DRAMに対する効果的なRAMBleed攻撃の提案

西尾 和浩[1,a)]　瀧田 愼[2,b)]　廣友 雅徳[3,c)]　白石 善明[1,d)]　森井 昌克[1,e)]

概要：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．一方で，DRAM 内で意図しないビット反転が生じる Rowhammer と呼ばれる問題が指摘されている．Rowhammer を利用した攻撃は多数存在し，ソフトウェアで容易に実行可能であるため，コンピュータシステムのセキュリティへの影響が懸念されている．RAMBleed は Rowhammer により意図的に引き起こしたビット反転を観測することで直接アクセスできない情報を読み取るサイドチャネル攻撃である．RAMBleed の課題として，攻撃にかかる時間があげられる．本稿では，攻撃時間を短縮する目的で，従来の RAMBleed の攻撃手順を見直し，ビット反転確率の高いビット位置を利用した RAMBleed 攻撃を提案する．提案手法と従来の RAMBleed を実装し，秘密情報の回復精度と攻撃時間の比較実験を行った．実験の結果，提案手法では，従来の RAMBleed の回復精度を保ったまま，秘密情報の回復時間を約 60%に短縮できることを示した．

キーワード：サイドチャネル攻撃，DRAM，Rowhammer, RAMBleed

Effective RAMBleed Attack on DRAM

Kazuhiro Nishio[1,a)]　Makoto Takita[2,b)]　Masanori Hirotomo[3,c)]　Yoshiaki Shiraishi[1,d)]
Masakatu Morii[1,e)]

Abstract: The high-capacity DRAM has improved the performance of computers. On the other hand, a problem called Rowhammer, in which unintended bit flips occur in DRAM, has been pointed out. There are many attacks using Rowhammer and can be easily executed by software, which is a serious problem in computer system security. RAMBleed is a side channel attack that reads information that cannot be accessed directly by observing bit flips caused by Rowhammer. The issue of RAMBleed is the time it takes to attack. In this paper, in order to reduce the attack time, we review the conventional RAMBleed attack procedure and propose a RAMBleed attack that uses bit positions with high bit flip probability. We implemented the proposed method and conventional RAMBleed, and compared them. As a result, we show that the proposed method can reduce the attack time by about 60% while maintaining the recovery accuracy of conventional RAMBleed.

Keywords: side channel attack，DRAM，Rowhammer, RAMBleed

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

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) nishio@stu.kobe-u.ac.jp
b) takita@sis.u-hyogo.ac.jp
c) hirotomo@cc.saga-u.ac.jp
d) zenmei@port.kobe-u.ac.jp
e) mmorii@kobe-u.ac.jp

© 2020 Information Processing Society of Japan

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 に関する攻撃の評価やその対策に関する研究が重要である．

Rowhammer はソフトウェアを用いて容易に実現でき，DRAM の特定のビットにアクセスを繰り返すことより，その近隣でビット反転が生じる．Rowhammer を利用すれば，ビット反転によりメモリ上の値を書き換えて，データの完全性を阻害する攻撃が可能である．さらに，2019 年に Kwong らは 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ている [5]．RAMBleed は，Rowhammer により意図的に引き起こした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 を実行し，秘密情報を読み取るにはいくつか課題がある．その中の一つは，攻撃プロセスの中で攻撃対象のプロセスを制御し，回復したい情報をメモリ上の意図した位置に誘導する必要があるということである．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．その他の課題として，攻撃にかかる時間である．特に，攻撃対象のプロセスに接続し，秘密情報を配置させたあと，Rowhammer によってビット反転を誘発し，ビット反転から秘密情報を読み取るといった時間が長いと，攻撃が検知されやすくなり，攻撃する上での障害となる．

前者の課題の解決策として，著者らは，従来の RAMBleed の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 と同程度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．

後者の課題を解決する目的で，本稿では，Rowhammer により反転しやすいビットのみを利用した RAMBleed を提案する．ビット反転しやすいビットのみを攻撃に使用することで，秘密情報の読み取り精度が向上する．その結果，精度よりも時間を重視した攻撃が可能となり，攻撃時間の短縮が実現できる．RAMBleed を実行する際には，事前準備として Rowhammer を行いビット反転が生じるビット位置を把握する．実際の攻撃時には，このビット位置の近くに秘密情報を誘導し，ビット反転が生じるかどうかを観測することで秘密情報の回復を行っている．秘密情報の回復精度が下がる要因を分析したところ，事前準備の段階でビット反転が起きたビットであっても攻撃時にビット反転が起きない場合があることがわかった．また，その逆で，準備段階でビット反転が起きず，攻撃時にビット反転が起きる場合もある．このような要因を減らすために，準備段階での Rowhammer を繰り返し行い，ビット反転が生じやすいビット位置を把握し，その位置のみを攻撃時に利用することを提案する．

single-sided RAMBleed に対して，提案手法を適用し，秘密情報の回復時間と回復精度を比較した結果，回復精度を全く落とすことなく秘密情報の回復時間を約 60%に削減できることを確認した．また，従来の RAMBleed および single-sided RAMBleed に対して提案手法を適用し，読み取り精度の比較実験を行った結果，double-sided RAMbleed においては回復精度が約 96.2%から約 97.4%に，single-sided RAMbleed においては回復精度が約 95.4%から約 96.8%となり，おおよそ 1%の精度が向上した．

2. DRAM

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．

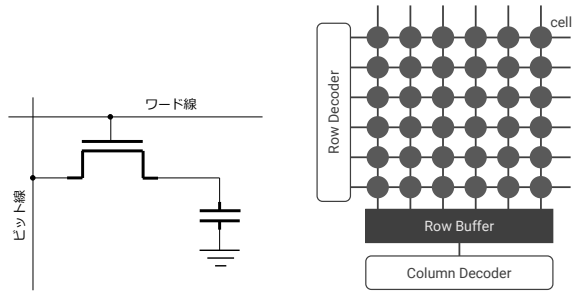
セルは図 1(b) のような行列の形に並べて配置されており，各行はワード線で，各列はビット線で接続されている．一定数の行をまとめて Bank と呼び，複数の Bank から DRAM チップが構成される．さらに，複数の DRAM チップをまとめて Rank と呼び，メモリモジュールである DIMM の片面に実装される．

DRAM 内でのデータの扱いは，8KiB から成る行単位でなされる．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].

3. RAMBleed

3.1 Rowhammer

Rowhammer は DRAM の特定の行に繰り返しアクセス



(a) セル　(b) Bank

図 1　DRAM の構造
Fig. 1　structure of DRAM

することによって，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．これを悪用して，意図的にビット反転を起こす攻撃を Rowhammer 攻撃と呼ぶ．ビット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，システムのセキュリティを脅かす重大な問題であると考えている．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．セル密度が増加するとともに，ワード線が互いに近接して配置されるようになった．その結果，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．Rowhammer によるビット反転の起こりやすさは，自身のビットとその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，充電されたセルのビット反転が最も起こりやすくなる．つまり，同じ列にある隣接した三つのビットの値が上から順に 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

2019 年に Kwong らによって Rowhammer をベースとする攻撃として，RAMBleed と呼ばれるサイドチャネル攻撃が提案された．RAMBleed では，攻撃者が Rowhammer によるビット反転を観察することで，その上下にあるビットの値を高い確率で推測する．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情報を読み取ることが可能となる．RAMBleed では，準備フェーズと回復フェーズの 2 段階により秘密情報の読み取りを行う．

準備フェーズと回復フェーズに共通して，Rowhammer 攻撃を行うためのメモリ領域を整える必要がある．攻撃者はメモリ上の隣接した 3 行を確保して，**図 2** に示すような

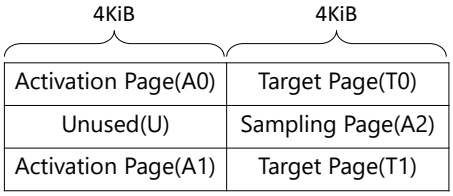


図 2　RAMBleed のメモリレイアウト
Fig. 2　double-sided RAMBleed

レイアウトを作る用意をする．それぞれの行に 4KiB のサイズのページが二つ含まれる．ここで，図 2 中の A0，A1，A2 は攻撃者のプロセスに割り当てるページ，T0，T1 は攻撃対象のプロセスに割り当てるページ，U は任意のページである．すなわち，攻撃者のプロセスからは A0，A1，A2 のみにアクセスでき，T0，T1 に直接アクセスすることはできない．

準備フェーズでは，秘密情報を読み取る前の準備として，Rowhammer によりビット反転が起こるビット位置を予め探索する．DRAM メモリのすべてのビットが Rowhammer によりビット反転することはなく，電荷の漏れが発生しやすい位置にのみ反転が生じる．そこで，図 2 のようなメモリ領域を確保したあと，T0 と T1 のページはすべてのビット値を 0 に，A2 のページはすべてのビット値を 1 にする．すなわち，Rowhammer によるビット反転が起こりやすい 0-1-0 の状態にする．その後，A0 と A1 にアクセスを繰り返す Rowhammer 攻撃を実行する．DRAM へのデータアクセスは行単位であるため，A0 にアクセスすると，A0 と T0 の 2 ページ分のデータが転送されることになる．同様に，A1 にアクセスすると A1 と T1 の 2 ページが転送される．そのため，A0 と A1 の 2 ページに繰り返しアクセスすることで，DRAM の同じ行にある T0 と T1 も同時に読み出され，A2 でビット反転を誘発することができる．A2 はデータを直接読み取ることができ，ビット反転が起こったかどうかを確認できる．A2 において，ビット値が 0 となっている箇所をビット反転の生じた位置として記録する．秘密情報の回復に必要なビット数が得られるまで，レイアウトを作成から繰り返し行い，ビット反転する位置を記録し続ける．

次に回復フェーズでは，図 2 のようなメモリ領域を確保したあと，T0 と T1 に秘密情報を格納する．T0 と T1，および秘密情報に直接アクセスすることはできないため，攻撃対象のプロセスを制御し，この位置へ秘密情報を誘導して格納させる必要がある．Kwong らは buddy allocator の動作を利用して，この誘導を実現している．

RAMBleed では，ビット配置によるビット反転の生じやすさの違いを利用して，情報の読み取りを行う．あるページ P の先頭から i 番目のビットを $P[i]$ で表す．事前準備により $A2[i]$ でビット反転が見つかっていたとする．図 2 のレイアウトを作成し，T0 と T1 に秘密情報を配置し，$A2[i]$

の値を 1 に設定する．その後，A0 と A1 に繰り返しアクセスし，A2 に対して Rowhammer 攻撃を行い，$A2[i]$ の値を確認する．$T0[i]$ と $T1[i]$ の値が 1 のとき，i 列は 1-1-1 の状態となるため，$A2[i]$ でビット反転は起こりにくく，$A2[i]$ の値は 1 のままである可能性が高い．反対に，$T0[i]$ と $T1[i]$ の値が 0 のとき，i 列は 0-1-0 となるため，$A2[i]$ でビット反転が起こりやすく，$A2[i]$ の値は 0 となる可能性が高い．したがって，$A2[i]$ の値は高確率で $T0[i]$ と $T1[i]$ の値に一致し，$A2[i]$ の値を確認することで，秘密情報が格納されたページに直接アクセスすることなく，情報を読み取ることができる．

3.3 single-sided RAMBleed

前節で説明した RAMBleed を実行するためには，T0 と T1 に秘密情報を格納する必要があり，Kwong らは buddy allocator の動作を利用して秘密情報を誘導する方法を示している．Kwong らが OpenSSH に対して行った実験では，意図したページに秘密情報を誘導できた確率は 83%程度であり，必ずしも成功するわけではない．さらに，T0 と T1 の 2 ページに誘導するときの成功確率は $0.83^2 = 68.89\%$ となってしまう．T0 と T1 のどちらか片方のみで誘導が成功する場合もあるが，成功しなかったページに格納される情報はランダムな値になるため，秘密情報の回復精度が劣化する．

著者らは RAMBleed における事前準備における反転する位置を探す際のメモリレイアウトを変えることで，回復時に秘密情報を格納するページを T0 のみにする新しい single-sided RAMBleed を提案している [6]．これにより，誘導の手順を 2 回から 1 回に削減でき，攻撃の実行のしやすさを改善できる．

従来の RAMBleed からの主な変更点は下記のとおりである．まず，図 2 のような 3 行のメモリ領域を確保する行うことは同様に行う．次に，Rowhammer は 0-1-0 の状態のときにビット反転が起こりやすいが，0-1-1 の状態でもビット反転は発生し，情報の回復が可能であることに着目してビット反転の位置を探す．すなわち，準備フェーズにおいて，T0 のページはすべてのビット値を 0 に，A2 と T1 のページはすべてのビット値を 1 にして，Rowhammer 攻撃を実行する．A2 において，ビット反転して 0 となったビット位置を記録する．回復フェーズでは，T0 のみに秘密情報を配置し，A2 と T1 はすべて 1 とする．$T0[i]$ の値が 1 のとき，i 列は 1-1-1 の状態となるため，$A2[i]$ でビット反転は起こりにくく，$A2[i]$ の値は 1 のままである可能性が高い．反対に，$T0[i]$ の値が 0 のとき，i 列は 0-1-1 となるため，$A2[i]$ でビット反転が 1-1-1 の状態より起こりやすく，$A2[i]$ の値は 0 となる可能性が高い．したがって，$A2[i]$ の値は高確率で $T0[i]$ の値に一致し，RAMBleed と同様に情報を読み取ることができる．

4. ビット反転確率の高いビット位置を利用した RAMBleed 攻撃

前章のとおり，RAMBleed はビット配置によるビット反転の生じやすさの違いを利用して，情報の読み取りを行う攻撃である．具体的には，隣接した 3 ビットが 0-1-0 のときには中央のビットが 0 に反転しやすい傾向があり，1-1-1 のときには中央のビットが反転しにくい傾向があることを利用して，左右のビットを推測している．このビット反転の起こりやすさは，メモリ内の各ビットで異なり，一般的には 0-1-0 の配置でも反転しないビットが多い．したがって，RAMBleed を実行する際には，まず準備フェーズで，0-1-0 のビット配置のときに中央のビットが反転する位置を探す必要がある．そして回復フェーズで，探索した位置において 0-1-0 の配置になっていれば，中央のビットは反転するだろうという考えのもと推測を行っている．しかしながら，回復フェーズで，0-1-0 の配置であっても反転しない場合があり，回復精度が下がる要因となっている．これはビット反転の起こりやすさがビットごとに様々であることに起因しており，準備フェーズにおいてビット反転の確率が小さいビットがたまたま反転して，回復フェーズで利用されていることが考えられる．

一般的に攻撃にかかる時間は，攻撃の成否に関わる重要な要素であり，RAMBleed においては回復フェーズが該当する．回復フェーズに行う Rowhammer で各ページにアクセスする回数を減らすことで，回復フェーズの時間を減らすことが可能である．その一方で，ビット反転する確率が低下するので，回復精度が低下する可能性がある．

回復精度を向上させるための単純な方法として，RAMBleed による秘密情報の回復を複数回繰り返し行い，その結果の多数決により秘密情報を推測するという方法が考えられる．この方法では，RAMBleed の回復フェーズの実行時間が伸び，攻撃する上での障害となってしまう．また，秘密情報を指定のページに誘導する回数も増えてしまい回復フェーズのコストが増大してしまう．

本稿では，Rowhammer によるビット反転のしやすさに着目し，準備フェーズを繰り返し行うことで，ビット反転する確率が高いビットのみを回復フェーズに利用する手法を提案する．すなわち，準備フェーズを複数回行い，その全てでビット反転が起きたビットのみを回復フェーズに利用する．この方法により，準備フェーズにおいて 0-1-0 の配置でビット反転したにも関わらず，回復フェーズにおいて 0-1-0 の配置でビット反転しないというビットを減らすことが可能となる．これによって，回復精度が上昇した分，Rowhammer で各ページにアクセスする回数を減らすことができ，回復フェーズの時間を削減することが可能となる．また，この手法は single-sided RAMBleed にも適用可能であり，その場合は 0-1-1 の配置で準備フェーズを繰り返せ

ば良い．

5. 実験と結果

本章では提案手法と既存手法による回復精度と攻撃の実行時間の比較を行う．1 ページ全体に秘密情報が格納されているとし，そのうち 5000 ビットを回復するのに必要な時間や秘密情報の回復精度を比較する．

5.1 実験環境

実験に用いた PC の構成やメモリは次の通りである．OS は Ubuntu 18.04，CPU は Core i5-4590，メモリは DDR3 4 GiB 1333 MHz non-ECC を 2 枚使用した．Rowhammer で各ページにアクセスする回数は文献 [1] で公開されているツールに倣って，ビット反転の検出では 100 万回，秘密情報の読み取りでは 200 万回とした．

5.2 従来の RAMBleed での比較

本節では，従来の RAMBleed とそれに提案手法を適用した攻撃を比較する．提案手法では準備フェーズを 3 回繰り返した場合と 5 回繰り返した場合で実験を行う．すなわち，3 回，あるいは 5 回ともビット反転した位置のみを回復フェーズに使う．回復フェーズでは，1-1-1 の状態で中央ビットの反転が起こらなかった，或いは 0-1-0 の状態で中央ビットの反転が起こった場合に情報を正しく読み取れていることになる．その他に，1-1-1 の状態で反転が起こった場合を False Positive，0-1-0 の状態で反転が起こらなかった場合を False Negative と表す．

実験結果を**表 1** に示す．準備時間は，準備フェーズに要した時間である．回復時間は，秘密情報の回復に要した時間である．既存手法に比べて，提案手法では秘密情報の回復精度を向上させることができている．これは，提案手法の方が，False Negative の割合が下がっているためである．つまり，準備フェーズにおいて 0-1-0 の配置でビット反転したにも関わらず，回復フェーズにおいて 0-1-0 の配置でビット反転しなかったという状況が減っていることがわかる．準備フェーズを繰り返すため，準備時間が大きくなるが，このフェーズでは攻撃対象のプロセスに無関係に実行できるため，準備時間は攻撃の実行しやすさには大きく影響しないものと考えている．3 回繰り返しと 5 回繰り返しを比較すると，回復精度の向上は見られるもののその伸びは小さくなっている．

5.3 single-sided RAMBleed での比較

本節では，single-sided RAMBleed とそれに提案手法を適用した攻撃を比較する．前節と同様に，提案手法では，準備フェーズを 3 回と 5 回で実験を行う．実験結果を**表 2** に示す．前節と同様に，提案手法は既存手法に比べて，準備時間は長くなるものの，秘密情報の回復精度は向上しており，single-sided RAMBleed においても提案手法が適用できることを示している．また，従来の RAMBleed と提案手法を適用した single-sided RAMBleed と比較すると，回復精度が低下していないことがわかる．つまり，回復精度を低下させることなく single-sided RAMBleed を適用することができ，秘密情報の配置手順を半分に削減することが可能となっているため，攻撃にかかるコストは削減できていると考えられる．

表 1 従来の RAMBleed での比較
Table 1 A comparison of existing RAMBleed

	既存手法	3 回繰り返し	5 回繰り返し
準備時間 (s)	3916	9060	13889
回復時間 (s)	1676	1676	1705
回復精度 (%)	96.20	97.24	97.40
False Positive(%)	0.97	0.98	0.87
False Negative(%)	2.83	1.78	1.73

表 2 single-sided RAMBleed での比較
Table 2 A comparison of single-sided RAMBleed

	既存手法	3 回繰り返し	5 回繰り返し
準備時間 (s)	8578	16903	25114
回復時間 (s)	1980	1974	1961
回復精度 (%)	95.44	96.18	96.88
False Positive(%)	1.37	1.16	1.12
False Negative(%)	3.19	2.66	2.00

表 3 アクセス回数による RAMBleed の比較
Table 3 A comparison of RAMBleed by number of accesses

	既存手法		3 回繰り返し	5 回繰り返し
アクセス回数	200 万回	80 万回	80 万回	80 万回
準備時間 (s)	8578	6559	14609	21415
回復時間 (s)	1980	1282	1269	1247
回復精度 (%)	95.44	95.00	95.52	95.62
False Positive(%)	1.37	0.72	0.87	0.91
False Negative(%)	3.19	4.28	3.61	3.47

次に，提案手法ではビット反転しやすいビットのみを利用することを念頭に置いて，回復フェーズで Rowhammer を実行する際にページアクセスを繰り返す回数を減らすことを検討した．Rowhammer で各ページにアクセスする回数を 80 万回に変更して，従来の RAMBleed でアクセス回数が 200 万回の場合と回復するのに必要な時間や秘密情報の回復精度を比較する実験を行った．その実験結果を**表 3** に示す．表 3 の結果より，回復精度を全く落とすことなく回復時間を約 60%に削減することに成功した．回復フェーズは攻撃対象のプロセスと関係が深く，攻撃時間の短縮は，攻撃の成功に大きな貢献があるものと考えられる．

6. 手法の制限事項

本稿では，従来の RAMBleed の攻撃手順の一部を修正し，秘密情報の回復精度を向上させる方法を提案した．ま

た，提案手法を利用することで，従来の精度を保ったまま，秘密情報の回復時間を約 6 割の時間に短縮することが可能であることも示した．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．
- 図 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．

以上の制限があるが，従来の RAMBleed についても同様である．文献 [5] では，OpenSSH の鍵を約 82%で回復できると報告しているが，上記に示した制限があるために，必ずしも全ての環境でこの精度を保つことができるわけではない．また，秘密情報を両方に置けた場合は double-sided RAMBleed に，片方にのみ置けた場合は single-sided RAMBleed になると主張しているが，実際に秘密情報がどのように置かれているかは攻撃者にはわからず，実環境の CPU やメモリの使用状況によって変化する．このため，実環境において厳密な回復の精度を導出することは難しいと考えられる．

7. むすび

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．しかし，RAMBleed の課題として，攻撃にかかる時間があげられる．本稿では，従来の RAMBleed の攻撃手順の一部を修正することで，回復フェーズでの Rowhammer におけるアクセス回数を削減したとしても，従来の RAMBleed に比べて回復精度を低下させずに，秘密情報の回復時間を約 60%に短縮することが可能となった．回復時間の短縮は攻撃の検知がされにくくなり，攻撃の成功に大きな貢献があると考えられる．

謝辞　本研究は JSPS 科研費 JP20K11810 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.