一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
ICM2020-15(2020-07)

Bloom Filterを用いたキャッシュポリューション攻撃の検知

芦原　大和†　　上山　憲昭†

† 福岡大学大学院 工学研究科 電子情報工学専攻　〒 814–0180 福岡市城南区七隈 8–19–1
E-mail: †td202001@cis.fukuoka-u.ac.jp, ††kamiyama@fukuoka-u.ac.jp

あらまし　Web 閲覧や動画配信サービスを快適に行うため、ネットワークの様々な場所にキャッシュサーバを設置しユーザの近くに存在するキャッシュサーバからデータを配信するキャッシュ配信が広く行われている．キャッシュ配信は，5G セルラーネットワークの基地局にキャッシュサーバを設置することでバックホールのトラヒック負荷を低減する MEC (mobile edge computing)，インターネット上の多数のネットワークに配置されたキャッシュサーバからコンテンツを配信する CDN (content delivery network)，次世代のネットワークとして期待を集める情報指向ネットワーク (ICN: information-centric networking) といった様々なネットワークで使用される．しかし悪意を持ったユーザが意図的に低人気のコンテンツに多数の要求を行うことでキャッシュの効果を低下させるキャッシュポリューション攻撃 (CPA : cache pollution attack) の問題が指摘されている．CPA 発生時にはできるだけ迅速に攻撃ホストを特定し，キャッシュを保護することが重要である．そこで少ないメモリ量とメモリアクセス回数で高精度に攻撃ホストを特定するため，本稿ではホスト ID とコンテンツ ID の組をキーとする Bloom Filter (BF) を用い，検知回数が BF 検知閾値を超えると CPA ホストとして検知する方式を提案する．また継続的な CPA ホスト検知を可能とするため，BF を 2 つ用意して交互に運用することを提案する．そして計算機シミュレーションによる性能評価を行い，提案方式を用いることで正常ホストの誤検知を防ぎつつ，CPA ホストを高精度に検出しキャッシュヒット率の低下を回避可能なことを示す．

キーワード　キャッシュ，キャッシュポリューション攻撃，Bloom Filter

Detecting Cache Pollution Attack Using Bloom Filter

Takakazu ASHIHARA† and Noriaki KAMIYAMA†

† Graduate School of Engineering, Fukuoka University 8–19–1, Nanakuma, Jounan, Fukuoka 814–0180
E-mail: †td202001@cis.fukuoka-u.ac.jp, ††kamiyama@fukuoka-u.ac.jp

Abstract　To efficiently providing web browsing and video streaming services with reasonable quality, cache delivery in which digital data is delivered to users from cache servers located close to users has been widly used. For example, in the MEC (mobile edge computing), cache memories are provided at base stations of 5G cellular networks to reduce the traffic load in the backhaul networks. Cache servers are also provided in various networks in the CDN (content delivery network) and at routers in the ICN (information-centric networking). However, the cache pollution attack (CPA) which degrades the effect of caches by intentionally sending many requests to non-popular content items will be a problem in the cache networks. Quickly detecting the CPA hosts and protecting the cache servers is important to effectively utilize the cache resources. Therefore, in this paper, we propose a method of accurately detecting the CPA hosts with using a limited amount of memory resources. The proposed method is based on a Bloom filter using the combination of host ID and content ID as keys. We also propose to use two Bloom filters in parallel to continuously operate the proposed detection method of CPA hosts and to introduce a threshold on the number of detections in the Bloom filter to identify the CPA hosts.Through numerical evaluation, we show that the proposed method reduce the degradation of the cache hit ratio caused by the CPA while avoiding the false identification of legitimate hosts.

Key words　cache, cache pollution attack, Bloom filter

1. はじめに

Web 閲覧や動画配信サービスを快適に行うため，多数のネットワーク上に多数のキャッシュサーバを配置し，ユーザの近くに存在するキャッシュサーバからデータを配信する CDN (content delivery network) が広く行われている．また近年，スマホなどの移動端末においても動画コンテンツなどの大容量コンテンツを視聴するユーザが増加し，基地局間を接続する

バックホールネットワークのトラヒック量が増大し，その設備投資コストの増大や品質の低下が懸念される．そこでバックホールネットワークのトラヒック量の増加を回避するため，基地局にキャッシュサーバを配置し，ユーザからの配信要求に対しキャッシュサーバからコンテンツを配信する Mobile Edge Computing (MEC) が注目され，精力的に研究が進められている [7] [10]．さらに次世代のネットワークとして期待を寄せられている情報指向ネットワーク (ICN: Information-Centric Networking) においても，ルータにキャッシュメモリを用意しルータからコンテンツを配信する [3]．

しかしキャッシュサーバの容量は有限であるため，存在する全てのコンテンツをキャッシュすることはできず，新たにコンテンツをキャッシュする際に容量が不足する場合にはキャッシュ置換アルゴリズムを用いて選択したキャッシュ済みコンテンツの一部を削除する．通常，コンテンツの人気には偏りがあるため，高人気のコンテンツを優先的にキャッシュに残すことでキャッシュヒット率の向上とトラヒック量削減効果の向上が期待できる．そのため最後に要求されてからの経過時間が最大のコンテンツから削除する LRU (least recently used) がキャッシュ置換アルゴリズムとして広く用いられている．

ところで悪意を持ったユーザが意図的に低人気のコンテンツに多数の要求を行うことで，キャッシュの効果を低下させるキャッシュポリューション攻撃 (CPA: cache pollution attack) の問題が指摘されている [8] [12]．そこでこれまでに筆者らは，少ないメモリアクセス回数で迅速に CPA ホストを検知するため，ホスト ID とコンテンツ ID の組をキーとする Bloom Filter（BF）を用いて CPA ホストを検知する方式を提案した [1]．しかし本方式では，正常なホストを多数，誤って検出する問題と，時間の経過に伴い BF のビットマップが埋まることから，継続して CPA ホストを検知することができない問題があった．そこで本稿では，正常なホストの誤検知を回避するため， BF の検知回数に閾値を設け，BF による検知回数が閾値を超えた場合に CPA ホストとして検出することを提案する．また継続的な CPA ホストの検出を可能にするため，BF を 2 つ用意して交互に運用することを提案する．そして計算機シミュレーションにより，提案方式の有効性を確認する．以下，2 節では関連研究について，3 節では CPA について概略をまとめる．そして 4 節では提案方式の詳細を述べ，5 節で性能評価結果を示し，最後に 6 節で全体をまとめる．

2. 関 連 研 究

ICN を対象とした CPA の検知技術が数多く提案されている．例えば Yao らは，コンテンツを要求比率と要求の平均発生間隔に基づき，コンテンツを人気コンテンツと不人気コンテンツの 2 つにクラスタリングし，人気クラスタから不人気クラスタに移ったコンテンツ数の比率が閾値を超えた場合に CPA の発生を検出し，移ったコンテンツに対する Prefix を持つ配信要求 (Interest) に対してはコンテンツをキャッシュしないことを提案している [13]．また Guo らは攻撃者の Interest は特定のルータから生成されるため Interest の経路の多様性が低いことに着目し，要求が均一に発生する場合と比較した経路の多様性が閾値を下回る場合に検知する方式を提案している [6]．また Conti らは，ランダムにサンプルしたコンテンツセットに対し，各コンテンツの長期間における要求発生比率と，あるスナップショットにおける要求発生比率との差の総和が閾値を超えたときに CPA の発生を検知する方式を提案している [4]．

また Xu らは，CPA は同一の Prefix を用いて異なる多数のコンテンツが要求される点に着目し，複数のハッシュ関数とビットマップを用いて異なり数を計測する Flajolet-Martin sketch (FM Sketch) を用いて，異なり数が閾値を超えたときに CPA を検知する方式を提案している [11]．さらに Park らは，CPA 発生中はキャッシュされているコンテンツのランダム性が増加する点に着目し，コンテンツ名から得られるハッシュ値でキャッシュコンテンツを 2 値行列にマッピングし，高人気コンテンツを過去の 2 時点の行列との XOR をとって消去した行列のランクの統計的な量を閾値と比較することで CPA の発生を検知することを提案している [9]．しかしこれらの研究は ICN のルータのキャッシュに対する CPA に対象が限定されている．

一方，MEC のキャッシュに対する CPA を対象とした研究としては，Paul らの研究 [8] や Yang らの研究 [12] が見られる．Paul らは，HetNet の Femtocell に設置されたキャッシュを攻撃対象とする CPA の影響をシミュレーション評価している [8]．また Yang らは，CPA が Cooperative MEC の性能に与える影響を計算機シミュレーションにより分析し，攻撃者が全ノードをカバーする最小数のキャッシュに攻撃を行った場合の効果を明らかにしている [12]．しかしこれらの研究は CPA の影響の計算機シミュレーションによる評価結果を示すのに留まっている．

3. キャッシュポリューション攻撃

CPA には，Locality-disruption 型と False-locality 型の 2 種類の攻撃法が存在する [5]．一般にキャッシュの性能はコンテンツの人気の偏りが強いほど大きくなるため，Locality-disruption 型の CPA では，攻撃ホストは低人気のコンテンツに対して一様 に要求を発生させ，コンテンツの人気の偏りを緩やかにすることでキャッシュの効果を低下させる．一方 False-locality 型の CPA では，攻撃ホストは低人気のコンテンツに対して多数の要求を発生させコンテンツの人気の順位を入れ替えることで，正常ユーザの配信要求に対するキャッシュヒット率を低下させる．本稿の提案方式は，どちらの攻撃方法でも検出可能である．

本稿では CPA を一般化し，要求対象コンテンツを人気の最も低いものから C 個のコンテンツとする．ただしコンテンツ数を M とすると，$1 \leqq C \leqq M$ の範囲に C を設定する．文献 [2] において，筆者らは CPA ホストが要求するコンテンツの選択方法として，以下の 4 方式を定義した．

（1） Rand 型: CPA 対象コンテンツの中からランダムに選択したコンテンツを要求

（2） Smart fixed 型: 最も低人気のコンテンツから人気の昇順にコンテンツを要求

（3） Smart rand 型: CPA ホストごとにランダムに始点を決め，人気の昇順にコンテンツを要求

（4） Smart divide 型: 攻撃対象コンテンツ数/CPA ホストの間隔で等間隔に各 CPA ホストの要求開始点を決め，人気の昇順にコンテンツを要求

なお Smart 型の 3 方式においては，C 個の CPA 対象コンテンツの中で最も高人気のコンテンツを要求した次は，最も人気の低いコンテンツを要求対象とて選択し，以後，C 個のコンテンツを人気の昇順に選択する処理を反復する．

文献 [2] では，4 方式について計算機シミュレーションにて CPA の影響を比較したところ，Smart fixed 型は CPA ホスト数が増加するに伴いキャッシュヒット率の低下度合いが弱まる特性がみられた．Smart fixed 型以外の 3 つの CPA 方式では，キャッシュヒット率の低下効果は同等であり，攻撃対象コンテンツ数 C の広い領域で，また攻撃ホスト数が少ない領域で，その脅威が大きくなることを確認した．このことから本稿では，Smart 型 3 方式はキャッシュヒット率の低下効果が大きな Smart divide 型を，Smart 型と定義する．

4. 提案方式

4.1 Bloom Filter を用いた CPA ホストの検知

正常なホストは短い時間内に同一のコンテンツを複数回，視聴する可能性は低い．一方，CPA の攻撃ホストは短い時間内に同一コンテンツに対し複数回の視聴要求を発生させる．そのため各配信要求に対し，ホスト ID とコンテンツ ID の組をテーブルで管理し，同一ホストからの同一コンテンツに対する配信要求を検知すればよい．しかしホスト数とコンテンツ数の増加に伴い，管理テーブルに必要となるメモリ量とメモリアクセス回数が急増する．そこでメモリ量とアクセス回数を抑えながら効率的にキーの存在判定が可能である BF を採用する [1].

BF とは，空間効率のいい確率的データ構造である．k 個のハッシュ関数を用いて key に対する k 個のハッシュ値を生成する．生成されたハッシュ値に対応する bitmap 上の k 個の bit がすべて 1 であれば 1 度呼ばれた key，1 つでも 0 があれば 1 度も呼ばれていない key であると判別される．その後生成された k 個のハッシュ値に対応する bit を 1 にセットする．この仕組みのため，異なる key に対し同一のハッシュ値が生成され，新規のものが既出現と判断される偽陽性による誤検出の可能性がある．ホスト ID とコンテンツ ID をキーとして BF を用いて，キーの既出現性を判定することで，メモリサイズとメモリアクセス回数を抑えた CPA ホストの検知が可能となる．BF のサイズを M，偽陽性の目標エラー率を p とすると，BF への入力回数上限 N とハッシュ関数の個数 k は次式で与えられる．

$$N = -\frac{M(\log\ 2)^2}{\log\ p} \tag{1}$$

$$k = \frac{M}{N}\ \log\ 2 \tag{2}$$

筆者らが [1] で提案した方式では 1 つの bitmap のみを用いるため，ホスト ID とコンテンツ ID の組であるキーに対する bitmap のビットセットを反復していくにつれ，やがて bitmap のほとんどの位置が 1 で埋まり，全てのホストが CPA ホストとして検知されるため永続的な運用ができない．また，正常なユーザも同一コンテンツを複数回，視聴する可能性があり，また BF の偽陽性判定のより，CPA ホストとして誤検知される可能性がある．そこで次に，これら 2 つの問題に対処するための方法を述べる．

4.2 2 個の Bloom Filter の並列運用

BF を永続的に使用するため，BF 用に 2 つの Bitmap BF1 と BF2 を用意し，これらを切り替えながら運用することを提案する．各 Bitmap に挿入するキーの総数を N 以下に抑えることから，Bitmap への挿入キー数が N に達した時点で，もう一方の Bitmap に運用を切り替える．ただし一方の BF から他方の BF に切り替えた時点で，新規に稼働を始める BF の Bitmap の更新を開始すると，Bitmap の全てのビットが 0 の状態から始めることになり，CPA ホストの検知が困難である．そこで運用開始の閾値パラメタ α を導入し，運用中の Bitmap の挿入キー数が αN に達した時点で，もう一方の Bitmap へのキーの挿入も並行して行う．そのため $\alpha < 0.5$ のとき，3 個以上の Bitmap が必要になるので，$0.5 \leqq \alpha \leqq 1.0$ の範囲で α を設定する．以下の手順で 2 つの Bitmap を運用する．

（1） BF1 と BF2 を初期化 (全ビットをゼロに設定) した状態で運用を開始

（2） BF1 の挿入キー数が αN に達した時点で BF2 の更新も並行して開始

（3） BF1 の挿入キー数が N に達した時点で BF2 に運用を切り替え，BF 1 を初期化

（4） BF2 の挿入キー数が αN に達した時点で BF1 の更新も並行して開始

（5） BF2 の挿入キー数が N に達した時点で BF1 に運用を切り替え，BF2 を初期化

（6） (2)〜(5) を反復

図 1 に BF の並列運用を時系列で示す．ただし図 1(a) は $\alpha = 0.5$ の場合を，図 1(b) は $\alpha = 0.8$ の場合を各々示す．図中，BF1 (BF2) で青色 (黄色) に塗られた部分が各々の Bitmap の更新が行われている時間帯を表し，Main BF は CPA の検出に用いられる Bitmap を表す．図 1(a) に示すように，$\alpha = 0.5$ の場合は常時，両方の Bitmap が更新されるが，図 1(b) に示すように，$\alpha > 0.5$ の場合は一方の Bitmap のみ更新される時間が存在する．

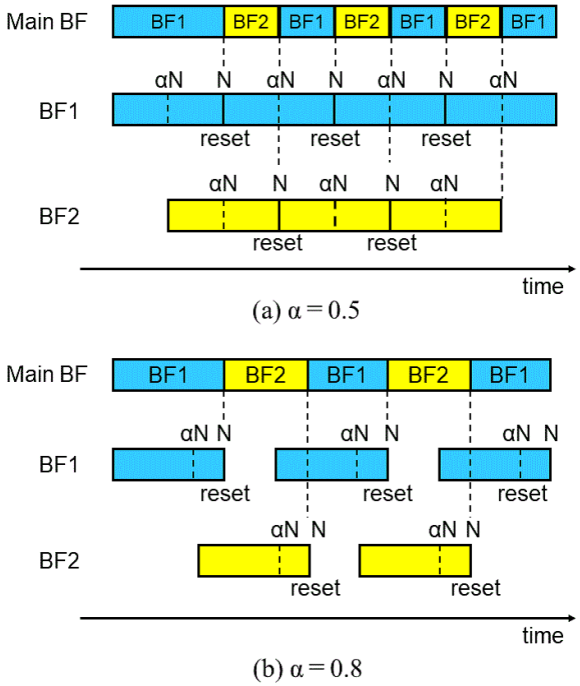

図 1 Diagram of updating two BF bitmaps

4.3 BF 検知回数の閾値の導入

正常ユーザの誤検知を減らすため，BF での検知回数に BF 検出閾値 Y を設け，Y 回検知されたホストを CPA として検知する．そのため BF 検知された各ホストの検知回数を記録するためのテーブル (BFDC: BF detection count) を用意し，ホスト u からの配信要求時には，まず BFDC の u に対する BF 検知回数 D_u をチェックし，$D_u < Y$ の場合はコンテンツ ID とホスト ID で BF 検索を行う．BF によって検知されなかった場合は，BF にコンテンツとホスト ID を記録し通常通りキャッシュを行う．BF によって検知された場合は，BFDC の D_u のカウンタをインクリメントする．$D_u = Y$ の場合は，キャッシュを行わない．

4.4 Bloom Filter の設計

Smart 型の CPA の場合，同一の CPA ホストが同一のコンテンツを要求するまでに，この CPA ホストから C 回の要求発生が必要である．一方で，提案方式の BF は N 回のキー入力まで受け付けることができるため，提案方式は BF による CPA ホストの検出を行うには，N が見たすべき下限値 N_{min}(検出リミット) が存在する．本節では N_{min} を導出する．全正常ユーザの毎秒要求数を R_n，全 CPA ユーザの毎秒要求数を R_c，CPA ホ

スト数を N_c とし，T を BF の 1 つの Bitmap にキーの挿入が開始されてからもう一方の Bitmap に切り替わるまでに要する時間とすると，

$$N_{min} = (R_c + R_n)T \tag{3}$$

となる．一方，1 つの CPA ホストから $C + Y$ 個の要求に対し，1 つの Bitmap にキーを挿入できれば CPA ホストの検知が可能なので，

$$\frac{R_c T}{N_c} = C + I \tag{4}$$

が得られる．(4) から得られる T を (3) に代入すると，

$$N_{min} = \frac{N_c}{R_c}(R_n + R_c)(C + I) \tag{5}$$

が得られる．(1) より，BF の 1 つの Bitmap に必要なメモリ量の下限値 M_{min} は，

$$M_{min} = -\frac{N_c(R_n + R_c)(C + I)\ \log\ p}{R_c(\log\ 2)^2} \tag{6}$$

となる．$R_n = 100$ (/seconds)，$R_c = 100$ (/seconds)，$N_c = 10$，$Y = 10$ とし，$C = 100$，500，1000，3000 としたときの M_{min} を，N_c と R_c に対し，図 2(a) と図 2(b) に各々プロットする．M_{min} は N_c の増加に対し線形に増加し，R_c の増加に対し反比例で減少する．

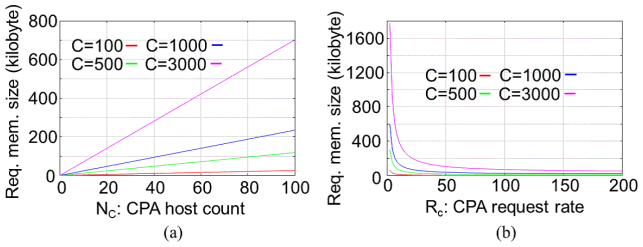

図 2 Required memory size against CPA host count and total request rate of CPA hosts

5. 性能評価

提案方式の有効性を計算機シミュレーションにより評価する．M 個のコンテンツに対し，人気の高いものから順に $1, 2, \cdots, M$ のラベリングがなされている．キャッシュ置換方式は LRU を想定し，コンテンツ数 $M = 10,000$ に対しキャッシュサイズは 500 とする．正常ホスト数 $N = 10,000$ で 6,000 秒間シミュレーションを行う．$N_c = 10$ 個の CPA 攻撃ホストが存在し，各々はコンテンツの人気順位を把握していることを想定し，2,500 秒から 5,800 秒の期間，攻撃ホストは CPA 対象コンテンツ $C = 1,000$ 個に対して，$M - C \leqq m \leqq M$ の範囲でコンテンツを要求する．CPA ホストは Smart 型を想定する．Smart 型攻撃は同一ホストが同一コンテンツを要求するための時間が最も長く提案方式で最も検出に時間がかかると考えられる攻撃方法である．

BF の運用開始パラメタ α を 0.5 に設定する．BF の各 Bitomap のメモリサイズは M_{min} より大きな 300KB に，BF の偽陽性の発生確率を $p = 0.01$ 設定し，式 (1)(2) に従い設計した BF の Bitmap を 2 つ用いる．BF の BF 検出閾値を $Y = 10$ とする．正常ホストはパラメタ $\theta = 0.7$ の Zipf 分布に従いランダムに選択したコンテンツを要求する．全 CPA ホストの総要求発生レートの，全正常ホストの総要求発生レートに対する比率を γ と定義し，特に断らない限り $\gamma = 1.0$ に設定する．結果は全て 10 回の施行における平均値で行う．

5.1 CPA の時系列特性

図 3 に，CPA ホスト数を $N_c = 60$ にしたときの，正常ホストのキャッシュヒット率の時系列グラフを示す．ただし，(i)CPA が発生しなかった場合，(ii)CPA が発生し提案方式を用いなかった場合，(iii)CPA が発生し提案方式を用いた場合，の 2 つの結果を各々示す．CPA が発生し提案方式を用いなかった場合，CPA の発生期間中，キャッシュヒット率の低下が見られる．しかし提案方式を用いた場合，一時的なキャッシュヒット率の低下が生じるものの，すぐに回復することが確認できる．

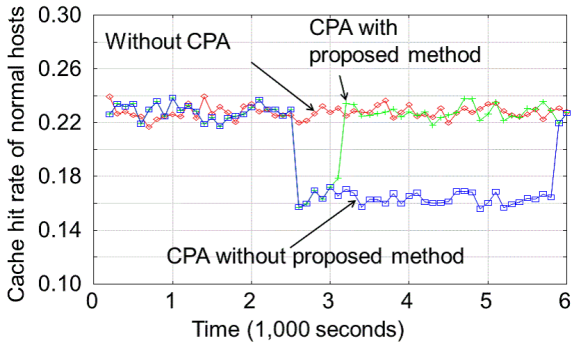

図 3 Time series of cache hit ratio of legitimate hosts under smart CPA

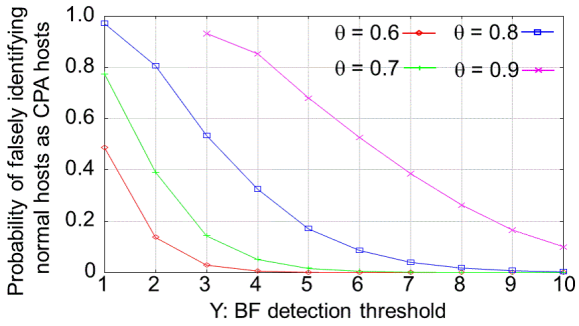

図 4 Probability of falsely identifying normal hosts as CPA hosts against threshold of BF detection

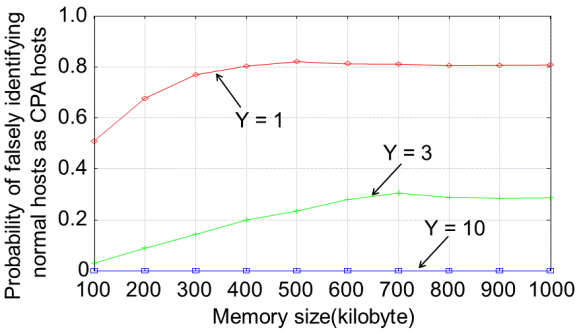

図 5 Probability of falsely identifying normal hosts as CPA hosts against memory size of each BF Bitmap

5.2 巻き添え率

BF は偽陽性により，一度も要求をしていないにもかかわら

ず要求があったと判断される False positive が発生する可能性がある．一方，同じ要求を繰り返したにもかかわらず検知されない False negative は発生しない．正常なホストが提案方式によって誤検出される割合を巻き添え率と定義する．巻き添えの原因は，BF の False Positive に加え，同一ホストが偶然，同一コンテンツを要求した場合にも発生する．

図 4 に，正常ホストの要求発生分布の偏りを決める Zipf パラメタ θ の 4 つの値に対し，BF 検出閾値 Y を変化させたときの巻き添え率をプロットする．θ に増加に伴い，要求対象コンテンツの偏りが増加するため，正常ホストの巻き添え率は増加する．また図 5 に，Y の 3 つの値に対し，BF の各 Bitmap に用いるメモリ容量を変化させたときの巻き添え率をプロットする．メモリサイズを増加させることで，より長時間，同一の Bitmap を CPA 検出に使用できるため，CPA の検出能力は向上するが，一方で正常ホストの巻き添え率が増加する．しかし BF 検出閾値 Y を増加させることで，巻き添え率を抑えることが可能である．例えば Y を 10 程度に設定することで，θ が 0.8 程度以下である場合は正常ホストの巻き添え率を 0.01 程度以下に抑えることが可能である．

5.3 検出時間の累積分布

次に，CPA が開始されてから各 CPA ホストの検出に要した時間を評価する．BF 検出閾値を $Y = 3, 10, 30$ の 3 つに設定し，Smart 型と Rand 型の CPA を各々実施した場合の，各 CPA ホストの検知に要した時間の累積分布を図 6 に示す．Rand 型と比較して Smart 型は CPA ホストが同一のコンテンツを要求するまでに要する時間が長いため，Smart 型 CPA の方が検出に要する時間が長い．各 CPA 方式においては，Y の増加に伴い CPA ホストの検出に要する時間が増加するが，Rand 型 CPA は Y の増加に伴う検出時間の増加程度が大きいが，Smart 型 CPA は $Y = 3$ の場合を除き，Y が増えても検出時間はほとんど増加しない．Smart 型では C のコンテンツを周回的に要求するため，2 回の同一コンテンツの要求が始まると連続して BF の検知がなされるためである．しかし Smart 型の $Y = 3$ の場合だけ，CPA ホストの検出時間が短い．これは，提案方式は BF の False positive のため CPA ホストが誤検知される結果と予想される．そのことを確認するため，図 7 に，BF の誤検知を考慮しない場合の CPA ホストの検知に要した時間の累積分布を同様に示す．Smart 型の $Y = 3$ の場合の結果が，Y の他の値の結果と，ほぼ同一になることが確認できる．

次に CPA 対象コンテンツ数 C を，500，1,000，3,000 に設定した場合の Smart 型 CPA と Rand 型 CPA における各 CPA ホストの検出に要した時間の累積分布を図 8 に示す．C の増加に伴い CPA ホスト検出に要する時間は増加するが，攻撃対象コンテンツ数が多くなるほど，CPA ホストが同一コンテンツを要求するまでに要する時間が増加するためである．Smart 型と Rand 型を比較すると，Smart 型の方が C の増加による影響を大きく受ける．

図 9 に，CPA ホスト数 N_c を，5，10，60 に設定した場合の Smart 型 CPA と Rand 型 CPA における CPA ホスト検出に要した時間の累積分布を同様に示す． N_c の増加に伴い CPA ホストの検出に要する時間が増加するが，CPA ホスト全体の要求発生レートを一定にしているため，N_c の増加に伴い各 CPA ホストの要求発生レートが減少し，同一コンテンツを 2 回，要求するまでの時間が増加するためである．CPA 対象コンテンツ数 C の増加と比較して，CPA ホスト数 N_c の増加が検出時間に与える影響が大きく，少量の CPA ホスト数の増加でも検出時間が大きく増加する．

次に，CPA ホストの総要求発生レートの正常ユーザの総要求発生レートに対する比率 γ を 0.5，1.0，1.5 に設定したときの，Smart 型 CPA と Rand 型 CPA における各 CPA ホストの検出に要した時間の累積分布を図 10 に示す．γ の減少に伴い CPA ホストの検出時間は増加するが，一方で，次節で述べるように γ の増加に伴いキャッシュヒット率の低下度合いは減少し，CPA の脅威は低下する．

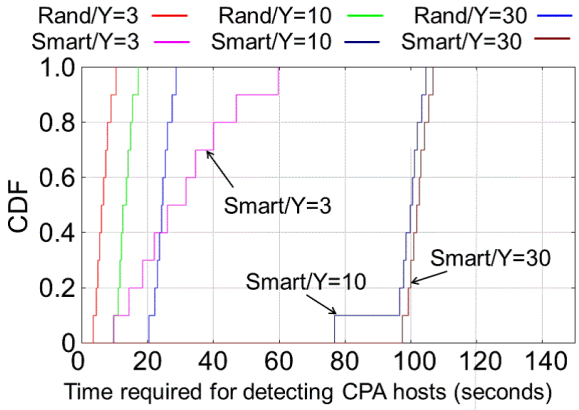

図 6 Cumulative distribution of time required for detecting each CPA host in various Y

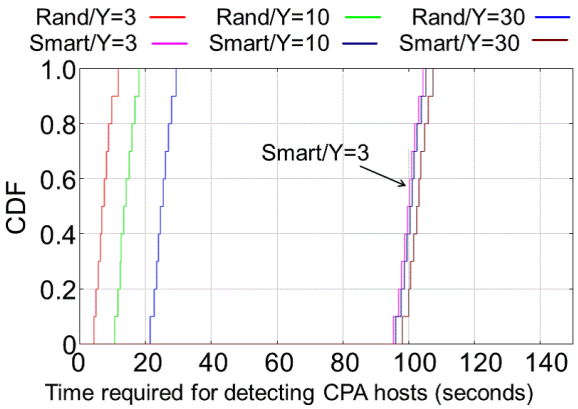

図 7 Cumulative distribution of time required for detecting each CPA host in various Y without considering detection of CPA hosts due to false positive of BFs

5.4 攻撃強度

本節では，CPA により生じた正常ホストのキャッシュヒット率の低下の影響度合い (攻撃強度) を評価する．図 3 の示すように，攻撃強度はキャッシュヒット率の低下量と，キャッシュヒット率の低下が継続した時間の長さの両方に依存し，各々が大きなほど攻撃強度も高くなる．そこで CPA の攻撃強度を，CPA が行われなかった場合のキャッシュヒット率と，CPA が行われた場合のキャッシュヒット率との差を，CPA が開始した時刻から，CPA ホストの検出に最も要した時間である 700 秒後の期間にわたり累積した値で定義する．表 1 に，提案方式を用いなかった場合と，提案方式で防御を行った場合の攻撃強度を，2 つの CPA 方式に対して各々，まとめる．ただし，パラメタの基本設定状態において，各パラメタを変更したときの結果を各々示す．提案方式を用いることで，最大で約 90 パーセント，攻撃強度を抑えることができる．CPA 対象コンテンツ数 C よりも，CPA ホスト数 N_c の増加により，攻撃強度が増加する度合いは大きい．CPA ホストの要求比 γ が小さなほど，

キャッシュヒット率の低下量は小さいが検出に要する時間が大きいため攻撃強度が大きくなる．

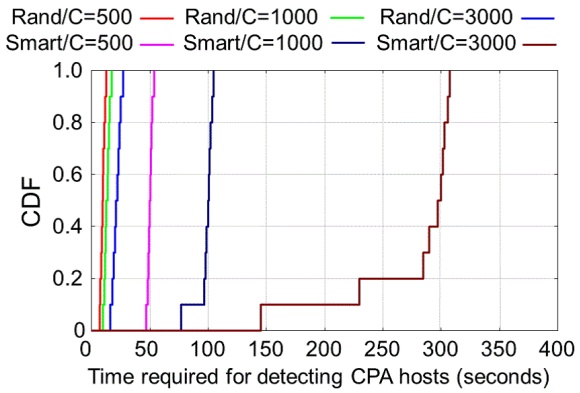

図 8　Cumulative distribution of time required for detecting each CPA host in various C

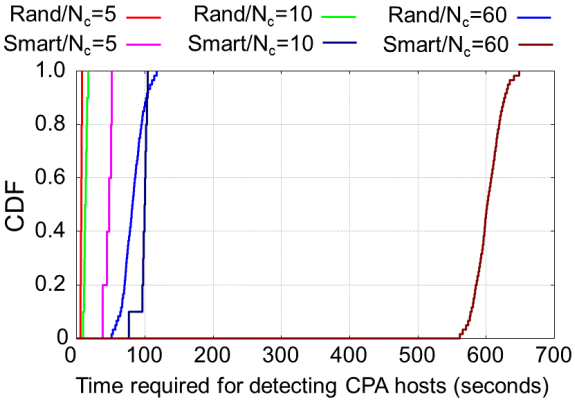

図 9　Cumulative distribution of time required for detecting each CPA host in various N_c

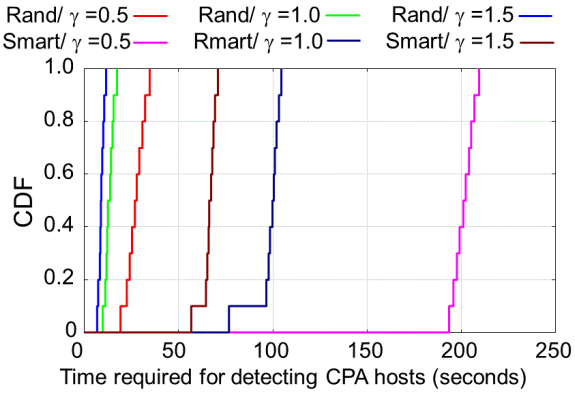

図 10　Cumulative distribution of time required for detecting each CPA host in various γ

6. 結　　論

本稿では，筆者らが以前提案した Bloom Filter (BF) を用いた CPA ホストの検出方式に対し，正常ホストの誤検知を避けるために BF の検知回数に閾値を設けることと，永続的な CPA ホストの検知を可能とするため 2 つの Bitmap を切り替えながら使用するという拡張を提案した．そして計算機シミュレーションによる数値評価により，例えば BF の検知閾値 Y を 10 程度に設定することで，正常ホストの巻き添え率を 0.01 程度以下に抑えながら，100 秒程度以内に CPA ホストを検知できることを確認した．しかし，BF の検出回数をリセットしない場合，長時間の運用後には正常ホストも検知される可能性が考えられる．そこで今後は，正常なホストが同一コンテンツを要求する傾向を分析し，BF 検出閾値のリセットを行う方式を検討する予定である．

謝辞　本研究成果は，JSPS 科研費 18K11283 の助成を受けたものである．ここに記して謝意を表す．

表 1　Attack strength of CPA

	Smart CPA	Rand CPA
Without proposed method	32.117	31.300
$Y = 3$	0.540	0.631
$Y = 10$	3.289	0.420
$Y = 30$	3.479	0.420
$C = 500$	0.262	0.420
$C = 1000$	3.289	0.420
$C = 3000$	14.416	0.420
$N_c = 5$	1.590	0.551
$N_c = 10$	3.289	0.420
$N_c = 60$	30.108	1.834
$\gamma = 0.5$	5.784	1.256
$\gamma = 1.0$	3.289	0.420
$\gamma = 1.5$	0.957	0.587

文　　献

[1] 芦原大和, 上山憲昭, ”MEC キャッシュにおけるキャッシュポリューション攻撃の検知”, 信学会 NS 研究会, NS2019-108, 2019 年 10 月.
[2] 芦原大和, 上山憲昭, ”MEC に対するキャッシュポリューション攻撃の影響分析”, 信学会 2020 年総合大会, B-6-32, 2020 年 3 月.
[3] J. Choi, J. Han, E. Cho, T. Kwon, and Y. Choi, “A Survey on Content-Oriented Networking for Efficient Content Delivery,” IEEE Commun. Mag., vol.49, no.3, pp.121-127, Mar. 2011.
[4] M. Conti, P. Gasti, and M. Teoli, A lightweight mechanism for detection of cache pollution attacks in Named Data Networking, Elsevier Computer Networks, vol. 57, no. 16, pp. 3178-3191, Nov. 2013.
[5] L. Deng, Y. Gao, Y. Chen, and A. Kuzmanovic, Pollution attacks and defenses for Internet caching systems, Computer Networks, 52, pp.935-956, 2008.
[6] H. Guo, X. Wang, K. Chang, and Y. Tian, Exploiting Path Diversity for Thwarting Pollution Attacks in Named Data Networking, IEEE Trans. Information Forensics and Security, vol. 11, no. 9, pp. 2077-2090, May 2016.
[7] J. Krolikowski, A. Giovanidis, and M. D. Renzo, Optimal Cache Leasing from a Mobile Network Operator to a Content Provider, IEEE INFOCOM 2018.
[8] S. Paul, A. Seetharam, A. Mukherjee, and M. K. Naskar, Investigating the Impact of Cache Pollution Attacks in Heterogeneous Cellular Networks, IEEE ICNP 2017.
[9] H. Park, I. Widjaja, and H. Lee, Detection of Cache Pollution Attacks Using Randomness Checks, IEEE ICC 2012.
[10] X. Wang, M. Chen, T. Taleb, et al., Cache in the Air: Exploiting Content Caching and Delivery Techniques in 5G Systems, IEEE Communications Magazine, vol. 52, no. 2, pp. 131-139, Feb. 2014.
[11] Z. Xu, B. Chen, N. Wang, Y. Zhang, and Z. Li, ELDA: Towards Efficient and Lightweight Detection of Cache Pollution Attacks in NDN, IEEE LCN 2015.
[12] C. Yang, H. Li, L. Wang, and D. Tang, Exploring the Behaviors and Threats of Pollution Attack in Cooperative MEC Caching, IEEE WCNC 2018.
[13] L. Yao, Z. Fan, J. Deng, X. Fan, and G. Wu, Detection and Defense of Cache Pollution Attacks Using Clustering in Named Data Networks, IEEE Trans. Dependable and Secure Computing, early access, Oct. 2018.