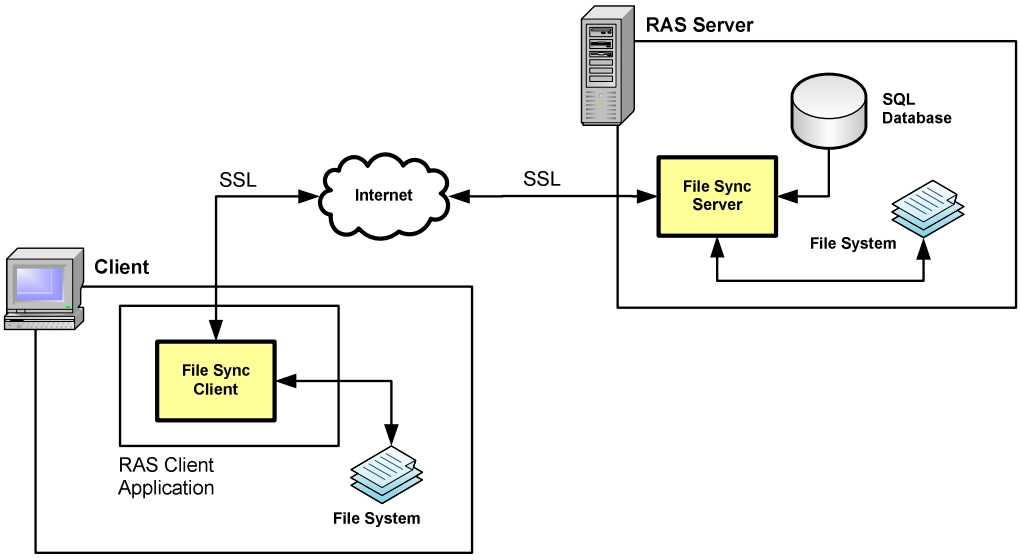
Remote Accounting Solutions, Inc.

Remote Accounting Solutions uses a technique to perform efficient file transfers and directory synchronization within the context of the Remote Accounting Solutions (RAS) services. File transfers utilize the rsync algorithm to effectively transfer only data that differs between the client and server files. To provide secure communications between the Tools and the Server, data will be transmitted via 256 bit SSL with a 2048 bit certificate. The algorithm is PKCS #1 SHA-1 With RSA Encryption.



rsync

Rsync is a fast and extraordinarily versatile file copying tool. It can copy locally, to/from another host over any remote shell, or to/from a remote rsync daemon. It offers
a large number of options that control every aspect of its behavior and permit very flexible specification of the set of files to be copied. It is famous for its delta-transfer algorithm, which reduces the amount of data sent over the network by sending only the differences between the source files and the existing files in the destination. Rsync is widely used for backups and mirroring and as an improved copy command for everyday use.

Rsync finds files that need to be transferred using a "quick check" algorithm (by default) that looks for files that have changed in size or in last-modified time. Any changes in the other preserved attributes (as requested by options) are made on the destination file directly when the quick check indicates that the file's data does not need to be updated.

Rsync uses a checksum method to perform this bit level data transfer. This method creates a short alphanumeric string based on the data it represents. Rsync first checks whether any data has changed by looking at the file size and modification date. If no data has changed, Rsync will not transfer any data, saving time and bandwidth. If files do not match, Rsync uses a checksum method called a rolling checksum on the changed files to see where it has been altered or appended. It will then transfer only the altered or appended data within the file. Rsync can cater for inserted or added data, removed data as well as shifted data, with a minimum transfer overhead.

In real terms, that means more efficient use of your bandwidth and data allowances. As Rsync will only transfer data that has changed and knows when file alterations or movements have occurred, your Internet based backups will take a lot less time when compared other methods such as FTP.

Rsync delivers substantial performance gains. With the ability to check what data is still the same, then append, remove or modify it as necessary to match the local source it can greatly reduce backup overhead

The problem

Imagine you have two files, A and B, and you wish to update B to be the same as A. The obvious method is to copy A onto B.

Now assume that A and B are quite similar, perhaps both derived from the same original file. To really speed things up you would need to take advantage of this similarity. A common method is to send just the differences between A and B down the link and then use this list of differences to reconstruct the file.

The problem is that the normal methods for creating a set of differences between two files rely on being able to read both files. Thus they require that both files are available beforehand at one end of the link. If they are not both available on the same machine, these algorithms cannot be used (once you had copied the file over, you wouldn't need the differences). This is the problem that rsync addresses.

The rsync algorithm efficiently computes which parts of a source file match some part of an existing destination file. These parts need not be sent across the link; all that is needed is a reference to the part of the destination file. Only parts of the source file which are not matched in this way need to be sent verbatim. The receiver can then construct a copy of the source file using the references to parts of the existing destination file and the verbatim material.

Trivially, the data sent to the receiver can be compressed using any of a range of common compression algorithms, for further speed improvements.

The algorithm

The server splits its copy of the file into fixed-size non-overlapping chunks, of size S, and computes two checksums for each chunk: the MD4 hash, and a weaker 'rolling checksum'. It sends these checksums to the Tool.

The Tool computes the rolling checksum for every chunk of size S in its own version of the file, even overlapping chunks. This can be calculated efficiently because of a special property of the rolling checksum: if the rolling checksum of bytes n through n + S − 1 is R, the rolling checksum of bytes n + 1 through n + S can be computed from R, byte n, and byte n + S without having to examine the intervening bytes. Thus, if one had already calculated the rolling checksum of bytes 1–25, one could calculate the rolling checksum of bytes 2–26 solely from the previous checksum, and from bytes 1 and 26.

The rolling checksum used in rsync is based on Mark Adler's adler-32 checksum, which is used in zlib, and which itself is based on Fletcher's checksum.

The Tool then compares its rolling checksums with the set sent by the server to determine if any matches exist. If they do, it verifies the match by computing the hash for the matching block and by comparing it with the hash for that block sent by the server.

The Tool then sends the server those parts of its file that did not match the server's blocks, along with information on where to merge these blocks into the server's version. This makes the copies identical. However, there is a small probability that differences between chunks in the Tool and server are not detected, and thus remains uncorrected. This requires a simultaneous hash collision in MD5 and the rolling checksum. It is possible to generate MD5 collisions, and the rolling checksum is not cryptographically strong, but the chance for this to occur by accident is nevertheless extremely remote. With 128 bits from MD5 plus 32 from the rolling checksum, and assuming maximum entropy in these bits, the possibility of a hash collision with this combined checksum is 2-(128+32) = 2-160. The actual possibility is a few times higher, since good checksums approach maximum output entropy.

If the Tool's and server's versions of the file have many sections in common, the utility needs to transfer relatively little data to synchronize the files.

While the rsync algorithm forms the heart of the rsync application that essentially optimizes transfers between two computers over TCP/IP, the rsync application supports other key features that aid significantly in data transfers or backup. They include compression and decompression of data block by block using zlib at sending and receiving ends, respectively, and support for protocols such as

ssh that enables encrypted transmission of compressed and efficient differential data using rsync algorithm.

Rsync is capable of limiting the bandwidth consumed during a transfer, a useful feature that few other standard file transfer protocols offer.

SSL

SSL protects data submitted over the Internet from being intercepted and viewed by unintended servers and as used by hundreds of thousands of websites in the protection of their online transactions with their customers, SSL is the de-facto industry standard Internet transaction security technology.

SSL is a layered protocol. At each layer, messages may include fields for length, description, and content. SSL takes messages to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, and transmits the result. Received data is decrypted, verified, decompressed, and assembled, then delivered to higher level clients.

An SSL session is stateful. It is the responsibility of the SSL Handshake protocol to coordinate the states of the client and server, thereby allowing the protocol state chines of each to operate consistently, despite the fact that the state is not exactly parallel. Logically the state is represented twice, once as the current operating state, and (during the handshake protocol) again as the pending state. Additionally, separate read and write states are maintained. When the client or server receives a change cipher spec message, it copies the pending read state into the current read state. When the client or server sends a change cipher spec message, it copies the pending write state into the current write state. When the handshake negotiation is complete, the client and server exchange change cipher spec messages, and they then communicate using the newly agreed-upon cipher spec.

Server Certificate

The server sends its certificate to the client. The server certificate contains the server's public key. The client will use this key to authenticate the server and to encrypt the premaster secret.

The client also checks the name of the server in the certificate to verify that it matches the name the client used to connect.

Server Key Exchange

This is a step in which the server creates and sends a temporary key to the client. This key can be used by the client to encrypt the Client Key Exchange message later in the process. The step is only required when the public key

algorithm does not provide the key material necessary to encrypt the Client Key Exchange message, such as when the server's certificate does not contain a public key.

Quickbooks file test

File 1 (original): 7,860,224 bytes
Time (ms): 538607
Time: 00:08:58.6

File 2 (updated with small change): 7,860,224 bytes
Time (ms): 53260
Time: 00:00:53.3
% Diff: 91%

The Sarbanes-Oxley Act requires all publicly traded companies to use SAS 70 Type II Certified datacenters. So you can protect yourself the same way publicly traded companies protect themselves, because all of RAS' servers are located in SAS 70 Type II Certified Datacenters.