Secure End-to-End VoIP System Based on Ethereum Blockchain

Elie F. Kfoury and David J. Khoury
Computer Science Department, American University of Science and Technology, Lebanon
Email: {ekfoury, dkhoury}@aust.edu.lb

Abstract — Blockchain is an emergent peer-to-peer technology that enables network decentralization and ensures availability by eliminating single points of failures. Furthermore, it provides data immutability by relying on consensus algorithms and protocols among peers to solve the trust concerns. In this article we provide a decentralized Blockchain-based keystore that holds the cryptographic public keys of users. This method eliminates the traditional mechanisms of distributing the public keys using central authorities (Certificate Authority). In a Voice over IP (VoIP) application, a caller can retrieve the public key of the callee from the Blockchain, and ensure the authenticity of the retrieved public key. Thus, the VoIP security complexity and intricacy are simplified on both signaling and media levels. Results of this solution implementation show that the call setup time is slightly affected compared to the existing secure VoIP solutions.

Index Terms—Blockchain, VoIP, Security, Ethereum, trust, decentralized, keystore, certificates.

I. Introduction

Voice over IP (VoIP) is a widespread technology used continuously by a huge number of end-users and corporates to deliver voice communications and media sessions over the IP protocol. Ensuring security and privacy in applications based on VoIP is vital since intercepting a multimedia session can result in eavesdropping the whole conversation by unintended parties and hence, violating users' privacies. In the last few years, several incidents related to VoIP attacks and breaches have occurred. For instance, VoIPTalk, a VoIP provider offering multimedia communication over the Internet, emailed its customers about a potential security breach involving Session Initiation Protocol (SIP) credentials after detecting external online attempts to exploit vulnerabilities in their infrastructure [1].

Security in VoIP applications appears to be a hassle especially in session key management. End-to-End encryption between end-users is critical as it forbids malicious parties and VoIP providers to listen to the media stream. Different schemes for key distribution exist but most of them require the use of a trusted third party that issues server and/or client certificates.

On the other hand, Blockchain is an emerging technology that provides decentralized network with no single point of failure and ensures data immutability through cryptographic functions and consensus algorithms and protocols. The Ethereum Blockchain is an open-source, public, distributed computing platform featuring smart contract (scripting) functionality [2]. The scripting functionality enables developers to write systems over the Blockchain and thus, benefit from the nature of distribution inherited from the Blockchain technology.

The objective of this article is to create a novel approach for trustless key distribution management based on the Ethereum Blockchain to resolve the complexity of SIP security and the third-party trust issues. The main contributions of this paper include: 1) Elimination of the public key infrastructure (PKI) and the Certificate Authority (CA), 2) Implementation of a generic handshake protocol for the PSK (Pre-shared Key) based security protocols (TLS-PSK, DTLS …) 3) Simplifying end-users wallet management functions, 4) Providing E2E multimedia (voice and video) encryption.

The paper starts with an introduction on Blockchain technology and VoIP systems. Section II describes the security contemporary security measures used against VoIP systems. Section III presents our proposed system to provide VoIP security based on Blockchain. Section IV discusses the implementation of the system and the results. We conclude the paper with the intended future work.

II. Background on VoIP and Security

VoIP security has been practiced for years by major carrier networks and researchers. Like any other system, the goal is to ensure the classical objectives of information security: Authentication, confidentiality, integrity, and availability. VoIP protocols are divided into two main categories: 1) Session control protocols (SIP, H.323 …) and 2) Media control protocols (RTP, RTCP …). Session control involves establishing and managing the session by negotiating the call parameters such as the available codecs, client's capabilities, encryption key, and others. Media control protocols are used to transfer audio and video streams between the end-users by carrying the media payload encoded by a pre-negotiated codec. Attacks on VoIP systems can target both protocols.

We list hereafter the basic SIP threats targeting the classical information security pillars mentioned earlier and the security mechanisms that are commonly deployed against these threats.

Manuscript received February 12, 2018; revised July 4, 2018.
doi:10.12720/jcm.13.8.450-455

1) Call/Session hijacking and impersonation

HTTP Digest mechanism defined in RFC 2617 [3] is used between users to proxies and users to users. This method is used to mitigate against call and registration hijacking and impersonation threats. It provides a way for a server or a UA (User Agent) to challenge other UAs or servers for authentication. When the server receives a request from the client, it generates a nonce value and sends it along with a realm to the client as a challenge. The client computes the response by hashing the nonce, username, password, and the realm and then sends it to the server. Finally, the server computes the same hash and compares it with the received response. The main drawback of this mechanism is the risk of having the SIP client credentials stolen, which can result in allowing the attacker to impersonate users without any proves that the legitimate client is actually using these credentials.

2) Eavesdropping on SIP signaling

A malicious party can track and record whom a user is communicating with by observing the SIP messages. The following mechanisms are used to mitigate against this threat:

a) Encryption at IP layer (IPSec): IPsec [4] can be established between two internet hosts and performed at the operating system. It is difficult for an application like SIP to know if IPsec is in place.

b) Transport Layer Security (TLS): SIP can utilize TLS to encrypt signaling messages as encryption on the transport layer masks application layer data. TLS provides only a single hop of confidentiality and authentication between UA and a proxy server; hence, end to end confidentiality is not fulfilled if one of the hops is not TLS protected. The main problem with TLS is the requirement of certificates and single hop encryption. Typically, there is more than one hop between two communicating UAs.

c) End to end confidentiality using S/MIME: Using S/MIME [5] protects only the Session Description Protocol (SDP) [6] part of the SIP message because the Header part should be sent in clear to be able for proxy server to route the message to the right destination. S/MIME allows UAs to encrypt SDP bodies within SIP and secure bodies without affecting the message header.

3) Eavesdropping on media

Malicious parties can track and monitor a media conversation between end users. Secure Real Time Protocol (SRTP) [7] is used to secure the RTP media streams and hence, mitigate this threat. The main limitation in this method is the requirement of a master session key distribution mechanism. SIP can help to establish a secure media session by exchanging the master key through the SDP part of the SIP messages. Multimedia Internet Keying (MIKEY) uses the SDP field for transporting a media key which can be carried in SDP in a=key-mgt-attributes. ZRTP is another protocol for media session exchange by using RTP's header extension mechanism.

4) Denial of service

Attacks against availability target service interruption. Examples of these threats are intentional call flooding, misuse attacks (invalid registration, call request, call control), and unintentional attacks (DNS, call billing…). Mitigating DOS attacks is not tackled in this paper as it can be achieved by deploying Anti-DDoS and Intrusion Prevention Systems (IPS).

As a conclusion, the ultimate security of VoIP based on SIP is not achieved by securing a single protocol. It involves securing a complete system and relies on several standardized encryption protocols including TLS, IPsec, and S/MIME for SIP signaling encryption, SRTP for RTP media encryption, and requires a PKI infrastructure and trusted certificates authorities. Authentication based on HTTP DIGEST is only good for authenticating to the first hop. SIP identity needs improvement through certificates and credentials. TLS protocol assures the transport security Hop-by-hop and S/MIME Suffers from the same key distribution issue as the TLS.

III. Proposed System

The main problem with the aforementioned security mechanisms is the requirement of having a public key infrastructure (PKI) to ensure rigid security. Therefore, certificates are mandatory to verify the authenticity of the client and the server. In our solution, we intend to eliminate the usage of the PKI by relying on the Blockchain network to store and manage the subscribers' public keys. Fig. 1 below illustrates overview architecture and the interfaces of the system.

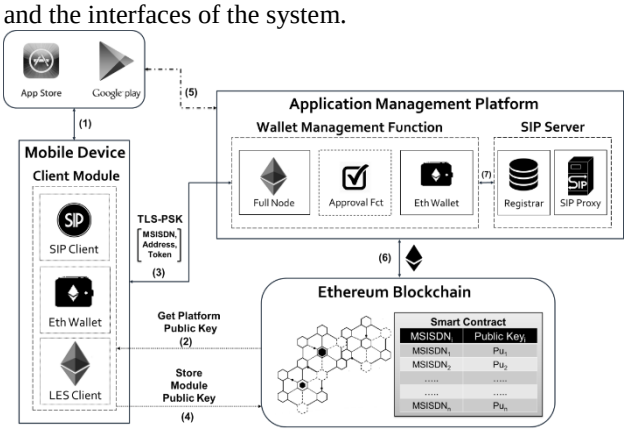

Fig. 1. System architecture and interfaces.

A. System Components

We describe hereby briefly on high level the main interfaces of the system:

The user downloads the mobile application from the Google Play store into an Android operating system (1). The application includes an interface to the Ethereum Blockchain network. The application generates an empty Ethereum wallet and securely requests Ether from the wallet management function in the server (2, 3). After the successful transfer of Ether, the mobile app generates and stores a key pair in the protected Android keystore. Moreover, it stores the public key in the Blockchain

through the smart contract (4) then requests approval from the wallet management server (3). The server validates the request transaction by contacting the Google Play service to verify the authenticity of the app (described later in more details) (5). If the request is authentic, the wallet management approves the storing transaction in the Blockchain (6). It should be mentioned that this solution does not affect in any means the VoIP architecture based on SIP. A new interface between the wallet management and SIP server is defined to automate the SIP credentials configuration (7).

The main components involved in this system are: 1) Mobile application, 2) Wallet management server and 3) Ethereum smart contract.

1) Mobile application (VoIP client)

The mobile app comprises the following components: a standard SIP client, a light Ethereum client that uses the Light Ethereum Subprotocol (LES) [8] to interface the Blockchain, and a dedicated module to interwork with the wallet management function in the server. The sequence of events that take place after the installation of the application on the phone is depicted in Fig. 2.

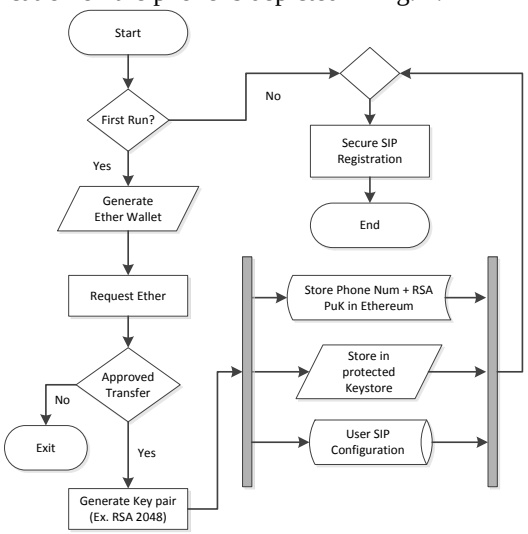

Fig. 2. Mobile application flowchart.

The application generates an empty Ethereum wallet on its first launch, and prompts the user to fill it with sufficient Ether to be able to store the key in the Blockchain network. The transfer of Ether to the client is mandatory as storing data or changing the state of a contract in the Blockchain requires transaction fees which are priced in Ether. To simplify this process for end-users, the application provides an in-app purchase dialog that prompts the user to pay for the Ether transfer in fiat currency. After the purchase is complete, the mobile application sends the newly generated wallet address, the purchase receipt, and the mobile phone number to the wallet management as an encrypted payload (interface 2 in Fig. 1). The encryption protocol used is described later in the Generic handshake protocol section (Section C). The application then generates the subscriber key pair and stores it in the protected Android keystore.

The wallet management validates the request and transfers sufficient Ether to the client. The validation process is described next in the wallet management function section (Section 3).

Upon successful transfer, the client stores its public key in the smart contract along with the receipt token and the subscriber number. More on validating the storage transaction in the Blockchain is explained next in the smart contract section (Section 2).

2) Smart contract

A smart contract is an account holding object that contains distributed code executed by the Ethereum Blockchain autonomously. Any user owning a wallet with sufficient Ether can deploy a smart contract to the Blockchain by sending it as a signed transaction. The contract code is written in Solidity [9], a high-level language with syntax similar to JavaScript designed to be used with the Ethereum Virtual Machine (EVM).

In our system, we developed a smart contract that holds the public keys of the subscribers by using a mapping data structure indexed by the receipt tokens. Fig. 3 illustrates the smart contract main functionalities:

a) *addClient(phoneNumber, publicKey, Receipt):* Any user can call the function addClient and insert a new record containing its public key in the Blockchain. However, this is vulnerable to identity theft: a malicious user might register the phone number and impersonate another user. To solve this, we created two mappings in the smart contract: pendingList, approvedList. The pendingList is a mapping that contains any client that calls the addClient function. In order to get approved, the wallet management must copy the entry from pending to the approved list (described next in the wallet management section (3)).

b) *approveClient(phoneNumber, Receipt)*: approves the client by moving the entry from the pending list to the approved list. Only the contract creator, which is typically the wallet management, has the right to approve subscribers.

c) *getClient(phoneNumber):* a function that accepts the phone number of the destination address and returns the public key.

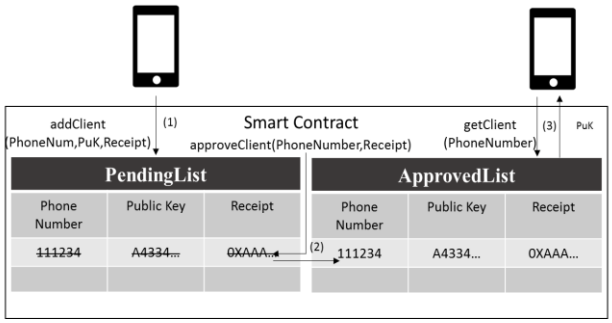

Fig. 3. Smart contract functionalities.

3) Wallet management function

The main functionalities of the wallet management are:

a) Ether Transfer: When the wallet management receives

the encrypted payload mentioned earlier in the VoIP Client section, it retrieves the receipt token which is signed by Google Play's application private key. The server verifies the receipt by decrypting the token using the application's public key. The aim of this verification is to make sure that the client requesting Ether has paid for the transfer fees. If the verification is successful, the server transfers Ether from its own wallet to the address extracted from the payload through its Ethereum client.

b) Storage Transaction Approval: After the mobile client receives the amount of Ether and stores the data in the smart contract, it contacts the wallet management requesting the storage approval by sending securely the receipt token. This is mandatory as the storage in the Blockchain network can be done by any user without any control. It is worth mentioning that a malicious user cannot overwrite a stored record even if the stored token is compromised. Furthermore, since the wallet management is the contract deployer, it can move the entry indexed by the receipt token from the pendingList to the approvedList.

c) SIP Credentials Configuration: Another important role of the wallet management is the storage of the SIP credentials (username and password) in the VoIP provider's database. The username is the subscriber's phone number, and the password is the hash of the receipt token. Hence, there is a need for an interface between the wallet management function and the SIP database.

B. Light Client Security

Considering that this platform will mainly be used on mobile devices, it is impractical or nearly impossible to download the whole Blockchain data to the phone as the size is continuously growing (~380 GB Dec 2017). As a result, the light client is developed to enable building Ethereum nodes that run on all computers and laptops large and small, smart phones, and even internet of things devices [10]. The size of the lightchain data by the time of writing (Dec 2017) is 0.005GB, which is affordable on a mobile device. It is worth noting that this lightchain data will be downloaded once only on the mobile for all Ethereum decentralized apps.

The main building block of the light client is the Merkle Tree [11]. It is a tree data structure that allows efficient and secure verification of the contents. To query the Blockchain, the client sends a request to light client servers. The server simply finds the object, fetches the Merkle branch (the list of hashes going up from the object to the tree root) and replies back to the light client with the branch.

Due to the usage of Merkle trees, a client can get a secure assurance about the state of a smart contract in logarithmic complexity.

C. Generic Handshake Protocol for the PSK (Pre-Shared Key) Based Security Protocols

An important contribution of this platform (smart contract in Ethereum) is the elimination of the public key infrastructure (PKI) and the Certificate Authority (CA), and the implementation of a generic handshake protocol for the PSK (Pre-shared Key) based security protocol like TLS-PSK, DTLS-PSK, SRTP, etc…

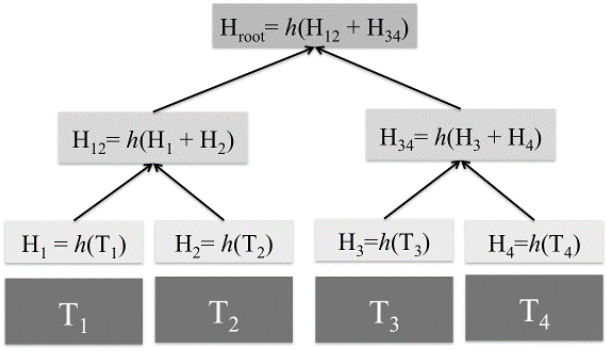

Fig. 4. Merkle tree

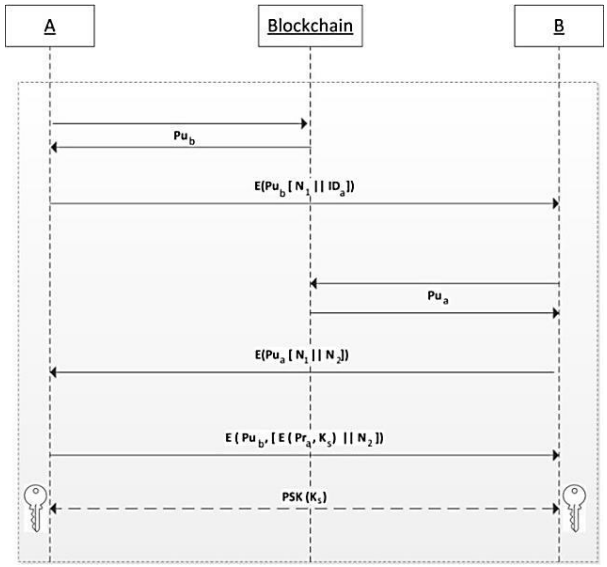

Fig. 5. Generic handshake protocol for the PSK

This out of band generic protocol is responsible for the distribution of the PSK key between two entities.

This protocol implements the Needham–Schroeder method [12] to distribute the Session key using asymmetric encryption.

1) The originator *A* requests the public key of the responder *B* from the smart contract in the Blockchain.
2) *A* encrypts with the public key of *B* Pu_b, a randomly generated nonce (N_1) which is used to identify this transaction uniquely, and the Identifier of *A* (ID_A)
3) *B* requests *A*'s public key Pu_a from the Blockchain.
4) *B* sends *A*'s nonce (N_1) and a new nonce generated by *B* (N_2) encrypted with Pu_a. This nonce (N_1) helps *A* ensure that the correspondent is *B* as only *B* can decrypt the message.
5) *A* returns (N_2), encrypted using *B*'s public key, to assure *B* that its correspondent is *A*.
6) *A* encrypts with Pr_a a generated secret key K_s and then encrypts the whole message along with (N_2) with Pu_b

7) *B* decrypts with Pu_a and Pr_b to retrieve the secret key.

D. End to End Secure VoIP System

The end to end security proposed in this paper does not affect the standard building blocks of a VoIP system based on SIP. But it describes a new security mechanism that is not related in any means to the existing solutions described previously. The security solution instead relies on the Ethereum Blockchain to hold the public keys of all subscribers. These keys will be used to produce session keys that secure the SIP control signaling between SIP clients and servers, and the media between two SIP clients. Hence we divide this section into two parts:

1) Establishment of a secure channel between VoIP client and SIP Server:

The SIP messages (requests and responses) are secured using TLS-PSK. The key is distributed using the generic handshake protocol for the PSK-based security protocols described in section C. After the establishment of the TLS session, the client securely sends a SIP Register message containing the credentials and registers to the server. All further SIP messages for setting up the call are sent encrypted. This method ensures authentication of the end client even if the SIP credentials are compromised. Hence, it mitigates against the SIP credentials theft discussed in the introduction.

2) Secure peer to peer Call setup:

Once the communication channel between the SIP client and the SIP server is secured, end-to-end encryption between endpoints can be set. The VoIP client fetches the public key of the callee from the Ethereum Blockchain and consequently generates a random session key to be used as a master key in SRTP protocol in order to encrypt the RTP voice packets between two endpoints. The master key is encrypted using the destination's public key, and then sent using the MIKEY protocol through the SDP part of the SIP invite as depicted in Fig. 6.

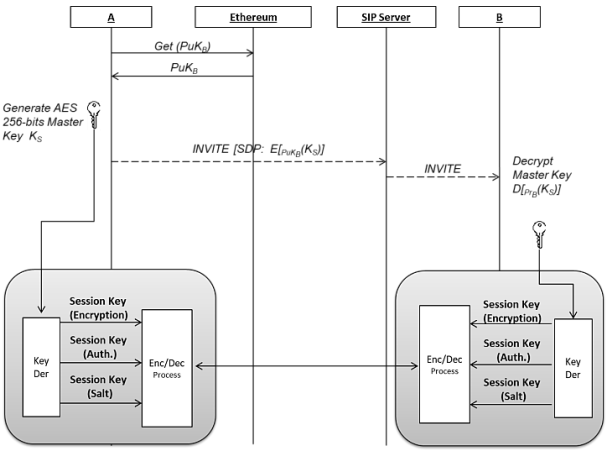

Fig. 6. End-to-End media security

Upon reception, the master key is decrypted using the private key, and then is passed to SRTP to be used as the session's master key.

Another alternative is to distribute the key using Station-to-Station (STS) [13] key agreement scheme as demonstrated in Fig 7. The main advantage of this approach is the usage of Diffie-Hellman as the key exchange (ensure Perfect Forward Secrecy).

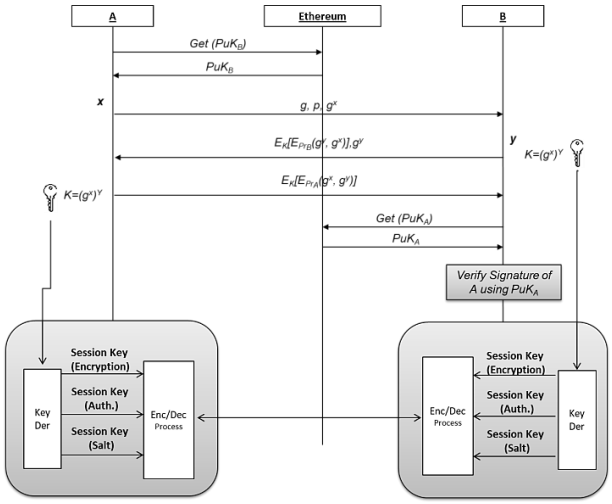

Fig. 7. STS End-to-End media security through RTCP

Since the STS protocol requires interaction between the endpoints to distribute and to authenticate the key, RTP Control Protocol (RTCP) [14] is used.

IV. Results and Simulation

To validate the proposed system, we implemented the solution using the following components: a) Kamailio: an Open Source SIP Server [15], b) RTPProxy: a high-performance media proxy for RTP streams, c) Geth Light Client (Ethdroid) [16]: Easy-to-use Ethereum Geth wrapper for Android, d) Solidity: Contract-oriented programming language for writing smart contracts, e) NodeJS: Server side scripting (Wallet management server), f) SIPDroid as the Android SIP client [17]. Fig. 8 demonstrates a SIP Invite message in which master key ***k*** is sent encrypted through the SDP payload.

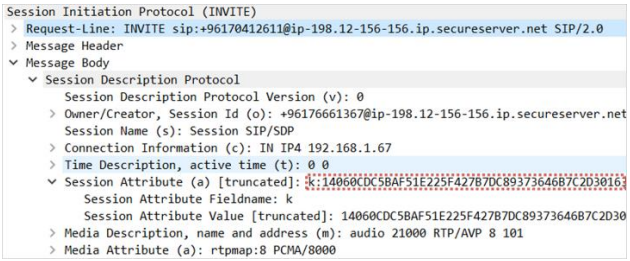

Fig. 8. SIP Invite with SDP Payload showing encrypted ***k***

Table I shows the time taken to setup a call when sending the key in the SDP part of the SIP message. Storing the public key in the Ethereum Testnet Blockchain took 11.5 seconds to be validated. This time is only spent once upon configuration. Retrieving the public key on the other hand took only 507ms; this amount of time is spent on each call setup.

TABLE I: CALL SETUP TIME RESULTS

Key retrieval from Blockchain	Call Setup Time
507ms	1218ms

The total call setup time is composed of: key retrieval from the Blockchain, symmetric key generation, public key encryption, SIP signaling, private key decryption, and SRTP key derivation. It took 1218ms to setup the end-to-end secure call between the parties.

V. CONCLUSIONS

In this paper, we have presented a novel approach for trustless key distribution management based on the Ethereum Blockchain to resolve the complexity of SIP security and the third-party trust issues. In another research project, we have proposed a method for enhancing trust based on the integration of GBA and CA [18]. Our implementation was tested on the Ethereum Testnet. Results showed that the solution performs well in terms of call setup time. Moreover, it has a minimal impact on the overall VoIP architecture. Regarding the future work, we are in the process of designing a generic secure key distribution framework based on the Ethereum Blockchain that works on any application and any platform (mobile app, servers, IoT devices, browsers…).

REFERENCES

[1] C. Osborne. (2017). VoIPtalk admits to possible data breach | ZDNet. [Online]. Available: http://www.zdnet.com/article/voiptalk-admits-to-possible-data-breach/

[2] Understanding Ethereum (Report). CoinDesk. June 24, 2016.

[3] RFC 2617 – IETF. HTTP Authentication: Basic and Digest Access Authentication.

[4] RFC 6071 – IETF. IP Security (IPsec) and Internet Key Exchange.

[5] RFC 5751 – IETF. Secure/Multipurpose Internet Mail Extensions.

[6] RFC 4566 – IETF: Session Description Protocol.

[7] RFC 3711 – IETF: The Secure Real-time Transport Protocol.

[8] Ethereum. Light client protocol. Ethereum Wiki. (May 2016). [Online]. Available: https://github.com/ethereum/wiki/wiki/Light-client-protocol

[9] Solidity Language. [Online]. Available: http://solidity.readthedocs.io/en/latest/

[10] Ethereum Blog. (2017). Merkling in Ethereum - Ethereum Blog. [Online]. Available: https://blog.ethereum.org/2015/11/15/merkling-in-ethereum/

[11] R. C. Merkle, "Method of providing digital signatures," U.S Patent US4309569, Jan. 5, 1982.

[12] R. M. Needham and M. D. Schroeder, "Using encryption for authentication in large networks of computers," *Communications of the ACM*, vol. 21, no. 12, pp. 993–999, Dec. 1978.

[13] S. Blake-Wilson and A. Menezes, "Unknown key-share attacks on the Station-to-Station (STS) protocol," in *Public Key Cryptography, Lecture Notes in Computer Science*, 1560, Springer, 1999, pp. 154–170.

[14] RFC 6642 – IETF. RTP Control Protocol (RTCP) Extension for a Third-Party Loss Report.

[15] Kamailio. The Kamailio SIP Server Project [Online]. Available: https://www.kamailio.org

[16] Ethdroid. Easy-to-use Ethereum Geth Wrapper for Android. [Online]. Available: https://github.com/ethmobile/ethdroid

[17] Sipdroid. Free SIP/VoIP Client for Android. [Online]. Available: http://sipdroid.org/

[18] D. Khoury and E. Kfoury, "Generic hybrid methods for secure connections based on the integration of GBA and TLS/CA," in *Sensors Networks Smart and Emerging Technologies*, 2017, pp. 1-4.

Elie F. Kfoury holds a B.S degree in Computer Science and has graduated with high distinction from AUST in 2015. He is currently pursuing an M.S degree and working as an academic and teaching assistant in the Computer Science and ICT departments at AUST. He has published several research papers in international and local conferences. His research interests include IoT, Blockchain, VoIP, distributed systems, and Information Security. He developed an IoT security platform based on Ethereum Blockchain which was selected by Ericsson Garage Startup.

David J. Khoury has More than 30+ years' experience in the Telecommunications and Technology field. He held different positions at Matra and Ericsson mainly in France and Sweden in the Research and Developments area, Product and System management. He holds a degree in Masters of Engineering in Telecommunications from E.S.I.B, Beirut Lebanon in 1983. He was leading a group for the development of the ISDN, where a generic platform for multiple accesses in the Ericsson main exchange AXE was developed. He was involved in the early evolution of the GSM towards an IP based network and as well the early studies of 3G/WCDMA, HSPA and LTE. The last 4 years he started teaching at AUST Beirut at the computer science department mainly the VoIP, security, cryptography, mobile systems, and Blockchain courses. He holds 4 US patents and a number of publications and he was the winner of Ericsson business Innovation 2003.