一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
ICSS2016-39 (2016-11)

車載ネットワークの構成証明に向けた ECU のソフトウェアベース改竄検知方式の検討

小山　卓麻† 　高橋　順子† 　星野　文学† 　田中　政志†

† 日本電信電話株式会社 NTT セキュアプラットフォーム研究所
〒 180–8585 東京都武蔵野市緑町 3–9–11
E-mail: †{koyama.takuma,takahashi.junko,hoshino.fumitaka,tanaka.ma}@lab.ntt.co.jp

あらまし　本稿では車載ネットワークに接続する制御機器（ECU）のソフトウェアの改竄検知方式を提案する．車外ネットワークと通信するコネクテッドカーおよび自動運転技術の研究開発が進む中，車載ネットワークに対する脅威として遠隔からの ECU のソフトウェア改竄が指摘されている．提案方式では既存方式が見逃しうる改竄を検知可能とするために ECU のメモリの空き領域に検証領域を拡張した．ソフトウェアがメモリサイズの約 9 割以上を占める条件下では既存方式より高速に検証可能である．

キーワード　自動車セキュリティ，改竄検知，ECU，構成証明，車載ネットワーク

Software-based Modification Detection of ECU for Automotive Network System Attestation

Takuma KOYAMA†, Junko TAKAHASHI†, Fumitaka HOSHINO†, and Masashi TANAKA†

† NTT Secure Platform Laboratories, NTT Corporation
Midori-cho 3–9–11, Musashino-shi, Tokyo, 180–8585 Japan
E-mail: †{koyama.takuma,takahashi.junko,hoshino.fumitaka,tanaka.ma}@lab.ntt.co.jp

Abstract　This paper proposes a software-based modification detection of the software of Electronic Control Units (ECU) in automotive networks. Many papers report vulnerabilities and risks of automotive networks with recent developing of out-vehicle connectivity and autonomous driving technologies. This paper deals with the issue of remote malicious modifications against ECU. We expand the scope to check to the whole of ECU memory which space written as random numbers. Our proposal can execute more rapidly than the previous study under a certain condition.

Key words　Automotive Security, Modification Detection, ECU, Attestation, In-vehicle network

1. はじめに

現代の自動車はエンジン制御や舵角操作といった自動車本来の機能からナビゲーションや外部通信といった情報システムまで，各機能を Electric Control Unit (ECU) というマイコンを多数用いて電子制御している．そして ECU 同士が通信することでより高度な制御を実現する車載ネットワークシステムが構築されている．車 1 台の ECU 搭載個数ならびに ECU に実装されるソフトウェアのコード総量は増大し続けており，高級車種では 1 台あたり 100 個以上の ECU が搭載され，ソフトウェアのコード総量は 1 億行以上と言われている [1], [2]．そのため ECU のソフトウェアのバグや脆弱性を完全に排除して出荷することは困難となりつつある．他方，地図データの更新や自動運転など高度なサービスを実現するために，通信キャリア網のデータ通信や近距離無線通信などを用いて自動車内外のネットワークが連携するコネクテッドカーと呼ばれる技術・システムが近年盛んに研究開発されている．車載ネットワークの不正動作に繋がりうる脅威として従来からアフターマーケットにおける ECU の交換やアマチュアレース用の ECU のソフトウェアの書き換え，想定外の機器のシステムへの取り付けが指摘されてきた．そしてコネクテッドカーの到来により，ソフトウェアのバグや脆弱性を悪用した遠隔からの不正な制御命令の実行やソフトウェア改竄の脅威も指摘されるようになった [3]〜[5]．

コネクテッドカーの安全性と利便性を高めるために，ECU のソフトウェアのバグや脆弱性の修正あるいは改良を円滑に行う無線配信による ECU のソフトウェアアップデート機能（リ

モートリプログラミング）が検討されている [6]．リモートリプログラミングが実現すれば，ソフトウェアに関する大規模なリコールが発生しても従来のように技術者が専用機器を車 1 台ずつ整備配線に結線して書き換える手間が不要となる．しかしリモートリプログラミング機能自体にバグや脆弱性が存在するとそれもまた攻撃者に悪用される危険性がある．このため攻撃が起きた場合に対応するための検知技術が要請される．検知の一手法として車載ネットワークシステム全体の完全性を検証する技術 (構成証明) が挙げられる．車載ネットワークシステムを構成する ECU の完全性を脅かす攻撃に着目すると，ECU のソフトウェアの改竄，物理的な接触を伴う交換，追加，除外が挙げられる．これらの攻撃に対して，構成証明の結果を車外のしかるべき者が確認することができれば，コネクテッドカーに対するサイバー攻撃や不正改造の早期発見が可能になる．

前述の ECU に対する物理的な接触を伴う攻撃は，人間が車輌 1 台ずつ攻撃する必要があるため攻撃可能な台数が遠隔攻撃ほど多くないと考えられる．また ECU に接触するためには車輌を一部分解しなければならない．そのため所有者以外による物理的攻撃のハードルは遠隔攻撃時よりも高いと言える [7]．一方，ECU のソフトウェアの改竄は遠隔から多量の車輌に対して行われる畏れがある．また搭乗者の安全を確保する上で，自動運転など高度な運転支援機能のソフトウェア制御による信頼性がアクチュエータなど機器の物理的信頼性と比較して，以前よりも重要になってきている．そのためコネクテッドカー時代において，構成証明の中でも ECU のソフトウェアの遠隔改竄を検知する技術の確立は重要な課題である．

そこで本稿ではコネクテッドカー時代を見据え，構成証明の重要な一要素である ECU のソフトウェアの遠隔改竄に対する検知方式を提案する．提案方式は，既存方式では発見できないソフトウェアの改竄を検出可能とするために ECU の CPU メモリの空き領域に乱数をパディングして検証領域を拡張した．ソフトウェアサイズがメモリサイズの約 9 割以上を占める条件下では既存方式より高速に検証可能である．なお本稿において，メモリとは ROM や Flash メモリといったソフトウェアが記憶される不揮発な記憶装置を指す．

2. 構成証明の関連・既存方式

2.1 TPM を用いたセキュアブート＋認証鍵共有方式 [8]

ECU の不正な物理的交換，ソフトウェアの改竄，車載ネットワーク内のメッセージのなりすまし攻撃を防ぐために，ECU へのセキュアブート機能の搭載および ECU 間認証鍵共有を行う方式が提案されている．全 ECU に TCG [9] が規格化した耐タンパ性を有するセキュリティチップ Trusted Platform Module（TPM）の搭載を要件とする．方式の概要として，まずエンジン始動時に全 ECU が自身の TPM を用いてセキュアブートを実行することによりソフトウェアを検証する．検証後，TPM 内に事前に保管されたマスタ秘密鍵を用いて ECU 間で認証およびセッション鍵共有する．共有鍵を用いたメッセージ認証符号（MAC）を車載ネットワーク内の全メッセージに付与する．不正に追加・交換されてなりすまし攻撃を試みる ECU に対して，受信 ECU が随時メッセージを検証しこれを検知する．

2.2 チェックサム関数 ICE を用いた改竄検知＋認証鍵共有方式 [10]

耐タンパ機器を搭載しない ECU のソフトウェアの改竄を検知するために，耐タンパ機器を搭載する検証 ECU が被検証 ECU を認証し，鍵共有を行う方式が提案されている．本方式では攻撃者による ECU への物理的な接触を伴う攻撃（改竄したい ECU のメモリを増設する，不正機器を取り付けるなど）は対象外としている．図 1 に本方式のプロトコルの一部を示す．実際には認証と鍵共有を同時に行うが，簡単のためにソフトウェアの改竄検知に寄与する認証および通信のみ抽出した．前提として，検証 ECU はレスポンスの期待値を算出するために被検証 ECU のソフトウェアデータを記憶している．そして被検証 ECU は自身のメモリに記憶されたソフトウェアをチェックサム関数 Indisputable Code Execution（ICE）[11], [12] と暗号学的ハッシュ関数に入力してレスポンスを生成する（図 1–4, 11 行）．図 2 に ICE の擬似コードを示す．ICE ではメモリのソフトウェアだけでなく CPU のプログラムカウンタやステータスレジスタの値をレスポンス生成に利用することで，ソフトウェアのデータの改竄だけでなく，データが別のアドレス番地に書き移されていないかまで検証できる（図 2–18 行）．メモリの空き領域にマルウェアが書き込まれ，元のソフトウェアが被検証 ECU 内に残されていれば ICE の正しい計算結果をマルウェアも算出可能である．このため ICE では計算結果だけでなく ICE 自体の処理時間，応答時間も改竄検知に利用する．マルウェアが ICE を偽装するためには少なくとも 1CPU サイクル以上 ICE よりも多く計算が必要になるように，CPU のアーキテクチャに合わせて ICE の実行コードを最適化し最小 CPU サイクル数で実装する．この CPU サイクル数の差を応答時間の長短から検証 ECU が検出できれば，マルウェアによる偽証を検知できる．そのため ICE の最小サイクル数で実装し，ループ回数 （図 2–7 行）と通信路の遅延を考慮した応答制限時間（図 1–8 行）を適切に設計する必要がある．

2.3 関連・既存方式の問題

まず方式 [8] は，自動車が使われる過酷で高負荷な環境で耐タンパ性を有しかつ長年性能を維持する機器を全 ECU に搭載するため製造導入コストが高いという問題がある．TPM を搭載すれば物理的接触を伴う攻撃を検知可能な場合はあるが，前述のとおり物理的攻撃は攻撃者のコストが高い割に効果範囲が狭い．さらに，例えば搭乗者および車輌の周囲に対する物理的危害が攻撃の目的であれば，駆動部品を不良品に交換したりボルトを緩めたりする方が ECU に接触して交換したり改竄したりするよりも容易である．

次に，被検証 ECU に TPM の搭載を要しない方式 [10] では以下 3 点の問題が挙げられる．1 点目は安全性に関して，検知できない改竄が存在する点である．まず，ICE およびハッシュ関数はメモリ内に正規のソフトウェアが格納されている領域だけを入力とする（図 1–4，11 行）．このためメモリの空き領域に不正コードが記憶されていた場合本方式では検出できない．正規のソフトウェアにバッファオーバーフローなどの脆弱性が

1:	V :	$r \xleftarrow{R} \{0,1\}^{128}$
2:	$V \to P$:	r (Challenge)
3:	V :	T_1 = Current time
4:	P :	$C \leftarrow ICE_P(r, memory_P) \in \{0,1\}^*$
5:		C' = truncated C into 64 bits
6:	$P \to V$:	C' (Response)
7:	V :	T_2 = Current time
8:		Verify $(T_2 - T_1) <$ ICE threshold
9:		Verify C' by recomputing $ICE_P(r, memory_P)$
10:		Abort if incorrect
11:	P :	$H_{mem} \leftarrow Hash(memory_P) \in \{0,1\}^{**}$
12:		H'_{mem} = truncated H_{mem} into 64 bits
13:	$P \to V$:	H'_{mem} (Response)
14:	V :	Verify H'_{mem} by recomputing $Hash(memory_P)$
15:		If H'_{mem} equals to expected value then stop

図 1　チェックサム関数 ICE を用いた改竄検知方式 [10] の一部抜粋．V：検証 ECU，P：被検証 ECU，*：被検証 ECU の汎用レジスタ数に依存する（MSP430 [13] の場合は 128-bit），**：ハッシュ関数に依存する（SHA-1 の場合は 160-bit）．

1:	Input: memory, $r = r_0 \| \| r_1 \| \| ... \| \| r_{n-1}, r_i \in \{0,1\}^*$
2:	Output: $C = c_0 \| \| c_1 \| \| ... \| \| c_{m-1}, c_i \in \{0,1\}^*$
3:	Variables: $[code_{start}, code_{end}]$: Verified memory area
4:	$daddr$: Address of current memory access
5:	b : Content of memory in $daddr$
6:	x : Variable for random walk for memory
7:	y : Times of loop decided as § 3.4
8:	PC : Content of Program Counter register
9:	SR : Conten of Status Register
10:	// Initialize valiables:
11:	If $m > n$ then, $r_{n+1} \leftarrow r_a \oplus r_b, ..., r_{m-1} \leftarrow r_i \oplus r_j \oplus ...$
12:	$c_0, c_1, ..., c_{m-1} \leftarrow r_0, r_1, ..., r_{m-1}$
13:	$x \leftarrow r_0 \oplus r_1 \oplus ... \oplus r_{n-1}$
14:	// ICE loop:
15:	for $l = y$ to 0 do
16:	$x \leftarrow x + (x^2 + 5) \bmod code_{end}$
17:	$daddr \leftarrow (daddr \oplus x) \wedge MASK + code_{start}$
18:	$c_j \leftarrow c_j + PC \oplus b + l \oplus c_{j-1} + x \oplus daddr + SR \oplus c_{j-2}$
19:	$c_j \leftarrow$ bitwise_left_rotate$(c_j, 1)$
20:	$j \leftarrow j + 1 \bmod m$
21:	end for

図 2　チェックサム関数 ICE の擬似コード．*：被検証 ECU の CPU レジスタ長．||：連結（Concatenation）を表す．

存在した場合，検証領域外の不正コードが実行される畏れがある．次に，ハッシュ関数の入力にチャレンジ（乱数）を用いないため出力が固定値である（図 1–11 行）．攻撃者はハッシュ値を事前計算できるため，リプレイ攻撃に対する改竄検知は実質的に ICE の出力によって確認することになる．また ICE は加算を用いて値を更新しているが，最上位ビットの桁上がりを考慮していない（図 2–18 行）．このため本来あるべきアドレスと最上位ビットだけが異なるアドレスに移されたとしても確率 1/2 で ICE の出力は同じ値になるため改竄を見逃す畏れがある．2 点目は方式の適用性に関して，本方式を適用できる CPU が限定される点である．プログラムカウンタの読み出しをソフトウェア（ICE）に許可しないプラットフォームでは ICE を実装できない．例えば Atmel 社製 Automotive AVR Family [14] や MIPS Technologies 社の MIPS アーキテクチャ [15] では実装できない．3 点目は安全性および設計コストに関して，ICE を CPU 毎に最適化しなければならない点である．自動車は 1 台あたり数十から百数十の ECU が搭載されており，制御内容によって ECU に搭載される CPU の性能は多様である．このため設計した ICE が最小サイクル数であること，ICE を偽証する同サイクル数のアルゴリズムが存在しないことを，被検証 ECU の CPU 毎に検証しなければならない．さらに現代の自動車において利用されている車載通信方式のメッセージ遅延に関する特徴から，ICE の適切なループ回数は車輌を設計する度に車載ネットワークの通信全体をシミュレーションしてみないと決定できない（3.2 節および 3.4.2 項を参照）．

3. 提案方式

3.1 方　　針

前述の関連・既存方式の問題点を解決するにあたり，提案方式では以下の方針をとった（図 3 を参照）．まず攻撃の容易さと影響範囲の観点，そしてコネクテッドカー時代におけるソフトウェア制御の重要性の観点から，本稿の脅威対象を遠隔によるソフトウェアの改竄に注目する．つまり TPM などの物理的攻撃を検知することに用いる製造コストが増加する機器を被検証 ECU には搭載しないこととする．被検証 ECU は自身でソフトウェアの完全性を検証するのではなく，車内の信頼の基点となる別の ECU と通信して完全性を証明する．

既存方式 [10] の問題点に対して次の方針をとった．まず，被検証 ECU の空き領域への不正コードの書き込みを検知するために，メモリ領域全体を検証範囲とする．通常，メモリの空き領域には 0xFFFF... などの固定値が連続して書き込まれているか，そもそも定義されていない．一方本方式では，ソフトウェア本体を書き込んだ後のメモリの空き領域に乱数をパディングして書き込み，ソフトウェア本体と乱数部を合わせたメモリ領域全体を検証範囲とする．次に，CPU に依存せずに実装可能とするために，ECU 用共通ソフトウェアアーキテクチャ AUTOSAR [16] が規定している暗号学的関数の中で比較的軽量な処理である乱数生成器とハッシュ関数を用いて実現する．メモリ全体に検証領域を拡張したことにより検証のレスポンス生成（ハッシュ値計算）に処理時間の大部分が割かれることになる．リプレイ攻撃耐性を有しながらもハッシュ関数の処理時間を少しでも短くするために，検証 ECU のチャレンジ（乱数）はハッシュ関数の入力としてではなく，ハッシュ関数の入力の範囲指定として用いることとした．

3.2 準　　備

攻撃者として以下の目的と能力を仮定する．攻撃者は被検証 ECU のソフトウェアを書き換えて悪意ある任意のコードを実行したい．その前段階として書き換え対象の被検証 ECU に対する検証 ECU の改竄検知を逃れようとする．あるいは改竄検知のチャレンジに対する正しいレスポンスを偽装して検証を誤

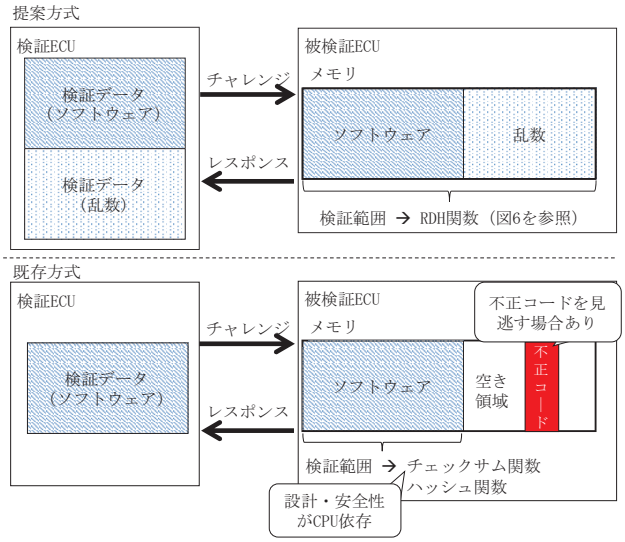

図 3　提案方式の概略と既存方式の問題点

認させようとする．攻撃者は以下の能力を有し，いずれも既存方式 [10] と同等の能力である．

（1） 被検証 ECU の物理的特徴（マイコンのクロック周期やメモリサイズなど）を知っている．

（2） 被検証 ECU が記憶しているソフトウェアのデータ（命令やデータ，秘密鍵など）を知っている．

（3） 任意個数の被検証 ECU のソフトウェアを自由に改竄できる．

（4） 車載ネットワークに正規の ECU 以外の機器 (PC や ECU) は追加できない．

（5） 正規 ECU のメモリを，より大容量なメモリに交換することはできない．

つぎに，車載ネットワークとして ECU 間の通信で採用されている通信方式 CAN [17] の特徴を述べる．CAN は OSI 参照モデルにおける物理層とデータリンク層を規定している．通信速度は 125Kbps から 1Mbps の範囲の固定値で，ペイロードは最大 64-bit である．バス接続型のブロードキャスト方式である．バスにメッセージが流れていないとき任意の ECU が任意のタイミングでメッセージを送信できる．メッセージヘッダの CAN-ID によって優先順位が予め決められており，複数の ECU が同時にメッセージを送信した場合は低優先度のメッセージの送信が一次停止し再送される．高優先度のメッセージを送信する場合でも，すでに低優先度のメッセージが CAN バス上に流れている場合は，低優先度のメッセージが送信完了するまで高優先度のメッセージは送信できない．このため CAN ではメッセージの最大遅延時間が保証されていない．

3.3 方 式 詳 細

図 4 に提案方式の機器構成を示す．

・車載 LAN： CAN に代表される車載ネットワーク方式．検証 ECU は全ての被検証 ECU と車載 LAN を通して直接通信できるものとする．

・検証 ECU： 耐タンパ性を有する機器を搭載する ECU．起動時のセキュアブートにより，自身のソフトウェアおよび記憶

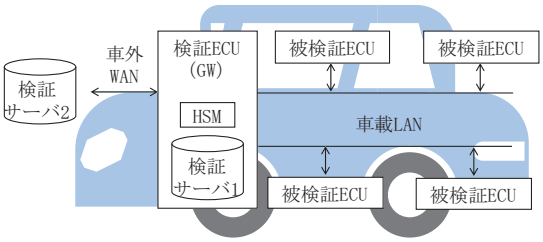

図 4　提案方式のネットワーク構成

する全データの完全性を検証できるものとする．本方式において車載ネットワーク内の信頼の基点となる．車載ネットワークシステム全体を検証するためには全ての被検証 ECU と通信できる必要があるため，ネットワーク構成の観点からゲートウェイ機能（GW）を有する ECU が検証 ECU としての機能を担うことが考えられる．なお提案方式では耐タンパ性を有する機器を TPM に限定せず，Hardware Security Module（HSM）と表記する．

・検証サーバ： 検証サーバ 1 の場合：被検証 ECU に対するチャレンジ（乱数）の生成と，被検証 ECU のメモリに書き込まれた正規のソフトウェアと乱数の記憶，そして検証結果を記憶する機能および機器．検証サーバ機能は検証 ECU が兼ねても良い．検証サーバ 2 の場合：しかし検証 ECU が全ての被検証 ECU のソフトウェアと乱数を記憶できる記憶領域を搭載することはコストが高い．このため本記憶機能を車外の検証サーバによって実現することも可能である．なお検証サーバと検証 ECU 間の通信（車外 WAN）は機器認証やパケットの暗号化が適切に行われ安全であると仮定する．検証サーバ 2 はチャレンジとその期待値を生成して検証 ECU に送信する．検証 ECU は送られてきたチャレンジと期待値を記憶し，そして期待値とレスポンスの比較結果を記憶すれば良い．検証 ECU は比較結果を検証サーバに送信する．検証 ECU が記憶するデータ総量はチャレンジと被検証 ECU の個数倍の期待値程度である．仮に被検証 ECU が 100 個，チャレンジと各期待値の長さが 64-bit であればデータ総量は高々 1KB である．

・被検証 ECU： 車輌の各機能を制御する ECU．耐タンパ性を有する機器を搭載しない．メモリにはソフトウェア本体と，ソフトウェア以外の領域には乱数が書き込まれている．

次に図 5， 6 に提案方式のプロトコルを示す．図 5 は被検証 ECU と検証 ECU（と検証サーバ 1）との 1 往復のチャレンジ・レスポンス検証である．CAN のペイロード長に合わせて，チャレンジおよびレスポンスのサイズを 64-bit とした．図 6 は Randomly Dimidiated Hash（RDH）関数を示す．RDH 関数は被検証 ECU と検証 ECU（の検証サーバ部 1）とがそれぞれチャレンジに従い被検証 ECU のメモリ領域全体のハッシュ値を計算する．RDH 関数はメモリ全体をチャレンジによって指定されたアドレス位置で 2 分割して，それぞれのアドレス領域のハッシュ値を計算する．RDH 関数はチャレンジに従いハッシュ関数に対する 2 回の入力開始アドレス位置と終了アドレス位置を変更するため，固定値であるメモリ領域全体を入力としているにも関わらず RDH 関数の出力は毎回変化する．

1:	V :	$r_0, r_1 \xleftarrow{R} \{0,1\}^{32}$
2:		$r \leftarrow r_0 \| r_1$
3:	$V \rightarrow P$:	r (Challenge)
4:	V :	$T_1 =$ Current time
5:	P :	$res_0, res_1 \leftarrow RDH(memory_P, r)$
6:		$res'_0, res'_1 =$ Truncated res_0, res_1 into 32 bits
7:		$Res \leftarrow res'_0 \| res'_1$
8:	$P \rightarrow V$:	Res (Response)
9:	V :	$T_2 =$ Current time
10:		Verify $(T_2 - T_1) <$ Allowed time
11:		Abort if incorrect
12:		Verify Res by recomputing $RDH(memory_P, r)$
13:		Store the result of verification (line.12)

図 5　提案方式のプロトコル．V：検証 ECU および検証サーバ，P：被検証 ECU．

1:	Input: memory, $r = r_0 \| r_1$
2:	Output: $res_0, res_1 \in \{0,1\}^*$
3:	Variables: $[addr_{start}, addr_{end}]$: Both ends of memory
4:	$mem_{[a,b]}$: Content of memory in $[a, b]$
5:	// Truncate inputs within the range of memory
6:	if $r_0, r_1 \geqq addr_{end}$ then, $r_0, r_1 \leftarrow r_0, r_1$ mod $addr_{end}$
7:	if $r_0 > r_1$ then, swap the values of r_0 and r_1
7:	// Compute responses
8:	$res_0 \leftarrow Hash(mem_{[r_0, r_1]})$
9:	$res_1 \leftarrow Hash(mem_{[r_1+1, addr_{end}]} \| mem_{[addr_{start}, r_0-1]})$

図 6　Randomly Dimidiated Hash（RDH）関数

検証サーバ 2 を用いた機器構成の場合は，図 5：1 行目の乱数生成および 12 行目の期待値生成を検証サーバ 2 が行い，検証 ECU に対して安全な経路で事前に送信しておく．検証 ECU は例えば走行開始前のエンジンを点火した際に，被検証 ECU と 3–13 行目を実施し，11 行目あるいは 13 行目の結果を検証 ECU 自身で保管するとともに検証サーバ 2 に対して返送する．検証 ECU は検証結果に従い安全な制御，対処をとる．例えば改竄を検知した場合は発進させない，徐行運転だけを許可する，検証サーバ 2 を通じてしかるべき者（自動車・部品メーカなど）に対して通知するなどの対処が考えられる．

3.4 性能評価

後述のとおり，提案方式は既存方式が検知可能な攻撃および既存方式が検知できない攻撃を検知可能である．また，ソフトウェアサイズが ECU のマイコンのメモリサイズの約 9 割を占める場合は既存方式より高速に検証可能である．

3.4.1 安全性

既存方式で検知可能なソフトウェアの改竄およびソフトウェアのデータが別のアドレス番地に移される改竄は，RDH 関数の出力から検知可能である．またメモリ全体にソフトウェアあるいは乱数が書き込まれているため，マルウェアを書き込む空き領域は存在しない．提案方式による検知をすり抜けようとする，あるいはレスポンスを偽証しようとする攻撃として以下の方法が挙げられる．

・事前計算・リプレイ攻撃：　RDH 関数を事前計算することは可能であるが，すべてのチャレンジに対して事前計算を行うことは総計算時間が膨大で困難であり，またその値全てを ECU 内に記憶しておくことは ECU の記憶容量の限界から困難である．チャレンジは乱数であるからレスポンスは毎回変わるため，リプレイ攻撃は困難である．

・ソフトウェアのデータ圧縮による領域確保攻撃：　ソフトウェアは符号によってある程度データサイズを圧縮できる．圧縮した符号データに改竄することで攻撃者は RDH 関数の入力となる正規のコードと乱数を ECU のメモリに記憶しながら，空いた領域に攻撃コードを書き込める．ただしこの攻撃では RDH 関数を計算するために圧縮したコードを RAM 領域に展開しなければならない．提案方式では被検証 ECU がレスポンス生成にかかる処理時間がほぼ一定であるため事前に予測できる．応答制限時間を適切に設定することでコードの展開による不正を検知可能である．

なお，既存方式と同様に提案方式はハッシュ関数に安全性の根拠を置いているため，ハッシュ関数の脆弱性やハッシュ値の衝突確率に依存して改竄を見逃してしまう場合がある．

表 1　提案方式と既存方式の処理速度．および提案方式の方が高速に検証可能なソフトウェアサイズ X, Y の条件

マイコン	提案方式	既存方式 [10]	条件|メモリサイズ
MSP430	44M cycle	5M + 920X cycle	$X > 42.1$ | 48 KB
TC1797	547M cycle	59M + 136Y cycle	$Y > 3.57$ | 4 MB

3.4.2 処理速度

表 1 に，提案方式と既存方式において被検証 ECU がレスポンスを生成するために必要な処理時間を CPU サイクル数を元に机上計算により比較した値を示す．対象は既存方式が速度評価に用いたマイコン，Texas Instruments 社製 16bit マイコン MSP430（16MHz，32KB ROM）[18] および Infineon 社製 32bit マイコン TC1797（180MHz，4MB ROM）[13] である．マイコンのメモリサイズに対してソフトウェアサイズが MSP430 の場合約 88%を超えた場合，また TC1797 の場合約 89%を超えた場合，提案方式が既存方式よりも高速に検証可能である．なおハッシュ関数の処理速度は，AUTOSAR が規定する暗号アルゴリズムに関する自動車向けマイコンにおける実装評価を参照した [19]．

各方式の処理速度を算出する．まず提案方式では，RDH 関数が入力とするデータ量はメモリサイズと等しい．このため提案方式で被検証 ECU がレスポンスを生成するのに要する時間はほぼ一定である．既存方式はハッシュ関数として SHA-1 を用いており，提案方式においても SHA-1 を用いた場合，MSP430 において RDH 関数にかかるサイクル数は約 44Mcycle(= 920.02[cycle/Byte] × 48[KB])，処理時間は約 2.7sec(= 44[Mcycle] × 16[MHz]) である．同様に TC1797 においてサイクル数は約 547Mcycle(= 136.74[cycle/Byte] × 4[MB])，処理時間は約 3.04sec (= 547[Mcycle]/180[MHz]) である．一方既存方式では，ソフトウェアサイズによって SHA-1 の処理時間が変化する．ソフトウェアサイズを X, Y[Byte] と置くと，MSP430 では $920.02X$[sec]，TC1797 では $136.76Y$[sec]

かかる．次に CPU の性能と通信遅延時間によって ICE の計算量が決まる．MSP430 における ICE の最小サイクル数実装は示されているが，TC1797 における ICE の最小サイクル数実装は示されていない [10], [12]．また車載ネットワークにおける最大通信遅延時間が既存方式では言及されていない．これらの値によって ICE のループ回数が異なるため，本稿では以下の仮定を置く．まず，TC1797 の ICE においても MSP430 と同一の 1 ループ 32cycle を要するものとする．なぜならば MSP430 の ICE の機械語実装に用いられている各命令は TC1797 でも実装されている．またマイコンのレジスタ長に合わせて ICE が設計されるためマイコンのデータバス幅や ALUbit 幅の増加が cycle 数の減少には直接結びつかない．そのためほぼ同程度の計算コストを要すると考えられる．次に最大通信遅延時間について，前述の通り CAN では最大通信遅延時間は保証されないため，ベンチマークを用いた検証結果における遅延時間を最大通信遅延時間として仮定する [20]．現在一般的に用いられている CAN の通信速度 500Kbps の場合，優先度が高い CAN-ID のメッセージにおいて 1 メッセージあたり最悪約 1.3msec の通信遅延が発生する．ICE の必要ループ回数は不等式 $n \geqq 2(c \times a/o)$ から導かれる [21]．それぞれ n[times]：ICE のループ回数，c[Hz]：マイコンのクロック周期，a[sec]：総遅延時間，o[cycle]：ICE の偽証に要する 1 ループあたりの余計な CPU 命令数，である．CAN は 1 メッセージのペイロード長が最大 64-bit である．ICE による検証のために合計 4 メッセージの通信を行うから，$a = 5.2$msec である．なお，図 1：1 行において 128-bit，6 行において 64-bit メッセージを送信しているため一見して合計 3 メッセージに見えるが，鍵共有のために方式全体では 6 行目において合計 128-bit メッセージを送信する．そのため合計 4 メッセージの遅延を考慮する必要がある．また $o = 1$cycle である [12]．以上より，ICE およびハッシュ関数による総処理時間は MSP430 で約 $5,324,800 + 920.02X$[cycle]，TC1797 で約 $59,904,000 + 136.74Y$[cycle] である．

つまりソフトウェアサイズが MSP430 において約 $X >$ 42.1KB（メモリサイズの約 88%），TC1797 において約 $Y >$ 3.57MB（メモリサイズの約 89%）のとき，提案方式の方が高速にソフトウェア全体を検証できる．

4. おわりに

本稿では構成証明の重要な要素である ECU のソフトウェアに対する遠隔改竄の検知方式を提案した．本方式の実現により，既存方式では検出できないソフトウェアの改竄を検知することが可能であり，また車外のしかるべき者が ECU のメモリに書き込まれているデータの完全性を確認することができる．今後は方式の実機による実装評価をもって提案方式の正確な性能を検証する．また検証に要する時間の短縮に向けて方式の改善を図りたい．そして物理的な交換などに対するソフトウェアベースの構成証明の検討を進めたい．

文　献

[1] 服部雅之，“自動車のディペンダビリティ設計と VLSI への要求，” JST CREST「ディペンダブル VLSI システムの基盤技術」研究領域平成 19 年度ワークショップ，pp.1–42，Dec. 2007.

[2] http://www.informationisbeautiful.net/visualizations/million-lines-of-code/

[3] C. Miller and C. Valasek, “Remote Exploitation of an Unaltered Passenger Vehicle,” Blackhat USA, pp.1–91, 2015.

[4] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, “Experimental Security Analysis of a Modern Automobile,” Proceedings of the 2010 IEEE Symposium on Security and Privacy (IEEE SP ’10), pp.447–462, IEEE Computer Society, Washington, DC, USA, 2010.

[5] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, , and T. Kohno, “Comprehensive Experimental Analyses of Automotive Attack Surfaces,” Proceedings of the 20th USENIX conference on Security (SEC’11), pp.6–6, USENIX Association, Berkeley, CA, USA, Aug. 2011.

[6] 中野学，松本勉，C. Vuillaume，小谷誠剛，自動車の情報セキュリティ ECU・車載 LAN・車外ネットワークの脅威と対策，日経 AutomotiveTechnology（編），日経 BP 社，東京，Dec. 2013.

[7] 高田広章，松本勉，“有識者による「車載組込みシステムの情報セキュリティ強化に関する提言，” Sept. 2013. http://www.ipa.go.jp/files/000034668.pdf

[8] 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，第 2014-ITS-58-8 巻，pp.47–54，情報処理学会，Sept. 2014.

[9] Trusted ComputingGroup. http://www.trustedcomputinggroup.org

[10] D. Bhattacharya, S. Neelakantan, J. Shokrollahi, and L. Hans, “Bootstrapping Trust in Automotive Networks with Different Types of ECUs,” Proceedings of the 12th Embedded Security in Cars Europe (escar Europe 2014), pp.1–31, Hamburg, Germany, Nov. 2014.

[11] A. Seshadri, M. Luk, A. Perrig, L. Doorn, and P. Khosla, “Using FIRE & ICE for Detecting and Recovering Compromised Nodes in Sensor Networks,” Technical Report CMU-CS-04-187, School of Computer Science, pp.1–25, Carnegie Mellon University, Pittsburgh, PA, Dec. 2004.

[12] A. Seshadri, M. Luk, A. Perrig, L. Doorn, and P. Khosla, “SCUBA: Secure Code Update By Attestation in Sensor Networks,” Proceedings of the 5th ACM workshop on Wireless security (WiSe ’06), pp.85–94, 2006.

[13] Infineon Technologies AG, “TC1797 32-Bit Single-Chip Microcontroller Data Sheet V1.3,” 2014.

[14] Atmel, “AVR 8-bit and 32-bit Microcontrollers”. http://www.atmel.com/products/microcontrollers/avr/

[15] M. Processors. https://imgtec.com/mips

[16] AUTomotive Open System ARchitecture, “AUTOSAR”. http://www.autosar.org

[17] International Organization for Standardization, “ISO 11898-1:2015 Road vehicles – Controller area network (CAN) - Part 1: Data link layer and physical signalling,” 2015.

[18] Texas Instruments Incorporated, “MSP430,” 2013. http://www.tij.co.jp/jp/lit/sg/jajb005i/jajb005i.pdf

[19] P. Murvay, A. Matei, C. Solomon, and B. Groza, “Development of an AUTOSAR Compliant Cryptographic Library on State-of-the-Art Automotive Grade Controllers,” In Proceedings of the 10th International Conference on Availability, Reliability and Security, pp.1–9, Aug. 2016.

[20] K. Tindell and A. Burns, “Guaranteeing Message Latencies on Controller Area Network (CAN),” Proceedings 1st International CAN Conference, pp.1–11, 1994.

[21] A. Seshadri, A. Perrig, L.V. Doorn, and P. Khosla, “SWATT: Software-based Attestation for Embedded Devices,” IEEE Symposium on Security and Privacy 2004, pp.272–282, May 2004.