2013 IEEE 26th Computer Security Foundations Symposium

Verified Computational Differential Privacy with Applications to Smart Metering

Gilles Barthe*, George Danezis‡, Benjamin Grégoire†, César Kunz*, Santiago Zanella-Béguelin‡
*IMDEA Software Institute, Spain. Email: {gilles.barthe,cesar.kunz}@imdea.org
†INRIA Sophia Antipolis – Méditerranée, France. Email: benjamin.gregoire@inria.fr
‡Microsoft Research, UK. Email: {gdane,santiago}@microsoft.com

***Abstract*—EasyCrypt is a tool-assisted framework for reasoning about probabilistic computations in the presence of adversarial code, whose main application has been the verification of security properties of cryptographic constructions in the computational model. We report on a significantly enhanced version of EasyCrypt that accommodates a richer, user-extensible language of probabilistic expressions and, more fundamentally, supports reasoning about approximate forms of program equivalence. This enhanced framework allows us to express a broader range of security properties, that notably include approximate and computational differential privacy. We illustrate the use of the framework by verifying two protocols: a two-party protocol for computing the Hamming distance between bit-vectors, yielding two-sided privacy guarantees; and a novel, efficient, and privacy-friendly distributed protocol to aggregate smart meter readings into statistics and bills.**

I. Introduction

Data mining holds the promise of delivering valuable information that can be used for purposes such as elaborating health policies or market strategies. To realize its potential, data mining must reconcile two objectives: providing useful results and protecting the privacy of individuals. Differential privacy [17] is one policy that achieves a good compromise between privacy and utility. In the database literature, a randomized mechanism is differentially private if it ensures that the privacy of individuals that contribute data to a statistical database is protected against malicious or dangerous queries. More formally, a randomized mechanism M is differentially private if it gives *similar* answers to queries on two databases D, D' that differ in at most one row. Quantitatively, (ϵ, δ)-differential privacy requires that for any set of answers S,

$$\mathsf{Pr}[M(D) \in S] \leq \exp(\epsilon)\ \mathsf{Pr}[M(D') \in S]\ +\ \delta$$

Differential privacy has two outstanding strengths: it is realizable by sanitization mechanisms that provide useful answers to honest queries, and it enjoys good composition properties. Over the last years, differential privacy has emerged as a de facto standard for privacy-preserving computation, which has led to the development of differentially private approximation algorithms for problems of practical interest, and sanitization mechanisms to answer a broad range of statistical queries.

Differential privacy provides information-theoretic guarantees that hold against computationally unbounded adversaries. As a consequence, it is usually too strong for distributed applications, which often rely on cryptographic constructions whose security holds only against computationally bounded adversaries. Computational differential privacy [34] is a relaxation of differential privacy that aims to provide privacy guarantees against such adversaries. This relaxation makes sense in the client/server setting, where all data is stored at a server, and also for distributed applications, where data is spread across participants. Although negative results indicate that little, if any, can be gained from moving beyond information-theoretic differential privacy in client/server applications [24], there is evidence that it can result in significant gains in efficiency and accuracy for distributed applications, and in particular for multi-party computation protocols [31].

Computationally differentially private two-party protocols have been developed for computing Hamming distance and threshold similarity between bit-vectors, and scalar product of integer vectors [1, 34]. Computational differential privacy can also account for the use of pseudo-random sources in otherwise information-theoretic differentially private algorithms [16]. In general, information-theoretic differentially private algorithms can be run in a distributed manner using secure multi-party computation to emulate a trusted third party with access to the data of all participants [8], but the resulting protocol will be in most cases only computationally differentially private.

Deductive verification of differential privacy

The advent of powerful Satisfiability Modulo Theory (SMT) solvers and automated theorem provers has turned deductive program verification into a practical technology for proving safety and correctness of complex software. However, applying state-of-the-art deductive program verification techniques to prove some security and privacy properties remains a significant challenge for two main reasons. First, many policies are most naturally expressed as 2-safety properties, i.e. properties of two executions of a program [12, 44], while most existing techniques can only reason directly about a single execution. Second, a large class of sensitive applications, such as cryptographic protocols or private data-mining algorithms, achieve their goals by means of randomized computations, but, with few exceptions, current verification tools are based on a deterministic or, in the best case, possibilistic model for programs. Differential privacy combines these two challenges, because it requires the ability to reason about quantitative properties of two executions of a probabilistic program.

DOI 10.1109/CSF.2013.26

This paper lays the foundations of a tool-assisted framework for the deductive verification of computationally differentially private algorithms. Our starting point is EasyCrypt [3], a framework for reasoning about probabilistic programs in the presence of adversarial code. To date, EasyCrypt has been mainly applied to verify the security of cryptographic primitives in the computational model. In order to allow reasoning about computational differential privacy, we have added support for proving judgments of the form

$$\vdash c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi$$

where c_1 and c_2 are probabilistic programs, assertions Ψ, Φ are relations over program states, $\alpha \geq 1$, and $0 \leq \delta \leq 1$ (see Section III-C for a formal definition of validity). The associated logic, coined apRHL, is described by Barthe et al. [5] and subsumes the relational logic implemented in previous versions of EasyCrypt, which corresponds to the case $\alpha = 1$, $\delta = 0$. An outstanding feature of apRHL is that it faithfully captures differential privacy. Concretely, if $M = c_1 = c_2$, the pre-condition Ψ is an appropriate adjacency relation on inputs, the post-condition Φ is the equality on the observable outputs of the algorithm, and $\alpha = \exp(\epsilon)$, then the above judgment is valid iff M is (ϵ, δ)-differentially private.

In addition, we have developed a library that gives first-class status to discrete probability distributions, and significantly enhances the expressive power of the programming language of EasyCrypt. This library allows users to define and specify probability distributions from which values can be sampled in programs, and to prove properties about them. These properties can then be used to verify rather than assume the privacy guarantees of standard mechanisms, such as the addition of symmetrically distributed noise for sanitizing numeric queries.

Applications

We apply the techniques described in this paper and our extension of EasyCrypt to obtain machine-checked proofs of two non-trivial examples of computationally differentially private protocols.

Our first example, inspired by the seminal paper on computational differential privacy [34], is a two-party protocol for computing Hamming distance between bit-vectors that achieves two-sided privacy guarantees.

Our second example is a novel distributed protocol for computing statistics and linear bills for smart metering infrastructures, without the use of a single trusted authority. The protocol guarantees the privacy of a single user even when all other users and authorities, except one, collude. Besides illustrating our verification techniques, the protocol offers major performance, compactness and robustness advantages over others in the literature [28, 30, 42].

Summary of contributions

The main contributions of the paper are:

1) A practical program verifier to reason about approximate relational equivalences of probabilistic programs;
2) A flexible framework to define and reason about discrete probability distributions;
3) Machine-checked privacy proofs for two practically relevant multi-party computation protocols, including a novel smart metering aggregation protocol.

Organization of the paper

The rest of the paper is structured as follows: Section II provides the necessary background on differential privacy and cryptography, and illustrates the workings of our framework on a two-party protocol for computing Hamming distance between bit-vectors. Section III describes the theoretical foundations of our extension of EasyCrypt, including its support for reasoning about approximate program equivalences and for defining new probabilistic expressions. Section IV illustrates the application of our extension of EasyCrypt to the verification of a protocol for computing aggregates of smart meter readings. We review related work in Section V and conclude with some perspectives for further work in Section VI.

Supplemental material accompanying this paper, including a version of EasyCrypt implementing the extensions we describe and the proofs of the examples we present, is available at

`http://easycrypt.gforge.inria.fr/csf13/`

II. Background and Motivating Example

In this section we review the notions of information-theoretic and computational differential privacy, both in the client-server setting, where privacy guarantees are one-sided, and in the two-party setting, where the privacy of the data of both participants must be guaranteed. Moreover, we show how various notions of differential privacy can be cast in a programming language setting as program equivalences, which can in turn be captured as judgments in a relational Hoare logic.

A. Preliminaries

We cast differential privacy in a programming language setting as a quantitative 2-safety property of probabilistic algorithms.

Definition 1 (Approximate differential privacy)**.** *A probabilistic algorithm M operating on inputs drawn from a set A with a metric d, and returning an output in B is (ϵ, δ)-differentially private if for all pair of inputs $a, a' \in A$ such that $d(a, a') \leq 1$ and for all $S \subseteq B$ it holds that*

$$\mathsf{Pr}[M(a) \in S] \leq \exp(\epsilon)\ \mathsf{Pr}[M(a') \in S]\ +\ \delta$$

The standard definition of ϵ-differential privacy corresponds to the case $\delta = 0$.

Approximate differential privacy enjoys good composition properties; it is closed with respect to sequential and parallel composition [5].

Theorem 1 (Sequential composition)**.** *Let M be an (ϵ, δ)-differentially private algorithm with outputs in B and for any $b \in B$, let $M'(b, \cdot)$ be an (ϵ', δ')-differentially private algorithm. Then, their sequential composition*

$$M''(a) \stackrel{def}{=} x \leftarrow M(a);\ y \leftarrow M'(x, a);\ \textbf{\textit{return}}\ (x, y)$$

is $(\epsilon + \epsilon', \delta + \delta')$-differentially private.

The notion of (ϵ, δ)-differential privacy is information-theoretic, in the sense that it guarantees privacy against all computationally unbounded adversaries who can observe the output of the algorithm. It is thus natural to ask whether something can be gained from restricting attention to efficient adversaries with only computationally bounded resources. Mironov et al. [34] put forward two ways of restricting Definition 1 to computationally bounded adversaries. The first, called *indistinguishability-based computational differential privacy* (IND-CDP), is a straightforward variant that restricts quantification to computationally bounded adversaries. The second, called *simulation-based differential privacy* (SIM-CDP), starkly separates the computational and information-theoretic aspects by requiring the existence of an ϵ-differentially private simulator whose output is computationally indistinguishable from the real algorithm.

Extending Definition 1 to a computational setting requires considering a family of algorithms $\{M_\eta\}$ indexed by a security parameter $\eta \in \mathbb{N}$ rather than a single algorithm. In the following, we say that an algorithm is probabilistic polynomial-time if it executes within time polynomial on the security parameter η and that a function $\nu : \mathbb{N} \to \mathbb{R}^+$ is *negligible* when

$$\forall c \in \mathbb{N}.\ \exists n_c \in \mathbb{N}.\ \forall n > n_c.\ \nu(n) < n^{-c}$$

Moreover, we assume that an algorithm M_η takes inputs in a set A with a metric d and outputs values in a set B_η.

Definition 2 (IND-CDP)**.** *A family of probabilistic algorithms $\{M_\eta\}_{\eta \in \mathbb{N}}$ is (ϵ, δ)-IND-CDP if for every probabilistic polynomial-time adversary $\mathcal{A}$ and every $a, a' \in A$ of size polynomial in η such that $d(a, a') \leq 1$ it holds that*

$$\mathsf{Pr}[\mathcal{A}(M_\eta(a)) = 1] \leq \exp(\epsilon(\eta))\ \mathsf{Pr}[\mathcal{A}(M_\eta(a')) = 1] + \delta(\eta)$$

When δ is negligible, we may simply say that $\{M_\eta\}_{\eta \in \mathbb{N}}$ is ϵ-IND-CDP.

Definition 3 (SIM-CDP)**.** *A family of probabilistic algorithms $\{M_\eta\}_{\eta \in \mathbb{N}}$ is (ϵ, δ)-SIM-CDP if there exists a family of algorithms $\{M'_\eta\}_{\eta \in \mathbb{N}}$, each of them $\epsilon(\eta)$-differentially private, such that for every probabilistic polynomial-time adversary $\mathcal{A}$ and any $a \in A$ of size polynomial in η it holds that*

$$|\mathsf{Pr}[\mathcal{A}(M_\eta(a)) = 1] - \mathsf{Pr}[\mathcal{A}(M'_\eta(a)) = 1]| \leq \delta(\eta)$$

When δ is negligible, we may simply say that $\{M_\eta\}_{\eta \in \mathbb{N}}$ is ϵ-SIM-CDP.

Note that the family of simulators $\{M'\}_{\eta \in \mathbb{N}}$ in the definition of SIM-CDP is just a theoretical artifact and need not be efficient. In general, neither do computationally differentially private algorithms need be efficient, although efficiency is a necessary and sufficient condition for an analog of Theorem 1 to hold for both IND-CDP and SIM-CDP [34].

Any algorithm that satisfies SIM-CDP also satisfies IND-CDP [34], but it is unknown whether the converse holds. However, by inverting the order of the quantifiers in the definition of SIM-CDP (requiring that *for all* pairs of adjacent inputs *there exists* a, possibly different, simulator), one obtains a definition that is equivalent to IND-CDP.

The above definitions extend naturally to interactive protocols. We restrict our attention here to two-party protocols in the honest-but-curious model (where participants are assumed to follow the protocol), but note that this extends to multi-party protocols, and that malicious participants can be forced to abide by the protocol by requiring that they accompany the messages they send with a proof of their *consistency*.

We represent each party in an interactive protocol by a sequence of probabilistic algorithms—one for each round—sharing local state, in a similar way to *interactive functions* [23]. In a two-party protocol Π, we denote $\textsc{view}^A_\Pi(x, y)$ the view of party A when the protocol is run with input x for A and y for the other party, i.e. the joint probability distribution over x, the messages exchanged, and the random coins of A.

Definition 4 (Differentially private two-party computation)**.** *A two-party protocol $\Pi = (A, B)$ is (ϵ, δ)-differentially private for party B if for all values x, $\textsc{view}^A_\Pi(x, \cdot)$ is (ϵ, δ)-differentially private. This definition extends to IND-CDP and SIM-CDP by considering a family of protocols $\{\Pi_\eta\}_{\eta \in \mathbb{N}}$ rather than a single protocol.*

Intuitively, a two-party protocol is differentially private for participant B if participant A gets only a differentially private view of B's secret input.

B. Computing Hamming distance privately

We illustrate the workings of our framework on a two-party protocol for computing the Hamming distance between ℓ-bit vectors (i.e. the number of positions at which they differ). This protocol achieves two-sided privacy guarantees and is inspired by a protocol first proposed by Mironov et al. [34] as a candidate for separating information-theoretic from computational differential privacy. The only differences are that in our protocol both parties learn an approximation of the Hamming distance of their inputs, and that we factor out the post-processing done by participants after the protocol terminates, which is only needed to argue that the protocol achieves good utility. Figure 1 describes the protocol in detail.

The protocol builds on a public key encryption scheme $(\mathcal{KG}, \mathcal{E}, \mathcal{D})$. The key generation algorithm $\mathcal{KG}(\eta)$ outputs a pair of keys (pk, sk); the public key pk includes a description of the plaintext space, which we assume to be $\mathbb{Z}_n$ for some n such that $n - \ell \in \Omega(\eta)$. We require the scheme be additively homomorphic and affine, i.e. there must exist efficient operations $\oplus_{pk}$ and $\otimes_{pk}$ such that for all $a, b \in \mathbb{Z}_n$:

1) $\mathcal{D}(\mathcal{E}_{pk}(a) \oplus_{pk} \mathcal{E}_{pk}(b)) = a + b \mod n$
2) $\mathcal{D}(\mathcal{E}_{pk}(a) \otimes_{pk} b) = ab \mod n$

INPUT: Alice holds $a \in \{0,1\}^\ell$, Bob holds $b \in \{0,1\}^\ell$

1) Alice generates a fresh pair of keys (pk, sk) using $\mathcal{KG}$ and computes $\tilde{a}_i = \mathcal{E}(a_i)$ for $i = 1 \ldots \ell$. It sends over to Bob the public key pk and the vector $\tilde{a}$.
2) Bob computes $\tilde{c}_i = \begin{cases} \tilde{a}_i & \text{if } b_i = 0 \\ 1 \ominus \tilde{a}_i & \text{otherwise} \end{cases}$ for $i = 1 \ldots \ell$, samples k_B from $\textsc{Geom}(\exp(\epsilon))$ and sends $h_B = \bigoplus_{i=1}^{\ell} \tilde{c}_i \oplus \mathcal{E}(k_B)$ to Alice.
3) Alice samples k_A from $\textsc{Geom}(\exp(\epsilon))$ and sends over $h_A = (\mathcal{D}(h_B) + k_A) \mod n$ to Bob.

Fig. 1. Two-party protocol for privately computing Hamming distance between bit-vectors

In the remainder, we omit the subscript pk from operations if there is no confusion, and write $1 \ominus x$ for $\mathcal{E}(1) \oplus (x \otimes (-1))$; when x is an encryption of a value $a \in \{0,1\}$, $\mathcal{D}(1 \ominus x) = \bar{a}$, the complement of a. One encryption scheme matching our requirements is Paillier cryptosystem [37], whose semantic security is based on the decisional composite residuosity assumption. In Paillier cryptosystem, $\oplus$ corresponds to multiplication modulo n^2 and $\otimes$ to exponentiation modulo n^2. To extend the proof to the malicious case, the BGN cryptosystem [9] could be used instead, which supports efficient non-interactive proofs of ciphertext consistency. We also note that the requirement that the encryption scheme is affine can be dropped at an extra cost in communication, by having Alice send both an encryption of each bit of its input vector and of its complement.

Each party protects its data by adding noise sampled from a symmetric geometric distribution. This *geometric mechanism* is the discrete counterpart of the Laplace mechanism used to sanitize real-valued queries. Formally, the distribution $\textsc{Geom}(\alpha)$ with scale parameter $\alpha > 1$ is the discrete distribution with support $\mathbb{Z}$ and probability mass function

$$p(k) \stackrel{\text{def}}{=} \frac{\alpha - 1}{\alpha + 1}\, \alpha^{-|k|}$$

Lemma 1 (Geometric mechanism [21])**.** *Let $f : A \to \mathbb{Z}$ be a function with sensitivity*

$$S_f \stackrel{def}{=} \max_{a,a' | d(a,a') \leq 1} |f(a) - f(a')|$$

*then $M(a) \stackrel{def}{=} k \xleftarrow{\$} \textsc{Geom}(\exp(\epsilon / S_f))$; **return** $f(a) + k$ is an ϵ-differentially private algorithm.*

We analyze the protocol in the honest-but-curious model; the privacy guarantees we prove reflect its asymmetry: the protocol is information-theoretic differentially private for Bob, but only computationally differentially private for Alice. However, both obtain similarly useful estimates of the Hamming distance between their inputs.

We first show that the protocol is ϵ-SIM-CDP for Alice. We represent a run of the protocol as a probabilistic program; we use a sugared version of the syntax presented in Section III-B. The view of each participant is obtained by projecting the output distribution on the relevant program variables. For instance, the view of Bob is given by the procedure $\textsc{View}^B_{\Pi_\eta}(b, a)$ in Figure 2. We show that this view is SIM-CDP using the simulator $F_\eta(b, a)$ in Figure 2. Instead of using a to compute the encrypted noisy estimate h_B of $h(a, b)$, the simulator uses the all-zero vector 0^ℓ; moreover, it computes h_A directly from $h(a, b)$ rather than from h_B.

If the underlying encryption scheme is semantically secure (equivalently, IND-CPA secure), no efficient adversary will be able to distinguish one or many encryptions of 0 from the encryption of any other value (even chosen adaptively). Indeed, we show by reduction that for any $a, b \in \{0,1\}^\ell$, the probability that an efficient adversary $\mathcal{B}$ distinguishes $\textsc{View}^B_{\Pi_\eta}(b, a)$ from $F_\eta(b, a)$ is the same as the IND-CPA advantage of some adversary $\mathcal{A}$ against the homomorphic encryption scheme. Formally, we define the IND-CPA advantage of an adversary $\mathcal{A}$ as the quantity

$$\mathbf{Adv}^{\text{IND-CPA}}_{\mathcal{A}}(\eta) \stackrel{\text{def}}{=} \left| \Pr\left[\mathcal{A}^{\mathcal{E}(\cdot)}(pk) = 1\right] - \Pr\left[\mathcal{A}^{\mathcal{E}(0)}(pk) = 1\right] \right|$$

where the probability is taken over the random coins of $\mathcal{A}$, its oracle, and $(pk, sk) \xleftarrow{\$} \mathcal{KG}(\eta)$.

The adversary $\mathcal{A}$ that we exhibit as witness behaves as $\textsc{View}^B_{\Pi_\eta}(b, a)$, but uses its oracle $\mathcal{E}(\cdot)$ to encrypt values, forwards the resulting *view* to $\mathcal{B}$, and returns the answer it gets back. Thus, when $\mathcal{A}$ is given $\mathcal{E}(\cdot)$ as oracle, it answers as $\mathcal{B}$ would do given a view of the real protocol, and when its oracle is $\mathcal{E}(0)$, it answers as $\mathcal{B}$ would do given a simulated view. We prove this using the non-approximate fragment of the logic of EasyCrypt, for which all the tools that have proven successful for reasoning about cryptographic constructions are available [3].

To see that $F_\eta(b, \cdot)$ is ϵ-differentially private, observe that the only output that depends on its second input is h_A, which is computed from the result of adding noise sampled from $\textsc{Geom}(\exp(\epsilon))$ to $h(a, b) + k_B$. But the sensitivity of Hamming distance (and $h(\cdot, b) + k_B$) is 1, and so it follows from Lemma 1 that $F_\eta(b, \cdot)$ is ϵ-differentially private. This part of the proof uses the novel extension of EasyCrypt described in next section for reasoning in the apRHL logic. Lemma 1 is derived from the probability mass function of the geometric distribution and reformulated as a parametric judgment:

$$\models x \xleftarrow{\$} \textsc{Geom}(\alpha) \sim_{\alpha^k, 0} y \xleftarrow{\$} \textsc{Geom}(\alpha) : \Psi \Rightarrow a + x = b + y$$

where $\Psi \stackrel{\text{def}}{=} 1 < \alpha \wedge |a - b| \leq k$. By combining this with the sequential composition rule [seq] of Fig. 3, one is left to prove a non-approximate judgment relating the prefix of $F_\eta(b, a)$ up to line 10 with itself, namely

$$\models F_\eta(b, a)_{[1-10]} \sim F_\eta(b', a')_{[1-10]} : h(a, a') \leq 1 \wedge b = b' \Rightarrow \Phi$$

procedure $\textsc{View}^{B}_{\Pi_\eta}(b, a)$
  $(pk, sk) \leftarrow \mathcal{KG}(\eta)$
  $h_B \leftarrow \mathcal{E}(0)$
  for $i \leftarrow 1$ **to** ℓ **do**
    $\tilde{a}[i] \leftarrow \mathcal{E}(a[i])$
    if $b[i] = 0$ **then** $\tilde{c} \leftarrow \tilde{a}[i]$ **else** $\tilde{c} \leftarrow 1 \ominus \tilde{a}[i]$
    $h_B \leftarrow h_B \oplus \tilde{c}$
  end for
  $k_B \xleftarrow{\$} \textsc{Geom}(\exp(\epsilon))$
  $h_B \leftarrow h_B \oplus \mathcal{E}(k_B)$
  $k_A \xleftarrow{\$} \textsc{Geom}(\exp(\epsilon))$
  $h_A \leftarrow (\mathcal{D}(h_B) + k_A) \mod n$
  return $(b, pk, \tilde{a}, k_B, h_B, h_A)$

procedure $F_\eta(b, a)$
  $(pk, sk) \leftarrow \mathcal{KG}(\eta)$
  $h_B \leftarrow \mathcal{E}(0)$
  for $i \leftarrow 1$ **to** ℓ **do**
    $\tilde{a}[i] \leftarrow \mathcal{E}(0)$
    if $b[i] = 0$ **then** $\tilde{c} \leftarrow \tilde{a}[i]$ **else** $\tilde{c} \leftarrow 1 \ominus \tilde{a}[i]$
    $h_B \leftarrow h_B \oplus \tilde{c}$
  end for
  $k_B \xleftarrow{\$} \textsc{Geom}(\exp(\epsilon))$
  $h_B \leftarrow h_B \oplus \mathcal{E}(0)$
  $k_A \xleftarrow{\$} \textsc{Geom}(\exp(\epsilon))$
  $h_A \leftarrow (h(a, b) + k_B + k_A) \mod n$
  return $(b, pk, \tilde{a}, k_B, h_B, h_A)$

Fig. 2. Bob's view of the protocol for computing Hamming distance (left) and simulator used in the proof of computational differential privacy (right)

where we mark with a single quote the variables of the program on the right-hand side, and define post-condition Φ as the relation

$$(b, pk, \tilde{a}, k_B, h_B) = (b', pk', \tilde{a}', k'_B, h'_B) \\ \wedge\ |h(a, b) - h(a', b')| \leq 1$$

Proving this boils down to showing that Hamming distance is 1-sensitive and that the joint distribution of the rest of the output variables of $F_\eta(b, \cdot)$ is fully determined by its first input b, on which both programs coincide.

The proof that the protocol is ϵ-differentially private for Bob is simpler. The view of Alice is defined analogously to Bob's but projecting on $(a, pk, sk, \tilde{a}, h_B, k_A, h_A)$ instead. Of these variables, only h_A and h_B depend on b, but h_B is an encryption of $h(a, b) + \textsc{Geom}(\exp(\epsilon))$ and h_A can be computed from h_B. So again, Lemma 1 implies that $\textsc{View}^{A}_{\Pi_\eta}(a, \cdot)$ is ϵ-differentially private.

III. Extending the EasyCrypt Framework

Providing support for reasoning about differential privacy required to extend EasyCrypt significantly. As a first step, we defined a library for describing discrete probability distributions, and extended the expression language of EasyCrypt with a new probabilistic sampling primitive $\mathsf{sample}(\cdot)$, taking as argument a representation of a distribution. In contrast, previous versions of EasyCrypt only allowed sampling from a pre-defined set of uniform distributions with finite support. Secondly, we have adapted the verification component of EasyCrypt to prove properties of user-defined probability distributions and to reason about approximate forms of program equivalence. This section presents these extensions.

A. Distributions

EasyCrypt features a polymorphically-typed expression language that can be extended by the user with new types and deterministic operators, but hitherto could not be extended with new probabilistic operators. In this work, we take advantage of this rich expression language to define an abstract data type of discrete probability distributions, and introduce a new primitive probabilistic operator $\mathsf{sample}(\cdot)$ to choose values from them.

We model probability distributions and events using an embedding of higher-order types into the first-order logic of SMT solvers. Our embedding is inspired by [26, 33]. Concretely, we introduce two polymorphic types: distr_A, of sub-probability distributions over a discrete sample space A, and event_A of events in the σ-algebra 2^A. Sub-probability distributions correspond to measures that may assign a value less than 1 to the entire probability space; this relaxation proved useful to encode runtime assertions (see Section III-B). Moreover, we define operators $@_E$ and $@_D$ of types $\mathsf{event}_A \times A \to \mathbb{B}$ and $\mathsf{distr}_A \times \mathsf{event}_A \to \mathbb{R}$, respectively, where $\mathbb{B}$ denotes the type of booleans and $\mathbb{R}$ the type of reals. The former operator corresponds to set membership, the latter corresponds to the measure defined by a sub-probability distribution. In the sequel we write $a \in E$ for $@_E(E, a)$ and $\mu(E)$ for $@_D(\mu, E)$.

We declare and axiomatize set-theoretical constants and operators over event_A, including union, intersection, complement, singleton sets, and constants denoting the empty set and the entire sample space. Note that we do not provide an operator for converting a boolean-valued function into an event; however, users can introduce for any function $\mathsf{f} : A \to \mathbb{B}$, an event $\mathsf{E_f} : \mathsf{event}_A$ together with the axiom $\forall (x : A).\ x \in \mathsf{E_f} \iff \mathsf{f}\ x = \mathsf{true}$.

Discrete sub-probability distributions in distr_A are axiomatized so that they respect the axioms of sub-probability measures, monotonicity and extensionality. Formally, for all $\mu, \mu' : \mathsf{distr}_A$ and $E, F : \mathsf{event}_A$, we assume:

1) $0 \leq \mu(E) \leq 1$
2) $\mu(\emptyset) = 0$
3) $\mu(E \cup F) = \mu(E) + \mu(F) - \mu(E \cap F)$
4) $\mu \leq \mu' \iff \forall a : A,\ \mu(\{a\}) \leq \mu'(\{a\})$
   where $\mu \leq \mu' \stackrel{\text{def}}{=} \forall E : \mathsf{event}_A.\ \mu(E) \leq \mu'(E)$.
5) $\mu = \mu' \iff \mu \leq \mu' \wedge \mu' \leq \mu$

Other useful properties are derived from these axioms as lemmas, e.g. $\forall E, F : \mathsf{event}_A,\ E \subseteq F \implies \mu(E) \leq \mu(F)$. Moreover, we define an operator denoting the weight of a sub-probability distribution and a predicate characterizing its support as follows:

$$\mathsf{weight}(\mu : \mathsf{distr}_A) \stackrel{\text{def}}{=} \mu(A)$$
$$\mathsf{support}(\mu : \mathsf{distr}_A, a : A) \stackrel{\text{def}}{=} 0 < \mu(\{a\})$$

In order to characterize differential privacy and prove the validity of apRHL judgments from first principles, we encode the notion of α-distance between sub-probability distributions of Barthe et al. [4]. The α-distance $\Delta_\alpha(\mu, \mu')$ between μ and μ' is defined as the smallest real number δ such that for all events E:

$$\mu(E) \leq \alpha\ \mu'(E) + \delta\ \wedge\ \mu'(E) \leq \alpha\ \mu(E) + \delta$$

Note that from the properties of discrete distributions above, one has $\Delta_\alpha(\mu, \mu') = 0$ iff for every singleton set $\{a\}$

$$\mu(\{a\}) \leq \alpha\ \mu'(\{a\})\ \wedge\ \mu'(\{a\}) \leq \alpha\ \mu(\{a\})$$

We encode α-distance by introducing an abstract operator

$$\Delta : \mathbb{R} \times \mathsf{distr}_A \times \mathsf{distr}_A \to \mathbb{R}$$

and axioms encompassing the above definition; in the sequel we write $\Delta_\alpha(\mu, \mu')$ for $\Delta(\alpha, \mu, \mu')$.

We provide instances of the above abstract data type of discrete sub-probability distributions for all distributions given as primitives in previous versions of EasyCrypt, including the uniform distribution over booleans, bitstrings, and integer intervals, as well as distributions that underlie differential privacy mechanisms. For instance, we formalize the geometric distribution by introducing an operator $\mathsf{Geom} : \mathbb{R} \to \mathsf{distr}_{\mathbb{Z}}$ and an axiom stating that for every $\alpha > 1$ and integer k:

$$\textsc{Geom}(\alpha)(\{k\}) \quad = \quad \frac{\alpha - 1}{\alpha + 1}\ \alpha^{-|k|}$$

Based on this axiom, we prove that for all integers a, b,

$$|a - b| \leq k\ \implies\ \Delta_{\alpha^k}(a + \textsc{Geom}(\alpha), b + \textsc{Geom}(\alpha)) = 0$$

This implies the validity of the apRHL judgment

$$\models x \xleftarrow{\$} \textsc{Geom}(\alpha) \sim_{\alpha^k, 0} y \xleftarrow{\$} \textsc{Geom}(\alpha) : \Psi \Rightarrow a + x = b + y$$

where $\Psi \stackrel{\text{def}}{=} 1 < \alpha \wedge |a - b| \leq k$, which corresponds to a direct embedding of Lemma 1. (Validity of apRHL judgments is defined formally in Section III-C; the derivation of the validity of this judgment in EasyCrypt is shown in the Appendix.)

B. Programs

Programs in EasyCrypt are written in an imperative programming language with procedure calls and probabilistic assignments, called pWHILE. Commands are defined inductively by the following grammar:

$\mathcal{C}$	::=	**skip**	nop
	|	$\mathcal{V} \leftarrow \mathcal{E}$	deterministic assignment
	|	$\mathcal{V} \xleftarrow{\$} \mathsf{sample}(\mathcal{E})$	probabilistic assignment
	|	**if** $\mathcal{E}$ **then** $\mathcal{C}$ **else** $\mathcal{C}$	conditional
	|	**while** $\mathcal{E}$ **do** $\mathcal{C}$	while loop
	|	$\mathcal{V} \leftarrow \mathcal{P}(\mathcal{E}, \ldots, \mathcal{E})$	procedure call
	|	$\mathcal{C};\ \mathcal{C}$	sequence

where $\mathcal{V}$ and $\mathcal{P}$ are sets of variable and procedure identifiers, respectively, and expressions $\mathcal{E}$ are built from primitive and user-defined constants and operators. In this work, we generalize probabilistic assignments to be of the form $x \xleftarrow{\$} \mathsf{sample}(\mu)$, where x is a variable of type A and μ is an expression of type distr_A. For conciseness, in code snippets we write simply $x \xleftarrow{\$} \mu$ instead of $x \xleftarrow{\$} \mathsf{sample}(\mu)$. Runtime assertions, which were introduced as primitives by Barthe et al. [5], can be encoded as macros: **abort** and **assert** e, where e is a boolean expression, respectively abbreviate sampling from the null sub-probability distribution, and the command **if** e **then skip else abort**.

Programs are given by a command together with an environment mapping procedures to their definitions; the denotation of a program is a function from an initial memory to a sub-probability distribution over final memories. Let $\mathbb{D}(X)$ be the set of discrete sub-probability distributions over X. A memory $m \in \mathcal{M}$ maps program variables to values. An expression e of type A denotes a function from memories to elements of the set-theoretical interpretation of A; an expression μ of type ditsr_A denotes a function from memories to sub-probability distributions over the set-theoretical interpretation of A. Finally, a program c denotes a function $[\![c]\!] : \mathcal{M} \to \mathbb{D}(\mathcal{M})$. For a detailed description of the semantics of pWHILE we refer the reader to Barthe et al. [5].

C. Reasoning about approximate relational properties

EasyCrypt provides three main verification components: an interactive but somewhat automated theorem prover to establish relational properties of probabilistic programs, a mechanism to derive inequalities between probabilities from proven relational properties, and an algorithm to directly compute (bounds of) the probability of an event w.r.t. the output distribution generated by a program.

In order to support reasoning about differential privacy, we first extended the program verifier of EasyCrypt to support reasoning about the approximate relational logic apRHL [4], a generalization of the pRHL logic supported in previous versions of EasyCrypt. Judgments in apRHL are of the form

$$\vdash c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi \tag{1}$$

with a *skew* $\alpha \geq 1$ and a *slack* $0 \leq \delta \leq 1$ (pRHL judgments correspond to the case $\alpha = 1, \delta = 0$). As for pRHL, pre- and post-conditions are first-order formulae built from the grammar:

$$\Psi, \Phi ::= \quad e \mid \neg\Phi \mid \Psi \wedge \Phi \mid \Psi \vee \Phi \mid \Psi \to \Phi \mid \forall x.\ \Phi \mid \exists x.\ \Phi$$

where e stands for a boolean expression over quantified logical variables and program variables tagged with either $\langle 1 \rangle$ or $\langle 2 \rangle$ to indicate whether they should be interpreted as variables in the program on the left- or right-hand side, respectively. A relational formula is interpreted as a relation on program memories. For example, the formula $x\langle 1 \rangle + y\langle 2 \rangle \leq z\langle 1 \rangle$ is interpreted as the relation

$$R = \{(m_1, m_2) \mid m_1(x) + m_2(y) \leq m_1(z)\}$$

Validity is defined in terms of an operator that lifts binary relations on memories to relations on distributions over memories; this *lifting* operator has close connections with probabilistic bisimulations, maximum flows in transport networks,

$$\frac{\models c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi' \quad \models c_1' \sim_{\alpha',\delta'} c_2' : \Phi' \Rightarrow \Phi}{\models c_1; c_1' \sim_{\alpha\alpha',\delta+\delta'} c_2; c_2' : \Psi \Rightarrow \Phi}\,[\mathsf{seq}]$$

$$\frac{\forall m_1, m_2.\ m_1 \Psi m_2 \implies \Delta_\alpha(f_1 \circ \mu_1, f_2 \circ \mu_2) \leq \delta \qquad f_1, f_2 \text{ bijective}}{\models x_1 \xleftarrow{\$} \mu_1 \sim_{\alpha,\delta} x_2 \xleftarrow{\$} \mu_2 : \Psi \Rightarrow f_1(x_1)\langle 1\rangle = f_2(x_2)\langle 2\rangle \wedge \mathsf{support}(\mu_1, x_1)\langle 1\rangle \wedge \mathsf{support}(\mu_2, x_2)\langle 2\rangle}\,[\mathsf{rand}]$$

$$\frac{\models c_1 \sim_{\alpha,\delta} c_2 : \varphi \wedge b_1\langle 1\rangle \wedge b_2\langle 2\rangle \wedge k = e \Rightarrow \varphi \wedge b_1\langle 1\rangle = b_2\langle 2\rangle \wedge k < e \qquad \varphi \wedge n \leq e \implies \neg b\langle 1\rangle}{\models \textbf{while } b_1 \textbf{ do } c_1 \sim_{\alpha^n, n\,\delta} \textbf{while } b_2 \textbf{ do } c_2 : \varphi \wedge b_1\langle 1\rangle = b_2\langle 2\rangle \wedge 0 \leq e \Rightarrow \varphi \wedge \neg b_1\langle 1\rangle \wedge \neg b_2\langle 2\rangle}\,[\mathsf{while}]$$

Fig. 3. Selected apRHL rules

and the Kantorovich metric [15]. Formally, the (α, δ)-lifting of a relation $R \subseteq A \times B$ is the relation $\sim_R^{\alpha,\delta}$ over $\mathbb{D}(A) \times \mathbb{D}(B)$ that holds for a pair of distributions μ_1, μ_2 iff there exists a distribution $\mu : \mathbb{D}(A \times B)$ such that:

1) $\forall a \in A, b \in B.\ 0 < \mu(\{(a,b)\}) \implies a\ R\ b$
2) $\pi_1(\mu) \leq \mu_1 \ \wedge\ \pi_2(\mu) \leq \mu_2$
3) $\Delta_\alpha(\pi_1(\mu), \mu_1) \leq \delta \ \wedge\ \Delta_\alpha(\pi_2(\mu), \mu_2) \leq \delta$

where the projections $\pi_1(\mu)$ and $\pi_2(\mu)$ are defined as:

$$\pi_1(\mu)(\{a\}) \stackrel{\text{def}}{=} \mu(\{a\} \times B) \qquad \pi_2(\mu)(\{b\}) \stackrel{\text{def}}{=} \mu(A \times \{b\})$$

Judgment (1) above is valid iff for all memories m_1 and m_2 the following holds:

$$m_1\ \Psi\ m_2 \implies (\llbracket c_1 \rrbracket\ m_1) \sim_\Phi^{\alpha,\delta} (\llbracket c_2 \rrbracket\ m_2)$$

From this, it can be proved [4, Lemma 5] that for any event[1] E that holds either in both or in neither of a pair of memories satisfying Φ:

$$m_1\ \Psi\ m_2 \implies \Pr[c_1, m_1 : E] \leq \alpha \Pr[c_2, m_2 : E] + \delta$$

which is exactly the kind of inequalities one needs to prove differential privacy. We plan to embed this mechanism to derive inequalities from apRHL judgments in EasyCrypt and extend the algorithm that directly computes (bounds of) the probability of events to handle our extended form of probabilistic assignments.

D. Verification of apRHL judgments

EasyCrypt uses a hybrid strategy to verify relational goals. It combines a weakest pre-condition transformer, with a tactic language for applying logic rules and semantics-preserving program transformations. The weakest pre-condition transformer can be applied to compute relational weakest pre-conditions w.r.t. fragments of deterministic, straight-line code, whereas tactics can be applied whenever ingenuity is required, and for reasoning about loops or adversarial code. Applying a tactic or the weakest pre-condition transformer results in one or several new verification goals, as well as a number of proof obligations in the form of first-order formulae, which are discharged by sending them to SMT solvers and automated theorem provers.

[1]Events in this setting are boolean expressions over program variables, and should not be confused with elements of the abstract data type defined in the previous section.

Goals that fall in the non-approximate pRHL fragment of the logic can be proven using all tactics previously available in EasyCrypt. To prove goals that fall outside this fragment, we have adapted and extended the existing weakest pre-condition transformer and the language of tactics. Figure 3 shows an excerpt of apRHL rules that we have implemented as tactics in our extension of EasyCrypt.

The sequential composition rule [seq] reflects the composition theorem of approximate differential privacy (Theorem 1). The corresponding tactic requires the user to provide the intermediate values α, α' and δ, δ' (if not given, they are assumed to be 1 and 0, respectively); these could be avoided by introducing existential variables, but we have not yet explored this possibility.

Rule [rand] is a generalization of the apRHL rule of the same name in [4], which could only establish the equality of the sampled variables as post-condition. This rule reduces proving a judgment relating two probabilistic assignments to proving an upper bound of the α-distance between two distributions. Thus, one can use it to prove differential privacy guarantees of standard mechanisms, like the geometric mechanism used in Section II. Given a distribution μ in A, and a bijection $f : A \to B$, $f \circ \mu$ in Figure 3 denotes the sub-probability distribution over B with measure $\mu(f^{-1}(\cdot))$. When $\delta = 0$, the verifier exploits the axiomatization of α-distance and generates a simplified verification condition of the form

$$\Psi \implies \forall a : B.\ \mu(\{a\}) \leq \alpha\, \mu'(\{a\}) \ \wedge\ \mu'(\{a\}) \leq \alpha\, \mu(\{a\})$$

Rule [while] may be used to relate two loops that execute in lock-step; it requires to provide an appropriate loop invariant. The fact that the loops progress in lock-step follows from the first premise, but one also needs to ensure that they both terminate after a given number of iterations. Therefore, the rule requires exhibiting a non-negative integer variant e and a (constant) bound n, and showing that: 1) a loop iteration strictly increases the value of e, and 2) whenever the expression e reaches the bound n, the loop guard becomes false.

IV. Applications to Smart Metering

Smart-meters collect information about household electricity usage or generation at a granularity ranging from 15 to 30 minutes. Aggregates of these results are used by different actors in the energy industry: energy distributors, providers and generators require timely statistics to support their business

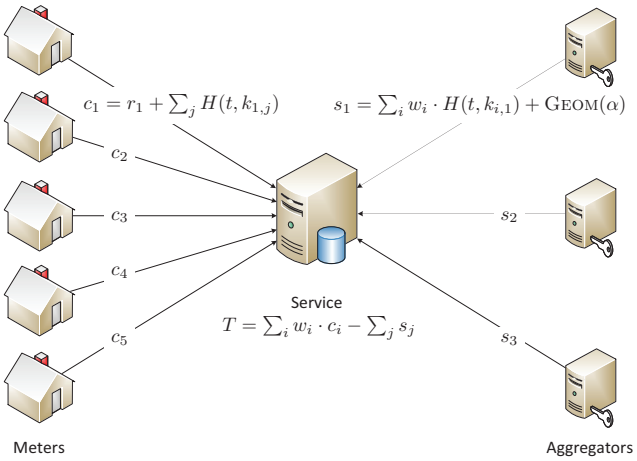

Fig. 4. Aggregation of readings using weights $\vec{w}$ in an infrastructure with 5 meters and 3 aggregators. The setup phase ensures that meter i and aggregator j share a key $k_{i,j}$.

processes; distributors require statistics about the peak loads that particular lines may be subject to; while providers need to know the exact consumption of their customer base for every settlement period. All entities require statistics to support the development of their future services or financial forecasting at different geographic or demographic levels. We present and verify a novel protocol that allows readings from smart meters to be aggregated in weighted sums and used by different entities to serve their needs, without compromising the privacy of customers by disclosing fine-grained consumption data. Furthermore, missing readings due to failing meters or unreliable networks do not impede the aggregation of the readings collected.

A. Aggregation protocol

Three types of entities are involved in the aggregation protocol. A set of *smart meters* that record and report readings r_i, a set of *aggregators* or *aggregation authorities* that facilitate private aggregation, and finally a *service provider* that wants to learn a weighted aggregate $T = \sum_i w_i \cdot r_i$. The goal of the protocol is to let the service provider compute an approximation of T without revealing too much about any individual reading r_i to any other entity beyond the meter that generated it. Figure 4 illustrates how the protocol works during the aggregation phase; no communication between readers and aggregators is needed.

Setup. In the setup phase each meter and each aggregator generates its own public/secret key pair. Every meter is then seeded with the public keys of all aggregators, and every aggregator with the public keys of all meters. In this way, each pair of a meter and an aggregator can derive a shared symmetric key.

Concretely, let $\mathcal{G}$ be a Diffie-Hellman group of prime order q and g a generator; these depend on the security parameter η which, for the sake of clarity, we omit from the presentation. Let P be the number of aggregators and N the number of meters participating in the protocol. In the setup phase, each of these entities generates a Diffie-Hellman key pair by uniformly choosing a random value $x \in \mathbb{Z}_q$ and setting x as their private key and publishing g^x as their public key. Each pair of meter and aggregator then exchange a shared key; if x is the secret key of the meter and y the secret key of the aggregator, their pairwise shared key is g^{xy}.

An honest-but-curious authority coordinates the setup phase by engaging all participating entities. The result is equivalent to executing the SETUP procedure in Fig. 5. The setup phase results in all meters being seeded with the valid public keys of all aggregation authorities, and *vice versa*. As a result, a shared key can be derived between all pairs of meters and aggregation authorities. Any secure mechanism to ensure this, including a certificate chain, an auditing mechanism or factory initialization can be used, but may have different systems' ramifications. The setup phase is only performed once per meter for a stable set of authorities. Public keys of new meters can be incrementally added to authorities.

Aggregation. Meters generate readings over time. We consider readings are associated with labels $t \in \mathcal{T}$. For example, t may be the label attached to all readings at a particular time period, or from a specific geographic area. The service provider wants to aggregate readings $r_{i,t}$ with label t from all meters and compute their weighted sum $T = \sum_i w_i \cdot r_{i,t}$.

First, the service provider requests a blinded reading $c_{i,t}$ from each meter i, computed as in the procedure GETREADING(i, t) in Fig. 5. The blinded reading $c_{i,t}$ is computed by each meter as the sum of its reading $r_{i,t}$ and ephemeral session keys shared with the aggregation authorities. These session keys are generated by applying a hash function $H : \mathcal{T} \times \mathcal{G} \to \mathbb{Z}_n$ to the label t of the requested reading, and the shared key $k_{i,j}$. The resulting value $c_{i,t}$ is a simple n-bit integer (e.g. a 32-bit machine integer, of course $n \ll q$), and the only computation involved in producing it on the meter is the application of a hash function and simple integer arithmetic.

The service provider may request readings from all meters, but some may fail to arrive due to meter malfunction or network unavailability or latency. The service provider gathers the identities of the meters that have responded in a set M and sends each aggregator j a request for an opening s_j computed using the procedure GETOPENING$(j, t, \vec{w}, M)$ in Fig. 5, to facilitate the aggregation of readings with label t from meters in M, using weights $\vec{w}$. Each aggregator computes the weighted sum of the session keys shared with each meter for tag t, and adds to the result noise sampled from the distribution GEOM(α). The scale parameter $\alpha = \exp(\epsilon/\Delta)$ of the distribution is chosen to offer ϵ-differential privacy on the basis of the sensitivity of a single reading being Δ, an upper bound on the maximum possible value of $w_i r_{i,t}$. The result of this sum, a single n-bit number, is returned to the service provider—the actual blinded readings $c_{i,t}$ need not be transmitted to the aggregators.

To get the sought result, the service provider sums the

```
1: procedure SETUP(N, P)
2:     g ← G*
3:     for j ← 1 to P do
4:         y_j ← Z_q
5:     end for
6:     for i ← 1 to N do
7:         x_i ← Z_q
8:         for j ← 1 to P do
9:             k_{i,j} ← g^{x_i y_j}
10:        end for
11:    end for

12: procedure GETREADING(i, t)
13:    c_{i,t} ← r_{i,t} + Σ_{j=1}^{P} H(t, k_{i,j})  mod n
14:    return c_{i,t}

15: procedure GETOPENING(j, t, w⃗, M)
16:    s_j ← Σ_{i∈M} w_i · H(t, k_{i,j})  mod n
17:    u_j ← GEOM(α)
18:    return s_j + u_j  mod n
19: procedure AGGREGATE(t, w⃗)
20:    M ← ∅
21:    for i ← 1 to N do
22:        c_{i,t} ← GETREADING(i, t)
23:        if c_{i,t} ≠ ⊥ then M ← M ∪ {i}
24:    end for
25:    for j ← 1 to P do
26:        s_j ← GETOPENING(j, t, w⃗, M)
27:    end for
28:    T ← Σ_{i∈M} w_i · c_{i,t} − Σ_{j=1}^{P} s_j  mod n
29:    return T
```

Fig. 5. Procedures used by participants of the distributed protocol for aggregating smart meter readings

weighted blinded readings, and subtracts from the result the responses from all the aggregators. The result is a noisy estimate of the aggregate value T, computed as in the procedure AGGREGATE in Figure 5.

The full aggregation process, initiated by the service provider, requires only $2 \cdot |M| \cdot |A|$ hash function evaluations, distributed across all meters and aggregators, no public key operations, and only modular multiplications and additions at the service provider. In terms of utility the noise added to the aggregate has mean zero and variance

$$P \sum_{k \in \mathbb{Z}} \frac{\alpha - 1}{\alpha + 1} \alpha^{-|k|} k^2 = \frac{2P\alpha}{(\alpha - 1)^2}$$

The size of n must be chosen to make integer overflow or underflow unlikely; Chebyshev's inequality can be used as a guide. Alternatively, aggregators may add noise from a truncated geometric distribution to avoid overflows or underflows, or to guarantee hard bounds on the added noise.

Billing. We note that the aggregation protocol can be adapted as a simple linear billing protocol by considering a single meter i and a set of labels L, rather than a set of meters and a single label. A service provider defines readings with labels $t \in L$ as originating from a single household within a billing period. The weights w_t represent the applicable tariff at the time of the reading. The final bill is, as expected, $T = \sum_{t \in L} w_t \cdot r_{i,t}$. Adding noise to bills is expensive, given that real money is at stake. Thus such an approach would have to be combined with other protocols that combine differentially private billing with rebates [13]. Alternatively, such queries may have no noise added to them, foregoing any protection based on differential privacy. We leave a security analysis of such protocols to future work.

B. Differential privacy

We prove that the protocol described above is computationally differentially private under mild assumptions on the group $\mathcal{G}$ and the hash function used to derive per-label shared keys. Namely, we require the group $\mathcal{G}$ and the hash function H satisfy the *Hash Diffie-Hellman assumption* (HDH).

Definition 5 (Hash Diffie-Hellman Assumption)**.** *Let $\mathcal{G}$ be a group of prime order q and $H : \mathcal{G} \to X$. The advantage $\mathbf{Adv}^{\mathsf{HDH}}_{\mathcal{A}}$ of an adversary $\mathcal{A}$ in solving the Hash Diffie-Hellman problem on $\mathcal{G}$ and H, is defined as the quantity*

$$\left|\mathsf{Pr}[\mathcal{A}(g, g^a, g^b, H(g^{ab})) = 1] - \mathsf{Pr}[\mathcal{A}(g, g^a, g^b, x) = 1]\right|$$

where the probability is taken over the random choices of $\mathcal{A}$, and uniformly chosen $g \in \mathcal{G}^*$ (the set of generators of $\mathcal{G}$), $a, b \in \mathbb{Z}_q$, and $x \in X$.

When the hash function H is a secure key-derivation function, the HDH assumption is weaker than the Decisional Diffie-Hellman assumption (DDH), in which the task of the adversary is to distinguish a random element in the group from g^{ab} given g, g^a, g^b, (i.e. if it is hard to solve DDH then it must be also hard to solve HDH, but the converse may not hold) [43]. However, HDH is in general stronger than the Computational Diffie-Hellman assumption, in which the task of the adversary is to compute g^{ab} given g, g^a, g^b.

We show in the honest-but-curious model that the protocol preserves privacy for a meter if there is at least one aggregator that behaves honestly, even if the rest of the aggregators, meters, and the service provider collude.

Theorem 2 (Aggregation privacy)**.** *When $\alpha = \exp(\epsilon/\Delta)$, the protocol described in the previous section is (ϵ, δ)-SIM-CDP for any meter against a coalition of all other meters and authorities, except one honest aggregator, where $\delta = \mathbf{Adv}^{\mathsf{HDH}}_{\mathcal{A}}$ for some adversary $\mathcal{A}$ against the HDH problem on $\mathcal{G}$ and $H(t, \cdot)$.*

In practice this means that if the honest aggregator limits the number of aggregation requests for any label t to ℓ, the protocol offers $(\ell\,\epsilon, \ell\,\delta)$ computational differential privacy (see Theorem 1).

Proof. Without loss of generality, we show that privacy holds for the meter $i = 1$ when the aggregator $j = 1$ is honest,

against a coalition C of all other meters, aggregators and the service provider. Let Π denote a run of the protocol for aggregating readings with label t using weights $\vec{w}$, and define the following abbreviations:

$$\begin{array}{rcl} \vec{r}_C & \stackrel{\text{def}}{=} & \{r_{i,t} \mid i \in [2 \dots N], t \in \mathcal{T}\} \\ \vec{r}_1 & \stackrel{\text{def}}{=} & \{r_{1,t} \mid t \in \mathcal{T}\} \\ \vec{x}_C & \stackrel{\text{def}}{=} & \{x_i \mid i \in [2 \dots N]\} \\ \vec{y}_C & \stackrel{\text{def}}{=} & \{y_j \mid j \in [2 \dots P]\} \\ \vec{u}_C & \stackrel{\text{def}}{=} & \{u_j \mid j \in [2 \dots P]\} \end{array}$$

The view of the coalition C is given by the following procedure:

procedure $\text{VIEW}_\Pi^C(\vec{r}_C, \vec{r}_1)$
  $\text{SETUP}(N, P)$
  $T \leftarrow \text{AGGREGATE}(t, \vec{w})$
  return $(\vec{r}_C, \vec{x}_C, \vec{y}_C, g, g^{x_1}, g^{y_1}, \vec{w}, t, T, \vec{c}, \vec{s}, \vec{u}_C)$

Note that we do not need to include in this view the keys shared between members of the coalition, or between them and honest participants, because they can be computed from $\vec{x}_C, \vec{y}_C, g, g^{x_1}$, and g^{y_1}.

We show that $\text{VIEW}_\Pi^C(\vec{r}_C, \cdot)$ is ϵ-SIM-CDP using the simulator F given in Fig. 6. We assume that the honest meter successfully responds to the service provider request to get its blinded reading; otherwise privacy holds trivially.

We first prove that for any value $\vec{r}$ of the readings, the probability that an adversary $\mathcal{B}$ distinguishes $\text{VIEW}_\Pi^C(\vec{r}_C, \vec{r}_1)$ from $F(\vec{r}_C, \vec{r}_1)$ is the same as the advantage of an adversary $\mathcal{A}$ in solving the HDH problem on $\mathcal{G}$ and $H(t, \cdot)$. The adversary $\mathcal{A}$ that we construct (parameterized by $\vec{r}$) is the following:

procedure $\mathcal{A}(g, gx, gy, h)$
  $u \xleftarrow{\$} \text{GEOM}(\alpha)$
  $c' \leftarrow (r_{1,t} + h) \mod n$
  $s' \leftarrow (w_1\, h + u) \mod n$
  $F.\text{SETUP}(N, P, g, gx, gy)$
  $T \leftarrow F.\text{AGGREGATE}(t, \vec{w}, c', s')$
  return $\mathcal{B}(\vec{r}_C, \vec{x}_C, \vec{y}_C, g, gx, gy, \vec{w}, t, T, \vec{c}, \vec{s}, \vec{u}_C)$

It is easy to see that when $\mathcal{A}$ is given as input a proper HDH tuple of the form $(g, g^a, g^b, H(t, g^{ab}))$, the *view* that it provides to $\mathcal{B}$ follows the same distribution as a view generated by $\text{VIEW}_\Pi^C(\vec{r}_C, \vec{r}_1)$. Indeed, g is a uniformly chosen generator of $\mathcal{G}$, g^a plays the role of g^{x_1}, g^b the role of g^{y_1}, and g^{ab} the role of $k_{1,1}$; the procedure $F.\text{SETUP}$ computes the keys of the rest of the meters and aggregators, and all other shared keys as in the protocol. The values of $c_{1,t}$ and s_1 computed by the procedure $F.\text{AGGREGATE}$ from $h = g^{ab}$ are also distributed as in the protocol. Intuitively, computations requiring the private data of the meter $i = 1$ and the aggregator $j = 1$ (i.e. $\vec{r}_1, x_1, y_1$) have been factored out from procedures SETUP and AGGREGATE.

When $\mathcal{A}$ is given instead a tuple of the form (g, g^a, g^b, h) where h is a value uniformly chosen from $\mathbb{Z}_n$, the *view* that it provides to the distinguisher $\mathcal{B}$ is distributed as a simulated view computed by $F(\vec{r}_C, \vec{r}_1)$. This not as straightforward to see; but it boils down to showing that for any values of w_1, $r_{1,t}$, and u, the program

$$c' \xleftarrow{\$} \mathbb{Z}_n;\ s' \leftarrow (w_1\,(c' - r_{1,t}) + u) \mod n$$

generates the same joint distribution over c', s' as the program

$$h \xleftarrow{\$} \mathbb{Z}_n;\ c' \leftarrow (r_{1,t} + h) \mod n;\ s' \leftarrow (w_1\, h + u) \mod n$$

This can be shown by an application of the non-approximate rule for random assignments of pRHL [4], which requires exhibiting a bijection f on $\mathbb{Z}_n$ such that for all $c' \in \mathbb{Z}_n$,

$$\begin{array}{rcl} c' & \equiv & r_{1,t} + f(c') \mod n \\ w_1\,(c' - r_{1,t}) + u) & \equiv & w_1\,(f(c') + u) \mod n \end{array}$$

The bijection $f(c') \stackrel{\text{def}}{=} (c' - r_{1,t}) \mod n$ satisfies these conditions.

To conclude the proof, we show that the simulator $F(\vec{r}_C, \cdot)$ in Fig. 6 is ϵ-differentially private. Formally, we consider that two inputs $\vec{r}_1$, $\vec{r'}_1$ are *adjacent* when

$$|r_{1,t} - r'_{1,t}| \leq \frac{\Delta}{\max_i w_i} \tag{2}$$

Since $F(\vec{r}_C, \vec{r}_1)$ (resp. $F(\vec{r}_C, \vec{r'}_1)$) uses directly the value $r_{1,t}$ (resp. $r'_{1,t}$) only to compute s', it suffices to show that $s' = (w_1\,(c' - r_{1,t}) + \text{GEOM}(\exp \epsilon/\Delta)) \mod n$ provides ϵ-differential privacy. We prove this applying the parameterized apRHL judgment corresponding to Lemma 1. This requires to show that $|w_1\,(c' - r_{1,t}) - w_1\,(c' - r'_{1,t})| \leq \Delta$, which follows from (2) above. □

C. Discussion

In general, the aggregation protocol above guarantees the following: If more than one meter and at least a single aggregator are honest, the adversary learns nothing about the partition of the resulting weighted sum between the different meter readings. In other words, any set of readings from honest meters, leading to the same weighted sum, are equally likely. In all cases, even when a single aggregator is honest, the protocol guarantees the privacy of honest readings. When more than one aggregator can be assumed to be honest, the utility of the protocol can be increased while maintaining the same privacy guarantee, by reducing the amount of noise each aggregator adds (i.e. increasing the scale parameter α).

The aggregators are oblivious to the readings or even the resulting weighted sum: they do not even need to see the blinded readings to execute their part of the protocol. Besides providing differential privacy, any subset of the aggregators can audit the privacy compliance of aggregation queries, perform access control, authorize only some aggregation queries, or limit the rate of aggregation queries to preserve privacy.

We observe that traditional encryption and signature schemes can be used to protect the confidentiality and integrity of all communications from third parties—a concern that is orthogonal to our presentation of the protocol.

V. Related Work

A. Relational verification of probabilistic programs

Pierce [38] classifies language-based approaches to differential privacy into three categories: runtime enforcement (PINQ, AIRAVAT), static enforcement (*Fuzz*, *DFuzz*), and

```
1:  procedure F(r⃗_C, r⃗_1)
2:      g ←$ G*;  x_1, y_1 ←$ Z_q;  c' ←$ Z_n
3:      u ←$ GEOM(α)
4:      s' ← (w_1 (c' − r_{1,t}) + u)  mod n
5:      F.SETUP(N, P, g, g^{x_1}, g^{y_1})
6:      T ← F.AGGREGATE(t, w⃗, c', s')
7:      return (r⃗_C, x⃗_C, y⃗_C, g, g^{x_1}, g^{y_1}, w⃗, t, T, c⃗, s⃗, u⃗_C)
8:  procedure F.SETUP(N, P, g, gx, gy)
9:      for j ← 2 to P do
10:         y_j ← Z_q;  k_{1,j} ← g x^{y_j}
11:     end for
12:     for i ← 2 to N do
13:         x_i ← Z_q;  k_{i,1} ← g y^{x_i}
14:         for j ← 2 to P do
15:             k_{i,j} ← g^{x_i y_j}
16:         end for
17:     end for
18: procedure F.AGGREGATE(t, w⃗, c', s')
19:     c_{1,t} ← (c' + Σ_{j=2}^{P} H(t, k_{i,j}))  mod n
20:     M ← {1}
21:     for i ← 2 to N do
22:         c_{i,t} ← GETREADING(i, t)
23:         if c_{i,t} ≠ ⊥ then M ← M ∪ {i}
24:     end for
25:     s_1 ← (s' + Σ_{i∈M} w_i H(t, k_{i,1}))  mod n
26:     for j ← 2 to P do
27:         s_j ← GETOPENING(j, t, w⃗, M)
28:     end for
29:     T ← Σ_{i∈M} w_i · c_{i,t} − Σ_{j=1}^{P} s_j  mod n
30:     return T
```

Fig. 6. Simulator used in the proof of Theorem 2. Procedures GETREADING and GETOPENING are as in Figure 5.

verification-based enforcement (CertiPriv). We briefly review these systems.

PINQ [32] is a platform that enforces differential privacy of C# programs that embed SQL-like queries. Enforcement is performed at runtime: for each query, the platform computes its sensitivity and subtracts the amount of privacy leaked by the query from a privacy budget. The soundness of PINQ rests on the composition theorems for differential privacy, and on the correctness of the Laplacian and exponential mechanisms. AIRAVAT [41] operates on similar principles, but is based on the MapReduce programming model.

Static approaches, originating from Reed and Pierce [39] and further developed in Gaboardi et al. [19], are based on linear type systems. These type systems exploit two essential properties of differential privacy: it can be achieved by output perturbation with noise scaled to sensitivity, and it is closed under sequential and parallel composition. Linear type systems can be automated and can handle effectively many interesting examples; D'Antoni et al. [14] use Z3 to solve constraints arising during type checking. However, these type systems cannot validate programs that achieve differential privacy without relying on standard building blocks for output perturbation, or programs that achieve computational differential privacy.

The verification-based approach followed in this paper is based on CertiPriv [4], which extends CertiCrypt [2] with support for reasoning about apRHL. CertiPriv is strictly more expressive than the framework presented in this paper because it allows users to short-circuit apRHL and reason directly about the semantics of programs. However, proofs must be built using the Coq proof assistant, which is time-consuming and requires expertise. Similar approaches that support relational program verification include Relational Hoare Type Theory [36] and Relational F* [6]; however, the former does not support probabilistic computations whereas the latter does not support approximate relational judgments as apRHL does. Recently, Chaudhuri et al. [11] have developed an automated method to prove sensitivity of imperative programs; their method can be used to choose adequate parameters for sanitization mechanisms in order to achieve differential privacy.

Our verification-based approach shares many similarities with the method developed by Tschantz et al. [45] for reasoning about differential privacy of interactive systems. In particular, their definition of differential privacy is also based on a lifting operator that closely resembles ours and their unwinding-based verification method can be understood as an abstract, language-independent, equivalent of apRHL. However, their method is currently limited to reason about $(\epsilon, 0)$-differential privacy.

B. Differentially private computation of Hamming distance

The problem of constructing a privacy-friendly two-party protocol for computing the Hamming distance between two bit-vectors is closely related to the problem of computing their inner product, and of securely computing set-intersection cardinality. Indeed, the inner product $\langle a, b\rangle$ of two bit-vectors $x, y \in \{0,1\}^n$ can be computed as $h(x, 0^n) + h(y, 0^n) - h(x, y)$, and thus any two-party protocol for approximating $h(x, y)$ can be converted into a protocol for approximating $\langle x, y\rangle$ (and *vice versa*) if, in addition, both parties release differentially private approximations of the Hamming weight of their inputs. Such a protocol would have applications in the context of data-mining, where many problems of practical interest can be reduced to computing the inner product of bit-vectors (see e.g. [18, 22, 46]).

Alaggan et al. [1] describe a computationally differentially private two-party protocol for computing the inner product of two bit-vectors based on the *shared scalar product* protocol of Goethals et al. [22] and a threshold homomorphic cryptosystem. As observed above, this protocol can be turned into a protocol for computing Hamming distance by having both parties release a differentially private estimate of the Hamming weight of their inputs. However, this comes at the expense of a

degradation in either privacy or utility due to the added noise, and in efficiency, due to the use of a threshold cryptosystem, which requires the participants to actively cooperate at the end of the protocol to decrypt the estimate of the inner product.

McGregor et al. [31] prove that for every $\zeta > 0$, any protocol for computing the Hamming distance between two ℓ-bit vectors that is (ϵ, δ)-differentially private must incur with probability $1 - \zeta$ an additive error $\Omega\left(\zeta\sqrt{\ell}\exp(-\epsilon)/\log(\ell)\right)$ when $\delta = o(1/\ell)$. In contrast, the (ϵ, δ)-SIM-CDP protocol we presented in Section II can be proven to incur only an additive error $1 + 1/\epsilon$ with probability at most $\exp(-1)/2$, independently of the length of the vectors. This implies a strong separation result between computational and information-theoretic differential privacy in the two-party setting; in contrast, no similar result is in sight for the client/server setting.

C. Smart meter aggregation & billing

Shi et al. [42] propose constructions for time-series aggregation of distributed data. Their techniques allow for compact ciphertexts, are aggregator oblivious, and achieve a flavor of computational differential privacy (in the random oracle model) provided a fraction of the participants is uncompromised. Yet, they lack robustness, and no result can be extracted in case a reading is missing. More recently, Chan et al. [10] achieve some degree of robustness at the cost of requiring more noise when readings are missing, and incurring a logarithmic to linear communication cost depending on the scheme variant.

Kursawe [29] first proposed the use of secret sharing for aggregation. The protocol requires communication complexity that is linear in the number of aggregation authorities, but also public key operations for every aggregation. In subsequent work [30] a number of schemes were proposed with public verifiability of the aggregation operation in mind, as well as protocols for communication efficient aggregation. None of these protocols is robust, and a single missing reading renders aggregation impossible.

A line of work starting with Garcia and Jacobs [20] uses homomorphic cryptosystems to support aggregation. Jawurek and Kerschbaum [28] extend this to support robustness: they achieve the same level of noise as our scheme, but at the computation and networking cost of one public key operation per reading, as well as several operations for every aggregation.

Our scheme achieves communication efficient aggregation, which means that messages are small, comparable in size to the readings themselves. No public key operations are required during aggregation, and an initial key distribution setup phase can be amortized over many aggregations. The threat model we use is similar to Jawurek and Kerschbaum [28] in that it requires a set of authorities to facilitate aggregation. These authorities are oblivious to the aggregation result. Meters proposed in Kursawe et al. [30] can be seeded with keys from our authorities, and perform our protocol without modification. In fact at a very high level the robust protocol we present here has the same roots as Kursawe [29] in secret sharing, but uses a one-off Diffie-Hellman exchange in the setup phase, to avoid subsequent computation and communication costs.

Privacy-friendly billing has received some previous attention: Jawurek et al. [27] as well as Rial and Danezis [40] propose schemes based on homomorphic commitments and zero-knowledge proofs to support a variety of billing policies, and in particular a linear policy very efficiently. Molina-Markham et al. [35] show how to make such protocols very efficient for the meters in terms of computation and communication. The downside of the above protocols is the requirement that, in addition to the meter, a user device is needed to compute bills. Our proposed protocol can be applied without the customer interacting with the service provider, but at the cost of requiring a small number of aggregation authorities.

VI. Conclusion

We have extended EasyCrypt with support for reasoning about user-defined sub-probability distributions, and for proving approximate relational equivalences of probabilistic programs. We have used this extension to verify differentially private algorithms and protocols, including two multi-party protocols that achieve differential privacy only against computationally bounded adversaries.

Besides the examples presented in Sections II and IV, we have verified several information-theoretic differentially private algorithms, including all examples from [5]. One of particular interest is the approximation algorithm for solving the Minimum Vertex Cover problem in graphs of Gupta et al. [25]. This case study is significant in three respects: it achieves differential privacy without resorting to standard mechanisms, it works on structured rather than numeric data, and its proof necessitates a more complex rule for loops [5, Figure 5], which we have also mechanized in EasyCrypt. CertiPriv [5] has been recently extended to reason about an asymmetric version of apRHL [7], which leads to a marginally tighter differential privacy bound for this example; it could be worthwhile to further extend EasyCrypt to support the same logic.

The extension of EasyCrypt that we presented in this paper, and in general the logic apRHL, can also be used for other purposes, including reasoning about the statistical distance between probability distributions generated by randomized programs; this has applications to information-theoretic notions of security in cryptography that are stated in terms of statistical rather than computational indistinguishability, such as statistical zero-knowledge.

There are several directions for further work. These include extending the protocols and proofs of Sections II and IV to the malicious model, and enhancing EasyCrypt with support for reasoning about continuous probability distributions and real-world implementations based on floating-point arithmetic. Another challenge is to provide tool-support for reasoning about the utility of differentially private algorithms, which would require more advanced methods to derive bounds about the probability of events w.r.t. distributions generated by programs (e.g. Chebyshev's inequality).

References

[1] M. Alaggan, S. Gambs, and A.-M. Kermarrec, "Private similarity computation in distributed systems: From cryptography to differential privacy," in *15th International Conference on Principles of Distributed Systems, OPODIS 2011*, ser. Lecture Notes in Computer Science, vol. 7109. Heidelberg: Springer, 2011, pp. 357–377.

[2] G. Barthe, B. Grégoire, and S. Zanella-Béguelin, "Formal certification of code-based cryptographic proofs," in *36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2009*. New York: ACM, 2009, pp. 90–101.

[3] G. Barthe, B. Grégoire, S. Heraud, and S. Zanella-Béguelin, "Computer-aided security proofs for the working cryptographer," in *Advances in Cryptology – CRYPTO 2011*, ser. Lecture Notes in Computer Science, vol. 6841. Heidelberg: Springer, 2011, pp. 71–90.

[4] G. Barthe, B. Grégoire, S. Heraud, F. Olmedo, and S. Zanella-Béguelin, "Verified indifferentiable hashing into elliptic curves," in *1st Conference on Principles of Security and Trust – POST 2012*, ser. Lecture Notes in Computer Science, vol. 7215. Heidelberg: Springer, 2012.

[5] G. Barthe, B. Köpf, F. Olmedo, and S. Zanella-Béguelin, "Probabilistic relational reasoning for differential privacy," in *39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2012*. New York: ACM, 2012, pp. 97–110.

[6] G. Barthe, C. Fournet, B. Grégoire, P.-Y. Strub, N. Swamy, and S. Zanella-Béguelin, "Probabilistic relational verification for cryptographic implementations," Unpublished manuscript, 2013.

[7] G. Barthe, B. Köepf, F. Olmedo, and S. Zanella-Béguelin, "Probabilistic relational reasoning for differential privacy," *TOPLAS*, 2013.

[8] A. Beimel, K. Nissim, and E. Omri, "Distributed private data analysis: Simultaneously solving how and what," in *Advances in Cryptology – CRYPTO 2008*, ser. Lecture Notes in Computer Science, vol. 5157. Heidelberg: Springer, 2008, pp. 451–468.

[9] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-dnf formulas on ciphertexts," in *2nd Theory of Cryptography Conference, TCC 2005*, ser. Lecture Notes in Computer Science, vol. 3378. Heidelberg: Springer, 2005, pp. 325–341.

[10] T.-H. H. Chan, E. Shi, and D. Song, "Privacy-preserving stream aggregation with fault tolerance," in *16th International Conference on Financial Cryptography and Data Security, FC 2012*, ser. Lecture Notes in Computer Science, vol. 7397. Heidelberg: Springer, 2012, pp. 200–214.

[11] S. Chaudhuri, S. Gulwani, R. Lublinerman, and S. Navidpour, "Proving programs robust," in *19th ACM SIGSOFT Symposium on the Foundations of Software Engineering and 13rd European Software Engineering Conference, ESEC/FSE 2011*. New York: ACM, 2011, pp. 102–112.

[12] M. R. Clarkson and F. B. Schneider, "Hyperproperties," *Journal of Computer Security*, vol. 18, no. 6, pp. 1157–1210, 2010.

[13] G. Danezis, M. Kohlweiss, and A. Rial, "Differentially private billing with rebates," in *13th International Conference on Information Hiding, IH 2011*, ser. Lecture Notes in Computer Science, vol. 6958. Heidelberg: Springer, 2011, pp. 148–162.

[14] P. L. D'Antoni, E. J. G. Arias, A. Haeberlen, and B. C. Pierce, "Type-based sensitivity analysis," Unpublished manuscript, 2013.

[15] Y. Deng and W. Du, "Logical, metric, and algorithmic characterisations of probabilistic bisimulation," Carnegie Mellon University, Tech. Rep. CMU-CS-11-110, March 2011.

[16] Y. Dodis, A. López-Alt, I. Mironov, and S. P. Vadhan, "Differential privacy with imperfect randomness," in *Advances in Cryptology – CRYPTO 2012*, ser. Lecture Notes in Computer Science, vol. 7417. Heidelberg: Springer, 2012, pp. 497–516.

[17] C. Dwork, "Differential privacy," in *33rd International Colloquium on Automata, Languages and Programming, ICALP 2006*, ser. Lecture Notes in Computer Science, vol. 4052. Heidelberg: Springer, 2006, pp. 1–12.

[18] C. Dwork and K. Nissim, "Privacy-preserving datamining on vertically partitioned databases," in *Advances in Cryptology – CRYPTO 2004*, ser. Lecture Notes in Computer Science, vol. 3152. Heidelberg: Springer, 2004, pp. 528–544.

[19] M. Gaboardi, A. Haeberlen, J. Hsu, A. Narayan, and B. C. Pierce, "Linear dependent types for differential privacy," in *40th ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages, POPL 2013*. New York: ACM, 2013, pp. 357–370.

[20] F. D. Garcia and B. Jacobs, "Privacy-friendly energy-metering via homomorphic encryption," in *6th International Workshop on Security and Trust Management, STM 2010*, ser. Lecture Notes in Computer Science, vol. 6710. Heidelberg: Springer, 2010, pp. 226–238.

[21] A. Ghosh, T. Roughgarden, and M. Sundararajan, "Universally utility-maximizing privacy mechanisms," *SIAM J. Comput.*, vol. 41, no. 6, pp. 1673–1693, 2012.

[22] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikäinen, "On private scalar product computation for privacy-preserving data mining," in *7th International Conference on Information Security and Cryptology – ICISC 2004*, ser. Lecture Notes in Computer Science, vol. 3506. Heidelberg: Springer, 2005, pp. 104–120.

[23] S. Goldwasser and M. Sipser, "Private coins versus public coins in interactive proof systems," in *18th Annual ACM Symposium on Theory of Computing, STOC 1986*. New Yotk: ACM, 1986, pp. 59–68.

[24] A. Groce, J. Katz, and A. Yerukhimovich, "Limits of computational differential privacy in the client/server setting," in *8th Theory of Cryptography Conference, TCC*

2011, ser. Lecture Notes in Computer Science, vol. 6597. Heidelberg: Springer, 2011, pp. 417–431.

[25] A. Gupta, K. Ligett, F. McSherry, A. Roth, and K. Talwar, "Differentially private combinatorial optimization," in *21st Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010*. SIAM, 2010, pp. 1106–1125.

[26] J. Hurd, "An LCF-style interface between HOL and first-order logic," in *18th International Conference on Automated Deduction, CADE-18*, ser. Lecture Notes in Artificial Intelligence, vol. 2392. Heidelberg: Springer, 2002, pp. 134–138.

[27] M. Jawurek, M. Johns, and F. Kerschbaum, "Plug-in privacy for smart metering billing," in *11h International Symposium on Privacy Enhancing Technologies, PETS 2011*, ser. Lecture Notes in Computer Science, vol. 6794. Heidelberg: Springer, 2011, pp. 192–210.

[28] M. Jawurek and F. Kerschbaum, "Fault-tolerant privacy-preserving statistics," in *12th International Symposium on Privacy Enhancing Technologies, PETS 2012*, ser. Lecture Notes in Computer Science, vol. 7384. Heidelberg: Springer, 2012, pp. 221–238.

[29] K. Kursawe, "Some ideas on privacy preserving meter aggregation," Radboud Universiteit Nijmegen, Tech. Rep. ICIS–R11002, January 2011.

[30] K. Kursawe, G. Danezis, and M. Kohlweiss, "Privacy-friendly aggregation for the smart-grid," in *11h International Symposium on Privacy Enhancing Technologies, PETS 2011*, ser. Lecture Notes in Computer Science, vol. 6794. Heidelberg: Springer, 2011, pp. 175–191.

[31] A. McGregor, I. Mironov, T. Pitassi, O. Reingold, K. Talwar, and S. P. Vadhan, "The limits of two-party differential privacy," *Electronic Colloquium on Computational Complexity (ECCC)*, vol. 18, p. 106, 2011.

[32] F. D. McSherry, "Privacy integrated queries: an extensible platform for privacy-preserving data analysis," in *35th SIGMOD International Conference on Management of Data, SIGMOD 2009*. New York: ACM, 2009, pp. 19–30.

[33] J. Meng and L. C. Paulson, "Translating higher-order clauses to first-order clauses," *J. Autom. Reasoning*, vol. 40, no. 1, pp. 35–60, 2008.

[34] I. Mironov, O. Pandey, O. Reingold, and S. Vadhan, "Computational differential privacy," in *Advances in Cryptology – CRYPTO 2009*, ser. Lecture Notes in Computer Science, vol. 5677. Heidelberg: Springer, 2009, pp. 126–142.

[35] A. Molina-Markham, G. Danezis, K. Fu, P. Shenoy, and D. Irwin, "Designing privacy-preserving smart meters with low-cost microcontrollers," in *16th International Conference on Financial Cryptography and Data Security, FC 2012*, ser. Lecture Notes in Computer Science, vol. 7397. Heidelberg: Springer, 2012, pp. 239–253.

[36] A. Nanevski, A. Banerjee, and D. Garg, "Verification of information flow and access control policies with dependent types," in *32nd IEEE Symposium on Security and Privacy, S&P 2011*. IEEE Computer Society, 2011, pp. 165–179.

[37] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology – EUROCRYPT 1999*, ser. Lecture Notes in Computer Science, vol. 1592. Heidelberg: Springer, 1999, pp. 223–238.

[38] B. C. Pierce, "Differential privacy in the programming languages community," 2012, invited tutorial at DIMACS Workshop on Recent Work on Differential Privacy across Computer Science.

[39] J. Reed and B. C. Pierce, "Distance makes the types grow stronger: a calculus for differential privacy," in *15th ACM SIGPLAN International Conference on Functional programming, ICFP 2010*. New York: ACM, 2010, pp. 157–168.

[40] A. Rial and G. Danezis, "Privacy-preserving smart metering," in *10th Annual ACM Workshop on Privacy in the Electronic Society, WPES 2011*. New York: ACM, 2011, pp. 49–60.

[41] I. Roy, S. T. V. Setty, A. Kilzer, V. Shmatikov, and E. Witchel, "Airavat: security and privacy for MapReduce," in *7th USENIX Conference on Networked Systems Design and Implementation, NSDI 2010*. Berkeley: USENIX Association, 2010, pp. 297–312.

[42] E. Shi, T.-H. H. Chan, E. G. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data," in *18th Annual Network and Distributed System Security Symposium, NDSS 2011*. The Internet Society, 2011.

[43] V. Shoup, "Using hash functions as a hedge against chosen ciphertext attack," in *Advances in Cryptology – EUROCRYPT 2000*, ser. Lecture Notes in Computer Science, vol. 1807. Heidelberg: Springer, 2000, pp. 275–288.

[44] T. Terauchi and A. Aiken, "Secure information flow as a safety problem," in *12th International Symposium on Static Analysis, SAS 2005*, ser. Lecture Notes in Computer Science, vol. 3672. Heidelberg: Springer, 2005, pp. 352–367.

[45] M. C. Tschantz, D. Kaynar, and A. Datta, "Formal verification of differential privacy for interactive systems," *Electronic Notes in Theoretical Computer Science*, vol. 276, pp. 61–79, 2011.

[46] J. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in *8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2002*. New York: ACM, 2002, pp. 639–644.

APPENDIX
EASYCRYPT PROOF OF THE GEOMETRIC MECHANISM

```
op Geom : real → int distr.

op p(α:real, k:int) = α^(- |k|) * ((α - 1) / (α + 1)).

axiom Geom_def : ∀ (α:real, k:int), 1 < α ⇒ mu_x(Geom(α), k) = p(α, k).

axiom mult_div_le : ∀ (x, y, z:real), 0 < z ⇒ x * (z ^ (-1)) ≤ y ⇒ x ≤ y * z.

axiom pow_monotonic : ∀ (x, a, b:real), 1 ≤ x ⇒ a ≤ b ⇒ x ^ a ≤ x ^ b.

axiom pow_pos : ∀ (x, y:real), 0 < x ⇒ 0 < x ^ y.

lemma pow_plus : ∀ (x, y, z:real), 0 < x ⇒ x ^ (y + z) = (x ^ y) * (x ^ z).

lemma pow_mult : ∀ (x, y, z:real), 0 < x ⇒ x ^ (y * z) = (x ^ y) ^ z.

lemma pow_inv : ∀ (x, y, z:real), 0 < x ⇒ (x ^ y) ^ (-1) = x ^ (- y).

lemma pow_ge_one : ∀ (x:real, a:int), 1 < x ⇒ 0 ≤ a ⇒ 1 ≤ x ^ a.

lemma abs_minus_sym : ∀ (a, b:int), |a - b| = |b - a|.

lemma pow_α_triangular :
  ∀ (α:real, k, a, b:int), 1 < α ⇒ |b - a| ≤ k ⇒ α ^ (|b| - |a|) ≤ α ^ k.

lemma pow_α_le :
  ∀ (α:real, k, a, b:int), 1 < α ⇒ |b - a| ≤ k ⇒ α^(- |a|) * α^(|b|) ≤ α ^ k.

lemma α_le : ∀ (α:real, k, a, b:int), 1 < α ⇒ |b - a| ≤ k ⇒
  let z = (α - 1) / (α + 1) in α^(- |a|) * z ≤ ((α ^ k) * α^(- |b|)) * z.

lemma Geom_distance :
  ∀ (α:real, k, a, b:int), 1 < α ⇒ |b - a| ≤ k ⇒ p(α, a) ≤ (α ^ k) * p(α, b).

lemma Geom_α_distance :
  ∀ (α:real, k, a, b: int, μ1, μ2:int distr),
    1 < α ⇒ |b - a| ≤ k ⇒
    (∀ (x:int), p(α, x - b) ≤ (α ^ k) * p(α, x - a)) ∧
    (∀ (y:int), p(α, y - a) ≤ (α ^ k) * p(α, y - b)).

lemma Geom_α_distance_pred :
  ∀ (α:real, k, a, b: int, μ1, μ2:int distr),
    1 < α ⇒ |b - a| ≤ k ⇒
    (∀ (x:int), mu_x(μ1, x) = mu_x(Geom(α), x - a)) ⇒
    (∀ (y:int), mu_x(μ2, y) = mu_x(Geom(α), y - b)) ⇒
    delta(α ^ k, μ1, μ2) = 0 ∧ delta_pred(α ^ k, μ1, μ2, 0).

lemma Geom_support : ∀ (α:real), 1 < α ⇒ ∀ (a:int), support(a, Geom(α)).

spec Geom_spec(α:real, k, a, b:int) :
  x = sample(Geom(α)) ~ y = sample(Geom(α)) : 1 < α ∧ |b - a| ≤ k =[α^k; 0]⟹ a + x = b + y
as lemma (x → a + x | x - a) (z → b + z | z - b).
```