情報処理学会論文誌 Vol.62 No.1 254–262 (Jan. 2021)

車載ECU向け省メモリソフトウェア更新方式

染谷 一輝[1] 鈴木 直希[2] 寺島 美昭[3] 清原 良三[2,a)]

受付日 2020年4月7日, 採録日 2020年10月6日

概要：コネクテッドカーの登場により自動運転車両や車車間，路車間の通信を利用した高度な制御のできる車両が増加しつつある．高度な制御を行うために，車載 ECU の数や，ソフトウェアの規模は増大している．ネットワークに接続可能であるがゆえのセキュリティ的な脅威も存在する．よって，ECU のソフトウェアの更新機能は必須であるといえ，携帯電話網などのネットワークを経由して更新することが望ましい．車外と車両を結ぶネットワークに比べ，車載ネットワークの主流である CAN は速度が遅く，更新時間全体から見ると車載ネットワーク転送時間がボトルネックとなる．そのため，車載 ECU のソフトウェア更新では差分更新機が基本となる．しかし，差分更新を適用するには各 ECU に十分な RAM が必要であり，RAM に余裕がない場合は利用できずにソフトウェアの転送量が大きくなるという課題がある．本論文では従来の汎用的なデータ圧縮方式を改良し，圧縮用の辞書を再利用することにより，RAM 容量が小さくても更新データのサイズが小さく，かつソフトウェア更新が可能な方式を提案する．提案方式を 2 種類の CPU の実行イメージに適用し，評価を行いその有効性を示す．

キーワード：ソフトウェア更新，ECU，圧縮，圧縮用辞書

Memory Saving Update Method for in-vehicle ECU

Kazuki Someya[1] Naoki Suzuki[2] Yoshiaki Terashima[3] Ryozo Kiyohara[2,a)]

Received: April 7, 2020, Accepted: October 6, 2020

Abstract: Vehicles with autonomous driving, V2V (Vehicle to Vehicle) and V2R (Vehicle to Roadside) functions will be shipped in the market. These vehicles have many ECUs (Electronic Control Units) and the size of software is very large for high level control. Moreover, there are security risks because of connected to the network. Therefore, the software update technologies by OTA (over the air) are required. However, the band width of popular in-vehicle network which is CAN (controller Area Network) is too narrow. This means the factor that causes the bottleneck is the data size for software updating. The differential compression technologies were proposed for software updating. However, this technologies require the large RAM for each ECU. There are some cases which cannot be applied the differential compression technologies, because the size of RAM is not sufficient. In this paper, we propose a new method with small RAM which is improved general purpose compression method with reusing the dictionary for compression. Then, we evaluate our method with two types of CPU architecture.

Keywords: software update, ECU, compression, dictionary for compression

1. はじめに

コネクテッドカーや自動運転車両の研究，開発が活発に行われており，車車間，路車間通信を活用した様々なアプリケーションが検討されている．なかには，外部から車両の動きを制御する場合もある．ネットワークに接続されることにより，外部からの情報に基づいて自動的に制御する

1 神奈川工科大学大学院
Graduate School of Kanagawa Institute of Technology, Atsugi, Kanagawa 243–0292, Japan
2 神奈川工科大学
Kanagawa Institute of Technology, Atsugi, Kanagawa 243–0292, Japan
3 創価大学
Soka University, Hachioji, Tokyo 192–8577, Japan
a) kiyohara@ic.kanagawa-it.ac.jp

© 2021 Information Processing Society of Japan

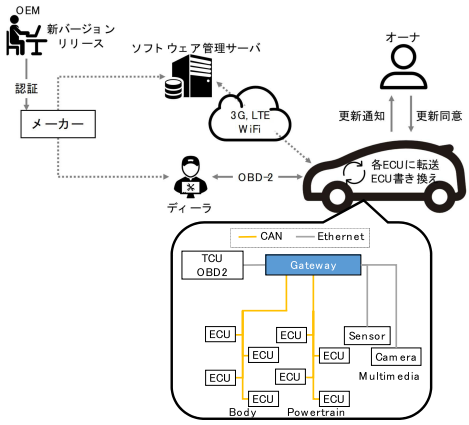


図 1　OTA アップデートシステム
Fig. 1　OTA update system.

ソフトウェアは攻撃を受ける可能性 [1], [2] を考慮する必要がある．また，研究開発がさかんであるがゆえに，機能も日々向上していくことが想定される．さらに，車載機器の制御を行う ECU（電子制御装置）のソフトウェアの不具合も多くなりつつある [3]．

従来のようにディーラに車両を持ち込んでのリコール対応や，機能の更新のほかにも，セキュリティ関係など緊急性のある更新も今後増加するこが予想され，手間とコストを考慮すると，携帯電話網などの無線通信を活用して OTA（Over-the-Air）で PC やスマートフォンのようにソフトウェアを更新することが望ましい．

OTA ソフトウェア更新システムのイメージを**図 1** に示す．車載 ECU は部品メーカ（サプライヤ）が開発していることが多く，サプライヤにおいてソフトウェアの新版を作成し，メーカで試験など確認を行い認証する．新版のソフトウェアは，メーカのサーバ経由 OTA で車両に配信されるか，ディーラで OBD2（On-board diagnostics 2）ポートから車両に送る．車両内では車載ネットワークを通じて各 ECU に更新データを配信し，ソフトウェアの書き換えを行う．車載ネットワークは一般的には CAN（Controller Area Network）[4] が利用されており，帯域が 500 Kbps 程度しかなく，車載ネットワーク上のデータ転送時間が更新時間のほとんどを占めることが分かっている [5]．FlexRay [6] など高速な車載ネットワークも提案されているが，コストの関係でそれほど普及していない．そこで将来的には車載イーサネットの組み合わされるものの，末端の ECU は CAN で接続されると想定できる．

すなわち，車載ネットワーク上を送信するデータサイズで更新時間が決まる．OTA での更新では，車載ネットワークを利用した更新中には，ユーザは車両を利用できない．そのため，更新時間を短くするためにデータサイズをいかに小さくするかが課題となる [5]．具体的には，文献 [5] に示されるように，代表的な ECU であるルネサス社製 ECU の RH850/FIL [7] を想定し，プログラムサイズ 256 KB から 2 MB 程度を更新するとして，更新時間 1 分未満を目標とする．すなわち，500 Kbps で 1 分で転送完了することを目標とするため，240 KB 程度にデータを抑えることを目標とする．

データ量を削減するためには，新版と旧版の違いのみを配信し，ECU 上で旧版に差分を適用して更新する差分更新技術を利用することが考えられる [5], [8]．

一方，車載 ECU には様々な種類があり，コストの関係で RAM の搭載が少ないものがある．文献 [5] でも示されているように差分更新ではフラッシュメモリの消去ブロックサイズ以上の RAM が必要であり，差分更新が適用できない場合もある．

そこで，本論文では，差分更新が適用できないようなフラシュメモリの消去ブロックのサイズよりも RAM 容量が少ない場合に，受信側のデータを活用することにより，従来からあるデータ圧縮方式の改良を提案する．具合的には，データを圧縮するための圧縮用辞書をあらかじめ出荷時にフラッシュメモリに記憶し，新版を作成する際にもこの旧版の辞書を再利用して圧縮する．提案方式を評価したうえでその有効性を示す．

2.　関連研究

ソフトウェア更新は PC のソフトウェアの更新などで一般の人にも身近なものになっている．一方で時間がかかるため困るときもあることがよく知られている．そのため更新時間を短くするための研究開発も多く実施されている．ソフトウェアの更新フローで時間がかかるのは，以下の 2 点である．

- デバイス上でのソフトウェアの書き換え
- デバイスへの更新データの転送

ここで，PC やスマートフォンでは回線速度も速く，デバイスの保存領域も十分あることが多いため，バックグラウンドで転送するため，後者のソフトウェアの書き換え時間を短くすることを考えなければならない．一方車載 ECU などでは個々のソフトウェアの規模は，PC やスマートフォンほど大きいわけではないため，書き換え時間は短く，遅いネットワーク上の転送時間が問題となる．

前者に関して文献 [9] はソフトウェアの差分が出にくいソフトウェアの構成法に関するもので，プログラム全体をモジュール化し，不具合などの更新でもプログラム上のアドレス参照部分の変化が少ないようにする手法で，ソフトウェアの書き換え量の削減も狙っている．大規模なソフトウェアでかつ，フラッシュメモリの書き換え時間がボトルネックとなるような場合に有効な手法である．しかし，車載 ECU のソフトウェアは個々の ECU を見ると大規模ではない．また，フラッシュメモリの書き換え時間がボトルネックになるわけでもない．

後者に関するデータ転送量の削減の研究は，筆者らも長

© 2021 Information Processing Society of Japan

年研究しており，携帯電話のソフトウェアの書き換えで実用化もされている [9], [10].

文献 [11], [12] は携帯電話に限らず組み込み機器を対象とした差分更新方式であり，また文献 [5], [13] は差分そのものを小さく表現するための工夫であり，車載 ECU のソフトウェア車載 ECU の更新には有効な手法であるが，メモリを消費するため，RAM に余裕のない場合には適用できない問題がある．

適用時にワークメモリが不足する場合を想定したアルゴリズムも提案されている [14], [15]．しかし，この方式においても消去ブロックサイズ以上の RAM は必要であり，消去ブロックのサイズに満たない RAM しかない場合には適用できない．

文献 [16] は，車載 ECU で RAM が小さい場合を想定しているが，汎用的な圧縮方式のメモリ使用量や解凍時間を評価しており，どのような汎用圧縮方式が適切かを示しているにすぎない．汎用的な圧縮方式は様々提案されている [26]．これらはそれぞれ，圧縮が高速なアルゴリズムや，解凍が高速なアルゴリズム，省メモリアルゴリズムなどがある．筆者らの先行研究である．文献 [16] で評価した結果，LZ77 系アルゴリズム [27] が車載 ECU のソフトウェアの圧縮には適していることが示されている．

さらに，ソフトウェアの更新においてはデータのダウンロード処理以外に，完全性の保証や安全性の保証が必要である．完全性の保証にはチェックサムや CRC といった従来手法で十分である．

安全性の保証には多くの研究がある．車載ネットワークの脆弱性をついた攻撃が報告されている．CAN はブロードキャストによる通信を行っており，CAN バスに接続を行えば容易に通信メッセージの盗聴が可能である．また，暗号化や認証の機能を持っていないため，悪意のある送信者から不正なメッセージを受信してしまう恐れがある．

車載ソフトウェアの標準化を進めている AUTOSAR [17] では，CAN メッセージにパケットカウンタと Message Authentication Code を付与することで，メッセージの完全性を保証する方式が策定されている [18]．MAC を付与することで，CAN メッセージの認証やリプレイ攻撃の阻止が可能となる．しかし，MAC を付与することで CAN メッセージのペイロードを圧迫してしまうため，データの転送効率は低下する．MAC による負荷を軽減するために，軽量な MAC 認証手法が提案されている [19]．カウンタ値をデータとは別のメッセージで送信することで，ペイロードの負荷を軽減している．

各 ECU がバス上のメッセージを監視し，自身の ID を持つメッセージが不正に送信されていたときに偽装メッセージを破棄する方法が提案されている [20]．ECU に簡易な変更を加えることで実現可能な手法であり，手法によるトラフィックの増加もないが，対象となる ECU を取り外されてしまうと機能しないという問題がある．

セキュアエレメントをサポートした EUC を用いることで，安全性のハードウェアレベルの信頼性を実現する手法が提案されている [21]．車両制御に関する ECU のセキュアブート，CAN のメッセージ単位の認証，更新データの署名と検証することで，エンジン始動から走行時，メンテナンス時の安全性を確保している．CAN メッセージの認証のために，分割した MAC を付加している．MAC サイズによりセキュリティ強度が変わるため，セキュリティ強度とペイロード負荷を加味して MAC サイズを決定することを勧めている．

CAN バスを監視しルールベースによる不正検知と無効化を行う ECU と，遠隔監視サーバによる機械学習を用いた異常検知による多層化した攻撃検知システムが提案されている [22]．教師なし学習による異常検知アルゴリズムを用いており，ソフトウェアアップデートにおいても更新データ転送の正常モデルを生成することで適用可能であると考えられる．

また，CAN 上の通信を安全にするために CAN メッセージの周期性を用いた異常検知手法が提案されている [23]．CAN 上の通信が一定の周期的性質を持つことに着目し，メッセージの周期性と出現順序の特徴から正常状態を抽出し異常を検知している．

コンシューマ機器向けのソフトウェアアップデートを車載機器向けに採用することで，インフォテインメントシステムソフトウェアのセキュアなソフトウェアアップデートを提案している [24]．提案されたアプローチでは，インフォテインメントシステム製造者以外の製作者によって作成されたソフトウェアではシステムを更新できないようにすることで，安全なソフトウェアアップデートを提供している．

ECU の認証と車載ネットワークの暗号化を行うために，ハッシュアルゴリズムと共通鍵暗号方式を採用したシステムが提案されている [25]．改ざんやなりすまし，リプレイ攻撃に対し低コストで対処することができることをシミュレーションにより示している．

このようにセキュリティに関しては多数の研究がある状況であるものの，いずれも更新時間を気にするものはなく，データの圧縮の重要性がより高いということができる．また，書き換え後の完全まで含めたシステムに関しても，多数研究されている [28], [29]，本論文では，車載機器までのダウンロードにおいて保証されているものとする．そこで，本論文では，LZ77 系の基本アルゴリズムを改良し，圧縮辞書を再利用する方式を提案し，車載 ECU の主流だる NOR 型のフラッシュメモリでの書き換えに対して有効性を示す．

© 2021 Information Processing Society of Japan

3. 車載 ECU ソフトウェア更新

3.1 車載ソフトウェア更新システム

ソフトウェアの更新は，図 1 に示すようにまずサプライヤにおいて不具合や脆弱性に対する対策などの修正を各 ECU 向けに行う．本論文の前提となるソフトウェア更新のシステムに関して以下に説明する．出荷元は，ECU ごとに次に示す情報を把握している．

- CPU の種類
- RAM サイズ
- NOR 型フラッシュメモリの消去ブロックサイズ
- フラッシュメモリの総容量

ここで，ある版から新しい版にソフトウェアを更新するために必要な更新データを生成する．部品メーカはこのデータの正しさなどを出荷元で適用試験を実施するとともに，カーメーカも適用試験をしたうえでリリースする．車両側は適切な更新プロトコルに基づいて，車両のソフトウェアのバージョンや ECU の種類といった情報を更新データの配信サーバと交換することにより適切なデータを OTA でカーナビや TCU（Telematics Control Unit）と呼ばれるゲートウェイなどにダウンロードし，そこから車両が停止しているときに，車載ネットワークを利用してデータを ECU に送り更新する．

データを生成する際には，差分更新に必要な RAM が確保できることが分かっていれば差分更新を，そうでなければ汎用のデータ圧縮方式などでデータを圧縮して送信する．こういったソフトウェア更新システムを前提とする．

3.2 車載 ECU の更新

ECU 上で，差分更新の場合は，**図 2** に示すフローで更新する．最初に RAM 上に更新のためのデータをダウンロードするとともに，次にフラッシュメモリの消去ブロック分のデータを RAM に読み込む．さらに，消去ブロックを消すとともに R 旧版のデータに差分を適用することにより RAM 上に新版のイメージを作成する．最後に新版のイメージを書き込む．差分データは消去ブロック単位で車載ネットワークを利用してダウンロードすればよいが，それでも RAM の容量は消去ブロックのサイズの倍以上は最低でも必要となる．

そのため，十分な RAM が搭載されていなければ差分適用は難しい．あるいは，フラッシュメモリに予備の消去ブロック 1 つ分を用意することができるのであれば，旧版データは RAM ではなく，予備の消去ブロックにコピーすることで RAM を節約することができる．しかし，フラッシュメモリの書き換え量は 2 倍になり，書き換え時間も無視できなくなるかもしれない．

一方汎用的な圧縮を利用して更新する場合は，**図 3** に示すフローになる．まずデータを車載ネットワークを利用してダウンロードし，RAM 上に置き，圧縮を解凍し，フラッシュメモリを消去してから書き込む．RAM が小さい場合は，何度もダウンロード展開し，書き込む操作を繰り返す．フラッシュメモリでは消去操作は消去ブロックをまとめて消すが，書き込む操作は逐次できる．本論文では，この処理のための車載ネットワーク上のダウンロードデータ量の削減を目的としている．

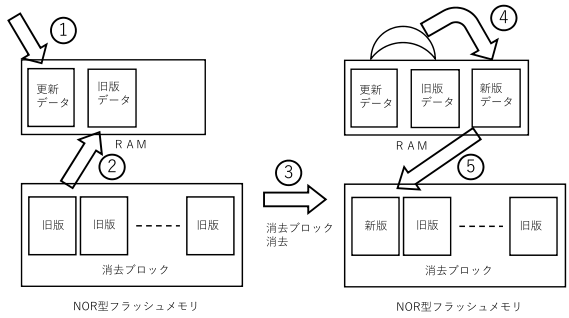


図 2 NOR 型フラッシュメモリの差分書き換え
Fig. 2 Software update by binary diff on NOR flash memory.

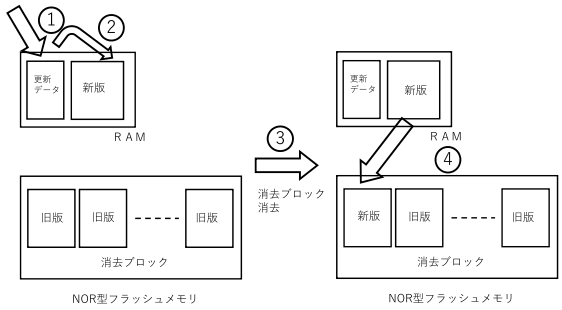


図 3 NOR 型フラッシュメモリの単純書き換え
Fig. 3 Software update by simple method on NOR flash memory.

4. 提案方式

4.1 データ圧縮

データ圧縮は古くから研究され多くのアルゴリズムやツールが存在する．しかし，その基本となるのは，ハフマン符号化 [30] および LZ77 [27] や，LZ78 [31] である．基本原理はすでに出てきたデータ列のならびが再度出てきたら短い符号に置き換える手法である．

先行研究 [16] で比較評価を行った結果，車載 ECU へのソフトウェアの配布はにおいては，圧縮率，伸長時のメモリ消費量，解凍速度の観点から LZ77 系の圧縮が良いことが示されている．工場出荷時にかかる圧縮の時間は，ECU 上の処理比べて高速な実行環境が準備でき，個々の ECU ソフトウェアの規模は大きくないことから無視してよい．差分適用に関してはメモリの消費量が多く圧縮率は高い．高速な実行環境でプログラムメモリの読み出し時間の方が展開時間より気になるようなケースに利用する圧縮方式の BPE [32] はメモリ消費量は小さいが，圧縮率が LZ77 系に

© 2021 Information Processing Society of Japan

比べ良くない．そこで，本論文では，広く実装されおり，ソースコードの改変の容易さも考え，LZ77 をベースとしている Zstandard [33] を車載 ECU のソフトウェア更新向けに改良することとした．

4.2　提案辞書再利用方式

LZ77 は記号列を一致位置，一致長，次の不一致記号という 3 種類の値に，順にデータをスキャンしては置き換えることにより圧縮する方式である．記号列を探す範囲をスライドウインドウとして，置き換えのパターンを辞書として行う．圧縮する際には，データ全体を検索しやすいように工夫することが多く，これを圧縮用辞書と呼ぶことにする．一方，一致位置や一致長が指すデータを展開用辞書と呼ぶことにする．圧縮時には，データをスキャンしながら圧縮用の辞書を引き，同じ記号列を見つけると置き換える操作を繰り返し**図 4** の従来方式に示すように最終的には初めて出た記号列が辞書のデータとなり，2 回目以降に現れた同じ記号列が一致位置，一致長という形で表現される．なお，不一致記号は，同じデータが辞書中になかったデータなので，圧縮データ中にそのまま埋め込まれる．なお，短い記号列で，一致長と一致位置で示す方がデータが多くなる場合も不一致記号として扱う．

図 4 に提案方式の概要を示す．旧版は，通常に圧縮する．繰返しパターンは，一致位置と一致長という情報に置き換えられることにより圧縮されることは元の方式と同じである．ここで展開用の辞書を圧縮データ中に埋め込まず独立したものとして取り出し，出荷する ECU のフラッシュメモリに保存しておく．これにより，初見のデータも含めて，一致位置，一致長という形と不一致記号の列として表現することができる．図 4 の上半分では，*ABC* にあたる部分が複数回出てくるため辞書に置かれ，データにはその参照情報が置かれデータを小さく表現できる．さらに展開用辞書として取り出し，辞書として保存して出荷する．

圧縮時には旧版で使った圧縮用辞書を利用するとともに，展開用辞書への参照に置き換えることで圧縮する．展開時には，フラッシュメモリ上にあらかじめおいてある辞書を利用して復号する．図 4 の下半分にあるように，新版を従来どおり圧縮するのに比べると提案方式では繰り返し部分ではデータ圧縮率は劣ることになるが，辞書部分を送らなくてすむため，送信のためのデータサイズは削減できる．

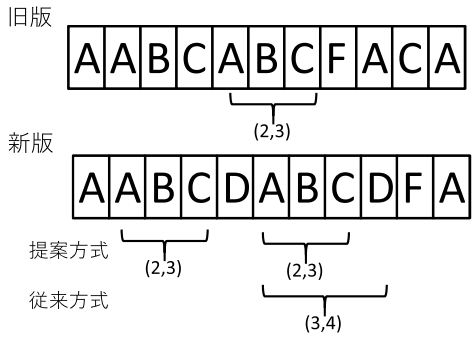


図 4　圧縮辞書
Fig. 4　Dictionary for compression.

図 5 に具体的な工場出荷時から ECU 上の状態および更新時の工場での処理と ECU 上での処理を示す．出荷時の版を旧版と呼ぶことにする．工場では，旧版ソフトウェアは LZ77 ベースの圧縮を行う．圧縮を行わない形で，直接実行可能なように ECU の NOR 型フラッシュメモリには保存する．また，この圧縮時の辞書情報を解凍用の辞書として，フラッシュメモリなどに保存しておき，一緒に出荷する．

ソフトウエア更新時には工場で新版のソフトウェアを旧版の圧縮時に利用した辞書を用いて圧縮する．このときの辞書を除いたデータを新版のデータとして，OTA で旧版を搭載している車両に配布する．

車両上では，該当 ECU の上で，旧版のソフトウェアを消して，新版の圧縮データと解凍用の辞書を利用して解凍を行い，新版のデータを RAM 上に作成のうえ，ECU のフラッシュメモリに保存する．車両の ECU 上での提案方式の動作例を**図 6** に示す．

消去ブロックサイズに比べ RAM に十分な容量がない場

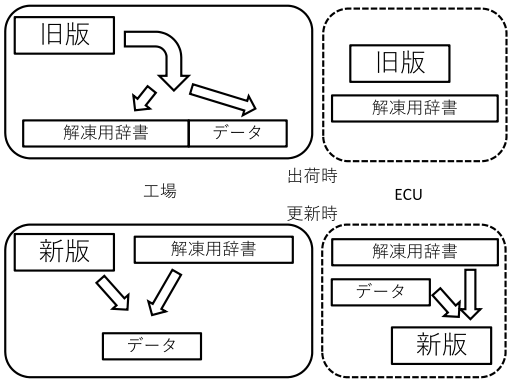


図 5　ECU のソフトウェア更新更新フロー
Fig. 5　Flow of software update on ECU.

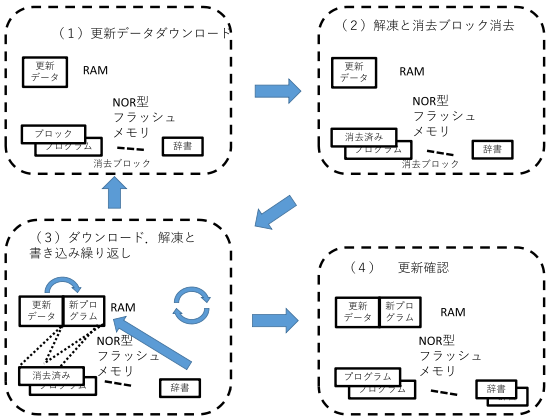


図 6　ECU のフラッシュメモリ更新
Fig. 6　Update of flash memories on ECU.

© 2021 Information Processing Society of Japan

合であるため，消去ブロック上に書き込みデータは複数回に分割して ECU に車載ネットワークを経由してダウンロードされる．

(1) 最初のデータをダウンロードする．
(2) 消去ブロックを消去する．
(3) 旧版の ECU 上にある辞書を利用して圧縮データを RAM 上に展開し，消去済みの消去ブロックの一部に書き込む．この (3) の処理を終えると，再びデータをダウンロードしては，RAM 上に新版データを作成し，1 回前に書き込んだフラッシュメモリに追加書き込みを行う．この繰返しの後，消去ブロックのサイズ分書き込みを終えると，再度 (1) から (3) の処理を次の消去ブロックに対して適用する．
(4) 最後にすべての書き込みが正しく書き込みを終えたことをチェックサムや，CRC といった従来の手法で確認する．

図 6 は例であり，適宜，書き込みが正しくできたことを確認するための処理 (4) を (1) から (3) の中に入れる場合もある．

5. 実験と評価

5.1 評価対象

提案方式の評価のために，固定長の命令長である ARM プロセッサ向けのマシン命令列と可変長の命令列となるインテル系のマシン命令列で評価を実施した．過去の辞書を使えなくなる可能性の高いパターンとしては以下の 3 点がある．

- ソースコード上での行の追加や削除などの影響で，レジスタの使用方法が変わる場合．
- 命令列の追加や削除によって参照 0 が変わることにより，アドレスが変わった関数などの呼び出し元が影響を受けてデータが変わる場合．
- 新規に追加された命令列により，旧版にはなかったデータが出てきた場合．

これらの影響を知るために，簡単なテキストエディタプログラムを利用して，意図的にこのような状況が起きるようにして評価を実施した．対象プログラムは 4 つの版を作成した．そのサイズなどの基本情報を**表 1** に示す．4 つの版は順次バージョンアップしたもので，A 版から B 版へは一部プログラムの変更である．B 版から C 版へは一部のデータの削除と追加である．C 版から D 版へは新たな関数の追加とその関数を呼ぶ部分の追加とした．そのため，D 版の追加部分約 4.5 K バイトは，A 版で作成した辞書を利用する方式では圧縮が効果が低くなる．

表 1 評価対象データ
Table 1 Feature of evaluation data.

バージョン名	ARM 向け データサイズ（bytes）	intel 向け データサイズ（bytes）
A	24,088	32,104
B	24,248	32,232
C	24,400	32,344
D	28,928	37,168

5.2 圧縮率評価

表 1 のデータを圧縮したときの圧縮データサイズと辞書を再利用して圧縮したサイズを**表 2**，**表 3** に示す．バージョン A で作成した展開用辞書をもとに残りのバージョンを作成した結果で辞書部分を含まない．バージョン A に関しても展開用辞書を抜き出したデータサイズを示している．両表の圧縮サイズは，従来の Zstandard で圧縮した場合の更新データのサイズである．提案方式の方が辞書を再利用することでどの程度小さくできるかを示している．

結果からどちらのアークテクチャでもデータサイズは約 30%程度に削減できていることが分かり，提案方式による展開用辞書の利用よる圧縮率向上の効果を確認できた．もし評価データの 10 倍から 100 倍のサイズのデータを対象とした場合のデータ提案方式のデータサイズは，40 KB から 400 KB 程度になるため，元のプログラムサイズと更新内容に依存するものの，目標をある程度クリアできている．

5.3 辞書によるフラッシュメモリ使用量とのトレードオフ

辞書は ECU のフラッシュメモリに配置する．辞書のサイズが大きいとフラッシュメモリを使うことになる．すなわち，出荷する ECU のフラッシュメモリに置く辞書のサイズと更新データはトレードオフの関係になる．数多くのパターンを配置しておけば，データを小さく表現できる．

表 2 圧縮サイズと提案方式データサイズ（ARM）
Table 2 Size of compression data and proposed method (ARM).

バージョン名	圧縮サイズ (bytes)	提案方式データサイズ (bytes)
A	10,744	2,780
B	10,873	2,733
C	11,077	3,975
D	11,839	3,736

表 3 圧縮サイズと提案方式データサイズ（Intel）
Table 3 Size of compression data and proposed method (Intel).

バージョン名	圧縮サイズ (bytes)	提案方式データサイズ (bytes)
A	11,411	3,539
B	11,576	3,655
C	11,772	3,935
D	12,954	5,060

© 2021 Information Processing Society of Japan

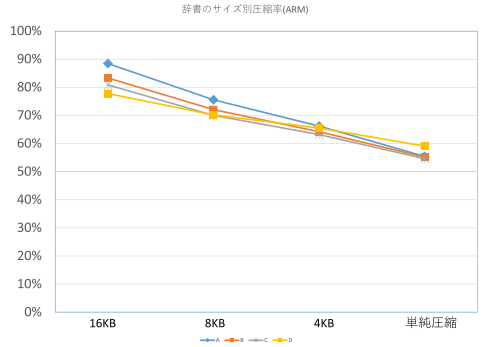


図 7　辞書サイズごとのデータサイズ（ARM）
Fig. 7　Size of data for each dictionary size (ARM).

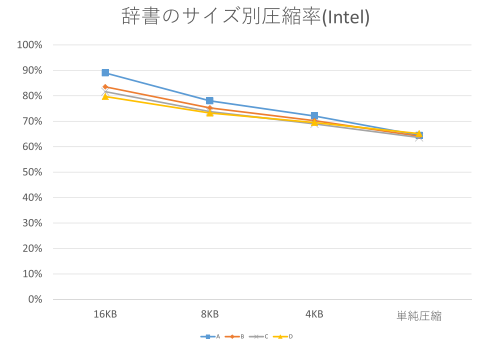


図 8　辞書サイズごとのデータサイズ（Intel）
Fig. 8　Size of data for each dictionary size (Intel).

そこで，辞書のサイズの上限を決め，上限を超える場合は，参照数の少ないものは辞書に置かないとして評価した．辞書のサイズの上限は，16 KB，8 KB，4 KB とした場合の更新用データサイズを**図 7**，**図 8** に示す．

辞書のサイズを小さくすると明らかに圧縮率は低下することを示している．しかしプログラムサイズが 20 KB から 30 KB 程度であることからプログラムサイズに対して，最低でも 15%から 20%程度の辞書を置く余裕があれば圧縮率 70%程度が達成できることが分かる．逆にいえば，フラッシュメモリの余裕が少ないと提案方式の効果が小さくなる．もし余裕がない場合は提案方式を適用できない．フラッシュメモリの余裕の度合いによっては提案方式の効果が得られる場合と得られない場合がある．なお，1 章で示した例で，全体のプログラムサイズが大きい場合で，2 MB 程度の ECU の場合には，600 KB 程度のフラッシュメモリがあれば，600 KB 程度にデータを圧縮できることを示している 240 KB 以下にするためには，1 MB 以上のフラッシュメモリを必要とし，目標を達成しない場合もある．

5.4　複数回更新における圧縮率への影響

最初の版から変更点が大きいほど，レジスタの使い方や関数館の参照関係などアドレスのずれなども多くなる．また新たなコードの追加も多いと考えれ，同じ命令列が少なくなるあめ，辞書を再利用する圧縮効果が小さくなる．単純に出荷したときの版の辞書がどの程度使えるのかで決まってくるため，改版を何回か繰り返すこととは直接的には関係がない．しかし，改版を繰り返すほど出荷時のコードとは違いが多くなる場合が多く，出荷版の辞書を使った場合の圧縮率は悪くなる．

運用上更新に時間をかけていい場合として，カーディーラなどでの点検時や車検時がある．あるいはドライバが許可すれば時間のかかる更新も可能になる．そこで，複数回の更新に対して，途中で圧縮用の辞書も置き換えた場合に関して評価を実施した．**表 4** に各バージョンのデータサイズと通常圧縮時のデータサイズを示す．

また，**図 9** にバージョン 4 のときに圧縮用の辞書を更新した場合のデータサイズを示す．辞書を更新することにより，データサイズは削減はできることは分かるが，そこから複数回更新するとやはり圧縮は大きくなる傾向にある．しかしデータを普通に圧縮するのに比べてデータ量を抑えられることには変わらず，提案手法は有効であることが分かる．

表 4　複数回更新の場合（ARM）
Table 4　Case of multiple update.

バージョン	データサイズ（bytes）	通常圧縮（bytes）
1	76,268	24,143
2	82,262	26,030
3	85,622	26,558
4	103,390	38,496
5	150,626	56,489
6	150,682	56,508
7	153,802	56,933

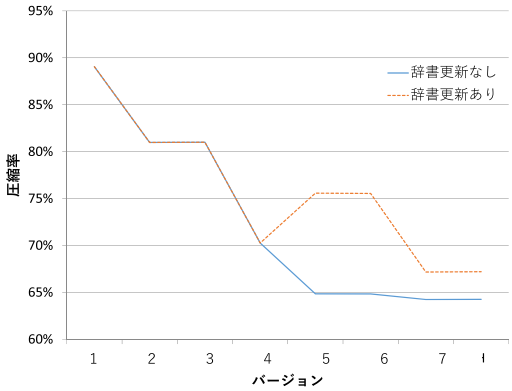


図 9　複数回の更新（ARM）と辞書の更新
Fig. 9　Multiple update (ARM) and dictionary update.

6.　考察

車載 ECU の更新はデータサイズが小さければ小さいほど時間が短くなる．OTA アップデートにおいては，ドライバが運転できなくなる時間に関係するため，この運転不可能時間をできるだけ短くすることが目標となる．実際には，ドライバは自宅や自社の駐車場に置き，一晩動かさないことも多く，長い時間を更新に利用できる場合も多い．

© 2021 Information Processing Society of Japan

しかし，家の駐車場に戻ったから一定時間は利用しないという保証はなく，自動的に判定するのも難しい．

実際の ECU のソフトウェアは，たとえば文献 [5] によれば，パワートレイン系で 2 Mbyte 弱で，1 MByte 以上のソフトウェアも多い．そのため圧縮しないと転送時間は，帯域をすべて使ったとしても数十秒かかるが，ソフトウェア更新にすべての帯域を使えることは少なく，10%程度とされており数分かかる．そのため，本提案方式のように削減できることでユーザの利便性が向上する．

しかし，条件によっては更新時間が十分短くなったとはいえない場合もある．したがって，ドライバが更新の許可を行うなど，何らかの手段で更新時間を考慮する必要がある．運用上は，緊急性のある更新とそうでない更新に分けた運用が考えられ，自宅や会社の駐車場におき，すぐに運転する予定がない場合などに時間のかかる更新をすることになる．したがって，緊急性のある更新をいかに早く行うかが課題となる．今後は運用と合わせた検討が必要になると想定され，離れた場所でも更新の許可を与えられるようなスマートフォンとも連携したシステムの検討も必要である．

7. おわりに

本論文では，車載 ECU の更新に関して，RAM の容量が，NOR 型フラッシュメモリの消去ブロックのサイズに比べて十分な容量がなく，差分更新を適用できないような場合に，更新データを小さく表現する手法を検討した．圧縮時にできる旧版の辞書を活用する手法を提案し，LZ77 系のアルゴリズムに実装し，その有効性を評価した．トレードオフとして，フラッシュメモリを使用するため，圧縮用の辞書のサイズに制限を与えた場合の圧縮率への影響を評価し，1.3 倍程度のフラッシュメモリがあると効果的あれば十分効果があることを示した．さらに複数回更新することにより，圧縮率の低下が想定されるが，辞書の更新を適切なタイミングで行えば，その影響を抑えられることも示した．

今後の課題として運用面を考えたシステムの構築と評価が必要である．また，セキュリティ面の課題も大きいため，あわせてセキュリティ面の検討と実装が必要である．また，実際の ECU ソフトウェアとそのバグフィックス状態のソフトウェアで評価をする必要もある．

参考文献

[1] Jafarnejad, S., Codeca, L., Bronzi, W., Frank, R. and Engel, T.: A Car Hacking Experiment: When Connectivity Meets Vulnerability, *2015 IEEE Globecom Workshops* (*GC Wkshps*), San Diego, CA, pp.1–6 (2015).

[2] Kumar, A.D., Chebrolu, K.N.R., Vinayakumar, R. and Soman, K.P.: A Brief Survey on Autonomous Vehicle Possible Attacks, Exploits and Vulnerabilities, arXiv:1810.04144 (2018).

[3] Ministry of Land, Infrastructure, Transport and Tourism: Vehicle Recall, available from ⟨http://www.mlit.go.jp/en/jidosha/vehicle_recall_17.html⟩ (accessed 2020-03-16).

[4] Bosch, R.: CAN specification version 2.0, Rober Bousch GmbH, Postfach, 300240 (1991).

[5] 寺岡秀敏，中原章晴，黒澤憲一：車載 ECU 向け差分更新方式，情報処理学会論文誌コンシューマ・デバイス&システム（CDS），Vol.7, No.2, pp.41–50 (2017).

[6] Lorenz, S.: The FlexRay electrical physical layer evolution, *SPECIAL EDITION HANSER automotive FLEXRAY*, pp.14–16 (2010).

[7] ルネサス社，入手先 ⟨https://www.renesas.com/jp/ja/products/microcontrollers-microprocessors/rh850/rh850f1x/rh850f1l.html⟩.

[8] Bogdan, D., Bogdan, R. and Popa, M.: Delta flashing of an ECU in the automotive industry, *2016 IEEE 11th International Symposium on Applied Computational Intelligence and Informatics* (*SACI*), pp.503–508 (2016).

[9] 星　誠司，一瀬晃弘，野瀬康弘，細川篤司，武市真知，矢野英司：無線通信を利用した「ソフトウェア更新」システム，NTTDoCoMo テクニカルジャーナル，Vol.11, No.4, pp.36–41 (2004).

[10] 清原良三，栗原まり子，小宮章裕，高橋　清，橘高大造：携帯電話ソフトウェアの更新方式，情報処理学会論文誌，Vol.46, No.6, pp.1492–1500 (2005).

[11] 清原良三，栗原まり子，三井　聡，木野茂徳：携帯電話ソフトウェア更新のためのバージョン間差分表現方式，電子情報通信学会論文誌 B，Vol.J89-B, No.4, pp.478–487 (2006).

[12] 清原良三，三井　聡，木野茂徳：組込みソフトウェア向けバイナリー差分抽出方式，電子情報通信学会論文誌 D，Vol.J90-D, No.6, pp.1375–1382 (2007).

[13] Onuma, Y., Nozawa, M., Terashima, Y. and Kiyohara, R.: Improved Software Updating for Automotive ECUs: Code Compression, *IEEE 40th Annual Computer Software and Applications Conference* (*COMPSAC*), pp.319–324 (2016).

[14] Burns, R., Stockmeyer, L. and Long, D.D.E.: In-place reconstruction of version differences, *IEEE Trans. Knowledge and Data Engineering*, Vol.15, No.4, pp.973–984 (2003).

[15] Nakanishi, T., Shih, H., Hisazumi, K. and Fukuda, A.: A software update scheme by airwaves for automotive equipment, *International Conference on Informatics, Electronics and Vision*, pp.1–6 (2013).

[16] Onuma, Y., Terashima, Y., Nakamura, S. and Kiyohara, R.: Compression method for ECU software updates, *10th International Conference on Mobile Computing and Ubiquitous Network* (*ICMU*), pp.1–6 (2017).

[17] AUTOSAR, available from ⟨https://www.autosar.org/⟩ (accessed 2020-04).

[18] AUTOSAR, Requirements on Secure Onboard Communication, available from ⟨https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_SRS_SecureOnboardCommunication.pdf⟩ (accessed 2018-01).

[19] 中野将志，久保田貴也，汐崎　充，藤野　毅：先進運転支援システムを搭載した自動車に対する制御乗っ取り攻撃の脅威分析，情報処理学会研究報告システムと LSI の設計技術（SLDM），Vol.2016-SLDM-175, No.3, pp.1–6 (2016).

[20] 畑　正人，田邉正人，吉岡克成，大石和臣，松本　勉：不正送信阻止：CAN ではそれが可能である，情報処理学会セキュリティシンポジウム 2011 論文集，pp.624–629

© 2021 Information Processing Society of Japan

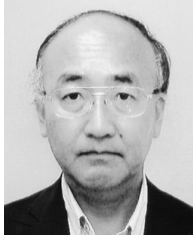
(2011).
[21] 竹森敬祐，溝口誠一郎，川端秀明，窪田 歩：セキュアブート＋認証による車載制御システムの保護，情報処理学会研究報告高度交通システムとスマートコミュニティ（ITS），Vol.2014-ITS-58, No.8, pp.1–8 (2014).
[22] 芳賀智之，岸川 剛，鶴見淳一，松島秀樹，高橋良太，佐々木崇光：機械学習による車載ネットワーク攻撃検知システム，パナソニック技報，Vol.63, No.1, pp.16–21 (2017).
[23] 栗田 萌，渡辺俊貴，山野 悟：ログの順序パターンに基づく異常検知手法の提案と CAN のログへの適用，情報処理学会研究報告マルチメディア通信と分散処理（DPS），Vol.2017-DPS-170, No.28, pp.1–7 (2017).
[24] Kapilan, G., Arumugam, K. and Chamundeswari, A.: An Approach for Secure Software Update in Infotainment System, *Proc. 10th Innovations in Software Engineering Conference* (*ISEC*), pp.127–131 (2017).
[25] Qin, Z., Li, F., Wu, Y.-H. and Wang, C.: New ECU Attestation and Encryption Mechanism for In-Vehicle Communication, *DEStech Trans. Engineering and Technology Research*, (*SSME-IST*) (2016).
[26] Uthayakumar, J., Vengattaraman, T. and Dhavachelvan, P.: A survey on data compression techniques: From the perspective of data quality coding schemes data type and applications, *J. King Saud Univ. Comput. Inf. Sci.* (2018).
[27] Ziv, J. and Lempel, A.: A universal algorithm for sequential data compression, *IEEE Trans. Information Theory*, Vol.23, No.3, pp.337–343 (1977).
[28] Steger, M., Boano, C.A., Niedermayr, T., Karner, M., Hillebrand, J., Roemer, K. and Rom, W.: An Efficient and Secure Automotive Wireless Software Update Framework, *IEEE Trans. Industrial Informatics*, Vol.14, No.5, pp.2181–2193 (2018).
[29] Chowdhury, T., Lesiuta, E., Rikley, K., Lin, C.-W., Kang, E., Kim, B.G., Shiraishi, S., Lawford, M. and Wassyng, A.: Safe and Secure Automotive Over-the-Air Updates, *Computer Safety, Reliability, and Security, SAFECOMP 2018*, Gallina, B., Skavhaug, A. and Bitsch, F. (Eds), Lecture Notes in Computer Science, Vol.11093, Springer, Cham (2018).
[30] Huffman, D.A.: A Method for the Construction of Minimum-Redundancy Codes, *Proc. IRE*, Vol.40, No.9, pp.1098–1101 (1952).
[31] Ziv, J. and Lempel, A.: Compression of individual sequences via variable-rate coding, *IEEE Trans. Information Theory*, Vol.24, No.5, pp.530–536 (1978).
[32] Gage, P.: A New Algorithm for Data Compression, *The C Users Journal*, Vol.12, No.2, pp.23–38 (1994).
[33] Collet, Y.: Zstandard – Real-time data compression algorithm (2015), available from ⟨http://facebook.github.io/zstd/⟩.

染谷 一輝 （学生会員）

2019 年神奈川工科大学情報学部情報工学科卒業，同年神奈川工科大学大学院工学研究科情報工学専攻前期課程入学．ソフトウェア更新技術の研究に従事．

鈴木 直希

2020 年神奈川工科大学情報学部情報工学科卒業，同年明電システムソリューション株式会社入社．在学中ソフトウェア更新技術の研究に従事．



寺島 美昭 （正会員）

1984 年埼玉大学工学部電子工学科卒業，同年三菱電機（株）入社．2015 年より創価大学理工学部情報システム工学科教授．分散システム開発と評価手法，無線ネットワーク信頼性に関する研究に従事．博士（工学）．IEEE，電子情報通信学会各会員．

清原 良三 （正会員）

1985 年大阪大学大学院工学研究科応用物理学専攻前期課程修了．同年三菱電機（株）入社．1988 年より（財）新世代コンピュータ技術開発機構に出向，1993 年三菱電機（株）に復職．2012 年より神奈川工科大学情報学部情報工学科教授．組込み機器のソフトウェア更新，JavaVM 実装方式，組込み機器の信頼性，ITS に関する研究に従事．2008 年大阪大学大学院情報科学研究科博士後期課程修了．博士（情報科学）．電子情報通信学会，ACM 各会員，IEEE Senior Member，本会フェロー．

© 2021 Information Processing Society of Japan