BGZF block-level encryption for VCF, BCF, BAM

(and anything BGZF compressed)

petr.danecek@sanger.ac.uk

2016-11-16

BGZF

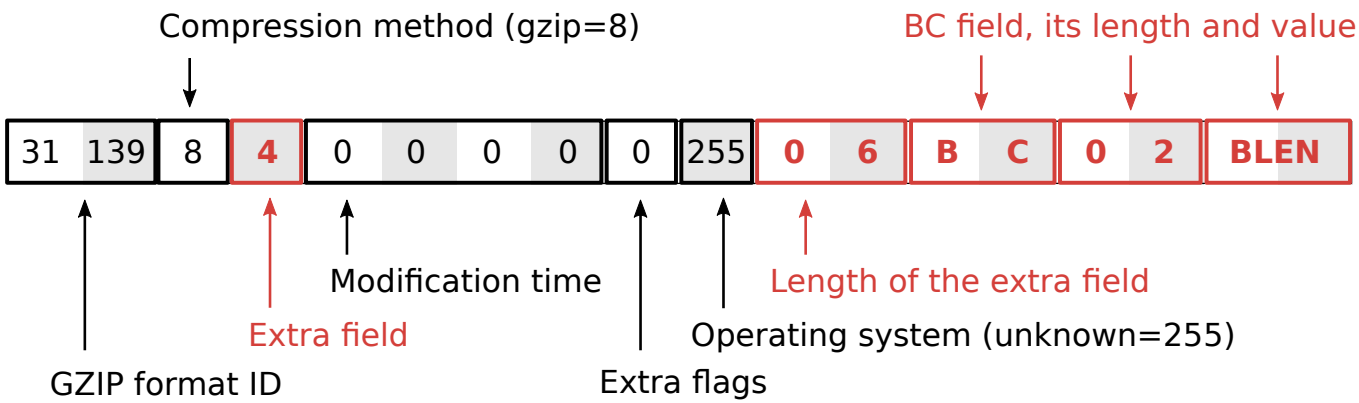
Compression format compatible with GZIP

- used for VCF, BCF, BAM compression
- indexable with tabix, .tbi, and .csi

BGZF file typically consists of many compressed blocks

The size of compressed blocks (BLEN) is limited to 2^{16} bytes

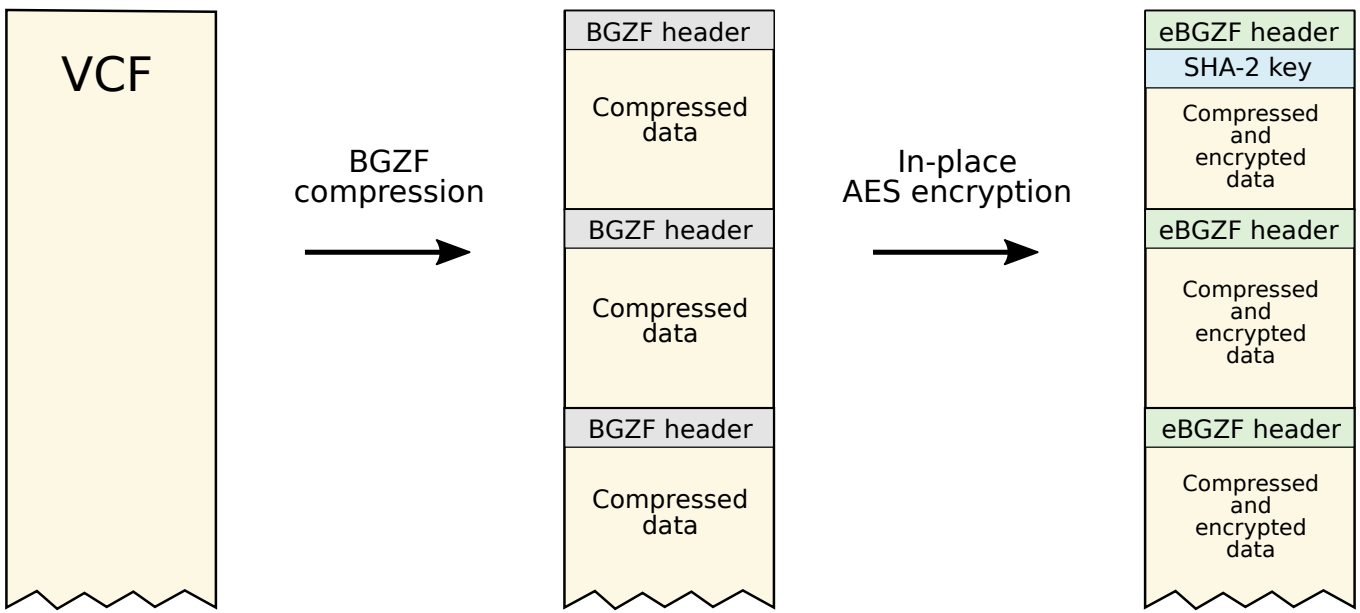
BGZF header, extension of GZIP, RFC 1952



BGZF encryption via AES-256

Advanced Encryption Standard

- symmetric key algorithm (the same key is used for encryption and decryption)
- used by governments, militaries, banks
- no known practical attacks

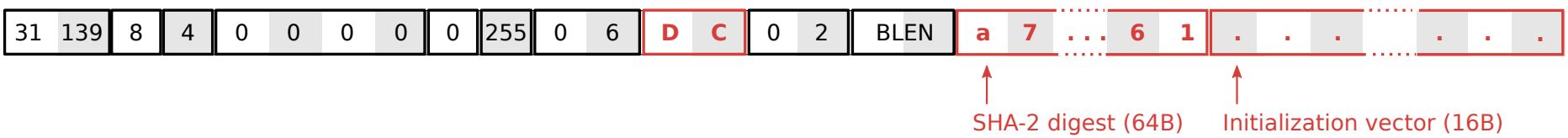


AES-encrypted BGZF

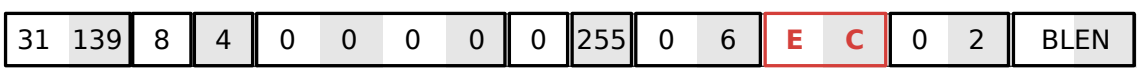
Encrypted blocks use EC or DC tag instead of BC

- EC blocks were in-place encrypted using AES-256 after the compression
- DC block comes first in the encrypted file and contains SHA-2 hash digest of the key used for encryption and the initialization key

First block



All other encrypted blocks



Proof of concept implementation in HTSlib

Transparent encryption and decryption

- openssl-aware HTSlib is required (and nothing else)
- for example, no code changes were required in BCFtools

Library of keys

- plain text file secured only by standard unix permissions
- location stored in the environment variable HTS_KEYS

```
cat hts-keys.txt
# [1] Public SHA256 digest of the key   [2] Private symmetric encryption key
d3f29f62f78df[....]be8f35590516e65ac3   63f89d73e188[...]6c0664d6a3427d82534
0a731c7703376[....]c7f578d9bea9ef0ac2   1820f1666f2d[...]98ecd75a32965f17e0a
7ae24d96d386a[....]04bf0e5cd9a08183d2   484a7bbb8eb1[...]21b5ebfa0a09e34847a
```

Encryption

- controlled by the environment variable HTS_ENC

```
export HTS_KEYS=hts-keys.txt

# plain compression
bgzip -c in.vcf > plain.vcf.gz

# encrypted output
HTS_ENC=0a731c7703376[....]c7f578d9bea9ef0ac2 bgzip -c in.vcf > enc.vcf.gz

# indexing and querying encrypted file with tabix
tabix enc.vcf.gz
tabix enc.vcf.gz 1:100000-100000
```

Practical aspects

Performance

- the cost is negligible, the following tests on a 248M BCF:

```
bcftools-1.3 view test.bcf -Ob -o normal.bcf    # 94.98user 2.18system 1:37.29elapsed
bcftools-ssl view test.bcf -Ob -o enc.bcf       # 97.03user 1.68system 1:38.73elapsed
```

Create random keys

```
# Random key and the public SHA-256 hash
KEY=‘dd if=/dev/urandom bs=1 count=32 2>/dev/null | xxd -ps -c32‘
HASH=‘echo $KEY | openssl sha256 | sed ’s,^.*= ,,’‘
echo -e "$HASH\t$KEY" > hts-keys.txt

# The (trimmed) result:
cat hts-keys.txt
e047893a7f886[....]e54a9cd24f0b430267   beb91c3fa95a[...]04c3052041e02419fba
```

Download and test the code

```
git clone --branch=crypto https://github.com/pd3/htslib.git
cd htslib
autoheader
autoconf
./configure --enable-openssl
make
./test/test.pl

# the only modification in bcftools is compilation with openssl
cd ..
git clone --branch=crypto https://github.com/pd3/bcftools.git
cd bcftools
make
```