5. HMAC and the Pseudorandom Function

The TLS record layer uses a keyed Message Authentication Code (MAC) to protect message integrity. The cipher suites defined in this document use a construction known as HMAC, described in [HMAC], which is based on a hash function. Other cipher suites MAY define their own MAC constructions, if needed.

In addition, a construction is required to do expansion of secrets into blocks of data for the purposes of key generation or validation. This pseudorandom function (PRF) takes as input a secret, a seed, and an identifying label and produces an output of arbitrary length.

In this section, we define one PRF, based on HMAC. This PRF with the SHA-256 hash function is used for all cipher suites defined in this document and in TLS documents published prior to this document when TLS 1.2 is negotiated. New cipher suites MUST explicitly specify a PRF and, in general, SHOULD use the TLS PRF with SHA-256 or a stronger standard hash function.

First, we define a data expansion function, P_hash(secret, data), that uses a single hash function to expand a secret and seed into an arbitrary quantity of output:

```
P_hash(secret, seed) = HMAC_hash(secret, A(1) + seed) +
                       HMAC_hash(secret, A(2) + seed) +
                       HMAC_hash(secret, A(3) + seed) + ...
```

where + indicates concatenation.

A() is defined as:

```
A(0) = seed
A(i) = HMAC_hash(secret, A(i-1))
```

P_hash can be iterated as many times as necessary to produce the required quantity of data. For example, if P_SHA256 is being used to create 80 bytes of data, it will have to be iterated three times (through A(3)), creating 96 bytes of output data; the last 16 bytes of the final iteration will then be discarded, leaving 80 bytes of output data.

TLS's PRF is created by applying P_hash to the secret as:

```
PRF(secret, label, seed) = P_<hash>(secret, label + seed)
```

The label is an ASCII string. It should be included in the exact form it is given without a length byte or trailing null character. For example, the label "slithy toves" would be processed by hashing the following bytes:

```
73 6C 69 74 68 79 20 74 6F 76 65 73
```