ホワイトペーパー

産業機器
すべてのインテル® FPGA デバイスで機能する IP Xiphera の信頼の基点

FPGA ベースの セキュリティー・ソリューション

著 者

Matti Tommiska
Xiphera Ltd.
(フィンランド、エスポー市)
matti.tommiska@xiphera.com

Mark Jervis
インテル コーポレーション
プログラマブル・ソリューションズ事業本部
(英国、ハイウィコム市)
mark.jervis@intel.com

Karl Wachswender
インテル コーポレーション
プログラマブル・ソリューションズ事業本部
(英国、ハイウィコム市)
karl.wachswender@intel.com

目 次

はじめに

インダストリー 4.0 という用語で表される第 4 次産業革命[1]では、物理、デジタル、生物学の各分野のテクノロジーの融合により、IoT など、相互に接続されたサイバー・フィジカル・システムが作り上げられます。サイバー・フィジカル・システムと人間の間のリアルタイムのコミュニケーションによって、スマート・ファクトリー、クラウドベースの自動化、あるいは AI を活用した予知保全など、新たなビジネスチャンスが創出されます。

インダストリー 4.0 と IoT には間違いなく長期的な展望を期待できますが、主要技術の要件を満たさなければ、その可能性を十分に引き出すことはできません。そのような要件には、ユビキタスな接続、24 時間 365 日の可用性、低レイテンシーなども含まれますが、中心となるのは、「センサーからクラウドまで」すべてに及ぶ包括的なエンドツーエンドのセキュリティーです。最重視すべきセキュリティー要件に、IoT デバイスおよびそれらのデバイスが生成、処理、伝送するデータの保護[2]があります。関連する標準規格、コピー防止、サプライチェーンのセキュリティー規制を遵守しながら、同時にパフォーマンス要件、シームレスな可用性、使いやすさを実現しなければなりません。

企業にとっての主な課題は、ここでは産業機器メーカーも例外ではなく、上記のような高度なセキュリティー要件を具体的な安全性仕様として自社の製品に実装することと関連します。この実装の課題をさらに複雑にしているのが、経年とともに多様化してきたハードウェア・アーキテクチャー、産業用のセキュリティー要件に特化した標準化の遅れ[3]、セキュリティーと(機能)安全性の複合体であり、セキュリティーに関する知識や専門用語の全般的な不足も否定できません。

産業分野では、近年ますます巧妙化するサイバー攻撃の一例として、SCADA システムを標的とする Stuxnet、産業制御システム (ICS) を狙った Triton、送配電ネットワークを不正操作する CrashOverride などが挙げられます。これらの攻撃は氷山の一角に過ぎず、またその詳細についてはすべてが一般に知られているわけではありませんが、共通する特徴として「ソフトウェア・ベースの攻撃である」という性質があります。また、一般に公開されている情報によると、これらの攻撃は基盤となる暗号アルゴリズムを解読するものではありません(「暗号とセキュリティー」を参照)。つまり着目すべきは、暗号解読という壁を突破されたのではなく、セキュリティー実装の脆弱性を悪用されたという点です。

その結果、現在のインターネット[4]におけるセキュリティーの課題を生じさせるのも解決するのも、ほぼ常にソフトウェアでした。そして、これらの課題に対処する情報セキュリティー・ソリューション (侵入検出、マルウェア攻撃からの防御、(分散型) DoS 攻撃に対する耐障害性など) を提供する企業は、数十億ドル規模の産業となっています。しかし、これまでのソフトウェア中心のアプローチでは、インダストリー 4.0 と IoT でのセキュリティー対策には直接適用できず、その理由も、要求される耐用年数、アップデート性、消費電力、製品のフォームファクターと多岐にわたります。ハードウェア・ベースのセキュリティーの特長については、「プロセッサーと比較した FPGA の優位性」で詳しく説明します。

注 :

[1] 第 1 次から第 3 次の産業革命は、一般に蒸気動力と機械化、電力と大量生産、コンピューターと自動化の急速な導入拡大と定義されています。

[2] ここでの保護は、データの認証、機密性、真正性を意味します。これらについては「暗号とセキュリティー」で詳しく説明します。

[3] 最近発行された IEC 62443-4-2 では、Industrial Automation and Control System (IACS : 産業用オートメーションおよび制御システム) を構成するコンポーネントに対するサイバー・セキュリティーの技術的要件を定義していますが、IEC 62443-4-2 での「コンポーネント」は半導体コンポーネントを指しません。

[4] 「Internet of Things (モノのインターネット)」の普及に対して、「Internet of People and Organizations (人と組織のインターネット)」と呼ぶこともあります。

興味深いことに、防衛市場ではかなり以前からハードウェア・ベースのセキュリティー・ソリューションが採用されており、その理由も上記の理由とほぼ同様です。インダストリー 4.0 と IoT の導入も重要なインフラストラクチャー（送電、エネルギー、石油化学業界）に広がるため、現在の防衛市場のセキュリティー要件が、今後のインダストリー 4.0 と IoT のセキュリティー要件にも適用できると言えるのではないでしょうか。

このホワイトペーパーでは、以下の構成で解説します。セキュリティーの根本は暗号とその完全な実装に基づいていることから、まず「暗号とセキュリティー」で暗号の概要について説明します。

次にハードウェア・ベース（具体的には FPGA ベース）のセキュリティーの特長を「プロセッサーと比較した FPGA の優位性」で、続いて FPGA ベースの通信エンドポイントの実装例を示します。セキュリティー設計に不可欠なコンポーネントが信頼の基点（Root-of-Trust）です。FPGA ベースの信頼の基点に関するソリューションについて「信頼の基点」で説明し、最後に「まとめ」で概要を述べます。

暗号とセキュリティー

暗号は、一般に攻撃者と呼ばれる不正な第三者からデータや通信を保護する技術であり科学です。暗号を中心とする最新の情報セキュリティーでは、暗号アルゴリズムと暗号プロトコルを、信用できないネットワークを介した通信の保護、格納データへの不正アクセスの防止、ユーザーの認証といったさまざまな目的に用います。このようなシステムで特定のセキュリティー目的に暗号を実装する例を、表 1 に示します。

セキュリティーの目的	説 明
機密性（秘匿性）	認証済みの利用者（正当な送信者と受信者）のみが情報にアクセスできるようにする。
完全性	偶発的または意図的な操作から情報を保護する。
真正性	あるエンティティー（人物、団体）が主張どおりのエンティティーであること、またはデータの出所が主張どおりであることを保証する。
否認防止	人物 / 団体が以前の言動を否定できないようにする。

表 1. セキュリティーの目的

このようなセキュリティーの目的は、コンピューター・システムに暗号プロトコルを実装することで確実に実現されます。暗号プロトコルを構成するのが、特定のセキュリティー特性を保護する十分に確立されたアルゴリズムである暗号プリミティブです。

暗号プリミティブは、使用する鍵のタイプで分類できます。

- **対称暗号方式 :** 送信者と受信者がともに同じ鍵をもつ暗号アルゴリズム。

  対称暗号化では、送信者が暗号アルゴリズムで鍵を使用して平文テキスト（メッセージ）を暗号テキストに変換します。この暗号テキストは、鍵がなければ攻撃者が解読することはできないため、通信の機密性が保たれます。受信者は同じ鍵を使用して暗号テキストを元の平文テキストに復号します。セキュリティーを確保するには、該当の鍵が攻撃者の手に渡らないようにすることが不可欠です。対称暗号化アルゴリズムの 1 つが、Advanced Encryption Standard（AES : 高度暗号化標準）です。

  メッセージ認証コード（MAC）では、送信者が暗号アルゴリズムで鍵を使用して認証タグを導出し、このタグがメッセージとともに受信者に送信されます。受信者は、同じ鍵と受信したメッセージを使用してもう一方のタグを算出し、2 つのタグが一致した場合のみメッセージを受け入れます（そうでない場合、メッセージまたはタグが送信者の送信したものとは異なります）。攻撃者は鍵がないため有効なタグを作り出すことができず、通信の真正性（および完全性）が確保されます。ハッシュベースの MAC（HMAC）は MAC アルゴリズムの例です。

  認証付き暗号化は、機密性と真正性の保護を 1 つの暗号プリミティブにまとめた方式で、その一例が AES の Galois/Counter Mode（AES-GCM）です。

- **非対称暗号方式 :** 通信の一方が秘密鍵と公開鍵のペアを生成する暗号アルゴリズム。公開鍵は、逆変換が極めて困難と考えられる[5]数学関数を使用して秘密鍵から生成されるので、（攻撃者に）公開されても、セキュリティーが犠牲になることはありません。

  公開鍵暗号方式では、暗号化に公開鍵が使用されますが、復号には秘密鍵が必要です。よって、だれもが暗号化できますが、その暗号を解読できるのは受信者だけになります。RSA（考案者の Ron Rivest 氏、Adi Shamir 氏、Leonard Adleman 氏にちなんで命名）、El-Gamal などもこの暗号方式の例です。

  デジタル署名では、メッセージの署名に秘密鍵を、署名の正当性の検証に公開鍵を使用します。そのため、だれもが公開鍵を使用して、メッセージに署名したのが主張する署名者本人であることを確認でき、署名者はこれを後から否定することはできません（否認防止）。この方式には、Digital Signature Algorithm（DSA）、その楕円曲線版である ECDSA も含まれます。

  鍵交換プロトコルでは、非対称暗号により秘密鍵を 2 者間で共有します。この秘密鍵はその後、対称暗号に使用でき、通常こちらのほうが非対称暗号よりもはるかに効率的とされています。楕円曲線 Diffie-Hellman（ECDH）は鍵交換プロトコルの 1 つです。

注 :

[5] 「極めて困難」とは、実際には不可能と解釈すべきレベルです。

- **その他の暗号方式 :** 鍵を全く介さない暗号アルゴリズムもあります。

  よく知られている手法が暗号ハッシュ関数です。ハッシュ関数は、任意長の入力を取り込み、固定長の出力（ハッシュ）を生成します。ハッシュはメッセージの指紋とみなすことができます。ハッシュから入力メッセージを得ることが不可能でなくてはならず、入力メッセージと同一のハッシュを持つ別のメッセージ、つまり2つのメッセージが同じハッシュとなること（衝突）があってはなりません。暗号プロトコルでのハッシュ関数の用途は幅広く、MAC、鍵導出関数（KDF）など、暗号プリミティブの構成にも使用されます。SHA-2（Secure Hash Algorithm）は、ハッシュ関数による暗号の一例です。

既存の暗号方式はケルクホフスの原理に基づいています。暗号システムの詳細は、秘密鍵を除いてすべて攻撃者にも知られていることが前提となるため、セキュリティーは鍵の秘匿性にのみ左右されます。つまり、対称暗号でも非対称暗号でも、（秘密）鍵の予測が困難であることが必須であり、暗号システムにはランダムな鍵を導出する規則性のないソースが必要になります。この乱数発生器（RNG）は、真性乱数発生器（TRNG）と擬似乱数発生器（PRNG）の2つに分類されます。TRNGでは、物理的エントロピー・ソースから、予測不可能な特定の物理現象に基づいた真にランダムなビットを導出します。一方PRNGは、シード値（初期値）からランダムに見えるビット列を生成する確定的アルゴリズムです。通常はTRNGを用いる場合でも、PRNGを併用してTRNGで生成されるビット列に起こり得る偏りを解消します。

標準的な暗号プロトコルは複数の暗号プリミティブを実装し、例えばトランスポート・レイヤー・セキュリティー（TLS）ハンドシェイク・プロトコルでは、デジタル署名による証明書の検証や、クライアント / サーバー間の共有秘密鍵を生成する鍵交換に非対称暗号を使用します。その後、TLSレコードプロトコルで対称暗号により暗号化とMAC（認証付き暗号化）のプロセスを処理し、クライアントとサーバーの間のバルク通信で機密性と完全性を確保します。

暗号プロトコルの例については、「実装例」と「信頼の基点」で紹介します。

プロセッサーと比較したFPGAの優位性

現在のセキュリティー実装の大半は、ソフトウェアでの暗号プロトコルの実装に基づいており、ほとんどの場合サードパーティーの暗号ソフトウェア・ライブラリーが採用され、広く利用されているオペレーティング・システムが稼動する汎用プロセッサー向けにコンパイルされ、そのOS上で実行されます。

ソフトウェア・ベースの暗号が最も普及している一方で、セキュリティーをハードウェアに直接実装するという動向も、特に重要な組込みシステムで増えつつあります。この動向の誘因と実現方法については、次のセクションで詳しく説明します。

既存のソフトウェア・ベースのセキュリティー実装は全体的に複雑であるため、悪意のある第三者にいくつもの潜在的な標的を呈することになります（技術用語で言う「攻撃面」の拡大）。攻撃の標的にされる例としては次のようなものが挙げられますが、これもほんの一部に過ぎません。

- オペレーティング・システム
- デバイスドライバー
- 暗号プリミティブの実装（例えばサードパーティーの暗号ライブラリーを利用した場合など）
- コンパイラーの最適化と、プロセッサー世代間で起こり得るマイクロアーキテクチャーの変更
- 極めて深いソフトウェア・スタック
- キャッシュとメモリーの管理
- 鍵管理（例えばバッファー・オーバーフローのバグは、秘密鍵を含め、あらゆるものを漏洩させる危険性があります）
- セキュリティー・アルゴリズムの実装における完全制御の欠如

上記で挙げた攻撃面に加え、ソフトウェア・ベースのセキュリティー実装では必要なパフォーマンス（スループット、レイテンシー）が満たされていないこともあります。多くの場合、消費電力も課題となります。

重要な点として、ソフトウェア・ライブラリー（暗号ライブラリーを含む）とオペレーティング・システムの両方に対しアップデート性（それも産業用システムの耐用年数全体にわたるアップデート性）を維持することは、克服が難しい保守面での課題の1つではないでしょうか。例えばIoTデバイスには、ライフサイクル全体を通じてバグ修正[6]を行い最新の状態を維持することが求められますが、耐用年数は数十年にも及びます。これは負荷の高いタスクであり、ソフトウェア・ベースのセキュリティー・ソリューションで総保有コストを実質的に増加させる要因です。

最近ではセキュリティーに対する考え方も進歩し、基盤となるプロセッサー・アーキテクチャーのセキュリティーも疑問視されるようになりました。これまでは基盤のハードウェア・プロセッサーは安全であると当然のように考えられていましたが、プロセッサー・マイクロアーキテクチャーのパフォーマンス強化を目的とする最適化（アウトオブオーダー実行、投機的分岐実行など）を悪用する攻撃（Meltdown、Spectre、Foreshadowなど）が明らかになったことから、この根本の前提条件に疑念が生じています。こうしたセキュリティー上の弱点のほとんどに更新プログラムが適用されていますが、新たな攻撃が出現する可能性は無視できません。

以上で説明した理由から、ハードウェア・ベースのセキュリティー・ソリューションが広く求められるようになり、再プログラム可能なハードウェア、特にFPGAにセキュリティーを実装する動向が高まっています。

注 :

[6] 例えば、cve.mitre.org（一般に公開されているサイバー・セキュリティーの脆弱性のリストを維持管理しているサイト）で「openssl」を検索すると、288件の検索結果が表示されます（2019年3月11日現在）。

FPGA は集積回路であり、その機能は通常ハードウェア記述言語 (VHDL、Verilog) で指定され、導入後も現場での設計者または顧客による再プログラムが可能です。現在市場で入手可能な FPGA の大多数は、揮発性 SRAM テクノロジーに基づいており、つまり FPGA の機能は外部の不揮発性メモリーに格納されるコンフィグレーション・ファイルで設定することになります。最新の FPGA ファミリーがサポートする暗号化および認証付きのコンフィグレーションでは、コンフィグレーション・ファイルが外部の不揮発性コンフィグレーション・メモリーに平文で格納されることはなく、起動時に FPGA がコンフィグレーション・ファイルの内容を復号し、認証します。

従来、再プログラム可能なロジック (FPGA) のセキュリティー実装での用途といえば、パフォーマンスを重視する暗号プロトコルのアルゴリズムを高速化してホスト・プロセッサー・システムのオフロードを行うことでした。高速化に用いられるデザイン技術は通常、パイプライン化、並列化、ループ展開です。この高速化には、コスト削減と省電力の点でも、AES などの対称暗号が特に適していました。

しかし、ここまで説明してきたとおり、FPGA ベースのセキュリティーを実装するメリットは、パフォーマンスの向上に限られるものではなく、セキュリティー機能を FPGA ロジックに適切に実装することで、最終製品のセキュリティー・レベルも強化できるということです。

FPGA ベースの暗号実装には、ほかにも次のような技術的優位性が挙げられます。

- アルゴリズムとプロトコルの機敏性とアップデート性
- FPGA セキュリティー機能の内蔵
  - 暗号化、認証付きコンフィグレーション
  - 改ざん防止機能
  - パーシャル・リコンフィグレーション (一部のケース)
  - Red/Black 分離をハードウェアに実装するデザイン手法
- アルゴリズム実装のより厳格な制御 (特に重要な鍵管理など)

実際には、FPGA ベースのセキュリティー・プロトコルの実装では、個別の Intellectual Property (IP) ブロックを組み合わせます。次のセクションで実装例を示します。

実装例

「プロセッサーと比較した FPGA の優位性」で説明したとおり、安全な通信チャネルの設定には、複数の暗号プリミティブで構成する完全な暗号プロトコルが欠かせません。そのため、FPGA ベースの実装には、暗号プリミティブ、プロトコルで使用する鍵用のセキュア・キー・ストレージ、制御の各 IP ブロックの組み合わせが必要です。ここでは、安全性が保証されていないネットワーク (インターネット) を介したリモートサーバーへのセキュアな接続を FPGA で設定する方法を例に、必要な IP ブロックと合わせて説明します。スループットの要件は低から中程度にしつつも、FPGA リソースの要件はできるだけ低く抑えたい産業用アプリケーションに、特に焦点を合わせました。

セキュアな接続を確立する方法は複数ありますが、この例では、FPGA がプロトコルでクライアントとして動作する TLS のようなユースケースを想定します。クライアントは接続先が本物のサーバーであり、悪意のある第三者 (中間者攻撃) ではないと確信できるよう、サーバーの真正性を検証する必要があります。クライアントはサーバーの長期公開鍵を保持し、その公開鍵を使用してサーバーのメッセージをデジタル署名で認証できることが前提です。[7] 鍵交換では、セッションごとに異なる鍵 (いわゆる一時鍵) で、どの長期鍵にも結びつかない鍵を導出する必要があります。これにより前方秘匿性を確保でき、将来漏洩が起きても過去の通信は保護されます。鍵交換が完了した後は、セッション鍵によりバルク通信で交換するメッセージの機密性、完全性、真正性が保護されます。

上記の方法は次のような FPGA ベースの IP ブロックの組み合わせで実現できます。

1. RNG : 推奨は TRNG。PRNG のセキュアなシーディングを実行します。
2. ハッシュ・アルゴリズム : プロトコルで次の項目を含めた複数用途に実装します。
   - PRNG
   - バルク通信の完全性を確保する MAC アルゴリズム (HMAC など)
   - ハッシュ・アルゴリズムでインスタンス化できる KDF (HMAC ベースの HKDF)
3. デジタル署名および鍵交換用の非対称暗号 : 推奨の方法は楕円曲線暗号 (ECC) 用 IP ブロックの使用です。それ以外の主な選択肢は RSA ですが、効率は大幅に低下します。非対称暗号の演算は、セキュアな接続を確立するプロセスの中でも最も負荷の高い演算処理ですが、セッションの開始時にのみ必要な演算のためレイテンシーは問題にはならず、FPGA リソースの使用率を最小限に抑えた IP ブロックを選択できます。
4. 対称暗号 : バルク通信の保護に使用します。ここで最も一般的な選択肢は、AES を安全な演算モードで使用することです。例えば、AES-GCM は機密性、完全性、真正性を同時に確保できます (いわゆる認証付き暗号化)。
5. セキュア・キー・ストレージ : セキュリティーが重要な情報すべてを FPGA に格納します。

注 :

[7] TLS では、相互に信頼できる第三者 (認証機関) が発行した証明書を使用して、公開鍵に対する信用を確立しますが、この例では、汎用性を失うことなく、信頼できる公開鍵が FPGA に存在するものとします。

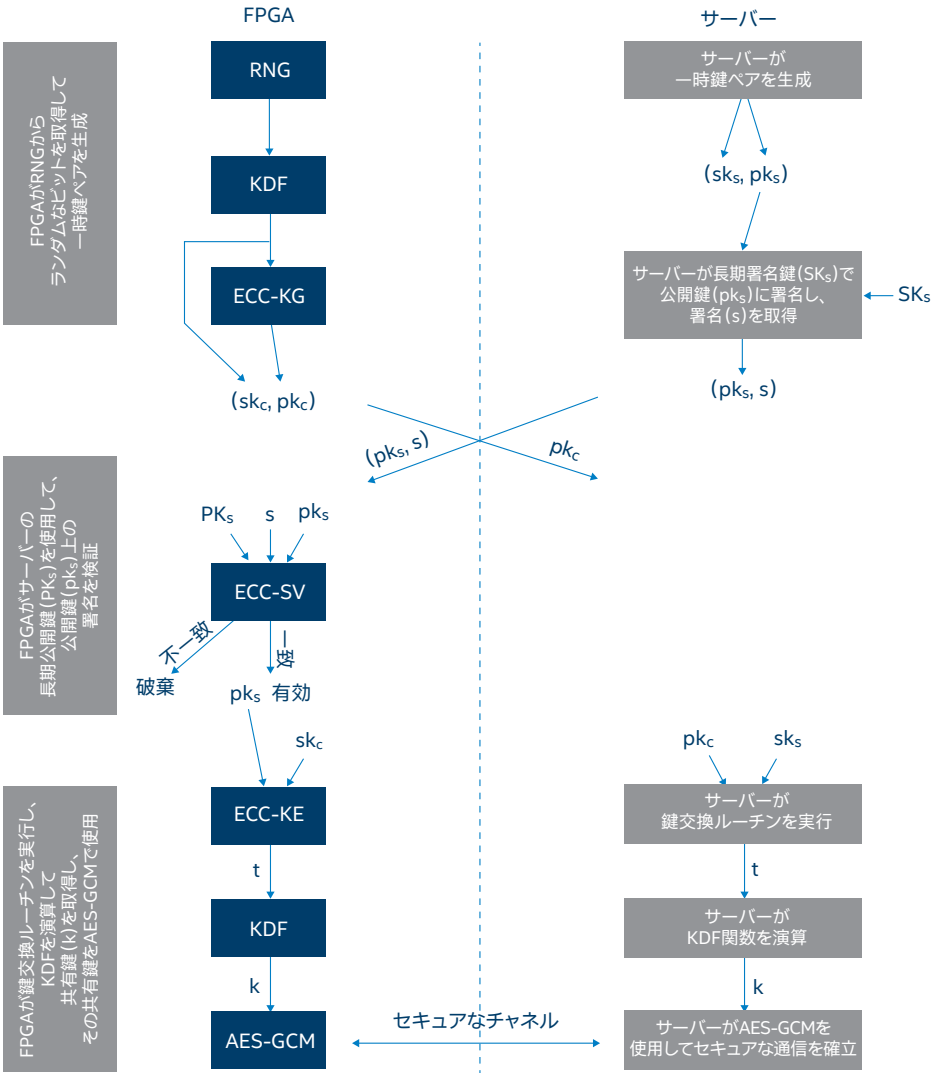


図 1. FPGA クライアントとリモートサーバーの間のセキュアな接続の確立

IP ブロックを組み合わせることにより、次の手順でリモートサーバーとのセキュアな接続を確立します(図 1 の流れ図を参照)。

1. FPGA が RNG、KDF(ハッシュ・アルゴリズム IP ブロック)、ECC IP ブロックを使用して、一時鍵ペア (クライアントの秘密鍵 *skc* と公開鍵 *pkc*) を生成します。公開鍵 *pkc* はサーバーに送信されます。
2. サーバー側でも自分の一時鍵ペア (サーバーの秘密鍵 *sks* と公開鍵 *pks*) を生成します。この一時公開鍵 *pks* に長期的秘密鍵 SKs で署名し、公開鍵 *pks* と署名 s をクライアントに送信します。
3. FPGA は、ECC に基づく非対称暗号 IP ブロックを使用して、サーバーの一時公開鍵 *pks* に添付された署名 s を検証します。この検証は、FPGA にあるサーバーの長期公開鍵 PKs を使用して実行され、この検証が成功すれば、一時公開鍵の出所が確かに正規のサーバーであり、攻撃者ではないことが確証されます。
4. FPGA は自分の一時秘密鍵 *skc* とサーバーの一時公開鍵 *pks* を使用して、非対称暗号 IP ブロックにより、ECC ベースの鍵交換に使用する共有秘密値 t を算出します。次に、この共有秘密値を KDF(ハッシュ・アルゴリズム IP ブロック) に入力して、バルク通信に使用する共有秘密鍵 k を導出します。
5. サーバーは自分の一時秘密鍵 *sks* とクライアントの一時公開鍵 *pkc* を使用して、(鍵交換演算と KDF 関数により)同じ共有秘密鍵 k を算出します。
6. FPGA は共有秘密鍵 k と AES-GCM(対称暗号 IP ブロック) を使用してサーバーとのセキュアな通信を確立します。AES-GCM で使用する初期化ベクトルの生成には RNG を使用できます。セッションが終了すると、すべての一時鍵は(共有秘密鍵も含み)削除されます。

上記の手順では、FPGA(クライアント) だけがサーバーの真正性を検証しました。FPGA の真正性も同様に、手順 1 および 2 で長期公開鍵を使用して検証できます。あるいは、上記で確立した安全な通信チャネルを介して別の認証メカニズムを適用することも可能です。この方法では FPGA に信頼の基点が必要です。信頼の基点については次のセクションで説明します。

信頼の基点

信頼の基点(Root-of-Trust) は、信頼できるコンピューティング・ユニット内の、システムが常に信頼できる機能のセットと定義できます。したがって、信頼の基点によりシステム全体に信頼性を構築することが可能になります。さらに、システム内の各デバイスに固有の ID、つまり秘密鍵を割り当てることが重要で、この鍵から公開鍵などの主要要素を導出できます。

ハードウェア・ベースの信頼の基点の必要性は、重要な IoT アプリケーションで認識されていました。これを組込みシステムで実現する最も一般的な方法は、トラステッド・プラットフォーム・モジュール (TPM) とも呼ばれる外部のセキュリティー・チップを使用することです。ただし、外部のセキュリティー・チップには必ずしもアプリケーションに必要な機能が備わっているとは限らず、例えば対称暗号化のスループットが十分ではないことも、設計コスト、ボード面積、消費電力が増大してしまう場合もあります。

すでに揮発性 FPGA を使用しているデザインでは、固有の秘密鍵を暗号化コンフィグレーション・ファイルに組み込むことでハードウェア・ベースの信頼の基点機能を実装したいという意向があるかもしれません。しかし、この方法では、メーカーがまず FPGA ベースの最終製品ごとに固有のコンフィグレーション・ファイルを生成し、発売後も各製品を特定のコンフィグレーション・ファイルに結び付けるためデータベースを維持していく必要があります。すべての製品の機能で同一の FPGA デザインを採用していても、これは変わりません。このようなコンフィグレーション・ファイルの生成や維持管理のタスクにはコストと時間がかかり、負担となるのは目に見えています。

Xiphera ではこの課題を解決するため、必要なのは FPGA とそのコンフィグレーション・メモリーのみという[8]、FPGA ベースの信頼の基点機能を提供するソリューションを開発しました。Xiphera の信頼の基点ソリューションは、現在利用可能な Xiphera IP ブロックを基盤とし、複数の FPGA から成るバッチ[9] に同一のコンフィグレーション・ファイルを適用できるようにしています。

FPGA ベースの信頼の基点ソリューションが主な対象としているのは、FPGA ベースのセキュリティーの新たな設計 / 導入ですが、一部のケースでは既存環境への組込みも可能です。ただし、FPGA に利用可能なリソースが十分にあり、再プログラムを実行できる場合に限ります。

FPGA ベースの信頼の基点機能は、RNG、KDF、セキュア・キー・ストレージの各 IP ブロックの組み合わせにより実現され、セキュリティーが重要な機能のすべてが確実に FPGA 内に実装されます。セキュリティー重視の主要要素と一般データとを厳密に分離し、鍵が外部からアクセスできないこと、暗号化による保護なしでは鍵が決して FPGA の外部に出ないことが徹底されます。

注目すべき点は、FPGA ベースの信頼の基点ソリューションに、顧客が独自に選択した暗号セットでも、または必要に応じて特定の標準規格でも、統合できることです。同様に、暗号アルゴリズムのパフォーマンス (スループットなど) をカスタマイズすることもできます。例えば、バルク通信プロトコルに認証付き暗号化アルゴリズムとして AES-GCM を使用している場合、スループットを 100Mbps/1Gbps/10Gbps に設定できますが、どのバージョンも同じ信頼の基点ソリューションを基盤として構築することが可能です。

FPGA ベースの信頼の基点ソリューションは、外部の制御ロジックまたはホスト・プロセッサーをインターフェイスとして接続することができ、このインターフェイスは FPGA の外部でも内部でも(ソフトコアでもハードコアでも)問題ありません。これは通信プロトコルのうちセキュリティーが重要でない部分(TLS over TCP/IP など) の処理に有効です。

まとめ

インダストリー 4.0 は現在進行中かつ加速中の第 4 次産業革命ですが、インダストリー 4.0 の完全な導入を成功させるためには、包括的なエンドツーエンドのセキュリティーが必要です。これまで明らかになったセキュリティー侵害の大半は、ソフトウェア・ベースのセキュリティーと暗号の実装に潜む脆弱性を巧妙に突いた攻撃により発生しています。そのため、重要な機能をハードウェアに直接実装することが、重要なセキュリティー要件を満たすためには望ましい手法だと認識されるようになりました。ハードウェア・ベースのセキュリティー実装には、ほかにもパフォーマンスの向上、消費電力の削減といった優位性が挙げられます。

ハードウェア・ベースのセキュリティー設計において FPGA は有力な候補であり、再プログラムが可能なことから、設計フェーズで暗号アルゴリズムやセキュリティー・プロトコルのさまざまなインタラクションを繰り返せるだけでなく、設計者は実装の完全な制御と把握を維持することができます。一般に、FPGA ベースのセキュリティーでは、個別の IP ブロックを組み合わせて、鍵管理を含めたセキュリティー・プロトコル全体を構築します。

Xiphera が提供する信頼の基点ソリューションで必要とされるのは、揮発性 SRAM ベースの FPGA と外部コンフィグレーション・メモリーのみです。この信頼の基点ソリューションでは、FPGA が固有の秘密鍵を生成し、初期化中にその鍵から追加の秘密鍵と公開鍵を導出できるので、電源をオフにして再度オンにしても同じ鍵が復元されます。この仕組みによって、揮発性 FPGA を使用した信頼の基点のセキュアなハードウェア・モジュールの構築が実現されます。

注 :

[8] インテル® MAX® 10 FPGA の場合、コンフィグレーション・フラッシュが同一パッケージ内にあるため、Xiphera の信頼の基点ソリューションはシングルチップ・ソリューションとして提供されます。

[9] バッチは、コンフィグレーションが同一の FPGA グループと考えることができ、顧客あるいは製品ごとに固有の違いはありますが、バッチサイズには実用上の数値的な上限 / 下限はありません。

用語集

攻撃面 : システムを攻撃しようとする相手から標的にされる危険性のあるシステムのさまざまなポイント。

認証付き暗号化 : 機密性(秘匿性) と真正性(AES-GCM など) の両方を提供する暗号化システム。

バッファー・オーバーフロー : バッファーの限界を超えてデータが書き込まれる(読み込まれる)事象。

コンフィグレーション・ファイル : 特定のデザインを実装するよう FPGA を構成するファイル。

暗号解読 : 暗号による保護を破ることを目的とした、暗号システムの研究分野。

暗号アルゴリズム : 暗号演算を実行する特定のアルゴリズム (AES など)。

暗号プロトコル : 特定の暗号アルゴリズムを適用して目的のセキュリティー機能を実装するプロトコル (TLS など)。

暗号 : データや通信を不正な第三者から保護する科学であり技術。

デジタル署名 : デジタル文書の真正性を検証する技術。紙の文書に対する手書き署名に相当。

楕円曲線暗号 : 楕円曲線と呼ばれる数学的構造を利用した非対称暗号システム。楕円曲線離散対数の困難性を安全性の根拠とする暗号。

暗号化 / 復号 : 送信者と受信者のみが読めるようにメッセージの内容をエンコードおよびデコードするプロセス。

一時的鍵 : 1 回のセッションに限り生成 / 使用される暗号鍵。

Field Programmable Gate Array(FPGA) : ロジックレベルでデバイスに演算を実装するプログラム可能な集積回路。

前方秘匿性 : 長期的機密性が損なわれた場合でも、以前に使用されたセッション鍵は侵害されないことを保証する機能。

ハッシュ関数 : 任意の長さのメッセージを固定長の値(ハッシュ) にマッピングする関数。

Intellectual Property(IP)ブロック : 特定の機能を実装するデジタルデザインのコンポーネント (暗号アルゴリズムの実装など)。

鍵交換 : 安全性が保証されていないネットワークを介して 2 者間で秘密鍵を交換できる技術。

鍵管理 : 暗号システム内で暗号鍵の管理(生成、保管、使用など) を行うプロセス。

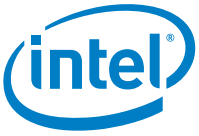
メッセージ認証コード : メッセージを認証し、改ざんを防止する技術。

乱数発生器 : ランダムなビットを生成するコンポーネントまたは関数。真性乱数発生器では、予測不可能な物理現象に基づく物理的エントロピー・ソースから乱数を取得しますが、擬似乱数発生器は、シード値からランダムに見えるビット列を生成する確定的アルゴリズムです。

Red/Black 分離 : 暗号化されていない(Red)情報と暗号化されている(Black)情報を厳密に分離すること。

信頼の基点 : システムが常に信頼することのできる(暗号)機能のセット。

トランスポート・レイヤー・セキュリティー(TLS) : インターネット上でクライアントとサーバーの間にセキュアな通信チャネルを提供するプロトコル。



WP-01297-1.0/JP