾋﾞｯﾄ (0=ﾌﾟﾛｸﾞﾗﾑ、1=非ﾌﾟﾛｸﾞﾗﾑの意味です。)

ﾋｭｰｽﾞ ﾋﾞｯﾄ名	ATmega8	ATmega8515	ATmega16	ATmega169	ATmega32	ATmega64	ATmega128
BOOTSZ1	0						
BOOTSZ0	0						1
BOOTRST	0						

ﾒﾓﾘとﾌﾞｰﾄﾛｰﾀﾞを保護するように施錠ﾋﾞｯﾄを設定することが推奨されますが、それはﾋｭｰｽﾞ ﾋﾞｯﾄが設定されてしまった後でだけです。施錠ﾋﾞｯﾄはAVR Studioを用いて設定することができます。ﾌｧｰﾑｳｪｱが暗号化される時に命令行の引数としてそれらが定義されていれば、ﾌｧｰﾑｳｪｱ更新中にBLS施錠ﾋﾞｯﾄも設定されます。推奨施錠ﾋﾞｯﾄ設定は次の通りです。

- ﾒﾓﾘ施錠ﾋﾞｯﾄ: これらはﾒﾓﾘへの認められていないｱｸｾｽを防ぐように設定されるべきです。

  注: ﾒﾓﾘが施錠されてしまった後はﾃﾞﾊﾞｲｽを消去することなく実装書き込み経由でｱｸｾｽすることができません。
- ﾌﾞｰﾄﾛｰﾀﾞ領域用保護形態: SPMとLPMの命令はBLSに対して読み書きを許されるべきではありません。これはﾌﾞｰﾄﾛｰﾀﾞを不正にする応用領域内のﾌｧｰﾑｳｪｱを保護して解読鍵を安全に保ちます。
- 応用領域用保護形態: 応用領域をｱｸｾｽするSPMとLPMの命令に関して全く制限を設定すべきではなく、さもなければﾌﾞｰﾄﾛｰﾀﾞがそれをﾌﾟﾛｸﾞﾗﾐﾝｸﾞできません。

注: ﾃﾞﾊﾞｲｽが正しく施錠されない場合にﾒﾓﾘはISPｲﾝﾀｰﾌｪｰｽ経由でｱｸｾｽすることができ、ﾌｧｰﾑｳｪｱを暗号化する全体的な意味がないことを理解することが重要です。

下表は存在するAVRﾏｲｸﾛ ｺﾝﾄﾛｰﾗに対する推奨施錠ﾋﾞｯﾄ設定を一覧にします。施錠ﾋﾞｯﾄの詳細な説明についてはﾃﾞｰﾀｼｰﾄをご覧ください。

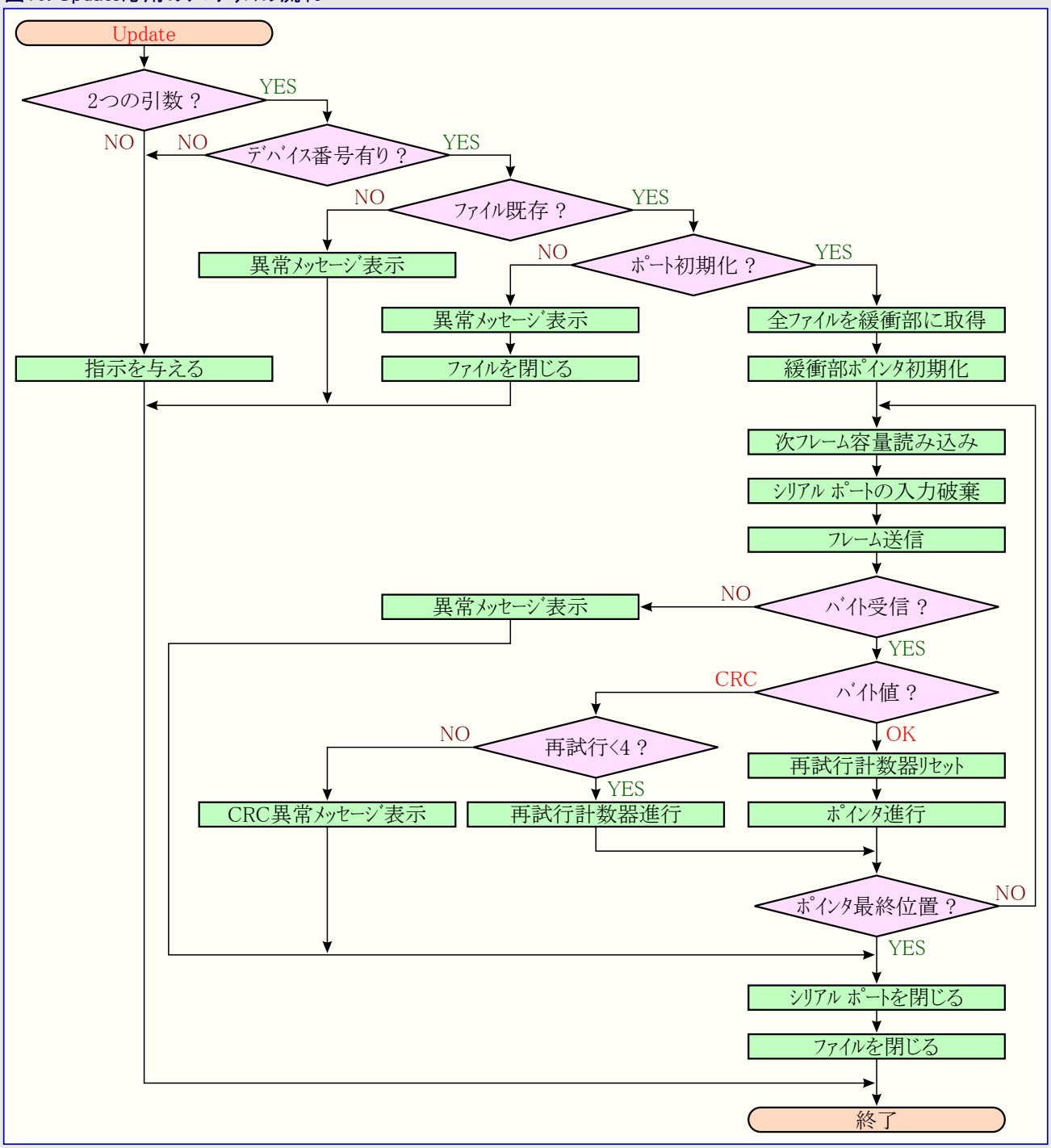
表10. 推奨施錠ﾋﾞｯﾄ設定

ﾋｭｰｽﾞ ﾋﾞｯﾄ名	ATmega8	ATmega8515	ATmega16	ATmega169	ATmega32	ATmega64	ATmega128
BLB12:BLB11	0 0						
BLB02:BLB01	1 1						

PC応用 – Update

Update応用はMicrosoftのVisual C++ 6.0版を使用して作成されました。これは目的対象に(EEPROM、ﾌﾗｯｼｭ、施錠ﾋﾞｯﾄの情報を含む)暗号化ﾌｧｲﾙを送るのに使用されます。ﾃﾞｰﾀはPC上のｼﾘｱﾙ ﾎﾟｰﾄ経由で目的対象ﾊｰﾄﾞｳｪｱ上のUSARTへ直接的に送ることができます。ﾌﾟﾛｸﾞﾗﾑの流れは図10.で図解されます。

図10. Updateｱﾌﾟﾘｹｰｼｮﾝのﾌﾟﾛｸﾞﾗﾑの流れ



Update応用はCreate応用で生成されたﾌｧｲﾙを読みます。このﾌｧｲﾙは暗号化したﾃﾞｰﾀの1つまたはより多くの連結されたﾌﾚｰﾑから成ります。応用は一度に1ﾌﾚｰﾑのﾃﾞｰﾀを送信し、ﾌﾞｰﾄﾛｰﾀﾞからの応答を待つ間に一時停止します。次のﾌﾚｰﾑは応答が受信された後にだけ送信され、そうでなければ応用はﾌﾚｰﾑを再送するか、または通信を閉じるかのどちらかです。

Update応用はｺﾏﾝﾄﾞ ﾌﾟﾛﾝﾌﾟﾄから走行します。このｺﾏﾝﾄﾞ ﾌﾟﾛﾝﾌﾟﾄ引数は**表11.**で一覧にされます。

表11. Updateｱﾌﾟﾘｹｰｼｮﾝのｺﾏﾝﾄﾞ ﾌﾟﾛﾝﾌﾟﾄ引数

引数	説明
<filename.ext>	転送されるべき暗号化されたﾌｧｲﾙへのﾊﾟｽ
-COMn	ｼﾘｱﾙ ﾎﾟｰﾄ、nはｼﾘｱﾙ ﾎﾟｰﾄ番号

更新ｼｽﾃﾑが応用とEEPROMのﾌｧｲﾙで指示されたﾌﾗｯｼｭ ﾒﾓﾘとEEPROMのそれらの部分を更新するだけであることに注意すべきです。応用領域のCRC検査が許可されている、またはCreateﾂｰﾙで消去任意選択が選択されている場合、全応用ﾒﾓﾘはﾌﾟﾛｸﾞﾗﾐﾝｸﾞする(書き込む)前に解除(消去)されます。

ハードウェア構成設定

目的対象ハードウェアは暗号化したファームウェアがブートローダに送られ得る前に正しく構成設定されなければなりません。この応用記述では目的対象としてSTK500が使用されると仮定されます。STK500は以下の通りに形態設定されるべきです。

1. シリアル ケーブルを用いて("RS232 CTRL"と記されたコネクタ経由で)STK500をPCに接続してください。STK500を電源ONしてください。
2. 前で記述されたように、ブートローダをダウンロードして(書き込んで)ヒューズと施錠ビットを設定するのにAVR Studioを使用してください。STK500を電源OFFしてください。
3. シリアル ケーブルを"RS232 SPARE"と記されたコネクタに移動してください。
4. デバイスのRXD/TXD線をRS232 SPAREと記されたコネクタのRXDとTXDに接続してください。PD0がRXD、PD1がTXDになります。
5. PD8(PORTDの8番ピン)をSW7(SWITCHESの8番ピン)に接続してください。
6. STK500をONに切り替える間、SW7を押して保持してください。これはブートローダを開始して更新動作形態に設定します。
7. SW7スイッチを開放してください。
8. 暗号化したデータを目的対象へ送るのに今やPC上のUpdate応用を使用することができます。

性能

以下の項は実行時間とコード量に関してシステム性能を要約します。

速度

受信、復号、データ プログラミング(書き込み)のために目的対象デバイスに必要とされる時間は以下の要素に依存します。

- ファイル容量。転送される必要があるより多くのデータは、より長い時間専有します。
- ボーレート。より高い転送速度は転送時間をより短くします。
- 目的対象AVR速度。より高いクロック周波数は復号時間をより短くします。
- フラッシュ ページのプログラミング(書き込み)時間。これはデバイス不変で変更できません。
- 鍵数。単一鍵DESは3つの鍵の3DESよりも解読がより高速です。
- その他設定。例えば、応用領域のCRC検査は短い時間を専有します。

単一鍵DESについて、全体時間は右式を用いて予測することができます。

$$T_{DES} \fallingdotseq FS \times \left(\frac{5}{BR} + \frac{1}{60+f} \right) + 3$$

ここでFSはバイトでのファイル容量、BRはボーレート(転送速度)、fはMHzでのクロック周波数です。これが予測を与えるだけで、例えばATmega128は僅かに異なる(定数60を54によって置換)ことに注意してください。

既定設定は9600ボーの転送速度と8MHzのクロック周波数です。これは右のように式を簡単化します。

$$T_{DES} \fallingdotseq 0.0026 \times FS + 3$$

3DESについて、全体時間は右の式を用いて予測することができます。

$$T_{3DES} \fallingdotseq FS \times \left(\frac{5}{BR} + \frac{1}{22+f} \right) + 3$$

既定設定に於いて、式は右のように簡単化することができます。

$$T_{3DES} \fallingdotseq 0.0062 \times FS + 3$$

DES解読自体はDESに関して塊当たり245k周期、3DESに関して塊当たり724k周期かかります。これはDESに関して32バイト/s/MHz、3DESに関して10バイト/s/MHzの単位処理量を与えます。

量

下表は目的対象デバイスとコンパイラ任意選択に関してブートローダのコード量を要約します。ATmega128で走行する3DESブートローダを除いて全てのブートローダ任意選択が2Kバイト内に適合することに注目されるかもしれません。

表12. バイトでのブートローダ容量

デバイス	CRC不許可		応用領域のCRC許可		
	DESなし	DES (56ビット)	DES (56ビット)	3DES (112ビット)	3DES (168ビット)
ATmega8	436	1684	1716	1818	1914
ATmega8515	436	1684	1716	1818	1914
ATmega16	476	1752	1786	1888	1984
ATmega169	500	1776	1810	1910	2008
ATmega32	476	1752	1786	1888	1984
ATmega64	504	1810	1844	1946	2042
ATmega128	588	1897	1945	2049	2143

暗号鍵が全く与えられない場合にブートローダがDES/3DES支援なしで構築されることに注目されるべきです。この応用記述はその後に標準ブートローダ システムとして実行し、ブートローダ支援を持つどのAVRでも使用することができます。

要約

この応用記述はﾌﾞｰﾄﾛｰﾀﾞ能力を持つAVRﾏｲｸﾛ ｺﾝﾄﾛｰﾀﾞへ安全にﾃﾞｰﾀを転送する方法が提供されています。この資料は保護されたｼｽﾃﾑを構築する時に実行されるべき技法も強調されています。AVR設計の安全性を増すために、要約として以下の問題が考慮されるべきです。

- 暗号化した形式でのﾀﾞｳﾝﾛｰﾄﾞを支援するﾌﾞｰﾄﾛｰﾀﾞの実装。(製造中に)ﾌﾞｰﾄﾛｰﾀﾞが最初にｲﾝｽﾄｰﾙされる時に将来のﾌｧｰﾑｳｪｱ更新に必要とされる解読鍵が搭載されなければなりません。ﾌｧｰﾑｳｪｱはその後に部外者から内容を保護し、暗号化して更新することができます。
- 応用とﾌﾞｰﾄﾛｰﾀﾞの領域を保護するためにAVRの施錠ﾋﾞｯﾄを使用。ﾃﾞﾊﾞｲｽからの読み込みを防ぐように施錠ﾋﾞｯﾄが設定されると、ﾒﾓﾘ内容は取り出すことができません。施錠ﾋﾞｯﾄが設定されない場合、ﾌｧｰﾑｳｪｱを暗号化する必要がありません。
- 配布前のﾌｧｰﾑｳｪｱ暗号化。暗号化されたﾌｧｰﾑｳｪｱは正しい解読鍵なしでどんな外部実体に対しても役に立ちません。
- 暗号鍵を安全に保護。暗号鍵は2つの場所にだけ格納されるべきで、それは施錠ﾋﾞｯﾄによって保護されているﾌﾞｰﾄﾛｰﾀﾞ内と、製造業者でのﾌｧｰﾑｳｪｱ開発台です。
- 連鎖暗号化ﾃﾞｰﾀ。ﾃﾞｰﾀが連結される時に暗号化された各塊は直前の塊に依存します。結果として、等しい平文は異なる暗号化出力を生成します。
- ﾌｧｰﾑｳｪｱに於いて標準的で予測可能な様式を避ける。殆どのﾌﾟﾛｸﾞﾗﾑは侵入者の手助けのために働くだけの低いｱﾄﾞﾚｽへ飛ぶことで始まる割り込みﾍﾞｸﾀ表のような、共通する枠組みと何らかの予測可能な様式を持ちます。一定の数で未使用領域を穴埋めすることも避けてください。
- 方法を隠す。使用されつつある算法やどの鍵長かに言及する必要は全くありません。侵入者がｼｽﾃﾑについて知らないことが良いことです。暗号化の方法を知ることは或る攻撃者を受け流すと主張されるかもしれませんが、方法について何も知らないことは労力を増して更に多くを防ぐかもしれません。
- 必要とされるなら、ﾌﾞｰﾄﾛｰﾀﾞは応用領域の消去にも使用されるかもしれません。多くの攻撃の試みは通常の動作環境からﾃﾞﾊﾞｲｽを取り外して暴き台での通電を含みます。例えば、LCDが失われている、またはﾒﾓﾘ内にCRC誤りがあることの検出が(ﾌﾞｰﾄﾛｰﾀﾞ領域と解読鍵を含む)全ﾒﾓﾘの完全な消去を始めるかもしれません。
- 更新用の外部通信ﾁｬﾈﾙを使用することが不適または不可能な応用では、ﾌｧｰﾑｳｪｱをATMELのCryptoMemory®ﾃﾞﾊﾞｲｽに格納することができます。このﾒﾓﾘは更新が必要とされる時にﾃﾞﾊﾞｲｽのｽﾛｯﾄに簡単に挿入することができるﾘﾑｰﾊﾞﾌﾞﾙ ｽﾏｰﾄ ｶｰﾄﾞとして外装することができます。ﾏｲｸﾛ ｺﾝﾄﾛｰﾗはﾌﾞｰﾄに於いてCryptoMemoryの存在に関して調べて必要とされる時にﾌｧｰﾑｳｪｱ更新を持ってきます。
- 安全なﾊｰﾄﾞｳｪｱだけを使用。ﾊｰﾄﾞｳｪｱが構造的欠陥を持つ場合、強力な暗号規約は役に立ちません。AVRﾏｲｸﾛ ｺﾝﾄﾛｰﾗでは安全性の問題が全く報告されていません。

この一覧は非常に長く作ることができますが、その目的は単に設計者を正しい方向に向けさせることです。敵の理解力や忍耐力を見くびらないでください。

参照

[1] Douglas Stinson: Cryptography: Theory and Practice, CRC Press, second edition, 1996

[2] Electronic Frontier Foundation: Cracking DES, 1998: http://www.eff.org/descracker.html

[3] Electronic Privacy Information Center: http://www.privacy.org

[4] International Association for Cryptologic Research: http://www.swcp.com/~iacr

[5]Man Young Rhee: Cryptography and Secure Data Communications, McGraw-Hill, 1994

[6] SSH Communications Security: http://www.ssh.com

本社

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131, USA
TEL 1(408) 441-0311
FAX 1(408) 487-2600

国外営業拠点

Atmel Asia
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
TEL (852) 2245-6100
FAX (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-Yvelines
Cedex
France
TEL (33) 1-30-60-70-00
FAX (33) 1-30-60-71-11

Atmel Japan
104-0033 東京都中央区
新川1-24-8
東熱新川ﾋﾞﾙ 9F
ｱﾄﾒﾙ ｼﾞｬﾊﾟﾝ株式会社
TEL (81) 03-3523-3551
FAX (81) 03-3523-7581

製造拠点

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
TEL 1(408) 441-0311
FAX 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
TEL 1(408) 441-0311
FAX 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3
France
TEL (33) 2-40-18-18-18
FAX (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex
France
TEL (33) 4-42-53-60-00
FAX (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR
Scotland
TEL (44) 1355-803-000
FAX (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn
Germany
TEL (49) 71-31-67-0
FAX (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Biometrics
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex
France
TEL (33) 4-76-58-47-50
FAX (33) 4-76-58-47-60

文献請求
www.atmel.com/literature

本応用記述はATMELのAVR230応用記述(doc2541.pdf Rev.2541D-04/05)の翻訳日本語版です。日本語では不自然となる重複する形容表現は省略されている場合があります。日本語では難解となる表現は大幅に意訳されている部分もあります。必要に応じて一部加筆されています。頁割の変更により、原本より頁数が少なくなっています。

必要と思われる部分には( )内に英語表記や略称などを残す形で表記しています。

青字の部分はﾘﾝｸとなっています。一般的に赤字の0,1は論理0,1を表します。その他の赤字は重要な部分を表します。