$X = 128$

$Y = \sout{256}\ 384$

Assumptions:
1. Only the server has the RSA private key.
2. RSA will not be broken (you have to have the priv. key to decrypt).
3. None of the hash function can be reversed.

$$H1(ps, rc, rs) = MD5(ps + SHA('A' + ps + rc + rs)) + MD5(ps + SHA('BB' + ps + rc + rs)) + MD5(ps + SHA('CCC' + ps + rc + rs))$$

$$H2(s, rc, rs) = HMAC\_MD5(key=s, message=(rc + rs + pad2)) + HMAC\_SHA(\text{''}\langle same \rangle\text{''}) + HMAC\_WHIRLPOOL(\text{''}\langle same \rangle\text{''})$$

$$H3(s, rc, rs) = HMAC\_MD5(key=s, message=(rs + rc + pad3)) + HMAC\_SHA(\text{''}\langle same \rangle\text{''}) + HMAC\_WHIRLPOOL(\text{''}\langle same \rangle\text{''})$$

$$H4(s, rc, rs) = PBKDF2\_HMAC\_WHIRLPOOL(password=s, salt=salt4, h4\_iters, aes\_key\_len)$$

$$H5(s, rc, rs) = PBKDF2\_HMAC\_WHIRLPOOL(password=s, salt=salt5, h5\_iters, aes\_key\_len)$$

---

pad2 = "<rand stuff>"

pad3 = "<different rand stuff>"

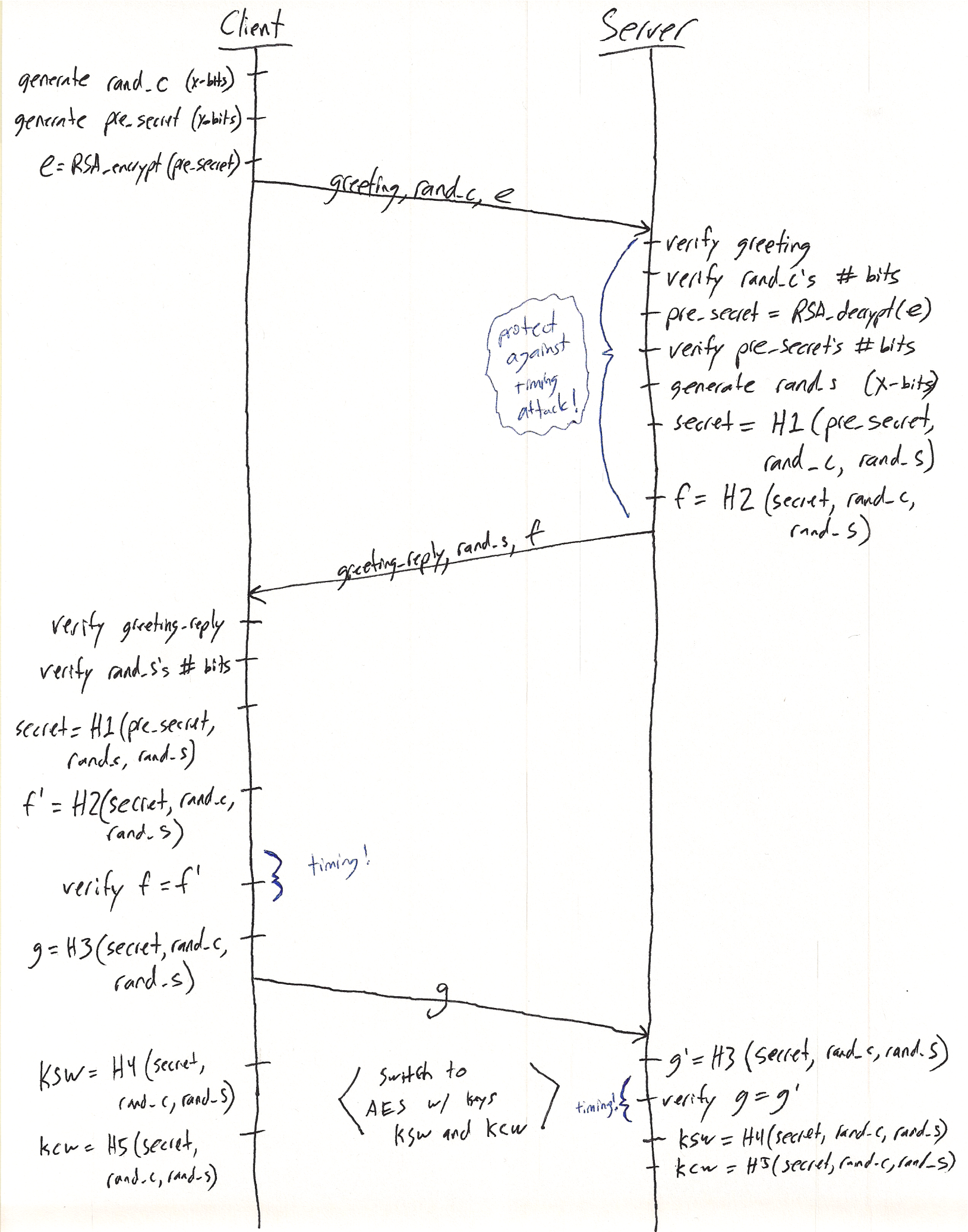
salt4 = "<diff rand stuff>"

salt5 = "<diff rand stuff>"

h4_iters = 1000

h5_iters = 1000

aes_key_len = 256


Client
Server
generate rand_c (x-bits)
generate pre_secret (y-bits)
e = RSA_encrypt(pre_secret)
greeting, rand_c, e
verify greeting
verify rand_c's # bits
pre_secret = RSA_decrypt(e)
verify pre_secret's # bits
generate rand_s (x-bits)
secret = H1(pre_secret, rand_c, rand_s)
f = H2(secret, rand_c, rand_s)
protect against timing attack!
greeting-reply, rand_s, f
verify greeting-reply
verify rand_s's # bits
secret = H1(pre_secret, rand_c, rand_s)
f' = H2(secret, rand_c, rand_s)
verify f = f'
timing!
g = H3(secret, rand_c, rand_s)
g
g' = H3(secret, rand_c, rand_s)
verify g = g'
timing!
ksw = H4(secret, rand_c, rand_s)
kcw = H5(secret, rand_c, rand_s)
switch to AES w/ keys ksw and kcw
ksw = H4(secret, rand_c, rand_s)
kcw = H5(secret, rand_c, rand_s)