How to Establish the Root of Trust in Zoom's E2E Protocol? Reliably, Securely and Conveniently Validating the Meeting Security Code via Speech-to-Text Transcription

-- A proposal based on a large body of prior research in secure and usable key distribution/agreement --

Nitesh Saxena (saxena@uab.edu)
University of Alabama at Birmingham

Blum et al. recently published a white paper describing the Zoom's proposed End-to-End Encryption (E2EE) protocol and architecture [1], with a roadmap of work to be done in various phases. Perhaps the most important phase of this protocol is "*Phase I: Client Key Management*," where the authors describe the key management protocol based on which the encryption of the media content (audio/video/text) will be performed. This is a leader/host driven protocol that relies on the (long-term) public key of the leader, whereby the symmetric "meeting key" (mk) is essentially to be encrypted with each participant's own public key authenticated/signed by the leader's public key, and distributed to each participant over the broadcast signaling channel ("bulletin board" in the terminology of [1]). That is, when each participant receives the (signed) ciphertext, it will decrypt it to learn the meeting key and also be sure that the meeting key is indeed generated by the public key of the leader by verifying the signature. All of this means that simply sending the public key of the leader over the signaling channel is not sufficient as an attacker (a "man-in-the-middle" or MITM) can insert its own public key over the insecure signaling channel, thereby compromising the security of the entire protocol and E2EE completely. ***This attack, although active, is extremely easy for the adversary to perform.*** Since the Zoom's server controls the signaling channel, it will be a cake walk for an adversary, who has compromised this server or if the Zoom were under coercion from law enforcement, to change the leader's actual public to the attacker's own public key. Thus, it is extremely important to address this critical vulnerability. ***It is not an option, it is a must have***. To counter this, it is essential to authenticate the public key of the leader ----- this is precisely what we call as the "root of trust" in the proposed E2EE protocol because if this authentication is not done right, you may lose all security. The game will be over.

In this document, we review Zoom's proposal to validate the authenticity of the leader's public key, define some very fundamental and subtle security+usability problems with the Zoom's approach, and then introduce a new solution – foundations of which have already been studied in our recent work – to address many of these problems. We also provide items for future work that needs to be done towards transitioning this new solution into Zoom's E2EE protocol in practice. The proposer and his research team is happy to work with the Zoom's researchers and engineers in making this transition possible. We appreciate the feedback from Zoom.

1. Zoom's Proposal for Leader's Public Key Authenticity Validation

The Zoom's proposal [1] to address the leader public key authentication is very simple (simplicity in Zoom's protocol is actually a key strength, since complicated protocols often struggle to get deployed in real world). They basically suggest creating a "*meeting security code*" (MSC) which is nothing but a second pre-image resistant hash of the leader's public key, encoded into human-readable words. This security code will then be verbally announced by the leader during the call, which each participant must compare with its own local view of the code (i.e., the hash of the participant's view of the leader's public key they received over the signaling channel – this may have been tampered with by the MITM) which will be displayed by the client machine of the participant. If the "announced code" matches with the "displayed code", all will be deemed well, otherwise any participant who sees a mismatch must alert the leader and others over the call to abort the protocol and start afresh (due to the most likely presence of the MITM attacker during the key management protocol). The protocol is depicted in Figure 1.

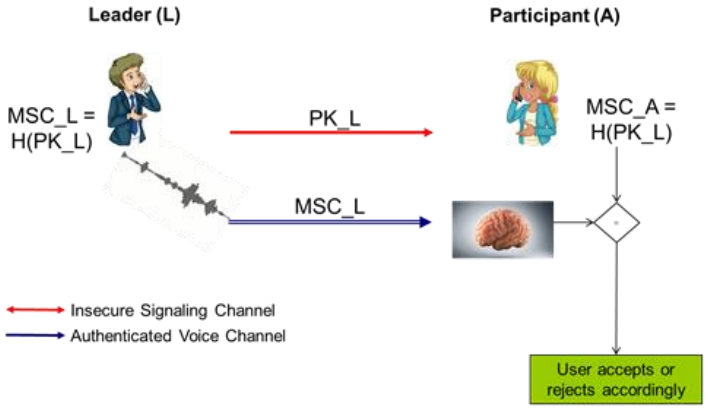

Figure 1: Zoom's leaver public key authenticity validation approach. It is susceptible to skip-through and human errors, greatly reducing the usability and security.

2. Fundamental Problems with Zoom's Meeting Security Code Authenticity Validation

All of the above sounds very great, right? Unfortunately not --- there are several subtle, vexing, and fundamental issues with such a security code validation mechanism, which we layout below:

1. ***Susceptibility to Click-Through Behavior***: Participants want to quickly get started on the call – their motivation is *not* security. In this light, they may not pay attention to this process requiring the comparison of meeting secuirty code, and simply **skip-through** the process (i.e., accept without looking or fully comparing). So, if there was an MITM attacker present and the code actually mismatches, no one will notice it. All of the security will be lost as a result. Such skip-through behavior is a well-known phenomenon in related applications of device pairing [2] as well other user-centered security tasks such as security warnings [3], where people just get habituated to security pop-ups, simply accept without due diligence and move on.
2. ***Susceptibility to Human Errors in the Code Comparison Task***: Now, let's say the participants are security conscious (like anyone who is reading this article ☺), and they do have a strong motivation to maintain security. Great, right? Actually….no. Comparing codes in this fashion is a cognitively taxing task, especially given that the code is a full domain hash and will contain multiple words. This will lead to two issues:

a) **False Positives (**aka False Accepts**)**: participants may sometimes not be able to detect the mismatch in the announced code vs. the displayed code. That is, an MITM attack may go unnoticed.
b) **False Negatives (**aka False Rejects**)**: participants may sometimes accidentally think that the matching codes are actually mismatching, thereby alerting the leader and others, which will mean the entire key management protocol will need to be aborted and restarted. Meaning, here a call that was not subject to an attack would need to be re-run for key management unnecessarily (which is already a pain!), which will necessitate participants to compare codes again, perhaps leading to errors (and most likely false positives) in the next round as participants will start getting tired and cognitively overwhelmed by the repeated process. This means the attacker may actually succeed if present in the next run.

3. ***Susceptibility to Dual Tasking Human Errors***: The meeting code comparison task is *not merely a comparison task*. It is actually a dual task, whereby, in addition to the actual comparison of the announced code with the displayed code, each participant must also validate whether the voice announcing the meeting code is indeed the voice of the leader (we call this second task the "**leader voice verification**" task). Unlike code comparison, leader voice verification task is actually an implicit task that perhaps happens naturally in the background at the same time the comparison computation happens in the brain. However, it is still something that would add cognitive burden on the user. As a result, we can expect that there will be **false negatives** resulting from this task alone, i.e., the rejection of the voice of the leader as being invalid (even though the code may be deemed as matching). These false negatives will again force a re-run of the protocol. Similarly, we can expect that there will be **false positives** resulting from this task alone, i.e., the acceptance of the attacker's voice (e.g., someone who naturally sounds similar to the leader or even a strong deep fake attacker who creates a close enough replica of the leader's voice based on some prior voice samples of the leader's voice) speaking the same code that the participant's client machine displays. These false positives will again lead to the success of the MITM attack in the protocol. These shallow and deep fake attacks have already been shown to be a vulnerability in our own prior study [4].

In short, the Zoom's approach to meeting security code validation will have severe consequences for the usability of the protocol and the security of the protocol, due to human skip-through and human errors in the dual task of comparison of the code and verification of the leader's voice.

3. Our Approach: Addressing the Above Problems with Speech-to-Text Transcription

In this document, we set out to address many of the aforementioned fundamental problems facing Zoom's Meeting Security Code validation approach. We suggest a new model using which Meeting Codes can be validated reliably, significantly improving both security and usability of the current design. Our proposal carefully leverages the strengths of humans and machines to make the checksum comparison and speaker verification tasks significantly more robust to errors/skip-through and address the MITM attacks at the signaling channel as well as the MITM attacks based on shallow or deep fake voice impersonation.

We posit that ***it is absolutely <u>unnecessary</u> to require the human participants to manually compare*** the announced meeting code with the displayed code. The main idea behind our approach, initially proposed in our work [5], is to automate the process of code comparison by using the current advancement in human speech transcription technology. This approach simply requires the leader to announce the meeting code (exactly like in the original proposal by Zoom [1]), but now, instead of the human participant, an NLP algorithm running on the participant's client machine will automatically transcribe the spoken code and performs the comparison on behalf of the participant (Figure 2). Automating the code comparison task provides several key advantages over the Zoom's approach [1], as listed below (visualized in Figure 3).

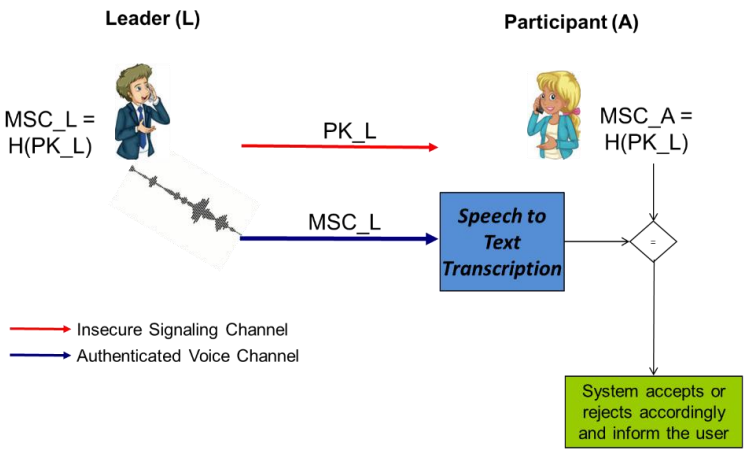

Figure 2: Our proposed approach. The comparison task will be performed automatically, thus addressing skip-through and human errors, thereby greatly improving the security and usability of code validation.

1. The probability of the success of signaling channel MITM, due to human errors or skip-through behavior in the code comparison task, could be highly reduced (or even eliminated).
2. The overall code validation task becomes much more reliable since the user only needs to perform a *single task* (voice validation only, with no need for code comparison) which greatly reduces the cognitive burden on the user and hence the accuracy in the voice verification task, which may help reduce the corresponding false negatives in the voice verification task and the false positives in the presence of voice-based MITM attacks using shallow or deep fake technology.
3. Longer meeting codes can be supported without noticeable increase in the errors. Why do we need longer codes? Hash functions keep getting compromised, may need longer hashes to counter such threats as the attacks evolve.

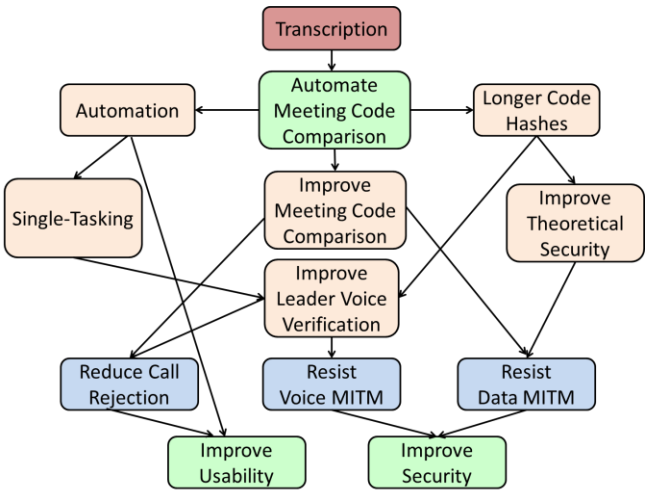

Figure 3 The path to improved security and usability in our proposed approach

4. Our Work Demonstrating the Practical Significance of the Proposed Idea

This is not merely a theoretical idea, it has great practical value in our opinion. We have performed a preliminary exploration of this new design for E2EE systems that use 20-bit or 40-bit short authenticated strings [5], which are basically the fingerprints of the key exchange protocol, as meeting codes. This proof of concept design evaluated the idea of automated checksum comparison by using IBM Watson Speech to Text standard transcription engine. The results of our investigation shows that using automated meeting code comparison, our approach can: (1) drastically reduce the chances of false positives under signaling channel MITM to 0%, in contrast to manual code comparison design (which is over 28% due to human errors) [5], and (2) reduce the overall false negatives down to about 5%, much lower than the traditional design (which is over 20% due to human errors) [5]. This design of transcription-based automated meeting code comparison is based on 4-word and 8-word code using a *natural language dictionary* with which the transcription engine was trained.

In our recent (unpublished) work, we have extended this initial design to support a full-length meeting code hash (just like the Zoom's proposed meeting code) based on a *specialized dictionary of phonetically-distinct words* (PGP Words) based on Google Speech API. The general purpose speech to text engines, such as Google Speech API, use deep learning techniques to recognize the natural language based on the context. Since they may not be perfectly applicable to a sequence of words with no context, our comparison scheme suggests correction rules to carefully improve the accuracy of transcription of key fingerprints. More specifically, our comparison is case-insensitive, accepts variant spelling and parts of the speech, and can detect word deletions and insertions. We recruited several Amazon Turk users who recited the security code's isolated words and evaluate the automated full-length meeting code verification tool on over 15,360 words spoken by these participants in our dataset. The results show that our model can offer a 0% false positive rate, and about 5% word error rate. We further showed how we can achieve an acceptably low false negative rate of 1.2% by pre-evaluating by relaxing the requirements of the number of accepted words while still retaining the 0% false positive rate.

5. Further Improvement, Refinements and Challenges

a. **Transitioning to practice**: While we do have an effective prototype, some more (engineering) work is needed to optimize, refine and further test it towards the deployment in the Zoom's specific environment. To this end, we could test other speech transcription engines, including Zoom's own in-built transcription system.
b. **Offline Verification**: When the leader announces the meeting security code, we suggest it can also be recorded and the recorded audio file can be posted to the Bulletin Board. When a new participant joins the call at a later point of time, he or she can fetch the recording and verify the public key of the leader independent of the call. Such recordings may also prove useful for forensics purposes to analyze the presence of attacks, especially the deep fake attacks. Automating the comparison process will certainly be helpful also for this offline verification, besides real-time verification for the participants who are present at the time of executing the key exchange protocol.
c. **Out of Band Verification**: The authors of [1] suggested that meeting security code could also be verified via an out of band mechanism. One possibility for this is the offline verification approach

we suggested above – here rather than posting to the Bulletin Board, the recording of the meeting code announced by the leader can be sent over, let's say, an email to each participant. This is still better than sending a textual code via email (which is usually an insecure channel) since the code is spoken by the leader and provides authenticity implicitly.

d. **Potential Speech Impediments**: We are cognizant to the fact that certain speech impediments may make it difficult for the transcription approach to work well. An easiest solution might be to outsource the task of announcing the code from the leader with the potential impediment to another participant without the impediment. Another possibility is that if such impediments are met, the secuirty code can be compared via the manual approach. More work can be conducted to design specific word dictionaries which may work with certain known speech impediments when transcribing.

References

[1] Blum et al. End-to-End Encryption for Zoom Meetings. Zoom's Security White Paper, May 2020.

[2] Kuo C., Walker J., Perrig A. (2007) Low-Cost Manufacturing, Usability, and Security: An Analysis of Bluetooth Simple Pairing and Wi-Fi Protected Setup. In: Financial Cryptography and Data Security. FC 2007.

[3] Anderson, B., Kirwan, B., Eargle, D., Howard, S., Vance, A. 2015. "How Polymorphic Warnings Reduce Habituation in the Brain—Insights from an fMRI Study," Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI), Seoul, Korea, pp. 2883–2892.

[4] M. Shirvanian and N. Saxena. Wiretapping via Mimicry: Short Voice Imitation Man-in-the-Middle Attacks on Crypto Phones. In ACM Conference on Computer and Communications Security (CCS), November 2014.

[5] M. Shirvanian and N. Saxena. CCCP: Closed Caption Crypto Phones to Resist MITM Attacks, Human Errors and Click-Through. In ACM Conference on Computer and Communications Security (CCS), October/November 2017.