4D-03

セキュリティ技術の事業への貢献における課題解決の取り組み

高務　健二†　梅崎　一也†　宮崎　剛†
富士電機株式会社†

1 はじめに

IoT の進展に伴い，IoT 機器が乗っ取られるなどのセキュリティインシデントの事例が増えてきており，組込機器のセキュリティ対策の重要性が高まっている[1]．IoT 機器は攻撃者にとって扱い易い大量の踏み台（乗っ取り対象）であり，インターネットに直接接続するため，様々な脅威に晒される．その対策の 1 つに，不正なソフトウェアの起動・書込みを防止するセキュアブート／アップデート技術がある．

近年我々は，組込機器向けのセキュリティ基盤技術として，各マイコンベンダから提供され始めた“Root of Trust”に関するセキュリティ機能を搭載したマイコンについて調査しており，今回，セキュアブート／アップデート技術を製品に適用できるようにするための製品化開発を行った．開発においては，製品仕様上の制約の観点で，提供された状態のままでは適用できないため，制約を満たすよう工夫する必要があった．

組込機器では，部品単価がコスト競争に繋がるため，搭載する機能を少ない ROM 使用量で実装できることが求められ，さらに，製品仕様として処理時間等各種許容値が制約付けられる．そのため，製品化開発においては，ROM 使用量と演算性能の両立が課題だった．

本稿では，起動時間短縮やセキュリティ対策機能自体の ROM 使用量削減など，製品適用における課題解決について述べる．

2 課題解決の取り組み

2.1 課題設定

マイコンベンダからは，セキュリティ機能を持った汎用マイコンとセットで，セキュアブート／アップデート機能のファームウェアが提供されている．これを製品へ適用するにあたり，他の製品へも展開しやすいプラットフォームとするため，図 1に示すように各製品の既存のブートアプリをそのまま活用する構成とした．

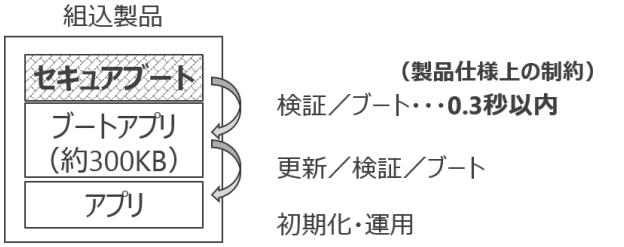

図 1　製品適用でのセキュアブートの位置付け

まず，ワンチップマイコンの内蔵 ROM の空き容量が製品展開においてアプリ開発の制約となるため，セキュアブートの ROM 使用量はマイコンベンダ提供時のものを上限として可能な限り削減する必要があった．

また，従来の製品の立ち上がり時間に対して追加処理となるセキュアブートは，300KB のブートアプリを 0.3 秒以内にブートできることが条件だったが，提供時の性能は，ROM 使用量・演算性能の選択肢として用意されている 2 つのブート方式とも 0.8 秒以上要するため，性能を改善する必要があった．

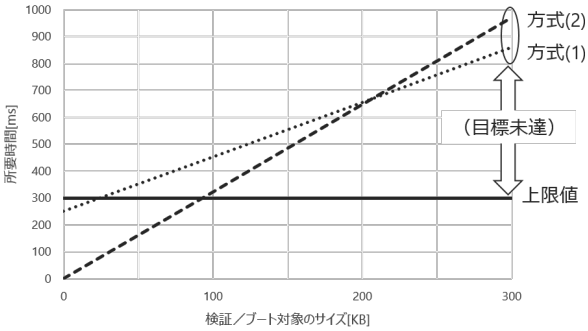

図 2　提供されたセキュアブートの性能

2.2 ROM 使用量削減の検討

ブートアプリ自体の更新はしないため，実装するセキュアブートの機能は，“Root of Trust”と直結する CPU 依存処理を吸収し，ブートアプリをセキュアにブートするものだけに限定し，ダウンロードやアップデートなどの機能やそれらから呼び出される復号処理などを無効化することとした．無効化においては，品質確保の観点から，マイコンベンダ提供のファームウェアにできるだけ手を加えないよう，無効化

Efforts to resolve issues in contributing to business through security technology
Kenji Takatsukasa†, Kazuya Umezaki†, Tsuyoshi Miyazaki†
†Fuji Electric Co.Ltd

する機能の関数を呼び出さないようにする状態遷移制御に書き換え，呼び出し先関数テーブルなどの参照を無効（NULL ポインタ）化し，コンパイラの最適化オプションと組み合わせて ROM 使用量を削減することとした．

2.3 高速化の検討

高速化の基本方針を表 1の通り検討した．

表 1　高速化の基本対策の比較検討

対策	開発費	コスト	セキュリティ	備考
アクセラレータ利用	○	×	◎	チップ単価が+1$
アプリ小規模化	×	○	◎	アプリ開発側の追加開発必要
検証範囲縮小	○	○	×	検証しない範囲にリスク
(ブート後に追加検証)	×	○	◎	大幅な改造が必要
代替アルゴリズム	△	○	○	直ちに危険になる訳ではない
高速版ライブラリ利用	**○**	**○**	**◎**	**改造が最も少ない**

各種暗号エンジンの API を用意したライブラリは，マイコンベンダからバイナリ形式で提供されており，演算を高速化した実装タイプのものも使用するライブラリの選択肢として提供されている．開発による不具合の作り込みのリスクを鑑み，自作暗号エンジンは使わず，評価された提供物をできるだけそのままの形で使うこととした．

次に，セキュアブート機能で設定可能な選択肢として用意されている 2 つのブート方式において使用されている暗号アルゴリズムの組み換えを検討した．ブート対象の検証では，署名を検証するアルゴリズムと，署名に使われる検証対象の Hash 値を算出するアルゴリズムが使われている．図 2における各アルゴリズムの所要時間の内訳について，署名検証は検証対象のサイズに関わらず固定で要するものであるため Y 軸との交点が，Hash 値算出は署名検証分を差し引いたものが，それぞれ該当し，速度の優劣については表 2の関係にある．

表 2　提供方式のアルゴリズムの速度比較

	署名検証	Hash 値算出
方式(1)	×	○
方式(2)	○	×

本環境では，アルゴリズム自体にセキュリティ上の優劣はないため，対策方式として，方式(2)の署名検証アルゴリズムと方式(1)の Hash 値算出アルゴリズムを組み合わせることとした．

2.4 対策の結果

【ROM 使用量削減】

高速版ライブラリの利用は，開発環境上ではコンパイラでインポートするライブラリを差し替えるだけで対応できるが，一般的な高速化実装の傾向と同様に ROM 使用量が増加する．ただし，未使用機能の無効化でそれ以上の削減ができているため，全体としては提供時の 60%程度のサイズにまで圧縮できた．また，無効化した更新機能だけが使用する 300KB 以上の作業用 ROM 領域を，アプリ／ブートアプリで自由に使用できるようになった．

【高速化】

図 3に示すように，高速版ライブラリを利用しただけでは図 2からの改善はあっても目標未達だったが，アルゴリズムの組み換えも行った対策方式では目標の 0.3 秒以下を達成できた．

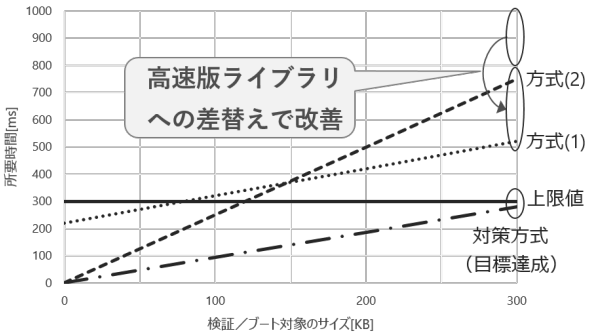

図 3　高速版ライブラリ利用時の性能

3 おわりに

本稿では，セキュリティ対策技術の実製品への適用における課題とその解決に向けた取り組みについて述べた．今後，社内の他の製品への展開を進めて行く．

参考文献

[1] IPA，顕在化した IoT のセキュリティ脅威とその対策
https://www.ipa.go.jp/files/000059579.pdf