IoT SAFE環境下での署名処理が通信プロトコル性能に与える影響の評価

迫本 澪[1,a)]　渡邉 龍星[,b)]　森 達哉[,c)]

概要：IoT デバイスの急速な普及に伴いセキュリティ確保が喫緊の課題となっており，SIM カード内で秘密鍵を安全に保持して署名処理を提供する IoT SAFE 規格が有望な解決策として注目されている．しかしながら，署名処理時間が実用上のボトルネックとなり得るにもかかわらず，その定量評価は乏しい．本研究では，IoT SAFE 対応 SIM カードを用いた署名処理の実機評価を行い，TLS-VPN，MQTT over TLS，CoAP over DTLS，HTTP over TLS の各プロトコルでの影響を分析した．その結果，IoT SAFE による署名処理はソフトウェア実装比で約 68 倍（約 680 ms）の遅延を示し， 各プロトコルの通信確立遅延は署名処理時間に依存することを確認した．したがって，プロトコル単位ではなく通信パターン単位で IoT SAFE の適用可能性を評価すべきであると提言する．これに基づき，代表的な通信パターンごとに適用可能性評価表を作成し，セッション再利用，トークンキャッシュ，非同期処理などの最適化指針を体系化した．本研究の知見は，IoT SAFE の導入可否判断および実用性能を確保する実装設計に関するガイドラインを提供する．

キーワード：SIM，IoT SAFE，IoT セキュリティ，署名性能評価，セキュリティプロトコル

Impact Assessment of Signature Processing on Communication Protocol Performance under IoT SAFE

Rei Sakomoto[1,a)]　Ryusei Watanabe[,b)]　Tatsuya Mori[,c)]

Abstract: With the rapid proliferation of IoT devices, ensuring security has become an urgent challenge. The IoT SAFE standard, which securely stores private keys on SIM cards and offloads signature operations to the SIM, has emerged as an approach to hardening device authentication. However, despite the potential for signature processing time to become a bottleneck, its performance impact has seen limited evaluation. In this study, we performed an empirical evaluation of signature processing using IoT SAFE-enabled SIM cards and analyzed the impact on communication performance across protocols: TLS-based VPN, MQTT over TLS, CoAP over DTLS, and HTTP over TLS. Our results showed that IoT SAFE incurs approximately 68-fold higher end-to-end latency (about 680 ms) than a pure software implementation, and we confirmed that connection establishment delays across all protocols depend on signature processing time. Therefore, we propose that IoT SAFE applicability should be evaluated on a per-communication-pattern basis rather than per-protocol. Based on this insight, we developed a suitability assessment matrix for representative communication patterns. We systematized optimization guidelines, including session reuse, token caching, and asynchronous processing. This research provides guidelines for IoT SAFE adoption decisions and implementation design to ensure practical performance.

Keywords: SIM, IoT SAFE, IoT Security, Signature Performance Evaluation, Security Protocol

[1] 株式会社エヌ・エフ・ラボラトリーズ
[2] 早稲田大学
[3] NICT
[4] 理研 AIP

[a)] rei.sakomoto@nflabs.jp
[b)] ryusei.watanabe@nsl.cs.waseda.ac.jp
[c)] mori@nsl.cs.waseda.ac.jp

1. はじめに

2025 年に約 199 億台，2034 年には約 406 億台に達すると予測される IoT デバイスの爆発的な増加は，従来のソフトウェアベースのセキュリティモデルの限界を露呈させている [24]．Industroyer や TRITON による攻撃では重要インフラの制御通信が傍受・改ざんされ，産業用制御システムにおける認証機構と通信保護の不備が顕在化した [5, 17]．また，Hajime や Remaiten といった IoT マルウェアは，感染デバイスのメモリ領域から暗号鍵を抽出する機能を実装しており，ソフトウェアによる鍵管理方式の構造的な問題を示している [11]．これらの脅威に対し，GSMA が策定した IoT SAFE 規格は，SIM カードのセキュアエレメントを活用して秘密鍵を耐タンパ領域で保護し，暗号化通信と鍵管理の両面でセキュリティを実現する [9, 10]．IoT SAFE は，既存の SIM カードインフラを活用し，専用の Hardware Security Module（HSM）や Trusted Platform Module（TPM）を新規導入することなくハードウェアセキュリティを実現できる．

しかしながら，IoT SAFE の実装には複数の技術的制約が存在する．第一に，標準でサポートされる暗号アルゴリズムは楕円曲線暗号の secp256r1 および brainpoolP256r1 に限定されており，他方式の採用はベンダ依存となっている [9]．第二に，SIM カードの演算能力に起因する署名処理の遅延は，リアルタイム性を要求するアプリケーションにおいて重大な制約となり得る．第三に，適用可能なセキュリティプロトコルは TLS/DTLS に限定されており，IPSec や OSCORE など他方式との互換性を欠いている．これらの制約は IoT SAFE の適用範囲を制限する要因となるが，既存研究においては実用性への影響を定量的に評価した事例が不足している．

そこで本研究では，IoT SAFE の署名処理性能が IoT 通信プロトコルに与える影響を実機評価により分析し，制約を踏まえた実用的な設計指針を提示することを目的とする．既存研究が IoT SAFE の基盤技術や個別プロトコルの評価に留まるのに対し，本研究は実機を使用して SIM カードの署名処理性能を評価し，通信パターンごとの適用判断基準を導出する点で差別化される．また，セッション再利用やトークンキャッシュ等の最適化手法を体系化し，性能制約下での実装方策を提示する．

実機評価では，IoT SAFE 対応 SIM カードを用いて ECDSA-P256 署名処理時間を測定し，署名処理に約 480 ms，エンドツーエンドで約 680 ms の遅延を確認した．次に，IoT 通信プロトコル (TLS-VPN，MQTT over TLS，CoAP over DTLS，HTTP over TLS) における通信確立遅延を分析した結果，遅延はプロトコル種別に依存せず，主に署名処理と SIM カード通信に起因することが判明した．これらの知見に基づき，通信パターン別に IoT SAFE の適用可能性を評価し，実装・運用指針を体系化した．

本研究の主な貢献は以下の通りである:

- IoT SAFE 対応 SIM カードを用いた ECDSA-P256 署名処理の実機評価
- IoT 通信プロトコルにおける通信確立遅延の影響分析と，通信パターン別の適用可能性評価フレームワークの提案
- セッション再利用，トークンキャッシュ，非同期処理など，IoT SAFE の性能制約を緩和する実装・運用指針の体系化

本論文の以降の章は次のように構成されている．第 2 章では，IoT SAFE の技術概要と関連研究について述べる．第 3 章では，IoT SAFE 署名処理性能の実機評価手法と結果を示す．第 4 章では，各種プロトコルへの影響評価と性能評価を行う．第 5 章では，通信パターン別の設計指針と IoT SAFE 実装戦略を提案する．第 6 章では，将来技術への対応を含む今後の展望について議論し，最後に第 7 章で結論を述べる．

2. 関連研究

IoT SAFE. IoT SAFE の標準仕様と運用シナリオは GSMA のホワイトペーパーにより初めて体系的に整理・普及が進んだ [10]．Pirker らは，IoT SAFE の API 設計を再構成し，ハッシュ処理の分離やコマンド簡素化により，TLS ハンドシェイク時のセキュアエレメントとの通信回数を最大 40%削減可能であることを示している [20]．特に「ハッシュ計算」と「署名要求」を分離する設計が性能改善に効果的であるとされる．また，Krishnan らは eSIM とブロックチェーンを組み合わせたゼロタッチプロビジョニングの自動化とセキュリティ強化を提案した [13]．

セキュアエレメントの性能評価. Noseda らは複数の汎用セキュアエレメントに対し，secp256r1（P-256）による ECDSA 署名処理の実行時間および消費エネルギーを詳細に測定した [19] ．その結果，最大 60%のエネルギー削減が可能であることが明らかとなった．IoT SAFE のような JavaCard ベースの実装では、署名 1 回あたり数百ミリ秒の遅延が発生するため、頻繁な署名が求められるプロトコルにおいては運用上の工夫が不可欠である．

IoT 向けセキュアプロトコルの性能評価. Gabriele らは，Cortex-M 系上で TLS ／ DTLS1.2 と 1.3 を実装・比較し，消費メモリや RAM の影響は構成により増減が見られ，エネルギ消費も設定に依存して増減があるため，「TLS1.3 が一概に高速」という結論には至らないことを示した [22]．Laaroussi らは，CoAP/DTLS と MQTT/TLS の遅延を実測し，ネットワーク RTT が 60–150 ms の条件でも MQTT の追加遅延は比較的限定的であったと報告した [14]．Gentile らは，OpenWRT ルータ上で OpenVPN（TLS ベース）を評価し，転送スループットは安定する一方，初回ハンド

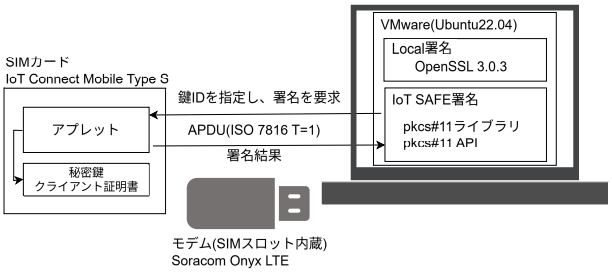

図 1 実験環境構成図

シェイクのオーバーヘッドが性能に影響し得ることを指摘している [7]．

プロトコル選定に関するサーベイ． Wytrebowicz らは MQTT，CoAP，AMQP など主要メッセージング・プロトコルを体系的に整理し，遅延，帯域効率，エネルギ消費，およびセキュリティ機能を軸にユースケース別の実用的な選定ガイドラインを提示した [25]．

本研究の位置づけ． 以上の関連研究を踏まえると，IoT SAFE の基盤技術や個別の IoT 通信プロトコルに関する研究は進展しているものの，IoT SAFE を実際の IoT システムに適用する際の包括的な評価と実装指針は十分に検討されていない．本研究では，これらの課題に対して実機評価に基づく定量的分析を行い，IoT SAFE を用いた実装時の通信パターン別の設計指針，および運用面での最適化手法を提示する．これにより，セキュリティ理論と実装現実のギャップを埋め，IoT SAFE の実用的な活用を促進することを目指す．

3. IoT SAFE 署名処理性能評価

本章では，IoT SAFE 対応 SIM カードを用いて ECDSA-P256 署名処理の性能を定量的に評価する．ここでは，ローカルホスト上で署名するソフトウェア署名 (以降，SW 署名) と IoT SAFE 署名処理時間の比較，および署名演算そのものと，その前段に必要となるセッション確立・認証・鍵検索の処理時間内訳に焦点を当てる．

3.1 実験環境

実験環境を図 1 に示す．SIM カードへのアクセスにはベンダ提供 PKCS#11 ライブラリ（`libpkcs11_iotsafe.so`）を使用し，Python 環境から `python-pkcs11` ラッパーを介して署名処理を実行した．

3.2 測定対象と API 呼び出しフロー

表 1 に PKCS#11 における署名処理の典型的な API 呼び出しシーケンスを示す [1]．呼び出しは表の上から順に実行される．また，本研究では，署名処理の性能特性を詳細に分析するため，表 2 に示す 3 つの測定モードを設定した．`sign_only` モードでは，セッションと鍵ハンドルを再利用することで，純粋な SIM 内署名処理時間を計測する．

表 1 PKCS#11 における代表的な署名呼び出し

API	概要
C_OpenSession	SIM カードと論理セッションを確立しハンドル取得
C_Login	PIN 検証によるユーザ認証
C_FindObjects	秘密鍵オブジェクトの検索とハンドル取得
C_SignInit	署名メカニズムとパラメータの初期化
C_Sign	データのハッシュ値を SIM カードに送信し署名を生成・取得

表 2 測定モードと API 区間

モード	計測区間
`local_sw`	OpenSSL `dgst -sha256 -sign` コマンド全体
`sign_only`	C_SignInit → C_Sign（署名処理のみ）
`total`	C_OpenSession → C_Sign（全処理）

表 3 署名処理遅延統計値（$n = 100$）

モード	平均 [ms]	σ	P50	P90	P99	>1s
local_sw	11.5	4.0	10.0	16.6	24.5	0
sign_only	476.5	9.2	475.1	483.5	494.8	0
total	677.7	177.2	642.2	671.9	1739.4	4

一方，`total` モードでは，実運用を想定し，毎回新規にセッション確立から署名完了までの全処理時間を計測する．

3.3 測定方法

各モードにおいて以下の手順で 100 回の試行を行った：

(1) **local_sw**：OpenSSL を用いた SW 署名のベースラインとして，P-256 秘密鍵（`privkey.pem`）により 32 バイトの固定データに対する署名を生成．

(2) **sign_only**：一度確立したセッション内で，同一の鍵ハンドルを用いて署名処理（C_SignInit/C_Sign）のみを繰り返し実行．

(3) **total**：各試行ごとに新規セッション確立（C_OpenSession），PIN 認証（C_Login），鍵検索（C_FindObjects）から署名完了までの全工程を実行．

署名対象データは全モード共通で 32 バイトの固定文字列（`"12345678901234567890123456789012"`）を使用した．各試行の処理時間を `time.perf_counter()` でマイクロ秒精度にて計測した．

3.4 結果

3.4.1 SW 署名と IoT SAFE の性能比較

表 3 に署名処理遅延の計測結果を示す．SW 署名（`local_sw`）の中央値 10 ms に対し，IoT SAFE の純粋な署名処理（`sign_only`）は約 475 ms と，**約 48 倍の処理時間**を要することが確認された．この差は主に SIM カードの計算能力の制約と APDU 通信オーバーヘッドに起因すると考えられる．

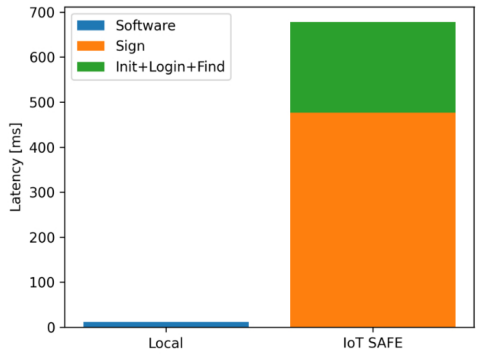

図 2 IoT SAFE 署名処理時間の平均値内訳

3.4.2 前処理フェーズのオーバーヘッド分析

図 2 に署名処理時間の内訳を示す．エンドツーエンドの処理時間（`total`）は平均 677.7 ms であり，純粋な署名処理時間（`sign_only`）の 476.5 ms に加えて，約 201 ms の追加オーバーヘッドが発生している．この追加時間は，セッション確立（C_OpenSession），PIN 認証（C_Login），および鍵検索（C_FindObjects）の各処理と，それらに伴う APDU 通信の累積によるものである．また，`total` モードでは標準偏差が 177.2 ms と大きく，4 回の試行で 1 秒を超える外れ値が観測された．これは，SIM カードとの通信において散発的に発生する遅延や，ホストシステムの影響によるものと推測される．

4. IoT 環境で使用されるプロトコルの概要とワークロード評価

本章では，第 3 章での評価結果を基に，IoT SAFE の署名処理性能が実際の IoT 通信プロトコルに与える影響を定量的に評価する．評価対象として，IoT 環境で主に使用されている TLS-VPN，MQTT over TLS，CoAP over DTLS，HTTP over TLS の 4 つのプロトコルを選定した．これらは標準化団体（IETF，OASIS）や主要 IoT プラットフォーム（AWS IoT Core [2]，Azure IoT Hub [16]，Google Cloud IoT [8]）で広く採用されているため，IoT SAFE の適用可能性を包括的に評価できる．

4.1 認証方式における署名処理の特性

前述の 4 つのプロトコルは，いずれも通信のセキュリティを確保するために認証機構を必要とする．IoT 環境では，デバイス認証の方式として主に mTLS（mutual TLS）と JWT（JSON Web Token）が採用されており，これらの認証処理において公開鍵暗号に基づく署名処理が実行される．これらの認証方式では署名処理が必須であるが，その実行頻度は認証方式やプロトコルの特性によって大きく異なる．本研究で着目する署名処理性能は，この署名処理の実行頻度に応じて各プロトコルに異なる影響を与えるため，各認証方式における署名処理の頻度特性を理解することが重要である．

表 4 認証方式別の署名処理特性

項目	mTLS	JWT
署名タイミング	接続確立時（1 回）	トークン発行時
署名頻度	低（セッション再利用可）	中〜高（有効期限依存）

4.1.1 mTLS（mutual TLS）

mTLS は，TLS ハンドシェイク時にクライアント証明書による相互認証を行う方式である．署名処理は CertificateVerify メッセージで 1 回のみ発生し，確立されたセッションは Session Resumption や Session Ticket により再利用可能なため，署名頻度を大幅に削減できる．AWS IoT Core や Azure IoT Hub など主要 IoT プラットフォームで標準採用されている [2, 16]．

4.1.2 JWT（JSON Web Token）

JWT は，アプリケーション層で動作するトークンベース認証である．クライアントは秘密鍵でトークンに署名し，サーバは公開鍵で検証する．トークンの有効期限に応じて定期的な再署名が必要となるため，設定によっては高頻度の署名処理を要求する可能性がある．Google Cloud IoT などで採用されている [8]．

4.2 対象プロトコルの概要と選定理由

4.2.1 TLS-VPN

OpenVPN に代表される TLS-VPN は，産業用 IoT や遠隔監視において既存ネットワークインフラを活用しながらセキュアな通信を実現する [7]．VPN の性質上，一度接続が確立されれば可能な限り維持される設計となっており，安定した環境では数時間から数日間の継続も可能なため，初回の署名処理オーバーヘッドの影響は相対的に小さい．

4.2.2 MQTT over TLS

MQTT は，低帯域幅環境でも効率的に動作する Pub/Sub 型プロトコルで，センサーデータ収集やテレメトリで広く使用される [14]．Keep-Alive メカニズムにより接続状態の維持を試みる設計となっているが，実際の接続継続時間はネットワーク環境や省電力要件に大きく依存する．安定した環境では署名処理の影響は初回接続時に限定されるが，頻繁な再接続が発生する環境では性能への影響を考慮する必要がある．

4.2.3 CoAP over DTLS

CoAP は，制約の厳しいデバイス向けに設計された REST 型プロトコルで，6LoWPAN や NB-IoT 環境での利用を想定している [22]．UDP ベースであるため通常の CoAP はステートレスだが，DTLS を適用した場合，セッション再開機能によりハンドシェイクのオーバーヘッドを軽減できる．ただし，リソース制約や省電力要件により，セッション維持期間は実装や運用方針に大きく依存する．

表 5　プロトコル評価用追加環境

評価項目	環境・ツール
TLS-VPN	OpenVPN 2.5.5
HTTPS Server	nginx 1.21
MQTT Broker	Mosquitto 2.0
CoAP Server	libcoap
JWT 検証	OpenSSL 3.0.3
測定ツール	`openssl s_client`, `curl`, `paho-mqtt`
試行回数	各条件 30 回 (平均値を使用)

表 6　通信確立遅延（単位：ms）

Protocol	SW	IoT SAFE	差分
TLS–VPN	1262	2061	+799
MQTT over TLS	59	757	+698
CoAP over DTLS	27	676	+653
HTTPS	17	678	+661

4.2.4　HTTP over TLS

HTTP over TLS(以降, HTTPS) は，REST API や OTA 更新など汎用的な用途で使用される．接続パターンが多様で，長期 Keep-alive から短命な接続まで幅広い [25].

4.3　性能評価

4.3.1　評価環境

本章の評価は，第 3 章で使用した図 1 の環境を基本とし，各プロトコルの評価に必要な追加のサーバ環境とツールを表 5 に示す．すべてのサーバは Docker 環境で構築し，ネットワーク遅延の影響を排除するためローカルホスト上で実行した．

4.3.2　mTLS 通信確立遅延評価

IoT SAFE 実装が各プロトコルの通信確立時間に与える影響を評価するため，ローカルホスト上で実行するソフトウェア実装（以降，SW 実装）との比較測定を行った．

測定方法．測定は，接続開始から各プロトコルで通信が可能となる以下の時点まで計測した：

- **VPN（OpenVPN）**：VPN トンネル確立完了
- **MQTT over TLS**：MQTT CONNECT ACK 受信
- **CoAP over TLS**：CoAP ACK 受信
- **HTTPS**：HTTP レスポンス受信完了

これらの測定により，TLS ハンドシェイクだけでなく，各プロトコル固有の初期化処理を含む通信確立遅延を評価した．

測定結果．表 6 に，mTLS を用いた各プロトコルの通信確立遅延の測定結果を示す．

測定結果から，全プロトコルで SW 実装と IoT SAFE 実装の通信確立遅延の差分が 650–800 ms であり，この遅延差分は第 3 章の結果から IoT SAFE 内での署名処理に起因すると考えられる．したがって，通信確立遅延はプロトコルに依存せず，主に暗号演算と SIM カードとの通信に依存することが示された．

なお，HTTPS については，セッション再利用時の測定も実施した．IoT SAFE 実装の初回通信確立時には 678 ms を要していたが，セッション再利用時は証明書・秘密鍵参照省略の最適化により，SW 実装と同等の遅延性能を達成した．この最適化では，SIM カードへの APDU コマンドが発生せず，SIM カードとの通信が不要となる．本結果は，セッション再利用時に限定されるものの，適切な実装最適化により通信確立遅延を解消可能であると示している．

4.3.3　JWT 認証遅延評価

JWT 認証方式については，HTTPS を対象として評価を実施した．JWT は初回のトークン生成時に署名処理を必要とするが，その後はトークンを再利用することで署名処理を回避できる特性を持つ．

JWT 署名処理時間．JWT 署名処理において，SW 実装が 15 ms で完了するのに対し，IoT SAFE 実装では 650 ms を要した．これは約 43 倍の増加であるが，トークンの有効期限を適切に設定し，キャッシュを活用することで，署名処理時間の影響を最小限に抑えることが可能である．

JWT 認証を用いた HTTPS 通信．JWT 認証を含む HTTPS の通信確立遅延を評価したところ，初回トークン生成時には署名処理の遅延が発生するものの，生成済みトークンを使用した 2 回目以降の接続では実装方式による差がほとんど見られなかった．トークンキャッシュ使用時の通信確立遅延は，SW 実装と IoT SAFE 実装の両方で約 10 ms となり，ほぼ同等の性能を示した．本結果は，HTTPS セッション再利用時と同様に，SIM カードアクセスの回避が性能向上の鍵となることを示している．

4.3.4　考察

性能評価から，以下の知見が得られた：

- **署名処理による一貫した遅延**：IoT SAFE はフルハンドシェイク時に 660–800 ms の追加遅延を発生させ，この値はプロトコルに依存しない
- **セッション再利用の効果**：セッション再利用により，SIM カードへの通信が発生せず SW 実装と同等の遅延性能を示す
- **JWT 認証の優位性**：トークン生成時は 650 ms（SW 実装の約 43 倍）を要するが，生成済みトークン使用時は両実装で約 10 ms と同等の性能を示す

本評価では，各プロトコルに共通する基本的な性能特性を明らかにしたが，測定された遅延が実用上どの程度の影響を与えるかは，各プロトコルの通信パターンに大きく依存する．実際の IoT 環境での適用可能性を判断するには，より詳細な通信パターンでの分析が不可欠である．

5.　通信パターン別設計指針と IoT SAFE 実装戦略

本章では，第 4 章で得られた評価を基に，実際の IoT シ

ステムにおける IoT SAFE の適用戦略を提示する．まず，代表的な 6 つの通信パターンに対する IoT SAFE の適用可能性を評価し，次に実装・運用時の具体的な最適化手法と留意点を示す．

5.1 通信パターン別の適用シナリオ設計

第 4 章の評価結果から，IoT SAFE の適用可能性は単一のプロトコル特性だけでなく，システム全体の通信パターンと業務要件の組み合わせによって決定されることが明らかになった．表 7 に，産業分野で一般的な 6 つの通信パターンと，それぞれに対する IoT SAFE の適用可能性評価の結果を示す．

5.2 IoT SAFE 実装・運用の最適化戦略

表 7 に示した各通信パターンにおいて，IoT SAFE の実用的な性能を実現するための最適化手法を以下に示す．これらは既存文献で効果が実証されている手法である．

- **セッション再利用／接続プール**： TLS 1.3 の PSK 再開モードや TLS 1.2 の Session Ticket メカニズムを活用することで，再接続時の完全なハンドシェイクを回避できる場合がある [22]．また，MQTT や HTTPS における Keep-alive パラメータの適切な設定により，接続の維持コストを最小化できる．
- **トークン認証の設計**：JWT などのトークンベース認証では，有効期限を業務要件に応じて適切に設定し，トークンのキャッシュを実装することで，署名処理の頻度を大幅に削減できる [23]．特にバッチ処理では，処理ウィンドウ内で有効なトークンを再利用する設計が有効である [25]．
- **非同期処理・事前認証**：資格情報の取得・更新やセッション確立を業務処理から切り離してバックグラウンドで先行し，実行時は認証済みセッションまたは有効な短寿命トークンを即時利用する．特にイベント駆動通信では，疎結合・メッセージ指向の設計と組み合わせることで，要求パスから認証遅延を分離できる [15]．
- **堅牢なエラーハンドリング**：SIM カードへのアクセスエラーやネットワーク断に対して指数バックオフで再試行し，発行済みトークンやセッション情報を適切にキャッシュして復旧時間を短縮する．なお，リプレイ耐性の観点から，署名値そのものの再利用は行わない [19]．

5.2.1 運用面の監視指標

IoT SAFE の実運用においては，以下の指標を継続的に監視し，最適化効果を評価することが重要である：

- 署名処理成功率および平均処理時間
- セッション再利用の頻度
- SIM カードアクセスエラー頻度
- 認証関連のタイムアウト発生率

本章で示した設計指針と最適化手法を適切に適用することで，IoT SAFE の高いセキュリティ性と実用的な性能を両立できる．ただし，100 ms 以下の決定論的な遅延が要求されるリアルタイム制御システムにおいては，PSK やハードウェアセキュリティモジュールなど，IoT SAFE 以外のソリューションも併せて検討すべきである．

6. 議論

6.1 制約事項

本研究の制約事項について議論する．

- **評価観点の限定**: 今回は実測レイテンシを基に表 7 を作成したが，スループットや消費電力は一般的な IoT 環境の評価で重視されるにもかかわらず，本評価には十分に反映できていない．
- **遅延内訳の未分解**: 各プロトコルの通信確立遅延を計測し，署名処理に大きく依存することを示した．一方，初期ハンドシェイクやアプリ層の確立処理，ネットワーク条件の寄与は未分解で，これらを切り分けて短縮するアプローチの検証は今後の課題である．
- **網羅性不足**: 評価は 4 種のプロトコルと ECDSA-P256 署名に依存し，多様なプロトコルと暗号アルゴリズムの組合せに対する What-if 分析は未実施である．

以上を踏まえ，本研究では主に署名処理遅延の評価に焦点を当てたが，今後は評価観点とカバレッジを拡張する必要がある．以下，そのための発展的な課題を述べる．

6.2 計算性能と耐量子暗号への適応

暗号アルゴリズム選定に関する発展的課題として，将来的な耐量子暗号への移行がある．NIST 標準の PQC アルゴリズム（CRYSTALS-KYBER, CRYSTALS-DILITHIUM, FALCON，SPHINCS+）を IoT SAFE に適用する際，以下の技術的課題が存在する [18]：

- **鍵・署名サイズ**: 従来の暗号と比較して大幅に大きな鍵・署名サイズを持つ（Dilithium2: 2.4 KB 署名，Falcon-512: 690 バイト署名）
- **メモリ制約**: 現行 SIM カード（RAM 数 KB）での実装が困難
- **処理時間**: 署名生成時間がさらに増加し，リアルタイム性が更に悪化

次世代セキュアエレメントの専用暗号コプロセッサにより署名処理時間の短縮が期待されるが，Noseda らの研究を参考に，どの程度の CPU 性能向上で実用的な処理時間を達成できるか定量的に評価する必要がある [19]．

---

*1 適用可能性評価：○ = 遅延要件を満たし実用的，△ = 設計工夫により実用可能，× = リアルタイム要件との両立困難

*2 初回認証後の長期セッション維持により条件付きで適合

表 7　IoT 通信パターンにおけるプロトコル特性と IoT SAFE 適用可能性評価[*1]

通信パターン	想定業界・用途	技術要件	一般的なプロトコル	工夫なしの適用可能性	IoT SAFE 実装上の課題と対策	工夫後の適用可能性	根拠文献
周期的データ収集	スマートファクトリ，環境モニタリング，スマートメータ	• 軽量性（省電力） • 多対一通信 • QoS 調整可能	MQTT/TLS (QoS 0-1) CoAP/DTLS (Non-Confirmable)	△ (初回認証 500 ms 以上)	課題：初回認証遅延 対策：セッション再利用，Keep-alive 最適化	○	[4, 22, 25]
イベント駆動通信	産業安全監視，ホームセキュリティ，医療アラート	• 到達保証 • 低遅延通知 • 省電力待機	CoAP/DTLS (Confirmable) MQTT/TLS (QoS 2) LwM2M/CoAP	△ (再接続時 500 ms 以上)	課題：スリープ中の認証維持 対策：事前認証，セッション保持	○	[14, 21]
リアルタイム制御	ドローン，産業オートメーション，自動運転	• 超低遅延 (100 ms) • 決定論的通信 • 最小オーバーヘッド	専用プロトコル/UDP WebSocket/TLS OPC UA Pubsub over TSN	× (mTLS 認証時の遅延)	課題：認証処理による遅延 対策：PSK 併用，ハードウェア暗号化	△[*2]	[6, 12]
リモート管理	製造設備保守，エネルギー監視，医療機器管理	• 双方向通信 • セッション管理 • 複雑操作対応	HTTPS/TLS (REST) TLS-VPN LwM2M/CoAP	△ (VPN 確立時の遅延)	課題：接続確立時間 対策：VPN 常時接続，証明書キャッシュ	○	[7]
バルク転送	自動車 OTA，スマート家電，ファームウェア更新	• 大容量転送 • 断続接続対応 • 再開機能	HTTPS/TLS CoAP Block-wise QUIC/TLS 1.3	△ (初回 TLS ハンドシェイク遅延)	課題：大容量データの認証 対策：チャンク単位認証，差分更新	○	[3, 26]
バッチデータ収集	スマートメータ，医療機器データ，環境センサ	• 一括処理効率 • 認証最小化 • データ圧縮	HTTPS/TLS+JWT MQTT/TLS (QoS 1) AMQP/TLS	○	課題：バッチ間の認証 対策：JWT 短期+更新，トークンキャッシュ	○	[2, 16]

6.3　総合シミュレーション環境

実用化に向けて，異なるプロトコル，暗号アルゴリズム，ネットワーク環境など，様々な条件下で IoT SAFE 性能を評価する必要がある．そのような評価を実現するための統合シミュレーション環境は以下にあげる機能を実装するものであり，その開発は今後の課題である．

(1) **プロトコルエミュレーション**: 主要 IoT プロトコルの動作エミュレーション
(2) **暗号処理モデリング**: 各種暗号アルゴリズムの署名処理時間を正確にモデル化
(3) **ネットワーク条件シミュレーション**: 遅延，パケットロス等の現実的なネットワーク条件の再現
(4) **What-if 分析**: 異なるパラメータ組み合わせでの性能予測

7.　結論

本研究では，IoT SAFE による署名処理性能がプロトコルに与える影響を評価し，実用的な設計指針を提示した．実機評価の結果，IoT SAFE による ECDSA-P256 署名処理は，ソフトウェア実装と比較して約 48 倍（約 475 ms）の遅延が発生することが明らかになった．また，セッション確立や認証処理を含むエンドツーエンドでは約 680 ms の遅延が確認された．プロトコルへの影響評価では，IoT SAFE 実装時には 660–800 ms の追加遅延を発生し，この値はプロトコルに依存しないことが示された．また，セッションの再利用時および JWT トークンキャッシュ時には，SW 実装と同等の遅延性能になることを確認した．これらの知見に基づき，通信パターン別の IoT SAFE 適用可能性を評価し，設計指針を体系化した．セッション再利用，非同期処理及びトークンキャッシュなどの設計工夫により，多くのユースケースで IoT SAFE の高いセキュリティと実用的な性能を両立できることを示した．

参考文献

[1] PKCS#11 cryptographic token interface specification version 3.1, February 2024. https://docs.oasis-open.org/pkcs11/pkcs11-spec/v3.1/pkcs11-spec-v3.1.pdf.

[2] Amazon Web Services. Client authentication. https://docs.aws.amazon.com/iot/latest/

developerguide/client-authentication.html, 2024. Accessed: 2025-06-24.

[3] C. Bormann and Z. Shelby. Block-Wise Transfers in the Constrained Application Protocol (CoAP). RFC 7959, Aug. 2016.

[4] M. Dammak et al. Performance evaluation of iot secure mqtt protocol deployment over tls/ssl. *IEEE Internet of Things Journal*, 7(9):8056–8069, 2020.

[5] Dragos, Inc. Crashoverride: Analysis of the threat to electric grid operations. Technical report, Dragos, Inc., June 2017. Version 2.20170613.

[6] C. Eymüller, J. Hanke, A. Hoffmann, M. Kugelmann, and W. Reif. Real-time capable opc-ua programs over tsn for distributed industrial control. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 278–285, 2020.

[7] A. F. Gentile, D. Macrì, F. De Rango, M. Tropea, and E. Greco. A vpn performances analysis of constrained hardware open source infrastructure deploy in iot environment. *Future Internet*, 14(9), 2022.

[8] Google Cloud. Device authentication and credential management. https://cloud.google.com/architecture/connected-devices/iot-platform-product-architecture, 2024. Accessed: 2025-06-24.

[9] GSM Association. Iot.06 iot security applet interface test specification. Technical report, GSMA, May 2021. Version 1.0.

[10] GSMA. IoT SAFE: SIM Applet For Secure End-2-End Communication, 2021.

[11] S. Herwig, K. Harvey, G. Hughey, R. Roberts, and D. Levin. Measurement and analysis of hajime, a peer-to-peer iot botnet. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. Internet Society, February 2019.

[12] A. Koubâa et al. Dronemap planner: A service-oriented cloud-based management system for the internet-of-drones. *Ad Hoc Networks*, 86:46–62, 2019.

[13] P. Krishnan, K. Jain, S. Poojara, S. Srirama, T. Pandey, and R. Buyya. esim and blockchain integrated secure zero-touch provisioning for autonomous cellular-iots in 5g networks. *Computer Communications*, 216, 02 2024.

[14] Z. Laaroussi and O. Novo. A performance analysis of the security communication in coap and mqtt. In *2021 IEEE 18th Annual Consumer Communications Networking Conference (CCNC)*, pages 1–6, 2021.

[15] T. Lodderstedt, J. Bradley, A. Labunets, and D. Fett. Best Current Practice for OAuth 2.0 Security. RFC 9700, Jan. 2025.

[16] Microsoft Azure. Authenticate identities with X.509 certificates. https://learn.microsoft.com/en-us/azure/iot-hub/authenticate-authorize-x509, 2024. Accessed: 2025-06-24.

[17] S. Miller, N. Brubaker, D. Kapellmann Zafra, and D. Caban. Triton actor ttp profile, custom attack tools, detections, and att&ck mapping. https://cloud.google.com/blog/topics/threat-intelligence/triton-actor-ttp-profile-custom-attack-tools-detections, April 2019. FireEye Mandiant Threat Intelligence.

[18] National Institute of Standards and Technology. NIST Announces First Four Quantum-Resistant Cryptographic Algorithms, July 2022. Accessed: 2025-07-17.

[19] M. Noseda, L. Zimmerli, T. Schläpfer, and A. Ruest. Performance analysis of secure elements for iot. *IoT*, 3:1–28, 12 2021.

[20] D. Pirker, T. Fischer, C. Reiter, H. Witschnig, and C. Steger. Towards a more flexible iot safe implementation. In *2021 24th Euromicro Conference on Digital System Design (DSD)*, pages 376–380, 2021.

[21] S. Raza, H. Shafagh, K. Hewage, R. Hummen, and T. Voigt. Lithe: Lightweight secure coap for the internet of things. 13(10):3711–3720, 2013.

[22] G. Restuccia, H. Tschofenig, and E. Baccelli. Low-power iot communication security: On the performance of dtls and tls 1.3, 2020.

[23] K. Shingala. Json web token (jwt) based client authentication in message queuing telemetry transport (mqtt). 03 2019.

[24] Transforma Insights. Current iot forecast highlights. https://transformainsights.com/research/forecast/highlights, June 2025. Selected highlights from the Connected Things forecast database.

[25] J. Wytrebowicz, K. Cabaj, and J. Krawiec. Messaging protocols for iot systems—a pragmatic comparison. *Sensors*, 21(20), 2021.

[26] K. Zandberg et al. Secure firmware updates for constrained iot devices using open standards: A reality check. In *IEEE Access*, volume 7, pages 71907–71920. IEEE, 2019.