



Differential Privacy meets Verifiable Computation: Achieving Strong Privacy and Integrity Guarantees

Downloaded from: <https://research.chalmers.se>, 2026-04-14 00:27 UTC

Citation for the original published paper (version of record):

Tsaloli, G., Mitrokotsa, A. (2019). Differential Privacy meets Verifiable Computation: Achieving Strong Privacy and Integrity Guarantees. ICETE 2019 - Proceedings of the 16th International Joint Conference on e-Business and Telecommunications, 2: 425-430. <http://dx.doi.org/10.5220/0007919404250430>

N.B. When citing this work, cite the original published paper.

Differential Privacy meets Verifiable Computation: Achieving Strong Privacy and Integrity Guarantees

Georgia Tsaloli and Aikaterini Mitrokotsa

Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg, Sweden

Keywords: Verifiable Computation, Differential Privacy, Privacy-preservation.

Abstract: Often service providers need to outsource computations on sensitive datasets and subsequently publish statistical results over a population of users. In this setting, service providers want guarantees about the correctness of the computations, while individuals want guarantees that their sensitive information will remain private. Encryption mechanisms are not sufficient to avoid any leakage of information, since querying a database about individuals or requesting summary statistics can lead to leakage of information. *Differential privacy* addresses the paradox of learning nothing about an individual, while learning useful information about a population. *Verifiable computation* addresses the challenge of proving the correctness of computations. Although verifiable computation and differential privacy are important tools in this context, their interconnection has received limited attention. In this paper, we address the following question: *How can we design a protocol that provides both differential privacy and verifiable computation guarantees for outsourced computations?* We formally define the notion of *verifiable differentially private computation* (VDPC) and what are the minimal requirements needed to achieve VDPC. Furthermore, we propose a protocol that provides verifiable differentially private computation guarantees and discuss its security and privacy properties.

1 INTRODUCTION

Recent progress in ubiquitous computing has allowed data to be collected by multiple heterogeneous devices, stored and processed by remote, untrusted servers (cloud) and, subsequently, used by third parties. Although this cloud-assisted environment is very attractive and has important advantages, it is accompanied by serious security and privacy concerns for all parties involved. Individuals whose data are stored in cloud servers want privacy guarantees on their data. Cloud servers are untrusted and thus, often need to perform computations on encoded data, while service providers want integrity guarantees about the correctness of the outsourced computations.

Encryption techniques can guarantee the confidentiality of information. Nevertheless, querying a database about individuals or requesting summary statistics can lead to leakage of information. In fact, by receiving the response to different statistical queries, someone may draw conclusions about individual users and could be exploited to perform differencing and reconstruction attacks (Dwork and Roth, 2014) against a database. *Differential privacy* (Dwork et al., 2006) addresses the paradox of learning nothing

about an individual, while learning useful information about a population. *Verifiable delegation of computation* (Gennaro et al., 2010) provides reliable methods to verify the correctness of outsourced computations. Although differential privacy and verifiable computation have received significant attention separately, their interconnection remains largely unexplored.

In this paper, we investigate the problem of *verifiable differentially private computation* (VDPC), where we want not only integrity guarantees on the outsourced results but also differential privacy guarantees about them. This problem mainly involves the following parties: (i) a *curator* who has access to a private database, (ii) an *analyst* who wants to perform some computations on the private dataset and then publish the computed results, and (iii) one or more *readers (verifiers)* who would like to verify the correctness of the performed computations as well as that the computed results are differentially private.

(Narayan et al., 2015) introduced the concept of verifiable differential privacy and proposed a system (VerDP) that can be employed to provide verifiable differential privacy guarantees by employing VFuzz (Narayan et al., 2015), a query language for computations and Pantry (Braun et al., 2013), a sys-

tem for proof-based verifiable computation. Although VerDP satisfies important privacy and integrity guarantees, it is vulnerable to malicious analysts, who may intentionally leak private information.

In this paper, we formalize the notion of verifiable differentially private computation and provide its formal definition as well as the requirements that need to be satisfied in order to achieve it. Furthermore, we propose a detailed protocol that can be employed to provide verifiable differential privacy guarantees and we discuss the security and privacy properties it satisfies and why it resolves the identified weaknesses in VerDP (Narayan et al., 2015).

Paper Organization: The paper is organized as follows. In Section 2, we describe the related work. In Section 3, we provide an overview of the general principles of the main building blocks – verifiable computation and differential privacy – where *verifiable differentially private computation* (VDPC) is based on. In Section 4, we describe the minimal requirements needed to achieve verifiable differentially private computations, provide the formal definition of VDPC and describe our proposed public VDPC protocol. In Section 5, we discuss the security and privacy properties that our protocol satisfies and how it resolves open issues in the state-of-the-art. Finally, Section 6 concludes the paper.

2 RELATED WORK

Verifiable computation (VC) is inherently connected to the problem of verifiable differential privacy. VC schemes based on fully homomorphic encryption (Gennaro et al., 2010; Chung et al., 2010) naturally offer input-output privacy because the inputs and correspondingly the outputs are encrypted. However, they do not provide public verifiability in case multiple readers want to verify the correctness of outsourced computations. VC schemes based on homomorphic authenticators have some restrictions with respect to the supported function class, nevertheless, some offer input privacy. For instance, (Fiore et al., 2014) have proposed a VC scheme for multivariate polynomials of degree 2, that offers input privacy, yet provides no public verifiability. (Parno et al., 2012) showed how to construct a VC scheme with public verifiability from any attribute-based encryption (ABE) scheme. If the underlying ABE scheme is attribute-hiding, the function’s input is encoded in the attribute and, thus, the VC scheme is input private. However, existing VC schemes largely ignore the problem of VDPC.

(Narayan et al., 2015) proposed the VerDP system that provides important privacy and integrity guarantees based mainly on the Pantry (Braun et al., 2013) VC scheme and the VFuzz (Narayan et al., 2015) language that can be employed to guarantee the use of differentially private functions. However, in the VerDP system, the analyst has access to the dataset and thus, jeopardizes the privacy of the data. We address this challenge by making sure that analysts may access only encoded data. Equally important is the random noise used to guarantee differentially private computations. We make sure that this noise is generated based on a randomness u on which the reader and the curator mutually agree and therefore, no one can control the random noise term on his own.

3 PRELIMINARIES

In this section, we briefly overview the general principles and recall the definitions of the main building blocks – verifiable computation and differential privacy – where verifiable differentially private computation is based on.

3.1 Verifiable Computation

A *verifiable computation* (VC) scheme can be employed when a client wants to outsource the computation of a function f to an external untrusted server. The client should be able to verify the correctness of the returned result (by the server), with less computation cost than performing the computation itself. Below we provide the definition of a verifiable computation scheme (VC) as introduced by (Parno et al., 2012) and (Fiore and Gennaro, 2012).

Definition 1 (Verifiable Computation (Parno et al., 2012; Fiore and Gennaro, 2012)). *A verifiable computation scheme VC is a 4-tuple of PPT algorithms (KeyGen, ProbGen, Compute, Verify) which are defined as follows:*

- $\text{KeyGen}(f, 1^\lambda) \rightarrow (\text{PK}_f, \text{EK}_f)$: *This randomized key generation algorithm takes the security parameter λ and the function f as inputs and outputs a public key PK_f to be used for encoding the function input and a public evaluation key EK_f to be used for evaluating the function f .*
- $\text{ProbGen}_{\text{PK}_f}(\mathbf{m}) \rightarrow (\sigma_{\mathbf{m}}, \text{VK}_{\mathbf{m}})$: *This randomized problem generation algorithm takes as input the key PK_f and the function input \mathbf{m} and outputs $\sigma_{\mathbf{m}}$, which is the encoded version of the function input, and $\text{VK}_{\mathbf{m}}$, which is the verification key used in the verification algorithm.*

- $\text{Compute}_{\text{EK}_f}(\sigma_m) \rightarrow \sigma_y$: Given the key EK_f and the encoded input σ_m , this deterministic algorithm encodes the function value $y = f(m)$ and returns the result, σ_y .
- $\text{Verify}_{\text{VK}_m}(\sigma_y) \rightarrow y \cup \perp$: The deterministic verification algorithm takes as inputs the verification key VK_m and σ_y and returns either the function value $y = f(m)$, if σ_y represents a valid evaluation of the function f on m , or \perp , if otherwise.

We note here that a publicly verifiable computation scheme is a VC scheme, where the verification key VK_m is public, allowing in this way the public verification of the outsourced computation.

3.2 Differential Privacy

Differential privacy is a framework which can be employed in order to achieve strong privacy guarantees. Intuitively, via a differentially private mechanism, it is impossible to distinguish two neighboring databases (i.e., databases that differ in a single record) storing private data when the same query is performed in both, regardless of the adversary's prior information. For instance, differential privacy implies that it is impossible, given the result of the queries, to determine if an individual's record is included in the databases or not. (Dwork et al., 2006) formally introduced differential privacy as follows:

Definition 2 (ϵ -Differential Privacy (Dwork et al., 2006)). A randomized mechanism $\mathcal{M} : \mathcal{D} \rightarrow \mathcal{R}$ with domain \mathcal{D} and range \mathcal{R} satisfies ϵ -differential privacy if for any two adjacent inputs $d, d' \in \mathcal{D}$ and for any subset of outputs $S \subseteq \mathcal{R}$ it holds that:

$$\Pr[\mathcal{M}(d) \in S] \leq e^\epsilon \Pr[\mathcal{M}(d') \in S]$$

In this notation, ϵ denotes the privacy parameter which is defined in positive real numbers. A privacy promise can be considered strong when its ϵ is close to zero.

Let us consider a deterministic real-valued function $f : \mathcal{D} \rightarrow \mathbb{R}$. A common method used to approximate the function f with a differentially private mechanism is by employing additive noise incorporated in f 's sensitivity S_f . The sensitivity S_f of a function f is defined as the maximum possible distance between the replies to queries (i.e., $|f(d) - f(d')|$) addressed to any of the two neighboring databases (d and d'). By intuition, larger sensitivity demands a stronger countermeasure. A commonly employed differentially private mechanism, the Laplace noise mechanism, is defined as follows:

$$\mathcal{M}(d) \triangleq f(d) + \text{Laplace}(\lambda)$$

We should note here that except of providing strong privacy guarantees, it is rather important to consider the utility of the data when employing a differentially private mechanism. Indeed, each query may lead to leakage of information about the data stored in the queried database. By increasing the noise, we can provide high privacy guarantees but this could also lead to low utility of the used information. Thus, achieving a good trade-off between utility and privacy is what needs to be achieved by an efficient differentially private mechanism.

4 VERIFIABLE DIFFERENTIALLY PRIVATE COMPUTATION

We consider the problem of *verifiable differentially private computations* where a curator (e.g., National Institute of Health (NIH)) has access to a private database m (e.g., medical information). An analyst (e.g., a researcher) wants to perform some computations on the private dataset and publish the results. One or more readers would like to verify not only the correctness of the computations but also that the computations were performed with a differentially private algorithm. To guarantee differential privacy, the analyst has to execute a differentially private query f on a dataset m , resulting in an output y , without having access to m . The reader then should be convinced that y is the correct result.

In differential privacy (DP), we assume that the algorithm is randomized which we can model by adding a uniformly distributed input u to a deterministic function $g(f, \epsilon, u, m)$. More precisely, the reader should be able to verify that:

$$y = g(f, \epsilon, u, m) = f(m) + \tau(f, \epsilon, u)$$

where $u \sim \mathcal{U}$ (i.e., \mathcal{U} denotes the Uniform distribution) and ϵ determines the level of differential privacy provided. In the right-hand side, we have decomposed the differentially private (DP) computation into two parts: $f(m)$, which produces the exact query response, and τ , which transforms the uniform noise appropriately in order to achieve the required differential privacy guarantees. For any given query denoted by f , the noise described by τ is fixed (e.g., Laplace or Gaussian noise). All in all, the reader needs to verify that all the following *requirements* hold:

1. g is an ϵ -DP computation for the query f . The way this is done is outside the scope of this paper and could be achieved using the VFuzz language (Narayan et al., 2015).

2. u is from a high-quality pseudo-random source.
3. y is correctly computed from g , which is evaluated on the dataset m with the randomness u .

We first propose the definition of a *publicly verifiable differentially private computation scheme* VDPC_{Pub} which is based on Parno et al.'s (Parno et al., 2012) publicly verifiable computation VC definition. Subsequently, we describe a concrete protocol that can be employed in order to provide public verifiability (i.e., all readers are able to verify the correctness of the computations) as well as the privacy and the integrity guarantees required in this setting.

Definition 3. (Publicly Verifiable Differentially Private Computation) *A publicly verifiable differentially private computation scheme VDPC_{Pub} is a 5-tuple of PPT algorithms (Gen_DP, KeyGen, ProbGen, Compute, Verify) which are defined as follows:*

- $\text{Gen_DP}(f, \epsilon, u) \rightarrow g(f, \epsilon, u, \cdot)$: Given the function f , the randomness u and the value ϵ , this algorithm computes $g(f, \epsilon, u, \cdot) = f(\cdot) + \tau(f, \epsilon, u)$ with τ denoting the used noise.
- $\text{KeyGen}(g, 1^\lambda) \rightarrow (\text{PK}_g, \text{EK}_g)$: Given the security parameter λ and the function g , the algorithm outputs a public key PK_g to be used for encoding the function input and a public evaluation key EK_g to be used for evaluating the function g .
- $\text{ProbGen}_{\text{PK}_g}(u || m) \rightarrow (\sigma_{u || m}, \text{VK}_{u || m})$: This algorithm takes as input the public key PK_g and the function input m as well as the randomness u and outputs $\sigma_{u || m}$, which is the encoded version of the function input, and $\text{VK}_{u || m}$, which is used for the verification.
- $\text{Compute}_{\text{EK}_g}(\sigma_{u || m}) \rightarrow \sigma_y$: Given the evaluation key EK_g and the encoded input $\sigma_{u || m}$, this algorithm encodes the function value $y = g(f, \epsilon, u, m)$ and returns the result, σ_y .
- $\text{Verify}_{\text{VK}_{u || m}}(\sigma_y) \rightarrow y \cup \perp$: Given the verification key $\text{VK}_{u || m}$ and the encoded result σ_y , this algorithm returns either the function value $y = g(f, \epsilon, u, m)$, if σ_y represents a valid evaluation of the function g on m , or \perp , if otherwise.

We assume that the reader has verified that g is an ϵ -DP computation for the query f when u is uniformly distributed, using for instance the VFuzz language (Narayan et al., 2015). We suggest the employment of a zero-knowledge protocol to prove that indeed u comes from a high quality pseudo-random source and a VDPC_{Pub} scheme to prove that y is the correct computation of g in the underlying data m . We should note here that, in our definition, we focus on publicly verifiable computation. We can adjust it

though in cases where a single verifier (reader) is required and thus, a verifiable computation scheme with secret verification could be incorporated accordingly.

4.1 A Publicly Verifiable Differentially Private Protocol

We consider three main parties in our protocol: (i) the curator, (ii) the analyst, and (iii) one or more readers (verifiers). The reader should obtain the desired result y , a proof about its correctness as well as differential privacy guarantees about the performed computation.

The curator has access to some data generated by multiple data subjects (e.g., patients). As soon as the curator collects the data, he chooses the randomness u that will be used to generate the stochasticity in the ϵ -DP function g . The randomness u is not revealed to any other party but it is committed so that the curator cannot change it. On the other hand, the curator should convince the reader that the randomness he selected is indistinguishable from a value selected from a uniform distribution and then use this value u to compute the ϵ -DP function $g(f, \epsilon, u, m)$. The analyst, even though he does not have any access to the dataset m but some encoded information $\sigma_{u || m}$ about the dataset and the randomness u respectively, he is still able to generate the encoded value σ_y of $y = g(f, \epsilon, u, m)$. We should note that the computation of the encoded value σ_y is of high complexity and thus, the analyst is the one who will perform this operation instead of the curator. Eventually, the reader, having the ϵ -DP function value y and the public verification key $\text{VK}_{u || m}$, is able to verify that what he has received from the analyst is correct.

The workflow of our protocol (depicted in Fig. 1) is separated into two phases. The first phase focuses on the *differential privacy* requirements, while the second phase is related to the *verifiable computation* of the function g .

We need to stress here that in the first phase, the main issue we want to address is to make sure that the random noise used to compute the differentially private function g from the query function f is generated with a good source of randomness u . This corresponds to the second requirement that needs to be satisfied in order to achieve verifiable differentially private computations as listed in Section 4. To meet the above requirement, we incorporate in our protocol, a zero-knowledge protocol that allows us to guarantee that the random noise is generated based on a randomness u (on which the reader and the curator mutually agree). We describe the two phases in detail:

- PHASE 1: We are interested in computing an ϵ -DP function g that depends on the query f . In

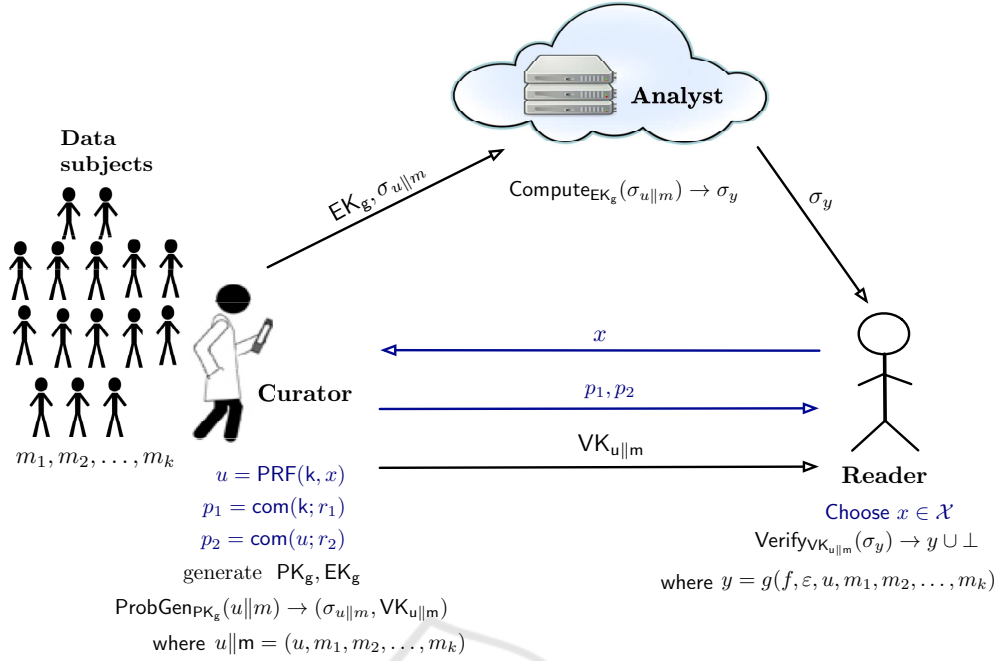


Figure 1: The workflow of achieving public verifiable differential privacy.

our model, we do not want to place all the trust in the curator. Thus, the curator should agree with the reader on the randomness u that will be used in computing the (Laplace) noise transformation. However, only the curator should know the value of u . To achieve this, a protocol is run between the reader and the curator. More precisely, (i) the reader chooses an input $x \in \mathcal{X}$, where \mathcal{X} is the input space and sends it to the curator, and (ii) the latter picks a secret key $k \in \mathcal{K}$ of a pseudorandom function PRF, where \mathcal{K} is the key space of a PRF, computes the randomness $u = \text{PRF}(k, x)$ and the commitments $p_1 = \text{com}(k; r_1)$ and $p_2 = \text{com}(u; r_2)$, where r_1, r_2 denote randomness. After sending p_1, p_2 to the reader, the curator runs an interactive zero-knowledge protocol to convince the reader that the statement (x, p_1, p_2) is true. The statement (x, p_1, p_2) is true, when the following holds: $\exists k, r_1, u, r_2$ such that:

$$p_1 = \text{com}(k; r_1) \wedge p_2 = \text{com}(u; r_2) \wedge u = \text{PRF}(k, x).$$

In this way, the reader and the curator agree on the randomness u while its value is available only to the curator.

- PHASE 2: Now, let us see how we may employ the *publicly verifiable differentially private computation* scheme VDPC_{Pub} in our system.
 - (a) The curator collects the data from the data subjects (e.g., patients) into a dataset $m = \{m_1, \dots, m_k\}$.

- (b) The curator runs $\text{KeyGen}(g, 1^\lambda) \rightarrow (\text{PK}_g, \text{EK}_g)$ to get a short public key PK_g that will be used for input delegation and the public evaluation key EK_g which will be used from the analyst to evaluate the function g . The curator sends EK_g to the analyst.
- (c) $\text{ProbGen}_{\text{PK}_g}(u||m) \rightarrow (\sigma_{u||m}, \text{VK}_{u||m})$ is run by the curator to get a public encoded value of u and m , called $\sigma_{u||m}$, and a public value $\text{VK}_{u||m}$ which will be used for the verification. The curator sends $\sigma_{u||m}$ to the analyst and publishes $\text{VK}_{u||m}$. We note that the curator can directly send $\sigma_{u||m}$ to the analyst after computing it (i.e., no need to store the value). Note also that, neither the analyst nor the reader can retrieve any information about the dataset m from $\sigma_{u||m}$.
- (d) The analyst runs the $\text{Compute}_{\text{EK}_g}(\sigma_{u||m}) \rightarrow \sigma_y$ algorithm using the evaluation key EK_g and $\sigma_{u||m}$ in order to obtain the encoded version of the function $g(f, \varepsilon, u, m_1, \dots, m_k) = y$. After performing the evaluation he publishes σ_y .
- (e) Now, the reader uses $\text{VK}_{u||m}$ and σ_y to run the algorithm $\text{Verify}_{\text{VK}_{u||m}}(\sigma_y) \rightarrow y \cup \perp$ which indicates whether σ_y represents the valid ε -DP output of f or not.

5 DISCUSSION

By employing our proposed protocol, the readers (verifiers) can get strong guarantees about the integrity and correctness of the computed result. To provide strong privacy guarantees, we need to make sure that the random noise used to compute the differentially private function g is generated with a good source of randomness. To address this, we incorporate in the VDPC_{Pub} protocol, a zero-knowledge protocol. The latter allows us to guarantee that the random noise is generated based on a randomness u on which the reader and the curator mutually agree and thus, no one can control the random noise term on his own. Of course the DP level achieved via our solution depends on the differentially private mechanism employed and can be tuned based on the selected ϵ parameter in order to achieve a good balance of utility and privacy. Curious readers that want to recover private data are not able to get any additional information than the result of the computation. Moreover, our protocol is secure against malicious analysts, since analysts have access to encoded information about the dataset m but do not obtain neither the dataset itself nor the randomness u . Our security requirement is that the computations performed by the analysts should be correct and they should not be able to get any additional information besides what is required to perform the computation. This is achieved assuming that the employed (publicly) VC scheme is secure. Contrary to our approach, in the VerDP system the analyst can access the dataset and thus, poses a privacy risk. In our approach this is addressed by allowing the analysts to access only an encoded form of the data.

6 CONCLUSION

Often, when receiving an output, we want to confirm that it is computed correctly and that it does not leak any sensitive information. Thus, we require not only confidentiality on the used data but also differential privacy guarantees on the computed result, while at the same time, we want to be able to verify its correctness. In this paper, we formally define the notion of verifiable differentially private computations (VDPC) and we present a protocol (VDPC_{Pub}) that can be employed to compute the value of a differentially private function g (i.e., denoting the ϵ -DP computation for a function f), as well as to check the correctness of the computation.

VDPC is an important security notion that has received limited attention in the literature. We believe

that verifiable differentially private computation can have important impact in a broad range of application scenarios in the cloud-assisted setting that require strong privacy and integrity guarantees. However, a rather challenging point for any verifiable differentially private computation protocol is for the end user to be able to verify that a VDPC scheme is used, and how we can make sure that he understands the use of the epsilon parameters. More precisely, we are interested in further investigating how a link can be made between the formal verification process and a human verification of the whole process, especially at the user level.

ACKNOWLEDGEMENTS

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

REFERENCES

- Braun, B., Feldman, A. J., Ren, Z., Setty, S., Blumberg, A. J., and Walfish, M. (2013). Verifying computations with state. In *Proceedings of SOSP*, pages 341–357.
- Chung, K.-M., Kalai, Y. T., and Vadhan, S. P. (2010). Improved delegation of computation using fully homomorphic encryption. In *CRYPTO*, volume 6223, pages 483–501.
- Dwork, C., McSherry, F., Nissim, K., and Smith, A. D. (2006). Calibrating noise to sensitivity in private data analysis. In *Proceedings of TCC*, pages 265–284.
- Dwork, C. and Roth, A. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407.
- Fiore, D. and Gennaro, R. (2012). Publicly verifiable delegation of large polynomials and matrix computations, with applications. In *Proceedings of CCS*, pages 501–512.
- Fiore, D., Gennaro, R., and Pastro, V. (2014). Efficiently verifiable computation on encrypted data. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 844–855.
- Gennaro, R., Gentry, C., and Parno, B. (2010). Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Advances in Cryptology—CRYPTO 2010*, pages 465–482.
- Narayan, A., Feldman, A., Papadimitriou, A., and Haeberlen, A. (2015). Verifiable differential privacy. In *Proceedings of the Tenth European Conference on Computer Systems*, page 28.
- Parno, B., Raykova, M., and Vaikuntanathan, V. (2012). How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *Proceedings of TCC*, pages 422–439.