圧縮辞書の再利用による車載 ECU 向けデータ圧縮方式

染谷一輝[1]　寺島美昭[2]　清原良三[3]

概要： 自動運転車両の研究が活発に行われている．実際に日本でも自動運転レベル 3 の自動車が登場しつつあり，高度な制御が要求される．その制御を行うため，車載 ECU の数の肥大化，その中身であるソフトウェアの大規模化が進んでいる．一方でネットワークに接続し，便利な分，セキュリティ面での脅威も存在する．事故につながるバグや外部から攻撃される危険性を考えると ECU のソフトウェアを迅速に更新する必要があると言える．また更新には携帯網などのネットワークを経由することが対応速度的に望ましい．データ受信部となるところまでは携帯網で比較的早い通信速度でデータ受信できるが，自動車内部のネットワークは 500Kbps ととても遅い．そのため車載ネットワークでは差分更新が基本となる．しかし差分更新するためには各 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ない．本論文では従来の汎用的なデータ圧縮方式を改良し，プリセット辞書を利用することでデータを小さくし，RAM 容量に余裕がなくともソフトウェア更新が可能な方式を提案する．

Data compression method for in-vehicle ECU by reusing compression dictionaries

KAZUKI SOMEYA[1]　TERASHIMA YOSHIAKI[2]　RYOZO KIYOHARA[3]

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．中には，自動的に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，研究開発が盛んであるが故に，機能も日々向上されていくことが想定される．また，エンジンやトランスミッション，エアバックなど多くの車載機器の制御をしている ECU(電子制御装置)はソフトウェアの大規模化によるプログラミングミスなどにより，不具合も多くなりつつある[3].

このような事故を招きかねない状況においては，早急にセキュリティの強化や不具合を取り除くことが求められる．従来のようにディーラーに車両を持ち込んでのリコールや，機能の更新をするのではなく，セルラー網などの無線通信を活用して OTA(Over-the-Air)でのデータ配信や，スマートフォンのようにソフトウェアをアップデートすることが望ましい.

OTA での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 は通常部品メーカ(OEM)が開発していることが多く，OEM にてソフトウェアの新しい版を作成し，メーカにて試験など確認をして認証する．新版のソフトウェアは, メーカのサーバ経由で OTA を通して車両に配信されるか，ディーラで OBD2 ポートから車両に配信する．車両内は車載ネットワークを通じて各 ECU に配信し，ソフトウェアの書き換えをする．車載ネットワークは一般的には CAN(Controller Area Network)[4]が利用されている．帯域が 500Kbps 程度しかなく，更新時間のほとんどを配信時間で占めることがわかっている[6]．FlexRay[5]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．将来的には車載イーサネットの組み合わさるものの，末端の ECU は CAN で接続されることになる．即ち，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ているため更新中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はデータサイズを如何に小さくするかが課題となる[6]．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[6][7].

一方，車載 ECU には様々な種類があり，コストの関係で

1 神奈川工科大学大学院
Graduate School of Kanagawa Institute of Technology
2 創価大学
Soka University
3 神奈川工科大学
Kanagawa Institute of Technology

© 2020 Information Processing Society of Japan

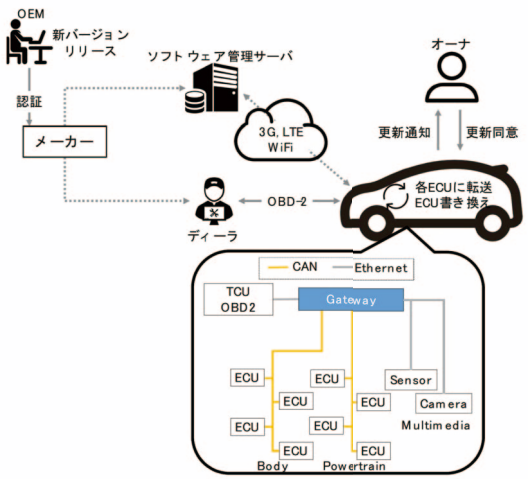

図 1. OTA アップデートのイメージ

RAM の搭載が少ないものがある．文献[cite bottoleneck]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．

本論文では，差分更新が適用できないような RAM が小さな場合にも単なるデータ圧縮に比べデータ量を削減する圧縮方式として，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ことを提案し，評価した上でその有効性を示す．

2. 関連研究

ソフトウェア更新は PC ソフトウェアの更新などで身近なものになっている．一方で時間がかかるため困る時もあることがよく知られている．そこで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．

- デバイスへの更新データの配信
- デバイス上でのソフトウェアの書き換え

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配信する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配信時間が問題となる．

前者のデータ配信量の削減の研究は，筆者らも長年研究しており，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[8][9]．

文献[8]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ある．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．

文献[10][11]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る．また[6][12]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が提案されている[13][14]．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[15]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[25]．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[15]で評価した結果，L77 系アルゴリズム[26]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と考える．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しており，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[16]では，CAN メッセージにパケットカウンタと MAC(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[17]．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，データの配信効率は低下する．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[18]．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．

© 2020 Information Processing Society of Japan

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．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[20]．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する 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[21]．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配信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[22]．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[23]．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の製造者により作成されていないソフトウェアで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[24]．シミュレーションにより，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．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 の主流たる 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．

3. 車載 ECU ソフトウェア更新

3.1 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 OEM(部品メーカ）にて不具合や脆弱性に対する対策など，各 ECU 向けに修正をする．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．出荷元では，ECU ごとに次に示す情報はわかっている．

- 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上でリリースする．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時に，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．このようなソフトウェア更新システムを前提とする．

3.2 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．このようなフローを実行する．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良いが，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．まずデータは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

© 2020 Information Processing Society of Japan

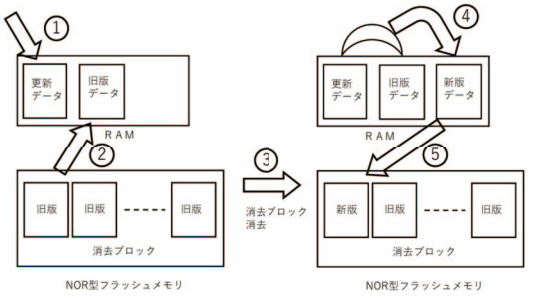

図 2. NOR 型フラッシュメモリの差分書き換え

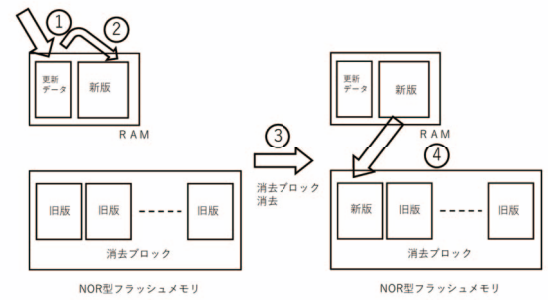

図 3. NOR 型フラッシュメモリの単純書き換え

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. フラッシュメモリの消去はブロックまとめてであるが，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．

多数研究されている[27][28]．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．

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．その基本となるのは，ハフマン符号化[29]および LZ77[26]や，LZ78[30]である．原理としては，既に出てきたデータ列の並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[15]で比較評価をした結果，車載 ECU へのソフトウェアの配布においては，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[31]は消費するメモリ量は小さいが，圧縮率が LZ77 系に比べ良くない. 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，Zstandard[32]を車載 ECU のソフトウェア更新向けに改良することとする．

4.2 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ていく．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利用する方式である．**図 4** に提案方式の概要を示す．旧版は，通常に圧縮する．繰り返しパターンは，どこからどの長さという情報に置き換えられることにより圧縮される．どこからどの長さのオリジナルの部分が辞書となる．この辞書部分と置き換え部分を合わせることにより圧縮データとなる. 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．提案方式はこの辞書にあたる部分を出荷する ECU のフラッシュメモリに配置しておく．圧縮時には旧版で使った辞書を利用してデータの繰り返し部分を表現しておき，圧縮展開時には，フラッシュメモリ上にあらかじめおいてある辞書(以下，プリセット辞書と呼ぶ)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 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 ベースの圧縮をする．圧縮をしない形で，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にて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．この時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 上に作成の上，ECU のフラッシュメモリに保存する．

図 6 には，車両の ECU 上での動作例を詳しく説明する．消去ブロックサイズに比べ RAM に十分な容量がない場合であるため，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

© 2020 Information Processing Society of Japan

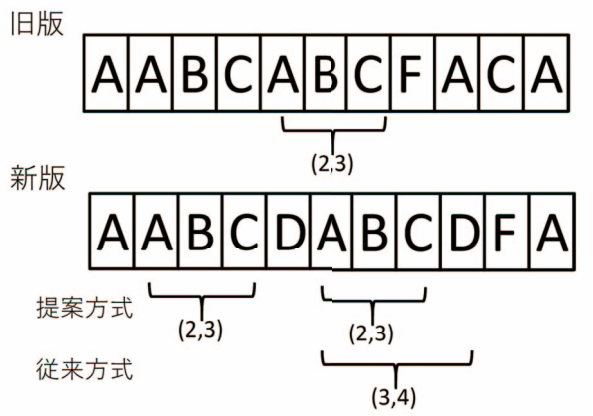

図 4. 圧縮辞書

に書きこむ.

(4) (3)の処理終了後, 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む. この繰り返しの後, 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.

(5) 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にて確認する.

図 6 は例であるため, 適宜正しく書き込みを確認するための処理を(1)から(3)の中に入れる場合もある.

5. 実験

実際にマイコンボードに書き込む用のバイナリファイルを対象とする. Toppers[33]の APS カーネル, バージョン 1.9.0 ~ 1.9.3 までの 4 つを用いてバイナリファイルをそれぞれビルドする. バージョン毎のデータサイズを**表 1** に示す. プロセッサは固定長命令の ARM 向け, Cortex-M4 とする.

1.9.0 を初期バージョンとし, バージョンアップによってソフトウェア更新される度に初期バージョン用に作成したプリセット辞書がどの程度有用であるか実験する. 過去の辞書を用いても圧縮率が悪くなる可能性として, ソースコードの変更による影響でレジスタの使用方法が変わる場合や, 機械語の命令列の変更によって参照アドレスが変わる場合がある.

実験は以下の 2 つをする. また圧縮には Zstandard を用いる.

- 通常圧縮とプリセット辞書を用いた圧縮率の比較
- プリセット辞書を用いた複数回バージョンアップ

6. 評価

6.1 圧縮率の評価

通常圧縮とバージョン 1.9.0 を対象に生成した, プリセット辞書を用いた圧縮の 2 つを**表 2** に示す. またプリセット辞書のサイズは 31,304bytes であった.

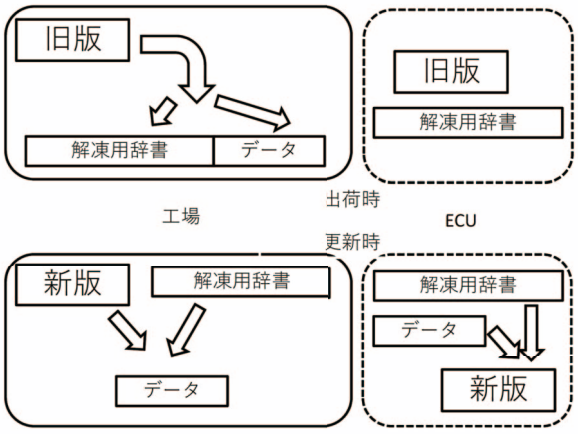

図 5. ECU のソフトウェア更新フロー

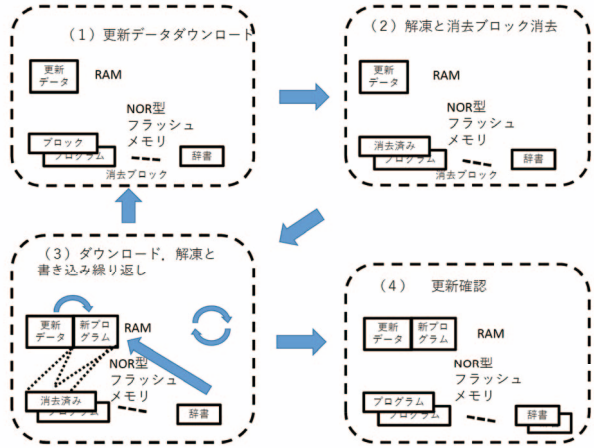

図 6. ECU のフラッシュメモリ更新

通常圧縮と比べ, プリセット辞書を用いた圧縮はとても圧縮率が良い. しかしプリセット辞書のデータサイズが 31,304bytes と元のデータサイズよりも大きいため, 通常圧縮と比べて一概に優れているとは言えない. またバージョン 1.9.1 と 1.9.2 の間で圧縮効率が悪くなったことが見受けられる. これはデータサイズが増えたため, それによってレジスタの使い方や参照アドレスの変更によるものと考えられる.

この実験より, バージョンアップデートによる差異が少ない場合には, プリセット辞書を初期から導入することで, 後々のデータサイズをかなり減らせることがわかる.

© 2020 Information Processing Society of Japan

表 1. 評価対象のデータ

バージョン	データサイズ(bytes)
1.9.0	28,742
1.9.1	28,742
1.9.2	28,746
1.9.3	28,746

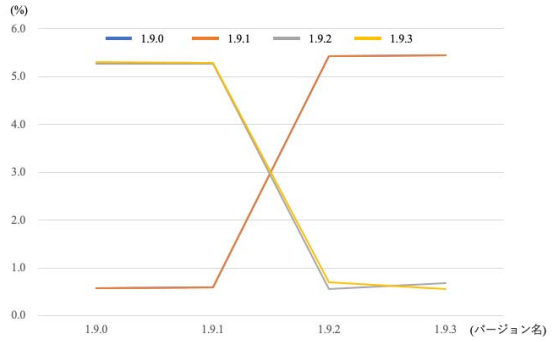

図 7. 辞書の更新点

6.2 複数回バージョンアップ

6.1 よりプリセット辞書は圧縮率を高めてくれることがわかった．しかしバージョン 1.9.1 から 1.9.2 へバージョンアップした際には僅かではあるが圧縮率が落ちている．ここではどのバージョンでプリセット辞書を更新するか評価をした．**図 7** に結果を示す．横軸がそのバージョンでビルドしたバイナリファイル，縦軸が圧縮率，4 つの折れ線がそのバージョンで生成したプリセット辞書となっている．

大きな変化はバージョン 1.9.1 と 1.9.2 の間にしかなく，1.9.0 と 1.9.1，1.9.2 と 1.9.3 がそれぞれ同じような動きとなった．

この実験より，本論文で対象としたバイナリファイルでは圧縮率が大きく悪化することはなかった．ファイルサイズが小さい，即ち，前バージョンとの変更点が少ない場合はプリセット辞書のデータサイズが大きいため，頻繁に更新することは非効率であり，初期のプリセット辞書を使用し続けることに問題がない結果となった．

7. まとめ

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．

通常圧縮とプリセット辞書を用いた圧縮の比較をすることでどの程度の変更点で，どの程度の圧縮率へ影響を及ぼすか評価した．さらにバージョンアップする毎に初期に使用していたプリセット辞書を用いた圧縮は効果を発揮し

表 2. 圧縮率比較

バージョン	通常圧縮(%)	辞書圧縮(%)
1.9.0	62.463	0.574
1.9.1	62.463	0.591
1.9.2	62.464	5.423
1.9.3	62.468	5.451

にくくなるため，どのような場合にプリセット辞書を更新するべきか評価した．

今後の課題として，大きな変更点があるバージョンアップを行う際の影響を図るため，より変更点がある評価が必要であると考える．また複数回バージョンアップでもバージョン間の変更点が少なく，データサイズが小さかったため，変更点が大きいデータの実験が必要である．

参考文献

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

[2] Amara Dinesh Kumar, Koti Naga Renu Chebrolu, Vinayakumar R, Soman KP, ”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,” http://www.mlit.go.jp/en/jidosha/vehicle_recall_17.html (accessed 16-March 2020)

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

[5] S. Lorenz, “The flexray electrical physical layer evolution,” SPECIAL EDITION HANSER automotive FLEXRAY, pp.14-16, 2010

[6] 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

[7] D. Bogdan, R. Bogdan and M. Popa, ”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

[8] 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

[9] 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

[10] 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

[11] 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

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

[13] R. Burns, L. Stockmeyer and D. D. E. Long, ”In-place reconstruction of version differences,” IEEE Transactions on Knowle

© 2020 Information Processing Society of Japan

dge and Data Engineering, vol. 15, no. 4, pp.973-984, 2003
[14]T. Nakanishi, H. Shih, K. Hisazumi and A. Fukuda, ”A software update scheme by airwaves for automotive equipment,” International Conference on Informatics, Electronics and Vision, pp.1-6, 2013
[15]Y. Onuma, Y. Terashima, Sumika, Nakamura and R. Kiyohara, ”Compression method for ECU software updates, ”Tenth International Conference on Mobile Computing and Ubiquitous Network (ICMU), pp.1-6, 2017
[16]AUTOSAR, https://www.autosar.org (accessed 2018/1)
[17]AUTOSAR, Requirements on Secure Onboard Communication, https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_SRS_SecureOnboardCommunication.pdf (accessed 2018/1)
[18]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
[19]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, 2011
[20]竹森敬祐, and 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
[21]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
[22]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
[23]Gandhi Kapilan Kulayan Arumugam, Arumugam Chamundeswari, ”An Approach for Secure Software Update in Infotainment System,” Proceedings of the 10th Innovations in Software Engineering Conference(ISEC) pp.127-131, 2017
[24]Zhou Qin, Fei Li, Yi-Huai Wu, Chao Wang, ”New ECU Attestation and Encryption Mechanism for In-Vehicle Communication,” DEStech Transactions on Engineering and Technology Research, (ssme-ist), 2016
[25]J. Uthayakumar, T. Vengattaraman, P. Dhavachelvan, ”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
[26]J. Ziv and A. Lempel, ”A universal algorithm for sequential data compression,” IEEE Transactions on Information Theory, vol. 23, no. 3, pp.337-343, 1977
[27]Marco Steger, Carlo Alberto Boano, Thomas Niedermayr, Michael Karner, Joachim Hillebrand, Kay Roemer, Werner Rom, ”An Efficient and Secure Automotive Wireless Software Update Framework,” IEEE Transactions on Industrial Informatics, vol. 14, no. 5, pp.2181- 2193, 2018
[28]Thomas Chowdhury, Eric Lesiuta, Kerianne Rikley, Chung-Wei Lin, Eunsuk Kang, BaekGyu Kim, Shinichi Shiraishi, Mark Lawford, Alan Wassyng, ”Safe and Secure Automotive Over-the-Air Updates, ” Gallina B., Skavhaug A., Bitsch F. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2018. Lecture Notes in Computer Science, vol 11093. Springer, Cham
[29]D. A. Huffman, ”A Method for the Construction of Minimum-Redundancy Codes,” Proceedings of the IRE, vol. 40, no. 9, pp.1098-1101, 1952
[30]J. Ziv and A. Lempel, ”Compression of individual sequences via variable-rate coding,” IEEE Transactions on Information Theory, vol. 24, no. 5, pp.530-536, 1978
[31]Gage P., “A New Algorithm for Data Compression,” The C Users Journal, Vol.12, No.2, pp.23-38, 1994
[32]Yann Collet, ”Zstandard - Real-time data compression algorithm” http://facebook.github.io/zstd/, 2015
[33]Toppers，https://toppers.com (accessed 2020/5)

© 2020 Information Processing Society of Japan