Assignment 4

Lynne Diep

October 2017

1 Total space in resolution

Abstract: We show $\Omega(n^2)$ lower bounds on the total space used in resolution refutations of random k-CNFs over n variables, and of the graph pigeonhole principle and the bit pigeonhole principle for n holes. This answers the long-standing open problem of whether there are families of k-CNF formulas of size $O(n)$ requiring total space $\Omega(n^2)$ in resolution, and gives the first truly quadratic lower bounds on total space. The results follow from a more general theorem showing that, for formulas satisfying certain conditions, in every resolution refutation there is a memory configuration containing many clauses of large width.

The violations I found in this abstract is the author's use of formulas. Additionally the word "we" is used in the abstract, and it would be better suited if the abstract were in passive voice. One of the positives of this abstract is its length; the abstracts is short, clear, and self-contained.

Revised Abstract: The purpose is to show $\Omega(n^2)$ lower bounds on the total space used in resolution refutations of random k-CNFs over n variables, and of the graph pigeonhole principle and the bit pigeonhole principle for n holes. This answers the long-standing open problem of whether there are families of k-CNF formulas of size $O(n)$ requiring total space $\Omega(n^2)$ in resolution, and gives the first truly quadratic lower bounds on total space. The results follow from a more general theorem showing that, for formulas satisfying certain conditions, in every resolution refutation there is a memory configuration containing many clauses of large width.
Riviser's note: It is bad practice to have formulas in an abstract, but if the abstract loses precision when eliminating them then there is no choice but to include it. I have highlighted the formulas.

2 Outsourcing Private RAM Computation

Abstract: We construct the first schemes that allow a client to privately outsource arbitrary program executions to a remote server while ensuring that: (I) the client's work is small and essentially independent of the complexity of the computation being outsourced, and (II) the server's work is only proportional to the run-time of the computation on a random access machine (RAM), rather than its potentially much larger circuit size. Furthermore, our solutions are non-interactive and have the structure of reusable garbled RAM programs, addressing an open question of Lu and Ostrovsky (Eurocrypt 2013). We also construct schemes for an augmented variant of the above scenario, where the client can initially outsource a large private and persistent database to the server, and later outsource arbitrary program executions with read/write access to this database.

Our solutions are built from non-reusable garbled RAM in conjunction with new types of reusable garbled circuits that are more efficient than prior solutions but only satisfy weaker security. For the basic setting without a persistent database,we can instantiate the required reusable garbled circuits using indistinguishability obfuscation. For the more complex setting with a persistent database we need stronger notions of obfuscation. Our basic solution also requires the client to perform a one-time preprocessing step to garble

a program at the cost of its RAM run-time, and we can avoid this cost using stronger notions of obfuscation. It remains an open problem to instantiate these new types of reusable garbled circuits under weaker assumptions, possibly avoiding obfuscation altogether.

The violations that I found in this abstract is the authors' use of references in the abstract. Additionally, the author abbreviated "random access machine" which is unnecessary, and the abstract is not in passive form.

Revised Abstract: Construction of the first schemes that allow a client to privately outsource arbitrary program executions to a remote server while ensuring that: (I) the client's work is small and essentially independent of the complexity of the computation being outsourced, and (II) the server's work is only proportional to the run-time of the computation on a random access machine, rather than its potentially much larger circuit size. Furthermore, the solutions are non-interactive and have the structure of reusable garbled RAM programs. Additional construction of schemes for an augmented variant of the above scenario, where the client can initially outsource a large private and persistent database to the server, and later outsource arbitrary program executions with read/write access to this database.

The solutions are built from non-reusable garbled random access machine in conjunction with new types of reusable garbled circuits that are more efficient than prior solutions but only satisfy weaker security. For the basic setting without a persistent database, the required reusable garbled circuits using indistinguishability obfuscation can be represented. For the more complex setting with a persistent database there is a need for stronger notions of obfuscation. The basic solution also requires the client to perform a one-time preprocessing step to garble a program at the cost of its random access machine run-time, and this cost can be avoided by using stronger notions of obfuscation. It remains an open problem to instantiate these new types of reusable garbled circuits under weaker assumptions, possibly avoiding obfuscation altogether.

References

[1] Ilario Bonacina, Nicola Galesi, and Neil Thapen. *Total space in resolution.* Electronic Colloquium on Computational Complexity. Revision 1 of Report No. 38 (2014), 18 pages.

[2] Craig Gentry, Shai Halevi, Mariana Raykova and Daniel Wichs. *Outsourcing Private RAM Computation* Cryptology ePrint Archive, Report 2014/148 (2014). DOI: http://eprint.iacr.org/2014/148