```
**********************************************************
    9157 Thu Mar 20 14:23:49 2008
new/usr/src/common/crypto/sha1/amd64/sha1-x86_64.pl
6665607 Need a SHA256/SHA384/SHA512 implementation optimized for 64-bit x86
**********************************************************
______unchanged_portion_omitted_


#
# void sha1_block_data_order(SHA1_CTX *ctx, const void *inpp, size_t blocks);
#

# Arguments:
$ctx="%rdi";    # 1st arg
$inp="%rsi";    # 2nd arg
$num="%rdx";    # 3rd arg

# reassign arguments in order to produce more compact code
$ctx="%r8";
$inp="%r9";
$num="%r10";

# Temporaries:
$xi="%eax";
$t0="%ebx";
$t1="%ecx";
# State information from SHA-1 context:
$A="%edx";
$B="%esi";
$C="%edi";
$D="%ebp";
$E="%r11d";
# Temporary:
$T="%r12d";

@V=($A,$B,$C,$D,$E,$T);

sub PROLOGUE {
my $func=shift;
$code.=<<___;
ENTRY_NP($func)
        /* EXPORT DELETE START */
        push    %rbx
        push    %rbp
        push    %r12
        mov     %rsp,%rax
        mov     %rdi,$ctx       # reassigned argument
        sub     \$'8+16*4',%rsp
        mov     %rsi,$inp       # reassigned argument
        and     \$-64,%rsp
        mov     %rdx,$num       # reassigned argument
        mov     %rax,'16*4'(%rsp)

        mov     0($ctx),$A
        mov     4($ctx),$B
        mov     8($ctx),$C
        mov     12($ctx),$D
        mov     16($ctx),$E
___
}

sub EPILOGUE {
my $func=shift;
$code.=<<___;
        mov     '16*4'(%rsp),%rsp
        pop     %r12
        pop     %rbp
        pop     %rbx
        /* EXPORT DELETE END */
        ret
SET_SIZE($func)
______unchanged_portion_omitted_
```



```
$code=<<___;
#if !defined(lint) && !defined(__lint)
        .ident  "@(#)sha1-x86_64.pl     1.2     08/03/20 SMI"
        .ident  "@(#)sha1-x86_64.pl     1.1     08/03/02 SMI"
#include <sys/asm_linkage.h>
___


&PROLOGUE("sha1_block_data_order");
$code.=".align  4\n.Lloop:\n";
for($i=0;$i<20;$i++)    { &BODY_00_19($i,@V); unshift(@V,pop(@V)); }
for(;$i<40;$i++)        { &BODY_20_39($i,@V); unshift(@V,pop(@V)); }
for(;$i<60;$i++)        { &BODY_40_59($i,@V); unshift(@V,pop(@V)); }
for(;$i<80;$i++)        { &BODY_20_39($i,@V); unshift(@V,pop(@V)); }
$code.=<<___;
        / Update and save state information in SHA-1 context
        add     0($ctx),$E
        add     4($ctx),$T
        add     8($ctx),$A
        add     12($ctx),$B
        add     16($ctx),$C
        mov     $E,0($ctx)
        mov     $T,4($ctx)
        mov     $A,8($ctx)
        mov     $B,12($ctx)
        mov     $C,16($ctx)

        xchg    $E,$A   # mov   $E,$A
        xchg    $T,$B   # mov   $T,$B
        xchg    $E,$C   # mov   $A,$C
        xchg    $T,$D   # mov   $B,$D
                        # mov   $C,$E
        lea     '16*4'($inp),$inp
        sub     \$1,$num
        jnz     .Lloop
___
&EPILOGUE("sha1_block_data_order");
$code.=<<___;
.asciz  "SHA1 block transform for x86_64, CRYPTOGAMS by <appro\@openssl.org>"

#else
        /* LINTED */
        /* Nothing to be linted in this file--it's pure assembly source. */
#endif /* !lint && !__lint */
___

################################################################################

$code =~ s/\'([^\']*)\'/eval $1/gem;
print $code;
close STDOUT;
```

****
1758 Thu Mar 20 14:23:51 2008
new/usr/src/common/crypto/sha2/amd64/THIRDPARTYLICENSE
6665607 Need a SHA256/SHA384/SHA512 implementation optimized for 64-bit x86
****

```
Copyright (c) 2006, CRYPTOGAMS by <appro@openssl.org>
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

      * Redistributions of source code must retain copyright notices,
        this list of conditions and the following disclaimer.

      * Redistributions in binary form must reproduce the above
        copyright notice, this list of conditions and the following
        disclaimer in the documentation and/or other materials
        provided with the distribution.

      * Neither the name of the CRYPTOGAMS nor the names of its
        copyright holder and contributors may be used to endorse or
        promote products derived from this software without specific
        prior written permission.

ALTERNATIVELY, provided that this notice is retained in full, this
product may be distributed under the terms of the GNU General Public
License (GPL), in which case the provisions of the GPL apply INSTEAD OF
those given above.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDER AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

```
**********************************************************
      31 Thu Mar 20 14:23:52 2008
new/usr/src/common/crypto/sha2/amd64/THIRDPARTYLICENSE.descrip
6665607 Need a SHA256/SHA384/SHA512 implementation optimized for 64-bit x86
**********************************************************
```

```
   1 PORTIONS OF SHA2 FUNCTIONALITY
```

```
**********************************************************
   11174 Thu Mar 20 14:23:52 2008
new/usr/src/common/crypto/sha2/amd64/sha512-x86_64.pl
6665607 Need a SHA256/SHA384/SHA512 implementation optimized for 64-bit x86
**********************************************************
```

```
#!/usr/bin/env perl
#
# ====================================================================
# Written by Andy Polyakov <appro@fy.chalmers.se> for the OpenSSL
# project. Rights for redistribution and usage in source and binary
# forms are granted according to the OpenSSL license.
# ====================================================================
#
# sha256/512_block procedure for x86_64.
#
# 40% improvement over compiler-generated code on Opteron. On EM64T
# sha256 was observed to run >80% faster and sha512 - >40%. No magical
# tricks, just straight implementation... I really wonder why gcc
# [being armed with inline assembler] fails to generate as fast code.
# The only thing which is cool about this module is that it's very
# same instruction sequence used for both SHA-256 and SHA-512. In
# former case the instructions operate on 32-bit operands, while in
# latter - on 64-bit ones. All I had to do is to get one flavor right,
# the other one passed the test right away:-)
#
# sha256_block runs in ~1005 cycles on Opteron, which gives you
# asymptotic performance of 64*1000/1005=63.7MBps times CPU clock
# frequency in GHz. sha512_block runs in ~1275 cycles, which results
# in 128*1000/1275=100MBps per GHz. Is there room for improvement?
# Well, if you compare it to IA-64 implementation, which maintains
# X[16] in register bank[!], tends to 4 instructions per CPU clock
# cycle and runs in 1003 cycles, 1275 is very good result for 3-way
# issue Opteron pipeline and X[16] maintained in memory. So that *if*
# there is a way to improve it, *then* the only way would be to try to
# offload X[16] updates to SSE unit, but that would require "deeper"
# loop unroll, which in turn would naturally cause size blow-up, not
# to mention increased complexity! And once again, only *if* it's
# actually possible to noticeably improve overall ILP, instruction
# level parallelism, on a given CPU implementation in this case.
#
# Special note on Intel EM64T. While Opteron CPU exhibits perfect
# perfromance ratio of 1.5 between 64- and 32-bit flavors [see above],
# [currently available] EM64T CPUs apparently are far from it. On the
# contrary, 64-bit version, sha512_block, is ~30% *slower* than 32-bit
# sha256_block:-( This is presumably because 64-bit shifts/rotates
# apparently are not atomic instructions, but implemented in microcode.

#
# OpenSolaris OS modifications
#
# Sun elects to use this software under the BSD license.
#
# This source originates from OpenSSL file sha512-x86_64.pl at
# ftp://ftp.openssl.org/snapshot/openssl-0.9.8-stable-SNAP-20080131.tar.gz
# (presumably for future OpenSSL release 0.9.8h), with these changes:
#
# 1. Added perl "use strict" and declared variables.
#
# 2. Added OpenSolaris ENTRY_NP/SET_SIZE macros from
# /usr/include/sys/asm_linkage.h, .ident keywords, and lint(1B) guards.
#
# 3. Added perl function &lea_offset_eax_register_register() to handle
#       OpenSolaris as(1) bug.
#
# 4. Removed x86_64-xlate.pl script (not needed for as(1) or gas(1)
# assemblers).  Replaced the .picmeup macro with assembler code.
```



```
#
# 5. Added 8 to $ctx, as OpenSolaris OS has an extra 4-byte field, "algotype",
# at the beginning of SHA2_CTX (the next field is 8-byte aligned).
#

use strict;
my ($code, $func, $TABLE, $SZ, @Sigma0, @Sigma1, @sigma0, @sigma1, $rounds,
        @ROT, $A, $B, $C, $D, $E, $F, $G, $H, $T1, $a0, $a1, $a2, $i,
        $ctx, $round, $inp, $Tbl, $_ctx, $_inp, $_end, $_rsp, $framesz);
my $output = shift;
open STDOUT,">$output";

#
# OpenSSL library:
# void sha512_block_data_order(SHA512_CTX *ctx, const void *in, size_t num);
# void sha256_block_data_order(SHA256_CTX *ctx, const void *in, size_t num);
#
# OpenSolaris OS:
# void SHA512TransformBlocks(SHA2_CTX *ctx, const void *in, size_t num);
# void SHA256TransformBlocks(SHA2_CTX *ctx, const void *in, size_t num);
# Note: the OpenSolaris SHA2 structure has an extra 8 byte field at the
# beginning (over OpenSSL's SHA512 or SHA256 structure).
#

if ($output =~ /512/) {
        $func="SHA512TransformBlocks";
        $TABLE="K512";
        $SZ=8;
        @ROT=($A,$B,$C,$D,$E,$F,$G,$H)=("%rax","%rbx","%rcx","%rdx",
                                        "%r8", "%r9", "%r10","%r11");
        ($T1,$a0,$a1,$a2)=("%r12","%r13","%r14","%r15");
        @Sigma0=(28,34,39);
        @Sigma1=(14,18,41);
        @sigma0=(1,  8, 7);
        @sigma1=(19,61, 6);
        $rounds=80;
} else {
        $func="SHA256TransformBlocks";
        $TABLE="K256";
        $SZ=4;
        @ROT=($A,$B,$C,$D,$E,$F,$G,$H)=("%eax","%ebx","%ecx","%edx",
                                        "%r8d","%r9d","%r10d","%r11d");
        ($T1,$a0,$a1,$a2)=("%r12d","%r13d","%r14d","%r15d");
        @Sigma0=( 2,13,22);
        @Sigma1=( 6,11,25);
        @sigma0=( 7,18, 3);
        @sigma1=(17,19,10);
        $rounds=64;
}

$ctx="%rdi";    # 1st arg
$round="%rdi";  # zaps $ctx
$inp="%rsi";    # 2nd arg
$Tbl="%rbp";

$_ctx="16*$SZ+0*8(%rsp)";
$_inp="16*$SZ+1*8(%rsp)";
$_end="16*$SZ+2*8(%rsp)";
$_rsp="16*$SZ+3*8(%rsp)";
$framesz="16*$SZ+4*8";


sub ROUND_00_15()
{ my ($i,$a,$b,$c,$d,$e,$f,$g,$h) = @_;

$code.=<<___;
```

```
128         mov     $e,$a0
129         mov     $e,$a1
130         mov     $f,$a2

132         ror     \$$Sigma1[0],$a0
133         ror     \$$Sigma1[1],$a1
134         xor     $g,$a2                  # f^g

136         xor     $a1,$a0
137         ror     \$'$Sigma1[2]-$Sigma1[1]',$a1
138         and     $e,$a2                  # (f^g)&e
139         mov     $T1,'$SZ*($i&0xf)'(%rsp)

141         xor     $a1,$a0                 # Sigma1(e)
142         xor     $g,$a2                  # Ch(e,f,g)=((f^g)&e)^g
143         add     $h,$T1                  # T1+=h

145         mov     $a,$h
146         add     $a0,$T1                 # T1+=Sigma1(e)

148         add     $a2,$T1                 # T1+=Ch(e,f,g)
149         mov     $a,$a0
150         mov     $a,$a1

152         ror     \$$Sigma0[0],$h
153         ror     \$$Sigma0[1],$a0
154         mov     $a,$a2
155         add     ($Tbl,$round,$SZ),$T1   # T1+=K[round]

157         xor     $a0,$h
158         ror     \$'$Sigma0[2]-$Sigma0[1]',$a0
159         or      $c,$a1                  # a|c

161         xor     $a0,$h                  # h=Sigma0(a)
162         and     $c,$a2                  # a&c
163         add     $T1,$d                  # d+=T1

165         and     $b,$a1                  # (a|c)&b
166         add     $T1,$h                  # h+=T1

168         or      $a2,$a1                 # Maj(a,b,c)=((a|c)&b)|(a&c)
169         lea     1($round),$round        # round++

171         add     $a1,$h                  # h+=Maj(a,b,c)
172 ___
173 }

175 sub ROUND_16_XX()
176 { my ($i,$a,$b,$c,$d,$e,$f,$g,$h) = @_;

178 $code.=<<___;
179         mov     '$SZ*(($i+1)&0xf)'(%rsp),$a0
180         mov     '$SZ*(($i+14)&0xf)'(%rsp),$T1

182         mov     $a0,$a2

184         shr     \$$sigma0[2],$a0
185         ror     \$$sigma0[0],$a2

187         xor     $a2,$a0
188         ror     \$'$sigma0[1]-$sigma0[0]',$a2

190         xor     $a2,$a0                 # sigma0(X[(i+1)&0xf])
191         mov     $T1,$a1

193         shr     \$$sigma1[2],$T1
```



```
194         ror     \$$sigma1[0],$a1

196         xor     $a1,$T1
197         ror     \$'$sigma1[1]-$sigma1[0]',$a1

199         xor     $a1,$T1                 # sigma1(X[(i+14)&0xf])

201         add     $a0,$T1

203         add     '$SZ*(($i+9)&0xf)'(%rsp),$T1

205         add     '$SZ*($i&0xf)'(%rsp),$T1
206 ___
207         &ROUND_00_15(@_);
208 }

210 #
211 # Execution begins here
212 #

214 $code=<<___;
215 #if !defined(lint) && !defined(__lint)
216         .ident  "@(#)sha512-x86_64.pl   1.2     08/03/20 SMI"
217 #include <sys/asm_linkage.h>

219 ENTRY_NP($func)
220         push    %rbx
221         push    %rbp
222         push    %r12
223         push    %r13
224         push    %r14
225         push    %r15
226         mov     %rsp,%rbp               # copy %rsp
227         shl     \$4,%rdx                # num*16
228         sub     \$$framesz,%rsp
229         lea     ($inp,%rdx,$SZ),%rdx    # inp+num*16*$SZ
230         and     \$-64,%rsp              # align stack frame
231         add     \$8,$ctx                # Skip OpenSolaris field, "algotype"
232         mov     $ctx,$_ctx              # save ctx, 1st arg
233         mov     $inp,$_inp              # save inp, 2nd arg
234         mov     %rdx,$_end              # save end pointer, "3rd" arg
235         mov     %rbp,$_rsp              # save copy of %rsp

237         /.picmeup $Tbl
238         / The .picmeup pseudo-directive, from perlasm/x86_64_xlate.pl, puts
239         / the address of the "next" instruction into the target register
240         / ($Tbl).  This generates these 2 instructions:
241         lea     .Llea(%rip),$Tbl
242         /nop    / .picmeup generates a nop for mod 8 alignment--not needed here

244 .Llea:
245         lea     $TABLE-.($Tbl),$Tbl

247         mov     $SZ*0($ctx),$A
248         mov     $SZ*1($ctx),$B
249         mov     $SZ*2($ctx),$C
250         mov     $SZ*3($ctx),$D
251         mov     $SZ*4($ctx),$E
252         mov     $SZ*5($ctx),$F
253         mov     $SZ*6($ctx),$G
254         mov     $SZ*7($ctx),$H
255         jmp     .Lloop

257 .align  16
258 .Lloop:
259         xor     $round,$round
```

```
___
        for($i=0;$i<16;$i++) {
                $code.="        mov     $SZ*$i($inp),$T1\n";
                $code.="        bswap   $T1\n";
                &ROUND_00_15($i,@ROT);
                unshift(@ROT,pop(@ROT));
        }
$code.=<<___;
        jmp     .Lrounds_16_xx
.align  16
.Lrounds_16_xx:
___
        for(;$i<32;$i++) {
                &ROUND_16_XX($i,@ROT);
                unshift(@ROT,pop(@ROT));
        }

$code.=<<___;
        cmp     \$$rounds,$round
        jb      .Lrounds_16_xx

        mov     $_ctx,$ctx
        lea     16*$SZ($inp),$inp

        add     $SZ*0($ctx),$A
        add     $SZ*1($ctx),$B
        add     $SZ*2($ctx),$C
        add     $SZ*3($ctx),$D
        add     $SZ*4($ctx),$E
        add     $SZ*5($ctx),$F
        add     $SZ*6($ctx),$G
        add     $SZ*7($ctx),$H

        cmp     $_end,$inp

        mov     $A,$SZ*0($ctx)
        mov     $B,$SZ*1($ctx)
        mov     $C,$SZ*2($ctx)
        mov     $D,$SZ*3($ctx)
        mov     $E,$SZ*4($ctx)
        mov     $F,$SZ*5($ctx)
        mov     $G,$SZ*6($ctx)
        mov     $H,$SZ*7($ctx)
        jb      .Lloop

        mov     $_rsp,%rsp
        pop     %r15
        pop     %r14
        pop     %r13
        pop     %r12
        pop     %rbp
        pop     %rbx

        ret
SET_SIZE($func)

___

if ($SZ==4) {
# SHA256
$code.=<<___;
.align  64
.type   $TABLE,\@object
$TABLE:
        .long   0x428a2f98,0x71374491,0xb5c0fbcf,0xe9b5dba5
        .long   0x3956c25b,0x59f111f1,0x923f82a4,0xab1c5ed5
```



```
        .long   0xd807aa98,0x12835b01,0x243185be,0x550c7dc3
        .long   0x72be5d74,0x80deb1fe,0x9bdc06a7,0xc19bf174
        .long   0xe49b69c1,0xefbe4786,0x0fc19dc6,0x240ca1cc
        .long   0x2de92c6f,0x4a7484aa,0x5cb0a9dc,0x76f988da
        .long   0x983e5152,0xa831c66d,0xb00327c8,0xbf597fc7
        .long   0xc6e00bf3,0xd5a79147,0x06ca6351,0x14292967
        .long   0x27b70a85,0x2e1b2138,0x4d2c6dfc,0x53380d13
        .long   0x650a7354,0x766a0abb,0x81c2c92e,0x92722c85
        .long   0xa2bfe8a1,0xa81a664b,0xc24b8b70,0xc76c51a3
        .long   0xd192e819,0xd6990624,0xf40e3585,0x106aa070
        .long   0x19a4c116,0x1e376c08,0x2748774c,0x34b0bcb5
        .long   0x391c0cb3,0x4ed8aa4a,0x5b9cca4f,0x682e6ff3
        .long   0x748f82ee,0x78a5636f,0x84c87814,0x8cc70208
        .long   0x90befffa,0xa4506ceb,0xbef9a3f7,0xc67178f2
___
} else {
# SHA512
$code.=<<___;
.align  64
.type   $TABLE,\@object
$TABLE:
        .quad   0x428a2f98d728ae22,0x7137449123ef65cd
        .quad   0xb5c0fbcfec4d3b2f,0xe9b5dba58189dbbc
        .quad   0x3956c25bf348b538,0x59f111f1b605d019
        .quad   0x923f82a4af194f9b,0xab1c5ed5da6d8118
        .quad   0xd807aa98a3030242,0x12835b0145706fbe
        .quad   0x243185be4ee4b28c,0x550c7dc3d5ffb4e2
        .quad   0x72be5d74f27b896f,0x80deb1fe3b1696b1
        .quad   0x9bdc06a725c71235,0xc19bf174cf692694
        .quad   0xe49b69c19ef14ad2,0xefbe4786384f25e3
        .quad   0x0fc19dc68b8cd5b5,0x240ca1cc77ac9c65
        .quad   0x2de92c6f592b0275,0x4a7484aa6ea6e483
        .quad   0x5cb0a9dcbd41fbd4,0x76f988da831153b5
        .quad   0x983e5152ee66dfab,0xa831c66d2db43210
        .quad   0xb00327c898fb213f,0xbf597fc7beef0ee4
        .quad   0xc6e00bf33da88fc2,0xd5a79147930aa725
        .quad   0x06ca6351e003826f,0x142929670a0e6e70
        .quad   0x27b70a8546d22ffc,0x2e1b21385c26c926
        .quad   0x4d2c6dfc5ac42aed,0x53380d139d95b3df
        .quad   0x650a73548baf63de,0x766a0abb3c77b2a8
        .quad   0x81c2c92e47edaee6,0x92722c851482353b
        .quad   0xa2bfe8a14cf10364,0xa81a664bbc423001
        .quad   0xc24b8b70d0f89791,0xc76c51a30654be30
        .quad   0xd192e819d6ef5218,0xd69906245565a910
        .quad   0xf40e35855771202a,0x106aa07032bbd1b8
        .quad   0x19a4c116b8d2d0c8,0x1e376c085141ab53
        .quad   0x2748774cdf8eeb99,0x34b0bcb5e19b48a8
        .quad   0x391c0cb3c5c95a63,0x4ed8aa4ae3418acb
        .quad   0x5b9cca4f7763e373,0x682e6ff3d6b2b8a3
        .quad   0x748f82ee5defb2fc,0x78a5636f43172f60
        .quad   0x84c87814a1f0ab72,0x8cc702081a6439ec
        .quad   0x90befffa23631e28,0xa4506cebde82bde9
        .quad   0xbef9a3f7b2c67915,0xc67178f2e372532b
        .quad   0xca273eceea26619c,0xd186b8c721c0c207
        .quad   0xeada7dd6cde0eb1e,0xf57d4f7fee6ed178
        .quad   0x06f067aa72176fba,0x0a637dc5a2c898a6
        .quad   0x113f9804bef90dae,0x1b710b35131c471b
        .quad   0x28db77f523047d84,0x32caab7b40c72493
        .quad   0x3c9ebe0a15c9bebc,0x431d67c49c100d4c
        .quad   0x4cc5d4becb3e42b6,0x597f299cfc657e2a
        .quad   0x5fcb6fab3ad6faec,0x6c44198c4a475817
___
}
$code.=<<___;

#else
```

```
392         /* LINTED */
393         /* Nothing to be linted in this file--it's pure assembly source. */
394 #endif /* !lint && !__lint */
395 ___

397 $code =~ s/\`([^\`]*)\`/eval $1/gem;
398 print $code;
399 close STDOUT;
```

```
**********************************************************
   31347 Thu Mar 20 14:23:53 2008
new/usr/src/common/crypto/sha2/sha2.c
6665607 Need a SHA256/SHA384/SHA512 implementation optimized for 64-bit x86
**********************************************************
```

```
   1 /*
   2  * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
   2  * Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
   3  * Use is subject to license terms.
   4  */

   6 #pragma ident   "@(#)sha2.c     1.8     08/03/05 SMI"
   6 #pragma ident   "@(#)sha2.c     1.7     07/04/10 SMI"


   8 /*
   9  * The basic framework for this code came from the reference
  10  * implementation for MD5.  That implementation is Copyright (C)
  11  * 1991-2, RSA Data Security, Inc. Created 1991. All rights reserved.
  12  *
  13  * License to copy and use this software is granted provided that it
  14  * is identified as the "RSA Data Security, Inc. MD5 Message-Digest
  15  * Algorithm" in all material mentioning or referencing this software
  16  * or this function.
  17  *
  18  * License is also granted to make and use derivative works provided
  19  * that such works are identified as "derived from the RSA Data
  20  * Security, Inc. MD5 Message-Digest Algorithm" in all material
  21  * mentioning or referencing the derived work.
  22  *
  23  * RSA Data Security, Inc. makes no representations concerning either
  24  * the merchantability of this software or the suitability of this
  25  * software for any particular purpose. It is provided "as is"
  26  * without express or implied warranty of any kind.
  27  *
  28  * These notices must be retained in any copies of any part of this
  29  * documentation and/or software.
  30  *
  31  * NOTE: Cleaned-up and optimized, version of SHA2, based on the FIPS 180-2
  32  * standard, available at http://www.itl.nist.gov/div897/pubs/fip180-2.htm
  33  * Not as fast as one would like -- further optimizations are encouraged
  34  * and appreciated.
  35  */

  37 #include <sys/types.h>
  38 #include <sys/param.h>
  39 #include <sys/systm.h>
  40 #include <sys/sysmacros.h>
  41 #define _SHA2_IMPL
  42 #include <sys/sha2.h>
  43 #include <sys/sha2_consts.h>

  45 #ifdef _KERNEL
  46 #include <sys/cmn_err.h>
  46 #ifndef _KERNEL

  48 #else
  49 #include <strings.h>
  50 #include <stdlib.h>
  51 #include <errno.h>

  53 #pragma weak SHA256Update = SHA2Update
  54 #pragma weak SHA384Update = SHA2Update
  55 #pragma weak SHA512Update = SHA2Update

  57 #pragma weak SHA256Final = SHA2Final
```



```
  58 #pragma weak SHA384Final = SHA2Final
  59 #pragma weak SHA512Final = SHA2Final

  60 #endif  /* !_KERNEL */

  62 #ifdef _KERNEL
  63 #include <sys/cmn_err.h>
  61 #endif  /* _KERNEL */

  63 static void Encode(uint8_t *, uint32_t *, size_t);
  64 static void Encode64(uint8_t *, uint64_t *, size_t);

  66 #if     defined(__amd64)
  67 #define SHA512Transform(ctx, in) SHA512TransformBlocks((ctx), (in), 1)
  68 #define SHA256Transform(ctx, in) SHA256TransformBlocks((ctx), (in), 1)

  70 void SHA512TransformBlocks(SHA2_CTX *ctx, const void *in, size_t num);
  71 void SHA256TransformBlocks(SHA2_CTX *ctx, const void *in, size_t num);

  73 #else
  74 static void SHA256Transform(SHA2_CTX *, const uint8_t *);
  75 static void SHA512Transform(SHA2_CTX *, const uint8_t *);
  76 #endif  /* __amd64 */

  78 static uint8_t PADDING[128] = { 0x80, /* all zeros */ };

  80 /* Ch and Maj are the basic SHA2 functions. */
  81 #define Ch(b, c, d)     (((b) & (c)) ^ ((~b) & (d)))
  82 #define Maj(b, c, d)    (((b) & (c)) ^ ((b) & (d)) ^ ((c) & (d)))

  84 /* Rotates x right n bits. */
  85 #define ROTR(x, n)      \
  86         (((x) >> (n)) | ((x) << ((sizeof (x) * NBBY)-(n))))

  88 /* Shift x right n bits */
  89 #define SHR(x, n)       ((x) >> (n))

  91 /* SHA256 Functions */
  92 #define BIGSIGMA0_256(x)        (ROTR((x), 2) ^ ROTR((x), 13) ^ ROTR((x), 22))
  93 #define BIGSIGMA1_256(x)        (ROTR((x), 6) ^ ROTR((x), 11) ^ ROTR((x), 25))
  94 #define SIGMA0_256(x)           (ROTR((x), 7) ^ ROTR((x), 18) ^ SHR((x), 3))
  95 #define SIGMA1_256(x)           (ROTR((x), 17) ^ ROTR((x), 19) ^ SHR((x), 10))

  97 #define SHA256ROUND(a, b, c, d, e, f, g, h, i, w)                       \
  98         T1 = h + BIGSIGMA1_256(e) + Ch(e, f, g) + SHA256_CONST(i) + w;  \
  99         d += T1;                                                        \
 100         T2 = BIGSIGMA0_256(a) + Maj(a, b, c);                           \
 101         h = T1 + T2

 103 /* SHA384/512 Functions */
 104 #define BIGSIGMA0(x)    (ROTR((x), 28) ^ ROTR((x), 34) ^ ROTR((x), 39))
 105 #define BIGSIGMA1(x)    (ROTR((x), 14) ^ ROTR((x), 18) ^ ROTR((x), 41))
 106 #define SIGMA0(x)       (ROTR((x), 1) ^ ROTR((x), 8) ^ SHR((x), 7))
 107 #define SIGMA1(x)       (ROTR((x), 19) ^ ROTR((x), 61) ^ SHR((x), 6))
 108 #define SHA512ROUND(a, b, c, d, e, f, g, h, i, w)                       \
 109         T1 = h + BIGSIGMA1(e) + Ch(e, f, g) + SHA512_CONST(i) + w;      \
 110         d += T1;                                                        \
 111         T2 = BIGSIGMA0(a) + Maj(a, b, c);                               \
 112         h = T1 + T2

 114 /*
 115  * sparc optimization:
 116  *
 117  * on the sparc, we can load big endian 32-bit data easily.  note that
 118  * special care must be taken to ensure the address is 32-bit aligned.
 119  * in the interest of speed, we don't check to make sure, since
```

```
 * careful programming can guarantee this for us.
 */

#if     defined(_BIG_ENDIAN)

#define LOAD_BIG_32(addr)       (*(uint32_t *)(addr))

#else   /* little endian -- will work on big endian, but slowly */

#define LOAD_BIG_32(addr)       \
        (((addr)[0] << 24) | ((addr)[1] << 16) | ((addr)[2] << 8) | (addr)[3])
#endif


#if     defined(_BIG_ENDIAN)

#define LOAD_BIG_64(addr)       (*(uint64_t *)(addr))

#else   /* little endian -- will work on big endian, but slowly */

#define LOAD_BIG_64(addr)       \
        (((uint64_t)(addr)[0] << 56) | ((uint64_t)(addr)[1] << 48) |    \
            ((uint64_t)(addr)[2] << 40) | ((uint64_t)(addr)[3] << 32) | \
            ((uint64_t)(addr)[4] << 24) | ((uint64_t)(addr)[5] << 16) | \
            ((uint64_t)(addr)[6] << 8) | (uint64_t)(addr)[7])

#endif


#if     !defined(__amd64)
/* SHA256 Transform */

static void
SHA256Transform(SHA2_CTX *ctx, const uint8_t *blk)
{

        uint32_t a = ctx->state.s32[0];
        uint32_t b = ctx->state.s32[1];
        uint32_t c = ctx->state.s32[2];
        uint32_t d = ctx->state.s32[3];
        uint32_t e = ctx->state.s32[4];
        uint32_t f = ctx->state.s32[5];
        uint32_t g = ctx->state.s32[6];
        uint32_t h = ctx->state.s32[7];

        uint32_t w0, w1, w2, w3, w4, w5, w6, w7;
        uint32_t w8, w9, w10, w11, w12, w13, w14, w15;
        uint32_t T1, T2;

#if     defined(__sparc)
        static const uint32_t sha256_consts[] = {
                SHA256_CONST_0, SHA256_CONST_1, SHA256_CONST_2,
                SHA256_CONST_3, SHA256_CONST_4, SHA256_CONST_5,
                SHA256_CONST_6, SHA256_CONST_7, SHA256_CONST_8,
                SHA256_CONST_9, SHA256_CONST_10, SHA256_CONST_11,
                SHA256_CONST_12, SHA256_CONST_13, SHA256_CONST_14,
                SHA256_CONST_15, SHA256_CONST_16, SHA256_CONST_17,
                SHA256_CONST_18, SHA256_CONST_19, SHA256_CONST_20,
                SHA256_CONST_21, SHA256_CONST_22, SHA256_CONST_23,
                SHA256_CONST_24, SHA256_CONST_25, SHA256_CONST_26,
                SHA256_CONST_27, SHA256_CONST_28, SHA256_CONST_29,
                SHA256_CONST_30, SHA256_CONST_31, SHA256_CONST_32,
                SHA256_CONST_33, SHA256_CONST_34, SHA256_CONST_35,
                SHA256_CONST_36, SHA256_CONST_37, SHA256_CONST_38,
                SHA256_CONST_39, SHA256_CONST_40, SHA256_CONST_41,
                SHA256_CONST_42, SHA256_CONST_43, SHA256_CONST_44,
```



```
                SHA256_CONST_45, SHA256_CONST_46, SHA256_CONST_47,
                SHA256_CONST_48, SHA256_CONST_49, SHA256_CONST_50,
                SHA256_CONST_51, SHA256_CONST_52, SHA256_CONST_53,
                SHA256_CONST_54, SHA256_CONST_55, SHA256_CONST_56,
                SHA256_CONST_57, SHA256_CONST_58, SHA256_CONST_59,
                SHA256_CONST_60, SHA256_CONST_61, SHA256_CONST_62,
                SHA256_CONST_63
        };
#endif  /* __sparc */
#endif

        if ((uintptr_t)blk & 0x3) {             /* not 4-byte aligned? */
                bcopy(blk, ctx->buf_un.buf32,  sizeof (ctx->buf_un.buf32));
                blk = (uint8_t *)ctx->buf_un.buf32;
        }

        /* LINTED E_BAD_PTR_CAST_ALIGN */
        w0 =  LOAD_BIG_32(blk + 4 * 0);
        SHA256ROUND(a, b, c, d, e, f, g, h, 0, w0);
        /* LINTED E_BAD_PTR_CAST_ALIGN */
        w1 =  LOAD_BIG_32(blk + 4 * 1);
        SHA256ROUND(h, a, b, c, d, e, f, g, 1, w1);
        /* LINTED E_BAD_PTR_CAST_ALIGN */
        w2 =  LOAD_BIG_32(blk + 4 * 2);
        SHA256ROUND(g, h, a, b, c, d, e, f, 2, w2);
        /* LINTED E_BAD_PTR_CAST_ALIGN */
        w3 =  LOAD_BIG_32(blk + 4 * 3);
        SHA256ROUND(f, g, h, a, b, c, d, e, 3, w3);
        /* LINTED E_BAD_PTR_CAST_ALIGN */
        w4 =  LOAD_BIG_32(blk + 4 * 4);
        SHA256ROUND(e, f, g, h, a, b, c, d, 4, w4);
        /* LINTED E_BAD_PTR_CAST_ALIGN */
        w5 =  LOAD_BIG_32(blk + 4 * 5);
        SHA256ROUND(d, e, f, g, h, a, b, c, 5, w5);
        /* LINTED E_BAD_PTR_CAST_ALIGN */
        w6 =  LOAD_BIG_32(blk + 4 * 6);
        SHA256ROUND(c, d, e, f, g, h, a, b, 6, w6);
        /* LINTED E_BAD_PTR_CAST_ALIGN */
        w7 =  LOAD_BIG_32(blk + 4 * 7);
        SHA256ROUND(b, c, d, e, f, g, h, a, 7, w7);
        /* LINTED E_BAD_PTR_CAST_ALIGN */
        w8 =  LOAD_BIG_32(blk + 4 * 8);
        SHA256ROUND(a, b, c, d, e, f, g, h, 8, w8);
        /* LINTED E_BAD_PTR_CAST_ALIGN */
        w9 =  LOAD_BIG_32(blk + 4 * 9);
        SHA256ROUND(h, a, b, c, d, e, f, g, 9, w9);
        /* LINTED E_BAD_PTR_CAST_ALIGN */
        w10 =  LOAD_BIG_32(blk + 4 * 10);
        SHA256ROUND(g, h, a, b, c, d, e, f, 10, w10);
        /* LINTED E_BAD_PTR_CAST_ALIGN */
        w11 =  LOAD_BIG_32(blk + 4 * 11);
        SHA256ROUND(f, g, h, a, b, c, d, e, 11, w11);
        /* LINTED E_BAD_PTR_CAST_ALIGN */
        w12 =  LOAD_BIG_32(blk + 4 * 12);
        SHA256ROUND(e, f, g, h, a, b, c, d, 12, w12);
        /* LINTED E_BAD_PTR_CAST_ALIGN */
        w13 =  LOAD_BIG_32(blk + 4 * 13);
        SHA256ROUND(d, e, f, g, h, a, b, c, 13, w13);
        /* LINTED E_BAD_PTR_CAST_ALIGN */
        w14 =  LOAD_BIG_32(blk + 4 * 14);
        SHA256ROUND(c, d, e, f, g, h, a, b, 14, w14);
        /* LINTED E_BAD_PTR_CAST_ALIGN */
        w15 =  LOAD_BIG_32(blk + 4 * 15);
        SHA256ROUND(b, c, d, e, f, g, h, a, 15, w15);

        w0 = SIGMA1_256(w14) + w9 + SIGMA0_256(w1) + w0;
```

```
249         SHA256ROUND(a, b, c, d, e, f, g, h, 16, w0);
250         w1 = SIGMA1_256(w15) + w10 + SIGMA0_256(w2) + w1;
251         SHA256ROUND(h, a, b, c, d, e, f, g, 17, w1);
252         w2 = SIGMA1_256(w0) + w11 + SIGMA0_256(w3) + w2;
253         SHA256ROUND(g, h, a, b, c, d, e, f, 18, w2);
254         w3 = SIGMA1_256(w1) + w12 + SIGMA0_256(w4) + w3;
255         SHA256ROUND(f, g, h, a, b, c, d, e, 19, w3);
256         w4 = SIGMA1_256(w2) + w13 + SIGMA0_256(w5) + w4;
257         SHA256ROUND(e, f, g, h, a, b, c, d, 20, w4);
258         w5 = SIGMA1_256(w3) + w14 + SIGMA0_256(w6) + w5;
259         SHA256ROUND(d, e, f, g, h, a, b, c, 21, w5);
260         w6 = SIGMA1_256(w4) + w15 + SIGMA0_256(w7) + w6;
261         SHA256ROUND(c, d, e, f, g, h, a, b, 22, w6);
262         w7 = SIGMA1_256(w5) + w0 + SIGMA0_256(w8) + w7;
263         SHA256ROUND(b, c, d, e, f, g, h, a, 23, w7);
264         w8 = SIGMA1_256(w6) + w1 + SIGMA0_256(w9) + w8;
265         SHA256ROUND(a, b, c, d, e, f, g, h, 24, w8);
266         w9 = SIGMA1_256(w7) + w2 + SIGMA0_256(w10) + w9;
267         SHA256ROUND(h, a, b, c, d, e, f, g, 25, w9);
268         w10 = SIGMA1_256(w8) + w3 + SIGMA0_256(w11) + w10;
269         SHA256ROUND(g, h, a, b, c, d, e, f, 26, w10);
270         w11 = SIGMA1_256(w9) + w4 + SIGMA0_256(w12) + w11;
271         SHA256ROUND(f, g, h, a, b, c, d, e, 27, w11);
272         w12 = SIGMA1_256(w10) + w5 + SIGMA0_256(w13) + w12;
273         SHA256ROUND(e, f, g, h, a, b, c, d, 28, w12);
274         w13 = SIGMA1_256(w11) + w6 + SIGMA0_256(w14) + w13;
275         SHA256ROUND(d, e, f, g, h, a, b, c, 29, w13);
276         w14 = SIGMA1_256(w12) + w7 + SIGMA0_256(w15) + w14;
277         SHA256ROUND(c, d, e, f, g, h, a, b, 30, w14);
278         w15 = SIGMA1_256(w13) + w8 + SIGMA0_256(w0) + w15;
279         SHA256ROUND(b, c, d, e, f, g, h, a, 31, w15);

281         w0 = SIGMA1_256(w14) + w9 + SIGMA0_256(w1) + w0;
282         SHA256ROUND(a, b, c, d, e, f, g, h, 32, w0);
283         w1 = SIGMA1_256(w15) + w10 + SIGMA0_256(w2) + w1;
284         SHA256ROUND(h, a, b, c, d, e, f, g, 33, w1);
285         w2 = SIGMA1_256(w0) + w11 + SIGMA0_256(w3) + w2;
286         SHA256ROUND(g, h, a, b, c, d, e, f, 34, w2);
287         w3 = SIGMA1_256(w1) + w12 + SIGMA0_256(w4) + w3;
288         SHA256ROUND(f, g, h, a, b, c, d, e, 35, w3);
289         w4 = SIGMA1_256(w2) + w13 + SIGMA0_256(w5) + w4;
290         SHA256ROUND(e, f, g, h, a, b, c, d, 36, w4);
291         w5 = SIGMA1_256(w3) + w14 + SIGMA0_256(w6) + w5;
292         SHA256ROUND(d, e, f, g, h, a, b, c, 37, w5);
293         w6 = SIGMA1_256(w4) + w15 + SIGMA0_256(w7) + w6;
294         SHA256ROUND(c, d, e, f, g, h, a, b, 38, w6);
295         w7 = SIGMA1_256(w5) + w0 + SIGMA0_256(w8) + w7;
296         SHA256ROUND(b, c, d, e, f, g, h, a, 39, w7);
297         w8 = SIGMA1_256(w6) + w1 + SIGMA0_256(w9) + w8;
298         SHA256ROUND(a, b, c, d, e, f, g, h, 40, w8);
299         w9 = SIGMA1_256(w7) + w2 + SIGMA0_256(w10) + w9;
300         SHA256ROUND(h, a, b, c, d, e, f, g, 41, w9);
301         w10 = SIGMA1_256(w8) + w3 + SIGMA0_256(w11) + w10;
302         SHA256ROUND(g, h, a, b, c, d, e, f, 42, w10);
303         w11 = SIGMA1_256(w9) + w4 + SIGMA0_256(w12) + w11;
304         SHA256ROUND(f, g, h, a, b, c, d, e, 43, w11);
305         w12 = SIGMA1_256(w10) + w5 + SIGMA0_256(w13) + w12;
306         SHA256ROUND(e, f, g, h, a, b, c, d, 44, w12);
307         w13 = SIGMA1_256(w11) + w6 + SIGMA0_256(w14) + w13;
308         SHA256ROUND(d, e, f, g, h, a, b, c, 45, w13);
309         w14 = SIGMA1_256(w12) + w7 + SIGMA0_256(w15) + w14;
310         SHA256ROUND(c, d, e, f, g, h, a, b, 46, w14);
311         w15 = SIGMA1_256(w13) + w8 + SIGMA0_256(w0) + w15;
312         SHA256ROUND(b, c, d, e, f, g, h, a, 47, w15);

314         w0 = SIGMA1_256(w14) + w9 + SIGMA0_256(w1) + w0;
```



```
315         SHA256ROUND(a, b, c, d, e, f, g, h, 48, w0);
316         w1 = SIGMA1_256(w15) + w10 + SIGMA0_256(w2) + w1;
317         SHA256ROUND(h, a, b, c, d, e, f, g, 49, w1);
318         w2 = SIGMA1_256(w0) + w11 + SIGMA0_256(w3) + w2;
319         SHA256ROUND(g, h, a, b, c, d, e, f, 50, w2);
320         w3 = SIGMA1_256(w1) + w12 + SIGMA0_256(w4) + w3;
321         SHA256ROUND(f, g, h, a, b, c, d, e, 51, w3);
322         w4 = SIGMA1_256(w2) + w13 + SIGMA0_256(w5) + w4;
323         SHA256ROUND(e, f, g, h, a, b, c, d, 52, w4);
324         w5 = SIGMA1_256(w3) + w14 + SIGMA0_256(w6) + w5;
325         SHA256ROUND(d, e, f, g, h, a, b, c, 53, w5);
326         w6 = SIGMA1_256(w4) + w15 + SIGMA0_256(w7) + w6;
327         SHA256ROUND(c, d, e, f, g, h, a, b, 54, w6);
328         w7 = SIGMA1_256(w5) + w0 + SIGMA0_256(w8) + w7;
329         SHA256ROUND(b, c, d, e, f, g, h, a, 55, w7);
330         w8 = SIGMA1_256(w6) + w1 + SIGMA0_256(w9) + w8;
331         SHA256ROUND(a, b, c, d, e, f, g, h, 56, w8);
332         w9 = SIGMA1_256(w7) + w2 + SIGMA0_256(w10) + w9;
333         SHA256ROUND(h, a, b, c, d, e, f, g, 57, w9);
334         w10 = SIGMA1_256(w8) + w3 + SIGMA0_256(w11) + w10;
335         SHA256ROUND(g, h, a, b, c, d, e, f, 58, w10);
336         w11 = SIGMA1_256(w9) + w4 + SIGMA0_256(w12) + w11;
337         SHA256ROUND(f, g, h, a, b, c, d, e, 59, w11);
338         w12 = SIGMA1_256(w10) + w5 + SIGMA0_256(w13) + w12;
339         SHA256ROUND(e, f, g, h, a, b, c, d, 60, w12);
340         w13 = SIGMA1_256(w11) + w6 + SIGMA0_256(w14) + w13;
341         SHA256ROUND(d, e, f, g, h, a, b, c, 61, w13);
342         w14 = SIGMA1_256(w12) + w7 + SIGMA0_256(w15) + w14;
343         SHA256ROUND(c, d, e, f, g, h, a, b, 62, w14);
344         w15 = SIGMA1_256(w13) + w8 + SIGMA0_256(w0) + w15;
345         SHA256ROUND(b, c, d, e, f, g, h, a, 63, w15);

347         ctx->state.s32[0] += a;
348         ctx->state.s32[1] += b;
349         ctx->state.s32[2] += c;
350         ctx->state.s32[3] += d;
351         ctx->state.s32[4] += e;
352         ctx->state.s32[5] += f;
353         ctx->state.s32[6] += g;
354         ctx->state.s32[7] += h;
355 }


358 /* SHA384 and SHA512 Transform */

360 static void
361 SHA512Transform(SHA2_CTX *ctx, const uint8_t *blk)
362 {

364         uint64_t a = ctx->state.s64[0];
365         uint64_t b = ctx->state.s64[1];
366         uint64_t c = ctx->state.s64[2];
367         uint64_t d = ctx->state.s64[3];
368         uint64_t e = ctx->state.s64[4];
369         uint64_t f = ctx->state.s64[5];
370         uint64_t g = ctx->state.s64[6];
371         uint64_t h = ctx->state.s64[7];

373         uint64_t w0, w1, w2, w3, w4, w5, w6, w7;
374         uint64_t w8, w9, w10, w11, w12, w13, w14, w15;
375         uint64_t T1, T2;

377 #if     defined(__sparc)
378         static const uint64_t sha512_consts[] = {
379                 SHA512_CONST_0, SHA512_CONST_1, SHA512_CONST_2,
380                 SHA512_CONST_3, SHA512_CONST_4, SHA512_CONST_5,
```

```
381                 SHA512_CONST_6, SHA512_CONST_7, SHA512_CONST_8,
382                 SHA512_CONST_9, SHA512_CONST_10, SHA512_CONST_11,
383                 SHA512_CONST_12, SHA512_CONST_13, SHA512_CONST_14,
384                 SHA512_CONST_15, SHA512_CONST_16, SHA512_CONST_17,
385                 SHA512_CONST_18, SHA512_CONST_19, SHA512_CONST_20,
386                 SHA512_CONST_21, SHA512_CONST_22, SHA512_CONST_23,
387                 SHA512_CONST_24, SHA512_CONST_25, SHA512_CONST_26,
388                 SHA512_CONST_27, SHA512_CONST_28, SHA512_CONST_29,
389                 SHA512_CONST_30, SHA512_CONST_31, SHA512_CONST_32,
390                 SHA512_CONST_33, SHA512_CONST_34, SHA512_CONST_35,
391                 SHA512_CONST_36, SHA512_CONST_37, SHA512_CONST_38,
392                 SHA512_CONST_39, SHA512_CONST_40, SHA512_CONST_41,
393                 SHA512_CONST_42, SHA512_CONST_43, SHA512_CONST_44,
394                 SHA512_CONST_45, SHA512_CONST_46, SHA512_CONST_47,
395                 SHA512_CONST_48, SHA512_CONST_49, SHA512_CONST_50,
396                 SHA512_CONST_51, SHA512_CONST_52, SHA512_CONST_53,
397                 SHA512_CONST_54, SHA512_CONST_55, SHA512_CONST_56,
398                 SHA512_CONST_57, SHA512_CONST_58, SHA512_CONST_59,
399                 SHA512_CONST_60, SHA512_CONST_61, SHA512_CONST_62,
400                 SHA512_CONST_63, SHA512_CONST_64, SHA512_CONST_65,
401                 SHA512_CONST_66, SHA512_CONST_67, SHA512_CONST_68,
402                 SHA512_CONST_69, SHA512_CONST_70, SHA512_CONST_71,
403                 SHA512_CONST_72, SHA512_CONST_73, SHA512_CONST_74,
404                 SHA512_CONST_75, SHA512_CONST_76, SHA512_CONST_77,
405                 SHA512_CONST_78, SHA512_CONST_79
406         };
407 #endif  /* __sparc */
401 #endif


410         if ((uintptr_t)blk & 0x7) {             /* not 8-byte aligned? */
411                 bcopy(blk, ctx->buf_un.buf64,  sizeof (ctx->buf_un.buf64));
412                 blk = (uint8_t *)ctx->buf_un.buf64;
413         }

415         /* LINTED E_BAD_PTR_CAST_ALIGN */
416         w0 =  LOAD_BIG_64(blk + 8 * 0);
417         SHA512ROUND(a, b, c, d, e, f, g, h, 0, w0);
418         /* LINTED E_BAD_PTR_CAST_ALIGN */
419         w1 =  LOAD_BIG_64(blk + 8 * 1);
420         SHA512ROUND(h, a, b, c, d, e, f, g, 1, w1);
421         /* LINTED E_BAD_PTR_CAST_ALIGN */
422         w2 =  LOAD_BIG_64(blk + 8 * 2);
423         SHA512ROUND(g, h, a, b, c, d, e, f, 2, w2);
424         /* LINTED E_BAD_PTR_CAST_ALIGN */
425         w3 =  LOAD_BIG_64(blk + 8 * 3);
426         SHA512ROUND(f, g, h, a, b, c, d, e, 3, w3);
427         /* LINTED E_BAD_PTR_CAST_ALIGN */
428         w4 =  LOAD_BIG_64(blk + 8 * 4);
429         SHA512ROUND(e, f, g, h, a, b, c, d, 4, w4);
430         /* LINTED E_BAD_PTR_CAST_ALIGN */
431         w5 =  LOAD_BIG_64(blk + 8 * 5);
432         SHA512ROUND(d, e, f, g, h, a, b, c, 5, w5);
433         /* LINTED E_BAD_PTR_CAST_ALIGN */
434         w6 =  LOAD_BIG_64(blk + 8 * 6);
435         SHA512ROUND(c, d, e, f, g, h, a, b, 6, w6);
436         /* LINTED E_BAD_PTR_CAST_ALIGN */
437         w7 =  LOAD_BIG_64(blk + 8 * 7);
438         SHA512ROUND(b, c, d, e, f, g, h, a, 7, w7);
439         /* LINTED E_BAD_PTR_CAST_ALIGN */
440         w8 =  LOAD_BIG_64(blk + 8 * 8);
441         SHA512ROUND(a, b, c, d, e, f, g, h, 8, w8);
442         /* LINTED E_BAD_PTR_CAST_ALIGN */
443         w9 =  LOAD_BIG_64(blk + 8 * 9);
444         SHA512ROUND(h, a, b, c, d, e, f, g, 9, w9);
445         /* LINTED E_BAD_PTR_CAST_ALIGN */
```



```
446         w10 =  LOAD_BIG_64(blk + 8 * 10);
447         SHA512ROUND(g, h, a, b, c, d, e, f, 10, w10);
448         /* LINTED E_BAD_PTR_CAST_ALIGN */
449         w11 =  LOAD_BIG_64(blk + 8 * 11);
450         SHA512ROUND(f, g, h, a, b, c, d, e, 11, w11);
451         /* LINTED E_BAD_PTR_CAST_ALIGN */
452         w12 =  LOAD_BIG_64(blk + 8 * 12);
453         SHA512ROUND(e, f, g, h, a, b, c, d, 12, w12);
454         /* LINTED E_BAD_PTR_CAST_ALIGN */
455         w13 =  LOAD_BIG_64(blk + 8 * 13);
456         SHA512ROUND(d, e, f, g, h, a, b, c, 13, w13);
457         /* LINTED E_BAD_PTR_CAST_ALIGN */
458         w14 =  LOAD_BIG_64(blk + 8 * 14);
459         SHA512ROUND(c, d, e, f, g, h, a, b, 14, w14);
460         /* LINTED E_BAD_PTR_CAST_ALIGN */
461         w15 =  LOAD_BIG_64(blk + 8 * 15);
462         SHA512ROUND(b, c, d, e, f, g, h, a, 15, w15);

464         w0 = SIGMA1(w14) + w9 + SIGMA0(w1) + w0;
465         SHA512ROUND(a, b, c, d, e, f, g, h, 16, w0);
466         w1 = SIGMA1(w15) + w10 + SIGMA0(w2) + w1;
467         SHA512ROUND(h, a, b, c, d, e, f, g, 17, w1);
468         w2 = SIGMA1(w0) + w11 + SIGMA0(w3) + w2;
469         SHA512ROUND(g, h, a, b, c, d, e, f, 18, w2);
470         w3 = SIGMA1(w1) + w12 + SIGMA0(w4) + w3;
471         SHA512ROUND(f, g, h, a, b, c, d, e, 19, w3);
472         w4 = SIGMA1(w2) + w13 + SIGMA0(w5) + w4;
473         SHA512ROUND(e, f, g, h, a, b, c, d, 20, w4);
474         w5 = SIGMA1(w3) + w14 + SIGMA0(w6) + w5;
475         SHA512ROUND(d, e, f, g, h, a, b, c, 21, w5);
476         w6 = SIGMA1(w4) + w15 + SIGMA0(w7) + w6;
477         SHA512ROUND(c, d, e, f, g, h, a, b, 22, w6);
478         w7 = SIGMA1(w5) + w0 + SIGMA0(w8) + w7;
479         SHA512ROUND(b, c, d, e, f, g, h, a, 23, w7);
480         w8 = SIGMA1(w6) + w1 + SIGMA0(w9) + w8;
481         SHA512ROUND(a, b, c, d, e, f, g, h, 24, w8);
482         w9 = SIGMA1(w7) + w2 + SIGMA0(w10) + w9;
483         SHA512ROUND(h, a, b, c, d, e, f, g, 25, w9);
484         w10 = SIGMA1(w8) + w3 + SIGMA0(w11) + w10;
485         SHA512ROUND(g, h, a, b, c, d, e, f, 26, w10);
486         w11 = SIGMA1(w9) + w4 + SIGMA0(w12) + w11;
487         SHA512ROUND(f, g, h, a, b, c, d, e, 27, w11);
488         w12 = SIGMA1(w10) + w5 + SIGMA0(w13) + w12;
489         SHA512ROUND(e, f, g, h, a, b, c, d, 28, w12);
490         w13 = SIGMA1(w11) + w6 + SIGMA0(w14) + w13;
491         SHA512ROUND(d, e, f, g, h, a, b, c, 29, w13);
492         w14 = SIGMA1(w12) + w7 + SIGMA0(w15) + w14;
493         SHA512ROUND(c, d, e, f, g, h, a, b, 30, w14);
494         w15 = SIGMA1(w13) + w8 + SIGMA0(w0) + w15;
495         SHA512ROUND(b, c, d, e, f, g, h, a, 31, w15);

497         w0 = SIGMA1(w14) + w9 + SIGMA0(w1) + w0;
498         SHA512ROUND(a, b, c, d, e, f, g, h, 32, w0);
499         w1 = SIGMA1(w15) + w10 + SIGMA0(w2) + w1;
500         SHA512ROUND(h, a, b, c, d, e, f, g, 33, w1);
501         w2 = SIGMA1(w0) + w11 + SIGMA0(w3) + w2;
502         SHA512ROUND(g, h, a, b, c, d, e, f, 34, w2);
503         w3 = SIGMA1(w1) + w12 + SIGMA0(w4) + w3;
504         SHA512ROUND(f, g, h, a, b, c, d, e, 35, w3);
505         w4 = SIGMA1(w2) + w13 + SIGMA0(w5) + w4;
506         SHA512ROUND(e, f, g, h, a, b, c, d, 36, w4);
507         w5 = SIGMA1(w3) + w14 + SIGMA0(w6) + w5;
508         SHA512ROUND(d, e, f, g, h, a, b, c, 37, w5);
509         w6 = SIGMA1(w4) + w15 + SIGMA0(w7) + w6;
510         SHA512ROUND(c, d, e, f, g, h, a, b, 38, w6);
511         w7 = SIGMA1(w5) + w0 + SIGMA0(w8) + w7;
```

```
        SHA512ROUND(b, c, d, e, f, g, h, a, 39, w7);
        w8 = SIGMA1(w6) + w1 + SIGMA0(w9) + w8;
        SHA512ROUND(a, b, c, d, e, f, g, h, 40, w8);
        w9 = SIGMA1(w7) + w2 + SIGMA0(w10) + w9;
        SHA512ROUND(h, a, b, c, d, e, f, g, 41, w9);
        w10 = SIGMA1(w8) + w3 + SIGMA0(w11) + w10;
        SHA512ROUND(g, h, a, b, c, d, e, f, 42, w10);
        w11 = SIGMA1(w9) + w4 + SIGMA0(w12) + w11;
        SHA512ROUND(f, g, h, a, b, c, d, e, 43, w11);
        w12 = SIGMA1(w10) + w5 + SIGMA0(w13) + w12;
        SHA512ROUND(e, f, g, h, a, b, c, d, 44, w12);
        w13 = SIGMA1(w11) + w6 + SIGMA0(w14) + w13;
        SHA512ROUND(d, e, f, g, h, a, b, c, 45, w13);
        w14 = SIGMA1(w12) + w7 + SIGMA0(w15) + w14;
        SHA512ROUND(c, d, e, f, g, h, a, b, 46, w14);
        w15 = SIGMA1(w13) + w8 + SIGMA0(w0) + w15;
        SHA512ROUND(b, c, d, e, f, g, h, a, 47, w15);

        w0 = SIGMA1(w14) + w9 + SIGMA0(w1) + w0;
        SHA512ROUND(a, b, c, d, e, f, g, h, 48, w0);
        w1 = SIGMA1(w15) + w10 + SIGMA0(w2) + w1;
        SHA512ROUND(h, a, b, c, d, e, f, g, 49, w1);
        w2 = SIGMA1(w0) + w11 + SIGMA0(w3) + w2;
        SHA512ROUND(g, h, a, b, c, d, e, f, 50, w2);
        w3 = SIGMA1(w1) + w12 + SIGMA0(w4) + w3;
        SHA512ROUND(f, g, h, a, b, c, d, e, 51, w3);
        w4 = SIGMA1(w2) + w13 + SIGMA0(w5) + w4;
        SHA512ROUND(e, f, g, h, a, b, c, d, 52, w4);
        w5 = SIGMA1(w3) + w14 + SIGMA0(w6) + w5;
        SHA512ROUND(d, e, f, g, h, a, b, c, 53, w5);
        w6 = SIGMA1(w4) + w15 + SIGMA0(w7) + w6;
        SHA512ROUND(c, d, e, f, g, h, a, b, 54, w6);
        w7 = SIGMA1(w5) + w0 + SIGMA0(w8) + w7;
        SHA512ROUND(b, c, d, e, f, g, h, a, 55, w7);
        w8 = SIGMA1(w6) + w1 + SIGMA0(w9) + w8;
        SHA512ROUND(a, b, c, d, e, f, g, h, 56, w8);
        w9 = SIGMA1(w7) + w2 + SIGMA0(w10) + w9;
        SHA512ROUND(h, a, b, c, d, e, f, g, 57, w9);
        w10 = SIGMA1(w8) + w3 + SIGMA0(w11) + w10;
        SHA512ROUND(g, h, a, b, c, d, e, f, 58, w10);
        w11 = SIGMA1(w9) + w4 + SIGMA0(w12) + w11;
        SHA512ROUND(f, g, h, a, b, c, d, e, 59, w11);
        w12 = SIGMA1(w10) + w5 + SIGMA0(w13) + w12;
        SHA512ROUND(e, f, g, h, a, b, c, d, 60, w12);
        w13 = SIGMA1(w11) + w6 + SIGMA0(w14) + w13;
        SHA512ROUND(d, e, f, g, h, a, b, c, 61, w13);
        w14 = SIGMA1(w12) + w7 + SIGMA0(w15) + w14;
        SHA512ROUND(c, d, e, f, g, h, a, b, 62, w14);
        w15 = SIGMA1(w13) + w8 + SIGMA0(w0) + w15;
        SHA512ROUND(b, c, d, e, f, g, h, a, 63, w15);

        w0 = SIGMA1(w14) + w9 + SIGMA0(w1) + w0;
        SHA512ROUND(a, b, c, d, e, f, g, h, 64, w0);
        w1 = SIGMA1(w15) + w10 + SIGMA0(w2) + w1;
        SHA512ROUND(h, a, b, c, d, e, f, g, 65, w1);
        w2 = SIGMA1(w0) + w11 + SIGMA0(w3) + w2;
        SHA512ROUND(g, h, a, b, c, d, e, f, 66, w2);
        w3 = SIGMA1(w1) + w12 + SIGMA0(w4) + w3;
        SHA512ROUND(f, g, h, a, b, c, d, e, 67, w3);
        w4 = SIGMA1(w2) + w13 + SIGMA0(w5) + w4;
        SHA512ROUND(e, f, g, h, a, b, c, d, 68, w4);
        w5 = SIGMA1(w3) + w14 + SIGMA0(w6) + w5;
        SHA512ROUND(d, e, f, g, h, a, b, c, 69, w5);
        w6 = SIGMA1(w4) + w15 + SIGMA0(w7) + w6;
        SHA512ROUND(c, d, e, f, g, h, a, b, 70, w6);
        w7 = SIGMA1(w5) + w0 + SIGMA0(w8) + w7;
```



```
        SHA512ROUND(b, c, d, e, f, g, h, a, 71, w7);
        w8 = SIGMA1(w6) + w1 + SIGMA0(w9) + w8;
        SHA512ROUND(a, b, c, d, e, f, g, h, 72, w8);
        w9 = SIGMA1(w7) + w2 + SIGMA0(w10) + w9;
        SHA512ROUND(h, a, b, c, d, e, f, g, 73, w9);
        w10 = SIGMA1(w8) + w3 + SIGMA0(w11) + w10;
        SHA512ROUND(g, h, a, b, c, d, e, f, 74, w10);
        w11 = SIGMA1(w9) + w4 + SIGMA0(w12) + w11;
        SHA512ROUND(f, g, h, a, b, c, d, e, 75, w11);
        w12 = SIGMA1(w10) + w5 + SIGMA0(w13) + w12;
        SHA512ROUND(e, f, g, h, a, b, c, d, 76, w12);
        w13 = SIGMA1(w11) + w6 + SIGMA0(w14) + w13;
        SHA512ROUND(d, e, f, g, h, a, b, c, 77, w13);
        w14 = SIGMA1(w12) + w7 + SIGMA0(w15) + w14;
        SHA512ROUND(c, d, e, f, g, h, a, b, 78, w14);
        w15 = SIGMA1(w13) + w8 + SIGMA0(w0) + w15;
        SHA512ROUND(b, c, d, e, f, g, h, a, 79, w15);

        ctx->state.s64[0] += a;
        ctx->state.s64[1] += b;
        ctx->state.s64[2] += c;
        ctx->state.s64[3] += d;
        ctx->state.s64[4] += e;
        ctx->state.s64[5] += f;
        ctx->state.s64[6] += g;
        ctx->state.s64[7] += h;

}
#endif  /* !__amd64 */


/*
 * Encode()
 *
 * purpose: to convert a list of numbers from little endian to big endian
 *   input: uint8_t *   : place to store the converted big endian numbers
 *          uint32_t *  : place to get numbers to convert from
 *          size_t      : the length of the input in bytes
 *  output: void
 */

static void
Encode(uint8_t *_RESTRICT_KYWD output, uint32_t *_RESTRICT_KYWD input,
    size_t len)
{
        size_t          i, j;

#if     defined(__sparc)
        if (IS_P2ALIGNED(output, sizeof (uint32_t))) {
                for (i = 0, j = 0; j < len; i++, j += 4) {
                        /* LINTED: pointer alignment */
                        *((uint32_t *)(output + j)) = input[i];
                }
        } else {
#endif  /* little endian -- will work on big endian, but slowly */
                for (i = 0, j = 0; j < len; i++, j += 4) {
                        output[j]       = (input[i] >> 24) & 0xff;
                        output[j + 1]   = (input[i] >> 16) & 0xff;
                        output[j + 2]   = (input[i] >>  8) & 0xff;
                        output[j + 3]   = input[i] & 0xff;
                }
#if     defined(__sparc)
        }
#endif
}
```

unchanged_portion_omitted_

```
#endif /* _KERNEL */

/*
 * SHA2Update()
 *
 * purpose: continues an sha2 digest operation, using the message block
 *          to update the context.
 *   input: SHA2_CTX *  : the context to update
 *          void *      : the message block
 *          size_t      : the length of the message block, in bytes
 *          size_t    : the length of the message block in bytes
 *  output: void
 */

void
SHA2Update(SHA2_CTX *ctx, const void *inptr, size_t input_len)
{
        uint32_t        i, buf_index, buf_len, buf_limit;
        const uint8_t   *input = inptr;
        uint32_t        algotype = ctx->algotype;
#if defined(__amd64)
        uint32_t        block_count;
#endif  /* !__amd64 */


        /* check for noop */
        if (input_len == 0)
                return;

        if (algotype <= SHA256_HMAC_GEN_MECH_INFO_TYPE) {
        if (ctx->algotype <= SHA256_HMAC_GEN_MECH_INFO_TYPE) {
                buf_limit = 64;

                /* compute number of bytes mod 64 */
                buf_index = (ctx->count.c32[1] >> 3) & 0x3F;

                /* update number of bits */
                if ((ctx->count.c32[1] += (input_len << 3)) < (input_len << 3))
                        ctx->count.c32[0]++;

                ctx->count.c32[0] += (input_len >> 29);

        } else {
                buf_limit = 128;

                /* compute number of bytes mod 128 */
                buf_index = (ctx->count.c64[1] >> 3) & 0x7F;

                /* update number of bits */
                if ((ctx->count.c64[1] += (input_len << 3)) < (input_len << 3))
                        ctx->count.c64[0]++;

                ctx->count.c64[0] += (input_len >> 29);
        }

        buf_len = buf_limit - buf_index;

        /* transform as many times as possible */
        i = 0;
        if (input_len >= buf_len) {

                /*
                 * general optimization:
                 *
                 * only do initial bcopy() and SHA2Transform() if
```



```
                 * buf_index != 0.  if buf_index == 0, we're just
                 * wasting our time doing the bcopy() since there
                 * wasn't any data left over from a previous call to
                 * SHA2Update().
                 */
                if (buf_index) {
                        bcopy(input, &ctx->buf_un.buf8[buf_index], buf_len);
                        if (algotype <= SHA256_HMAC_GEN_MECH_INFO_TYPE)
                        if (ctx->algotype <= SHA256_HMAC_GEN_MECH_INFO_TYPE)
                                SHA256Transform(ctx, ctx->buf_un.buf8);
                        else
                                SHA512Transform(ctx, ctx->buf_un.buf8);

                        i = buf_len;
                }

#if !defined(__amd64)
                if (algotype <= SHA256_HMAC_GEN_MECH_INFO_TYPE) {

                        for (; i + buf_limit - 1 < input_len; i += buf_limit) {
                        if (ctx->algotype <= SHA256_HMAC_GEN_MECH_INFO_TYPE)
                                SHA256Transform(ctx, &input[i]);
                        }
                } else {
                        for (; i + buf_limit - 1 < input_len; i += buf_limit) {
                        else
                                SHA512Transform(ctx, &input[i]);
                        }
                }

#else
                if (algotype <= SHA256_HMAC_GEN_MECH_INFO_TYPE) {
                        block_count = (input_len - i) >> 6;
                        if (block_count > 0) {
                                SHA256TransformBlocks(ctx, &input[i],
                                    block_count);
                                i += block_count << 6;
                        }
                } else {
                        block_count = (input_len - i) >> 7;
                        if (block_count > 0) {
                                SHA512TransformBlocks(ctx, &input[i],
                                    block_count);
                                i += block_count << 7;
                        }
                }
#endif  /* !__amd64 */

                /*
                 * general optimization:
                 *
                 * if i and input_len are the same, return now instead
                 * of calling bcopy(), since the bcopy() in this case
                 * will be an expensive noop.
                 * will be an expensive nop.
                 */

                if (input_len == i)
                        return;

                buf_index = 0;
        }

        /* buffer remaining input */
        bcopy(&input[i], &ctx->buf_un.buf8[buf_index], input_len - i);
}
```

```
 876 /*
 877  * SHA2Final()
 878  *
 879  * purpose: ends an sha2 digest operation, finalizing the message digest and
 880  *          zeroing the context.
 881  *   input: uchar_t *   : a buffer to store the digest
 848  *   input: uchar_t *   : a buffer to store the digest in
 882  *                      : The function actually uses void* because many
 883  *                      : callers pass things other than uchar_t here.
 884  *          SHA2_CTX *  : the context to finalize, save, and zero
 885  *  output: void
 886  */

 888 void
 889 SHA2Final(void *digest, SHA2_CTX *ctx)
 890 {
 891         uint8_t         bitcount_be[sizeof (ctx->count.c32)];
 892         uint8_t         bitcount_be64[sizeof (ctx->count.c64)];
 893         uint32_t        index;
 894         uint32_t        algotype = ctx->algotype;

 896         if (algotype <= SHA256_HMAC_GEN_MECH_INFO_TYPE) {

 863         if (ctx->algotype <= SHA256_HMAC_GEN_MECH_INFO_TYPE) {
 897                 index  = (ctx->count.c32[1] >> 3) & 0x3f;
 898                 Encode(bitcount_be, ctx->count.c32, sizeof (bitcount_be));
 899                 SHA2Update(ctx, PADDING, ((index < 56) ? 56 : 120) - index);
 900                 SHA2Update(ctx, bitcount_be, sizeof (bitcount_be));
 901                 Encode(digest, ctx->state.s32, sizeof (ctx->state.s32));

 903         } else {
 904                 index  = (ctx->count.c64[1] >> 3) & 0x7f;
 905                 Encode64(bitcount_be64, ctx->count.c64,
 906                     sizeof (bitcount_be64));
 907                 SHA2Update(ctx, PADDING, ((index < 112) ? 112 : 240) - index);
 908                 SHA2Update(ctx, bitcount_be64, sizeof (bitcount_be64));
 909                 if (algotype <= SHA384_HMAC_GEN_MECH_INFO_TYPE) {
 876                 if (ctx->algotype <= SHA384_HMAC_GEN_MECH_INFO_TYPE) {
 910                         ctx->state.s64[6] = ctx->state.s64[7] = 0;
 911                         Encode64(digest, ctx->state.s64,
 912                             sizeof (uint64_t) * 6);
 913                 } else
 914                         Encode64(digest, ctx->state.s64,
 915                             sizeof (ctx->state.s64));
 916         }

 918         /* zeroize sensitive information */
 919         bzero(ctx, sizeof (*ctx));
 920 }
```

*unchanged_portion_omitted_*

```
**********************************************************
    3404 Thu Mar 20 14:23:54 2008
new/usr/src/lib/libmd/Makefile.com
6665607 Need a SHA256/SHA384/SHA512 implementation optimized for 64-bit x86
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# ident "@(#)Makefile.com       1.5     08/03/05 SMI"
# ident "@(#)Makefile.com       1.4     08/03/02 SMI"
#

# $LIBRARY is set in lower makefiles so we can have platform and
# processor optimised versions of this library via libmd_psr and libmd_hwcapN

#LIBRARY= libmd.a
VERS= .1

OBJECTS= md4.o md5.o $(MD5_PSR_OBJECTS) sha1.o $(SHA1_PSR_OBJECTS) \
                sha2.o $(SHA2_PSR_OBJECTS)
OBJECTS= md4.o md5.o $(MD5_PSR_OBJECTS) sha1.o $(SHA1_PSR_OBJECTS) sha2.o

# Use $(SRC) to include makefiles rather than ../../ because the
# platform subdirs are one level deeper so it would be ../../../ for them
include $(SRC)/lib/Makefile.lib
include $(SRC)/lib/Makefile.rootfs

LIBS =          $(DYNLIB) $(LINTLIB)
SRCS = \
        $(COMDIR)/md4/md4.c \
        $(COMDIR)/md5/md5.c \
        $(COMDIR)/sha1/sha1.c \
        $(COMDIR)/sha2/sha2.c

COMDIR= $(SRC)/common/crypto

$(LINTLIB) :=   SRCS = $(SRCDIR)/$(LINTSRC)
LDLIBS +=       -lc

SRCDIR =        ../common
COMDIR =        $(SRC)/common/crypto

CFLAGS += $(CCVERBOSE) $(C_BIGPICFLAGS)
CFLAGS64 += $(C_BIGPICFLAGS)
CPPFLAGS += -I$(SRCDIR)
```



```
# The md5 and sha1 code is very careful about data alignment
# but lint doesn't know that, so just shut lint up.
LINTFLAGS += -erroff=E_SUPPRESSION_DIRECTIVE_UNUSED
LINTFLAGS64 += -erroff=E_SUPPRESSION_DIRECTIVE_UNUSED


ROOTLINT= $(LINTSRC:%=$(ROOTLIBDIR)/%)

.KEEP_STATE:

all: $(LIBS) fnamecheck

lint: lintcheck

pics/%.o: $(COMDIR)/md4/%.c
        $(COMPILE.c) -I$(COMDIR)/md4 -o $@ $<
        $(POST_PROCESS_O)

pics/%.o: $(COMDIR)/md5/%.c
        $(COMPILE.c) -I$(COMDIR)/md5 $(INLINES) -o $@ $<
        $(POST_PROCESS_O)

pics/%.o: $(COMDIR)/sha1/%.c
        $(COMPILE.c) -I$(COMDIR)/sha1 -o $@ $<
        $(POST_PROCESS_O)

pics/%.o: $(COMDIR)/sha1/sparc/$(PLATFORM)/sha1_asm.s
        $(COMPILE.s) -P -DPIC -D_ASM -o $@ $<
        $(POST_PROCESS_O)

pics/%.o: $(COMDIR)/sha2/%.c
        $(COMPILE.c) -I$(COMDIR)/sha2 -o $@ $<
        $(POST_PROCESS_O)

#
# Used when building links in /platform/$(PLATFORM)/lib for libmd_psr.so.1
#

LIBMD_PSR_DIRS = $(LINKED_PLATFORMS:%=$(ROOT_PLAT_DIR)/%/lib)
LIBMD_PSR_LINKS = $(LINKED_PLATFORMS:%=$(ROOT_PLAT_DIR)/%/lib/$(MODULE))

LIBMD_PSR64_DIRS = $(LINKED_PLATFORMS:%=$(ROOT_PLAT_DIR)/%/lib/$(MACH64))
LIBMD_PSR64_LINKS = $(LINKED_PLATFORMS:%=$(ROOT_PLAT_DIR)/%/lib/$(MACH64)/$(MODU

INS.slink6 = $(RM) -r $@; $(SYMLINK) ../../$(PLATFORM)/lib/$(MODULE) $@ $(CHOWNL

INS.slink64 = $(RM) -r $@; $(SYMLINK) ../../../$(PLATFORM)/lib/$(MACH64)/$(MODUL

$(LIBMD_PSR_DIRS):
        -$(INS.dir.root.bin)

$(LIBMD_PSR_LINKS): $(LIBMD_PSR_DIRS)
        -$(INS.slink6)

$(LIBMD_PSR64_DIRS):
        -$(INS.dir.root.bin)

$(LIBMD_PSR64_LINKS): $(LIBMD_PSR64_DIRS)
        -$(INS.slink64)

include $(SRC)/lib/Makefile.targ
```

```
**********************************************************
    1686 Thu Mar 20 14:23:55 2008
new/usr/src/lib/libmd/amd64/Makefile
6665607 Need a SHA256/SHA384/SHA512 implementation optimized for 64-bit x86
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# ident "@(#)Makefile   1.6     08/03/20 SMI"
# ident "@(#)Makefile   1.3     08/03/02 SMI"
#

LIBRARY= libmd.a

MD5_PSR_OBJECTS = md5_amd64.o
SHA1_PSR_OBJECTS = sha1-x86_64.o
SHA2_PSR_OBJECTS = sha512-x86_64.o sha256-x86_64.o

include ../Makefile.com
include $(SRC)/lib/Makefile.lib.64

CLEANFILES += md5_amd64.s sha1-x86_64.s sha512-x86_64.s sha256-x86_64.s
CLEANFILES += md5_amd64.s sha1-x86_64.s

# This prevents <sys/asm_linkage.h> from including C source:
AS_CPPFLAGS += -D_ASM

install: all $(ROOTLIBS64) $(ROOTLINKS64) $(ROOTLINT64)

pics/%.o: %.s
pics/md5_amd64.o: md5_amd64.s
        $(COMPILE.s) -o $@ ${@F:.o=.s}
        $(POST_PROCESS_O)

pics/sha1-x86_64.o: sha1-x86_64.s
        $(COMPILE.s) -o $@ ${@F:.o=.s}
        $(POST_PROCESS_O)

md5_amd64.s: $(COMDIR)/md5/amd64/md5_amd64.pl
        $(PERL) $? $@

sha1-x86_64.s: $(COMDIR)/sha1/amd64/sha1-x86_64.pl
        $(PERL) $? $@

sha512-x86_64.s: $(COMDIR)/sha2/amd64/sha512-x86_64.pl
```



```
        $(PERL) $? $@

sha256-x86_64.s: $(COMDIR)/sha2/amd64/sha512-x86_64.pl
        $(PERL) $? $@
```

```
**********************************************************
    6448 Thu Mar 20 14:23:57 2008
new/usr/src/lib/pkcs11/pkcs11_softtoken/Makefile.com
6665607 Need a SHA256/SHA384/SHA512 implementation optimized for 64-bit x86
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# ident "@(#)Makefile.com       1.6     08/03/05 SMI"
# ident "@(#)Makefile.com       1.5     08/02/26 SMI"
#
# lib/pkcs11/pkcs11_softtoken/Makefile.com
#

LIBRARY = pkcs11_softtoken.a
VERS= .1

LCL_OBJECTS = \
        softGeneral.o           \
        softSlotToken.o         \
        softSession.o           \
        softObject.o            \
        softDigest.o            \
        softSign.o              \
        softVerify.o            \
        softDualCrypt.o         \
        softKeys.o              \
        softRand.o              \
        softSessionUtil.o       \
        softDigestUtil.o        \
        softAttributeUtil.o     \
        softObjectUtil.o        \
        softDESCrypt.o          \
        softEncrypt.o           \
        softDecrypt.o           \
        softEncryptUtil.o       \
        softDecryptUtil.o       \
        softSignUtil.o          \
        softVerifyUtil.o        \
        softMAC.o               \
        softRSA.o               \
        softRandUtil.o          \
        softKeysUtil.o          \
        softARCFourCrypt.o      \
        softDSA.o               \
```



```
        softDH.o                \
        softAESCrypt.o          \
        softCrypt.o             \
        softKeystore.o          \
        softKeystoreUtil.o      \
        softSSL.o               \
        softASN1.o              \
        softBlowfishCrypt.o     \
        softEC.o

ASFLAGS = $(AS_PICFLAGS) -P -D__STDC__ -D_ASM $(CPPFLAGS)

AES_COBJECTS = aes_cbc_crypt.o  aes_impl.o
BLOWFISH_COBJECTS = blowfish_cbc_crypt.o blowfish_impl.o
ARCFOUR_COBJECTS = arcfour_crypt.o
DES_COBJECTS = des_cbc_crypt.o des_impl.o des_ks.o

ECC_COBJECTS = \
        ec.o ec2_163.o ec2_mont.o ecdecode.o ecl_mult.o ecp_384.o \
        ecp_jac.o ec2_193.o ecl.o ecp_192.o ecp_521.o \
        ecp_jm.o ec2_233.o ecl_curve.o ecp_224.o ecp_aff.o ecp_mont.o \
        ec2_aff.o ec_naf.o ecl_gf.o ecp_256.o oid.o secitem.o \
        ec2_test.o ecp_test.o

MPI_COBJECTS = mp_gf2m.o mpi.o mplogic.o mpmontg.o mpprime.o

RSA_COBJECTS = rsa_impl.o
BIGNUM_COBJECTS = bignumimpl.o

AES_OBJECTS = $(AES_COBJECTS) $(AES_PSR_OBJECTS)
BLOWFISH_OBJECTS = $(BLOWFISH_COBJECTS) $(BLOWFISH_PSR_OBJECTS)
ARCFOUR_OBJECTS = $(ARCFOUR_COBJECTS) $(ARCFOUR_PSR_OBJECTS)
DES_OBJECTS = $(DES_COBJECTS) $(DES_PSR_OBJECTS)

ECC_OBJECTS = $(ECC_COBJECTS) $(ECC_PSR_OBJECTS)
MPI_OBJECTS = $(MPI_COBJECTS) $(MPI_PSR_OBJECTS)
RSA_OBJECTS = $(RSA_COBJECTS) $(RSA_PSR_OBJECTS)
SHA1_OBJECTS = $(SHA1_COBJECTS) $(SHA1_PSR_OBJECTS)
SHA2_OBJECTS = $(SHA2_COBJECTS) $(SHA2_PSR_OBJECTS)
BIGNUM_OBJECTS = $(BIGNUM_COBJECTS) $(BIGNUM_PSR_OBJECTS)

BER_OBJECTS = bprint.o decode.o encode.o io.o

# Sparc userland uses a floating-point implementation of
# Montgomery multiply.  So, USE_FLOATING_POINT is defined here
# for Sparc targets.
#
# x86 does not use floating-point for the kernel or userland.
#
# Sparc has only one integer implementation of big_mul_add_vec()
# and friends, so these functions are called directly.
# So, HWCAP (HardWare CAPabilities) is not defined for Sparc.
#
# x86 has multiple integer implementations to choose from.
# Hardware features are tested at run time, just once,
# on first use.  So, big_mul_add_vec() and friends must be
# called through a function pointer.
#
# AMD64 has a 64x64->128 bit multiply instruction, which makes
# things even faster than i386 SSE2 instructions.  Since there
# is no run-time testing of features, as there is for SSE2,
# there is no need to call big_mul_add_vec() and friends through
# functions pointers, and so HWCAP is not defined.
#
# For now i386 and amd64 use the C code version of mont_mulf
```

```
125 OBJECTS = \
126         $(LCL_OBJECTS)          \
127         $(AES_OBJECTS)          \
128         $(BLOWFISH_OBJECTS)     \
129         $(ARCFOUR_OBJECTS)      \
130         $(DES_OBJECTS)          \
131         $(MPI_OBJECTS)          \
132         $(RSA_OBJECTS)          \
135         $(SHA1_OBJECTS)         \
136         $(SHA2_OBJECTS)         \
133         $(BIGNUM_OBJECTS)       \
134         $(BER_OBJECTS)          \
135         $(ECC_OBJECTS)

137 AESDIR=         $(SRC)/common/crypto/aes
138 BLOWFISHDIR=    $(SRC)/common/crypto/blowfish
139 ARCFOURDIR=     $(SRC)/common/crypto/arcfour
140 DESDIR=         $(SRC)/common/crypto/des
141 ECCDIR=         $(SRC)/common/crypto/ecc
142 MPIDIR=         $(SRC)/common/mpi
143 RSADIR=         $(SRC)/common/crypto/rsa
144 BIGNUMDIR=      $(SRC)/common/bignum
145 BERDIR=         ../../../libldap5/sources/ldap/ber

147 include $(SRC)/lib/Makefile.lib

149 #       set signing mode
150 POST_PROCESS_SO +=      ; $(ELFSIGN_CRYPTO)

152 SRCDIR= ../common

154 SRCS =  \
155         $(LCL_OBJECTS:%.o=$(SRCDIR)/%.c) \
156         $(AES_COBJECTS:%.o=$(AESDIR)/%.c) \
157         $(BLOWFISH_COBJECTS:%.o=$(BLOWFISHDIR)/%.c) \
158         $(ARCFOUR_COBJECTS:%.o=$(ARCFOURDIR)/%.c) \
159         $(DES_COBJECTS:%.o=$(DESDIR)/%.c) \
160         $(MPI_COBJECTS:%.o=$(MPIDIR)/%.c) \
161         $(RSA_COBJECTS:%.o=$(RSADIR)/%.c) \
166         $(SHA1_COBJECTS:%.o=$(SHA1DIR)/%.c) \
167         $(SHA2_COBJECTS:%.o=$(SHA2DIR)/%.c) \
162         $(BIGNUM_COBJECTS:%.o=$(BIGNUMDIR)/%.c) \
163         $(BIGNUM_PSR_SRCS) \
164         $(ECC_COBJECTS:%.o=$(ECCDIR)/%.c)

166 # libelfsign needs a static pkcs11_softtoken
167 LIBS    =       $(DYNLIB)
168 LDLIBS  +=      -lc -lmd -lcryptoutil

170 CFLAGS  +=      $(CCVERBOSE)
171 CPPFLAGS += -I$(AESDIR) -I$(BLOWFISHDIR) -I$(ARCFOURDIR) -I$(DESDIR) \
172             -I$(ECCDIR) -I$(MPIDIR) -I$(RSADIR) -I$(SRCDIR) -I$(BIGNUMDIR) \
173             -D_POSIX_PTHREAD_SEMANTICS -DMP_API_COMPATIBLE \
174             -DNSS_ECC_MORE_THAN_SUITE_B

176 LINTFLAGS64 += -errchk=longptr64

178 ROOTLIBDIR=     $(ROOT)/usr/lib/security
179 ROOTLIBDIR64=   $(ROOT)/usr/lib/security/$(MACH64)

181 LINTSRC = \
182         $(LCL_OBJECTS:%.o=$(SRCDIR)/%.c) \
183         $(AES_COBJECTS:%.o=$(AESDIR)/%.c) \
184         $(BLOWFISH_COBJECTS:%.o=$(BLOWFISHDIR)/%.c) \
185         $(ARCFOUR_COBJECTS:%.o=$(ARCFOURDIR)/%.c) \
186         $(DES_COBJECTS:%.o=$(DESDIR)/%.c) \
```



```
187         $(RSA_COBJECTS:%.o=$(RSADIR)/%.c) \
194         $(SHA1_COBJECTS:%.o=$(SHA1DIR)/%.c) \
195         $(SHA2_COBJECTS:%.o=$(SHA2DIR)/%.c) \
188         $(BIGNUM_COBJECTS:%.o=$(BIGNUMDIR)/%.c) \
189         $(BIGNUM_PSR_SRCS)

191 .KEEP_STATE:

193 all:    $(LIBS)

195 lint:   $$(LINTSRC)
196         $(LINT.c) $(LINTCHECKFLAGS) $(LINTSRC) $(LDLIBS)

198 pics/%.o:       $(AESDIR)/%.c
199         $(COMPILE.c) -o $@ $<
200         $(POST_PROCESS_O)

202 pics/%.o:       $(BLOWFISHDIR)/%.c
203         $(COMPILE.c) -o $@ $<
204         $(POST_PROCESS_O)

206 pics/%.o:       $(ARCFOURDIR)/%.c
207         $(COMPILE.c) -o $@ $<
208         $(POST_PROCESS_O)

210 pics/%.o:       $(DESDIR)/%.c
211         $(COMPILE.c) -o $@ $<
212         $(POST_PROCESS_O)

214 pics/%.o:       $(ECCDIR)/%.c
215         $(COMPILE.c) -o $@ $<
216         $(POST_PROCESS_O)

218 pics/%.o:       $(MPIDIR)/%.c
219         $(COMPILE.c) -o $@ $<
220         $(POST_PROCESS_O)

222 pics/%.o:       $(RSADIR)/%.c
223         $(COMPILE.c) -o $@ $<
224         $(POST_PROCESS_O)

234 pics/%.o:       $(SHA1DIR)/%.c
235         $(COMPILE.c) -o $@ $<
236         $(POST_PROCESS_O)

238 pics/%.o:       $(SHA2DIR)/%.c
239         $(COMPILE.c) -o $@ $<
240         $(POST_PROCESS_O)

226 pics/%.o:       $(BIGNUMDIR)/%.c
227         $(COMPILE.c) -o $@ $(BIGNUM_CFG) $<
228         $(POST_PROCESS_O)

230 pics/%.o:       $(BERDIR)/%.c
231         $(COMPILE.c) -o $@ $< -D_SOLARIS_SDK -I$(BERDIR) \
232                 -I../../../libldap5/include/ldap
233         $(POST_PROCESS_O)

235 include $(SRC)/lib/Makefile.targ
```

```
**********************************************************
    1948 Thu Mar 20 14:23:59 2008
new/usr/src/lib/pkcs11/pkcs11_softtoken/amd64/Makefile
6665607 Need a SHA256/SHA384/SHA512 implementation optimized for 64-bit x86
**********************************************************
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #
  21 #
  22 # Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
  23 # Use is subject to license terms.
  24 #
  25 # ident "@(#)Makefile   1.5     08/03/05 SMI"
  25 # ident "@(#)Makefile   1.4     08/02/26 SMI"
  26 #
  27 # lib/pkcs11/pkcs11_softtoken/amd64/Makefile

  29 AES_PSR_OBJECTS =
  30 ARCFOUR_PSR_OBJECTS =  arcfour_crypt_amd64.o
  31 DES_PSR_OBJECTS =
  32 RSA_PSR_OBJECTS =
  33 SHA1_PSR_OBJECTS =
  33 BIGNUM_PSR_OBJECTS = bignum_amd64.o bignum_amd64_asm.o
  34 BIGNUM_PSR_PICS = $(BIGNUM_PSR_OBJECTS:%=pics/%)
  35 BIGNUM_CFG = -DPSR_MUL
  36 BIGNUM_PSR_SRCS = \
  37         $(BIGNUMDIR)/amd64/bignum_amd64.c \
  38         $(BIGNUMDIR)/amd64/bignum_amd64_asm.s

  40 pics/bignum_amd64.o := amd64_COPTFLAG = -xO3

  42 include ../Makefile.com
  43 include ../../../Makefile.lib.64

  45 install: all $(ROOTLIBS64) $(ROOTLINKS64)

  47 $(BIGNUM_PSR_PICS) := CFLAGS += $(C_BIGPICFLAGS) $(BIGNUM_CFG)

  49 LINTFLAGS64 += $(BIGNUM_CFG)

  51 pics/arcfour_crypt_amd64.o: $(ARCFOURDIR)/amd64/arcfour_crypt_amd64.s
  52         $(COMPILE.s) -o $@ $(AS_BIGPICFLAGS) \
  53                 $(ARCFOURDIR)/amd64/arcfour_crypt_amd64.s
  54         $(POST_PROCESS_O)

  56 pics/%.o: $(BIGNUMDIR)/$(MACH64)/%.c
  57         $(COMPILE.c) -o $@ $(C_BIGPICFLAGS) $(BIGNUM_CFG) $<
  58         $(POST_PROCESS_O)
```



```
  60 pics/%.o: $(BIGNUMDIR)/$(MACH64)/%.s
  61         $(COMPILE.s) -o $@ $(AS_BIGPICFLAGS) $(BIGNUM_CFG) $<
  62         $(POST_PROCESS_O)
```

****
1605 Thu Mar 20 14:24:00 2008
new/usr/src/lib/pkcs11/pkcs11_softtoken/i386/Makefile
6665607 Need a SHA256/SHA384/SHA512 implementation optimized for 64-bit x86
****

```
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #
  21 #
  22 # Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
  23 # Use is subject to license terms.
  24 #
  25 # ident "@(#)Makefile   1.6     08/03/05 SMI"
  25 # ident "@(#)Makefile   1.5     08/02/26 SMI"
  26 #
  27 # lib/pkcs11/pkcs11_softtoken/i386/Makefile

  29 AES_PSR_OBJECTS =
  30 ARCFOUR_PSR_OBJECTS =
  31 DES_PSR_OBJECTS =
  32 RSA_PSR_OBJECTS =
  33 SHA1_PSR_OBJECTS =
  33 BIGNUM_PSR_OBJECTS = bignum_i386.o bignum_i386_asm.o
  34 BIGNUM_CFG = -DPSR_MUL -DHWCAP
  35 BIGNUM_PSR_SRCS = \
  36         $(BIGNUMDIR)/i386/bignum_i386.c \
  37         $(BIGNUMDIR)/i386/bignum_i386_asm.s

  39 include ../Makefile.com

  41 CPPFLAGS += -DMP_USE_UINT_DIGIT

  43 install: all $(ROOTLIBS) $(ROOTLINKS)

  45 DYNFLAGS += -M $(BIGNUMDIR)/i386/cap_mapfile

  47 pics/bignum_i386.o := COPTFLAG = -xO3

  49 pics/%.o:       $(BIGNUMDIR)/$(MACH)/%.c
  50         $(COMPILE.c) -o $@ $(BIGNUM_CFG) $<
  51         $(POST_PROCESS_O)

  53 pics/%.o:       $(BIGNUMDIR)/$(MACH)/%.s
  54         $(COMPILE.s) -o $@ $(BIGNUM_CFG) $<
  55         $(POST_PROCESS_O)
```

```
**********
    5864 Thu Mar 20 14:24:03 2008
new/usr/src/tools/opensolaris/license-list
6665607 Need a SHA256/SHA384/SHA512 implementation optimized for 64-bit x86
**********
   1 usr/closed/cmd/man/src/util/solbookv1/THIRDPARTYLICENSE
   2 usr/closed/cmd/ndmpd/LICENSE
   3 usr/closed/lib/smartcard/ocfserv/opencard/core/event/THIRDPARTYLICENSE
   4 usr/src/cmd/agents/snmp/THIRDPARTYLICENSE
   5 usr/src/cmd/ast/THIRDPARTYLICENSE
   6 usr/src/cmd/backup/dump/THIRDPARTYLICENSE
   7 usr/src/cmd/bnu/THIRDPARTYLICENSE
   8 usr/src/cmd/checkeq/THIRDPARTYLICENSE
   9 usr/src/cmd/checknr/THIRDPARTYLICENSE
  10 usr/src/cmd/cmd-inet/THIRDPARTYLICENSE.kcmd
  11 usr/src/cmd/cmd-inet/sbin/ifparse/THIRDPARTYLICENSE
  12 usr/src/cmd/cmd-inet/usr.bin/THIRDPARTYLICENSE.rcp
  13 usr/src/cmd/cmd-inet/usr.bin/THIRDPARTYLICENSE.rsh
  14 usr/src/cmd/cmd-inet/usr.bin/ftp/THIRDPARTYLICENSE
  15 usr/src/cmd/cmd-inet/usr.bin/pppd/THIRDPARTYLICENSE
  16 usr/src/cmd/cmd-inet/usr.bin/pppd/plugins/THIRDPARTYLICENSE.minconnect
  17 usr/src/cmd/cmd-inet/usr.bin/pppd/plugins/THIRDPARTYLICENSE.passwd
  18 usr/src/cmd/cmd-inet/usr.bin/pppdump/LICENSE.top
  19 usr/src/cmd/cmd-inet/usr.bin/rdist/THIRDPARTYLICENSE
  20 usr/src/cmd/cmd-inet/usr.bin/telnet/THIRDPARTYLICENSE
  21 usr/src/cmd/cmd-inet/usr.lib/in.mpathd/THIRDPARTYLICENSE
  22 usr/src/cmd/cmd-inet/usr.sbin/THIRDPARTYLICENSE.arp
  23 usr/src/cmd/cmd-inet/usr.sbin/THIRDPARTYLICENSE.comsat
  24 usr/src/cmd/cmd-inet/usr.sbin/THIRDPARTYLICENSE.rlogind
  25 usr/src/cmd/cmd-inet/usr.sbin/THIRDPARTYLICENSE.route
  26 usr/src/cmd/cmd-inet/usr.sbin/ifconfig/THIRDPARTYLICENSE
  27 usr/src/cmd/cmd-inet/usr.sbin/in.ftpd/LICENSE
  28 usr/src/cmd/cmd-inet/usr.sbin/in.rdisc/THIRDPARTYLICENSE
  29 usr/src/cmd/cmd-inet/usr.sbin/in.routed/THIRDPARTYLICENSE
  30 usr/src/cmd/cmd-inet/usr.sbin/traceroute/THIRDPARTYLICENSE
  31 usr/src/cmd/compress/THIRDPARTYLICENSE
  32 usr/src/cmd/csh/THIRDPARTYLICENSE
  33 usr/src/cmd/eeprom/THIRDPARTYLICENSE
  34 usr/src/cmd/eqn/THIRDPARTYLICENSE
  35 usr/src/cmd/fs.d/udfs/fsck/THIRDPARTYLICENSE
  36 usr/src/cmd/fs.d/ufs/THIRDPARTYLICENSE
  37 usr/src/cmd/ipf/tools/IPFILTER.LICENCE
  38 usr/src/cmd/lastcomm/THIRDPARTYLICENSE
  39 usr/src/cmd/ldap/THIRDPARTYLICENSE
  40 usr/src/cmd/look/THIRDPARTYLICENSE
  41 usr/src/cmd/lp/cmd/lptest/THIRDPARTYLICENSE
  42 usr/src/cmd/man/src/THIRDPARTYLICENSE
  43 usr/src/cmd/man/src/util/THIRDPARTYLICENSE
  44 usr/src/cmd/man/src/util/instant.src/THIRDPARTYLICENSE
  45 usr/src/cmd/man/src/util/nsgmls.src/COPYING
  46 usr/src/cmd/man/src/util/solbookv2/THIRDPARTYLICENSE
  47 usr/src/cmd/mdb/common/libstand/THIRDPARTYLICENSE
  48 usr/src/cmd/mt/THIRDPARTYLICENSE
  49 usr/src/cmd/perl/5.8.4/distrib/ext/Cwd/THIRDPARTYLICENSE
  50 usr/src/cmd/perl/THIRDPARTYLICENSE
  51 usr/src/cmd/refer/THIRDPARTYLICENSE
  52 usr/src/cmd/script/THIRDPARTYLICENSE
  53 usr/src/cmd/sendmail/THIRDPARTYLICENSE
  54 usr/src/cmd/soelim/THIRDPARTYLICENSE
  55 usr/src/cmd/ssh/THIRDPARTYLICENSE
  56 usr/src/cmd/stat/vmstat/THIRDPARTYLICENSE
  57 usr/src/cmd/tbl/THIRDPARTYLICENSE
  58 usr/src/cmd/tcpd/THIRDPARTYLICENSE
  59 usr/src/cmd/terminfo/THIRDPARTYLICENSE
  60 usr/src/cmd/tip/THIRDPARTYLICENSE
  61 usr/src/cmd/ul/THIRDPARTYLICENSE
```



```
  62 usr/src/cmd/units/THIRDPARTYLICENSE
  63 usr/src/cmd/vgrind/THIRDPARTYLICENSE
  64 usr/src/cmd/vi/THIRDPARTYLICENSE
  65 usr/src/cmd/which/THIRDPARTYLICENSE
  66 usr/src/cmd/xntpd/THIRDPARTYLICENSE
  67 usr/src/cmd/xstr/THIRDPARTYLICENSE
  68 usr/src/common/crypto/sha1/amd64/THIRDPARTYLICENSE
  69 usr/src/common/crypto/sha2/amd64/THIRDPARTYLICENSE
  70 usr/src/common/openssl/LICENSE
  71 usr/src/grub/grub-0.95/COPYING
  72 usr/src/lib/gss_mechs/mech_krb5/THIRDPARTYLICENSE
  73 usr/src/lib/krb5/THIRDPARTYLICENSE
  74 usr/src/lib/libast/THIRDPARTYLICENSE
  75 usr/src/lib/libbc/THIRDPARTYLICENSE
  76 usr/src/lib/libbsdmalloc/THIRDPARTYLICENSE
  77 usr/src/lib/libcmd/THIRDPARTYLICENSE
  78 usr/src/lib/libdll/THIRDPARTYLICENSE
  79 usr/src/lib/libgss/THIRDPARTYLICENSE
  80 usr/src/lib/libinetutil/common/THIRDPARTYLICENSE
  81 usr/src/lib/libkmf/THIRDPARTYLICENSE
  82 usr/src/lib/libldap5/THIRDPARTYLICENSE
  83 usr/src/lib/libmp/common/THIRDPARTYLICENSE
  84 usr/src/lib/libpp/THIRDPARTYLICENSE
  85 usr/src/lib/libresolv/THIRDPARTYLICENSE
  86 usr/src/lib/libresolv2/THIRDPARTYLICENSE
  87 usr/src/lib/libsasl/THIRDPARTYLICENSE
  88 usr/src/lib/libshell/THIRDPARTYLICENSE
  89 usr/src/lib/libtecla/THIRDPARTYLICENSE
  90 usr/src/lib/pam_modules/authtok_check/THIRDPARTYLICENSE
  91 usr/src/lib/passwdutil/THIRDPARTYLICENSE
  92 usr/src/lib/pkcs11/include/THIRDPARTYLICENSE
  93 usr/src/stand/lib/tcp/THIRDPARTYLICENSE
  94 usr/src/tools/ctf/dwarf/THIRDPARTYLICENSE
  95 usr/src/ucbcmd/basename/THIRDPARTYLICENSE
  96 usr/src/ucbcmd/echo/THIRDPARTYLICENSE
  97 usr/src/ucbcmd/from/THIRDPARTYLICENSE
  98 usr/src/ucbcmd/groups/THIRDPARTYLICENSE
  99 usr/src/ucbcmd/ln/THIRDPARTYLICENSE
 100 usr/src/ucbcmd/ls/THIRDPARTYLICENSE
 101 usr/src/ucbcmd/plot/THIRDPARTYLICENSE
 102 usr/src/ucbcmd/sum/THIRDPARTYLICENSE
 103 usr/src/ucbcmd/test/THIRDPARTYLICENSE
 104 usr/src/ucbcmd/tset/THIRDPARTYLICENSE
 105 usr/src/ucbcmd/users/THIRDPARTYLICENSE
 106 usr/src/ucbcmd/whereis/THIRDPARTYLICENSE
 107 usr/src/ucbcmd/whoami/THIRDPARTYLICENSE
 108 usr/src/ucblib/libcurses/THIRDPARTYLICENSE
 109 usr/src/ucblib/libtermcap/THIRDPARTYLICENSE
 110 usr/src/ucblib/libucb/THIRDPARTYLICENSE
 111 usr/src/uts/common/gssapi/mechs/krb5/THIRDPARTYLICENSE
 112 usr/src/uts/common/inet/ip/THIRDPARTYLICENSE.rts
 113 usr/src/uts/common/inet/tcp/THIRDPARTYLICENSE
 114 usr/src/uts/common/io/THIRDPARTYLICENSE.etheraddr
 115 usr/src/uts/common/io/aac/THIRDPARTYLICENSE
 116 usr/src/uts/common/io/afe/THIRDPARTYLICENSE
 117 usr/src/uts/common/io/chxge/com/THIRDPARTYLICENSE
 118 usr/src/uts/common/io/ib/clients/rds/THIRDPARTYLICENSE
 119 usr/src/uts/common/io/mxfe/THIRDPARTYLICENSE
 120 usr/src/uts/common/io/sfe/THIRDPARTYLICENSE
 121 usr/src/uts/common/io/wpi/fw-wpi/LICENSE
 122 usr/src/uts/common/sys/THIRDPARTYLICENSE.agpgart
 123 usr/src/uts/common/sys/THIRDPARTYLICENSE.icu
 124 usr/src/uts/common/sys/THIRDPARTYLICENSE.unicode
 125 usr/src/uts/common/sys/i2o/THIRDPARTYLICENSE
 126 usr/src/uts/common/zmod/THIRDPARTYLICENSE
 127 usr/src/uts/intel/THIRDPARTYLICENSE
```

```
 128 usr/src/uts/intel/io/acpica/THIRDPARTYLICENSE
 129 usr/src/uts/intel/io/amr/THIRDPARTYLICENSE
 130 usr/src/common/crypto/ecc/THIRDPARTYLICENSE
 131 usr/src/common/mpi/THIRDPARTYLICENSE
 132 usr/src/lib/libsmbfs/smb/THIRDPARTYLICENSE.apple
 133 usr/src/lib/libsmbfs/smb/THIRDPARTYLICENSE.boris_popov
 134 usr/src/lib/libsmbfs/smb/THIRDPARTYLICENSE.bsd4
 135 usr/src/lib/libsmbfs/smb/THIRDPARTYLICENSE.microsoft
```

```
**********************************************************
    4323 Thu Mar 20 14:24:06 2008
new/usr/src/uts/common/sys/sha2.h
6665607 Need a SHA256/SHA384/SHA512 implementation optimized for 64-bit x86
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
 * Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#ifndef _SYS_SHA2_H
#define _SYS_SHA2_H

#pragma ident   "@(#)sha2.h     1.5     08/03/05 SMI"
#pragma ident   "@(#)sha2.h     1.4     06/03/28 SMI"

#include <sys/types.h>          /* for uint_* */

#ifdef  __cplusplus
extern "C" {
#endif

#define SHA2_HMAC_MIN_KEY_LEN   8       /* SHA2-HMAC min key length in bits */
#define SHA2_HMAC_MAX_KEY_LEN   INT_MAX /* SHA2-HMAC max key length in bits */

#define SHA256_DIGEST_LENGTH    32      /* SHA256 digest length in bytes */
#define SHA384_DIGEST_LENGTH    48      /* SHA384 digest length in bytes */
#define SHA512_DIGEST_LENGTH    64      /* SHA512 digest length in bytes */

#define SHA256_HMAC_BLOCK_SIZE  64      /* SHA256-HMAC block size */
#define SHA512_HMAC_BLOCK_SIZE  128     /* SHA512-HMAC block size */

#define SHA256                  0
#define SHA256_HMAC             1
#define SHA256_HMAC_GEN         2
#define SHA384                  3
#define SHA384_HMAC             4
#define SHA384_HMAC_GEN         5
#define SHA512                  6
#define SHA512_HMAC             7
#define SHA512_HMAC_GEN         8

/*
 * SHA2 context.
 * The contents of this structure are a private interface between the
```



```
 * Init/Update/Final calls of the functions defined below.
 * Callers must never attempt to read or write any of the fields
 * in this structure directly.
 * in this strucutre directly.
 */
typedef struct  {
        uint32_t algotype;              /* Algorithm Type */

        /* state (ABCDEFGH) */
        union {
                uint32_t s32[8];        /* for SHA256 */
                uint64_t s64[8];        /* for SHA384/512 */
        } state;
        /* number of bits */
        union {
                uint32_t c32[2];        /* for SHA256 , modulo 2^64 */
                uint64_t c64[2];        /* for SHA384/512, modulo 2^128 */
        } count;
        union {
                uint8_t         buf8[128];      /* undigested input */
                uint32_t        buf32[32];      /* realigned input */
                uint64_t        buf64[16];      /* realigned input */
        } buf_un;
} SHA2_CTX;
______unchanged_portion_omitted_
```

```
**********************************************************
    2794 Thu Mar 20 14:24:10 2008
new/usr/src/uts/intel/sha2/Makefile
6665607 Need a SHA256/SHA384/SHA512 implementation optimized for 64-bit x86
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
# Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
#ident  "@(#)Makefile   1.5     08/03/20 SMI"
#ident  "@(#)Makefile   1.3     06/11/02 SMI"
#
#       This makefile drives the production of the sha2 crypto kernel module.
#
#       intel architecture dependent
#


#
#       Path to the base of the uts directory tree (usually /usr/src/uts).
#
UTSBASE = ../..
COMDIR = $(COMMONBASE)/crypto/sha2

#
#       Define the module and object file sets.
#
MODULE          = sha2
LINTS           = $(SHA2_OBJS:%.o=$(LINTS_DIR)/%.ln)
SHA2_OBJS_32    =
SHA2_OBJS_64    = sha512-x86_64.o sha256-x86_64.o
SHA2_OBJS       += $(SHA2_OBJS_$(CLASS))
OBJECTS         = $(SHA2_OBJS:%=$(OBJS_DIR)/%)
LINTS           = $(SHA2_OBJS:%.o=$(LINTS_DIR)/%.ln)
ROOTMODULE      = $(ROOT_CRYPTO_DIR)/$(MODULE)
ROOTLINK        = $(ROOT_MISC_DIR)/$(MODULE)

#
#       Include common rules.
#
include $(UTSBASE)/intel/Makefile.intel

#
#       Override defaults
#
CLEANFILES += sha512-x86_64.s sha256-x86_64.s
```



```
#
#       Define targets
#
ALL_TARGET      = $(BINARY)
LINT_TARGET     = $(MODULE).lint
INSTALL_TARGET  = $(BINARY) $(ROOTMODULE) $(ROOTLINK)

#
# For now, disable these lint checks; maintainers should endeavor
# to investigate and remove these for maximum lint coverage.
# Please do not carry these forward to new Makefiles.
#
LINTTAGS        += -erroff=E_BAD_PTR_CAST_ALIGN
LINTTAGS        += -erroff=E_SUPPRESSION_DIRECTIVE_UNUSED
LINTTAGS        += -erroff=E_PTRDIFF_OVERFLOW
LINTTAGS        += -erroff=E_ASSIGN_NARROW_CONV

#
#       Default build targets.
#
.KEEP_STATE:

def:            $(DEF_DEPS)

all:            $(ALL_DEPS)

clean:          $(CLEAN_DEPS)

clobber:        $(CLOBBER_DEPS)

lint:           $(LINT_DEPS)

modlintlib:     $(MODLINTLIB_DEPS)

clean.lint:     $(CLEAN_LINT_DEPS)

install:        $(INSTALL_DEPS)

$(ROOTLINK):    $(ROOT_MISC_DIR) $(ROOTMODULE)
        -$(RM) $@; ln $(ROOTMODULE) $@

#
#       Include common targets.
#
include $(UTSBASE)/intel/Makefile.targ

$(OBJS_DIR)/%.o: %.s
        $(COMPILE.s) -o $@ ${@F:.o=.s}
        $(POST_PROCESS_O)

$(OBJS_DIR)/%.ln: %.s
        @($(LHEAD) $(LINT.c) ${@F:.ln=.s} $(LTAIL))

sha512-x86_64.s: $(COMDIR)/amd64/sha512-x86_64.pl
        $(PERL) $? $@

sha256-x86_64.s: $(COMDIR)/amd64/sha512-x86_64.pl
        $(PERL) $? $@
```