自動車サービス連携のセキュリティアーキテクチャの提案

2008MI011 朝倉 知也 2008MI079 岩井 大

指導教員 青山 幹雄

1. はじめに

現在，自動車の車載システムと通信技術の発達により，自動車はナビゲーションシステムなどの様々な外部サービスと車載システムを連携させることが可能となっている．

本稿では，外部サービス連携のセキュリティに着目し，ユーザから車載システムまでのセキュアな通信を保証するアーキテクチャを提案する．

2. 問題の背景

現在，車載システムと外部サービス間の連携はメーカごとにインタフェース/プロトコルなどの通信方式が異なる．そのためサードパーティサービスの利用が困難である．

この問題を解決するため SOA (Service-Oriented Architecture)[7]を適用し，SOAP や REST (REpresentational State Transfer)などの標準化されたプロトコルを用いるアーキテクチャが提案されている[3]．

3. 研究課題

SOA に基づく車載サービスブローカアーキテクチャでは，外部システムとユーザ間のプロトコルが定義されていない．また，通信時のセキュリティを保証していない (図 1)．

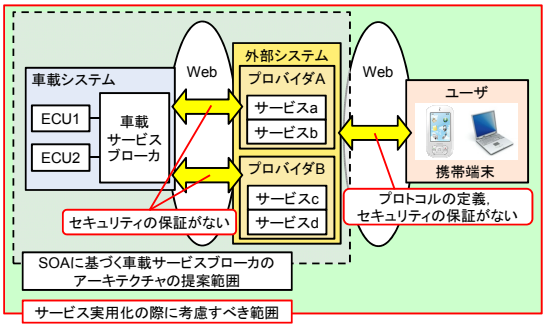


図 1 参考アーキテクチャの提案範囲とその課題

4. 関連研究

4.1. OSGi(Open Service Gateway initiative)

OSGi[9]は機器をネットワークに接続し，相互にサービスを提供するための仕様である．

OSGi は Java で実装され，JVM(Java Virtual Machine)上でミドルウェアである OSGi フレームワークが動作する．OSGi フレームワーク上でソフトウェアコンポーネントである Bundle が複数個動作し，連携することが可能である．

OSGi 仕様では，利用価値の高いサービスを汎用化した Bundle を Basic Service Bundle と呼び，これらを利用することで共通的に使用される機能を開発者が実装する必要がない．それに対し，各事業者や機器に依存し標準化していない Bundle のことを Application Bundle と呼ぶ．

また，OSGi は組み込みシステムの利用が前提で開発されたため省メモリで動作可能であり，システムのリソース制約の影響を受けにくい．

4.2. SOA に基づく車載サービスブローカアーキテクチャ

本アーキテクチャは OSGi を用いた車載サービスブローカを自動車に搭載し，Basic Service Bundle として提供されるプロトコル変換機能を持つ Bundle を利用することで，車載システムと外部サービス間の連携で SOAP/REST を容易に利用できる [3]．

車載サービスブローカの OSGi フレームワーク上では以下の 4 つの Bundle が実装され動作し，Bundle 間連携を行うことで車載ネットワークと外部サービスを連携する．

(1) HTTP 通信機能を持つ HTTP Server Bundle
(2) SOAP/POX メッセージの生成と解析を行う
SOAP/POX Proxy Processor Bundle
(3) テレマティクスサービス機能を持つ Application Bundle
(4) 車載ネットワークと通信を行う Interface Gateway Bundle

4.3. 自動車ネットワークサービスにおけるセキュリティ

自動車ネットワークサービスのセキュリティを保証するためには，自動車に対する攻撃を明確にする必要がある[4]．自動車に対する攻撃概要と具体例を表 1 に示す．

表 1 自動車に対する攻撃概要とその具体例

攻撃発生場所	概要	具体例
近接	整備員などになりすました第三者の車両への直接攻撃	ECUのプログラム改ざん，個人情報の吸い出し
中間 (持ち込み機器)	車内の機器(ドライブレコーダなど)からの攻撃	運転情報の改ざん，漏えい
広域 ネットワーク経由	車外サービス連携の際に発生する外部ネットワークからの攻撃	盗聴による個人情報流出，虚偽メッセージの表示

4.4. 情報セキュリティ

情報セキュリティとは，企業や団体内の情報システム，情報システムを利用して得た情報を正確に運用することである[5]．情報セキュリティの目的は，機密性，完全性，可用性があり，内容や範囲，対象によって達成すべき目的は区分されている．

Web サービスなどを対象とするネットワークセキュリティの対策方法として，不正アクセス防止や秘密鍵を用いた暗号化がある．例として，デジタル証明書を用いる SSL，署名や証明書を用いて SOAP メッセージを暗号化する WS-Security[10]が挙げられる．

5. アプローチ

本稿では，SOA に基づく車載サービスブローカアーキテクチャを拡張し，ユーザから車載システムまでの End-to-End でセキュアな通信を保証する新たな連携方法を提案する．

前提条件として，外部システムはユーザがブラウザを用いて実行する，もしくは Web アプリケーションとし，外部システム自体のセキュリティは保証されているものとする．

ユーザと外部システム間の通信は HTTPS で行い，セキュリティとして SSL を用いる．また，外部システムと車載システム間の通信は SOAP/REST で行い，セキュリティとしてそれぞれ WS-Security/SSL を用いる．異なるセキュリティレベルに応じ，SOAP(WS-Security)と REST(SSL)を選択可能とする(図 2)．

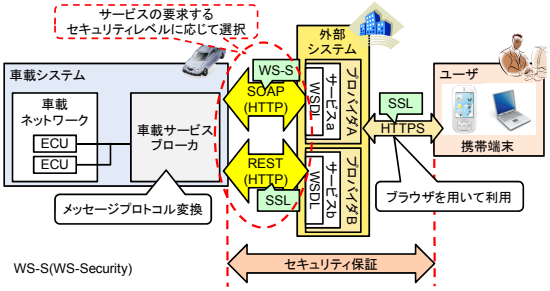


図 2 提案する連携方法

6. 提案アーキテクチャ

6.1. 自動車サービス連携のセキュリティアーキテクチャ

ユーザから車載システムまでの End-to-End で必要とされるセキュリティレベルを満たし，セキュアな通信を保証するアーキテクチャを提案する(図 3)．

SOA に基づくアーキテクチャと同様に車載サービスブローカに OSGi を用いることで，SOAP/REST を用いた連携が可能になる．セキュアな通信を実現するため WS-Security と SSL を用い，Web アプリケーションの要求するセキュリティレベルに応じて選択する．

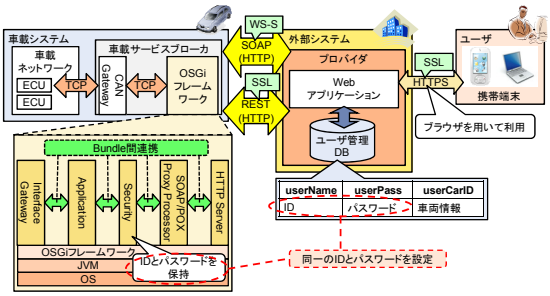


図 3 自動車サービス連携のセキュリティアーキテクチャ

6.2. 実現するセキュリティ機能

提案アーキテクチャでは，OSGi フレームワーク上に認証機能を持つ Security Bundle を追加する．Security Bundle は外部システムとの通信時に登録されたユーザ情報を用いて認証を行い，対応する Application Bundle と連携する．外部システムでも同様に，データベースに格納したユーザ情報を用いて認証を行う．これにより，第三者からの不正アクセスを防ぐことができる．また，ユーザと外部システム間，外部システムと車載システム間の通信は WS-Security/SSL を用いて暗号化され，個人情報の流出，盗聴を防ぐことができる(図 4)．

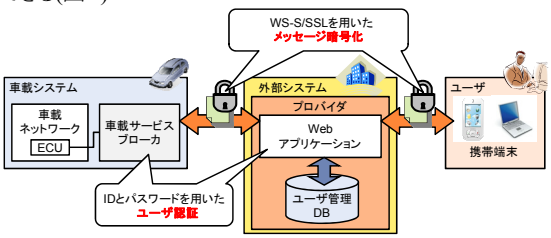


図 4 実現するセキュリティ機能

6.3. Security Bundle

Security Bundle のサービスは Axis2 Bundle によってデプロイされ，外部から利用可能となる[1]．外部システムは車載システムと通信する際に，ID，パスワード，ServiceName を送信する．Security Bundle は外部システムから送信された ID とパスワードを用いて認証を行う．認証後，ServiceName で指定されたサービスを持つ Application Bundle と連携する(図 5)．

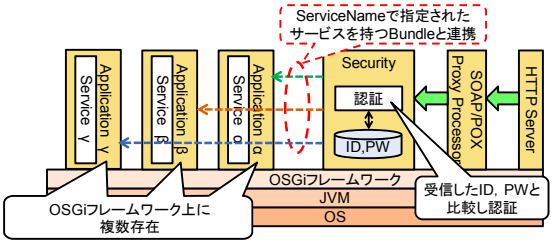


図 5 Security Bundle

6.4. セキュリティの選択

外部システムと車載システム間の通信には，SOAP(WS-Security)と REST(SSL)を選択することが可能である．外部システムの要求するセキュリティレベルに応じ，SOAP(WS-Security)と REST(SSL)から適切なセキュリティを選択する(図 6)．

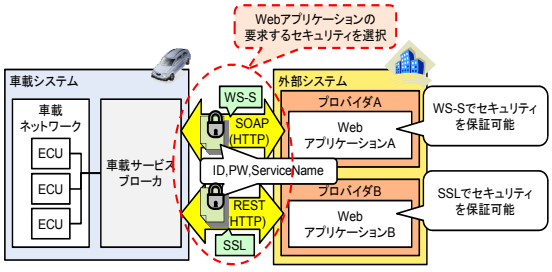


図 6 セキュリティの選択

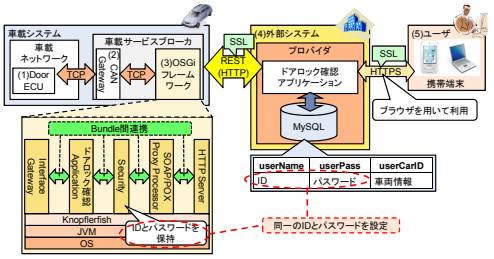
7. 提案アーキテクチャのプロトタイプ

7.1. 利用シナリオ

プロトタイプの利用シナリオとして，遠隔からドアのロック状態を確認する「ドアロック確認システム」を選択する．前提条件として，ドアロック確認アプリケーションの要求するセキュリティレベルは SSL で満たせるものとする．また，比較対象として，セキュリティ機能が無いドアロック確認システムを実装する．

7.2. 実装環境

車載サービスブローカの OSGi フレームワークには，オープンソースの Knopflerfish[6]を用いる．また，外部システムには Apache Tomcat[2]，データベースに MySQL[8]を用いる．車載システム内の通信は TCP 接続で行い，外部システムと車載システム間の通信は REST(SSL)で行う(図 7)．プロトタイプは Java で実装し，規模は外部システムが 108 行，車載システムが 216 行となった．



	(1)PC1 (DoorECU)	(2)PC2 (CAN Gateway)	(3)PC3 (OSGiフレームワーク)	(4)PC4 (外部システム)	(5)PC5 (ユーザ)
OS	Windows XP SP3	Windows XP SP3	Windows Vista SP2	Windows Vista	Windows XP SP2
CPU	Intel (R) Pentium (R)4 2.80GHz	Intel (R) Pentium (R)4 2.00GHz	Intel (R) core (TM)2 Duo 2.00GHz	Intel (R) Pentium (R) Dual 1.60GHz	Intel (R) Pentium (R)4 2.00GHz
メモリ	760MB	1.00GB	2.00GB	2.5GB	1.00GB
JDK	JDK 1.6.0_29	JDK 1.6.0_29	JDK 1.6.0_29	JDK 1.6.0_29	—
Eclipse SDK	Eclipse3.5.0	Eclipse3.5.0	Eclipse3.5.0	Eclipse3.5.0	—
OSGi Framework	—	—	Knopflerfish 3.1.0	—	—
Tomcat	—	—	—	Tomcat 6.0	—
MySQL	—	—	—	MySQL 5.5.17	—
ブラウザ	—	—	—	—	Internet Explorer 8
ネットワーク	1000BASE-T				

図 7 プロトタイプの開発/実行環境

7.3. プロトタイプの振る舞い

セキュリティ機能が有る通信では SSL 通信と認証を行い，ドアロック状態を取得する．セキュリティ機能が無い通信では，SSL 通信と認証を行わない．以下にセキュリティ機能が有る通信とセキュリティ機能が無い通信の振る舞いと測定範囲を示す(図 8，図 9，表 2)．

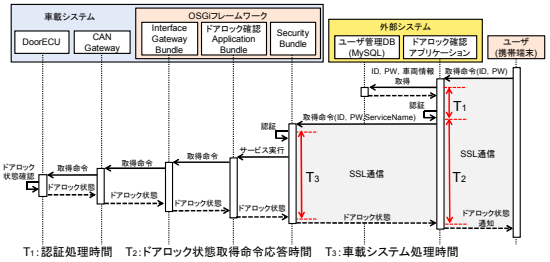


図 8 セキュリティ機能が有る通信のシーケンス図

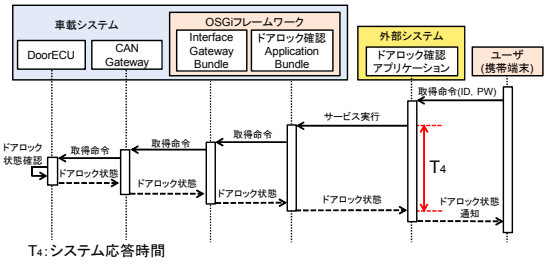


図 9 セキュリティ機能が無い通信のシーケンス図

表 2 測定範囲

		セキュリティ機能が有る通信	セキュリティ機能が無い通信
システム応答時間		T1+T2	T4
車載システム応答時間		T3	
セキュリティ処理時間	認証処理時間	T1	—
	外部システムと車載システム間のSSL通信処理時間	T2-T3	

8. プロトタイプに基づく評価

8.1. プロトタイプの性能評価

実装したプロトタイプを実行し，ドアロック確認アプリケーションの起動から終了までの応答時間を測定することで，性能評価を行う．DoorECU と CAN Gateway は Java で実装したスタブであるため，正式な車載システムの応答時間を測定することはできない．そのため，セキュリティ機能の有無に着目し，セキュリティ機能が有る通信とセキュリティ機能が無い通信を比較し，セキュリティ機能の処理時間を計測する．

応答時間の測定範囲はドアロック確認アプリケーションの起動から終了までであり，また，車載サービスブローカ内のユーザ認証は簡易的な実装である．よって，セキュリティ機能の処理時間は表 2 で示したように，外部システムで行うユーザ認証の処理時間(T1)と，外部システムと車載システム間の SSL 通信の処理時間(T2-T3)の合計である．

8.2. 応答時間の測定と平均値の算出

セキュリティ機能が有る通信とセキュリティ機能が無い通信をそれぞれ 1000 回実行し，応答時間を測定した(図 10，図 11)．セキュリティ機能が有る通信の測定範囲は図 8 の T1+T2，セキュリティ機能が無い通信の測定範囲は図 9 の T4 である．

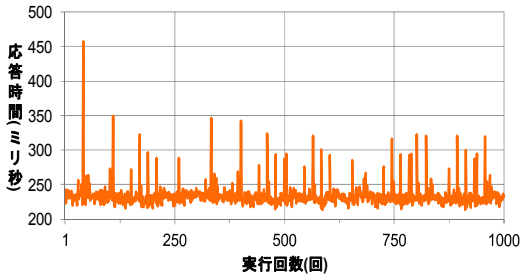


図 10 セキュリティ機能が有る通信の応答時間

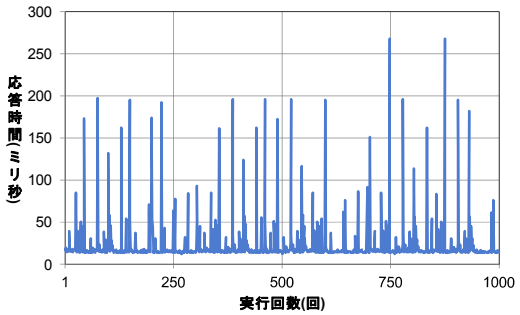


図 11 セキュリティ機能の無い通信の応答時間

図 10, 図 11 の測定結果では定期的に大幅な遅延が発生しているが, JVM のガベージコレクションによると推定できる. そこで, 測定結果の上位約 5%を特異値とし, 特異値を除いた応答時間を補修値とする. それぞれの 100 回ごとの補修値平均と補修値全体の平均, 標準偏差を以下に示す(図 12).

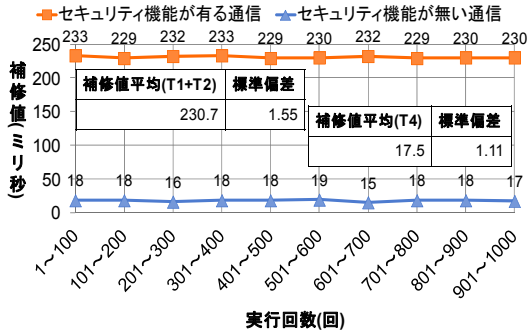


図 12 補修値平均と標準偏差

8.3. セキュリティ機能の処理時間

図 12 より, 応答時間は実行回数に依存せず, ほぼ一定であった. また, セキュリティ機能が有る通信とセキュリティ機能が無い通信の応答時間の差が, セキュリティ機能の処理時間となる. 外部システムで行うユーザ認証(T1)の平均応答時間は 30.4 ミリ秒, 外部システムと車載システム間の SSL 通信(T2-T3)の平均応答時間は 175.2 ミリ秒であった(表 3). この結果から, セキュリティ機能が有る通信の応答時間の内, セキュリティ機能の処理時間が 9 割以上を占め, 特に SSL 通信処理に時間がかかることを確認した.

表 3 セキュリティ機能の処理時間

	補修値平均(ミリ秒)
セキュリティ機能の有る通信(T1+T2)	230.7
セキュリティ機能の無い通信(T4)	17.5
セキュリティ機能の処理時間	213.2

	平均応答時間(ミリ秒)
認証処理時間(T1)	30.4
外部システムと車載システム間のSSL通信処理時間(T2-T3)	175.2

9. 考察

9.1. 車載システムからユーザまでのセキュアな通信

SOA に基づく車載サービスブローカアーキテクチャでは, 通信時のセキュリティ保証はされていない. 提案アーキテクチャでは, ユーザと外部システム間の通信に SSL, 外部システムと車載システム間の通信に WS-Security と SSL を適用することで, 車載システムからユーザまでのセキュアな通信を実現した.

9.2. プロトタイプの応答時間

プロトタイプとしてドアロック確認システムを開発し, セキュリティ機能の処理時間を測定した. その結果, 応答時間の 9 割以上がセキュリティ機能の処理時間であった. また, セキュリティ機能の処理時間の 8 割以上が SSL 通信の処理時間であることを確認した. これは, Java では SSL 通信の暗号処理に時間がかかるためだと考えられる. この問題の解決方法として, JNI(Java Native Interface)を用いて C 言語で実装されたネイティブコードと連携し, SSL 通信処理のパフォーマンスを改善する方法がある.

10. 今後の課題

今後の課題として, 以下の 4 点が挙げられる.

(1) SSL 通信処理のパフォーマンス改善
(2) セキュリティポリシに基づくセキュリティ選択のコード化
(3) WS-Security を用いたプロトタイプの実装
(4) 正式な ECU と CAN Gateway の利用

11. まとめ

本稿では, SOA に基づく車載サービスブローカアーキテクチャを拡張し, End-to-End でセキュアな通信を保証する新たなアーキテクチャを提案した. 異なるセキュリティレベルの要求に応じ, セキュリティとして WS-Security と SSL を選択可能とした. さらに, プロトタイプを開発し, 応答時間とセキュリティ処理時間を測定した.

参考文献

[1] Apache Axis2, http://axis.apache.org/axis2/java/core/.
[2] Apache Tomcat, http://tomcat.apache.org/.
[3] 濱千代 正弥, 片桐 雅仁, 自動車ネットワークサービスの連携アーキテクチャ 南山大学 2010 年度卒業論文, 2011.
[4] 情報処理推進機構, http://www.ipa.go.jp/.
[5] 情報セキュリティ標準テキスト編集委員会, 情報セキュリティ, オーム社, 2006.
[6] Knopflerfish OSGi-Open Source OSGi Service Platform, http://www.knopflerfish.org/index/html/.
[7] D. Krafzig, et al., Enterprize SOA, Prentice Hall, 2005.
[8] MySQL, http://www-jp.mysql.com/.
[9] OSGi(Open Service Gateway initiative) Alliance, http://www.osgi.org/Main/HomePage.
[10] C. Wells, Ajax アプリケーション&Web セキュリティ, オーム社, 2008.