```
static OSStatus
SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa, SSLBuffer signedParams,
                                 uint8_t *signature, UInt16 signatureLen)
{
    OSStatus        err;
    SSLBuffer       hashOut, hashCtx, clientRandom, serverRandom;
    uint8_t         hashes[SSL_SHA1_DIGEST_LEN + SSL_MD5_DIGEST_LEN];
    SSLBuffer       signedHashes;
    uint8_t         *dataToSign;
    size_t          dataToSignLen;

    signedHashes.data = 0;
    hashCtx.data = 0;

    clientRandom.data = ctx->clientRandom;
    clientRandom.length = SSL_CLIENT_SRVR_RAND_SIZE;
    serverRandom.data = ctx->serverRandom;
    serverRandom.length = SSL_CLIENT_SRVR_RAND_SIZE;


    if(isRsa) {
        /* skip this if signing with DSA */
        dataToSign = hashes;
        dataToSignLen = SSL_SHA1_DIGEST_LEN + SSL_MD5_DIGEST_LEN;
        hashOut.data = hashes;
        hashOut.length = SSL_MD5_DIGEST_LEN;

        if ((err = ReadyHash(&SSLHashMD5, &hashCtx)) != 0)
            goto fail;
        if ((err = SSLHashMD5.update(&hashCtx, &clientRandom)) != 0)
            goto fail;
        if ((err = SSLHashMD5.update(&hashCtx, &serverRandom)) != 0)
            goto fail;
        if ((err = SSLHashMD5.update(&hashCtx, &signedParams)) != 0)
            goto fail;
        if ((err = SSLHashMD5.final(&hashCtx, &hashOut)) != 0)
            goto fail;
    }
    else {
        /* DSA, ECDSA - just use the SHA1 hash */
        dataToSign = &hashes[SSL_MD5_DIGEST_LEN];
        dataToSignLen = SSL_SHA1_DIGEST_LEN;
    }

    hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
    hashOut.length = SSL_SHA1_DIGEST_LEN;
    if ((err = SSLFreeBuffer(&hashCtx)) != 0)
        goto fail;

    if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
        goto fail;
```

```
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
        goto fail;
        goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;

    err = sslRawVerify(ctx,
                       ctx->peerPubKey,
                       dataToSign,              /* plaintext */
                       dataToSignLen,           /* plaintext length */
                       signature,
                       signatureLen);
    if(err) {
        sslErrorLog("SSLDecodeSignedServerKeyExchange: sslRawVerify "
                    "returned %d\n", (int)err);
        goto fail;
    }

fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;

}
```