

Description of the MedicIP Communication Platform with Emphasis on Data Security Issues

Normand Bédard^a, Alain Houle^a, Guy Lépine^b, Frédéric Mailhot^a

^a Université de Sherbrooke, Sherbrooke (Qc), Canada

^b M5T, Sherbrooke (Qc), Canada

Abstract

MedicIP, a system for communicating by voice and exchanging data between medical personnel during an emergency situation has been designed and implemented. The security aspects of the system are reviewed, including authentication of participants, data integrity and access control, robust resistance to external attacks. The performance of the various protection elements is evaluated.

Keywords: Patient Care Team, Computer Security, Secure Multimedia Conference

Introduction

Effective communication between ambulance crew and medical personnel at a hospital emergency department is known to have a very positive impact on both patient care quality and efficiency of the emergency operations [1]. As an example, making it possible for the ambulance crew to share real-time electrocardiograms (ECGs) through multimedia conferencing with an emergency physician and a cardiologist can surely help in decision making for patient transportation and can also contribute to make sure a medical team is properly prepared to receive the patient [2-3]. We present the MedicIP communication platform, which is actually being developed at Université de Sherbrooke. In particular, our interest lies in the security aspects of such a distributed communication platform. First, we describe the architecture of the MedicIP platform itself. We then present solutions for security issues that have been identified. The MedicIP vision: multimedia communication over IP anytime, anywhere.

The MedicIP project began in 2005 with the idea of building a communication system able to support simultaneous voice communication and real-time ECG transmission from an ambulance to a hospital. The MedicIP platform allows more than point-to-point communication: it also enables multimedia conferencing for up to ten parties [4]. In a typical case scenario, an ambulance crew establishes a communication link with an emergency physician at a first-line hospital. Depending on the analysis of the physiological signals received from

the ambulance, the emergency physician has multiple options: 1) he can decide to prepare his team to receive the patient; 2) he can invite another physician, potentially a specialist from another institution, into the conference to get his opinion; 3) he can decide to reroute the ambulance to a more appropriate institution given this institution is able to receive the patient.

The MedicIP platform is an IP (Internet Protocol) -based communication platform. Since the beginning of the project, we did not consider the physical radio links required for ambulance communication and assumed that any involved party has access to an IP network. Future work will concentrate on the radio communication aspects as many new technologies are now emerging (e.g. APCO P25 standard [5]).

The MedicIP signaling functions, required to establish communication sessions between parties, are implemented using the SIP protocol (RFC 3261) [6]. A full-mesh conferencing protocol was implemented over the SIP middleware [7]. The full-mesh conferencing scheme is not the most bandwidth efficient one as $N(N-1)/2$ data streams are required in opposition to only N data streams for the more conventional client-server conferencing scheme. However, as it is not very likely that more than ten parties would be involved in the kind of emergency situation we are aiming, bandwidth efficiency is not our first concern. The important issue is the reliability of the system because of the critical nature of the MedicIP task. The full-mesh conferencing scheme is then an excellent candidate as it does not have a single point of failure. Being a distributed scheme, any party can leave an ongoing conference without leaving other parties in the dark. This is not the case with the conventional client-server scheme where the central server must be always reachable and available.

With respect to the type of information that can be communicated using the MedicIP platform, we began by considering the simultaneous transmission of voice and real-time ECG. Voice signals are encoded according to conventional voice-over-IP (VoIP) standards [8]. We selected ecgML [9] as a representation format for ECGs. Based on the XML technology, ecgML is intrinsically compatible with SIP technology. Data streams are transported using the RTP protocol (RFC

3550) [6]. Since we combine different types of information for simultaneous transmission, we are exploiting multimedia communication over IP (MMoIP).

In January 2007, a team of six undergraduate computer engineering students received the mandate to redesign the MedicIP platform and to incorporate a sophisticated graphical user interface (GUI) (Figure 1). That work, made under the umbrella of a final-year design project in our computer engineering curriculum at Université de Sherbrooke, also included interaction of the MedicIP platform with a presence server used to locate physicians and specialists [10].

Since January 2008, the development of the MedicIP platform has been focused on security issues, under the project name SecureMedic. The remaining of this paper reports our work on these issues.

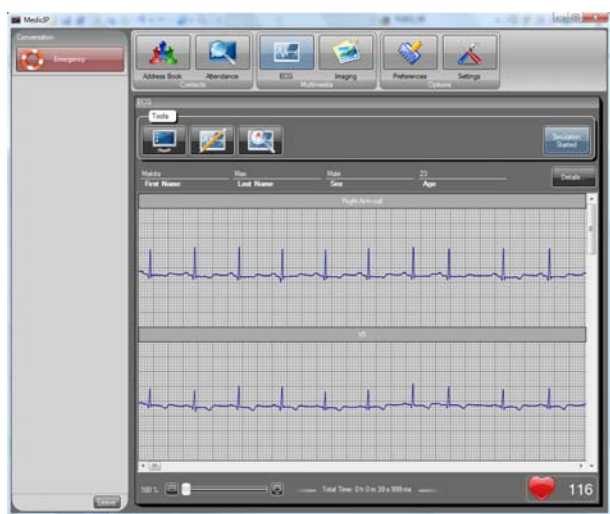


Figure 1- The MedicIP GUI

Methods

MedicIP is intended as a communication tool both between distant physicians and also between emergency personnel and ambulance crews on their way to the emergency departments. It is a Windows-based software client application that runs on traditional computers. The software system consists of approximately 45,000 lines of C# code, which has been extensively tested and validated by a team of physicians.

User authentication

The SecureMedic project completely redesigns the authentication process of the MedicIP prototype. The initial objectives and goals of the SecureMedic authentication architecture answer problems identified in the MedicIP prototype:

- Have a transparent mutual authentication between the server software and the client software. This avoids unauthorized malicious entities to form part of a communication.
- Have secure communications between entities during

an authentication.

- Have a robust and reliable centralized authentication server.
- Have an efficient way for users to authenticate themselves.
- Have flexible authentication alternatives if there are network problems or if the authentication server is unavailable.
- Allow users to securely receive their credentials after a successful authentication.

The SecureMedic architecture is a PKI (Public Key Infrastructure). Its role is to provide services that allow the authentication of entity by another. Two major elements are present in the SecureMedic PKI:

- Private keys
- Public keys, embedded into X.509v3 digital certificates

The system is based on 2048 bits RSA keys and the SecureMedic architecture uses three kind of digital certificates:

- Generic server certificate (root). This certificate holds the server software credential. The private key associated to the generic server certificate's public key is hardcoded into the server software. Doing so lowers the burden of certificate management.
- Generic client certificate. This certificate, signed by the root certificate, holds the client software credentials. The private key associated with the generic client certificate's public key is hardcoded in the client software.
- User certificates. Each user credentials are held within a unique certificate, signed by the root certificate. Initially, the authentication server has all the client's certificates and associated private keys.

The next sequence diagram shows how the user authentication and the mutual authentication between client and authentication server works.

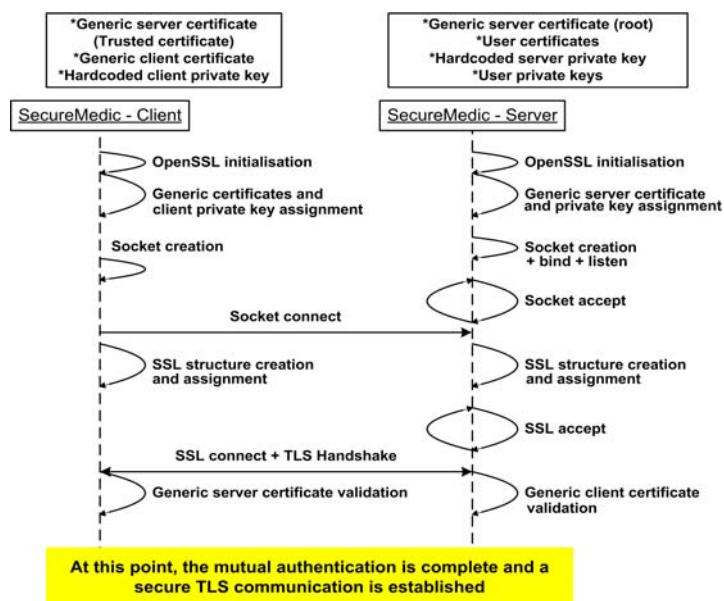


Figure 2 - Device mutual authentication

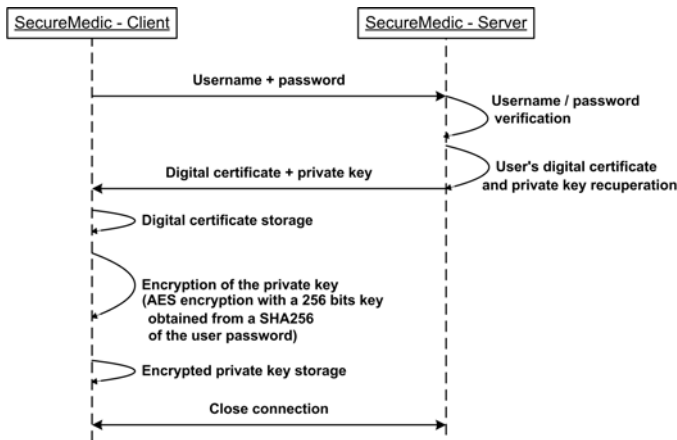


Figure 3 - User authentication

For performance reasons, the server software is multithreaded. Each socket connection creates a new thread that is responsible for user authentication and mutual device authentication. The authentication server also includes security protections against some types of Denial of Service (DoS) attacks. Another thread, called the “Cleaner”, is responsible for cleaning up inactive socket connection threads, possibly caused by a DoS. This thread can be seen as a garbage collector for the connection list.

The authentication server contains a list which can be filled with a maximum number of threads. If the list is full, there is no socket connection listening. Otherwise, socket connections are possible and a thread can be created and added to the list.

The “Cleaner” thread has an internal timer which depends on the number of threads in the list; the value of the timeout is inversely proportional to the number of connections in the list. If there is a small number of connections in the list, there is no disadvantage to let a long timeout running. However, a large number of connections in the list indicates higher traffic on the network, causing congestion. This is why the “Cleaner” thread is advantageous to cleanup the list more rapidly. When the timer expires, the “Cleaner” kills and removes the oldest connection from the list and restarts its timer.

Connections stay in that list until they complete a TLS handshake (see Figure 2). The handshake allows the client and the authentication server to authenticate the peer and agree to a symmetric keys used for data [11]. Upon the successful completion of the TLS handshake (which includes entity authentication), the connection is removed from the list and the user authentication takes place. Every thread is associated with a timer which ensures none of the threads will consume computer resources forever if the authentication is not complete. The “Cleaner” thread does not manage threads that are removed from the list.

This method protects the system against malicious socket connections which could try to overload the authentication server. Indeed, if the list is full, no socket will be made created, and no thread will be created, meaning that no computer resources are used by those connection attempts. Although malicious connections are possible when the list is not empty, they will be killed by the “Cleaner” thread if they are inactive or will be killed by the authentication server if they fail the TLS handshake. Since there is no complete solution against a DoS [12] attack, this method is not immune to the authentication server refusing legitimate connections. It’s a trade-off between security, robustness and reliability. Our method assures that a legitimate authentication request present in the list will be processed within a predefined time period.

To summarize, the list is used as a buffer protection, which allows the authentication server to filter the legitimate authentication requests. It is also called a perimeter defense.

Privacy of information exchanges

A MMoIP communication exchange is done over a TCP/IP network. This type of network is intrinsically insecure. Therefore, additional protocols need to be used in order to provide communication security over this insecure network.

First, the SIP signalling must be protected. The MedicIP software client uses SIP, which in turn supports TLS (RFC5246) [6], the ubiquitous protocol for securing communications over TCP (RFC793) [6]. This protocol provides:

- Server authentication (the entity which will receive and process the SIP requests)
- Optional client authentication. In the process of securing the MedicIP application, client authentication using X.509v3 certificates has been chosen
- Data privacy, using a negotiated symmetric-key algorithm
- Data integrity, using an also negotiated HMAC algorithm

Note, however, that SIP over TLS only secures the conference signalling between the parties. The actual multimedia data exchanged need to be protected by other means. This is where SRTP/SRTCP comes into play. When initiating a multimedia session with SIP, the session description (i.e. the devices’ multimedia capabilities) is specified using SDP (RFC4566) [6] and negotiated as per RFC3264 [6]. The negotiation results in a chosen codec, ptime, IP address and port for the exchange to take place, usually using RTP.

However, RTP is insecure and additional protocols need to be used to protect the multimedia data. Therefore, a secure version of RTP, SRTP (RFC3711) [6], is put in place, providing a symmetric-key algorithm to protect the

multimedia communication. The key needs to be exchanged by other means than SRTP. Usually, the key exchange will be done using MIKEY (RFC3830) [6], inserted in the SDP exchange. The SDP exchange being a payload of SIP packets, the latter being secured with TLS, there is no way an attacker could eavesdrop on the keying material.

Therefore, SIP over TLS is used to protect the conference signalling, in conjunction with SRTP/SRTCP to protect the multimedia exchanges.

Security of stored data

Secure data storage was not an initial objective of the MedicIP development team. Both the client and authentication server lack security measures to ensure data privacy and integrity. SecureMedic fixes a number of those problems on the authentication server side, in particular data previously stored in unprotected XML files.

First of all, the authentication server has been completely redesigned in C++, on OpenBSD (www.openbsd.org), considered by many experts as one of the most secure operating system.

A local database, whose access is authorized only with the required credentials, contains all the information needed by the system (user accounts, digital certificates and private keys filenames, user privileges, hashed passwords, etc.).

Users' digital certificate files and private keys indicate specific file access restrictions, which are managed by the operating system. In addition to the data protection, all the hard drive is encrypted with the TrueCrypt (www.truecrypt.org) which allows on-the-fly encryption and decryption of needed files by the authentication server.

On the client side, MedicIP was storing two types of data: address book and ecgML files. No file restrictions protected that data, therefore anybody who had access to the computer could read, modify or delete it. With SecureMedic, all that confidential information is the AES symmetric key cryptography algorithm (AES is a standard adopted by the U.S. Government [13]). The AES key is generated on demand using the user's private key with a Password-Based Key Derivation Function (PBKDF2) that is part of RSA Laboratories' Public-Key Cryptography Standards (PKCS) series, specifically PKCS #5 V2.0 [14].

Discussion

SecureMedic uses existing security concepts or products with the aim of fixing security vulnerabilities of an existing prototype.

This means that the project has limits introduced by the technological choices made. In the case of the authentication scheme using the PKI's concept, there are some restrictions. Indeed, the architecture developed does not offer all the functionalities of a complete Internet X.509 public key infrastructure certificate management protocol (RFC 2510) [6]. The development of a complete PKI architecture could itself be a project, and will possibly be added to SecureMedic in the future.

As discussed before, we introduced a protection against a flood DoS attack that could overload the authentication server with authentication requests. Many other types of DoS attacks exist, like infrastructure DoS, service DoS, network DoS, application DoS, etc. SecureMedic does not claim to be able to counter advanced and sophisticated DoS attacks.

One of our objectives was to have flexible authentication alternatives if there are network problems or if the authentication server is unavailable. Two solutions are currently being designed:

- Relayed authentication
- Local authentication

For the relayed authentication, it is possible that a client is unable to reach the authentication server, but is able to reach another client that has been already authenticated. With the use of the digital certificate that signed the client devices (two client devices can do mutual authentication), a protocol will be designed to allow a client to use another authenticated client as a relay to "talk" to the authentication server and authenticate himself.

As a last resort, a client device will be able to authenticate a user locally with a local history of successful authentications. This history will be encrypted with the client generic private key and hash functions will ensure that no modifications have been made on the history. Because the user's digital certificate and private key are stored on a client device (see Figure 3), the client software will be able to use it for other uses (discussed in the above privacy of information exchanges section). Since the prototype is used in a highly critical situation involving patients' lives, it is important to have this trade-off between security and accessibility, resulting in a functional system even if the central authentication server is unavailable. A 99.99% server uptime rate involves a 52 minutes downtime period over a year, which indicates the necessity of that trade-off.

Conclusion

The MedicIP prototype fills a need in the traditional way patients are transported to a hospital, by allowing ambulance crew and emergency personnel to efficiently interact earlier in the process.

Because this product uses IP network and Internet as a communication medium, the security aspects of the project needed to be addressed.

SecureMedic focuses on some important security aspects: secure authentication process, secure information exchanges and secure data storage. Since computer systems are never completely secure, future development is planned on SecureMedic to minimize the risks associated with a networked environment.

Acknowledgements

We thank Université de Sherbrooke's Department of Electrical Engineering and Computer Engineering, which sponsored the team of six undergraduate students who worked on the latest version of the MedicIP platform. The members of that team, Benoit Beauchemin, Francis Beaulé, Marc Boissonneault, Jean-François Desjean-Gauthier, Guillaume Dubeau and Francis Ruel, were essential to the design and realization of the latest prototype and have all our gratitude.

For information, please contact

Alain Houle: alain.houle@usherbrooke.ca
Normand Bédard: [nor-
mand.bedard@usherbrooke.ca](mailto:mand.bedard@usherbrooke.ca)
Frédéric Mailhot : frederic.mailhot@usherbrooke.ca
Guy Lépine: glepine@m5t.com

References

- [1] Rowlands, A., "An evaluation of pre-hospital communication between ambulances and an accident and emergency department", *Journal of Telemedicine and Telecare*, (2003), 35-37.
- [2] McNamara et al., "Effect of door-to-balloon time on mortality in patients with ST-segment elevation myocardial infarction", *J.Am.Coll.Cardiol.*, 2006, vol.47, pp.2180-2186.
- [3] Swor et al., "Prehospital 12-Lead ECG: Efficacy or Effectiveness?", *Prehospital Emergency Care*, July/September 2006, vol.10, no.3, pp.374-377.
- [4] Dorais-Joncas, A., Elleuch, W. and Houle, A.C., "A Conference Mechanism for the Simultaneous Transmission of Voice and Medical Information using SIP", 19th IEEE Canada Conference on Electrical Engineering and Computer Engineering, 2006, Ottawa, Ontario, Canada.
- [5] TIA/EIA 102.BAAA Project 25 – FDMA Common Air Interface – New Technology Standards Project – Digital Radio Technical Standards, 1998.
- [6] Internet Engineering Task Force, www.ietf.org/rfc.html
- [7] Lennox, J. et Schulzrinne, H., "A protocol for reliable decentralized conferencing", NOSSDAV'03 : Proceedings of the 13th International Workshop on Networks and Operating Systems Support for Digital Audio and Video, New York, NY, United State of America, 72-81, ACM Press.
- [8] ITU-T G.729, "G.729 : Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear prediction (CS-ACELP)", 2007.
- [9] H. Wang, F. Azuaje, B. Jung, N. Black, "A markup language for electrocardiogram data acquisition and analysis (ecgML)," *BMC Medical Informatics and Decision Making*, May 2003.
- [10] Friedlander, J., Loganathan, K., Murphy, R., Pattabhiraman, R.V. and Vemuri, K.V., "Are you there? Reflections on presence server architectures", *Bell Labs Technical Journal*, 10(2006), 77-82.
- [11] Rescola E. *SSL and TLS Designing and Building Secure Systems*. 2000; 449p.
- [12] C. Peikari, A. Chuvakin. *Security Warrior*. 2004
- [13] CNSS Policy No.15. National Policy on the Use of the Advance Encryption Standard (AES) to Protect National Security Systems and National Security Information. June 2003.
- [14] RSA Laboratories, "PKCS #5: Password-Based Cryptography Standard", www.rsa.com/rsalabs/node.asp?id=2127