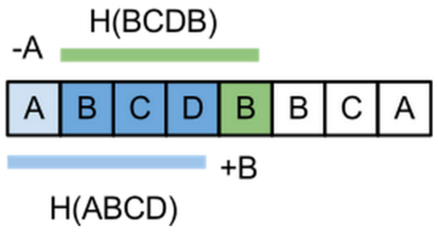
Parallelizing the Rabin-Karp String Matching Algorithm
CS5220 project proposal for iyv2

String matching is a classic computer science problem that is also applicable for security, more specifically in deep packet inspection. It involves searching for specific patterns of characters within an input string. To define the problem more concretely, we begin by defining a finite set, or alphabet, of elements. This alphabet could be {0, 1} if each item is a single bit, or it could be {00, 01, ... , FF} if each item is a full byte. The string and patterns are vectors of this alphabet and the entire input string is searched to find which patterns are contained within all its possible substrings.

The Rabin-Karp Algorithm uses hashing as an efficient way to compare many characters at once. Before the input stream is analyzed, the patterns that the algorithm is searching for much be hashed. Once the patterns have been preprocessed, the algorithm begins to hash the input string using a rolling hash and compares the output hash with each of the hashes of the patterns. If any of the hashes match, then the algorithm will do a character by character comparison between the input and that matched pattern to determine if the matched hashes actually do indicate that the two strings are identical.

To increase the performance of the algorithm, a rolling hash is used. A rolling hash creates a new hash from a previous hash by subtracting the amount that corresponds with the character to be removed and adds the amount that corresponds to the character to be added. Fig. 3 illustrates a brief example. This allows the algorithm to take one pass through the input string without having to recompute the entire hash each time it moves to the next character within the input string.



A Rolling Hash Function Illustration [1]

References
[1] http://www.infoarena.ro/blog/rollinghash?action=download&file=image10.png