Computer Security Symposium 2020
26 - 29 October 2020

OpenSSLに対するRAMBleed攻撃

冨田 千尋[1,a)]　瀧田 愼[2]　廣友 雅徳[3]　白石 善明[1]　森井 昌克[1]

概要：DRAMの行に繰り返しアクセスすることでその近隣の行内でビット反転を誘発する，Rowhammerと呼ばれる攻撃が存在する．Rowhammerを利用して，アクセス権限のない秘密情報を読み取るRAMBleedと呼ばれるサイドチャネル攻撃が提案され，OpenSSHで利用される秘密情報を回復できることが指摘されている．本稿では，OpenSSLによるSSL/TLS通信が実装されたWebサーバに対してRAMBleedを行い，サーバの秘密鍵の回復を試みた．OpenSSLでは，RAMBleedを実行する前に，秘密情報が物理メモリ上に割り当てられており，従来の方法ではTarget Pageに秘密鍵を誘導することができない．そこで，物理メモリ上のデータを強制的にスワップ領域に移動させ，物理メモリ上に再度割り当てさせる手法を提案し，秘密鍵をTarget Pageに誘導できる可能性を示した．

キーワード：rowhammer, rambleed, openssl, 秘密鍵, サイドチャネル攻撃

RAMBleed Attack against OpenSSL

Chihiro Tomita[1,a)]　Makoto Takita[2]　Masanori Hirotomo[3]　Yoshiaki Shiraishi[1]
Masakatu Morii[1]

Abstract: There is an attack called Rowhammer on the DRAM of the computer. A side-channel attack called RAMBleed, which reads secret information without access authority using Rowhammer, has been proposed. It is pointed out that RAMBLeed can recover secret information used in OpenSSH. In this paper, we attempt to recover the private key of OpenSSL by attacking the server with RAMBleed. In OpenSSL, the secret information is allocated on the physical memory before RAMBleed is executed. Then, the secret key cannot be guided to the target page by the conventional method. In this paper, we propose a method to move a secret key on physical memory to swap area and reassign it on physical memory. Also, we show the possibility of inducing the secret key to a target page.

Keywords: Rowhammer, RAMBleed, openssl, secret key, side-channel attack

1. はじめに

計算機やスマートフォンには一般的にDRAMと呼ばれる半導体メモリが使用されている．半導体の微細化，高密度化が進むとともに，大容量のDRAMが作られるようになり計算機の性能が向上している．その一方で，高密度化によりDRAM内のビット同士の物理的距離が近づいた結果，あるビットへのアクセスが別のビットに影響を与え，意図しないビット反転が生じることが指摘されている．この現象はRowhammer[1]と呼ばれ，それを悪用した権限昇格攻撃[2]，故障利用攻撃[3]，DoS攻撃[4]などが提案され，コンピュータシステムのセキュリティを脅かす重大な問題となっている．そのため，Rowhammerに関する様々な攻撃の評価やその対策について十分に議論する必要がある．

Rowhammerはソフトウェアを用いて容易に実行でき，DRAMの特定のページにアクセスを繰り返すことより，その近隣のページでビット反転を引き起こすことができる．

[1] 神戸大学大学院工学研究科
Graduate School of Engineering, Kobe University
[2] 兵庫県立大学社会情報科学部
School of Social Information Science, University of Hyogo
[3] 佐賀大学理工学部
Faculty of Science and Engineering, Saga University
[a)] tomita@stu.kobe-u.ac.jp

Rowhammer を悪用すると，ビット反転によってメモリ上の値を書き換えて，データの完全性を阻害する事ができる．Kwong らが 2019 年に Rowhammer に基づく攻撃として，RAMBleed と呼ばれるサイドチャネル攻撃を提案した [5]．RAMBleed は，意図的に引き起こした Rowhammer によるビット反転を観察することにより，アクセス権限のない秘密情報の一部を高い精度で回復する攻撃である．これにより，openSSH で利用される秘密鍵の情報を取得できることが指摘されている．RAMBleed を実行するためには，攻撃者のプロセスの中で攻撃対象のプロセスを制御し，回復したい情報を DRAM 上の意図したページに誘導する必要がある．実働している計算機では様々なプロセスが動作しているため，この誘導は必ずしも成功するとは限らない．また，従来の RAMBleed では数ビットの情報を回復するために，それが含まれる情報を 2 箇所に誘導する必要がある．

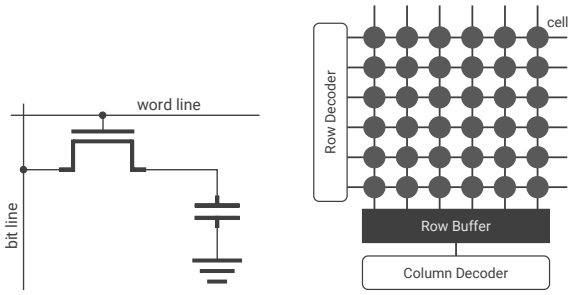
筆者らは，従来の RAMBleed からメモリ領域の扱い方を改善することで，秘密情報の誘導を 1 箇所のみに行う single-sided RAMBleed を提案している [6]．これにより，RAMBleed を実行する際の課題である秘密情報の誘導の回数を削減することが可能となった．また，従来の RAMBleed と同等である約 94%の確率で情報を回復することが可能であることを実際に計算機を利用した実験により示している．

本稿では，OpenSSL を用いて実装された SSL/TLS サーバで利用される秘密鍵の情報の回復を試みる．RAMBleed により秘密情報を回復するために，その情報を物理メモリ上のビット反転が生じやすい位置に誘導する必要がある．Kwong らはこれを，Linux のメモリ割り当てアルゴリズムである Buddy Allocator を利用して，秘密情報が物理メモリ上に配置されるタイミングを測ることにより実現した．OpenSSL の動作を観察したところ，OpenSSL の秘密情報は RAMBleed を実行する前に物理メモリ上に割り当てられており，その情報を利用してプロセスが動作することが分かった．このとき，Kwong らの誘導方法では，秘密情報を新たに意図した位置に割り当てることができない．そこで本研究では，攻撃対象のプロセスが保有する物理メモリ上のデータを強制的に全てスワップ領域に退避させることで，秘密情報へのアクセス時に物理メモリの割り当てを発生させる手法を提案する．RAMBleed を実行する前に，物理メモリ上に秘密情報が割り当てられていることを想定した実験を行い，提案手法により意図した位置に秘密情報を割り当てできることを確認した．

2. DRAM

2.1 DRAM の構造

DRAM(Dynamic Random Access Memory) は，図 1(a) のようにキャパシタとトランジスタから構成されるセルで 1 ビットの情報を保持する．充電された状態で 1，充電されていない状態で 0 を表すセルを true セルと呼ぶ．逆に，充電された状態で 0，充電されていない状態で 1 を表すセルを anti セルと呼ぶ．本稿では，簡単のために true セルのみを考慮して説明する．



(a) セル　(b) Bank

図 1　DRAM の構造

セルは図 1(b) のような行列の形に並べて配置されており，各行はワード線で，各列はビット線で接続されている．一定数の行をまとめて Bank と呼び，複数の Bank から DRAM チップが構成される．さらに，複数の DRAM チップをまとめて Rank と呼び，メモリモジュールである DIMM の片面に実装される．

DRAM 内でのデータの扱いは，8KiB から成る行単位でなされる．次のようにしてデータへアクセスする．目的のデータが含まれる行のワード線の電圧を高くすることで，キャパシタをビット線に接続される（行をアクティブにする）．これによって，1 行分のデータを図 1(b) に示す行バッファへ転送される．その後，CPU は行バッファ内にあるデータから目的の列にアクセスして，データの読み取り/書き込みを行う．

DRAM セルのキャパシタに貯められている電荷は，時間の経過とともに徐々に失われていく．これによるデータの損失を防ぐために定期的にキャパシタを充電し直す必要があり，この動作をリフレッシュをいう．この間隔をリフレッシュサイクルといい，DDR3 と DDR4 で通常は 64ms である [7]．

2.2 Linux buddy allocator

Linux では，システムに搭載されているメモリを，ページと呼ばれる領域（一般的には 4KiB）を最小単位として管理する．buddy allocator というアルゴリズムを使い，このページの割り当てや解放を行う．カーネルは，メモリの全ての空き領域を，物理的に連続するページごとに一つのブロックとしてオーダー 0 からオーダー 10 に分類する．オーダー n は 2^n ページのブロックで構成され，各オーダーでメモリブロックは，スタックのような first-in-last-out(FILO) のデータ構造をしている．

プロセスからメモリ割り当ての要求があった場合，buddy allocator は必要な領域に対して最小のブロックを割り当て

る．ただし，ユーザ空間からはオーダー 0 の要求のみが可能となっている．ユーザ空間からメモリ割り当てを要求されると，サイズに関わらずオーダー 0 にあるブロックから順に割り当てる．

3. RAMBleed

3.1 Rowhammer

Rowhammer は DRAM の複数の行にアクセスを繰り返すことによって，その近隣の行でビット反転が誘発されるという現象である [1]. 反転を起こしたいメモリ領域に直接アクセスすることなくデータを書き換えることが可能となるため，コンピュータシステムのセキュリティにおける重大な問題であると考えられている．ソフトウェアを使用して意図的にビット反転を起こさせる攻撃を，Rowhammer 攻撃という．

Rowhammer は，DRAM のセル密度の増加やキャパシタのサイズ縮小が原因となって発生し始めた．セル密度が増加に伴い，ワード線が互いに近接して配置された結果，あるワード線の電圧が上げられることで近隣のワード線の電圧が部分的に高くなり，その位置にあるセルから電荷が失われる．短時間に同じ行を何度もアクティブにすることでこの影響は大きくなり，リフレッシュが行われる前にノイズマージンを超えるだけの電荷が失われると，ビット反転が起こる．

文献 [1] では，多くの DDR3 DRAM モジュールで Rowhammer によるビット反転が発生するとされている．なお，ビット反転の起こりやすさはハードウェアに依存する．

Rowhammer によるビット反転の起こりやすさは，上下にあるビットの値に依存する．具体的には，充電されたセルの上下に充電されていないセルがある場合に，ビット反転は最も起こりやすくなる [1]．簡単な例を以下に示す．同じ列にある隣接した 3 つのビットの値を上から順に $x-y-z$ と表すこととする．true セルの場合，ビットの値が 1 のセルは充電されている．そのため，$0-1-0$ の状態が最もビット反転が起こりやすい．この状態で上下のビットを含む行に繰り返しアクセスすることで，真ん中のビットで反転が起こり $0-0-0$ に変化する可能性がある．

3.2 RAMBleed

Rowhammer をベースとした攻撃は, そのほとんどがメモリ上のデータを書き換えて, その信頼性を脅かすものであった．Kwong らはデータの機密性に影響を与える RAMBleed を提案した [5]．RAMBleed は，攻撃者が Rowhammer によるビット反転を利用して，その上下にあるビットの値を高い確率で推測する攻撃である．これにより，攻撃者は直接アクセスできないメモリ領域に格納されている秘密鍵などといった情報を読み取ることが可能となる．また，攻撃にはメモリの割り当てと解放，攻撃プロセスが所有するメモリ領域へのアクセスができれば十分であるため，攻撃者は特権を必要としない．RAMBleed では，次の 3.2.1 項から 3.2.3 項の手順で情報を読み取る．

3.2.1 メモリレイアウトの作成

攻撃者はまず，物理メモリ上の隣接した 3 行を確保して，図 2(a) に示すレイアウトを作成する．それぞれのページサイズは 4KiB であり，各行に 2 ページが含まれる．ここで，図中の A0，A1，A2 で示した部分は攻撃者のプロセスに割り当てるページであり，T0 及び T1 で示した部分は攻撃対象となるプロセスに割り当てるページである．

レイアウトを作成した後，攻撃者は Rowhammer 攻撃を実行する．ここで，DRAM 内での行バッファへの転送は行単位でまとめて行われるため，例えば攻撃者が A0 のページにアクセスすると A0 と T0 の 2 ページ分のデータが行バッファへ転送されることとなる．そのため，A0 と A1 の 2 ページに何度も繰り返しアクセスして Rowhammer 攻撃を行うことで，A2 内でのビット反転を誘発することができる．また，A2 は攻撃者のプロセスに割り当てるページのため，攻撃者はデータの読み書きが可能であり，ビット反転が起こったかどうかを確認できる．

3.2.2 秘密情報の配置

攻撃者は秘密情報を回復するために，攻撃対象となるプロセスを起動し，Target Page の位置に回復したい情報を含むページを誘導する必要がある．Kwong らは，buddy allocator の動作を利用した FFS（Frame Feng Shui）と呼ばれる以下の方法を示した [5]．この方法では，攻撃対象のプロセスが秘密情報にアクセスするまでに割り当てを受ける物理メモリページ数を，攻撃者が知っていることが前提となる．

step 1. ページの確保

攻撃対象のプロセスが起動後，$n+1$ ページ目に目的の秘密情報を格納すると仮定する．このとき，攻撃者は n ページ分の領域を確保しておく．

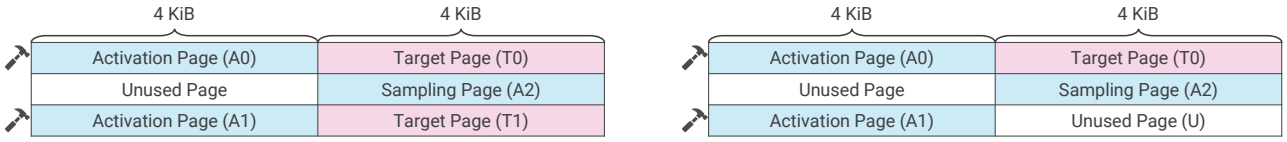
step 2. ページの解放

攻撃者は，秘密情報を置かせたいページ（Target Page）を解放する．その後，step 1 で割り当てられた n ページを全て解放する．この結果，図 3 のように，buddy allocator のオーダー 0 の最上位に n ページ分の領域があり，その直後に Target Page がある状態になる．

step 3. 攻撃対象プロセスの起動

step 2 の直後に，攻撃者は攻撃対象のプロセスを起動することで，オーダー 0 の初めの n ページが割り当てられた後，Target Page に秘密情報が格納される．

これを 2 回行うことで，T0 と T1 の位置に同じデータを置かせることができる．



(a) double-sided RAMBleed　　(b) single-sided RAMBleed

図 2　RAMBleed のメモリレイアウト

図 3　秘密情報を配置中の buddy allocator の様子

3.2.3　情報の読み取り

RAMBleed では，ビット反転のデータ依存性を利用した秘密情報の読み取りを行う．まず，あるページ P の先頭から i 番目（$i \in \{0, 1, ..., 32767\}$）のビットを $P[i]$ で表す．A2 に反転が起こるビットが含まれていると，Target Page に格納されているデータの一部を次のようにして読み取ることができる．攻撃者はあらかじめ A2 内でビット反転が起こる位置を特定し，それを保持しておく．ここでは，$A2[i]$ でビット反転が見つかったとする．次に，$A2[i]$ の値を 1 にする．その後，A0 と A1 へ繰り返しアクセスし，A2 に対して Rowhammer 攻撃を行う．最後に，攻撃者は $A2[i]$ の値を確認する．

$T0[i]$ と $T1[i]$ の値が共に 1 の場合，この 1 列は $1-1-1$ となるため，$A2[i]$ でのビット反転が起こりにくい．逆に，$T0[i]$ と $T1[i]$ の値が共に 0 の場合，この 1 列は $0-1-0$ となるため，$A2[i]$ でビット反転が起こり $0-0-0$ となりやすい．よって，Rowhammer 攻撃を行った後は高い確率で $T0[i], T1[i]$ の値と $A2[i]$ の値は一致する．したがって，$A2[i]$ の値を確かめることで攻撃者は情報の一部を高い確率で回復できる．図 2(a) のようなレイアウトを用いるとき，Sampling Page に対して上下の両方の行に Target Page を配置することから，double-sided RAMBleed と呼ばれている．一方で，図 2(b) のように，Sampling Page に対して上下のどちらか片方のみに Target Page がある場合を single-sided RAMBleed と呼ぶ．double-sided RAMBleed の途中で秘密情報の配置に失敗したときに，single-sided RAMBleed となることがある．single-sided RAMBleed の場合は，ページ U の位置には任意のデータが格納される．そのため，$T0[i]$ の値と $U[i]$ の値は一致しないことがあり，読み取り精度は低下する．

4.　OpenSSL に対する RAMBleed 攻撃

本章では，OpenSSL を用いて実装した SSL/TLS サーバが保有する RSA 秘密鍵を RAMBleed により回復する手法について述べる．

RAMBleed を実行するためには，秘密情報を Target Page に誘導する必要がある．そのため，SSL/TLS サーバがアクセスのリクエストを受けたときに，どのタイミングで秘密鍵にアクセスするかを調査した．サーバプロセスがあるクライアントからアクセスリクエストを受けたとき，サーバプロセスはハンドシェイクの過程で秘密鍵にアクセスしていることが分かった．しかしながら，サーバプロセスのメモリの使用状況の分析と仮想メモリダンプの解析を行った結果，その秘密鍵はクライアントからのリクエストを受ける前の時点で秘密鍵が物理メモリ上にロードされていることが分かった．

FFS により秘密鍵を Target Page に誘導するためには，アクセスリクエストの後にサーバプロセスが物理メモリに秘密鍵を割り当てる必要がある．今回の SSL/TLS サーバのプロセスのように，リクエストの前に秘密鍵に物理メモリ上が割り当てられている場合には，FSS の手順の中で別の位置，すなわち Target Page に割り当てを変更することができず，従来の手順により RAMBleed を実行することができない．

本稿では，秘密鍵を含むページに物理メモリを割り当てるタイミングを考慮して，回復したい秘密情報に物理メモリが割り当てられている場合でも，RAMBleed を実行可能にする手法を提案する．すなわち，秘密情報に割り当てられた物理メモリを，Target Page に変更する手法を提案する．

秘密鍵を含むページに物理メモリが割り当てられるタイミングは 2 通りある．

(1) サーバプロセスが起動してから初めて秘密鍵にアクセスしたとき
(2) サーバプロセスがスワップ領域にある秘密鍵にアクセスしたとき

RAMBleed による攻撃を始めた後に，（1）のタイミングが発生するプロセスでは，従来の FFS により秘密情報を Target Page に誘導することが可能である．例えば，OpenSSH では，サーバにアクセスする際に新規にプロセスが起動して，そのプロセスで利用する秘密鍵に物理メモリを割り当てるため，（1）のタイミングを利用して，Target Page への誘導が可能である．

一方で，OpenSSL を利用したプロセスのように，RAMBleed による攻撃を始める前に秘密鍵が物理メモリ上にロードされている場合は，（1）のタイミングを利用できないため，（2）のタイミングを待つことになる．しかしなが

ら，RAMBleed による攻撃の実行中に，自然に物理メモリ上のデータがスワップ領域に移動し，再度そのデータにアクセスをするタイミングを待つことは容易ではない．

そこで提案手法では，秘密情報を意図的にスワップ領域に移動させる方法を示し，（2）のタイミングを半強制的に発生させる手順を組み込む．具体的には，攻撃プロセスから大量の物理メモリ割り当てを要求し，サーバプロセスが保有する物理メモリ上のデータを全てをスワップ領域に退避させる．物理メモリの空き容量が少ない状態で新たに物理メモリの割り当てを要求すると，ページ置換アルゴリズムに従って選ばれた物理メモリ上のデータを一時的にスワップ領域に退避させること（ページアウト）により，物理メモリに空きページを用意する．この動作を利用して，サーバプロセスで利用される秘密情報を含むデータを強制的にスワップ領域に移動することができる．この状態で，攻撃対象のプロセスが秘密情報にアクセスすれば，秘密情報に物理メモリが改めて割り当てられて，スワップ領域から物理メモリ上へロードされるため，FFS により Target Page に秘密鍵の誘導することが可能となる．

OpenSSL のように，RAMBleed を始める前に物理アドレス上に秘密情報がロードされている場合であっても，上記の方法により秘密情報を Target Page に誘導でき，RAMBleed を実行することで，秘密情報を回復可能である．

5. 秘密情報の誘導精度の評価

本章では，提案手法による秘密情報の誘導を SSL/TLS サーバに対して行い，Target Page にどの程度誘導できるかを測定し，その結果について述べる．

5.1 実験環境

実験には OS は Ubuntu 18.04，CPU は Core i5-4590，メモリは DDR3 4 GiB 1333 MHz non-ECC を 2 枚の計算機を利用した．

5.2 実験方法

ここでは，攻撃対象のプロセスが起動後，$n+1$ ページ目のアクセスで秘密情報へのアクセスが発生すると仮定する．また，秘密情報は攻撃対象のプロセスが起動する前に，物理メモリ上にロードされているものとする．実験の手順は次のとおりである．

(1) RAMBleed に用いるメモリレイアウトの Target Paeg に見立てたページを物理メモリ上に配置し，/proc/self/pagemap インターフェースを用いて物理アドレスを記録する．また，OpenSSL を用いたサーバープロセスの挙動に倣い，誘導する対象となる秘密情報の入ったページと，そのページにアクセスするまでにアクセスされる n ページは物理メモリ上にマッピングしておく．これらは mmap システムコールの発行時に MAP_POPULATE フラグを立てることで実現する．
(2) mmap を用いて 8GB の物理メモリ割り当てを要求し，物理メモリ上にあるデータをスワップ領域上へ退避させる．このとき，先ほど確保した物理メモリページがスワップ領域に入る．
(3) n ページの Dummy Page を物理メモリ上に確保する．
(4) Dummy Page，Target Page の順番に物理メモリへのマッピングを解き，メモリを開放する．その直後に（1）で確保した n ページ，秘密情報が入ったページにアクセスする．本実験では，memset を用いてページ内に文字列を書き込むことでページへのアクセスを行った．
(5) 秘密情報の入ったページと Target Page の物理アドレスを比較し，一致しているかどうかを確認する．

以上のようにして，Dummy Page 数 n を増やしながら秘密情報を含むページが Target Page として確保した物理メモリ上の位置に格納されているかどうかを検証した．

5.3 実験結果と考察

ある Dummy Page 数に対して複数回誘導実験を行い，成功率を求めた結果を表 1 に示す．実験の結果，一定の確率で Target Page に秘密情報を含む物理メモリページを誘導できることが確認できた．誘導に失敗する原因として，攻撃プロセスが解放した Dummy Page が対象のプロセスよりも早く他のプロセスへの物理メモリ上割り当てに使用されてしまった結果，秘密情報を含むページが Target Page 以外の物理メモリページへ割り当てられるということが考えられる．また，Dummy Page の数が大きいほど，他のプロセスに Dummy Page が割り当てられる可能性が高くなるため，誘導の成功率が低下している．

表 1 Dummy Page 数ごとの誘導成功率

Dummy Page 数	試行回数	成功数	成功率 (%)
1	50	33	66.00
2	30	14	46.67
3	30	16	53.33
4	30	12	40.00
5	30	8	26.67
6	30	10	33.33
7	30	10	33.33
8	30	7	23.33
9	30	9	30.00
10	30	6	20.00
20	10	2	20.00
30	10	0	0.00
40	10	0	0.00
50	10	0	0.00

6. 手法の制限事項

本稿では，OpenSSL のように，攻撃の実行前に秘密情報が物理メモリ上にロードされる場合においても，RAMBleed により秘密情報の回復を実行する手法を提案した．しかし，以下に示す理由で，提案手法がいかなる環境に対して適用できるわけではない．

- Rowhammer によってビット反転が起こるビットの割合は，メモリの規格や個体によって変化する．もしビット反転する割合が小さいと，準備フェーズに要する時間が長くなる．さらに，ビット反転する割合が小さすぎると，秘密情報の回復に必要なビット数を得られなくなることもある．したがって，一定数以上ビット反転するメモリを使用する必要がある．
- 2 を作成する際に，物理アドレスからメモリ上への位置へマッピングを利用しているが，全ての CPU で明らかになっているわけではない．そのため，マッピングが不明な CPU が用いられていた場合，メモリレイアウトの作成ができず，RAMBleed が実装できない．したがって，マッピングが明らかになっている CPU を使用する必要がある．
- 今回の実験では，攻撃対象のプロセスが起動後，$n+1$ ページ目のアクセスで秘密情報へのアクセスが発生すると仮定している．一般のサーバ環境では，攻撃プロセス，攻撃対象のプロセスの他に様々なプロセスが動作しているため，n の値を確実に推定できるとは限らない．

以上の制限は従来の RAMBleed についても同様であり，必ずしも全ての環境で RAMBleed にを保つことができるわけではない．秘密情報を Target page に誘導する際には，三つ目の課題を評価することが重要となり，OpenSSL を用いて実装した SSL/TLS サーバにおいても，n の値をどの程度推測できるかの評価を行っていく予定である．

7. まとめ

DRAM の高密度化に伴い，特定のビットに繰り返しアクセスすることで，他のビットに影響が起こる Rowhammer の問題が生じている．また，Rowhammer を秘密情報の回復に悪用した攻撃として，RAMBleed が提案されている．RAMBleed は直接アクセスできない秘密情報の一部を高確率で回復することができ，コンピュータシステムのセキュリティを脅かす問題となっている．本稿では，OpenSSL を用いて実装した SSL/TLS サーバが保有する RSA 秘密鍵を RAMBleed により回復することを試みた．OpenSSL では，RAMBleed を実行する前に，秘密情報が物理メモリ上に割り当てられており，従来の方法では Target Page に秘密鍵を誘導することができない．そこで，物理メモリ上のデータを強制的にスワップ領域に移動させ，物理メモリ上に再度割り当てさせる手法を提案し，秘密鍵を Target Page に誘導できる可能性を示した．

謝辞 本研究は JSPS 科研費 20K11810 の助成を受けたものです．

参考文献

[1] Y. Kim, R. Daly, J. Kim, C. Fallin, J. Lee, D. Lee, C.Wilkerson, K. Lai, and O. Mutlu, “Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors,” ACM/IEEE 41st International Symposium on Computer Architecture, ISCA 2014, Minneapolis, MN, USA, June 14-18, 2014, pp.361–372, 2014.

[2] M. Seaborn and T. Dullien, “Exploiting the dram rowhammer bug to gain kernel privileges,” Black Hat, vol.15, p.71, 2015.

[3] S. Bhattacharya and D. Mukhopadhyay, “Curious case of rowhammer: Flipping secret exponent bits using timing analysis,” Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings, pp.602–624, 2016.

[4] Y. Jang, J. Lee, S. Lee, and T. Kim, “Sgx-bomb: Locking down the processor via rowhammer attack,” Proceedings of the 2nd Workshop on System Software for Trusted Execution, SysTEX@SOSP 2017, Shanghai, China, October 28, 2017, pp.5:1–5:6, 2017.

[5] A. Kwong, D. Genkin, D. Gruss, Y. Yarom, ”RAMBleed: Reading Bits in Memory Without Accessing Them, “ In 41st IEEE Symposium on Security and Privacy (S&P), 2020.

[6] 長濱拓季, 瀧田愼, 廣友雅徳, 森井昌克, “効果的な single-sided RAMBleed の提案,” 電子情報通信学会技術研究報告（情報通信システムセキュリティ），2020 年 3 月.

[7] “JEDEC. Standard No. 79-3F. DDR3 SDRAM Specification,” July 2012.