```
**********************************************************
    5271 Mon Mar 24 15:58:26 2008
new/deleted_files/usr/src/common/crypto/arcfour/amd64/arcfour_crypt_amd64.s
6652716 Need an ARCFOUR implementation optimized for Intel EM64T
**********************************************************
#if !defined(lint) && !defined(__lint)
/  ARCFOUR implementation optimized for AMD64.
/
/  Author: Marc Bevand <bevand_m (at) epita.fr>
/  Licence: I hereby disclaim the copyright on this code and place it
/  in the public domain.
/
/  The code has been designed to be easily integrated into openssl:
/  the exported RC4() function can replace the actual implementations
/  openssl already contains. Please note that when linking with openssl,
/  it requires that sizeof(RC4_INT) == 8. So openssl must be compiled
/  with -DRC4_INT='unsigned long'.
/
/  The throughput achieved by this code is about 320 MBytes/sec, on
/  a 1.8 GHz AMD Opteron (rev C0) processor.


/ ***** BEGIN LICENSE BLOCK *****
/ Version: MPL 1.1/GPL 2.0/LGPL 2.1
/
/ The contents of this file are subject to the Mozilla Public License Version
/ 1.1 (the "License"); you may not use this file except in compliance with
/ the License. You may obtain a copy of the License at
/ http://www.mozilla.org/MPL/
/
/ Software distributed under the License is distributed on an "AS IS" basis,
/ WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License
/ for the specific language governing rights and limitations under the
/ License.
/
/ The Original Code is "Marc Bevand's fast AMD64 ARCFOUR source"
/
/ The Initial Developer of the Original Code is
/ Marc Bevand <bevand_m@epita.fr> .
/ Portions created by the Initial Developer are
/ Copyright (C) 2004 the Initial Developer. All Rights Reserved.
/
/ Contributor(s):
/
/ Alternatively, the contents of this file may be used under the terms of
/ either the GNU General Public License Version 2 or later (the "GPL"), or
/ the GNU Lesser General Public License Version 2.1 or later (the "LGPL"),
/ in which case the provisions of the GPL or the LGPL are applicable instead
/ of those above. If you wish to allow use of your version of this file only
/ under the terms of either the GPL or the LGPL, and not to allow others to
/ use your version of this file under the terms of the MPL, indicate your
/ decision by deleting the provisions above and replace them with the notice
/ and other provisions required by the GPL or the LGPL. If you do not delete
/ the provisions above, a recipient may use your version of this file under
/ the terms of any one of the MPL, the GPL or the LGPL.
/
/ ***** END LICENSE BLOCK *****

        .ident  "@(#)arcfour_crypt_amd64.s      1.1     08/01/02 SMI"

/
/ void arcfour_crypt(ARCFour_key *key, uchar_t *in,
/               uchar_t *out, size_t len);
/
/ The following is Marc Bevand's RC4 implementation optimized for
/ AMD64.  It has been lifted intact, except for minor interface
```



```
/ changes to get along with Solaris crypto common code (the parameter
/ order and the key struct element order are both different).
/ This function works for both aligned and unaligned data ('in' and 'out').
/ The key and key elements must be aligned.
/
/ Register Usage
/ rax           data[x]
/ rbx           ARG(len)
/ rcx           key->i (aka x)
/ rdx           key->j (aka y)
/ rsi           ARG(in)
/ rdi           ARG(out)
/ rbp           key->arr (aka data or d)
/ rsp           stack
/ r8            8 bytes of rc4 stream
/ r9            temp
/ r10-r15       unused
/

#include <sys/asm_linkage.h>


        ENTRY_NP(arcfour_crypt)
        /* EXPORT DELETE START */
                                        / load parameters
        push    %rbp
        push    %rbx
        mov     %rdi,           %rbp    / rbp = ARG(key)
                                        / rsi = ARG(in)
        mov     %rdx,           %rdi    / rdi = ARG(out)
        mov     %rcx,           %rbx    / rbx = ARG(len)

                                        / load key indices and key
        mov     2048(%rbp),     %rcx    / rcx x = key->i
        mov     2056(%rbp),     %rdx    / rdx y = key->j
                                        / rbp d = key->arr
        inc     %rcx                    / x++
        and     $255,           %rcx    / x &= 0xff
        lea     -8(%rbx,%rsi),  %rbx    / rbx = in+len-8
        mov     %rbx,           %r9     / tmp = in+len-8
        mov     (%rbp,%rcx,8),  %rax    / tx = d[x]
        cmp     %rsi,           %rbx    / cmp in with in+len-8
        jl      .Lend                   / jump if (in+len-8 < in)

.Lstart:
        add     $8,             %rsi            / increment in
        add     $8,             %rdi            / increment out

        / generate the next 8 bytes of the rc4 stream into %r8
        mov     $8,             %r11            / byte counter
1:      add     %al,            %dl             / y += tx
        mov     (%rbp,%rdx,8),  %ebx            / ty = d[y]
        mov     %ebx,           (%rbp,%rcx,8)   / d[x] = ty
        add     %al,            %bl             / val = ty + tx
        mov     %eax,           (%rbp,%rdx,8)   / d[y] = tx
        inc     %cl                             / x++           (NEXT ROUND)
        mov     (%rbp,%rcx,8),  %eax            / tx = d[x]     (NEXT ROUND)
        movb    (%rbp,%rbx,8),  %r8b            / val = d[val]
        dec     %r11b
        ror     $8,             %r8             / (ror does not change ZF)
        jnz     1b

        / xor 8 bytes
        xor     -8(%rsi),       %r8
        cmp     %r9,            %rsi            / cmp in+len-8 with in
        mov     %r8,            -8(%rdi)
```

```
128         jle     .Lstart                         / jump if (in <= in+len-8)

130 .Lend:
131         add     $8,             %r9             / tmp = in+len

133         / handle the last bytes, one by one
134 1:      cmp     %rsi,           %r9             / cmp in with in+len
135         jle     .Lfinished                      / jump if (in+len <= in)
136         add     %al,            %dl             / y += tx
137         mov     (%rbp,%rdx,8),  %ebx            / ty = d[y]
138         mov     %ebx,           (%rbp,%rcx,8)   / d[x] = ty
139         add     %al,            %bl             / val = ty + tx
140         mov     %eax,           (%rbp,%rdx,8)   / d[y] = tx
141         inc     %cl                             / x++           (NEXT ROUND)
142         mov     (%rbp,%rcx,8),  %eax            / tx = d[x]     (NEXT ROUND)
143         movb    (%rbp,%rbx,8),  %r8b            / val = d[val]
144         xor     (%rsi),         %r8b            / xor 1 byte
145         movb    %r8b,           (%rdi)
146         inc     %rsi                            / in++
147         inc     %rdi                            / out++
148         jmp     1b

150 .Lfinished:                                     / save key indices i & j
151         dec     %rcx                            / x--
152         movb    %dl,            2056(%rbp)      / key->j = y
153         movb    %cl,            2048(%rbp)      / key->i = x
154         pop     %rbx
155         pop     %rbp

157         /* EXPORT DELETE END */

159         ret
160         SET_SIZE(arcfour_crypt)

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

```
**********************************************************
    1758 Mon Mar 24 15:58:27 2008
new/usr/src/common/crypto/arcfour/amd64/THIRDPARTYLICENSE
6652716 Need an ARCFOUR implementation optimized for Intel EM64T
**********************************************************
```

```
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.
```

```
**********************************************************
      34 Mon Mar 24 15:58:29 2008
new/usr/src/common/crypto/arcfour/amd64/THIRDPARTYLICENSE.descript
6652716 Need an ARCFOUR implementation optimized for Intel EM64T
**********************************************************
```

```
   1 PORTIONS OF ARCFOUR FUNCTIONALITY
```

```
**********************************************************
   10756 Mon Mar 24 15:58:31 2008
new/usr/src/common/crypto/arcfour/amd64/arcfour-x86_64.pl
6652716 Need an ARCFOUR implementation optimized for Intel EM64T
**********************************************************
```

```
#!/usr/bin/env perl
#
# ====================================================================
# Written by Andy Polyakov <appro@fy.chalmers.se> for the OpenSSL
# project. The module is, however, dual licensed under OpenSSL and
# CRYPTOGAMS licenses depending on where you obtain it. For further
# details see http://www.openssl.org/~appro/cryptogams/.
# ====================================================================
#
# 2.22x RC4 tune-up:-) It should be noted though that my hand [as in
# "hand-coded assembler"] doesn't stand for the whole improvement
# coefficient. It turned out that eliminating RC4_CHAR from config
# line results in ~40% improvement (yes, even for C implementation).
# Presumably it has everything to do with AMD cache architecture and
# RAW or whatever penalties. Once again! The module *requires* config
# line *without* RC4_CHAR! As for coding "secret," I bet on partial
# register arithmetics. For example instead of 'inc %r8; and $255,%r8'
# I simply 'inc %r8b'. Even though optimization manual discourages
# to operate on partial registers, it turned out to be the best bet.
# At least for AMD... How IA32E would perform remains to be seen...

# As was shown by Marc Bevand reordering of couple of load operations
# results in even higher performance gain of 3.3x:-) At least on
# Opteron... For reference, 1x in this case is RC4_CHAR C-code
# compiled with gcc 3.3.2, which performs at ~54MBps per 1GHz clock.
# Latter means that if you want to *estimate* what to expect from
# *your* Opteron, then multiply 54 by 3.3 and clock frequency in GHz.

# Intel P4 EM64T core was found to run the AMD64 code really slow...
# The only way to achieve comparable performance on P4 was to keep
# RC4_CHAR. Kind of ironic, huh? As it's apparently impossible to
# compose blended code, which would perform even within 30% marginal
# on either AMD and Intel platforms, I implement both cases. See
# rc4_skey.c for further details...

# P4 EM64T core appears to be "allergic" to 64-bit inc/dec. Replacing
# those with add/sub results in 50% performance improvement of folded
# loop...

# As was shown by Zou Nanhai loop unrolling can improve Intel EM64T
# performance by >30% [unlike P4 32-bit case that is]. But this is
# provided that loads are reordered even more aggressively! Both code
# pathes, AMD64 and EM64T, reorder loads in essentially same manner
# as my IA-64 implementation. On Opteron this resulted in modest 5%
# improvement [I had to test it], while final Intel P4 performance
# achieves respectful 432MBps on 2.8GHz processor now. For reference.
# If executed on Xeon, current RC4_CHAR code-path is 2.7x faster than
# RC4_INT code-path. While if executed on Opteron, it's only 25%
# slower than the RC4_INT one [meaning that if CPU µ-arch detection
# is not implemented, then this final RC4_CHAR code-path should be
# preferred, as it provides better *all-round* performance].

# Intel Core2 was observed to perform poorly on both code paths:-( It
# apparently suffers from some kind of partial register stall, which
# occurs in 64-bit mode only [as virtually identical 32-bit loop was
# observed to outperform 64-bit one by almost 50%]. Adding two movzb to
# cloop1 boosts its performance by 80%! This loop appears to be optimal
# fit for Core2 and therefore the code was modified to skip cloop8 on
# this CPU.

#
```



```
# OpenSolaris OS modifications
#
# Sun elects to use this software under the BSD license.
#
# This source originates from OpenSSL file rc4-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 some comments, "use strict", and declared all variables.
#
# 2. Added OpenSolaris ENTRY_NP/SET_SIZE macros from
# /usr/include/sys/asm_linkage.h, .ident keywords, and lint(1B) guards.
#
# 3. Changed function name from RC4() to arcfour_crypt() and RC4_set_key()
# to arcfour_key_init(), and changed the parameter order for both to that
# used by OpenSolaris.
#
# 4. The current method of using cpuid feature bits 20 (NX) or 28 (HTT) from
# function OPENSSL_ia32_cpuid() to distinguish Intel/AMD does not work for
# some newer AMD64 processors, as these bits are set on both Intel EM64T
# processors and newer AMD64 processors.  I replaced this with code to use CPUID
# instruction subfunction EAX=0 to determine if we're running on "GenuineIntel"
# or not. The result decides whether to use a
#       * 1-byte key array (label .LRC4_CHAR, optimal on Intel EM64T) or a
#       * 4-byte key array (Labels .Lloop1 and .Lloop8, optimal on AMD64).
#
# 5. Removed x86_64-xlate.pl script (not needed for as(1) or gas(1) assemblers).
#
# 6. Removed Lcloop8 code (slower than Lcloop1 on EM64T and not used on AMD64).
#

use strict;
my ($code, $dat, $inp, $out, $len, $idx, $ido, $i, @XX, @TX, $YY, $TY);
my $output = shift;
open STDOUT,">$output";

#
# Parameters
#

# OpenSSL:
# void RC4(RC4_KEY *key, unsigned long len, const unsigned char *indata,
#       unsigned char *outdata);
#$dat="%rdi";       # arg1
#$len="%rsi";       # arg2
#$inp="%rdx";       # arg3
#$out="%rcx";       # arg4

# OpenSolaris:
# void arcfour_crypt(ARCFour_key *key, uchar_t *in, uchar_t *out, size_t len);
$dat="%rdi";        # arg1
$inp="%rsi";        # arg2
$out="%rdx";        # arg3
$len="%rcx";        # arg4

#
# Register variables
#
# $XX[0] is key->i (aka key->x), $XX[1] is a temporary.
# $TX[0] and $TX[1] are temporaries.
# $YY is key->j (aka key->y).
# $TY is a temporary.
#
@XX=("%r8","%r10");
@TX=("%r9","%r11");
$YY="%r12";
```

```
128 $TY="%r13";
129
130 $code=<<___;
131 #if !defined(lint) && !defined(__lint)
132
133         .ident  "@(#)arcfour-x86_64.pl  1.1     08/03/01 SMI"
134
135 #include <sys/asm_linkage.h>
136
137
138 ENTRY_NP(arcfour_crypt)
139         /* EXPORT DELETE START */
140
141         or      $len,$len # If (len == 0) return
142         jne     .Lentry
143         ret
144 .Lentry:
145         push    %r12
146         push    %r13
147
148         / Set $dat to beginning of array, key->arr[0]
149         add     \$8,$dat
150         / Get key->j
151         movl    -8($dat),$XX[0]#d
152         / Get key->i
153         movl    -4($dat),$YY#d
154
155         /
156         / Use a 1-byte data array, on Intel P4 EM64T,
157         / which is more efficient there,
158         / or a 4-byte data array (for AMD AMD64).
159         /
160
161         / If RC4_CHAR flag set (Intel EM64T), then use 1-byte array
162         cmpl    \$-1,256($dat)
163         je      .LRC4_CHAR
164         / otherwise use 4-byte integer array (AMD64)
165         inc     $XX[0]#b
166         movl    ($dat,$XX[0],4),$TX[0]#d
167         test    \$-8,$len
168         jz      .Lloop1
169         jmp     .Lloop8
170
171 .align  16
172 .Lloop8:
173         /
174         / This code is for use with a 4-byte integer data array, which is
175         / more efficient on AMD64 Athlon and Opteron-class processors.
176         /
177 ___
178 for ($i=0;$i<8;$i++) {
179 $code.=<<___;
180         add     $TX[0]#b,$YY#b
181         mov     $XX[0],$XX[1]
182         movl    ($dat,$YY,4),$TY#d
183         ror     \$8,%rax                        # ror is redundant when $i=0
184         inc     $XX[1]#b
185         movl    ($dat,$XX[1],4),$TX[1]#d
186         cmp     $XX[1],$YY
187         movl    $TX[0]#d,($dat,$YY,4)
188         cmove   $TX[0],$TX[1]
189         movl    $TY#d,($dat,$XX[0],4)
190         add     $TX[0]#b,$TY#b
191         movb    ($dat,$TY,4),%al
192 ___
193 push(@TX,shift(@TX)); push(@XX,shift(@XX));     # "rotate" registers
```



```
194 }
195 $code.=<<___;
196         ror     \$8,%rax
197         sub     \$8,$len
198
199         xor     ($inp),%rax
200         add     \$8,$inp
201         mov     %rax,($out)
202         add     \$8,$out
203
204         test    \$-8,$len
205         jnz     .Lloop8
206         cmp     \$0,$len
207         jne     .Lloop1
208 ___
209 $code.=<<___;
210 .Lexit:
211         /
212         / Cleanup and exit code
213         /
214         / --i to undo ++i done at entry
215         sub     \$1,$XX[0]#b
216         / set key->i
217         movl    $XX[0]#d,-8($dat)
218         / set key->j
219         movl    $YY#d,-4($dat)
220
221         pop     %r13
222         pop     %r12
223         ret
224 .align  16
225 .Lloop1:
226         add     $TX[0]#b,$YY#b
227         movl    ($dat,$YY,4),$TY#d
228         movl    $TX[0]#d,($dat,$YY,4)
229         movl    $TY#d,($dat,$XX[0],4)
230         add     $TY#b,$TX[0]#b
231         inc     $XX[0]#b
232         movl    ($dat,$TX[0],4),$TY#d
233         movl    ($dat,$XX[0],4),$TX[0]#d
234         xorb    ($inp),$TY#b
235         inc     $inp
236         movb    $TY#b,($out)
237         inc     $out
238         dec     $len
239         jnz     .Lloop1
240         jmp     .Lexit
241
242
243 .align  16
244 .LRC4_CHAR:
245         /
246         / This code is for use with a 1-byte integer data array, which is
247         / more efficient on Intel P4 EM64T-class processors.
248         /
249         add     \$1,$XX[0]#b
250         movzb   ($dat,$XX[0]),$TX[0]#d
251         jmp     .Lcloop1
252
253 .align  16
254 .Lcloop1:
255         add     $TX[0]#b,$YY#b
256         movzb   ($dat,$YY),$TY#d
257         movb    $TX[0]#b,($dat,$YY)
258         movb    $TY#b,($dat,$XX[0])
259         add     $TX[0]#b,$TY#b
```

```
260         add     \$1,$XX[0]#b
261         / Intel Optimization (preload $TY and $XX[0]):
262         movzb   $TY#b,$TY#d
263         movzb   $XX[0]#b,$XX[0]#d
264         movzb   ($dat,$TY),$TY#d
265         movzb   ($dat,$XX[0]),$TX[0]#d
266         xorb    ($inp),$TY#b
267         lea     1($inp),$inp
268         movb    $TY#b,($out)
269         lea     1($out),$out
270         sub     \$1,$len
271         jnz     .Lcloop1
272         jmp     .Lexit

274         /* EXPORT DELETE END */
275         ret
276 SET_SIZE(arcfour_crypt)
277 ___


280 #
281 # Parameters
282 #

284 # OpenSSL:
285 # void RC4_set_key(RC4_KEY *key, int len, const unsigned char *data);
286 #$dat="%rdi";       # arg1
287 #$len="%rsi";       # arg2
288 #$inp="%rdx";       # arg3

290 # OpenSolaris:
291 # void arcfour_key_init(ARCFour_key *key, uchar_t *keyval, int keyvallen);
292 $dat="%rdi";        # arg1
293 $inp="%rsi";        # arg2
294 $len="%rdx";        # arg3

296 # Temporaries
297 $idx="%r8";
298 $ido="%r9";

300 $code.=<<___;
301         / int arcfour_crypt_on_intel(void);
302 .extern arcfour_crypt_on_intel

304 ENTRY_NP(arcfour_key_init)
305         /* EXPORT DELETE START */

307         / Find out if we're running on Intel or something else (e.g., AMD64).
308         / This sets %eax to 1 for Intel, otherwise 0.
309         push    %rdi            / Save arg1
310         push    %rsi            / Save arg2
311         push    %rdx            / Save arg3
312         call    arcfour_crypt_on_intel
313         pop     %rdx            / Restore arg3
314         pop     %rsi            / Restore arg2
315         pop     %rdi            / Restore arg1

317         / Set $dat to beginning of array, key->arr[0]
318         lea     8($dat),$dat
319         lea     ($inp,$len),$inp
320         neg     $len
321         mov     $len,%rcx
322         / Zeroed below, as %eax contains a flag from arcfour_crypt_on_intel():
323         /xor    %eax,%eax
324         xor     $ido,$ido
325         xor     %r10,%r10
```



```
326         xor     %r11,%r11

328         /
329         / Use a 1-byte data array, on Intel P4 EM64T,
330         / which is more efficient there,
331         / or a 4-byte data array (for AMD AMD64).
332         /
333         cmp     \$1,%eax        / Test if Intel
334         mov     \$0,%eax        / Zero eax without modifying flags
335         je      .Lc1stloop      / If Intel then use a 1-byte array,
336         jmp     .Lw1stloop      / otherwise use a 4-byte array.

338 .align  16
339 .Lw1stloop:
340         / AMD64 (4-byte array)
341         mov     %eax,($dat,%rax,4)
342         add     \$1,%al
343         jnc     .Lw1stloop

345         xor     $ido,$ido
346         xor     $idx,$idx
347 .align  16
348 .Lw2ndloop:
349         mov     ($dat,$ido,4),%r10d
350         add     ($inp,$len,1),$idx#b
351         add     %r10b,$idx#b
352         add     \$1,$len
353         mov     ($dat,$idx,4),%r11d
354         cmovz   %rcx,$len
355         mov     %r10d,($dat,$idx,4)
356         mov     %r11d,($dat,$ido,4)
357         add     \$1,$ido#b
358         jnc     .Lw2ndloop
359         jmp     .Lexit_key

361 .align  16
362 .Lc1stloop:
363         / Intel EM64T (1-byte array)
364         mov     %al,($dat,%rax)
365         add     \$1,%al
366         jnc     .Lc1stloop

368         xor     $ido,$ido
369         xor     $idx,$idx
370 .align  16
371 .Lc2ndloop:
372         mov     ($dat,$ido),%r10b
373         add     ($inp,$len),$idx#b
374         add     %r10b,$idx#b
375         add     \$1,$len
376         mov     ($dat,$idx),%r11b
377         jnz     .Lcnowrap
378         mov     %rcx,$len
379 .Lcnowrap:
380         mov     %r10b,($dat,$idx)
381         mov     %r11b,($dat,$ido)
382         add     \$1,$ido#b
383         jnc     .Lc2ndloop
384         movl    \$-1,256($dat)

386 .align  16
387 .Lexit_key:
388         xor     %eax,%eax
389         mov     %eax,-8($dat)
390         mov     %eax,-4($dat)
```

```
392         /* EXPORT DELETE END */
393         ret
394 SET_SIZE(arcfour_key_init)
395 .asciz  "RC4 for x86_64, CRYPTOGAMS by <appro\@openssl.org>"

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

403 $code =~ s/#([bwd])/$1/gm;

405 print $code;

407 close STDOUT;
```

```
**********************************************************
    1726 Mon Mar 24 15:58:33 2008
new/usr/src/common/crypto/arcfour/arcfour.h
6652716 Need an ARCFOUR implementation optimized for Intel EM64T
**********************************************************
/*
 * 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 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#ifndef _ARCFOUR_H
#define _ARCFOUR_H

#pragma ident   "@(#)arcfour.h  1.7     08/03/01 SMI"
#pragma ident   "@(#)arcfour.h  1.6     07/03/29 SMI"

#ifdef __cplusplus
extern "C" {
#endif

#include <sys/types.h>
#include <sys/crypto/common.h>

#define ARCFOUR_MIN_KEY_BYTES   1

#define ARCFOUR_MAX_KEY_BYTES   256

#define ARCFOUR_MIN_KEY_BITS    (ARCFOUR_MIN_KEY_BYTES << 3)
#define ARCFOUR_MAX_KEY_BITS    (ARCFOUR_MAX_KEY_BYTES << 3)

typedef arcfour_state_t ARCFour_key;

void arcfour_key_init(ARCFour_key *key, uchar_t *keyval, int keyvallen);
void arcfour_crypt(ARCFour_key *key, uchar_t *in, uchar_t *out, size_t len);
#ifdef  sun4u
void arcfour_crypt_aligned(ARCFour_key *key, size_t len, uchar_t *in,
    uchar_t *out);
#endif  /* sun4u */
#ifdef  __amd64
int arcfour_crypt_on_intel(void);
#endif  /* _amd64 */

#ifdef  __cplusplus
}
_____unchanged_portion_omitted_
```

```
**********************************************************
    3539 Mon Mar 24 15:58:37 2008
new/usr/src/common/crypto/arcfour/arcfour_crypt.c
6652716 Need an ARCFOUR implementation optimized for Intel EM64T
**********************************************************
```

```
/*
 * 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.
 */

#pragma ident   "@(#)arcfour_crypt.c    1.7     08/03/01 SMI"
#pragma ident   "@(#)arcfour_crypt.c    1.6     08/01/02 SMI"

#include "arcfour.h"

#if defined(__amd64)
#ifdef _KERNEL
#include <sys/x86_archext.h>
#include <sys/cpuvar.h>

#else
#include <sys/auxv.h>
#endif  /* _KERNEL */
/*
 * Use hand-tuned, processor-specific assembly version of arcfour_crypt()
 * for 64-bit x86:
 */
#define USE_PSR_VERSION_OF_ARCFOUR_CRYPT
#endif  /* __amd64 */

#if !defined(__amd64)
/* Initialize the key stream 'key' using the key value */
void
arcfour_key_init(ARCFour_key *key, uchar_t *keyval, int keyvallen)
{
/* EXPORT DELETE START */

        uchar_t ext_keyval[256];
        uchar_t tmp;
        int i, j;

        /* Normalize key length to 256 */
        for (i = j = 0; i < 256; i++, j++) {
                if (j == keyvallen)
                        j = 0;
```



```
                ext_keyval[i] = keyval[j];
        }

        for (i = 0; i < 256; i++)
                key->arr[i] = (uchar_t)i;

        j = 0;
        for (i = 0; i < 256; i++) {
                j = (j + key->arr[i] + ext_keyval[i]) % 256;
                tmp = key->arr[i];
                key->arr[i] = key->arr[j];
                key->arr[j] = tmp;
        }
        key->i = 0;
        key->j = 0;

/* EXPORT DELETE END */
}


#if !defined(USE_PSR_VERSION_OF_ARCFOUR_CRYPT)
/*
 * Encipher 'in' using 'key'.
 * in and out can point to the same location
 */
void
arcfour_crypt(ARCFour_key *key, uchar_t *in, uchar_t *out, size_t len)
{
        size_t ii;
        uchar_t tmp, i, j;

/* EXPORT DELETE START */

        /*
         * The sun4u has a version of arcfour_crypt_aligned() hand-tuned for
         * the cases where the input and output  buffers are aligned on
         * a multiple of 8-byte boundary.
         */
#ifdef  sun4u
        int index;

        index = (((uint64_t)(uintptr_t)in) & 0x7);

        /* Get the 'in' on an 8-byte alignment */
        if (index > 0) {
                i = key->i;
                j = key->j;
                for (index = 8 - (uint64_t)(uintptr_t)in & 0x7;
                    (index-- > 0) && len > 0;
                    len--, in++, out++) {
                        i = i + 1;
                        j = j + key->arr[i];
                        tmp = key->arr[i];
                        key->arr[i] = key->arr[j];
                        key->arr[j] = tmp;
                        tmp = key->arr[i] + key->arr[j];
                        *out = *in ^ key->arr[tmp];
                }
                key->i = i;
                key->j = j;

        }
        if (len == 0)
                return;

        /* See if we're fortunate and 'out' got aligned as well */
```

```
121         if ((((uint64_t)(uintptr_t)out) & 7) != 0) {
122 #endif  /* sun4u */
123                 i = key->i;
124                 j = key->j;
125                 for (ii = 0; ii < len; ii++) {
126                         i = i + 1;
127                         j = j + key->arr[i];
128                         tmp = key->arr[i];
129                         key->arr[i] = key->arr[j];
130                         key->arr[j] = tmp;
131                         tmp = key->arr[i] + key->arr[j];
132                         out[ii] = in[ii] ^ key->arr[tmp];
133                 }
134                 key->i = i;
135                 key->j = j;
136 #ifdef  sun4u
137         } else {
138                 arcfour_crypt_aligned(key, len, in, out);
139         }
140 #endif  /* sun4u */

142 /* EXPORT DELETE END */
143 }
145 #else

147 /*
148  * Return 1 if executing on Intel, otherwise 0 (e.g., AMD64).
149  */
150 int
151 arcfour_crypt_on_intel(void)
152 {
153 #ifdef _KERNEL
154         return (cpuid_getvendor(CPU) == X86_VENDOR_Intel);
155 #else
156         uint_t  ui;
157         (void) getisax(&ui, 1);
158         return ((ui & AV_386_AMD_MMX) == 0);
159 #endif  /* _KERNEL */
160 }
161 #endif  /* !__amd64 */
141 #endif  /* !USE_PSR_VERSION_OF_ARCFOUR_CRYPT */
```

```
**********************************************************
    2018 Mon Mar 24 15:58:41 2008
new/usr/src/lib/pkcs11/pkcs11_softtoken/amd64/Makefile
6652716 Need an ARCFOUR implementation optimized for Intel EM64T
**********************************************************
```

```
#
# 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   1.5     08/03/01 SMI"
# ident "@(#)Makefile   1.4     08/02/26 SMI"
#
# lib/pkcs11/pkcs11_softtoken/amd64/Makefile

AES_PSR_OBJECTS =
ARCFOUR_PSR_OBJECTS =  arcfour-x86_64.o
ARCFOUR_PSR_OBJECTS =  arcfour_crypt_amd64.o
DES_PSR_OBJECTS =
RSA_PSR_OBJECTS =
SHA1_PSR_OBJECTS =
BIGNUM_PSR_OBJECTS = bignum_amd64.o bignum_amd64_asm.o
BIGNUM_PSR_PICS = $(BIGNUM_PSR_OBJECTS:%=pics/%)
BIGNUM_CFG = -DPSR_MUL
BIGNUM_PSR_SRCS = \
        $(BIGNUMDIR)/amd64/bignum_amd64.c \
        $(BIGNUMDIR)/amd64/bignum_amd64_asm.s

pics/bignum_amd64.o := amd64_COPTFLAG = -xO3

include ../Makefile.com
include ../../../Makefile.lib.64

#
# Overrides
#
CLEANFILES      += arcfour-x86_64.s

install: all $(ROOTLIBS64) $(ROOTLINKS64)

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

LINTFLAGS64 += $(BIGNUM_CFG)

pics/arcfour-x86_64.o: arcfour-x86_64.s
        $(COMPILE.s) -o $@ $(AS_BIGPICFLAGS) ${@F:.o=.s}
pics/arcfour_crypt_amd64.o: $(ARCFOURDIR)/amd64/arcfour_crypt_amd64.s
```



```
        $(COMPILE.s) -o $@ $(AS_BIGPICFLAGS) \
                $(ARCFOURDIR)/amd64/arcfour_crypt_amd64.s
        $(POST_PROCESS_O)

arcfour-x86_64.s: $(ARCFOURDIR)/amd64/arcfour-x86_64.pl
        $(PERL) $? $@

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

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

```
**********************************************************
   16832 Mon Mar 24 15:58:46 2008
new/usr/src/uts/common/sys/crypto/common.h
6652716 Need an ARCFOUR implementation optimized for Intel EM64T
**********************************************************
```

```
/*
 * 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.
 */

#ifndef _SYS_CRYPTO_COMMON_H
#define _SYS_CRYPTO_COMMON_H

#pragma ident   "@(#)common.h   1.27    08/03/01 SMI"
#pragma ident   "@(#)common.h   1.26    08/01/04 SMI"

/*
 * Header file for the common data structures of the cryptographic framework
 */

#ifdef __cplusplus
extern "C" {
#endif

#include <sys/types.h>
#include <sys/uio.h>
#include <sys/stream.h>
#include <sys/mutex.h>
#include <sys/condvar.h>


/* Cryptographic Mechanisms */

#define CRYPTO_MAX_MECH_NAME 32
typedef char crypto_mech_name_t[CRYPTO_MAX_MECH_NAME];

typedef uint64_t crypto_mech_type_t;

typedef struct crypto_mechanism {
        crypto_mech_type_t      cm_type;        /* mechanism type */
        caddr_t                 cm_param;       /* mech. parameter */
        size_t                  cm_param_len;   /* mech. parameter len */
} crypto_mechanism_t;
```

*_____unchanged_portion_omitted_*

```
#endif  /* _SYSCALL32 */
```



```
#endif /* _KERNEL */

/*
 * The measurement unit bit flag for a mechanism's minimum or maximum key size.
 * The unit are mechanism dependant.  It can be in bits or in bytes.
 */
typedef uint32_t crypto_keysize_unit_t;

/*
 * The following bit flags are valid in cm_mech_flags field in
 * the crypto_mech_info_t structure of the SPI.
 *
 * Only the first two bit flags are valid in mi_keysize_unit
 * field in the crypto_mechanism_info_t structure of the API.
 */
#define CRYPTO_KEYSIZE_UNIT_IN_BITS     0x00000001
#define CRYPTO_KEYSIZE_UNIT_IN_BYTES    0x00000002
#define CRYPTO_CAN_SHARE_OPSTATE        0x00000004 /* supports sharing */


/* Mechanisms supported out-of-the-box */
#define SUN_CKM_MD4                     "CKM_MD4"
#define SUN_CKM_MD5                     "CKM_MD5"
#define SUN_CKM_MD5_HMAC                "CKM_MD5_HMAC"
#define SUN_CKM_MD5_HMAC_GENERAL        "CKM_MD5_HMAC_GENERAL"
#define SUN_CKM_SHA1                    "CKM_SHA_1"
#define SUN_CKM_SHA1_HMAC               "CKM_SHA_1_HMAC"
#define SUN_CKM_SHA1_HMAC_GENERAL       "CKM_SHA_1_HMAC_GENERAL"
#define SUN_CKM_SHA256                  "CKM_SHA256"
#define SUN_CKM_SHA256_HMAC             "CKM_SHA256_HMAC"
#define SUN_CKM_SHA256_HMAC_GENERAL     "CKM_SHA256_HMAC_GENERAL"
#define SUN_CKM_SHA384                  "CKM_SHA384"
#define SUN_CKM_SHA384_HMAC             "CKM_SHA384_HMAC"
#define SUN_CKM_SHA384_HMAC_GENERAL     "CKM_SHA384_HMAC_GENERAL"
#define SUN_CKM_SHA512                  "CKM_SHA512"
#define SUN_CKM_SHA512_HMAC             "CKM_SHA512_HMAC"
#define SUN_CKM_SHA512_HMAC_GENERAL     "CKM_SHA512_HMAC_GENERAL"
#define SUN_CKM_DES_CBC                 "CKM_DES_CBC"
#define SUN_CKM_DES3_CBC                "CKM_DES3_CBC"
#define SUN_CKM_DES_ECB                 "CKM_DES_ECB"
#define SUN_CKM_DES3_ECB                "CKM_DES3_ECB"
#define SUN_CKM_BLOWFISH_CBC            "CKM_BLOWFISH_CBC"
#define SUN_CKM_BLOWFISH_ECB            "CKM_BLOWFISH_ECB"
#define SUN_CKM_AES_CBC                 "CKM_AES_CBC"
#define SUN_CKM_AES_ECB                 "CKM_AES_ECB"
#define SUN_CKM_AES_CTR                 "CKM_AES_CTR"
#define SUN_CKM_AES_CCM                 "CKM_AES_CCM"
#define SUN_CKM_RC4                     "CKM_RC4"
#define SUN_CKM_RSA_PKCS                "CKM_RSA_PKCS"
#define SUN_CKM_RSA_X_509               "CKM_RSA_X_509"
#define SUN_CKM_MD5_RSA_PKCS            "CKM_MD5_RSA_PKCS"
#define SUN_CKM_SHA1_RSA_PKCS           "CKM_SHA1_RSA_PKCS"
#define SUN_CKM_SHA256_RSA_PKCS         "CKM_SHA256_RSA_PKCS"
#define SUN_CKM_SHA384_RSA_PKCS         "CKM_SHA384_RSA_PKCS"
#define SUN_CKM_SHA512_RSA_PKCS         "CKM_SHA512_RSA_PKCS"
#define SUN_CKM_EC_KEY_PAIR_GEN         "CKM_EC_KEY_PAIR_GEN"
#define SUN_CKM_ECDH1_DERIVE            "CKM_ECDH1_DERIVE"
#define SUN_CKM_ECDSA_SHA1              "CKM_ECDSA_SHA1"
#define SUN_CKM_ECDSA                   "CKM_ECDSA"

/* Shared operation context format for CKM_RC4 */
typedef struct {
#if defined(__amd64)
        uint32_t i, j;
        uint32_t arr[256];
typedef uint64_t arcfour_key_int_t;
```

```
 194 #else
 195         uchar_t arr[256];
 196         uchar_t i, j;
 193 typedef uchar_t arcfour_key_int_t;
 197 #endif /* __amd64 */

 196 typedef struct {
 197         arcfour_key_int_t arr[256];
 198         arcfour_key_int_t i, j;
 198         uint64_t pad;           /* For 64-bit alignment */
 199 } arcfour_state_t;
```

_____unchanged_portion_omitted_

```
**********************************************************
    2610 Mon Mar 24 15:58:50 2008
new/usr/src/uts/intel/arcfour/Makefile
6652716 Need an ARCFOUR implementation optimized for Intel EM64T
**********************************************************
#
# 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   1.8     08/03/01 SMI"
# ident "@(#)Makefile   1.7     08/01/02 SMI"
#
#       This makefile drives the production of the ARCFOUR KEF provider.
#
#       intel implementation architecture dependent
#


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

#
#       Define the module and object file sets.
#
MODULE          = arcfour
LINTS           = $(ARCFOURPROV_OBJS:%.o=$(LINTS_DIR)/%.ln)
ARCFOURPROV_OBJS_32 =
ARCFOURPROV_OBJS_64 = arcfour-x86_64.o
ARCFOURPROV_OBJS_64 = arcfour_crypt_amd64.o
ARCFOURPROV_OBJS += $(ARCFOURPROV_OBJS_$(CLASS))
OBJECTS         = $(ARCFOURPROV_OBJS:%=$(OBJS_DIR)/%)
ROOTMODULE      = $(ROOT_CRYPTO_DIR)/$(MODULE)

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

#       set signing mode
ELFSIGN_MOD     = $(ELFSIGN_CRYPTO)

#
#       Define targets
#
```



```
ALL_TARGET      = $(BINARY)
LINT_TARGET     = $(MODULE).lint
INSTALL_TARGET  = $(BINARY) $(ROOTMODULE)

#
# Overrides
#
CPPFLAGS        += -I$(COM_DIR)
CLEANFILES      += arcfour-x86_64.s

#
# 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_PTRDIFF_OVERFLOW

#
#       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)

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

$(OBJS_DIR)/arcfour-x86_64.o: arcfour-x86_64.s
        $(COMPILE.s) -o $@ ${@F:.o=.s}
$(OBJS_DIR)/arcfour_crypt_amd64.o: $(COM_DIR)/amd64/arcfour_crypt_amd64.s
        $(COMPILE.s) -o $@ $(COM_DIR)/amd64/arcfour_crypt_amd64.s
        $(POST_PROCESS_O)

$(OBJS_DIR)/arcfour-x86_64.ln: arcfour-x86_64.s
        @($(LHEAD) $(LINT.s) ${@F:.ln=.s} $(LTAIL))
$(OBJS_DIR)/arcfour_crypt_amd64.ln: $(COM_DIR)/amd64/arcfour_crypt_amd64.s
        @($(LHEAD) $(LINT.c) $(COM_DIR)/amd64/arcfour_crypt_amd64.s $(LTAIL))

arcfour-x86_64.s: $(COM_DIR)/amd64/arcfour-x86_64.pl
        $(PERL) $? $@
```