Generic Hybrid Methods for Secure Connections based on the Integration of GBA and TLS/CA

David Khoury, Elie Kfoury
Computer Science Department
American University of Science and Technology
Beirut, Lebanon
dkhoury@aust.edu.lb, ekfoury@aust.edu.lb

Abstract—**The Generic Bootstrapping Architecture (GBA) is standardized by the Third Generation Partnership Project (3GPP) to provide authentication and data confidentiality based on the existing Mobile Network Operator's (MNO) infrastructure. Unfortunately, this method is not well deployed as the MNO is not always fully trusted by its clients. On the other hand, Transport Layer Security (TLS) is a well-spread protocol that provides strong authentication and data confidentiality based on Public Key Infrastructure (PKI) and certificate distribution. However, distributing certificates for each client remains impractical especially for Internet of Things (IoT) devices. In this paper, we propose hybrid methods that integrate both mechanisms to provide mutual authentication based on two trusted authorities. Each proposed method is deployed depending on the adopted devices and security requirements. Finally we estimate the impact of each of these methods and we conclude with the intended future work.**

Keywords—GBA, 3GPP, MNO, TLS, PKI, CA Hybrid, Authentication, Confidentiality, IoT, Trust.

I. Introduction

In the last few years, several considerable efforts were spent to provide a general security framework that can ensure the classical objectives of information security: Authentication, confidentiality, integrity, availability, and non-repudiation. Authentication is the first step in almost any network session initiation. Hence, being non-authenticated disrupt the whole session and violates the other aforementioned security concerns. The PKI is a widespread authentication method often used in conjunction with a trusted certificate authority (CA) to secure the transmitted data and verify the identity of the contacted server [1]. Authentication not only involves validating the server's identity, but the identity of the client as well. In PKI this issue was partially solved by assigning a certificate to the client to create an identity. While this architecture seems possible, distributing certificates for each client remains impractical especially for IoT devices as there will exist according to Ericsson [2] 28 billion devices by 2021, and it's not possible to provision such number of certificates.

On the other hand, 3GPP standardized the GBA [3] that provides a general mechanism based on 3GPP Authentication and Key Agreement (AKA) [4] to install a shared secret between the client and the server. Hence, it leverages the mobile security infrastructure by using permanent secret stored in the smart card (SIM) and the Home Subscriber Server (HSS) on the mobile operator's network. Since almost all clients are subscribers to the MNO, GBA can be used to provide client's authentication and solve the raised limitation in PKI. However, so far, this solution is not heavily adopted due to the security (authentication, confidentiality, and integrity) dependability on one private sector organization which is not always fully trusted by its clients. In general, a system or an infrastructure is considered as "Trusted" if it provides a trusted service, but at the same time, it is not possible to verify it. This definition is used as a hypothesis for our proposed scheme.
The objective of this article is to provide a generic security framework that relies on two trusted authorities in order to provide the appropriate security mechanism depending on the device type and the level of security required in the application.

The main contributions of this paper are: 1) Provide a Trusted authentication and double layer end-to-end encryption for non-constrained devices based on two authorities. 2) Revisit the GBA architecture to provide authentication only and use TLS/CA for encrypting the data, and 3) Use GBA with TLS/CA in capillary networks for resource constrained devices. The rest of this paper is organized as follows: Section II overviews the theoretical background of both existing security infrastructures, GBA and TLS/CA. Section III introduces the proposed methods and their advantages comparing to the existing architectures. Results are demonstrated in section IV. Section V provides some concluding remarks and the future work.

II. Background

As discussed previously, there exist two authentication mechanisms, the well-known TLS/CA and the 3GPP GBA. In this section, we overview these two mechanisms and we highlight their strong points and their weaknesses.

A. *Generic Bootstrapping Architecture (GBA)*

GBA is an authentication architecture standardized by the 3GPP that enables the authentication of a client and a server based on existing mobile network operator infrastructure. The method consists on validating the identity of the client based on a token (e.g SIM card of the subscriber) and generating a common shared key between the client and the server. It has been recognized that the mobile infrastructure could be leveraged to enable application functions in the network and on the user side to establish shared keys. Therefore, 3GPP can provide the "bootstrapping of application security" to authenticate the subscriber by defining a Generic Bootstrapping Architecture based on AKA protocol [3]. The main components of GBA are: 1) The Network Application

Function (NAF): element which forms the authentication function of the server side, and 2) The Bootstrapping Server Function: an intermediary element in Cellular networks which provides application independent functions for mutual authentication.

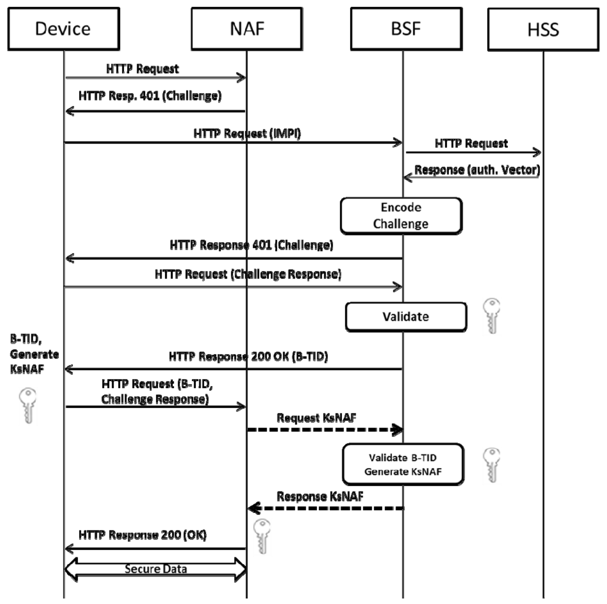

Fig. 1. GBA sequence diagram

Fig. 1. demonstrates the main objects involved in a GBA procedure and they interact with one another and in what order. The steps for a GBA authentication can be summarized as follows: 1) The client (device) access the Network Application Function (NAF) by sending HTTP request. The NAF replies with a bootstrapping initiation message. 2) The client contacts the Bootstrapping Server Function (BSF) with its identity as parameter. The BSF then retrieves the authentication vector from the HSS based on the received identity. 3) The BSF encodes a challenge, and sends it to the client for verfication. 4) The client sends the challenge response to the BSF through an HTTP request. 5) The BSF authenticates the subscriber using by validates the response as it is done in AKA protocol. 6) If validated, the BSF returns back to the client a Bootstrapping Transaction Identifier (B-TID). 7) The client derives a NAF-specific shared key material (KsNAF) based on B-TID. 8) The client sends an HTTP request to the NAF containing as parameters the B-TID and the challenge response. 9) The NAF request from the BSF the KsNAF for the corresponding B-TID. Note here that a secure channel between the NAF and the BSF is a pre-condition for GBA. 10) The NAF replies back to the client with a 200 OK message indicating that it is ready to start transmitting secure data. 11) The client and the server can now transmit secure data based on a key KsNAF generated by the client and the BSF (MNO): To exchange a message **M**, the symmetric key KsNAF is used for encryption and decryption:

$$C \leftrightarrow S \{ M \}_{KsNAF} \qquad \textbf{(1)}$$

Another alternative is to use the KsNAF for setting up a TLS-PSK (Pre-Shared keys) session.

Based on the above explanation we can clearly deduce two main advantages of GBA: its feasibility compared to PKI due to the elimination of the user enrollment and keys deployment phases, and its portability and simplicity as a result of using the well-known HTTP digest mechanism.

The main disadvantage is that the operator may not be trustworthy from confidentiality and integrity point of view as the time limited KsNAF is generated by the mobile infrastructure.

B. Transport Layer Security / Certificate Authority

Transport Layer Security (TLS) is a general purpose security service implemented as a set of protocols that run on top of TCP to provide secure communication security over a computer network [5]. It is the successor of Secure Socket Layer (SSL), yet still often reffered to as SSL.

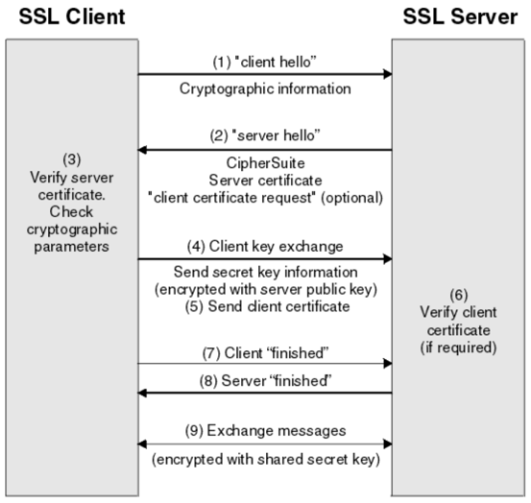
SSL Client
SSL Server
(1) "client hello"
Cryptographic information
(2) "server hello"
CipherSuite
Server certificate
"client certificate request" (optional)
(3) Verify server certificate. Check cryptographic parameters
(4) Client key exchange
Send secret key information (encrypted with server public key)
(5) Send client certificate
(6) Verify client certificate (if required)
(7) Client "finished"
(8) Server "finished"
(9) Exchange messages
(encrypted with shared secret key)

Fig. 2. TLS/SSL hanshake (IBM: An overview of the SSL or TLS handshake

Fig. 2. demonstrates the interaction between the SSL client and the SSL server. SSL handshake can be summarized in four phases: In phase 1, the client sends a client-hello message and the security capabilities to the server which returns a server-hello back to the client. In phase 2, the server sends its certificate, key exchange, and optionally a certificate request to the client. The client validates the server's identity using the CA signature. In phase 3, client sends key exchange and its certificate if requested. In phase 4, they change cipher suite and finish the hanshake protocol.

At the end of the TLS handshake protocol, the symmetric key K_{TLS} is generated and used for encryption and integrity of the messages exchanged between the client and the server.

While TLS/CA offers the advantages of authenticating the server as well as ensuring confidentiality between the communicating parties, it also has some drawbacks: 1) The client is often not authenticated since it is inconvenient to configure the client to use certificate especially by non-technical users; moreover, as mentioned previously, it is impractical to assign and provision million of certificates in the IoT case. 2) Risk of having a rogue CA or subverting a CA: DigiNotar was a well-established and reputable certificate authority included in all major browsers as a root CA. In June 2011, an intruder successfully penetrated DigiNotar server and issued his first rogue certificate. By the end of the summer, he would go on to issue 531 rogue certificates for domains ranging from aol.com and microsoft.com to mossad.gov.il and cia.gov [6]. In 2001, a notable case of VeriSign subversion occurred, when it issued two certificates to a person claiming to represent Microsoft [7].

III. Proposed Methods

In this section, we elucidate three proposed methods and their applications on three different scenarios. The adoption of each method depends on the type of application requiring security hardening as well as the type of the available devices. We also highlight the advantages of each method compared to the aforementioned existing infrastructures.

A. Method 1: Integrating GBA and TLS

This method combines the use of GBA and TLS. GBA provides a simple mean to authenticate the client and encrypt the data using a time limited session key generated by the BSF as explained earlier. When the bootstrapping method is established, the client initiates another secure connection based on TLS protocol towards the NAF. As a result, the communication between the client and the NAF has a double layer authentication and encryption as shown in Fig. 3, with two different session keys, each generated by a different authority: The first is provided by the MNO and the second by the TLS/CA mechanism.

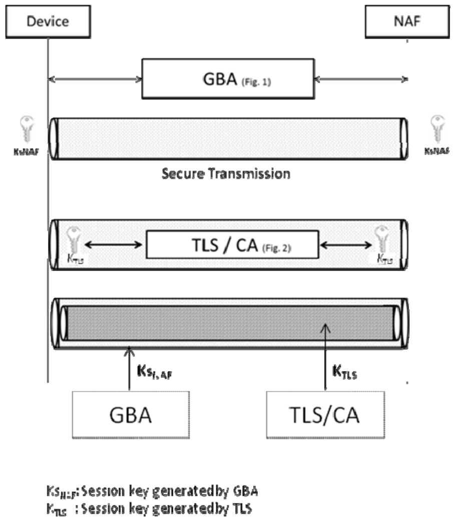

Fig. 3. GBA/TLS integration

Hence, a message **M** exchanged between the client and the server is masked using the following encryption function:

$$C \leftrightarrow S \{ \{ M \}_{KsNAF} \}_{K_{TLS}} \qquad \textbf{(2)}$$

This method provides several advantages over the existing infrastructures, namely: 1) Two trusted authorities mutually excluding each other, prohibiting data eavesdropping and alteration by both the MNO and the CA, 2) Hardening data confidentiality by providing a double layer protection using two keys, KsNAF and K_{TLS}. 3) Elimination of client certificate as the client is already authenticated during GBA bootstrapping. 4) Double server authentication since the NAF is authenticated in GBA and in TLS. 5) Eradicate rogue and fraudulent CA certificates: GBA solves this issue by design since a secure channel between the NAF and the BSF is a precondition for bootstrapping. If the client is contacting a fake NAF, the bootstrapping process will stop as soon as the NAF tries to request KsNAF from the BSF. 6) Increased level of trust**:** While the entire system security depends on trust, this method offers an additional authority to trust, thus decreasing the probability of having an entity act against your interest.

This method can be used in security critical applications that require mutual authentication, as well as superior encryption. For instance, in the scenarios of mobile payment, digital identity, online voting, etc… Moreover, GBA combined with TLS provides authentication beyond passwords based on a hardware token (SIM or e-SIM).

B. Method 2: Securing IoT Devices

In this method, we aim at providing a secure framework for IoT devices. According to RFC 7228 [8], IoT or constrained devices are classified into three classes as illustrated in Table I: C0 devices use gateway for basic communication, C1 devices use IoT specific protocol stack and connect without the need of a gateway, and C2 devices support regular IP protocol stack and connect directly to a remote application.

TABLE I. IoT Constrained Devices Classes

Name	**data size (e.g., RAM)**	**code size (e.g., Flash**
Class 0, C0	<< 10 KiB	<< 100 KiB
Class 1, C1	~ 10 KiB	~ 100 KiB
Class 2, C2	~ 50 KiB	~ 250 KiB

Since the main disadavantage the previous method is the computational load on the client, our proposal in this method is to modify the GBA architecture by removing the KsNAF entirely from the bootstrapping process as shown in Fig. 4 (grayed out part) in order to minimize the computational load.

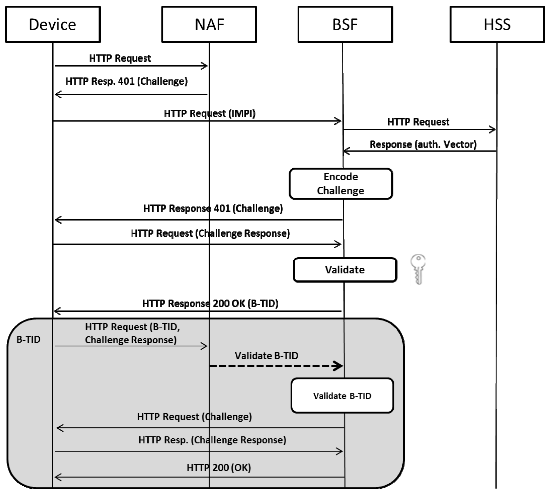
Device
NAF
BSF
HSS
HTTP Request
HTTP Resp. 401 (Challenge)
HTTP Request (IMPI)
HTTP Request
Response (auth. Vector)
Encode Challenge
HTTP Response 401 (Challenge)
HTTP Request (Challenge Response)
Validate
HTTP Response 200 OK (B-TID)
B-TID
HTTP Request (B-TID, Challenge Response)
Validate B-TID
Validate B-TID
HTTP Request (Challenge)
HTTP Resp. (Challenge Response)
HTTP 200 (OK)

Fig. 4. Compacted GBA

As a result, we relied on GBA architecture to authenticate the device only, followed by a TLS connection as done in the previous method to secure the transactions. Regarding TLS overhead, this method requires the use of TLS session resumption technique to avoid full handshake by storing secret information about the previous sessions on the server and reusing them. This method is applicable on C1 and C2 devices connected directly to the MNO. The main advantage here is that the IoT device is authenticated without the need to have a client certificate and the data is protected during transmission.

C. Method 3: C0 and Non-3GPP devices

C0 devices are not expected to connect directly to the server. Instead, these devices will use an intermediary gateway which is computationally decent to connect to a remote server. In this proposed method, we tackle C0 devices as well as non-3GPP hosts as depicted in Fig. 5.

According to Ericsson [9], the delegated GBA method can be used after establishing a bootstrapping process with the gateway to distribute session keys between the end-devices and the NAF, providing end-to-end encryption. Each session key is a function of the original bootstrapping key of the gateway, the device ID, and the NAF domain name. However, the delegated GBA has a main disadvantage which is inherited from GBA: the MNO is able to intercept the data between the NAF and the end-devices.

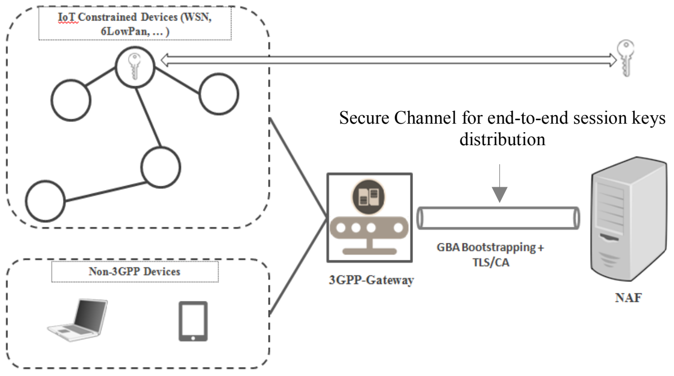
IoT Constrained Devices (WSN, 6LowPan, ...)
Secure Channel for end-to-end session keys distribution
GBA Bootstrapping + TLS/CA
3GPP-Gateway
NAF
Non-3GPP Devices

Fig. 5. Non-3GPP constrained / unconstrained

In the method 3, this problem was solved by relying on the proposed method #1 to establish a GBA+TLS connection between the gateway and the NAF as depicted in Fig. 5. This secure connection is used to distribute the session keys between the end devices and the NAF. As a result, the distribution of the session keys between the NAF and the end-devices is done securely since the MNO cannot intercept the data encrypted with K_{TLS}.

IV. Results Estimation and Discussions

In this section, we estimate the impact of the proposed solutions on the devices in terms of performance, memory consumption, and feasibility. Method II proposes a compact version of GBA followed by a TLS connection. GBA was implemented and tested on a C2 device in [10]. Table II demonstrates the results.

TABLE II. GBA RESULTS

RAM	**ROM**	**Time for 1 GBA run**	**Energy**
< 5 kB	44 kB	1.5 s	150mJ

Though the results are efficient, we expect more improvement as the KsNAF is no longer included in the solution and the data is not encrypted using MNO's key. On the other hand TLS is tested on the same device, and took 3423 ms to establish the handshake [11]. It is worth noting that this amount of time is spent one time only since the method implements session resumption. Regarding the first method, the performance and the memory consumption are negligible as the clients must have decent computational power and RAM size. Finally, C0 devices only perform symmetric encryption since key generation is done securely on the gateway.

V. Conclusions and Future Work

GBA has many advantages to the end user as well for the mobile operator as it can provide them with a new business model. Unfortunately, due to the trust reluctance from the clients, it is not heavily adopted. In this paper, we proposed a generic security framework based on the integration of GBA and TLS to provide mutual authentication and solve the trust issue. The next step is to implement the proposed methods and to establish a test bed using the BSF and the NAF software from Ericsson public servers, aiming at testing accurately the performance and its feasibility on different types of devices.

References

[1] C. Adams and S. Lloyd, Understanding PKI, 1st ed. Boston [u.a.]: Addison-Wesley, 2007.

[2] "Ericsson Mobility Report", 2016. [Online]. Available: https://www.ericsson.com/res/docs/2016/ericsson-mobility-report-2016.pdf.

[3] 3GPP; Technical Specification Group Services and System Aspects. "Generic Authentication Architecture (GAA); Generic bootstrapping architecture http://www.3gpp.org/DynaReport/33220.htm

[4] 3GPP; Technical Specification Group Services and System Aspects; "3G security". http://www.3gpp.org/DynaReport/33103.htm

[5] RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2", Tools.ietf.org, 2017. [Online]. Available: https://tools.ietf.org/html/rfc5246

[6] P. Bright, "Comodo hacker: I hacked DigiNotar too; other CAs breached", Ars Technica, 2017. [Online]. Available: https://arstechnica.com/security/2011/09/comodo-hacker-i-hacked-diginotar-too-other-cas-breached/

[7] M. Johnson, Cyber crime, security and digital intelligence, 1st ed. London: Routledge, 2016.

[8] RFC 7228 - Terminology for Constrained-Node Networks", Tools.ietf.org, 2017. [Online]. Available: https://tools.ietf.org/html/rfc7228.

[9] Ericsson, "Bootstrapping Security", 2016. [Online]. Available: https://www.ericsson.com/assets/local/publications/white-papers/wp-iot-security.pdf

[10] Using Generic Bootstrapping Architecture with Constrained Devices", Tools.ietf.org, 2017. [Online]. Available: https://tools.ietf.org/html/draft-sethi-gba-constrained-01

[11] ArduinoLibs: Cryptographic Library", Rweather.github.io, 2017. [Online]. Available: https://rweather.github.io/arduinolibs/crypto.html