Speed estimation:

New user registration: estimated speed cost is equal to speed of transfer of data over network, plus iteration over all clients to check if same name already exists, plus addition of the client into the vector.

User Authentication: Our model does not feature authentication as such, after registration of a new client, the client has to generate new public and private keys, and then using asymetric communication generate symetric key, which will be used for further communication. So estimated cost of this operation is estimated cost of generating pair of RSA keys + estimated cost of generating AES-256 symetric key + speed cost of transfering both over the network.

Obtain list of users: estimated cost is getting list of all client names from list of clients, and then writing these values one by one into json, and then encrypt it using previously generated aes key + speed of network transfer + decryption on the client side using aes key.

Prepare protected message for another user: first, body of message and type of message are written into json, then the json is transformed into QbyteArray and passed gcm.encryptAndTtag function,
which allocates array of size of the byte array + size of expected tag + size of length variable. Then using memcpy, the data of byte array is copient into the array, which is then passed to internal mbedtls_gcm_crypt_and_tag function.

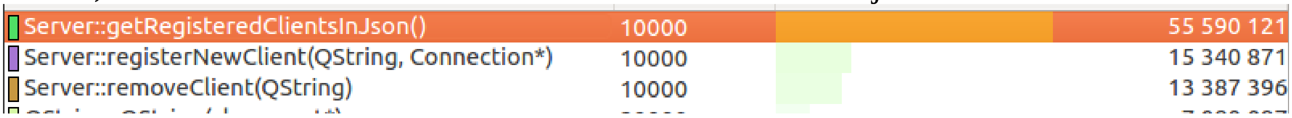
Unprotect message from another user: Copy of QbyteArray is made using sub() function to get to the encrypted data and get pointer to the raw data from the byte array and pass it to mbedtls_gcm_crypt_and_tag() function, so estimated cost is rougly the cost of copying the byte array and cost of mbedtls_gcm_crypt_and_tag().

Send communication request: simple json is created and inserted into values representing type of message and name of client, then the json is transformed into QbyteArray and passed gcm.encryptAndTag function,
which allocates array of size of the byte array + size of expected tag + size of length variable. Then using memcpy, the data of byte array is copient into the array, which is then passed to internal mbedtls_gcm_crypt_and_tag() function.

Get client info: simple json is created and inserted into values representing type of message and name of client, then client info is written into the json using write() function then the json is transformed into QbyteArray and passed gcm.encryptAndTag function,
which allocates array of size of the byte array + size of expected tag + size of length variable. Then using memcpy, the data of byte array is copient into the array, which is then passed to internal mbedtls_gcm_crypt_and_tag() function.

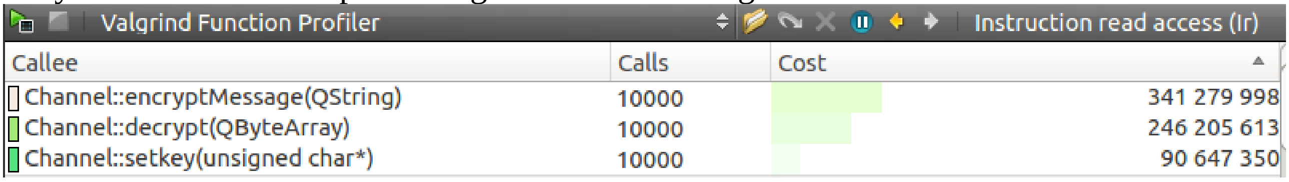
Send communication reply: json is created and inserted into values representing type of message, name of the client as well as result of the reply, so either accept or decline,then the json is transformed into QbyteArray and passed gcm.encryptAndTag function, which allocates array of size of the byte array + size of expected tag + size of length variable. Then using memcpy, the data of byte array is copient into the array, which is then passed to internal mbedtls_gcm_crypt_and_tag() function.

Server functions:
Relatively fast compared to cryptographic functions, most of the time is taken by get all clients, which iterates over clients and inserts their names into json



Server::getRegisteredClientsInJson()	10000	55 590 121
Server::registerNewClient(QString, Connection*)	10000	15 340 871
Server::removeClient(QString)	10000	13 387 396

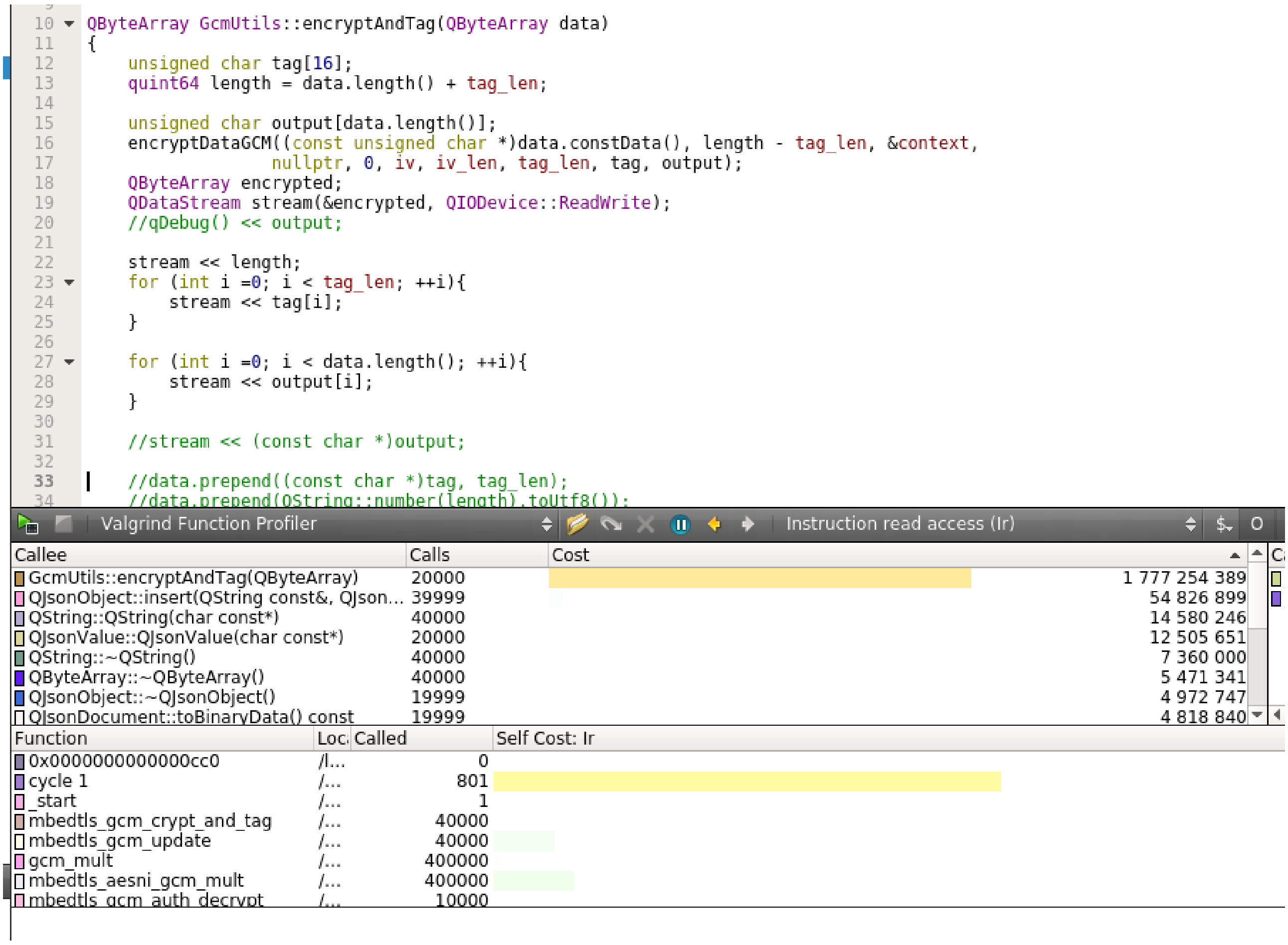
Client functions:
They all take a lot more processing time due to calling mbedtls functions



Callee	Calls	Cost
Channel::encryptMessage(QString)	10000	341 279 998
Channel::decrypt(QByteArray)	10000	246 205 613
Channel::setkey(unsigned char*)	10000	90 647 350

Optimization: in function gcm.encryptAndTag(), first implementation was to write encrypted string returned by mbedtls_gcm_crypt_and_tag() using operator <<, but we had to write it character by character because there was no ending 0 or there was ending 0 in the middle of the string. The optimized version uses memcpy function into offseted array of chars, then prepends it with tag and length. Then resulting byte array is assigned the array of chars using createFromRaw() function. As can be seen in the picture, this has reduced the cost of this operation by more than 50%.

Here encrypt and tag function can be seen, with total cost of 1 777 millions

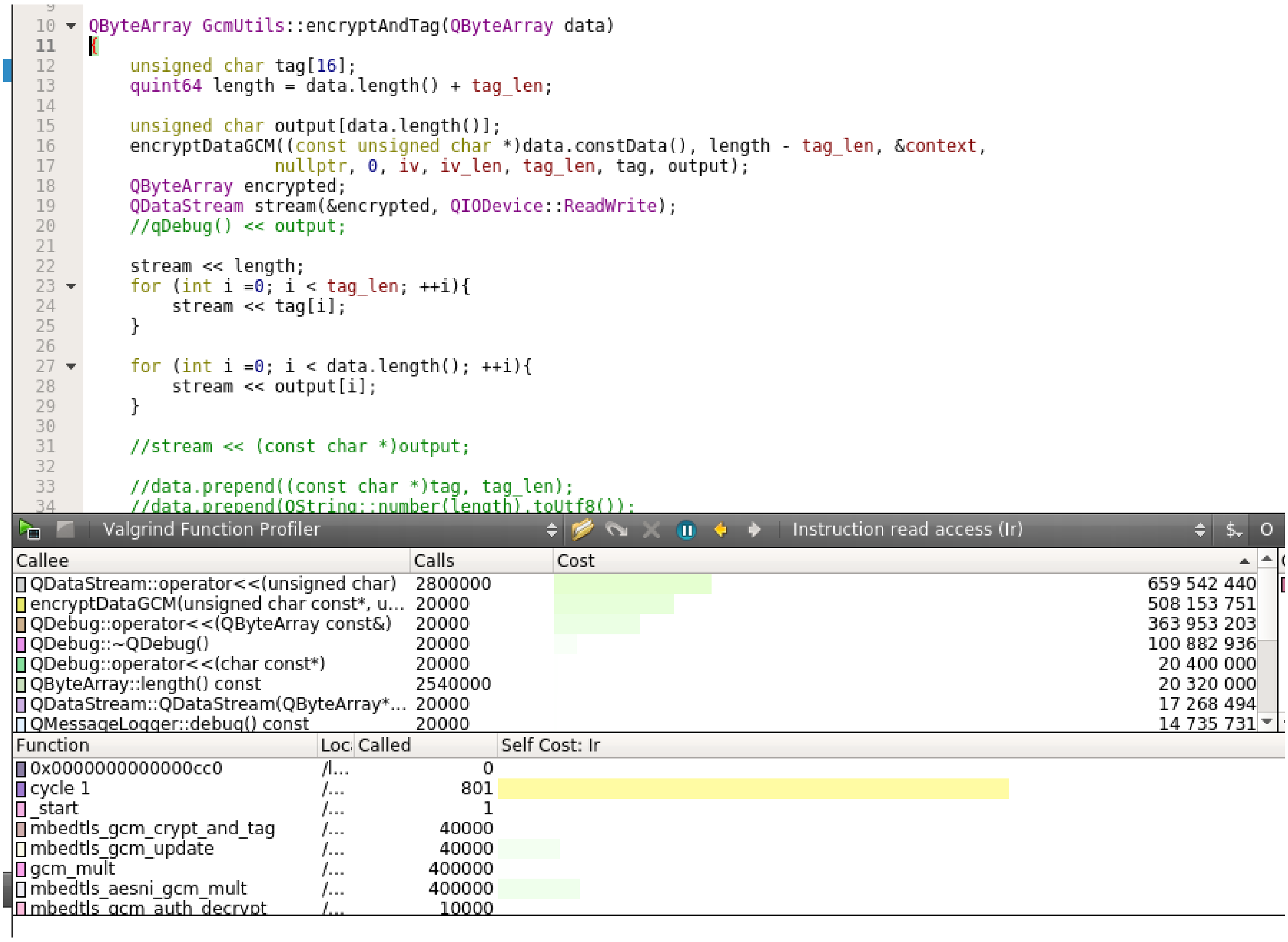


```
QByteArray GcmUtils::encryptAndTag(QByteArray data)
{
    unsigned char tag[16];
    quint64 length = data.length() + tag_len;

    unsigned char output[data.length()];
    encryptDataGCM((const unsigned char *)data.constData(), length - tag_len, &context,
                  nullptr, 0, iv, iv_len, tag_len, tag, output);
    QByteArray encrypted;
    QDataStream stream(&encrypted, QIODevice::ReadWrite);
    //qDebug() << output;

    stream << length;
    for (int i =0; i < tag_len; ++i){
        stream << tag[i];
    }

    for (int i =0; i < data.length(); ++i){
        stream << output[i];
    }

    //stream << (const char *)output;

    //data.prepend((const char *)tag, tag_len);
    //data.prepend(QString::number(length).toUtf8());
```

Callee	Calls	Cost
GcmUtils::encryptAndTag(QByteArray)	20000	1 777 254 389
QJsonObject::insert(QString const&, QJson…	39999	54 826 899
QString::QString(char const*)	40000	14 580 246
QJsonValue::QJsonValue(char const*)	20000	12 505 651
QString::~QString()	40000	7 360 000
QByteArray::~QByteArray()	40000	5 471 341
QJsonObject::~QJsonObject()	19999	4 972 747
QJsonDocument::toBinaryData() const	19999	4 818 840

Function	Loc	Called	Self Cost: Ir
0x0000000000000cc0	/l…	0	
cycle 1	/…	801	
_start	/…	1	
mbedtls_gcm_crypt_and_tag	/…	40000	
mbedtls_gcm_update	/…	40000	
gcm_mult	/…	400000	
mbedtls_aesni_gcm_mult	/…	400000	
mbedtls_gcm_auth_decrypt	/…	10000	

Here we can see encrypt and tag from inside. Notice QdataStream::operator<< has cost of 659 millions.



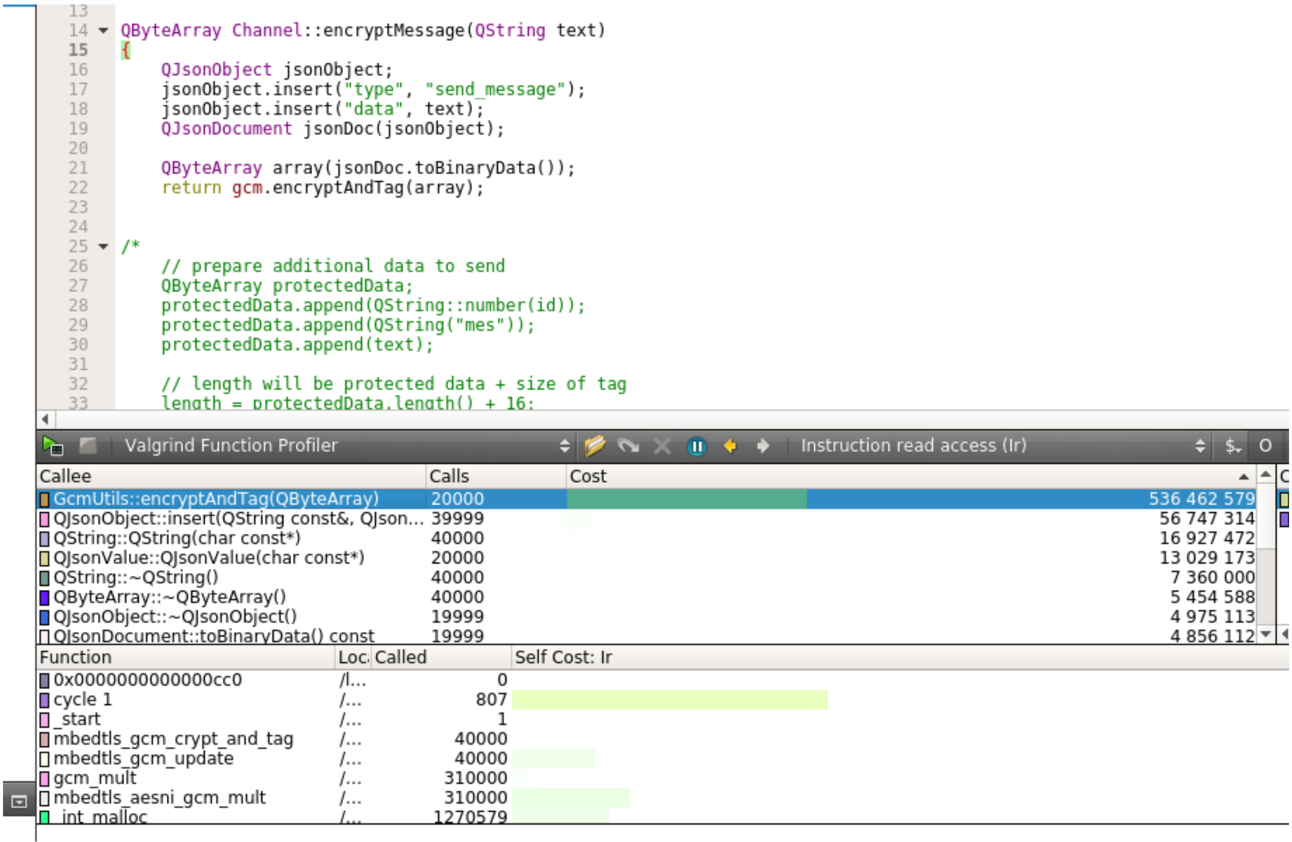
```
QByteArray GcmUtils::encryptAndTag(QByteArray data)
{
    unsigned char tag[16];
    quint64 length = data.length() + tag_len;

    unsigned char output[data.length()];
    encryptDataGCM((const unsigned char *)data.constData(), length - tag_len, &context,
                   nullptr, 0, iv, iv_len, tag_len, tag, output);
    QByteArray encrypted;
    QDataStream stream(&encrypted, QIODevice::ReadWrite);
    //qDebug() << output;

    stream << length;
    for (int i =0; i < tag_len; ++i){
        stream << tag[i];
    }

    for (int i =0; i < data.length(); ++i){
        stream << output[i];
    }

    //stream << (const char *)output;

    //data.prepend((const char *)tag, tag_len);
    //data.prepend(QString::number(length).toUtf8());
```

Callee	Calls	Cost
QDataStream::operator<<(unsigned char)	2800000	659 542 440
encryptDataGCM(unsigned char const*, u...	20000	508 153 751
QDebug::operator<<(QByteArray const&)	20000	363 953 203
QDebug::~QDebug()	20000	100 882 936
QDebug::operator<<(char const*)	20000	20 400 000
QByteArray::length() const	2540000	20 320 000
QDataStream::QDataStream(QByteArray*...	20000	17 268 494
QMessageLogger::debug() const	20000	14 735 731

Function	Loc	Called	Self Cost: Ir
0x0000000000000cc0	/l...	0	
cycle 1	/...	801	
_start	/...	1	
mbedtls_gcm_crypt_and_tag	/...	40000	
mbedtls_gcm_update	/...	40000	
gcm_mult	/...	400000	
mbedtls_aesni_gcm_mult	/...	400000	
mbedtls_gcm_auth_decrypt	/...	10000	

Here is optimized version of encryptAndTag. Notice That the cost is reduced down to 536 millions.

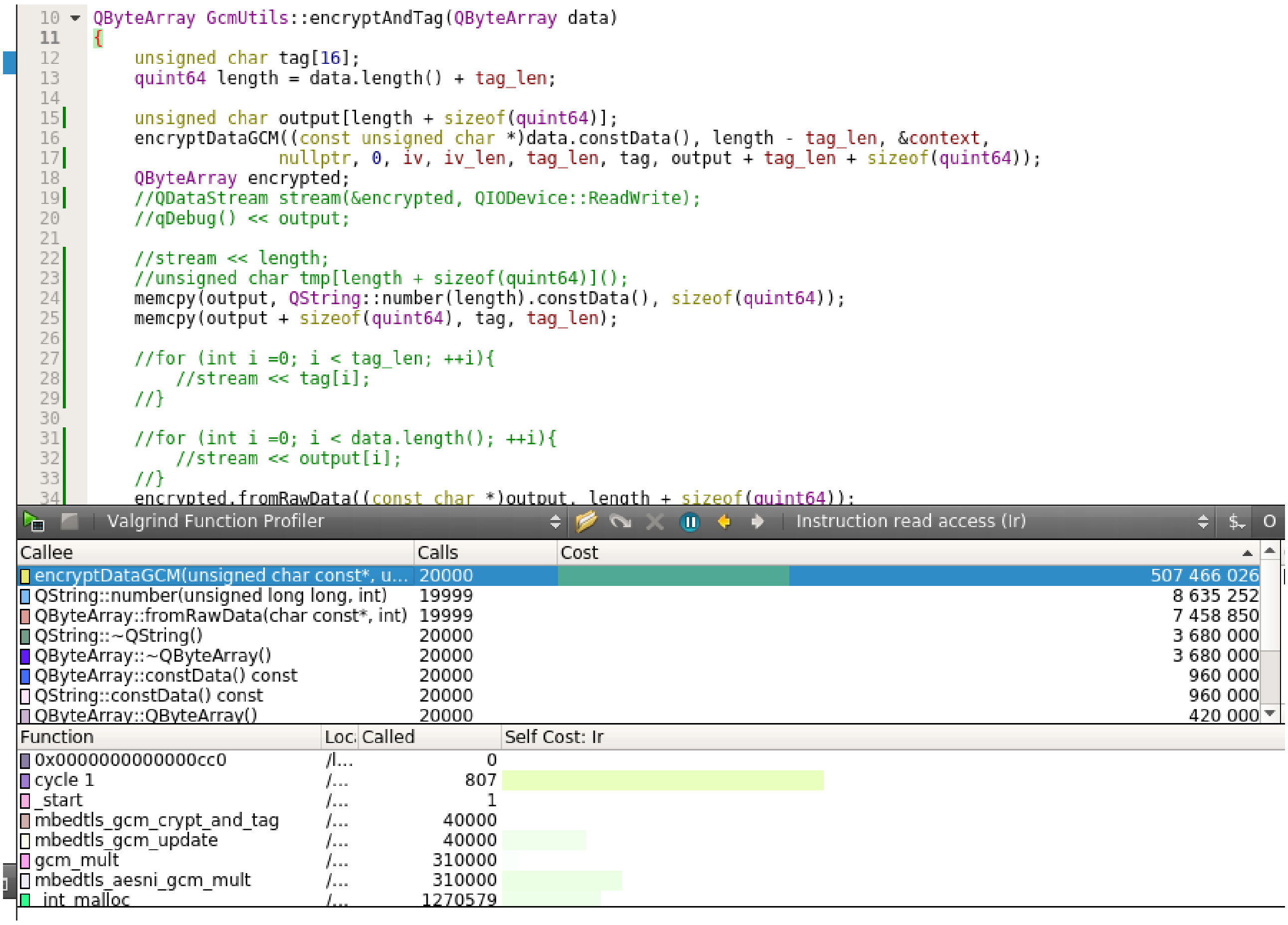
Here is optimized version of encryptAndTag from inside. Notice there is no QdataStream::operator <<, only encrytDataGcm().



```
QByteArray Channel::encryptMessage(QString text)
{
    QJsonObject jsonObject;
    jsonObject.insert("type", "send_message");
    jsonObject.insert("data", text);
    QJsonDocument jsonDoc(jsonObject);

    QByteArray array(jsonDoc.toBinaryData());
    return gcm.encryptAndTag(array);


/*
    // prepare additional data to send
    QByteArray protectedData;
    protectedData.append(QString::number(id));
    protectedData.append(QString("mes"));
    protectedData.append(text);

    // length will be protected data + size of tag
    length = protectedData.length() + 16;
```

Callee	Calls	Cost
GcmUtils::encryptAndTag(QByteArray)	20000	536 462 579
QJsonObject::insert(QString const&, QJson...	39999	56 747 314
QString::QString(char const*)	40000	16 927 472
QJsonValue::QJsonValue(char const*)	20000	13 029 173
QString::~QString()	40000	7 360 000
QByteArray::~QByteArray()	40000	5 454 588
QJsonObject::~QJsonObject()	19999	4 975 113
QJsonDocument::toBinaryData() const	19999	4 856 112

Function	Loc	Called	Self Cost: Ir
0x0000000000000cc0	/l...	0	
cycle 1	/...	807	
_start	/...	1	
mbedtls_gcm_crypt_and_tag	/...	40000	
mbedtls_gcm_update	/...	40000	
gcm_mult	/...	310000	
mbedtls_aesni_gcm_mult	/...	310000	
_int_malloc	/...	1270579	



```
QByteArray GcmUtils::encryptAndTag(QByteArray data)
{
    unsigned char tag[16];
    quint64 length = data.length() + tag_len;

    unsigned char output[length + sizeof(quint64)];
    encryptDataGCM((const unsigned char *)data.constData(), length - tag_len, &context,
                   nullptr, 0, iv, iv_len, tag_len, tag, output + tag_len + sizeof(quint64));
    QByteArray encrypted;
    //QDataStream stream(&encrypted, QIODevice::ReadWrite);
    //qDebug() << output;

    //stream << length;
    //unsigned char tmp[length + sizeof(quint64)]();
    memcpy(output, QString::number(length).constData(), sizeof(quint64));
    memcpy(output + sizeof(quint64), tag, tag_len);

    //for (int i =0; i < tag_len; ++i){
        //stream << tag[i];
    //}

    //for (int i =0; i < data.length(); ++i){
        //stream << output[i];
    //}
    encrypted.fromRawData((const char *)output, length + sizeof(quint64));
```

Callee	Calls	Cost
encryptDataGCM(unsigned char const*, u...	20000	507 466 026
QString::number(unsigned long long, int)	19999	8 635 252
QByteArray::fromRawData(char const*, int)	19999	7 458 850
QString::~QString()	20000	3 680 000
QByteArray::~QByteArray()	20000	3 680 000
QByteArray::constData() const	20000	960 000
QString::constData() const	20000	960 000
QByteArray::QByteArray()	20000	420 000

Function	Loc:	Called	Self Cost: Ir
0x0000000000000cc0	/l...	0	
cycle 1	/...	807	
_start	/...	1	
mbedtls_gcm_crypt_and_tag	/...	40000	
mbedtls_gcm_update	/...	40000	
gcm_mult	/...	310000	
mbedtls_aesni_gcm_mult	/...	310000	
_int_malloc	/...	1270579	