Auditable messages with hash chain in instant messaging apps

Andrea E. Komo[1], **Marcos A. Simplicio Jr.**[1]

[1] Escola Politécnica – Universidade de São Paulo (USP)

{aerina, mjunior}@larc.usp.br

Abstract. *Instant messaging applications have been used as corporate tools, so the messages exchanged in these systems have been used as negotiation records. However, by design, most of such apps do not provide any verification feature to confirm the integrity of the conversations. Analyses show that it is possible to surreptitiously modify records in popular apps like WhatsApp and Telegram. Aiming this issue, this work proposes a message structure based on hash chain, to ensure the integrity and the possibility to audit conversations. Besides, we propose a design with selective disclosure to improve privacy during audits, and this solution is architecture-independent, so it can be integrated with any instant message app.*

1. Introduction

In this document, we present a summary of the work **"An efficient method to provide auditable messages exchanged in instant messaging applications" [Komo 2023]**, the Master of Science dissertation of Andrea Erina Komo defended and approved on November 29, 2022. This work was supervised by Prof. Dr. Marcos A. Simplicio Jr., and it was supported by Ripple's University Blockchain Research Initiative in partnership with the Universidade de São Paulo (USP).

1.1. Problem and Motivation

Instant messaging (IM) apps have become official work tools, therefore new security requirements have emerged creating a gap in the literature. Among the apps available on the market, security is mainly concentrated on the messages' confidentiality, focusing on end-to-end encryption [Cohn-Gordon et al. 2017]. In contrast, integrity, authenticity, and non-repudiation of the conversations have not received as much attention, which often hinders the verification of the truthfulness of the messages exchanged [Schliep et al. 2017, Schliep and Hopper 2018]. Not all applications ensure that messages have not been manufactured or modified and that the conversation has not undergone any changes in content or order. In a hypothetical situation, if the messages are stored in a central database and an attacker modifies the records, then false evidence can be created. After all, the messages exchanged often have the value of a receipt for the interlocutors.

Table 1 presents an example comparing the original chat versus the chat with the messages' order changed. As we can observe, the conversation's meaning is very different in both situations. In the original chat, the person $\mathcal{B}$ has a healthy life, and in the modified conversation it's the opposite.

So, the problem identified in this work is the lack of reliable mechanisms to verify the integrity of conversations in IM apps. And the motivation for creating this feature is the need to check whether a conversation is integrated or not in dispute situations, especially in corporate conversations.

Authentic chat	**Modified chat (change order)**
$\mathcal{A}$: Do you go to the gym?	$\mathcal{A}$: Do you go to the gym?
$\mathcal{B}$: Twice a week.	$\mathcal{B}$: No.
$\mathcal{A}$: Do you smoke?	$\mathcal{A}$: Do you smoke?
$\mathcal{B}$: No.	$\mathcal{B}$: Twice a week.

Table 1. Example of conversation with modified message sequence. [Komo 2023]

1.2. Goals and Contributions

Aiming to improve communications security in IM apps, this work proposes an efficient and secure communication architecture that ensures integrity, authenticity, and non-repudiation of conversations. More precisely, the scheme intends to protect each message and guarantee exchanged messages' sequences, making reliable audits possible in IM apps. The goal is to confirm one of two situations at auditing:
- The conversation presented is authentic, so that the sequence of the messages is exactly how the conversation took place; or
- The conversation presented has undergone some kind of change, proving that this record is fake and unreliable.

In this work, we do not pretend to identify which type of modification happened in the conversation, nor discover the original char from the modified chat presented.

The solution is based on blockchain technology and uses signed hash chains, similar to systems like Git [Torvalds 2005], Bitcoin [Nakamoto 2008], and SecureTCG [Simplicio et al. 2014]. With this mechanism, it is possible to create a reliable and verifiable record in IM apps. The work's principal contribution is this chat integrity verification feature, however, the proposed complete architecture was also concerned with other characteristics such as conversation privacy, solution efficiency in mobile systems and app communication. The complete architecture is detailed in Section 3.

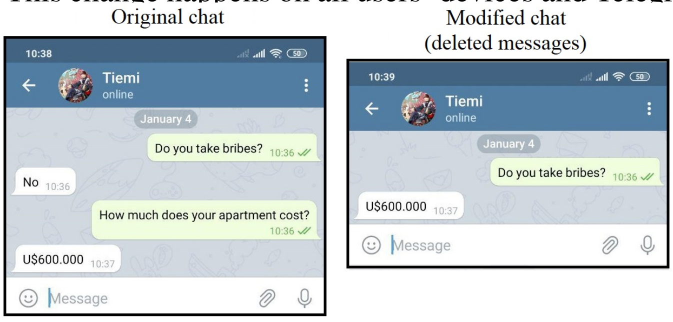
1.3. Outline

The rest of this article is organized as follows. In Section 2, we comment on the existing IM apps as related works, showing why they do not fulfill this work goals. Section 3 explains our proposed architecture. Section 4 presents the prototypes used as proofs of concept and the results obtained from them, and also lists the publications of this research. And Section 5 concludes this summary by highlighting the main points of the work.

2. Related Works

Security protocols currently used for IM apps to guarantee exchanged messages' integrity and authenticity do not involve digital signatures. They often use schemes based on symmetric keys [Cohn-Gordon et al. 2017], as Messages Authentication Codes (MACs) and Authenticated Encryption (AE) [Simplicio et al. 2013]. This is the case with popular apps such as WhatsApp, Telegram [Telegram 2019], Signal, and Threema [Rösler et al. 2018]. Due to the characteristics of these symmetric schemes, the access to received messages also allows their manipulation, enabling the repudiation of messages sent or received. To manipulate local contents, it would be enough: access the memory space where the messages are stored, manipulate the message, and recalculate the authentication data set of the modified message, using the same key used to verify the original content. In this case, it is possible to identify discrepancies between the contents stored on the devices of the users participating in the communication, however, it would not be possible to assess who was responsible for the manipulation.

At apps with a central server, in principle, would be possible to verify the occurrence of local data manipulations when comparing them with the data stored on the server. However, this is not always possible in practice, depending on how the application is designed. One example, in particular, is the case of *Telegram*, which uses just client-server encryption in conversations by default (i.e., when the "secret chat" feature is not enabled). Figure 1 shows an example of how a conversation could be manipulated at Telegram, simply using the message deletion feature, resulting in a chat with a very different meaning from the original. This change happens on all users' devices and Telegram's server, if you select the option '



Do you take bribes? 10:36
No 10:36
How much does your apartment cost? 10:36
U$600.000 10:37

Do you take bribes? 10:36
U$600.000 10:37

Figure 1. Conversation with deleted messages. Source: Authors by *Telegram*.

Considering apps with end-to-end encryption (E2EE), it will be even more difficult to detect the chat modification, because it focuses on confidentiality of the messages exchanged. Although this type of mechanism is important for the preservation of users' privacy, it compromises the requirements of a corporate scenario where it is wanted to have conversations recorded as reliable documents for later consultations. After all, given E2EE, the system does not allow central servers to access the contents of messages, so there is no tamper-proof environment from which the original messages can be retrieved.

Some instant messaging applications with E2E security even avoid having a central server, using peer-to-peer technologies so that there are no records of communication metadata (e.g., instants of message sending) [Komo et al. 2018, Rogers et al. 2018, Gultsch 2014]. This increases the independence and privacy of the system, but also makes it difficult to prove the content of conversations at a strategic and neutral point.

To the best of our knowledge, one of the few works in the literature that takes message integrity requirements into account after receiving them in instant messaging applications is the protocol proposed in [Schliep and Hopper 2018]. The integrity of the conversation is guaranteed by the combination of the NAXOS key agreement protocol [LaMacchia et al. 2007] and symmetric encryption protocol authenticated with associated data, the AES-GCM with random initialization vectors [A. Mcgrew and Viega 2004]. An ephemeral NAXOS key is generated for each message based on the previous message and then that key is used to encrypt the current message. However, the solution also seeks to provide plausible deniability, i.e., by design it does not provide the non-repudiation property, leaving an unsatisfactory gap for its use in corporate settings.

3. Proposed Solution

Aiming to ensure auditability in instant messaging systems, this work proposes to use chained and signed cryptographic hashes [Hu et al. 2005]. Considering a sequence with

$\mathcal{N}$ registered messages, to insert a new message $\mathcal{N}+1$ it is necessary that the new record presents the new message and the cryptographic hash of the previous record $\mathcal{N}$. The last information is then signed, allowing any manipulation to be detectable.

In addition, instead of saving the message's plain text in the blocks, we propose to use a pseudorandom function (PRF) to save messages' identifiers $\mathcal{X}$, for example, using a hash function and saving the message's hash in the blocks, as illustrated in Figure 2. This allows a **selective disclosure** of messages, improving system **privacy**, so that only the messages to be audited are exposed and not the entire conversation.

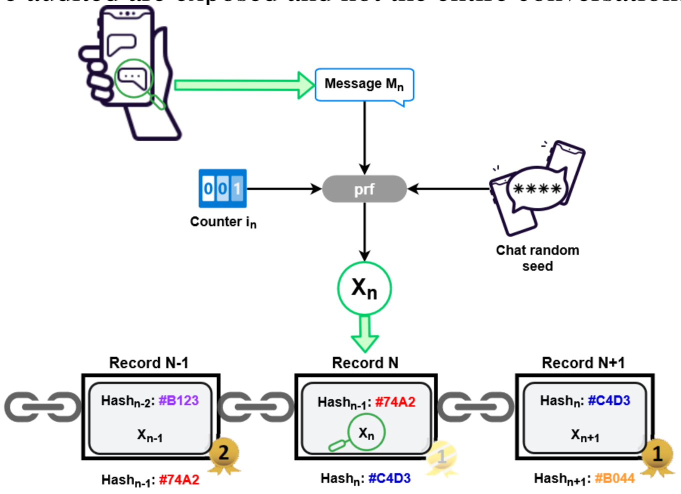


Figure 2. Hash chain structure for messages. Assuming that user 1 sent messages $\mathcal{N}$ and $\mathcal{N}+1$, only his last signature needs to be stored. [Komo 2023]

More formally, we have the following logic for creating the chain of messages of a conversation (note that each $block_n$ is digitally signed):

1. $block_0 = (\emptyset, X_1)$, where $\emptyset$ is a string of bits filled with zeros, indicating the beginning of the communication, and $X_1 = prf(M_1|seed|i_1)$ when M_1 is the conversation's first message, $seed$ is the Diffie-Hellman key exchanged in the chat, and i is the counter of messages in this chat;
2. $block_n = (h_{n-1}, X_n)$, for $n \geq 1$, where $h_{n-1} = Hash(block_{n-1})$ and $X_n = prf(M_n|seed|i_n)$ when M_n is the conversation's n-th message.

Notice that we propose to use $\emptyset$ zeros values over an Initialization Vector (IV) in the first block. A reason for this choice is that we do not need a security number in this part to ensure the security of the system, so the $\emptyset$ value is enough and demands fewer computer resources than an IV. Another reason is that with the $\emptyset$ we can be sure that it is the start of the chain, preventing improper messages to be added before the first register.

Due to the properties of cryptographic hashing algorithms, any change in the input data leads to changes in the algorithm's output hash. In this way, a sequential link is created that ensures the integrity of each individual record and also the order of the records. At the same time, if the last signature in the chain is valid, this is sufficient to ensure the complete conversation's non-repudiation, so that previous signatures of the same user can be discarded, optimizing memory use. Signatures of other users, who sent previous messages, must be preserved to ensure non-repudiation up to that point of the communication. By using such structure for exchanging messages between users, it allows reliable audits of conversations by anyone who has access to a sequence of exchanged messages, just reconstructing the chain and checking each block's hashes and final signatures.

If the intention is to reveal only a portion of messages, omitting previous and subsequent blocks, then the integrity check of the revealed content is still possible as long as the digital signature for the last block is available. In this case, only the value $\mathcal{X}$ of the omitted messages would be available for analysis, which would not allow the recovery of the message itself except for possible brute-force attacks.

3.1. Attack resistance analysis

Below we have some theorems showing how the proposed architecture is robust against some attacks, meeting the desired characteristics of reliable records, assuming that at least one of the interlocutors is honest.

Theorem 1 (Discover chat content)**.** *Let prf be a pseudo-random function, M_n is the n-th message of a chat, $cont_n$ is the message counter, $seed_c$ is the chat c unique seed number, and $X_n = prf(M_n|cont_n|seed_c)$. Given X_n, it should be computationally infeasible to find the message M_n for X_n under prf.*

Proof (sketch). Suppose that prf is a cryptographic hash function. Consider cryptographic hash function requirements, specifically, first preimage resistance, then by construction is computationally infeasible to find a first preimage M_n for X_n. □

Theorem 2 (Editing message content)**.** *Let h be a cryptographic hash function, H_n is the hash generated over n-th message of a chat so that $H_n = h(H_{n-1}|X_n)$. Given M_n, it should be computationally infeasible to find another message M'_n such that $H'_n = h(H_{n-1}|X'_n)$, $X'_n = prf(M'_n|cont_n|seed_c)$ and $H'_n = H_n$.*

Proof (sketch). The proof builds upon the hash function's second preimage resistance. Given M_n, it should be computationally infeasible to find a second preimage M'_n for M_n under h. □

Theorem 3 (Changing the order of messages)**.** *Let h be a cryptographic hash function, H_n is the hash generated over n-th message of a chat so that $H_n = h(H_{n-1}|X_n)$. Given the hash chain H_{n-1}, H_n, H_{n+1}, and H_{n+2} related, respectively, to the messages' sequence M_{n-1}, M_n, M_{n+1}, and M_{n+2}. If M_n and M_{n+1} switch positions, then this change is noticed in the hash chain.*

Proof (sketch). The proof builds upon the hash chain construction and hash function collision resistance. The new messages' sequence will generate new hashes values different from those saved in the hash chain blocks. Because of hash function collision resistance, with the exchange, it should be computationally expected that H_n is different from H_{n+1} used to generate H_{n+2}, and H_{n+1} be different from H_{n-1} used to generate H_n, and H_{n-1} be different from H_n used to generate H_{n+1}. When checking the hash chain, these 3 points of divergence will be detected against the saved record blocks. □

Theorem 4 (Message removal)**.** *Let h be a cryptographic hash function, H_n is the hash generated over n-th message of a chat so that $H_n = h(H_{n-1}|X_n)$. Given the hash chain H_{n-1}, H_n, and H_{n+1} related, respectively, to the messages' sequence M_{n-1}, M_n, and M_{n+1}. If the record related to M_n is deleted, this change is noticed in the hash chain.*

Proof (sketch). The proof builds upon the hash chain construction and the underlying hash function's collision and second pre-image resistance. Specifically, the deletion of M_n would only go undetected if H_{n-1} is identical to H_n, or if record M_{n-1} is modified so its hash matches H_n. The first condition only happens with negligible probability, whereas the second is computationally infeasible. □

Theorem 5 (Message insertion)**.** *Let h be a cryptographic hash function, H_n is the hash generated over n-th message of a chat so that $H_n = h(H_{n-1}|X_n)$. Given the hash chain H_{n-1}, H_n, and H_{n+1} related, respectively, to the messages' sequence M_{n-1}, M_n, and M_{n+1}. If a new record related to $M_{n.5}$ is inserted in the middle of the chain between M_n and M_{n+1}, then this change will be noticed in the hash chain.*

Proof (sketch). This theorem can be proof considering hash chain structure construction and hash function collision resistance and second preimage resistance. Because of hash function collision resistance, with the insertion, it should be computationally improbable that $H_{n.5}$ be the same value as H_n used to generate H_{n+1}. Furthermore, because hash function second preimage Resistance, given message M_n, it should be computationally infeasible to find a second preimage $M_{n.5}$ for M_n under h. Then considering $H_{n.5} \neq H_n$, when checking the hash chain, it will use $H_{n.5}$ to generate H_{n+1} and these point of divergence will be detected. □

It is relevant to note that, in all theorems, the architecture is robust for identifying breaches of conversation integrity, although it cannot (and is not its purpose) reverse the detected changes. Besides, what ensures that the altered messages' hashes can not replace the original hashes is the fact that the message at the end of the chain is always digitally signed by the message sender. At the same time, this guarantees authenticity and non-repudiation in communications.

4. Results

Following the architecture proposed were developed two proof of concept (PoC):

- **Android app PoC**: Available in the following GitHub repository `https://github.com/Erina-chan/app_eri-chain`. The focus of this PoC is to prove the efficiency of the solution in Android systems. To the benchmark was used a Samsung Galaxy A22 with Android version 12, 4GB RAM memory, and 2 GHz Octa-Core. The test performed used a message with 100 characters. If we consider the full processes to send and receive a message in the proposed architecture, the sending time is the sum of algorithms' times (ECDH exchange + 3 x SHA3-256 + PGP encryption + ECDSA sign) and the receiving time is the sum of algorithms' times (ECDH exchange + 3 x SHA3-256 + ECDSA verify + PGP decryption). If we add the respective average times measured at Samsung Galaxy A22, we have about 5.756ms to the sending and 26.450ms to the receiving, generating a total time of 32.206ms. In the literature, [Keates 2016] says that the better response time is $\approx$250ms, and [Funk et al. 2020] considers that the most accepted time range is between 0s and 2s. Seeing out benchmark times, we conclude that the architecture proves to be efficient for the standards in the literature.
- **Python chat PoC**: Available in the following GitHub repository `https://github.com/Erina-chan/hashchat`. The focus of this PoC is to prove the auditability of the conversation with the proposed architecture. This PoC generates a chat and this respective hash chain that can be exported and audited in another program that verifies the chain's construction. The results meet the project goal to prove if the conversation is authentic or fake. And these tests also proved that the audit can be performed in an external and independent system making the verification much more fair and reliable.

4.1. Publications

This work produced the following publications:

- Conference Paper: KOMO, A. E.; ARAKAKI, B. O.; SIMPLICIO JR., M. A.; LEVY, M. R. Aplicativo de troca de mensagens instantâneas utilizando comunicação P2P. In: Anais Estendidos do XVIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais. Porto Alegre, RS, Brasil: SBC, 2018. p.65–72. Available in: `https://sol.sbc.org.br/index.php/sbseg_estendido/article/view/4143`
- Conference Paper: KOMO, A. E.; SIMPLICIO JR., M. A. Solução para habilitar conversas integras e auditáveis em aplicativos de troca de mensagens instantâneas. In: Anais do XIX Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais. Porto Alegre, RS,Brasil: SBC, 2019. Available in: `https://sbseg2019.ime.usp.br/anais/196912.pdf`

The paper "Solução para habilitar conversas integras e auditáveis em aplicativos de troca de mensagens instantâneas" was awarded as the best complete article at the XIII Workshop de Trabalhos de Iniciação Científica e de Graduação (WTICG) of the XIX Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg).

5. Conclusion

In this article, we summarised the [Komo 2023] master dissertation work. We show the difficulty to audit a conversation at the instant messaging apps, and we specified the problem this can generate in the corporate scenario where the conversation records are considered business documents.

Our solution was based on blockchain technology, however, we created an efficient architecture using only the hash chain to build conversation records. The proofs of concept tests concluded that the solution is efficient in Android mobile system and meets the initial proposal of the project satisfactorily.

References

[A. Mcgrew and Viega 2004] A. Mcgrew, D. and Viega, J. (2004). The galois/counter mode of operation (GCM).

[Cohn-Gordon et al. 2017] Cohn-Gordon, K., Cremers, C., Dowling, B., Garratt, L., and Stebila, D. (2017). A formal security analysis of the signal messaging protocol. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 451–466.

[Funk et al. 2020] Funk, M., Cunningham, C., Kanver, D., Saikalis, C., and Pansare, R. (2020). Usable and acceptable response delays of conversational agents in automotive user interfaces. In *12th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, AutomotiveUI '20, page 262–269, New York, NY, USA. Association for Computing Machinery. Available at: `https://doi.org/10.1145/3409120.3410651`. Accessed June 2022.

[Gultsch 2014] Gultsch, D. (2014). Conversations. Available at: `https://conversations.im/`. Accessed Aug. 2021.

[Hu et al. 2005] Hu, Y.-C., Jakobsson, M., and Perrig, A. (2005). Efficient constructions for one-way hash chains. In *Proc. of the 3rd Int. Conf. on Applied Cryptography and Network Security (ACNS'05)*, pages 423–441, Berlin, Heidelberg. Springer-Verlag.

[Keates 2016] Keates, S. (2016). Measuring acceptable input: What is "good enough"? page 713–723. Universal Access in the Information Society volume 16. Available at: `https://doi.org/10.1007/s10209-016-0498-4`. Accessed June 2022.

[Komo 2023] Komo, A. E. (2023). *An efficient method to provide auditable messages exchanged in instant messaging applications*. Dissertation (Master of Science) - Corrected version, Universidade de São Paulo. Available at: `https://doi.org/10.11606/D.3.2022.tde-22052023-143703`. Accessed May 2023.

[Komo et al. 2018] Komo, A. E., Arakaki, B. O., Simplicio Jr., M. A., and Levy, M. R. (2018). Aplicativo de troca de mensagens instantâneas utilizando comunicação P2P. In *Anais Estendidos do XVIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, pages 65–72, Porto Alegre, RS, Brasil. SBC. Available at: `https://sol.sbc.org.br/index.php/sbseg_estendido/article/view/4143`. Accessed Jan. 2019.

[Komo and Simplicio Jr. 2019] Komo, A. E. and Simplicio Jr., M. A. (2019). Solução para habilitar conversas integras e auditáveis em aplicativos de troca de mensagens instantâneas. In *Anais do XIX Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, Porto Alegre, RS, Brasil. SBC. Available at: `https://sbseg2019.ime.usp.br/anais/196912.pdf`. Accessed Jan. 2020.

[LaMacchia et al. 2007] LaMacchia, B., Lauter, K., and Mityagin, A. (2007). Stronger security of authenticated key exchange. In Susilo, W., Liu, J. K., and Mu, Y., editors, *Provable Security*, pages 1–16, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Nakamoto 2008] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`.

[Rogers et al. 2018] Rogers, M., Saitta, E., Grote, T., Dehm, J., Erlingsson, E., Tyers, B., and Grigg, J. (2018). Briar [online]. `https://briarproject.org/`.

[Rösler et al. 2018] Rösler, P., Mainka, C., and Schwenk, J. (2018). More is less: On the end-to-end security of group chats in Signal, WhatsApp, and Threema. In *IEEE European Symposium on Security and Privacy (Euro S&P)*, pages 415–429.

[Schliep and Hopper 2018] Schliep, M. and Hopper, N. (2018). End-to-end secure mobile group messaging with conversation integrity and deniability. Cryptology ePrint Archive, Report 2018/1097. `https://eprint.iacr.org/2018/1097`.

[Schliep et al. 2017] Schliep, M., Kariniemi, I., and Hopper, N. (2017). Is Bob sending mixed signals? In *Proc. of the 2017 on Workshop on Privacy in the Electronic Society*, WPES '17, pages 31–40, New York, NY, USA. ACM.

[Simplicio et al. 2013] Simplicio, M., Oliveira, B., Margi, C., Barreto, P., Carvalho, T., and Näslund, M. (2013). Survey and comparison of message authentication solutions on wireless sensor networks. *Ad Hoc Networks*, 11(3):1221–1236.

[Simplicio et al. 2014] Simplicio, M., Santos, M., Leal, R., Gomes, M., and Goya, W. (2014). SecureTCG: a lightweight cheating-detection protocol for P2P multiplayer online trading card games. *Security and Communication Networks*, 7(12):2412–2431.

[Telegram 2019] Telegram (2019). MTProto mobile protocol [online]. `core.telegram.org/mtproto`.

[Torvalds 2005] Torvalds, L. (2005). Git [online]. `http://www.git-scm.com/`.