Future implementation / improvement considerations

Used SecretKey, IV and Salt

- In the present implementation, the SecretKey, IV are all given as a plain string, which is provided by the user. In order to enhance the security of the system, it is recommended that a key be derived and that the responsibility for secure key generation be transferred from the user.

More AES operation modes and improve existing

- Implementation of additional AES operating modes (e. g. CBC, etc.)
- Implementation of Padding for AES CTR mode

More Hashing operation modes

- Implementation of implementing additional hashing algorithms.

Use of timestamp services

- With the use of timestamp services one can protect against the potential manipulation of the timestamp and to guarantee the authenticity of the sequence of events.

Specify a specific security provider

- In the present implementation, no specific security provider is specified, a choice that increases portability and aligns with oracle guidelines. This approach also eliminates the introduction of another conditional dependency into the log4j framework.However, in the context of high-security applications, it may be preferable to specify a particular provider to ensure adherence to particularly stringent safety standards.
- It is also noteworthy that the provider BouncyCastle is incompatible with the currently implemented AES CTR mode due to its improper flushing behaviour, which is likely attributable to its use of a BufferedOutputStream. In contrast, the default security provider functions without issues. See here.

Increase Performance for large log files

- At present, in the event that encryption is employed without overwriting the log file that is currently encrypted, the file is first decrypted and subsequently encrypted once more. This method is elementary, yet it is disadvantageous for large files. A more optimal approach would be to implement continuous encryption using AES in CTR mode. Although this mode has been implemented, appending encrypted data necessitates that the IV and counter remain consistent across sessions. In the event that the same IV is reused with appended data, this can result in significant cryptographic vulnerabilities. To address this challenge, the implementation of an IV lifecycle management protocol is imperative.

Expanded Charset

- The present implementation utilizes the UTF-8 and encompasses the majority of symbols, though not all. An alternative solution, whether independent or expanded, might be preferable.