紀要 専電研

Vol.05

令和４年11月

全国専門学校電気電子教育研究会

目 次

AI のモジュール化による帰納的学習方法の実現性について

日本工学院専門学校 テクノロジーカレッジ 電子・電気科
後藤 武尊
佐藤 優樹

要約

本研究は，先ず人工知能，機械学習，深層学習の位置付けをサーベイし，各々の定義を再確認した.その結果，人工知能(AI)を広義の意味で捉えることにより，Raspberry Pi による AI のモジュール化という着想を得た.この妥当性を検証するためにチャットロボットを製作し，その実現性について検討を行った.その結果，従来の演繹的な AI に関する学習方法だけでなく，AI をモジュールとして活用する帰納的学習方法も有効であるということがわかった.

1. はじめに

現在，「スマート」という接頭語がつくエレクトロニクス製品には，人工知能や音声認識機能が搭載されている.そもそも人工知能は，人間の脳を構成するニューロンを人工的に再現することを目指しており，情報の拡張性という意味で，ネットワーク技術とは親和性が高い(Goodfellow et al, 2016).

ここで情報の拡張性を階層的に捉えると，人工知能が機能するためには，機械学習が必要で，更に，より人間の思考に近づけるためには深層学習が必須となる.この様に，人工知能という領域自体がネットワーク性を持つため，我々は，つい学術の深淵に嵌まってしまう(松尾, 2015).

次に，プログラミング視点で人工知能を捉えた場合，機械学習に関する書籍があまりにも多いことに驚く.その為，人工知能，機械学習，深層学習，一体何から手を付ければ良いのか迷ってしまう(ピックオーバー, 2020).人工知能に関する書籍の執筆者は，学者からプログラマー，あるいは経営コンサルタントまで，その経歴やバックグラウンドは多岐に渡っている.従って，執筆者各人のこれまで学習してきた過程や方法が全く異なるため，当然ながら書籍に記載されている内容も各々異なっている(松尾，2015; ピックオーバー，2020; Raschka & Mirjalili, 2020; 猪狩 他, 2021).実は，この情報量の多さが，人工知能や機械学習の初学者にとって，非常に混乱を招く原因となっているのではないか，というのが本研究のリサーチ・クエスチョンになっている.つまり，漠然と興味本位で人工知能を学ぼうとする初学者にとって，「機械学習，深層学習は難しい(閾が高い)」と見えてしまうのである.そこで本研究は，先ず「人工知能とは一体どの様なものか」という，全体像を掴む目的でサーベイを実施した.その結果，Raspberry Pi (Raspberry Pi, 2022)というモジュールを活用することによって，人工知能を，その仕組みから理解し積み上げていく演繹的学習法ではなく，人工知能を活用することから始める帰納的学習法が有効なのではないかについて，その実践を試みたことについて報告する.具体的には Raspberry Pi を AI モジュールとして活用し，チャットロボットを製作した.

2. 人工知能とは何か －人工知能に関するサーベイ－

現在，人工知能(AI)，機械学習(ML: Machine Learning)，深層学習(DL: Deep Learning)という言葉が巷で氾濫しているが，その違いは曖昧である.そこでこれまでの AI に関する学術研究の経緯に立って整理すると，AI, ML, DL の定義は図 1 の様になる.

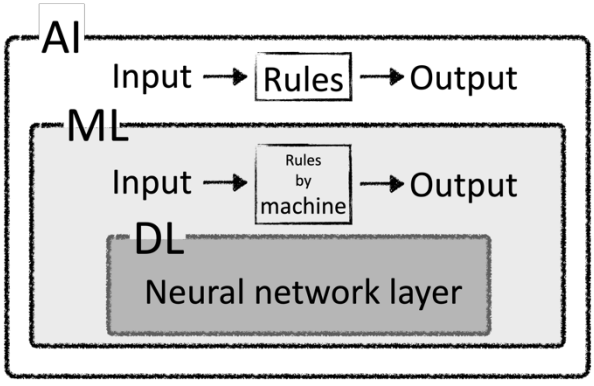

図 1.人工知能，機械学習,および深層学習の定義

AI という言葉は，最近登場した様に思われるが，その研究の根源は 1950 年代である.(Dartmouth, 1956).近年，AI という言葉がブームになっているが，実は第 3 次 AI ブームであり，このブームを支えているが深層学習で，これが巷の人工知能ブームの真相である.

図 1 を基に，AI を広義的に定義すれば，ある「決められたルール」に従ってインプットを行うと，アウトプットを返すものとなる.具体的には，電卓などが相当する.我々が何らかの数字を電卓に入力すると，「四則演算というルール」によって，計算結果を返していると考えることができる.

そして，AI という領域の中に機械学習(ML)という領域が存在し，ML は，ルール自体を機械が決めるものを指す.具体的には，機械に何らかの過去のデータを入力することによって，そのデータを基に機械がルールを決定する.そして出来上がったルールに対してインプットすることによって特定のアウトプットをかえすという仕組みである.従って，敢えて AI と ML の違いを示すならば，AI におけるルールは人間が決めてもよいのに対し，ML ではルールを機械が決める，として両者を分けることができる(松尾， 2015; ピックオーバー, 2020).

そして，機械学習領域の中に，更に深層学習(DL)という領域がある.厳密に言えば，機械学習におけるひとつの手法が深層学習となる.また，ML における学習方法のひとつにニューラル・ネットワークがあり，この手法を更に複雑にしたものが DL に相当する.具体的にはニューラル・ネットワークを層構造にすることによって複雑な問題を解ける様にしている.AI と ML の場合と比べ，ML と DL との間には大きな違いはない(ピックオーバー, 2020).一般的に ML は，人間が特徴量，すなわちインプットするデータの意味付けを行うのに対し，DL は特徴量が不要な場合がある，ということが挙げられる.具体的な例を使って説明すれば，商品棚にバナナ，リンゴ，パン，ヨーグルト，チョコレートが並んでいて，どの様に陳列すれば売れ行きが上がるかという分析について考える.ML の場合，先ず，「バナナの単価」，「バナナの売上げ」，「バナナの陳列位置」などの情報をインプットする.以下，他の商品についても同様に情報をインプットする.つまりある程度，売れ筋のパターンというルールができる程度に変数(特徴量)を決めてやる必要がある.一方，DL は棚の情報をまるごと画像として認識し，それをピクセル情報としてインプットできるため，特徴量を分けずに最適解を得ることができる.この DL の特徴を活かし，2019 年には Banerjee ら(2019)がノーベル経済学賞を受賞している.

以上，AI, ML, DL の違いについて簡単に説明してきたが，その定義をまとめると;

深層学習⊂機械学習⊂人工知能

と記述することができる.

この「広義の AI」という考えに基づき，本研究の位置付けについて以下に述べる.

2-1. チャットロボット製作の動機

なぜチャットロボットを製作したかという理由については，筆者のひとりが，中学時代に体験したことに在る.ある体験コーナーで，日本語を入力すると自動で英語に翻訳された言葉が返ってくるキャラクターの応答に魅せられたのである.それ以来，この様なロボットをいつか自分で作ってみたいと思うようになり，それが専門学校への進学に繋がっていった.そして，チャットロボットが一体どの様な技術で動作しているのかについてリサーチした結果，前述の AI と呼ばれる学術領域に自分の関心があることがわかった.つまり AI の仕組みより，AI の使い方に関心があるのである.これにより，AI を自作することを目的として，限られた時間内での製作に臨む場合，AI を用いて如何に多くのことができるかを目指すより，寧ろ機能をコンパクトにし，期限内に自作できる範囲を明確にすることによって実現可能な設計方針を立てることが可能となった.

2-2. モジュールとしての AI

限られた時間内で製作物を完成させる為には，ゴールまでにいくつのプロセス(ハードル)があるかを事前に見積もることが重要になる(加藤 他, 2019).

チャットロボットの製作実現に関して，指導教員と共にその実現性を見積もった結果，本研究では，AI を，その仕組みから紐解く時間を捻出でき

ないという問題が生じた.この問題を解決するため，本研究では，AI 技術自体をひとつの「技術モジュール」として扱い，その組み合わせによって所望の動作を実現する，アジャイル型アプローチ（Rubin, 2012）を採用した.

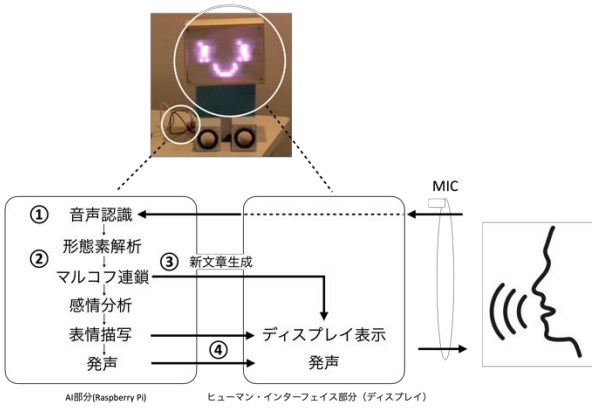

図 2. チャットロボットのシステムブロック図

一般に，チャットロボットの文書生成にはさまざまな方法がある(Goodfellow et al, 2016).その中で，機械学習を用いる方法は，相手が話した内容に対して高確率で的確に返答できるというメリットがある一方，機械学習による推論には，相当の高負荷処理が必要になるというデメリットがある.そこで，限られた製作時間の中で確実にチャットロボットを完成させる為には，各々の処理がモジュールあるいはユニット毎に完結している，モジュラー型アーキテクチャ(Ulrich, 1995)が適切であるという判断に至った（後藤 他, 2021）[1].

出典: https://www.switch-science.com/catalog/4110/

図 3. Raspberry Pi 3 Model A+

図 2 に本研究のシステム概略を示す.チャットロボットが成立する最小限の機能として;

人間が話した文章をマイクで拾い，①それを AI が「音声と認識」し，②文書生成処理を行なう.③そこで得られた文章中の単語を組み替えることで新しい文章を生成し，処理の応答として，④得られた新文章を音声および感情表現で返す.
という一連の動作になる.ここで，音声認識から発声に至るまでの処理を Raspberry Pi に行わせることによって，AI のモジュール化を試みた. 具体的には，人間が話した会話（文章）をマイクで拾い，形態素解析とマルコフ連鎖による文書生成処理を Raspberry Pi [2]の中で行う.そして，得られた文章内の単語を組み替えることによって新しい文章を生成する，という手順である.この様に，AI エンジン部分を Raspberry Pi としてモジュール化することによって，文書生成・音声認識・ロボットの発声をスタンドアローンで処理することが可能となった.これは正に，前述した広義の AI に相当する.

3. チャットロボットの設計

本研究の設計プロセスを大きく分けると，人工知能を司るソフトウェア設計と，表情付き対話応答を司るヒューマン・インターフェイス設計（ハード設計）になる.以下，順に説明していく.

3-1. AI の設計プロセス

図 4 は，AI 部分の設計プロセスを示している.AI に実装する機能を感情対話表現に絞り，その一連の動作表現に必要な「音声認識」，「形態素解析」，「マルコフ連鎖」，「感情解析」，「表情描写」，「発声」を技術モジュールとして取り入れた.

3-2. 音声認識

音声認識には，オフラインで使用できる音声認識ライブラリ”Julius”を使用した（Julius, 2022).Julius に付属しているディクテーションキットと音声認識パッケージを用いて，Raspberry Pi に接続する USB マイクから入力された音声を「文字おこし」する.Julius は他の音声認識プログラムとは異なり，音声を import する場合，「モジュールモード」で起動する必要がある.Julius は，モジュールモードでは仮想音声認識サーバーとしてコンピューター内に構築される.そして，ポート番号や IP アドレスが割り振られ，クライアントと TCP/IP 接続

[1] モジュラー型アーキテクチャに関しては巻末の Appendix を参照のこと.

[2] 本研究では Raspberry Pi 3 Model A+を採用.

を行いながら音声処理を行っていく.よってロボットと会話する際，メインのプログラムとは別に，仮想音声認識サーバーを起動するプログラムをバックグラウンドで動作させる必要がある(Julius, 2022).

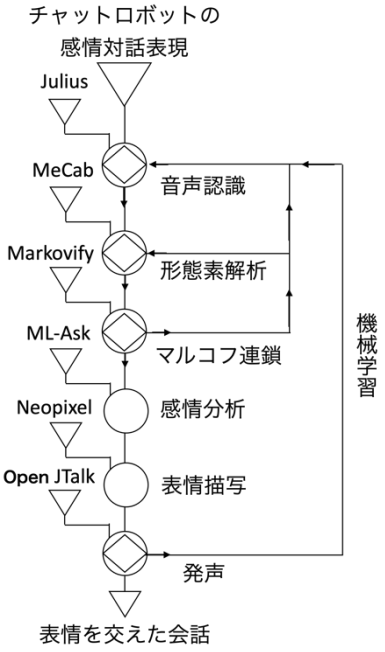

図 4. 本研究の設計アルゴリズム(AI モジュール)

3-3. 形態素解析

音声を認識した後，今度はそこから得られた情報を形態素解析する.形態素解析とは，構成されている文章内の単語を，名詞，動詞，助詞など，品詞として意味を持つ最小単位に分割し，各々の単語の品詞や活用を割り出す操作である.この技術は近年，テキストマイニング領域においてポピュラーな手法になっている(Silge & Robinson, 2018). 本研究では，テキストマイニング手法で最も汎用的である”MeCab”(MeCab, 2006)を形態素解析のエンジンとして採用した.そして，形態素解析で得られた最小単位の単語をそれぞれスペースで区切る，「分かち書き」を施す.”MeCab”には形態素解析用の辞書が付属しているが，特定の人物名や商品名等，固有名詞への対応を図るため，オープンソースで製作された辞書”mecab-ipadic-NEologd”(MeCab, 2020)を更に導入してより詳細な分かち書きを施した.

3-4. マルコフ連鎖

マルコフ連鎖とは，過去の状態によらず，現在の状況において，確率を決めて文章の遷移または推移を決めるアルゴリズムである.例を使って説明すると，「私は犬を飼っている」と「私はマグロのお寿司が好きだ」という，2 つの文章がある場合，冒頭の「私は」までは両文章ともに共通している.しかし直後の「犬」と「マグロ」で目的格が変わっていることがわかる.この場合，マルコフ連鎖では「私は」から次の品詞に遷移するとき，それぞれ 1/2 の確率で選択されると判断する.この時，マルコフ連鎖では言葉の意味や品詞自体を理解はしない.あくまで同じ語句を含んだ言葉があった場合のみ，両者を比較する.従って，意味の通らない文章を生成する可能性が高くなるが，ひと通り，形態素解析を終えることができれば，日本語以外の言語にも対応できるというメリットを享受できる.本研究では，マルコフ連鎖のライブラリとして”Markovify”(Markovify, 2015)を採用している.

3-5. 感情分析

ロボットに感情を表現させることは，本研究の遡及ポイントのひとつである.この場合，言葉から感情を分析する機能と表情を描写する機能，つまり，ロボットの「声と表情」が感情を表現する上で欠かせない.

感情分析については，Python 用感情分析ライブラリ”ML-Ask”(ML-Ask, 2017)を採用した.ML-Ask は，与えられた文章中の言葉を読み込み，「喜，怒，哀，怖，恥，好，厭，昂，安，驚」という，10 種類の感情を抽出できる.感情分析は ML-Ask に内蔵された感情表現辞典から行っており，収録されている言葉数はおよそ 2100 語である(ML-Ask, 2017).

3-6. 表情描写

ロボットの表情描写は，マイコン入りフルカラー LED テープ(Neopixel)を採用した.通常 LED(発光ダイオード)を配線する場合，抵抗も同時に配線する必要があり，配線作業は複雑になる.Neopixel の場合，直列に接続された LED テープを 3 つのケーブルで一括指定することができる.配線をシンプルにすることができる.

3-7. 会話データ

本研究で製作したチャットロボットは，USB マイク

を介して入力された音声データを文章化し，これを基に語句間の繋がりを推定し，新たな文章を生成したものを返事として返す.このアルゴリズムの場合，最初の会話時点において，ロボット内に語彙(ボキャブラリ)データをストアしていない為，話しかけても返答できない状態が生じる.この状態を回避するために，予め「こんにちは」など，簡単な話し言葉が含まれたテキストファイルを別途製作した(図 5)．

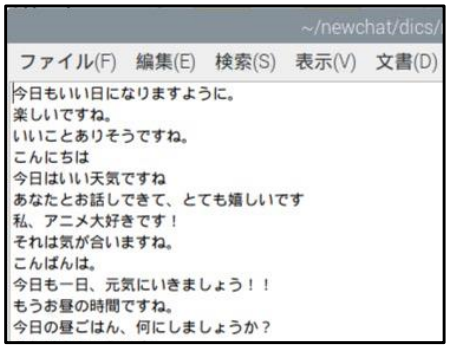

図 5. デフォルト値としてのテキストファイル

4. ヒューマン・インターフェイスの設計

本研究のロボット製作過程は大きく分けて，「AI 実装(プログラム製作/ソフトウェア設計)」と「ヒューマン・インターフェイス設計(外装設計/ロボット表示部の製作/ハードウェア設計)」になる.設計を始めた時点では，機能の搭載如何によってボタン配置等のハード設計で変更を加える必要があるため，前述した AI 実装設計の後，本設計に着手した.

4-1. ロボット表情表現の製作/外装製作

ロボットの顔ともいうべき表情表現部分(外装製作)は，図 6 に示す様に，LED 点滅を変化させることによって，感情を表現している[3] [4].

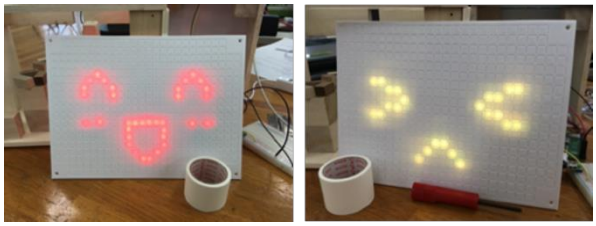

図 6. 製作したチャットロボットの感情表現

LED で点灯する部分を 3D プリンターで，その他の部分はプラ板もしくは木材を加工して製作した.

4-2. LED による表情の描写

ロボットの顔部分については，3D-CAD を用いて設計を行った(図 7)．

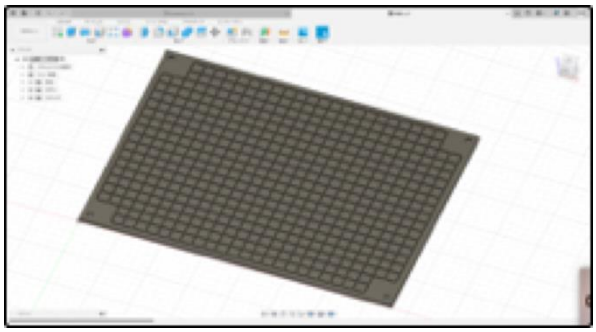

図 7. CAD による LED プレートの設計

設計データを基に，3D プリンターを用いてカスタム設計したものが図 8 になる.設計したプレートは，LED を点灯させると，ドット自体が顔の輪郭を描写し，かつ表情自体も表現する.従って，顔の輪郭や表情をより豊かに表現するために，プレートには，メッシュ状の溝を施している.

そして，このプレート裏面に Neopixel が貼り付けられている.ここで LED の間隔を調整するために，LED テープを山折り・谷折り構造にした上で，配線コードをハンダ付けし，LED テープは直列に接続されている.Neopixel は色の一括指定や，各 LED を指定して点灯・消灯させることが可能であるため，顔の表情は Excel を用いて描写し，出力された2進数コードを Python プログラムに組み込んで表示している(Python, 2017).

図 8. メッシュ溝構造を有する LED プレート

4-3. 組立て

その他，チャットロボットに必要な建材は，適宜 3D プリンタ，プラスチック板，フェルトを用いており，これらを組み合わせて構築されている.図 9 にチャットロボットの完成形を示す.

[3] https://qiita.com/hamahamabe/items/9216df345bb908eb5199

[4] https://muscle-keisuke.hatenablog.com/entry/2017/12/13/125951

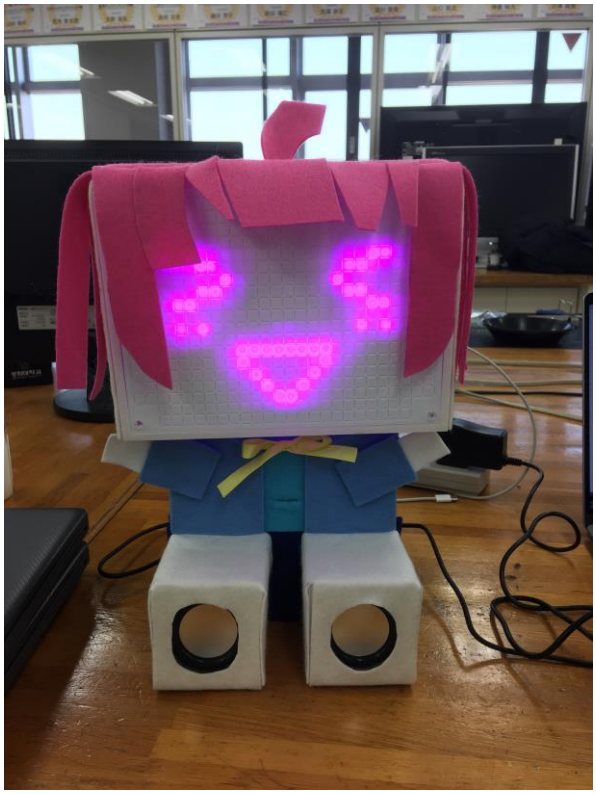

図 9. 完成したチャットロボット

顔の部分については，LED プレートと 3D プリンタ側に穴を設け，ビスで組み込んでいる.胴体部分については3D プリンタとプラスチック板を用い，接着加工で固定している.最後にフェルト生地で，LED プレートとスピーカー以外の部分を覆い，外観を整えた.

5. 問題点−設計と実装のギャップ

ここまで，チャットロボットの設計から製作に至るまでの過程について説明を行った.本項では，設計と実装のギャップとして，経験したいくつかの点について述べる.

人工知能の導入について

「人と会話するロボット」を作る上で，ML を用いたプログラム設計については以前から構想を温めていた.しかしながら，いくつかの理由により断念せざるを得なかった.主な理由として，Raspberry Pi の性能不足が挙げられる.ML は，主に GPU（グラフィックス・プロセッシング・ユニット）を使用するため，GPU を搭載しない Raspberry Pi では機械学習による推論は難しい.この事を踏まえて，狭義の意味での人工知能が使える様に，Raspberry Pi から別のサーバーにインターネットでアクセスし，サーバー内に構築した人工知能と通信する，という別の方法を模索していた.この場合，オフライン環境でも使用できるというメリットは失われるが，ML による，もっと人間の会話に近い会話が実現できるため，筆者のひとりが中学時代に体験した感動再現により近づけることが期待できる.

6. まとめ

本研究は，先ず AI に関するサーベイを実施することによって，広義の AI に着目し，Raspberry Pi による AI のモジュール化を図った.一般に「機械学習を実装する」ことを実現しようする場合，その仕組みから理解しなければならないと考え，学ぶことの多さに挫折する危険性がある.その意味で，AI の全体像を把握するという目的で，サーベイは非常に有効であった.また，本研究で示した様に，AI をモジュールとして扱うことによって，限られた時間内で確実にモノを仕上げるという「デリバリー感覚」を体得できたことは今後の学際活動においてともて有益と思われる.

本研究の出発点は，AI の仕組みを学ぶことではなく，AI を活用する，楽しむことにあった.このグランドデザインを実現するために，AI をモジュール化するという帰納的思考が設計製作プロセスに有効であることが確かめられた.

最後に，本研究から得られたインプリケーションは，「学習という，一種の努力を継続する環境をいかに構築するか」，ということである.努力を継続できる環境には;

1. 良質なコンテンツ
2. 良質な学習環境
3. 如何に努力を継続できるか

が必要である（山本, 2020; 日本工学院専門学校, 2021）.

1.については，Raspberry Pi の存在がとても大きかった.改めて，標準化技術，モジュール化のもつバーゲニング・パワーを実感できた.2.については，暖め続けてきた構想を実現できる実践教育の場の存在が先ず挙げられる.そして，先生方から直接指導を受けられたということが，本研究の実現に繋がっている.3.については，本研究をきっかけとして，今後の ML の本格的な実装への足がかりとする予定である.

謝辞

本研究を進めるにあたり，日本工学院専門学校テクノロジーカレッジ 電子・電気科 三須健吾先生，知久雅治先生，森田秀之先生には，有益なアドバイスを頂きました.ここに謝意を表します.

Appendix

A-1. アーキテクチャという考え方

おそらく「アーキテクチャ」という言葉を聞いた時に，多くの人が最初に思い浮かべる分野は，建築分野における構造物だと思われる.Paraft(2016) は，建築業界では，建設する行為そのものを，”construction”や”build”とし，建設行為だけでなく設計やデザイン性も含めた「概念」を”architecture”と称し，IT 業界のそれとは意味が異なると説明している.また，吉田&野城(2005) は，建築を人工物とした上で，モジュラー/インテグラル型アーキテクチャを説明し，建築プロセスと作業側との情報の擦り合わせについて分析を試みている. 本論文における「アーキテクチャ」は，主に技術経営(MoT)領域におけるコンテキストで用いている.その理由は，複雑なシステムは「多様に関連し合う多くの部分から成る人工物(artifacts)」であるという Simon(1969)の考えが根拠となっているからである.この考えに基づくと，製品は，時間の経過と共に構成要素やその関係性が変化していくシステム構造を持つと定義できる.つまりシステムが複雑になるとは，「システムを構成する要素（モジュール）の数が増加する」，「構成要素間の関係性が増加する」，ことを意味する.ここで，物事を分割する際の「境界」について，その隔たり構造に着目すれば，境界は社会学における組織セクショナリズムや，エンジニアリング領域における専門分野と言う論点を与える.また，境界面で互いにどの様なつながり方をしているのかに着目すれば，アーキテクチャ理論におけるインターフェイスという視点を与えることになる(Ulrich, 1995; Baldwin & Clark, 2001).

A-2. インテグラル型アーキテクチャとモジュラー型アーキテクチャ

Simon(1969), Ulrich(1995) を基に，モジュール間のつながりに着目するのが，アーキテクチャという考え方である.例えば，製品を構成する機能と構造をどの様につなぐかに関する基本的な設計構想は製品アーキテクチャとなる.

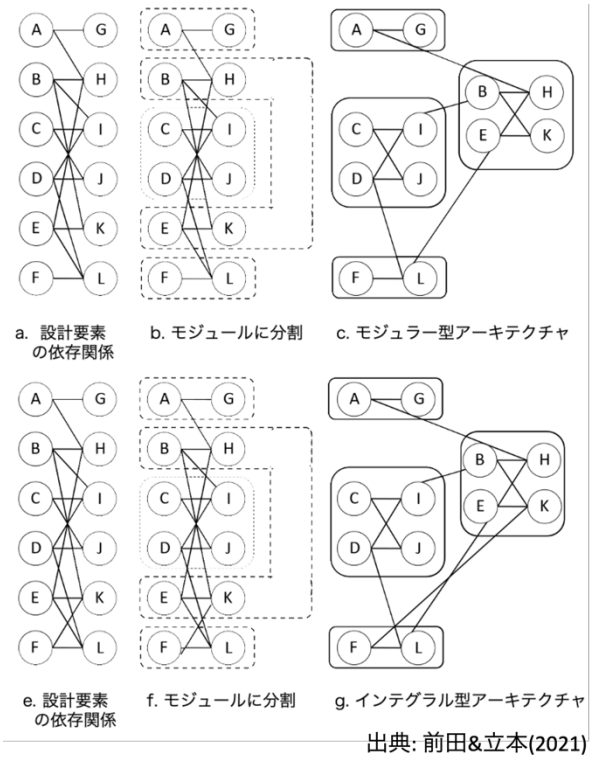

図 10. モジュラー型アーキテクチャとインテグラル型アーキテクチャ

図 10 は，製品アーキテクチャの視点でシステム設計要素における依存関係の違いを示したものである.同図(a)の様な依存関係にある設計要素を，つながり毎にモジュールにまとめると，(c)の様になる.モジュールに分けられた 4 つのブロックは，それぞれが 1:1 という，シンプルな接続をしていることがわかる.これをモジュラー型アーキテクチャという.

これに対して，図 10-(e)の様な依存関係にあるシステムをモジュールに分割し，まとめると(g)の様な，各モジュール間に複数のつながりが存在する，4 つのブロックになる.これをインテグラル型アーキテクチャという.Baldwin & Clark(2001) は，このつながりの形(インターフェイス)に着目し，「モジュール間の連結ルールが決まると，個々のモジュールの設計や改善は，他のモジュールから自立して行われる」という「デザイン・ルール」(Baldwin & Clark, 2001) を示し，インターフェイスの重要性を主張した.これは，インターフェイスが定義できれば，それと切り離してモジュール内を独立に進化させることが可能になることを意味している(Baldwin & Clark, 2001; Langlois & Robertson, 1992; Fine, 1998; 立本, 2013).

参考文献

後藤武尊，佐藤優樹，三須健吾，知久雅治，（2021）．プライベート・ディスカッション，卒業研究， 11 月．

猪狩宇司，今井翔太，江間有紗（2021).「深層学習教科書 ディープラーニング G 検定公式テキスト」，第 2 版，翔泳社.

加藤勇夫，太田結隆，越島一郎（2019). リーン&アジャイルマネジメントプログラムマネジメントに関する基礎的考察，Journal of International Association, Vol.13, No.2, pp.60-80.

前田篤志，立本博文（2021). アーキテクチャ理論からみた技術世代変化への対応について，イノベーション・マネジメント，No.18，法政大学イノベーションマネジメント.

松尾豊（2015).「人工知能は人間を超えるか -ディープラーニングの先にあるもの-」，角川書店.

日本工学院専門学校（2021). 教育設計図，https://www.neec.ac.jp/education/career_design/（2022 年 3 月 20 日確認）.

ピックオーバー，C（2020).「人工知能-グラフィック・ヒストリー」，ニュートンプレス.

立本博文（2013).「アーキテクチャ研究再考」，「人工物」複雑化の時代-設計立国日本の産業競争力-，藤本隆宏 編，pp.133-168，有斐閣.

山本祐平（2020).「学習環境のイノベーション」，東京大学出版会.

吉田敏，野城智（2005). アーキテクチャ概念による建築の設計，生産システムの記述に関する考察，日本建築学会計画系論文集，第 589 号，pp.169-175.

Baldwin, K.Y.and Clark, K.B. (2001).“Design Rules, The Power of Modularity Volume 1”, MIT Press, 邦訳「デザイン・ルール -モジュール化パワー」，東洋経済新報社, 2004.

Banerjee A, Duflo, E, and Kremer, M (2019). “For their experimental approach to alleviating global poverty, ”「開発途上国の貧困対策への実験的アプローチ」, 2019 年ノーベル経済学賞.

Dartmouth, (1956). The Dartmouth Summer Research Project on Artificial Intelligence.

Fine, C.H.(1998). Clockspeed: Winning industry control in the age of temporary advantage. Massachusetts: Perseus books.

Goodfellow, I, Bengio, Y, and Courville, A. (2016). “Deep Learning”, The MIT Press.

Julius (2022). https://github.com/julius-speech/julius（2022 年 3 月 18 日確認）.

Langlois, R.N, and Robertson, P.L. (1992). Networks and Innovation in a Modular System: Lessons from the Microcomputer and Stereo Component Industries, Research Policy, Vol.21, No.4, pp.297-313.

Markovify (2015). https://github.com/jsvine/markovify（2022 年 3 月 19 日確認）.

MeCab（2006). http://taku910.github.io/mecab/（2022 年 3 月 19 日確認）.

MeCab (2020). https://github.com/neologd/mecab-ipadic-neologd（2022 年 3 月 19 日確認）.

ML-Ask (2017). https://github.com/ikegami-yukino/pymlask（2022 年 3 月 19 日確認）.

Paraft (2016). https://paraft.jp/r000016001893（2022 年 3 月 20 日確認）.

Python (2017). https://sandmark.hateblo.jp/entry/2017/10/07/141339（2022 年 2 月 28 日確認）.

Raschka, S. and Mirjalili, V. (2020).「Python 機械学習プログラミング -達人データサイエンティストによる理論と実践」，第 3 版，インプレス.

Raspberry Pi (2022). https://www.raspberrypi.com（2022 年 3 月 19 日確認）.

Rubin, K, S (2012). “Essential Scrum: A Practical Guide to the Most Popular Agile Process”, Addison-Wesley Professional.

Silge, J., and Robinson, D. (2018).「R によるテキストマイニング -tidytext を活用したデータ分析と可視化の基礎-」，オライリージャパン.

Simon, H. (1969). ″The Sciences of the Artificial″, MIT Press.

Ulrich, K.T. (1995). The Role of Product Architecture in the Manufacturing Firm, Research Policy 24 419-440.

ネットワークオーディオシステムのモジュール化について

日本工学院専門学校　テクノロジーカレッジ　電子・電気科
稲垣映人
佐藤優樹

要約
本論文は、ネットワークオーディオシステムをモジュール化したことについて報告する。モジュール化の目的は、視聴者が楽曲コレクションを最適に鑑賞できるように構築することにある。これにより、電源およびネットワークの自己完結化を特徴とするシステムが構築できた。

1. はじめに

我々が日々生きていく中で、リフレッシュというものは非常に大切である。仕事の繁忙期など、なかなか休みが取れない時こそ、リフレッシュは効果的である。ハイキング、ジョギング、映画鑑賞、のんびり過ごす等、人それぞれにリフレッシュの形態は異なる。その中で、趣味を持つということもまた、自身の気持ちを切り替えるという意味では、とても有効である（乾&森田, 2021）。その中で蒐集趣味というものがある。蒐集とは、趣味・嗜好品を集め、保存・保管し、鑑賞を楽しむ一連の行為をさす。代表的な例として、硬貨蒐集や切手蒐集が挙げられる。もっと長い時間軸で見れば、美術館や博物館に展示されている美術品や古文書、化石などの蒐集は、美術館や博物館が行うことによって成り立っており、人類の文化・科学の発展に大きく貢献している。

音楽についても蒐集という行為が成り立つ。例えば、ビンテージ楽器や楽譜の蒐集などである。記録された楽曲の蒐集という意味では、レコードの蒐集に端を発して、音楽番組をテープに保存・収集（エアチェック）したり、CD(Compact Disc)を集めるのもまた蒐集とみなせる[1]。

これまで、蒐集したレコードやCDは、プレイヤーという媒体を介して視聴していたが、ICT 技術の発展によって、楽曲の保存形式やその視聴形態は多様化してきた。たとえば、購入した CD をリッピング[2]し、物理メディアを伴わないデジタルデータとして蒐集したり、インターネットを介して音声を送受信、あるいは楽曲データのダウンロード販売や音楽ストリーミングサービスが出現したりしている[3,4]。楽曲データのデジタル化と通信技術の発達によって、音楽を聴く行為が機器や場所に制約されないようになってきた。楽曲を、記録メディアに入ったものとしてではなく、楽曲そのものとして扱う時代になったと言える。

これらの変化により、オーディオを視聴するシステムにも変化が起きている。従来は、自宅に設えたオーディオシステムが、コレクションを楽しむ唯一の方法だった。携帯カセット・CD プレイヤーや携帯音楽プレイヤーによるポータブルな音楽視聴も可能ではあったが、蒐集家は記録時間・容量などの制約[5]により自身のコレクションから厳選したものだけを手中に収めて楽しまなければならなかった。これが通信の高規格化によって、随時、ネットワークにアクセスできるようになったため、ネットワーク上に音楽ファイルを保存すれば、いつでも、どこでも、あらゆるデバイスで視聴の嗜好、気分に合わせて選択視聴することができる。

2. ネットワークオーディオシステム

一般に、ネットワーク上に置かれた音楽ファイルを嗜好に合わせて必要なものだけを端末にダウンロードして視聴する方式をネットワークオーディオシステムという。その構成は、楽曲データを保存・配信をおこなうサーバー、楽曲の選択・再生や楽曲情報の表示をおこなうコントローラー、楽曲データを実際に再生するプレイヤーからなり、これらを互いにネットワーク接続している。図 1は、一般的なネットワークオーディオの構成で、既存のネットワーク(図では上流ルーター)に対してこれらの機器を接続している。

---

[1] 筆者の一人も、大伯父が自宅の一室を丸々使って本棚にびっしりとレコードや CD を蒐集しているのを見たことがある。ある意味、蒐集という行為は、日常生活の中で当たり前であった。

[2] 記録メディアに保存されているデジタルデータをパソコンなどに取り込むこと。吸い出しとも。

[3] https://www.spotify.com/

[4] https://music.amazon.co.jp/

[5] LP レコードの制約に合わせてカセットテープの容量は 46 分であった。

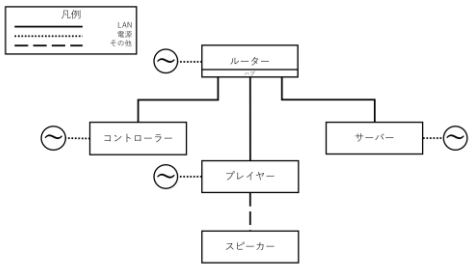

図 1. 一般的なネットワークオーディオの接続例

しかしながら、この構成は市場に出回っている機器が相互に連携するための枠組みに沿ったものであり、個々のケースの最適解は別の形を取り得る。よって、本論文ではネットワークオーディオを「自由に音楽を楽しむ」という目的で、その構成自体を再検討したネットワークオーディオシステムについて報告する。

2. 製作概要

2-1. 視聴者の嗜好にマッチするシステムの構築

前項で説明したように、楽曲の保存形式やその視聴形態は様々である。にもかかわらず、市販のネットワークプレイヤーに搭載されているソフトウェアは、従来の家電の組み込みソフトウェアの域を出ておらず、特にストリーミングサービス対応については遅れている。そこで、視聴者自身の嗜好に合わせたシステムを自作することで、視聴者の使い勝手を向上させることを設計目標とした。

2-2. ネットワークオーディオシステムのモジュール化

視聴者の環境をストレスフリーで実現するためには、USB モジュールの様に、当該モジュールとそれを動作させるシステム間のインターフェイスが規定されていることが重要になる。従って、オーディオシステムをモジュール化するためには;

① 独立した LAN: Local Area Network の中にシステムを収め、②全ての構成機器の電源供給を PoE(Power over Ethernet[6])にすることによって、モジュール化を実現した。本来の PoE の用途は、主に、電源供給の難しい箇所に設置するネットワーク機器の電源確保に用いられている。従って、本報告の様な PoE の用途は全く新しい使用方法となる。

更に、インターネットを通じてLAN外からもNAS(Network Attached Storage[7])内のデータへアクセスを可能とするために、VPN(Virtual Private Network[8])機能も実装した。

2-3. 構成

以上の要件を踏まえて、図 2 のような構想設計をした。

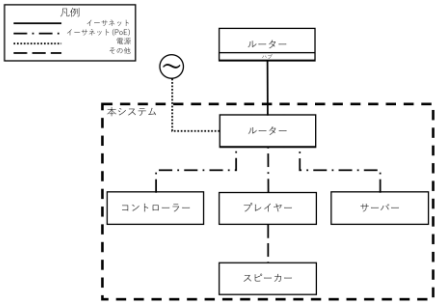

図 2. 提案システムの構想設計

図 1 に示した一般的な接続例に比べて、本システムから外に伸びる接続専用線は、イーサネット1本、電源1本になっている。これは、この2本のケーブルをインターフェイスとして、提案システムとルーターがモジュラー化(モジュラー型アーキテクチャ)(Ulrich, 1995)していることを示している。

なお、単にコントローラーで操作し、プレイヤーを使ってサーバーに保存されている楽曲を再生するだけであれば、本システムにおいても一般的なネットワークオーディオにおいても上流ルーターは必要ないが、多くのルーターに付属しているDHCP サーバーの機能は、各種機器を自動設定するならば必要になる。

---

[6] Power over Ethernet: イーサネットケーブル(UTP)で電力伝送をする規格。IEEE802.3af、IEEE 802.3at で標準化されている。

[7] Network Attached Storage: ネットワークに接続された記憶機器のこと。FTP や Samba(ファイル共有)などにより保存されたファイルへのアクセスを提供する。DLNA に準拠していれば、ネットワークオーディオにおけるサーバーとしてふるまうこともできる。

[8] Virtual Private Network: ネットワーク機器(VPN Client)と VPN サーバーの間で仮想的な接続を行い、仮想的な LAN の構築や VPN サーバーが存在するネットワークへの接続などを行う技術。

3. 製作手順

3.1 ルーターの環境構築・接続

まず、ルーターの環境構築をおこなった。今回の製作では独立した LAN 内にシステムを収めるが、最終的にインターネットに接続することを見据えてのシステム構築であるため、ルーターが必須となる。LANの外からVPNによりアクセスできるようにするためにVPNサーバーが必要になり、ルーターとVPNサーバーを兼用できるように、Raspberry Piをルーターとして環境構築を図った。

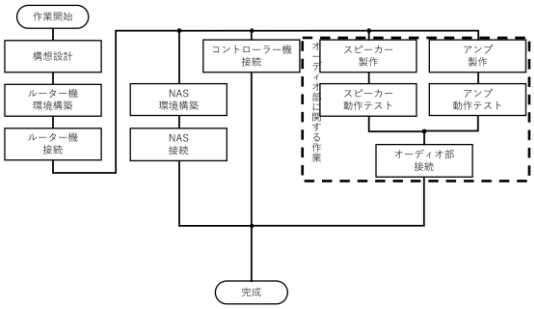

図 3. 製作フローチャート

3.2 NAS 環境構築・接続

次に、NAS の環境構築をおこなった。NAS にも Raspberry Pi 4 model B 4GB を採用した。

3.3 スピーカー製作・動作テスト

2way スピーカーを製作、高音を補強するツイーターには「Fountek NeoX1.0」というリボンツイーターを、フルレンジスピーカーユニットには「Markaudio Alpair6v2 Metal」を採用した。「Markaudio Alpair6v2 Metal」の推奨エンクロージャー設計を基本にして、上部にツイーターを取り付け、それを覆う箱を取り付ける設計とした。

ツイーターのカップリングコンデンサに PARC Audio のセラミックコンデンサー0.47μF を採用、低音をカットして自然なクロスオーバーになるようにした。図 4 は仮接続の様子である。

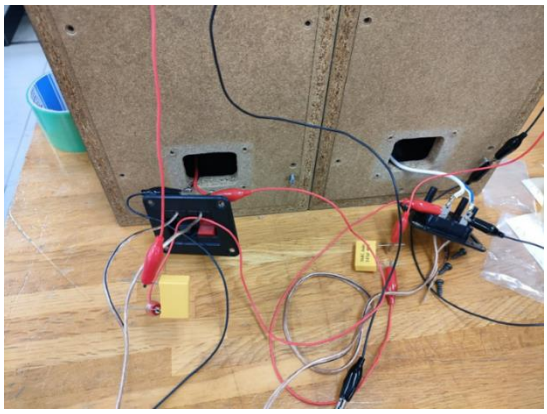

図 4. 仮接続で音を確認している様子

3.4 アンプ製作・動作テスト

アンプの製作に関しては、自作キットを組み込んだ。

3.6 完成

自分の創意工夫がつまった作品に仕上げた時、その製品やシステムに固有の名称をつけたくなるのは人の常であろう。今般の作品もそれに漏れず、「ネットワークオーディオシステム INAS: Inagaki Network Audio System」と名付けることにした。

図 5. 完成した提案システム

課題

当初の目標であった VPN 接続については、未だ成功していない。より時間をかけて VPN 接続が出来るように検討を継続していく予定である。

参考文献

Ulrich, K.T. (1995). The Role of Product Architecture in the Manufacturing Firm, Research Policy 24 419-440.

乾里穂, 森田健一 (2021). 大学生の趣味がストレスに及ぼす影響, 定塚山大学心理科学論集, 第 4 号, pp.59-64.

コロナ禍における「ものづくり」の力　～人と人をつなぐカプセル用玩具の製作～

日本工学院八王子専門学校
テクノロジーカレッジ　機械設計科　本田　ゆりか

1. はじめに

2020 年 4 月に日本工学院八王子専門学校へ入学直後から新型コロナウィルスの感染が拡大し，全国的に活動制限がされた状況下においても，オンラインと対面のハイブリッド授業や，感染対策を行った実習など多くのものづくりの機会が与えられ，私は無事に希望する職種へ内定頂き卒業することができた．日本工学院八王子専門学校では授業や実習の他に，有志による自由な作品製作を行う特長的なものづくり学習が行われており，私は学生間の交流をはかる事が出来るような作品製作を目指し，2020 年度にクラスメイトとの交流を目的とした『カプセルトイ販売機』を設計・製作し，2021 年度は後輩に向けた『クレーンマシン』の自主製作を行う事を計画した．

自身の作品の製作過程及び成果を記し，活動を通じて得た学びを広く共有する事を目的として本稿にて報告する．

2. 製作の背景

幼少期より工作が好きで，作品を通じ両親が「笑顔」になる事に達成感と，ものづくりが持つ魅力を感じていた．専門学校では機械知識や設計ツールの習得など技術的な学びを進める一方，ものづくりを通じて社会に貢献する「プロ」としての自覚を強く持つようになり，ものづくりの対象が「自分の為」だけではなく「世の中の為」へと広がった．

改めて，自身のものづくりの原点である「人を笑顔にするものづくり」を通じて，コロナ禍においてもクラスメイトや先生方の笑顔を生み出し，心が繋がる製作を実践し，その過程において自身の機械技術の習得と人間性の向上を図ることを目的としてカプセルトイ販売機とクレーンマシンの製作を行った．

3. カプセルトイ販売機の計画策定

販売機の製作では，まずQCDの目標を設定した．

Q:品質の視点

広い世代に親しみやすい「昭和レトロ」な販売機をモチーフとして，実際に使用する事ができ，使用時におけるユーザの怪我や破損が無いこと．

C:コストの視点

学校の機材（3D プリンタ，レーザ加工機）を用いて，市販されている販売機のコスト以下の金額で製作を実現すること．

D:納期の視点

6か月で製作を完了し，クラスメイトとの交流の機会を生み出し，且つ製作の成果を自身の就職活動での自己アピールへと繋げること．

4. カプセルトイ販売機の設計

QCD の達成にあたり，改善設計を含めた設計期間は3か月と設定し，市販のカプセルトイ販売機を調査し具体的な構造設計を行った．設計には授業で使用する3次元 CAD（Autodesk Fusion360）を用いて，放課後や授業の空き時間，自宅にて作業を行った．設計した3次元 CAD のモデルを図1に示す．

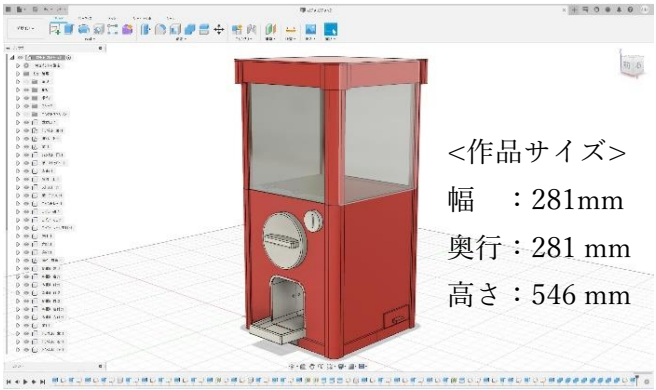

図1．Fusion360 で設計した初期の3次元モデル

カプセルトイ販売機の設計において，コインによるノブの回転機構は特に重要な機械要素である．段ボール玩具や無通電の機械的な販売機の構造を参考として，コインを入れる事でカプセル排出を即するノブの回転機構を検討した．その構造の詳細を図２に示す．

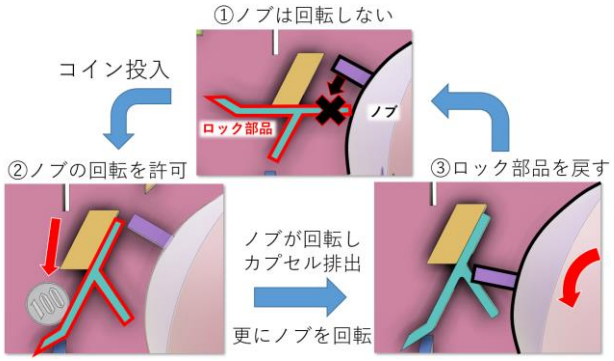

図2. コインによる回転機構部

５．カプセルトイ販売機のデザインレビュー

カプセルトイ販売機の設計が一旦完了した後，部品製作開始前にデザインレビューを先生と実施した．その際に 43 点の課題を抽出し，一つずつ改善設計を行った．最終的な設計変更箇所は全部で 62 か所に及んだが，特に大きな課題は２点あり，以下にその概要と解決策を示す．

（１）生産性

学科で保有する加工機を前提に設計を行った際，３Ｄプリンタで出力可能な「部品サイズ」の制約と，材料の熱収縮による部品の変形が発生した．そのため，一体であった部品を分割してサイズを小型化する事で熱収縮を抑え，整形可能なサイズに変更した．一例を図３および図４に示す．

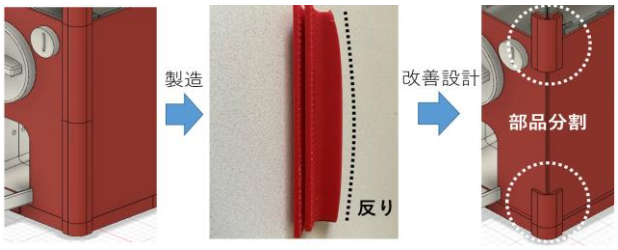

図3. 部品の分割による3Dプリントの改善①

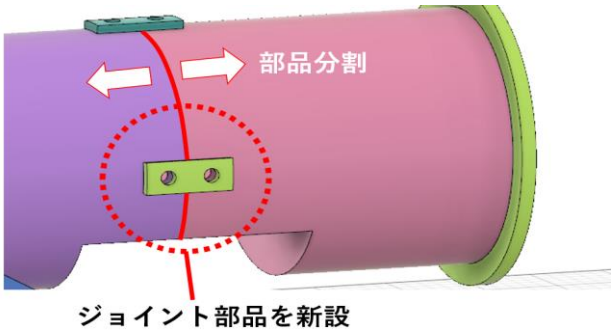

図4. 部品の分割による3Dプリントの改善②

（２）組み立て時の作業性

構造は成立をしているが実際の組み立て時にスペースが少なく，汎用工具が使用できない部位が発生し，改めて作業スペースの見直しが必要となった．図５で示す様に，実際の工具を３次元モデルでアッセンブリ内に配置し，モデル上で作業に必要なスペースの確認を出来るようにした．

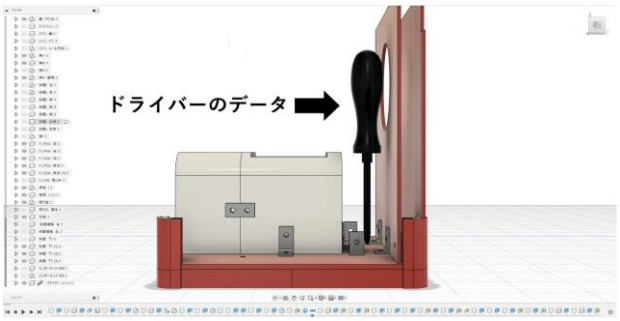

図5. モデル上の工具スペースの確認

６．カプセルトイ販売機の部品加工と組み立て

カプセルトイ販売機および主な新規部品は，３Ｄプリンタとレーザ加工機で部品加工を行った．主材料は ABS とアクリル板を使用し，材料費・製作期間ともに目標を達成した．完成した３次元モデルとその作品を図６に示す．

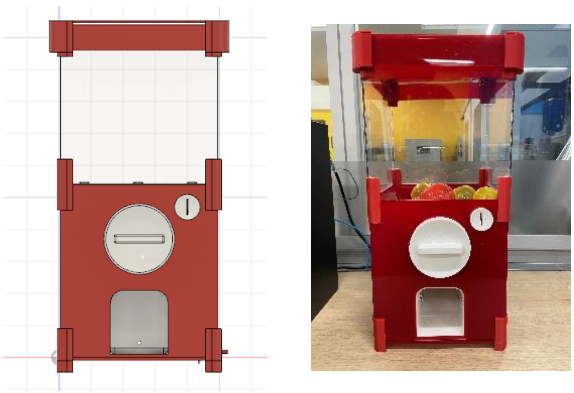

図6. 3次元モデル(左側)と完成作品(右側)

７．カプセルトイ販売機製作による学びと成果

カプセルトイ販売機の製作活動を通じた学びを以下に纏める．

・予め目標を設定してＰＤＣＡを回す事で，製品の完成度が向上し，且つ自己成長へと繋がった．

・ものづくりは一人では出来ない．設計者は構造の成立性だけではなく，次工程の部品製作や組立易さを尊重することの大切さを理解することができた．

・多くの設計変更や現物での調整は，職場では大きロスを生むが，この経験を自身のノウハウとして蓄積ることができた．

８．クレーンマシンの設計と背景

2020 年度（１年次）でのカプセルトイ販売機の製作はクラスメイトの交流を目的としていたが，２年次は後輩に「ものづくりの楽しさ」を伝えることを新たに目標と設定し，ものづくりの楽しさを一緒に体験できる課題として，クレーンマシンの設計・製作を行うこととした．

クレーンマシンの設計は可動構造が多く機構部が複雑になるため，入学直後の１年生を主体とした設計・製作においてはすべて自作することは難しい．ここで私は田宮模型の楽しい工作シリーズの各種ギアボックスや４チャンネルリモコンボックスなどの汎用コンポーネントを組み合わせることにより，設計に必要な知識のハードルを下げ，かつ所定の機能を達成することができるようにした．横方向の送り機構にはロープウェイ工作キット，前後方向の送り機構はキャタピラの工作キットを使うことで機能を実現した．挟む機構の部分は，汎用のタミヤキットの基本パーツを使用することで，ネジや穴などのスケールを同じとすることができた．その一例を図７に示す．

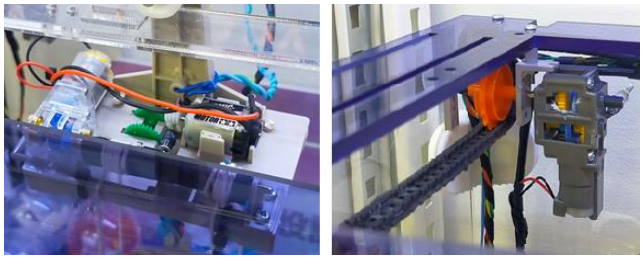

図7. クレーンの主なコンポーネント

設計では，Fusion 360 のクラウド機能を活用して，複数人で手分けして部品をモデリングし，モデル上にレイアウトしながら設計を行うことで，コロナ禍による時間制限の回避と時間短縮を図ることができた．その３次元ＣＡＤのモデルと完成作品を図８に示す．

図8. 3次元モデル(左側)と完成作品(右側)

(作品サイズ 幅:450 ㎜ 奥行:300 ㎜ 高さ:630 ㎜)

９．クレーンマシンの部品製作と組立て

田宮模型のキット以外の筐体部品の多くは，トイカプセル販売機と同様に 3D プリンタによる成形と，アクリル板をレーザ加工機によってカットして製作した．部品の製作は，これまでの経験によりスムーズに進んだが，それ以上に組立てと調整に時間を要した．

ヒモ，バネといった部品は，モデル上で具体的な機能設計をすることが難しく，一旦組み立てて動作確認をした後に，途中に知恵を出し合いながら材料を変更したり，取り付方法を変更しながら調整を行った．景品を掴む爪の部分においては，ゴムの爪の強さを調整するためのゴムの強さをギアボックスのトルクに合わせて調整しました．また，爪のユニットを巻き上げるための巻き取り機構においては，シャフトの直径が小さすぎて巻取に時間がかかり過ぎてしまうため，厚いテープを貼って直径を増やして調整した．また，爪全体を上下させる際に，爪を巻き上げるモータの配線にガイドが無かったため，左右前後の移動時に他のユニットに絡まるという問題も発生し，手元にある硬い電線でコイル状のガイドを作成することで解決した．その様子を図９に示す．

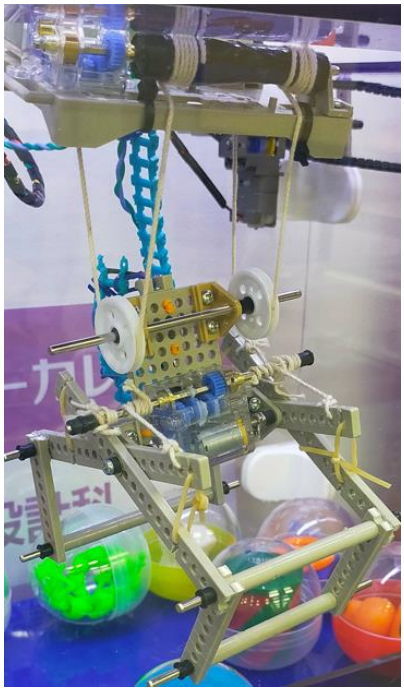

図9. ゴム・ヒモの組み立て調整部分

多くの調整を経て，コントローラを操作して景品をクレーンで吊り上げ，回収用の穴に落とすという一連の動作を達成できるようになった．しかし，繰返し使用すると巻取りのヒモが偏って外れてしまうなど，繰り返し精度に課題があることがわかった．また，巻取用のヒモのシャフトに固定する際に，テープや接着剤で固定したため，調整がしにくいという課題も見つかり，調整機構を取り入れることが今後の課題である．

10. クレーンマシン製作による学びと成果

クレーンマシンの製作による学びと成果を以下にまとめる．

- 複雑な可動機構も要素ごとに分解することで，田宮模型の工作キットが活用できるシンプルでわかりやすい構造へと落し込む事ができた．
- ３次元ＣＡＤ上で詳細設計が難しい部分は，部分的にモックアップを作成し検証する．
- 動作不良の部位を特定し根本原因を対策した．
- 組立て後の調整を考慮した設計が重要である．
- 長期間トラブルなく動作することができる実際の製品の品質の高さを実感した．

11. 最後に

カプセルの中に入れた景品は，クラスメイト全員に協力してもらい製作した．更にカプセルトイ販売機の製作活動に対し Autodesk 社の取材（＊1）受け，取組みを広く共有すると同時に，貴重な学びの場を得る事が出来た．

誰もが子どものころに親しんだカプセルトイ販売機・クレーンマシンをクラスメイトと一緒に製作することで，コロナ禍においてもクラスメイトや後輩との絆を「ものづくり」によって深めることができた．これから機械設計を学ぼうと考えている方にも，人を思いやり，人と人を笑顔でつなげる様なワクワクする機械作品の製作を実践して欲しいと考える．

自信もこの経験を活かし，人を笑顔にするものづくりを通じて「若きつくりびと」として社会に貢献していく決意である．

12. 謝辞

本報告を行う上で，協力してくれたクラスメイトをはじめ，学校関係者の皆様に感謝申し上げます．

著者　Yurika Honda
著者連絡先 g020d1926@g.neec.ac.jp
*指導教員（奥住智也） okuzumitmy@stf.neec.ac.jp
*指導教員（吉川求） kikkawamtm@stf.neec.ac.jp

（＊1）Autodesk 社　取材の動画
Youtube　QR コード

Ansible を用いたサーバ環境の自動構築

中山千拓† 二橋翼† 朴鋒洙†

概要：VMware ESXi (以下，ESXi)を使い RedHat 系 Linux である CentOS，Debian 系 Linux である Ubuntu の 2 種類の Linux ディストリビューションの仮想マシンを立て，Ansible を用いてその各サーバにおける環境構築の自動化ができるかを検証する．

Automated construction of server environments using Ansible

CHIHIRO NAKAYAMA† TSUBASA NIHASHI† PARK BONGSOO†

Abstract: We will test whether Ansible can be used to automate the construction of environments on VMware ESXi (ESXi) virtual machines running two different Linux distributions, CentOS (Red Hat Linux) and Ubuntu (Debian Linux).

1. はじめに

1.1 背景

近年 Ansible をはじめとして，複数のサーバ構築やクラウド環境の設定変更を統合管理できる構成管理ツールが注目を浴びている．その理由の一つにはクラウドリソースのサービス化やインフラリソース管理の API 化に伴い，ビジネス変化に合わせたシステム構築が容易になっていることが挙げられる．こうした開発環境の変化によって，迅速かつ低コストでシステム構築を行えることが，企業のシステム投資の指標となっている．

つまり，構成管理ツールは迅速なシステム構築や柔軟な変更をもとに，コストの削減や開発スピードの向上を提供し，ビジネスにおける IT 戦略をサポートする役割を担っている．

1.2 概要/目的

RedHat 社が開発している構成管理ツールである Ansible を使用して，ESXi に作成した 2 種類の Linux ディストリビューションにおけるサーバ環境設定の自動化を検証する．この検証を通して，大規模なシステム環境の構築をする際の時間短縮や人的脅威の削減を目指す．

2. Ansible について

Ansible とは，RedHat 社が開発するオープンソースの Python 製構成管理ツールである．

2.1 特徴

Ansible の特徴は 4 つある．

(1) シンプル

Ansible における一連の処理内容を記述する Playbook は，YAML 形式で定義する．YAML 形式は可読性が高く初心者でも扱いやすいうえ，学習が容易である．

(2) パワフル

多数のサーバ，クラウド，ネットワーク機器に対応しており，様々な機器を操作するためのモジュールを使用することができる．

(3) エージェントレス

Ansible は SSH を利用したエージェントレスのため，構築対象となる機器側に専用ソフトのインストールは不要である．SSH を利用して安全にリモート操作で作業できるため，人員を常駐させる必要が無くなる．

(4) 冪等性(べきとうせい)

冪等性とは「同じ操作を何度繰り返しても，同じ結果が得られる性質」のことである．

例えば，ソフトウェアのインストール作業の自動化する事を想定する．その際，独自スクリプトで冪等性を考慮しない場合，既存の設定を上書きして初期状態に戻ってしまう他に，思わぬサービスに影響を及ぼす可能性がある．

このようなトラブルを防ぐためには，本来インストールされるはずのディレクトリのチェックや，プロセスが起動していないかの確認作業を事前に行い，既にインストールされている場合は何もしない判断が必要である．複数回実行しても，冪等性が担保されているツールを利用すれば，何度実行しても同じ結果が得られる．

このように Ansible は作業を実行する前に条件を判断し，必要が無ければ作業をスキップする機能が備わっている．

† 日本電子専門学校
Japan Electronics College

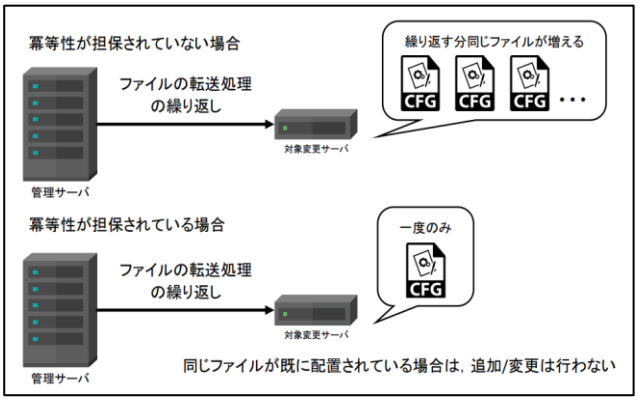

図 1　冪等性

2.2　システム構成

システム構成の詳細について説明する．

2.2.1 システム構成概要

管理者は，管理 PC から Ansible-Playbook コマンドを実行し，SSH 経由でターゲットである各ノードに対し環境設定を行う．ターゲットとなるノード情報は，Inventory ファイルに記載されている．

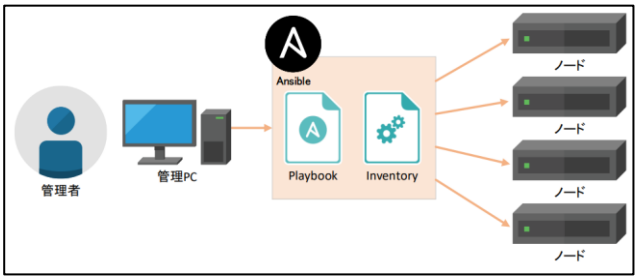

図 2　システム構成

2.2.2 ネットワーク環境

本研究のネットワーク環境は，VMware ESXi 内に実装されており，各仮想マシンの仮想 NIC が仮想スイッチに接続されている．これにより，仮想マシン間やインターネットに接続されている．

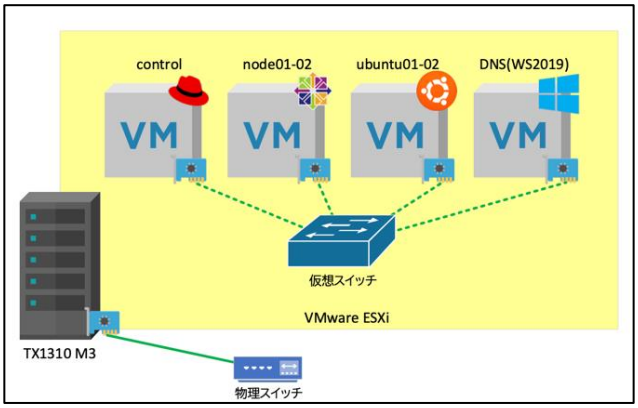

図 3　トポロジ図

表 1　アドレステーブル

ホスト名	IP アドレス	プレフィックス	デフォルトゲートウェイ
WS2019	192.168.1.102	/24	192.168.1.254
control	192.168.1.110	/24	192.168.1.254
node01	192.168.1.111	/24	192.168.1.254
node02	192.168.1.112	/24	192.168.1.254
ubuntu01	192.168.1.113	/24	192.168.1.254
ubuntu02	192.168.1.114	/24	192.168.1.254

2.2.3 ハードウェア

使用したサーバ等のスペック情報を記載する．サーバは，中山の自宅にある Fujitsu Primergy TX1310 M3 を使用した．

表 2　管理 PC のスペック情報

CPU	Intel E3-1225v6 CPU @2.90GHz
Memory	24GB
SSD/HDD	SSD 250GB + HHD 1TB
Ethernet	Intel Ethernet Server Adapter I350-T4V2

表 3　ノードの OS 情報

control	Red Hat Enterprose Linux release 8.5 (Ootpa)
node01-02	CentOS Linux release 8.4.2105
ubuntu01-02	Ubuntu Servsr 20.04.3 LTS
DNS(WS2019)	WindowsServer2019

2.2.4 ソフトウェア

使用したソフトウェアの種類およびバージョン情報を記載する．

表 4　ソフトウェアバージョン

Ansible	2.9.27
Python	3.6.8

Ansible 実行環境として，Python2.7 もしくは Python3 がインストールされている必要がある．

3.　実装

実際にシステムを構築した過程と，構築完了段階のシステムについて説明する．

3.1　事前準備（SSH 公開鍵認証設定）

Ansible サーバから SSH 経由でターゲットとなるノードの管理を行うので，サーバ側で秘密鍵および公開鍵を作成し，公開鍵を各ノードに登録する必要がある．

3.2　Ansible インストール

Ansible は，OS パッケージマネージャー（yum や apt）または pip を使用して，安定したバージョンをインストールすることが推奨されている．

3.3 Ansible の環境設定

Ansible の環境ファイル名は，ansible.cfg である．ここでは，通信方法やログ出力等，Ansible の実行を記述する．設定ファイルは，図 4 の通りである．

```
/home/ansible/ansible.cfg
[defaults] ←グループ名
forks = 15
log_path = $HOME/.ansible/ansible.log
host_key_checking = False
gathering = smart
gather_subnet = all
```

図 4　ansible.cfg の設定

図 4 で使用したパラメータを以下で説明する．

- **forks**
  並列実行する数を指定する
- **log_path**
  実行結果のログを保存するパスを指定する
- **host_key_checking**
  known_hosts の key をチェックするかを指定する
- **gathering**
  対象ホストの情報を集めるか否かを指定する
- **gather_subnet**
  対象ホストの収集する情報を指定する

3.4 Inventory 作成

Inventory ファイルにはターゲット情報を記述する．Ansible-Playbook を実行する際の，ターゲット指定に必要となる．

最も一般的には図 3 のように INI 形式で作成する．

```
/home/ansible/var_inventory.ini
[redhat] ←グループ名
node01.ansible.esxi71.local ansible_host=192.168.1.111
  ↑ホスト名，IP アドレス
node02.ansible.esxi71.local ansible_host=192.168.1.112

[ubuntu]
ubuntu01.ansible.esxi71.local ansible_host=192.168.1.113
ubuntu02.ansible.esxi71.local ansible_host=192.168.1.114

[control]
#control.ansible.esxi71.local ansible_host=192.168.1.110

[node:children] ←グループ変数
redhat
ubuntu
```

図 5　Inventory ファイル内容

3.5 Playbook のディレクトリ構造

ディレクトリ構造の主要ファイルを説明する．

- **group_vars**
  ステージ毎に異なるグループ変数を定義する．変数を設定する事で，コードや設定変更の簡略化に繋がる．
- **linux_initialsetup.yml**
  Linux 初期設定をするためのメイン実行ファイル．明示した全ての処理をタスク管理している．また，ファイルの編集により，処理項目の変更も可能である．
- **roles**
  Ansible の推奨ディレクトリ構成の（細分化するための）「ロール」ディレクトリである．ロールに従った構成にすることで，変数を自動でファイルから読み込む，転送するファイルの相対パス指定が楽になるなど利点が生まれる．
- **templates**
  モジュールの 1 つである．これを使用することで，Ansible マシンのローカルに存在するテンプレートファイルを管理対象機器側へ転送することができる．
- **Jinja2**
  テンプレートエンジンライブラリである．httpd.conf などの設定ファイルを，変数とテンプレートから生成する用途に使用する．
- **main.yml**
  ロールを実行する際に最初に読み込まれる必須ファイルである（HTML でいう index.html のようなもの）．main.yml 以外のファイルは，main.yml から include モジュールを使用し，Playbook を分割した時と同様に，タスクのみを直接記述し使用する事が多い．

3.6 Playbook 作成

Playbook には一連の処理内容を記述する．本研究では，サーバの初期設定構築を想定し Playbook を作成した．

Playbook は 1 つのファイルで完結させ作成する事も可能だが，細分化することにより拡張性や柔軟性という利点が生まれるため，本研究では，role の仕組みに基づきディレクトリ構成をする．

作成した処理内容は全 8 項目である．項目毎の設定内容と，ディレクトリ構造を併せて明示する．

①　Firewall 設定

ファイアウォールとは，外部ネットワークからの攻撃や不正アクセスを防止する為に必要となる，基本的なセキュリティである．自動起動有効化にする設定を記述した．

```
/home/ansible/roles/common
firewalld ←firewalld 用 Dir
  └ tasks ←Playbook 用 Dir
      └ main.yml
```

図 6　firewalld ディレクトリ構造と説明

② **SELinux 設定**

SELinux とは，強制アクセス制御機能を実現するセキュリティの仕組みである．本来有効にする事が望ましいが，SELinux のルール外の動作を行うとブロックされてしまう．

本研究では検証が目的の為，無効化にする設定を行った．

```
/home/ansible/roles/common
selinux ←SELinux 用 Dir
 └ tasks ←Playbook 用 Dir
       └ main.yml ←主動作
```

図 7　selinux ディレクトリ構造と説明

③ **パッケージ管理**

初期段階で必要と思われる開発ツールパッケージのインストール，全パッケージを最新版にアップデートする設定である．

```
/home/ansible/roles/common
 packages ←packages 用 Dir
  ├ tasks ←Playbook 用 Dir
  │   ├ apt_update.yml ←Debian 系の動作
  │   ├ dnf_update.yml ←RedHat 系の動作
  │   └ main.yml ←上記 2 項目をまとめた動作
  └ vars ←変数用 Dir
      ├ Debian.yml ←Debian 用変数ファイル
      └ RedHat.yml ←RedHat 用変数ファイル
```

図 8　packages ディレクトリ構造と説明

④ **ロケール設定**

ロケール設定とは，特定の地域や言語で利用できるようにする各種設定のことである．ロケール，タイムゾーン，キーボード配列の設定は以下の構造となる．

```
/home/ansible/roles/common
locale ←locale 用 Dir
 ├ defaults ←デフォルト値用 Dir
 │  └ main.yml ←デフォルト値 locale 設定動作
 └ tasks ←Playbook 用 Dir
       ├ timezone.yml
       ├ locale.yml
       ├ keymap.yml
       └ main.yml ←上記 3 項目をまとめた動作
```

図 9　locale ディレクトリ構造と説明

⑤ **ユーザ管理**

ユーザ作成，ユーザが sudo コマンドを使用する際に必要となる wheel にグループを登録する設定を行った．

```
/home/ansible/roles/common
users ←users 用 Dir
 ├ tasks ←Playbook 用 Dir
 │   └ main.yml
 ├ templates ←テンプレート格納用 Dir
 │   ├ admin_sudoers.j2 ←RedHat 用ファイル
 │   └ member_sudoers.j2 ←Debian 用ファイル
 └ vars ←users 変数設定用 Dir
     ├ debian.yml ←Debian 用変数ファイル
     └ redhat.yml ←RedHat 用変数ファイル
```

図 10　users ディレクトリ構造と説明

⑥ **リゾルバ管理**

リゾルバとは，名前解決のする仕組みのことである．nameserver の指定，DNS 無効化の設定を行った．

```
/home/ansible/roles/common
resolver ←resolver 用 Dir
 ├ files ←filemodule 用格納 Dir
 │   ├ nsswitch.conf ←4 の設定ファイル
 │   └ resolv.conf ←3 の設定ファイル
 ├ tasks ←Playbook 用 Dir
 │   └ main.yml ←1~4 の動作
 └ templates ←テンプレート格納用 Dir
     └ hosts.j2 ←2 のテンプレートファイル
```

図 11　resolver ディレクトリ構造と説明

⑦ **NTP 設定**

NTP とは，PC の内部時刻を正しく設定する為に，ネットワーク上で時刻表同期をするプロトコルである．同期 NTP サーバプールの指定，サービスの自動起動有効化の設定とした．

```
/home/ansible/roles/common
chronyd ←chronyd 用 Dir
 ├ files ←filemodule 用格納 Dir
 │   ├ debian.conf ←Debian 用 1 設定ファイル
 │   └ redhat.conf ←RedHat 用 2 設定ファイル
 └  tasks ←Playbook 用 Dir
      └ main.yml ←1，2 の動作
```

図 12　chronyd ディレクトリ構造と説明

⑧ **http 設定**

Web サーバを使う上で必要となる Apache の設定ファイルである．デフォルトページの作成，Basic 認証の追加，ポート解放，サービスの自動起動有効化，ファイル権限，所有グループの設定とした．

```
/home/ansible/roles/common
apache ←apache 用 Dir
  ├ files ←filemodule 用格納 Dir
  │   └ basic.conf ←設定ファイル
  ├ tasks ←Playbook 用 Dir
  │   ├ apache2.yml ←Debian 用の動作
  │   ├ httpd.yml ←RedHat 用の動作
  │   └ main.yml ←上記 2 項目をまとめた動作
  └ templates ←テンプレート格納用 Dir
      └ index.j2 ←設定ファイル
```

図 13　httpd ディレクトリ構造と説明

4.　評価

4.1　Playbook の実行

ansible-playbook -i［イベントリファイル］［Playbook ファイル］コマンドで実行する事ができる．Ansible の特徴である冪等性の検証のため，同じ Playbook の実行を 2 回行う．

1 回目の実行完了時は，change の数（設定変更完了）がカウントされているのが確認できる．

```
ok: [node01.ansible.esxi71.local]
 ------------
< PLAY RECAP >
 ------------
  \ ^__^
   \(,_u u/

node01.ansible.esxi71.local : ok=49   change
d=19   unreachable=0    failed=0    skipped=
2    rescued=0    ignored=0
```

図 14　1 回目実行完了結果

2 回目の実行完了時は，change の項目が 0 になっている事が確認できた．これにより，1 回目と同じ Playbook の実行を行なった事により Ansible の特徴である冪等性が正しく機能し，何も変更しなかった事が確認できる．

```
ok: [node01.ansible.esxi71.local]
 ------------
< PLAY RECAP >
 ------------
  \ ^__^
   \(,_u u/

node01.ansible.esxi71.local : ok=48   change
d=0    unreachable=0    failed=0    skipped=
3    rescued=0    ignored=0
```

図 15　2 回目実行完了結果

4.2　確認

CentOS（RedHat 系）と Ubuntu（Debian 系）への設定が正しく反映されたか確認を行う．

4.2.1 Firewalld の状態確認

下図の通り両者とも Firewalld が有効になっている事が確認できた．

```
[ansible@node01 ~]$ sudo firewall-cmd --list
-all
                public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens192
  sources:
  services: cockpit dhcpv6-client http ssh
  ports: 9100/tcp
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

図 16　Firewalld 状態（CentOS）

```
ansible@ubuntu01:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
9100/tcp                   ALLOW       Anywh
ere
Apache                     ALLOW       Anywh
ere
9100/tcp (v6)              ALLOW       Anywh
ere (v6)
Apache (v6)                ALLOW       Anywh
ere (v6)
```

図 17　Firewalld 状態（Ubuntu）

4.2.2 インストールされたパッケージ確認

下図の通り両者ともパッケージのインストールが正常に行われた事が確認できた．

```
[ansible@node01 ~]$ rpm -qa  | grep -e epel-
n36 -e httpd-2.4.37
chrony-4.1-1.el8.x86_64
httpd-2.4.37-43.module_el8.5.0+1022+b541f3b
epel-release-8-11.el8.noarch
python36-3.6.8-38.module_el8.5.0+895+a459ec
tar-1.30-5.el8.x86_64
bash-completion-2.7-5.el8.noarch
git-2.27.0-1.el8.x86_64
```

図 18　パッケージ確認（CentOS）

```
ansible@ubuntu01:~$ apt list |grep -e ^apache
letion -e ^chrony/ -e ^tree/
apache2-bin/focal-updates,focal-security,now
apache2-utils/focal-updates,focal-security,no
chrony/focal-updates,focal-security,now 3.5-6
git-all/focal-updates,focal-security 1:2.25.1
python3-pip/focal-updates,now 20.0.2-5ubuntu1
python3-pipdeptree/focal 0.13.2-1build1 amd64
python3.8/focal-updates,focal-security,now 3.
tar/focal-updates,focal-security,now 1.30+dfs
tree/focal,now 1.8.0-1 amd64 [installed]
wget/focal-updates 1.20.3-1ubuntu2 amd64 [upg
```

図 19　パッケージ確認（Ubuntu）

4.2.3 Basic 認証の確認

サーバに接続に接続を試みると，下図の通り両者とも Basic 認証が正常に動作する事が確認できた．

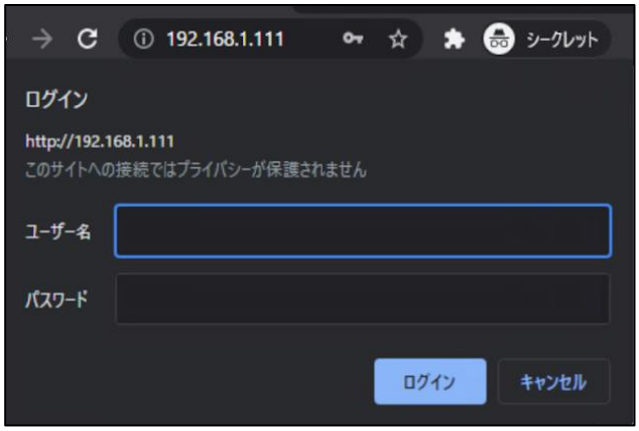

図 20　認証確認（CentOS）

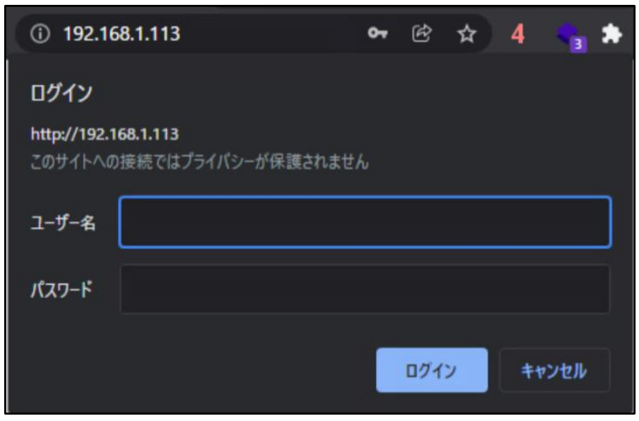

図 21　認証確認（Ubuntu）

5.　まとめ

テーマであった VMware ESXi を使い 2 種類のディストリビューションの仮想マシンを立て，Ansible を用いてその各サーバ（RedHat 系，Debian 系）環境構築の自動化を行う事ができ，全て正しく設定された事も確認する事ができた．

本成果により，Ansible を用いる事で大規模なシステム環境の構築をする際の，時間短縮や人的脅威の削減を可能にする事が実現できるという結論に至った．

唯一 Ansible の欠点として感じた部分として，Playbook を作成する際，Ansible で対応していないコマンドが存在し，止むを得ず shell モジュール，command モジュールを使う場合において，冪等性の担保ができない点であるが，工夫次第では可能である．shell モジュールを使用した例を挙げると，ロケール設定が該当するが，変数と条件判定を組み合わせた Playbook を作成する事により，shell モジュールを使用しても冪等性を担保する事ができた．Ansible は一度作成した Playbook を再利用，組み合わせて利用するなどと，柔軟性，拡張性，冪等性に秀でる強みがあるため，それを最大限活かすシンプルかつ応用の効く構成をする事が求められる．

今後の展望として，本研究で使用した自動化の実行エンジンである Ansible Engine と併せて，組織において包括的に管理運用を実施する Ansible Tower を用いる事により，組織が持つ IT 機器を全て管理し，誰でも自動構築の実行を扱う事まで可能となるので，Ansible を使い組織全体の IT 機器の管理，構築の自動化を目指したい．

6.　参考文献

[1]佐藤学，　横地晃，　塚本正隆，　畠中幸司，　北山晋吾．
Ansible 実践ガイド（第 3 版）．インプレス出版，　2021
[2]横地晃
自動化ツール　Ansible をはじめよう【Day1】，　株式会社エーピーコミュニケーションズ，　2021，P07 - P37.
[3][Ansible] service モジュールの基本的な使い方（サービスの起動・停止・自動起動の有効化など）
https://tekunabe.hatenablog.jp/entry/2019/02/24/ansible_service_intro (参照　2021-10-11)
[4]katakoda asnible
https://www.katacoda.com/irixjp (参照　2021-10-11)
[5]【Ansible】初心者が　Syntax Error while loading YAML.did not find expected で嵌った場合と対処法
https://hamutetublog.com/ansible-syntax-error/ (参照　2021-10-15)

アズテック認証システム

大植友博† 丹羽雅人† 松本朝香†

概要：本稿では，顔認証と声紋認証による認証システムと認証情報の管理，および，リソース監視を実行するシステム群を設計，構築した．現在のコロナ禍における感染症対策として，認証に際し，機器に接触することなくスムーズな速度で実装をすることを目標に構築をした．認証精度と脆弱性対策については，今後も改善が必要である．

Aztech Authentication System

TOMOHIRO OUE† MASATO NIWA† ASAKA MATSUMOTO†

Abstract: In this paper, we designed and built an authentication system using face recognition and voice print recognition, and a set of systems for managing authentication information and monitoring resources. As a countermeasure against infectious diseases caused by covid-19, which are currently prevalent in the world, we built the system with the goal of realizing smooth authentication without contacting the devices required for authentication. We believe that further improvement of the authentication and countermeasures against vulnerabilities are necessary in the future.

1. はじめに

1.1 背景

認証とは，何らかの手段を用いて対象の正当性や真正性を確かめることである．

インターネット上のセキュリティ問題が声高に叫ばれている昨今，より強固な認証が求められている．しかし，一般に普及している 2D 認証には本人判定に精度やなりすまし等の課題があり，3D 認証は導入にコストがかかるため普及していないのが現状である．このように，実際に普及するかどうかは「ユーザビリティ」と「コスト」に左右される．安全で快適なサービスを普及させるには，容易な導入と過不足のない機能性を持ったシステムを構築することが重要である．さらに，感染症の流行により非接触で日常を過ごす場面が増えている．

こうした状況を鑑み，我々は，安価かつ非接触で認証が完了する多要素認証入退室管理システムを構築することに決めた．

1.2 目的

顔認識と声紋認識を用いた多要素認証入退室管理システムを構築する．OSS を用いることで安価で普及しやすく実用的なサービスを構築し，プログラミング・セキュリティ・OSS に関する知見と技術力を向上させる．また，現在コロナ禍であることを考慮し，実用面では非接触で認証ができ，ストレスを感じない程度の速度が出ることを目標にした．

2. Aztech Authentication System

2.1 機能一覧

(1) 認証機能

顔認証と声紋認証による生体認証を行う機能である．認証したいときに，非接触で認証を行うことが可能である．

(2) 入退室監視機能

データベースサーバから入退室ログを抽出して，web ページに表示する機能である．誰が，いつ，入室または退室しているのかをリアルタイムに表示する．簡潔に現在の状況を調査したい場合に利用されることを想定している．

(3) サービス監視機能

当システムで利用しているサーバ，ネットワークのリソースを監視，追跡するための機能である．リソース監視を通してサイバー攻撃の早期検知，障害復旧時に利用されることを想定している．

2.2 システム構成

システム構成の詳細について説明する．

2.2.1 システム構成概要

クライアント PC では常に顔認証システムである YOLO と、声紋認証である Voice プログラムが動作しており，利用者が内蔵カメラに検出されると認証が始まる．また，物体検知後，閾値以上の音を検知すると声紋認識も開始され，2 つの判定結果を総合して本人だと認識する．さらに認証が通るたび、次回認証用のデータを解析用サーバに送り，次回用のモデルを再構築し，クライアントに送る仕組みにした．

† 日本電子専門学校
Japan Electronics College

2.2.2 ネットワーク環境

システム周辺のネットワークに関して説明する．今回 GCP 無料枠により IP アドレスは全て動的であるため，アドレステーブルの記載は割愛する．以下，図 1 はシステム構成図，図 2 はトポロジ図を示している．

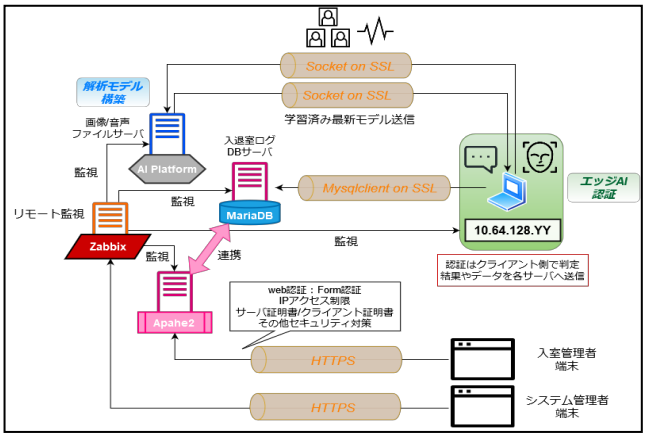

図 1　システム構成図

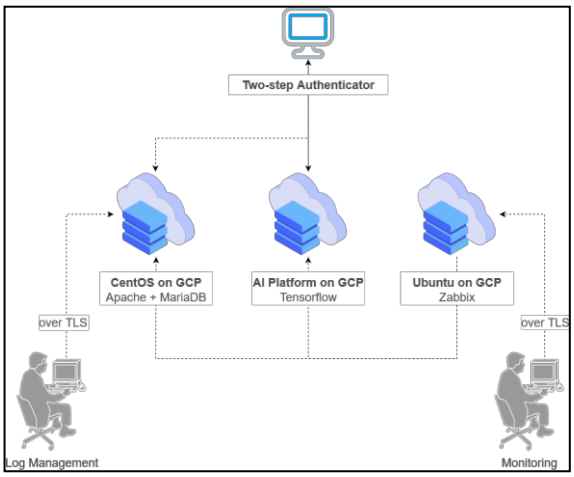

図 2　接続構成図

2.2.3 ハードウェア

下表 1-1，1-2 に使用したマシンスペックを記載する．

表 1-1　GCP インスタンス スペック

項目	スペック
マシンタイプ	E2-micro
リージョン	us-west1 (オレゴン)
ゾーン	a, b
ブートディスク	標準永続ディスク (30 GB)
イメージ	CentOS, Ubuntu20.04LTS

表 1-2　クライアント PC スペック

項目	スペック
CPU	AMD Ryzen 7 3700U with Radeon Vega Mobile Gfx　2.30 GHz
RAM	16GB
OS	Windows 10 Home

2.2.4 各サーバ使用 OS＆導入ソフトウェア

使用した GCP 上のサーバのスペック情報を記載する．

表 2　ソフトウェアバージョン

役割	OS	ソフトウェア
web サーバ兼 ログ管理 DB サーバ	CentOS7.9	Apache2.4.6, Mariadb 10.5
監視サーバ	Ubuntu 20.04 LTS	Zabbix 5.4
クライアント PC	Windows 10	——

表 3　必要ライブラリ (Client)

ライブラリ	バージョン
matplotlib	3.2.2
numpy	1.18.5
opencv-python	4.1.2
Pillow	7.1.2
PyYAML	5.3.1
requests	2.23.0
scipy	1.4.1
torch	1.7.0
torchvision	0.8.1
tqdm	4.41.0
tensorboard	2.4.1
pandas	1.1.4
seaborn	0.11.0
pyaudio	0.2.11
librosa	0.8.1
keras	2.7.0
keras-Preprocessing	1.1.2
matplotlib	3.5.0
Pillow	8.4.0
pyAudio	0.2.11
pycparser	2.21
pydub	0.25.1
tensorflow	2.7.0

3.　認証スコープの実装

実際にシステムを構築した過程と，構築完了段階のシステムについて，説明する．

3.1　リアルタイム顔認証

顔認証には YOLOv5 を使用している．YOLO にはサンプルコードが付属しており，今回は要件を満たすよう編集し使用した．5 秒以上同一人物が精度 70%（背景白の場合）で映ることが基準となっている．この状態で声紋認識の結果が一致すれば認証成功となり，ログ監視 DB へのリクエスト，学習用データが各サーバへ送信される．

下図 3 は，認証実行中の画面をキャプチャしたものである．顔認証プログラムのロード後，声紋認証プログラムもロードされ，2 段階認証が並行して行なわれる．顔認証プログラムは，5 秒間継続して本人だと判定すると赤枠のように本人確認が完了したことを示すメッセージを出力する．

図 3　実装中画面-1

なお，認証システムに非登録の人間に対しては下図 4 のように認証されないため白黒画面に “INVALID” として出力するようにし，ユーザにも分かりやすい仕様にしている．

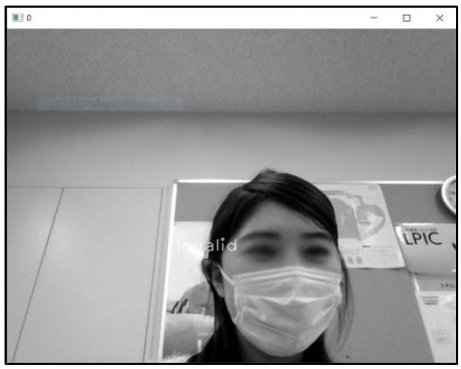

図 4　INVALID 画面

3.2　リアルタイム声紋認証

声紋認証では録音したデータから音響特徴量を抽出したものをネットワーク通している．サンプリングレートを 44100Hz で取得しているため学習時，予測時に処理時間が長くなってしまうことを考え入力層はスペクトログラムに変換したものを通している．下図 5 が示すように，入力層はメルスペクトログラム，mfcc，stft の 3 つを結合させ一つの大きな画像として扱い，ニューラルネットワークに通している．

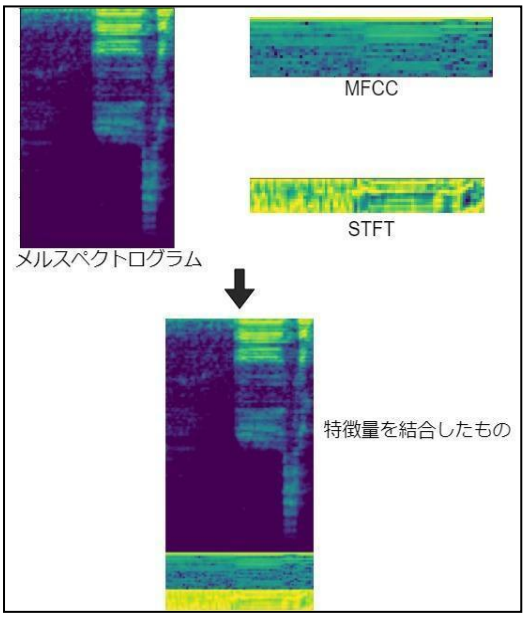

図 5　特徴量結合

録音したデータに対し，この一連の処理を行っただけでは入力層のサイズがデータの長さによって左右されてしまい，引き延ばし，0 パディングといったリサイズ処理が必要になってしまう．しかしリサイズを行ってしまうと横軸を時間としているため精度が落ちてしまうと考え，データを 1 秒で分割し，それぞれに対して処理を行った．結果として出力される値は複数の値になってしまうが，これは最頻値を取ることで一つのデータに一つの結果を返せるようにした．しかし，分割箇所によっては無音部分が多く含まれたものが処理を受けてしまうこともあり，精度が悪くなってしまった．この問題に対して，録音時に出来るだけ無音空間を排除するように変更を加えた．

前提条件として非接触での録音があったため，声を基準に録音するかを判断している．規定値を越える音が観測された時点から 1 秒間録音している．話している間は規定値以上の音が継続しているため，録音開始位置はその都度更新され，1 秒以上の録音が出来るようになっている．今回は規定値に dBFS の-3dB にしている．dBFS は PC 内で扱える最大数を 0dB としたもので，マイクの設定によって変わってしまうため，これがほかの環境でも使える値とは限らない．

下図 6 は録音停止処理について示したものである．沈黙が 1 秒以上継続した場合は一時停止され，そこから 5 秒間待機し，5 秒以内に音が発生した場合に発生源は同じものと見なし，一時停止されていたデータに追記されていき，沈黙が 5 秒以上継続された場合に録音終了となる．

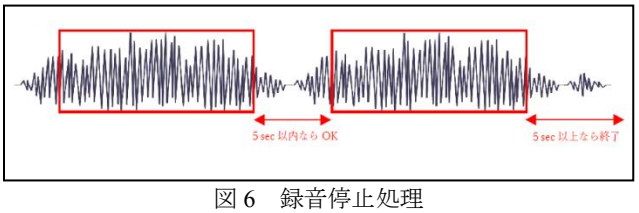

図 6　録音停止処理

しかし，これらの変更を行った結果，精度は少ししか上がらなかった．これは同一人物が話したとしても，得られるデータが大きく変わる点である．この点を考慮し声紋認証に使われる言葉を決定し，録音サイズを 3 秒と決める変更を行ったところ精度は改善された．

下図 7 は，認証実行中の画面をキャプチャしたものである．声紋認証プログラムは，録音完了後，下図 7 のように解析経過および解析結果をコンソール上に出力する．

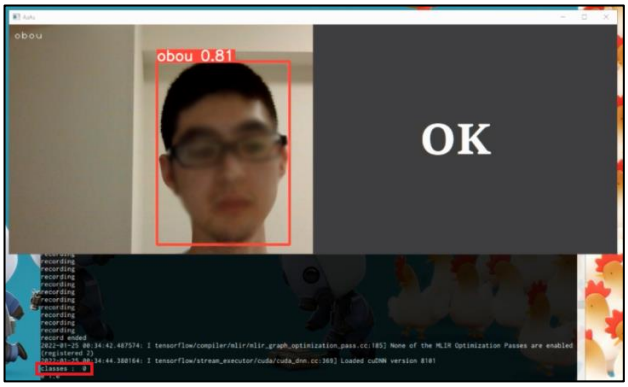

図 7　実装中画面-2

3.3　学習用データ転送

次回用モデルを構築するにあたってクライアントからサーバへデータを転送する必要がある．今回はその転送に暗号化した socket を使用している．socket には Dataset クラスをインスタンス化し，オブジェクトの pickle ライブラリを使用し送信可能なフォーマットに変換し送信している．Dataset クラスには yolo のアノテーションデータ，yolo の web カメラの画像，録音した音声データ，音声データのラベルを値として持っている．

サーバが受け取ると元のフォーマットに直し，それぞれ指定したディレクトリに出力している．yolo は学習時に yaml 形式のアノテーションデータが必要なため別ディレクトリに保存し，音声データはファイル名に「0_202101121515．wav」のように正解ラベル，取得した日時を入れている．

3.4　設定ファイル

各プログラムは「config.ini」から設定を読み込んでいるため，変更点が出たときに変更しやすくなっている．ファイルの形式は下図 8 のようになっている．

```
[セクション名]
パラメータ＝値
```

図 8　設定ファイルの形式

設定ファイルは大きく分け 4 つのセクションに分けられている．下表 4 は各セクションとその役割の表である．

表 4　各セクションの概要

セクション	概要
voice	声紋認証の録音開始閾値，学習済みモデル等
yolo	yolo に渡すオプション等
sql	IP アドレス，ssh に使用する key ファイル等
dataset_server	モデル構築用サーバの IP アドレス等

3.5　バックエンドの構築

以下は FIrewall の開放ポートである．なお，GCP では SSH と HTTP, HTTPS はデフォルトでポートが解放されているため特に設定の必要はない．

mariadb-up：3306

mariadb-down：3306

3.5.1 Apache

使用バージョン：Apache2.4.6

認証方式としては Form 認証を採用した．user ID と session ID を紐づけて管理することで response cookie の有効期間中は，クライアント側はユーザ ID とパスワードを入力せずにログイン可能となるため，今後サービス展開が充実した際に，シングルサインオンの実現が容易になる．今回，Form 認証を行なうにあたってモジュール（mod_session）の追加インストールと「/etc/httpd/conf.modules.d/01-session.conf」で有効化した．追加で有効化する必要があったのは下図 9 において赤枠で囲われた部分である．

```
LoadModule session_module modules/mod_session.so
LoadModule session_cookie_module modules/mod_session_cookie.so
LoadModule session_dbd_module modules/mod_session_dbd.so
LoadModule auth_form_module modules/mod_auth_form.so

LoadModule request_module modules/mod_request.so
LoadModule session_crypto_module modules/mod_session_crypto.so
```

図 9　Apache 追加モジュールの有効化

さらに，ディレクトリ毎にアクセス制限をかけられる設定とクライアントに SSL 強制接続を要求するための設定を下図 10 のように修正した．web server のセキュリティ対策については後述する．

```
<IfModule mode_rewrite.c>
   RewriteEngine On
   RewriteCond %{HTTPS} off
   RewriteRule
^(.*)$   https://%{HTTP_HOST}%{REQUEST_URI}
[R,L]
</IfModule>
```

図 10　Apache メイン設定ファイル (抜粋)

3.5.2 MariaDB

使用バージョン：MariaDB10.5

認証プログラムの結果をを格納するデータベースを構築した．以下を行なった．

・MariaDB のバージョンアップ
・MariaDB の TLS 化 （自己署名証明書）
・テーブル作成
・権限設定

TLS 化に関しては，openssl コマンドで証明書を発行し，下図 11 のようにサーバの設定ファイル「/etc/my.cnf.d/server.cnf」に証明書があるパスを追記した．

```
[mariadb-10.2]
log_error
ssl_ca = /etc/my.cnf.d/certificates/ca-cert.pem
ssl_key = /etc/my.cnf.d/certificates/server-key.pem
ssl_cert = /etc/my.cnf.d/certificates/server-cert.pem
```

図 11　MariaDB メイン設定ファイル(抜粋)

ここで，証明書を格納するディレクトリは my.cnf.d 配下に設定する必要がある．bundle がないディレクトリに証明書を格納すると，have_ssl が ”DISABLED” となってしまう．これは SSL 証明書と鍵で復号できていないことを表している．

TLS が有効化されると，下図 12 のように “have_openssl”, “have_ssl” の両方に YES が表示されることが確認できる．

```
MariaDB [(none)]> show variables like '%ssl%';
+---------------------+----------------------------------------+
| Variable_name       | Value                                  |
+---------------------+----------------------------------------+
| have_openssl        | YES                                    |
| have_ssl            | YES                                    |
| ssl_ca              | /etc/my.cnf.d/certificates/ca-cert.pem |
| ssl_capath          |                                        |
| ssl_cert            | /etc/my.cnf.d/certificates/server-cert.pem |
| ssl_cipher          |                                        |
| ssl_crl             |                                        |
| ssl_crlpath         |                                        |
| ssl_key             | /etc/my.cnf.d/certificates/server-key.pem |
| version_ssl_library | OpenSSL 1.0.2k-fips  26 Jan 2017       |
+---------------------+----------------------------------------+
```

図 12　ssl status on MariaDB

下図 13 のとおり，wireshark で暗号化されている．

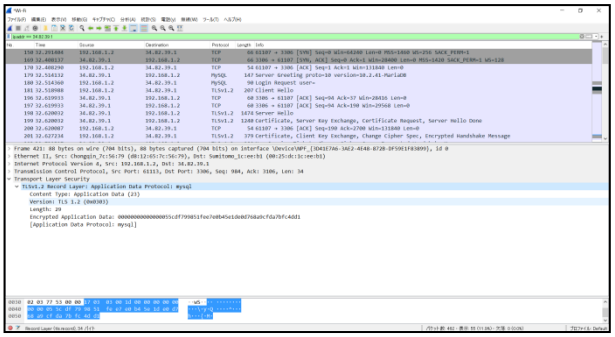

図 13　暗号化の確認

下図 14 は，今回使用するため各カラムに設定を施したテーブルである．

```
+-----------+-------------------------+------+-----+---------+-------+
| Field     | Type                    | Null | Key | Default | Extra |
+-----------+-------------------------+------+-----+---------+-------+
| Entrydate | date                    | YES  |     | NULL    |       |
| Entrytime | time                    | YES  |     | NULL    |       |
| ID        | int(4) unsigned zerofill | YES  |     | NULL    |       |
| Name      | varchar(20)             | YES  |     | NULL    |       |
| Exittime  | time                    | YES  |     | NULL    |       |
| Staying   | int(11)                 | YES  |     | NULL    |       |
+-----------+-------------------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

図 14　テーブル設定

ID に関しては，「ゼロ埋め処理」のため “zerofill” を指定した．ゼロ埋めとは，指定された桁数に値を揃えるために，本来定義したい数値の左側に不足している桁数分「0」を付加する処理のことを指す．”ZEROFILL”を指定した場合，自動的に”UNSIGNED” も設定される．これは，マイナスの値の格納は出来なくなり，0 以上の整数値データのみが取り扱い可能となるオプションである．

下図 15 は，権限付与の結果である．

```
MariaDB [(none)]> show grants for 'ssl_user'@'%';
+-----------------------------------------------------------------------------------------------------------------------------------------------------------+
| Grants for ssl_user@%                                                                                                                                     |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'ssl_user'@'%' IDENTIFIED BY PASSWORD '*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19' REQUIRE SSL |
| GRANT SELECT, INSERT, UPDATE, CREATE, DROP ON `Auth_management`.`test_ssl` TO 'ssl_user'@'%'                                                              |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

図 15　権限設定

以上で MariaDB の設定が完了した．

3.6　フロントエンドの構築

3.6.1 PHP

使用バージョン：PHP7.1

今回は，ログインページ（login.html）とデータベース管理画面（success_log.php）を作成した．データベース管理画面はリアルタイムで DB の情報を取得できるよう PHP で動的な web サイトを作成した．また，検索機能をつけることで利便性を向上させた．検索機能は，mysqli モジュールを使用してオブジェクト指向でコーディングした．工夫した点としては以下 2 点である．

・POST メソッドの使用
・empty 関数の使用

一点目に関して，一般的に，情報の登録などを伴わない情報の取得などは GET メソッドで実装されることが多いが，今回は POST メソッドを使用した．サーバへ送る SQL 文のデータ量とセキュリティ面を考慮したためである．

二点目に関して，MariaDB のカラムの設定において未入力はデフォルトで NULL が格納されているようになっている．isset 関数を用いると，未入力のカラムでも検索画面に出力されてしまう．したがって empty 関数を用いることで意図しない結果が出力されないようにした．

以下，図 16 と図 17 は完成した各 web サイトである．

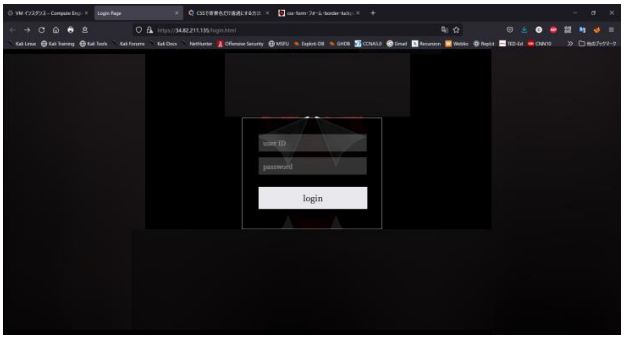

図 16　ログインフォーム画面 (login.html)

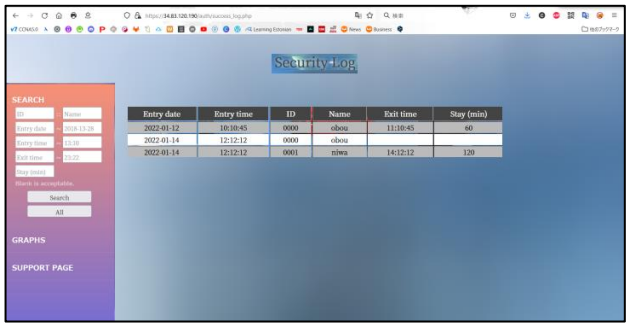

図 17　ログ管理画面 (success_log.php)

図 17 を見ると 1/14 は 2 名が入室したことが分かる．ユーザ “niwa” は 2 時間の滞在ののち退室したことが分かるが，ユーザ “obou” は退室時刻および滞在時間が空欄であり，まだ在室しているであろうことが分かる．これは入室時点では入室時刻のみタイムスタンプが押され，その後再び同一人物が認証を行った際に検出された「ユーザ名」と「退室時刻が NULL であるカラム」を持つレコードを探し，そのレコードの退室時刻を現在の時刻に UPDATE することで実現している．

認証をかけるディレクトリとして「/var/www/html/auth」を新規作成し，認証後に参照されるページ（success_log.php）を格納した．

以上でフロントエンドの構築は完了である．

4.　監視スコープの実装

認証システムが稼働しているインスタンスとは別に新規インスタンスを作成し，Ubuntu 20.0.4LTS をインストールした．Ubuntu に zabbix server を，CentOS に Zabbix agent をインストールし，AI Platform はエージェントレスで監視する構成である．

4.1　Zabbix

使用バージョン：Zabbix5.4

一連の認証システムが安定動作しているか監視するシステムは Zabbix を用いて構築した．以下を行なった．

・Zabbix server dashboard の TLS 化 （自己署名証明書）
・Zabbix server - Zabbix agent 間の TLS 化 （事前共有鍵 psk）
・ローカルおよびリモートホスト監視設定
・トリガー/アクション/アラーム設定 （slack）
・Map および Graph の設定

4.1.1 Zabbix server の設定

以下は Firewall の解放ポートである．

zabbix-up：10050, 10051
zabbix-down：10050,10051

監視対象としたのは，下表 5 の通りである．

表 5　監視対象一覧

監視対象ホストおよびアイテム
IaaS (CentOS，Ubuntu)： CPU，memory，チェックサム監視，ファイル更新監視
PaaS (AI Platform)：Ping 監視，SSH 監視
Apache (CentOS)：HTTPS 監視，web シナリオ監視，ユーザ数監視
MariaDB (CentOS)：MySQL 監視

4.1.2 Zabbix agent の設定

使用バージョン：zabbix_agentd （MIRACLE ZBX）4.0.36-2

Zabbix の監視方法にはエージェントレス監視，トラップ監視，ポーリング，アクティブチェックの 4 種類がある．今回は，CentOS にポーリング監視，AI Platform にエージェントレス監視を実装した．

設定ファイルにサーバの情報を追記し，Web UI で必要な設定を行なった．下図 18 は，各リモートホストに対し，監視対象や暗号化設定を追加した結果である．

図 18　Zabbix 設定画面 (抜粋)

4.1.3 障害発生時の挙動確認

障害時通知として，Zabbix server のアクションと Slack API の設定を行なったので，障害時にどのような挙動を示すのか確認した．図 19 は，監視対象ホストの IP アドレスが変わったときのキャプチャである．接続先が見つからないため，”not available” となっている．障害から復旧すると Problem （オレンジ）から Recovery （グリーン）の表示になった．

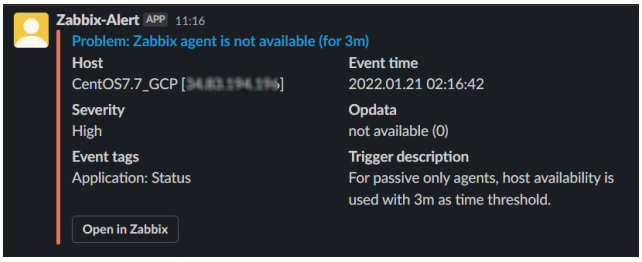

図 19　zabbix alert via slack

ここで製作過程において実際に有用なアラートを受け，対応したケースがあるので紹介する．図 20 は swap 領域が足りずアラートが出た際のキャプチャである（既に解決したので Recovery: グリーンとなっている）．

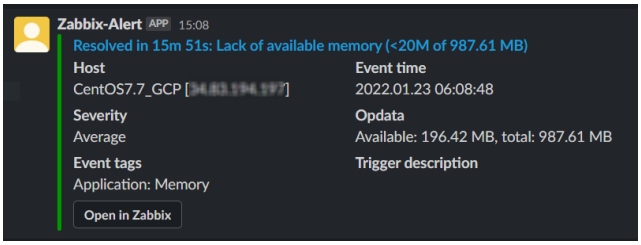

図 20　zabbix alert swap

今回は無料枠でのみの運用であったため，デフォルトの RAM 容量が少なくサーバがダウンしかけた．SSH 接続もできなくなったため，急いで対応を行ない，解決することができた．下図 21，下図 22 は対応前と対応後のメモリ一覧である．

```
[root@samurai-db log]# free -tm
              total        used        free      shared  buff/cache   available
Mem:            987         312         422           7         252         532
Swap:             0           0           0
Total:          987         312         422
[root@samurai-db log]# vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 433196   2208 256536    0    0   585    34  180  225  1  1 93  4  0
[root@samurai-db log]#
```

図 21　memory before

```
[root@samurai-db /]# free -m
              total        used        free      shared  buff/cache   available
Mem:            987         322          59           7         605         513
Swap:          1023           0        1023
```

図 22　memory after

調べてみると，swap が 0 であることが分かった．GCP をはじめとするクラウドやレンタルサーバでは swap 領域が確保されていないことが多いようである．クラウド上のサーバは SSH への不正アクセスや DoS 攻撃が頻繁に発生しログファイルの肥大化とメモリの逼迫を起こすケースが多いため，事前の確認を怠らないようにすべきである．

下図 23 は，全ての設定が完了した時点でのダッシュボードである．

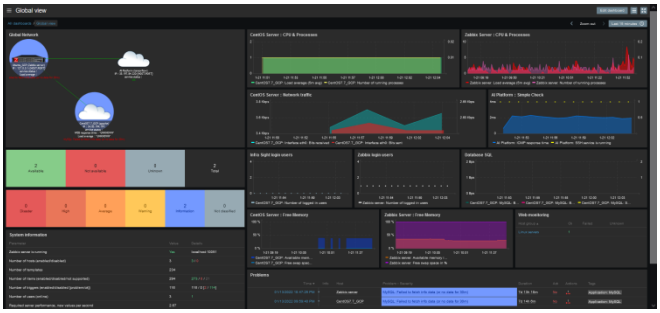

図 23　Zabbix ダッシュボード

5.　評価

5.1　Functionality

必要な機能を満たしているかに関して，課題として以下 4 点が挙げられる．

① yolo による顔認証における課題

yolo では，同一人物が一定時間以上映ることでその本人だと判断しているが，クライアントの性能や負荷によって認証に使われる画像の枚数が変わることになり，精度が安定せず，基準を満たした認証機能を保てない可能性がある．

改善策としては，検証に使う枚数と時間を顔認証の結果として考慮するべきである．最低枚数，最低時間を設定することで高い性能のクライアントであったとしても一定時間を必要とし，低い性能のクライアントでも枚数が得られる．

② 声紋認証における課題

周りが騒音である場合には使えないという点である．録音開始位置に使用される規定値は，設定ファイルから簡単に変えられるようになっているため，多少のノイズには対応できる．しかし，認証プログラムに渡す前にノイズ除去等の処理を行わないため，精度に影響が出てしまうと考えられる．

③ 認証タイミングにおける課題

声紋認証が終わったタイミングで顔認証の結果を参照しているため，タイミングのずれで，正しく認証が通らない可能性がある．声紋検証終了時に yolo が一定時間経過していない場合，現状では本人判定 “不可”と判断されてしまう．

改善策としては，声紋認証と顔認証の結果をそれぞれ一定時間保持し，他のプログラムがその結果を定期的に確認することである．一定時間保持することでタイミングのずれは影響しなくなる．またこのような変更を行うことで，各認証プログラムが保持すべきデータを GUI 表示を行うためのステータス値，認証の結果，のように明確にすることができ，新しく指紋認証等認証方式を採用する際も容易に増やすことができる．

④ 監視スコープにおける課題

学習構築モデルサーバとして AI Platform (PaaS) を選択したため，Zabbix agent や snmp agent 等の監視エージェントをインストールすることができず，簡易監視しか実装することが出来なかった．また，Zabbix を TLS に対応させたことにより MySQL 監視でデータ抽出が出来なくなってしまった．zabbix_get コマンドではオプションを付与することでリモートから情報を対象サーバに対し MySQL 監視のデータを抽出することはできるものの，Web UI を通すとデータが反映されないので Web UI の MySQL Template のパラメータを TLS に対応させたスクリプトに変更する必要があると思われるが，解決に至っていない．現状では，本来想定していた以下の監視をすることができていない．

・次回用モデル構築に必要なデータがクライアントからサーバに送られ指定のディレクトリに格納されたかのチェック
・独自プログラムの動作確認
・MySQL 監視 on TLS

5.2　Reliability

一定の条件下で安定して期待された役割を果たすことができるかについて，エラー発生時の処理が不十分であるという課題が挙げられる．ログ出力先の Mariadb サーバが停止するなどプログラムの流れに障害が発生した場合停止し

てしまう．しかし，障害点がどこであるか Zabbix で分かるようにはなっている．

5.3 Usability

使いやすさに関しては，おおむね良好と判断した．
認証の速度として検出速度は **30FPS** である．実証の結果，どのくらい話すかにもよるが認証開始から認証終了まで全部で **10~15 秒程度**で判定可能である．これは，リアルタイムで実装するにあたり十分な速度が出ていると判断してよい．なお，長く話せば話すほど精度は良くなるので 5 秒以上の音声吹込みが望ましい．

5.4 Vulnerability

Web サーバには，現在の認証情報などの機密情報を表示するため，ブラウザを介した入力に有害なスクリプトが挿入されることや，不審なリクエストが送られることで，情報を盗難される危険性があるため，対策が必要である．そのため，我々は評価に脆弱性の項目を設けた．

主に，Web アプリケーションの脆弱性診断ツールの一種「nikto（ver2.1.6）」と「OWASP ZAP」を用いてローカルから検証を行った．「nikto」とは Netspark 社（英）がスポンサーをしている，OSS Web アプリケーションセキュリティスキャナーである．Perl で開発された CUI であるためほとんどの OS で動作し非常に有用なツールである．今回は安定版の Nikto 2.1.6 を使用した．図 24 は，対策を行う前の nikto を用いた検出結果である．

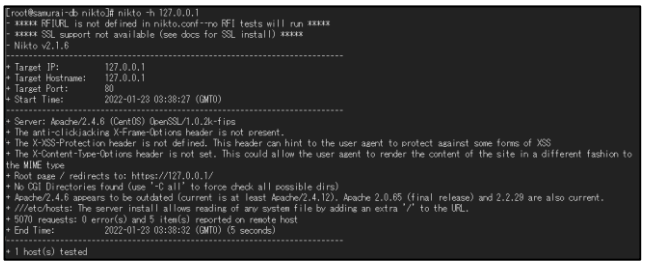

図 24　nikto check before

この結果を踏まえ，Apache メイン設定ファイルに図 25 の内容を追記した．

```
#Enable Security stuff
ServerSignature Off
ServerTokens Prod
TraceEnable Off

#X-XSS-Protection
Header always set X-XSS-Protection "1; mode=block"

#DDoS Protection
LimitRequestBody 10485760
LimitRequestFields 50
```

図 25　Apache メイン設定ファイル(抜粋)

下図 26 は上記対策後のキャプチャである．各種バージョン情報の隠蔽やクロスサイト系の脆弱性に対して対策が出来たことが分かる．しかし，パストラバーサルの脆弱性が残ったままである．

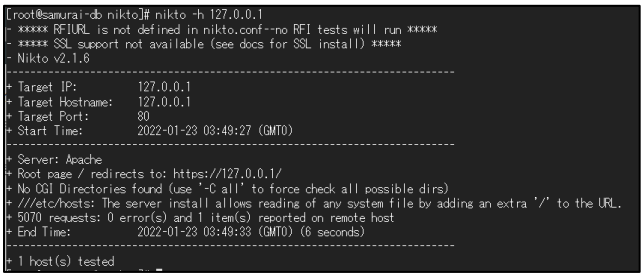

図 26　nikto check after

nikto は使い勝手の良いツールではあるが安定版が 2013 年リリースのもので開発が盛んでないことから，別のツール「OWASP ZAP」を用いた診断も併せて行った．今回は，対象 Web サーバにリクエストを送信し，脆弱性を検出する．OWASP ZAP は GUI と組み合わせると非常に使い勝手の良いツールであるが，不要なリソースを削減するためにインスタンスには GUI を導入しておらず，daemon モードで動作させた．そのため，xml 形式のレポートをそのまま読むことになる．

図 27 は，ログインページに Basic 認証を実装していた時の実際のレポートである．また，検出された脆弱性の概要を表 6 に示す．なお，OWASP　ZAP のレポートはいずれも一部抜粋となっている．

図 27　OWASP ZAP のレポート(1)

表 6　検出された脆弱性(1)

CWE 番号	説明	対策
CWE1021	レンダリングされたユーザインターフェースレイヤまたはフレームの不適切な制限	X-Frame-Options ヘッダーを HTTP レスポンスに含めない
CWE693	保護メカニズムの不具合	X-Content-Type-Options を nosniff に変更

以上の ZAP によるレポートを受け，図 28 のように対策をした．CWE1021 に対しては，X-Frame-Options の使用を無効かする設定を，CWE693 については X-Content-Type-Options の nosniff をレスポンスヘッダに含める設定をした．

```
<IfModule mod_headers.c>
        Header always append X-Frame-Options “DENY”
        Header always set X-Content-Type-Options nosniff
</IfModule>
```

図 28　CWE1021 の対策

対策の有効性を確認するために，もう一度 OWASP ZAP を実行し，レポートを確認する．出力したレポートを図 29 にその概要を表 7 に示す．

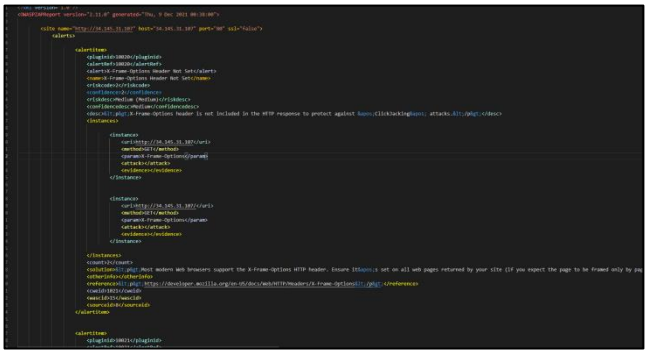

図 29　OWASP　ZAP のレポート(2)

表 7　検出された脆弱性(2)

CWE 番号	説明	対策
CWE525	各 Web ページおよび関連するフォームフィールドをキャッシュする範囲を適切に指定していない	キャッシュの暗号化や不必要に情報をキャッシュに保存しないなど

表 7 のように CWE1021 が検出されなくなったことが確認できた．新たに CWE525 が検出された．しかし CWE525 については，公共のデバイスでの利用を考慮しない．対策する場合は web サーバ側で強制的にキャッシュの保持を禁止することや，キャッシュの暗号化などが有効になる．

次に，Form 認証を実装後にフォーム認証のディレクトリである/auth ディレクトリに OWASP　ZAP による脆弱性調査を行った．図 30 が実際のレポート，表 8 にその概要を示す．

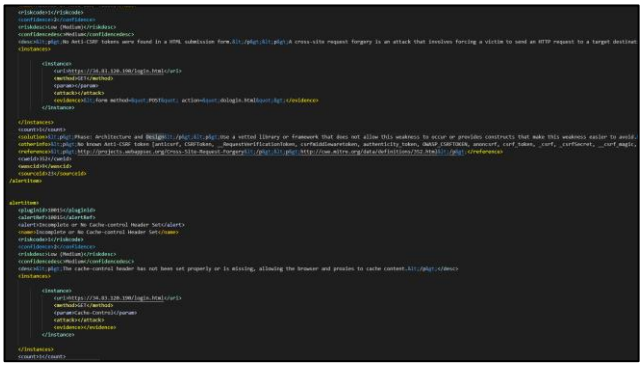

図 30　OWASP　ZAP のレポート(3)

表 8　検出された脆弱性(3)

CWE 番号	説明	対策
CWE352	クロスサイトリクエストフォージェリと呼ばれる攻撃．攻撃者がクライアントを騙し，意図しないリクエストを Web サーバに送信させる可能性がある	リクエスト中にトークンと呼ばれる乱数を送信するタグを挿入しておき，リクエストを受け付けた際にトークンの正当性を確認する
CWE525	各 Web ページおよび関連するフォームフィールドをキャッシュする範囲を適切に指定していない	キャッシュの暗号化や不必要に情報をキャッシュに保存しないなど

CWE352 の対策としては，シンクロナイザートークンパターンという方法で対策を考えている．サーバ側でトークンを生成する．クライアントによってリクエストが発行されると，サーバは、ユーザーセッションで見つかったトークンと比較し，リクエスト内のトークンの有効性を検証する．トークンがリクエスト内で見つからなかった場合，またはクライアントから送られてきたトークンの値がユーザーセッション内の値と一致しない場合は，リクエストを中止する．

6.　まとめ

本研究では，python を用いた顔認証＋声紋認証システムの開発を中心に，各種サーバの構築およびセキュリティ設定，ペネトレーションツールを使用したシステムの評価を行なった．マシンスペックにもよるが，今回用意したノート PC ではストレスを感じない程度の速度を出すことができ，背景がシンプルな無地であれば実際に使用できる非接触型のシステムを構築することができたと言える．本プロジェクトの Docker イメージを作成し，公開および配布によって誰でも容易に多要素認証を導入できる．

評価に関して，実施したのはアプリケーションレベルのみなので、IPS の導入を検討すべきだと感じている．

今後の発展性としては，本製作を基盤として入退室を伴う事業を展開している顧客に対し，そのニーズにあわせてシステムをカスタマイズしていく，所謂 Sler のような開発もおもしろいと思う．また，実際にハードウェアと組み合わせてドア付近に設置できるような IoT 機器にし，パッチを当てるのが難しい IoT 機器特有の脆弱性の問題を Zabbix と Vuls を組み合わせることにより，リアルタイムで監視および通知できるシステムまで拡張できれば実用性に富んだものができると考えている．

参考文献

[1] “MySQL で利用可能な整数型について解説！”. https://style.potepan.com/articles/19318.html, (2022-1-8)
[2] “zerizeri のエンジニア技術ブログ - zabbix5.0 の監視項目設定 -”. https://security-blog-it.com/197/, (2022-1-4)
[3] “Zabbix Documentation 2.0 - 1 Zabbix の定義 -”. https://www.zabbix.com/documentation/2.0/jp/manual/concepts/definitions, (2022-1-4)
[4] “CWE-1021: Improper Restriction of Rendered UI Layers or Frames”.　https://cwe.mitre.org/data/definitions/1021.html, (2022-1-9)
[5] “CWE-693: Protection Mechanism Failure”.

https://cwe.mitre.org/data/definitions/693.html, (2022-1-9)
[6] “CWE-352: Cross-Site Request Forgery (CSRF)”. https://cwe.mitre.org/data/definitions/352.html, (2022-1-9)
[7] “Cross-Site Request Forgery Prevention”. https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html#synchronizer-token-pattern, (2022-1-9)
[8] “CWE-525: Use of Web Browser Cache Containing Sensitive Information”. https://cwe.mitre.org/data/definitions/525.html, (2022-1-9)

汎用ロジックICで実装する換字式暗号回路

日本電子専門学校 電子応用工学科　　井口陽太 (20EO0101)

2022/02/03

概要

様々な情報をコンピュータで扱う現代，データ保護はますます重要視されている．データを狙う脅威は日々進化しており，それに対抗する手段もまた進化している [1]．データ保護の手段の一つである**暗号化** とその関連技術*1 は，通信やブロックチェーン [2] など，多様な分野・技術でデータの安全性・完全性の担保に利用されており，私たちの生活の安心を支えている．

それらは従来，汎用プロセッサで処理*2 するケースが多かったが，近年ではセキュリティや処理効率の観点から**専用プロセッサ**で処理するケースが増えている．ハードウェア実装は，今となっては HDL やツールを用いることによって簡単に設計できてしまう．しかしながら，ツールで自動生成された回路は効率的である一方，人間にとって非常に煩雑であるケースが多く，動作原理を理解するのは非常に難しい．また，ツールに頼りすぎることはエンジニアとしての質の低下につながり，ツールのバグにより自動生成された回路が意図しない動作をするケースなどに遭遇した際，迅速に問題解決できないことが懸念される．

以上を踏まえ，本研究では **暗号化と論理回路の理解を目的として**，古典・現代暗号の調査，また，簡素な多表換字式暗号アルゴリズムをハードウェア実装した,.

1　まえがき

個人情報や決済情報などをコンピュータで扱うようになり，社会はますます便利になった．今やスマートフォン 1 台で電車・バスに乗降でき，自動販売機やコンビニエンスストア・飲食店などで決済できる．近年は NFC *3 決済サービス*4 や，バーコード（QR コード）決済サービスの普及・成長 [3][4] が著しく，決済事情が一変した人も多いだろう．

多くの人々が何気なく利用している便利なサービスだが，そのようなサービスは大前提として安心・安全に利用できるものでなければならない．利用者の個人情報は攻撃者にとって恰好の的であり，事実として，そういったデリケートな情報を取り扱う端末・サーバは日々，様々な攻撃に脅かされている．それでも情報が漏洩せずに安寧が保たれているのは，ファイアウォールや EPP・EDR，暗号化をはじめとしたセキュリティ技術，また，物理サーバの所在地秘匿，メンテナンス・操作する人間の限定，技術者による迅速な対応といった，原始的な対策・緻密なインシデント管理の賜物である．

本研究では，セキュリティ技術の中でも根本的な技術である「暗号化」と，近年のトレンドである「専用ハードウェアによる処理」に焦点を絞り，最新・応用技術を理解する足掛けとして，古典・現代暗号の調査，並びに簡素な多表換字式暗号アルゴリズムのハードウェア実装に取り組んだ．

2　暗号とは

暗号とは，当事者にしか分からない法則に則り，元の情報*5 を第三者にとって一見意味がない情報に置き換えたものである [5]．暗号の歴史は長く，紀元前より用いられている．現代では生活の安心・安全を支える重要な技術となっており，日々，安全性の検証や新技術の研究が進められている．

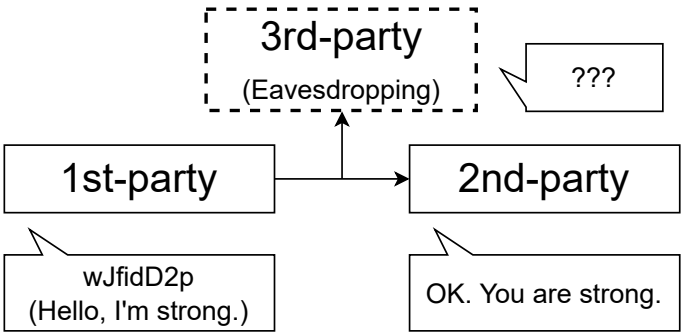

図 1　暗号の概念

2－1　何故暗号化が必要なのか

住所や決済情報が流出しても困らないという人はいないだろう．住所が流出すれば犯罪に巻き込まれ物理的な被害を被る恐れがあり，決済情報が流出すれば経済的な被害を被る恐れがある．そういった被害を防ぐため，私たちは安心・安全を脅かす脅威からデータを保護しなければならない．

特に，不特定多数のノードを経由して通信するインターネットでは，暗号化は非常に重要である．平文で通信した場合，経由するノードで改ざん・盗聴といった攻撃を受けてしまう可能性がある．もし，それが個人情報を取り扱うサービスを提供するサーバであった場合，攻撃者はそれを狙わない理由はない．

しかし，データを狙う脅威は通信経路に限らない．通信先サーバや私たちが普段使用しているコンピュータにも潜んでいる可能性があり，一概に通信を暗号化すればデータを保護できるとは言えない．暗号化対象のデータ及び暗号化で使用する鍵を適切に管理する，または不正なプログラムがコンピュータ上で動作していないか監視・対処するなど，根本的な対策も重要である．

---

*1　共通鍵暗号，公開鍵暗号，ハッシュ関数など．

*2　暗号アルゴリズムをソフトウェアで実装し，CPU などのメインプロセッサでの処理する．

*3　Near Field Communication.

*4　1）　我が国では NFC Type F（FeliCa）を用いたものが主流である．
　　2）　近年，我が国で普及し始めたクレジットカードの非接触式決済は NFC Type A/B を用いている．
　　また，NFC Type A/B は我が国の運転免許証やマイナンバーカードなどでも用いられている．

*5　言葉やデジタルデータなど，通信・会話において伝達するものを指す．

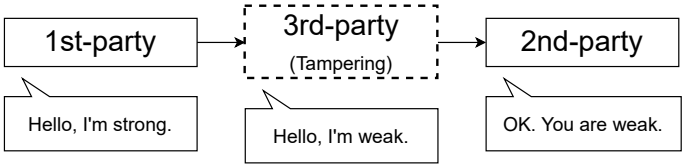

図 2　通信の改ざん

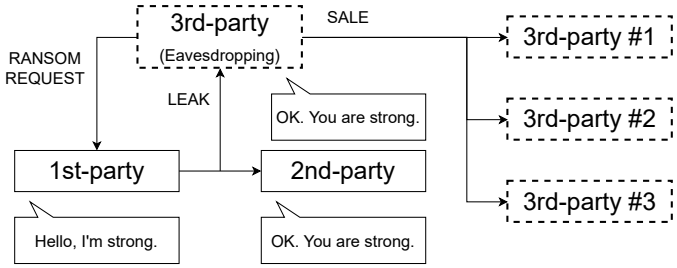

図 3　通信の盗聴

2－2　暗号化以外での通信保護

通信経路においてデータを保護する手段は，暗号化だけではない．原始的なものとして，データを電磁的記憶媒体を用いて直接手渡すといった方法がある．伝送線を介さないこの方法は，最も安全で確実性が高い．しかし，通信網が発達した現在においては非効率であり，また，紛失や物理的な攻撃によってデータが漏洩する恐れがある，

その他に，通信相手と専用線で結び，第三者が介入できない状態で通信するといった方法もある．しかし，もし伝送線を特定されてしまった場合，伝送線から射出される電磁ノイズなどから伝達内容を解析・特定されてしまう恐れがある．伝送線に物理的な防御を施さなかった場合には，伝送線を切断され通信の妨害，あるいは盗聴されてしまう恐れがある．

どちらの手段も不特定多数との通信には適しておらず，時間的・経済的・資源的に無駄が多い．現在のコンピュータネットワークの形態と合致せず，現在となっては特殊な事情がある場面でしか用いられていない．

以上のことから，共有の伝送線上でデータを暗号化して伝達することは，とても理にかなっていることが分かる．もっとも，暗号化と前述の手段を組み合われば，セキュリティの観点からして非常に強力な対策となる．

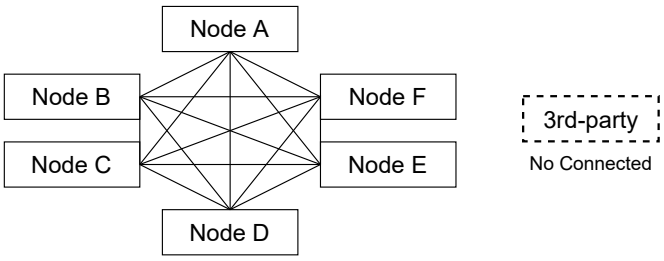

図 4　専用線による通信

2－3　古典暗号と現代暗号

暗号史は長く，古来より様々な手法で情報は暗号化されてきた．有名なものとして「シーザー暗号」や「エニグマ」といった暗号を知る人は多いだろう．この二つを含め，旧時代で用いられた暗号は，大きな括りとして「古典暗号[*6] 」に分類される．対して，現在主流な AES や RSA といった暗号は「現代暗号」に分類される．暗号は原則，「アルゴリズム（方法）」と「鍵」に分離して考えることができ，その扱いによって二つを分類することができる．

一般的には「アルゴリズム[*7] を公開しても信頼性が損なわれない暗号」を「現代暗号」，そうでないものを「古典暗号」と分類されており，「現代暗号」をより詳細に説明すると，「現代の数学理論に基づいて考え出され，優れた安全性・堅牢性が証明・評価されている暗号」である．先人の執筆物では，二つの中間として「近代暗号」という分類を設けているケースがあるが，本稿では「近代暗号」も「古典暗号」として取り扱う．

2－3－1　古典暗号

古典暗号は現代暗号以前に用いられた暗号である．大抵のものはアルゴリズムから鍵を特定することが容易なため，鍵と共にアルゴリズムも隠蔽しなければ安全性が担保されない．しかし，アルゴリズムを隠蔽しても絶対に解読されない保証はなく，単純なアルゴリズムであった場合，ヒントを必要とせずに解読されてしまう可能性が高い．

例として，シーザー暗号は暗号化前の数値（文字）に対して決まった数値を加算（決まった文字分ずらす）という方法で暗号化しており，「加算」がアルゴリズム，「決まった文字」が鍵と対応している．シーザー暗号は非常に単純なアルゴリズムであるため，鍵が不明でも一文字ずつずらして試行すれば鍵を特定できてしまう．

単一換字式暗号では，一見すると解読できないように見えるが，「頻度分析」を用いて解読することができる．一つの表を用いた暗号では，暗号化前の数値と暗号化後の数値の組み合わせは一意となるため，数値の出現頻度は隠蔽されない．そのため，どの数値がどの程度の頻度で出現するか分析することによって，パターンから元のデータを特定できてしまう可能性が高い．

古典暗号の解読は人間の手作業では難しいが，処理能力が非常に高い現代のコンピュータでは高速に処理することができ，解読されてしまう可能性が極めて高いため，脆弱な暗号とされている．そのため，現代ではその処理能力を以てしても解読できない，高度な手法の暗号が利用されている．

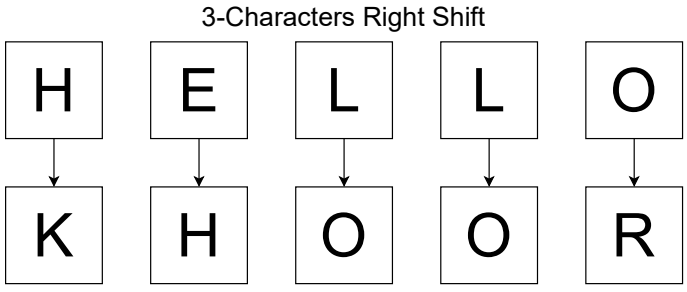

図 5　シーザー暗号

*6 1）　主に単純演算や表を用いた単一換字によるものが多い．
2）　発展的なものとして，複数の表を用いたものもある．
3）　近代では，機械を用いてより複雑に表を組み合わせた暗号も用いられた．
＊）　「換字」は「かえじ」と読む．慣習的には「かんじ」と読めるが，他の言葉との混同を避けるため，このように読む．

*7　「アルゴリズム」本来の使われ方として，加算するだけのような単純な処理はそれに当たらないが，本稿では内容の都合上，「アルゴリズム」は「アルゴリズム（本来の使われ方）」と「方法」の両方を指すものとする．

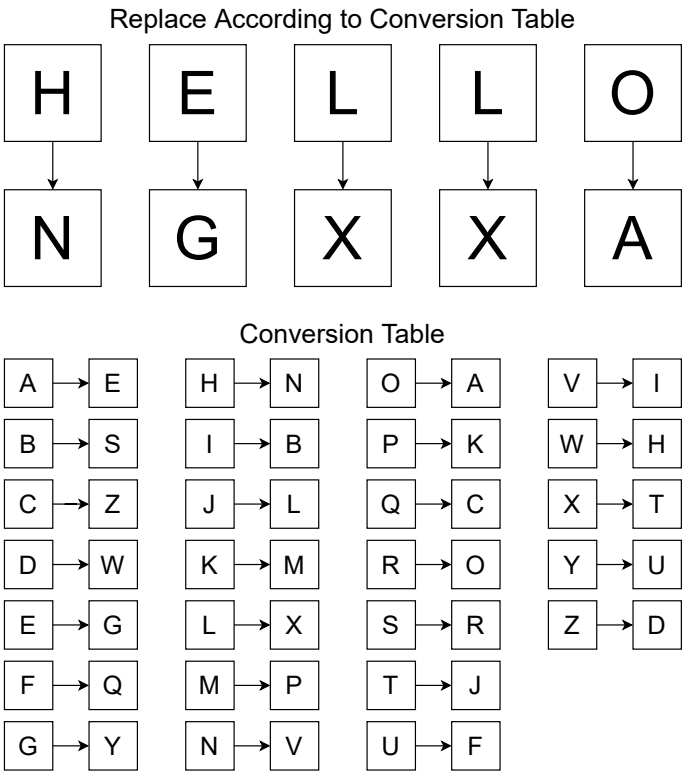

図 6　単一換字式暗号

2－3－2　現代暗号

現代暗号は古典暗号に対して，アルゴリズムを公開しても安全性を担保できる暗号 [6] である．現代の数学理論に基づいて考え出され，解読の難しさが証明・評価されている．何より，発表されてから今日まで，幾人もの研究者や攻撃者によって 議論・攻撃されてもなお，破られていない*8 のが非常に強力な根拠である．逆に，公開されていない暗号アルゴリズムは客観的な評価がないため，その安全性を証明できない．一般的には独自の暗号アルゴリズムは危険であり，利用すべきではないとされている．

現代暗号はさらに「共通鍵暗号*9 」と「公開鍵暗号」に分類でき，それぞれの用途や特性は異なる．現代暗号は現代においてデータ保護の要であり，通信・ストレージの保護・デジタル署名など，私たちの安心の根幹を支えている．

2－3－2－1　共通鍵暗号

共通鍵暗号は暗号化と復号に共通の鍵を用いる暗号*10 である．主なものとして，脆弱性が発見され現在では非推奨となっている DES と，DES の後継であり現在実用されている AES がある．

AES はアメリカ国立標準技術研究所が公募し，採用された Rijndael *11 をベースとして，ブロック長 128 bit 固定，鍵長 128 bit/192 bit/256 bit から選択できる暗号である．2001 年 3 月に正式に公表されたが，現在に至るまで解読されておらず，その安全性・信頼性は極めて高い．

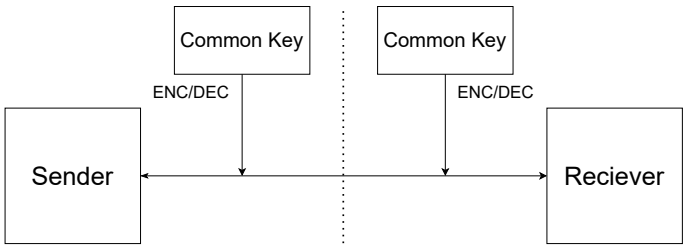

図 7　共通鍵暗号

2－3－2－2　公開鍵暗号

公開鍵暗号は暗号化と復号に異なる鍵を用いる暗号である．「現代暗号」の狭義ではこちらを指すことがある．主なものとして，素因数分解を利用した RSA や，楕円曲線を利用した暗号が実用されている．

いずれも，演算が困難*12 であることを安全性の根拠としており，現在最高の演算性能を持つ古典コンピュータを以てしても膨大な時間を要するとされている．

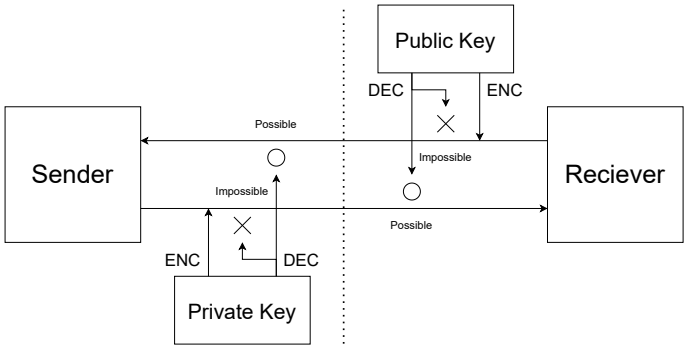

図 8　公開鍵暗号

2－3－2－3　速度検証

前述の通り，暗号化は平和な世界であれば本来必要のない処理である．そのため，暗号アルゴリズムはできるだけ高速であることが望ましい．特性が異なる AES (共通鍵暗号) と RSA (公開鍵暗号) だが，処理時間は一体どの程度差があるのか．それぞれ同じ条件で処理時間を計測した結果*13 を表 1 に示す．

結果は歴然で，AES が非常に高速である．AES は並べ替えや XOR といった高速な演算を組み合わせて構成されるアルゴリズムであり，リソースが潤沢な現在のコンピュータであ

*8　2013 年時点で，鍵長が 1024 bit の RSA はスーパーコンピュータを用いて 1 年未満で解読できるとされている．

*9　「共通鍵」は「対称鍵」と言うこともある．

*10　暗号化と復号に共通の鍵を用いる点では，古典暗号もまた「共通鍵暗号」に分類できるが，古典暗号は「古典暗号」としての分類であり，現代暗号の共通鍵暗号とは区別する．

*11　Rijndael は鍵長・ブロック長ともに 128bit から 256bit まで 32bit 単位で指定できる．

*12　効率的に解くアルゴリズムが確立しておらず，現在知られている最も効率的なアルゴリズムを用いてもなお演算に膨大な時間を要する．つまり，現実的な時間で解くことが難しいということを「困難」としている．

*13　元のデータは 10 MiB，AES は鍵長 256 bit，RSA は鍵長 2048 bit（ハッシュ関数を SHA-1 とする OAEP パディング）で暗号化・復号に掛かる時間を計測した．

*14　RSA は非常に大きな素数の積の素因数分解が困難であることを暗号の安全性の根拠としているため，暗号単位は 1024 bit や 2048 bit（現在主流）といった非常に大きなサイズとなる．
現在主流な家庭向けコンピュータのレジスタサイズは 64 bit であるため，そのような大きなサイズのデータを処理するには分割しなければならない．

れば比較的高速に処理することができる．対して，RSA は暗号単位が非常に大きなサイズであるため，分割して処理[*14] しなければならないうえ，累乗や剰余といった比較的重い演算をするため非常に時間が掛かる．

AES は非常に高速に処理できるため，一見すると総当たり攻撃での解読が懸念されるが，AES-256 の鍵長は 256 bit であり，2^{256} の鍵が存在する．一回の処理に 20 ms 掛かるとすると全ての鍵を試すには，式 1 [*15] より，およそ 7.32×10^{51} 京年を要する[*16]．しかし，AES では初期化ベクトル (IV)[*17] という概念が存在しており，同一の鍵でも初期化ベクトルによって処理後のデータが変化する仕様のため，全ての組み合わせを試すには更に時間を要する．

また，RSA を総当たりで解読するには，積が 2048 bit になる二つの素数を全て試す必要があるうえ，復号一回に AES のおよそ 2894 倍の時間を要する[*18]．RSA の原理上，公開鍵の一つである二つの素数の積を素因数分解できれば解読できるのだが，現在，巨大な合成数を効率的に素因数分解できるアルゴリズムは見つかっておらず，現在最も効率が良いとされるアルゴリズムを用いたとしても現実的な時間で解けない [7]．

解読に膨大な時間を要するとなると，マシンの故障，宇宙線などに起因するメモリのビット反転といった，物理的原因による演算の中断・誤結果の出力といった可能性が非常に高い．そもそも，処理を開始した人間は処理が終了するまでに死亡している可能性が高い．以上のことから，解読は事実上不可能だと考えるのが妥当である．

$$\mathrm{T\ years} = \frac{2^{\mathrm{B}}}{\mathrm{P} \times 60 \times 60 \times 24 \times 365} \tag{1}$$

T = Processing Years
B = Key Length (Unit: bit)
P = Processing per Minute

表 1　AES と RSA の速度比較

unit: ms

	AES [E]	AES [D]	RSA [E]	RSA [D]
#1	17	20	4923	65288
#2	18	15	4901	65324
#3	24	17	4929	65272
#4	18	15	4903	65211
#5	18	16	4893	65155
#6	19	46	4885	65176
#7	22	17	4878	65056
#8	19	17	4890	65289
#9	20	47	4895	65217
#10	19	15	4926	65159
Avg	19.4	22.5	4902.3	65124.7

Environment : Windows 11 21H2(22000.434)
.NET Framework 4.7.2
CPU : AMD Ryzen 9 5900HS 3.30 GHz
RAM : DDR4 16.0 GB 4266 MHz

2－3－2－4　鍵配送問題の解決

前節では AES と RSA の処理時間を比較し，AES が RSA と比較して非常に高速であることを確認できた．この結果より，公開鍵暗号の存在意義が疑問視されるが，公開鍵暗号は現代の通信において非常に大きな役割を担っている．公開鍵暗号は共通鍵暗号の大きな弱点である鍵配送問題を解決するために利用されている．

共通鍵暗号は暗号化と復号に同じ鍵を用いるという性質上，送信者と受信者が鍵を共有しなければならない．その場合，「送信者と受信者が現実世界で交換」あるいは「通信で交換」の二つしか方法がない．前者は前述した通り，非常に非合理的であり現代のコンピュータネットワークに適していない．また，後者は通信を秘匿するための鍵を通信経路上に平文で流すこととなり，その鍵を利用した通信の安全性が危ぶまれる．

しかし，公開鍵暗号は前述の通り暗号化と復号に用いる鍵が非対称であり，共通鍵暗号で大きな問題であった鍵配送問題は存在しない．問題があるならば，前節で検証した通り処理に時間を要することだろう．

現代の通信では，通信開始時に共通鍵暗号の鍵を公開鍵暗号によって暗号化して配送することで．暗号処理時間の問題も鍵配送問題を解決している．このように，共通鍵暗号と公開鍵暗号の良い点をそれぞれ生かした暗号を「ハイブリッド暗号」と呼ぶ．

図 9　ハイブリッド暗号

2－4　暗号処理専用プロセッサ

現代の通信やデータ保管において暗号化はほぼ必須と言っても過言ではない．しかし，AES や RSA といった現代暗号は強力である反面，非常に複雑な処理・手順を踏む必要があり，プロセッサに掛かる負荷が高い．また，メインシステムで鍵を管理することは，セキュリティの観点から望ましくない．そのため，近年では暗号やハッシュ処理専用のプロセッサ[*19] の需要が高まっている．

専用プロセッサを用いると，暗号処理でメインシステムのリソースを消費しない[*20] ため，スループットが向上することが期待できる．しかし，現在主流なプロセッサは従来と比較して格段に性能が向上しているため，恩恵は微々たるものかもしれない．ただ，メインシステムと分離することはセキュリティの観点からして非常に望ましく，鍵を専用プロセッサで生成・保管すれば，メインシステムは鍵を知らずに暗号処理でき，仮にメインシステムが攻撃を受けても鍵が流出する心配がない．

*15　閏年は考慮しない．
*16　初期化ベクトルが既知の場合．
*17　AES の初期化ベクトルのサイズはブロックサイズ（128 bit）と等しい．
*18　計測結果より算出．実行環境や鍵長によっては短くなるか，あるいは長くなる．
*19　GPU や TPU などのプロセッサも，メインシステムのリソース占有や演算効率を理由としてメインシステムから分離している．
*20　専用プロセッサの処理性能が低いとボトルネックとなる．

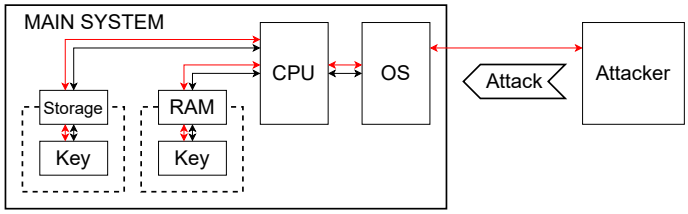

図 10　メインシステムで鍵を保管

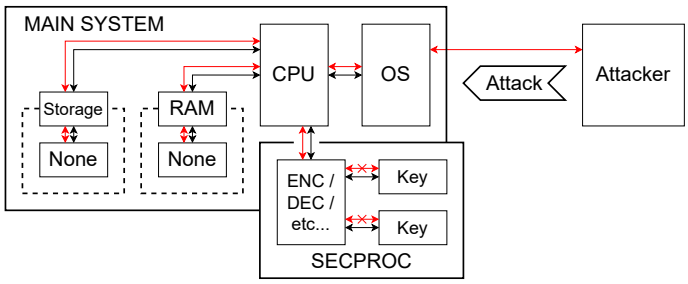

図 11　専用プロセッサで鍵を保管

3　設計・製作

第 2 章までの内容を踏まえ，簡易な暗号回路を設計した．方針としては実用性・拡張性を重視し，マイコンなどから制御しやすい I/O と，脱着・変更可能な制御回路を用意した．

3－1　システム概略

システムは制御側（マイコンなど）から見て，以下の I/O で操作・状態確認できる設計とした．

信号線名：入出力種別／概略

CS：入力／回路選択
EXE：入力／実行
DATA：入力／入力データ
PRC：出力／実行可否
WND：出力／データ入力待機
OUT：出力／出力データ

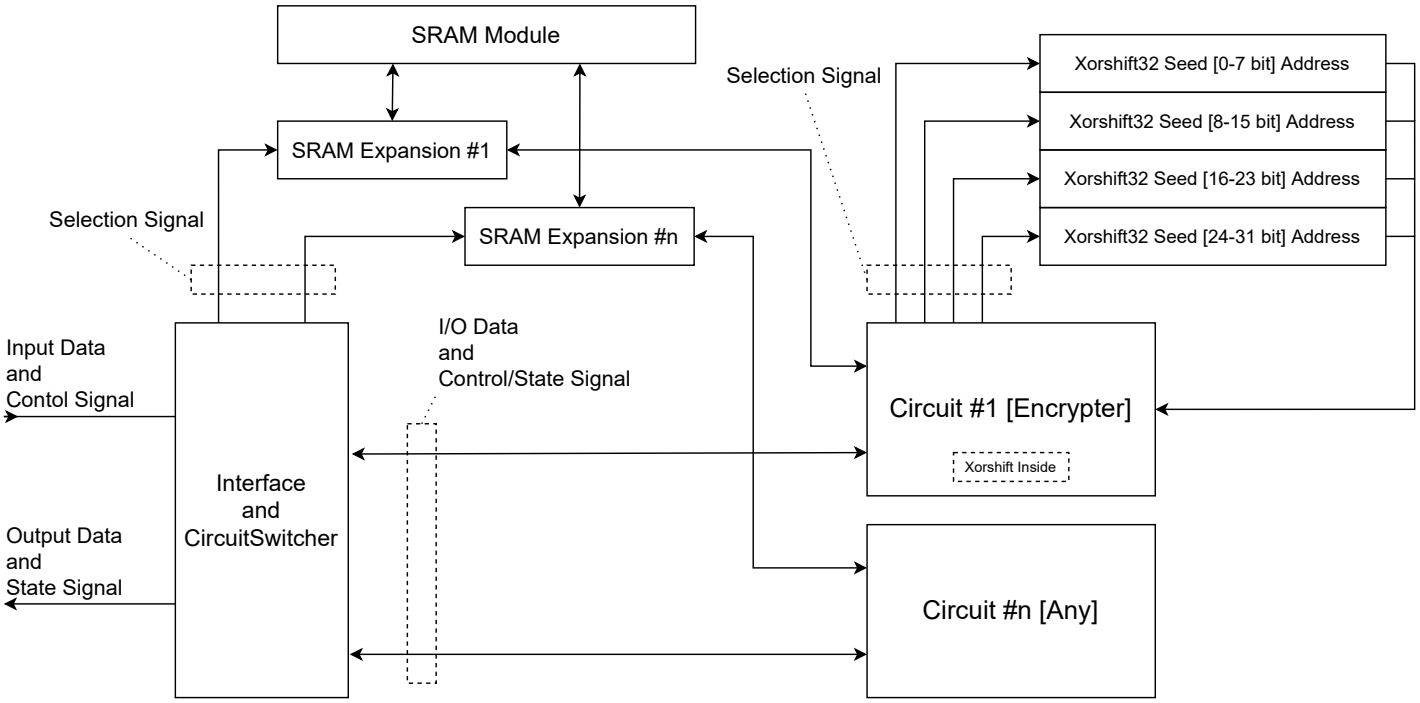

図 12　システム

3－1－1　制御モジュール

外部との入出力，接続する回路の制御を司るモジュールである．拡張性を考慮し，制御・回路選択・信号分配はそれぞれ独立させた．

本稿では回路選択モジュール・信号分配モジュールの説明を省略するが，動作の概略としては，制御モジュールの CS に HIGH 入力した状態で EXE を HIGH 入力すると，CS_DATA[0-7] がラッチされ，回路選択モジュールは CS_DATA[0-7] の状態に対応した信号を出力し，信号分配モジュールは回路選択モジュールから入力された信号に応じた回路を選択する．動きとしては，デマルチプレクサそのものである．

DATA は 1 byte のデータを入力できるため，回路選択モジュール・信号分配モジュールを対応させれば，理論上 256 の回路を接続することができる．しかし，実際に実装する場合はファンイン・ファンアウトをしっかりと考慮する必要がある．

RS-FF1 の R には，POR と発振防止を目的として遅延回路（RC 回路）を接続した．リセット条件に EXE の反転が必要な理由は，意図しない連続実行を防止するためである．

また，CMP 信号を RS-FF2 でラッチしている理由は，CMP の入力時間を定義しないためである．RS-FF2 で CMP がラッチされた時点で，接続回路に入力される EXE が LOW になり，リセット信号が正常に入力されたことを接続回路側で判定することができる．並んで，先述の RS-FF1 に接続された遅延回路によって制御モジュールがリセットされるまでに時間的猶予があるため，接続回路側は余裕をもって終了処理を行える．

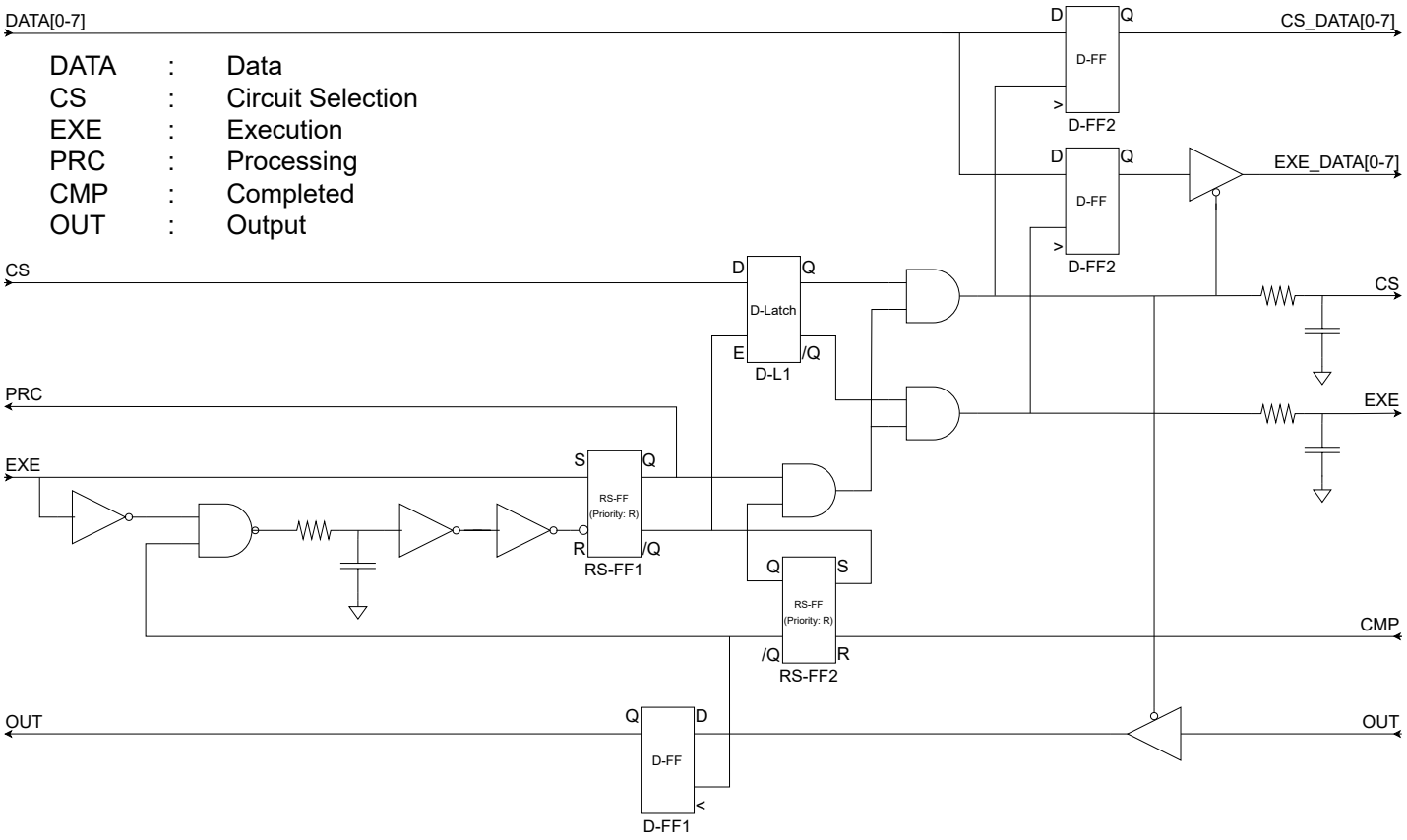

図 13　制御モジュール

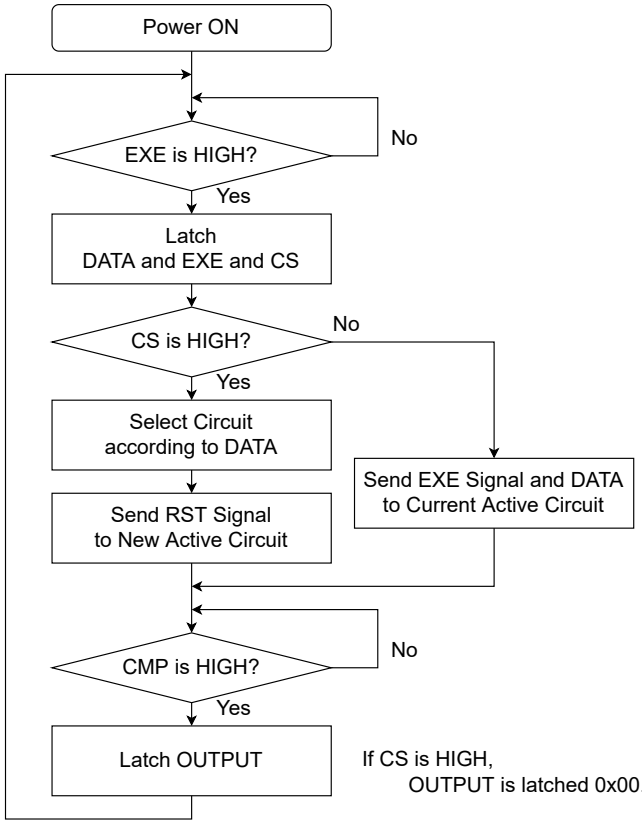

図 14　制御モジュールのフローチャート

3－1－2　SRAMモジュール

本システムでは M68AF127B を採用した．M68AF127B にはイネーブル端子が二つあり，柔軟な設計が可能である．本システムでは，イネーブル端子の一本をチップ切り替え，もう一本を書き込み開始信号線[*21] として利用する．また，共通の信号線で複数の回路からモジュールにアクセスできるよう，トライステートバッファと回路選択モジュールの信号を用いて，各回路のメモリモジュールへのアクセスを管理できる設計とした．

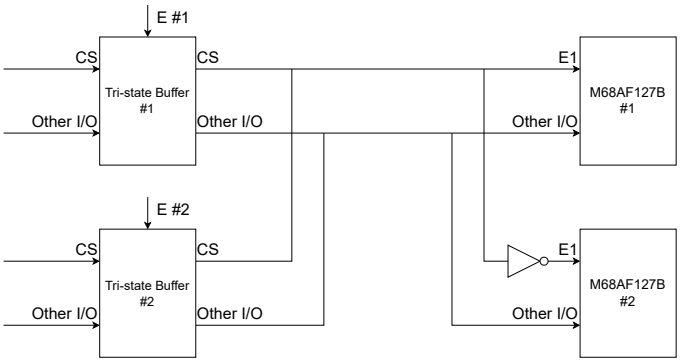

図 15　SRAM モジュール

3－1－3　暗号化モジュール

暗号処理を司るモジュールである．単体では動作せず，SRAM モジュール、Xorshift モジュール（後述）、リングカウンタモジュール（説明省略）と連携して動作する．

[*21] イネーブル信号が LOW から HIGH になったときに書き込みを開始する仕様である．（ただし，二つのイネーブル端子のうち一方がでパッシブとなっている場合を除く．）

３－１－３ー１　アルゴリズム

メモリの読み出し／書き出し・シフト演算のみで構成される非常なシンプルな設計とした．0x00-0xFF のデータをランダムに並べたものを一つの表として，並べた複数の表から暗号単位ごとに乱数生成アルゴリズムによって表を選択する．暗号化後のデータは 2 byte のアドレスのうち，上位 1 byte をテーブル番号，下位 1 byte を対象データとして，当該アドレスに存在するデータである．

Xorshift モジュールを独立させることによって，異なる乱数生成モジュールに変更することができる．また，アドレスの下位 1 byte に DATA をそのまま入力するため，間に加工回路を接続することでより柔軟な暗号処理を可能とした．

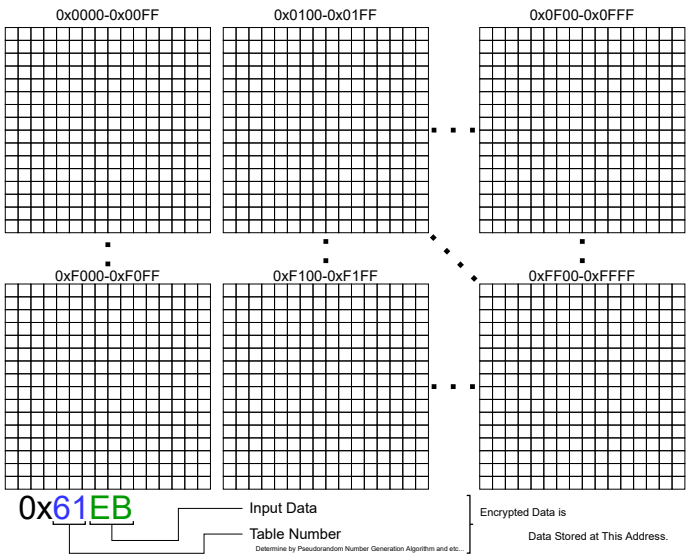

図 16　暗号化モジュールのアルゴリズムのイメージ

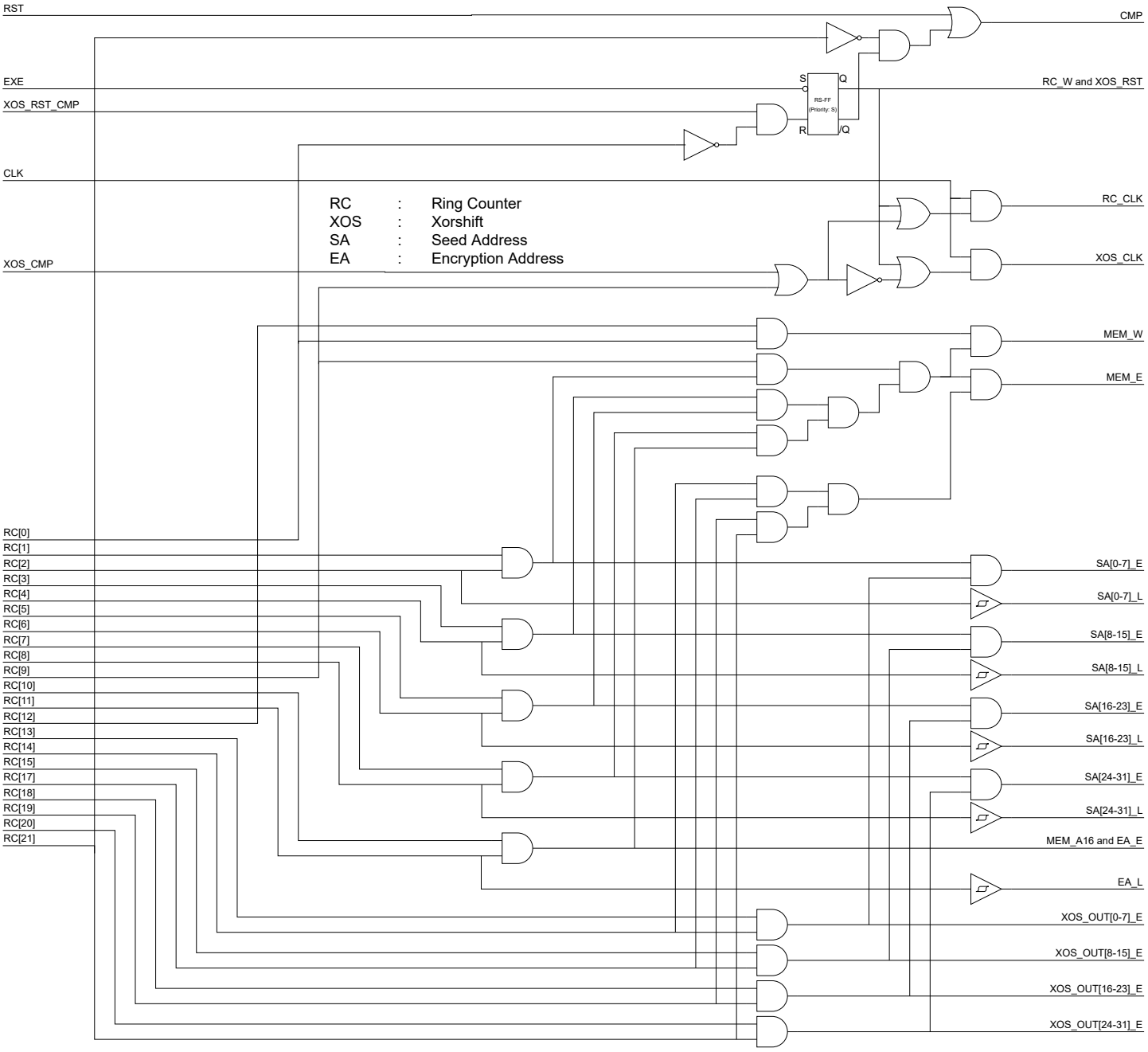

図 17　暗号化モジュール

3－1－5　疑似乱数生成モジュール

暗号化モジュールの表選択に用いる疑似乱数を生成するモジュールである．本システムでは Xorshift を採用した．Xorshift は疑似乱数生成アルゴリズムの一つであり，単純な処理で構成されているため，非常に軽量でありハードウェアで実装しやすい．

設計初期段階では，単純さと処理速度の面から，XOR 素子を直列で接続した回路を考えていたが，コストが想定より高かったため，トライステートバッファと D フリップフロップを用いたクロック駆動回路[*22] へと変更した．

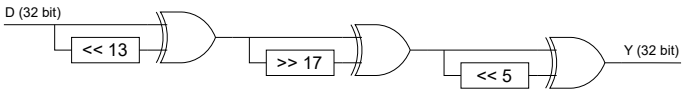

図 18　設計初期段階の Xorshift

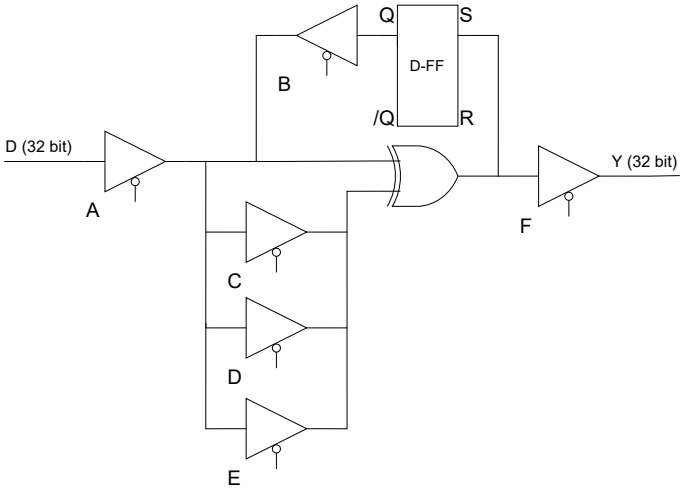

図 19　設計変更後の Xorshift

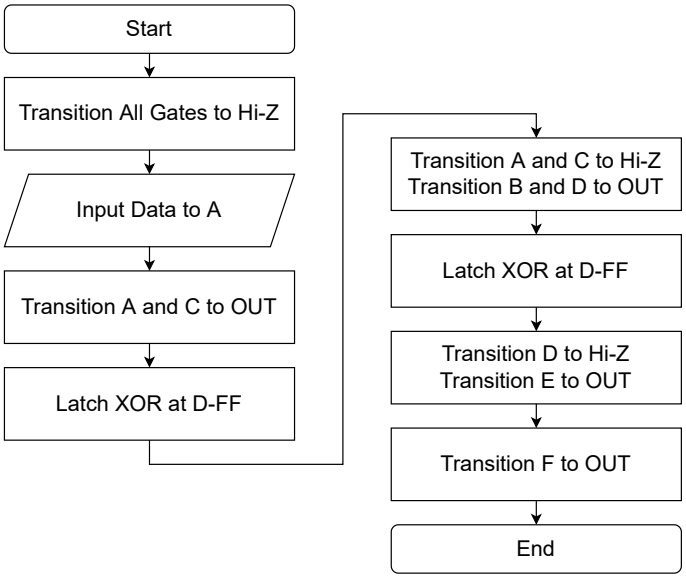

図 20　Xorshift（設計変更後）のフローチャート

3－2　速度検証

表 1 と同じ環境で速度を検証[*23] したところ，予想外に AES より低速であることを確認できた．

しかし，AES と RSA は.NET Framework [*24] 標準ライブラリを利用しており，Windows ネイティブのコードで実行されていることが考えられるため，完全に公平ではない恐れがあることは留意したい．

なお，共通鍵暗号の特性上，暗号化／復号に掛かる負荷はほぼ等しいため，暗号化速度のみの比較とする．また，本アルゴリズムは異なる二つのレギュレーションで計測した．表選択も含めて一つのアルゴリズムであるため，B は比較としては不適切ではあるが，B では AES より高速であることを確認できた．

A．　テーブル選択に剰余演算を使用
B．　テーブル選択を事前に処理

表 2　独自アルゴリズムと AES の暗号化速度比較

unit: ms

	Orig A [E]	Orig B [E]	AES [E]
#1	30	16	17
#2	30	15	18
#3	29	16	24
#4	29	16	18
#5	35	17	18
#6	31	15	19
#7	31	15	22
#8	29	17	19
#9	29	16	20
#10	29	15	19
Avg	30.2	15.8	19.4

Environment　:　Windows 11 21H2(22000.434)
.NET Framework 4.7.2
CPU　:　AMD Ryzen 9 5900HS 3.30 GHz
RAM　:　DDR4 16.0 GB 4266 MHz

3－3　懸念点

本アルゴリズムは非常にシンプルで軽量である一方，消費メモリや表選択アルゴリズム，その他の要素においていくつかの問題が散見する．

3－3－1　メモリ消費量

本アルゴリズムでは表が鍵となるため，メモリ消費量は表の数に比例する．RSA-2048 の鍵の長さが 256 byte（2048 bit）であるのに対して，本アルゴリズムでは一つの表で 256 byte の領域を消費する．暗号強度を上げるために複数の表が必要であるため，本アルゴリズムでは実用にあたり多量のメモリが必要となる．

3－3－2　変換表の質

本アルゴリズムはあくまで「表に則って変換する」だけのアルゴリズムであるため，暗号強度は表に依存する．重複した

[*22] XOR 素子を 64 個削減．
[*23] 表 1 の AES の記録は表 1 と同様．
[*24] .NET Framework と.NET Core は異なる．

表，変換前と変換後のデータが同じ表，変換前と変換後のデータの一致率が高い表などを用いると暗号強度は低下する．暗号強度をある程度担保するためには，表生成においてルールを設ける必要がある．

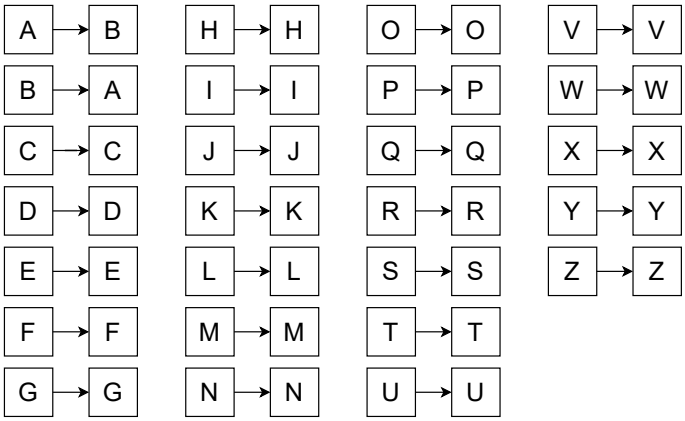

図 21　低質な変換表

3－3－3　疑似乱数の質

変換表の質に並んで疑似乱数生成アルゴリズムも非常に重要である．極端な偏りがあるアルゴリズムを採用した場合，表の使用頻度に影響してしまう．良い表を複数用いたとしても，使われなければ意味がない．また，特定の表の使用頻度が高いと頻度分析によって解読されてしまう可能性が上がり，暗号強度が下がってしまう．

図 22 から図 25 は適当に与えたシード値をもとに Xorshift (32 bit) を 10000 回試行し，8 bit 毎の 0 (0x00) から 255 (0xFF) の出現頻度をグラフ化したものである．同様に，図 26 から図 29 は線形合同法を用いて同じ条件[*25] で試行した．

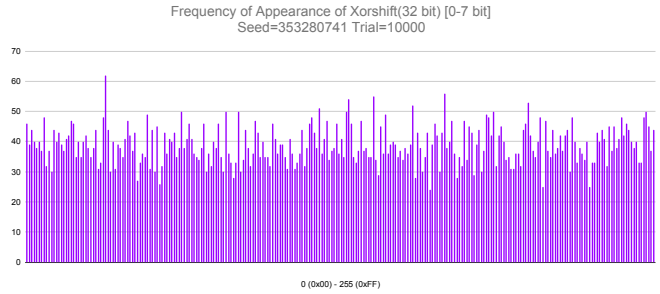

図 22　Xorshift32 [0-7 bit] における値の出現頻度

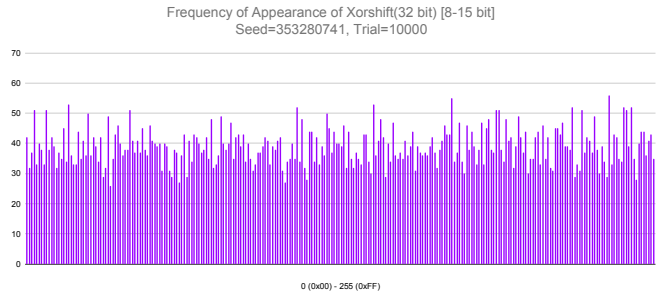

図 23　Xorshift32 [8-15 bit] における値の出現頻度

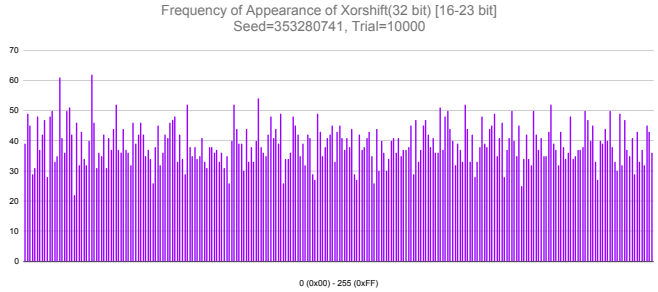

図 24　Xorshift32 [16-23 bit] における値の出現頻度

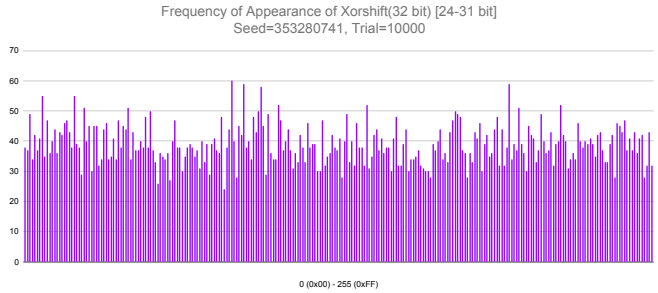

図 25　Xorshift32 [24-31 bit] における値の出現頻度

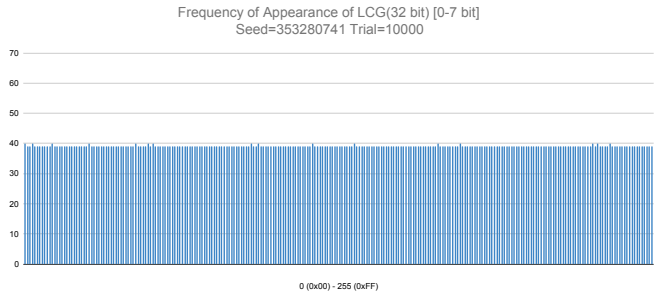

図 26　ＬＣＧ 32 [0-7 bit] における値の出現頻度

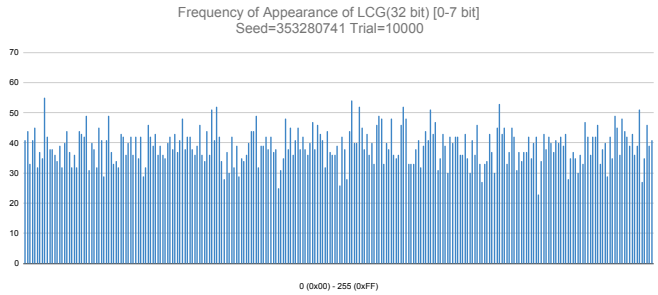

図 27　LCG32 [8-15 bit] における値の出現頻度

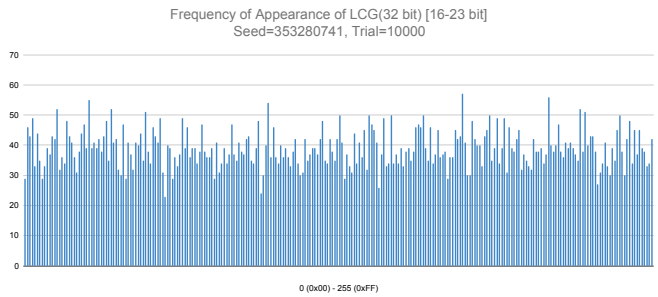

図 28　LCG32 [16-23 bit] における値の出現頻度

[*25] 定数 A と C の値は次の通りである．$A = 1664525, C = 1013904223$

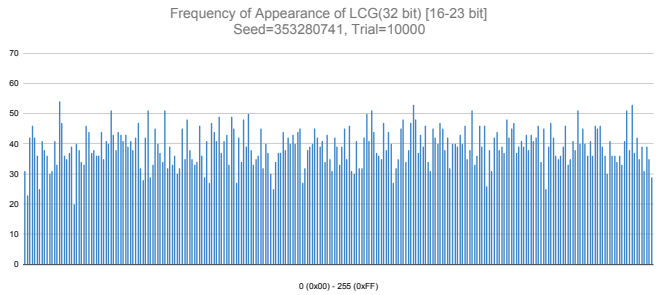

図 29　LCG32 [24-31 bit] における値の出現頻度

4　あとがき

本研究で実装したアルゴリズムは，古典暗号を僅かに拡張したに過ぎず実用は危険である．しかし，それ故に現代暗号と比較して非常に軽量である．だからこそ，工夫によって実用に耐えうる暗号強度を実現できるのではないかと考える．例えば，ダミーデータをランダムに散りばめて元のデータサイズを隠蔽，前の暗号化前データを次の Xorshift のシード値に組み込むといった工夫で暗号強度を高めることができる．

量子コンピュータの研究が進む中，現代暗号も安全が脅かされるとされている [8][9][10]．今後，更に複雑な暗号が考案・開発されることは必至であり，現在「現代暗号」と呼ばれる暗号が，「古代暗号」に分類される日も来るだろう．しかしながら，新時代のより強力な暗号技術が開発されたとしても，今までに培われた暗号技術は今後の技術発展で必要である．今後もセキュリティ及び暗号技術の動向を注視したい．

参考文献

[1] 総務省 - サイバーセキュリティタスクフォース（第 27 回）- 資料 27-1『サイバー攻撃の最近の動向等について』
[2] 電子情報通信学会 - 通信ソサイエティマガジン - 2020 年夏号 No.53『今だからこそ知りたい ブロックチェーンの基礎』
[3] 消費者庁 - インターネット消費者取引連絡会（第 42 回）- 資料 1 三菱 UFJ リサーチ&コンサルティング『キャリア決済を中心としたキャッシュレス決済の動向整理』
[4] 総務省 - 商務・サービスグループ キャッシュレス推進室 -『キャッシュレスの現状及び意義』
[5] 独立行政法人 情報処理推進機構 - 暗号技術 Q&A - 1．暗号の鍵とは何ですか？ https://www.ipa.go.jp/security/enc/qa.html
[6] 国立情報学研究所 - 市民講座 (2006) 第 3 回『現代暗号〜ネット社会の情報を守る暗号技術とは〜』
[7] 国立研究開発法人 情報通信研究機構 - NICT NEWS 2013 年 3 月号 No.426『暗号の安全性評価』
[8] 日本銀行金融研究所 -『量子暗号通信の仕組みと開発動向』
[9] 日本銀行金融研究所 -『量子コンピュータが共通鍵暗号の安全性に与える影響』
[10] 日本銀行金融研究所 -『量子コンピュータの解読に耐えうる「格子暗号」の最新動向』

全国専門学校電気電子教育研究会　加盟校

【会員校】

東北電子専門学校	980-0013	宮城県仙台市青葉区花京院 1-3-1
新潟工科専門学校	950-0932	新潟県新潟市中央区長潟 2-1-4
中央工学校	114-8543	東京都北区王子本町 1-26-17
専門学校東京テクニカルカレッジ	164-8787	東京都中野区東中野 4-2-3
日本電子専門学校	169-8522	東京都新宿区百人町 1-25-4
読売理工医療福祉専門学校	112-0002	東京都文京区小石川 1-1-1
日本工学院専門学校	144-8655	東京都大田区西蒲田 5-23-22
日本工学院八王子専門学校	192-0983	東京都八王子市片倉町 1404-1
名古屋工学院専門学校	456-0031	愛知県名古屋市熱田区神宮 4-7-21
阪神自動車航空鉄道専門学校	660-0893	兵庫県神戸市長田区林山町 27-1
九州電気専門学校	812-0018	福岡県福岡市博多区住吉 4-4-5
麻生情報ビジネス専門学校	812-0016	福岡市博多区博多駅南 2-12-32
熊本工業専門学校	861-8038	熊本県熊本市東区長嶺東 5-1−1

【賛助会員】

株式会社ビーフォーシー	168-0065	東京都杉並区浜田山 4-16-18

紀要 専電研 Vol.5

発行日：令和 4 年 11 月 29 日
発　行：全国専門学校電気電子教育研究会
発行者：船山世界
事務局：〒169-8522 東京都新宿区百人町 1-25-4
学校法人電子学園 日本電子専門学校 内
TEL 03-3369-9333(職員室)　E-Mail kookawa@jec.ac.jp