Verified Time Balancing of Security Protocols
A Case Study

Student Name: Donovan Crichton
Student Number: u6881864

1 A description of the ZRTP Protocol up to public key exposure.

1.1 The Message Preamble

All ZRTP Messages are prefixed by a preamble, message length and message type.

- Let pr represent the message preamble $0x505a$ followed by a 16-bit message length in 32-bit words, followed by the message type 2-word block string (e.g. "hello").

2 The Hello Messages

- Let v represent the 4-character long string containing the version of the ZRTP protocol.
- Let c represent the client identification string which identifies the vendor and release of the ZRTP software.
- Let $h3$ represent the following:
  - Let $h0$ represent a 256-bit random nonce.
  - Let $h1$ represent a hash of $h0$ with SHA-256.
  - Let $h2$ represent a hash of $h1$ with SHA-256.
  - Let $h3$ represent a hash of $h2$ with SHA-256.
- Let z represent the 96-bit long unique identifier for the ZRTP endpoint. This is a random number generated at installation time to act as a key when looking up shared secrets in a local cache.
- Let fs represent a 4-bit long sequence of flags. The leading bit is 0, the second represents a signature-capable flag, the third a MiTM flag to identify that this message was sent from a PBX. The fourth and final bit represents a passive flag which is only used on devices that send hello, but will never send commit messages.

- Let hc represent the number (count) of hashing algorithms.
- Let cc represent the number of cipher algorithms.
- Let ac represent the number of auth tag types.
- Let kc represent the number of key agreement types.
- Let sc represent the number of SAS types.
- Let hs represent the ordered set of hashing algorithms.
- Let cs represent the set of cipher algorithms.
- Let as represent the set of SRTP auth tag types.
- Let ks represent the set of key agreement types.
- Let ss represents the set of short authentication string types.
- Let m represents the machine authentication code that finishes the ZRTP message.

2.0.1 Alice says Hello to Bob

$alice_0(pr, v, c, h3, z, fs, hc, cc, ac, kc, sc, hs, cs, as, ks, ss, m)$

2.0.2 Bob receives Alice's Hello

$bob_0(pr_{a0}, v_{a0}, c_{a0}, h3_{a0}, z_{a0}, fs_{a0}, hc_{a0}, cc_{a0}, ac_{a0}, kc_{a0}, sc_{a0}, hs_{a0}, cs_{a0}, as_{a0}, ks_{a0}, ss_{a0}, m_{a0})$

2.0.3 Bob acknowledges Alice's hello via HelloAck

$bob_1(pr)$

2.0.4 Alice receives Bob's HelloAck

$alice_1(pr_{b1})$

2.0.5 Bob says Hello to Alice

$bob_2(pr, v, c, h3, z, fs, hc, cc, ac, kc, sc, hs, cs, as, ks, ss, m)$

2.0.6 Alice receives Bob's Hello

$alice_2(pr_{b2}, v_{b2}, c_{b2}, h3_{b2}, z_{b2}, fs_{b2}, hc_{a0}, cc_{b2}, ac_{b2}, kc_{b2}, sc_{b2}, hs_{b2}, cs_{b2}, as_{b2}, ks_{b2}, ss_{b2}, m_{b2})$

2.0.7 Alice acknowledges Bob's Hello via HelloAck

$alice_3(pr)$

2.0.8 Bob receives Alice's HelloAck

$bob_3(pr_{a3})$

2.1 The Commit Message

- Let ha represent the negotiated hash algorithm.
- Let ca represent the negotiated cipher algorithm.
- Let at represent the negotiated auth-tag type.
- Let kt represent the negotiated key-agreement type.
- Let st represent the negotiated short authentication string type.
- Let hvi represent the initiators hash value where:
  - Let svi represent a random number.
  - Let g represent the base.
  - Let p represent the prime modulus.
  - svi, g, and p are all functions of kt such that:
  - Let $pvi = mod(g^{svi}, p)$.
  - Let $||$ denote bit-wise concatenation.
  - Let $dh2i$ represent the initiators DH Part 2 Message.
  - Let $hellor$ represent the responders Hello Message.
  - $hvi = ha(dh2i||hellor)$

2.1.1 Bob prepares a DH Part 2 Message

$bob_4(pr, h1, rs1i, rs2i, asi, psi, pvi, m)$

2.1.2 Bob sends a Commit to Alice

$bob_5(pr, h2, z, ha, ca, at, kt, st, hvi, m)$

2.1.3 Alice receives Bob's Commit

$alice_4(pr_{b5}, h2_{b5}, z_{b5}, ha_{b5}, ca_{b5}, at_{b5}, kt_{b5}, st_{b5}, hvi_{b5}, m_{b5})$

2.2 The DH Part 1 Message

- Let $rstr$ represent a message denoting "Responder".
- Let $s1r$ represent the responders first shared secret if it exists, or null otherwise.
- Let $s2r$ represent the responders second shared secret if it exists, or null otherwise.
- Let $rs1r = m(s1r, rstr)$ represent the non-invertible hash of the responders first retained shared secret.
- Let $rs2r = m(m2r, rstr)$ represent the non-invertible hash of the responders second retained shared secret.
- Let $auxr$ represent the responders auxiliary shared secret.
- Let $rh3$ represent the responder H3 value.
- Let $asr = m(auxr, rh3)$ represent the non-invertible hash of the responders auxiliary shared secret.
- Let $pbxr$ represent the responders pbx shared secret.
- Let $psr = m(pbx, rstr)$ represent the non-invertible hash of the responders pbx shared secret.
- Let pvr represent the responders hash value where:
  - Let svr represent a random number.
  - Let g represent the base.
  - Let p represent the prime modulus.
  - svr, g, and p are all functions of kt such that:
  - Let $pvr = mod(g^{svr}, p)$.

2.2.1 Alice sends DH Part 1 to Bob

$alice_5 = (pr, h1, rs1r, rs2r, asr, psr, pvr, m)$

2.2.2 Bob receives Alice's DH Part 1

$bob_6 = (pr_{a5}, h1_{a5}, rs1r_{a5}, rs2r_{a5}, asr_{a5}, psr_{a5}, pvr_{a5}, m_{a5})$

2.2.3 Bob checks Alice's DH Part 1 and calculates DH Result

- if $pvr_{a5} = p_{b5}$ or $pvr_{a5} = p_{b5} - 1$ then terminate protocol. Otherwise continue:
- Let $dhr = mod(pvr_{a5}^{svi_{b5}}, p)$ represent the DH result.

2.3 The DH Part 2 Message

- Let $istr$ represent a message denoting “Initiator”.
- Let $s1$ represent the initiators first shared secret if it exists or null otherwise.
- Let $s2$ represent the initiators second shared secret if it exists or null otherwise.
- Let $rs1i = m(s1, istr)$ represent the non-invertible hash of the initiators first retained shared secret.
- Let $rs2i = m(s2, istr)$ represent the non-invertible hash of the initiators second retained shared secret.
- Let $auxi$ represent the auxiliary shared secret.
- Let $ih3$ represent the initiators H3 value.
- Let $asi = m(auxi, ih3)$ represent the non-invertible hash of the initiators auxiliary shared secret.
- Let $pbxi$ represent the pbx shared secret.
- Let $psi = m(pbx, istr)$ represent the non-invertible hash of the initiators pbx shared secret.

2.3.1 Bob sends DH Part 2 to Alice

$bob_7 = bob_4$

2.3.2 Alice receives Bob's DH Part 2

$alice_6(pr_{b7}, h1_{b7}, rs1i_{b7}, rs2i_{b7}, asi_{b7}, psi_{b7}, pvi_{b7}, m_{b7})$

2.3.3 Alice checks Bob's DH Part 2 and calculates DH Result

- if $pvi_{b7} = p_{a5}$ or $pvr_{b7} = p_{a5} - 1$ then terminate protocol. Otherwise continue:
- if $hash(bob_7 || alice_0) \neq hvi_b5$ then terminate protocol. Otherwise continue:
- Let $dhr = mod(pvr_{b7}^{svi_{a5}}, p)$ represent the DH result.

3 A Simplification of ZRTP

This section details a simplified model of the ZRTP protocol with the purpose of identifying relevant computation for timing observability.

3.1 Negotiated Protocols

To simplify the formal verification of timing observability, this simplified protocol will forgo the agreement process between the parties and all parties will assume the following:

- Let ha represent the agreed-upon hash algorithm SHA-256.
- Let ca represent the agreed-upon cipher algorithm AES-1.
- Let at represent the agreed-upon auth tag type HS32 - based on HMAC-SHA1 in RFC 3711.
- Let kt represent the agreed-upon key agreement type DH3k, p=3072 Bit Prime as per RFC 3526 section 4.
- Let st represent the agreed-upon short authentication string type B32

3.2 The Hello and HelloAck Messages

If this protocol assumes pre-agreement on protocol parameters by Alice and Bob, and assumes that both Alice and Bob pre-compute $h0$, $h1$, $h2$, and $h3$ then the "Hello" messages and the corresponding "HelloAck" messages can be considered to have a constant time, and thus possibly be omitted for the purpose of timing observability.
while $h3$ can be pre-calculated, it is important for the responder to send this to the initiator as this allows the responder to validate against the initiators DH Part 2 message.

- Let $h0$ represent a 256 bit random nonce.
- Let $h1$ represent SHA-256 of $h0$
- Let $h2$ represent SHA-256 of $h1$
- Let $h3$ represent SHA-256 of $h2$

3.2.1 Alice sends Hello

$alice_0(h3_{a0})$

3.2.2 Bob sends Hello

$bob_0(h3_{b0})$

3.3 The Commit Message

Given the assumptions above, the commit message contains the following components that are required to be computed by the initiator.

- Let hvi represent the initiators hash value under DH3k key agreement type, such that:
  - Let svi represent a random 3072 bit number under the DH3k key agreement type.
  - Let $p = 2^{3072} - 2^{3008} - 1 + 2^{64}(2^{2942}\pi + 1690314)$ represent the prime number under the DH3k key agreement type.
  - Let $g = 2$ represent the generator under the DH3k key agreement type.
  - Let $pvi = mod(g^s vi, p)$ represent the public value of the initiator.
  - Let $dh2i$ represent the calculation of the DH Part 2 message by the initiator in order to compute part of the hvi.
  - Let $h3_{a0}$ represent Alice's Hello message that Bob has received.
  - Let $||$ denote bit-wise concatenation.
  - Then $hvi = ha(dh2i||h3_{a0})$

3.3.1 Bob pre-calculates DH Part 2

$bob_1 = (h1_{b0}, pvi)$

3.3.2 Bob sends a Commit to Alice

$bob_2(h2_{b1}, hvi)$

3.4 The DH Part 1 Message

The DH Part 1 message is composed primarily of book keeping parameters regarding previous shared secrets between the responder and initiator. These details will be omitted in this simplified model in the interest of brevity as they do not require any computing beyond a simple looking and hash.

- Let pvr represent the responders public hash value under the DH3k key agreement type where:
  - Let svr represent a random 3072 bit number under the DH3k key agreement type.
  - Let $p = 2^{3072} - 2^{3008} - 1 + 2^{64}(2^{2942}\pi + 1690314)$ represent the prime number under the DH3k key agreement type.
  - Let $g = 2$ represent the generator under the DH3k key agreement type.
  - Let $pvr = mod(g^s vr, p)$ represent the public value of the initiator.

3.4.1 Alice sends DH Part 1 to Bob

$alice_1(h1_{a0}, pvr)$

3.4.2 Bob calculates the DH Result.

$$bob_3(pvr_{a1}, svi_{b1}, p_{b1}) = \begin{cases} \text{Terminate Protocol,} & \text{if } pvr_{a1} = p_{b1} \\ \text{Terminate Protocol,} & \text{otherwise if } pvr_{a1} = p_{b1} - 1 \\ \text{Let } dhr = mod(pvr_{a1}^{svi_{b1}}, p), & \text{otherwise.} \end{cases}$$

3.5 The DH Part 2 Message

This message is a mirror of the DH Part 1 Message from the initiators viewpoint. The calculation of the pvi is covered under section 3.3.

3.5.1 Bob sends DH Part 2 to Alice

$bob_4(h2_{b0}, pvi_{b1})$

3.5.2 Alice calculates the DH Result.

$$alice_2(pvr_{a1}, svi_{b1}, p_{b1}) = \begin{cases} \text{Terminate Protocol,} & \text{if } pvr_{a1} = p_{b1} \\ \text{Terminate Protocol,} & \text{otherwise if } pvr_{a1} = p_{b1} - 1 \\ \text{Terminate Protocol,} & \text{otherwise if } ha(bob_4 || alice_0) \neq hvi_{b2} \\ \text{Let } dhr = mod(pvr_{a1}^{svi_{b1}}, p), & \text{otherwise.} \end{cases}$$

4 A Prototype Domain Specific Language

4.1 Values

A value v is given by the following:
let $b32$ represent a 32 bit integer.

$$\begin{aligned} v \ ::= & \ val(b32) \\ & \mid add(v_1, v_2) \\ & \mid sub(v_1, v_2) \\ & \mid xor(v_1, v_2) \\ & \mid rightrotate(v_1, v_2) \\ & \mid rightshift(v_1, v_2) \end{aligned}$$

4.2 *Prg*: a DSL for the constrained protocol.

let n represent a natural number.
let v represent the value type.
let s represent the string type.
let $vec(n,t)$ represent the vector type of length n of values of type t.

$$
\begin{aligned}
prg(n) \ ::= \ & \text{Terminate:} \\
& halt : prg(1) \\
& \text{Assign a constant:} \\
& \mid assc(name : s, val : v, cont : prg(k)) : prg(k+1) \\
& \text{Assign an array:} \\
& \mid assa(name : s, vals : vec(size, v), cont : prg(k)) : prg(k + size + 1) \\
& \text{Conditional branching:} \\
& \mid cond(pred : prg(n), true : prg(m), false : prg(m), cont : prg(k)) : prg(k + m + n + 1) \\
& \text{Repetition:} \\
& \mid do(counts : n, iter : prg(k), cont : prg(m)) : prg(k * n + m) \\
& \text{Do nothing:} \\
& \mid skip(cont : prg(k)) : prg(k+1)
\end{aligned}
$$

4.3 Describing SHA-256

4.3.1 Initialising the hash values:

$$
\begin{aligned}
\text{let } hs : vec(8, v) \text{ represent } & [val(6a09e667), val(bb67ae85), val(3c6ef372), \\
& val(a54ff53a), val(510e527f), val(9b05688c), \\
& val(1f83d9ab), val(5be0cd19)].
\end{aligned}
$$

$initHashValue : prg(n) \mapsto prg(n + 8 + 1)$
$initHashValue(p) = assa(\text{“}hs\text{”}, hs, p)$

4.3.2 Initialising the round constants:

$$
\begin{aligned}
\text{let } ks : vec(64, v) \text{ represent } [&val(428a2f98), val(71374491), val(b5c0fbcf),\\
&val(e9b5dba5), val(3956c25b), val(59f111f1),\\
&val(923f82a4), val(ab1c5ed5),\\
&val(d807aa98), val(12835b01), val(243185be),\\
&val(550c7dc3), val(72be5d74), val(80deb1fe),\\
&val(9bdc06a7), val(c19bf174),\\
&val(e49b69c1), val(efbe4786), val(0fc19dc6),\\
&val(240ca1cc), val(2de92c6f), val(4a7484aa),\\
&val(5cb0a9dc), val(76f988da),\\
&val(983e5152), val(a831c66d), val(b00327c8),\\
&val(bf597fc7), val(c6e00bf3), val(d5a79147),\\
&val(06ca6351), val(14292967),\\
&val(27b70a85), val(2e1b2138), val(4d2c6dfc),\\
&val(53380d13), val(650a7354), val(766a0abb),\\
&val(81c2c92e), val(92722c85),\\
&val(a2bfe8a1), val(a81a664b), val(c24b8b70),\\
&val(c76c51a3), val(d192e819), val(d6990624),\\
&val(f40e3585), val(106aa070),\\
&val(19a4c116), val(1e376c08), val(2748774c),\\
&val(34b0bcb5), val(391c0cb3), val(4ed8aa4a),\\
&val(5b9cca4f), val(682e6ff3),\\
&val(748f82ee), val(78a5636f), val(84c87814),\\
&val(8cc70208), val(90befffa), val(a4506ceb),\\
&val(bef9a3f7), val(c67178f2)]
\end{aligned}
$$

$initRoundConsts : prg(n) \mapsto prg(n + 64 + 1)$
$initRoundConsts(p) = assa(\text{“}ks\text{”}, ks, p)$

4.3.3 Padding the original message.

$$
\begin{aligned}
\text{let } ps(ms) : vec(8, v) \mapsto vec(16, v) \text{ represent } [&ms_0, ms_1, ms_2, ms_3, ms_4, ms_5, ms_6, ms_7\\
&val(80000000), val(00000000), val(00000000),\\
&val(00000000), val(00000000), val(00000000),\\
&val(00000000), val(00000100)]
\end{aligned}
$$

$pad256 : vec(8, v) \mapsto prg(n) \mapsto prg(n + 16 + 1)$
$pad256(ms, p) = assa(\text{“}ms\text{”}, ps(ms), p)$