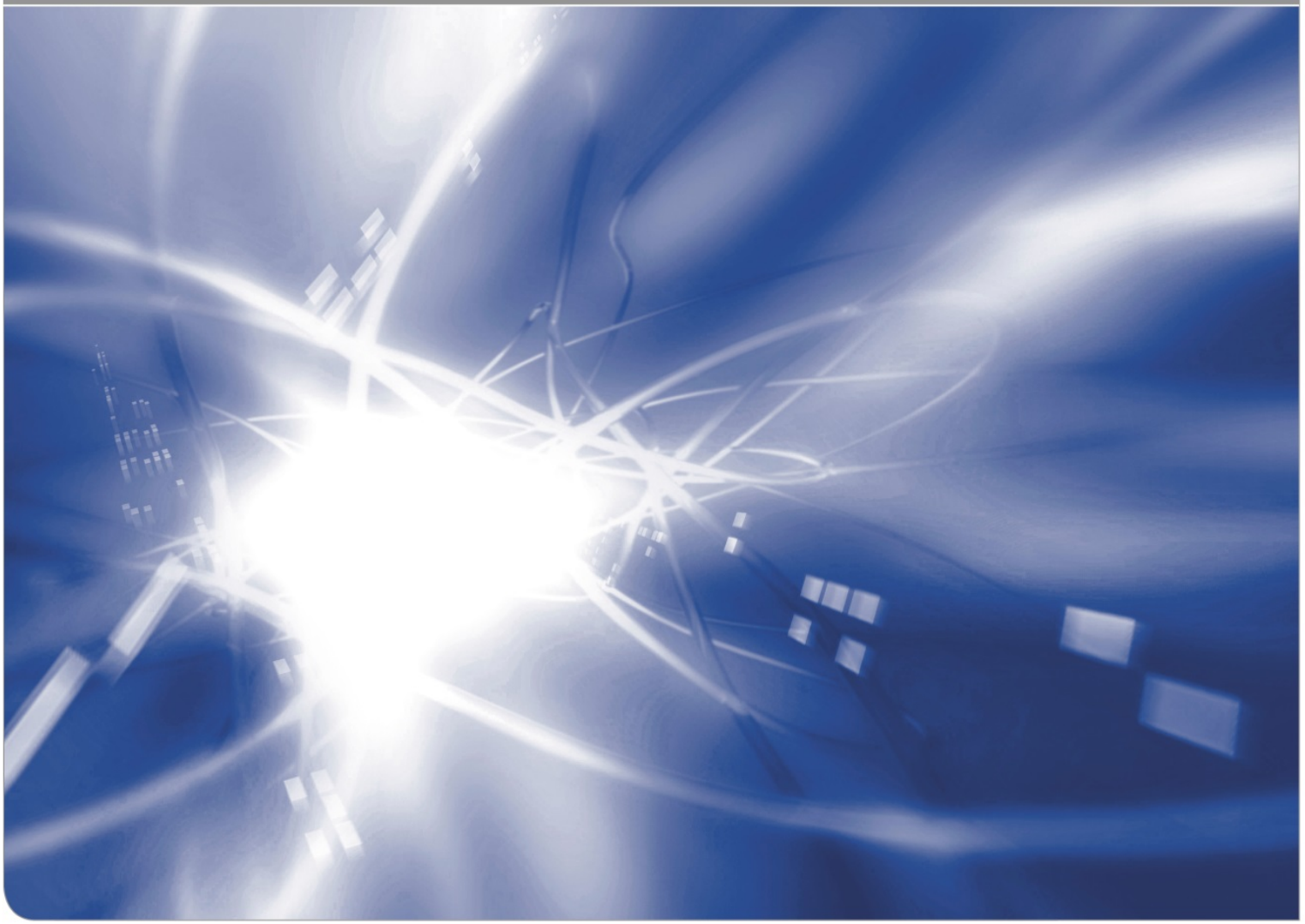
Karlsruhe Institute of Technology

Swellfish Privacy: Exploiting Time-Dependent Relevance for Continuous Differential Privacy

Technical Report

by Christine Tex, Martin Schäler and Klemens Böhm

KIT SCIENTIFIC WORKING PAPERS 151



Institute for Program Structures and Data Organization
Am Fasanengarten 5
76131 Karlsruhe

Impressum

Karlsruher Institut für Technologie (KIT)
www.kit.edu

2020

ISSN: 2194-1629

Swellfish Privacy: Exploiting Time-Dependent Relevance for Continuous Differential Privacy

Christine Tex
Karlsruhe Institute of Technology
Karlsruhe
christine.tex@kit.edu

Martin Schäler
Karlsruhe Institute of Technology
Karlsruhe
martin.schaeler@kit.edu

Klemens Böhm
Karlsruhe Institute of Technology
Karlsruhe
klemens.boehm@kit.edu

ABSTRACT

Today, continuous publishing of differentially private query results is the de-facto standard. The challenge hereby is adding enough noise to satisfy a given privacy level, and adding as little noise as necessary to keep high data utility. In this context, we observe that privacy goals of individuals vary significantly over time. For instance, one might aim to hide whether one is on vacation only during school holidays. This observation, named *time-dependent relevance*, implies two effects which – properly exploited – allow to tune data utility. The effects are time-variant sensitivity (TEAS) and time-variant number of affected query results (TINAR). As today's DP frameworks, by design, cannot exploit these effects, we propose Swellfish privacy. There, with policy collections, individuals can specify combinations of time-dependent privacy goals. Then, query results are Swellfish-private, if the streams are indistinguishable with respect to such a collection. We propose two tools for designing Swellfish-private mechanisms, namely, temporal sensitivity and a composition theorem, each allowing to exploit one of the effects. In a realistic case study, we show empirically that exploiting both effects improves data utility by one to three orders of magnitude compared to state-of-the-art w-event DP mechanisms. Finally, we generalize the case study by showing how to estimate the strength of the effects for arbitrary use cases.

KEYWORDS

differential privacy, streams, monitoring, privacy policies, data utility, time-dependent relevance

1 INTRODUCTION

The digitization of everyday live goes along with the availability of countless streams of personal data. Examples are locations continuously recorded by smart phones, or power consumptions recorded by smart meters. Monitoring these streams is expected to lead to new insights, facilitating many new applications [20]. To reduce data volume and to compensate individual bias, instead of individual data, one often monitors the results of statistical queries of the streams. Examples of such queries are `Counts` [9, 14], `Histograms` [18, 23] and `Sums` [8].

However, disclosure even of query results may compromise the privacy of individuals. This is mainly due to so-called differencing attacks [7]. To avoid this, one aims to sanitize the query results. Here, differential privacy [5] (DP) and its extensions [7, 12, 14, 16] are the current gold standard. The idea is to give a provable statistical indistinguishability guarantee of the query results computed on two databases that differ in two confidential *patterns*, i.e., sequences of events. Examples for such patterns are trajectories or sequences of physical activities. With a certain residual risk, it is not possible to determine from the query results whether an individual performed any confidential pattern. To achieve this, a privacy mechanism adds a well-defined amount of noise to each query result, ensuring a specific privacy level. Sanitizing query results is a balancing act between adding enough noise to ensure privacy, and adding as little noise as necessary to keep high data utility.

An important observation is that privacy goals of the same individual tend to be different for different times. We dub this observation *time-dependent relevance* of privacy goals.

Example 1.1 (Time-Dependent Relevance)**.** *Suppose that a smart meter records the power consumption every minute, and that one aims to monitor the* `Sum` *of power consumed in a city. Now think of an individual with two privacy goals: The first one is that he wants to hide whether he took a shower in the morning. Second, he wants to hide whether he cooked at lunch time or had (unhealthy) food delivered. He does not aim to hide anything else.*

In the example, the individual aims to hide two different appliances, i.e., fractions of his power consumption, during two different time periods of the day. We call such time periods *relevance intervals*. DP-like frameworks for streams do not take this observation into account so far. Instead, they would hide the worst case, i.e., *all* appliances *all the time*. – Time-dependent relevance of privacy goals has two effects, *time-variant sensitivity* and *time-variant number of affected query results*. They allow to tune data utility beyond designing new mechanisms for existing privacy definitions.

Example 1.2 (Time-Variant Sensitivity)**.** *The amount of power consumed to warm up water for showering, and thus to be hidden in the morning, is larger than the one needed for cooking at lunch time.*

Example 1.3 (Time-Variant Number of Affected Query Results)**.** *Taking a shower lasts 10 minutes at best, while cooking takes, say, at least an hour.*

Example 1.2 reveals that the fraction of the power consumption to be hidden is smaller at lunch time than in the morning. Furthermore, there is even nothing to hide in the evening and afternoon. This is the *time-variant sensitivity* effect (TEAS effect). Example 1.3 indicates that there are fewer power-consumption values monitored that are affected by the respective appliance usage in the morning than at lunch time. This is the *time-variant number of affected query results effect (TINAR effect)*.

In this paper, we design and evaluate a privacy framework which takes time-dependent relevance of privacy goals into account and allows to exploit these effects. Related work does not do this by design. It either is not designed for streams [3, 12, 16], or it cannot exploit the effects [14]. This is because the level of abstraction selected does not allow to model time-dependent privacy goals.

Challenges

Designing and evaluating the envisioned privacy framework is challenging. It is not obvious how to define the building blocks a DP-like framework consists of, like neighboring streams, utilization of the privacy budget, and the composition theorem [14]. This already holds for a single privacy goal and becomes even more challenging when supporting combinations of multiple – possibly concurrent – privacy goals, as we now explain.

Single Privacy Goal. Initially, taking time-dependent relevance into account means that one needs a notion that lets individuals specify *when* the framework must provide indistinguishability between *which* patterns. All building blocks of the framework must be aware of this notion.

Combinations of Privacy Goals. Think of an individual who specifies several pairs of patterns, each having a relevance interval. Then the intervals can overlap. To illustrate, an individual specifies that it should be hidden whether he took a shower anytime today, and, *additionally*, whether he cooked at lunch time. So it must be possible to hide *concurrent* privacy goals. This affects the definition of neighboring streams, the utilization of the privacy budget, and the composition theorem. First, one must capture that neighboring streams differ by pattern pairs from arbitrarily many concurrent privacy goals. Second, concurrent goals must share the privacy budget. Otherwise, they would be independent. Third, the query results affected by the patterns are not consecutive in time. This makes composition theorems based on rolling windows infeasible.

Contributions

We propose Swellfish privacy, a framework for continuous monitoring taking time-dependent relevance into account. We make the following contributions:

- We define the concepts stream policies and policy collections. A stream policy enables an individual to specify a privacy goal together with a relevance interval. A policy collection allows to specify several goals, possibly with different relevance intervals. We define Swellfish privacy by using these concepts, including an iterative definition of neighboring streams to allow for combinations of privacy goals.
- We prove that Swellfish privacy is a generalization of Blowfish privacy [12] to the streaming setting.
- To design Swellfish-private mechanisms exploiting the effects, we propose two tools. Each one corresponds to one effect. The first tool, corresponding to the TEAS effect, is named temporal sensitivity. The second tool, corresponding to the TINAR effect, is a composition theorem. With these tools, we adjust existing, and design new, mechanisms featuring different sampling and budget allocation techniques.
- We design and perform a realistic case study from the area of power consumption monitoring. We evaluate the utility of exploiting each effect in isolation and compare our mechanisms to the state-of-the-art. The study reveals that exploiting the effects improves data utility by up to three orders of magnitude, and that our mechanisms feature the best utility.
- We show how to estimate the strength of the effects for arbitrary use cases.

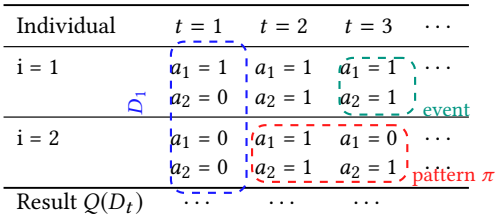


Figure 1: Continuous monitoring of query results.

2 PRELIMINARIES

In this section, we first recapitulate common notation on data streams and stream prefixes from literature. We then give a brief overview of privacy frameworks from literature. We recapitulate the ϵ-differential privacy framework as a foundation for everything that follows, as well as policy-driven frameworks. The latter improve data utility by allowing individuals to specify their actual privacy goals. However, both frameworks do not support infinite streams. We then introduce w-event differential privacy, the current state-of-the-art privacy framework for streams. It in turn does not support the specification of privacy goals.

2.1 Notation

Following [14] for most parts, we now introduce our notation. Let $I = \{1, .., n\}$ be a set of individuals and $\mathcal{A} = \{a_1, .., a_k\}$ a set of activities, like physical activities or appliance usages. An *activity configuration* is a function $\gamma : \mathcal{A} \to \mathcal{U}$ assigning each activity in $\mathcal{A}$ a value in the universe $\mathcal{U}$. Examples for universes are Boolean values stating whether one performs an activity, or real numbers stating the power consumption of the appliances. An *event* over $\mathcal{A}$ is an aggregation, like Sum or the identity function of an activity configuration. For an illustration, see Figure 1. There, the universe $\mathcal{U}$ are Boolean values indicating whether one performed an activity. Further, events are activity configurations, i.e., there is no aggregation. In contrast, in, say, power consumption streams, events are the sums of the consumption values of the appliances. For an arbitrary but fixed time stamp t, $D_t = (D_t[1], .., D_t[n])$ is a *static database* containing the event each individual $i \in I$ performs at time t. A query $Q : D_t \to \mathbb{R}^{\text{dim}}$ maps a static database D_t to *dim* values being monitored. Examples are histograms reflecting the number of individuals per location (*dim* = number of bins), or the sums of the power consumption values of individuals (*dim* = 1). A *continuous stream* $S = (D_1, D_2, \cdots)$ is an infinite sequence of continuously collected static databases at equidistant time stamps. τ is the respective time resolution. All static databases in a stream contain events over the same activity set $\mathcal{A}$. The *stream prefix* $S_p := (D_1, .., D_p)$ of a stream S is a finite prefix of the stream of length p. If $J = [t, t']$ is an interval, the *sub-stream* S_J is given by $S_J = (D_t, .., D_{t'})$. A *pattern* of length T is a finite sequence $\pi = [\pi_1, .., , \pi_T]$ of events. Static databases and streams may fulfill *constraints* on the events and patterns contained in them. A *monitoring system* publishes the results of a query over a stream continuously. $Q(S)$ stands for the application of Q to every static database of stream S, i.e.,

$$Q(S) = (Q(D_1), Q(D_2), \cdots).$$

A *privacy mechanism* $\mathcal{M}$ is a randomizing function having as input a static database or a stream prefix, and outputting a sanitized query result $Q+\eta$ that fulfills a privacy definition, e.g., differential privacy. Consequently, the output space Range($\mathcal{M}$) of $\mathcal{M}$ equals the output space of Q. In this paper, we aim at publishing $Q(S)$ continuously in a differentially private manner while taking time-dependent relevance into account.

2.2 ϵ-Differential Privacy

Differential privacy (DP) [5] is the current state-of-the-art privacy framework for static databases. The rationale is to hide all patterns an individual can perform in a published query result. Formally, let D be a static database, where each row corresponds to one individual. DP states that the output of a mechanism should be indistinguishable for any possible existing pair of *neighboring databases*. Databases D, D' are neighbors if one obtains one from the other by removing or adding *all* data of one individual, i.e., one row. A data curator sets the desired *privacy level* ϵ. It usually lies between 0.1 and 1.0. A smaller value means better privacy.

Definition 2.1 (Differential Privacy [5]). *A randomized mechanism $\mathcal{M}$ gives ϵ-differential privacy if for all neighboring databases D and D' and all $R \in$ Range($\mathcal{M}$) holds,*

$$Pr[\mathcal{M}(D) = R] \leq e^{\epsilon} \cdot Pr[\mathcal{M}(D') = R].$$

To publish differentially private query results, the Laplace mechanism [7] is frequently used. It adds noise that follows the zero-mean Laplace distribution Lap(λ) to each of the *dim* outputs of a query Q. The scale $\lambda = \frac{\Delta Q}{\epsilon}$ depends on the privacy budget ϵ as well as on the *global sensitivity* $\Delta Q = \max_{D,D'} ||Q(D) - Q(D')||_1$ of Q. The latter is the maximum difference in the query results for neighboring databases. For instance, Δ`Histogram` $= k$, i.e., the maximum number of activities one individual can perform concurrently.

2.3 Policy-Driven Privacy

Since DP is too strong in various use cases, and data utility is low, policy-driven privacy frameworks [12, 16] have been proposed. They are generalizations of DP, allowing individuals to specify their privacy goals with policies. To define Swellfish privacy we rely on the same idea, but generalize policies so that they feature time-dependent relevance. To prepare this, we now review discriminative secret pairs [12, 16], which are one building block of a privacy policy [12] we review afterwards. Further, we introduce Blowfish privacy [12], which Swellfish privacy will generalize.

2.3.1 Discriminative Secret Pairs. A *secret* is a statement of arbitrary nature on the values, e.g., events, in a database, which can or cannot be true for a specific database. A *discriminative secret pair* (s, s') is a tuple of two secrets s and s' that are mutually exclusive, i.e., cannot both be true. It describes properties of databases an adversary should not be able to distinguish between. An example is ("Christine consumes 1kW at time t", "Christine consumes 5kW at time t"). For more examples, we refer to [12, 16]. A privacy policy contains a set P of discriminative secret pairs.

Particularly relevant in this paper are sets of discriminative secret pairs defined by a distance measure and threshold [12]: If the domain of the database is associated with a distance measure d, such as the Manhattan distance, a *distance-based set of discriminative secret pairs* based on threshold θ

$$P^{d,\theta} = \{(s^i(x), s^i(y)) | d(x, y) \leq \theta\}$$

formalizes that an adversary should not be able to distinguish between all secrets $s^i(x), s^i(y)$ about data values x, y of individual i that are close to each other. For example, he should not be able to distinguish between events differing by at most θ activities.

Table 1: Comparison of privacy frameworks.

Framework	Intra-Individual	Inter-Individual
Static Setting		
original DP [5]	✓	✗
Policy-Driven [12, 16]	✓	✓
Streaming Setting		
w-event DP [14]	✓ in window of size $\leq w$	✗
Swellfish privacy	✓ within each relevance interval	✓ within each relevance interval

2.3.2 Privacy Policies. A set of discriminative secret pairs P and activities $\mathcal{A}$ are two of three inputs of a privacy policy $\phi = (\mathcal{A}, P, C)$. The third input C is auxiliary knowledge in the form of deterministic dependencies that an adversary may has about the values in the database, possibly helping him to infer private information from sanitized query results. One differs between intra- and inter-individual dependencies [2, 19, 25], see Table 1. Intra-individual constraints describe dependencies between values of *one* individual, while inter-individual constraints refer to values of *different* individuals.

2.3.3 Blowfish Privacy. Blowfish privacy is given if an adversary cannot distinguish between query results computed on two databases that are neighboring with respect to a privacy policy. This means that they differ in a minimal number of discriminative secret pairs, see Definition 2.2. If $C = \emptyset$, i.e., there are no constraints, neighboring databases differ in one arbitrary secret pair. In consequence, the policy-specific sensitivity is generally lower than the global one needed in the DP framework. This in turn means that one needs less noise to achieve the desired privacy level.

Definition 2.2 (Neighboring databases w.r.t. a policy [12]). *Let $\phi = (\mathcal{A}, P, C)$ be a policy and D, D' two databases. Further, let $\mathcal{T}(D, D') \subseteq P$ be the set of discriminative secret pairs (s, s') so that secret s is true in D, and secret s' is true in D', and $\Delta(D, D') = D_1 \backslash D_2 \cup D_2 \backslash D_1$. Then D and D' are neighbors, i.e., $D, D' \in \mathcal{N}(\phi)$, if*
(1) they both fulfill the constraints in C,
(2) $\mathcal{T}(D, D') \neq \emptyset$
(3) there is no database D'' fulfilling C so that
- *$\mathcal{T}(D, D'') \subset \mathcal{T}(D, D')$, or*
- *$\mathcal{T}(D, D'') = \mathcal{T}(D, D')$ and $\Delta(D'', D) \subset \Delta(D', D)$.*

2.4 w-Event Differential Privacy

Differential privacy and Blowfish privacy are designed for static databases. Applied to streams, DP would protect *any*, possibly *infinitely long*, pattern of one individual, which is unreasonable. To counter this, the w-event differential privacy framework [14], the current reference point for streams, has been proposed. In the following, we review w-event differential privacy and sketch how to design mechanisms that satisfy it. We will use the mechanisms as competitors in our experiments.

The definition of w-event differential privacy is an adaptation of Definition 2.1, providing indistinguishability of w-neighboring stream prefixes. Here, w-neighboring stream prefixes differ by *any* pattern pair of length smaller than or equal to w, occurring *anywhere* in the stream. To design w-event mechanisms, an important observation is Theorem 2.1 from [14]. It states that one can design a w-event ϵ-differentially private mechanism $\mathcal{M}$, by taking one ϵ_t-differentially private mechanism $\mathcal{M}_t$ for each time stamp t as a sub-mechanism. If, in every rolling window of size w, it holds that the sum of the budgets ϵ_t does not exceed ϵ, the mechanism $\mathcal{M}$ is w-event ϵ-differentially private. For instance, by leveraging Theorem 2.1, one can design the Uniform mechanism [14]. It implements each sub-mechanism with the Laplace mechanism, i.e., $\mathcal{M}_t = \mathcal{M}_Q$, with a budget of $\epsilon_t = \frac{\epsilon}{w}$ each. That is, the privacy budget is split uniformly over the time stamps in each rolling window.

Theorem 2.1 (Composition [14]). *Let $\mathcal{M}$ be a mechanism that takes as input a stream prefix $S_p = (D_1, .., D_p)$ and outputs $R = (r_1, .., r_p)$. Assume that one can decompose $\mathcal{M}$ into p sub-mechanisms $\mathcal{M}_1, .., \mathcal{M}_p$, s.t. $\mathcal{M}_t(D_t) = r_t$, each $\mathcal{M}_t$ has independent randomness and achieves ϵ_t- differential privacy. Then $\mathcal{M}$ satisfies w-event differential privacy if*

$$\forall t \in [1, p] : \sum_{k=t-w+1}^{t} \epsilon_k \leq \epsilon.$$

Literature proposes more sophisticated mechanisms as well [1, 9, 14, 18, 23, 24] that use *sampling*, possibly combined with *dynamic budget allocation* and *dynamic grouping*, to improve data utility. Here, sampling means that a mechanism publishes the current (perturbed) statistics for *some* time stamps only. For the other ones, it re-publishes the statistics from the last sampling time stamp. The rationale is to save budget at non-sampling time stamps.

3 SWELLFISH PRIVACY

In this section, we propose Swellfish privacy. It is a framework for differentially private continuous monitoring of infinite streams that takes time-dependent relevance into account. We first introduce three use cases and highlight their peculiarities. Second, we propose our definition of Swellfish privacy and point out how it hides combinations of privacy goals. Third, we show that Swellfish privacy generalizes Blowfish privacy to the streaming setting.

3.1 Use Cases

In this section, we introduce three common use cases, namely, physical activity, power consumption, and location monitoring. For each of them, we illustrate how events, patterns and the monitored stream look like. Second, the illustrations will reveal that the challenge of combining privacy goals manifests itself differently in the use cases, so we discuss the consequences for Swellfish privacy.

3.1.1 Physical Activity Monitoring. In this use case, activities $a_1, .., a_k \in \mathcal{A}$ correspond to physical activities, like *running* or *eating*. An event is an activity configuration $\gamma : \mathcal{A} \to \mathcal{U} = \{1, 0\}$ assigning every activity a logical value, like in Figure 1. A pattern is a sequence of such events, e.g., a jogging round. Here, one typically monitors histograms stating how many individuals performed which activities. Generally, an individual can perform some, but not all physical activities concurrently. For instance, one might sit and eat, but not sit and run at the same time.

3.1.2 Power Consumption Monitoring. Here, activities $a_1, .., a_k \in \mathcal{A}$ correspond to the usage of appliances, like the *washing machine*. At time t, every appliance consumes a specific amount of power. An activity configuration $\gamma : \mathcal{A} \to \mathcal{U} = \mathbb{R}_0^+$ assigns every appliance an amount of power. An event is the sum over these values of an activity configuration. Thus, $D_t[i] = \sum_{a_v \in \mathcal{A}} \gamma_t(a_v)$ is the total power consumed by residential unit i. As result, patterns to be hidden are vectors of real numbers. Here, one monitors the sum of power consumption values of all residential units in, say, a city. In such a unit, several (possibly all) appliances run at the same time.

3.1.3 Location Monitoring. Here, activities $a_1, .., a_k \in \mathcal{A}$ correspond to visits of locations. As an individual is at one location at a time, $\gamma : \mathcal{A} \to \mathcal{U} = \{1, 0\}$ is true for exactly one location. An event is the location for which γ is true, and a pattern is a trajectory represented by a sequence of locations. Here, one monitors histograms stating how many individuals are at each location.

If privacy goals are relevant at the same time, one must hide the combination of the respective patterns. However, the above illustrations indicate that there are use-case specific constraints limiting the space of activities, and therefore patterns that *could* take place concurrently: In the power-consumption use case, there are no such constraints, while with location monitoring, there are no concurrent location visits. In the physical-activity use case, some, but not all activities can be performed concurrently. In the remainder, we therefore only consider events and patterns that fulfill a set of use-case dependent *event constraints* $C_\gamma \subseteq C$.

3.2 Definition of Swellfish Privacy

In this section, we define Swellfish privacy. To this end, in Section 3.2.1, we first propose our notion allowing individuals to specify multiple privacy goals. In Section 3.2.2, we define neighboring stream prefixes so that the definition copes with concurrent privacy goals. Section 3.2.3 contains our definition of Swellfish privacy. It features the sharing of privacy budget between concurrent goals.

3.2.1 Notion for Specifying Privacy Goals. We now propose two concepts our notion is built on, namely, stream policies and policy collections. First, stream policies are a deployment of general policies [12, 16] (cf. Section 2.3) for the streaming setting. With them, an individual can formalize a single privacy goal referring to one relevance interval. Second, to allow for multiple, possibly concurrent, privacy goals, we propose policy collections.

Privacy Goal	**Stream Policy** $\phi = (\mathcal{A}, P, C(J), J)$ Activities $\mathcal{A}$	Set of Discriminant Stream Policies P	$C_\gamma \subseteq C(J)$	J
1. Physical Activity Hide whether next Friday, one goes jogging (= a_1) for up to one hour during working time, or not.	{jogging, sitting, ..}	$\{sp_j(\pi = \begin{bmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & .. & .. \end{bmatrix}, \quad \pi' = \begin{bmatrix} 0 & 0 & 0 & 0 \\ x_1 & x_2 & .. & .. \end{bmatrix})$ $\mid \forall x_t = (\gamma_t(a_2), .., \gamma_t(a_k)), j \in J^T\}$	cf. application	04/03/2020, 8 a.m. - 6 p.m.
2. Power Consumption Hide whether one cooked with his stove (=a_1), that consumes 7.5 kW, for up to one hour on lunch time next Friday.	{stove, dryer, ..}	$\{sp_j(\pi = [\pi_1 = \sum_{a \in \mathcal{A}} \gamma_1(a), .., \pi_4], \quad \pi' = [\pi'_1 = \sum_{a \in \mathcal{A}} \gamma'_1(a), .., \pi'_4]$ $\mid \lvert\pi_t - \pi'_t\rvert \leq 7.5 \; \forall t, \forall j \in J^T\}$	$\emptyset$	04/03/2020, 11 a.m. - 1 p.m.
3. Location Monitoring Hide whether one does a side trip to Pizza Hut (= a_2), on ones 45 min. jogging round in the city park (= a_1) next Saturday evening.	{city park, Pizza Hut, ..}	$\{sp_j(\pi = [a_1, a_1, a_1], \quad \pi' = [a_2, a_1, a_1]),$ $sp_j(\pi = [a_1, a_1, a_1] \quad \pi' = [a_1, a_2, a_1]),$ $sp_j(\pi = [a_1, a_1, a_1] \quad \pi' = [a_1, a_1, a_2])$ $\mid \forall \, j \in J^T\}$	$\{\gamma \mid \oplus_{a_v \in \mathcal{A}} \gamma(a_v) = 1\}$	04/05/2020, 5 p.m. - 8 p.m.

Figure 2: Privacy goals and resulting stream policies assuming a time resolution τ = 15 min. of the underlying streams.

Stream Policies. So-called *stream policies* consist of pairs of secrets. A secret $s_j^i(\pi)$ states that individual i performs pattern π at time interval $[j, j+T]$. Here, j is called the *start time*, and $T := |\pi|$ is the length of the pattern. To illustrate, consider the pattern

$$\pi = \begin{bmatrix} a_1 = 1 & a_1 = 0 \\ a_2 = 1 & a_2 = 1 \end{bmatrix}$$

of length $T = 2$. The columns represent the events and the rows the activities configured in the event. Further, consider the example stream in Figure 1. There, the secret $s_{j=2}^{i=2}(\pi)$ is true, as the above pattern starts at time $j = 2$. In contrast, the secret $s_1^2(\pi)$ is *not* true. In the remainder, in line with related work [12], we assume that all secrets considered belong to the same individual i, and we omit the superscript. Now, a discriminative stream secret pair $\text{sp}_j(\pi, \pi')$ is a pair of two secrets, consisting of different, but equal-length, patterns π, π' with the *same* start time j. For readability, in the remainder, *secret pair*, is short for *discriminative stream secret pair*.

Definition 3.1 (Discriminative Stream Secret Pair)**.** *A discriminative stream secret pair*

$$sp_j(\pi, \pi') := (s_j(\pi), s_j(\pi'))$$

is a tuple of two secrets having the same start time j and referring to two equal-length patterns $\pi \neq \pi'$.

Requiring the same start time makes sure that the secrets are mutually exclusive, as unequal patterns differ in at least one event, and one individual can perform only one event at a time. It is possible to extend Swellfish privacy so that different start times are allowed, as long the respective secrets are still mutually exclusive.

One secret pair may not be enough to specify a privacy goal. For instance, one generally needs secret pairs for different start time stamps. Namely, one may want to hide the occurrence of patterns at any time during a relevance interval, and not only at a specific one. In consequence, a *set* of secret pairs P belonging to different pattern pairs π, π' is one of four elements of a stream policy ϕ.

Definition 3.2 (Stream Policy)**.** *A stream policy $\phi = (\mathcal{A}, P, C(J), J)$ is a four-tuple containing a set of activities $\mathcal{A}$, secret pairs P, a relevance interval J, and deterministic constraints $C(J)$ on events in interval J so that*

$$P = \{sp_j(\pi_1, \pi'_1), \cdots, sp_j(\pi_q, \pi'_q), , \cdots \mid \forall j \in J^{T_q}\}$$

and all patterns π_q, π'_q fulfill $C(J)$. Here, $J^T \subseteq J$ is the interval that begins at the same time stamp as J, but ends $T-1$ time stamps earlier.

The remaining elements are as follows: The set of activities $\mathcal{A}$ contains the activities recorded in the stream. The relevance interval J is the interval when the privacy goal must be satisfied. The set of deterministic constraints $C(J)$ is the public knowledge an adversary might have about the events having taken place during J (cf. Section 2.3.3). In particular, it contains the event constraints $C_\gamma \subseteq C(J)$. For convenience we assume that, if P contains a secret pair belonging to the patterns π, π', it must contain such a pair for *every* possible start time stamp $j \in J^T$.

In the remainder, our examples focus on power consumption, as does our case study. In this use case, typical sets of secret pairs are distance-based, cf. Section 2.3. See Figure 2 for an example. In the streaming setting, they are defined as follows.

Definition 3.3 (Distance-Based Stream Policy)**.** *A distance-based stream policy is a stream policy of the form*

$$P_T^{d,\theta} = \{(sp_j(\pi, \pi') \mid \forall j \in J^T \quad \text{and } \pi, \pi' \text{ with } d(\pi_t, \pi'_t) \leq \theta_t \; \forall t \in [1, T]\}.$$

There are two differences two the static setting: Threshold θ generally is a vector, and we use subscript T to represent the length of the pattern contained in the policy.

Policy Collections. A policy collection is a set of stream policies with possibly different relevance intervals.

Definition 3.4 (Policy Collection)**.** *A policy collection Φ is a set of stream policies $\Phi = \{\phi_1, \phi_2 \cdots, \phi_{|\Phi|}\}$ so that the policies contain the same activities $\mathcal{A}$ and the same constraints for a given time stamp,.*

In the remainder, we use the notion of relevant subsets of stream policies of a policy collection.

Definition 3.5 (Relevance of Stream Policies and Time Stamps)**.** *For a time stamp t, a stream policy ϕ is* relevant *if $t \in J$. Similarly, time stamp t is relevant if there is at least one relevant policy at time t. For a policy collection Φ, the set $\Phi_t \subseteq \Phi$ contains all stream policies from Φ that are relevant at time t.*

While the definition of a policy collection appears to be straightforward, defining neighboring stream prefixes that are aware of policy collections is more difficult.

3.2.2 Neighboring Stream Prefixes. Neighboring stream prefixes are stream prefixes for which one must provide indistinguishability. This means that one must perceive streams as neighbors if they are neighbors with respect to a single policy, but also with respect to several ones. The latter means that neighboring streams can differ regarding one secret pair from *each* stream policy in a policy collection. However, not all combinations of secret pairs from different stream policies *can* result in neighboring streams. It may hold that two secrets from different policies cannot be true at the same time. However, if they can be true, we argue that the individual intends to hide the respective secret pair combination , i.e., the concatenation of secrets from different policies. Secret pair combinations might result in additional patterns to be hidden.

Example 3.1 (Combining Secret Pairs)**.** *Suppose that the secret pairs $sp_{j_1=1}(\pi_1, \pi_1')$ and $sp_{j_2=3}(\pi_2, \pi_2')$ belong to different stream policies. Further, let $\pi_1' = [loc_1, loc_7, loc_3]$ and $\pi_2 = [loc_3, loc_5]$ be given. In these patterns, the last location in π_1' is equal to the first location in π_2. This means that the respective secrets can be true at the same time, because $sp_{j_1=1}$ starts at time $j_1 = 1$, and $sp_{j_2=3}$ at time $j_2 = 3$. This in turn means that the individual intends to hide the individual patterns, as well as the pattern $[loc_1, loc_7, loc_3, loc_5]$.*

The following definition of neighboring stream prefixes ensures that there is a pair of neighboring stream prefixes for every possible combination of secret pairs.

Definition 3.6 (Neighboring Stream Prefixes w.r.t. a Policy Collection)**.** *Let Φ be a policy collection. The stream prefixes S_p, S_p' are neighbors with respect to Φ, i.e., $S_p, S_p' \in \mathcal{N}(\Phi)$ iff there exists*

(1) a sequence $[\phi^0, .., \phi^k]$ of stream policies from Φ containing each stream policy at most once, and allowing to construct

(2) a sequence $[S_p^0 = S_p, .., S_p^l, .., S_p^{k+1} = S_p']$ of stream prefixes, starting with S_p and ending with S_p',

so that for all $0 \leq l \leq k$ it holds that $(S_p^l, S_p^{l+1}) \in \mathcal{N}(\phi^l)$.

Example 3.2 (Neighboring Stream Prefixes)**.** *Consider Figure 3. It contains a policy collection, together with stream prefixes. Assuming that the stream policies belong to individual $i = 1$, it shows that $S_5, S_5' \in \mathcal{N}(\Phi)$ for Case (a) and (b). For instance, consider Case (a). The stream prefixes S_5^0 and S_5^1 are neighbors with respect to policy ϕ^0, as they differ by a pattern of length $T = 1$ consuming $\theta = 1$ kW at time stamp $t = 3 \in J_1$. Similarly, S_5^1 and S_5^2 are neighbors with respect to policy ϕ^1. Also observe that, due to constraints, the power values of a second individual can change as well.*

3.2.3 Swellfish privacy. We now define Swellfish privacy.

Definition 3.7 ((ϵ, Φ)-Swellfish privacy)**.** *Let $\mathcal{M}$ be a randomized mechanism that has as input a stream prefix S_p of arbitrary size, Φ be a policy collection, and $\epsilon > 0$. The mechanism $\mathcal{M}$ gives (ϵ, Φ)-Swellfish privacy iff*

$$Pr[\mathcal{M}(S_J) = R] \leq e^{\epsilon} \cdot Pr[\mathcal{M}(S_J') = R]$$

for every relevance interval J of a stream policy $\phi \in \Phi$.

Here, the semantics of the privacy budget ϵ is particularly interesting. The definition says that one has budget ϵ for each relevance interval. In consequence, if there is only one privacy goal, one has budget ϵ for this goal. However, if several policies are relevant at the same time t, they share the budget, meaning that the polices depend on each other.

3.3 Relationship to Blowfish Privacy

In this section, we relate Swellfish privacy to Blowfish privacy [12], before we discuss the relationship to other privacy frameworks in the remainder. While Blowfish privacy is defined for the static setting and allows for one policy only, Swellfish privacy is defined for the streaming setting and allows for policy collections. However, reduced to the static setting, Swellfish privacy is equivalent to Blowfish privacy. To prove this, we show (1) that one can implement (ϵ, ϕ_B)-Blowfish privacy with Swellfish privacy, and (2) vice versa. To this end, we consider a definition of Swellfish privacy restricted to a static database given in Definition 3.8.

Definition 3.8 (Static (ϵ_t, Φ_t)-Swellfish privacy)**.** *Let $\mathcal{M}_t$ be a randomized mechanism that has as input a static database D_t. The mechanism $\mathcal{M}_t$ gives (ϵ_t, Φ_t)-Swellfish privacy iff for all neighboring databases $(D_t, D_t') \in \mathcal{N}(\Phi_t)$, and all $R \in Range(\mathcal{M})$, holds*

$$Pr[\mathcal{M}_t(D_t) = R] \leq e^{\epsilon} \cdot Pr[\mathcal{M}_t(D_t')) = R].$$

We now show both directions separately.

Blowfish → Swellfish privacy. Given a Blowfish policy $\phi_B = (\mathcal{A}, P, C)$, let $\phi_S = (\mathcal{A}, P, C, J)$ with $J = [t, t]$ be a stream policy. Both policies contain the same activities, constraints and set of secret pairs. The relevance interval of ϕ_S is restricted to the time stamp t, i.e., to one time stamp only. Then, providing (ϵ_t, Φ_S)-Swellfish privacy for the policy collection $\Phi_S = \{\phi_S\}$ is equivalent to providing (ϵ_t, ϕ_B)-Blowfish privacy, as $\Phi_t = \Phi_S$. As result, a Swellfish private mechanism provides Blowfish privacy.

Swellfish → Blowfish privacy. Let Φ_S be a policy collection. W.l.o.g., assume that at time t, the two stream policies $\Phi_{S,t} = \{\phi, \phi'\}$ are relevant. Let $s_t(\pi), s_t'(\pi')$ any two secrets from ϕ, ϕ', that can be simultaneously true at time t. For each such two secrets, we construct a new secret covering both of them, i.e., both are true. Then, the stream secret pairs containing $s_t(\pi)$ and $s_t'(\pi')$, we replace these secrets with this new secret. By doing this for all stream secret pairs, we obtain a new set of secret pairs P_B, and policy $\phi_B = (\mathcal{A}, P_B, C, J = [t, t])$. For this, it holds that, Swellfish neighbors $(D_t, D_t') \in \mathcal{N}(\Phi_{S,t})$ are also Blowfish neighbors, i.e., $(D_t, D_t') \in \mathcal{N}(\phi_B)$.

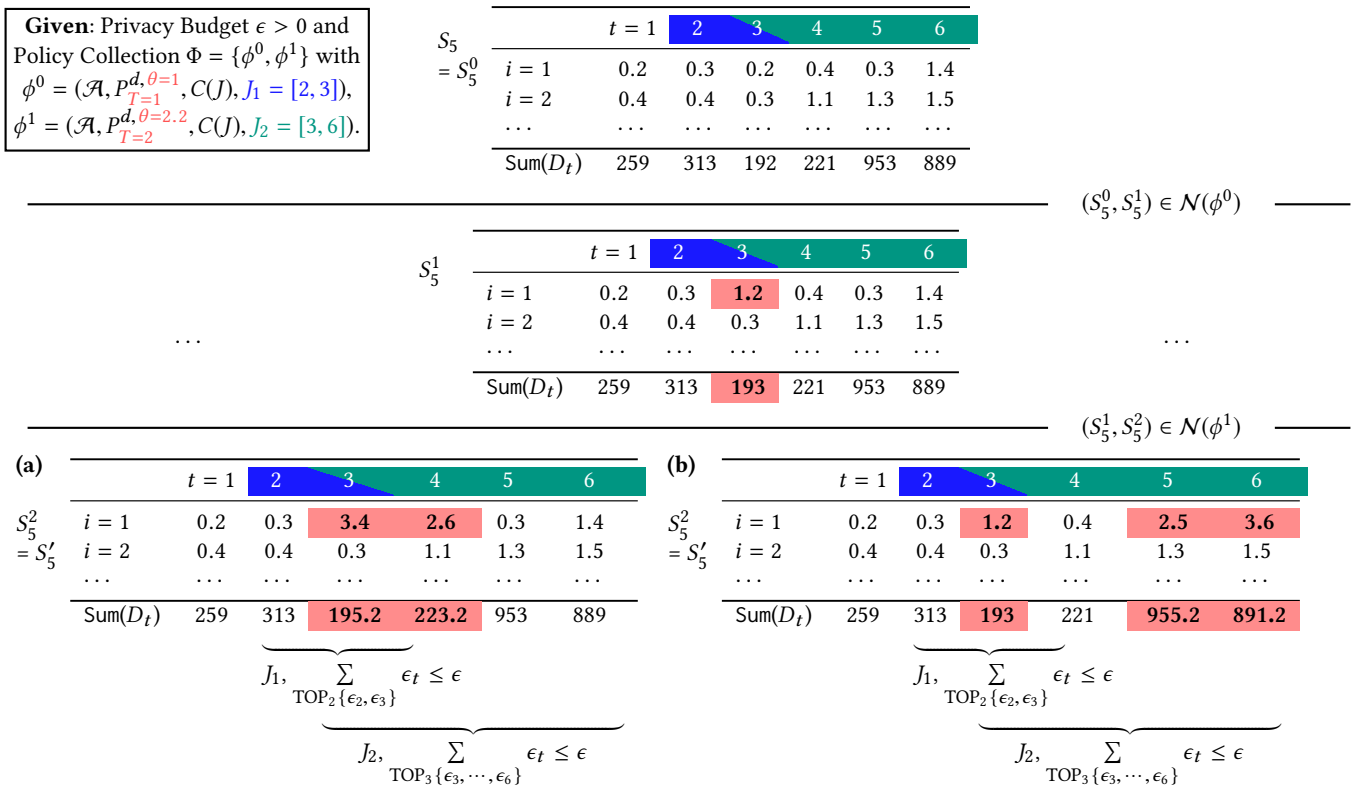


Figure 3: Illustration of our iterative definition of neighboring stream prefixes.

4 MECHANISM DESIGN

After defining Swellfish privacy, we now design mechanisms providing it. We first provide two tools, each one allowing to exploit one of the effects featured in the introduction. Second, we propose concrete mechanisms, both baseline mechanisms and three new mechanisms. In contrast to the baseline, the latter ones can exploit the TEAS and TINAR effect.

4.1 Toolbox

Our aim is to design mechanisms that consist of independent (ϵ_t, Φ_t)-Swellfish private sub-mechanisms $\mathcal{M}_1, .., \mathcal{M}_p$, where $\mathcal{M}_t$ outputs the private query result at time t. To facilitate this, the first tool is the temporal sensitivity, allowing to exploit the TEAS effect. With this, one can design sub-mechanisms based on Laplace perturbation. The second tool is a composition theorem, allowing to exploit the TINAR effect. It constrains how to distribute the privacy budget among the sub-mechanisms.

4.1.1 Temporal Sensitivity. To provide Swellfish privacy, one can add Laplace noise to the query results that is proportional to the *temporal sensitivity* of the query, see Definition 4.1. Generally, for every time stamp t, the temporal sensitivity is different, as different policies may be relevant.

Definition 4.1 (Temporal Sensitivity). *Let Φ be a policy collection. For a query $Q : D \to \mathbb{R}^{dim}$, the temporal sensitivity at time t is*

$$\Delta_Q^t(\Phi) = \max_{(S_p, S_p') \in \mathcal{N}(\Phi)} ||Q(D_t) - Q(D_t')||_1.$$

For illustration, see Example 4.1.

Example 4.1 (Temporal Sensitivity). *Consider again the policy collection from Figure 3. There, at time $t = 3$, in the worst case, neighboring stream prefixes differ by $|\text{Sum}(D_t) - \text{Sum}(D_t')| = 3.2$ kW. The rationale is that both stream policies ϕ^0 and ϕ^1 featuring thresholds 1.0 and 2.2 are relevant there. In contrast, at $t = 4$, the temporal sensitivity is 2.2, as only policy ϕ^1 is relevant. At $t = 1$, the temporal sensitivity is 0, as no policy is relevant.*

Theorem 4.1 states that adding noise that is proportional to the temporal sensitivity at time t yields (ϵ_t, Φ_t)-Swellfish privacy for a specific time stamp t.

Theorem 4.1. *Let t be an arbitrary, but fixed time stamp. The mechanism $\mathcal{M}_t$ given by*

$$\mathcal{M}_t(D_t) = Q(D_t) + Lap(\frac{\Delta_Q^t(\Phi)}{\epsilon})$$

provides (ϵ_t, Φ_t)-Swellfish privacy at time t.

Proof: As Swellfish privacy is equivalent to Blowfish privacy in the static setting, this holds due to Theorem 5.1 from [12]. □

4.1.2 Composition Theorem. Distributing the budget between the sub-mechanisms is challenging due to concurrent privacy goals. The rationale is that the time stamps affected by the goals are generally not consecutive. To illustrate, consider Case (b) in Figure 3. There, time stamps 3, 5 and 6 are affected, which are not consecutive.

Defining and Determining Goal-Affected Time Stamps. Given two specific neighboring stream prefixes, we refer to a time stamp t that is affected by any goal as goal-affected time stamp.

Definition 4.2 (Goal-Affected Time Stamp)**.** *Let Φ be a policy collection, and $S_p, S'_p \in \mathcal{N}(\Phi)$. A time stamp t is a* goal-affected time stamp *iff $D_t \neq D'_t$.*

For instance, in Figure 3, in both cases, time stamp $t = 3$ is a goal-affected time stamp, but $t = 2$ is not. Nevertheless, different neighboring streams may feature different goal-affected time stamps. For instance, $t = 4$ in Case (a) in Figure 3 is a goal-affected time stamp, but not in Case (b). Thus, to provide indistinguishability for all possible neighboring stream prefixes, one must protect the *maximum* number of goal-affected time stamps over all neighboring stream prefixes. For a given relevance interval J, we abbreviate this number with $\delta(J)$, i.e.,

$$\delta(J) = \max_{S_p, S'_p \in \mathcal{N}(\Phi)} |\{t \mid D_t \neq D'_t\}|.$$

One can determine $\delta(J)$ as follows. If during J only one policy is relevant, $\delta(J)$ is the length of the longest pattern in the policy. Otherwise, one can determine an upper bound on $\delta(J)$ with Algorithm 1. It determines $\delta(J)$ by using the length of the patterns of all relevant policies, and the number of time stamps in which they overlap. To illustrate the algorithm, consider Figure 3 and $J = J_2$. Here, the algorithm initializes $\delta^0_\phi(J_2) = 1$ and $\delta^1_\phi(J_2) = 2$ in Line 3 with the length of the respective patterns. As both policies overlap at time $t = 3$ only, it increments both numbers in Line 5 by 1. As result, $\delta(J_2) = 2 + 1 = 3$.

Algorithm 1 determine $\delta(J)$

```
1:  procedure CalcDeltaJ(Φ, J)
2:      for φ ∈ Φ_J do                          ▷ policies relevant during J
3:          δ_φ(J) := max{|π| | sp_J(π, π') ∈ P}
4:          for φ' ∈ Φ_J, φ' ≠ φ, J ∩ J' ≠ ∅ do   ▷ overlap. policies
5:              δ_φ(J) += min{|J ∩ J'|, max{|π| | sp_J(π, π') ∈ P'}}
6:          end for
7:          δ_φ(J) = min{δ_φ(J), |J|}
8:      end for
9:      δ(J) = max{δ_φ(J) | φ ∈ Φ_J}
10: end procedure
```

Protecting all Possible Goal-Affected Time Stamps. With this, we arrive at Theorem 4.2, the composition theorem. The intuition is as follows. For each relevance interval J, one must protect $\delta(J)$ time stamps in this interval. However, in case many stream policies overlap with each other, the goal-affected time stamps are distributed arbitrarily over J. Therefore, the theorem presumes that they can lie *anywhere* in the interval. However, as one knows that one has to protect only $\delta(J)$ time stamps, it is sufficient to make sure that the budget consumed at any $\delta(J)$ time stamps does not exceed ϵ.

Theorem 4.2 (Composition). *Let $\mathcal{M}$ be a mechanism having as input a stream prefix $S_p = (D_1, .., D_p)$, and outputting $R = (r_1, .., r_p)$. Assume that we can decompose $\mathcal{M}$ into p sub-mechanisms $\mathcal{M}_1, .., \mathcal{M}_p$, s.t. $\mathcal{M}_t(D_t) = r_t$, where each $\mathcal{M}_t$ has independent randomness and achieves (ϵ_t, Φ_t)- Swellfish privacy. Then $\mathcal{M}$ satisfies (ϵ, Φ)-Swellfish privacy if*

$$\forall \phi \in \Phi : \sum_{\epsilon_t \in \mathcal{X}} \epsilon_t \leq \epsilon. \tag{1}$$

$\mathcal{X} = TOP_{\delta(J)}\{\epsilon_t | t \in J\}$ is the set of the highest $\delta(J)$ budgets spent during J.

Proof: Neighboring sub-streams S_J and S'_J differ in at most $\delta(J)$ time stamps. As (1) all mechanisms use independent randomness, (2) are (ϵ_t, Φ_t)-Swellfish private, and (3) Equation 1 holds, we have

$$\begin{aligned}\frac{Pr[\mathcal{M}(S_J) = R]}{Pr[\mathcal{M}(S'_J) = R]} &\overset{(1)}{=} \max_{\mathcal{Y} \in \mathbb{P}_{\delta(J)}(J) \cap [1,p]} \prod_{t \in \mathcal{Y}} \frac{Pr[\mathcal{M}_t(D_t) = r_t]}{Pr[\mathcal{M}_t(D'_t) = r_t]} \\ &\overset{(2)}{\leq} \max_{\mathcal{Y} \in \mathbb{P}_{\delta(J)}(J) \cap [1,p]} \prod_{t \in \mathcal{Y}} e^{\epsilon_t} \\ &= \max_{\mathcal{Y} \in \mathbb{P}_{\delta(J)}(J) \cap [1,p]} \exp(\sum_{t \in J} \epsilon_t) \overset{(3)}{\leq} \exp(\epsilon).\end{aligned}$$

$\mathbb{P}_{\delta(J)}(J)$ contains all sets from the power set of J of size $\delta(J)$. □

4.2 Mechanisms Satisfying Swellfish privacy

In this section, we design mechanisms satisfying Swellfish privacy. To this end, we first propose a baseline. Then we discuss how we can do better than the baseline in terms of utility, by exploiting the effects with the tools just introduced. In this context, we propose three Swellfish-private mechanisms that feature different sampling and budget allocation strategies and exploit both effects.

4.2.1 w-Event DP Baseline. Swellfish privacy differs from w-event DP in the specification of privacy goals and inter-individual constraints only (cf. Table 1). Thus, in the absence of inter-individual constraints, a $(w = \max_{\phi \in \Phi} |J|)$-event private mechanism satisfies Swellfish privacy.

Theorem 4.3. *Let Φ be a policy collection where all $C(J)$ does not contain any inter-individual constraint, $\epsilon > 0$ and let $\mathcal{M}$ be a w-event DP mechanism. If $w \geq \max_{(\mathcal{A}, P, C(J), J) \in \Phi} |J|$, then $\mathcal{M}$ provides (ϵ, Φ)-Swellfish privacy.*

Proof: Let $\phi = (\mathcal{A}, P, C(J), J) \in \Phi$. Then, $|J| \leq |J'| = w$ where J' is the relevance interval that corresponds to

$$(\mathcal{A}, P, C(J), J) = \arg\max_{(\mathcal{A}, P, C(J), J) \in \Phi} |J|.$$

Let S_J, S'_J be sub-streams of the stream prefixes S_p, S'_p, which differ during interval J only. Then, S_p, S'_p are neighboring stream prefixes for $w \geq |J|$. As $\mathcal{M}$ is $w = |J'|$-event differentially private, it follows that

$$\frac{Pr[\mathcal{M}(S_J) = R]}{Pr[\mathcal{M}(S'_J) = R]} = \frac{Pr[\mathcal{M}(S_p) = R]}{Pr[\mathcal{M}(S'_p) = R]} \leq \exp(\epsilon).$$ □

4.2.2 Improving the Baseline. However, due to the TEAS and TINAR effect, we hypothesize that one can have a higher utility than with w-event DP baseline in many cases. First, w-event mechanisms scale Laplace noise proportionally to the global sensitivity, but the temporal sensitivity $\Delta_Q^t(\Phi)$ tends to be smaller if there are no inter-individual constraints. Therefore, scaling Laplace noise with the temporal sensitivity is expected to improve data utility. In particular, this is the case for time stamps t where no policy is relevant, as one can publish the true statistics there. From a correctness point of view, one can in general replace the global sensitivity in any w-event mechanism with the temporal one and obtains a Swellfish-private mechanism. We use such mechanisms in our experimental study to investigate the influence of the TEAS effect. However, mechanisms using sampling and dynamic budget allocation may presume that the sensitivity is constant over time. An example is RescueDP [23] whose adaptive sampling component needs the sensitivity of the *next*, so far unknown sampling point to calculate the sampling point.

Second, the number of time stamps $\delta(J)$ to be protected in a relevance interval J might be different for each interval. So it may be much smaller than w. Therefore, distributing the budget in line with our composition theorem should improve data utility further. As the composition theorem is not constructive, there generally are many possibilities to this end. However, as the budget-distribution strategy of a mechanism typically builds on its sampling strategy, we now propose Swellfish-private mechanisms featuring typical sampling strategies. They range from pre-defined sampling rates over permanent sampling to data-adaptive sampling, exploiting both effects.

4.2.3 Proposed Mechanisms. In general, a Swellfish-private mechanism $\mathcal{M}$ consists of private sub-mechanisms $\mathcal{M}_t$, structured as stated in Algorithm 2. Each mechanism publishes the true query results if there are no relevant policies. Otherwise, it decides whether to sample the query results, and depending on this decision, perturbs the query results with some output perturbation budget ϵ_t^{op}, or approximates the results with the perturbed query results r_l of the last sampling point l. The mechanisms that follow differ in the procedures isSamplingPoint() and budgetAllocation(), that are given as follows.

Algorithm 2 Swellfish Mechanism-Framework

1: **procedure** $\mathcal{M}_t(\epsilon_t, \Phi, l)$
2: **if** $|\Phi_t| == 0$ **then return** $Q(D_t)$ ▷ no relevant policies
3: **else**
4: **if** isSamplingPoint(t) **then** ▷ poss. samp. budget ϵ_t^s
5: $\epsilon_t^{\text{op}} \leftarrow$ budgetAllocation(..)
6: $r_t = Q(D_t) + \text{Lap}(\frac{\Delta_t^Q}{\epsilon_t^{\text{op}}})$ ▷ output perturbation
7: $l = t$ ▷ $\epsilon_t = \epsilon_t^s + \epsilon_t^{\text{op}}$
8: **else** $r_t = r_l$ ▷ approximation, $\epsilon_t = \epsilon_t^s$
9: **end if**
10: **end if**
11: **return** r_t
12: **end procedure**

Predefined-Rate Sampling with UnicornIS. The mechanism UnicornIntervalSample (UnicornIS) samples at most once per relevance interval, and assigns the full budget $\epsilon_t^{\text{op}} = \epsilon$ available to each relevance interval to perturb the respective query result. Specifically, the first sampling time stamp is $t = 0$, and the second time stamp t' is the one for which $\Phi_{t'} \cap \Phi_0 = \emptyset$ holds, i.e., the policies relevant at time $t = 0$ are not relevant any more. So it is Swellfish-private by design. As the mechanism samples infrequently, its error is dominated by the approximation error at non-sampling time stamps. So it is expected to perform well if the query results are constant over time, or if the stream policies have short relevance intervals.

Permanent Sampling with UnicornPS. In case the query results vary over time, sampling more frequently may improve utility. So we propose the mechanism UnicornPermanentSample (UnicornPS) that implements those two procedures as follows. First, it samples every time stamp, i.e., isSamplingPoint() always returns true. Second, budgetAllocation() features a budget-distribution strategy that takes advantage of our composition theorem, cf. Algorithm 3.

This strategy is an ensemble of two sub-strategies targeting at different distributions of policies over the lifetime of the stream. The aim is to spent in each relevance interval the *entire* budget ϵ – and not more (violating privacy) or less (non-optimal utility). Furthermore, we aim to distribute the budget preferably homogeneous over each relevance interval, to achieve constant high utility over time. To achieve this, both sub-strategies target at different distributions of stream policies over time. The first sub-strategy, named *uniform*, distributes the budget uniformly over each relevance interval, being is a baseline strategy from literature [9, 14]. With respect to our aim, one sees that if there is only one relevant policy during a relevance interval, it spends exactly budget ϵ available for this interval. However, in case there is more than one relevant policy, it spends never more, but mostly less than ϵ. The rationale is that, in Line 16, for a given time stamp t, as final budget ϵ_t^{op}, the mechanism uses the *minimum* of all policies relevant at time t, to not violate privacy. If this budget ϵ_t^{op} is smaller than the policy-specific budget $\epsilon_{t,\phi}^{\text{op}}$, the mechanism spends less than ϵ budget in the relevance interval of this policy. We say that, if $\epsilon_t^{\text{op}} < \epsilon_{t,\phi}^{\text{op}}$, policy ϕ is dominated by another policy. So, for dominated policies, we do not achieve the mentioned aim. To encounter this, we propose the second sub-strategy, named *absorb-and-distribute*. It spends the difference between the final budget ϵ_t^{op} spent actually, and policy-specific budget, at later time stamps where the policy is not dominated anymore (*budget absorption*). To this end, it determines the remaining budget $\epsilon_{\text{rm},\phi}^{\text{op}}$ that the mechanism can spend during the remaining time stamps in the relevance interval, and proposes to distribute this remaining budget uniformly among the remaining time stamps in J. As spending budget does not affect privacy if the budget is not in $\text{TOP}_{\delta(J)}$, the smallest budget in $\text{TOP}_{\delta(J)}$ does not count when calculating the remaining budget.

Data-Adaptive Sampling with Unicorn. Sampling every time stamp might be a waste of budget if there are times where the query results vary only slightly and the approximation error would be small. So we propose the mechanism Unicorn that does data-adaptive sampling, but uses the same budget allocation procedure as UnicornPS already involving to distributed budget saved by non-sampling.

In literature, there are two types of data-adaptive sampling. The first one is dissimilarity-based sampling [14]. Here, one samples iff the approximation error is higher than the expected perturbation error. To decide this, one determines the dissimilarity between the published value at the last sampling point and the current query result. However, as one uses the actual query result for the sampling decision, one must ensure that the procedure is private as well. This costs a part of the privacy budget. The alternative is PID-based sampling [9, 24] coming with no additional cost. Here, one determines the next sampling time stamp by using a PID controller known from electrical engineering. As mentioned, such a sampling strategy contradicts with the temporal sensitivity. So we focus on the first type of strategy, cf. Algorithm 4. In line with related work, Unicorn spends $\frac{\epsilon}{2}$ budget per relevance interval for the sampling decision as well as for output perturbation. However, different splits of the budget are possible.

Theorem 4.4. *The mechanism UnicornPS and Unicorn fulfill (ϵ, Φ)-Swellfish privacy.*

Proof: Intuitively, privacy of Unicorn holds for the following reason: Given arbitrary $\delta(J)$ time stamps in a relevance interval, at these time stamps, the mechanism spends at most $\frac{\epsilon}{2}$ for dissimilarity perturbation, as well as for output perturbation. As UnicornPS uses the same budget-allocation procedure, the following arguments prove that it spends at most ϵ budget for output perturbation at arbitrary $\delta(J)$ time stamps in a relevance interval.
Formally, consider Unicorn, and let J be a relevance interval, and $\mathcal{X} = \text{TOP}_{\delta(J)}\{\epsilon_t = \epsilon_t^{\text{s}} + \epsilon_t^{\text{op}} | t \in J\}$. We prove that (1) $\sum_{\epsilon_t \in \mathcal{X}} \epsilon_t^{\text{s}} \leq \frac{\epsilon}{2}$ and (2) $\sum_{\epsilon_t \in \mathcal{X}} \epsilon_t^{\text{op}} \leq \frac{\epsilon}{2}$. Then, with (1), (2) and Theorem 4.2, the claim follows. Equation (1) holds, as $\sum_{\epsilon_t \in \mathcal{X}} \epsilon_t^{\text{s}} \leq \sum_{\epsilon_t \in \mathcal{X}} 0.5 \cdot \frac{\epsilon^t}{\delta(J)} = \frac{\epsilon}{2}$ in the procedure IsSamplingPoint(). Concerning ϵ_t^{op}, we discern between the following cases: If there is only one relevant policy, and the mechanism samples every time stamp, then the mechanism uses sub-strategy 1 for budget allocation only. in consequence, Equation (2) holds for the same reasons as Equation (1). Otherwise, sub-strategy 2 distributes the saved or dominated budget over the remaining time stamps. Here, Line 7 in budgetAllocation() ensures Equation (2). □

Algorithm 3 Budget allocation of UnicornPS ($\epsilon^{\text{op}} = \epsilon$) and Unicorn($\epsilon^{\text{op}} = 0.5 \cdot \epsilon$).

1: **procedure** budgetAllocation($t, \epsilon^{\text{op}}, \Phi$)
2: $\epsilon_t^{\text{op}} = \infty$
3: **for** $\phi \in \Phi_t$ **do**
4: // *Sub-Strategy 1 – uniform:*
5: $\epsilon_{t,c1}^{\text{op}} \leftarrow \frac{\epsilon^{\text{op}}}{\delta(J)}$
6: // *Sub-Strategy 2 – absorb-and-distribute:*
7: $\epsilon_{\text{rm},\phi}^{\text{op}} \leftarrow \epsilon^{\text{op}} - \sum_{\epsilon_{t'}^{\text{op}} \in \mathcal{X}} \epsilon_{t'}^{\text{op}}$
where $\mathcal{X} = \text{TOP}_{\delta(J)}\{\epsilon_t^{\text{op}} | t \in J\}$
8: **if** no. sampling time stamps during $J \geq \delta(J)$ **then**
9: $\epsilon_{\text{rm},\phi}^{\text{op}} -=$ smallest pub. budget spent during J
10: **end if**
11: open_rel_ts $\leftarrow J$.end $- t + 1$
12: $\epsilon_{t,c2}^{\text{op}} \leftarrow \frac{\epsilon_{\text{rm},\phi}^{\text{op}}}{\text{open\_rel\_ts}}$
13: $\epsilon_{t,\phi}^{\text{op}} \leftarrow \max\{\epsilon_{t,c1}^{\text{op}}; \epsilon_{t,c2}^{\text{op}}\}$
14: **end for**
15: $\epsilon_t^{\text{op}} = \min_{\phi \in \Phi_t}\{\epsilon_{t,\phi}^{\text{op}}\}$
16: **return** ϵ_t
17: **end procedure**

Algorithm 4 Sampling Strategy of Unicorn.

1: **procedure** isSamplingPoint(t, ϵ^s, Φ,Q)
2: $\epsilon_t^{\text{s}} = 0.5 \cdot \frac{\epsilon}{\max_{\phi \in \Phi_t} \delta(J)}$
3: dis $\leftarrow |Q(D_t) - r_l| + \text{Lap}(\frac{\Delta_t^Q}{\epsilon_t^{\text{s}}})$
4: $\epsilon_t^{\text{op}} \leftarrow$ budgetAllocation(t, ϵ, Φ), $\lambda \leftarrow \frac{\Delta_t^Q}{\epsilon_t^{\text{op}}}$
5: **if** dis $>$ λ **then**
6: **return** True
7: **else return** False
8: **end if**
9: **end procedure**

5 CASE STUDY – POWER-CONSUMPTION MONITORING

In this section, we evaluate Swellfish privacy experimentally in the form of a case study. Its objectives are to quantify the utility improvement of the TEAS and TINAR effect, and to compare the utility of Swellfish-private mechanisms to w-event competitors from literature, including the state-of-the-art. We first describe the methodology, and second state and discuss the results.

5.1 Methodology

We perform an intrinsic evaluation, to quantify the effects, as well as an extrinsic evaluation comparing our mechanisms with the state-of-the-art. Our experiments rely on the use case of power-consumption monitoring. In both evaluations, we aim at using (1) a broad range of competitors, (2) real-world power consumption data streams featuring different query results over time and (3) realistic policy collections of private households. In the remainder of this section, we describe how we deal with these challenges.

5.1.1 Mechanism Selection. Selecting meaningful competitors is challenging due to the wide range of existing w-event mechanisms. We select the mechanisms for our study as follows. For reproducibility, the implementations are publicly available[1].

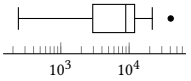
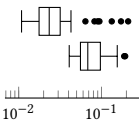
Intrinsic evaluation. To evaluate the TEAS effect, we compare mechanisms exploiting this effect with mechanisms that are identical on an algorithmic level, but use the the temporal sensitivity instead of the global one. As mentioned, this is not possible for every mechanism. As stated in Table 2, we use as competitors the mechanisms Sample, Uniform, BD and BA proposed in the original article on w-event DP [14]. While Uniform samples every time stamp, Sample samples every w-th time stamp, and BD and BA use

[1] https://git.scc.kit.edu/nb0387/swellfish-public

Table 2: Mechanisms compared in the intrinsic evaluation.

	w-event	vs.	Swellfish	Implementation
TEAS Effect				
■	Uniform [14]	vs.	TSUniform	use Δ_t^Q
■	Sample [14]	vs.	TSSample	use Δ_t^Q
■	BD [14]	vs.	TSBD	use Δ_t^Q
■	BA [14]	vs.	TSBA	use Δ_t^Q
TINAR effect				
	w-event	vs.	Swellfish	
■	Uniform [14]	vs.	TINARUniform	UnicornPS using ΔQ and SubStrategy 1 only
■	Sample [14]	vs.	TINARSample	UnicornIS using ΔQ

Table 3: Statistics of generated policy collections.

Parameters w-event competitors		
Parameter		Value
Global sensitivity	ΔQ	27.57
Window size per policy collection Φ	w	
Estimation of MAE Improvement of Uniform by Effect		
Effect	Calculation	Value
TEAS effect	$\varnothing_t \frac{\Delta_Q^t}{\Delta Q}$	
TINAR effect	$\varnothing_t \frac{\max_{\phi \in \Phi_t} \delta(J)}{w}$	

dissimilarity-based sampling, but differ in the budget-allocation strategy. The prefix 'TS' indicates that the respective mechanism uses the temporal sensitivity. As discussed, the composition theorem is not constructive. This means that there is no general rule to adapt a mechanism proposed in literature, such that it exploits the TINAR effect. This makes it challenging to evaluate the influence of this effect. However, we can design unbiased counterparts for the Uniform and Sample mechanism. To this end, we keep the key idea of their non-data adaptive budget distribution strategy, by using adjustments of the mechanisms UnicornPS and UnicornIS, see Table 2. No such unbiased mechanisms for BD and BA exist.

Extrinsic evaluation. Here, we compare (1) our proposed mechanisms UnicornIS, UnicornPS and Unicorn to (2) the current state-of-the-art w-event DP mechanism RescueDP [24] that cannot exploit any effect, and (3) the best variant of the w-event mechanisms used in the intrinsic evaluation. Preliminary experiments indicate that UnicornPS, UnicornIS, TSBD and TSBA outperform their competitors from Table 2. So to address (3), we include TSBD and TSBA in our extrinsic evaluation.

5.1.2 Data Streams. We use the GEFCom 2012 data set [13] already used in [8]. This real-world data consists of the summed-up hourly power consumption values of 20 different US zones over 4.5 years ($p = 152,277$). The zones have different average query results over time. E.g., zones 4 and 8 have the smallest (0.5 and 3.77 MW) consumption, while zone 18 has the highest one with 213.57 MW.

5.1.3 Generation of Policy Collections. Literature reveals that there are no freely available policy collections, but proposes generators simulating usage patterns of appliances in private households. As consequence, we use a such one to generate policy collections. We use the one proposed in [11]. We generate 55 households , which is the number of households in a current project [2]. Each household results in one policy collection. For each household, *each* appliance-usage pattern scheduled by the generator becomes one stream policy. This means that we protect all appliance usages challenging our approach, as one needs to fulfill many privacy goals. The stream policy $\phi = (\mathcal{A}, P, C(J), J)$ per appliance usage is generated as follows: The set $\mathcal{A}$ of activities is given by the appliances in the household, and we have no constraints ($C(J) = \emptyset$). We set $P = P_T^{d,\theta}$, where threshold θ is the power an appliance consumes, and T is the length of the respective appliance usage pattern. The relevance interval J is $4 \cdot T$ time stamps long and is located around the usage time scheduled by the generator. Multiple concatenated patterns of one appliance result in one stream policy either. Table 3 lists statistics of the resulting policy collections. It reveals the window length w we have to use for our w-event competitors, and it illustrates the expected MAE improvement for mechanisms sampling every time stamp. We set the global sensitivity of our competitors to $\Delta Q = 27.57$, which is the sum of consumption values associated with all possible activities. For reproducibility, the policy collections are publicly available[3].

5.1.4 Statistical Soundness. As all mechanisms used in our study are based on adding random noise, we repeat every experiment for every combination of data stream, mechanism and policy collection 100 times to eliminate statistical bias. We measure data utility in the usual way, using the mean absolute (MAE) and relative (MRE) error. The higher the error is, the lower is the data utility. For brevity, we focus on the MRE results – the key results are the same with both metrics. We perform all experiments for $\epsilon = 1.0$. We tested other values as well and observed that the key results are equal.

5.2 Results

We now state and discuss the results of our experiments. As the policy collections feature different temporal sensitivities and maximum numbers of possible goal-affected time stamps, we use boxplots displaying the distribution of error values among the collections.

5.2.1 Intrinsic Evaluation. Figures 4 and 5 show the decrease of the MRE due to the TEAS and TINAR effect. A value of 1 means that both mechanisms provide the same data utility. A smaller value indicates a decrease of the MRE and a greater one an increase. The zones are ordered by increasing average query result over time.

[2] esquire-projekt.de

[3] https://git.scc.kit.edu/nb0387/swellfish-public

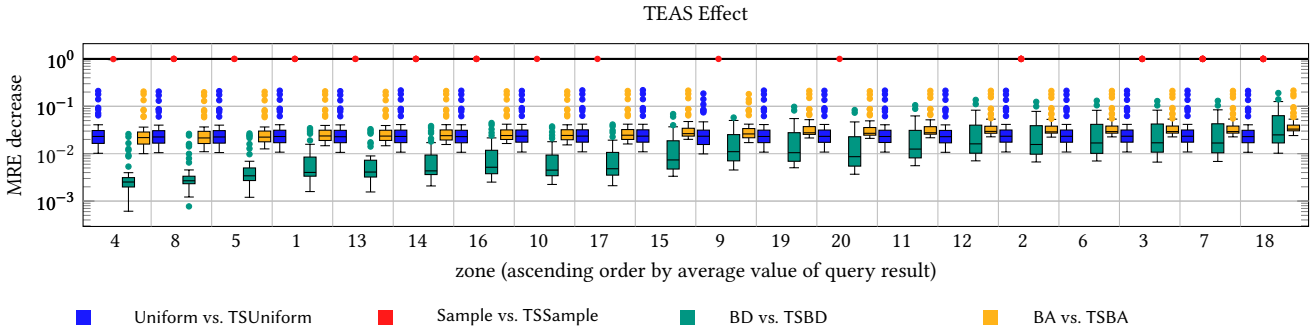


Figure 4: Results of the intrinsic evaluation – TEAS effect.

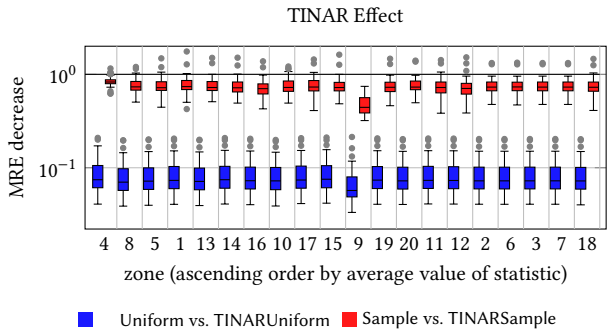


Figure 5: Results of the intrinsic evaluation – TINAR effect.

TEAS Effect. We observe three key findings: (1) General large error decrease, (2) no decrease if a fixed sampling rate is used, and (3) highest decrease for BD. Regarding (1), for all mechanisms except Sample, the MRE decreases by one to three orders of magnitude. The decrease becomes smaller for zones with higher power consumption. This is expected, and in line with previous results [8]. Regarding (2), exploiting the TEAS effect does not influence the utility of Sample. The rationale is as follows. Generally, mechanisms profit from the TEAS effect at sampling time stamps only. Sample samples every w-th time stamp only. In consequence, its MRE is dominated by the approximation error at non-sampling time stamps, which is nearly independent from the sensitivity used. Regarding (3), BD profits most from the TEAS effect. As a result of its budget-allocation strategy, BD samples at the beginning of each window only. As the windows are long, it samples infrequently, meaning that its MRE is also dominated by the approximation error. While TSBA samples infrequently as well, it additionally publishes true query results in the meantime, if the temporal sensitivity is zero. This reduces the total approximation error significantly. However, though BA has the same sampling strategy as BD, BA profits much less from the TEAS effect. By its design, after sampling, BA has to skip various time stamps where it must not publish. Thus, it cannot publish true query results at the skipped time stamps.

TINAR Effect. Considering Figure 5, we observe (1) an MRE decrease by an order of magnitude for the Uniform mechanism, but (2) an increase of MRE for a few policy collections of Sample. This means that, first, Uniform profits from the TINAR effect a lot. This is expected, as it samples every time stamp with a reduced noise scale. Furthermore, we observe a small MRE decrease for Sample for most collections. This is because TINARSample samples more often than Sample. However, it samples at most once per relevance interval, which is rare, leading to a small improvement only. Despite this general MRE decrease, there is an increase of MRE for a few policy collections. The rationale is that the time stamps where TINARSample and Sample sample differ. As the sampling strategies of both mechanisms is not data-adaptive, Sample may select the better sampling time stamps, leading to a lower MRE.

5.2.2 Extrinsic Evaluation. Figure 6 shows the MRE of the competitors (upper plot) and of Swellfish-framework mechanisms (bottom plot). To assess the influence of sampling on mechanism utility, we add the mechanism Unicorn* into our study. It is a non-private variant of Unicorn featuring perfect sampling. In other words, it uses the budget only for query result perturbation, i.e., $\epsilon^{\text{op}} = \epsilon$, and does not perturb the dissimilarity value in the procedure IsSamplingPoint(). We observe three key findings we now discuss: (1) Best utility by mechanisms from Swellfish framework, (2) UnicornIS is best for publishing small query results, and (3) marginal difference in utility of Swellfish framework mechanisms performing dynamic budget allocation. Regarding (1), in general, the competitors from literature perform worse than our mechanisms except for UnicornIS. This is expected, as they exploit only the TEAS (TSBD, TSBA) or even none of the effects (RescueDP). Regarding (2), considering our mechanisms only, we observe that UnicornIS performs worst for all zones except for Zone 4, having the smallest average query result. For this zone, it also performs better than the competitors. As, additionally, the MRE of UnicornIS is almost independent from the zone, this mechanism is suitable continuous publishing query results that are small. Regarding (3), by comparing UnicornPS with Unicorn and its non-private variant Unicorn*, we observe small utility differences. This indicates that almost perfect sampling decisions yield only little utility improvement in the Swellfish-framework.

Summing up, exploiting the TEAS and TINAR effect improves the utility of adjusted existing mechanisms by orders of magnitude. Furthermore, it allows to design high-utility Swellfish-private mechanisms. The results indicate that, to this end, a good budget allocation is more important than a good sampling strategy.

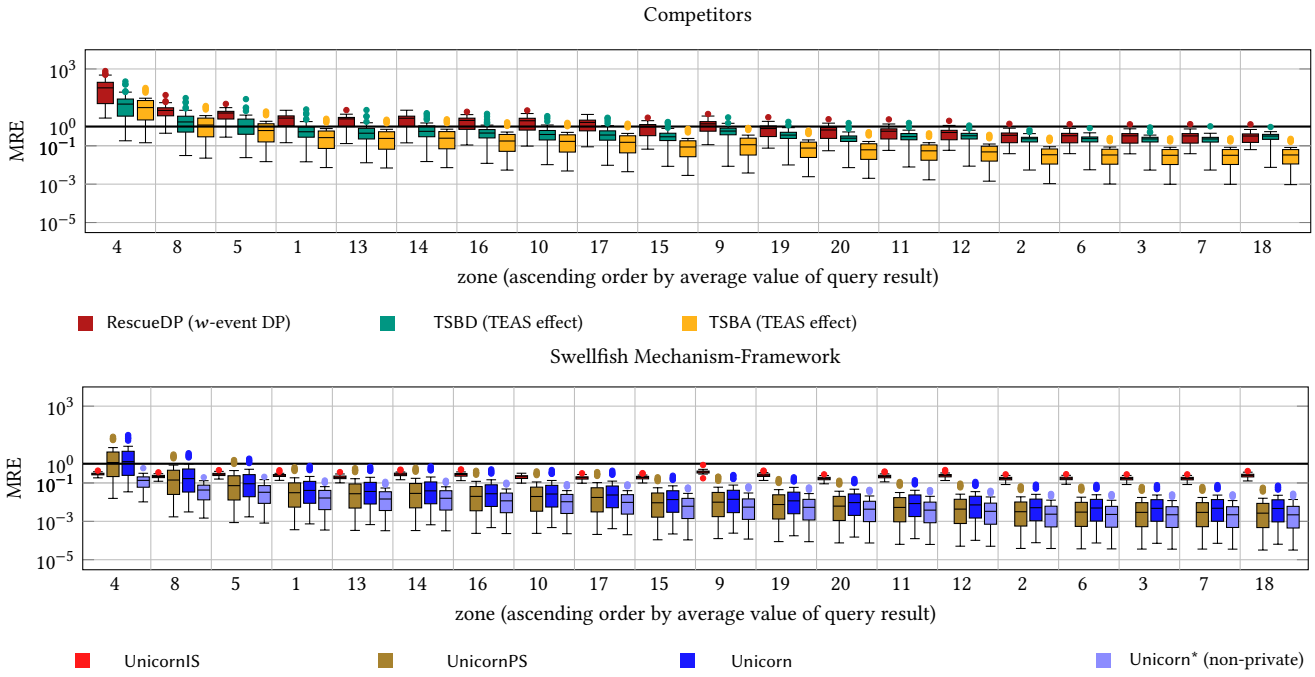


Figure 6: Results of the extrinsic evaluation.

6 GENERALIZATION OF CASE STUDY

The results from the case study indicate that – in the power-consumption use case – exploiting the effects improves data utility from one to three orders of magnitude. However, it is natural to ask what improvement to expect for other use cases. Abstractly, for non-sampling mechanisms, we can estimate the MAE improvement of TEAS knowing the average temporal sensitivity, i.e., predict the improvement of TSUniform over Uniform.

Theorem 6.1. *Given a policy collection Φ, the improvement of the MAE of TSUniform over Uniform is given by*

$$\frac{MAE\ TSUniform}{MAE\ Uniform} = \varnothing_t \frac{\Delta_Q^t}{\Delta Q}.$$

Proof: According to its definition, the mechanism Uniform adds Laplace noise with scale $\lambda_t^U = \frac{\Delta Q \cdot \epsilon}{w}$ to the query result at time t. Here, $w = \max_{\phi \in \Phi} \delta(J)$. Similarly, TSUniform uses scale $\lambda_t^{\text{TSU}} = \frac{\Delta_Q^t \cdot \epsilon}{w}$. As the scales equal the expected MAE, we obtain the claim by dividing both scales. □

Similarly, knowing the average size of the maximum $\delta(J)$ over all stream policies, we can estimate the improvement of the TINAR effect in isolation.

Theorem 6.2. *Given a policy collection Φ, the improvement of the MAE of TINARUniform over Uniform is given by*

$$\frac{MAE\ TINARUniform}{MAE\ Uniform} = \varnothing_t \frac{\max_{\phi \in \Phi_t} \delta(J)}{w},$$

where $w = \max_{\phi \in \Phi} \delta(J)$.

Proof: By definition, the mechanism Uniform adds Laplace noise with scale $\lambda_t^U = \frac{\Delta Q \cdot \epsilon}{w}$ to the query result at time t. Similarly, TINARUniform uses scale $\lambda_t^{\text{TIU}} = \frac{\Delta_Q \cdot \epsilon}{\max_{\phi \in \Phi_t}}$. As the scales equal the expected MAE, we obtain the claim by dividing both scales. □

The respective values for the case study are given in Table 3. The MAE values we obtained in our study are in line with them. We now give an intuition regarding the expected temporal sensitivity and number of goal-affected time stamps we expect in the location and physical activity monitoring use case. To illustrate, the TEAS effect tends to be high if the global sensitivity of the query is high, and if there are few sensitive privacy goals. In the location-monitoring use case, the global sensitivity equals 1, as there are no concurrent activities. The temporal sensitivity is either 0 (no relevant policy) or 1. Thus, one profits from the TEAS effect if there are few relevant policies. However, one should expect a smaller improvement than in the case study, as the difference between global and temporal sensitivity is smaller. This is different in the physical-activity-monitoring use case, as the global sensitivity is given by the number of activities an individual can perform, and we expect that an individual usually wants to hide only a few activities. Further, the TINAR effect is high if relevance intervals are – compared to the number of goal-affected time stamps – long. An example is hiding short trajectories in long relevance intervals.

7 RELATED WORK

Related work proposes DP-based privacy frameworks to bound the sensitivity, or number of protected time stamps, as well as frameworks that focus on a specific use case.

Bounding Sensitivity. Bounding the sensitivity has been done before in the static setting. The smoothed sensitivity [21] is a smooth upper bound on the local sensitivity and thus higher than our temporal sensitivity. As discussed, Swellfish privacy is a generalization of Blowfish [12] to streams. Following results from [12], in the absence of constraints, this also holds for Pufferfish privacy [15, 16]. Metric-based privacy [3] features distance-based policies. However, all these approaches are defined for the static setting and do not take time-dependent relevance into account.

Bounding Number of Protected Time Stamps. Bounding the number of time stamps one has to protect has been done before in the streaming setting. However, event-level DP [6] hides only one event, and w-event DP [14] hides all possible patterns anywhere in the stream. Both frameworks cannot take time-dependent relevance into account, lacking the notion of time-dependent goals. Much work exists on designing new w-event mechanisms. The latest ones exploit features of the streams, like small query results [24]. Many of them use sampling [18, 24] and filtering [4, 24] techniques. We can show that Swellfish privacy inherits post-processing immunity from the DP framework, and therefore, filtering can be used. However, they are orthogonal to exploiting time-dependent relevance. Thus, their evaluation is beyond the scope of this paper.

Use-Case Specific Frameworks. For the power-consumption use case, specific frameworks that implement individual privacy goals exist [10, 17, 22]. But they are not suitable for data streams or focus on privacy goals and guarantees different from differential privacy. Work that focuses on specifics of the location-monitoring use case [1] and hides *l-trajectories* is applicable to streams, but does not account for the time-dependent relevance problem.

8 CONCLUSIONS AND FUTURE WORK

In this paper, we consider the time-dependent relevance of privacy goals when continuously publishing differentially-private query results. To this end, we propose Swellfish privacy, a privacy framework exploiting time-dependent relevance to increase utility by exploiting two effects. The effects are the time-variant sensitivity and the time-variant number of affected query results, that related work cannot exploit by design. In order to exploit them, our notion of a policy collection allows individuals to specify combinations of possibly concurrent time-dependent privacy goals. Then, the iterative definition of neighboring databases, and the assignment of the privacy budget to relevance intervals in the privacy definition, ensures that privacy goals with overlapping relevance intervals are hidden concurrently. To exploit the effects in privacy mechanisms, we propose two tools. A realistic case study in the power-consumption use case shows that adjusting existing mechanisms so that they exploit the effects leads to one to three orders of magnitude utility improvement. At the same time, designing new mechanisms with the Swellfish mechanism framework yields mechanisms outperforming such adjusted existing mechanisms significantly. We generalize the study by specifying how to determine the expected utility improvement of the effects given an arbitrary policy collection.

We see various directions for future work. As usual, policy collections are presumed to be public knowledge. However, individuals may deem their policy collections sensitive information. So we will investigate how to keep them secure with appropriate encryption frameworks. Next, as our study reveals that budget allocation yields the highest utility improvement, we aim to investigate budget-allocation strategies in more detail.

REFERENCES

[1] Y. Cao and M. Yoshikawa. 2016. Differentially Private Real-Time Data Publishing over Infinite Trajectory Streams. *IEICE Trans. Inf. Syst.* 99, 1 (2016), 163–175.

[2] Y. Cao, M. Yoshikawa, Y. Xiao, and L. Xiong. 2019. Quantifying Differential Privacy in Continuous Data Release Under Temporal Correlations. *IEEE Trans. Knowl. Data Eng.* 31, 7 (2019), 1281–1295.

[3] K. Chatzikokolakis, M. Andrés, N. Bordenabe, and C. Palamidessi. 2013. Broadening the Scope of Differential Privacy Using Metrics. In *Proc. Int'l Symp. on Privacy Enhancing Technologies (PETS) (LNCS)*, Vol. 7981. Springer, 82–102.

[4] Y. Chen, A. Machanavajjhala, M. Hay, and G. Miklau. 2017. PeGaSus: Data-Adaptive Differentially Private Stream Processing. In *Proc. Int'l Conf. on Computer and Communications Security (CCS)*. ACM, 1375–1388.

[5] C. Dwork. 2011. Differential Privacy. In *Encyclopedia of Cryptography and Security*. Springer.

[6] C. Dwork, M. Naor, T. Pitassi, and G. Rothblum. 2010. Differential privacy under continual observation. In *Proc. Symp. on Theory of Computing (STOC)*. ACM, 715–724.

[7] C. Dwork and A. Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* 9, 3-4 (2014), 211–407.

[8] G. Eibl et al. 2018. The influence of differential privacy on short term electric load forecasting. *Energy Inform* 1, 48 (2018).

[9] L. Fan and L. Xiong. 2014. An Adaptive Approach to Real-Time Aggregate Monitoring With Differential Privacy. *IEEE Trans. Knowl. Data Eng.* 26, 9 (2014), 2094–2106.

[10] F. Fioretto and P. Van Hentenryck. 2019. OptStream: Releasing Time Series Privately. *J. Artif. Intell. Res.* 65 (2019), 423–456.

[11] S. Gottwalt et al. 2011. Demand side management – A simulation of household behavior under variable prices. *Energy Policy* 39, 12 (2011), 8163–8174.

[12] X. He, A. Machanavajjhala, and B. Ding. 2014. Blowfish privacy: Tuning privacy-utility trade-offs using policies. In *Proc. Int'l Conf. on Management of Data (SIGMOD)*. ACM, 1447–1458.

[13] T. Hong, P. Pinson, and S. Fan. 2014. Global energy forecasting competition 2012. *Int. J. Forecast.* 30, 2 (2014), 357–363.

[14] G. Kellaris et al. 2014. Differentially Private Event Sequences over Infinite Streams. *Proc. VLDB Endow.* 7, 12 (2014), 1155–1166.

[15] S. Kessler, E. Buchmann, and K. Böhm. 2015. Deploying and Evaluating Pufferfish Privacy for Smart Meter Data. In *Proc. Int'l Conf. on Ubiquitous Intelligence and Computing (UIC)*. IEEE, 229–238.

[16] D. Kifer and A. Machanavajjhala. 2014. Pufferfish: A framework for mathematical privacy definitions. *ACM Trans. Database Syst.* 39, 1 (2014), 3:1–3:36.

[17] F. Laforet, E. Buchmann, and K. Böhm. 2015. Individual privacy constraints on time-series data. *Inf. Syst.* 54 (2015), 74–91.

[18] H. Li et al. 2015. Differentially private histogram publication for dynamic datasets: an adaptive sampling approach. In *Proc. Int'l Conf. on Information and Knowledge Management (CIKM)*. ACM, 1001–1010.

[19] C. Liu, S. Chakraborty, and P. Mittal. 2016. Dependence Makes You Vulnberable: Differential Privacy Under Dependent Tuples.. In *Annual Network and Distributed System Security Symposium (NDSS)*. The Internet Society.

[20] S. Muthukrishnan. 2005. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science* 1, 2 (2005).

[21] K. Nissim, S. Raskhodnikova, and A. Smith. 2007. Smooth sensitivity and sampling in private data analysis. In *Proc. Annual Symp. on Theory of Computing (STOC)*. ACM, 75–84.

[22] C. Tex et al. 2018. PrivEnergy – a Privacy Operator Framework Addressing Individual Concerns. In *e-Energy*. ACM, 426–428.

[23] Q. Wang et al. 2018. Real-time and spatio-temporal crowd-sourced social network data publishing with differential privacy. *TDSC* (2018).

[24] T. Wang et al. 2019. Adaptive Differentially Private Data Stream Publishing in Spatio-temporal Monitoring of IoT. In *Int'l Performance Computing and Communications Conf. (IPCCC)*. IEEE, 1–8.

[25] T. Zhu et al. 2015. Correlated differential privacy: Hiding information in non-IID data set. *IEEE Trans. Inf. Forensics Secur.* 10, 2 (2015), 229–242.

KIT Scientific Working Papers
ISSN 2194-1629

www.kit.edu