Apache Commons Crypto: Another wheel of Apache Commons

By Xianda Ke

Dev@Intel

APACHECON Europe

intel®

About me

- xianda.ke(at)intel.com

- Now, I’m contributing to:
    - Apache Pig(Pig on Spark)
    - Apache Commons Crypto
    - OpenJDK

Agenda

- Brief introduction to cryptography
- Why we create another wheel?
- Features & API samples
- accelerating big data Apache Commons Crypto
- How to contribute

History

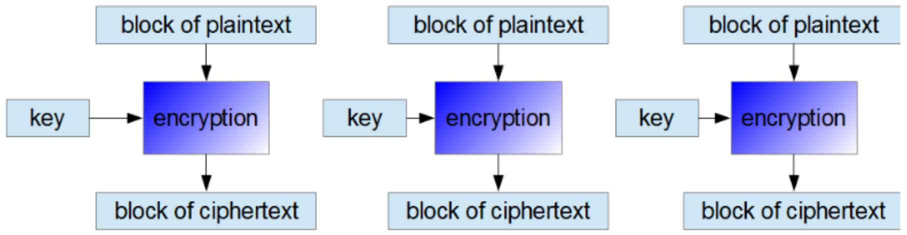
- Intel’s big data team created Chimera
  - a cryptographic library optimized with hardware acceleration
  - it wraps OpenSSL
- Apache Commons Crypto stems from Chimera

A glance at cryptographic primitives

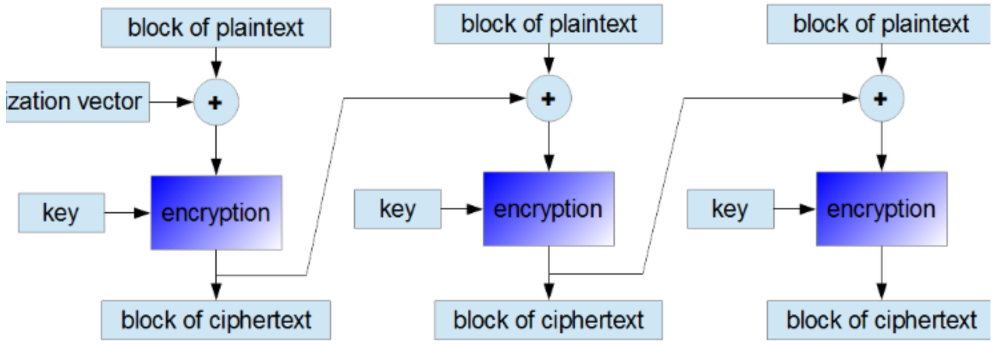
- Symmetric encryption(Secret Key) : **confidentiality**
  DES/3DES, RC4, AES, …
- Asymmetric encryption (Public Key): **authentication /key exchange**
  RSA, Diffie-Hellman, ECC, ...
- Hash algorithms: **integrity**
  MD5 , SHA-1, SHA-2, GMAC, …
- Authenticated Encryption : **confidentiality + integrity**
  RC4 + HmacMD5, RC4 + HMAC-SHA-1, AES-GCM, …
- Random Number generator
- ...

block cipher **mode** of operation?

It blurs the cipher output to avoid creating identical output ciphertext blocks from identical input data.



Encryption in the ECB mode



Encryption in the CBC mode

JCE(Java Cryptography Extension) in hand, why we create another wheel?

Intel’s engineers are working in performance-sensitive area(big data), they need a handy Java cryptographic library, which is

secure **&** **fast**

the popular symmetric encryption algorithms:

DES is broken

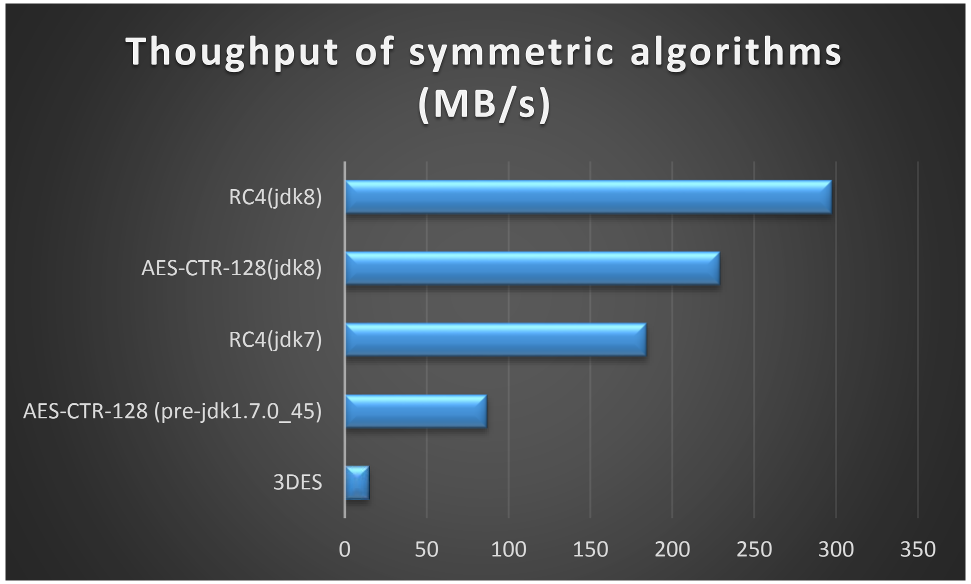
3DES is too slow

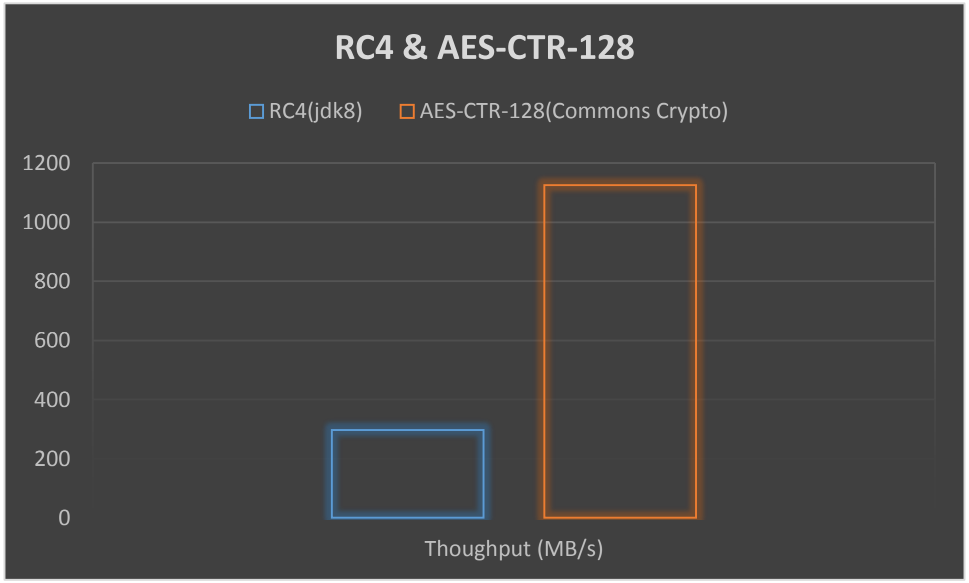
- RC4 is well known for its speed.

  Insecure, IETF has published RFC 7465 to prohibit the use of RC4, 2015

Throughput of symmetric encryption algorithms

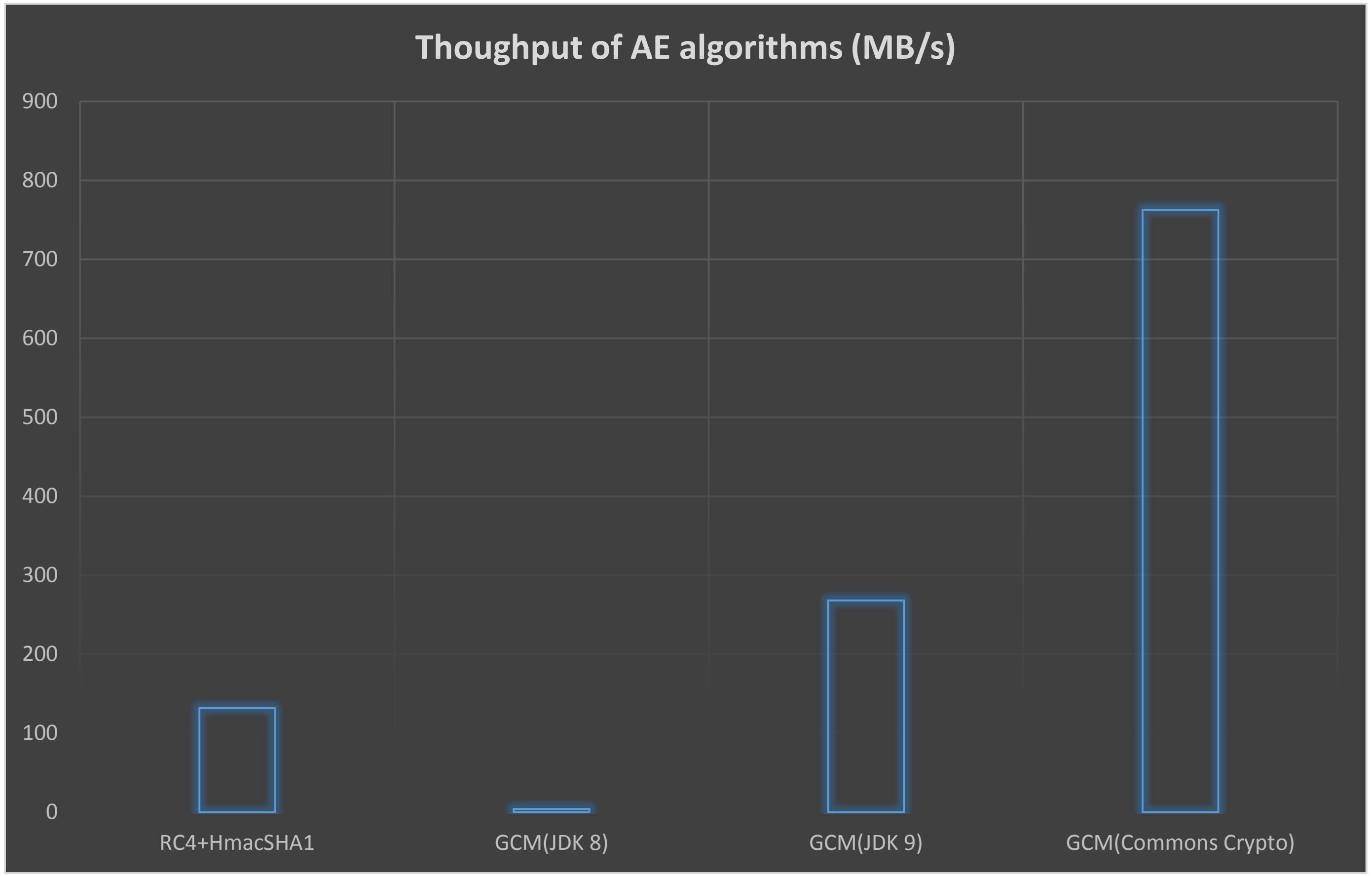
AES is secure and considered as industry standard. However, its performance is not good enough in JCE





*CPU: Xeon(R) CPU E5-2690 v2 @ 3.00GHz. Memory: DDR3, 64 G

Throughput of Authenticated Encryption algorithms



RC4 + HmacMD5 (broken)

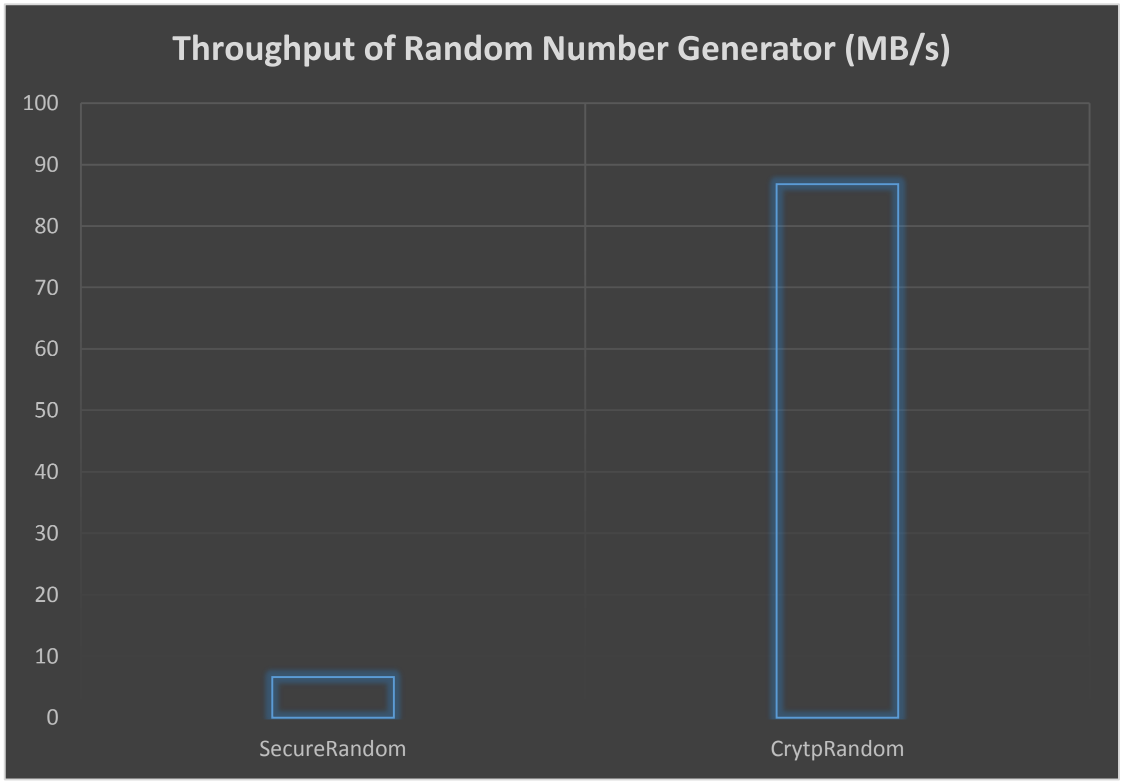
RC4 + HmacSHA1 (mostly)

AES-GCM: designed for high performance, however it has poor performance in Java

a better alternative: AES-GCM with Commons Crypto!

Random Number Generator (RNG)

- Random is Pseudo-Random Number Generator(PRNG), cryptographically insecure
- SecureRandom is cryptographically strong, but has poor performance
- CryptoRandom is True Random Number Generator, and is ~13X faster than SecureRandom



Commons Crypto outperforms JCE(Java 8)!

- AES has advantages over RC4/3DES in both security and speed
- AES-CTR/AES-CBC in Commons Crypto is 5~7X faster than JCE (java 8)
- AES-GCM in Commons Crypto 60~190X faster than JCE(Java 8)
- TRNG CryptoRandom is ~13X faster than SecureRandom

AES is **secure**, and it can be very **fast** with Commons Crypto. That’s why we create a wheel

How does Commons Crypto get so fast?

- OpenSSL is the low-level engine

- Native code enables hardware acceleration
  - Intel® Advanced Encryption Standard New Instructions (AES-NI)
  - Intel® Carry-Less Multiplication Instruction (PCLMULQDQ for GCM)
  - Intel® Secure Key (True Random Number Generator)
  - Other platforms(SPARC, PowerPC) acceleration…

- Optimized algorithms & implementation
  - Shay Gueron's papers, parallelism(Pipelined instructions), …

JCE is also becoming faster:

JDK 7

JDK-7184394 (add intrinsics to use AES instructions) jdk1.7.0_45
Hardware acceleration is enable, but the implementation is not optimized.

JDK 9:

JDK-8073108 AES-GCM, ~60X gain, but still falls far behind Commons Crypto

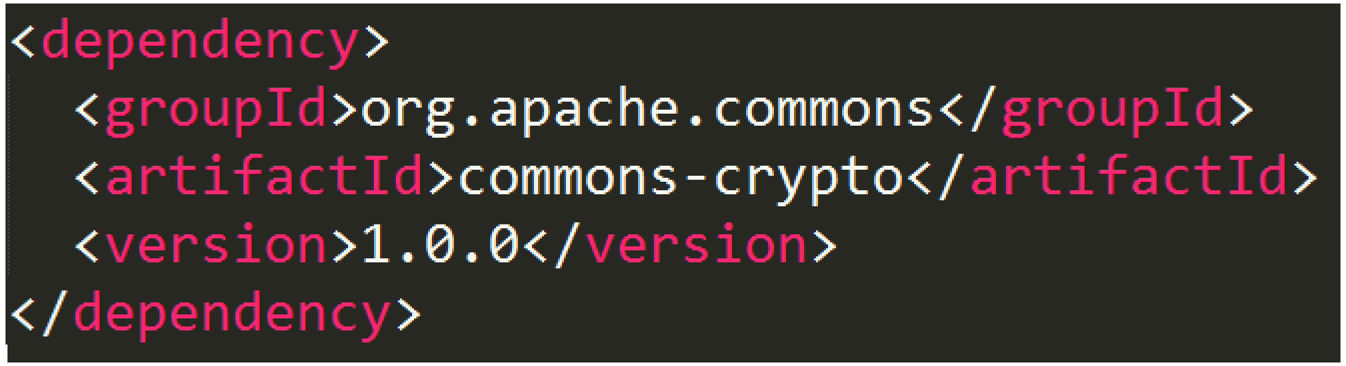
Collaborating with JVM team, we have contributed two patches to HotSpot(JDK 9):
JDK-8143925 x86 AES-CTR, 5~8X gain compared with JDK 8
JDK-8152354 x86 AES-CBC Decryption, 15%~50% gain compared with JDK 8

Java 9 ? We have to wait for years to adopt Java 9 in production… ☹

Apache Commons Crypto’s features

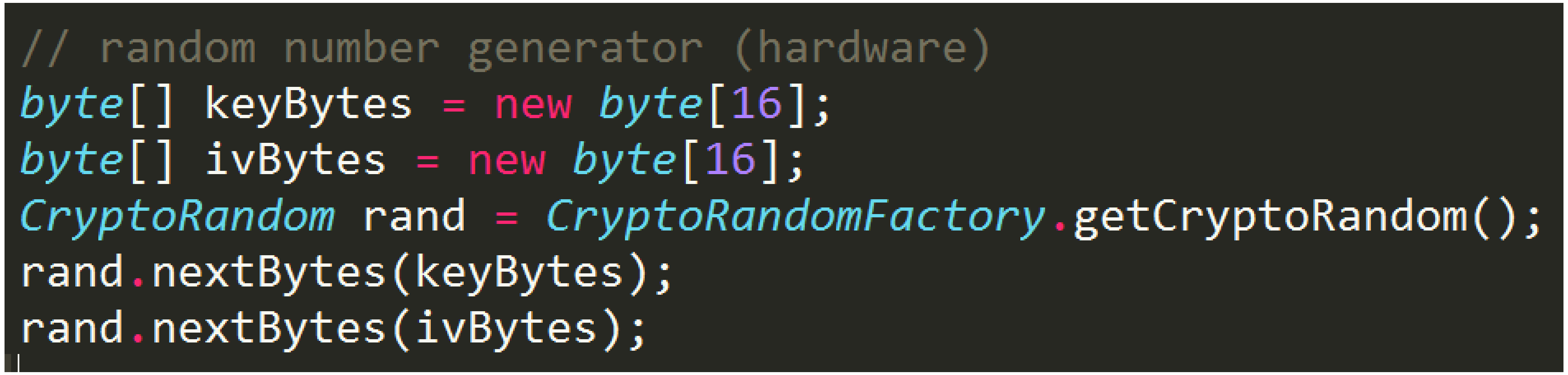
- Cipher API for low level cryptographic operations. (AES-CBC, AES-CTR)
- Java stream API for high level stream encryption/decryption.
- Secure true random number generator.

maven repo:

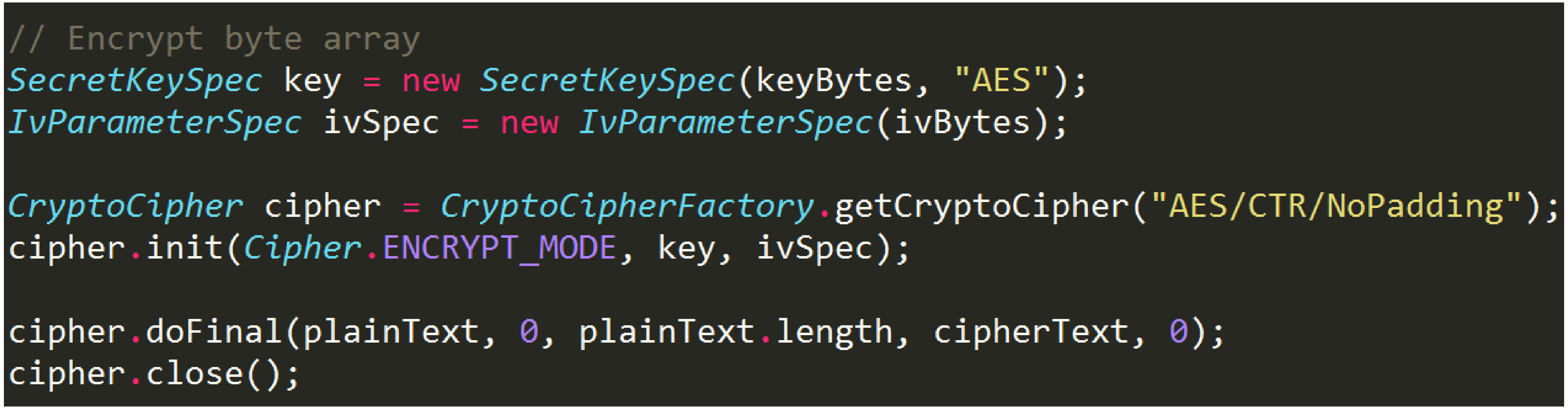


```
<dependency>
  <groupId>org.apache.commons</groupId>
  <artifactId>commons-crypto</artifactId>
  <version>1.0.0</version>
</dependency>
```

API & Samples



```java
// random number generator (hardware)
byte[] keyBytes = new byte[16];
byte[] ivBytes = new byte[16];
CryptoRandom rand = CryptoRandomFactory.getCryptoRandom();
rand.nextBytes(keyBytes);
rand.nextBytes(ivBytes);
```

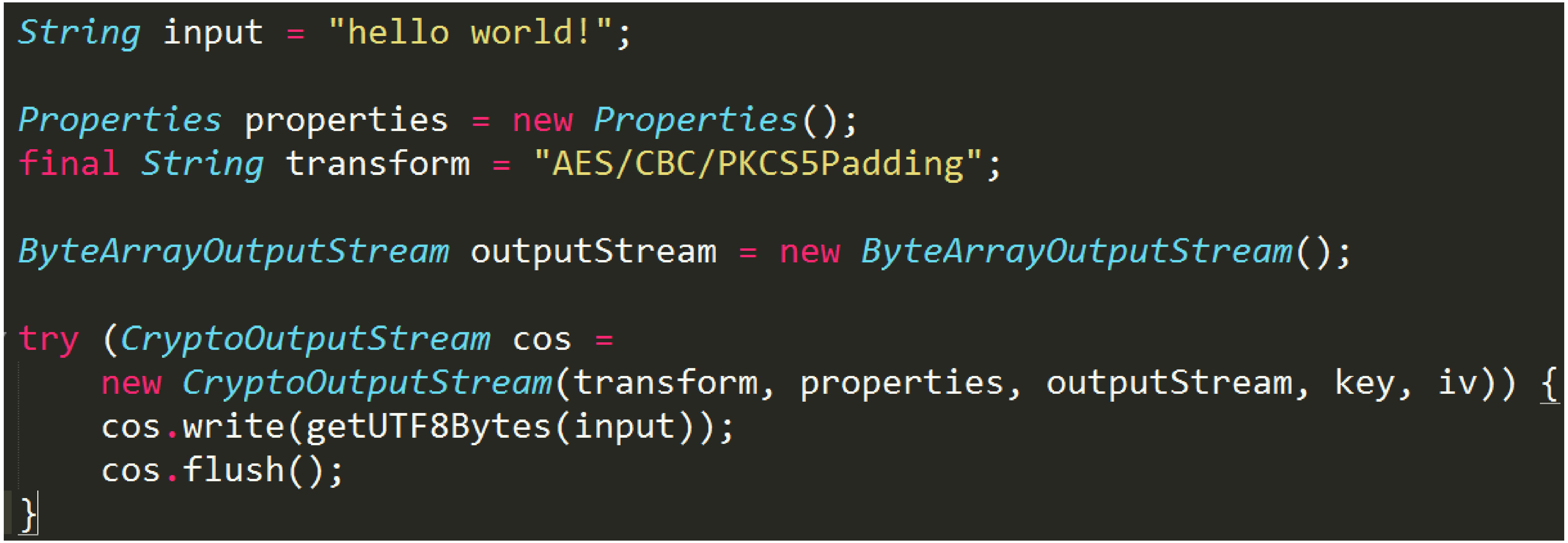


```java
// Encrypt byte array
SecretKeySpec key = new SecretKeySpec(keyBytes, "AES");
IvParameterSpec ivSpec = new IvParameterSpec(ivBytes);

CryptoCipher cipher = CryptoCipherFactory.getCryptoCipher("AES/CTR/NoPadding");
cipher.init(Cipher.ENCRYPT_MODE, key, ivSpec);

cipher.doFinal(plainText, 0, plainText.length, cipherText, 0);
cipher.close();
```

API & Samples

```java
// Decrypt ByteBuffer
ByteBuffer inBuffer = ByteBuffer.allocateDirect(1000);
ByteBuffer outBuffer = ByteBuffer.allocateDirect(1000);
inBuffer.put(cipherText);
inBuffer.flip();

cipher.init(Cipher.DECRYPT_MODE, key, ivSpec);
cipher.doFinal(inBuffer, outBuffer);
cipher.close();
```

API & Samples



```java
String input = "hello world!";

Properties properties = new Properties();
final String transform = "AES/CBC/PKCS5Padding";

ByteArrayOutputStream outputStream = new ByteArrayOutputStream();

try (CryptoOutputStream cos =
     new CryptoOutputStream(transform, properties, outputStream, key, iv)) {
     cos.write(getUTF8Bytes(input));
     cos.flush();
}
```

Intel is accelerating big data with Commons Crypto

- Add encrypted shuffle in spark (SPARK-5682)

- Optimize HDFS Encrypted Transport performance (HDFS-6606)
  Replace 3DES & RC4 with AES

- Improve performance for RPC encryption (HBASE-16414)
  1.2X ~ 2.2X performance gain

in development :

- AES support for over-the-wire encryption(SPARK-13331)
  brings 12.5% overall performance
- Optimize Hadoop RPC encryption performance(HADOOP-10768)

Status & Plan:

Status:

- v1.0.0 is released. AES-CBC, AES-CTR are supported
- AES-GCM (will be released in v1.1.0)

Plan: to cover more facets in cryptography, such as:

- Support Asymmetric key algorithms: RSA, DSA…
- More hashing algorithms: SHA-1, SHA-256…

How to contribute:

- SCM: https://github.com/apache/commons-crypto
- Apache Bugtracker (JIRA): https://issues.apache.org/jira/

Thanks to the contributors

- Aaron T Myers
- Andrew Wang
- Chris Nauroth
- Colin P. McCabe
- Dapeng Sun
- Dian Fu
- Dong Chen
- Ferdinand Xu
- Haifeng Chen
- Marcelo Vanzin
- Uma Maheswara Rao G
- Yi Liu
- Colin Ma
- Xianda Ke
- Ke Jia
- George Kankava
- …

Thanks

Q & A