

LWIG Working Group
Internet-Draft
Intended status: Informational
Expires: August 17, 2014

M. Sethi
V. Lehtovirta
P. Salmela
Ericsson
February 13, 2014

Using Generic Bootstrapping Architecture with Constrained Devices
draft-sethi-gba-constrained-01

Abstract

This document discusses the use of the 3GPP Generic Bootstrapping Architecture (GBA) for authenticating and securing constrained devices. While GBA re-uses the 3GPP credentials, it does not require mobile network access, such as LTE, but requires only IP connectivity. Though building devices that employ GBA is obviously well known, this document specifically focuses on techniques necessary to minimize memory and energy consumption which is essential for constrained device networks.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 17, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Terminology	5
1.2. Rational for using GBA	5
2. Reason for low power implementation of GBA	6
3. Implementation Considerations	7
3.1. Full HTTP stack is not needed	7
3.2. Resource-Constrained AES implementations are widely available	7
3.3. Purging unnecessary functionality from memory after bootstrapping	7
3.4. Complete State Machine or Complex Error Handling Are Not Needed	8
4. Implementation Status	9
5. Future standardization work	12
6. Security Considerations	13
7. Acknowledgments	14
8. IANA Considerations	15
9. Informative References	16
Authors' Addresses	18

1. Introduction

Generic Bootstrapping Architecture (GBA) is part of the 3GPP standard [3gppts33220] based on 3GPP Authentication and Key Agreement (3GPP AKA). GBA is an application independent mechanism to provide a client application (running on the User equipment (UE)) and an application server with a shared session secret. This shared session secret can subsequently be used to authenticate and protect the communication between the client application and the application server. GBA authentication is based on the permanent secret shared between the UE's Universal Integrated Circuit Card (UICC), for example SIM card, and the corresponding profile information stored within the cellular network operator's Home Subscriber System (HSS) database. The permanent shared secret is used to generate a time-limited master key on the UE and the network operator. The UE and network operator derive a session key from the master key, and the network operator distributes this to the appropriate application server. This session key can then allow authentication and protection of the communication channel between the UE and application server. Figure 1 provides a high-level overview of GBA.

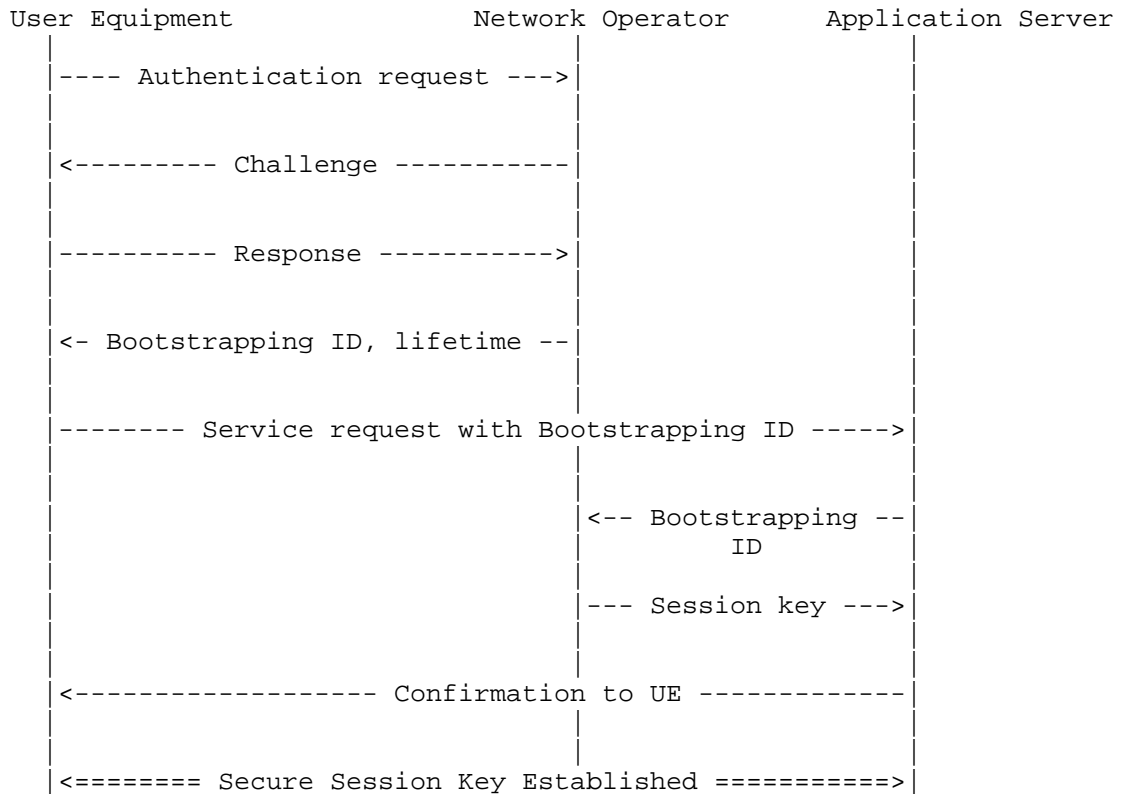


Figure 1: High Level Overview of GBA

As shown in Figure 1, the UE initially identifies itself to the network operator and requests for authentication. The network operator responds by sending a challenge that can be correctly responded to, only with the permanent secret stored on the UICC. The UE responds to the challenge using the permanent secret on the UICC. The network operator then verifies the response and if it is found to be valid, it derives a master key and returns a bootstrapping ID to the UE. The UE then derives a session key from the master key and initiates a secure connection with the application server with which it wants to communicate and provides the bootstrapping ID obtained in the previous step. The application server uses the bootstrapping ID to obtain a session key from the network operator. The session key is derived from the master key by the network operator. The network operator authenticates the application server before generating and revealing the session key. Once the application server is authenticated, it receives the session key. The application authenticates the original message from UE with this session key. If

authenticated correctly, the application server sends a confirmation message to the UE indicating that it can now protect the communication with the session key. While in this scenario, the UE is responsible for initiating secure communication with the application server, the GBA standard also allows the application server to inform the UE to perform GBA authentication for access to the requested resource. In 3GPP terminology, the application server is referred to as Network Application Function (NAF) and the cellular operator interface is referred to as Bootstrapping Server Function (BSF). The bootstrapping ID is known as the Bootstrapping Transaction Identifier (B-TID)

The current GBA standard requires HTTP and TCP along with crypto algorithms like AES and MD5 to be supported on the UE. While the implementation details may be obvious for a UE, using GBA for authentication and channel security on resource-constrained devices may be non-trivial. This draft provides implementation guidelines for using GBA authentication on resource-constrained devices.

1.1. Terminology

This draft uses terminologies for constrained devices defined in [I-D.ietf-lwig-terminology].

1.2. Rational for using GBA

In principle, GBA can be used to protect any application traffic, e.g. with TLS, as GBA provides shared keys for applications to use. Using shared keys avoids the need for complex calculations of asymmetric crypto and the need for verifying certificate chains and checking revocation lists. Instead, GBA relies on the security infrastructure of the network operator. Even though GBA re-uses the credentials stored on the UICC, use of GBA does not require mobile network access, such as LTE, but requires only IP connectivity to the network operator infrastructure. As the credentials of the UICC are used by GBA, there is no need to remember or store passwords to authenticate to applications, and in principle a button-free authentication can be implemented.

2. Reason for low power implementation of GBA

As explained in [I-D.draft-ietf-lwig-cellular-00] there are many situations where focus on a low energy consuming implementation is unnecessary. This would be the case, when for example, the devices are connected to the main, or receive power over wired communications media, such as in Power-over-Ethernet (PoE) devices. Such devices require a limited amount of optimization for energy efficiency. Similarly, devices that are directly connected to the mains do not require extreme optimizations and/or hacks for saving power. Also devices that can gather power from energy harvesting do not necessarily need optimization techniques.

However, wirelessly connected battery powered sensors and actuators are the future and existence of wired infrastructure for communication or power might be impossible or impractical. Such devices require highly efficient implementations of security and network protocols as changing batteries frequently is not possible because of the sheer number of devices and the fact that some of these devices may be physically inaccessible once installed.

Unfortunately, much of our current network and security technology is built with different objectives in mind. Networked devices are assumed to be "always on", and require frequent re-charging. However, most sensor and actuator applications require long battery lifetimes, in the order of years or even a decade, if not longer. While some devices may already reach multi-year lifetimes with continuous improvement in low-power electronics and radio technology, it is rather reasonable to note that battery lifetimes are generally too short currently. Therefore, to use GBA as a mechanism for secure bootstrapping for constrained device applications, a more focused effort is required to ensure an energy-efficient implementation. This memo is a first step for defining considerations and guidelines to achieve an energy-efficient GBA implementation.

3. Implementation Considerations

In this section we describe implementation guidelines that apply to GBA on constrained devices in particular but can be used in general for implementing other similar related network protocols.

3.1. Full HTTP stack is not needed

GBA relies on HTTP digest authentication and requires the client to implement an HTTP stack. Although HTTP client libraries for constrained devices [small-http] are available, they are often unnecessary. This is because the number of HTTP messages required for a GBA-run is small and having an entire library consumes more memory than the templates for these messages. Thus, the required messages can be handcrafted and sent over TCP. Thanks to the plain text nature of HTTP/1.1 this is rather easy to do. An example of a minimalistic HTTP packet is shown here:

```
char httpFirstRequestFormat[82] = "GET /naf/resource HTTP/1.1\r\n"  
                                "Host: pl23.example.net:8080\r\n"  
                                "Connection: Keep-Alive\r\n"  
                                "\r\n";
```

Similarly, the response can easily be parsed from the packets received from the transport layer without necessarily performing all error checks. If an unknown packet or unknown data is received, the application can simply re-start from the first exchange.

3.2. Resource-Constrained AES implementations are widely available

AES is rather easy to implement. There are many open source implementations available specifically for resource-constrained environments [gladman], [avr-crypto-lib]. There are several resources [jpkaps06],[shammid] that show that AES can be implemented with low memory and energy consumption. Also many constrained devices and platforms such as the MicaZ motes are equipped with an AES hardware engine as part of the IEEE 802.15.4 (Zigbee)-compliant RF module.

3.3. Purging unnecessary functionality from memory after bootstrapping

Once a GBA bootstrapping round has finished, only the session key (KsNAF) and B-TID need to be retained in the memory. Optimally, the master key (Ks) and the key derivation function can be retained in the memory if the device will be connecting to multiple NAFs. If a GBA run is only used for authentication, then all GBA related code can be purged from the memory. This means that any libraries or code used for application, transport or cryptography during a GBA run can

be purged and loaded back into the memory when the lifetime of the key expires. This is especially useful with long lived session keys as the device can switch to other application (CoAP) and routing/transport (RIPL/UDP) protocol after secure bootstrapping.

3.4. Complete State Machine or Complex Error Handling Are Not Needed

A typical GBA implementation requires a state-machine to track messages that have been sent and received from the BSF and NAF. However, implementing such a state-machine with appropriate error handling can be rather complex for resource-constrained device. It is therefore advisable for such devices to implement a simple hard fail-over starting from the first message in case of an error, unknown packet or timeout. However, there also needs to be a limit to the number of fail-overs as the device might drain its battery from too many failed GBA authentication tries.

4. Implementation Status

In this section we describe our GBA implementation that was developed for prototyping GBA on constrained devices using the principles described in the previous section. We used an Arduino Mega prototyping board [mega] and an Ethernet Shield for communication. The Mega board has an 8-bit ATmega2560 microprocessor.

For AES, we used Brian Gladman's byte oriented AES implementation [gladman]. Similarly, we used easily available C implementations of SHA256 [sha256] and MD5 [md5]. The default Ethernet and TCP libraries available with the Arduino platform were used for communication.

We tested this GBA implementation against standard 3GPP BSF and NAF interfaces running on our public servers. The sequence of messages for the resource-constrained (sensor/actuator) device in our implementation is as follows:

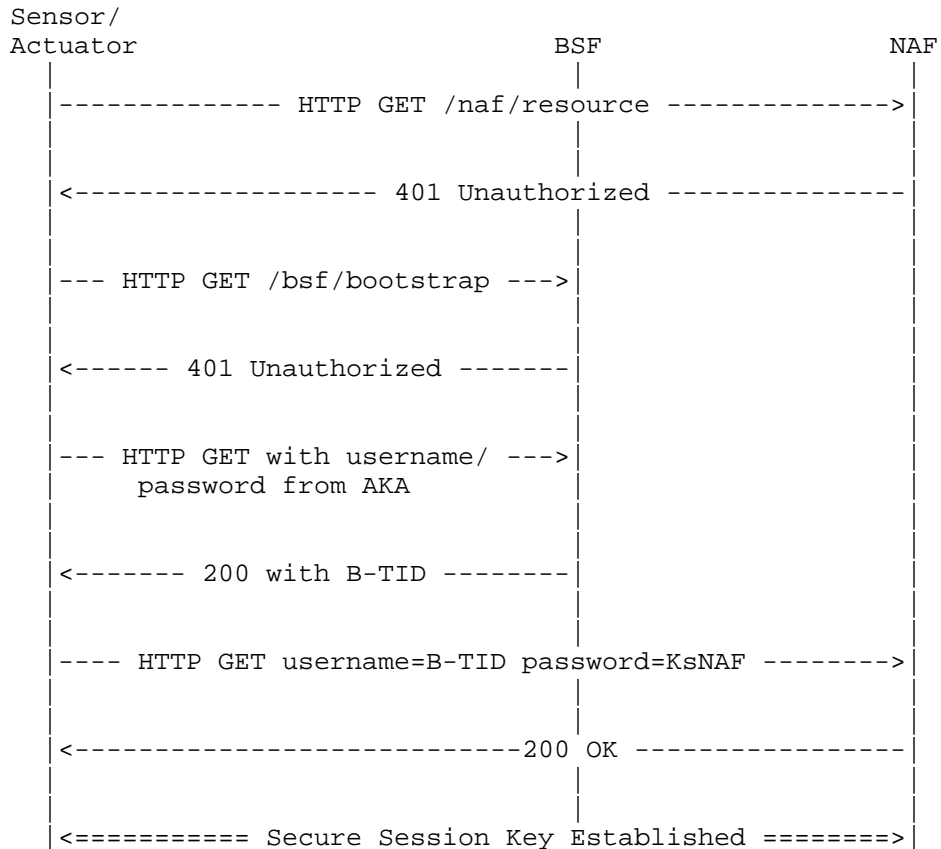


Figure 2: Prototype message sequence

The sensor/actuator begins by contacting the service with which it wishes to communicate (NAF in 3GPP terminology). This means that the sensor sends a HTTP GET for the resource which it wishes to access. The service/NAF in-turn replies with a HTTP 401 Unauthorized with WWW-Authenticate indicating that it requires digest authentication for access to that resource. The HTTP response header also contains the realm prefixed with "3GPP-Bootstrapping@" to indicate that a GBA run can be performed to obtain the appropriate keys for digest authentication. The sensor then contacts the BSF hosted by the operator by sending a HTTP GET. This again results in a 401 unauthorized with the HTTP header containing "algorithm=AKAv1-MD5" indicating that AKA authentication needs to be run. The sensor then creates and sends another HTTP GET containing the username and password from AKA to the BSF. The BSF now replies with an 200 OK message whose body contains the B-TID and the lifetime for which the

B-TID and the derived master key (Ks) would be valid. Finally, the sensor contacts the service/NAF with a HTTP GET containing the B-TID as the username and KsNAF as the password. KsNAF is obtained from the master key Ks and the fully qualified domain name (FQDN) of the NAF. If the service/NAF replies with a 200 OK, the sensor is assured of successful authentication. The sensor and NAF can then use the session key for TLS/DTLS in PSK mode. This key can also be used for message level encryption and/or integrity protection by using, for example, the EAX [mihir04] mode.

Some approximate results from our prototype are listed in the table below:

RAM consumption	<5 kB
ROM consumption	44 kB
Time for 1 GBA run	1.5 s
Energy ($W = U * I * t$)	150mJ
Additional HTTP messages sent/received	8

Prototype Details

Table 1

U=Operating Voltage (5V), I=Current drawn (0.02A for Atmega2560),
t=Execution time (1.5 s)

The energy estimate and time for 1 GBA run would vary significantly depending on the network access (which in our case was Ethernet).

5. Future standardization work

While we have shown how the current GBA standard can be used as a method for secure bootstrapping of constrained devices, it would also be useful in the future for 3GPP and IETF to define GBA over protocols other than HTTP/TCP.

6. Security Considerations

GBA is used for authenticating a device based on the 3GPP subscription credentials stored in the device. Normally the credentials are stored on a UICC, but also embedded UICCs (eUICC) are possible. eUICCs are targeted for scenarios where the 3GPP subscription should be changeable remotely, without having to physically exchange the card in the device. Both UICCs and eUICCs provide hardware based protection for the subscription credentials used for authentication in 3GPP networks. However, in addition it is also essential to protect the access to the (e)UICC from applications to protect against Man-in-the-Middle (MitM) type of attacks [3gpptr33905].

If a UICC or eUICC is not available, GBA digest is an option [3gppts33220]. GBA digest uses SIP digest credentials, basically a username/password pair and secret deducted from TLS, to do GBA. The security requirement defined by 3GPP for the SIP digest credentials are that they need to be securely stored within the terminal.

7. Acknowledgments

The authors would like to thank Bengt Sahlin, Jari Arkko and Ari Keranen for the interesting discussions on this topic.

8. IANA Considerations

This document has no IANA actions.

9. Informative References

- [3gpptr33905]
3GPP, "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Recommendations for Trusted Open Platforms", September 2012,
<<http://www.3gpp.org/DynaReport/33905.htm>>.
- [3gppts33220]
3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (GBA) (Release 12)", March 2013,
<<http://www.3gpp.org/DynaReport/33220.htm>>.
- [I-D.draft-ietf-lwig-cellular-00]
Arkko, J., Eriksson, A., and A. Keranen, "Building Power-Efficient CoAP Devices for Cellular Networks", draft-ietf-lwig-cellular-00 (work in progress), December 2013.
- [I-D.ietf-lwig-terminology]
Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained Node Networks", draft-ietf-lwig-terminology-06 (work in progress), August 2013.
- [avr-crypto-lib]
"AVR Crypto-lib", December 2013,
<<http://avrcryptolib.das-labor.org/trac/wiki/AES>>.
- [gladman] "Byte Oriented AES (Low Resource Version)", December 2013,
<<http://gladman.plushost.co.uk/oldsite/AES/>>.
- [jpkaps06]
Kaps, JP. and B. Sunar, "Energy comparison of AES and SHA-1 for ubiquitous computing", Emerging Directions in Embedded and Ubiquitous Computing. Springer Berlin , 2006.
- [md5] "MD5", December 2013,
<<https://github.com/tzakis/ArduinoMD5/>>.
- [mega] "Arduino Mega", December 2013,
<<http://arduino.cc/en/Main/arduinoBoardMega2560>>.
- [mihir04] Bellare, M., Rogaway, P., and D. Wagner, "The EAX mode of operation", Fast Software Encryption. Springer Berlin Heidelberg , 2004.

- [sha256] "SHA256", December 2013,
<<https://github.com/Cathedrow/Cryptosuite>>.
- [shammid] Didla, S., Ault, A., and S. Bagchi, "Optimizing AES for
embedded devices and wireless sensor networks", 4th
International Conference on Testbeds and research
infrastructures for the development of networks &
communities , 2008.
- [small-http]
"HttpClient for Arduinos", December 2013,
<<https://github.com/amcewen/HttpClient>>.

Authors' Addresses

Mohit Sethi
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: mohit.m.sethi@ericsson.com

Vesa Lehtovirta
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: vesa.lehtovirta@ericsson.com

Patrik Salmela
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: patrik.salmela@ericsson.com

