Patrick Ah-Fat and Michael Huth*

Protecting Private Inputs: Bounded Distortion Guarantees With Randomised Approximations

Abstract: Computing a function of some private inputs while maintaining the confidentiality of those inputs is an important problem, to which Differential Privacy and Secure Multi-party Computation can offer solutions under specific assumptions. Research in randomised algorithms aims at improving the privacy of such inputs by randomising the output of a computation while ensuring that large distortions of outputs occur with low probability. But use cases such as e-voting or auctions will not tolerate large distortions at all. Thus, we develop a framework for randomising the output of a privacy-preserving computation, while guaranteeing that output distortions stay within a specified bound. We analyse the privacy gains of our approach and characterise them more precisely for our notion of sparse functions. We build randomisation algorithms, running in linearithmic time in the number of possible input values, for this class of functions and we prove that the computed randomisations maximise the inputs' privacy. Experimental work demonstrates significant privacy gains when compared with existing approaches that guarantee distortion bounds, also for non-sparse functions.

Keywords: Quantitative information flow, utility metric

DOI 10.2478/popets-2020-0053
Received 2019-11-30; revised 2020-03-15; accepted 2020-03-16.

1 Introduction

Computing a function on private inputs while maintaining the confidentiality of those inputs is desired in a variety of different scenarios. In e-voting, auctions, financial audits or statistical benchmarking, parties that hold private data are willing to enter the computation of a function that may involve other participants, to the extent that their personal data is kept "confidential" throughout the process. More precisely, as the outcome of those computations is generally intended to be made available to others, the participants would like to be able to estimate and control the information that such public information might leak on their own private input.

Secure Multi-party Computation (SMC) [1, 2] provides cryptographic protocols that enable several participants to collaboratively compute a public function of their own private inputs without resorting to any trusted third party. Moreover, the security of those protocols ensures that no information will leak on the private inputs during the protocol, apart from information flowing from the intended public output. In Quantitative Information Flow (QIF) [3, 4], one acknowledges that computing a function of some private inputs inevitably — and sometimes, intentionally — reveals some information about the secret inputs. For example, this is the case in the inevitable information flow produced by the output of SMC. The aim is then to design and apply measures of uncertainty that allow one to quantify such leakage to inform decision making.

In the randomisation of the output in Differential Privacy (DP), the precision of the outputs is a statistical concept [5]. A randomised mechanism is evaluated by how likely its randomised output will be close to the original output, closeness being defined via some distance or utility function. Two prominent techniques for this are the Laplace mechanism [6] and the Exponential mechanism [7]. In the Laplace mechanism, the probability to return a given distorted output decreases exponentially with its distance to the original output. In the Exponential mechanism, the same exponential decrease occurs but for a utility function. While these approaches provide strong statistical guarantees, they may be unsuitable when large deviations from the original output are unacceptable even under low probability.

For example, an e-voting protocol that guarantees that the elected president could be any of the candidates who receive very few votes with a very low probability could be exploited by an attacker. Similarly, in auctions, users may not accept that one's revenue could be drastically lower than the actual one, even with low probability, or this may conflict with regulatory constraints.

In this paper, we therefore focus on randomising mechanisms that ensure that the output of the ran-

Patrick Ah-Fat: Imperial College London, E-mail: patrick.ah-fat14@imperial.ac.uk
***Corresponding Author: Michael Huth:** Imperial College London, E-mail: m.huth@imperial.ac.uk

domised computation will *always* be contained within a specifiable distance to the actual output, making such guarantees non-probabilistic. We will use concepts from QIF and DP, to formalise our notion of a *randomised approximation of a function* and to formulate theoretical bounds for the entropy gain that this class of functions can provide. We measure privacy gains through *min-entropy*, widely used in cryptography to measure the vulnerability of a secret. Our approach can be extended to other measures of entropy.

Contributions. We summarise our main contributions:

1. We first introduce the notion of randomised Δ-approximation of a function that adheres to a specifiable output distortion bound Δ.
2. We formally investigate the privacy gains that those randomised Δ-approximations can provide.
3. We define *sparse* functions and characterise the privacy gains for those functions.
4. We design deterministic Δ-approximations under uniform and non-uniform prior beliefs and prove that they maximise the inputs' privacy.
5. We experimentally demonstrate significant privacy gains of Δ-approximations for polynomial and non-polynomial functions under more general entropies.

Outline of Paper. Related work is discussed in Section 2. Section 3 justifies the assumptions made in our work. We recall some concepts of QIF and DP in Section 4. Section 5 formalises our setting and assumptions. Our randomised approximation and their privacy gains is the subject of Section 6. In Section 7, we characterise privacy gains for the class of sparse functions and design a deterministic approximation that maximises inputs' privacy. In Section 8, we build a deterministic approximation that maximises privacy gains under non-uniform prior beliefs. The effectiveness of our approximations is experimentally demonstrated in Section 9. Potential generalisations of our work are discussed in Section 10 and Section 11 concludes.

Notations. For a set D, its cardinality is $|D|$. Let $\Omega(D)$ be the set of random variables with range D, represented by probability distributions whose support is contained in D. For any integers a and b, we will write $[\![a; b]\!]$ for $\{z \in \mathbb{Z} \mid a \leq z \leq b\}$, and $a \bmod b$ denotes the residue of a modulo b. The floor of a real number x is denoted by $\lfloor x \rfloor$. Throughout, we present distributions as Python dictionaries, e.g., $\{4\colon \frac{1}{2}, 8\colon \frac{1}{2}\}$ represents the uniform distribution over $\{4, 8\}$. For n in $\mathbb{N}$, a *linear distribution* over $[\![1; n]\!]$ refers to the triangular distribution with mode n, i.e. to the distribution $\{k\colon \frac{2(n-k+1)}{n(n+1)} \mid 1 \leq k \leq n\}$. Given random variable X and value x, the event "$X = x$" will be abbreviated by "x" when there is no ambiguity, and its probability will be denoted by $p(x)$. Similarly, we write $\sum_{x \in D}$ for $\sum_x$ when D is determined from context. We write log for the logarithm in base 2.

2 Related Works

We discuss related work that constitutes the foundations and the motivations of our present work.

Secure Multi-party Computation. Secure Multi-party Computation [1, 2, 8–11] is a domain of cryptography that provides advanced protocols which enable several participants to compute a public function of their own private inputs without having to rely on any other trusted third party or any external authority. Those protocols enable the participants to compute a function in a decentralised manner, while ensuring that no information leaks about the private inputs, other than what can be inferred from the public output. The commonly called "acceptable leakage" which is further studied in this paper, is the information that can be inferred by an attacker about the private inputs given the knowledge of the public output alone.

Quantitative Information Flow. The purpose of Quantitative Information Flow (QIF) [3, 12] is to provide frameworks and techniques based on information theory and probability theory for measuring the amount of information that leaks from a secret. Shannon entropy [13] reflects the minimum number of binary questions required to recover a secret on average, while the min-entropy is an indicator of the probability to guess a secret in one try [3, 4, 14]. Richer measures such as Rényi entropy [15] and the g-entropy [16] have been introduced in order to quantify some specific properties of a secret. Generalised entropies have been proposed in order to unify those different concepts [17, 18].

Differential Privacy. Differential Privacy (DP) [5, 19] formalises privacy concerns and introduces techniques that provide users of a database with the assurance that their personal details will not have a significant impact on the output of the queries performed on the database. More precisely, it proposes mechanisms which ensure that the outcome of the queries performed on two databases differing in at most one element will be statistically indistinguishable. Moreover, minimising the distortion of the outcome of the queries while ensuring privacy is an important trade-off that governs DP. Al-

though DP is particularly suited for guaranteeing privacy in statistical computations involving a large number of parties, its effectiveness diminishes when a small number of parties are involved in the computation. For example, in a two-party computation, a DP mechanism would ensure that the output would not be sensibly affected when half of the data is changed. In this case, the utility of the computed function can thus be drastically impaired by the low number of parties. Precision guarantees in DP generally ensure that the probability of a DP mechanism to return a given distorted output diminishes exponentially with its distance — or more generally, its negative utility — to the non-distorted output. In contrast, our work is meant to make such distortion guarantees hard, yielding non-probabilistic bounds.

Information Flow in Secure Multi-party Computation. Recent works [17, 20, 21] have adapted techniques stemming from QIF to the setting of SMC in order to propose a model that allows us to reason about the acceptable leakage. In this model, the set of parties willing to compute a public function f is partitioned into three sets: a set of attackers, a set of targets and a set of spectators, holding the respective input vectors $\mathbf{x}_{\mathbb{A}}$, $\mathbf{x}_{\mathbb{T}}$ and $\mathbf{x}_{\mathbb{S}}$. The attackers are those parties willing to share the value of their inputs and to take advantage of the public output of the computation $f(\mathbf{x}_{\mathbb{A}}, \mathbf{x}_{\mathbb{T}}, \mathbf{x}_{\mathbb{S}})$ in order to learn as much information as possible on their targets' inputs, while the remaining parties are called spectators. From the point of view of the attackers, the inputs $\mathbf{x}_{\mathbb{T}}$ and $\mathbf{x}_{\mathbb{S}}$ are unknown values and are thus modelled as random variables $X_{\mathbb{T}}$ and $X_{\mathbb{S}}$, further deemed to be independent since targets and spectators are supposed to be honest parties who provide their inputs without being influenced by any other information. The attackers' prior belief on those inputs will represent the prior distributions $\pi_{\mathbb{T}}$ and $\pi_{\mathbb{S}}$ of those random variables. The output of the function f is then also considered as a random variable defined as $O = f(\mathbf{x}_{\mathbb{A}}, X_{\mathbb{T}}, X_{\mathbb{S}})$. The privacy of the targeted parties is then expressed as the conditional entropy $\mathrm{H}(X_{\mathbb{T}} \mid \mathbf{x}_{\mathbb{A}}, O)$ of the targeted inputs given knowledge of the attackers' inputs and the conditional knowledge of the output. The choice of the entropy measure H depends on the users' privacy concerns and is left general in [17].

Those works have also introduced some randomising mechanisms for enhancing the targeted parties' entropy. The main idea of their techniques is to randomise the output O with a random noise Φ contained in a restricted range so that the noise introduced by computing $O + \Phi$ does not exceed a given distortion bound. Those methods take inspiration from the Laplace mechanism [6], in that a critical assumption of their work is that this random noise Φ is independent of the function output O. They then formally investigate the relationships between $\mathrm{H}(X_{\mathbb{T}} \mid \mathbf{x}_{\mathbb{A}}, O)$ and $\mathrm{H}(X_{\mathbb{T}} \mid \mathbf{x}_{\mathbb{A}}, O + \Phi)$. They further design some methods for determining optimal probability distributions for this noise Φ in order to maximise $\mathrm{H}(X_{\mathbb{T}} \mid \mathbf{x}_{\mathbb{A}}, O + \Phi)$.

In contrast, in this paper we will randomise the output O of a secure computation with some noise which *does* depend on the output O. We will show that a noise that depends on O offers more flexibility that Algorithms 1 and 2 use to produce privacy that cannot be reached with a noise that is independent from O.

3 Objectives and Methodology

To motivate our objectives and convey our methodology, let us consider the informal setting where a set of parties wishes to collaboratively compute a public function f of their private inputs and we wish to protect the privacy of a set of inputs Y. Let Z be the set of all the other inputs of the computation. The aim is to control and reduce the information leaked by the public output $f(Y, Z)$ on private inputs Y. We do this by substituting the computation of f with that of a randomised function g that approximates the output of f while protecting the privacy of Y.

The rest of this section answers the following questions and, in doing so, explains our objectives and approach taken:

Q1 How should privacy be measured, i.e. how do we quantify information leakage on private inputs Y ?

Q2 What utility metric should we use in order to assess the output difference between functions f and g ?

Q3 What type of functions f and inputs Y and Z does our developed approach support ?

A1: Privacy measure. We think that there is no technical concept of privacy that is appropriate in all contexts. Rather, mapping such concepts to use cases through the latter's requirements is more appropriate — also when considering compliance with the EU General Data Protection Regulation (GDPR) [22], say. Differential privacy and quantitative information flow are examples of such measures of privacy. Although we believe that pursuing research in both directions is crucial, this work focuses on the quantitative approach and information leakage will be measured by means of min-

entropy, which is widely used in QIF and cryptography [4, 14, 23]. Our approach is not wedded to the latter: in Section 9 we experimentally demonstrate that our mechanisms offer significant privacy gains when the more general notion of g-entropy is used.

A2: Utility metric. How to measure the utility of a randomisation mechanism will similarly depend on requirements of the use case, and perhaps on user expectations. Most research on DP and QIF uses a statistical notion of some distance in order to characterise utility, where high utility means that the distortion is small with a high probability, and large distortions happen only with small probability [7, 24, 25]. We believe that other utility metrics can be better suited to the requirements and expectations of some use cases. Here, we therefore focus on a particular utility metric which intentionally differs from those used extensively in DP and QIF. This will allow us to build mechanisms that ensure that the distortion they introduce will *never* exceed a given threshold. Our work should thus raise awareness of the fact that the commonly accepted statistical notion of utility used in DP and QIF is not universally the best choice, and that the preference between different utility measures is highly dependent on the nature of a use case and its users.

Example 1 conveys that choosing a utility measure is not straightforward and argues that the ability to choose between a statistical utility measure and ours based on guaranteed bounds of distortions is a benefit.

Example 1. *Claire holds some private medical record y and wishes to know whether she is infected by a serious disease. To do so, she must enter the computation of a function f that involves private information z of others. The result $o = f(y, z)$ returned by that function is an integer between* 1 *and* 100*. If the result is lower or equal to* 50*, then Claire is not infected, otherwise she is infected. The result of this function must be public, as this serious disease is highly contagious. An analysis reveals that result o leaks sensitive, private information and that its publication would thus deeply invade her privacy. Therefore, the proposal is to not publish o but an integer approximation thereof. For this, the doctors let Claire choose between two different solutions:*

S1. *Approximation o_1 adds random noise to o such that the probability of o_1 is exponentially decreasing with its distance to o: approximation o_1 is equal to o with probability* 98%*, but could be arbitrarily far from o with a low, but non zero probability. For example, in the worst case where $o = 51$, then o_1 would be > 50 with a probability* 99% *and would thus lead to the correct diagnostic. Probabilities are selected for illustrative purposes. In contrast, if $o = 100$, then o_1 would be > 50 with probability* 99.99%*. To summarise, this approximation yields a correct diagnostic with a high probability of* 99% *in the worst case, but would always be prone to misdiagnosis with a probability of at least* 0.01%*.*

S2. *Approximation o_2 adds random noise of $-1, 0$ or $+1$ to the output o. In the worst case where o is equal to* 50 *or* 51*, this would thus introduce a high misclassification rate of* 33% *and would not be reliable information. However, in all the other cases, i.e. in* 98% *of the cases, this randomisation method would yield a* 100% *accurate diagnosis.*

Which solution **S1** *or* **S2** *should Claire choose? In the first case, Claire has very high probability for a correct diagnosis overall, but she would always run a small risk of misdiagnosis. In the second case, there is a small risk that Claire would not have enough information to have a reliable diagnosis, but she could have a* 100% *accurate diagnosis in the majority of the cases.*

We believe that answering this question might not be straightforward for Claire and that her choice could depend on the her own personality, background or personal judgement. Offering Claire such a choice, by explaining benefits and risks of each choice, is bound to lead to more informed use of privacy-preserving technology.

The choice of utility metric can also depend on the usage of the function to randomise. Statistical utilities, such as the expected gain recalled in Definition 6 or those used in DP for database access, have advantages when repeated privacy-preserving computations are required, so that closeness is achieved on average. A privacy budget may be used to accommodate such compositions of computation.

When computation is meant to be performed only once on some inputs, e.g. in e-voting, the notion of maximal distortion introduced in Definition 6 might be preferable to a statistical one — e.g. Claire might actually prefer this. In Example 1, a better expected gain would be achieved by o_1 from solution **S1** (98% against 33% for o_2) while o_2 from **S2** would yield a better utility in terms of maximal distortion (1 against 99 for o_1).

Naturally, it would be of interest to derive solutions that combine the benefits of statistical utilities with the non-probabilistic guarantee of hard distortion bounds. We mean to investigate this in future work and offer some first insights on this in Section 9.

Extant works study the trade-off between utility and privacy in a variety of scenarios [25, 26]. One can fix

a utility bound and then optimise for privacy. But giving users the choice of a utility that is more tailored to their needs might remove unnecessary utility constraints and offer them even higher privacy. There should therefore be value in offering such choice.

The second solution above might be prone to composition attack, e.g. invoking twice the same randomising technique to the same set of inputs would reveal more information on secret inputs y. DP can offer such robustness against composition attacks. We here work in the setting of SMC and are interested in randomising functions that are meant to be used in a unique computation that will not be repeated with the same inputs.

A3: Function and input types. Since our work means to enhance the privacy in secure multi-party computations, we follow the same assumptions that SMC protocols are based on and so work with polynomial functions on integer inputs and outputs.

While some of our results apply to the combinatorial concept of sparse functions, defined below, we aim to generalise these results to wider classes of integer functions. That doing this is feasible is corroborated by experiments on non-sparse functions in Section 9, where our approach also achieves high privacy gains.

4 Background

Quantitative Information Flow has provided a variety of entropy measures. We will focus on min-entropy [4, 16]:

Definition 1. *Let Y and O be discrete random variables with respective domains D_Y and D_O. The conditional Bayes vulnerability $\mathrm{V}(Y \mid O)$ of Y given O and the conditional min-entropy $\mathrm{H}(Y \mid O)$ of Y given O are:*

$$\begin{aligned} \mathrm{V}(Y \mid O) &= \sum_{o \in D_O} p(o) \max_{y \in D_Y} p(y \mid o) \\ \mathrm{H}(Y \mid O) &= -\log \mathrm{V}(Y \mid O) \end{aligned}$$

The relationship between inputs and outputs of a randomised process is formalised in QIF via the notion of statistical channel [23], which we recall next.

Definition 2. *Let $\mathbb{D}_Y$ and $\mathbb{D}_O$ be finite sets. Let C be a $|\mathbb{D}_Y| \times |\mathbb{D}_O|$ matrix.*

1. *Then C is a channel matrix with input domain $\mathbb{D}_Y$ and output domain $\mathbb{D}_O$ if all its elements are ranged between 0 and 1 and if the sum of each row equals 1. For all y in $\mathbb{D}_Y$ and o in $\mathbb{D}_O$, element $C[y, o]$ of the channel matrix C represents $p(o \mid y)$, and the triple $(\mathbb{D}_Y, \mathbb{D}_O, C)$ is called a channel.*
2. *Channel matrix C is* deterministic *if each row contains exactly one 1, i.e. that each input $y \in \mathbb{D}_Y$ is mapped to one output $o \in \mathbb{D}_O$ with probability 1.*

We now recall the notion of randomised mechanism [5].

Definition 3. *Let $\mathbb{D}_Y$ and $\mathbb{D}_O$ be finite sets. A randomised function $\mathscr{F}$ with domain $\mathbb{D}_Y$ and output range $\mathbb{D}_O$ is a function $\mathscr{F} \colon \mathbb{D}_Y \longrightarrow \Omega(\mathbb{D}_O)$ whose inputs y are elements of $\mathbb{D}_Y$ and which outputs a random variable $\mathscr{F}(y)$ ranged in $\mathbb{D}_O$. For all y in $\mathbb{D}_Y$ and o in $\mathbb{D}_O$, $p(\mathscr{F}(y) = o)$ denotes the probability that $\mathscr{F}(y)$ equals o.*

We represent randomised functions as channel matrices, to unify approaches and draw from results on channels. A randomised function $\mathscr{F} \colon \mathbb{D}_Y \longrightarrow \Omega(\mathbb{D}_O)$ with domain $\mathbb{D}_Y$ and range $\mathbb{D}_O$ represents and can be represented as a channel matrix between $\mathbb{D}_Y$ and $\mathbb{D}_O$: We can build one from the other by setting $C[y, o] = p(\mathscr{F}(y) = o)$ for all $y \in \mathbb{D}_Y$ and $o \in \mathbb{D}_O$. Based on this representational equivalence, we define *deterministic* functions:

Definition 4. *A deterministic function is a randomised function whose channel matrix is deterministic.*

We note that a set-theoretic function $f \colon \mathbb{D}_Y \longrightarrow \mathbb{D}_O$ with domain $\mathbb{D}_Y$ and range $\mathbb{D}_O$ represents and can be represented as a deterministic function $\mathscr{F} \colon \mathbb{D}_Y \longrightarrow \mathbb{D}_O$. This follows from the fact that we can build the one from the other by noting the equivalence $p(\mathscr{F}(y) = o) = 1$ iff $f(y) = o$ for all $y \in \mathbb{D}_Y$ and o in $\mathbb{D}_O$.

We next introduce the notions of image and inverse image of a randomised function, used subsequently:

Definition 5. *Let $\mathbb{D}_Y$ and $\mathbb{D}_O$ be finite sets, y in $\mathbb{D}_Y$, o in $\mathbb{D}_O$, and $\mathscr{F} \colon \mathbb{D}_Y \longrightarrow \Omega(\mathbb{D}_O)$ a randomised function. The image $\mathscr{F}_*(y)$ of y by $\mathscr{F}$ and the inverse image $\mathscr{F}^*(o)$ of o by $\mathscr{F}$ are:*

$$\begin{aligned} \mathscr{F}_*(y) &= \{o \in \mathbb{D}_Y \mid p(\mathscr{F}(y) = o) > 0\} \\ \mathscr{F}^*(o) &= \{y \in \mathbb{D}_X \mid p(\mathscr{F}(y) = o) > 0\} \end{aligned}$$

5 Information Flow of SMC

We now present our framework. Let n be a positive integer and $f \colon \mathbb{Z}^n \longrightarrow \mathbb{Z}$ be an n-ary function where n parties $P_1, \cdots, P_n$ hold the respective private integer inputs $x_1, \cdots, x_n$. These parties wish to compute the

output value of the function evaluated on their private inputs: $o = f(x_1, \cdots, x_n)$. Function f is public and the output o can be opened publicly. But the values of the secret inputs should remain private, apart from information leaked by the public output o.

Such a functionality can be achieved by delegation of this computation to a trusted third party, e.g. a secure cloud environment; use of SMC; use of secure hardware such as Intel's SGX, and so forth. The work we present here is agnostic to such choices as it focuses on the information leakage of the output o on the private inputs.

Assumption 1. *Attacker Ann is an external observer of this computation of o and aims to infer as much information as possible on some targeted private inputs.*

Let $S \subseteq [\![1; n]\!]$ and let $y = (x_k)_{k \in S}$ be a vector of the inputs indexed by S. Let $z = (x_k)_{k \in [\![1;n]\!] \setminus S}$ capture the remaining inputs. Vector y represents those inputs that Ann wishes to attack, i.e. Ann wants to learn as much information as possible on y. We write z to denote the remaining inputs. The pair $x = (y, z)$ denotes any element of the domain $\mathbb{D} = \mathbb{D}_Y \times \mathbb{D}_Z \subset \mathbb{Z}^n$.

Below, we will write $o = f(x)$ and $o = f(y, z)$ interchangeably. For all y in $\mathbb{D}_Y$, the partially evaluated function $f_y \colon \mathbb{D}_Z \longrightarrow \mathbb{D}_O$ satisfies $\forall z \in \mathbb{D}_Z \colon f_y(z) = f(y, z)$.

To Ann, the input vectors y and z are unknown. This uncertainty is modelled by random variables $Y \in \Omega(\mathbb{D}_Y)$ and $Z \in \Omega(\mathbb{D}_Z)$ and we set $X = (Y, Z)$. We further assume that Y and Z are independent, i.e. parties holding y and z cannot choose their inputs depending on the other parties' inputs. This is a desirable property in our use cases such as auctions and e-voting. The random variable $O = f(Y, Z)$ corresponds to the output of f and has domain $\mathbb{D}_O = \{f(y, z) \mid y \in \mathbb{D}_Y, z \in \mathbb{D}_Z\}$.

With this in place, we define a measure of privacy for input Y: the information that attacker Ann learns on her targeted input Y by learning the value of the output O will be measured by $\mathrm{H}(Y) - \mathrm{H}(Y \mid O)$. As $\mathrm{H}(Y)$ is a constant, we will focus on computing the conditional min-entropy of Y given O, expressed as $\mathrm{H}(Y \mid O)$.

6 Randomised Approximations

We now introduce randomised approximations, which add noise to the output of a given function f in order to improve the privacy of its inputs. We also introduce the utility measure Λ_{max} studied in this work.

Definition 6. *Let $\mathbb{D}_{O'}$ be a subset of $\mathbb{Z}$ and $\mathscr{G} \colon \mathbb{D} \longrightarrow \Omega(\mathbb{D}_{O'})$ a randomised function. Then:*

1. *$\mathscr{G}$ is a randomised approximation of f, denoted $\mathscr{G} \propto f$, if there is a randomised function $\mathscr{H} \colon \mathbb{D}_O \longrightarrow \Omega(\mathbb{D}_{O'})$, called output randomisation function, such that: $\forall x \in \mathbb{D} \colon \mathscr{G}(x) = \mathscr{H}(f(x))$. Moreover, $\mathscr{G}$ is a deterministic approximation of f, denoted $\mathscr{G} \propto^{\mathrm{d}} f$, if $\mathscr{H}$ is deterministic.*
2. *We define the maximal loss $\Lambda_{max}(\mathscr{G})$ as:*
$$\Lambda_{max}(\mathscr{G}) = \max_{\substack{x \in \mathbb{D} \\ o' \in \mathscr{G}_*(x)}} |f(x) - o'|$$
3. *The expected gain $\Gamma_{exp}(\mathscr{G})$ is defined in [27, 28] as:*
$$\Gamma_{exp}(\mathscr{G}) = \sum_{o \in \mathbb{D}_O} p(o) \cdot p(\mathscr{H}(o) = o)$$

Note that for a guess domain $\mathcal{W}$, the expected gain can be further refined with a generic gain function $g \colon \mathcal{W} \times \mathbb{D}_Y \longrightarrow [0, 1]$, which we set to the identity gain function g_{id} in this work, where $g_{id}(w, y) = 1$ if $w = y$ and 0 otherwise. Function Γ_{exp} will be used in the evaluations in Section 9. Our work focuses on designing algorithms that are optimal when the utility is fixed in terms of Λ_{max}, i.e. we want to randomise the output of f while imposing a maximal distortion threshold:

Definition 7. *For Δ in $\mathbb{N}$, a randomised approximation $\mathscr{G}$ of f is a randomised Δ-approximation of f, denoted $\mathscr{G} \propto_\Delta f$, if $\Lambda_{max}(\mathscr{G}) \leq \Delta$, i.e. if:*
$$\forall x \in \mathbb{D}, \forall o' \in \mathscr{G}_*(x) \colon |f(x) - o'| \leq \Delta$$
Moreover, $\mathscr{G}$ is a deterministic Δ-approximation of f if $\mathscr{G}$ is deterministic, denoted $\mathscr{G} \propto^{\mathrm{d}}_\Delta f$.

We provide theoretical bounds for the entropy gain that randomised Δ-approximations offer:

Theorem 1. *Given Δ in $\mathbb{N}$ and a randomised Δ-approximation $\mathscr{G}$ of f:* $\mathrm{H}(Y \mid \mathscr{G}(X)) \geq \mathrm{H}(Y \mid f(X))$.

Proof. Since $\mathscr{G}$ is a randomised approximation of f, there is a randomised function $\mathscr{H} \colon \mathbb{D}_O \longrightarrow \Omega(\mathbb{D}_{O'})$ such that: $\forall x \in \mathbb{D} \colon \mathscr{G}(x) = \mathscr{H}(f(x))$. The function $f \colon \mathbb{D}_X \longrightarrow \mathbb{D}_O$ can be represented as a deterministic function $\mathscr{F} \colon \mathbb{D}_X \longrightarrow \Omega(\mathbb{D}_O)$, which in turn can be represented as a channel matrix C_f with domain $\mathbb{D}_X$ and range $\mathbb{D}_O$. Let us also consider the channel matrix $C_{\mathscr{H}}$ of $\mathscr{H}$. Then, by Definition 11 from [23], the randomised function $\mathscr{G}$ represents a cascade of the channels $(\mathbb{D}_X, \mathbb{D}_O, \mathscr{F})$ and $(\mathbb{D}_O, \mathbb{D}_{O'}, \mathscr{H})$. The inequality then follows from the data-processing inequality for the min-entropy in Theorem 5 of [23]. □

Theorem 2. *Given Δ in $\mathbb{N}$ and a randomised Δ-approximation $\mathscr{G}$ of f we have:*

$$\mathrm{H}(Y \mid \mathscr{G}(X)) \leq \mathrm{H}(Y \mid f(X)) + \log(2\Delta + 1)$$

Proof. Since $\mathscr{G}$ is a randomised approximation of f, there is a randomised function $\mathscr{H}\colon \mathbb{D}_O \longrightarrow \Omega(\mathbb{D}_{O'})$ such that: $\forall x \in \mathbb{D}\colon \mathscr{G}(x) = \mathscr{H}(f(x))$. Recall the random variable $O = f(X)$. Let us define the random variable $O' = \mathscr{G}(X) = \mathscr{H}(O)$. By definition, we have $\mathrm{V}(Y \mid O') = \sum_{o'} \max_y p(y)\, p(o' \mid y)$. The law of total probabilities with conditional probabilities gives us

$$\mathrm{V}(Y \mid O') \quad = \quad \sum_{o'} \max_y p(y) \sum_o p(o' \mid y, o)\, p(o \mid y)$$

Moreover, by defining $\mathscr{G}$ as the composition of f and $\mathscr{H}$, and as discussed in Theorem 4 from [23], we know that for all $y \in \mathbb{D}_Y, o \in \mathbb{D}_O$ and $o' \in \mathbb{D}_{O'}$, if $p(o \mid y) > 0$ then $p(o' \mid y, o) = p(o' \mid o)$. We thus have $p(o' \mid y, o)\, p(o \mid y) = p(o' \mid o)\, p(o \mid y)$ which ensures that

$$\mathrm{V}(Y \mid O') = \sum_{o'} \max_y p(y) \sum_o p(o' \mid o)\, p(o \mid y) \qquad (1)$$

As the sum of positive numbers is not lower than their maximum, we have

$$\begin{aligned} \mathrm{V}(Y \mid O') \quad &\geq \quad \sum_{o'} \max_o \max_y p(y)\, p(o' \mid o)\, p(o \mid y) \\ &\geq \quad \sum_{o'} \max_{o \in \mathscr{H}^*(o')} \max_y p(y)\, p(o' \mid o)\, p(o \mid y) \end{aligned}$$

since $p(o' \mid o)$ is non-zero only when $o \in \mathscr{H}^*(o')$.

As the maximum of some elements is not lower than their average, we have

$$\begin{aligned} \mathrm{V}(Y \mid O') \quad \geq \quad & \sum_{o'} \frac{1}{|\mathscr{H}^*(o')|} \\ & \sum_{o \in \mathscr{H}^*(o')} \max_y p(y)\, p(o' \mid o)\, p(o \mid y) \end{aligned}$$

As $\mathscr{G}$ is a randomised Δ-approximation of f, for all $o' \in \mathbb{D}_{O'}$, we have $\mathscr{H}^*(o') \subseteq [\![o' - \Delta; o' + \Delta]\!]$ and thus $|\mathscr{H}^*(o')| \leq 2\Delta + 1$. Thus:

$$\begin{aligned} \mathrm{V}(Y \mid O') \quad \geq \quad & \frac{1}{2\Delta + 1} \sum_{o'} \\ & \sum_{o \in \mathscr{H}^*(o')} \max_y p(y)\, p(o' \mid o)\, p(o \mid y) \end{aligned}$$

The summation $\sum_{o'} \sum_{o \in \mathscr{H}^*(o')}$ ranges over all pairs (o, o') with $o \in \mathscr{H}^*(o')$, i.e. $o' \in \mathscr{H}_*(o)$. So we have:

$$\begin{aligned} \mathrm{V}(Y \mid O') \quad \geq \quad & \frac{1}{2\Delta + 1} \sum_{o} \\ & \left(\sum_{o' \in \mathscr{H}_*(o)} p(o' \mid o) \right) \max_y p(y)\, p(o \mid y) \\ \geq \quad & \frac{1}{2\Delta + 1} \sum_{o} \max_y p(y)\, p(o \mid y) \end{aligned}$$

Taking $-\log$ on both sides proves the claim. □

The following notions are useful for subsequent examples and for comparing our work to existing ones.

Definition 8. *1. We define the uniformly randomised Δ-approximation $\mathscr{G}_\Delta^{uni}$ of f by its output randomisation function $\mathscr{H}_\Delta^{uni}$ for all o in $\mathbb{D}_O$ and o' in $\mathbb{Z}$ as:*

$$p(\mathscr{H}_\Delta^{uni}(o) = o') = \begin{cases} \frac{1}{2\Delta+1} & \text{if } |o - o'| \leq \Delta \\ 0 & \text{otherwise} \end{cases}$$

2. A randomised approximation $\mathscr{G}$ of f, characterised by output randomisation $\mathscr{H}$, is independent if $\mathscr{H}(o)$ is independent of o for all o in $\mathbb{D}_O$.

3. An independently randomised approximation $\mathscr{G}_\Delta^{opt}$ of f is optimal if for all independently randomised Δ-approximations $\mathscr{G}$ of f, we have:

$$\mathrm{H}(Y \mid \mathscr{G}(X)) \leq \mathrm{H}(Y \mid \mathscr{G}_\Delta^{opt}(X))$$

4. We define the Δ-truncation $\mathscr{G}_\Delta^{trunc}$ of f by its deterministic output randomisation h_Δ^{trunc} as:

$$\forall o \in \mathbb{D}_O\colon h_\Delta^{trunc}(o) = o - (o \bmod (2\Delta + 1)) + \Delta$$

Approximation $\mathscr{G}_\Delta^{uni}$ maps each output to all the values that are within a distance Δ with equal probability while $\mathscr{G}_\Delta^{trunc}$ gathers all the values of intervals of length $2\Delta + 1$ around their central value with probability 1. Extant work has focused on independently randomised approximations of f. In particular, the work in [17] proposes some methods for building independently randomised approximations that are optimal. However, making the output of a randomised approximation $\mathscr{H}(o)$ independent of the concerned output o is restrictive. We relax this assumption and build bespoke randomised approximations of f that randomise the output depending on its value, which can lead to optimal schemes that indeed further enhance the inputs' privacy.

7 Sparse Function Randomisation

We give more precise estimations of the entropy gain that randomised approximations provide and design algorithms that maximise the privacy of targeted inputs Y. We do this for sparse functions, introduced next.

Definition 9. *For Δ in $\mathbb{N}$, function f is Δ-sparse if:*

$$\forall y \in \mathbb{D}_Y, \forall (z, z') \in \mathbb{D}_Z \times \mathbb{D}_Z : \Big(z \neq z'\Big) \implies \Big(|f(y,z) - f(y,z')| > \Delta\Big)$$

Sparsity generalises the notion of injectivity: f is 0-sparse iff for all y in $\mathbb{D}_Y$, f_y is injective. Also, for all Δ in $\mathbb{N}$, any $(\Delta+1)$-sparse function is Δ-sparse. We give examples of sparse functions when y and z are 1 dimensional. Function f defined as $f(y,z) = 2y + 4z$ is 3-sparse. For all $a \neq 0$ in $\mathbb{Z}$, function f defined as $f(y,z) = 2y + az$ is $(|a|-1)$-sparse. Term $2y$ can be replaced by any function of y without affecting the sparsity of f. Sparsity of f may depend on the input domains. Function f with $f(y,z) = 2y + z^2$, e.g., is 4-sparse for $\mathbb{D}_Z = \{z \in \mathbb{Z} \mid z \geq 2\}$ but when $\mathbb{D}_Z = \mathbb{Z}$ we have $f(y,z) = f(y,-z)$ which violates sparsity. We can evaluate the privacy of Y for sparse functions more precisely.

Assumption 2. *Let $\mathscr{G} : \mathbb{D} \longrightarrow \Omega(\mathbb{D}_{O'})$ be a randomised Δ-approximation of f and $\mathscr{H} : \mathbb{D}_O \longrightarrow \Omega(\mathbb{D}_{O'})$ its randomised function. The random variable O' is $\mathscr{G}(X) = \mathscr{H}(O)$ and f is assumed to be (2Δ)-sparse.*

We can characterise the information that $\mathscr{G}(X)$ leaks on private input Y more precisely, which allows us to build optimal approximations for maximising this entropy.

Theorem 3. *We have:*

$$\mathrm{V}(Y \mid \mathscr{G}(X)) = \sum_{o'} \max_{o} p(o' \mid o) \max_{y} p(y)\, p(o \mid y)$$

Proof. From Equation (1), we have:

$$\mathrm{V}(Y \mid O') \quad = \quad \sum_{o'} \max_{y} p(y) \sum_{o} p(o' \mid o)\, p(o \mid y)$$

Now, $p(o' \mid o) \neq 0$ only when $o \in \mathscr{H}^*(o')$, and so:

$$\mathrm{V}(Y \mid O') = \sum_{o'} \max_{y} p(y) \sum_{o \in \mathscr{H}^*(o')} p(o' \mid o)\, p(o \mid y)$$

However, as $\mathscr{G}$ is a randomised Δ-approximation of f, for all o' in $\mathbb{D}_{O'}$ and o in $\mathscr{H}^*(o')$, we have $|o - o'| \leq \Delta$. Thus, for all o_1 and o_2 in $\mathscr{H}^*(o')$, the triangular inequality ensures that $|o_1 - o_2| \leq |o_1 - o'| + |o_2 - o'| \leq 2\Delta$. As f is (2Δ)-sparse, we then get that for all y in $\mathbb{D}_Y$, there is at most one z in $\mathbb{D}_Z$ such that $f(y,z) \in \mathscr{H}^*(o')$. By definition of $\mathbb{D}_{O'}$, for all o' in $\mathbb{D}_{O'}$ and y in $\mathbb{D}_Y$, there exists a unique o in $\mathscr{H}^*(o')$ such that $p(o \mid y)$ is non-zero. Thus, the sum $\sum_{o \in \mathscr{H}^*(o')}$ can be substituted for $\max_{o \in \mathscr{H}^*(o')}$, which implies:

$$\mathrm{V}(Y \mid \mathscr{G}(X)) = \sum_{o'} \max_{o \in \mathscr{H}^*(o')} p(o' \mid o) \max_{y} p(y)\, p(o \mid y)$$

Since $p(o' \mid o)$ is non-zero only when o is in $\mathscr{H}^*(o')$, this concludes the proof. □

We can simplify the expression of $\mathrm{V}(Y \mid \mathscr{G}(X))$ when Y and Z are uniformly distributed.

Corollary 1. *Let Y and Z be uniformly distributed. Then we have:*

$$\mathrm{V}(Y \mid \mathscr{G}(X)) = \frac{1}{|\mathbb{D}_Y| \cdot |\mathbb{D}_Z|} \cdot \sum_{o'} \max_{o} p(o' \mid o)$$

Proof. From Theorem 3, we know that $\mathrm{V}(Y \mid \mathscr{G}(X)) = \sum_{o'} \max_o p(o' \mid o) \max_y p(y)\, p(o \mid y)$. By definition of $\mathbb{D}_O$, for all o in $\mathbb{D}_O$ this is a pair (y,z) in $\mathbb{D}_Y \times \mathbb{D}_Z$ with $f(y,z) = o$ such that $p(o \mid y) = p(z)$. Since Y and Z are uniform, this concludes the proof. □

Deterministic $\mathscr{G}$ have a compact form of $\mathrm{V}(Y \mid \mathscr{G}(X))$:

Corollary 2. *Let Y and Z be uniformly distributed and $\mathscr{G}$ a deterministic approximation of f. Then we have:*

$$\mathrm{V}(Y \mid \mathscr{G}(X)) = \frac{|\mathbb{D}_{O'}|}{|\mathbb{D}_Y| \cdot |\mathbb{D}_Z|}$$

Proof. As $\mathscr{G}$ is deterministic, so is $\mathscr{H}$ by definition. For all o' in $\mathbb{D}_{O'}$ and o in $\mathscr{H}^*(o')$, we have $p(o' \mid o) = p(\mathscr{H}(o) = o')$ and thus $p(o' \mid o) = 1$ as $\mathscr{H}$ is deterministic. By Corollary 1, the claim follows. □

Based on Corollary 2, when inputs are uniformly distributed and $\mathscr{G}$ is deterministic, then $\mathrm{H}(Y \mid \mathscr{G}(X))$ is maximised when the number of possible distorted outputs $|\mathbb{D}_{O'}|$ is minimised. This guides us in building a randomised Δ-approximation $\mathscr{G}$ of f that maximises the inputs' privacy. Algorithm 1 builds $\mathscr{G}_\Delta^{DET}$, a deterministic Δ-approximation of f that minimises the number of possible distorted outputs $|\mathbb{D}_{O'}|$ under those assumptions. It explores possible outputs in $\mathbb{D}_O$ in increasing order, and lets the deterministic output randomisation function h merge as many of these outputs into the same distorted output o' as possible, given distortion bound Δ. Green lines use variables that are only used in the proof of Theorem 4, not in the actual computation.

Algorithm 1 Deterministic output merging $\mathscr{G}_\Delta^{DET}$

Inputs: $f\colon \mathbb{D} \longrightarrow \mathbb{Z}, \mathbb{D}_Y \subseteq \mathbb{Z}, \mathbb{D}_Z \subseteq \mathbb{Z}, \Delta \in \mathbb{N}$
Output: $\mathscr{G}_\Delta^{DET}\colon \mathbb{D} \longrightarrow \Omega(\mathbb{Z})$

1: $\mathbb{D}_O \leftarrow f(\mathbb{D}) = \{f(y,z) \mid y \in \mathbb{D}_Y, z \in \mathbb{D}_Z\}$
2: $o' \leftarrow \min(\mathbb{D}_O) - \Delta - 1$
3: /* $j \leftarrow 0$ */
4: **for** $o \in \mathbb{D}_O$ in increasing order **do**
5: **if** $o - o' > \Delta$ **then**
6: $o' \leftarrow o + \Delta$
7: /* $c_j \leftarrow o$ */
8: /* $j \leftarrow j + 1$ */
9: $h(o) \leftarrow o'$
10: $\mathscr{G}_\Delta^{DET} \leftarrow h \circ f$
11: **return** $\mathscr{G}_\Delta^{DET}$

The complexity of this algorithm is dominated by computing and sorting the output space $\mathbb{D}_O$, which is in $\mathcal{O}(n \log n)$ where $n = |\mathbb{D}_Y| \cdot |\mathbb{D}_Z|$. This computes deterministic Δ-approximations of f that maximise the targeted input's privacy for uniformly distributed inputs:

Corollary 3 (of Theorem 4)**.** *Let Y and Z be uniformly distributed. Then:*

$$\mathscr{G} \propto_\Delta^d f \implies \mathrm{H}(Y \mid \mathscr{G}(X)) \leq \mathrm{H}(Y \mid \mathscr{G}_\Delta^{DET}(X))$$

Proof. This is based on the fact that Algorithm 1 minimises the number of possible distorted outputs. This will be a direct implication of the next Theorem 4. □

For any randomised approximation of f with uniformly distributed inputs, $\mathscr{G}_\Delta^{DET}$ has the highest entropy for Y over all possible randomised Δ-approximations of f:

Theorem 4. *Let Y and Z be uniformly distributed over their respective domains $\mathbb{D}_Y$ and $\mathbb{D}_Z$. Then we have:*

$$\mathrm{H}(Y \mid \mathscr{G}(X)) \leq \mathrm{H}(Y \mid \mathscr{G}_\Delta^{DET}(X))$$

Proof. Let C be the set of variables c_j created in Line 7 of Algorithm 1. Each time this line is executed, a new distorted output o' is created in Line 6. By Corollary 2, $\mathrm{V}(Y \mid \mathscr{G}_\Delta^{DET}(X))$ equals $|C|/(|\mathbb{D}_Y| \cdot |\mathbb{D}_Z|)$. We show that the vulnerability $\mathrm{V}(Y \mid \mathscr{G}(X))$ of an arbitrary randomised Δ-approximation of f is greater or equal to that. For all j in $[\![1; |C|]\!]$, set $\mathbb{C}_j = [\![c_j - \Delta; c_j + \Delta]\!]$. We first show that for all j in $[\![1; |C| - 1]\!]$, we have $\max \mathbb{C}_j < \min \mathbb{C}_{j+1}$. Line 6 ensures that $h(c_j) = c_j + \Delta$, where h is the deterministic output randomisation function used in Algorithm 1. And c_{j+1} is created when the test in Line 5 succeeds. Thus, $c_{j+1} - h(c_j) > \Delta$ which implies $c_{j+1} - (c_j + \Delta) > \Delta$ and so $c_{j+1} - c_j > 2\Delta$, which ensures that $\max \mathbb{C}_j < \min \mathbb{C}_{j+1}$. So $(\mathbb{C}_j)_{1 \leq j \leq |C|}$ is a collection of disjoint subsets of $\mathbb{Z}$. By Corollary 1:

$$\begin{aligned} \mathrm{V}(Y \mid \mathscr{G}(X)) &= \frac{1}{|\mathbb{D}_Y| \cdot |\mathbb{D}_Z|} \cdot \sum_{o'} \max_o p(o' \mid o) \\ &\geq \frac{1}{|\mathbb{D}_Y| \cdot |\mathbb{D}_Z|} \cdot \sum_{j=1}^{|C|} \sum_{o' \in \mathbb{C}_j} \max_o p(o' \mid o) \end{aligned}$$

Fixing a value of o in $\mathbb{D}_O$ does not decrease this vulnerability, thus we may set $o = c_j$ so that:

$$\begin{aligned} \mathrm{V}(Y \mid \mathscr{G}(X)) &\geq \frac{1}{|\mathbb{D}_Y| \cdot |\mathbb{D}_Z|} \cdot \sum_{j=1}^{|C|} \sum_{o' \in \mathbb{C}_j} p(o' \mid O = c_j) \\ &\geq \frac{1}{|\mathbb{D}_Y| \cdot |\mathbb{D}_Z|} \cdot \sum_{j=1}^{|C|} p(O' \in \mathbb{C}_j \mid O = c_j) \end{aligned}$$

As $\mathscr{G}$ is a Δ-approximation of f, for all j in $[\![1; |C|]\!]$ we have $\mathscr{H}_*(c_j) \subseteq \mathbb{C}_j$, and so $p(O' \in \mathbb{C}_j \mid O = c_j) = 1$. □

We illustrate the privacy gains that this optimal randomisation $\mathscr{G}_\Delta^{DET}$ offers, and compare it to the independently randomised approximations in [17].

Example 2. *Let Y and Z be 1-dimensional and uniformly distributed over $[\![1; 30]\!]$ and a in $[\![0; 9]\!]$. Function $f\colon \mathbb{Z} \times \mathbb{Z} \longrightarrow \mathbb{Z}$ with $f(y,z) = 3y^2 - ayz + 2y - 4z$ is 3-sparse and thus 2Δ-sparse for $\Delta = 1$. We compare the efficiency of our randomised function $\mathscr{G}_\Delta^{DET}$ to methods $\mathscr{G}_\Delta^{uni}, \mathscr{G}_\Delta^{opt}$ and $\mathscr{G}_\Delta^{trunc}$ where $\mathscr{G}_\Delta^{opt}$ refers to an optimal independently randomised Δ-approximation of f as introduced in [17]. Figure 1 shows the entropies $\mathrm{H}(Y \mid f(X))$ and $\mathrm{H}(Y \mid \mathscr{G}(X))$ for the aforementioned functions $\mathscr{G}$, which are all greater than $\mathrm{H}(Y \mid f(X))$, consistent with Theorem 1. Also, the privacy gains that those methods offer do not exceed $\log(2\Delta + 1)$, consistent with Theorem 2. Methods $\mathscr{G}_\Delta^{uni}$, $\mathscr{G}_\Delta^{opt}$, and $\mathscr{G}_\Delta^{trunc}$ produce comparable entropies. But $\mathscr{G}_\Delta^{DET}$ offers an additional and significant privacy gain compared to the previous methods. By Theorem 4, $\mathscr{G}_\Delta^{DET}$ has the greatest entropy of all randomised Δ-approximations.*

8 Non-Uniform Priors

We now design a *deterministic* approximation for sparse functions with *non-uniform* inputs that maximises the targeted input's privacy.

Assumption 3. *In this section, inputs Y and Z are* non-*uniform, $\pi_Y \in \Omega(\mathbb{D}_Y)$ and $\pi_Z \in \Omega(\mathbb{D}_Z)$ denote their*

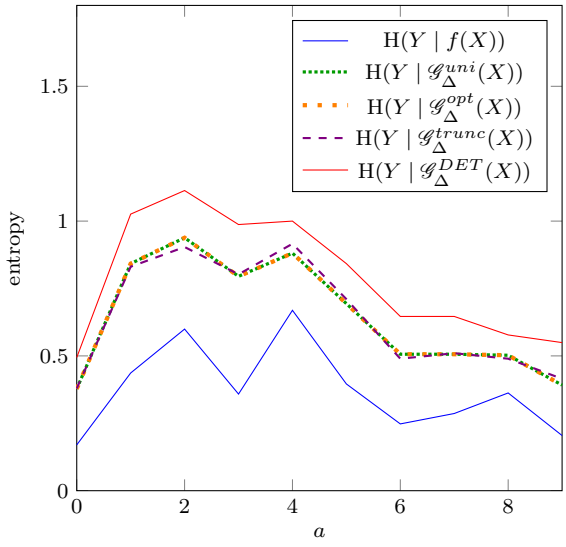

Fig. 1. Privacy gains of $\mathscr{G}_\Delta^{DET}$ compared to those of other randomised Δ-approximations.

respective probability distributions, the prior knowledge that an external observer has on the inputs Y and Z.

Our aim can be phrased as this optimisation problem:

$$\underset{\mathscr{G} \propto_\Delta^d f}{\text{maximise}} \ \mathrm{H}(Y \mid \mathscr{G}(X)) \qquad \text{(OPT)}$$

In words, given a function $f\colon \mathbb{D} \longrightarrow \mathbb{D}_O$ and a distortion bound Δ in $\mathbb{N}$, we want to find a deterministic approximation $\mathscr{G}$ of f which maximises $\mathrm{H}(Y \mid \mathscr{G}(X))$. This involves building a (deterministic) function h such that $\mathscr{G} = h \circ f$ will be a deterministic Δ-approximation of f that best protects the inputs' privacy. To this end, let $(\mathbb{Z}^{\mathbb{D}_O})_\Delta$ denote the set of functions $h\colon \mathbb{D}_O \longrightarrow \mathbb{Z}$ which introduce a distortion bounded by Δ and $[\mathbb{Z}^{\mathbb{D}_O}]_\Delta$ as set of increasing functions in $(\mathbb{Z}^{\mathbb{D}_O})_\Delta$:

$$(\mathbb{Z}^{\mathbb{D}_O})_\Delta = \{h \in \mathbb{Z}^{\mathbb{D}_O} \mid \forall o \in \mathbb{D}_O\colon |h(o) - o| \leq \Delta\}$$

and $[Z^{\mathbb{D}_O}]_\Delta$ is the set of those h in $(\mathbb{Z}^{\mathbb{D}_O})_\Delta$ such that $o_1 \leq o_2$ implies $h(o_1) \leq h(o_2)$ for all (o_1, o_2) in $(\mathbb{D}_O)^2$.

By Theorem 3 and as $\mathscr{G}$ ought to be deterministic, our optimisation problem OPT can be rephrased as:

$$\underset{h \in (\mathbb{Z}^{\mathbb{D}_O})_\Delta}{\text{minimise}} \ \sum_{o'} \max_{o \in h^{-1}(o')} \max_y p(y)\, p(o \mid y)$$

To reduce the search space, we prove that any function in $(\mathbb{Z}^{\mathbb{D}_O})_\Delta$ can be made increasing without increasing the resulting vulnerability:

Theorem 5. *For all h in $(\mathbb{Z}^{\mathbb{D}_O})_\Delta$, there is a k in $[\mathbb{Z}^{\mathbb{D}_O}]_\Delta$ such that $\mathrm{V}(Y \mid k(O)) \leq \mathrm{V}(Y \mid h(O))$.*

This theorem, proved in Appendix A, reduces the search space of OPT via an equivalent optimisation problem:

$$\underset{h \in [\mathbb{Z}^{\mathbb{D}_O}]_\Delta}{\text{minimise}} \ \sum_{o'} \max_{o \in h^{-1}(o')} \max_y p(y)\, p(o \mid y) \qquad (2)$$

We can use this to design a dynamic algorithm for building $\mathscr{G}_\Delta^{DYN}$ as a deterministic Δ-approximation of f that maximises the inputs' privacy over all deterministic approximations. The intuition of that algorithm is to build a hash map d such that for all o in $\mathbb{D}_O$, entry $d[o]$ equals $\max_y p(y)\, p(o \mid y)$. We will then build an array A indexed from 1 which contains the sorted elements of $\mathbb{D}_O$. We will build an array S indexed from 1 such that for all j in $[\![1; |\mathbb{D}_O|]\!]$, we have:

$$S[j] = \min_{h \in [\mathbb{Z}^{\mathbb{D}_O}]_\Delta} \sum_{o'} \max_{\substack{o \in h^{-1}(o') \\ o \leq A[j]}} d[o]$$

Note that $S[0]$ is initialised to 0. We build array S by visiting all the outputs in increasing order, such that at the end of the algorithm, $s[|\mathbb{D}_O|]$ will contain the minimal vulnerability that can be achieved by a deterministic Δ-approximation of f. The idea is here that each value o_1 in $\mathbb{D}_O$ can be mapped with other values o_2 into the same output o'. As h is assumed to be increasing, all values of o that are merged into the same output o' need to be consecutive with respect to the ordered set $\mathbb{D}_O$. Let j be in $[\![1; |\mathbb{D}_O|]\!]$. We will thus allow value $A[j]$ to be merged with a number of its predecessors. We record in the cell array $N[j]$ the index of the smallest *neighbour* of $A[j]$, i.e. the smallest value of i such that $A[i]$ has to be merged with $A[j]$ in order to achieve $S[j]$. We note that $A[j] - A[N[j]]$ cannot exceed 2Δ as we need to build a Δ-approximation of f. This design is shown in Algorithm 2 and illustrated in Figure 2. The number of iterations of the inner loop in Line 16 for dynamically building the elements of arrays S and N is bounded by constant 2Δ. Thus the complexity of this algorithm is also dominated by computing and sorting the output space $\mathbb{D}_O$, which is in $\mathcal{O}(n \log n)$ where $n = |\mathbb{D}_Y| \cdot |\mathbb{D}_Z|$.

The randomisation that this algorithm produces is the best deterministic Δ-approximation of f, i.e. $\mathscr{G}_\Delta^{DYN}$ is a solution of the optimisation problem OPT:

Theorem 6. *For all randomised Δ-approximation $\mathscr{G}$ of f with $\mathscr{G} \propto_\Delta^d f$: $\mathrm{H}(Y \mid \mathscr{G}(X)) \leq \mathrm{H}(Y \mid \mathscr{G}_\Delta^{DYN}(X))$.*

Proof. Let d be a hash map with $d[o] = \max_y p(y)\, p(o \mid y)$ for all o in $\mathbb{D}_O$. As f is assumed to be Δ-sparse for some Δ in $\mathbb{N}$, for all y in $\mathbb{D}_Y$, f_Y is injective and for all z in $\mathbb{D}_Z$ we have that $p(O = f(y, z) \mid y) = p(z)$. Moreover, for a given j in $[\![1; |\mathbb{D}_O|]\!]$, Lines 13 to 23 of Algorithm 2

Algorithm 2 Construction of the optimal deterministic Δ-approximation $\mathscr{G}_\Delta^{DYN}$

Inputs: $f : \mathbb{D} \longrightarrow \mathbb{D}_O, \Delta \in \mathbb{N}, \pi_Y \in \Omega(\mathbb{D}_Y), \pi_Z \in \Omega(\mathbb{D}_Z)$

Output: $\mathscr{G}_\Delta^{DYN} : \mathbb{D} \longrightarrow \Omega(\mathbb{Z})$

$\mathbb{D}_O \leftarrow f(\mathbb{D}) = \{f(y,z) \mid y \in \mathbb{D}_Y, z \in \mathbb{D}_Z\}$
$d \leftarrow \{o\colon 0 \mid o \in \mathbb{D}_O\}$
for $(y,z) \in \mathbb{D}_Y \times \mathbb{D}_Z$ **do**
    $o \leftarrow f(y,z)$
    $d[o] \leftarrow \max(d[o], p(y) * p(z))$
$A \leftarrow \mathbb{D}_O$ as a sorted array, indexed from 1
/* dynamically build S and N */
$S[0] = 0$
$j_{min} = 1$
for j in $[\![1; |\mathbb{D}_O|]\!]$ **do**
    while $A[j] - A[j_{min}] > 2\Delta$ **do**
        $j_{min} \leftarrow j_{min} + 1$
    $m \leftarrow d[A[j]]$
    $s_{min} \leftarrow m + S[j-1]$
    $n_{min} \leftarrow j$
    for i in $\{j-1, \cdots, j_{min}\}$ in (-1) steps **do**
        $m \leftarrow \max(m, d[A[i]])$
        $s \leftarrow m + S[i-1]$
        if $s < s_{min}$ **then**
            $s_{min} \leftarrow s$
            $n_{min} \leftarrow i$
    $S[j] \leftarrow s_{min}$
    $N[j] \leftarrow n_{min}$
/* reconstruct output randomisation function h */
$j \leftarrow |\mathbb{D}_O|$
while $j > 0$ **do**
    $n \leftarrow N[j]$
    $o' \leftarrow \lfloor (A[n] + A[j])/2 \rfloor$
    for i in $[\![n; j]\!]$ **do**
        $h[i] \leftarrow o'$
    $j = n - 1$
return $\mathscr{G}_\Delta^{DYN}$ where $\mathscr{G}_\Delta^{DYN} = h \circ f$

explicitly solve an optimisation problem whose optimal value is assigned to $S[j]$, specified by:

$$\underset{j_{min} \le i \le j}{\text{minimise}} \quad (S[i-1] + \max_{i \le l \le j} d[A[l]]) \tag{3}$$

where j_{min} is the minimal index with $A[j] - A[j_{min}] \le 2\Delta$, i.e. $A[j_{min}]$ is the smallest element that an output randomisation function h can merge with j.

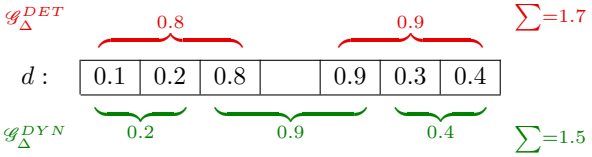

Fig. 2. Run of $\mathscr{G}_\Delta^{DET}$ and $\mathscr{G}_\Delta^{DYN}$ on example array d for $\Delta = 1$. In red, $\mathscr{G}_\Delta^{DET}$ minimises the number of distorted outputs o' while $\mathscr{G}_\Delta^{DYN}$ in green minimises $\sum_{o'} \max_{o \in h^{-1}(o')} d[o]$. Braces represent functions h and we display each $\max_{o \in h^{-1}(o')} d[o]$.

Let k be in $[\![1; |\mathbb{D}_O|]\!]$ and define the property P_k as:

$$P_k \equiv \left(S[k] = \min_{h \in [\mathbb{Z}^{\mathbb{D}_O}]_\Delta} \sum_{o'} \max_{\substack{o \in h^{-1}(o') \\ o \le A[k]}} d[o] \right) \tag{4}$$

We will prove P_k by induction on k. We know that P_1 holds. Indeed, as we have set $S[0] = 0$, Equation (3) ensures that the algorithm will set $S[1] = A[1]$. Moreover, the right hand side of Equation (4) also equals $A[0]$ as its innermost max contains only this element. For the inductive step, let j be in $[\![2; |\mathbb{D}_O|]\!]$ and assume that P_k holds for all k in $[\![1; j-1]\!]$. To prove P_j, we need to consider the choice of the value of $h(A[j])$. Let us take h in $[\mathbb{Z}^{\mathbb{D}_O}]_\Delta$. As h is increasing, the value of $h(A[j])$ can either be greater than that of $h(A[j-1])$, or equal to $h(A[l])$ for all l in $[\![i; j-1]\!]$ such that $A[j] - A[i] \le 2\Delta$. Let i be minimal in $[\![1; j]\!]$ with $h(A[j]) = h(A[i])$. Then, by distinguishing the case where $o' = h(A[j])$, we have

$$\begin{aligned} \sum_{\substack{o \in h^{-1}(o') \\ o \le A[j]}} d[o] &= \max_{o \in h^{-1}(o')} d[o] + \sum_{o'} \max_{\substack{o \in h^{-1}(o') \\ o \le A[i-1]}} d[o] \\ &= \max_{A[i] \le o \le A[j]} d[o] + \sum_{o'} \max_{\substack{o \in h^{-1}(o') \\ o \le A[i-1]}} d[o] \end{aligned}$$

In this case, choosing an h in $[\mathbb{Z}^{\mathbb{D}_O}]_\Delta$ is equivalent to choosing the aforementioned value of i, and to then select h_1 in $[\mathbb{Z}^{\mathbb{D}(i)}]_\Delta$ where $\mathbb{D}(i) = \{o \in \mathbb{D}_O \mid o \le A[i-1]\}$. The right hand side of Equation (4) thus becomes:

$$\min_{\substack{1 \le i \le j \\ A[j] - A[i] \le \Delta}} \min_{h \in [\mathbb{Z}^{\mathbb{D}(i)}]_\Delta} \left(\max_{\substack{A[i] \le o \\ o \le A[j]}} d[o] + \sum_{o'} \max_{\substack{o \in h^{-1}(o') \\ o \le A[i-1]}} d[o] \right)$$

which can be rewritten as:

$$\min_{\substack{1 \le i \le j \\ A[j] - A[i] \le \Delta}} \left(\max_{\substack{A[i] \le o \\ o \le A[j]}} d[o] + \min_{h \in [\mathbb{Z}^{\mathbb{D}(i)}]_\Delta} \sum_{o'} \max_{\substack{o \in h^{-1}(o') \\ o \le A[i-1]}} d[o] \right)$$

By definition of j_{min} and assumption on $S[i-1]$, this expression solves the optimisation problem in (3). □

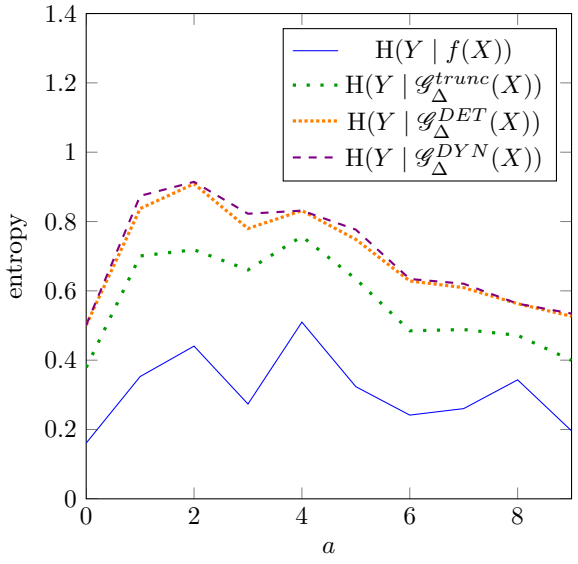

Fig. 3. Privacy gains for $\mathscr{G}_\Delta^{DYN}$ compared to those of other deterministic Δ-approximations.

We illustrate the privacy gains that $\mathscr{G}_\Delta^{DYN}$ provides as compared to other deterministic approximations of f.

Example 3. *Reconsider Example 2 with the same values for a, $\Delta = 1$, and f, but inputs Y and Z now follow a linear distribution over $[\![1;30]\!]$. So f is 2Δ-sparse for all a in $[\![0;9]\!]$. As $\mathscr{G}_\Delta^{uni}$ and $\mathscr{G}_\Delta^{opt}$ are not deterministic, we compare the efficiency of our approximation $\mathscr{G}_\Delta^{DYN}$ to the deterministic randomisations $\mathscr{G}_\Delta^{trunc}$ introduced in Definition 8 and $\mathscr{G}_\Delta^{DET}$. Figure 3 shows the entropies $\mathrm{H}(Y \mid f(X))$ and $\mathrm{H}(Y \mid \mathscr{G}(X))$ for the different randomised functions $\mathscr{G}$ previously mentioned.*

We note that $\mathscr{G}_\Delta^{DYN}$ offers a significant privacy gain compared to $\mathscr{G}_\Delta^{trunc}$, and that it also provides a slight privacy gain compared to $\mathscr{G}_\Delta^{DET}$. By Theorem 6, $\mathscr{G}_\Delta^{DYN}$ produces the highest possible entropy of all *deterministic Δ-approximations of f.*

9 Evaluation

Since we also want to build randomisations that are effective in more general and practical situations, beyond the theoretical scope of our results, we now compare the effectiveness of our approximations to other deterministic and randomised approximations under non-uniform prior beliefs. We test our approach against sparse and non-sparse functions and report experimental results on the privacy gains for the more general g-entropy.

We conduct a series of four experiments. A summary of the parameters chosen in those experiments is seen in Table 1. Each experiment selects a polynomial f in 2 or 3 variables. We denote targeted variables by y, y_1 and y_2 and spectators variables by z, z_1 and z_2. Experiments E1 and E2 have functions of 1 or 2 targeted inputs and 1 spectator input, experiments E3 and E4 have functions of 1 targeted input with 2 spectators inputs. The selected polynomials are of the following forms:

E1 $f(y,z) = ay^2 + bz^2 + cyz + dy + ez$

E2 $f(y_1,y_2,z) = ay_1^2 + by_2^2 + cz^2 + dy_1y_2 + ey_1z + fy_2z + gy_1 + hy_2 + iz$

E3 $f(y,z_1,z_2) = ayz_1 + byz_2 + cz_1z_2 + dy + ez_1 + fz_2$

E4 $f(y,z_1,z_2) = ay^2 + bz_1^2 + cz_2^2 + dyz_1 + eyz_2 + fz_1z_2 + gy + hz_1 + iz_2$

where a, b, c, d, e, f, g, h and i represent generic coefficients. Each experiment performs 1000 iterations. Each such iteration randomly samples a set of coefficients from a range specified in Table 1, determining the polynomial on which we performed our privacy analyses. Constant terms in polynomials are ignored since they do not affect the targeted entropy. The generic forms of the functions introduced earlier are summarised by their arguments and the degree of the polynomial in Table 1, which also specifies the input ranges used, the distortion bound, the entropy for which the privacy of the targeted inputs is evaluated, and whether the polynomial is sparse – and if so, for which value of Δ.

Experiment E4 measures privacy via the general notion of g-entropy introduced in [16] and uses gain function g_2 that measures the information gained on the parity of a targeted input. For input Y ranged in D_Y, function $g_2\colon \{\texttt{even}, \texttt{odd}\} \times D_Y \longrightarrow \{0,1\}$ is defined as $g_2(w,y) = 1$ if y has parity w, and 0 otherwise. Inputs are assigned a *non-uniform* prior probability: for each iteration and input, we drew a probability distribution from a symmetric Dirichlet distribution of concentration parameter 1 on its range. Each analysis compares the privacy gains of our new methods $\mathscr{G}_\Delta^{DET}$ and $\mathscr{G}_\Delta^{DYN}$ to other approximations: $\mathscr{G}_\Delta^{uni}$ that adds uniformly distributed noise, $\mathscr{G}_\Delta^{trunc}$ that truncates the output, and the optimal independent randomisation $\mathscr{G}_\Delta^{opt}$ in [17].

We also evaluate how randomising mechanisms widely used in DP would perform under our utility and privacy metrics. The intent is not to make an exhaustive comparison between methods that are intrinsically aimed at solving different problems, but to demonstrate that there might not exist a method of privacy preservation that is always best in all use cases, as suggested in Example 1. To that end, we introduce the discrete

Table 1. Experiment parameters.

Experiment	E1	E2	E3	E4
Arguments	(y, z)	(y_1, y_2, z)	(y, z_1, z_2)	(y, z_1, z_2)
deg(f)	2	2	1+mixed	2
Coefficients	$[\![1; 50]\!]$	$[\![1; 20]\!]$	$[\![1; 20]\!]$	$[\![1; 20]\!]$
Inputs	$[\![2; 50]\!]$	$[\![1; 40]\!]$	$[\![1; 20]\!]$	$[\![1; 20]\!]$
Sparsity	7	5	no	no
Bound Δ	3	1	1	1
Entropy	min	min	min	g_2

Table 2. Comparison between average resulting entropy $\mathrm{H}(Y \mid \mathscr{G}(O))$ of different randomising methods $\mathscr{G}$.

Experiment	E1	E2	E3	E4
f	0.0095	0.2155	0.4001	0.0547
$\mathscr{G}_\Delta^{uni}$	0.0320	0.4356	0.7815	0.1265
$\mathscr{G}_\Delta^{trunc}$	0.0321	0.4361	0.7819	0.1261
$\mathscr{G}_\Delta^{opt}$	0.0322	0.4374	0.7889	0.1276
$\mathscr{G}_\Delta^{lap}$	0.0207	0.3879	0.6897	0.1136
$\mathscr{G}_\Delta^{DET}$	0.0546	0.4799	0.8069	0.1473
$\mathscr{G}_\Delta^{DYN}$	0.0561	0.5452	0.8823	0.1638

equivalent of the truncated Laplace mechanism $\mathscr{G}_\Delta^{lap}$. It generates an additive noise that follows the discrete analogue of the Laplace distribution [29] which is then truncated to respect the maximal distortion bound Δ. Parameter p will be set to 0.3 in our experiments.

Definition 10. *The Laplace Δ-approximation $\mathscr{G}_\Delta^{lap}$ of f with parameter p is defined by its output randomisation function $\mathscr{H}_\Delta^{lap}$ for all o in $\mathbb{D}_O$ and o' in $\mathbb{Z}$ as:*

$$p(\mathscr{H}_\Delta^{lap}(o) = o') = \begin{cases} \frac{1}{s} \cdot \frac{1-p}{1+p} & \text{if } \delta = 0 \\ \frac{1}{2s} \cdot \frac{1-p}{1+p} \cdot p^\delta & \text{if } 0 < \delta \leq \Delta \\ 0 & \text{otherwise} \end{cases}$$

where $s = \sum_{k=0}^{\Delta} \frac{1-p}{1+p} \cdot p^k$ and $\delta = |o - o'|$.

As for our experimental results, we compute the average entropy produced by each of the randomisations over the 1000 iterations for each of the four experiments, and display the results in Table 2. Figure 4 plots a set of 20 data points randomly sampled from the 1000 iterations of experiment 1, sorted by their value of $\mathrm{H}(Y \mid O)$, illustrating the performance of those approximations. Similar figures are drawn for experiments E2, E3 and E4 in Appendix B in the respective Figures 8, 9 and 10.

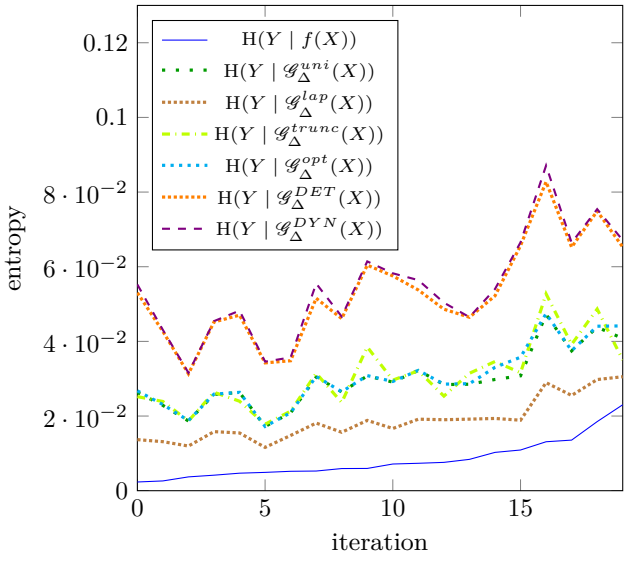

Fig. 4. Experiment E1: Privacy gains offered by $\mathscr{G}_\Delta^{DET}$ and $\mathscr{G}_\Delta^{DYN}$ in comparison to those of other Δ-approximations.

Experiments E1 and E2 empirically demonstrate that $\mathscr{G}_\Delta^{DYN}$ and $\mathscr{G}_\Delta^{DET}$ produce a significantly higher min-entropy than all the other deterministic and randomised Δ-randomisations that we have collected in the literature, empirically extending the scope of Theorem 6.

Experiment E3 and E4 show that $\mathscr{G}_\Delta^{DYN}$ and $\mathscr{G}_\Delta^{DET}$ do offer some privacy gains compared to all the other Δ-randomisations even in the case where the function to randomise is not sparse. This corroborates that the randomisations we developed in this paper also seem to offer effective privacy gains beyond the scope of our theoretical results for sparse functions. Experiment E4 also shows that the privacy gains of our randomisations are effective not only for the min-entropy but also for the g-entropy as a utility measure of privacy.

We note that the naive randomisation $\mathscr{G}_\Delta^{uni}$ already provides decent privacy gains compared to the original entropy $\mathrm{H}(Y \mid O)$. The optimal randomisation for independent noise $\mathscr{G}_\Delta^{opt}$ naturally always produces higher entropy than $\mathscr{G}_\Delta^{uni}$ and $\mathscr{G}_\Delta^{lap}$. As $\mathscr{G}_\Delta^{trunc}$ adds noise that is dependent on the output, it occasionally provides more privacy than $\mathscr{G}_\Delta^{opt}$. The truncated Δ Laplace mechanism $\mathscr{G}_\Delta^{lap}$ is the method which yields the smallest privacy gain. This is natural as this mechanism provides a stronger utility guarantee which combines a statistically small distortion with a maximal distortion bound.

In this section, we studied a variety of different mechanisms $\mathscr{G}$ in order to evaluate the algorithms $\mathscr{G}_\Delta^{DET}$ and $\mathscr{G}_\Delta^{DYN}$ that we introduced in this work. Privacy was measured in terms of min-entropy and g-entropy,

Table 3. Comparison between average expected gain $\Gamma_{exp}(\mathscr{G})$ of different randomising methods $\mathscr{G}$.

Experiment	E1	E2	E3	E4
$\mathscr{G}_\Delta^{uni}$	0.143	0.333	0.333	0.333
$\mathscr{G}_\Delta^{trunc}$	0.140	0.326	0.324	0.315
$\mathscr{G}_\Delta^{opt}$	0.118	0.310	0.301	0.333
$\mathscr{G}_\Delta^{lap}$	0.545	0.625	0.625	0.625
$\mathscr{G}_\Delta^{DET}$	0.015	0.238	0.270	0.194
$\mathscr{G}_\Delta^{DYN}$	0.815	0.319	0.290	0.399

while utility was measured in terms of maximal loss Λ_{max} introduced in Definition 6. More precisely, all of the mechanisms evaluated here were tailored to provide the utility guarantee $\Lambda_{max}(\mathscr{G}) \leq \Delta$ for a certain bound Δ. We are also interested in empirically evaluating the utility of our randomising algorithms with respect to other popular utility notions, such as the expected gain Γ_{exp} defined in Definition 6, where we would expect our algorithms to perform worse than e.g. $\mathscr{G}_\Delta^{lap}$ which is designed to provide strong guarantees in terms of Γ_{exp}. We thus ran another set of 1000 iterations of our four experiments and we report in Table 3 the expected gain $\Gamma_{exp}(\mathscr{G})$ averaged over all iterations for the same set of mechanisms $\mathscr{G}$. We plot the values of $\Gamma_{exp}(\mathscr{G})$ for the first 20 iterations of experiment E1 and E2 in Figures 5 and 6 respectively. The results of E3 and E4 are similar to that of E2 and are displayed in Appendix C.

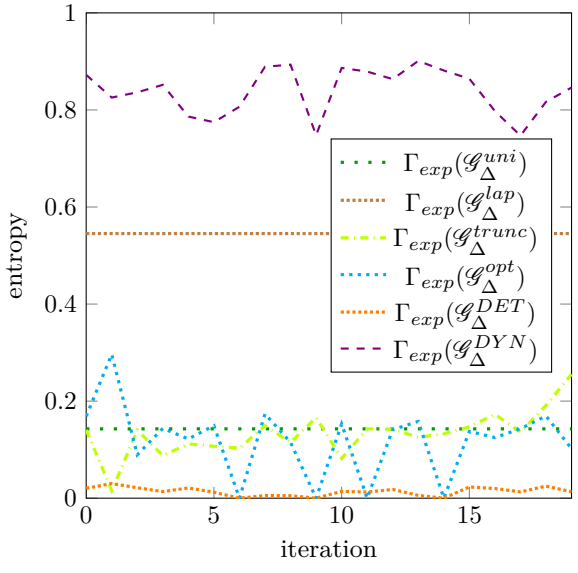

Fig. 5. Experiment E1: Expected gain provided by $\mathscr{G}_\Delta^{DET}$ and $\mathscr{G}_\Delta^{DYN}$ in comparison to those of other Δ-approximations.

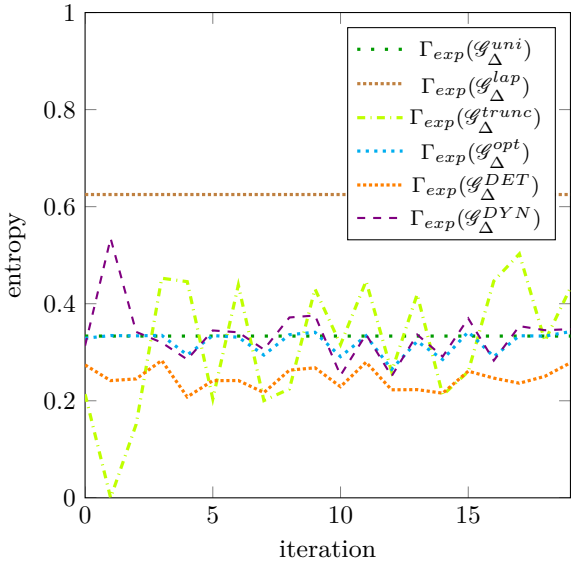

Fig. 6. Experiment E2: Expected gain provided by $\mathscr{G}_\Delta^{DET}$ and $\mathscr{G}_\Delta^{DYN}$ in comparison to those of other Δ-approximations.

As expected, we can see that $\mathscr{G}_\Delta^{lap}$ performs better than our algorithms in terms of expected gain in experiments E2, E3 and E4. However, we can notice that $\mathscr{G}_\Delta^{DYN}$ produces a significantly higher Γ_{exp} than $\mathscr{G}_\Delta^{lap}$ in experiment E1. This is an interesting result, since the design of $\mathscr{G}_\Delta^{lap}$ itself guarantees a certain utility value of Γ_{exp}. On the other hand, $\mathscr{G}_\Delta^{DYN}$ is only designed to maximise the resulting entropy and still provides a higher expected gain than $\mathscr{G}_\Delta^{lap}$. Naturally, raising the value of the parameter p of the Δ-Laplace approximation would increase its expected gain, but would also further decrease the resulting privacy gain. It would be of interest to study the influence of the value of Δ, the type of function, and the type of inputs on the effectiveness of our algorithms $\mathscr{G}_\Delta^{DYN}$ in terms of expected gain, a topic that we mean to study in the future.

Non-polynomial functions. Although our functions for the experiments E1 to E4 encompass a large variety of low degree polynomial functions, real world applications of SMC might involve polynomials of very high degree, or more generally interactive functions, such as the max function in auctions. Therefore we now perform two experiments on non-polynomial functions. Experiments E5 and E6 are described as follows. We place ourselves in the setting of Experiment E1 and we sample $n = 5$ functions $f_0, \cdots, f_4$ of the form of E1. We then build the following functions:

E5 $f(y,z) = \max\{f_j(y,z) \mid 0 \leq j < n\}$

E6 $f(y,z) = f_j(y,z)$ where $j = ((y+z) \bmod n)$

where E5 computes the maximal output of the n functions f_j while E6 interactively selects the output of one of the f_j depending on the values of y and z. We ran 20 iterations of experiment E5 and we experimentally demonstrate in Figure 7 that our algorithms also provide significant privacy gains on those non-polynomial functions compared to other existing randomising mechanisms. A similar graph is drawn for experiment E6 in Figure 13 in Appendix D.

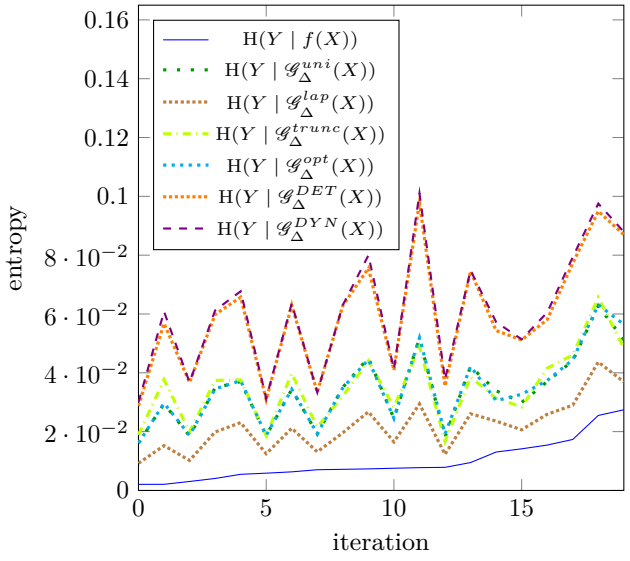

Fig. 7. Experiment E5: Privacy gains offered by $\mathscr{G}_\Delta^{DET}$ and $\mathscr{G}_\Delta^{DYN}$ in comparison to those of other Δ-approximations.

10 Discussion and Future Works

Deceitful adversaries. Recall the observer Ann, *external* to the computation of f, who wants to learn a private input y given the knowledge of the output $o = f(y, z)$. Let us now assume that Ann is a party in this computation who holds a private input a and enters the computation of $o = f(a, y, z)$. Ann can then actively choose her value of a to maximise what she can learn about the value of y.

In the setting of Example 2, let Ann be a party. Ann knows her own input value $x = a$ and can control this value. The randomised approximations $\mathscr{G}$ of f are then implicitly parametrised by a. In such settings, a randomised approximation $\mathscr{G}$ of f is no longer represented by its output randomisation function $\mathscr{H}$ only, as the information that Ann learns on y from output $\mathscr{G}(a, y, z)$ would not only depend on the value of $f(y, z)$, but also on the value of a. Instead, we ought to represent $\mathscr{G}$ as $\mathscr{G}(a, y, z) = \mathscr{J}(f(a, y, z), a)$ where $\mathscr{J}$ would play the role of an output randomisation function, with an additional dependency on the attacker's input a.

Independence of inputs. In this work, we assumed that targeted and spectators' inputs Y and Z were independent. However, although targets and spectators would not share information before entering a computation, their inputs might be correlated or bound to some constraint that would make them dependent on each other. Consider, for example, two competing companies that are willing to enter their revenues into some benchmarking analysis but do not want to share information between each other. Their revenues might not be completely independent as they would both be influenced by common factors such as current supply and demand. In such cases, it would be more appropriate to replace prior probabilities $p(y)$ and $p(z)$ by joint probabilities $p(y, z)$ represented as an $|\mathbb{D}_Y| \times |\mathbb{D}_Z|$ matrix.

We could adapt our results to the assumption that Y and Z are not independent by replacing $p(y) * p(z)$ in Line 5 of Algorithm 2 with the joint probability $p(y, z)$. As the independence assumption is only used in the construction of the hash map d, this substitution would extend the validity of our theoretical results of Section 7 and 8 to non-independent inputs.

Using other privacy measures could also help us to integrate this assumption. For example, the min-capacity [14, 23], defined as the maximal multiplicative Bayes leakage over all priors on the secret, would be expressed as $\sum_o \max_y p(o \mid y)$ and would be independent of the prior distribution for Y. Thus, Algorithm 1 would apply for any joint distribution for Y and Z given that the marginal distribution of Z is uniform.

Limitations and applicability. Secure multi-party computation is an active field of research, but real world and industrial deployments of SMC are still limited [30, 31]. However, we believe that our approach could be useful in future applications of SMC, even when restricted to low degree polynomial functions.

Federated learning [32] enables parties to compute machine learning algorithms on their private data in a decentralised manner. A server iteratively computes some *linear combination* of some data that has been locally aggregated by the parties and returns the output to the parties, as described in Algorithm 1 in [32]. Some works have studied the possibility to replace the server by an SMC protocol in order to compute the secure aggregation [33]. One could thus ask how much information about their private inputs is leaked by the output of those linear combinations. Our experiments E1 to E4

suggest that our approach could improve the privacy of inputs for such low degree polynomials.

Experiments E5 and E6 show the effectiveness of our approach on interactive functions that are non-polynomial, suggesting that our algorithms could be applied to more general functions in the future.

For example, some works have focused on privacy preserving logistic regressions [34]. It would be of interest to apply our randomising mechanisms in this scenario. Each party would hold an input (x_i, y_i), all together forming vectors x and y. The output of the simplest linear regression with ordinary least squares would be $\beta = \frac{cov(x,y)}{var(x)}$ and $\alpha = \bar{y} - \beta\bar{x}$. The natural problem that arises here is that of handling floating point values in our model.

Another example where randomised mechanisms have been studied is that of auctions [7]. In the setting of digital goods auction, each party P_j bids the highest price p_j at which she is ready to pay a given item. The output of the auction is the price that maximises the sellers' benefits, i.e. v_k where $k = argmax_j(p_j * j)$ and we assume that the bids $p_1, \cdots, p_n$ are ordered in descending order. Randomising mechanisms from DP exist that protect the bidders' inputs privacy [7] and it would be interesting to compare the results with our approach.

Due to the combinatorial essence of our approach, our experiments currently rely on a small number of inputs, typically lower than 5. It would be interesting to explore some methods to scale our algorithms in the future. However, we believe that focusing on a small number of inputs is of particular interest in many practical situations. Indeed, some of the few secure computations that have been deployed in real life to date involved a large number of parties but relied on a very limited number of servers, on which the secure computation was executed [30]. Our approach could thus be effective in computations involving a larger number of parties by being applied to the servers' inputs and output.

Moreover, Differential Privacy is not suited for computations with a limited number of inputs. For example, a differentially private mechanism used on a computation involving two inputs would ensure that the output distribution is unaffected if half of the inputs is changed, which would naturally yield an extremely poor utility. In such cases where DP is not applicable, our approach could be a viable privacy-enhancing alternative.

Also due to efficiency reasons, our current experiments involved small input spaces. Research in this direction has produced closed-form formulas for computing the privacy of inputs in affine computations, which scale to arbitrarily large input spaces [21], and we would like to take advantage of such findings to make our analyses scalable in the future.

11 Conclusion

Controlling the confidentiality of secret inputs when performing some computations on such private data is a well studied paradigm. Differential Privacy provides techniques that enhance the participants' privacy by randomising the output of a computation, while guaranteeing that distorted outputs will be statistically close to the intended output. However, certain use cases may not tolerate distortions that exceed a given threshold at all, even with low probability. Therefore recent works have investigated the privacy gains that can be obtained by randomising a function with an additive noise, independent of the output of the function, and bounded by a maximal distortion threshold.

In this paper, we developed randomising mechanisms that can distort the output of a function dependent on its value, while guaranteeing that the distortion these mechanisms introduce does not exceed a given threshold. We have formalised this concept by introducing the notion of randomised approximation. We formally investigated the privacy gains that randomised approximations can offer, under a given distortion bound. Then we presented algorithms for building specific deterministic approximations and proved that these approximations maximise the targeted inputs' entropy for so called sparse functions. Our experiments, for both min-entropy and g-entropy, showed that these optimal mechanisms offer additional privacy gains for non-sparse functions and non-uniform input distributions as well, when compared to mechanisms that add noise independently of the output.

Acknowledgements. This research project was supported by the UK EPSRC grants EP/N020030/1 and EP/N023242/1.

Source code. The source code of our evaluative experiments described in Section 9 is publicly available at https://github.com/pahfat/PoPETs2020.

References

[1] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 162–167. IEEE, 1986.

[2] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. of the twentieth annual ACM symposium on Theory of computing*, pages 1–10. ACM, 1988.

[3] Geoffrey Smith. On the foundations of quantitative information flow. In *International Conference on Foundations of Software Science and Computational Structures*, pages 288–302. Springer, 2009.

[4] Christian Cachin. *Entropy measures and unconditional security in cryptography*. PhD thesis, Swiss Federal Institute of Technology Zurich, 1997.

[5] Cynthia Dwork. Differential privacy: A survey of results. In *International Conf. on Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.

[6] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[7] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, pages 94–103. IEEE, 2007.

[8] Andrew C Yao. Protocols for secure computations. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 160–164. IEEE, 1982.

[9] Adi Shamir. How to share a secret. *CACM*, 22(11):612–613, 1979.

[10] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 73–85. ACM, 1989.

[11] David Chaum, Claude Crépeau, and Ivan Damgard. Multiparty unconditionally secure protocols. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 11–19. ACM, 1988.

[12] Pasquale Malacaria. Algebraic foundations for quantitative information flow. *Mathematical Structures in Computer Science*, 25(02):404–428, 2015.

[13] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

[14] Geoffrey Smith. Quantifying information flow using min-entropy. In *Quantitative evaluation of systems (QEST), 2011 eighth international conference on*, pages 159–167. IEEE, 2011.

[15] Alfréd Rényi et al. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. The Regents of the University of California, 1961.

[16] Mário S. Alvim, Kostas Chatzikokolakis, Catuscia Palamidessi, and Geoffrey Smith. Measuring information leakage using generalized gain functions. In *Computer Security Foundations Symposium (CSF), 2012 IEEE 25th*, pages 265–279. IEEE, 2012.

[17] Patrick Ah-Fat and Michael Huth. Optimal accuracy-privacy trade-off for secure computations. *IEEE Transactions on Information Theory*, 2018.

[18] MHR Khouzani and Pasquale Malacaria. Relative perfect secrecy: Universally optimal strategies and channel design. In *Computer Security Foundations Symposium (CSF), 2016 IEEE 29th*, pages 61–76. IEEE, 2016.

[19] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[20] Patrick Ah-Fat and Michael Huth. Secure multi-party computation: Information flow of outputs and game theory. In *International Conference on Principles of Security and Trust*, pages 71–92. Springer, 2017.

[21] Patrick Ah-Fat and Michael Huth. Scalable information flow analysis of secure three-party affine computations. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 2967–2971. IEEE, 2019.

[22] Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 2017.

[23] Barbara Espinoza and Geoffrey Smith. Min-entropy as a resource. *Information and Computation*, 226:57–75, 2013.

[24] Lalitha Sankar, S Raj Rajagopalan, and H Vincent Poor. Utility-privacy tradeoffs in databases: An information-theoretic approach. *IEEE Transactions on Information Forensics and Security*, 8(6):838–852, 2013.

[25] Mário S Alvim, Miguel E Andrés, Konstantinos Chatzikokolakis, Pierpaolo Degano, and Catuscia Palamidessi. Differential privacy: on the trade-off between utility and information leakage. In *International Workshop on Formal Aspects in Security and Trust*, pages 39–54. Springer, 2011.

[26] Reza Shokri. Privacy games: Optimal user-centric data obfuscation. *Proceedings on Privacy Enhancing Technologies*, 2015(2):299–315, 2015.

[27] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. *SIAM Journal on Computing*, 41(6):1673–1693, 2012.

[28] Ehab Elsalamouny, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. A differentially private mechanism of optimal utility for a region of priors. In *International Conference on Principles of Security and Trust*, pages 41–62. Springer, 2013.

[29] Seidu Inusah and Tomasz J Kozubowski. A discrete analogue of the laplace distribution. *Journal of statistical planning and inference*, 136(3):1090–1102, 2006.

[30] Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, et al. Secure multiparty computation goes live. In *International Conference on Financial Cryptography and Data Security*, pages 325–343. Springer, 2009.

[31] Ivan Damgård, Kasper Damgård, Kurt Nielsen, Peter Sebastian Nordholt, and Tomas Toft. Confidential benchmarking based on multiparty computation. In *International Conference on Financial Cryptography and Data Security*, pages 169–187. Springer, 2016.

[32] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of

deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

[33] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for federated learning on user-held data. *arXiv preprint arXiv:1611.04482*, 2016.

[34] Kamalika Chaudhuri and Claire Monteleoni. Privacy-preserving logistic regression. In *Advances in neural information processing systems*, pages 289–296, 2009.

Appendices

A Proof of Theorem 5

For a function h in $(\mathbb{Z}^{\mathbb{D}_O})_\Delta$ we build an increasing function k in $[\mathbb{Z}^{\mathbb{D}_O}]_\Delta$ with $\mathrm{V}(Y \mid k(O)) \leq \mathrm{V}(Y \mid h(O))$. We first present a base case, how to disentangle two unsorted outputs, and then build on that an algorithm for constructing the increasing function k.

Base case: Let o_1 and o_2 be in $\mathbb{D}_O$ and assume that $o_1 < o_2$ and that $h(o_1) > h(o_2)$. If such outputs exist, then h is not increasing and we build a function k with $k(o_1) \leq k(o_2)$ and $\mathrm{V}(Y \mid k(O)) \leq \mathrm{V}(Y \mid h(O))$. Function k is a copy of function h, except that we impose $k(o_1) = k(o_2)$, which equals $h(o_1)$ or $h(o_2)$.

First, let us show that $|o_1 - h(o_2)| \leq \Delta$. As $h \in (\mathbb{Z}^{\mathbb{D}_O})_\Delta$, we have $o_2 - h(o_2) \leq \Delta$. By assumption, $o_1 - o_2 < 0$. By adding those inequalities we get $o_1 - h(o_2) \leq \Delta$. Conversely, we have $h(o_1) - o_1 \leq \Delta$ and $h(o_2) - h(o_1) < 0$. By summation, we get $h(o_2) - o_1 \leq \Delta$. We thus have $|o_1 - h(o_2)| \leq \Delta$, and a symmetric argument yields $|o_2 - h(o_1)| \leq \Delta$. Thus, with k a copy of h, we can set $k(o_1) \leftarrow h(o_2)$ or $k(o_2) \leftarrow h(o_1)$ while still guaranteeing that k belongs to $(\mathbb{Z}^{\mathbb{D}_O})_\Delta$.

Let us now see which assignment ensures that $\mathrm{V}(Y \mid k(O)) \leq \mathrm{V}(Y \mid h(O))$. For all o in $\mathbb{D}_O$, we set $d[o] = \max_y p(y)\, p(o \mid y)$. Let us for now assume that:

$$\max_{o \in h^{-1}(h(o_1))} d[o] \geq \max_{o \in h^{-1}(h(o_2))} d[o] \tag{5}$$

Then we decide to set $k(o_2) \leftarrow h(o_1)$. Let us show that this does not increase vulnerability. We have $k^{-1}(h(o_2)) = h^{-1}(h(o_2)) \setminus \{o_2\}$, so:

$$\max_{o \in k^{-1}(h(o_2))} d[o] \geq \max_{o \in h^{-1}(h(o_2))} d[o]$$

Conversely, we have $k^{-1}(h(o_1)) = h^{-1}(h(o_1)) \cup \{o_2\}$, but we know that $d[o_2] \leq \max\{d[o] \mid o \in h^{-1}(h(o_2))\} \leq \max\{d[o] \mid o \in h^{-1}(h(o_1))\}$ by assumption. Thus $\max k^{-1}(h(o_1))\} \geq \max\{d[o] \mid o \in h^{-1}(h(o_1))\}$. We note that for all o' in $\mathbb{D}_{O'}$ which are different from $h(o_1)$ and $h(o_2)$, as k is a copy of h, we still have $k^{-1}(o') = h^{-1}(o')$ and thus $\sum_{o'} \max\{d[o] \mid o \in k^{-1}(o')\} \leq \sum_{o'} \max\{d[o] \mid o \in h^{-1}(o')\}$. Whenever Equation (5) is not satisfied, we set $k(o_1) \leftarrow h(o_2)$ and a similar argument ensures that the vulnerability does not increase.

Notation: This process of building a copy k of a function h that merges the image of two unsorted outputs o_1 and o_2 will be denoted as $k \leftarrow \mathrm{merge}(h, (o_1, o_2))$.

General case: Algorithm 3 outputs an increasing function k that does not increase vulnerability.

Algorithm 3 Building an increasing function given h in $(\mathbb{Z}^{\mathbb{D}_O})_\Delta$

Inputs: $\mathbb{D}_O \subseteq \mathbb{Z}, h \in (\mathbb{Z}^{\mathbb{D}_O})_\Delta$
Output: $k \in [\mathbb{Z}^{\mathbb{D}_O}]_\Delta$
Post-Condition: $\mathrm{V}(Y \mid k(O)) \leq \mathrm{V}(Y \mid h(O))$

```
1:  k ← h
2:  𝔼 ← ∅
3:  for o in 𝔻_O in increasing order do
4:      𝔼 ← 𝔼 ∪ {o}
5:      /* Invariant 1: k is increasing on 𝔼 \ {o} */
6:      if k is not increasing on 𝔼 then
7:          𝔽 ← {q ∈ 𝔼 | k(o) < k(q)}
8:          for q in 𝔽 in increasing order do
9:              k ← merge(k, (q, o))
10:     /* Invariant 2: k is increasing on 𝔼 */
11: return k
```

We show that function k, returned by this algorithm, is in $[\mathbb{Z}^{\mathbb{D}_O}]_\Delta$. We enumerate elements of $\mathbb{D}_O$ as $\mathbb{D}_O = (o_1, o_2, \cdots, o_{|\mathbb{D}_O|})$. We prove that the algorithm satisfies the invariants stated in green: when entering the for-loop in Line 5, k is increasing on $\mathbb{E} \setminus \{o\}$ and when exiting in Line 10, k is increasing on $\mathbb{E}$. We can see that during the first iteration, $\mathbb{E} \setminus \{o\}$ is the empty set and thus Invariant 1 holds. It is obvious to see that Invariant 2 at a given iteration implies Invariant 1 at the next iteration. Let us now assume that k is increasing on $\mathbb{E} \setminus \{o\}$ and show that Invariant 2 holds at the end of the iteration. Consider an iteration of the loop in Line 3 such that the condition on Line 6 succeeds. We study the variables of the program in their current state. Let i be such that o_{i+1} denotes the first element of $\mathbb{F}$. For j with $0 \leq j \leq |\mathbb{F}|$, define property P_j as:

$$\begin{aligned} P_j \quad = \quad & (k \text{ is increasing on } \mathbb{E} \setminus \{o\}) \\ & \wedge (k(o_{i+j}) \leq k(o) \leq k(o_{i+j+1})) \end{aligned}$$

We prove by induction on the size of $\mathbb{F}$ that each merge operation in Line 9 complies with this property. We note that $P_{i+|\mathbb{F}|}$ means that k is increasing on $\mathbb{E}$.

For the base case, we know by Invariant 1 that k is increasing on $\mathbb{E} \setminus \{o\}$, and by definition of $\mathbb{F}$ that $k(o) \geq k(o_i)$ and $k(o) \leq k(o_{i+1})$. For the inductive step and j with $1 \leq j \leq |\mathbb{F}|$, let us assume that P_{j-1} holds. We know that the merge operation in Line 9 will either set $k(o) \leftarrow k(o_{i+j})$ or $k(o_{i+j}) \leftarrow k(o)$. However, by assumption, we know that $(k(o_{i+j-1}) \leq k(o) \leq k(o_{i+j}))$ and that k is increasing on $\mathbb{E} \setminus \{o\}$. And thus the merge operation will maintain those properties.

B Empirical Privacy Evaluation

This section contains Figures 8, 9 and 10 that are graphical illustrations of the empirical evaluation of our algorithms in terms of privacy for experiments E2, E3 and E4 respectively.

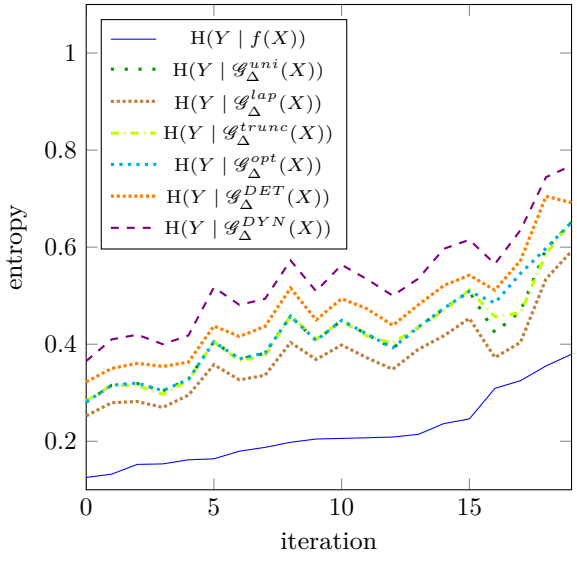

Fig. 8. Experiment E2: Privacy gains offered by $\mathscr{G}_\Delta^{DET}$ and $\mathscr{G}_\Delta^{DYN}$ in comparison to those of other Δ-approximations.

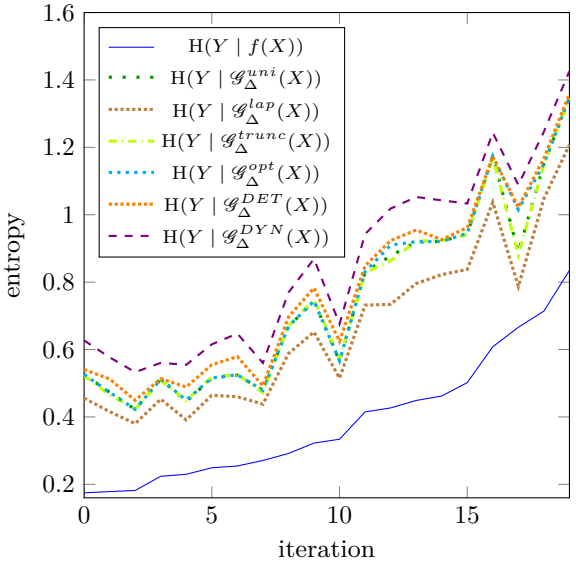

Fig. 9. Experiment E3: Privacy gains offered by $\mathscr{G}_\Delta^{DET}$ and $\mathscr{G}_\Delta^{DYN}$ in comparison to those of other Δ-approximations.

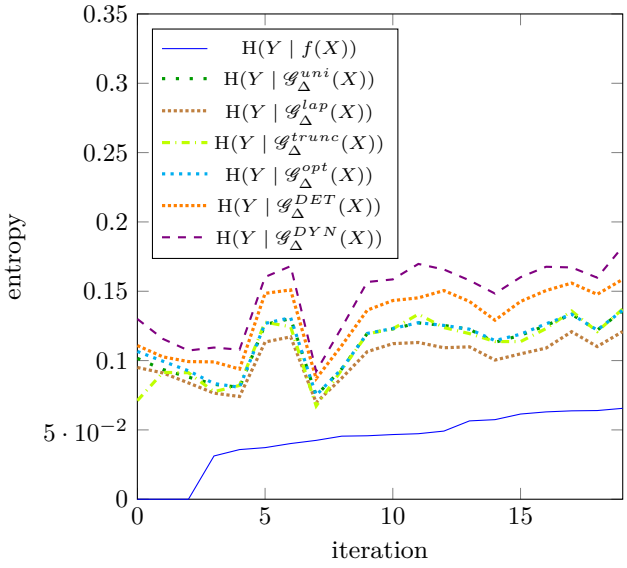

Fig. 10. Experiment E4: Privacy gains offered by $\mathscr{G}_\Delta^{DET}$ and $\mathscr{G}_\Delta^{DYN}$ in comparison to those of other Δ-approximations.

C Empirical Utility Evaluation

This section contains Figures 11 and 12 that are graphical illustrations of the empirical evaluation of our algorithms in terms of privacy for experiments E3 and E4 respectively.

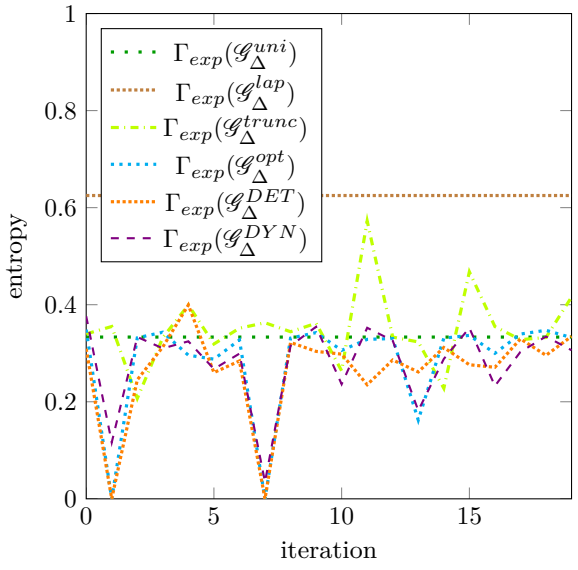

Fig. 11. Experiment E3: Expected gain provided by $\mathscr{G}_\Delta^{DET}$ and $\mathscr{G}_\Delta^{DYN}$ in comparison to those of other Δ-approximations.

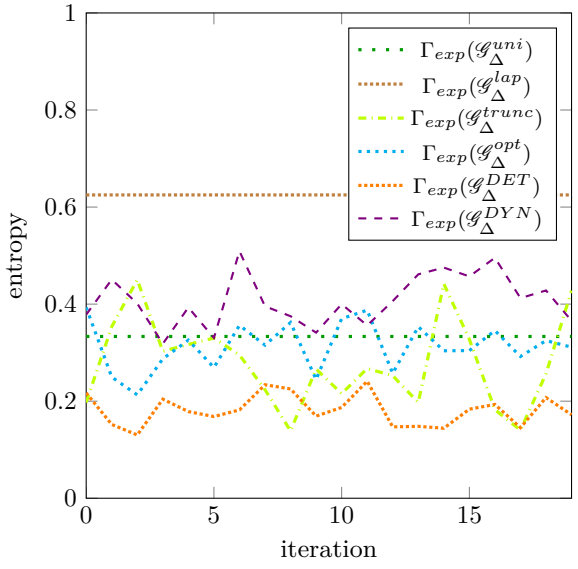

Fig. 12. Experiment E4: Expected gain provided by $\mathscr{G}_\Delta^{DET}$ and $\mathscr{G}_\Delta^{DYN}$ in comparison to those of other Δ-approximations.

D Privacy Evaluation on Non-Polynomial Functions

This section contains Figure 13 that is a graphical illustration of the empirical evaluation of our algorithms in terms of privacy on non-polynomial functions for experiment E6.

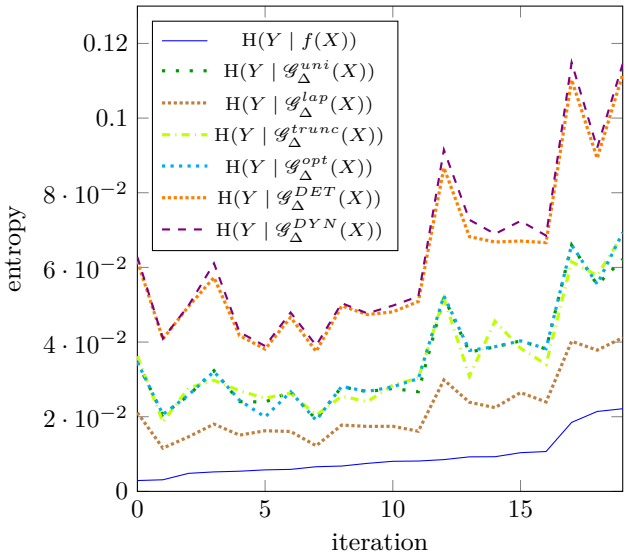

Fig. 13. Experiment E6: Privacy gains offered by $\mathscr{G}_\Delta^{DET}$ and $\mathscr{G}_\Delta^{DYN}$ in comparison to those of other Δ-approximations.