Merkle Puzzles Revisited - Finding Matching Elements Between Lists

Bruce Christianson[1] and David Wheeler[2]

[1] Computer Science Department, University of Hertfordshire, Hatfield
[2] Computer Laboratory, University of Cambridge
England, Europe

Abstract. Consider the following problem. A and B each have a N-element set of bit-strings. They wish to find all collisions, in other words to find the common strings of their sets or to establish that there are none. How much data must A and B exchange to do this?

Problems of this type arise in the context of Merkle puzzles, for example where A and B propose to use the collision between two randomly constructed lists to construct a cryptographic key.

Here we give a protocol for finding all the collisions. Provided the number of collisions is small relative to $N/\log_2 N$ the protocol requires on the order of $\log_2 N$ messages and the total amount of data which A and B need exchange is about $4.5N$ bits. The collision set can also be determined in three messages containing a total of at most $9N$ bits provided $N < 2^{1023}$.

Introduction

Suppose that A and B each have an N-element set of fixed-length bit-strings. By first hashing with an appropriate random hash function if necessary, we can assume that the bit-strings appear to be uniformly distributed. A and B arrange their respective strings in lexical order, and form a table. They may use a fixed part of each string rather than the entire string, provided the parts are sufficiently long to distinguish all pairs of strings.

A *collision* is a string which is in both tables. It is assumed in what follows that the number of collisions is small relative to $N/\log_2 N$. It may be known *a priori* that there is exactly one collision, or some other small number, or the precise number may be unknown, and may include zero. A and B wish to find all collisions between their tables, by exchanging only a small amount of data per table entry.

In the next few sections, we discuss the properties of some protocols to do this. It turns out that the amount of data which the partners are required to exchange can be made less than five bits per list element, regardless of N. Following this, we give an application of our solution to cryptography, in the context of a Merkle puzzle.

Basic Protocol

The basic protocol is a recursive construction. Imagine a list containing, in lexical order, all the 2^n possible table entries (not just the actual ones). We divide this list into a number of intervals of nearly equal size. Initially this number is chosen to be $M = N \log_2 e \approx 3N/2$ where N is the number of entries actually in each table. We say that an interval is *occupied* for A if it contains an entry which occurs in A's table, otherwise it is *empty* for A. The number of intervals M is chosen so that the probability of each interval being occupied is about $1/2$ for each participant, and all these probabilities are nearly independent.

Each party informs the other which intervals are occupied and which are empty. This communication requires one bit per interval, and the coding of the information is optimal. On average half the intervals will be empty for each participant. Only intervals which are occupied for both participants survive to the next round.

Eliminating the intervals which A regards as empty obviously eliminates no entries from A's table, although it does eliminate about half the entries from B's table. However the information from B will eliminate roughly half of the intervals which A regards as occupied, as well as half the intervals which A regards as empty, since their selections are uncorrelated. For each participant, on average half the table entries and one quarter of the intervals thus survive to the next round.

The next round begins by dividing each surviving interval in half. This restores the ratio $N/M = \log_2 e$, so that the probability of occupation is still one half, but the size of the problem is now halved.

Message and Bit Counts

The protocol continues for about $\log_2 N$ rounds. The i-th round exchanges two messages (one in each direction), each containing $M2^{1-i}$ bits of information.

However, if B waits to receive A's messages before responding, the length of B's messages can be halved.

The number of messages can also be reduced if A and B take it in turns to send the first message of a round combined with the second message of the previous round giving $\log_2 N$ messages.

In this case the first message (from A) contains M bits, and the i-th message (from A or B according as i is odd or even) contains about $M2^{2-i}$ bits for $1 < i \leq \log_2 N$.

The total bit-length of all messages is thus less than the geometric sum $3M$. (A's total is $M + 2M/3$ bits. B's total is $4M/3$.) This is less than $4.5N$, ie less than five bits per table entry.

In all cases to eliminate chance we send extra bits of the candidate collision strings to verify the collision. This requires at most $\log_2 N$ bits per collision, which is negligible since the number of collisions is assumed small relative to $N/\log_2 N$. Each collision also causes an interval to be occupied throughout the protocol: this adds one bit per collision per round, which is also negligible by hypothesis on the number of collisions.

Effect of Statistical Deviations

At each round, small statistical deviations from the expected ratios occur. Although the relative size of the deviation introduced at each round increases as the protocol proceeds, the effects are not cumulative: deviations from the previous rounds tend to be systematically damped out.

To see this, assume that at the i-th round there are M_i occupied intervals and N_i^A entries in A's table. Let $\lambda_i^A = N_i^A/M_i, \epsilon_i^A = \lambda_i^A - \log_e 2$. To first order, the proportion of occupied intervals for A is $1 - \exp -\lambda_i^A = (1 + \epsilon_i^A)/2$. Neglecting the statistical errors introduced during the i-th round, we therefore have to first order $\epsilon_{i+1}^A = (1 - \log_e 2)\epsilon_i^A \approx \epsilon_i^A/3$ and similarly for B.

Reducing the Number of Rounds

We may wish to restrict the number of rounds. This can be done by pipelining several rounds into the same pair of messages, at the cost of a modest increase in the total number of bits required per initial table entry.

If A places 16 rounds in the first message, then B can produce a second message of almost equal length containing 2^{16} rounds. If the table contains more than 2^{65536} entries, then A can reply with a third message containing $2 \uparrow (2^{16})$ rounds, and so on.

In practice a reasonable trade-off might be for A to place two rounds in the first message, and B to place ten rounds in the second message. Provided $N < 2^{1023}$, a total of three long messages suffices, with a combined total $6M < 9N$ bits. This is twice the minimum value required by the basic protocol. A fourth short message is required if both parties wish to know the locations of the collisions.

Application to a Merkle Puzzle

Suppose that A and B wish to agree a 40-bit shared secret. They publicly agree a prefix c, and then each generates a few (k) million 40-bit random values r_i. Each then forms a table (in lexical order) of the values $h(c|r_i)$, where h is a randomizing hash function.

By the birthday paradox, the tables are likely to contain about k^2 collisions, and these can be found by the participants using the protocol described above, exchanging on the order of $10k$ million bits of data. The value of r corresponding to the first collision is the shared secret, or alternatively the unused bits of the collision. The work required by an attacker to find r is of the order of 10^{12} rather than 10^6.

If no collisions are found then the exchange is completely repeated or else N is increased, until a solution is found. The chance that no collisions are found is about $\exp -k^2$.

Conclusion

We have shown that Merkle-type puzzles can be solved at a total communications cost which is less than five bits per table entry, regardless of the of the number N of table entries, provided that the number of messages exchanged is allowed to increase to $\log_2 N$.

The usual protocols for a Merkle puzzle require one message in each direction, effectively forcing one of the participants to transmit their table to the other, at a communications cost which increases nonlinearly with the number of table entries. (The precise cost under this restriction depends upon the details of the protocol used, but even here a modified form of our protocol requires on the order of $1.5 \log_2 N$ bits per table entry, considerably fewer than Merkle's original algorithm which for a 40 bit key with $\log_2 N \approx 22$ requires A to send about 70 bits per table entry.)

Alternatively, both the bits per table entry and the number of messages can be held to $\deg_2 N$, where $\deg_2 1 = 0, \deg_2 N = 1 + \deg_2 [\log_2 N]$.