Computer Security Symposium 2017
23 - 25 October 2017

ネットワーク隔離された ARM CPU ベース PC からのデータ漏洩

山本 健太[†1]　廣瀬 幸[†2]　齊藤 泰一[†2]

概要：エアギャップネットワークは公衆ネットワークから完全に切り離されている．エアギャップネットワーク上のコンピュータから情報を抽出する手法は近年実証されつつある．Guri らは x86 CPU の特定の命令を呼び出し，CPU と RAM の間にあるメモリバスから電磁波を発生させ，B-ASK 変調された信号を GSM 周波数帯で送信することを可能にした．
本研究では ARM アーキテクチャ CPU を搭載したコンピュータで B-ASK 変調されたデータ信号を AM 周波数帯で送信する。我々は NEON 命令を使用し，ARM ベースのコンピュータから放射される電磁波によって B-ASK データを送信する技術を提案する．我々は，システムバスから電磁波を効果的に放射するために x86 CPU と異なるアプローチを採用している．

キーワード：エアギャップネットワーク，データ抽出，ARM，SIMD, NEON

Data Exfiltration from Air-Gapped computers based on ARM CPU

Kenta Yamamoto[†1]　Miyuki Hirose[†2]　Taiichi Saito[†2]

Abstract: Air-gapped network is a network isolated from public networks. The techniques of data exfiltration from air-gapped networks have been recently demonstrated. Guri et al., present a technique, called "GSMem", that can exfiltrate data from air-gapped computers over GSM frequencies, 850 MHz to 900 MHz. GSMem makes it possible to send data using radio wave leaked out from system-bus between CPU and RAM. They adopt a method that generates binary amplitude shift keying (B-ASK) data with x86 SIMD instructions.
We present a technique to use NEON instructions and transmit B-ASK modulated data by radio wave radiated from ARM based computer (i.g. Raspberry Pi 2 and 3). We observe it sends binary data using radio wave (about 1000kHz ~ 1700kHz) leaked out from system-bus between ARM CPU and RAM. We adopt new approache for ARM CPU to effectively radiate EM wave from the system-bus. Our program code can also run on Android machines based on ARM CPU (e.g. ASUS Zenpad 3S 10 and OnePlus 3).

Keywords: Air-gapped Network, Data exfiltration, ARM, SIMD, NEON

1. はじめに

エアギャプネットワークは，インターネット等の公のネットワークから物理的かつ論理的に独立したネットワークである．複数のコンピュータ，またはそれらが属するネットワークそのものを他のネットワークから切り離すことでセキュリティレベルを上げることができる．産業用制御システムや機密保持を担当するシステムはエアギャップネットワーク内で稼働することがある．これらのシステムに必須ではないネットワーク，すなわちパブリックネットワークからは物理的に切断されている．いくつかの組織では，Wi-Fi や Bluetooth の使用や，USB フラッシュドライブなどのリムーバブルストレージへのアクセスの制限により，データの漏洩を防ぐこともある．

†1 東京電機大学大学院
Graduate School of Tokyo Denki University
5 Senju, Asahi-cho, Adachi-ku, Tokyo 120-8551, JAPAN
†2 東京電機大学
Tokyo Denki University
5 Senju, Asahi-cho, Adachi-ku, Tokyo 120-8551, JAPAN

一方，Mordechai らによる Air-Hopper[1]は，ディスプレイケーブルから漏出する電磁波を利用してエアギャップを介したデータ送信手法を提案した．Hanspach[2]らは，コンピュータに接続されたスピーカーから超音波を出してエアギャップを介したデータ送信手法を提案した．このように，エアギャップネットワーク上で動作するコンピュータから情報を抽出する手法は近年実証されつつあるが，依然として研究されている．

本稿では，コンピュータの電磁放射の現象を利用したデータ送信の手法について言及する．

現代のコンピュータは様々な波長や強度で電磁放射(Electromagnetic Radiation)するような構造になっている．Electromagnetic Radiation (EMR)の現象を利用して，Wi-Fi や Bluetooth などの無線通信の使用が禁止されたエリアでエアギャップを介してデータを抽出する試みがある．

Ben-Gurion 大学の研究チームである Guri らは CPU と RAM の間にあるメモリバスから電磁波を発生させ，B-ASK 変調された信号を送信するマルウェア GSMem[3]を発表した．GSMem は x86 アーキテクチャで動作するコ

ンピュータにおいて，メモリバスの帯域を占有するようなCPU 命令を発行しメモリバスからの電磁波の漏出を促す．

本研究は，ARM アーキテクチャの CPU を搭載したコンピュータで，B-ASK 変調されたデータ信号を AM 周波数帯(1,000kHz – 1,600kHz)で送信する．

変調された信号を送信するためにはメモリバスに効率的にアクセスできる CPU 命令とアルゴリズムが必要となる．GSMem では効率的にメモリバスへのアクセスを行うために CPU キャッシュを介さずにデータ操作が可能な x86 SIMD 命令を採用している．一方，ARM アーキテクチャの CPU は明示的に CPU キャッシュを回避する命令は用意されていない．我々は効率的にメモリバスの帯域を占有するための ARM CPU 命令として NEON 命令を採用し，CPU キャッシュをできるだけ使用させないためのアルゴリズムを提案する．

本アルゴリズムを採用したプログラムを実行し，ARM コンピュータから漏出した電磁波をスペクトラムアナライザで観測する．

2. 技術背景

本稿では，セルラーGSM および AM ラジオ周波数帯において B-ASK 変調した信号を送信する技術について言及する．このセクションでは，先行研究及び我々の提案手法をより理解しやすくするための基礎技術概要を提供する．

2.1 セルラーネットワーク

セルラーネットワークには世代があり第二世代(2G)，第三世代(3G)，第四世代(4G)がある．それぞれの世代に独自のプロトコルやアーキテクチャが実装されている．2G, 3G, 4G の通信規格はそれぞれ GSM, UMTS, LTE と呼ばれる．各規格とも利用する周波数の組み合わせを定めたバンドと呼ばれる周波数帯域が存在する．

先行研究で利用する GSM のバンドは 850MHz から 900MHz となる．

2.2 B-ASK 変調

B-ASK 変調はデジタル信号を送信する変調技術に一つである．送信するデータのビット列に対応させて振幅を変化させる．搬送波の周波数や位相は固定され，振幅のみが変調に利用される．送信データのビットが”1”の時，搬送波の振幅は大きくなり，送信データのビットが”0”の時は，搬送波の振幅は小さくなる．

図 1 は送信ビット列の振幅表現であり，図 2 は送信する信号(上)と B-ASK 変調によって変調された信号(下)のイメージである．ビット列の振幅表現と変調信号の振幅が対応していることがわかる．

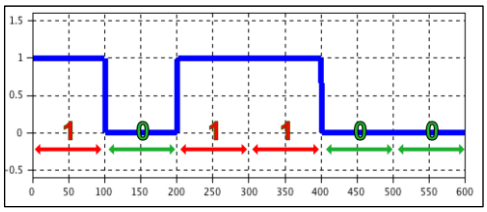

図 1. B-ASK による送信データのビット表現

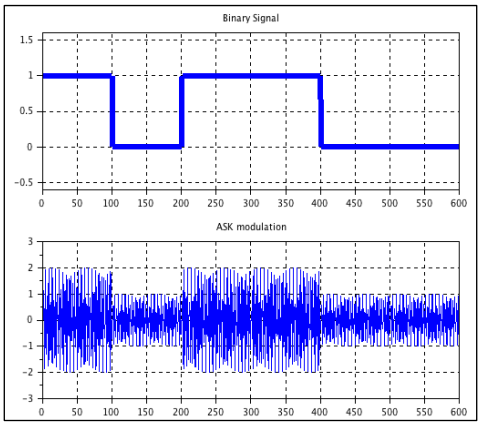

図 2. 送信する信号(上)と変調された信号(下)

3. 先行研究

3.1 GSMem

Guri らが提案した GSMem は，x86 アーキテクチャに基づく CPU を搭載したコンピュータからエアギャップを介してデータを抽出できるマルウェアである．CPU がメモリバスを介して RAM へアクセスするとき，GSM 周波数帯をカバーする電磁波が漏出する現象を利用する．彼らは GSMem に感染させたコンピュータを送信機に見立てて，B-ASK 変調した信号を GSM 周波数帯で送信する．B-ASK 変調はデータをビット列に変換し，”1”と”0”のビットに応じて振幅を変化させることでデータを送る方式である．

GSMem では，RAM へのアクセスを行っている場合は“1”，アクセスを行わない場合は “0” と表現する．RAM アクセスを行うと特定周波数においてパワーが増加することから，”1”と”0”の表現に振幅の差が生じる．

受信機は，受信した電磁波の振幅によって信号の復調を試みる．

本手法は RAM へのアクセスが電磁波漏出を促す．そのため CPU が持つデータキャッシュを使用せずに，直接 RAM とデータをやり取りするようなアルゴリズムであることが望ましい．彼らはより効率的に RAM へのアクセスを発生させるため，x86 アーキテクチャ CPU で利用可能な SSE2 命令「MOVNTDQ」を使用した．MOVNTDQ 命令は，CPU キャッシュを介さずにデータをメモリアドレスにストアする命令である．C++向けライブラリの関数名は _mm_stream_si128 となる．

図 3 は GSMem が任意のデータを B-ASK 変調して送信す

るコンセプトコードである．
送信するデータ，bit_index のビットを読み取り，それが 1 であった場合は MOVNTDQ 命令を tx_time で決定した時間 T だけ実行し続ける．読み取ったビットが 0 であった場合は tx_time で決定した時間 T だけプログラムをスリープさせる．

```
1: buffer ←ALIGNED_ALLOCATE(16,4096)
2: tx_time ← 500000
3: for bit_index ← 0 to 32 do
4:     if (data[bit_index] = 1) then
5:         start_time ←CURRENT_TIME()
6:         while (tx_time > CURRENT_TIME() - start_time) do
7:             buffer_ptr ← buffer
8:             for i ← 0 to buffer_size do
9:                 SIMDNTMOV(buffer_ptr, 128bit_register)
10:                buffer_ptr ← buffer_ptr + 16
11:            end for
12:        end while
13:    else
14:        SLEEP(tx_time)
15:    end if
16: end for
```

図 3. GSMem によるデータ送信アルゴリズム[a]

3.2 System-bus-radio

fulldecent は GSMem のアルゴリズムを用いて音声波を生成し，AM 周波数帯で信号を送信するプログラム System-bus-radio [4]を発表した．System-bus-radio は音声波の周波数fから$period = 2/\lambda$を求める．1 と 0 のデータを$period$時間毎に交互に生成することで周波数fの音声波を得ることができる．図 4 は音声周波数fから$period$を求めて音声波を生成しているイメージである．

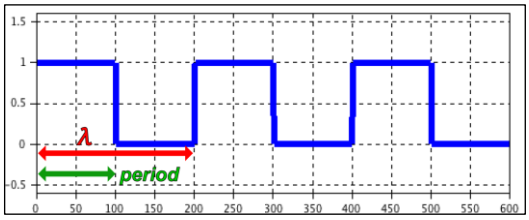

図 4. System-bus-radio による音声波生成

信号の受信機として AM ラジオが利用できる．変調された信号は振幅変調であるから，信号を受信した AM ラジオは既存の回路で復調可能である．ラジオのスピーカーからは，System-bus-radio が生成した音声が出力される．
実際のコードには _mm_stream_si128 が使用されるが，バリエーションとして SIMD を採用しないものもある．

[a]. Guri, M., Kachlon, A., Hasson, O., Kedma, G., Mirsky, Y. and Elovici, Y., (2015). GSMem: data exfiltration from air-gapped computers over GSM frequencies. In 24th USENIX Security Symposium (USENIX Security 15) (854).

4. 提案手法

我々は ARM アーキテクチャの CPU を搭載したコンピュータで B-ASK 変調された信号を送信するアルゴリズムを提案する．GSMem では, Guri らは, x86 アーキテクチャで使用できる SIMD 命令 MOVNTDQ を採用した．彼らはこの命令が効率的かつ大量にメモリバスにデータを流すことができることを示した．しかしながら，x86 とマイクロアーキテクチャが異なる ARM ではこれを利用することができない．我々は，効率的にメモリバスにアクセスするための CPU 命令の選定とアルゴリズムを提案する．

4.1 ARM アーキテクチャと問題点

ARM アーキテクチャの CPU を搭載したコンピュータの多くは x86 コンピュータ同様 DDR メモリを搭載していることがあるが，デバイスの省電力化のために LPDDR が採用されることがある．LPDDR メモリは 1 チャネル毎に 32 bit のデータ幅をもつ．DDR はマルチチャネルに対応しており，その場合のデータ幅は 32 bit の倍数になる．ARM 本来の 32 bit/64 bit ARM アーキテクチャでメモリバスの帯域を使い切ることは可能だと思われるが，マルチチャネル環境下や，割り込み処理等で常に効率的なメモリバスの帯域占有は困難になる場合がある．

我々は，ARM コンピュータでより確実にメモリバスを占有できる命令セットとして NEON を採用した．NEON は ARM アーキテクチャにおける SIMD 命令セットである．SIMD(Single Instruction Multiple Data) は 1 度の命令で複数のデータを扱う事ができる．NEON は ARMv7 以降で使用可能で，64 bit および 128 bit の命令セットである．1 つの命令で最大 128 bit のデータを取り扱うことができるため，メモリバスの帯域占有に向いている．

NEON は SIMD 命令であり，ARM 従来の命令セットとは別に各種オペレーションが存在する．x86 の SIMD には CPU キャッシュを使用せずにデータ操作を行う命令が予め用意されている．しかし，NEON の命令セットには，明示的に CPU キャッシュを使用せずにデータ操作を行う命令が存在しない．ARM CPU には命令キャッシュとデータキャッシュが存在し，それぞれのキャッシュを無効にする CPU モードが存在する．しかし，モードの切り替えには CPU の特権モードが必要となるため，アルゴリズムを特殊な条件下でしか動作させることができない．CPU キャッシュの無効化は現実的に使用できる方法ではない．

そのため，我々は CPU キャッシュが有効であっても RAM へのアクセスを発生させるようなアルゴリズムと，NEON 命令セットの中からアルゴリズムの実行に最適な命令を選択することが必要になる．次のセクションで CPU のデータキャッシュを回避するためのアルゴリズムを提案する．

4.2 アルゴリズム

CPU のデータキャッシュを回避するためには，CPU がデータキャッシュを参照した際にヒットしなければよい．

我々は, CPU データキャッシュを回避するためのアルゴリズムとして次のコンセプトコードを提案する．(図 5)
メモリ上からデータをロードする単純なアルゴリズムだが，ロード命令を実行するたびに，ロードするアドレスを変化させることで，毎回異なるデータを操作することになり，CPU データキャッシュにヒットしにくくなると思われる．

ロードする際に実行される SIMD 命令として，我々は，一度の命令の発行でより複数のデータがやり取りされる“VLD1.32”を採用した．C++言語ライブラリでの関数名は vld1q_s32 である．VLD1.32 は RAM 上に保存された 32bit 幅のデータ 4 つを単一のベクタとしてレジスタにロードする命令である．一度に 128 bit のデータをロードできる．

```
1 :   ptr = (int32_t *)malloc(size)
2 :   for(int i=0; i<=n; i++):
3 :      ptr[i] = i;
4 :
5 :   data_bits[] = {1, 0, 1, 0,}
6 :   period = 500000                              // nanoseconds
7 :   for data in data_bit :
8 :      if ( data == 1):
9 :         i = 0
10:         start = now()
11:         while (period > now() - start):
12:         int32 var[4]={*(ptr+i),*(ptr+i+1),*(ptr+i+2),*(ptr+i+3) }
13:            va = vld1q_s32(var)          // VLD1.32
14:            i++
15:            if(i==limit) i=0
16:      if ( data == 0):
17:         sleep(period)
```

図 5. CPU データキャッシュを回避するアルゴリズム

本アルゴリズムは大きく分けてメモリの確保と，データロード部分に別れる．RAM アクセスを発生させる場合はメモリに確保したデータをロードし続ける．

まず，1 行目ではメモリの確保を行う．32 bit 単位のデータ配列を *size* バイト確保する．確保するメモリはすくなくとも L1 キャッシュより大きなサイズを確保するのが有効なようだ．例えば ARM Cortex-A シリーズの一つである Cortex-A53 アーキテクチャは L1 キャッシュとして最大 64KiB 搭載可能であるから, 65536 バイト以上の割り当てが必要となる．

2-3 行目では割り当てたメモリに 32 bit データを初期化する．この時，全て同じ値で初期化すると，CPU のデータキャッシュにストアされ，RAM へのアクセスが発生しない可能性があるため，すべてユニークかつランダムなデータを追加した．図 3 では配列のインデックス値と同期した整数を格納している．

data_bits は送信するデータバイナリの配列とする. period は 500 ミリ秒を表している．GSMem によると period を小さくするとより高いビットレートが得られるが，エラーレートが増加する．GSMem のアルゴリズムに倣って同様の 500 ミリ秒を設定した．

7 行目の外側の for 文は data_bits の要素数だけループする．data_bits の要素が 1 であるとき，メモリ操作をして電磁波を発生させる．12 - 14 行目は vld1q_s32 (VLD1.32)によるロード命令を実行している．period 時間，すなわち 500 ミリ秒の間，ロード命令 vld1q_s32 を実行し続ける．

12 行目では vld1q_s32 を実行するため, ロードするメモリアドレスを指定している．vld1q_s32 命令を使用して指定された 4 つのアドレスから単一ベクタとしてレジスタにデータをロードしている．

このとき，メモリアドレスの指定方法は 11 行目ループ内が実行される毎に 4 バイトずつずれていく． すなわち vld1q_s32 によってロードするベクタは実行される事に異なるものになる．

図 4 は, vld1q_s32 初回にロードするメモリデータである．1 つのブロックは 32 bit の整数データである．vld1q_s32 は配列上の 1 から 4 ブロックを一度にロードする．

図 5 は，2 回目のロード時の実行イメージである．ロードする配列の要素は，2 から 5 番目のブロックから読み出される．3 回目以降も，1 ブロックずつずらしてレジスタに読み込まれる．読み出した 128 bit レジスタは毎回異なるデータとなるので CPU のデータキャッシュにヒットしにくくなる．

なお, 15 行目では確保した配列の最後までロードし終えた場合の処理である．この場合は，配列のインデックス指定変数”i”を 0 にリセットし，配列の先頭要素からロードし直す．

最後に，16 行目では，data_bits の要素が 0 であるときの処理である．アルゴリズムを period 時間スリープする．

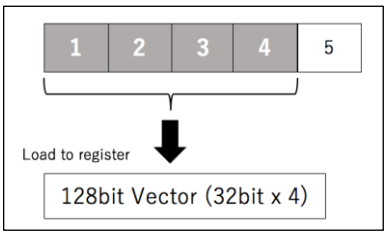

図 6. vld1q_s32 初回実行イメージ

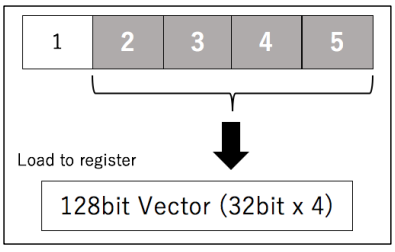

図 7. vld1q_s32 2 回目の実行イメージ

5. 周波数測定

本稿で提案したアルゴリズムを元に，データ送信プログラムを作成した[b]．本プログラムは fulldecent による System-bus-radio[4]に提案アルゴリズムを適用した改良プログラムである．ARM CPU を搭載したコンピュータでプログラムを実行した際の周波数測定の結果を示す．測定デバイスは，Raspberry Pi 3 および，ASUS Zenpad 3S 10 Z500M である．

Raspberry Pi 3 は ARM アーキテクチャのシリーズに一つである，Broadcom 社製 BCM2837 を搭載する．BCM2837 は，ARM Cortex-A53 CPU コアを 4 つ搭載する．LPDDR2 の容量 1GB の RAM を搭載する[5]．OS は Raspbian Linux を採用した．

ASUS Zenpad 3S 10 Z500M は MediaTek 製 MT8176 を搭載する．MT8176 は内部的に ARM Cortex-A72 を 2 コア，Cortex-A53 を 4 コア搭載する．LPDDR3 容量 4GB の RAM を搭載する．RAM はデュアルチャネルで接続される[6]．OS は Android 7.0 となる．

実験は電波暗室で行った．

5.1 実験セットアップ

図 8 に周波数測定の実験セットアップを示す．測定にはアンテナはモノポールアンテナを使用し，スペクトルアナライザにより測定した．DUT（被試験デバイス；device unser test）はタブレット端末 ASUS Zenpad 3S 10 Z500M および Raspberry Pi 3 である．

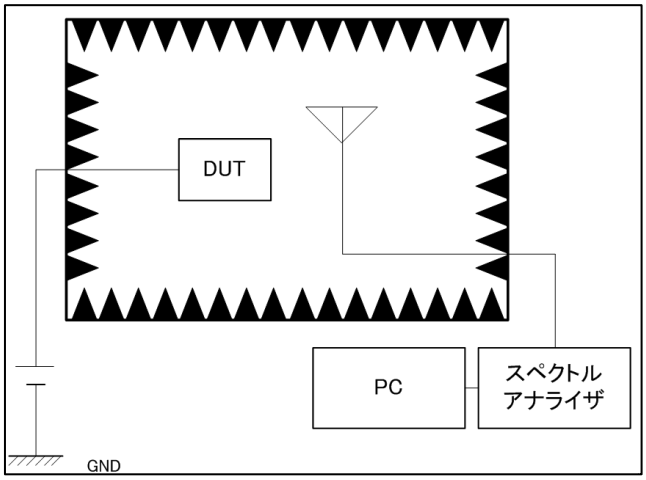

図 8　実験セットアップ

5.2 測定結果

ASUS Zenpad 3S 10 Z500M および Raspberry pi 3 の測定結果を，それぞれ図 9 および 10 に示す．プログラム実行時は”on”，実行していない状態では”off”とする．on 状態では，off 状態で発生していない周波数が確認できる．

測定結果より PC では 1.2 – 1.4 MHz，Raspberry pi で 1.5 MHz 周辺の周波数が観測でき，およそ 4 dB 前後の信号強度であることを確認した．

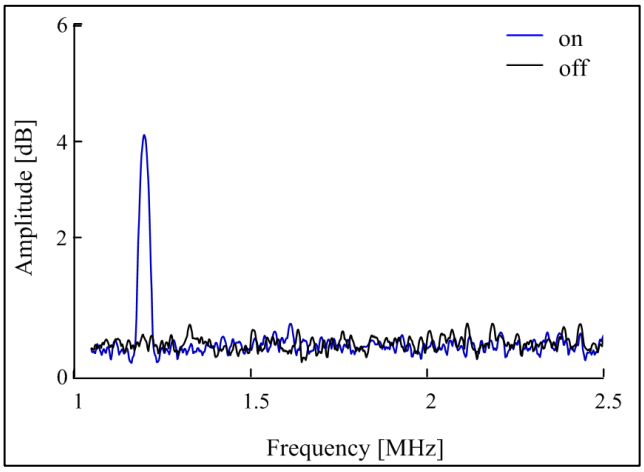

図 9　測定周波数スペクトルの一例
（ASUS Zenpad 3S 10　Z500M）．on はプログラム実行状態．off はプログラム実行していない状態．

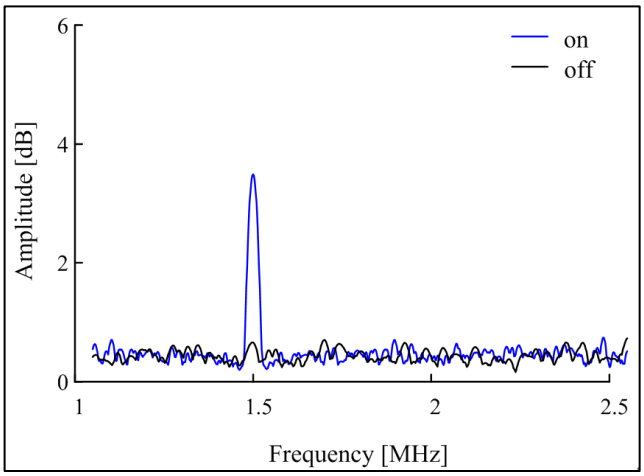

図 10　測定周波数スペクトルの一例
（Raspberry pi 3）．on はプログラム実行状態．off はプログラム実行していない状態．

6. まとめ

本稿では，ARM アーキテクチャに基づく CPU を搭載したコンピュータにおいて，コンピュータのメモリバスから発生する電磁放射(EMR)の現象を利用して B-ASK 変調されたデジタルデータを送信するアルゴリズムを提案，および測定実験をした．EMR を効果的に発生させるためにはメモリバスに連続で大量のデータを流す必要があり，先行研究では x86 CPU で明示的に CPU のキャッシュを使用せずにメモリとのデータをやり取りする命令を採用した．ARM アーキテクチャの SIMD には明示的に CPU キャッシュを回避する命令がない．我々は，データをロードする SIMD 命令(VLD1.32)を採用した．VLD1.32 命令を実行す

[b] 山本健太(gorillanet). (2017) System-bus-radio-neon, https://github.com/gorillanet/system-bus-radio-neon/ (参照 2017-08-25)

る毎に, ロードするデータを異なるものにすることで CPU データキャッシュを回避するようなアルゴリズムを提案した.

周波数測定では，提案したアルゴリズムを用いて周波数特性を測定した．提案アルゴリズムを実装したプログラムは，図 9 および図 10 より，うまく EMR を放出できていることが確認できた．実験に使用した 2 種類の異なるデバイスは，それぞれ異なる周波数で信号を確認できる．
この周波数はプログラム上で決定できるものではなく，基盤設計・基盤構成にから決定されるものと思われる．筐体設計や素材により信号強度も異なる場合がある．今後の課題として，放出された電磁波の周波数特定やプログラム上からある程度周波数を決定するアルゴリズムの考案などがあげられる．

更に, 今後の課題として, ARM CPU を搭載した Android デバイスにおいての詳細な動作を確認する．Android は携帯電話・タブレット OS として高いシェアを獲得しており, より多くのデバイスを攻撃対象にすることが期待できる．Android に搭載される SoC (System-on-a-Chip)は多くが ARM アーキテクチャだが，それらはカスタマイズされた ARM CPU であることがある．ARM 社以外のカスタマイズされた CPU においての EMR 放射特性を調査することが課題となる．
また，Android OS 独自の省電力機能により，SIMD 命令が連続して発行できない現象を確認済みである．Android OS 上で安定して SIMD 命令を発行するためのアルゴリズムの考案も今後の課題となる．

参考文献

[1] G. Mordechai, G. Kedma, A. Kachlon and Y. Elovici, "AirHopper: Bridging the air-gap between isolated networks and mobile phones using radio frequencies," in Malicious and Unwanted Software: The Americas (MALWARE), 2014 9th International Conference on, IEEE, 2014, pp. 58-67.

[2] M. Hanspach and M. Goetz, "On Covert Acoustical Mesh Networks in Air.," Journal of Communications, **vol. 8**, 2013.

[3] Guri, M., Kachlon, A., Hasson, O., Kedma, G., Mirsky, Y. and Elovici, Y., (2015). GSMem: data exfiltration from air-gapped computers over GSM frequencies. In 24th USENIX Security Symposium (USENIX Security 15) (pp. 849-864).

[4] fulldecent. (2016) System-bus-radio, https://github.com/fulldecent/system-bus-radio/ (参照 2017-08-25)

[5] RASPBERRY PI FOUNDATION (2017), Raspberry Pi 3 Model B - Raspberry Pi, https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ (参照 2017-08-25)

[6] MediaTek.(2016), MediaTek MT8176 for Tablets | MediaTek , https://www.mediatek.com/products/tablets/mt8176 (参照 2017-08-27)