ҚАЗАҚСТАН РЕСПУБЛИКАСЫ
БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РЕСПУБЛИКИ КАЗАХСТАН

ҚазҰТЗУ ХАБАРШЫСЫ

ВЕСТНИК КазНИТУ

VESTNIK KazNRTU

№2 (126)

АЛМАТЫ 2018 МАРТ

Учредитель:

Казахский национальный исследовательский технический университет
имени К.И. Сатпаева

Регистрация:

Министерство культуры, информации и общественного согласия
Республики Казахстан № 951 – Ж “25” 11. 1999 г.

Основан в августе 1994 г. Выходит 6 раз в год

Адрес редакции:

г. Алматы, ул. Сатпаева, 22,
каб. 616, тел. 292-63-46
Nina. Fedorovna. 52 @ mail.ru

Пыркова А.Ю., Темирбекова Ж.Е.

Симметриялық шифрлеуді mbed платформасында жүзеге асыру

Түйіндеме. Бұл мақала криптографиялық бағдарламалау және BLE Nano жиынтығы микроконтроллерінде жұмыс істейтін кодтық ортаны құрайтын Arm Mbed TLS платформасы туралы жазылған. Бағдарламасының тиімділігін арттыру үшін С орнына негізгі тіл ретінде С ++ пайдаланылды.

Түйін сөздер: BLE Nano kit микроконтроллері, Mbed платформасы, Arm Mbed TLS, криптография.

UDC 004.056.55

A. Pyrkova, Zh.E. Temirbekova
(Al-farabi Kazakh national university, Almaty, Republic of Kazakhstan,
temyrbekovazhanerke2@gmail.com)

POSSIBILITIES OF USING A BLE NANO KIT MICROCONTROLLER TO DEVELOP CRYPTOGRAPHIC LIBRARIES

Abstract. Nowadays, the number of IoT (Internet of Things) devices are growing rapidly around the world with wide range of purposes. Many types of these devices collect data and depending on their purpose, some of the data can be highly sensitive for the user. This calls for security on the device which has to secure the collected data and send it to a server through the Internet. Using TLS (Transport Layer Security) is a great way to provide such a security and one part of it is the data encryption which is the main focus of this thesis. The problem is that IoT devices are often made out of microprocessors having a low computational power and performance.

This article compiles some of the programming and debugging techniques writing the mbed platform, the code environment running on our BLE Nano kit microcontroller. C++ was used as the core language instead of C, in order to increase programming effectiveness. This turned out to be a good choice, but came at the cost of increased program size. Thus, a big part of optimizing was spent on reducing program and data size. Then demonstrate.The main goal of this thesis is to determine AES (Advanced Encryption Standard) symmetric encryption algorithms, security and space complexity using a specific encryption library from ARM mbed.

Keywords: IOT, BLE Nano kit microcontroller, C++, mbed platform, Arm Mbed TLS, AES cryptography.

BLE Nano kit is the smallest Bluetooth 4.1 Low Energy (BLE) development board in the market [1] . The core is Nordic nRF51822 (an ARM Cortex-M0 SoC plus BLE capability) running at 16MHz with ultra low power consumption.

Developing a Bluetooth Smart enabled 'appcessory' (accessory device + companion application) is easier than ever [2]. You can quickly produce prototypes and demos target for IoT and other interesting projects. BLE Nano could operate under 1.8V to 3.3V, therefore it works with a lot of electronic components.

Features:

- Smallest BLE development board, only 18.5mm x 21.0mm
- Nordic nRF51822 ARM Cortex-M0 SoC supports both BLE Central and BLE Peripheral roles
- 2.4 G Hz transceiver
- Ultra low power consumption
- Support voltage from 1.8V to 3.3V
- Software development using mbed.org, GCC, Keil or Arduino
- Lots of libraries and examples available
- Easy firmware deployment with the MK20 USB board
- Work with our free Android App and iOS App

BLE Nano kit microcontroller supported hardware platforms on mbed. The mbed Microcontrollers are a series of ARM microcontroller development boards designed for rapid prototyping [3-4].

The mbed platform is a free C/C++ compiler and IDE provided by ARM and suitable for programming most ARM microcontrollers. The IDE is Web based, something which may back some of us down. I was myself skeptical about the practicality of such IDE compilers, however I must say the mbed compiler has proven to be easy to use, fast and (reasonably) friendly, even for a C/C++ compiler [5-6].

The many available platform libraries are easy to use and well designed, specially for starters, resembling (vaguely) the simplicity you may have enjoyed when using Arduino libraries. Also, mbed

encourages and eases sharing with others your own libraries, or any piece/snippet of code. You have to register a username in the platform from the very beginning, that will allow you to store your projects online, comment, share code and participate in the community forums comfortably [7-8]. Compiling and programming your projects is really easy. You can literally have the blink test application running in a few minutes.

mbed has a dedicated team which develops and maintains a very nice Bluetooth Low Energy API. You can rest assured that using this API for learning to develop BLE appcessories will be much easier (and faster) than trying to do so by using directly the BLE libraries (Soft Device) provided by Nordic [9-10]. The BLE API includes also several application examples.

On the not-so-positive side of mbed platform, you have zero control over the compiler/linker. This has proven to be a showstopper for our own BLE development [11-12]. At the time of this writing, within mbed you are only allowed to develop BLE applications for the nRF51822 along with the S110 Soft Device, which only supports peripheral mode. Hence, you can quickly start to develop applications with mbed which only require the Peripheral Role. But if you want to develop an application requiring the Central Role, you will have to switch to a full-fledged GCC (or Keil or IAR) C/C++ compiler [13].

Entry-level Online Compiler–Nothing to Install: Browser-based IDE, Immediately compiling examples or writing your own, Best in class RealView Compiler in the back end (Figure 1).

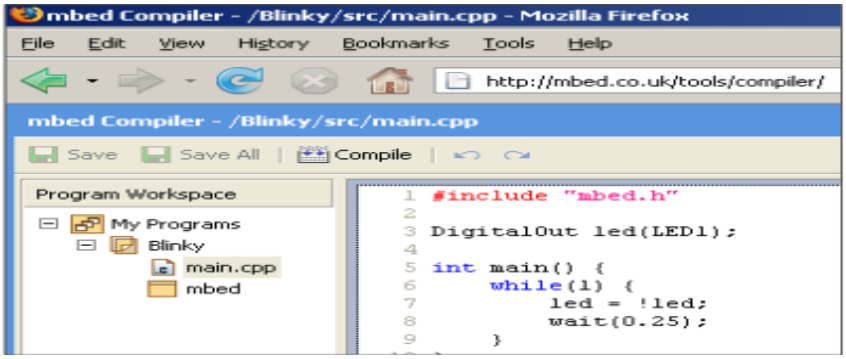

Figure 1. mbed Compiler C++ on a Microcontroller

The first firmware's for our BLE Nano kit microcontrollers were written in the "traditional" language C++ [14-15]. C++ is a pretty convoluted "multi paradigm" programming language: supporting standard structured programming, object-oriented programming as well as metaprogramming. It is one of the few languages giving the programmer absolute control over memory allocation and placement and functions-call overhead, which is crucial on a restricted environment.

Using BLE Nano Kit microcontroller tested program in the ARM mbed platform (read characters).

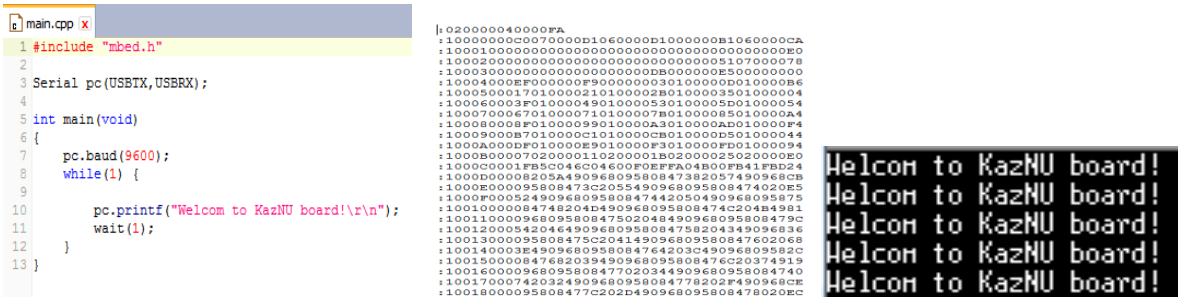

Figure 2. Read characters in C++

In order to provide security of data transmitted back and forth from an IoT device toa server, the communication has to be encrypted. Cryptography is the practice and study of techniques for secure communication and has been around for a long time. One of the cryptographic protocol is the TLS (Transport layer security) protocol where the data are encrypted with ciphers to protect them from third parties through the Internet.

The use of the ARM mbed TLS library with the BLE Nano microcontroller is an appropriate match, because they are made by the same company, so compatibility errors are less likely and cause security problems.

Arm mbed TLS makes it easy for developers to include cryptographic and SSL / TLS functions in their (embedded) products, which facilitates this functionality with minimal code printing [16]. It offers an SSL / TLS library with an intuitive API and readable source code and includes a complex set of tests [17]. You can create it out of the box in most systems or manually select and configure functions.

The mbed TLS library provides a set of cryptographic components that you can use and compile separately, and include or exclude, using a single configuration header file. mbed TLS also provides a central SSL / TLS module, which is based on cryptographic components, abstraction layers and support components to ensure full protocol implementation for SSL and TLS.

Cryptographic library

The cryptographic part of mbed TLS has abstraction layers for Public Key cryptography, Hashing (Message Digests) and Symmetric Ciphers [18]. It also contains standards-based random number generators and an entropy pool.

All cryptographic algorithms are implemented as loosely-coupled modules. You can just take the appropriate header files and source code files and drop them in your project as needed.

mbed TLS provides the most commonly used algorithms, such as AES. mbed TLS supports an alternative implementation for most of its cryptographic modules. The usual use case is for hardware accelerated cryptographic engines. There are several methods for alternative implementations: special function replacement and complete module replacement [19].

The configuration file contains the cryptography modules, which you can replace with alternative implementation [20]. These are named MBEDTLS_<MODULE NAME>_ALT. In order to support alternative implementation for a module, uncomment the corresponding *_ALT definition. Function replacement is according to the function name, with the suffix of _ALT. To support hardware entropy source, enable MBEDTLS_ENTROPY_HARDWARE_ALT in the configuration file.

AES is based on a design principle known as a substitution-permutation network, a combination of both substitution and permutation, and is fast in both software and hardware. AES is a variant of Rijndael which has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits. By contrast, the Rijndael specification *per se* is specified with block and key sizes that may be any multiple of 32 bits, with a minimum of 128 and a maximum of 256 bits.

In each round, the AES uses the four following operations [21]:

•SubBytes:Each byte of the array is transformed using a nonlinear substitution box called the AES S-Box.The S-Box in the AES has been carefully constructed and the cipher uses only one S-Box throughout the encryption.

•ShiftRows: Is a transposition step which ensures that the last threerows of the array are shifted by adifferent number of byte positions.

•MixColumns: Mixes each column in the array to create even more diffusion.

•Addkey:Using bitwise XOR, each byte of the array is mixed witha byte of a sub-key material, alsocalled round-key [22]. The sub-key ismade by "key expansion" and is derived from the main cipher key using a Rijndael key-schedule.

The overall structure of the cipher begins with a key expansion before going in to the pre-whitening which only includes AddKey. Then the rounds performing Sub-Bytes, ShiftRows, MixColumns and Add-Key will loop for $n-1$ rounds where n is the numbers of rounds. The n-th round or final round does not include MixColumns and produce the output ciphertext.

AES context structure. buf is able to hold 32 extra bytes, which can be used [23]:

- for alignment purposes if VIA padlock is used.
- to simplify key expansion in the 256-bit case by generating an extra round key.

AES-CBC buffer encryption/decryption. Length should be a multiple of the block size (16 bytes).

Upon exit, the content of the IV is updated so that you can call the function same function again on the following block(s) of data and get the same result as if it was encrypted in one call. This allows a "streaming" usage. If on the other hand you need to retain the contents of the IV, you should either save it manually or use the cipher module instead. param ctx-AES context, param mode-MBEDTLS_AES_ENCRYPT or MBEDTLS_AES_DECRYPT, param length-length of the input data, param iv -initialization vector (updated after use), param input-buffer holding the input data, param output-buffer holding the output data, return 0 if successful, or MBEDTLS_ERR_AES_INVALID_INPUT_LENGTH.

small_aes.c

```
#include "small_aes.h"

static const unsigned int rcon[] = {
    0x01000000, 0x02000000, 0x04000000, 0x08000000,
    0x10000000, 0x20000000, 0x40000000, 0x80000000,
    0x1B000000, 0x36000000,
};


static const unsigned int Te[256] = {
    0xc66363a5U, 0xf87c7c84U, 0xee777799U, 0xf67b7b8dU,
    0xfff2f20dU, 0xd66b6bbdU, 0xde6f6fb1U, 0x91c5c554U,
    0x60303050U, 0x02010103U, 0xce6767a9U, 0x562b2b7dU,
    0xe7fefe19U, 0xb5d7d762U, 0x4dababe6U, 0xec76769aU,
    0x8fcaca45U, 0x1f82829dU, 0x89c9c940U, 0xfa7d7d87U,
    0xeffafa15U, 0xb25959ebU, 0x8e4747c9U, 0xfbf0f00bU,
    0x41adadecU, 0xb3d4d467U, 0x5fa2a2fdU, 0x45afafeaU,
    0x239c9cbfU, 0x53a4a4f7U, 0xe4727296U, 0x9bc0c05bU,
    0x75b7b7c2U, 0xe1fdfd1cU, 0x3d9393aeU, 0x4c26266aU,
    0x6c36365aU, 0x7e3f3f41U, 0xf5f7f702U, 0x83cccc4fU,
    0x6834345cU, 0x51a5a5f4U, 0xd1e5e534U, 0xf9f1f108U,
    0xe2717193U, 0xabd8d873U, 0x62313153U, 0x2a15153fU,
    0x0804040cU, 0x95c7c752U, 0x46232365U, 0x9dc3c35eU,
    0x30181828U, 0x379696a1U, 0x0a05050fU, 0x2f9a9ab5U,
    0x0e070709U, 0x24121236U, 0x1b80809bU, 0xdfe2e23dU,
    0xcdebeb26U, 0x4e272769U, 0x7fb2b2cdU, 0xea75759fU,
    0x1209091bU, 0x1d83839eU, 0x582c2c74U, 0x341a1a2eU,
```

small_aes.h

```
#ifndef _SMALL_AES_H_
#define _SMALL_AES_H_

#include "string.h"

#ifdef __cplusplus
    extern "C" {
#endif

enum {
    SMALL_AES_ENCRYPTION = 0,
    SMALL_AES_DECRYPTION = 1,
    SMALL_AES_BLOCK_SIZE = 16
};

typedef struct AES {
    unsigned int key[60];
    unsigned int  rounds;

    unsigned int reg[SMALL_AES_BLOCK_SIZE / sizeof(unsigned int)];
} AES;

```

terminal output

```
plaintext message: 536f6d652074686696e6773206172652062657474657220 6c65667420756e7265616400
ciphertext: c57f7afb94f14c7977d785d08682a2596bd62ee9dcf216b8cccd997afee9b402f5de1739e8e6467aa363749e
decrypted: 536f6d652074686696e6773206172652062657474657220 6c65667420756e7265616400

DONE
```

Figure 3. Performance of AES ciphers in mbed platform.

AES also has one advantage over other encryption algorithms. As the name suggests, AES is an industry standard, it is not patented and is not free for everyone. One of the attributes of the AES algorithm is that it must reliably work on all platforms as hardware and software, so AES is well documented and distributed throughout the world. This means that if the AES breaks, regardless of who it is, the headers will not concern the company that used the hacked AES, they will be about fleurs into something that many companies and people use.

Security is fundamental for the successful rollout of the Internet of Things. Edgenodes are currently the weakest link in ensuring IoT security and the protection ofcryptographic key. The best way to achieve lockdown is by protected hardware. It is the only way to keep those keys and other secrets away from prying eyes. An IoT device can only be as secured as its weakest link.

REFERENCES

[1]Red Bear Company, http://redbearlab.com/blenano/, Accessed: 18th January, 2018.

[2]Jose Angel, BLE Nano hardware development kit for Bluetooth Low Energy, 2015

[3]Aguilar, S.; Vidal, R.; Gomez, C. Opportunistic Sensor Data Collection with Bluetooth Low Energy. *Sensors* - 2017, - p. 159.

[4]Gomez, C.; Oller, J.; Paradells, J. Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology. *Sensors* - 2012, - pp. 11734–11753.

[5]Nieminen, J.; Savolainen, T.; Isomaki, M.; Patil, B.; Shelby, Z.; Gomez, C. *IPv6 over BLUETOOTH(R) Low Energy*; RFC 7668; IETF: Fremont, CA, USA, - 2015.

[6]Gogic, A.; Mujcic, A.; Ibric, S.; Suljanovic, N. Performance Analysis of Bluetooth Low Energy Mesh Routing Algorithm in Case of Disaster Prediction. *Int. J. Comput. Electr. Autom. Control Inf. Eng.* - 2016, - *3*, - pp. 175–181.

[7]Kim, H.-S.; Lee, J.; Jang, J.W. BLEmesh: A Wireless Mesh Network Protocol for Bluetooth Low Energy Devices. In Proceedings of the 2015 3rd International Conference on Future Internet of Things and Cloud (FiCloud), Rome, Italy, 24–26 August 2015.

[8]Maharjan, B.K.; Witkowski, U.; Zandian, R. Tree Network Based on Bluetooth 4.0 for Wireless Sensor Network Applications. In Proceedings of the 6th European Embedded Design in Education and Research Conference (EDERC), Milan, Italy, 11–12 September 2014.

[9]K. Townsend, R. Davidson, Getting Started with Bluetooth Low Energy. -2014. pp. 105-108

[10]Embedded C++ Technical Committee. "The embedded C++ programming guide lines". http://www.caravan.net/ec2plus/guide.html

[11]Tsai, C. Lai, C. Vasilakos, V. (2014). Future internet of things: Open issue and challanges. ACM/SpringerWirelessNetworks,. doi:10.107/s11276-014-0731-0.

[12] Li, S.; Xu, L.D. and Zhao, S., "The internet of things: a survey", Information Systems Frontiers, vol. 17, No.2, pp 243-259, April 2015, DOI:10.1007/s10796-014-9492-7

[13] Atmel Corporation,"Integrating the Internet of Things: Necessary building blocks for broad market adoption", San Jose, USA: Atmel, 0776 Corporate IOT WhitePaper US 102014.

[14] Knudsen, L.R.; Robshaw, M.J. The Block Cipher Companion, New York: Springer-Verlag, - 2011. – pp. 225-228

[15] Nermin Hajdarbegovic, "Are We Creating An Insecure Internet of Things (IoT) Security Challenges and Concerns", 2015. [Online] Available:https://www.toptal.com/it/are-we-creating-an-insecure-internet-of-thing.

[16] Jessica Chani Cahuana, Chalmers, A Search for a Convenient Data Encryption Algorithm For an Internet of Things Device. – 2016. - p. 102

[17] Bartlett, Alan and Richard Silverman "SSH: The Secure Shell The Definitive Guide." - 31 July 2001. – p. 98

[18] Dierks, T. & C. Allen. "The TLS Protocol Version 1.0." - 1999. -p. 100

[19] Hiller, Janine S. and Ronnie Cohen. Internet Law & Policy. Upper Saddle River: Prentice Hall, - 2000.

[20] Martin, Franck. "SSL Certificates HOWTO." - 20 October 2002.

[21] Alex Biryukov and Dmitry Khovratovich, Related-key Cryptanalysis of the Full AES-192 and AES-256, "Archived copy". Archived *from the original on 2009-09-28*. Retrieved 2018-02-16.

[22] "Rijndael". Archived from the original on January 25, 2018. Retrieved March 1, 2018.

[23] Daemen, Joan; Rijmen, Vincent. "AES Proposal: Rijndael". National Institute of Standards and Technology. – 2003. - p. 10.

Пыркова А.Ю., Темирбекова Ж.Е.

Возможности использование микроконтроллера BLE Nano Kit для разработки криптографических библиотек

Аннотация. В настоящее время количество устройств IoT (Internet of Things) быстро растет во всем мире с широким спектром задач. Многие типы этих устройств собирают данные и в зависимости от их назначения некоторые данные могут быть очень чувствительны для пользователя. Это требует безопасности на устройстве, которое должно защищать собранные данные и отправлять их на сервер через Интернет. Использование TLS (Transport Layer Security) - отличный способ обеспечить такую безопасность, и одна из них - это шифрование данных, которое является основным направлением этого тезиса. Проблема в том, что устройства IoT часто изготавливаются из микропроцессоров с низкой вычислительной мощностью и производительностью.

В этой статье описаны некоторые методы программирования и отладки, написанные на платформе mbed, среда кода, работающая на нашем микроконтроллере BLE Nano. C ++ использовался как основной язык вместо C, чтобы повысить эффективность программирования. Таким образом, большая часть оптимизации была потрачена на сокращение размера программы и данных. Затем продемонстрировано. Основная цель этого статья - определить симметричные алгоритмы шифрования AES (Advanced Encryption Standard), безопасность и сложность пространства, используя специальную библиотеку шифрования из ARM mbed.

Ключевые слова: IOT, микроконтроллер BLE Nano kit, C ++, платформа mbed, Arm Mbed TLS, криптография AES.

Пыркова А.Ю., Темирбекова Ж.Е.

Криптографиялық кітапханаларды дамыту үшін BLE Nano Kit микроконтроллерді қолдану мүмкіндіктері

Түйіндеме. Қазіргі уақытта IoT (Интернет заттар) құрылғыларының саны бүкіл әлем бойынша тез арада дамып келеді. Бұл құрылғылардың көптеген түрлері деректерді жинайды және олардың мақсаттарына байланысты кейбір деректер пайдаланушыға өте сезімтал болуы мүмкін. Бұл жинақталған деректерді қорғап, Интернет арқылы серверге жіберуге тиіс құрылғыдағы қауіпсіздікті талап етеді. TLS (Transport Layer Security) - бұл қауіпсіздікті қамтамасыз етудің керемет тәсілі және оның бір бөлігі деректерді шифрлау болып табылады, ол осы мақаланың негізгі бағыты болып табылады. Мәселе мынада, IoT құрылғылары жиі есептеу қуаты мен өнімділігі төмен микропроцессорлардан жасалады.

Бұл мақала Mbed платформасында жазатын программалау және BLE Nano kit жинағының микроконтроллерінде жұмыс істейтін шифрлау кодын есептеу ортасы болып табылады . Бағдарламаның тиімділігін арттыру үшін C орнына негізгі тіл ретінде C++ пайдаланылды. Бұл жақсы таңдау болып шықты, бірақ бағдарламаның көлемін ұлғайтты. Осылайша оңтайландырудың үлкен бөлігі бағдарлама мен деректер көлемін азайтуға жұмсалды. Содан кейін есептеу нәтижесі бейнеленген. Осы мақаланың негізгі мақсаты ARM платформасында арнайы шифрлау кітапханасын пайдаланып, AES (Advanced Encryption Standard) симметриялы шифрлау алгоритмін анықтау болып табылады.

Түйін сөздер: IoT, BLE Nano kit микроконтроллері, C ++, Mbed платформасы, Arm Mbed TLS, AES криптографиясы.

Химия-металлургия ғылымдары

Химико-металлургические науки

Chemical and metallurgical sciences

Редакторы:
Н.Ф. Федосенко
Верстка на компьютере:
Л.Т. Касжанова

Подписано в печать 29.03.2018 г.
Формат 60х84 $^{1}/_{8}$. Усл. п.л 37,8.
Тираж 500 экз. Заказ № 207.

Адрес редакции:
ул. Сатпаева, 22, КазНИТУ каб. 616, тел. 292-63-46 ,Nina.Fedorovna. 52 @ mail.ru
Департамент маркетинга и коммуникаций КазНИТУ
Казахского национального исследовательского технического университета имени К.И. Сатпаева