AuthentiCall: Efficient Identity and Content Authentication for Phone Calls

Bradley Reaves, *North Carolina State University;* Logan Blue, Hadi Abdullah, Luis Vargas, Patrick Traynor, and Thomas Shrimpton, *University of Florida*

https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/reaves

This paper is included in the Proceedings of the 26th USENIX Security Symposium

August 16–18, 2017 • Vancouver, BC, Canada

ISBN 978-1-931971-40-9

AuthentiCall: Efficient Identity and Content Authentication for Phone Calls

Bradley Reaves
North Carolina State University
reaves@ufl.edu

Logan Blue
University of Florida
bluel@ufl.edu

Hadi Abdullah
University of Florida
hadi10102@ufl.edu

Luis Vargas
University of Florida
lfvargas14@ufl.edu

Patrick Traynor
University of Florida
traynor@cise.ufl.edu

Thomas Shrimpton
University of Florida
teshrim@cise.ufl.edu

Abstract

Phones are used to confirm some of our most sensitive transactions. From coordination between energy providers in the power grid to corroboration of high-value transfers with a financial institution, we rely on telephony to serve as a trustworthy communications path. However, such trust is not well placed given the widespread understanding of telephony's inability to provide end-to-end authentication between callers. In this paper, we address this problem through the AuthentiCall system. AuthentiCall not only cryptographically authenticates both parties on the call, but also provides strong guarantees of the integrity of conversations made over traditional phone networks. We achieve these ends through the use of formally verified protocols that bind low-bitrate data channels to heterogeneous audio channels. Unlike previous efforts, we demonstrate that AuthentiCall can be used to provide strong authentication before calls are answered, allowing users to ignore calls claiming a particular Caller ID that are unable or unwilling to provide proof of that assertion. Moreover, we detect 99% of tampered call audio with negligible false positives and only a worst-case 1.4 second call establishment overhead. In so doing, we argue that strong and efficient end-to-end authentication for phone networks is approaching a practical reality.

1 Introduction

Telephones remain of paramount importance to society since their invention 140 years ago, and they are especially important for sensitive business communications, whistleblowers and journalists, and as a reliable fallback when other communication systems fail. When faced with critical or anomalous events, the default response of many organizations and individuals is to rely on the telephone. For instance, banks receiving requests for large transfers between parties that do not generally interact call account owners. Power grid operators who detect phase synchronization problems requiring careful remediation speak on the phone with engineers in adjacent networks. Even the Federal Emergency Management Agency (FEMA) recommends that citizens in disaster areas rely on phones to communicate sensitive identity information (e.g., social security numbers) to assist in recovery [29]. In all of these cases, participants depend on telephony networks to help them validate claims of identity and integrity.

However, these networks were never designed to provide end-to-end authentication or integrity guarantees. Adversaries with minimal technical ability regularly take advantage of this fact by spoofing Caller ID, a vulnerability enabling over $7 billion in fraud in 2015 [34]. More capable adversaries can exploit weaknesses in core network protocols such as SS7 to reroute calls and modify content [15]. Unlike the web, where mechanisms such as TLS protect data integrity and allow experts to reason about the identity of a website, the modern telephony infrastructure simply provides no means for anyone to reason about either of these properties.

In this paper, we present AuthentiCall, a system designed to provide end-to-end guarantees of authentication and call content integrity over modern phone systems (e.g., landline, cellular, or VoIP). While most phones have access to some form of data connection, that connection is often not robust or reliable enough to support secure VoIP phone calls. AuthentiCall uses this often low-bitrate data connection to mutually authenticate both parties of a phone call with strong cryptography *before* the call is answered. Even in the worst case, this authentication adds at most a negligible 1.4 seconds to call establishment. Once a call is established, AuthentiCall binds the call audio to the original authentication using specialized, low-bandwidth digests of the speech in the call. These digests protect the integrity of call content and can distinguish legitimate audio modifications attributable to the network from

99% of maliciously tampered call audio even while a typical user would expect to see a false positive only once every six years. Our system is the first to use these digests to ensure that received call audio originated from the legitimate source and has not been tampered with by an adversary. Most critically, AuthentiCall provides these guarantees for standard telephone calls without requiring changes to any core network.

Our work makes the following contributions:

- **Designs Channel Binding and Authentication Protocols:** We design protocols that bind identities to phone numbers, mutually authenticate both parties of a phone call, and protect call content in transit.
- **Evaluates Robust Speech Digests for Security:** We show that proposed constructions for digesting speech data in systems that degrade audio quality can be made effective in adversarial settings in real systems.
- **Evaluates Call Performance in Real Networks:** Our prototype implementation shows that the techniques pioneered in AuthentiCall are practical and performant, adding at most only 1.4 seconds to phone call establishment in typical settings.

We are not the first to address this problem [2, 9, 17, 21, 43, 47, 56, 77]. However, other approaches have relied upon weak heuristics, fail to protect phone calls using the public telephone network, are not available to end users, neglect to protect call content, are trivially evaded, or add significant delay to call establishment. AuthentiCall is the only system that authenticates phone calls and content with strong cryptography in the global telephone network with negligible latency and overhead. We compare AuthentiCall to other existing or proposed systems in Section 9.

The remainder of this paper is organized as follows: Section 2 provides background information about the challenges underlying authentication in telephony networks; Section 3 describes our assumptions about adversaries and our security model in detail; Section 4 gives a formal specification of the AuthentiCall system; Section 5 discusses how analog speech digests can be used to achieve call content integrity; Section 6 provides details of the implementation of our system; Section 7 shows the results of our experiments; Section 8 offers additional discussion; Section 9 analyzes related work; and Section 10 provides concluding remarks.

2 Background

Modern telephony systems are composed of a mix of technologies. As shown in Figure 1, the path between a caller and callee may transit through multiple networks consisting of mobile cores, circuit-switched connections and packet-switched backbones. While the flow of a call across multiple network technologies is virtually invisible to customers, significant transformations occur to call audio between source and destination. Whereas the content of data packets on the Internet should not be modified between source and destination, call audio is transcoded by gateways to ensure that it is compatible with the underlying network. As such, *users of the global telephony infrastructure can only be guaranteed that an approximate but not bitwise identical representation of their voice will be delivered to the other end of the call.*

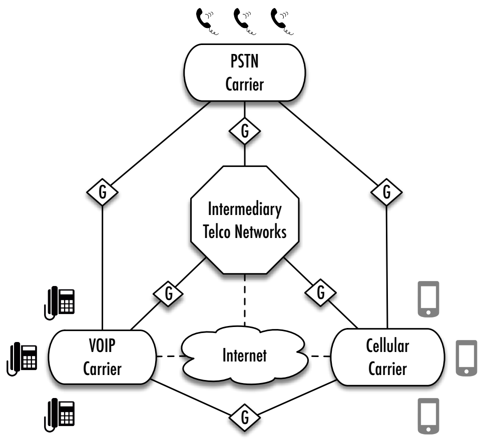


Figure 1: In the modern phone network, calls are often routed through gateways at network boundaries that remove authentication information and modify call audio.

Any other data that may be generated by a user or their home network is not guaranteed to be delivered or authenticatable end-to-end. That is, because the underlying technologies are heterogeneous, there is no assurance that information generated in one system is passed (much less authenticated) to another. This has two critical implications. The first is that any proofs of identity a user may generate to their provider are not sent to the other end of the call. For instance, a mobile phone on a 4G LTE connection performs strong cryptographic operations to prove its identity to its provider. However, there exists no means to share such proofs with a callee within this system let alone one in another provider's network. Second, claims of identity (e.g., Caller ID) are sent between providers with no means of verifying said claims. As evidenced by greater than $7 billion in fraud in 2015 [34], it is extremely simple for an adversary to trick a receiver into believing any claim of identity. There is no simple solution as calls regularly transit multiple intermediate networks between the source and destination.

It is increasingly common that modern phones have simultaneous access to at least low-bitrate data channels. VoIP phones naturally have a secondary data channel, the majority of mobile phones allow users to both talk and use data networks simultaneously, and even some circuit-switched connections (e.g., ISDN) provide phones with a data connection. The presence of these data services does *not* mean that all calls can be simply converted to VoIP. For example, cellular data in many places does not support the high data-rate or quality of service necessary for intelligible calls. Moreover, it is unlikely that any provider will entirely scrap their non-VoIP infrastructure. Accordingly, we argue that the presence of this low-bitrate data channel creates opportunities to develop a uniform means of end-to-end authentication across the heterogeneous mechanisms for delivering call audio.

3 Security Model

In order to authenticate voice calls and content, AuthentiCall will face adversaries with a range of capabilities. The simplest adversary will attempt to commit phone fraud by spoofing Caller ID when calling a target [59, 60]. An equivalent form of this attack may occur by the adversary tricking their target to call an arbitrary number under their control (e.g., via spam or phishing) and claiming to represent some other party (e.g., a financial institution) [46]. Additionally, this adversary may perform a call forwarding attack, which forces a target calling a legitimate number to be redirected to the adversary. Lastly, the adversary may place a voice call concurrent with other legitimate phone calls in order to create a race condition to see which call arrives at the destination first. In all of these cases, the goal of the adversary is to claim another identity for the purpose of extracting sensitive information (e.g., bank account numbers, usernames, and passwords).

A more sophisticated adversary may gain access to a network core via vulnerabilities in systems such as SS7 [15], or improperly protected legal wiretapping infrastructure [74]. This adversary can act as a man-in-the-middle, and is therefore capable of redirecting calls to an arbitrary endpoint, acting as an arbitrary endpoint, hanging up one side of a call at any point in time, and removing/injecting audio to one or both sides. Such an adversary is much more likely to require nation-state level sophistication, but exists nonetheless. Examples of both classes of adversary are shown in Figure 2.

Given that the bitwise encoding of audio is unlikely to be the same at each endpoint, end-to-end encryption is not a viable means of protecting call content or integrity across the heterogeneous telephony landscape. Moreover, while we argue that the majority of phones have access to at least a low-bandwidth data connection, solutions that demand high-speed data access at all times (i.e., pure VoIP calls) do not offer solutions for the vast majority of calls (i.e., cellular calls). Finally, we claim no ability to make changes throughout the vast and disparate technologies that make up the core networks of modern telephony and instead focus strictly on addressing this problem in an end-to-end fashion.

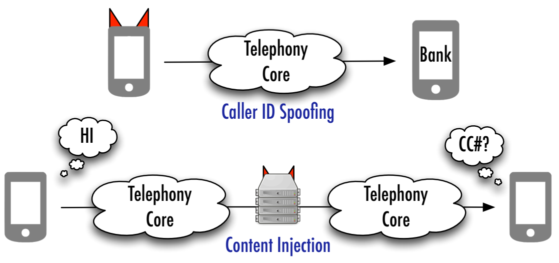


Figure 2: Broad overview of attacks possible on Caller ID and call content in current telephony landscape.

We define four participants: the Caller (R), the Callee (E), the Server (S), and the Adversary (Adv). Callers and Callees will register with the AuthentiCall service as described in the next section and will generate credentials[1] that include a public key. AuthentiCall will achieve the following security goals in the presence of the above-described adversaries:

1. **(G1) Proof of Number Ownership:** During the process of registration, R will actively demonstrate ownership of its claimed Caller ID to S before it receives a signed certificate.
2. **(G2) Authentication of the Caller:** E will be able to cryptographically verify the identity of R prior to accepting an incoming call.
3. **(G3) Authentication of the Callee:** R will be able to cryptographically verify the identity of E as soon as the call begins.
4. **(G4) Integrity Protection of Call Content:** R and E will be able to verify that the analog voice content has not been meaningfully altered, or that new content has not been injected by a man in the middle. Additionally, both will also be protected against concurrent call attacks.
5. **(G5) Proof of Liveness:** Both R and E will be able to detect if the other party is no longer on the call, perhaps as the result of a man in the middle attempting to engage in the call after the initial authentication.

We note that AuthentiCall does not provide confidentiality guarantees. While recent work has shown how to build systems that support anonymous calling [31], encrypting call audio end-to-end in lossy, heterogeneous telephone networks remains an open problem.

4 Protocol Design and Evaluation

Previously, we saw that AuthentiCall has five security goals to meet, and this section describes the three protocols that AuthentiCall uses to achieve these goals. These

[1]The details of which are described in depth in Section 4.

are the *Enrollment*, *Handshake*, and *Call Integrity* protocols.

These protocols make use of certificates issued to each client that indicate that a particular client controls a specific phone number. In prior work we proposed a full public key infrastructure for telephony [56] called a "TPKI" that would have as its root the North American Numbering Plan Administration with licensed carriers acting as certificate authorities. This PKI would issue an authoritative certificate that a phone number is owned by a particular entity, and AuthentiCall could enforce that calls take place between the entities specified in those certificates. While AuthentiCall can leverage the proposed TPKI, a fully-deployed TPKI is not necessary as AuthentiCall can act as its own certificate authority (this is discussed further in the enrollment protocol).

All of these protocols make use of a client-server architecture, where an AuthentiCall server acts as either an endpoint or intermediary between user clients. There are several reasons for this design choice. First, having a centralized relay simplifies the development of AuthentiCall. Although there are risks of adding a centralized point on a distributed infrastructure, our design minimizes them by distributing identity verification to a certificate authority and only trusting a central server to act as a meeting point for two callers. Second, it allows the server to prevent abuses of AuthentiCall like robodialing [71] by a single party by implementing rate limiting. The server can authenticate callers before allowing the messages to be transmitted, providing a mechanism for banning misbehaving users. Finally, all protocols (including handshake and enrollment) implement end-to-end cryptography. Assuming the integrity of the AuthentiCall certificate authority infrastructure and the integrity of the client, no other entity of the AuthentiCall network can read or fabricate protocol messages. We also assume that all communications between clients and servers use a secure TLS configuration with server authentication.

Our protocols have another goal: no human interaction except for choosing to accept a call. There are two primary reasons for this. First, it is well established that ordinary users (and even experts) have difficulty executing secure protocols correctly [76]. Second, in other protocols that rely on human interaction, the human element has been shown to be the most vulnerable [63].

The following subsections detail the three protocols in AuthentiCall. First, the enrollment protocol ensures that a given AuthentiCall user actually controls the phone number they claim to own (G1). The enrollment protocol also issues a certificate to the user. Second, the handshake protocol mutually authenticates two calling parties at call time (G2 and G3). Finally, the call integrity protocol ensures the security of the voice channel and the content it carries (G4 and G5).

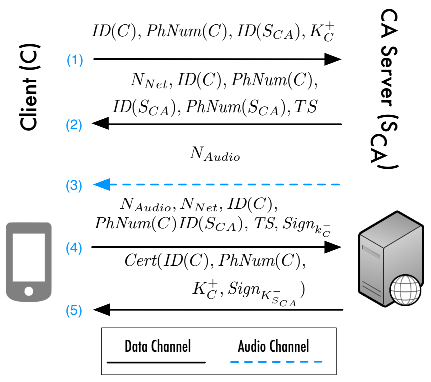

Client (C)
CA Server (S_{CA})
$ID(C), PhNum(C), ID(S_{CA}), K_C^+$
(1)
$N_{Net}, ID(C), PhNum(C),$
$ID(S_{CA}), PhNum(S_{CA}), TS$
(2)
N_{Audio}
(3)
$N_{Audio}, N_{Net}, ID(C),$
$PhNum(C) ID(S_{CA}), TS, Sign_{k_C^-}$
(4)
$Cert(ID(C), PhNum(C),$
$K_C^+, Sign_{K_{S_{CA}}^-})$
(5)
Data Channel
Audio Channel

Figure 3: Our enrollment protocol confirms phone number ownership and issues a certificate.

4.1 Enrollment Protocol

The enrollment protocol ensures that a client controls a claimed number and establishes a certificate that binds the identity of the client to a phone number. For our purposes, "identity" may be a user's name, organization, or any other pertinent information. Binding the identity to a phone number is essential because phone numbers are used as the principal basis of identity and routing in phone networks, and they are also used as such with AuthentiCall. The enrollment protocol is similar to other certificate issuing protocols but with the addition of a confirmation of control of the phone number.

Figure 3 shows the details of the enrollment protocol. The enrollment protocol has two participants: a client C and an AuthentiCall enrollment server S_{CA}. In message 1, C sends an enrollment request with S_{CA}'s identity, C's identity info, C's phone number, and C's public key. In message 2, the server sends a nonce N_{Net}, the identities of C and S_{CA} and the phone numbers of C and S_{CA} with a timestamp to ensure freshness, liveness, and to provide a "token" for this particular authentication session.

In message 3, the server begins to confirm that C controls the phone number it claims. The number is confirmed when S_{CA} places a call to C's claimed phone number. When the call is answered, S_{CA} transmits a nonce *over the voice channel*. Having S_{CA} call C is a critical detail because intercepting calls is far more difficult than spoofing a source number.[2] Using a voice call is important because it will work for any phone – including VoIP devices that may not have SMS access.

In message 4, C sends both N_{Net} and N_{Audio} along with the IDs of server, clients, a timestamp, and a signature covering all other fields. This final message concludes the proof of three things: possession of N_{Net}, the ability

[2]We will revisit the threat of call interception later in this subsection.

to receive a call by providing N_{Audio} and possession by C of the private key K_C^- by virtue of signing the message.

In message 5, S_{CA} replies with a signed certificate issued to C. This completes the enrollment protocol.

We note that this protocol is subject to the same limitations on certifying identity as every other Internet certificate authority. In particular, we will require an out-of-band process to verify identity for high-value certificates, and will require the ability to authenticate supporting documentation. AuthentiCall can also use other authoritative information sources like CNAM[3] lookups to verify number ownership in some cases. While no system or process is perfect, these types of policies have been largely effective on the Internet.

We also note that this is a trust-on-first-use (TOFU) protocol. While the protocol is secure in the presence of passive adversaries on both the data and voice networks, if an adversary can actively intercept a call addressed to a victim phone number (and also supply any out-of-band identity confirmation), they may be able to obtain a certificate for a number they illicitly control. If a TPKI were deployed, this attack would not be possible. Even without a TPKI, the likelihood of a successful attack is limited. Success is limited because the attack would eventually be detected by the legitimate owner when they attempt to register or authenticate using the legitimate number. To further protect against the prior attack, our protocol meets an additional goal: human interaction is not required for enrollment and confirming control of the claimed phone number. This means that automatic periodic reverification of phone number control is possible. This is important to prevent long-term effects of a brief phone number compromise, but also for more mundane issues like when phone numbers change ownership.

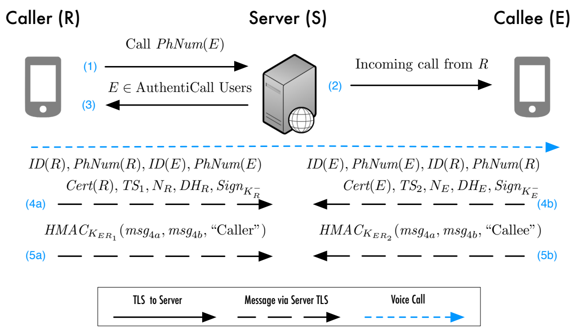
4.2 Handshake Protocol

The handshake protocol takes place when a caller intends to contact a callee. The caller places a voice call over the telephone network while simultaneously using a data connection to conduct the handshake protocol.

The handshake protocol consists of two phases. The first indicates to the AuthentiCall server and the calling party that a call is imminent. The second phase authenticates both parties on the call and establishes shared secrets. These secrets are only known end-to-end and are computed in a manner that preserves perfect forward secrecy. Figure 4 shows the handshake protocol.

Prior to the start of the protocol, we assume that C has

[3]CNAM is the distributed database maintained by carriers that maps phone numbers to the names presented in traditional caller ID. While spoofing a number is trivial, CNAM lookups occur out-of-band to call signaling and results could only be spoofed by a carrier, not a calling party.



Caller (R)
Server (S)
Callee (E)
Call $PhNum(E)$
(1)
$E \in$ AuthentiCall Users
(3)
Incoming call from R
(2)
$ID(R), PhNum(R), ID(E), PhNum(E)$
$Cert(R), TS_1, N_R, DH_R, Sign_{K_R^-}$
(4a)
$ID(E), PhNum(E), ID(R), PhNum(R)$
$Cert(E), TS_2, N_E, DH_E, Sign_{K_E^-}$
(4b)
$HMAC_{K_{ER_1}}(msg_{4a}, msg_{4b}, \text{“Caller”})$
(5a)
$HMAC_{K_{ER_2}}(msg_{4a}, msg_{4b}, \text{“Callee”})$
(5b)
TLS to Server
Message via Server TLS
Voice Call

Figure 4: Our handshake protocol mutually authenticates both parties.

connected to S via TLS, meaning S has properly authenticated itself to C. After connecting C authenticates itself to S, by either presenting a username/password pair or by signing a challenge with its private key.

The first phase consists of messages 1–3. In message 1, a caller R indicates to an AuthentiCall server S that R would like to place a call to the callee E. In message 2, S informs the callee E that an authenticated voice call is incoming.

In message 3, S informs R whether E is an AuthentiCall user or not, but does not provide information about E's presence or availability. Message 3 has several aims. The first is to protect the privacy of E. A strawman mechanism to protect privacy is for AuthentiCall to provide no information about E until E agrees to accept the call. However, this presents a problem: if an adversary tampers or blocks messages from E, it prevents E from participating in the handshake, and R would have to assume (in the absence of outside knowledge) that E is not a participant in AuthentiCall. This would allow an adversary to evade AuthentiCall. To solve this problem, S simply indicates to R whether or not R should expect to complete an AuthentiCall handshake for this call if E is available and chooses to accept the call. This reveals only E's preference to authenticate a phone call, and nothing about her availability or whether she has even chosen to accept or reject a call. Protecting this information is important because if an unwanted callee knows that a user is available, they may call repeatedly or use that information in other undesirable ways (e.g., harassment or telemarketing). If message 3 indicates that E is not an AuthentiCall user but E does not choose to accept the call, R must simply wait for the call request to time out. From R's perspective, this is no different from dialing and waiting for a busy signal or voicemail and should add little to no latency to the call. If message 3 indicates that E is not an AuthentiCall user, the protocol ends at this step and R is forced to fallback to an insecure call.

The second handshake phase authenticates R and E and consists of messages 4A-B and 5A-B. These messages are indicated by letters A and B because the messages contain the same fields for caller and callee respectively. They can be computed independently and sent in parallel, reducing round trip latencies.

Message 4 contains all information necessary for a Diffie-Hellman key establishment authenticated with a signature key defined in the certificate of R or E. It also contains identity information for R or E, the calling or called phone number, a timestamp, and a nonce. Each side also provides a Diffie-Hellman share, and the entire message is signed with the public key in the certificate issued by AuthentiCall.

After message 4, both sides combine their Diffie-Hellman secret with the share they received to generate the derived secret. Each client then generates keys using the Diffie-Hellman result, the timestamps of both parties, and the nonces of both parties. These keys are used to continue the handshake and to provide keys for the integrity protocol.

Message 5A and 5B contain an HMAC of messages 4A and 4B along with a string to differentiate message 5A from message 5B. The purpose of this message is to provide key confirmation that both sides of the exchange have access to the keys generated after messages 4A and 4B. This message concludes the handshake protocol.

4.3 Call Integrity Protocol

The call integrity protocol binds the handshake conducted over the data network to the voice channel established over the telephone network. Part of this protocol confirms that the voice call has been established and confirms when the call ends. The remainder of the messages in this protocol exchange content authentication information for the duration of the call. This content integrity takes the form of short "digests" of call audio (we discuss these digests in detail in the following section). These digests are effectively heavily compressed representations of the call content; they allow for detection of tampered audio at a low bit rate. Additionally, the digests are exchanged by both parties and authenticated with HMACs.

Figure 5 shows the details of the call integrity protocol. The protocol begins *after the voice call is established*. Both caller R and callee E send a message indicating that the voice call is complete. This message includes a timestamp, IDs of the communicating parties and the HMAC of all of these values. The timestamp is generated using the phone clock which is often synchronized with the carrier.[4] These messages are

[4]In this setting, loose clock synchronization (approximately one minute) is sufficient; if necessary, S can also provide a time update at login.

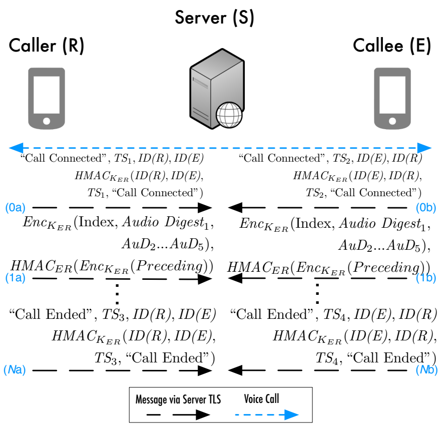

Server (S)
Caller (R)
Callee (E)
"Call Connected", $TS_1, ID(R), ID(E)$ $HMAC_{K_{ER}}(ID(R), ID(E), TS_1$, "Call Connected")
"Call Connected", $TS_2, ID(E), ID(R)$ $HMAC_{K_{ER}}(ID(E), ID(R), TS_2$, "Call Connected")
(0a)
(0b)
$Enc_{K_{ER}}$(Index, $Audio\ Digest_1, AuD_2 ... AuD_5$), $HMAC_{ER}(Enc_{K_{ER}}(Preceding))$
$Enc_{K_{ER}}$(Index, $Audio\ Digest_1, AuD_2 ... AuD_5$), $HMAC_{ER}(Enc_{K_{ER}}(Preceding))$
(1a)
(1b)
"Call Ended", $TS_3, ID(R), ID(E)$ $HMAC_{K_{ER}}(ID(R), ID(E), TS_3$, "Call Ended")
"Call Ended", $TS_4, ID(E), ID(R)$ $HMAC_{K_{ER}}(ID(E), ID(R), TS_4$, "Call Ended")
(Na)
(Nb)
Message via Server TLS
Voice Call

Figure 5: Our call integrity protocol protects all speech content.

designed to prevent attacks where a call is redirected to another phone. One possible attack is an adversary maliciously configuring call forwarding on a target; the handshake would be conducted with the target, but the voice call would be delivered to the adversary. In such a case, the target would not send a "call established" message and the attack would fail.

Once the voice call begins, each side will encrypt and send the other audio digests at a regular interval. It is important to note that we use unique keys generated during the handshake for encryption, message authentication codes, and digest calculation. The messages also guarantee freshness because the index is effectively a timestamp, and the message authentication codes are computed under a key unique to this call. Timestamps in messages 1-N are indexed against the beginning of the call, negating the need for a synchronized clock. In order to prevent redirection attacks, the messages are bound to the identities of the communicating parties by including the IDs in the HMACs and by using keys for the HMACs that are unique to the call.

When the voice call ends, each side sends a "call concluded" message containing the client IDs, a timestamp, and their HMAC. This alerts the end point to expect no more digests. It also prevents a man-in-the-middle from continuing a call that the victim has started and authenticated.

4.4 Evaluation

Our protocols use standard constructions for certificate establishment, certificate-based authentication, authenticated key establishment, and message authentication. We therefore believe our protocols are secure based on inspection. Nevertheless, we used ProVerif [20] to further analyze the handshake and enrollment protocols. Our ProVerif code can be found in our technical report [55]. The analysis verified that our handshake protocol establishes and never leaks the secret key. The protocol also provides authentication and perfect forward secrecy for both the caller and callee. The enrollment protocol is verified to never leak the private keys of either party. This property allows us to assert that both signatures and certificates cannot be forged.

5 Speech Digest Design and Evaluation

The previous section describes how AuthentiCall enrolls and authenticates users prior to a call. During a call, AuthentiCall needs a way to summarize speech content in order to authenticate audio using a low-bandwidth data connection. To accomplish this goal, we leverage research from an area of signal processing that produces techniques that are known as "perceptual hashes" or "robust hashes." Robust digests have been developed for a wide domain of inputs, including music, images, and speech, but their applicability has remained limited. Unlike cryptographic hashes, which change drastically with small changes in input, robust hashes give very similar outputs for similar inputs. By definition, a robust digest cannot provide collision resistance (or second preimage resistance) because collisions are the property that make them useful. In this paper, we call these techniques "speech digests" to avoid confusion with cryptographic hashes. To our knowledge, this work presents one of the first uses of robust speech digests for security.

A speech digest has two goals. First, it must accurately summarize the content of the call. However, it is not necessary for this summary to be lossless or meaningful for human interpretation. We are also concerned more with semantics (i.e., words spoken) than we are with speaker voice characteristics (e.g., tone, identity) or extraneous features like noise. Second, the digest must be robust to non-semantic changes in audio.

Because of ambient or electronic noise, intermittent loss, and the use of differing encodings throughout the phone network, the audio transmitted by a phone will not be the same as the audio received. In particular, the audio received is practically guaranteed to *not* be identical on a bit level to the audio sent by the phone. This means that common data digest approaches like cryptographic hashes will fail.

While the original phone system used analog transmission of voice, it is now common in every telephone network (landline, VoIP, cellular, etc.) for speech to be digitized and compressed using an audio codec. At network boundaries, it is common for audio to be decoded and recoded into a different codec (known as transcoding). Codecs used in the phone network are highly lossy and drastically distort the call audio, and so have the potential to significantly impact audio digest performance. Because some phone systems (especially cellular and VoIP) use lossy networks for transmission, frames are routinely lost. For example, loss rates of 4% are considered nominal for cellular voice [12].

These legitimate modifications caused by the phone network must be distinguished from changes to audio induced by an adversary. The following subsections provide a description of the speech digests we use in AuthentiCall and a thorough analysis of the performance of these digests for telephone calls.

5.1 Construction and Properties

There are a number of constructions of speech digests, and they all use the following basic process. First, they compute derived features of speech. Second, they define a compression function to turn the real-valued features into a bit string. We use the construction of Jiao et al. [36] called `RSH`. We chose this technique over others because it provides good performance on speech at a low-bitrate, among other properties. We note that the original work did not evaluate the critical case where an adversary can control the audio being hashed. Our evaluation shows that `RSH` maintains audio integrity in this crucial case. The construction also selects audio probabilistically; we show in Appendix B that the digest indeed covers all of the semantic content in the input audio. Finally, to our knowledge we are the first to use any robust speech digest for an authentication and integrity scheme.

For space reasons, and because we do not claim the design of the `RSH` digest as a research contribution, we provide a detailed description of the actual computation of an `RSH` digest in Appendix A. However, the remainder of this subsection will provide details necessary for the rest of this paper. `RSH` computes a 512-bit digest for each 1 second of audio, and the digest can be thought of as a heavily compressed version of the audio in the call. The digest is computed probabilistically using a keyed pseudorandom number generator with a key derived during the handshake (Section 4.2) in AuthentiCall. The probabilistic nature of the digest ensures that digests of the same phrase (e.g., "Hello") differ and cannot simply be replayed. Digests are computed by the caller and are received and verified by the callee. The verifying party computes the digest of the received audio and the Hamming distance between the calculated and received digests. Because degradation of audio over a phone call is expected, digests will not match exactly. However, the Hamming distance between two audio digests (equivalent to the bit error rate (BER)) quantifies the change in the audio. By setting an appropriate threshold on BER, maliciously modified audio can be detected.

5.2 Implementation and Evaluation

Now that we have seen how RSH digests are computed, we can evaluate properties of RSH digests. This includes effects of legitimate transformations and the results of comparing digests of unrelated audio samples (as might be generated by an adversary). We also describe how we use digests to detect tampered audio.

We implement RSH using Matlab, and we deploy it in our AuthentiCall prototype by using the Matlab Coder toolbox to generate C code that is compiled as an Android native code library. We use the TIMIT audio corpus [30] which is a standard test dataset for speech processing systems. It consists of high-fidelity recordings of 630 male and female speakers reading 10 English sentences constructed for phonetic diversity. Because RSH computes hashes of one second of audio, we split the TIMIT audio data into discrete seconds of audio corresponding to a unique section of audio from a speaker and sentence. This resulted in 22,487 seconds of unique audio.

5.2.1 Robustness

Robustness is one of the most critical aspects of our speech digests, and it is important to show that these digests will not significantly change after audio undergoes any of the normal processes that occur during a phone call. These include the effects of various audio encodings, synchronization errors in audio, and noise. To test robustness, we generate modified audio from the TIMIT corpus and compare the BER of digests of standard TIMIT audio to digests of degraded audio. We first downsample the TIMIT audio to a sample rate of 8kHz, which is standard for most telephone systems. We used the sox [5] audio utility for downsampling and adding delay to audio to model synchronization error. We also used sox to convert the audio to two common phone codecs, AMR-NB (Adaptive Multi-Rate Narrow Band) and GSM-FR (Groupe Spécial Mobile Full-Rate). We used GNU Parallel [67] to quickly compute these audio files. To model frame loss behavior, we use a Matlab simulation that implements a Gilbert-Elliot loss model [32]. Gilbert-Elliot models bursty losses using a two-state Markov model parameterized by probabilities of individual and continued losses. We use the standard practice of setting the probability of an individual loss (p) and probability of continuing the burst ($1-r$) to the desired loss rate of 5% for our experiments. We also use Matlab's agwn function to add Gaussian white noise at a 30 decibel signal to noise ratio.

Figure 6 shows boxplots representing the distribution of BER rates of each type of degradation tested. All degradations show a fairly tight BER distribution near the median with a long tail. We see that of the effects tested, 10ms delay has the least effect; this is a result of the fact that the digest windows the audio with a high overlap. For most digests, addition of white noise also has little effect; this is because LSF analysis discards all frequency information except for the most important frequencies. We see higher error rates caused by the use of audio codecs like GSM-FR and AMR-NB; these codecs significantly alter the frequency content of the audio. We can also see that a 5% loss rate has negligible effect on the audio digests. Finally, we see that combining transcoding, loss, delay, and noise has an additive effect on the resulting digest error — in other words, the more degradation that takes place, the higher the bit error. These experiments show that RSH is robust to common audio modifications.

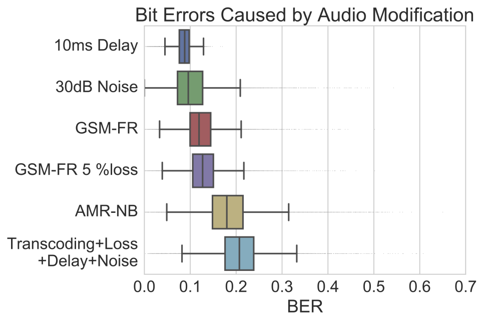


Figure 6: These box plots show the distribution of digests bit error rates as a result of various audio degradations. These error rates are well below the rates seen by adversarial audio, shown in Figure 7

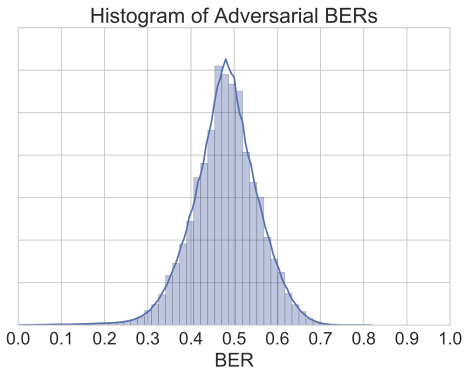


Figure 7: This graph shows the histogram and kernel density estimate of digest of adversarial audio on over 250 million pairs of 1-second speech samples. While the majority of legitimately modified audio has digest errors less than 35%, adversarial audio has digest BERs averaging 47.8%.

5.2.2 Adversarial Audio

While robustness is essential, the ultimate goal of these digests is to detect maliciously tampered or injected audio, which we term "adversarial audio." Such an analysis has not been previously performed. To validate the ability of RSH to detect adversarial audio we compute the BER of digests of every pair of seconds of TIMIT audio discussed in the previous section. This dataset includes 252,821,341 pairs of single seconds of audio. For this test, we use the same key for every hash; this models the situation where an adversary can cause the target to receive audio of its choice but not modify the associated digest.

We find that the mean BER between two distinct audio pairs is 0.478. A histogram and kernel density estimate of these values is also shown in Figure 7. This plot shows that the bit error is normally distributed with a mean and median of 0.478 and 0.480 (respectively). The expected bit error for two random bit strings is 50%, and the mean seen for RSH bit error is close to the optimal, best possible distance between two adversarial digests.

Because the TIMIT corpus contains speakers speaking several identical sentences, we can investigate the resilience of the digest to more specific adversarial scenarios in two important ways. First, we can look at whether using different speech from the same speaker can create a false positive. If so, this would be a serious problem because an adversary could use recorded words from the target speaker undetected. Second, we can determine if a different speaker uttering the same words causes false positives. This test indicates to what extent the digest is protecting *content* instead of speaker characteristics.

We found that digests from the same speaker speaking different content are accepted at practically the same rate as audio that differs in speaker and content. At a BER detection threshold of 0.384 (derived and discussed in the following subsection), the detection rate for different content spoken by the same speaker is 0.901482, while the detection rate for different content spoken by a different speaker is 0.901215. However, identical phrases spoken by different speakers results in a much higher rate of collision and a detection rate of 0.680353. This lower detection rate is not a problem for AuthentiCall because it is still high enough to detect modified call audio with high probability. More importantly, it indicates that RSH is highly sensitive to changes in call content.

5.2.3 Threshold selection and performance

Distinguishing legitimate and illegitimate audio requires choosing a BER threshold to detect tampered audio. Because the extreme values of these populations overlap, a tradeoff between detection and false positives must be made. The tradeoff is best depicted in a ROC curve in Figure 8. This figure shows the true positive/false positive tradeoff measured on the adversarial audio and two legitimate modifications – GSM encoding and a combination of GSM, AMR-NB, 5% frame loss, 10ms delay, and 30dB of white noise. This combination represents an approximate "worst case" of legitimate audio. Figure 8 shows excellent performance in terms of distinguishing audio. For GSM-only audio, we see an area-under-curve of 0.998, and for the "worst case" audio, we see an area-under-curve of 0.992. However, because digests will be used at a high rate (one per second), even with a very small false positive rate, alerting users for every individual detection will likely result in warning fatigue. As a result, the most important metric for evaluating a threshold is minimizing the user's likelihood of a false positive. This problem suggests trading off sensitivity to short changes in call content for a lower false positive rate. To reduce overhead and network load, AuthentiCall sends digests in groups of five. To provide high detection rates while limiting false positives, AuthentiCall alerts the user if any 3 out of 5 digests are greater than the BER threshold. We model true and false performance of this scheme as a set of five Bernoulli trials — successful authentication for true positives and successful digest collision for false positives. Thus, we can compute 3-out-of-5 performance using the binomial distribution.

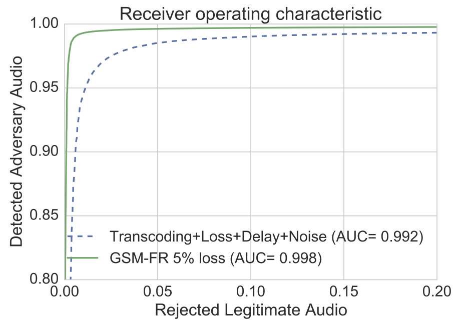


Figure 8: The digest performance ROC graph shows that digests can easily distinguish between legitimate and substituted audio, even in the presence of transcoding, loss, delay, and noise. These results are computed over digests of a single second. The graph is scaled to show the extreme upper corner.

After this analysis, we selected an individual-digest BER threshold of 0.384. This corresponds to an individual adversary audio true positive detection rate of 0.90, while presenting a 0.0058 false positive rate against our "worst-case" audio and a 0.00089 false positive rate against clean GSM-FR encoded audio. Using our "three-out-of-five" alerting scheme, the probability of detecting 3 or more seconds of tampered audio is 0.992. The false positive rate is drastically reduced: the false positive rate is 1.96×10^{-6}, and for clean GSM-FR audio the false positive rate is 7.02×10^{-9}. This corresponds to a false alert on average every 425.1 hours of talk time for worst case audio, and for GSM-FR audio one false positive every 118,766 hours. The average British mobile phone user only places 176 minutes per month of outbound calls [65]; assuming inbound and outbound talk time are roughly equal, the average user only places 70.4 hours of calls per year. This means that the average AuthentiCall user would only see a false alert *once every six years.*

5.2.4 Limitations

No security solution is perfect, and our use of audio digests have some limitations. The chief limitation is that audio digests cannot detect altered audio less than one second in length. This limitation is simply a result of the constraints of doing low-bitrate authentication of mutable and analog data.

While the digests are not perfect, we argue that they are secure against most adversaries. We note that audio digests have two purposes: 1) to provide a guarantee that the voice call established was the one that was negotiated in the handshake and 2) that the voice content has not significantly changed during the call. These two goals deal with adversaries of different capabilities. In particular, intercepting and modifying call audio requires far more advanced access and capability than simply spoofing a caller ID during a handshake already occurring. Audio digests will detect the first scenario within five seconds of audio, and it will also quickly detect changes that effect any three seconds in five for the second scenario.

In limited circumstances, it may be possible for a man-in-the-middle adversary to make small modifications to the received audio. For the second attack to be successful in the presence of these digests, a number of conditions must hold: First, the adversary can change no more than two seconds out of every five seconds of audio. Second, the adversary must change the audio in a way that would sound natural to the victim. This would mean that the changed audio would have to conform to both the current sentence pattern as well as the speaker's voice. While voice modification algorithms exist (e.g., Adobe VoCo [10] and Lyrebird [11]), modifying an existing sentence in an ongoing conversation is beyond the abilities of current natural-language processing. Also, since our digests depend on the semantic call content, changes to the structure of a sentence (and not necessary audible voice) would alert the user. Finally, in addition to the substantial difficulty of these limits, the adversary must also do all of this in soft-real-time.

Additionally, our threat model assumes the adversary has access to the audio (but not the keys) that generated the digest and thus second preimage resistance is a relevant property. Note that our security argument rests in the computational difficulty of finding a series of collisions in real-time using semantically relevant audio. The protection that RSH would provide for preimage resistance (given an arbitrary digest but no corresponding audio) depends primarily on the security of the keyed-pseudorandom selection of audio segments for each digest. Evaluating this property is interesting but not immediately relevant to the security of our system.

Nevertheless, a user is still not defenseless against such an attack. While we believe such attempts would likely be noticeable and suspicious to the human ear, users could also receive prompts from AuthentiCall when individual digests fail. These prompts could recommend that the user ask the opposing speaker to elaborate their prior point or to confirm other details to force the adversary to respond with enough tampered audio that the attack could be detected.

6 System Implementation

The previous sections described the protocol design and characterized our speech digests. In this section, we describe our AuthentiCall client and server implementation, and in the following section evaluate its performance.

Server: Our server was implemented in Java, using Twilio's Call API to call clients during the registration phase to share the audio nonce that confirms control of a phone number. Google Cloud Messaging (GCM) is used to generate a push notification to inform clients of incoming calls.

Client: Our prototype AuthentiCall client consists of an Android app, though we anticipate that in the future AuthentiCall will be available for all telephony platforms, including smartphones, VoIP phones, PBXs, and even landlines (with additional hardware similar in concept to legacy Caller ID devices that uses a wireless or wired LAN data connection).

A TLS connection is used to establish a secure channel between client and server. We implement the AuthentiCall protocol in Java using the Spongy Castle library [1]. The audio digests were implemented in Matlab, compiled to C, and linked into the app as native code. In our implementation, digest protocol messages contain five seconds of audio digests.

We use RSA-4096 to as our public key algorithm and SHA-3 for the underlying hash function for HMACs. To reduce handshake time, we use a standard set of NIST Diffie-Hellman parameters hardcoded into the client. These are NIST 2048-bit MODP group with a 256-bit prime order subgroup from RFC5114 [41]. We also use the HMAC-based key derivation algorithm used by TLS 1.2 described in RFC 5869 [39]. Upon registration, the

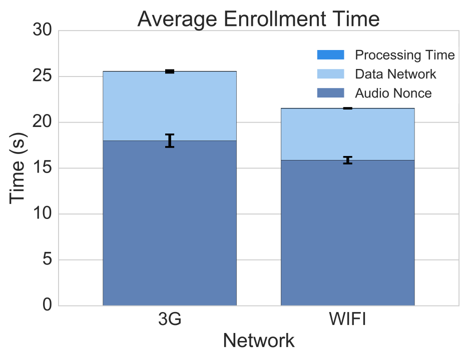


Figure 9: Enrollment takes less than 30 seconds and is a one time process that may be done in the background.

server issues the client an X509 certificate. This consists of a user's claimed identity, phone number, validity, public key and signature of the CA.

Audio Nonces: As described in Section 4, the AuthentiCall enrollment protocol sends a nonce through the voice channel to ensure that an client can receive a voice call. We use a 128-bit random nonce. In our implementation, the nonce is encoded as touch-tones (DTMF[5]). DTMF tones were used because they are faithfully transmitted through every telephone system and were simple to send and detect. There are 16 possible touch-tone digits [6], so each tone can represent an encoded hexadecimal digit. These tones are transmitted for 200ms each with a 100ms pause between tones. This provides a bit rate of 13.3 bits per second for a nonce transmission time of 9.6 seconds. This transmission time comprises the bulk of the time spent in the enrollment protocol.

7 Results

Our AuthentiCall implementation allows us to test its performance in enrollment, call handshakes, and detecting modified call audio in real phone calls.

7.1 Experiment Setup

Before describing individual experiments, we describe our experiment testbed. The AuthentiCall server was placed on an Amazon Web Services (AWS) server located in Northern Virginia. We used the same network provider, AT&T, and the same cellular devices, Samsung Galaxy Note II N7100s, across all experiments. The enrollment and handshake experiments were carried out 20 times over both WiFi and 3G, and digest exchange tests were done 10 times using WiFi. Digest exchange was done over WiFi as this experiment was used to validate content protection, not delivery speed. In all experiments, calls used a 3G voice channel.

We evaluate 3G and WiFi because our research phones do not support 2G-only operation. We note that not all wireless access is created equal, and actual speeds depend on many factors including network congestion, transmission power, and interference.

[5]Dual-Tone Multi-Frequency tones are the sounds made by dialing digits on a touch-tone phone.

[6]Four DTMF tones are not available on consumer phones but provide additional functionality in some special phone systems

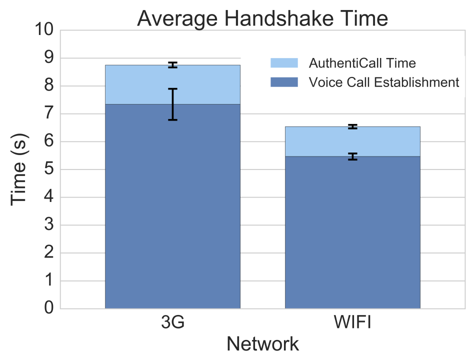


Figure 10: AuthentiCall adds 1 to 1.41 seconds to the phone call establishment, making the overhead effectively unnoticeable to users.

7.2 Enrollment Protocol

Our first experiments measure the user enrollment time. We measure the time from the instant a user begins enrollment to when the user receives the last protocol message, including all protocol messages and the audio nonce. For clients, enrollment is a one-time process that is done before the first call can be placed, analogous to activating a credit card. Figure 9 shows the average time of enrollment using 3G and WiFi to exchange protocol messages. The main contributor to the enrollment time comes from the transmission of the audio nonce which is used to establish ownership. Though the enrollment times over 3G and WiFi are 25 and 22 seconds respectively, this protocol requires no user interaction.

7.3 Handshake Protocol

We next measure the time to complete an entire handshake, including data messages and voice call setup. We note that voice call setup time is substantial, and requires many seconds even without AuthentiCall. We believe the most important performance metric is additional latency experienced by the end user. As shown in Figure 10, AuthentiCall only adds 1.07 seconds for WiFi or 1.41 seconds on 3G data to the total call establishment time (error bars indicate standard error). We believe that this will be unnoticeable to the user for several reasons. First, call establishment time varies significantly. This is normal network behavior, not an artifact introduced by AuthentiCall. In our 3G experiments our additional handshake time is approximately equal to the standard error in voice call establishment. We also note that our test phones were in the same location connected to the same tower, so the voice call setup time is likely lower than a typical call. In fact, our measured times are very close to the published estimates of 6.5 seconds for call setup by the tower between both phones [4]. Finally, we note that this is substantially faster than Authloop [56] which takes nine seconds to perform authentication *after call delivery*.

7.4 Speech Digest Performance

Our final experiments evaluate our speech digest accuracy over real call audio. In these 10 calls, we play 10 sentences from 10 randomly selected speakers in the TIMIT corpus through the call, and our AuthentiCall implementation computed the sent and received digests. In total this represented 360 seconds of audio. For simplicity, a caller sends audio and digests, and a callee receives the audio and compares the received and locally computed digests. We also compared these 10 legitimate call digests with an "adversary call" containing different audio from the hashes sent by the legitimate caller. To compare our live call performance to simulated audio from Section 5, we first discuss our individual-hash accuracy.

Figure 11 shows the cumulative distribution of BER for digests of legitimate audio calls and audio sent by an adversary. The dotted line represents our previously established BER threshold of 0.348.

First, in testing with adversarial audio, we see that 93.4% of the individual fraudulent digests were detected as fraudulent. Our simulation results saw an individual digest detection rate of 90%, so this means that our real calls see an even greater performance. Using our 3-out-of-5 standard for detection, we detected 96.7%. This test shows that AuthentiCall can effectively detect tampering in real calls. Next, for legitimate calls, 95.5% of the digests were properly marked as authentic audio. Using our 3-out-of-5 standard, we saw no five-second frames that were marked as tampered.

While our individual hash performance false positive rate of 4.5% was low, we were surprised that the performance differed from our earlier evaluation on simulated degradations. Upon further investigation, we learned that our audio was being transmitted using the AMR-NB codec set to the lowest possible quality setting (4.75kbps); this configuration is typically only used when reception is exceptionally poor, and we anticipate this case will be rare in deployment. Nevertheless, there

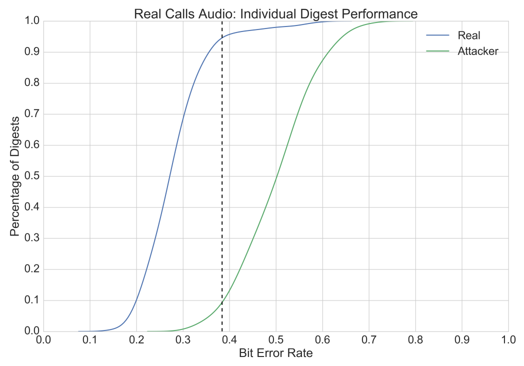


Figure 11: This figure shows that 93.4% of individual digests of adversarial audio are correctly detected while 95.5% of individual digests of legitimate audio are detected as authentic. Using a 3-out-of-5 detection scheme, 96.7% of adversarial audio is detected.

are several mechanisms that can correct for this. One option would be to digest audio *after* compression for transmission (our prototype uses the raw audio from the microphone); such a scheme would reduce false positives partially caused by known-good transformation of audio. Another option is to simply accept these individual false positives. Doing so would result in a false alert on average every 58 minutes, which is still acceptable as most phone calls last only 1.8 minutes [3].

8 Discussion

We now discuss additional issues related to AuthentiCall.

Applications and Use Cases: AuthentiCall provides a mechanism to mitigate many open security problems in telephony. The most obvious problems are attacks that rely on Caller ID fraud, like the perennial "IRS scams" in the United States. Another problem is that many institutions, including banks and utilities, use extensive and error-prone challenge questions to authenticate their users. These challenges are cumbersome yet still fail to stop targeted social engineering attacks. AuthentiCall offers a strong method to authenticate users over the phone, increasing security while reducing the authentication time and effort.

Another valuable use case is emergency services, which have faced "swatting" calls that endanger the lives of first responders [73] as well as denial of service attacks that have made it impossible for legitimate callers to receive help [8]. AuthentiCall provides a mechanism to allow essential services to prioritize authenticated calls in

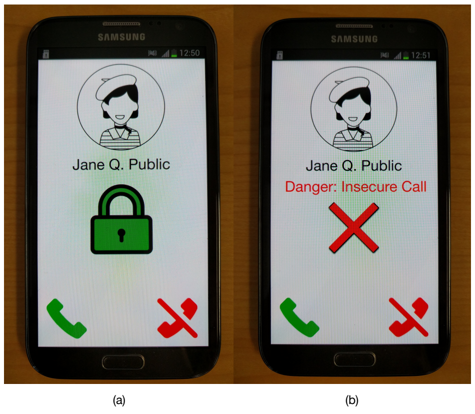


Figure 12: Before the call is answered, AuthentiCall indicates if the call is authenticated or unauthenticated

such a scenario while answering other calls opportunistically. While such a proposal would need to be reviewed by public policy experts and stakeholders, we provide a mitigation to a problem with no clear solution.

Server Deployment: AuthentiCall relies on a centralized server infrastructure to facilitate authenticated calls while minimizing abuse. AuthentiCall, including server infrastructure, could be provided by a carrier or an independent organization. While a centralized model is simplest to test our hypothesis that auxiliary data channels can be used to authenticate traditional voice calls, we intend to study decentralized and privacy-preserving architectures in future work.

Cellular Network Load: Systems that make use of the cellular network must be careful not to increase signaling load on the network in a harmful way [26, 40, 62]. We believe that AuthentiCall will not cause network harm because in modern networks (3G and 4G), data signaling is no longer as expensive as a voice call, and simultaneous voice and data usage is now commonplace.

Certificate Management: Any system that relies on certificates must address certificate revocation and expiration. AuthentiCall's centralized model allows the server to deny use of any revoked certificate, drastically simplifying revocation compared to CRLs or protocols like OCSP. Similar to Let's Encrypt [7], AuthentiCall certificates can have short lifetimes because certificate renewal using our enrollment protocol is fast and requires no human interaction. Our certificate authority proposal is one of many possible designs. As mentioned in Section 4, AuthentiCall could also make use of the proposed Telephony PKI [56]. In this scenario, certificate lifetime would be determined by the TPKI, which would also issue a certificate revocation list.

Why IP data: We chose IP data over other channels because it provides reliable and fast data transmission for most existing devices including smartphones, VoIP phones, and even landlines if provided with suitable hardware. As an example, SMS as a transmission carrier would be impractical. Bandwidth is low, and delivery is slow and not guaranteed [69]. In particular, the average time to send one SMS message is 6.4 seconds [53], meaning that AuthentiCall using SMS would require a minimum of *38.4 seconds* — effectively increasing call setup time by a factor of 5. If data connections are not available, users could use a system like Authloop to authenticate their calls. [56]

Why Not Biometrics: Robust speech digests are a superior solution for content integrity than voice biometrics for several reasons. First, voice authentication is simply not secure in adversarial settings [38]. Second, voice biometrics would assume that the call would only consist of a single party (e.g., speakerphones would not be supported). By contrast, audio digests are speaker independent and can be computed locally with no additional knowledge about the other party.

Denial of Service Adversaries may attempt to break the security of AuthentiCall by selectively dropping protocol messages, but AuthentiCall can detect these attacks and fail to complete a call or end an in-progress call. In the handshake, the client will not accept a voice call until the all authentication messages are complete. During the integrity protocol, the client can enforce tight timeouts of all messages and alert the user of an attack if expected messages do not arrive.

User Interface We have developed a complete working prototype of AuthentiCall for Android, including a preliminary simple user interface as shown in Figure 12. Along with previous research [72], this is one of the first interfaces to indicate secure Caller-ID, our prototype interface is intended to simply and clearly alert the user to the safety of the call. We note that indicating security in a user interface requires great care [13, 16], and we intend to formally study interface design for AuthentiCall in future work.

9 Related Work

Authentication has long been a concern in telephony networks. Chiefly motivating that concern has been the need to identify customers to bill for service usage [69]. The strength of such authentication mechanisms have varied widely, from easily breakable or weak authentication (e.g., 1G and 2G cellular) [18, 54] and authorization [42, 70, 75] to strong mutual authentication (e.g., LTE). However, all of these mechanisms do not provide authentication end-to-end.

Researchers have attempted to address the problem through one of two classes of solutions: heuristics or cryptography. In the case of the former, researchers have explored a wide range of solutions: blacklists of phone numbers [6, 44, 52], call-back verification [47], channel characterization [57], call data analysis [35, 45, 48, 58], carrier level ID extraction [68], timing [47], call provenance [17], name registries [22] and biometrics [14, 19, 27, 37]. The difficulty with these is that their defenses are probabilistic in nature and may be weak in various adversarial settings. Given the increasing number of attacks on machine learning algorithms [33,49,50], such techniques offer uncertain security properties.

As for cryptographic solutions, most have been VoIP-only (e.g., Zfone and Redphone) [2, 9, 21, 43, 77]. Such solutions not only require high bandwidth at all times, but also *cannot be extended to the heterogeneous global telephone network*. Additionally, they are susceptible to man-in-the-middle attacks [28, 63] and are difficult to use [25, 51, 61, 64]. Tu et al. have described how to modify SS7, the core telephony signaling protocol, to support authenticated Caller ID [72]. This protocol is not end-to-end (so the protocol is vulnerable to malicious network endpoints like IMSI-catchers [23, 24]), requires both endpoints to call from an SS7-speaking network, and most importantly would also require modifying core network entities throughout every network.

The solution closest to our own is Authloop [56]. Authloop uses a codec agnostic modem and a TLS-inspired protocol to perform authentication solely over the audio channel. While Authloop provides end-to-end authentication for heterogeneous phone calls, it has a number of limitations compared to AuthentiCall. The constrained bandwidth of the audio channel means that the handshakes are slow (requiring 9 seconds on average), authentication is one-sided, and content authentication is not possible. While Authloop can prevent some forms of man-in-the-middle attacks, it is vulnerable to attacks that replace content. Finally, because Authloop relies on the audio channel, users must answer all calls before they can be authenticated. AuthentiCall overcomes all of these limitations.

10 Conclusion

Telephone networks fail to provide even the most basic guarantees about identity. AuthentiCall protects voice calls made over traditional telephone networks by leveraging now-common data connections available to call endpoints. AuthentiCall cryptographically authenticates both call parties while only adding a worst case 1.4 seconds of overhead to the call setup time. Unlike other solutions that use voice calls, AuthentiCall also protects the content of the voice call with high accuracy. In so doing, AuthentiCall offers a solution to the constant onslaught of illicit or fraudulent bulk calls plaguing the telephone network.

11 Acknowledgments

The authors thank our shepherd, Adam Doupé, and our anonymous reviewers for their helpful comments. Gaby Garcia, Zlatko Najdenovski, and Arthur Shlain created icons used in Figure 12, licensed under the Creative Commons 3.0 License. This work was supported in part by the US National Science Foundation under grant numbers CNS-1617474 and CNS-1564446. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] Bouncy Castle. http://www.bouncycastle.org/.

[2] RedPhone :: Private Calls - Android Apps on Google Play. https://play.google.com/store/apps/details?id=com.littlebytesofpi.linphonesip&hl=en.

[3] Average call. https://www.statista.com/statistics/185828/average-local-mobile-wireless-call-length-in-the-united-states-since-1987/, 2012.

[4] Circuit-Switched Fallback: The First Phase of Voice Evolution for Mobile LTE Devices. Technical report, Qualcomm, 2012.

[5] Sox. http://sox.sourceforge.net/Main/HomePage, 2013.

[6] Finally! No more annoying Robocalls and Telemarketers. http://www.nomorobo.com/, 2016.

[7] Letsencrypt. https://letsencrypt.org/, 2016.

[8] Teen's iPhone Hack Gets Him Arrested for Unleashing DDoS on 911 System. https://www.neowin.net/news/teens-iphone-hack-gets-him-arrested-for-unleashing-ddos-on-911-system, 2016.

[9] The Zfone Project. http://zfoneproject.com/, 2016.

[10] Adobe Project VoCo. http://www.vocobeta.com/, 2017.

[11] Lyrebird. http://lyrebird.ai/, 2017.

[12] 3rd Generation Partnership Project. 3GPP TS 45.005 v12.1.0. Technical Report Radio transmission and reception. ftp://www.3gpp.org/tsg_ran/TSG_RAN/TSGR_17/Docs/PDF/RP-020661.pdf.

[13] D. Akhawe and A. P. Felt. Alice in Warningland: A Large-scale Field Study of Browser Security Warning Effectiveness. In *Proceedings of the 22nd USENIX Security Symposium*, 2013.

[14] F. Alegre, G. Soldi, and N. Evans. Evasion and obfuscation in automatic speaker verification. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 749–753, 2014.

[15] S. Alfonsi. Hacking Your Phone. 60 Minutes. `http://www.cbsnews.com/news/60-minutes-hacking-your-phone/`, 2016.

[16] C. Amrutkar, P. Traynor, and P. van Oorschot. An Empirical Evaluation of Security Indicators in Mobile Web Browsers. *IEEE Transactions on Mobile Computing (TMC)*, 14(5):889–903, 2015.

[17] V. Balasubramaniyan, A. Poonawalla, M. Ahamad, M. Hunter, and P. Traynor. PinDr0p: Using Single-Ended Audio Features to Determine Call Provenance. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2010.

[18] E. Barkan, E. Biham, and N. Keller. Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication. *Journal of Cryptology*, 21(3):392–429, 2008.

[19] R. Baumann, S. Cavin, and S. Schmid. Voice Over IP-Security and SPIT. *Swiss Army, FU Br*, 41:1–34, 2006.

[20] B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *Proceedings of the 14th IEEE Workshop on Computer Security Foundations*, 2001.

[21] R. Bresciani, S. Superiore, S. Anna, and I. Pisa. The ZRTP Protocol Security Considerations. Technical Report LSV-07-20, 2007.

[22] S. T. Chow, C. Gustave, and D. Vinokurov. Authenticating Displayed Names in Telephony. *Bell Labs Technical Journal*, 14(1):267–282, 2009.

[23] A. Dabrowski, G. Petzl, and E. Weippl. The Messenger Shoots Back: Network Operator Based IMSI Catcher Detection. In *19th International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2016)*, 2016.

[24] A. Dabrowski, N. Pianta, T. Klepp, M. Schmiedecker, and E. Weippl. IMSI-Catch Me If You Can: IMSI-Catcher-Catchers. In *Annual Computer Security Applications Conference (ACSAC)*, 2014.

[25] S. Egelman, L. F. Cranor, and J. Hong. You've Been Warned: An Empirical Study of the Effectiveness of Web Browser Phishing Warnings. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2008.

[26] W. Enck, P. Traynor, P. McDaniel, and T. La Porta. Exploiting Open Functionality in SMS-Capable Cellular Networks. In *Proceedings of the 12th ACM conference on Computer and communications security*, pages 393–404. ACM, 2005.

[27] U. Equivox. Speaker recognition–Part 1. *Biometric Technology Today*, page 10, 2004.

[28] N. Evans, F. Alegre, Z. Wu, and T. Kinnunen. Anti-spoofing, Voice Conversion. *Encyclopedia of Biometrics*, pages 115–122, 2015.

[29] Federal Emergency Management Agency. Call Toll-Free Number For Disaster Assistance. `https://www.fema.gov/news-release/2003/09/25/call-toll-free-number-disaster-assistance`, 2003.

[30] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett. DARPA TIMIT Acoustic-Phonetic Continous Speech Corpus CD-ROM. NIST speech disc 1-1.1. *NASA STI/Recon technical report n*, 93, 1993.

[31] S. Heuser, B. Reaves, P. K. Pendyala, H. Carter, A. Dmitrienko, W. Enck, N. Kiyavash, A.-R. Sadeghi, and P. Traynor. Phonion: Practical Protection of Metadata in Telephony Networks. *Proceedings of Privacy Enhancing Technologies*, 2017(1), July 2017.

[32] O. Hohlfeld, R. Geib, and G. Haßlinger. Packet Loss in Real-time Services: Markovian Models Menerating QoE Impairments. In *Quality of Service, 2008. IWQoS 2008. 16th International Workshop on*, pages 239–248. IEEE, 2008.

[33] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar. Adversarial Machine Learning. In *Proceedings of the ACM Workshop on Security and Artificial Intelligence*, 2011.

[34] M. Huffman. Survey: 11% of adults lost money to a phone scam last year. Consumer Affairs - `https://www.consumeraffairs.com/news/survey-11-of-adults-lost-money-to-a-phone-scam-last-year-012616.html`, 2016.

[35] N. Jiang, Y. Jin, A. Skudlark, W.-L. Hsu, G. Jacobson, S. Prakasam, and Z.-L. Zhang. Isolating and Analyzing Fraud Activities in a Large Cellular Network Via Voice Call Graph Analysis. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2012.

[36] Y. Jiao, L. Ji, and X. Niu. Robust Speech Hashing for Content Authentication. *IEEE Signal Processing Letters*, 16(9):818–821, 2009.

[37] Q. Jin, A. R. Toth, A. W. Black, and T. Schultz. Is Voice Transformation a Threat to Speaker Identification? In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2008.

[38] T. Kinnunen, Z.-Z. Wu, K. A. Lee, F. Sedlak, E. S. Chng, and H. Li. Vulnerability of Speaker Verification Systems against Voice Conversion Spoofing Attacks: The Case of Telephone Speech. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4401–4404. IEEE, 2012.

[39] H. Krawczyk and P. Eronen. HMAC-based Extract-and-Expand Key Derivation Function (HKDF). Technical report, 2010.

[40] P. P. Lee, T. Bu, and T. Woo. On the detection of signaling DoS attacks on 3G/WiMax wireless networks. *Computer Networks*, 53(15):2601–2616, 2009.

[41] M. Lepinski and S. Kent. Additional Diffie-Hellman Groups for Use with IETF Standards. RFC 5114, RFC Editor, January 2008.

[42] C.-Y. Li, G.-H. Tu, C. Peng, Z. Yuan, Y. Li, S. Lu, and X. Wang. Insecurity of Voice Solution VoLTE in LTE Mobile Networks. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, New York, NY, USA, 2015.

[43] T. H. A. C. Liath and R. Bresciani. The ZRTP Protocol Analysis on the Diffie-Hellman Mode. *Foundations and Methods Research Group*, 2009.

[44] J. Lindqvist and M. Komu. Cure for Spam over Internet Telephony. In *4TH IEEE Consumer Communications And Networking Conference (CCNC 2007)*, 2007.

[45] B. Mathieu, S. Niccolini, and D. Sisalem. SDRS: A Voice-over-IP Spam Detection and Reaction System. *IEEE Security & Privacy Magazine*, 6(6):52–59, Nov. 2008.

[46] N. Miramirkhani, O. Starov, and N. Nikiforakis. Dial One for Scam: A Large-Scale Analysis of Technical Support Scams. In *Proceedings of the 24th Network and Distributed System Security Symposium (NDSS)*, 2017.

[47] H. Mustafa, W. Xu, A. R. Sadeghi, and S. Schulz. You Can Call but You Can't Hide: Detecting Caller ID Spoofing Attacks. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 168–179, 2014.

[48] S. Mustafa, H. and Wenyuan Xu and Sadeghi, A.R. and Schulz. You Can SPIT, but You Can't Hide: Spammer Identification in Telephony Networks. In *2011 Proceedings IEEE INFOCOM*, pages 41–45, 2011.

[49] N. Papernot, P. McDaniel, M. F. Somesh Jha, Z. B. Celik, and A. Swami. The Limitations of Deep Learning in Adversarial Settings. In *Proceedings of the 1st IEEE European Symposium on Security and Privacy (Euro S&P)*, 2016.

[50] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swam. Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2016.

[51] M. Petraschek, T. Hoeher, O. Jung, H. Hlavacs, and W. Gansterer. Security and usability aspects of Man-in-the-Middle attacks on ZRTP. *Journal of Universal Computer Science*, (5):673–692.

[52] S. Phithakkitnukoon and R. Dantu. Defense against SPIT using community signals. *Intelligence and Security Informatics, 2009. ISI '09. IEEE International Conference on*, 2009.

[53] R. Pries, T. Hobfeld, and P. Tran-Gia. On the suitability of the short message service for emergency warning systems. In *2006 IEEE 63rd Vehicular Technology Conference*, volume 2, pages 991–995. IEEE, 2006.

[54] A. Ramirez. Theft Through Cellular "Clone" Calls. `http://www.nytimes.com/1992/04/07/business/theft-through-cellular-clone-calls.html`, April 7, 1992.

[55] B. Reaves, L. Blue, H. Abdullah, L. Vargas, P. Traynor, and T. Shrimpton. AuthentiCall: Efficient Identity and Content Authentication for Phone Calls. Technical Report FICS-TR-2017-0001, Florida Institute for Cybersecurity Research, University of Florida, Gainesville, FL, June 2017.

[56] B. Reaves, L. Blue, and P. Traynor. AuthLoop: End-to-End Cryptographic Authentication for Telephony over Voice Channels. *Proceedings of the 25th USENIX Security Symposium*, Aug. 2016.

[57] B. Reaves, E. Shernan, A. Bates, H. Carter, and P. Traynor. Boxed Out: Blocking Cellular Interconnect Bypass Fraud at the Network Edge. In *Proceedings of the 24th USENIX Security Symposium*, 2015.

[58] S. Rosset, U. Murad, E. Neumann, Y. Idan, and G. Pinkas. Discovery of Fraud Rules for Telecommunications-Challenges and Solutions. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 409–413, New York, NY, USA, 1999.

[59] M. Sahin and A. Francillon. Over-The-Top Bypass: Study of a Recent Telephony Fraud. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1106–1117. ACM, 2016.

[60] M. Sahin, A. Francillon, P. Gupta, and M. Ahamad. Sok: Fraud in telephony networks. In *Proceedings of the 2nd IEEE European Symposium on Security and Privacy*, 2017.

[61] S. E. Schechter, R. Dhamija, A. Ozment, and I. Fischer. The Emperor's New Security Indicators. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2007.

[62] J. Serror, H. Zang, and J. C. Bolot. Impact of Paging Channel Overloads or Attacks on a Cellular Network. In *Proceedings of the 5th ACM Workshop on Wireless Security*. Citeseer, 2006.

[63] M. Shirvanian and N. Saxena. Wiretapping via Mimicry: Short Voice Imitation Man-in-the-Middle Attacks on Crypto Phones. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 868 – 879, 2014.

[64] M. Shirvanian and N. Saxena. On the Security and Usability of Crypto Phones. In *Proceedings of the 31st Annual Computer Security Applications Conference*, pages 21–30, 2015.

[65] Statista. Average monthly outbound minutes. `https://www.statista.com/statistics/273902/average-monthly-outbound-mobile-voice-minutes-per-person-in-the-uk/`, 2013.

[66] Y. Stylianou. Voice Transformation: A Survey. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009.

[67] O. Tange et al. Gnu parallel-the command-line power tool. *;login: The USENIX Magazine: Volume 36, Number 1*, 2011.

[68] TrapCall. `https://www.trapcall.com/`, 2016.

[69] P. Traynor, P. McDaniel, and T. La Porta. *Security for Telecommunications Networks*. Number 978-0-387-72441-6 in Advances in Information Security Series. Springer, August 2008.

[70] G.-H. Tu, C.-Y. Li, C. Peng, Y. Li, and S. Lu. New Security Threats Caused by IMS-based SMS Service in 4G LTE Networks. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[71] H. Tu, A. Doupé, Z. Zhao, and G.-J. Ahn. SoK: Everyone Hates Robocalls: A Survey of Techniques against Telephone Spam. *2016 IEEE Symposium on Security and Privacy (S&P)*, 2016.

[72] H. Tu, A. Doupé, Z. Zhao, and G.-J. Ahn. Toward Authenticated Caller ID Transmission: The Need for a Standardized Authentication Scheme in Q.731.3 Calling Line Identification Presentation. In *Proceedings of the ITU Kaleidoscope (ITU)*, Nov. 2016.

[73] D. Tynan. The terror of swatting: how the law is tracking down high-tech prank callers. `https://www.theguardian.com/technology/2016/apr/15/swatting-law-teens-anonymous-prank-call-police`, Apr 2016.

[74] Vassilis Prevelakis and Diomidis Spinellis. The Athens Affair. *IEEE Spectrum*, June 2007.

[75] X. Wang and R. Zhang. VoIP Security: Vulnerabilities, Exploits and Defenses. *Elsevier Advances in Computers*, March 2011.

[76] A. Whitten and J. D. Tygar. Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. In *25th USENIX Security Symposium (USENIX Security 16)*, 1999.

[77] P. Zimmermann, A. Johnston, and J. Callas. RFC 6189 ZRTP: Media Path Key Agreement for Unicast Secure RTP. *Internet Engineering Task Force*, 2011.

A RSH Digest Construction

In this appendix, we describe the construction of the RSH digest used by AuthentiCall for channel binding and content integrity.

There are a number of constructions of speech digests, and they all use the following basic process. First, they compute derived features of speech. Second, they define a compression function to turn the real-valued features into a bit string. In this paper, we use the construction of Jiao et al. [36], which they call RSH. We chose this technique over others because it provides good performance on speech at a low-bitrate, among other properties. We note that the original work did not evaluate the critical case where an adversary can control the audio being hashed. Our evaluation shows that RSH maintains audio integrity in this crucial case. The construction also selects audio probabilistically; we show in Appendix B that the digest indeed protects all of the semantic content in the input audio. Finally, to our knowledge we are the first to use any robust speech digest for an authentication and integrity scheme.

Figure 13 illustrates how RSH computes a 512-bit digest for one second of audio. In the first step of calculating a digest, RSH computes the Line Spectral Frequencies (LSFs) of the input audio. LSFs are used in speech compression algorithms to represent the major frequency components of human voice, which contain the majority of *semantic* information in speech.

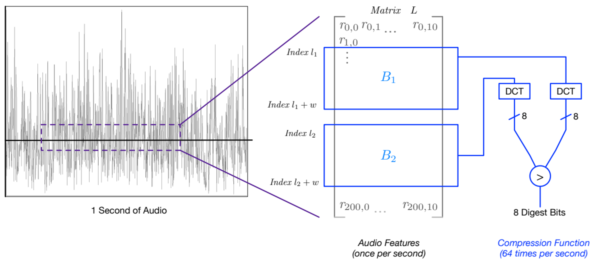

1 Second of Audio
Matrix L
$r_{0,0}$ $r_{0,1}$ … $r_{0,10}$
$r_{1,0}$
Index l_1
Index $l_1 + w$
Index l_2
Index $l_2 + w$
B_1
B_2
$r_{200,0}$ … $r_{200,10}$
DCT
DCT
8
8
>
8 Digest Bits
Audio Features (once per second)
Compression Function (64 times per second)

Figure 13: This figure illustrates the digest construction described in Section 5.1. Audio digests summarize call content by taking one second of speech data, deriving audio features from the data, and compressing blocks of those features into a bit string.

That is, LSFs represent phonemes – the individual sound units present in speech. While pitch is useful for speaker recognition, LSFs are not a perfect representation of all of the nuances of human voice. This is one reason why it is sometimes difficult for humans to confidently recognize voices over the phone. This means that the digest more accurately represents *semantic* content rather than the speaker's voice characteristics. This is important because a number of techniques are able to synthesize new speech that evades speaker recognition from existing voice samples [38,66]. Finally, LSFs are numerically stable and robust to quantization — meaning that modest changes in input yield small changes in output. In RSH, the input audio is grouped into 30ms frames with 25ms audio overlap between frames, and 10 line spectral frequencies are computed for each frame to create a matrix L.

The second phase of digest computation involves compressing the large amount of information about the audio into a digest. Because audio rarely changes on millisecond time scales, the representation L is highly redundant. To compress this redundant data, RSH uses the two-dimensional discrete cosine transform (DCT). The DCT is related to the Fourier transform, is computationally efficient, and is commonly used in compression algorithms (e.g., JPEG, MP3). RSH computes the DCT over different sections of the matrix L to produce the final digest. RSH only uses first eight DCT coefficients (corresponding to the highest energy components and discarding high-frequency information).

The second phase of digest computation – the compression function – uses the DCT algorithm in the computation of the bitwise representation of the audio sample. The following process generates 8 bits of a digest; it is repeated 64 times to generate a 512-bit digest.

1. Obtain a window size w and two window start indexes l_1 and l_2 from the output of a keyed pseudorandom function.
2. Select from L two blocks of rows. These blocks B_1 and B_2 contain all columns from $l_1 : l_1 + w$ and $l_2 : l_2 + w$ respectively.
3. Compress these individual blocks into eight coefficients each using the DCT.
4. Set eight digest bits by whether the corresponding coefficients of the first block (B_1) are greater than the coefficients

of the second block (B_2).

We note that sections of audio are selected probabilistically; we show in Appendix B that the probability that a section of audio is not used in a digest is negligible.

An important consideration is to note that the digest is *keyed*. By using a keyed pseudorandom function, repeated phrases generate verifiable unique digests. It also has the advantage that it makes it difficult to compute digests for audio without knowledge of the key, which in AuthentiCall is derived during the handshake for each call. In AuthentiCall, digests themselves are also authenticated using an HMAC to guarantee digest integrity in transit.

Digests of spoken audio are sent by both parties. The verifying party computes the digest of the received audio, then computes the hamming distance between the calculated and received digests. Because degradation of audio over a phone call is expected, digests will not match exactly. However, the Hamming distance between two audio digests — or bit error rate (BER) — is related to the amount of change in the audio. By setting an appropriate threshold on BER, legitimate audio can be distinguished from incorrect audio.

B Probabilistic Analysis of Robust Hashing

AuthentiCall uses the RSH speech digest algorithm [36], which probabilistically selects sections of audio for inclusion. The initial research did not establish whether all audio was included in every hash. In this appendix, we bound the probability that one or more 5ms sections of audio (which are individual rows in the matrix L) are not included. The analysis shows that it is possible for a few milliseconds of audio to be excluded – less than 25 milliseconds of audio. This is less than an individual phoneme, could not change semantic meaning of the audio, and losses of 25 milliseconds or more are common in audio transmission and typically go unnoticed by users. Accordingly, the digests effectively cover call content.

Fix an even integer $N > 0$, and fix a block width $w \in [2..N/2]$. Let $r \in [1..N]$ be a row index of the matrix L. We begin by computing the probability that in any particular trial, the r-th row is *not* covered by at least one of the two blocks B_1, B_2 used in the robust hashing algorithm. Recall that the "top" row of B_1 and B_2 are randomly selected each trial. Thus, let ℓ_1, ℓ_2 be uniform integers in the range $[1..N+1-w]$.

Let $X_r^{(i)}$ be an indicator random variable for the event that row f is covered by at least one of these blocks in the i-th trial. Then we observe that $X_r^{(i)} = 0$ iff the event $r \notin [\ell_1..\ell_1+w-1] \wedge r \notin [\ell_2..\ell_2+w-1]$ occurs. We have

$$\begin{aligned}\Pr\left[X_r^{(i)} = 0\right] &= \Pr\left[r \notin [\ell_1..\ell_1+w-1]\right] \cdot \\ &\quad \Pr\left[r \notin [\ell_2..\ell_2+w-1]\right] \\ &= \Pr\left[r \notin [\ell_1..\ell_1+w-1]\right]^2\end{aligned}$$

since ℓ_1, ℓ_2 are independent and identically distributed. There are two cases to consider. When $r \in [1..w-1]$ we have

$$\Pr\left[X_r^{(i)} = 0\right] = \left(1 - \frac{r}{N+1-w}\right)^2 \leq e^{-2r/(N+1-w)}$$

because there are only r values for ℓ_1 (resp. ℓ_2) that cause block B_1 (resp. B_2) to include the r-th row of L. When $r \geq w$, which is the common case when $N \gg w$, we have

$$\Pr\left[X_r^{(i)} = 0\right] = \left(1 - \frac{w}{N+1-w}\right)^2 \leq e^{-2w/(N+1-w)}.$$

To build some intuition for these probabilities, take $N = 200$ and $w = 51$ (the average value if w were selected uniformly from its range), $\Pr\left[X_1^{(i)} = 0\right] \leq 0.98$, i.e., the first row is almost certainly not covered in any particular trial. But this quickly decreases as r grows, and when $r = w$ (and beyond) we have $\Pr\left[X_r^{(i)} = 0\right] \leq 0.51$. Keep in mind that the robust hashing algorithm runs $t = 64$ independent trials, thus, defining the indicator $X_r = 1$ iff $\bigvee_{i=1}^{t} X_r^{(i)} = 1$, we have

$$\Pr\left[X_r = 0\right] \leq \begin{cases} e^{-t(2r)/(N+1-w)} & \text{when } r < w \\ e^{-t(2w)/(N+1-w)} & \text{when } r \geq w \end{cases}$$

Thus $\Pr[X_1 = 0] \leq (0.98)^{64} \approx 0.43$, and for $r \geq w$ we have $\Pr[X_r = 0] \leq (0.51)^{64} \approx 2^{-64}$. It is apparent that the first few rows of L are unlikely to be covered, but that the remaining rows are covered in some trial with overwhelming probability.

Continuing, let $X = \sum_{r=1}^{N} X_r$, i.e., the number of rows covered across all t trials. Additionally, let W be a uniform value in $[2..N/2]$. (Recall that in the robust hashing algorithm, the parameter w is chosen this way for each trial.) By linearity of expectation we have

$$\begin{aligned}\mathbb{E}[X \mid W = w] &= \sum_{r=1}^{N} \mathbb{E}[X_r \mid W = w] \\ &= \sum_{r=1}^{N} \Pr[X_r = 1] \\ &= N - \sum_{r=1}^{N} \Pr[X_r = 0] \\ &= N - \left(\sum_{r=1}^{w-1} \Pr[X_r = 0] + \sum_{r=w}^{N} \Pr[X_r = 0]\right) \\ &\geq N - \left(\sum_{r=1}^{w-1} e^{-t(2r)/(N+1-w)}\right. \\ &\qquad \left. + (N+1-w)e^{-t(2w)/(N+1-w)}\right)\end{aligned}$$

Again, when $N = 200, w = 51, t = 64$, we have $\mathbb{E}[X \mid W = 51] \geq 198.4$; on average, the number of rows missed is less than two. Finally, we define $f(w) = \mathbb{E}[X \mid W = w]$ and consider $\mathbb{E}[f(W)]$, which is the average number of rows covered over random choices of block width and block-starting rows. When $N = 200, t = 64$ we have $\mathbb{E}[f(W)] \geq 195.4$; thus fewer than five rows are completely missed, on average, across all trials. With overwhelming probability, it will be the first few rows that are missed. As we discussed at the beginning of this section, this audio would could not affect the semantics of the transmitted speech.