小型デバイスを用いた安全な通信の実現へ向けて

満保雅浩，西永俊文
金沢大学

2016年3月7日

IoTセキュリティに関連する話題

- IoTの一種であるスマートグリッドにおけるサイバーセキュリティの検討状況　（前半の内容）

- IoTで利用されるデバイスの通信機能の保護　（後半の内容）

スマートグリッド・サイバーセキュリティ標準化

- International Electrotechnical Commission (IEC), IEC Smart Grid Standardization Roadmap, June 2010.
  - サイバーセキュリティの議論があまりない

- NIST, NISTIR 7628: Guideline for Cyber Security in the Smart Grid, Vol.1-3, August 2010.
  - サイバーセキュリティについて考慮されている。しかし、不十分であることも指摘されている[1]

[1]: Aldar C.-F. Chan, Jianying Zhou, One smart grid cybersecurity standardization: Issue of designing with NISTIR 7628 , IEEE Communications Magazine, Vol.51, Issue 1, pp.58-65, January 2013.

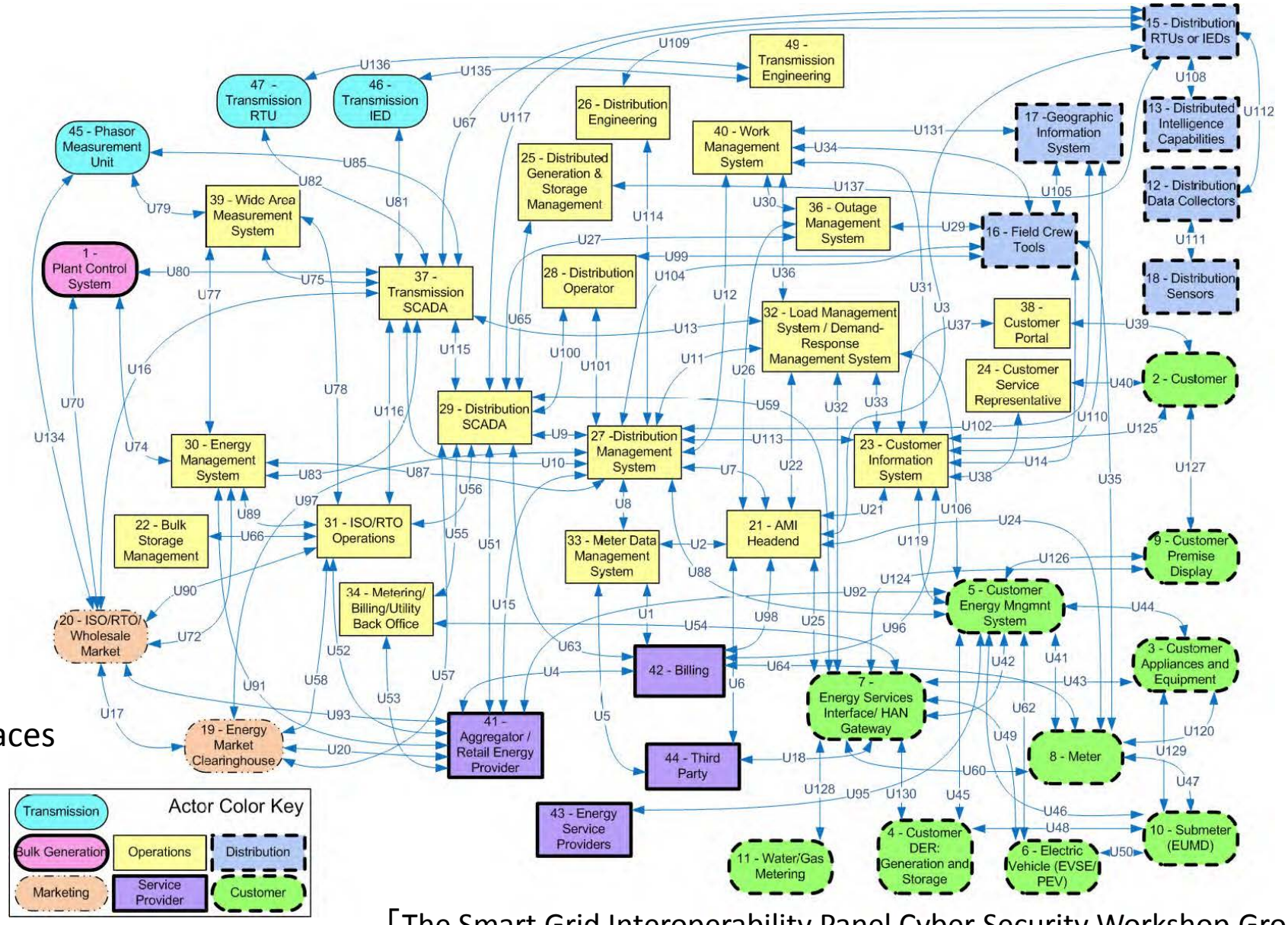
NISTIR 7628

ロリスク評価とセキュリティ分析：

トップダウン・アプローチとボトムアップ・アプローチの併用

- トップダウン・アプローチ
  - アクター（device, computer system, software program, individual, organization）をつなぐ論理的なインターフェースは分類されており、分類ごとにセキュリティ要件が設定される。
  - このセキュリティ要件で防ぎきれない攻撃は、ボトムアップ・アプローチで対応することになっている。
- ボトムダウン・アプローチ
  - 明確に定義しにくい問題を対象としており、設計者の経験に基づいて定義される非自明な要件をケース・バイ・ケースで対応するアプローチ

Logical Reference Model of NISTIR 7628



49 Actors
7 Domains
133 Logical interfaces
22 categories

「The Smart Grid Interoperability Panel Cyber Security Workshop Group, Introduction to NISTIR 7628 Guidelines for Smart Grid Cyber Security, September 2010」より

トップダウン・アプローチ： NISTIR 7628

1. 論理インターフェースを決定
2. 論理インターフェースが属する分類を検索
3. CIA(Confidentiality, Integrity, Availability)に関する優先度（Low, Moderate, High）とセキュリティ要件を定める
   - セキュリティ要件の例：
     SG.AC-12: Session Lock
     SG.AC-14: Permitted Actions w/o Identification or Authentication
     SG.IA-04: User Identification and Authentication
     SG.SC-03: Security Function Isolation
     SG.SC-05: Denial-of-Service Protection
     SG.SC-06: Resource Priority
     SG.SC-08: Communication Integrity
     SG.SC-26: Confidentiality of Information at Rest
     SG.SI-07: Software and Information Integrity

NISTIR 7628の問題点

- リスク評価とセキュリティ分析：トップダウン・アプローチとボトムアップ・アプローチの併用
- トップダウン・アプローチと問題点
  - アクター（device, computer system, software program, individual, organization）をつなぐ論理的なインターフェースは分類されており、分類ごとにセキュリティ要件が設定される。
  - このセキュリティ要件で防ぎきれない攻撃は、ボトムアップ・アプローチで対応することになっている。抜けが起りえる。
- ボトムダウン・アプローチ
  - 明確に定義しにくい問題を対象としており、設計者の経験に基づいて定義される非自明な要件をケース・バイ・ケースで対応するアプローチ

必ずしも、安全性を保障仕切れていない

EV(Electric Vehicle)とSmart Grid

- EVの充電：3.6-7.2 kW　（≈アメリカの家庭の夏の消費電力：3kW）

- 電力需要のピーク時：EVに充電された電気を売り戻す
  電力需要のピーク時以外： EVの充電を行う

スマートグリッド技術の必要性

NISTIR7628におけるEV ecosystemの課題

- Substitution Attackへの耐性がない[1]

(a) EVの正当性に関する
Challenge & Response認証

(b) Cardの正当性に関する
アクセス制御

IED(Intelligent Electronic Device)：通信用(a)とエネルギー制御用((b)に関連)の2種類のインターフェースを有する

図：文献[1]より

[1]: Aldar C.-F. Chan, Jianying Zhou, One smart grid cybersecurity standardization: Issue of designing with NISTIR 7628 , IEEE Communications Magazine, Vol.51, Issue 1, pp.58-65, January 2013.

NISTIR7628におけるEV ecosystemの課題

- Location privacyの保護が考慮されていない[1]
  - ➢NISTIR 7628では、Utility operatorが常に信頼できると仮定している
    ⇒　もし、Billing serversが信頼できないと、個人の位置を特定できる
      - user/EVのIDとcharging stationのID
      - 充電時間

□同様に、AMI(Advanced Metering Infrastructure)においても、個人情報を保護する問題が存在する。

- ➢Smart Metering Systemsにおいては、準同型暗号などの暗号技術を活用した、個々の消費量を隠しながら合計消費量を求める解決方法などが検討されている

[1]: Aldar C.-F. Chan, Jianying Zhou, One smart grid cybersecurity standardization: Issue of designing with NISTIR 7628 , IEEE Communications Magazine, Vol.51, Issue 1, pp.58-65, January 2013.

小型デバイスの安全な通信の研究：背景

- 省電力・低価格なマイコンや通信モジュールの登場・普及
- IoT対応を掲げた製品やサービスの登場
- 通信の安全対策需要の増加
- 各種組み込み向け暗号ライブラリの登場
- ハードウェア暗号機能を持ったマイコンの登場

IoTが連想させる事柄

- センサ等で情報を集め，解析し，人間の生活を支援するシステム
  - 例：睡眠の深さをセンシングし，起床30分前から暖房を入れ，起床をサポート
- センサ，解析システム，制御装置等が相互に通信しあい，目的を達成する
  - どれか一つだけではだめ
- セキュリティの確保が最重要
  - 機器や通信経路が安全でない場合，個人に直接の被害が発生し危険

IoT機器の安全対策手法

- 処理能力の高いものは対策が（比較的）容易

- 処理能力の低いもの（センサノード等）はどうする？
  - マイコンのハードウェアセキュリティ機能を用いる(後述)
  - ソフトウェア処理で対応する
    - 軽量暗号の利用
    - 組み込み向け暗号ライブラリの利用（後述）

ARMマイコンのセキュリティ対策

- 今まで（ARMv6M~v7Me（Cortex-M0 ~ M7））
  - コアの機能として暗号化は無い
  - 任意でつけられる内蔵または外部セキュリティ機能を使用する

- これから（ARMv8M）
  - TrustZone を用いた権限分離による安全対策
  - CryotoCellによる各種暗号機能の提供
    - 乱数生成，セキュアブートなど

NaCl(Networking and Cryptography library)

- 楕円曲線DH鍵共有などを備えた暗号ライブラリ
  - 鍵共有　　：楕円曲線DH鍵共有(Curve25519)
  - 署名　　：エドワーズ曲線デジタル署名(Ed25519)
  - ストリーム暗号　　：Salsa20
  - MAC　　：Poly1305
- 製作者はBernsteinとShwabe
- 暗号化から署名まで1つのAPIで行える
- マイコン向け実装（μNaCl）も開発中
  - AVR　　：Shwabeらにより実装済み
  - ARM Cortex-M3　：西永らにより実装済み
  - ARM Cortex-M0　：西永らにより実装済み

NaClの安全対策（一例）

攻撃	原因	対策
タイミング攻撃	秘密情報に依存する分岐を用いた高速化	秘密情報に依存した分岐を除去
キャッシュタイミング攻撃	データロード時のインデックスが秘密情報に依存していた.	秘密情報に依存したロードを除去

※タイミング攻撃： 処理の実行時間差を用いてパスワードなどを推測する攻撃

Shwabeらの行った高速化手法（一例）

- 高速剰余算計算
  - mod 2^{130}-5
  - mod 2^{255}-19
- 3段カラツバ法を適応した256-bit乗算

剰余算計算

- Poly1305などで用いるメルセンヌ風素数剰余算の式を
  シフト と 加算 のみにすることで高速に処理する
- 例
  - $x\ mod\ 2^{130} - 5$, (xは256-bitの整数)
    - $x = x_1 \cdot 2^{130} + x_0$ に分解
    - $2^{130}\ mod\ 2^{130} - 5 = 5$ より，$mod\ 2^{130} - 5$ 上において $2^{130} \equiv 5$
    - $2^{130} \equiv 5$ より $x = x_1 \cdot 2^{130} + x_0$ を $x \equiv x_1 \cdot 5 + x_0$ に変形
    - $5x_1 = (1 \ll 4)x_1 + x_1$ より，$x \equiv (1 \ll 4)x_1 + x_1 + x_0$
  - $x\ mod\ 2^{255} - 19$, (xは512-bitの整数)
    - $mod\ 2^{256} - 38$ を用いて同様の計算を行った後, $mod\ 2^{255} - 19$ を行い整合を取る

3段カラツバ法を適応した256-bit乗算

- カラツバ法
  - 大きな値の乗算の一部を低コストな加減算に変換することで，全体の実行時間を減らす手法
- 前提
  - Cortex-M0の乗算器は入力と出力が32-bitのため16-bit乗算器とみなす
- 特徴
  - 256-bit 乗算を 256, 128, 64-bitの3段に分けカラツバ法で計算
  - 2の補数を用いて符号判定処理から条件分岐を除去し，安全性と速度を向上
- 効果
  - 筆算方式に比べ，7,000サイクル高速

大きな乗算値の計算方法(n-bit x n-bit)

- 筆算方式

$A = 2^{n/2} \cdot A_h + A_l$

$B = 2^{n/2} \cdot B_h + B_l$

		A_h	A_l
	$\times$	B_h	B_l
		$A_h B_l$	$A_l B_l$
	$A_h B_h$	$A_l B_h$	
	$A_h B_h$	$A_h B_l + A_l B_h$	$A_l B_l$

より,

$$A \cdot B = 2^n \cdot A_h B_h + 2^{2/n}\left(A_h B_l + A_l B_h\right) + A_l B_l$$

- カラツバ方式
  - $L = A_l \cdot B_l$
  - $H = A_h \cdot B_h$
  - $M = (A_l - A_h) \cdot (B_l - B_h)$

のとき

$$A \cdot B = 2^n \cdot H + 2^{2/\mathrm{n}}(L + H - M) + L$$

筆算方式に比べ,
加減算が増えるが乗算を減らせる

2の補数を用いた条件分岐の除去(256-bit)

- 一般的なカラツバ法は以下
  - $A = 2^{128} \cdot A_h + A_l, \quad B = 2^{128} \cdot B_h + B_l, \quad L = A_l \cdot B_l, \quad H = A_h \cdot B_h$
  - $M = (A_l - A_h) \cdot (B_l - B_h)$
  - $A \cdot B = L + 2^{128}(L + H - M) + 2^{256} \cdot H$
- Mを以下のように扱うと，符号に寄らず処理を一定にできる
  - $|M| = |A_l - A_h| \cdot |B_l - B_h|$
  - 各減算処理後の符号は一時変数tにXORをしつつ格納する
    - ＋なら0，−なら0xffffffffとなる
  - 最後|M|全体をtでXORし，−1 ＊ tの値を|M|に足すことで2の補数を取る

参考: https://cryptojedi.org/peter/data/space-20141020.pdf

西永らの成果[2]

- Cortex-Mシリーズで”すべての機能が”動作するμNaClの実装
- Thumbアセンブリによるサイズを抑えた実装
- サイズと速度のバランスを保ったM0NaClの取り込み
- SRAMの初期値とSalsa20を用いた乱数生成機能の追加

[2] 西永俊文, 満保雅浩, μNaClの32-bit ARM Cortex-M0への実装, Proceedings of the 2016 Symposium on Cryptography and Information Security, SCIS2016, 2C4-5 (January 20, 2016).

M0NaClのコード取り込み

- 課題
  - DullらのCurve25519実装は非常に高速である.
    しかし，コードサイズが大きい
    - 256-bitの 乗算 と自乗算 を分けて高速実装
    - Subtractive Karatsuba法を用いた256-bit乗算を
      コードの再利用なしに全てアセンブリで実装
- 対応
  - サイズを削減して取り込む
    - 256-bitの 乗算 と自乗算 を統一
    - 256-bit乗算から128-bit乗算を関数化し，
      再利用する方式に変更

今後の課題

- 乱数生成機能の安全性強化