Simulation and Formal Verification of SIP/ZRTP Protocol Using UPPAAL

**Aishwarya Raghavan, P.P. Amritha,
M. Sethumadhavan**

Abstract--- The security of voice communication over the Internet Protocol is a continuously growing research area due to the rapid rise in its usage among consumers. With the advent of Voice-over-IP Protocols, the Real Time Protocol (RTP) was used to facilitate VoIP communications. To secure this communication, Secure Real Time Protocol (SRTP) was implemented to encrypt these voice packets. The SRTP requires a session key to be shared between the communicating entities. The challenging task of establishing a new, unused session key to secure each SRTP session was overcome by the key agreement protocol, Zimmermann Real-time Transport Protocol (ZRTP) which ensures confidentiality as well as a shield against Man-in-the-Middle attack. We firstly analyze the security properties of this protocol. Formal analysis is a mathematical technique that can be used to verify the correctness of the system. We simulated the complete ZRTP Protocol with the well-known formal analysis tool, Uppaal, and verified the existing security properties such as Deadlock Prevention, Liveliness, Safety and other protocol parameters mismatch detection using the Uppaal model checker engine. Temporal logic was used to design the queries to verify the properties.

Index Terms--- Temporal Logic, Timed Automata, UPPAAL, ZRTP.

I. INTRODUCTION

In today's world, we try to ensure that any technology that we use comes with the highest level of in-built security. To narrow down our scope, we need to ensure that any communication between two parties in any mode needs to be secured and no third party or attacker can understand or derive information from the communication. Almost everything that we use today comes with in-built internet facility or extensible internet facility. An entity connected to the internet is vulnerable to attackers who can perform sniffing, eavesdropping, Man in-the-Middle and Denial-of-Service (DoS) attacks.

We have reached the extent where even voice calls can be performed flawlessly over the internet. This technology is known as Voice-over-IP (VoIP) that is voice over the internet protocol. As these communications are over wireless network technologies, they are prone to any wireless attacks [1]. The very first protocol that was designed for VoIP communication was the RTP Protocol that just ensured unprotected wireless voice communication. As this was not secure, this communication was encrypted using a session key and protocol was then termed Secure RTP (SRTP). The freshness of the key generated in SRTP was questionable. If the session key generated for the SRTP was a repeated one, an attacker with access to the previous keys can brute force or reuse the keys to decrypt the communication. Thus, Philip Zimmermann came up with the Zimmermann Real-time Transport Protocol (ZRTP) which ensured a brand new session key for each SRTP session.

To ensure the correctness of the existing security properties in the ZRTP Protocol, we need a mathematically proven procedure to show proof of concept that the protocol has the security that it claims to provide. We chose to formally analyze this protocol with the help of timed automata and transactions in the Uppaal Model Simulation. Uppaal simulation itself performs basic sanity checks and helps us find if any transaction trace leads to a deadlock situation where the system's availability is put to test [2]. Once the simulation is designed, we use Uppaal's verifier engine that can check the states and their transactions and signal whether the model satisfies the properties or not. We formed the properties with the help of temporal logic to design the queries.

II. SECURITY ANALYSIS

The ZRTP Protocol is considered to be secure due to features like Man-in-the-middle attack prevention using the SAS Authentication. In [2] and [3], we see ZRTP Protocol analysis is done with AVISPA and ProVerif to check for Man-in-the-middle protection property. The protocol also rejects fake ZRTP messages and prevents DoS attacks. In [4], π-calculus and ProVerif have been used to verify the protocol's efficiency.

The session keys generated ensure that the message confidentiality and trust is maintained. Some researches as in [5], use SPIN to analyze the protocol. Even if a session's key is compromised, the protocol ensures forward secrecy where the keys of other/previous communications are unknown and not revealed to the attacker. The space complexity of the protocol is very efficient as it does not require any complex computations or big storage capabilities as it requires a feasible cache space. Any voice forgery/impersonation attack is curbed with the use of voice verification during SAS Authentication as seen in [6]. The communication in ZRTP is successful only when the cache which stores the session key on both the communication parties' side is empty. This ensures that no spoofing happens. To facilitate security features the protocol does not require any public key infrastructure or pre-shared secret [7].

Manuscript received September 16, 2019.

Aishwarya Raghavan, TIFAC-Core in Cyber Security, Amrita School of Engineering, Coimbatore, T.N, India.

P.P, Amritha, TIFAC-Core in Cyber Security, Amrita School of Engineering, Coimbatore, T.N, India.

M. Sethumadhavan, TIFAC-Core in Cyber Security, Amrita School of Engineering, Coimbatore, T.N, India.

III. ZRTP PROTOCOL

ZRTP Protocol is one of the best known protocols for encrypted voice communication over the internet. Fig. 1 shows the establishment of the communication and further voice data packet transfer. The complete protocol can be divided into three phases, namely, SIP Handshake, ZRTP Key Agreement protocol and the SRTP communication [8]. The caller is the entity that initiates the communication and the callee is the entity that the caller wants to communicate with. The SIP Server facilitates the initial SIP Handshake. In our scenario, we consider that the ZRTP Protocol is preceded with the SIP Telephonic Handshake. Each phase has been explained below.

A. SIP Handshake

The caller sends an invite to the SIP Server, which is then forwarded to the callee. Caller and Callee try to connect. Once connection is established, the ringing happens. Each of them have now joined the call and exchange an OK message, and a final ACK to confirm the connection establishment.

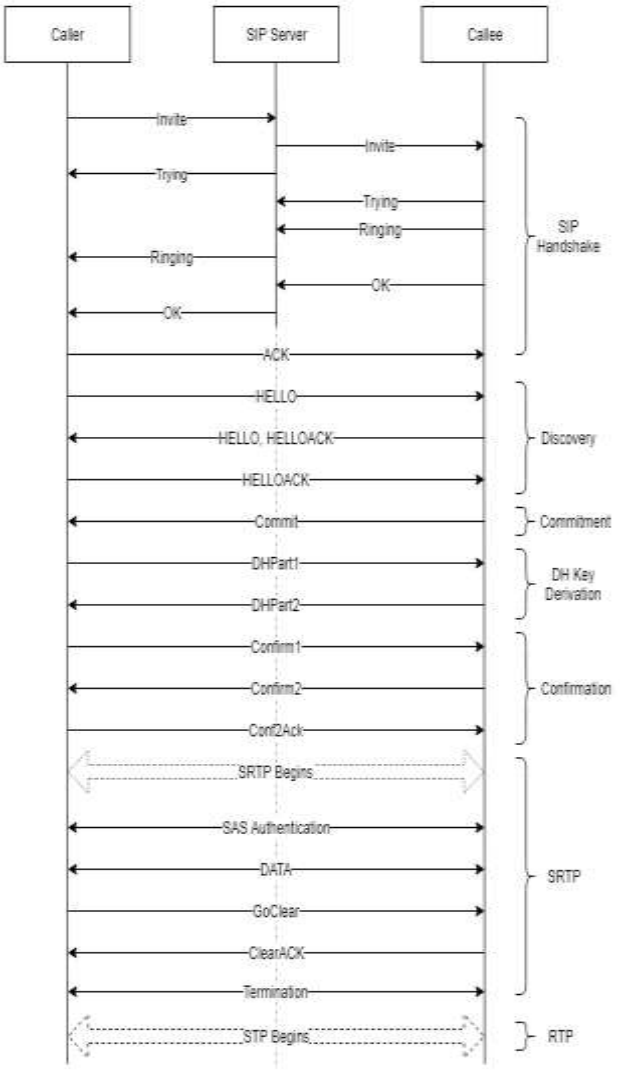

Fig. 1: The SIP/ZRTP Protocol

B. ZRTP Key Agreement Protocol

This phase is further divided into four phases, namely, Discovery Phase, Commitment Phase, Diffie Hellman Key Derivation Phase and then the Confirmation Phase. In the Discovery Phase, the caller and the callee exchange their ZRTP identifiers required for the successful connection. Each of them acknowledges the receipt of the other's identifiers. The Commitment Phase finalizes the identifiers that will be used by both the parties for the rest of the communication. In the Diffie Hellman Key Derivation Phase, each entity shares their ephemeral part of the key with the other, and they derive the session key for the SRTP communication. Finally, the Confirmation Phase will decide the life span of this session key that will be used by the SRTP communication.

C. SRTP Communication

The session key generated above is always fresh and can now be used to encrypt voice messages for the SRTP communication. Just after the SRTP is established, both parties have to verbally exchange a string, known as the SAS (Short Authentication String), which defends against Man-in-the-middle attack. One the SAS has been verified, the parties communicated by encrypting their messages using the session key. If any party wants to terminate the communication, it issues a GoClear command and the SRTP Session changes to a RTP Session after the ClearAck is issued by the other party. This is the complete working of the SIP/ZRTP Protocol.

IV. FORMAL SIMULATIOM

We have seen that the ZRTP Protocol has multiple states and multiple outcomes depending on the scenario that is encountered. We need a mechanism to model all these possible outcomes in one place, analyze their behavior and verify the security properties. For this purpose, we will be formally analyzing the model using the Uppaal tool. This tool can be used for formal analysis and verification of Cyber Physical and Real Time Systems by using timed automata and temporal logic [2]. The implementation of this tool has been done in Java.

The simulation can be performed by using state transitions and these state transitions can be enabled using receiver and sender commands. We can maintain state names and state variables in this part of the tool. The model is designed in the Editor and the transition trace of the model is analyzed in the Simulator. The property verification can be done using the Verifier of the tool. We used temporal logic to design the queries that check if our requirements have been satisfied by the model.

To understand the models, we need to be aware of the notations used.

1) A transition containing a '!' initiates another transition.
2) A transition containing a '?' waits for a transaction initiation
3) Each template consists of a single Start state represented by double circle.

4) Single circle states represent any intermediate state
5) States with c in the circle indicate a commit state which has only one outgoing transaction

The SIP/ZRTP Protocols has three components, namely, the Caller, the SIPServer and the Callee. Each of these components forms a template in the Uppaal model. The three templates can be seen in Fig. 2, Fig. 3 and Fig. 4.

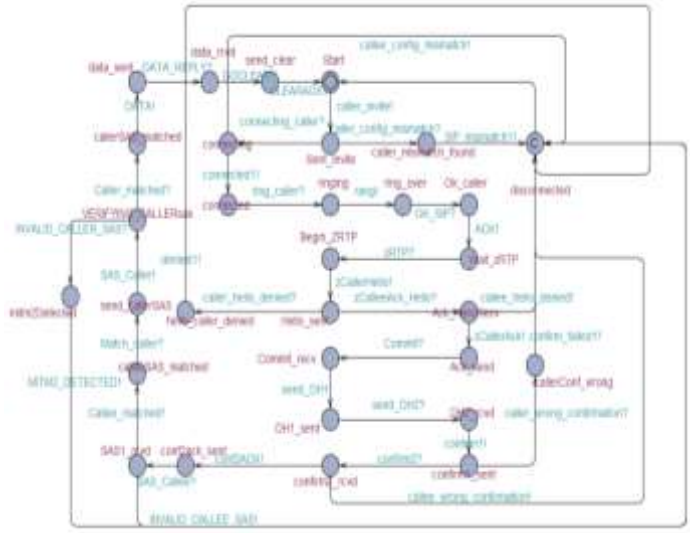

Fig. 2: Caller's Automaton

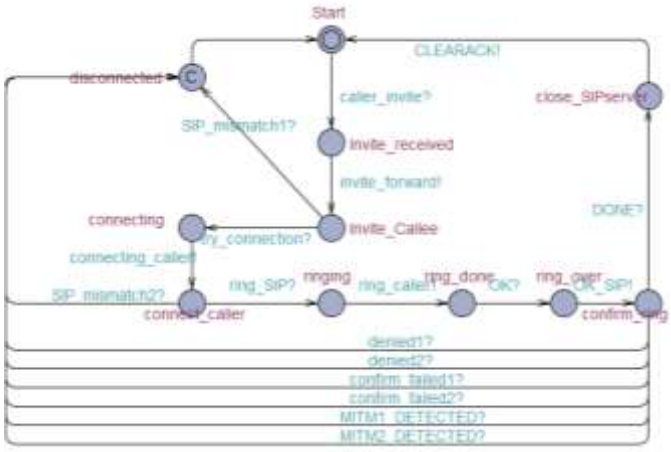

Fig. 3: SIP Server's Automaton

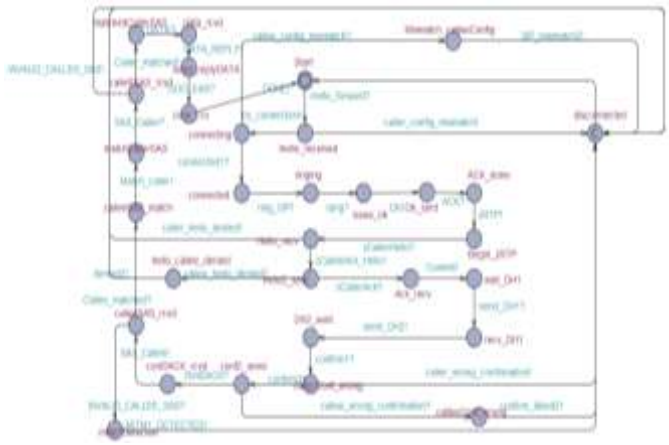

Fig. 4: Callee's Automaton

The detailed transaction trace(s) is explained below:

1) A caller invites the callee via the SIPServer using the *caller_invite!* Command.
2) The callee can reject the invite by issuing *caller_config_mismatch!* If it does not match the caller's requirements or the callee cannot connect with the caller at the moment. If no such scenario, the callee initiate connection with the caller via the SIPServer using *try_connection!* Command. Similarly, caller can disconnect with the callee if any requirements mismatch by issuing *callee_config_mismatch!*.
3) If the invite has been accepted on both the ends, it is followed by the ringing (*ring_caller!*) and ok (*OK!*) commands. The caller, then, sends the *ACK!* To acknowledge the completion of the SIP handshake.
4) Now, the ZRTP Phase begins and from here onward, there will be direct communication between the caller and callee, with no SIPServer in the middle.
5) Caller and Callee will verify and acknowledge each other's Hello through the *zCallerHello!* And *zCalleeAck_Hello!* Commands. The Hello commands contain that identifiers that need to be verified before the ZRTP Key is derived. If any mismatch in the identifiers such as in *caller_hello_denied!* And *callee_hello_denied!*, the communication will go to the disconnected commit states in the respective templates.
6) If the Hello's have been successfully verified, The *Commit!* Command will finalize the communication settings and identifiers for the secure communication.
7) Both sides now exchange their Diffie-Hellman key part with commands *send_DH1!* And *send_DH2!*
8) After the exchange, the key formation confirmation states may lead to disconnected commit states if there is any mismatch during the Diffie-Hellman key generation, as seen in *callee_wrong_confirmation!* or *caller_wrong_ confirmation!*.
9) On successful generation of Diffie-Hellman key, the *Conf2Ack!* marks the end of the ZRTP Key agreement phase and the SRTP Phase begins.
10) Both sides will now match their SAS Authentication strings verbally by issuing *SAS_Callee!* and *SAS_Caller!* to ensure no Man-in-the-middle. If detected, *MITM2_DETECTED!* and *MITM2_DETECTED!* will lead to disconnected state.
11) If the SAS authentication Phase is successful, the parties can now have a secure voice communication *DATA!* and *DATA_REPLY!*.
12) To terminate the session, we can use the *GOCLEAR!* Command. The session is closed only when the other party responds with a *CLEARACK!*. This closes all the channels created for this communication. Any of the two parties can choose to close the communication.
13) Hence, the above explanation shows that all scenarios have been considered and modeled so that the system can handle any scenario.

V. RESULTS

The above simulated model is verified in the Uppaal verifier using temporal logic. Each condition we want to verify should be written using temporal logic notations such as 'A[]' expression, whose condition all transaction must follow and 'E<>' expression, whose condition is followed by at least one transaction trace. We verified the below mentioned seven properties of the system.

A. Deadlock

This query verifies that no transition trace leads to a deadlock situation in the model and any situation lands in a final successful/unsuccessful state.

A[] not deadlock

B. Safety

Safety verifies that "something bad never happens" and in our model we verify that only when the caller initiates a call request, the callee responds with a connection acceptance. Upon caller's 'Sent_Invite' state, the SIPServer will move to the 'Invite_Callee' state. Then the callee tries to connect in state 'connecting' and they both exchange initial Hellos in states 'Hello_sent' and 'Hello2_sent'.

E<> (Caller.Sent_Invite imply SIPServer.Invite_Callee) and (Caller.Sent_Invite imply Callee.connecting) and (Caller.Hello_sent imply Callee.Hello2_sent)

C. Liveliness

Liveliness verifies that "something good eventually happens" and in our model we verify that on successful caller communication completion, the SIPServer and the callee terminate. Upon caller's 'send_clear', callee closes at state 'clear_rcv' and SIPServer at 'close_SIPserver'.

E<> (Caller.send_clear imply Callee.clear_rcv) and (Caller.send_clear imply SIPServer.close_SIPserver)

D. Invitation Rejection

This property verifies that if the caller's configurations (as in the first scenario) or the callee's configurations (as in the second scenario) are not suitable as in states 'caller_mismatch_found' and 'Mismatch_calleeConfig' respectively during the initiation of SIP handshake, the model leads to 'disconnected' states in the caller, callee and SIP Server.

E<> (Caller.caller_mismatch_found imply Caller.disconnected) and (Caller.caller_mismatch_found imply SIPServer.disconnected) and (Caller.caller_mismatch_found imply Callee.disconnected)

E<> (Callee.Mismatch_calleeConfig imply Caller.disconnected) and (Callee.Mismatch_calleeConfig imply SIPServer.disconnected) and (Callee.Mismatch_calleeConfig imply Callee.disconnected)

E. Discovery Hello Mismatch

This property verifies that if the caller's Hello parameters (as in the first scenario) or the callee's Hello parameters (as in the second scenario) are not compatible as in states 'hello_caller_denied' and 'hello_callee_denied' respectively during the ZRTP Discovery Phase, the model leads to 'disconnected' states in the caller, callee and SIP Server.

E<> (Caller.hello_caller_denied imply Caller.disconnected) and (Caller.hello_caller_denied imply SIPServer.disconnected) and (Caller.hello_caller_denied imply Callee.disconnected)

E<> (Callee.hello_callee_denied imply Caller.disconnected) and (Callee.hello_callee_denied imply SIPServer.disconnected) and (Callee.hello_callee_denied imply Callee.disconnected)

F. Invalid DH Key Generation

This property verifies that if the caller (as in the first scenario) or the callee (as in the second scenario) is not able to generate the DH Key as in states 'callerConf_wrong' and 'calleeConf_wrong' respectively during the ZRTP DH Phase, the model leads to 'disconnected' states in the caller, callee and SIP Server.

E<> (Caller.callerConf_wrong imply Caller.disconnected) and (Caller.callerConf_wrong imply SIPServer.disconnected) and (Caller.callerConf_wrong imply Callee.disconnected)

E<> (Callee.calleeConf_wrong imply Caller.disconnected) and (Callee.calleeConf_wrong imply SIPServer.disconnected) and (Callee.calleeConf_wrong imply Callee.disconnected)

G. Man-in-the-Middle Detection

This property verifies that if the caller (as in the first scenario) or the callee (as in the second scenario) is not able to show the same SAS Authentication string as in states 'mitm1Detected' and 'mitm2Detected' respectively during the SRTP Phase, the model leads to 'disconnected' states in the caller, callee and SIP Server.

E<> (Callee.mitm1Detected imply Caller.disconnected) and (Callee.mitm1Detected imply SIPServer.disconnected) and (Callee.mitm1Detected imply Callee.disconnected)

E<> (Caller.mitm2Detected imply Caller.disconnected) and (Caller.mitm2Detected imply SIPServer.disconnected) and (Caller.mitm2Detected imply Callee.disconnected)

VI. CONCLUSION

The existing security of the SIP/ZRTP Protocol shows that with the increasing use of VoIP over traditional wired telephonic communication, VoIP is preferred by users due to the ease of use, low cost and high security features.

As ZRTP Protocol provides security against information disclosure, Man-in-the-Middle and Denial-of-Service, by constantly securing this protocol, we can achieve good VoIP security. Formal verification of these security features assure that the VoIP communication provides end-to-end security with no requirement for pre-shared secret. Having said that, there are multiple possibilities that can occur as zero-day attacks.

Hence we need to explore on possible attacks on the existing VoIP Protocol, harden the protocol to stay immune to those attacks and then formally verify it's correctness to provide maximum security to users on the internet.

REFERENCES

1. P. Gupta and V. Shmatikov, "Security Analysis of Voice-overIP Protocols," 20th *IEEE Computer Security Foundations Symposium (CSF'07)*, Venice, 2007, pp. 49-63.
2. Jayasri K.S., Jevitha K.P., Jayaraman B. (2018) Verification of OAuth 2.0 Using UPPAAL. In: Mandal J., Sinha D. (eds) Social Transformation – Digital Way. CSI 2018. *Communications in Computer and Information Science*, vol 836. Pp. 58-67. Springer, Singapore
3. R. Bresciani, S. Superiore, S. Anna, and I. Pisa. The ZRTP protocol security considerations. Technical Report LSV-07-20, 2007
4. R. Bresciani and A. Butterfield, "A formal security proof for the ZRTP Protocol," 2009 *International Conference for Internet Technology and Secured Transactions, (ICITST)*, London, 2009, pp. 1-6.
5. X. Chang, Y. Qin, Z. Chen and B. Xing, "ZRTP-based Trusted Transmission of VoIP Traffic and Formal Verification," 2012 *Fourth International Conference on Multimedia Information Networking and Security, Nanjing*, 2012, pp. 560-563.
6. Petraschek, M. and Höher, Thomas and Jung, O. and Hlavacs, Helmut and Gansterer, Wilfried, Security and Usability Aspects of Man-in-the-Middle Attacks on ZRTP, *J. UCS Journal of Universal Computer Science*, 14 (5). pp. 673-692 Springer Verlag (2008)
7. O. Jung, M. Petraschek, T. Hoeher and I. Gojmerac, "Using SIP identity to prevent man-in-the-middle attacks on ZRTP," 2008 *1st IFIP Wireless Days, Dubai*, 2008, pp. 1-5.
8. Zimmermann, P., Johnston, A., Ed., and J. Callas, "ZRTP: Media Path Key Agreement for Unicast Secure RTP", *RFC* 6189, April 2011, https://www.rfceditor.org/info/rfc6189

AUTHORS PROFILE

Aishwarya Raghavan Currently pursuing final year Masters' Degree at Amrita School of Engineering, Coimbatore, India. Interested research areas include Zigbee Security and VOIP Security.

Amritha P. P Currently serves as Assistant Professor at TIFAC-CORE in Cyber Security, Coimbatore Campus. Research areas include Stegnography, Steganalysis, Information Hiding, Secret sharing and Cryptography.

M. Sethumadhavan Head of TIFAC-Centre of Relevance and Excellence in Cyber Security, an R&D centre at Amrita Vishwa Vidyapeetham Coimbatore campus since its inception in the year 2005. He leads the R&D that focuses on the areas of cyber security such as Cryptology, Post Quantum Cryptography, Steganalysis, Secure Coding, Computer Network Security, Digital Forensics etc. A central focus of his work has been to create innovative educational and research programs and develop world-class expertise in Cyber Security as the underlying vision.