Casey Devet

My work with Professor Ian Goldberg began in the Fall 2011 term, for which I received an NSERC Undergraduate Student Research Award. During this term, Professor Goldberg asked me to implement some new ideas relating to private information retrieval (PIR) in his open source software project Percy++.

Percy++ was originally developed in 2007 to implement the ideas in Professor Goldberg's 2007 paper *Improving the Robustness of Private Information Retrieval*. Suppose a client wishes to retrieve some data from a database, but does not want any of the database servers to know what data the client is retrieving. PIR schemes define how a client can send queries to a set of database servers in such a way that the responses from the servers can be combined to get the wanted data and that none of the servers knows which data the client wanted. Two important questions arise from this: what if a server does not respond, or responds incorrectly, and what if multiple servers collude to discover the wanted data. Professor Goldberg developed a scheme in which a client could be sure that their query would remain private as long as at most t servers collude and the data could still be recovered if at most v servers respond incorrectly. To do this, the PIR scheme would make the queries so that responses formed a Reed-Solomon codeword.

My work was to implement a new Reed-Solomon decoding scheme developed by H. Cohn and N. Heninger in their paper *Ideal forms of Coppersmith's theorem and Guruswami-Sudan list decoding*. This algorithm was theoretically more efficient than the previously used algorithm. After implementing this algorithm, I tested the implementation and found that, as hoped, it was more efficient than the previously used algorithm.

After this work, I also implemented a new decoding scheme based on new research by H. Cohn and N. Heninger in the paper *Approximate common divisors via lattices*, which if given multiple Reed-Solomon codewords could successfully decode codewords with more errors than was previously possible. The advantage of using this decoding scheme in PIR is the data can be recovered when there are more Byzantine servers than previously allowed, but requires that multiple query responses are available. This is very practical since a client will generally make multiple queries. If the number of servers that respond is k, then this scheme has a Byzantine robustness of $v < k - t - 1$. That is, if t is the number of servers that can collude without recovering the query, the data can be recovered if at most $k - t - 1$ servers respond incorrectly. This is better then the previously used algorithms which have Byzantine robustness of at most $v < k - \sqrt{kt}$. In fact, this new Byzantine robustness is the theoretically best possible for a PIR scheme that runs in polynomial time.

I continued my work with Professor Goldberg during the Winter 2012 term as a part-time undergraduate research assistant. During this term Professor Goldberg and I also developed a decoding scheme that used dynamic programming. If we assume that some number of servers are honest and some are Byzantine, then we can perform a decoding algorithm on all other responses and combine the results with our assumptions to get the decoding of all responses. I implemented this scheme in Percy++ and precomputed the best algorithm for each case.

Together with Professor Goldberg and Nadia Heninger, I discuss this work in the paper *Optimally Robust Private Information Retrieval*. In this paper we describe an algorithm that combines all of the Reed-Solomon decoding algorithms described above so that the best algorithm is used in each case and has a Byzantine robustness of $v < k - t - 1$. We conclude two things in this paper: that our algorithm is much faster than those used in previous PIR schemes and that we can make an optimally robust PIR scheme. These results suggest that PIR may be feasible, if not practical, in highly adversarial settings. This paper was recently accepted for presentation at the 21st USENIX Security Symposium, a top venue in the security research community.

My current work involves implementing this algorithm in Percy++ for a software release in the near future. I am also testing the latest algorithm from Cohn and Heninger to see why it fails in a very small number of cases so that we can use this knowledge to improve our algorithm.