2013 IEEE International Conference on Cloud Computing Technology and Science

Background Knowledge-Resistant Traffic Padding for Preserving User Privacy in Web-Based Applications

Wen Ming Liu*, Lingyu Wang*, Kui Ren†, Mourad Debbabi*
*Concordia Institute for Information Systems Engineering,Concordia University, Canada
{l_wenmin,wang,debbabi}@ciise.concordia.ca
†Department of Computer Science and Engineering, State University of New York at Buffalo, USA
kuiren@buffalo.edu

***Abstract*—While enjoying the convenience of Software as a Service (SaaS), users are also at an increased risk of privacy breaches. Recent studies show that a Web-based application may be inherently vulnerable to side-channel attacks which exploit unique packet sizes to identify sensitive user inputs from encrypted traffic. Existing solutions based on packet padding or packet-size rounding generally rely on the assumption that adversaries do not possess prior background knowledge about possible user inputs. In this paper, we propose a novel random ceiling padding approach whose results are resistant to such adversarial knowledge. Specifically, the approach injects randomness into the process of forming padding groups, such that an adversary armed with background knowledge would still face sufficient uncertainty in estimating user inputs. We formally present a generic scheme and discuss two concrete instantiations. We then confirm the correctness and performance of our approach through both theoretic analysis and experiments with two real world applications.**

***Keywords*—Privacy Preservation, Indistinguishability, Uncertainty, Web Application, Traffic Padding, Side-Channel Attack, Background Knowledge**

I. Introduction

Today's Web applications allow users to enjoy the convenience of Software as a Service (SaaS) through their feature-rich and highly interactive user interfaces. However, recent studies show that such features may also render Web applications vulnerable to side channel attacks, which employ observable information, such as a sequence of directional packet sizes and timing, to recover sensitive user inputs from encrypted traffic [7]. Intrinsic characteristics of Web applications, including low entropy inputs, diverse resource objects, and stateful communications render such attacks a pervasive and fundamental threat in the age of cloud computing.

Existing countermeasures include *packet-size rounding* (increasing the size of each packet up to the closest multiple of given bytes) and *random padding* (increasing each packet size up to a random value). Those straightforward approaches have been shown to incur a high overhead and require application-specific implementation, while still not being able to provide sufficient privacy guarantee [7]. A more recent solution, *ceiling padding*, inspired by similar approaches in privacy-preserving data publication, partitions packets into *padding groups* and increases the size of every packet inside a group to the maximum size within that group in order to provide required privacy guarantee [16]. However, an important limitation shared by most existing solutions is that they assume adversaries do not possess any background knowledge about possible user inputs; the privacy guarantee may cease to exist when such knowledge allows adversaries to refine their guesses of the user inputs.

A natural way to address the above issue is to apply the well-known concept of differential privacy [8], which provides provable resistance to adversaries' background knowledge. Nonetheless, applying differential privacy to traffic padding will meet a few practical challenges. Specifically, introducing noises is more suitable for statistical aggregates (e.g., COUNT) or their variants, which have more predictable, and relatively small sensitivity; it is less applicable to traffic padding which has less predictable and often unbounded sensitivity (due to diverse resource objects), and individual packets' sizes, instead of their statistical aggregates, are directly observable. Moreover, while the qualitative significance of the privacy parameter ϵ is well understood in the literature, the exact quantitative link between this value and the degree of privacy guarantee is what an application provider would need to convince users about the level of privacy guarantee, which has received less attention. Therefore, the discussion of differential privacy is beyond the scope of this paper and is regarded as a future direction.

In this paper, we propose a novel *random ceiling padding* approach to providing background knowledge-resistant privacy guarantee to Web applications. We first adopt an information theoretic approach to modeling a padding algorithm's resistance to adversaries' prior knowledge about possible user inputs. Armed with this new *uncertainty* privacy metric, we then design a generic scheme for introducing randomness into the previously deterministic process of forming padding groups. Roughly speaking, the scheme makes a random choice among all the valid ways for forming padding groups to satisfy the privacy requirement. Consequently, an adversary would still face sufficient uncertainty even if s/he can exclude certain number of possible inputs to refine his/her guesses of the true input. We show that our proposed scheme may be instantiated in distinct ways to meet different applications' requirements by discussing two examples of such instantiations. Finally, we confirm the correctness (the algorithms provide sufficient privacy guarantee) and performance (the padding and processing cost), through both theoretic analysis and experiments with two real world Web applications.

The contribution of this paper is twofold. First, the proposed random ceiling padding approach may lead to practical solutions for protecting user privacy in real-life Web applications. As evidenced by our experimental results, the two padding algorithms instantiated from the generic approach can provide required privacy guarantee with reasonable costs. Second, although we have focused on the traffic padding issue in this paper, similar principles can be readily applied in other domains, such as privacy preserving data publication [12], in order to enhance syntactic privacy metrics [23], [19] with the capability of resisting adversarial background knowledge.

The rest of the paper is organized as follows. The remainder of this section builds intuitions through a running example. Section II defines our models. Section III first introduces a generic scheme and briefly discusses its privacy, costs, and computational complexity, then instantiates it into two concrete padding methods. Section IV experimentally evaluates the performance of our algorithms. We review related work in Section V, and conclude the paper in Section VI.

Motivating Example

Consider a fictitious website which, upon the login of a user, displays information about the disease with which s/he is most recently associated. Table I shows a toy example of sizes and directions of encrypted packets for the diseases starting with the letter *C*. Clearly, the fixed patterns of directional sizes of the first, second, and last packets will allow an adversary to pinpoint packets corresponding to different diseases from the observed traffic. In this example, if an adversary observes a s-byte value to be 360 when a patient logins, s/he can infer that the patient was likely diagnosed $Cancer$ (note this example is simplified to facilitate discussions, and the traffic pattern may be more complicated in reality).

Diseases	Observed Directional Packet Sizes			
$Cancer$	801 $\rightarrow$,	$\leftarrow$ 54,	$\leftarrow$ 360,	60 $\rightarrow$
$Cervicitis$	801 $\rightarrow$,	$\leftarrow$ 54,	$\leftarrow$ 290,	60 $\rightarrow$
$Cold$	801 $\rightarrow$,	$\leftarrow$ 54,	$\leftarrow$ 290,	60 $\rightarrow$
$Cough$	801 $\rightarrow$,	$\leftarrow$ 54,	$\leftarrow$ 290,	60 $\rightarrow$
			(s bytes)	

TABLE I. User Inputs and Corresponding Packet Sizes

We now examine two existing solutions, *rounding* [7] and *ceiling padding* [16], when applied to this example. Both solutions aim to pad packets such that each packet size will no longer map to a unique disease. In this example, we should pad s-byte such that each packet size maps to at least $k = 2$ different diseases, namely, 2-*indistinguishability*. In Table II, the third column shows that a larger rounding size does not necessarily lead to more privacy, since rounding with $\Delta = 112$ and 176 cannot achieve privacy (the s-value of *Cancer* after padding is still unique), whereas $\Delta = 144$ does. Therefore, we may be forced to evaluate many Δ values before finding an optimal solution, which is clearly an impractical solution.

Diseases	s Value	Rounding (Δ)			Ceiling Padding
		112	144	176	
$Cancer$	360	448	432	528	360
$Cervicitis$	290	336	432	352	360
$Cold$	290	336	432	352	290
$Cough$	290	336	432	352	290
Padding Overhead(%)		18.4%	40.5%	28.8%	5.7%

TABLE II. Rounding and Ceiling Padding for Table I

Next, the last column in Table II shows that the ceiling padding approach [16] achieves 2-indistinguishability. When an adversary observes a 360-byte packet, s/he can only infer that the patient has either *Cancer* or *Cervicitis*, but cannot be sure which is true. However, if the adversary happens to also possess some background knowledge through outbound channels that, say, this particular patient is a male, then it is obvious now that the patient must have *Cancer*, since, in this case, the adversary can further exclude the gynecological disease, *Cervicitis*.

In this paper, we will adopt a different approach to traffic padding. Instead of deterministically forming padding groups, the server randomly (at uniform, in this example) selects one out of the three possible ways for forming a padding group. Therefore, we can see that a cancerous person will always receive a 360-byte packet, whereas the other patients have $\frac{2}{3}$ and $\frac{1}{3}$ probability to receive a 290-byte and 360-byte packet, respectively, as shown in Table III.

Cancerous Person

Possible Padding Group	s Value (Padded)
{*Cancer, Cervicitis*}	360
{*Cancer, Cold*}	360
{*Cancer, Cough*}	360

Person Diagnosed with Cervicitis

Possible Padding Group	s Value (Padded)
{*Cervicitis, Cancer*}	360
{*Cervicitis, Cold*}	290
{*Cervicitis, Cough*}	290

TABLE III. Proposed Solution for Table I

To see why this approach provides better privacy guarantee, suppose an adversary observes a 360-byte packet and knows the patient to be a male. Under the above new approach, the adversary can no longer be sure that the patient has *Cancer*, because the following five cases will equally likely lead to a 360-byte packet to be observed. In the first three cases, the patient has *Cancer* and the server selects either *Cervicitis*, *Cold*, or *Cough* to form the padding group. In the other two cases, the patient has either *Cold* or *Cough*, respectively, while the server selects *Cancer* to form the padding group. Consequently, the adversary now can only be 60%, instead of 100%, sure that the patient is associated with *Cancer*.

II. The Model

We first describe our traffic padding model in Section II-A. We then introduce the concept of *uncertainty* in Section II-B and the *random ceiling padding* method in Section II-C. Finally we define our cost metrics in Section II-D. Table IV lists our main notations.

a, $\vec{a}$, A_i or A	Action, action-sequence, action-set
s_i, v, $\vec{v}$, V_i or V	Flow, flow-vector, vector-sequence, vector-set
VA_i or VA	Vector-action set
$\vec{VA}$	Vector-action sequence
$dom(P)$	Dominant-Vector

TABLE IV. The Notation Table

A. Traffic Padding

We model the traffic padding issue from two perspectives, the *interaction* between users and servers, and the *observation* made by adversaries. For the *interaction*, we call an atomic input that triggers traffic an *action*, denoted as a, such as a keystroke or a mouse click. We call a sequence of actions that represents a user's complete input information an *action-sequence*, denoted as $\vec{a}$, such as a sequence of consecutive keystrokes entered into a search engine. We also call the collection of all the i^{th} actions in a set of action-sequences an *action-set*, denoted as A_i.

Correspondingly, for the *observation*, we denote a *flow-vector* as v to represent a sequence of *flows*, $\langle s_1, s_2, \ldots, s_{|v|} \rangle$, that is, the sizes of packets triggered by actions. We denote a *vector-sequence* as $\vec{v}$ to represent the sequence of flow-vectors triggered by an action-sequence, and a *vector-set* as V_i corresponding to the action-set A_i. Finally, given a set of action-sequences and corresponding vector-sequences, we define all the pairs of i^{th} actions and corresponding i^{th} flow-vectors as the *vector-action set*, denoted as VA_i or simply VA when no ambiguity is possible. For a given application, we call the collection of all the vector-action sets *vector-action sequence*, denoted as $\vec{VA}$.

B. Privacy Properties

We model the privacy requirement of a traffic padding scheme from two perspectives. First, when adversaries observe a flow-vector triggered by a single action, they should not be able to distinguish this action from at least $k-1$ other actions that could have also triggered that same flow-vector, which is formalized in the following.

Definition 1 (k-Indistinguishability): Given a vector-action set VA, a padding algorithm $\mathcal{M}$ with output range $Range(\mathcal{M}, VA)$, we say $\mathcal{M}$ satisfies k-indistinguishability w.r.t. VA (k is an integer) if

$$\forall (v \in Range(\mathcal{M}, VA)), |\{a : Pr(\mathcal{M}^{-1}(v) = a) > 0 \wedge a \in A\}| \geq k.$$

Example 1: Assume that there are only four possible diseases in Table II, then the ceiling padding solution as shown in the right column satisfies 2-indistinguishability. $\boxdot$

In the previous section, we have illustrated how adversaries' background knowledge may help them to breach privacy even though the k-indistinguishability may already be satisfied. Therefore, our objective here is to first formally characterize the amount of *uncertainty* faced by an adversary about the real action performed by a user (we will then propose algorithms to increase such uncertainty in the next section). For this purpose, we apply the concept of entropy in information theory to quantify an adversary's uncertainty in the following.

Definition 2 (Uncertainty): Given a vector-action sequence $\vec{VA}$, a padding algorithm $\mathcal{M}$, we define

- the uncertainty of $v \in Range(\mathcal{M}, VA)$, where $VA \in \vec{VA}$, is defined as $\varphi(v, VA, \mathcal{M}) =$
$$-\sum_{a \in A} (Pr(\mathcal{M}^{-1}(v) = a) log_2(Pr(\mathcal{M}^{-1}(v) = a));$$
- the uncertainty of algorithm $\mathcal{M}$ w.r.t. VA is defined as
$$\phi(VA, \mathcal{M}) = \sum_{v \in Range(\mathcal{M}, VA)} \varphi(v, VA, \mathcal{M}) \times Pr(\mathcal{M}(A) = v);$$
- the uncertainty of algorithm $\mathcal{M}$ w.r.t. $\vec{VA}$ is defined as
$$\Phi(\vec{VA}, \mathcal{M}) = \prod_{VA \in \vec{VA}} (\phi(VA, \mathcal{M}));$$

Example 2: To illustrate the above notions, following Example 1, the uncertainty of the flow 360, denoted as $\varphi(360, VA, \mathcal{M})$ (or simply $\varphi(360)$ hereafter when no ambiguity is possible) can be calculated as $\varphi(360) = -(\frac{1}{2} log_2(\frac{1}{2}) + \frac{1}{2} log_2(\frac{1}{2})) = 1$. Similarly, we have $\phi(VA) = \frac{\varphi(360)}{2} + \frac{\varphi(290)}{2} = 1$. Further, since the vector-action sequence is composed of a single vector-action set, $\Phi(\vec{VA}) = \phi(VA) = 1$. $\boxdot$

Finally, we model the privacy of a padding algorithm as its joint capabilities of satisfying k-indistinguishability and δ-uncertainty. Note that here the former serves as a basic privacy requirement (when no resistance to background knowledge is needed) while the latter can be regarded as an enhanced requirement. Both parameters may be adjusted according to different applications' unique requirements for privacy.

Definition 3 (δ-uncertain k-indistinguishability): An algorithm $\mathcal{M}$ gives δ-uncertain k-indistinguishability for a vector-action sequence $\vec{VA}$ if

- $\mathcal{M}$ w.r.t. any $VA \in \vec{VA}$ satisfies k-indistinguishability, and
- the uncertainty of $\mathcal{M}$ w.r.t. $\vec{VA}$ is not less than δ.

C. Padding Method

To be more self-contained, we first review the *ceiling padding* method [15], [16]. The method deterministically partitions a vector-action set into padding groups, each of which has a cardinality no less than k, and then breaks the linkage among actions in the same group by padding the flows to be identical, as described in the following.

Definition 4 (Dominance and Ceiling Padding[16]): Given a vector-set V, we define

- the *dominant-vector* $dom(V)$ as the flow-vector in which each flow is equal to the maximum of the corresponding flow among all the flow-vectors in V.
- a *ceiling-padded* group in V as a padding group in which every flow-vector is padded to the dominant-vector.

Clearly, the ceiling padding method is only designed to achieve the k-indistinguishability, and will not provide sufficient privacy guarantee if the adversary possesses prior background knowledge.

In this paper, we propose to introduce randomness into the process of forming padding groups per each user request. Specifically, to response to an action, we first select at random, from certain distributions, $k-1$ other actions to form the padding group. Then, we apply ceiling padding on the resultant group. To differentiate from the aforementioned fixed padding group and the original ceiling padding method, we call the group formed on the fly with randomness the *transient group*, and our method the *random ceiling padding* in the following.

Definition 5 (Transient Group and Random Ceiling Padding): We say a mechanism $\mathcal{M}$ is a *random ceiling padding* method if, when responding to an action a, it randomly selects $k-1$ other actions and pads the flow-vector of action a to be the dominant-vector among the corresponding flow-vectors of selected $k-1$ actions together with the original flow-vector of action a. We also call those k actions a *transient group*.

Example 3: To achieve 2-indistinguishability, a mechanism $\mathcal{M}$ selects uniformly at random 1 other action to form the transient group (Table II). Then, the following two cases could both lead to an observed $s = 360$ flow. First, the patient has $Cancer$ and $\mathcal{M}$ selects any one of the others to form the group (there are 3 possible transient groups in this case). Second, the patient does not have $Cancer$ but has one of the other threes, and $\mathcal{M}$ selects $Cancer$ to form the group. Each of them has only one possible transient group. Thus, $\varphi(360) = -(\frac{1}{2}log_2(\frac{1}{2}) + 3 \times \frac{1}{6}log_2(\frac{1}{6})) \approx 1.79$.

Now, if the adversary knows that the patient can not have $Cervicitis$ and observes the s-byte value to be 360, s/he will no longer be able to infer which disease the patient has. Formally, in this case, $\varphi(360) = -(\frac{3}{5}log_2(\frac{3}{5}) + 2 \times \frac{1}{5}log_2(\frac{1}{5})) = 1.37$. $\boxdot$

D. Cost Metrics

In addition to privacy requirements, we also need metrics for the communication and processing costs. For the former, we measure the proportion of packet size increases compared to the original flow-vectors. For the latter, we measure how many flow-vectors need to be padded among all the vectors in a $\vec{VA}$, as formalized in Definition 6 and 7, respectively.

Definition 6 (Expected Padding Cost): Given a vector-action sequence $\vec{VA}$, an algorithm $\mathcal{M}$,

- the *expected padding cost* of action a in $(a, v) \in VA$ where $VA \in \vec{VA}$ is defined as $pcos(a, VA, \mathcal{M}) =$

$$|| \sum_{v' \in Range(\mathcal{M}, VA)} (Pr(\mathcal{M}(a) = v') \times v') - v||_1;$$

- the *expected padding cost* of a vector-action set $VA \in \vec{VA}$ under algorithm $\mathcal{M}$ is defined as

$$pcos(VA, \mathcal{M}) = \sum_{(a,v) \in VA} (pcos(a, VA, \mathcal{M}));$$

- the *expected padding cost* of the vector-action sequence $\vec{VA}$ is defined as

$$pcos(\vec{VA}, \mathcal{M}) = \sum_{VA \in \vec{VA}} (pcos(VA, \mathcal{M}));$$

Definition 7 (Expected Processing Cost): The *expected processing cost* of a vector-action sequence $\vec{VA}$ under an algorithm $\mathcal{M}$ is defined as

$$rcos(\vec{VA}, \mathcal{M}) = \frac{\sum_{VA \in \vec{VA}} \sum_{(a,v) \in VA} (Pr(\mathcal{M}(a) \neq v))}{\sum_{VA \in \vec{VA}} |\{(a, v) : (a, v) \in VA\}|};$$

Surprisingly, while introducing randomness into the process of forming padding groups improves the privacy, this improvement does not necessarily come at a higher cost, as shown in Example 4 (we will only compare the cost with the original ceiling padding method hereafter, since ceiling padding has much less overhead than other methods, such as rounding, as shown in our previous work [16]).

Example 4: For ceiling padding shown in last column of Table II, the expected padding cost can be calculated as $pcos(VA, ceiling\ padding) = 70$, and the expected processing cost as $rcos(VA, ceiling\ padding) = 25\%$.

On the other hand, for the random ceiling padding $\mathcal{M}$ shown in Example 3, we have $pcos(VA, \mathcal{M}) = (360–360) + 3 \times ((\frac{1}{3} \times 360 + \frac{2}{3} \times 290) – 290) = 70$ and $rcos(VA, \mathcal{M}) = \frac{0+3\times\frac{1}{3}}{4} = 25\%$.

That is, these two methods actually lead to exactly the same expected padding and processing costs, while the latter clearly achieves higher uncertainty (with the same k-indistinguishability). $\boxdot$

III. The Algorithms

We first introduce a generic random ceiling padding scheme in Section III-A, and then discuss two example ways for instantiating the scheme into concrete algorithms in Section III-B. The main intention here is to show that the random ceiling padding method can potentially be instantiated in many different ways based on specific applications' needs. In the coming sections, we will further show that even those straightforward ways we describe here can still achieve good performance in terms of the privacy guarantee and the costs.

A. The Random Ceiling Padding Scheme

The main idea of our scheme is the following. In responding to a user input, the server will form a transient group on the fly by randomly selecting members of the group from certain candidates based on certain distributions (different choices of such candidates and distributions will lead to different algorithms, as demonstrated in Section III-B).

Our goal is two-fold. First, the privacy properties, k-indistinguishability and δ-uncertainty, need to be ensured. Second, the costs of achieving such privacy protection should be minimized. Clearly, a trade-off naturally exists between these two objectives. We will demonstrate how to address this trade-off through two instantiations of the general scheme with different methods of forming transient groups.

a) Outline of Scheme

The generic random ceiling padding scheme consists of two stages as shown in Table V. The first stage, a one-time process, derives the randomness parameters and accordingly determines the probability of an action being selected as a member of a transient group. Both δ and costs are related to k (which is considered as a given constant), the vector values and their cardinalities (which is determined for a given vector-action set), and the parameters of distribution from which the randomness is drawn. Clearly, to determine the randomness parameters such that δ is not less than a desired value while keeping the costs minimal is naturally an optimization problem. In this paper, we simplify the process by setting the size of each transient group to k to ensure the indistinguishability (the proof is omitted due to space limitations). Once the randomness parameters are set, then repeatedly, upon receiving an action a_0, the second stage selects, randomly following the results of stage one, $k - 1$ other actions from the corresponding action-set A of vector-action set VA to form the transient group, and then returns the dominant-vector of this transient group.

The computational complexity of random ceiling padding algorithm is $O(k)$ due to the following. First, the first stage of our scheme can be pre-calculated only once, when the vector-action set is given, and does not need to be repeatedly

Stage 1: One-Time Preprocessing	**Stage 2:** Real-Time Response
Input: the vector-action set VA, the privacy properties k_{min} and δ_{min}, the randomness generator G; **Output:** the parameters $\langle P \rangle$ of G **Method:** 1. **Let** V be the vector-set of VA, and A be the action-set of VA; 2. **If** $(\lvert VA \rvert \leq k_{min})$ **Return**; 3. Compute the distribution D_V of V; 4. Compute $\langle P \rangle$ based on its relation with δ, k, $pcos$, $rcos$, D_V when random ceiling padding is applied, such that (1). $k \geq k_{min}$ and $\delta \geq \delta_{min}$; (2). $pcos$ and $rcos$ are minimal; 5. **Return** $\langle P \rangle$;	**Input:** the vector-action set VA, the randomness parameters $\langle P \rangle$ of G, the privacy properties k_{min} and δ_{min}, the action a_0 **Output:** the flow-vector $v_0^{'}$; **Method:** 1. **Let** V be the vector-set of VA, and A be the action-set of VA; 2. Create A_C by randomly selecting $k_{min}-1$ actions from the subset of A based on $\langle P \rangle$ of G; 3. $A_C = A_C \cup \{a_0\}$; 4. **Let** V_C be the subset of vector-set V which corresponds to A_C; 5. **Return** the dominant-vector of V_C;

TABLE V. THE RANDOM CEILING PADDING SCHEME

evaluated each time when the scheme is invoked to respond to an action. Therefore, although it runs in polynomial time of N (N as the cardinality of the vector-action set), for continuous execution of the algorithm, the computational complexity for responding to each action is still $O(1)$. Second, to select k random actions without duplicates, Line 2 of second stage can be realized in $O(k)$ time with many standard algorithms. Finally, it takes $O(k)$ times to select the corresponding vector-set for the selected actions in Lines 3-4 and calculate their dominant-vector in Line 5.

b) Discussion on Privacy and Costs

We briefly discuss the privacy and costs of our scheme in the following. Note that, the formulations for uncertainty and costs related to the aforementioned parameters are not necessary to be straightforward. These formulations and the corresponding proofs are omitted due to space limitations and the more detailed analysis can be found in our technical report [17].

In privacy preserving traffic padding (PPTP), the adversary cannot collect the vector-action set even if s/he acts as a normal user of the application using random ceiling padding technique. The reason is as follows. First, the sample space with respect to the set of all possible transient groups is huge even for small-size vector-action set with reasonable k value. For example, when $|VA| = 100$ and $k = 20$, the cardinality of sample space for each action equals to $\binom{99}{19} \approx 2^{66}$, each of which is equally likely to be the resultant transient group per each user request when the randomness is drawn from a uniform distribution [21]. Second, since all users share one random process in the scheme, the distribution of events cannot be sufficiently approximated by collecting flow-vector values for a special action just as many times as the cardinality of its sample space.

In other words, in random ceiling padding, an action cannot be distinguished from at least other $k-1$ different actions based on the traffic triggered, which satisfies k-indistinguishablility. Moreover, the adversary cannot deterministically infer the action only by the observation even s/he can further remove a limited number of actions based on prior knowledge.

On the other hand, we have shown in previous examples that the padding cost and the processing cost of ceiling padding and random ceiling padding is deterministically incomparable. That is, these costs cannot simply be ordered based on the algorithms themselves and will depend on specific vector-action sets.

B. Instantiations of the Scheme

In this section, we discuss two example instantiations of the proposed random ceiling padding scheme, and illustrate two different ways for reducing the padding and processing costs while satisfying the privacy requirements. Basically, the two instantiations differ in their ways of selecting candidates for members of the transient group, in order to reduce the cost. First, to facilitate discussions, we introduce two new notions.

In Definition 8, intuitively, function $f_v(.)$ sorts a vector-action set VA based on the padding cost, and we denote the resultant totally ordered set (chain) under the binary relation $\succcurlyeq_v$ by $\langle VA \rangle_v$. The main objective of this step is to adjust the probability of an action being selected as a member of the transient group, in order to reduce the expected costs. Besides, in the case that each flow-vector in V includes a single flow, the flows (integers) ordered by the standard larger-than-or-equal relation $\geq$ is also a chain that is naturally identical for each v. Therefore, although the chain $\langle VA \rangle_v$ for different $v \in V$ may be different, in the rest of this paper, we will use a single chain (simplified as $\langle VA \rangle$) for analysis and experiment.

Definition 8 (Larger and Closer): Given a vector-action set VA, a pair $(a, v) \in VA$, define a function $f_v : V \to I$ (I for integers) as $f_v(v') = ||dom(\{v, v'\}) - v||_1$. Then, we say, w.r.t. (a, v),

- $(a', v') \in VA$ is larger than $(a'', v'') \in VA$, denoted by $(a', v') \succcurlyeq_v (a'', v'')$, if $f_v(v') > f_v(v'') \vee ((f_v(v') = f_v(v'')) \wedge (a' \succ a''))$, where $\succ$ is any predefined order on the action-sets;
- $(a', v') \in VA$ is closer to (a, v) than $(a'', v'') \in VA$ if $|f_v(v')| < |f_v(v'')|$.

a) Option 1: Randomness from Bounded Uniform Distribution

The step 2 of stage 2 in Table V may be realized in many different ways by choosing group members from different subsets of candidates and based on different distributions. Note that although choosing the members uniformly at random from all possible candidates certainly leads to highest possible uncertainty, this also will incur prohibitive processing cost.

In fact, the uncertainty of an algorithm can be dramatically increased even by a slight increase in the cardinality of possible candidates for forming the transient group.

This first option draws candidates from a uniform distribution. It also allows users to constraint the cardinality of candidate actions to be considered (c_t) and the number of such actions that are *larger* than given action (c_l). More specifically, given a vector-action set VA, and a pair (a_i, v_i) being the i^{th} pair of its corresponding chain $\langle VA \rangle$, the transient group of (a_i, v_i) will be selected uniformly at random from the sub-chain of the chain in the range of $[max(0, min(i-c_l, |VA|-c_t)), min(max(0, min(i-c_l, |VA|-c_t))+c_t, |VA|)]$ (complete algorithms will be omitted due to space limitations) .

The action in a transient group which is in the least position of the chain $\langle VA \rangle$ will determine the padding cost of (a, v) when a is performed. Thus, from this perspective, c_l should be as small as possible. However, c_l should also be sufficiently large. For example, if $c_l = 0$, each action should be deterministically padded. Moreover, the c_t determines the cardinality of possible transient groups. More possibilities of transient groups will complicate adversaries' tasks in attacking (collecting the data of directional packet sizes and analyzing the distribution of flow-vector information).

b) Option 2: Randomness from Normal Distribution

In this option, the action *closer* to a in the chain has higher probability to be selected as a member of a's transient group. To select a member, the distance between the selected action and the performed action a in the chain $\langle VA \rangle$ (that is, the difference of the positions) is drawn from normal distribution (rounded up to the nearest integer).

When the mean of normal distribution is set to zero, the two actions with equal distance in the both sides of the performed action a in the chain are equally likely selected. As mentioned before, the action in transient group with least position in the chain $\langle VA \rangle$ determines the padding cost. Thus, the mean can be adjusted to a positive integer, such that the actions in larger positions than a would have a higher chance to be selected, and consequently the expected cost will be reduced.

In addition, since increasing the standard deviation flattens the curve of the distribution and allows more chances to draw a value far from the mean, it yields a higher probability to select an action farther away from the performed one in the chain $\langle VA \rangle$. Thus, in practice, the standard deviation should be small enough to reduce the padding cost; it also should not be too small in order to prevent the adversary from collecting the data and analyzing the distribution of flow-vector values.

IV. EXPERIMENT

In this section, we evaluate the uncertainty and the cost under two implementation options of our scheme (see Section III) through experiments with two real world Web applications. First, Section IV-A describes the experimental settings, and then Section IV-B, IV-C, and IV-D present experimental results for randomness drawn from bounded uniform distribution and normal distribution, respectively.

A. Experimental Setting

We collect testing vector-action sets from two real-world web applications, a popular search engine $Engine$ (where users' searching keyword needs to be protected) and an authoritative drug information system $Drug$ (where users' health information needs to be protected). Specially, for $engine$, we collect flow-vectors with respect to query suggestion widget by crafting HTTP(S) requests to simulate the normal AJAX connection request. Note that infinite number of queries in a search engine does not mean infinite data, because a search engine will no longer provide autosuggestion feature once the query string exceeds certain length, so the number of action-sequences is still finite. For $drug$, we collect the vector-action set for all the drug information by mouse-selecting following the application's tree-hierarchical navigation. Such data can be collected by acting as a normal user of the applications without having to know internal details of the applications. For our experiment, these data are collected using separate programs whose efficiency is not our main concern in this paper.

We observe that the flows of $drug$ are more diverse than those of $engine$ evidenced by the standard deviations (σ) of the flows. The flows of $drug$ are also larger than those of $engine$ based on their means (μ). Besides, the flows of $drug$ are much more disparate in values than those of $engine$. These differences of flows will affect the costs as will be shown later.

We compare our solutions with the svmdGreedy (short for SVMD) [16] on four-letter combinations in $Engine$ and the last-level data in $Drug$ due to the following. First, one vector-action set is sufficient to demonstrate the results. Thus, we use a single vector-action set instead of vector-action sequence. Second, as reported in [16], rounding and random padding [7] lead to even larger overheads while they cannot ensure the privacy. Thus, we compare our results with SVMD only.

In the first option (see Section III-B), namely, *TUNI* option, we constraint the number of larger actions (c_l) and the minimal number of possible actions to be selected (c_t) when the probability of an action to be selected is drawn from uniform distribution. In the meantime, in the second option, namely, *NORM* option, we allow to adjust the mean (μ) and standard deviation (σ) when it is drawn from normal distribution.

All experiments are conducted on a PC with 2.20GHz Duo CPU and 4GB memory and we conduct each experiment 1000 times. To facilitate the comparisons, we use padding cost ratio, processing cost ratio to measure the relative overheads of the padding cost and processing cost, respectively.

B. Uncertainty and Cost v.s. k

The first set of experiments evaluates the uncertainty and cost of *TUNI* and *NORM* options against *SVMD*. Figure 1(a), (b), and (c) illustrate the padding cost, uncertainty, and processing cost against the privacy property k, respectively. In general, the padding and processing costs of all algorithms increase with k, while *TUNI* and *NORM* have less costs than those of *SVMD*. Meanwhile, our algorithms have much larger uncertainty for *Drug* and slightly larger for *Engine*.

C. Randomness Drawn from Bounded Uniform Distribution

Figure 2 illustrates the uncertainty and cost of *TUNI* option on both vector-action sets against the top limit c_l. As expected,

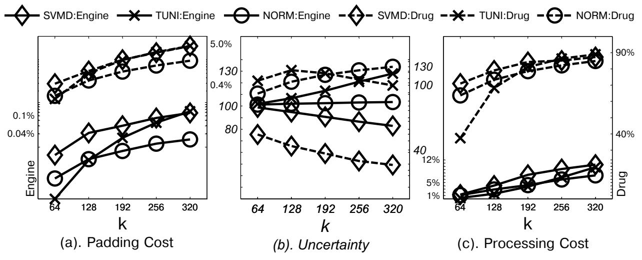

Fig. 1. Uncertainty and Cost Against k

SVMD is insensitive to c_l since it does not have the concept of c_l. On the other hand, both costs increase slowly with c_l for *TUNI*. This is because, larger c_l allows the algorithm to have more chances to select *larger* actions for transient group. The largest action in the transient groups determines the padding cost in this case, and a single *larger* action leads to an increase of processing cost. From the results, *TUNI* performs worse on *Drug* than on *Engine* w.r.t. cost. This is because, the more diverse in the flow of *Drug* leads to more chances to select *larger* action even with a small increase of c_l. Despite the slight increase of cost with c_l, *TUNI* generally has less cost and larger uncertainty than *SVMD* for both vector-action sets.

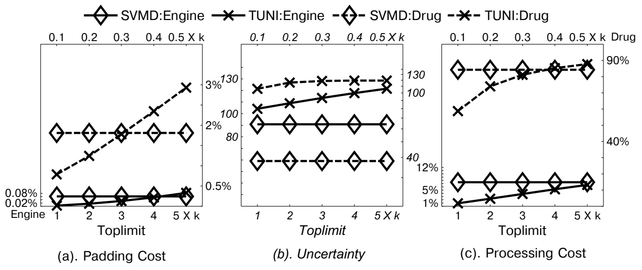

Fig. 2. Uncertainty and Cost for Bounded Uniform Distribution Against Top Limit

Figure 3 shows the uncertainty and cost against the minimal cardinality c_t. Similarly, *SVMD* is insensitive to c_t due to the same reason. Also, *TUNI* demonstrates same results on *engine* in terms of both uncertainty and cost regardless of the value of c_t. This is because, the constraint of minimal cardinality works only when the cardinality of possible actions is less than c_t after applying c_l parameter. In *engine*, the number of actions which have the smallest flow value is extremely larger than the c_t values in the experiment. In other words, c_t does not affect the results. For *drug*, the padding and processing costs increase slowly with c_t while the uncertainty decreases slowly.

D. Randomness Drawn from Normal Distribution

Figure 4 illustrates the uncertainty and cost of *NORM* option on both vector-action sets against the mean (μ) of normal random function. Compared with *SVMD*, *NORM* has less cost and yet higher uncertainty. The mean values do not affect the uncertainty and cost of *SVMD* since it does not take mean as a parameter. On the other hand, the cost of *NORM* decreases almost linearly with the increase of mean from 0 to 16, and rapidly as μ grows to 32 on both vector-action sets. In the meanwhile, the uncertainty of *NORM* slightly changes for

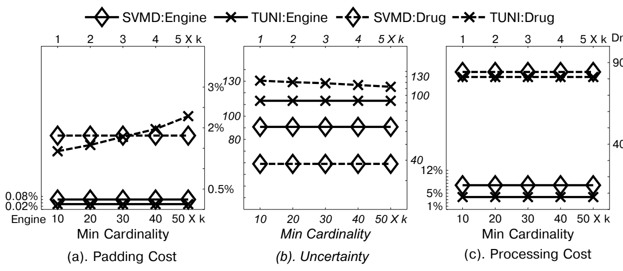

Fig. 3. Uncertainty and Cost for Bounded Uniform Distribution Against Minimal Cardinality

the mean from 0 to 16, and decreases rapidly when $\mu = 32$. This is because, when $\mu = 32$, *NORM* has negligible chance to select a *larger* actions for the group. In other words, the vectors need not to be padded in most cases. Thus, in practice, we must tune the parameters (μ and σ) to avoid such situation.

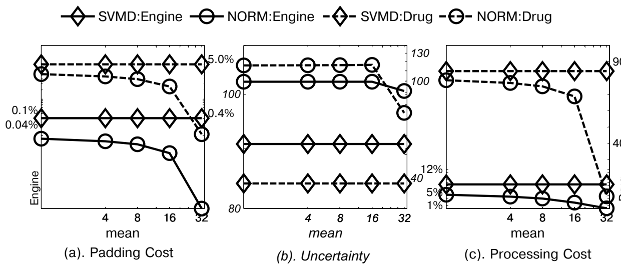

Fig. 4. Uncertainty and Cost for Normal Distribution Against Mean

Figure 5 shows the uncertainty and cost against the standard deviation σ of normal random function. Basically, all the three measurements decreases with the decrease of σ. Compared with *SVMD*, the less the σ, *NORM* exhibits better. This is as expected since the larger the standard deviation is, the flatter the curve of normal distribution is, and consequentially, the more chances to draw a value far from the mean, which is equal to select an action far from the performed one.

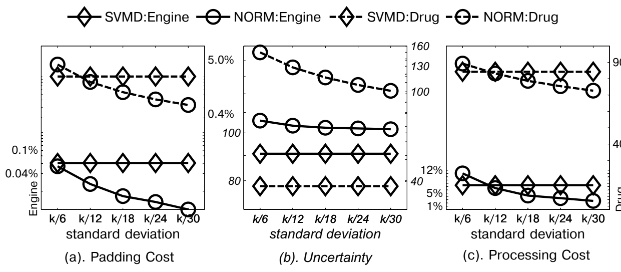

Fig. 5. Uncertainty and Cost for Normal Distribution Against Standard Deviation

V. Related Work

The privacy preserving issue has received significant attentions in different domains, including micro-data release [12], [23], [8], mobile network [11], social network [10], [20], and other web applications [4], [7], [27], [26]. In the context of privacy-preserving data publishing, since the introduction of the k-anonymity concept [23], much effort has been made on developing efficient privacy-preserving algorithms [1], [23], [13]. Meanwhile many other models are also proposed to enhance the k-anonymity model, such as l-diversity [19], t-closeness [14], and differential privacy [8].

Various side-channel leakages have been studied extensively in the literature. By measuring the amount of time taken to respond to decrypted queries, an attacker may extract OpenSSL RSA privacy keys [5]. By differentiating the sounds produced by keys, an attacker may recognize the key pressed [3]. Ristenpart *et al.* discovers cross-VM information leakage on Amazon EC2 [22]. Search histories may be reconstructed by session hijacking attack [6], while web-browsing histories may be compromised by cache-based timing attacks [9]. Saponas *et al.* show the transmission characteristics of encrypted video streaming may allow attackers to recognize the title of movies [24]. Meanwhile, much efforts have been made on developing techniques to mitigate the threats of such leakages. Countermeasures are suggested against the exposure of identification of encrypted web traffic [25]. HTTPO are proposed to prevent information leakages of encrypted HTTP traffic [18]. Timing mitigator is introduced to achieve any given bound on timing channel leakage [2]. Zhang *et al.* present an approach to verifying the VMs' exclusive use of a physical machine [28]. We have reported our privacy model and method in privacy-preserving traffic padding [16], in this paper, we enhance our previous works by mitigating the threats of background knowledge.

VI. Conclusion

In this paper, we have proposed a solution to reduce the impact of adversaries' background knowledge in privacy-preserving traffic padding. The approach can potentially be applied to other privacy-preserving issues, although we have focused on the traffic padding issues in this paper. We have also instantiated two algorithms by following the proposed solution. Our experiments with real-world applications confirmed the performance of the proposed solution in terms of both privacy and overheads. Our future work will apply the proposed approach to privacy-preserving data publishing to achieve syntactic privacy while mitigating the threat of adversaries' prior-knowledge.

Acknowledgment

The authors thank the anonymous reviewers for their valuable comments. This material is based upon work partially supported by Natural Sciences and Engineering Research Council of Canada under Discovery Grant N01035, Strategic Project Grant STPGP/396651-2010, and Canada Graduate Scholarship, and by US National Science Foundation through grant CNS-1117811. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsoring organizations.

References

[1] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Anonymizing tables. In *ICDT '05*, pages 246–258, 2005.

[2] A. Askarov, D. Zhang, and A. C. Myers. Predictive black-box mitigation of timing channels. In *CCS '10*, pages 297–307, 2010.

[3] Dmitri Asonov and Rakesh Agrawal. Keyboard acoustic emanations. *Security and Privacy, IEEE Symposium on*, page 3, 2004.

[4] M. Backes, G. Doychev, M. Dürmuth, and B. Köpf. Speaker recognition in encrypted voice streams. In *ESORICS '10*, pages 508–523, 2010.

[5] David Brumley and Dan Boneh. Remote timing attacks are practical. In *USENIX '03*, pages 1–1, 2003.

[6] C. Castelluccia, E. De Cristofaro, and D. Perito. Private information disclosure from web searches. In *PETS'10*, pages 38–55, 2010.

[7] S. Chen, R. Wang, X. Wang, and K Zhang. Side-channel leaks in web applications: A reality today, a challenge tomorrow. In *IEEE Symposium on Security and Privacy'10*, pages 191–206, 2010.

[8] Cynthia Dwork. Differential privacy. In *ICALP (2)*, pages 1–12, 2006.

[9] Edward W. Felten and Michael A. Schneider. Timing attacks on web privacy. In *CCS '00*, pages 25–32, 2000.

[10] Philip W. L. Fong, Mohd Anwar, and Zhen Zhao. A privacy preservation model for facebook-style social network systems. In *ESORICS '09*, pages 303–320, 2009.

[11] J. Freudiger, M. H. Manshaei, J-P Hubaux, and D. C. Parkes. On non-cooperative location privacy: a game-theoretic analysis. In *CCS '09*, pages 324–337, 2009.

[12] Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.*, 42:14:1–14:53, June 2010.

[13] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Incognito: Efficient fulldomain k-anonymity. In *SIGMOD*, pages 49–60, 2005.

[14] N. Li, T. Li, and S Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE '07*, pages 106–115, 2007.

[15] W.M. Liu, L. Wang, P. Cheng, and M. Debbabi. Privacy-preserving traffic padding in web-based applications. In *WPES '11*, pages 131–136, 2011.

[16] W.M. Liu, L. Wang, K. Ren, P. Cheng, and M. Debbabi. k-indistinguishable traffic padding in web applications. In *PETS'12*, pages 79–99, 2012.

[17] W.M. Liu, L. Wang, K. Ren, and M. Debbabi. Background knowledge-resistant traffic padding for preserving user privacy in web-based applications. Technical report, Concordia University, 2013. Available at http://users.encs.concordia.ca/~l_wenmin/cloudcom2013_long.pdf.

[18] Xiapu Luo, Peng Zhou, Edmond W. W. Chan, Wenke Lee, Rocky K. C. Chang, and Roberto Perdisci. Httpos: Sealing information leaks with browser-side obfuscation of encrypted flows. In *NDSS '11*.

[19] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1):3, 2007.

[20] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *IEEE Symposium on Security and Privacy '09*, pages 173–187, 2009.

[21] J.A. Rice. *Mathematical Statistics and Data Analysis.* second edition. Wadsworth, Belmont, California, 1995.

[22] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *CCS '09*, pages 199–212, 2009.

[23] Pierangela Samarati. Protecting respondents' identities in microdata release. *IEEE Trans. on Knowl. and Data Eng.*, 13(6):1010–1027, 2001.

[24] T. S. Saponas and S. Agarwal. Devices that tell on you: Privacy trends in consumer ubiquitous computing. In *USENIX*, pages 5:1–5:16, 2007.

[25] Q. Sun, D. R. Simon, Y-M Wang, W. Russell, V. N. Padmanabhan, and L Qiu. Statistical identification of encrypted web browsing traffic. In *IEEE Symposium on Security and Privacy '02*, pages 19–, 2002.

[26] C. V. Wright, S. E. Coull, and F. Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. In *NDSS '09*.

[27] Kehuan Zhang, Zhou Li, Rui Wang, XiaoFeng Wang, and Shuo Chen. Sidebuster: automated detection and quantification of side-channel leaks in web application development. In *CCS '10*, pages 595–606, 2010.

[28] Yinqian Zhang, Ari Juels, Alina Oprea, and Michael K. Reiter. Homealone: Co-residency detection in the cloud via side-channel analysis. In *IEEE Symposium on Security and Privacy'11*, pages 313–328, 2011.