

Get Started with Intel® Integrated Performance Primitives Cryptography

Contents

Chapter 1: Get Started with Intel® Integrated Performance Primitives Cryptography for Intel® oneAPI Base Toolkit

Get Started with Intel® Integrated Performance Primitives Cryptography for Intel® oneAPI Base Toolkit

1

Intel® Integrated Performance Primitives (Intel® IPP) Cryptography is a software library that provides a broad range of secure and efficient cryptographic algorithm implementations.

The library is delivered as a part of [Intel® oneAPI Base Toolkit](#). You may install [specific library version](#) as well. This get started guide assumes you have installed Intel IPP Cryptography library as a part of the toolkit.

Prerequisites (macOS*)

Set Environment Variables

After installing Intel IPP Cryptography, set the `IPPCRYPTOROOT`, `LD_LIBRARY_PATH` environment variables by running the script appropriate to your target platform architecture. The scripts are available in `<install dir>/ippcp/bin`.

By default, the `<install dir>` is `/opt/intel/oneapi`. See [Intel IPP high-level directories structure](#).

Build and Run Your First Intel® IPP Cryptography Application (macOS*)

The code example below represents a short application to help you get started with Intel IPP Cryptography:

```
#include "ippcp.h"
#include <stdio.h>

#define ippCPUID_MMX          0x00000001  /* Intel® Architecture MMX™
technology                      */
#define ippCPUID_SSE          0x00000002  /* Intel® Streaming SIMD
Extensions                      */
#define ippCPUID_SSE2        0x00000004  /* Intel® Streaming SIMD Extensions
2                               */
#define ippCPUID_SSE3        0x00000008  /* Intel® Streaming SIMD Extensions
3                               */
#define ippCPUID_SSSE3       0x00000010  /* Supplemental Streaming SIMD Extensions
3                               */
#define ippCPUID_MOVBE       0x00000020  /* The processor supports the MOVBE
instruction                      */
#define ippCPUID_SSE41       0x00000040  /* Intel® Streaming SIMD Extensions
4.1                             */
#define ippCPUID_SSE42       0x00000080  /* Intel® Streaming SIMD Extensions
4.2                             */
#define ippCPUID_AVX         0x00000100  /* Intel® Advanced Vector Extensions (Intel®
AVX)                             */
#define ippAVX_ENABLEDBYOS   0x00000200  /* The operating system supports Intel
AVX                             */
#define ippCPUID_AES         0x00000400  /* Intel® AES New Instructions (Intel® AES-
NI)                             */
#define ippCPUID_CLMUL       0x00000800  /* PCLMULQDQ
instruction                      */
#define ippCPUID_ABR         0x00001000  /*
```

```

Reserved                                                                    */
#define ippCPUID_RDRAND              0x00002000 /* Read Random Number
instructions                          */
#define ippCPUID_F16C                0x00004000 /* Float16
instructions                          */
#define ippCPUID_AVX2                0x00008000 /* Intel® Advanced Vector Extensions
2                                  */
#define ippCPUID_ADCOX              0x00010000 /* ADCX and ADOX
instructions                          */
#define ippCPUID_RDSEED             0x00020000 /* The RDSEED
instruction                          */
#define ippCPUID_PREFETCHW          0x00040000 /* The PREFETCHW
instruction                          */
#define ippCPUID_SHA                0x00080000 /* Intel® SHA
Extensions                          */
#define ippCPUID_AVX512F            0x00100000 /* Intel® Advanced Vector Extensions 512 (Intel®
AVX-512) Foundation instructions */
#define ippCPUID_AVX512CD           0x00200000 /* Intel AVX-512 Conflict Detection
instructions                          */
#define ippCPUID_AVX512ER           0x00400000 /* Intel AVX-512 Exponential & Reciprocal
instructions                          */
#define ippCPUID_AVX512PF           0x00800000 /* Intel AVX-512 Prefetch
instructions                          */
#define ippCPUID_AVX512BW           0x01000000 /* Intel AVX-512 Byte & Word
instructions                          */
#define ippCPUID_AVX512DQ           0x02000000 /* Intel AVX-512 DWord & QWord
instructions                          */
#define ippCPUID_AVX512VL           0x04000000 /* Intel AVX-512 Vector Length
extensions                          */
#define ippCPUID_AVX512VBMI         0x08000000 /* Intel AVX-512 Vector Bit Manipulation
instructions                          */
#define ippCPUID_MPX                0x10000000 /* Intel® Memory Protection
Extensions                          */
#define ippCPUID_AVX512_4FMADDPS     0x20000000 /* Intel AVX-512 DL floating-point single
precision                          */
#define ippCPUID_AVX512_4VNNIW      0x40000000 /* Intel AVX-512 DL enhanced word variable
precision                          */
#define ippCPUID_KNC                0x80000000 /* Intel® Xeon Phi™
Coprocessor                        */

int main(int argc, char* argv[])
{
    const IppLibraryVersion *lib;
    IppStatus status;
    Ipp64u mask, emask;

    /* Get Intel IPP Cryptography library version info */
    lib = ippcpGetLibVersion();
    printf("%s %s\n", lib->Name, lib->Version);

    /* Get CPU features and features enabled with selected library level */
    status = ippcpGetCpuFeatures( &mask );
    if( ippStsNoErr == status ) {
        emask = ippcpGetEnabledCpuFeatures();
        printf("Features supported by CPU\thy Intel® Integrated Performance Primitives Cryptography
\n");
        printf("-----\n");
        printf(" ippCPUID_MMX      = ");

```

```

printf("%c\t%c\t", ( mask & ippCPUID_MMX ) ? 'Y':'N', ( emask & ippCPUID_MMX ) ? 'Y':'N');
printf("Intel® Architecture MMX technology supported\n");
printf(" ippCPUID_SSE      = ");
printf("%c\t%c\t", ( mask & ippCPUID_SSE ) ? 'Y':'N', ( emask & ippCPUID_SSE ) ? 'Y':'N');
printf("Intel® Streaming SIMD Extensions\n");
printf(" ippCPUID_SSE2      = ");
printf("%c\t%c\t", ( mask & ippCPUID_SSE2 ) ? 'Y':'N', ( emask & ippCPUID_SSE2 ) ? 'Y':'N');
printf("Intel® Streaming SIMD Extensions 2\n");
printf(" ippCPUID_SSE3      = ");
printf("%c\t%c\t", ( mask & ippCPUID_SSE3 ) ? 'Y':'N', ( emask & ippCPUID_SSE3 ) ? 'Y':'N');
printf("Intel® Streaming SIMD Extensions 3\n");
printf(" ippCPUID_SSSE3      = ");
printf("%c\t%c\t", ( mask & ippCPUID_SSSE3 ) ? 'Y':'N', ( emask & ippCPUID_SSSE3 ) ?
'Y':'N');
printf("Supplemental Streaming SIMD Extensions 3\n");
printf(" ippCPUID_MOVBE      = ");
printf("%c\t%c\t", ( mask & ippCPUID_MOVBE ) ? 'Y':'N', ( emask & ippCPUID_MOVBE ) ?
'Y':'N');
printf("The processor supports MOVBE instruction\n");
printf(" ippCPUID_SSE41      = ");
printf("%c\t%c\t", ( mask & ippCPUID_SSE41 ) ? 'Y':'N', ( emask & ippCPUID_SSE41 ) ?
'Y':'N');
printf("Intel® Streaming SIMD Extensions 4.1\n");
printf(" ippCPUID_SSE42      = ");
printf("%c\t%c\t", ( mask & ippCPUID_SSE42 ) ? 'Y':'N', ( emask & ippCPUID_SSE42 ) ?
'Y':'N');
printf("Intel® Streaming SIMD Extensions 4.2\n");
printf(" ippCPUID_AVX        = ");
printf("%c\t%c\t", ( mask & ippCPUID_AVX ) ? 'Y':'N', ( emask & ippCPUID_AVX ) ? 'Y':'N');
printf("Intel® Advanced Vector Extensions (Intel® AVX) instruction set\n");
printf(" ippAVX_ENABLEDBYOS = ");
printf("%c\t%c\t", ( mask & ippAVX_ENABLEDBYOS ) ? 'Y':'N', ( emask & ippAVX_ENABLEDBYOS ) ?
'Y':'N');
printf("The operating system supports Intel® AVX\n");
printf(" ippCPUID_AES        = ");
printf("%c\t%c\t", ( mask & ippCPUID_AES ) ? 'Y':'N', ( emask & ippCPUID_AES ) ? 'Y':'N');
printf("Intel® AES instruction\n");
printf(" ippCPUID_SHA        = ");
printf("%c\t%c\t", ( mask & ippCPUID_SHA ) ? 'Y':'N', ( emask & ippCPUID_SHA ) ? 'Y':'N');
printf("Intel® SHA new instructions\n");
printf(" ippCPUID_CLMUL       = ");
printf("%c\t%c\t", ( mask & ippCPUID_CLMUL ) ? 'Y':'N', ( emask & ippCPUID_CLMUL ) ?
'Y':'N');
printf("PCLMULQDQ instruction\n");
printf(" ippCPUID_RDRAND      = ");
printf("%c\t%c\t", ( mask & ippCPUID_RDRAND ) ? 'Y':'N', ( emask & ippCPUID_RDRAND ) ?
'Y':'N');
printf("Read Random Number instructions\n");
printf(" ippCPUID_F16C        = ");
printf("%c\t%c\t", ( mask & ippCPUID_F16C ) ? 'Y':'N', ( emask & ippCPUID_F16C ) ? 'Y':'N');
printf("Float16 instructions\n");
printf(" ippCPUID_AVX2        = ");
printf("%c\t%c\t", ( mask & ippCPUID_AVX2 ) ? 'Y':'N', ( emask & ippCPUID_AVX2 ) ? 'Y':'N');
printf("Intel® Advanced Vector Extensions 2 instruction set\n");
printf(" ippCPUID_AVX512F     = ");
printf("%c\t%c\t", ( mask & ippCPUID_AVX512F ) ? 'Y':'N', ( emask & ippCPUID_AVX512F ) ?
'Y':'N');
printf("Intel® Advanced Vector Extensions 512 Foundation instruction set\n");

```

```

printf(" ippCPUID_AVX512CD  = ");
printf("%c\t%c\t", ( mask & ippCPUID_AVX512CD ) ? 'Y':'N', ( emask & ippCPUID_AVX512CD ) ?
'Y':'N');
printf("Intel® Advanced Vector Extensions 512 Conflict Detection instruction set\n");
printf(" ippCPUID_AVX512ER  = ");
printf("%c\t%c\t", ( mask & ippCPUID_AVX512ER ) ? 'Y':'N', ( emask & ippCPUID_AVX512ER ) ?
'Y':'N');
printf("Intel® Advanced Vector Extensions 512 Exponential & Reciprocal instruction set\n");
printf(" ippCPUID_ADCOX    = ");
printf("%c\t%c\t", ( mask & ippCPUID_ADCOX ) ? 'Y':'N', ( emask & ippCPUID_ADCOX ) ?
'Y':'N');
printf("ADCX and ADOX instructions\n");
printf(" ippCPUID_RDSEED    = ");
printf("%c\t%c\t", ( mask & ippCPUID_RDSEED ) ? 'Y':'N', ( emask & ippCPUID_RDSEED ) ?
'Y':'N');
printf("The RDSEED instruction\n");
printf(" ippCPUID_PREFETCHW = ");
printf("%c\t%c\t", ( mask & ippCPUID_PREFETCHW ) ? 'Y':'N', ( emask & ippCPUID_PREFETCHW ) ?
'Y':'N');
printf("The PREFETCHW instruction\n");
printf(" ippCPUID_KNC        = ");
printf("%c\t%c\t", ( mask & ippCPUID_KNC ) ? 'Y':'N', ( emask & ippCPUID_KNC ) ? 'Y':'N');
printf("Intel® Xeon Phi™ Coprocessor instruction set\n");
}
return 0;
}

```

This application consists of two sections:

1. Get the library layer name and version.
2. Show the hardware optimizations used by the selected library layer and supported by CPU.

To build the code example above, follow the steps:

1. Paste the code into the editor of your choice.
2. Make sure the compiler and Intel IPP Cryptography variables are set in your shell.
3. Compile with the following command: `icc ippcptest.cpp -o ippcptest -I$IPPCRYPTOROOT/
include -L$IPPCRYPTOROOT/lib/<arch> -lippcp.`
4. Run the application.

Training and Documentation

Document	Description
Intel® IPP Cryptography Developer Guide	Provides detailed guidance on Intel IPP Cryptography library configuration, development environment, and linkage modes.
Intel® IPP Cryptography Developer Reference	Contains detailed descriptions of the Intel IPP Cryptography functions.
Intel® Integrated Performance Primitives	Intel® IPP product page. See this page for support and online documentation.

Notices and Disclaimers

Intel, the Intel logo, Intel Atom, Intel Core, Intel Xeon Phi, VTune and Xeon are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

© Intel Corporation.

This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you (**License**). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

Product and Performance Information
Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex . Notice revision #20201201