Authenticated Data Structures

April 22nd, 2014

Motivation In certain settings, the use do not want to store the data remotely while have the ability to verify that the data is not altered by the storage provider. One simple straw man method is to locally maintain a copy of the whole data set, which will incur linear cost. Note that here usually we store the data to an untrusted server, but it does not mean that the data comes from this server. If the data is generated by the party who store the data, there is no way to verify it here because they can alter the data anyway. In other words, the are storing trusted data on untrusted data storage provider.

Merkle Hash Tree Merkle Hash tree is a crypto primitives that is originally used for signature, but can also be used to verify that data is consistent. Given an array of data, we can build a merkle hash tree from bottom up, in a way that every leaf node is the hash of an element in the array, and that every non-leaf node is the hash of concatenation of its children. Its security can be reduced to the security of hash functions used, that is we cannot forge any hash without breaking the hash function.

The server will store the data as well as the whole merkle hash tree, while the client will only store the root of the hash tree, a.k.a. digest. As we can see, in this case, the user only need to store constant amount of information. When the user want to retrieve something, it will conduct an interactive protocol with the server, which take $O(\log N)$ space directly and constant space if we stream the proof.

Contribution The paper design and implemented a generic way of constructing authenticated data structures, so that people not in crypto area can program a authenticated data structure without really understanding the details.

Shallow Projection In one word, shallow projection serialize the data up to but not pass the the authenticated values. For value that is not authenticated, it is just itself; for values that consists of a hash, it only take the digest.

combination of authenticated and unauthenticated programming the language is designed such that it is possible to programming with authenticated value and unauthenticated value together. Here further work will be done on

how to further speed up the query by hashing less number of levels. It also shows the advantages of generic authenticated data structure, which is easy to maintain and modify.

What if Prover and Verifier use different hash function If they use different hash function, then they will not be able to store the data at the very beginning. If we assume that the server changes the hash function in the middle, then he will be caught because it is of negligible probability that two hash function collide on the same input, where at least one of them is cryptographically secure hash function.