Private and Perennial Distributed Computation[*]

Shlomi Dolev[1] Juan Garay[2] Niv Gilboa[3] Vladimir Kolesnikov[4]
[1]Dept. of Computer Science, Ben-Gurion University
[2]AT&T Labs – Research
[3]Dept. of Computer Science, Ben-Gurion University and Deutsche Telekom Laboratories
[4]Bell Laboratories
dolev@cs.bgu.ac.il garay@research.att.com niv.gilboa@gmail.com
kolesnikov@research.bell-labs.com

Abstract: In this paper we consider the problem of n agents (servers) wishing to perform a given computation on behalf of a user, on common inputs and in a privacy preserving manner, in the sense that even if the entire memory contents of some of them are exposed, no information is revealed about the state of the computation, and where there is no *a priori* bound on the number of inputs. The problem has received ample attention recently in several domains, including cloud computing as well as *swarm computing* and Unmanned Aerial Vehicles (UAV) that collaborate in a common mission, and schemes have been proposed that achieve this notion of privacy for arbitrary computations, at the expense of one round of communication per input among the n agents.

In this work we show how to *avoid communication altogether* during the course of the computation, with the trade-off of computing a smaller class of functions, namely, those carried out by finite-state automata. Our scheme, which is based on a novel combination of secret-sharing techniques and the Krohn-Rhodes decomposition of finite state automata, achieves the above goal in an information-theoretically secure manner, and, furthermore, does not require randomness during its execution.

Keywords: Secure outsourcing of computation; Information-theoretic security; Finite-state automata; Krohn-Rhodes decomposition.

[*]A preliminary version of this paper appeared in *Proc. Second Symposium on Innovations in Computer Science* (ICS '11) under the title "Secret Sharing Krohn-Rhodes: Private and Perennial Distributed Computation."

1 Setting and Goal

Hiding the state of a computation performed by n agents from exposure of some of the agents' states has recently received attention from several application domains, in particular those where the sequence of inputs to the computation is potentially *unbounded*. Such is the case of cloud computing and *ad hoc* and "swarm" computing (see [4, 6, 18] and references therein). In this work we make progress in this area, by showing the feasibility of non-interactive private distributed computation on such unbounded input sequences.

Recently, Dolev *et al.* [4] presented schemes that support the type of "infinite" private computation mentioned above by implementing a distributed version of a so-called *strongly oblivious* universal Turing machine (TM)[1]. However, this is done at the expense of one round of communication per received input amongst the participants. In contrast, in this work we show *how to avoid communication altogether* during the course of the computation, with the trade-off of computing a smaller class of functions.

Specifically, we consider a distributed computation setting in which a party, which we refer to as the *user* (sometimes as the *dealer*), has a finite state automaton (FSA) $\mathcal{A}$ which accepts an (*a priori* unbounded) stream of inputs $x_1, x_2, \ldots$ received from an external source. We are interested in situations in which the user cannot perform the required computation, but instead delegates the responsibility to agents (servers) $P_1, \ldots, P_n$. Each of the agents receives all the inputs destined to $\mathcal{A}$ during its execution. The agents execute their distributed implementation of $\mathcal{A}$ (without communication!) and, at a given signal from the user, terminate the execution, compute the current state of $\mathcal{A}$, and return it as output.

Furthermore, there is an additional entity, called *the adversary* Adv, who is able to adaptively "corrupt" a subset of the agents (i.e., inspect their internal state) during the execution phase, up to a threshold[2] $t < n$, and our objective is to ensure that the agents' computation is as private as possible. We do not aim to maintain the privacy of the automaton $\mathcal{A}$; however, we wish to protect the secrecy of the state of $\mathcal{A}$ and the inputs' history. We note that Adv may have external information about the computation, such as partial inputs or length of the input sequence, state information, etc. This auxiliary information, together with the knowledge of $\mathcal{A}$, may exclude the protection of certain configurations, or even fully determine $\mathcal{A}$'s state. We stress that this cannot be avoided in any implementation, and we do not consider this an insecurity. Thus, our goal is to prevent the leakage or derivation by Adv of any knowledge from seeing the execution traces which Adv did not already possess.

2 Our Approach

We present a scheme that achieves the above goal in an information-theoretically secure manner (i.e., there are no bounds imposed on Adv's computational power), and does not require randomness during the execution of $\mathcal{A}$. Our scheme is based on a novel combination of secret-sharing techniques [17] and the Krohn-Rhodes decomposition of finite automata [12, 13]. Informally, Krohn-Rhodes theory states that any finite state automaton can be emulated by a combination (*cascade product*) of permutation automata and flip-flop automata. (A permutation automaton is any automaton such that each of its possible input symbols induces a permutation of the automaton's states.) The computation complexity per each received input symbol, and the storage complexity required by our scheme are a function only of (the decomposition of) $\mathcal{A}$, and not of the number of symbols processed. A trade-off for this is that, depending on $\mathcal{A}$, the number of components of its Krohn-Rhodes decomposition might be exponential in its number of states. We note, however, that for many interesting and relevant automata, there is a small Krohn-Rhodes decomposition. In the full paper we present examples of such automata families with a small Krohn-Rhodes representation.

For ease of exposition, in this presentation we concentrate on the case of passive corruptions—i.e., Adv is considered "honest but curious." How-

[1]An *oblivious* TM moves the tape heads through the same sequence of cells; a *strongly oblivious* TM is a Turing machine in which the movement of tape heads is a function only of the cell indices that the heads point to. Not every oblivious TM is strongly oblivious, since the movement of the tape heads may be a function of time, not only of space.

[2]We note that more general access structures may be naturally employed with our constructions.

ever, since our construction does not require communication among parties at the time when corruptions are allowed, it can be readily strengthened to handle active corruptions by employing secret-sharing schemes (e.g., *unverified* secret sharing [5, 16]) that are robust against disruptive behavior, and suitable for our scenario.

As noted earlier, swarms and sensor networks (e.g., [4, 6]) are areas that can potentially benefit from our scheme. Another area of great current interest where user privacy is critical is that of outsourcing computation and storage to the "cloud." Yet, a big challenge in making the shift in computing to the cloud infrastructure is finding a way to ensure the privacy of the computation. One possible approach is for the users to run the program distributively on several computing clouds in such a way that even if some of them collude and exchange information they still will not be able to learn the program and/or the data they use for the computation. Furthermore, the type of computation may be of a "never-ending" nature, such as ongoing sequence of tasks performed by an operating system; the output of a given task or state of an on-going system can then be revealed by receiving information from all or a sufficient number of cloud suppliers participating in the computation—very much like a terminal client is used to interface with remote computers. Our work addresses this scenario.

3 Related Work

Reactive k-secret sharing with no communication among agents participating in a swarm has been suggested in [6]. Several solutions that are able to withstand limited memory corruptions were presented, some of them based on the linearity of secret sharing, supporting addition and multiplication by constants. The last solution is based on maintenance of the vector of possible states by each agent, masking the actual state of the swarm (defined to be the one with a majority of copies) by redundant states (with fewer copies). In general, two states maintained by an agent may yield the same next state when a certain input is received, thus redundancy of states may be eliminated over time. Randomization is used in [6] in order to cope with such a convergence, randomly choosing a state for the extra copies in the vector when two or more states become equal. In contrast, in this work we show that it is possible to solve the problem of convergence to the same state in a deterministic way using Krohn-Rhodes decomposition. Furthermore, the scheme in this work is information-theoretically secure.

As mentioned above, the authors recently presented schemes that support the same type of *perennial* private computation considered here by implementing a universal Turing machine privately, with one round of communication per transition [4]. In this work we show how to avoid communication completely during the course of the actual execution, at the expense of computing a smaller class of functions.

The type of private computation we consider is also related to the problem of (information-theoretic, or unconditional) secure multi-party computation (MPC) [1, 3]. We perform a detailed comparison below.

Unbounded-input private computation vis-à-vis MPC. Recall that in secure multi-party computation, n parties ("players"), some of which might be corrupted, are to compute an n-ary (public) function on their inputs, in such a way that no information is revealed about them beyond what is revealed by the function's output. At a high level, we similarly aim in our context to ensure the correctness and privacy of the distributed computation. However, as we now argue, our setting is significantly different from that of MPC, and MPC solutions cannot be directly applied here.

Firstly, MPC aims to solve a different problem, that of protecting the players' individual inputs from Adv, who can corrupt some of them, learn their input and observe the communication they receive. In contrast, in our problem the inputs are common to all the players (but not *a priori* known to Adv, or revealed in case of corruption), and the goal is to protect the state of, as well as the inputs to the computation. (Therefore, we cannot in particular treat the common input as public information, and the shares received from the dealer as MPC input.)

Of course, an adequate representation (circuit-based, for example) of the MPC computation would be able to evaluate $\mathcal{A}$, with respect to a subset of corrupted players, and at least for the basic MPC setting, where there is a single (tuple of secret) input(s) out of which an output (tuple) is produced. But then comes our main feature, of multiple, possibly unbounded number of input symbols. This is reminiscent of secure *reactive* systems (e.g., [15]), where the computation is not limited to "one shot" as above, but instead processes inputs "in blocks" throughout several rounds of interaction. However, because all MPC solutions (and definitions) are explicitly tied to the length of the input, being able to handle unbounded number of inputs without communication does not seem immediate. This is what our Krohn-Rhodes-based approach achieves, at the expense of solving a narrower problem.

Looking at the relationship with MPC from another perspective, we note that it is the *combination* of our requirements of non-interactivity during the input-processing phase, information-theoretic security, and computation on inputs of unbounded length, that precludes the use of known MPC techniques. That is, with any of the three requirements removed, known techniques would allow stronger results to be achieved.

Indeed, we have discussed above the possibility of solutions in the setting where the inputs are bounded. Alternatively, if we only required computational secrecy, then the players could use fully homomorphic encryption [9] to maintain under encryption the current state of the computation on unbounded inputs (and carry shares of the scheme's private key). Further, if an *a priori* bound on the input length existed, players could simply encrypt their inputs with any public-key encryption scheme, again keeping shares of the decryption key. Finally, allowing interaction during the input processing phase can effectively bring us to the bounded-input setting, since interaction—and thus share updates using MPC—could occur after a certain fixed number of inputs has been processed.

References

[1] M. Ben-OR, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *STOC*, pages 1–10, 1988.

[2] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively Secure Multi-Party Computation. In *STOC*, pages 639–648, 1996.

[3] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols(extended abstract). In *STOC*, pages 11–19, 1988.

[4] S. Dolev, J. Garay, N. Gilboa, and V. Kolesnikov. Swarming secrets. In *47th Annual Allerton Conference on Communication, Control, and Computing*, 2009.

[5] D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. *J. ACM*, 40:1, pages 17–47, 1993.

[6] S. Dolev, L. Lahiani, and M. Yung. Secret swarm unit reactive k-secret sharing. In *INDOCRYPT*, pages 123–137, 2007.

[7] P. Domosi and C.L. Nehaniv. *Algebraic Theory of Automata Networks (SIAM Monographs on Discrete Mathematics and Applications, 11).* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2004.

[8] S. Eilenberg. *Automata, Languages, and Machines, Vol. B.* Academic Press, London, New York, NY, USA, 1976.

[9] C. Gentry. Fully Homomorphic Encryption Using Ideal Lattices. In *41st ACM Symposium on Theory of Computing (STOC)*, pages 169-178, 2009.

[10] F. Higgins, A. Tomlinson and K. Martin. Survey on Security Challenges for Swarm Robotics. *ICAS 2009*, pages 307–312.

[11] M. Ito, A. Saito, and T. Nishizeki. Secret sharing scheme realizing general access structure. In *IEEE Globecom*, pages 99–102, 1987.

[12] K.R. Krohn and J. L. Rhodes. Algebraic theory of machines, 1962.

[13] K.R. Krohn and J. L. Rhodes. Algebraic theory of machines i: prime decomposition theorems for finite semigroups and machines. *Transactions of the American Mathematical Society*, 116:450–464, 1965.

[14] S. Margolis Complexity of holonomy decomposition. Private communication, February, 2010.

[15] B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In *CCS '00: Proceedings of the 7th ACM conference on Computer and Communications Security*, pages 245–254, 2000.

[16] T. Rabin and M. Ben-Or. Verifiable secret

sharing and multiparty protocols with honest majority. In *STOC*, pages 73–85, 1989.

[17] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

[18] M. Weiser. The Computer for the 21th Century. *Scientific American*, September, 1991.

[19] H.P. Zeiger. Cascade synthesis of finite state machines. *Information and Control*, 10:419–433, 1967. Erratum: Information and Control 11(4): 471 (1967).