セキュアブートガイド

Armadillo-X2/Armadillo-IoT ゲートウェイ G4

Version 1.0.0
2025/06/25

株式会社アットマークテクノ [https://www.atmark-techno.com]

Armadillo サイト [https://armadillo.atmark-techno.com]

セキュアブートガイド: Armadillo-X2/Armadillo-IoT ゲートウェイ G4

株式会社アットマークテクノ

Version 1.0.0
2025/06/25

目次

図目次

表目次

1. はじめに

1.1. 本書の目的と対象読者

本書は Armadillo-X2 及び Armadillo-IoT ゲートウェイ G4 を使用するユーザーが以下を理解することを目的としています。

- 開発機に対してセキュアブートの有効化及びルートファイルシステムを暗号化する方法
- 量産機に対してセキュアブートの有効化及びストレージの暗号化を適用し開発したアプリケーションをインストールする方法
- 運用時にセキュアブート有効化済み個体に対してブートローダー及び Linux カーネルイメージを更新する方法

本書は、以下の読者を対象としています。

- Armadillo-X2 及び Armadillo-IoT ゲートウェイ G4 でセキュアブートの有効化を検討している

  セキュアブートの有効化を検討する上でこのドキュメントは役に立ちます。

- Armadillo での基本的な開発方法を理解している

  ATDE の使用方法、Armadillo Base OS の思想の理解、SWU イメージの使用方法、ABOS Web の使用方法などが含まれます。詳細については製品マニュアルをご参照ください。

  - Armadillo-X2 ：「製品マニュアル [https://manual.atmark-techno.com/armadillo-x2/armadillo-x2_product_manual_ja]」
  - Armadillo-IoT ゲートウェイ G4：「製品マニュアル [https://manual.atmark-techno.com/armadillo-iot-g4/armadillo-iotg-g4_product_manual_ja]」

- 基本的な Linux の扱い方を知っている

  ls や cd などの基本的なコマンドをターミナル上で実施するためです。

1.2. 本書の位置づけ

本書は Armadillo-X2 及び Armadillo-IoT ゲートウェイ G4 におけるセキュアブート有効化の実践方法を記したドキュメントです。

セキュアブートの理論的背景は最小限に留めていますので、必要に応じて関連文献をご参照ください。例えば、Armadillo-X2 及び Armadillo-IoT ゲートウェイ G4 では NXP の uboot-imx を使用していますが、より詳細な技術的内容については以下をご参照ください。

https://github.com/nxp-imx/uboot-imx/tree/lf_v2024.04/doc/imx

また、Armadillo を用いたアプリケーションの開発方法やハードウェアの仕様については触れておりません。それらについて知りたい場合は製品マニュアルをご参照ください。

- Armadillo-X2 ：「製品マニュアル [https://manual.atmark-techno.com/armadillo-x2/armadillo-x2_product_manual_ja]」

- Armadillo-IoT ゲートウェイ G4：「製品マニュアル [https://manual.atmark-techno.com/armadillo-iot-g4/armadillo-iotg-g4_product_manual_ja]」

1.3. JC-STAR とセキュアブートの関連

セキュリティ要件適合評価及びラベリング制度（JC-STAR: Labeling Scheme based on Japan Cyber-Security Technical Assessment Requirements）とは、独立行政法人 情報処理推進機構（IPA）が主導になり進めている日本のセキュリティ適合性評価制度です。

対象となる製品はインターネットプロトコル（IP）を使用したデータの送受信機能を持つ IoT 製品であり、Armadillo も含まれます。組織として脆弱性に対応する体制が整っており、製品のセキュリティ機能として必要な水準を満たことを評価した上で、その申請が通るとラベルが付与されます。

政府機関や重要インフラ事業者が製品選定を行う際に JC-STAR のラベルの有無を参考にすることが想定されます。そのため、特に国の重要な施設といった公共事業向けに使用される製品を開発する場合はラベルの取得を目指すことが推奨されます。

適合基準のレベルは 4 つ存在します。求められるセキュリティ水準に応じたセキュリティ技術要件として ★1 から ★4 までのレベルが設定されています。

★1 は 2025 年 3 月 25 日から申請の受付が始まりました。製品として最低限必要なセキュリティ機能を保持していることが条件ではありますが、この時点でセキュアブートを有効化することは要件にありません。

★2 以降 は製品類型ごとにセキュリティ要件が異なり、2025 年 5 月現在では詳細な情報は明かされていません。しかし、レベルが上がるにつれてセキュリティ要件が厳しくなり、セキュアブート有効化による保護が必須要件になることは十分に予想されます。

JC-STAR に関するより詳細な情報は以下の URL からご確認いただけます。

https://www.ipa.go.jp/security/jc-star/index.html

1.4. 本書の構成

本書はセキュアブートに関する最低限必要な知識の説明から始まり、その後に実際にセキュアブートを有効化する手順を示します。第２章以降の構成は以下のようになります。

２章　背景

この章ではセキュアブートを有効化するにあたり前提となる知識を説明します。セキュアブートがどのようなものなのか、なぜ行う必要があるのか、その仕組みを知ることでセキュリティにどのように役立つのか示します。セキュアブートについて理解がある読者はこの章を読み飛ばして３章に進んでも問題ありません。

３章　導入編

この章では開発用 Armadillo に対してセキュアブートを有効化するための手順を示します。セキュアブートを有効化するために必要なスクリプトの取得、環境のセットアップ方法についての説明から始まり、セキュアブートの有効化及びルートファイルシステム（rootfs）の暗号化を実際に行うことがこの章の目標です。

４章　量産編

アプリケーションの開発段階が終了すると、次に量産用 Armadillo に対してセキュアブートの有効化及び暗号化を適用し、さらに開発したアプリケーションをインストールする必要があります。この章で

は量産用 Armadillo に対してセキュアブートの有効化及び rootfs 用、ログ書き込み用、ファームウェア用、アプリケーション用パーティションの暗号化を行い、開発したアプリケーションをインストールする手順を示します。

5章　運用編

サイバー攻撃に対する防御策の一環として Linux カーネルイメージ及びブートローダーの更新は欠かせません。この章ではセキュアブート有効化済みの Armadillo にインストールされている署名済みブートローダーと署名済み Linux カーネルイメージの更新方法について説明します。

1.5. 表記上のルール

本書では以下のような意味でフォントを使いわけています。

表 1.1 使用しているフォント

フォント例	説明
本文中のフォント	本文
[PC ~]$ ls	プロンプトとユーザ入力文字列

本書に記載されているコマンドの入力例は、表示されているプロンプトによって、それぞれに対応した実行環境を想定して書かれています。「 / 」の部分はカレントディレクトリによって異なります。各ユーザのホームディレクトリは「 ~ 」で表します。

表 1.2 表示プロンプトと実行環境の関係

プロンプト	コマンドの実行環境
[PC /]#	作業用 PC の root ユーザで実行
[PC /]$	作業用 PC の一般ユーザで実行
[ATDE ~/]#	ATDE 上の root ユーザで実行
[ATDE ~/]$	ATDE 上の一般ユーザで実行
[armadillo /]#	Armadillo 上 Linux の root ユーザで実行
[armadillo /]$	Armadillo 上 Linux の一般ユーザで実行
[container /]#	Podman コンテナ内で実行
⇒	Armadillo 上 U-Boot の保守モードで実行

本書では以下のようにアイコンを使用しています。

注意を払わなければ、システム障害やデータ損失といった取り返しがつかない状態を引き起こすような重要な事柄に関する情報を記載します。

注意を促すものの、大きなリスクを伴わない情報を記載します。

重要な事柄ではあるものの、リスクや警告を伴わない情報を記載します。

役に立つ情報を記載します。

用語の説明や補足的な説明を記載します。

2. 背景

2.1. セキュリティの重要性

近年、組み込みデバイスの増加に伴い、ハッキングによる脅威も増加の一途を辿っています。サイバー攻撃による被害はそのデバイスの使用者はもちろんですが、信用の失墜や損害賠償といった開発元の企業が被る損害も計り知れません。そのようなリスクを最小限に抑えるために、開発する段階でセキュリティを意識することは重要と言えます。

組み込みデバイスへの攻撃は様々な方向から行われます。例えば以下のような攻撃方法が考えられます。

- データを抜き取るために eMMC をボードから引き剥がすなどの物理的な攻撃
- DDoS 攻撃などのネットワークを介した攻撃、またはボットネットとして使用される
- 電磁波を使用して故障やデータ改ざんを引き起こす攻撃
- Linux カーネルやアプリケーションの脆弱性をついた攻撃

上記は一例であり、他にも様々な攻撃方法があります。

ある方向のセキュリティ対策が強固な場合、攻撃者は回避可能な別の方向がないのか模索します。特に、IoT デバイスはネットワーク上のサービスとデータのやり取りを行います。通信路やパケットの暗号化、サーバーとデバイスの相互認証などの対策を講じたとしても、IoT デバイス上にあるソフトウェアに攻撃者のコードを組み込むことで対策を回避してシステムに侵入される可能性があるのです。

そのため、ある一点に対して強固なセキュリティ対策を施すよりも、システム全体で防御策を講じる多層防御の考え方が重要になります。とはいえ、全ての攻撃に対して対策を考えるのはコスト面を考えたとしても現実的ではありません。

デバイス上にどのような資産が保有されているのか、攻撃方法として何がありえるのか、もし攻撃された場合に影響範囲はどの程度なのかといったリスクアセスメントを行った上で、セキュリティ対策を施すことにより大きな費用対効果が得られます。

2.2. セキュアブートとは

何らかの手段でデバイスにエクスプロイトコードがインストールされることで Linux カーネルを含むソフトウェアが改ざんされることが考えられます。

セキュアブートは、起動ソフトウェアのデジタル署名を用いて正規ソフトウェアであることを確認してから起動する機能のことです。攻撃者によって作られた不正なコードを実行前に検出することができます。

デバイスにあらかじめ書き込まれた書き換え不可のハッシュ値を起点に、署名されたブートローダーおよび OS が正規のものであるか検証しながら順番に起動します。万が一、署名されたソフトウェアが改ざんされていた場合には起動に失敗します。

このプロセスがあることで、セキュアブートは電源投入から署名されたソフトウェアの起動までの起動プロセスを保護します。

ただし、起動後に署名されたソフトウェアが改ざんされた場合にはセキュアブートでは対処できません。そのため、セキュアブートを有効にしたとしても、パスワードを強固なものにする、JTAG を無効化し侵入口を塞ぐ、アプリケーションをセキュアな設計にするなどの対策は必須です。

2.3. チェーンオブトラスト

セキュアブートはチェーンオブトラスト (chain of trust) と表裏一体に実装されます。チェーンオブトラストとは、その名の通り、信頼を繋いでいく形態のことを指します。

ルートオブトラストと呼ばれる基礎となる情報から枝葉のように繋がれた情報を認証していくことで、繋がれた個々のコンポーネントだけでなくシステムを信頼できるものにしてくれます。セキュアブートは、起動時にソフトウェアを順番に認証することで信頼を次に繋いでいるのです。

セキュアブートによる保護範囲をどこまでにするかによりますが、起動時に認証されたソフトウェアで別の情報を認証すれば、チェーンオブトラストを繋げていくことができます。

また、IoT デバイスとクラウドサービスから構成されるような広範囲に及ぶシステムは特に信頼が必要になります。信頼できるセキュリティ基盤を構築するためには、構成するソフトウェアが正規のリリース物であることを確認することが重要になります。チェーンオブトラストを採用することでより信頼できる IoT システムとなり得るのです。

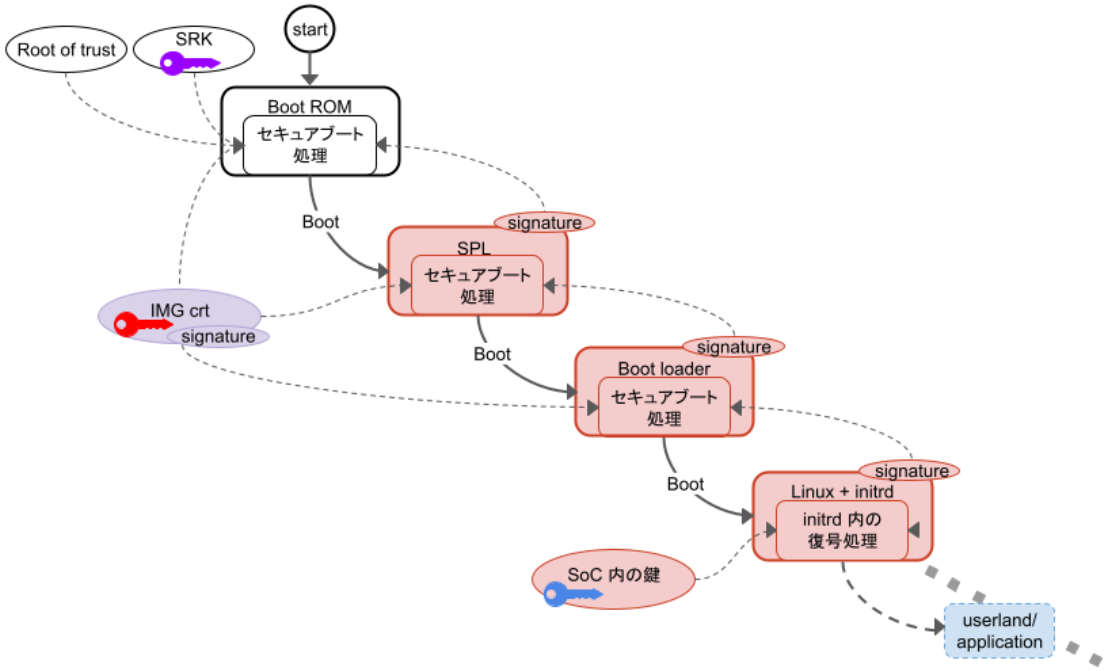

図 2.1 チェーンオブトラスト

では、どういったケースでセキュアブートを採用するべきでしょうか。セキュアブートはセキュアな組み込みデバイスを実現する上で最初のステップにするべき技術です。

しかし、導入するのであればコスト面にも配慮することが必要でしょう。まず、製造時の追加コストが必要です。鍵等を書き込む工程が必要になります。ただ書き込めばよいわけではなく、漏洩があってはいけないので物理的な隔離などセキュアに書き込む必要があります。メンテナンスにも追加のコストが必要です。

ソフトウェアのリリース時にはソフトウェアの署名が必要になります。署名鍵については物理的な隔離などの漏洩対策が必要になります。

費用対効果を検討してからの導入をお勧めします。

2.4. HAB とは

HAB (High Assurance Booting) は、NXP が提供するセキュアブートの実装です。i.MX 8M Plus の BootROM には HABv4 が組み込まれます。デフォルトでは無効な状態になっていますが、一度、eFuse に情報を書き込むことでセキュアブートが有効になり、それ以降、有効な状態のまま変更不可能になります。HABv4 で規定する仕様では次の情報をブートローダーイメージに追加することで BootROM が起動時にブートローダーの認証を行います。

- CSF (Command Sequence File)、IVT (Image Vector Table)
- SRK (Super Root Key)、CSF、IMG 署名確認鍵
- イメージの署名

NXP は署名ツールとして CST (Code Signing Tool) をリリースしています。本来、署名範囲などは環境によって様々な実装がなされるべきなので署名ツール自体にはその辺りの仕様が含まれません。ブートローダーの実装仕様に依存します。Armadillo Base OS で採用される uboot-imx のセキュアブート処理では、Trusted Firmware-A (ATF)、OP-TEE OS、Linux カーネルイメージまでが認証の対象になります。署名に関する概要は以下のとおりです。

- 署名確認用の鍵は X.509 証明書に対応する
- 署名は RSA/ECC に対応する
  - RSA 1024, 2048, 3072, 4096 bits
  - ECC NIST P-256, NIST P-384, NIST P-521
- 署名のダイジェストは SHA256 のみ

HAB の詳しい仕様は以下を参照してください。

i.MX Secure Boot on HABv4 Supported Devices

https://www.nxp.com/search?keyword=AN4581

Encrypted Boot on HABv4 and CAAM Enabled Devices

https://www.nxp.com/search?keyword=AN12056

以降でダウンロードする imx-boot ディレクトリ内のドキュメントもご参照ください。

imx-boot/uboot-imx/doc/imx/habv4/introduction_habv4.txt

imx-boot/uboot-imx/doc/imx/habv4/guides/
mx8m_secure_boot.txt

2.5. セキュアブートの有効化及びストレージの暗号化で保証される範囲

Armadillo-X2 および Armadillo-IoT ゲートウェイ G4 ではセキュアブートによる保護はブートローダーと Linux カーネルイメージに対して行われ、ユーザーランドで使用するソフトウェアに対しては機能しません。

代わりに、ストレージを暗号化して、Armadillo 起動時に復号することでユーザーランドが正規の Armadillo によって起動されたことを保証します。

表 2.1 セキュアブートの有効化及び暗号化で保証される範囲

ソフトウェア	セキュアブートの有効化	ストレージの暗号化
ブートローダー	○	×
Linux カーネルイメージ	○	×
ユーザーランドのソフトウェア	×	○

セキュアブートの有効化及びストレージの暗号化は起動時に正規のものであることを保証するための機能であり、**起動後にファイルが改ざんされるなどの状況には対処できません。**

起動後のセキュリティ対策は別の方法で行う必要があります。

2.6. セキュリティとして期待される効果

Armadillo-IoT ゲートウェイ G4/Armadillo-X2 と Armadillo Base OS を利用することで、以下のようなセキュリティ機能を実現することができます。

表 2.2 セキュリティとして期待される効果

機能	期待される効果
セキュアブートの有効化	ブートローダーと Linux Kernel の改竄を検出することができます。また、 署名とその検証によって、正規ソフトウェアのみの起動を強制させることができます。
ストレージの暗号化	ファイルがフラッシュメモリに保存されている間は暗号化によって ファイルは保護されます。 eMMC を外して他の個体に付け替えても復号できないので、保存されているアプリケーションやデータなどを外部から読み取るまたは改竄することはできません。 そのため、ファイルに情報資産が含まれるケースでの活用も考えられます。

以降の作業を行うことで、セキュアブートを有効化し、ストレージを暗号化することが出来ます。

ブートローダーの暗号化に関しては対応しておりません。

3. 導入編

3.1. 署名環境を構築する

まず、セキュアブートを有効化するために必要な環境を構築していきます。

3.2. 署名環境について

ここで利用する署名環境の構成は以下のとおりです。

表 3.1 署名環境

ツール/パッケージ	説明
ATDE(Atmark Techno Development Environment)	提供元 : アットマークテクノ Armadillo シリーズの開発環境として VirtualBox イメージとして配布されます。
CST(Code Signing Tool) パッケージ	提供元 : NXP HAB の署名を行うツールや鍵を生成するツールです。 Linux パッケージとして提供されています。
mkswu パッケージ	提供元 : アットマークテクノ Armadillo Base OS のファームウェアアップデートの仕組みである SWUpdate 用のアップデートファイルを生成するツールを含みます。
build-rootfs-[VERSION].tar.gz	提供元 : アットマークテクノ Armadillo Base OS のルートファイルシステムを生成するツールを含みます。 Alpine Linux をベースにアットマークテクノのパッケージを加えた構成になります。 **Armadillo-X2** 「開発環境・ツール [https://armadillo.atmark-techno.com/resources/software/armadillo-x2/tools]」 **Armadillo-IoT ゲートウェイ G4** 「開発環境・ツール [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-g4/tools]」
imx-boot-[VERSION].tar.gz	提供元 : アットマークテクノ ブートローダーのソースコードや、セキュアブートの有効化及びストレージ暗号化用の SWU イメージをつくるスクリプトを含みます。 **Armadillo-X2** 「ブートローダー [https://armadillo.atmark-techno.com/resources/software/armadillo-x2/boot-loader]」 **Armadillo-IoT ゲートウェイ G4** 「ブートローダー [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-g4/boot-loader]」
baseos-[VERSION].tar.zst	提供元 : アットマークテクノ Armadillo Base OS のルートファイルシステムのビルド済みバイナリです。 **Armadillo-X2** 「製品ソフトウェア [https://armadillo.atmark-techno.com/resources/software/armadillo-x2/baseos]」 **Armadillo-IoT ゲートウェイ G4** 「製品ソフトウェア [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-g4/baseos]」

3.2.1. uboot-imx のセキュアブートの実装仕様

セキュアブートの実装は uboot-imx のガイドに準拠しています。詳しい仕様は以下を参照してください。取得したソースツリー内にドキュメントがあります。

```
imx-boot-[VERSION]/uboot-imx/doc/imx/habv4/introduction_habv4.txt
imx-boot-[VERSION]/uboot-imx/doc/imx/habv4/guides/mx8m_secure_boot.txt
```

3.2.2. ATDE を準備する

製品マニュアルを参考に作業をしてください。

- Armadillo-X2：「仮想環境のセットアップ [https://manual.atmark-techno.com/armadillo-x2/armadillo-x2_product_manual_ja/ch03.html#sct.setup-environment]」
- Armadillo-IoT ゲートウェイ G4：「仮想環境のセットアップ [https://manual.atmark-techno.com/armadillo-iot-g4/armadillo-iotg-g4_product_manual_ja/ch03.html#sct.setup-environment]」

3.2.3. mkswu を準備する

mkswu は SWUpdate に対応したアップデートファイルを生成するツールです。開発環境である ATDE にインストールされています。

あらかじめ、mkswu を用いて生成した initial_setup.swu を Armadillo にインストールする必要があります。詳しくは製品マニュアルをご参照ください。

- Armadillo-X2：「Armadillo に初期化設定をインストールする [https://manual.atmark-techno.com/armadillo-x2/armadillo-x2_product_manual_ja/ch03.html#sct.setup-armadillo-with-vscode]」
- Armadillo-IoT ゲートウェイ G4：「Armadillo に初期化設定をインストールする [https://manual.atmark-techno.com/armadillo-iot-g4/armadillo-iotg-g4_product_manual_ja/ch03.html#sct.setup-armadillo-with-vscode]」

セキュアブート有効化に対応する mkswu は 4.0-1 以上になります。

SWU イメージの暗号化

SWU イメージはデフォルト設定では暗号化されません。通信路は TLS で守られても、ファイルで保管されているときには平文なので SWU イメージ内にある機密情報が漏洩する可能性があります。SWU イメージに漏洩すると問題のある情報資産を含めるときには必ず暗号化するべきです。

initial_setup.swu を作成するときに SWU イメージを暗号化する設定を行うことができます。詳しくは製品マニュアルをご参照ください。

- Armadillo-X2：「initial_setup.swu 初回生成時の各種設定 [https://manual.atmark-techno.com/armadillo-x2/armadillo-x2_product_manual_ja/ch03.html#fig.first-time-mkswu-setting]」
- Armadillo-IoT ゲートウェイ G4：「initial_setup.swu 初回生成時の各種設定 [https://manual.atmark-techno.com/armadillo-iot-g4/armadillo-iotg-g4_product_manual_ja/ch03.html#fig.first-time-mkswu-setting]」

バージョンの確認方法は以下のとおりです。

```
[ATDE ~]$ dpkg -l mkswu
ii  mkswu        7.6.2-1
```

図 3.1 mkswu のバージョン確認

3.2.4. セキュアブートの有効化及びストレージの暗号化に必要なスクリプトの取得

以下の URL から「ブートローダー ソース（u-boot 等）」を取得してください。

- Armadillo-X2：「ブートローダー [https://armadillo.atmark-techno.com/resources/software/armadillo-x2/boot-loader]」
- Armadillo-IoT ゲートウェイ G4：「ブートローダー [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-g4/boot-loader]」

取得したアーカイブ内にセキュアブートの有効化およびストレージ暗号化に必要なスクリプトが含まれています。

本書作成時点で利用したパッケージは以下のとおりです。

- imx-boot-2020.04-at26.tar.gz

ブートローダーのソースコードのアーカイブを展開します。以降、このアーカイブをホームディレクトリに展開した想定で説明します。

```
[ATDE ~]$ tar xaf imx-boot-[VERSION].tar.gz ❶
[ATDE ~]$ cd imx-boot-[VERSION] ❷
```

図 3.2 ブートローダーのソースコードのアーカイブを展開する

❶ [VERSION] はバージョンによって変化します

❷ imx-boot-[VERSION]に移動します。

3.2.5. セキュアブートの有効化及びストレージ暗号化に使用するスクリプトの説明

imx-boot-[VERSION] に存在する secureboot.sh というスクリプトを用いてセキュアブートの有効化及びストレージの暗号化を行います。

以降の作業で使用する secureboot.sh のオプションは次のとおりです。

表 3.2 以降で使用する secureboot.sh のオプション

オプション	説明
setup	imx-boot-[VERSION] と同じディレクトリ階層に、セキュアブートの有効化及びストレージ暗号化を行うための環境である secureboot_x2 ディレクトリを作成します。
build	セキュアブートの有効化及びルートファイルシステムの暗号化を行うための SWU イメージを作成します。
make_installer	開発用 Armadillo の環境を複製したインストールディスクメージを作成する SWU イメージを生成します。作成したインストールディスクイメージを量産用 Armadillo にインストールすることで以下のような環境を準備できます。 ・ セキュアブートの有効化 ・ ルートファイルシステム及びアプリケーション領域などの暗号化 ・ 開発用 Armadillo の環境の複製

3.2.6. 実行環境の生成

以下のように、imx-boot-[VERSION] 上で secureboot.sh の setup オプションを実行してください。

```
[ATDE ~/imx-boot-[VERSION]]$ ./secureboot.sh setup
imx-code-signing-tool package is needed to setup secureboot.
Install imx-code-signing-tool? [Y/n] ❶
...省略
OK: Linking /home/atmark/secureboot_x2/build-rootfs to /home/atmark/secureboot_x2/build-rootfs-[VERSION]
OK: Linking /home/atmark/secureboot_x2/imx-boot to /home/atmark/imx-boot-[VERSION]
Setup /home/atmark/secureboot_x2 successfully. Run './secureboot.sh build' next. ❷
```

図 3.3 secureboot.sh setup の実行

❶ Enter を押すと、CST の Linux パッケージをインストールします。

❷ secureboot_x2 ディレクトリが作成されます。

imx-boot-[VERSION] ディレクトリと同じ階層に secureboot_x2 ディレクトリが作成されます。

デフォルトでは、secureboot_x2 ディレクトリ内は以下の構成になっています。

```
secureboot_x2
├── build-rootfs -> /home/atmark/secureboot_x2/build-rootfs-[VERSION] ❶
├── build-rootfs-[VERSION] ❷
├── cst ❸
├── imx-boot -> /home/atmark/imx-boot-[VERSION] ❹
├── out ❺
├── secureboot.conf ❻
```

```
├── secureboot.sh -> /home/atmark/imx-boot-[VERSION]/secureboot.sh ❼
└── swu ❽
└── tmp ❾
```

❶ シンボリックリンク元が使用する build-rootfs ディレクトリになります。

❷ build-rootfs-[VERSION] が imx-boot-[VERSION] と同じ階層になければダウンロードします。

❸ CST によって生成された鍵などが配置されます。

❹ シンボリックリンク元が使用する imx-boot ディレクトリになります。

❺ Linux カーネルおよびブートローダーのイメージが配置されます。

❻ secureboot.sh の設定ファイルです。imx-boot-[VERSION]/secureboot.conf.example からコピーされます。

❼ シンボリックリンク元が使用する secureboot.sh になります。

❽ セキュアブートの有効化およびストレージ暗号化を行うための SWU イメージが配置されます。

❾ secureboot.sh を実行した時に生成される一時ファイルが配置されます。

CST(Code Signing Tool)は NXP からソースコードをダウンロードしたものを使用することも可能です。

https://www.nxp.com/search?keyword=IMX_CST_TOOL_NEW

既にダウンロードした CST を使用している場合は、secureboot_x2/secureboot.conf を以下のように修正してください。

以下では、ダウンロードしてきた CST（ cst-[VERSION] ）をホームディレクトリに配置していることを想定しています。

```
[ATDE ~]$ tar xaf IMX_CST_TOOL_NEW.tgz ❶
[ATDE ~]$ ls
cst-[VERSION] :(省略)
[ATDE ~]$ cat ../secureboot_x2/secureboot.conf

:(省略)

# CST directory where signing keys are kept. Keep preciously!
#CST="$SCRIPT_DIR/cst"
CST="/home/atmark/cst-[VERSION]" ❷

:(省略)
```

図 3.4 ダウンロードした IMX_CST_TOOL_NEW を使用する

❶ ダウンロードした CST のアーカイブを展開します。

❷ 展開した CST ディレクトリを絶対パスで CST 変数に指定してください。

以上で環境構築は完了です。

3.2.7. secureboot.conf の設定方法

secureboot.conf はセキュアブートイメージ生成スクリプト secureboot.sh の設定ファイルです。以下はコンフィグの一部です。

CST_ECC　既存の CA 証明書を利用するかどうかを設定します。

- y: EC を利用する
- n: RSA を利用する

CST_KEYLEN, CST_KEYTYPE　鍵長を設定する

表 3.3 セキュアブート用の鍵の種類

Algorithm	CST_KEYLEN	CST_KEYTYPE
RSA 1024	1024	1024_65537
RSA 2048	2048	2048_65537
RSA 3072	3072	3072_65537
RSA 4096	4096	4096_65537
EC NIST P-256	p256	prime256v1
EC NIST P-384	p384	secp384r1
EC NIST P-521	p521	secp521r1

CST_SOURCE_INDEX　SRK は最大 4 本まで持つことができます。この設定ではどの SRK を利用するのか設定します。設定値は 0 はじまりで、0 がデフォルトです。

CST_UNLOCK_SRK　SRK のリボーク時に利用します。デフォルト設定では攻撃や事故の防止のために、リボークができない状態になっています。#CST_UNLOCK_SRK=y のコメントを外すことでリボークが可能な状態になります。

LINUX_IMAGE,LINUX_DTB　FIT イメージに組み込むための Linux カーネルイメージとデバイスツリーブロブ (DTB) のパスを指定します。

LINUX_DTB_OVERLAYS_PREAPPLY　DTB に直接適用するデバイスツリーオーバーレイ（DTBO）ファイルを指定します。指定方法の例を示します。

```
LINUX_DTB_OVERLAYS_PREAPPLY=(
        armadillo_iotg_g4-aw-xm458.dtbo
        armadillo_iotg_g4-lte-ext-board.dtbo
)
```

LINUX_MODULES　必要であればインストールする Linux modules のパスを指定します。

LINUX_INITRD　必要であればインストールする initrd のパスを指定します。

3.3. セキュアブートの有効化及びルートファイルシステム（rootfs）の暗号化

開発用 Armadillo に対してセキュアブートの有効化及び rootfs を暗号化する流れは以下のとおりです。

1. 「3.3.1. セキュアブートの有効化および rootfs の暗号化を行う SWU イメージの作成」
2. 「3.3.2. SWU イメージのインストール」

3.3.1. セキュアブートの有効化および rootfs の暗号化を行う SWU イメージの作成

secureboot.sh の build を実行します。

この時に使用できるオプション引数は以下の通りです。

表 3.4 build の引数

オプション引数	説明
--image	使用する Linux カーネルイメージを指定できます。絶対パスで指定してください。secureboot.conf 内の LINUX_IMAGE 変数で指定することもできます。
--initrd	署名および rootfs の暗号化時に使用する initrd を指定できます。絶対パスで指定してください。secureboot.conf 内の LINUX_INITRD 変数で指定することもできます。
--dtb	使用する dtb ファイルを指定できます。絶対パスで指定してください。secureboot.conf 内の LINUX_DTB 変数で指定することもできます。
--dtbo	使用する dtbo ファイルを指定できます。絶対パスで指定してください。secureboot.conf 内の LINUX_DTB_OVERLAYS_PREAPPLY 変数で指定することもできます。
--modules	使用する modules ディレクトリを指定できます。絶対パスで指定してください。secureboot.conf 内の LINUX_MODULES 変数で指定することもできます。

at-dtweb（Device Tree をカスタマイズするツール）を利用して作成した dtbo ファイルを使用する場合は以下のように指定してください。

ここでは、作成した dtbo ファイル（ armadillo_iotg_g4-at-dtweb.dtbo ）をホームディレクトリに配置したとします。

```
[ATDE ~/imx-boot-[VERSION]]$ ./secureboot.sh build ¥
    --dtbo /home/atmark/armadillo_iotg_g4-at-dtweb.dtbo
```

図 3.5 カスタマイズした dtbo ファイルを使用する場合

at-dtweb についての説明は製品マニュアルをご参照ください。

- Armadillo-X2 :「Device Tree をカスタマイズする [https://manual.atmark-techno.com/armadillo-x2/armadillo-x2_product_manual_ja/ch03.html#sct.customize-dts]」
- Armadillo-IoT ゲートウェイ G4 :「Device Tree をカスタマイズする [https://manual.atmark-techno.com/armadillo-iot-g4/armadillo-iotg-g4_product_manual_ja/ch03.html#sct.customize-dts]」

オプション引数を指定せずに実行した場合は、デフォルトの dtb ファイルなどを使用します。imx-boot-[VERSION] ディレクトリ下で以下のように build を実行してください。

```
[ATDE ~/imx-boot-[VERSION]]$ ./secureboot.sh build
Logging build outputs to /home/atmark/secureboot_x2/tmp/build.log

::(省略)

Welcome to NXP firmware-imx-8.11.bin

You need to read and accept the EULA before you can continue. ❶

::(省略)

Do you accept the EULA you just read? (y/N) y ❷

::(省略)

Created /home/atmark/secureboot_x2/out/imx-boot_armadillo_x2.signed

Signing linux image...
Created /home/atmark/secureboot_x2/out/Image.signed

Building initrd for mmc (first time is slow)
Signing linux image (mmc)...
Created /home/atmark/secureboot_x2/out/Image.signed-mmc

Building 1_write_srk_install_kernel.swu...
Enter pass phrase for /home/atmark/mkswu/swupdate.key: ❸
Building 2_secureboot_close.swu...
Enter pass phrase for /home/atmark/mkswu/swupdate.key:
Building 3_disk_encryption.swu...
Enter pass phrase for /home/atmark/mkswu/swupdate.key:


Please install SWU images in the following order:
 - /home/atmark/secureboot_x2/swu/1_write_srk_install_kernel.swu ❹
 - /home/atmark/secureboot_x2/swu/2_secureboot_close.swu
 - /home/atmark/secureboot_x2/swu/3_disk_encryption.swu
```

❶ NXP の EULA の承諾を求められる場合は、下矢印キーを押し続けてください。

❷ 下矢印キーが入力された場合は消した後、y を入力してください。

❸ SWU イメージを３つ作成するので、パスワードの入力を３回求められます。

❹ SWU イメージが３つ作成されます。

ビルド後に以下の SWU イメージが作られていることをご確認ください。

```
[ATDE ~/imx-boot-[VERSION]]$ ls ../secureboot_x2/swu/
1_write_srk_install_kernel.swu  2_secureboot_close.swu  3_disk_encryption.swu
```

図 3.6 build コマンドにより作成された SWU イメージ

これらの SWU イメージを Armadillo にインストールすることでセキュアブートの有効化および Armadillo Base OS が保存されている rootfs パーティションの暗号化を実現できます。

secureboot.sh build を実行して生成された鍵（cst/keys）は今後も利用するものです。壊れにくい、セキュアなストレージにコピーしておくことをお勧めします。

鍵の更新は計画性を持って行ってください。たとえば開発時のみ利用する鍵、運用時に利用する鍵を使い分ける。また、鍵は定期的に更新が必要です。以下を参考にしてください。

BlueKrypt Cryptographic Key Length Recommendation

https://www.keylength.com/

参考までに上記の３つの SWU イメージが生成されるまでの流れを以降に示します。

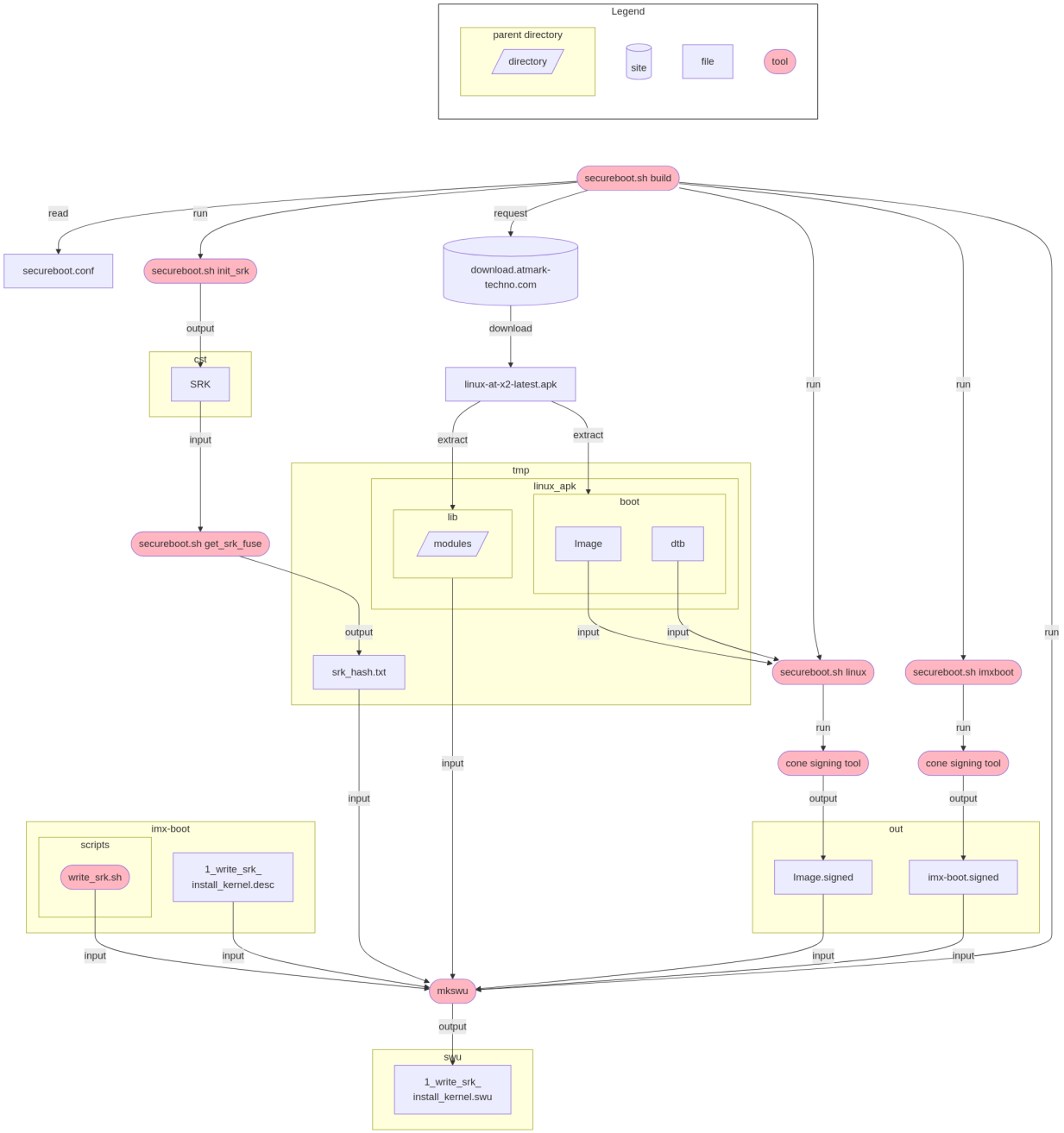

図 3.7 1_write_srk_install_kernel.swu を生成するまでの流れ

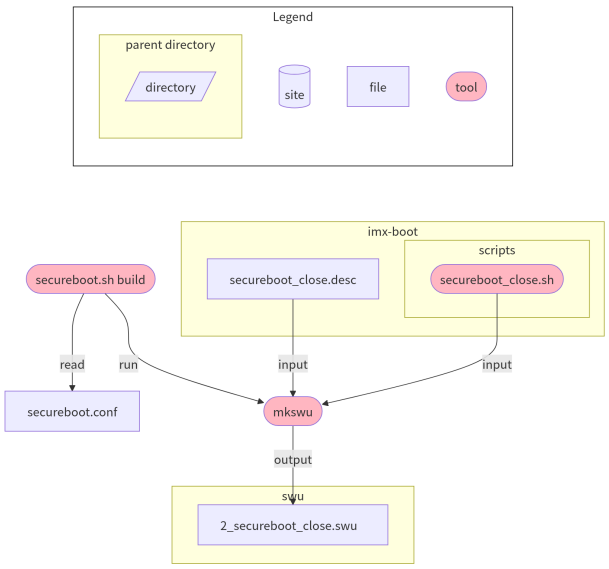

図 3.8 2_secureboot_close.swu を生成するまでの流れ

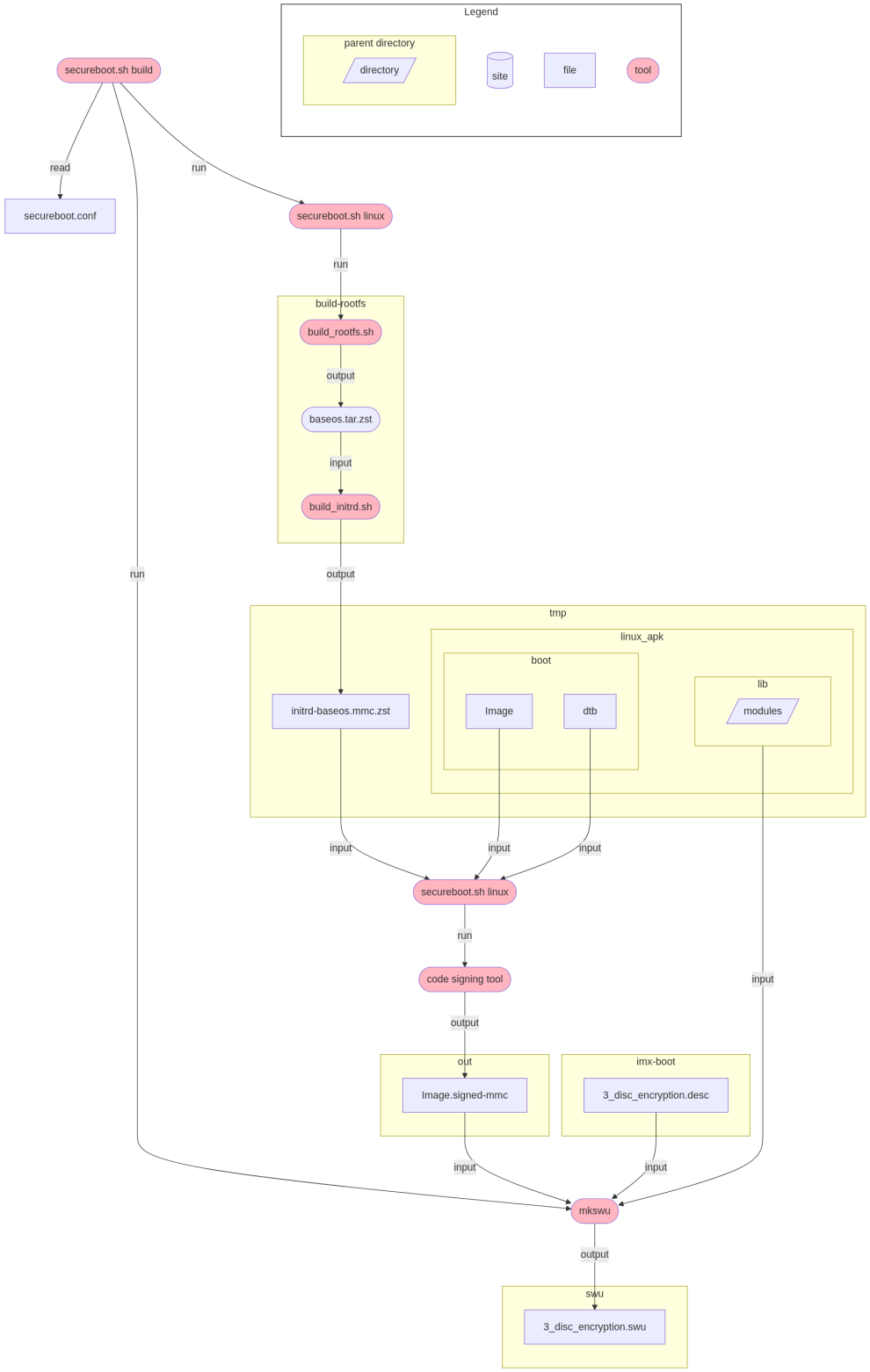

図 3.9 3_disk_encryption.swu を生成するまでの流れ

3.3.2. SWU イメージのインストール

secureboot.sh build によって作成された SWU イメージは以下になります。

表 3.5 生成された SWU イメージ

SWU イメージ名	説明
1_write_srk_install_kernel.swu	署名されたブートローダーおよび Linux カーネルイメージのインストール、SRK のハッシュの書き込みを行います。
2_secureboot_close.swu	セキュアブートの close 処理を行います。
3_disk_encryption.swu	ルートファイルシステム（rootfs）を暗号化します。暗号化に対応した Linux カーネルイメージもインストールします。

生成された SWU イメージを以下の順に Armadillo にインストールしてください。

- 1_write_srk_install_kernel.swu
- 2_secureboot_close.swu
- 3_disk_encryption.swu

SWU イメージのインストール方法については製品マニュアルをご参照ください。

- Armadillo-X2 :「SWU イメージのインストール [https://manual.atmark-techno.com/armadillo-x2/armadillo-x2_product_manual_ja/ch03.html#sct.swupdate-install]」
- Armadillo-IoT ゲートウェイ G4：「SWU イメージのインストール [https://manual.atmark-techno.com/armadillo-iot-g4/armadillo-iotg-g4_product_manual_ja/ch03.html#sct.swupdate-install]」

3.3.2.1. 1_write_srk_install_kernel.swu のインストール

1_write_srk_install_kernel.swu を Armadillo にインストールすることで以下の設定が行われます。

- 署名されたブートローダーおよび Linux カーネルイメージのインストール
- eFuse への SRK ハッシュの書き込み
- ブートローダーにおいて CFG_ENV_FLAGS_LIST_STATIC で指定された環境変数以外の変更の禁止
- ブートローダーのプロンプトの使用の禁止

1_write_srk_install_kernel.swu をインストールしただけでは、まだセキュアブートは有効になりません。

1_write_srk_install_kernel.swu が Armadillo に正常にインストールされた場合の起動ログを以下に示します。

```
:(省略)

Requesting system reboot ❶
[ 3801.975472] imx2-wdt 30280000.watchdog: Device shutdown: Expect reboot!
[ 3801.982451] reboot: Restarting system

:(省略)
```

```
Secure boot disabled ❷

HAB Configuration: 0xf0, HAB State: 0x66
No HAB Events Found! ❸

## Loading kernel from FIT Image at 40480000 ...
   Using 'armadillo' configuration
   Trying 'kernel' kernel subimage
     Description:  linux kernel
     Created:      2025-02-13  10:13:01 UTC
     Type:         Kernel Image
     Compression:  zstd compressed
     Data Start:   0x404800cc
     Data Size:    10353399 Bytes = 9.9 MiB
     Architecture: AArch64
     OS:           Linux
     Load Address: 0x80080000
     Entry Point:  0x80080000
   Verifying Hash Integrity ... OK

:(省略)

Starting kernel ...

:(省略)
```

図 3.10 1_write_srk_install_kernel.swu をインストールした後の起動ログ

❶ Armadillo を再起動します。

❷ この時点ではまだセキュアブートは有効ではありません。

❸ 「No HAB Events Found!」が表示されていることをご確認ください。

SRK のハッシュが書き込まれていることを以下のコマンドで確認できます。

```
[armadillo ~]# device-info -f secureboot
secureboot configured (not enforced) ❶
```

図 3.11 SRK ハッシュが書き込まれていることを確認する

❶ SRK のハッシュは書き込まれていますが、セキュアブートはまだ有効ではないことを示しています。

3.3.2.2. 2_secureboot_close.swu のインストール

2_secureboot_close.swu を Armadillo にインストールすることで、close 処理が行われてセキュアブートが有効になります。SRK ハッシュが書き込まれていない、もしくは /boot/Image が存在する場合はこの SWU イメージはインストール時にエラーになります。

以下、2_secureboot_close.swu が Armadillo に正常にインストールされた場合の起動ログを以下に示します。

```
:(省略)

Secure boot enabled ❶

HAB Configuration: 0xcc, HAB State: 0x99
No HAB Events Found! ❷

## Loading kernel from FIT Image at 40480000 ...
   Using 'armadillo' configuration
   Trying 'kernel' kernel subimage
     Description:  linux kernel
     Created:      2025-02-13  10:13:01 UTC
     Type:         Kernel Image
     Compression:  zstd compressed
     Data Start:   0x404800cc
     Data Size:    10353399 Bytes = 9.9 MiB
     Architecture: AArch64
     OS:           Linux
     Load Address: 0x80080000
     Entry Point:  0x80080000
   Verifying Hash Integrity ... OK

:(省略)

Starting kernel ...

:(省略)
```

図 3.12 2_secureboot_close.swu をインストールした後の起動ログ

❶ セキュアブートが有効であることをご確認ください。

❷ 「No HAB Events Found!」が表示されていることをご確認ください。

以下のコマンドでもセキュアブートが有効であることを確認できます。

```
[armadillo ~]# device-info -f secureboot
secureboot configured
```

図 3.13 セキュアブートが有効であることを確認する

3.3.2.3. 3_disk_encryption.swu のインストール

3_disk_encryption.swu を Armadillo にインストールすることで、Armadillo Base OS が保存されている rootfs パーティションが暗号化されます。

他のパーティション（log やコンテナやアプリケーションが保存されている appfs）は SWU によって暗号化できませんので、そちらも暗号化したい場合は 「4. 量産編」 の手順が完了した後にインストールして有効化してください。

以下、3_disk_encryption.swu が Armadillo に正常にインストールされた場合の確認方法を以下に示します。

```
:(省略)

Starting kernel ...

:(省略)

Welcome to Alpine Linux 3.20
Kernel 5.10.233-0-at on an aarch64 (/dev/ttymxc1)

armadillo login: root ❶
Password:
Welcome to Alpine!

The Alpine Wiki contains a large amount of how-to guides and general
information about administrating Alpine systems.
See <http://wiki.alpinelinux.org/>.


Please note this system is READ-ONLY with a read-write overlayfs,
after updating password make sure to save new password with
# persist_file /etc/shadow

You can change this message by editing /etc/motd.
Last update on Mon Feb 17 11:26:14 JST 2025, updated:
  extra_os.secureboot_init: 2 -> 3
  boot_linux: 1 -> 3
armadillo:~# lsblk ❷
NAME         MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINTS
mmcblk2      179:0    0   9.8G  0 disk
├─mmcblk2p1  179:1    0   300M  0 part
├─mmcblk2p2  179:2    0   300M  0 part
│ └─rootfs_1 252:0    0   299M  0 crypt /live/rootfs ❸

:(省略)
```

図 3.14 3_disk_encryption.swu をインストールした後の確認方法

❶ root にログインします。

❷ lsblk コマンドを実行します。

❸ /live/rootfs に crypt の表記があり、rootfs が暗号化されていることを確認できます。

3.4. セキュアブートが有効になっているかの確認

Linux カーネルが起動することを確認してください。Linux カーネルまで起動しない場合、uboot-imx のコマンドで HAB の状態を確認することができます。

Linux が起動するまでの正常な起動ログの例

```
: (省略)
Booting from mmc ...
```

```
## Checking Image at 40480000 ...
Unknown image format!
53522 bytes read in 22 ms (2.3 MiB/s)

Authenticate image from DDR location 0x40480000...

Secure boot enabled ❶

HAB Configuration: 0xcc, HAB State: 0x99
No HAB Events Found! ❷

## Flattened Device Tree blob at 45000000
   Booting using the fdt blob at 0x45000000
   Using Device Tree in place at 0000000045000000, end 0000000045010111

Starting kernel ...
:（省略）
```

❶　セキュアブートが有効な場合に表示されます

❷　問題がない場合はイベントが表示されません

ブートローダーに問題がある場合の起動ログの例

```
:（省略）
spl: ERROR:  image authentication unsuccessful
### ERROR ### Please RESET the board ###
:（省略）
```

Linux カーネルイメージに問題がある場合の起動ログの例

```
:（省略）
Authenticate image from DDR location 0x40480000...
bad magic magic=0x14 length=0xa1 version=0x0
bad length magic=0x14 length=0xa1 version=0x0
bad version magic=0x14 length=0xa1 version=0x0
Error: Invalid IVT structure

Allowed IVT structure:
IVT HDR      = 0x4X2000D1
IVT ENTRY    = 0xXXXXXXXX
IVT RSV1      = 0x0
IVT DCD      = 0x0
IVT BOOT_DATA = 0xXXXXXXXX
IVT SELF      = 0xXXXXXXXX
IVT CSF      = 0xXXXXXXXX
IVT RSV2      = 0x0
Authenticate Image Fail, Please check ❶
:（省略）
```

❶　認証に失敗しています

u-boot コマンドの hab_status

問題がない場合

```
u-boot=> hab_status

Secure boot enabled

HAB Configuration: 0xcc, HAB State: 0x99
No HAB Events Found!
```

Linux カーネルイメージの署名確認で問題がある場合

```
u-boot=> hab_status

Secure boot disabled

HAB Configuration: 0xf0, HAB State: 0x66

--------- HAB Event 1 -----------------
event data:
        0xdb 0x00 0x14 0x45 0x33 0x0c 0xa0 0x00
        0x00 0x00 0x00 0x00 0x40 0x1f 0xdd 0xc0
        0x00 0x00 0x00 0x20

STS = HAB_FAILURE (0x33)
RSN = HAB_INV_ASSERTION (0x0C)
CTX = HAB_CTX_ASSERT (0xA0)
ENG = HAB_ENG_ANY (0x00)
```

4. 量産編

開発用 Armadillo のセキュアブートの有効化を実施した後、量産用の Armadillo に対してもセキュアブートを有効にする必要があります。

また、量産用 Armadillo では rootfs の他にログ書き込み用、ファームウェア用、アプリケーション用パーティションの暗号化も行います。

ここでは、量産用 Armadillo のセキュアブートを有効化し、ストレージを暗号化するインストールディスクイメージを作成する方法を 2 通り示します。

1. 「4.1. 開発用 Armadillo の環境を複製する方法」

   この方法では、開発用 Armadillo の開発環境を量産用 Armadillo に複製します。

   開発用 Armadillo にある不要なデータを削除する必要はありますが、手順は「4.2. 量産用インストールディスクイメージを 1 から作る方法」に示す方法より簡単です。

2. 「4.2. 量産用インストールディスクイメージを 1 から作る方法」

   この方法では、必要なパッケージおよびソフトウェアのみを抽出し、それらを量産用インストールディスクイメージに組み込みます。

   手順は「4.1. 開発用 Armadillo の環境を複製する方法」よりも複雑になりますが、開発用 Armadillo の環境を複製したくない場合、ユーザーランドが起動せず SWU イメージによる複製が困難な場合にこちらの方法が有効です。

4.1. 開発用 Armadillo の環境を複製する方法

ここでは、セキュアブート有効化済みの開発用 Armadillo に initial_setup.swu がインストールされており、アプリケーションも配置されている前提です。

abos-ctrl make-installer コマンドを使用して、開発環境を複製したインストールディスクイメージを作成する方法を紹介します。

ここで紹介する量産用インストールディスクイメージを量産用 Armadillo にインストールすると、以下のような環境を作成できます。

- セキュアブートが有効化されている
- rootfs 用、ログ書き込み用、ファームウェア用、アプリケーション用パーティションが暗号化されている
- initial_setup.swu 及びアプリケーションが予めインストールされている

4.1.1. インストールディスクイメージ作成用の SWU イメージを生成する

secureboot.sh の make_installer オプションを使用します。以下に示すオプション引数があります。

表 4.1 make_installer の引数

オプション引数	説明
--cn	インストールディスクイメージの改ざん防止のために内部で使用している openssl で使用します。デフォルトでは 「Secure boot installer [ランダムな 10 桁の英数字]」 が使用されます。

以下のコマンドを実行してください。

```
[ATDE ~/imx-boot-[VERSION]]$ ./secureboot.sh make_installer
Logging build outputs to /home/atmark/secureboot_x2/tmp/make_installer.log

Downloading https://armadillo.atmark-techno.com/files/downloads/armadillo-iot-g4/image/baseos-x2-
installer-latest.zip
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  167M  100  167M    0     0  5480k      0  0:00:31  0:00:31 --:--:-- 14.7M

Building initrd for verity (first time is slow)
Signing linux image (verity)...
Created /home/atmark/secureboot_x2/out/Image.signed-verity

Building SWU...
Enter pass phrase for /home/atmark/mkswu/swupdate.key: ❶


Install following SWU image with an USB drive plugged in:
 - /home/atmark/secureboot_x2/swu/secureboot_make_installer.swu ❷
```

図 4.1 secureboot.sh make_installer の実行

❶ SWU イメージを作成するためのパスワードを入力してください。

❷ インストールディスクイメージ作成用の SWU イメージが生成されます。

secureboot_make_installer.swu が存在することをご確認ください。

```
[ATDE ~/imx-boot]$ ls /home/atmark/secureboot_x2/swu/secureboot_make_installer.swu
/home/atmark/secureboot_x2/swu/secureboot_make_installer.swu
```

図 4.2 secureboot.sh make_installer のよって作成された SWU イメージ

参考までに secureboot_make_installer.swu の生成時の流れを示します。

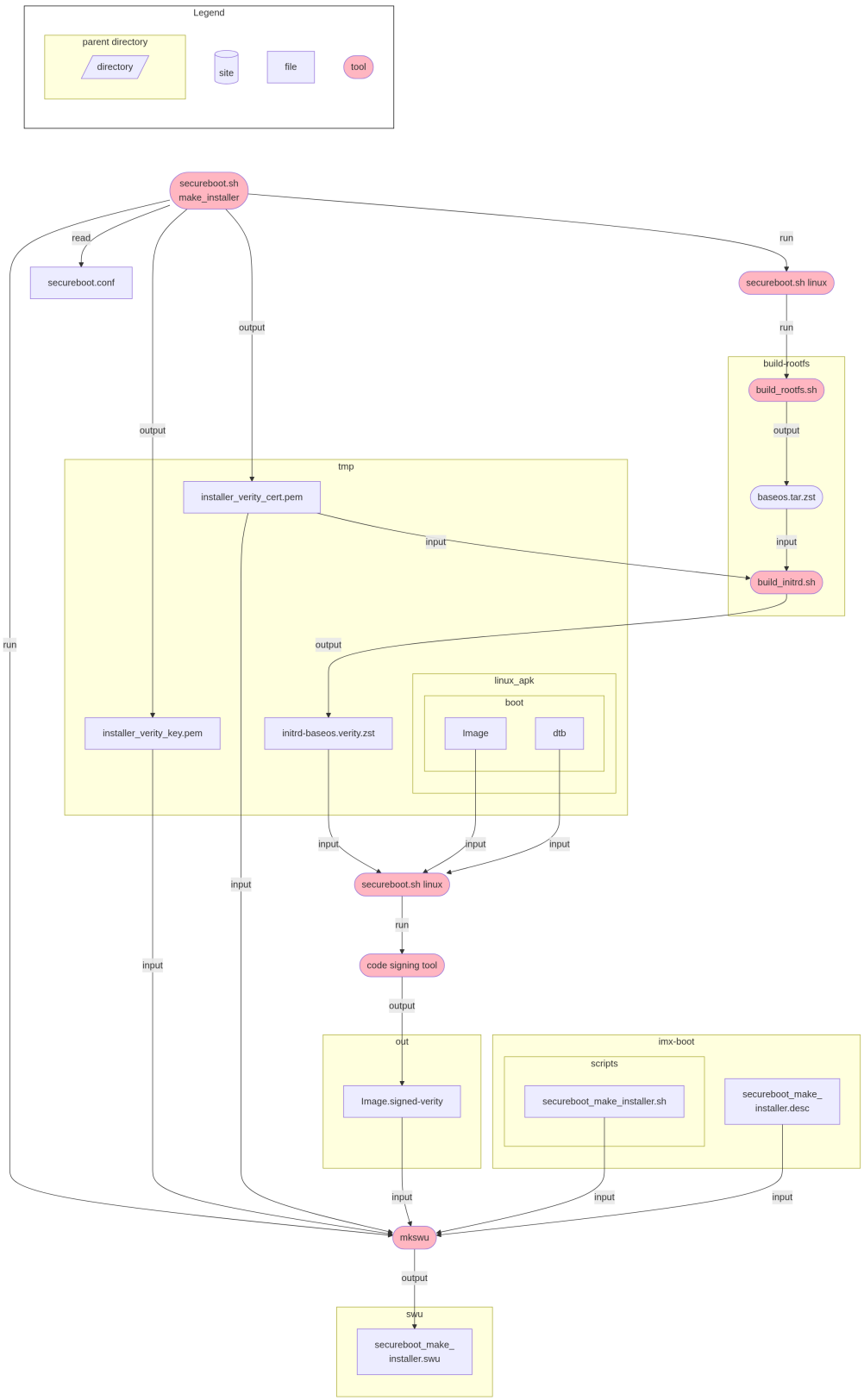

図 4.3 secureboot_make_installer.swu を生成するまでの流れ

4.1.2. Armadillo に USB メモリを挿入

Armadillo に電源を投入し、インストールディスクを保存するための USB メモリを挿入してください。

USB メモリは vfat もしくは ext4 形式でフォーマットし、空き容量が 10GB 以上のものを使用してください。Armadillo への USB メモリのマウントは不要です。

インストールディスクイメージは installer.img という名前で保存します。すでに同名のファイルが存在する場合は上書きされます。

4.1.3. インストールディスクイメージ作成用 SWU を Armadillo にインストールする

ABOS Web を使用して、生成した secureboot_make_installer.swu をインストールしてください。実行時は ABOS Web 上に 「図 4.4. secureboot_make_installer.swu インストール時のログ」 が表示されます。

```
secureboot_make_installer.swu をインストールします。

SWU アップロード完了

SWUpdate v2024.12.0-git20250115-r0

Licensed under GPLv2. See source distribution for detailed copyright notices.

[INFO ] : SWUPDATE running :  [print_registered_handlers] :     no handler registered.
[INFO ] : SWUPDATE running :  [main] : Running on armadillo-900 Revision at1
[INFO ] : SWUPDATE started :  Software Update started !
[INFO ] : SWUPDATE running :  [install_single_image] : Installing pre_script
[INFO ] : SWUPDATE running :  [read_lines_notify] : No base os update: copying current os over
[INFO ] : SWUPDATE running :  [read_lines_notify] : Waiting for btrfs to flush deleted subvolumes
[INFO ] : SWUPDATE running :  [install_single_image] : Installing Copy installer to USB device
[INFO ] : SWUPDATE running :  [install_single_image] : Installing Stop containers
[INFO ] : SWUPDATE running :  [install_single_image] : Installing Make installer image
[INFO ] : SWUPDATE running :  [read_lines_notify] : Secure boot enabled setting for production
Armadillo is already set
[INFO ] : SWUPDATE running :  [read_lines_notify] : ./
[INFO ] : SWUPDATE running :  [read_lines_notify] : ./installer_verity_cert.pem
[INFO ] : SWUPDATE running :  [read_lines_notify] : ./installer_verity_key.pem
[INFO ] : SWUPDATE running :  [read_lines_notify] : ./sd_copy/
[INFO ] : SWUPDATE running :  [read_lines_notify] : ./sd_copy/boot/
[INFO ] : SWUPDATE running :  [read_lines_notify] : ./sd_copy/boot/Image
[INFO ] : SWUPDATE running :  [read_lines_notify] : /target/mnt/installer.img-in-progress (425MB)
was bigger than 338MB and was not truncated.
[INFO ] : SWUPDATE running :  [read_lines_notify] : Checking if /target/mnt/installer.img-in-
progress can be used safely...
[INFO ] : SWUPDATE running :  [read_lines_notify] : Using installer image on image file.
[INFO ] : SWUPDATE running :  [read_lines_notify] : Growing /target/mnt/installer.img-in-progress
to fit verity partition
[INFO ] : SWUPDATE running :  [read_lines_notify] : Growing installer main partition
[INFO ] : SWUPDATE running :  [read_lines_notify] : Resize device id 1 (/dev/loop0p1) from 394.00MiB
to max
```

```
[ERROR] : SWUPDATE failed [0] ERROR : WARNING: the new size 0 (0.00B) is < 256MiB, this may be rejected by kernel
[INFO ] : SWUPDATE running :  [read_lines_notify] : Setting console to console=ttyLP0,115200 in installer
[INFO ] : SWUPDATE running :  [read_lines_notify] : Environment OK, copy 1
[INFO ] : SWUPDATE running :  [read_lines_notify] : Copying armadillo's root password to installer
[INFO ] : SWUPDATE running :  [read_lines_notify] : Copying boot image
[INFO ] : SWUPDATE running :  [read_lines_notify] : Installer will enable secure boot
[INFO ] : SWUPDATE running :  [read_lines_notify] : Copying rootfs
[INFO ] : SWUPDATE running :  [read_lines_notify] : Copying /opt/firmware filesystem
[INFO ] : SWUPDATE running :  [read_lines_notify] : Copying appfs
[INFO ] : SWUPDATE running :  [read_lines_notify] : At subvol app/snapshots/volumes
[INFO ] : SWUPDATE running :  [read_lines_notify] : At subvol app/snapshots/boot_volumes
[INFO ] : SWUPDATE running :  [read_lines_notify] : At subvol app/snapshots/boot_containers_storage
[INFO ] : SWUPDATE running :  [read_lines_notify] : Trying to shrink the installer partition...
[INFO ] : SWUPDATE running :  [read_lines_notify] : Shrinking the installer partition...
[INFO ] : SWUPDATE running :  [read_lines_notify] : Cleaning up and syncing changes to disk...
[INFO ] : SWUPDATE running :  [read_lines_notify] : Computing verity hashes...
[INFO ] : SWUPDATE running :  [read_lines_notify] : Installer updated successfully!
[INFO ] : SWUPDATE running :  [read_lines_notify] : -rw-------    1 root     root      409.1M Apr 2 16:49 /target/mnt/installer.img
[INFO ] : SWUPDATE running :  [read_lines_notify] : Installer successfully created!
[INFO ] : SWUPDATE running :  [install_single_image] : Installing post_script
[INFO ] : SWUPDATE running :  [read_lines_notify] : Removing unused containers
[INFO ] : SWUPDATE running :  Installation in progress
[INFO ] : SWUPDATE successful ! SWUPDATE successful !
swupdate exited
```

図 4.4 secureboot_make_installer.swu インストール時のログ

完了後、USB メモリを抜いてください。もし、エラーが出た場合は Armadillo の電源を再投入してやり直してみてください。

無事に生成が完了した場合、USB メモリ上に installer.img が保存されています。この installer.img を microSD に書き込むことでインストールディスクを作成することができます。

4.1.3.1. ATDE への量産用インストールディスクイメージのコピー

USB メモリにある installer.img を ATDE 上にコピーします。

PC に USB メモリを挿入してください。VirtualBox の左上のペインの [デバイス] → [USB] から、USB メモリを ATDE にマウントします。ここではホームディレクトリにコピーします。

```
[ATDE ~]$ sudo cp /media/atmark/<USB メモリをマウントしたディレクトリ>/installer.img ~/
```

図 4.5 ATDE に量産用インストールディスクイメージをコピーする

量産用インストールディスクイメージである installer.img を microSD に書き込む方法は「4.3. 量産用インストールディスクイメージを microSD に書き込む」をご参照ください。

4.2. 量産用インストールディスクイメージを 1 から作る方法

開発用 Armadillo の開発環境を複製せずに量産用インストールディスクイメージを作成する方法を紹介します。

ここで紹介する量産用インストールディスクイメージを量産用 Armadillo にインストールすると、以下のような環境を作成できます。

- セキュアブートが有効化されている
- rootfs 用、ログ書き込み用、ファームウェア用、アプリケーション用パーティションが暗号化されている
- initial_setup.swu 及びアプリケーション用 SWU イメージ（例として release.swu ）が予めインストールされている

4.2.1. 量産用インストールディスクイメージの作成

以下では secureboot.sh setup 及び secureboot.sh build を実行済みである前提です。

量産用インストールディスクイメージの作成には build-rootfs を使用します。build-rootfs に含まれるファイルの説明は製品マニュアルをご参照ください。

- Armadillo-X2：「Alpine Linux ルートファイルシステムをビルドする [https://manual.atmark-techno.com/armadillo-x2/armadillo-x2_product_manual_ja/ch06.html#sct.build-alpine-rootfs]」
- Armadillo-IoT ゲートウェイ G4：「Alpine Linux ルートファイルシステムをビルドする SWU インストール [https://manual.atmark-techno.com/armadillo-iot-g4/armadillo-iotg-g4_product_manual_ja/ch06.html#sct.build-alpine-rootfs]」

Linux カーネルの modules と dtb の入れ替え

はじめに、build-rootfs/ax2/packages の linux-at-x2 をコメントアウトします。

```
[ATDE ~/secureboot_x2]$ code build-rootfs/ax2/packages
# linux-at is not available on all arches

# linux-at-x2@atmark ❶

# wifi firwmares
wireless-regdb
linux-firmware-nxpwifi-9098

# encryped boot
cryptsetup
caam-decrypt
```

❶ コメントアウトします。

最新の Linux カーネルイメージと modules を取得するために、./secureboot.sh linux を実行してください。

```
[ATDE ~/secureboot_x2]$ ./secureboot.sh linux --initrd-type none --linux-update

Logging build outputs to /home/atmark/secureboot_x2/tmp/linux.log

Downloading https://download.atmark-techno.com/armadillo-iot-g4/baseos/linux-at-x2-latest.apk ❶
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
```

```
                                 Dload  Upload   Total   Spent    Left  Speed
100 16.1M  100 16.1M    0     0  3971k      0  0:00:04  0:00:04 --:--:-- 3971k

Signing linux image...
Created /home/atmark/secureboot_x2/out/Image.signed ❷
```

❶ 最新の linux apk パッケージがダウンロードされて tmp/linux_apk が作成されます。

❷ Image.signed が作成されますが、こちらは使用しませんので無視してください。

次に、lib/modules と boot/armadillo_iotg_g4.dtb を build-rootfs/ax2/resources にコピーします。

```
[ATDE ~/secureboot_x2]$ cp -r tmp/linux_apk/lib build-rootfs/ax2/resources/
[ATDE ~/secureboot_x2]$ cp tmp/linux_apk/boot/armadillo_iotg_g4.dtb ¥
    build-rootfs/ax2/resources/boot/
```

SWU イメージのインストール

initial_setup.swu や開発したアプリケーションを含む release.swu など、必要な SWU イメージは build-rootfs/ax2/image_installer/installer_swus に配置します。

SWU イメージはファイル名がアルファベット順にインストールされます。インストールされる順番を保証するため、1_initial_setup.swu、2_release.swu にように名前を変更します。

ここでは、initial_setup.swu は /home/atmark/mkswu に、release.swu は /home/atmark/my_project にあるとします。

```
[ATDE ~/secureboot_x2]$ mkdir build-rootfs/ax2/image_installer/installer_swus
[ATDE ~/secureboot_x2]$ cp /home/atmark/mkswu/initial_setup.swu ¥
    build-rootfs/ax2/image_installer/installer_swus/1_initial_setup.swu
[ATDE ~/secureboot_x2]$ cp /home/atmark/my_project/release.swu ¥
    build-rootfs/ax2/image_installer/installer_swus/2_release.swu
```

rootfs のビルド

最新の rootfs を作成するために、以下のコマンドを実行してください。

```
[ATDE ~/secureboot_x2]$ cd build-rootfs
[ATDE ~/secureboot_x2/build-rootfs]$ ./build_rootfs.sh -b ax2

:(省略)
Successfully built /home/atmark/secureboot_x2/build-rootfs/baseos-x2-[VERSION].[DATE].tar.zst
```

インストールディスクイメージ改ざん防止のためのキーペアの生成

作成するインストールディスクイメージの改ざんの防止のために openssl で作成したキーペアを生成します。

```
[ATDE ~/secureboot_x2/build-rootfs]$ cd ..
[ATDE ~/secureboot_x2]$ openssl req -x509 -nodes -days 3650 -newkey ec ¥
```

```
    -pkeyopt ec_paramgen_curve:prime256v1 ¥
    -keyout verity_key.pem ¥
    -out verity_cert.pem ¥
    -subj /CN="secureboot installer"

Generating an EC private key
writing new private key to 'verity_key.pem'
```

署名済み Linux カーネルイメージの作成

暗号化は SoC 内の鍵を使用します。正規のボード以外で eMMC 内のデータを復号して読み取ることが出来ないようにするために、暗号化された eMMC 以外起動することができない Linux カーネルイメージに署名して Image.signed-mmc を作成します。

```
[ATDE ~/secureboot_x2]$ ./secureboot.sh linux --initrd-type mmc

Logging build outputs to /home/atmark/secureboot_x2/tmp/linux.log

Building initrd for mmc (first time is slow)
Signing linux image (mmc)...
Created /home/atmark/secureboot_x2/out/Image.signed-mmc
```

次に、インストールディスクイメージの改ざん防止用の署名済み Linux カーネルイメージである Image.signed-verity を作成します。

```
[ATDE ~/secureboot_x2]$ ./secureboot.sh linux ¥
    --initrd-type verity ¥
    --verity /home/atmark/secureboot_x2/verity_cert.pem

Logging build outputs to /home/atmark/secureboot_x2/tmp/linux.log

Building initrd for verity (first time is slow)
Signing linux image (verity)...
Created /home/atmark/secureboot_x2/out/Image.signed-verity
```

改ざんを防止するために、microSD に Image.signed-verity を書き込み、この Linux カーネルイメージを用いて SD ブートを実行します。

そのために、Image.signed-verity を build-rootfs/ax2/image_installer/boot/Image としてコピーします。

```
[ATDE ~/secureboot_x2]$ mkdir build-rootfs/ax2/image_installer/boot
[ATDE ~/secureboot_x2]$ cp out/Image.signed-verity ¥
    build-rootfs/ax2/image_installer/boot/Image
```

署名済みブートローダーの作成

署名済みブートローダーである imx-boot_armadillo_ax2.signed を作成します。

```
[ATDE ~/secureboot_x2]$ ./secureboot.sh imxboot
```

```
Logging build outputs to /home/atmark/secureboot_x2/tmp/imxboot.log

Building imx-boot (boot loader)...
Created /home/atmark/secureboot_x2/out/imx-boot_armadillo_x2.signed
```

インストールディスクイメージの作成

インストールディスクイメージを作成する前に以下のコマンドを実行して、cryptsetup-bin をインストールします。cryptsetup-bin はストレージの暗号化に使用します。

```
[ATDE ~/secureboot_x2]$ sudo apt update && sudo apt upgrade
[ATDE ~/secureboot_x2]$ sudo apt install cryptsetup-bin
```

eMMC に書き込む rootfs を含むイメージを作成するために以下のコマンドを実行してください。

```
[ATDE ~/secureboot_x2]$ cd build-rootfs
[ATDE ~/secureboot_x2/build-rootfs]$ ./build_image.sh -b ax2

use default(outdir=/home/atmark/secureboot_x2/build-rootfs)
use default(output=baseos-x2-[VERSION].[DATE].img)
:(省略)
Successfully built /home/atmark/secureboot_x2/build-rootfs/baseos-x2-[VERSION].[DATE].img
```

次に、インストールディスクイメージを作成します。

作成するインストールディスクイメージを microSD に書き込み、その microSD で SD ブートを実行します。

SD ブートを実行すると、改ざん防止用の署名済み Linux カーネルイメージが起動し、先ほど作成したイメージと署名済みのブートローダー、署名済みの暗号化に対応した Linux カーネルイメージを eMMC に書き込みます。

以下のコマンドでは、eMMC に書き込まれるそれらのファイルを引数で指定して、インストールディスクイメージを作成します。

```
[ATDE ~/secureboot_x2/build-rootfs]$ PATH=/sbin:$PATH ./build_image.sh -b ax2 ¥
    --srk $(cat /home/atmark/secureboot_x2/tmp/srk_hash.txt) ¥
    --encrypt all ¥
    --boot /home/atmark/secureboot_x2/out/imx-boot_armadillo_x2.signed ¥
    --boot-linux /home/atmark/secureboot_x2/out/Image.signed-mmc ¥
    --verity /home/atmark/secureboot_x2/verity_cert.pem /home/atmark/secureboot_x2/verity_key.pem ¥
    --installer baseos-x2-[VERSION].[DATE].img

use default(outdir=/home/atmark/secureboot_x2/build-rootfs)
use default(output=baseos-[VERSION].[DATE]-installer.img)
:(省略)
Successfully built /home/atmark/secureboot_x2/build-rootfs/baseos-x2-[VERSION].[DATE]-
installer.img
```

baseos-x2-[VERSION].[DATE]-installer.img が microSD に書き込む量産用インストールディスクイメージとなります。

必要であれば、量産用インストールディスクイメージ名を管理しやすい名前に変更してください。ここでは、installer.img という名前でホームディレクトリにコピーします。

```
[ATDE ~/secureboot_x2/build-rootfs]$ cp baseos-x2-[VERSION].[DATE]-installer.img ~/installer.img
```

量産用インストールディスクイメージである installer.img を microSD に書き込む方法は「4.3. 量産用インストールディスクイメージを microSD に書き込む」をご参照ください。

4.3. 量産用インストールディスクイメージを microSD に書き込む

「4.1. 開発用 Armadillo の環境を複製する方法」または「4.2. 量産用インストールディスクイメージを 1 から作る方法」で作成したインストールディスクイメージ installer.img を microSD に書き込み、開発に使用した Armadillo 以外の個体にインストールします。

インストール先の Armadillo の eMMC 内のデータは上書きされて消えるため、以下のディレクトリにある必要なデータは予めバックアップを取っておいてください。

- /var/app
- /var/log
- container image

以下のコマンドを実行して、installer.img を microSD に書き込んでください。

installer.img を microSD に書き込む手順は初期化インストールディスクの作成方法と同じですので、製品マニュアルの以下の節を参考にしてください。

- Armadillo-X2：「初期化インストールディスクの作成 [https://manual.atmark-techno.com/armadillo-x2/armadillo-x2_product_manual_ja/ch03.html#sct.make-install-disk-tutorial]」
- Armadillo-IoT ゲートウェイ G4：「初期化インストールディスクの作成 [https://manual.atmark-techno.com/armadillo-iot-g4/armadillo-iotg-g4_product_manual_ja/ch03.html#sct.make-install-disk-tutorial]」

上記で作成したインストールディスクを用いて量産用 Armadillo に量産用イメージをインストールする手順は製品マニュアルの以下の節を参考にしてください。

- Armadillo-X2：「インストールディスクを使用する [https://manual.atmark-techno.com/armadillo-x2/armadillo-x2_product_manual_ja/ch03.html#sct.use-install-disk-tutorial]」
- Armadillo-IoT ゲートウェイ G4：「インストールディスクを使用する [https://manual.atmark-techno.com/armadillo-iot-g4/armadillo-iotg-g4_product_manual_ja/ch03.html#sct.use-install-disk-tutorial]」

実際の量産製造においては、自社でイメージ書き込みを実施する他に、アットマークテクノが提供する「 BTO サービス」 の「 ROM イメージ書込み」を利用する方法もあります。詳細については、下記ページの BTO サービスマニュアルの一覧から各製品の BTO サービスマニュアルを参照してください。

・ BTO サービスマニュアル [https://armadillo.atmark-techno.com/bto/manual]

4.4. 量産用 Armadillo の動作確認

量産用 Armadillo を起動します。起動時のログは 「3.4. セキュアブートが有効になっているかの確認」をご参照ください。

Armadillo にログイン後、以下のコマンドを実行すると、rootfs 用、ログ書き込み用、ファームウェア用、アプリケーション用パーティションが暗号化されていることを確認できます。

```
[armadillo ~]# lsblk
NAME          MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINTS
mmcblk2       179:0    0   9.8G  0 disk
├─mmcblk2p1   179:1    0   300M  0 part
│ └─rootfs_0  252:0    0   299M  0 crypt /live/rootfs ❶
├─mmcblk2p2   179:2    0   300M  0 part
├─mmcblk2p3   179:3    0    50M  0 part
│ └─mmcblk2p3 252:1    0    49M  0 crypt /var/log ❷
├─mmcblk2p4   179:4    0   200M  0 part
│ └─mmcblk2p4 252:2    0   199M  0 crypt /opt/firmware ❸
└─mmcblk2p5   179:5    0     9G  0 part
  └─mmcblk2p5 252:3    0     9G  0 crypt /var/tmp ❹
                                         /var/app/volumes
                                         /var/app/rollback/volumes
                                         /var/lib/containers/storage_readonly
:(省略)
```

図 4.6 ストレージが暗号化されていることを確認

❶ Type が crypt になっているため、rootfs 用パーティションが暗号化されています。

❷ Type が crypt になっているため、書き込みログ用パーティションが暗号化されています。

❸ Type が crypt になっているため、ファームウェア用パーティションが暗号化されています。

❹ Type が crypt になっているため、アプリケーション用パーティションが暗号化されています。

5. 運用編

コンテナ内のアプリケーションのアップデートはセットアップとは関係なく、とくに特別な対応なしで通常どおりの方法でアップデートが可能です。製品マニュアルを参考にしてください。

- Armadillo-X2 :「SWU インストール [https://manual.atmark-techno.com/armadillo-x2/armadillo-x2_product_manual_ja/ch06.html#sct.install-swu-with-abos-web]」
- Armadillo-IoT ゲートウェイ G4 :「SWU インストール [https://manual.atmark-techno.com/armadillo-iot-g4/armadillo-iotg-g4_product_manual_ja/ch06.html#sct.install-swu-with-abos-web]」

署名済みブートローダーや Linux カーネルを更新する場合は以下の手順に従ってください。

5.1. 署名済みブートローダーの更新方法

以下の URL における「ブートローダー ソース（u-boot 等）」から最新のブートローダーを取得してください。

- Armadillo-X2 :「ブートローダー [https://armadillo.atmark-techno.com/resources/software/armadillo-x2/boot-loader]」
- Armadillo-IoT ゲートウェイ G4 :「ブートローダー [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-g4/boot-loader]」

ブートローダーのソースコードのアーカイブを展開します。以降、このアーカイブをホームディレクトリに展開した想定で説明します。

```
[ATDE ~]$ tar xaf imx-boot-[VERSION].tar.gz ❶
```

図 5.1 最新のブートローダーのソースコードを展開する

❶ アーカイブを展開します。[VERSION] はバージョンによって変化します。

secureboot_x2/imx-boot のシンボリックリンク元を最新のものにつけかえます。

```
[ATDE ~]$ rm -rf secureboot_x2/imx-boot ❶
[ATDE ~]$ ln -s /home/atmark/imx-boot-[VERSION] /home/atmark/secureboot_x2/imx-boot ❷
```

図 5.2 シンボリックリンク元を最新のブートローダーに変更する

❶ シンボリックリンクを削除します。

❷ imx-boot-[VERSION] をシンボリックリンク元として secureboot_x2/imx-boot のシンボリックリンクを作成します。

最新のブートローダーに署名を行います。

```
[ATDE ~]$ cd imx-boot-[VERSION]
[ATDE ~/imx-boot-[VERSION]]$ ./secureboot.sh imxboot
Logging build outputs to /home/atmark/secureboot_x2/tmp/imxboot.log

Building imx-boot (boot loader)...
Created /home/atmark/secureboot_x2/out/imx-boot_armadillo_x2.signed ❶
```

図 5.3 最新のブートローダーに署名する

❶ 最新のブートローダー対して署名したファイルが作成されます。

最新の署名済みブートローダーをインストールするための SWU イメージを作成します。

```
[ATDE ~/imx-boot-[VERSION]]$ cd ~/mkswu ❶
[ATDE ~/mkswu]$ cp /usr/share/mkswu/examples/boot.desc . ❷
[ATDE ~/mkswu]$ cat boot.desc
swdesc_boot "/home/atmark/secureboot_x2/out/imx-boot_armadillo_x2.signed" ❸
[ATDE ~/mkswu]$ mkswu boot.desc
Enter pass phrase for /home/atmark/mkswu/swupdate.key: ❹
boot.swu を作成しました。❺
[ATDE ~/mkswu]$ mkswu --show boot.swu ❻
# boot.swu

# Built with mkswu [mkswu のバージョン]
# signed by "atmark"

swdesc_boot  --version boot [VERSION] /home/atmark/secureboot_x2/out/imx-boot_armadillo_x2.signed
```

図 5.4 最新の署名済みブートローダーを SWU イメージに組み込む

❶ mkswu ディレクトリに移動します。

❷ 使用する desc ファイル（boot.desc）を mkswu ディレクトリにコピーします。

❸ swdesc_boot の引数を最新の署名済みブートローダーのファイルパスに置き換えてください。

❹ SWU イメージを作成するためのパスワードの入力を求められます。

❺ boot.swu がカレントディレクトリに作成されます。

❻ boot.swu の中身を表示します。[VERSION] が最新であることをご確認ください。

作成した boot.swu を Armadillo にインストールすることでブートローダーをアップデートできます。

5.2. 署名済み Linux カーネルの更新方法

署名済み Linux カーネルを更新するために、secureboot.sh linux コマンドを使用します。

--linux-update 引数を指定することで最新の Linux カーネルイメージをダウンロードできます。

at-dtweb でカスタマイズした dtbo ファイルを Linux カーネルイメージに組み込みたい場合は、secureboot.sh linux コマンド実行時に --dtbo でその dtbo ファイルのパスを指定してください。

ストレージの暗号化を行っている場合は --initrd-type mmc を指定してください。

```
[ATDE ~]$ cd imx-boot-[VERSION]
[ATDE ~/imx-boot-[VERSION]]$ ./secureboot.sh linux --linux-update --initrd-type mmc
Logging build outputs to /home/atmark/secureboot_x2/tmp/linux.log

Downloading https://download.atmark-techno.com/armadillo-iot-g4/baseos/linux-at-x2-latest.apk ❶
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 16.1M  100 16.1M    0     0  6582k      0  0:00:02  0:00:02 --:--:-- 6579k

Building initrd for mmc (first time is slow)
Signing linux image (mmc)...
Created /home/atmark/secureboot_x2/out/Image.signed-mmc ❷
```

図 5.5 最新の Linux カーネルイメージに署名する

❶ 最新の Linux カーネルの apk の apk パッケージを取得します。

❷ 署名済みの暗号化対応の Linux カーネルイメージが作成されます。

SWU イメージを作成するための desc ファイルを作成します。

```
[ATDE ~/imx-boot-[VERSION]]$ cd ~/mkswu ❶
[ATDE ~/mkswu]$ cp /usr/share/mkswu/examples/encrypted_rootfs_linux_update.desc . ❷
[ATDE ~/mkswu]$ cat encrypted_rootfs_linux_update.desc ❸
# This example is intended for users with ENCRYPTED_ROOTFS.
# Ignore it if not using encryped rootfs.

# version must be updated everytime like normal updates
swdesc_option version=[LATEST_VERSION] ❹

# Image.signed is an a linux image with initrd built using
# imx-boot/secureboot.sh linux
swdesc_boot_linux "/home/atmark/secureboot_x2/out/Image.signed-mmc" ❺
```

図 5.6 最新の署名済み Linux カーネルイメージを SWU イメージに組み込む desc ファイルを作成する

❶ mkswu ディレクトリに移動します。

❷ 作成した署名済み Linux カーネルイメージを SWU イメージに組み込むための desc ファイル（encrypted_rootfs_linux_update.desc）を mkswu ディレクトリにコピーします。

❸ encrypted_rootfs_linux_update.desc の中身を表示します。

❹ [LATEST_VERSION] は更新する際の最新のバージョンに書き換えてください。

❺ 作成した署名済み Linux カーネルイメージのパスを swdesc_boot_linux の引数に指定してください。

SWU イメージを作成します。

```
[ATDE ~/mkswu]$ mkswu encrypted_rootfs_linux_update.desc
Enter pass phrase for /home/atmark/mkswu/swupdate.key: ❶
```

```
encrypted_rootfs_linux_update.swu を作成しました。❷
[ATDE ~/mkswu]$ mkswu --show encrypted_rootfs_linux_update.swu ❸
# encrypted_rootfs_linux_update.swu

# Built with mkswu [mkswu のバージョン]
# signed by "atmark"

swdesc_boot_linux  --version boot_linux [LATEST_VERSION] /home/atmark/secureboot_x2/out/
Image.signed-mmc
```

図 5.7 最新の署名済み Linux カーネルイメージが組み込まれた SWU イメージを作成する

❶ SWU イメージを作成するためのパスワードの入力を求められます。

❷ encrypted_rootfs_linux_update.swu がカレントディレクトリに作成されます。

❸ encrypted_rootfs_linux_update.swu の中身を表示します。[LATEST_VERSION] が最新であることをご確認ください。

作成した encrypted_rootfs_linux_update.swu を Armadillo にインストールすることで Linux カーネルイメージをアップデートできます。

5.3. SRK の無効化と切り替え

何らかのインシデント対応による鍵更新、また、鍵の定期更新などが必要な場合、その時点で利用している鍵を無効化して、別の鍵に切り替えることが可能です。

5.3.1. SRK の無効化 (revocation)

ここでは SRK1 (index 0) から SRK2 (index 1) に変更する例を説明します。

1. secureboot.conf の revocation のロックを解除する設定を有効にします

   デフォルトでは eFuse の revoke レジスタはロックされているので書き込みできません。ロックは HAB の設定で解除することができます。常にロックを解除すると攻撃者に悪用される可能性があるので通常はロックされるべきです。

   secureboot_x2/secureboot.conf を開いて、CST_UNLOCK_SRK=y のコメントを外してください。secureboot.conf の詳細については 「3.2.7. secureboot.conf の設定方法」 を参照してください。

   ```
   [ATDE ~]$ vi secureboot_x2/secureboot.conf
   ```

   図 5.8 secureboot.conf の CST_UNLOCK_SRK を編集

2. 署名済みイメージを書き込む

   環境に合わせて、署名済みか、暗号化＋署名済みのイメージを作成して、イメージを書き込んでください。

3. 再起動

4. Unlock を確認する

再起動時の uboot-imx のプロンプトを立ち上げてレジスタ値を確認します。以下のコマンドを実行してください。bit1 (SRK_REVOKE_LOCK) が 0 ならば Unlock 状態です。

```
u-boot=> md 0x30350050 1
30350050: 00007dbc ❶
```

図 5.9 Unlock 状態であることを確認する

❶ 7dbc の bit 1 が 0 なので unlock 状態

5. SRK を無効化する

ビットフィールドはビットは 0 はじまりで、鍵の番号は 1 はじまり (1,2,3,4) になります。bit0 が SRK1、bit1 が SRK2、bit2 が SRK3、bit3 が SRK4 です。

以下のコマンドはあくまで例なので、そのまま実行しないで下さい。

SRK1 を無効化する場合は以下のコマンドを実行してください。最終引数が無効化する鍵の設定値です。

```
u-boot=> fuse prog 9 3 1
```

図 5.10 SRK1 を無効化する

5.3.2. SRK の切り替え

ここでは SRK1 (index 0) から SRK2 (index 1) に変更する例を説明します。

1. SRK の変更

secureboot.conf の CST_SOURCE_INDEX を 0 から 1 に変更してください。secureboot.conf の詳細については 「3.2.7. secureboot.conf の設定方法」 を参照してください。

```
[ATDE ~]$ vi imx-boot-[VERSION]/secureboot.conf
```

図 5.11 secureboot.conf の CST_SOURCE_INDEX を編集

2. 再署名する

環境に合わせて、署名済みか、暗号化＋署名済みのイメージを作成して、イメージを書き込んでください。

セキュアブートガイド

改訂履歴

バージョン	年月日	改訂内容
1.0.0	2025/06/25	・初版発行

セキュアブートガイド
Version 1.0.0
2025/06/25