SMS OTP Security (SOS): Hardening SMS-Based Two Factor Authentication

Christian Peeters
University of Florida
Gainesville, Florida, USA
cpeeters@ufl.edu

Christopher Patton
University of Florida
Gainesville, Florida, USA
cjpatton@ufl.edu

Imani N. S. Munyaka
University of Florida
Gainesville, Florida, USA
shermani@ufl.edu

Daniel Olszewski
University of Florida
Gainesville, Florida, USA
dolszewski@ufl.edu

Thomas Shrimpton
University of Florida
Gainesville, Florida, USA
teshrim@ufl.edu

Patrick Traynor
University of Florida
Gainesville, Florida, USA
traynor@ufl.edu

ABSTRACT

SMS-based two-factor authentication (2FA) is the most widely deployed 2FA mechanism, despite the fact that SMS messages are known to be vulnerable to rerouting attacks, and despite the availability of alternatives that may be more secure. This is for two reasons. First, it is very effective in practice, as evidenced by reports from Google and Microsoft. Second, users prefer SMS over alternatives, because text messaging is already part of their daily communication. Accepting this practical reality, we developed a new SMS-based protocol that makes rerouting attacks useless to adversaries who aim to take over user accounts. Our protocol delivers one-time passwords (OTP) via text message in a manner that adds minimal overhead (to both the user and the server) over existing SMS-based methods, and is implemented with only small changes to the stock text-message applications that already ship on mobile phones. The security of our protocol rests upon a provably secure authenticated key exchange protocol that, crucially, does not place significant new burdens upon the user. Indeed, we carry out a user study that demonstrates no statistically significant difference between traditional SMS and our protocol, in terms of usability.

CCS CONCEPTS

• **Security and privacy → Formal security models**; **Usability in security and privacy**; **Mobile and wireless security**; **Multi-factor authentication**.

KEYWORDS

security;two-factor authentication;cellular;SMS;authenticated key-exchange

ACM Reference Format:
Christian Peeters, Christopher Patton, Imani N. S. Munyaka, Daniel Olszewski, Thomas Shrimpton, and Patrick Traynor. 2022. SMS OTP Security (SOS): Hardening SMS-Based Two Factor Authentication. In *Proceedings of the 2022 ACM Asia Conference on Computer and Communications Security (ASIA CCS '22), May 30–June 3, 2022, Nagasaki, Japan.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3488932.3497756

ASIA CCS '22, May 30–June 3, 2022, Nagasaki, Japan.

ACM ISBN 978-1-4503-9140-5/22/05…$15.00
https://doi.org/10.1145/3488932.3497756

1 INTRODUCTION

Text messaging is by far the most widely used communications technology, with more than 5 billion users sending over 8.3 trillion messages in 2017 [51]. Its pervasiveness and the binding of phone numbers to user identities have lead to many services being built on top of SMS. With relation to security, these services include account validation, anomaly reporting, and one-time passwords (OTPs) for two-factor authentication (2FA) [30, 71].

Despite the widespread belief that SMS provides a secure channel for communication, this is not the case. Adversaries have been able to use *redirection attacks* (i.e., SS7 redirection, SIM Swap) to intercept SMS messages and, with the (possible) exception of the last-mile link between the tower and cellular device, text messages are unencrypted. Attackers have moved funds from traditional financial entities [15], drained cryptocurrency wallets [32], and enabled spying on human rights advocates [38]. Multiple governmental bodies, including the National Institute of Standards and Technology (NIST) in the United States and the National Cyber Security Centre (NCSC) in the United Kingdom, have publicly recommended that SMS no longer be used to deliver OTPs.

These recommendations are sensible in theory, but they stand-in opposition to the momentum of real-world circumstances. In particular, SMS-based 2FA is the most widely used method of 2FA [27, 34], with many large websites (e.g., Venmo, US Social Security Administration, Samsung, and more) offering no other options for alternative authentication factors [6]. Users consistently pick SMS-based 2FA because of its usability and the associated comfort with text messaging [26, 82], even when offered demonstrably more secure options, such as hardware or software tokens (e.g., YubiKeys, Authy).[1] Additionally, in the developing world, SMS is the only realistic service with a sufficient user base to support significant 2FA deployment [63].

[1] Moreover, many of these solutions fall back to SMS-based 2FA when these tokens become unavailable, meaning that an OTP redirection resilient technique remains necessary.

Finally, both Google [45] and Microsoft [25] argue that SMS-based 2FA works well, with account takeovers virtually disappearing when compared to accounts without 2FA. Simply put, the preponderance of evidence suggests that SMS-based 2FA will remain dominant for the foreseeable future. Our position accepts this reality, and works to make the users' method of choice less vulnerable.

Contributions. In this paper, we develop a protocol, intended to be deployed as an add on to default messaging applications, to allow users to continue using SMS-based 2FA for OTPs while significantly improving its security.

- **Mitigate SMS Redirection Attacks:** We design and implement a protocol to protect OTPs sent via SMS from interception. Rather than send the OTP itself, the server sends a nonce from which the mobile device derives the OTP, using a secret key shared with the server (Similar to the HMAC-based one-time password (HOTP) protocol [52] that underlies a variety of hardware and software tokens.) To securely establish the shared secret, we propose and prove secure a new and novel authenticated key-exchange (AKE) scheme, which leverages existing communications and public-key infrastructure as-is and requires no trusted setup of the mobile device. These properties are made possible by the user, who intervenes during registration to pass messages between the mobile device and the server. This process closely mimics the usual registration procedure for SMS-based 2FA; the only difference is that the user is required to scan a QR code displayed on their web client (i.e., a browser) using their mobile device's camera. Our protocol also works when the web client is run on the phone, in which case this step is not needed.
- **Demonstrate Consistent Usability:** We conduct a user study to compare the usability of an implementation of our protocol to that of traditional SMS-based 2FA and show that our protocol exhibits no statistically significant difference within our user study.
- **Discuss Deployment Challenges:** We carefully consider many of the challenges of deploying such technology, including the asynchronous nature of text messaging and calling plans with pay-per-message requirements. We note that related proposals such as Messaging Layer Security (MLS) [19] fail to achieve these ends as they assume IP connections and require large numbers of messages.

We refer to our substitute for traditional SMS-based 2FA as the *SOS protocol*. "SOS" stands for "SMS OTP Security".

The remainder of this paper is organized as follows: Section 2 provides important background information; Section 3 describes the SOS protocol as well as the underlying design constraints and rationale; Section 4 describes our security goals and the formal security model; Section 5 provides implementation and user study details; Section 6 shows experimental results; Section 7 discusses deployment and other relevant issues; Section 8 enumerates related works; Section 9 provides concluding remarks; and the Appendix contains our formal specification and provable security treatment of our novel AKE protocol.

2 BACKGROUND

2.1 Two-Factor Authentication Mechanisms

The concept of two-factor authentication for computing can be seen as far back as the early 1980's [77], and since then many variants of 2FA have been demonstrated. Though the combination of authentication methods may differ, most common implementations of 2FA require a traditional password in conjunction with a second alternative authentication method. There have been many proposed forms of 2FA[2], including biometrics [3], hardware tokens [14], software tokens [2, 5], additional knowledge factors [8], device characteristics [16, 56], both current and historic location information [9], and phone numbers [12]. Despite this variety of possibilities, studies [42] and usage data [44] have shown that a majority of the current 2FA market consists of three variants: hardware tokens, software tokens, and SMS-based 2FA.

Although hardware tokens were the first to be adopted, SMS-based 2FA began seeing a rise in popularity due to the ubiquitous nature of mobile devices [42]. SMS-based 2FA came into use before the popularity of smartphones, which allowed it to become widely deployed before the introduction of software tokens and modern hardware tokens [42, 66]. SMS-based 2FA can be broken down into a two-phase protocol. The *registration phase* allows a user to associate their phone number with an account for a website or system. Once the user provides their phone number, the server sends an SMS message containing a randomly chosen one-time password (OTP), with a short validity period, using an IP-to-cellular gateway. The browser prompts the user to enter the OTP once the message is received. If the input matches what the server sent within the validity period, registration completes and the phone number is associated with a user's account for future authentication.

The *authentication phase* occurs during every subsequent login. As in the registration phase, an OTP is sent via SMS from the server to the device associated with a user's phone number. Authentication is complete once the user types the correct OTP in the web client and the server verifies it.

Since the initial deployment of SMS-based 2FA, both software and hardware tokens have continued to see a raise in popularity. Efforts have been made to push these into becoming the modern standard for 2FA due to the security benefits over SMS-based 2FA. However, despite the efforts at improving the usability of both types of tokens [16, 75, 81], they still trail SMS-based 2FA by a significant margin in terms of user preference and adoption [27, 44].

Indeed, there are many usability factors that may contribute to the lack of user interest in alternatives to SMS, which we further discuss in Section 8. Both hardware and software tokens require users to adopt a new and non-standardized technology which can be complicated without a technology background. The complexity of the alternative systems has even resulted in websites offering SMS-based 2FA as a backup, which eliminates any security advantages gained from using other 2FA methods[3]. Doing away with the most

[2]Throughout this paper we refer to the second authentication method as a form, type, and method of 2FA. It is implied that the first method of authentication implemented by these techniques is a traditional password.

[3]At the time of writing, this functionality is provided by Google and is defaultly enabled on accounts using 2FA that also have a registered phone number.

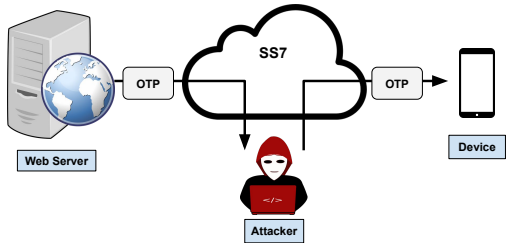


Figure 1: By controlling a node in the SS7 network, an attacker is able to reroute traffic. This includes rerouting 2FA OTP messages sent over SMS to themselves in plaintext. The attacker can then forward the message to the intended user in efforts to make them unaware of the attack.

preferred and familiar method of 2FA [26, 34] further discourages user adoption and continued use of any 2FA method whatsoever.

2.2 Rerouting Attacks on SMS

Although attacks on SMS-based 2FA have become more prevalent, vulnerabilities in SMS have been known for years [53]. There are two common attacks on SMS-based 2FA: message rerouting via Signaling System 7 (SS7) and what are commonly referred to as *SIM Swap* attacks. The former exploits weaknesses in legacy cellular technology, while the latter capitalizes on the lack of customer authentication provided by telecommunications companies.

2.2.1 Routing in Telecommunications. Telecommunications networks are a conglomeration of interconnected communication network technologies which include cellular, landline, and VoIP. The underlying protocols responsible for standard operations are unique to each network technology. In order to communicate with one another, telecommunications networks use the protocol suite known as SS7 [73]. This standard suite of protocols forms an all-digital network that is responsible for both signaling between core elements within a network and communication between different networks.

SS7 was built with weak security assumptions that allow adversaries to purchase access to the SS7 network core. This permits a variety of attacks, including SMS interception [1] that is outlined in Figure 1. Through this it is possible for attackers to recover 2FA OTPs [18, 32], which has resulted in compromised bank accounts [18], email addresses [32], and cryptocurrency wallets [55]. Though it is hard to quantify the regularity of SS7-based attacks, there is evidence that they are becoming more common [28, 32] and are exhibiting no signs of slowing down [72].

For years it has been common knowledge that SS7 networks possess an array of vulnerabilities [46, 50], but little has been done to address them. Proposed defensive mechanisms against SS7 based attacks primarily focus on network layer filtering [31, 37], but are largely ineffective [29]. A device level defense was proposed by researchers in 2018 [60], but only for call rerouting via SS7. Replacing SS7 has also been attempted through its packet-based anticipated successor called Diameter [11], but has shown to be ineffective due to Diameter being backwards compatable with SS7 [70].

Moreover, network providers have begun encapsulating SMS messaging in Session Initiation Protocol (SIP) messages. These messages are then transmitted over IP Multimedia Subsystem (IMS) core networks, which has increased the difficulty of intercepting

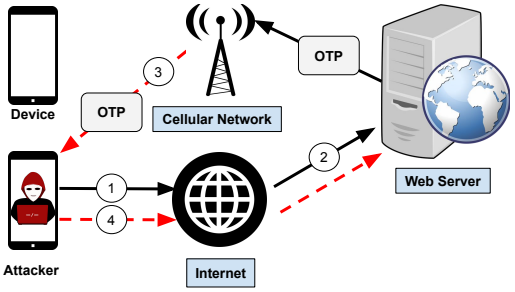


Figure 2: In a SIM Swap attack, an adversary obtains a replacement SIM card for a targeted user through a cellular network provider. This allows an attacker to receive all of a user's SMS messages, including 2FA OTP messages.

SMS messages containing OTPs, as messages can be encrypted using TLS. However, this is not an end-all solution to SMS rerouting attacks. This technique only works with 4G and 5G networks, and offers nothing to users that do not have access to modern cellular technologies. Additionally, adversaries can still successfully exploit existing vulnerabilities in both IMS and SIP, as well as execute out-of-band rerouting attacks such as a step-down attack with a rogue base station or a SIM Swap attack, which we discuss next.

2.2.2 SIM Swap Attacks. SIM Swap attacks [41] exploit the lack of customer authentication involved with replacing a lost or damaged Subscriber Identity Module (SIM) card. These cards contain the symmetric key used for device authentication with the network. Shortly after a replacement SIM card is activated, the original is deactivated. Often, new SIM cards can be acquired free of charge and with minimal identification. SIM Swap attacks have demonstrated to be an effective mechanism of compromising 2FA OTP messages for specific targets, and on multiple occasions for the same individual [62].

An attacker can execute SIM Swap attacks, visualized in Figure 2, by assuming the identity of a customer, or by contracting the assistance of an entry level employee working at one of the cell provider's stores [54]. Once the new SIM card has been activated, the attacker will receive all messages for a targeted phone number, including 2FA OTP messages. The user eventually becomes aware of the problem when their device no longer connects to the network, and regaining control of the phone number can be a lengthy process. In some cases, victims have been aware of the attack occurring but were unable to stop it before the attack was complete [79].

The motivation behind SIM Swap attacks can vary. In some cases adversaries have financial incentives and will attack 2FA-protected cryptocurrency wallets [17, 79]. There have been other cases where SIM Swap attacks have targeted celebrities, CEO's of large companies, and politicians in order to hijack social media and other personal accounts [59, 62]. Research into defenses against SIM Swap attacks is limited, though some early concepts exist [48]. Recently, some network providers have begun deploying additional security measures for SIM replacement, but this is still in early stages and does not defend against attacks with insider assistance (e.g., the help of an entry level employee). Currently, there are no widely available solutions to these attacks.

3 THE SOS PROTOCOL

We now enumerate the design constraints for SOS in Section 3.1 and give an informal description our threat model in Section 3.2.

3.1 Design Constraints and Rationale

The goal of SOS is to mitigate OTP-rerouting attacks while mimicking traditional SMS-based 2FA to the user (to the maximum extent possible) in both the registration and authentication procedures. To increase the likelihood of deployment, our protocol should use the existing communication and public-key infrastructure as-is. (We further discuss our adherence to our design objectives in Section 7.)

(D1) Minimize the Number of SMS Messages. From both the usability and deployment perspectives, it is critical that we minimize the number of messages used in our protocol's operations. SMS is an asynchronous service, with no guarantees on delivery time or the order in which messages are received. Additionally, cellular networks are only provisioned to accommodate typical network conditions: significant increases in traffic beyond this can result in limited service to users in affected areas. Sending large amounts of information over multiple SMS messages may result in a significant delivery time and potentially be prone to error during message assembly. Aside from decreasing reliability, high communication overhead can also have a financial impact on both the deploying server and user. Commonly, servers that deploy SMS functionality are charged by telecommunications providers based on the number of messages sent and received. Users may also have a similar plan from their telecommunications provider [7].

(D2) Mimic SMS-based 2FA. Because users prefer it to other 2FA mechanisms [26, 27, 82], our goal is to mimic the registration and authentication phases of traditional SMS-based 2FA. In particular, any added security functionality should remain mostly hidden from users so that regardless of the steps necessary to securely deliver an OTP via SMS, it is presented to users as if no protocol changes had been made. Achieving this would require our protocol to be incorporated into default messaging applications, which would allow most of the process to go unnoticed. The device then performs the steps to obtain an OTP from a message and present it to the user in the default messaging app as if it were sent in plaintext.

(D3) Minimize Changes to Existing Infrastructure. A protocol whose operation, or security, requires a major change in telecommunication infrastructure has little hope of deployment. Thus, we aim to use the current internet and telephony networks as they are.

Likewise, to foster adoption by enterprises wishing to employ SMS-based 2FA with their users, we aim to minimize the impact of implementing our system upon authentication servers. In the traditional protocol, the server simply generates an OTP, sends it to the device, and waits for the client's response; otherwise its operation is completely stateless and there is no trusted setup of the device (i.e., distribution of certificates or symmetric keys). Because these features make traditional SMS-based 2FA easy to deploy, our goal is to minimize any overhead.

3.2 Threat Model

Traditional SMS-based 2FA is vulnerable to rerouting attacks. Regardless of how the attack is realized—a compromised SS7 node, SIM Swap attacks, or otherwise—the goal of the attack is to impersonate a valid user to an authentication server. We now give an informal description of our threat model; we leave a formal treatment to Section 4.

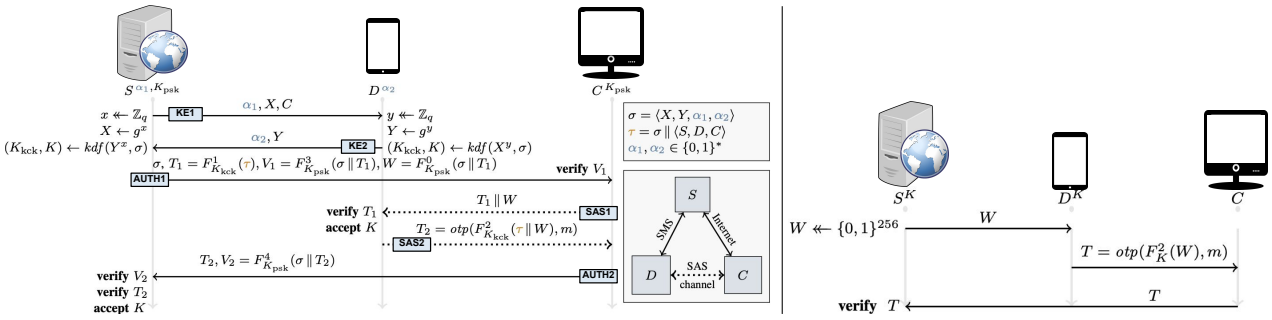
Modeling Assumptions. Typically, the first authentication factor is a password; hence impersonating a user to a server requires knowledge of the user's password and the ability to obtain the OTP. We will assume that, before the authentication phase but after registration, the adversary has obtained a user's password (e.g., by a phishing attack). We note that if the password is compromised prior to registration, there is no hope for security; the attacker can simply register itself with the password and its own phone number.

Finally, we assume that the user's mobile device is not under control of the attacker, neither by remotely controlling its software (e.g., by installing malware) nor by the theft of the physical device. Indeed, virtually all of the 2FA systems discussed in Section 8 are insecure in the presence of such an adversary. Atacks that compromise 2FA OTPs via malware are entirely different from the redirection attacks our set of protocols address. Potential solutions to those attacks have also been proposed [40].

Parties and Communication. We refer to the party requesting authentication and providing the authentication service as the "server". In practice, these may be distinct, organizationally separated entities, but we will think of them as being a single participant in the protocol. The other participants are: the "client", which is some endpoint in the Internet; and the "device", some endpoint in a cellular network. Both are under the physical control of the user. These might be co-located on the same physical machine (e.g., an Internet-capable smartphone); even so, we will think of them as being separate entities. These parties communicate as follows.

- The client and server communicate over the Internet. We assume they are capable of establishing a server-authenticated TLS session (e.g., HTTPS).
- The device and server communicate over the mobile telephony network, using only SMS messages.
- The device and client do not automatically communicate over a network; as discussed below, they communicate only by intervention on the part of the user.

Although our focus is on rerouting attacks, we will adopt a more conventional (and much stronger) threat model for network adversaries. We assume that the adversary *observes all* communication channels between the entities, including the SMS channel between the device and the server, the Internet connection between the server and the client, and the user-enabled channel between the client and the device. Further, the adversary is capable of *manipulating all* channels *except* the device-client channel. This assumption is what allows us to devise a protocol that does not require the device to be provisioned with any cryptographic assets beforehand. This setting is close to the usual Bellare-Rogaway model for formalizing security of entity authentication (EA) and authenticated-key exchange (AKE) [57]. The only difference is that we wish to weaken it in a way that captures the adversary's inability to manipulate the channel between the client and the device. This modification of the Bellare-Rogaway model was first explored by Vaudenay [74], who sought to "bootstrap" secure communication over an unauthenticated channel by leveraging an authenticated channel capable of



(a) SOS Registration Phase

(b) SOS Authentication Phase

Figure 3: The registration phase AKE protocol involves the server S, device D, and client C. It consists of two stages: the key exchange between S and D (KE1 and KE2), and the C-mediated key confirmation between S and D (AUTH1, SAS1, SAS2, AUTH2). The authentication phase begins with S choosing a random, 256-bit string W and sending it to D over SMS. The device derives an OTP T using the authentication key K and displays it. The key is then entered into the client and verified by the sever.

transmitting small amounts of information between parties. This so-called *short authenticated string (SAS)* channel allows communication that is subject to adversarial observation only.

Vaudenay envisioned SAS channels being realized by a human participant actually speaking the message, or reading a number displayed by one device and inputting it into another. Because these examples require human intervention, it is necessary from a practical standpoint that the messages are fairly short. Fifteen years later, we have higher-capacity SAS-channels available to us. For example, it is common for both smartphones and feature phones to ship with cameras and software capable of scanning QR codes, which are capable of encoding a few hundred bytes of information. Assuming the device is under the physical control of the user, it is possible to convey this information to the device without the opportunity for the adversary to manipulate it.

3.3 Protocol

Having established our design goals and threat model, we are now ready to describe the registration and authentication phases of the SOS protocol. Our solution to mitigating OTP-rerouting attacks against authentication is simple: rather than send the OTP over SMS, the OTP is derived from the symmetric key and a nonce using a pseudorandom function. This approach is similar to the TOTP and HOTP protocols [52] that underlie a variety of hardware/software tokens. There, too, the OTP is derived by applying a pseudorandom function (i.e., HMAC) to a nonce under a shared secret key. In the case of HOTP, the nonce is a counter stored by both the device and server that is incremented after each authentication attempt; for TOTP, the nonce is the current time, typically quantized in 30-second intervals. In contrast, our system uses a random challenge generated by the server and sent to the client via SMS. This avoids the need to synchronize state between the device and server (i.e., the counter for HOTP and the clock for TOTP.)

By integrating SOS into the device's operating system, it is possible for the authentication phase to perfectly mimic the authentication process of traditional SMS-based 2FA. (See Section 5 for details.) However, we need the server and device to share a key. We provide a protocol for securely establishing this shared secret that closely mimics the registration for traditional SMS-based 2FA.

3.3.1 Registration. Registration begins when the user logs in to the server. First, the client and server establish a server-authenticated TLS session (i.e., an HTTPS connection). Second, the user types in their password, thereby authenticating the client to the server. The user is then prompted to type in the phone number of the device being registered. Along with the phone number, the client also generates a symmetric key and sends it over the TLS-encrypted channel to the server. We call this key K_{psk} to denote the fact that it is the pre-shared key of the client and server. It is used to carry out an authenticated key-exchange (AKE) between the server and device. We formalize the security goal of this protocol in Section 4.

The AKE protocol is pictured in Figure 3a. It is comprised of three primitives: a Diffie-Hellman (DH) function (for a prime-order, cyclic group, where g denotes the group generator); the HKDF key-derivation function (kdf); and a pseudorandom function (F). (We suggest instantiations for these primitives in A.) The AKE protocol has two phases: the *key-exchange* phase in which the server and device exchange DH key shares g^x and g^y; and the *key-confirmation phase* in which the device and server prove to one another that they have engaged in matching conversations and agree on the DH shared secret g^{xy}.

Key Exchange. The server S speaks first, sending to the device D (over SMS) the following quantities: a string α_1 (defined in Section 5), its key share g^x, and the identity C of the client (i.e., the username). We choose a DH function for which group elements have a compact representation: in particular, group $\langle g \rangle$ is the main prime order subgroup on the Curve25519 elliptic curve [21]. (Points in this group can be encoded with just 32 bytes, well below the maximum length of a single SMS.) The device D replies to S with a string α_2 (also defined in Section 5) and its key share g^y.

Key Confirmation. Next, S and D confirm they agree on the values of α_1, α_2, g^x, and g^y, as well as the identities of the client C (i.e., the username), the device D (i.e., the phone number), and the server S (i.e., the domain of the service provider). Key confirmation exploits their mutual trust in the client C, who shares a key K_{psk} with S and who communicates with D over a SAS channel. We describe how the SAS channel is realized below.

The device and server derive two keys from g^{xy}: the *authentication key* K, which will be used in the authentication phase if registration succeeds; and the *key-confirmation key* K_{kck}, which

is used for key confirmation. The server S confirms to D that it has computed the correct key by computing a MAC T_1 over the transcript of the key exchange using K_{kck}. It then sends the following values to C: the transcript σ of its conversation with D, the MAC T_1, and a challenge W derived from K_{psk}.[4] It also computes a MAC over σ and T_1 using K_{psk} in order to ensure integrity of these values as they transit the Internet. This corresponds to the AUTH1 message, as shown in Figure 3a.

After verifying the MAC, the client transmits $T_1 \parallel W$ to the device on the SAS channel (SAS1). Upon receipt of these values, the device D first verifies T_1 using K_{kck}. If this succeeds, then D deems registration successful and outputs K. Finally, D confirms to S it has computed the correct key by computing a MAC over the transcript of the key exchange and the challenge W. Rather than transmit the MAC itself, the device "truncates" the MAC to obtain a string T_2 and sends it over the SAS channel (SAS2). (We describe truncation in a moment.) In the last step (AUTH2), client C transmits T_2 to S, computing a MAC over T_2 with K_{psk} for integrity as the messages transits the Internet. Finally, S verifies T_2 using K_{kck}. If this succeeds, then S deems registration successful and outputs K.

Realizing the SAS Channel. The first SAS-channel message (SAS1) is sent from the client to the device. We note that the means by which the SAS messages are transmitted between the client and device can vary, and the mediums we select for our implementation are what we believe minimize changes compared to traditional SMS-based 2FA. When the device and client are physically separated—suppose the user is registering their mobile phone using a PC—this is accomplished by having the client display a QR code (i.e., in the web browser) that encodes $T_1 \parallel W$. The user scans the code using the device's camera. When the device and client are co-located—suppose the user is registering on their Internet-capable smartphone—the SAS1 message is transmitted to the device directly via the mobile web browser. The user is then given a push notification from the browser, as it is important that the user gives explicit permission for registration to proceed for security purposes.

The second SAS-channel message (SAS2) is identical to the last step of registration in traditional SMS-based 2FA: the device displays an OTP that encodes T_2, and the user inputs the OTP into the client. The string T_2 is truncated to $\lceil \log m \rceil$ bits, where m is largest integer that may be encoded by an OTP, similar to the computation of the OTP in the HOTP and TOTP protocols. (See Section 7 for details.)

Changes to the Registration Process. From the point of view of the user, the execution of the AKE protocol between the device and the server is nearly completely transparent. The only difference in the process is an additional step of scanning a QR code to convey the SAS1 message from the device to the client when the two are physically separated. We evaluate the impact of this change on the overall user experience in Section 6.

3.3.2 Authentication. If registration succeeds, then the device and server agree on the authentication key K. During the authentication phase, this key is used in the following challenge-response protocol. The server requests authentication by choosing a random, 256-bit string W (the same length as the challenge in the registration phase) and sending W to the device over SMS. The device derives an OTP by computing a MAC over W under key K and truncating the MAC to obtain a $\lceil \log m \rceil$-bit string T. It then displays the OTP T to the user, who inputs it into the client. The client sends the OTP to the server, The server verifies T in the usual way, then accepts. This simple protocol is illustrated in Figure 3b.

[4] Challenge W might have been randomly generated, as it is during authentication. We chose to derive it in order to reduce the amount of randomness required for registration.

4 SECURITY ANALYSIS

The authentication protocol is sufficient to thwart OTP-rerouting attacks. In fact, assuming the device and server have securely exchanged a key, authentication enjoys roughly the same security properties as HOTP. It is crucial to HOTP's security that the counter is non-repeating (cf. [52, Theorem 1]); likewise, we must ensure that the challenge is non-repeating. We therefore choose a challenge length (32 bytes) that is long enough to ensure that the probability of randomly generated challenges colliding is negligible. (Applying a standard birthday argument, this probability is at most $q^2/2^{256}$, where q is the number of authentication attempts.)

The truly novel part of SOS is the AKE protocol that forms the core of the registration phase. Given its novelty, we sought to establish its security on firm formal foundations. We begin in Section 4.1 by formalizing the threat model of Section 3.2. Our security goal is that of *key indistinguishability* [57], which captures the adversary's inability to discern any useful information about the authentication key K, even when the client and server are under active attack: with the exception of messages written to the SAS channel, the adversary is capable of manipulating any message sent in the protocol (i.e., over the Internet or SMS). In addition to key indistinguishability, security in our setting implies that *forward secrecy* holds after the key K_{psk} shared by the client and server is compromised. In particular, this ensures that the authentication key K remains secret even after all other assets, including the password, are compromised.

Our main result (Theorem 1) reduces the GDH assumption [57] to the protocol's security, modeling HKDF (*kdf*) and HMAC (*F*) as random oracles [20]. Our reduction yields a tight security bound that describes the concrete security of the AKE protocol as an explicit function of the adversary's resources.

Intuitively, one would expect that the security of the protocol depends most crucially on the length of the OTP sent during registration (SAS2, Figure 3a). Our bound confirms this intuition: the shorter the OTP, the more likely it becomes that the adversary will be able to successfully attack some execution of the registration protocol and learn the authentication key. On the other hand, the success probability only increases significantly as a function of the number of sessions attacked, and not the adversary's computational power. This suggests that, even for a modest OTP length, targeted attacks are infeasible in practice.

Notation. Logarithms are base-2 unless the base is given explicitly. Let $[i..j] = \{x \in \mathbb{N} : i \le x \le j\}$; let $[j] = [1..j]$. A *string* is an element of $\{0,1\}^*$; let ε denote the empty string. Let x_i and $x[i]$ denote the i-th bit of string x. Let $x \parallel y$ denote the concatenation of x and string y. If $i \le j$ then let $x[i..j] = x_i \parallel \cdots \parallel x_n$ where $n = \min\{j, |x|\}$; otherwise let $x[i..j] = \varepsilon$. Let $\langle x_1, \ldots, x_n \rangle$ denote some invertible encoding of arbitrary values $x_1, \ldots, x_n$ as a string. A *tuple* is a finite sequence of values of any type; let ( )

denote the empty tuple. Similarly to strings, we let $X[i] = X_i$ denote the i-th element of tuple X and $X \| Y$ denote concatenation of tuples X and Y. Let $X.y = (X_1, \ldots, X_{|X|}, y)$ (in other words $X.y$ means to append y to X and return the new tuple). We write $y \in X$ if there exists $i \in [|X|]$ such that $X_i = y$. A *table* is an associative array mapping keys of any type to values of any type. We let T_k and $T[k]$ denote the value associated to key k in table T; if no such element exists, then we define T_k to be equal to $\perp$. When keys are pairs (i, j) we will sometimes write T_i^j instead of $T[i, j]$. Let $[\,]$ denote the empty table. Let Dom T denote the set of keys k for which which $T[k] \neq \perp$. We write $X \preceq Y$ if string (resp. tuple) X is a prefix of string (resp. tuple) Y. By convention, if $X = \perp$ then we say that $X \not\preceq Y$.

Let $x \leftarrow A(y)$ denote execution of a deterministic algorithm A on input of y and assigning the output to x. If A is randomized, then we write $x \twoheadleftarrow A(y)$. Let $[A(y)]$ denote the set of possible outputs of A when run on input y. Let $x \twoheadleftarrow \mathcal{X}$ denote sampling x uniform randomly from a finite set $\mathcal{X}$. Let $\rho \twoheadleftarrow \mathrm{Rand}(\mathcal{X}, \mathcal{Y})$ denote choosing a random function ρ from a set $\mathcal{X}$ to a set $\mathcal{Y}$.

All algorithms run in a security experiment are required to halt in a finite number of time steps. A *t-time* algorithm is one that halts after t time steps regardless of (the length of) its inputs, its random coins, or the responses to its oracle queries.

4.1 Formal Model

Here we define our syntax and security model for authenticated key exchange (AKE) protocols. Our starting point is the syntax and execution environment of the extended Canetti-Krawczyk (eCK) model of LaMacchia et al [43] (LLM07). We adapt these in order to account for multi-party protocols and enable communication over a SAS channel [74]. We define key indistinguishability along the same lines as the eCK model, but the notion of session freshness is left as a paramter of the experiment. (Our notion of freshness for our protocol defined in Section 3 dissallows ephemeral key reveals, but require forward secrecy under static key comprimise.)

4.1.1 Syntax. Syntactically, a protocol specifies how keys are generated and how each party responds to messages. Responses are computed as a function of the party's initial input and the sequence of messages received in a session so far, which we call a *conversation*. The conversation records the message's sender as well as the *channel* on which the message was received: either the standard, adversary-controlled channel, or the SAS channel.

Definition 1 (Conversation)**.** Fix $\mathcal{I} \subseteq \{0,1\}^*$. A *conversation for* $\mathcal{I}$ is a tuple $((j_1, M_1, z_1), \ldots, (j_\rho, M_w, z_w))$ where $w \geq 0$ is the number of messages received so far and for each $n \in [w]$, string M_n denotes the n-th message received, $j_n \in \{0,1\}^*$ denotes the sender of the message, and $z_n \in \mathbb{N}$ denotes the *channel* on which the message was sent. □

Definition 2 (Protocol)**.** Fix a finite set $\mathcal{I} \subseteq \{0,1\}^*$ of *parties*. A *protocol for* $\mathcal{I}$ is a triple of algorithms $\Pi = (\mathrm{SGen}, \mathrm{EGen}, \mathrm{Send})$ defined as follows.

- $(pk, sk) \twoheadleftarrow \mathrm{SGen}(\,)$: generates the *static key* (i.e., long-lived secret) and corresponding public key of each party. The outputs pk, sk are tables, where (pk_i, sk_i) denotes the public/static key pair of party $i \in \mathcal{I}$.
- $ek \twoheadleftarrow \mathrm{EGen}(i)$: generates an *ephemeral key* ek for party i, which constitutes the randomness used by i in a given *session*.
- $(X, \kappa, \sigma, \rho, \delta) \leftarrow \mathrm{Send}(sk, ek, \pi, \alpha, i)$: computes the output of party i in conversation π and with input $\alpha \in \{0,1\}^*$. Inputs sk, ek are party i's static key and the session's ephemeral key. The first output is the *outbound messages* $X \in (\{0,1\}^*, \mathbb{N})^*$, each of which is a pair (M, z) where M is the message and z is the *channel* on which the message is output. The remaining outputs are the *private output* and *session identifier (SID)* $\kappa, \sigma \in \{0,1\}^* \cup \{\perp\}$, the *partner identifiers (PIDs)* $\rho \in \mathcal{I}^* \cup \{\perp\}$, and the *decision* $\delta \in \{0,1\} \cup \{\perp\}$.

We say that Π is *p-party* if $|\mathcal{I}| = p$. We say that Π has *t-bit SAS-bandwidth* if each SAS message is at most t bits long, i.e., if for every pair $(M, z) \in X$ such that $z = 1$, it holds that $|M| \leq t$. We say that Π is a *k-bit key-exchange (KE) protocol* if each private output is k bits long, i.e., for every κ, if $\kappa \neq \perp$ then κ is a k-bit string. We say that Π is a (p, t, k)*-protocol* if it is p-party, has t bits of SAS-channel bandwidth, and is a k-bit KE protocol. □

Our syntax for protocols differs from LLM07 in a few respects. Most notably, our syntax supports multiple output channels by having Send output tuples of pairs (M, z) where M is an outbound message and z is the channel on which the message is transmitted. Looking ahead, our security experiment (see Figure 4) handles messages differently depending on the channel: when $z = 0$, message M is presumed to have been transmitted on the standard, adversary-controlled network; and when $z = 1$, the message is presumed to have been transmitted on the SAS channel.

4.1.2 Security. The KEY-IND^f experiment defined in Figure 4 is associated to a (p, t, k)-protocol Π, an adversary E, and a *freshness predicate* f, which determines the validity of the adversary's attack. Let $\mathcal{I}$ denote the set of identities associated with Π. The experiment begins by generating (pk, sk) according to SGen, choosing a random bit b, and running E on input of pk and with oracles that allow the adversary to initialize and send messages to sessions. A *session* is a pair $(s, i) \in \mathbb{N} \times \mathcal{I}$ identifying the s-th run of the protocol for party i. For each session (s, i) the experiment records the ephemeral key ek_s^i, private output κ_s^i, SID σ_s^i, PIDs ρ_s^i, and conversation π_s^i.

- **Init**(s, i, A): initializes a new *session* $(s, i) \in \mathbb{N} \times \mathcal{I}$ with input $A \in \{0,1\}^*$. For each session the experiment generates a fresh ephemeral key $ek_s^i \twoheadleftarrow \mathrm{EGen}(i)$ and initializes the conversation $\pi_s^i \leftarrow (\,)$.
- **Send**(s, i, j, M, z): sends message M from party j on channel z to session (s, i). It appends (j, M, z) to π_s^i and computes the output of i in conversation π_s^i. We keep track of all messages sent from i on the SAS channel: for each output message M' for which the corresponding channel is $z' = 1$, we insert M' into a set $\mathcal{Q}_i$ so that $\mathcal{Q}_i$ records the set of SAS-channel messages sent by party i over the course of the experiment. When the input M is sent on the SAS channel from party j (i.e., $z = 1$), the oracle checks that $M \in \mathcal{Q}_j$; if not the oracle halts and returns $\perp$. This enforces the authenticity property of the SAS channel (cf. [74, Section 2.1]).
- **Test**(s, i): returns κ_s^i if $b = 1$ and a random, k-bit string if not. This oracle may be queried at most once and may not be called unless κ_s^i is defined.

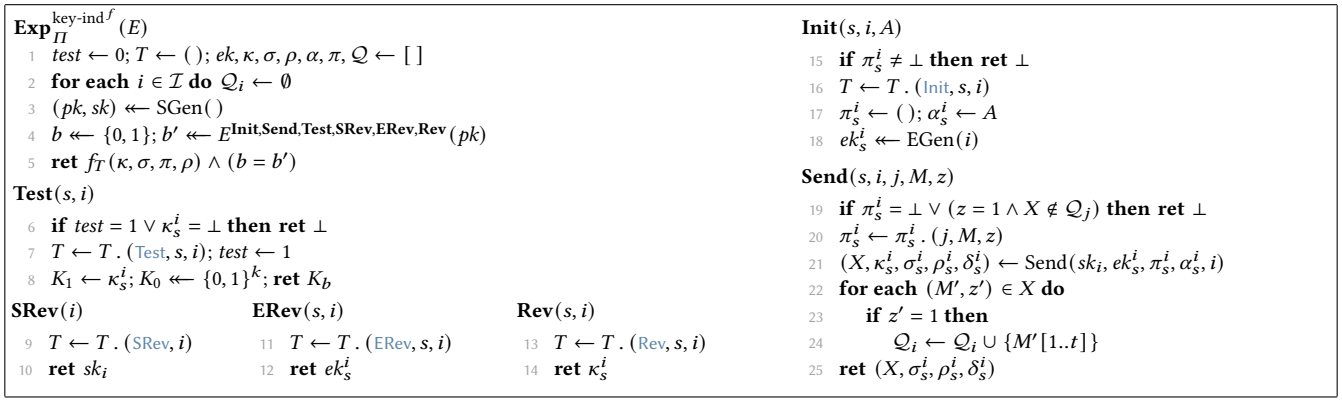


Figure 4: The KEY-IND^f experiment for (p, t, k)-protocol Π for $\mathcal{I}$ and adversary E. We silently enforce restrict E's oracle queries so that the query returns $\perp$ unless $s, z \in \mathbb{N}$, $i, j \in \mathcal{I}$, and $A, M \in \{0,1\}^*$.

- **SRev**(i), **ERev**(s, i), and **Rev**(s, i): reveals (resp.) the static key of party i, the ephemeral key of session (s, i), and the private output of session (s, i) to the adversary.

The experiment records the sequence of queries made by the adversary, as well as the session to which the query pertains, as a tuple T. We refer to T as the experiment *transcript*. The freshness predicate f is a function of the transcript and the values of tables $\kappa, \sigma, \rho, \pi$. The outcome of the experiment is $f_T(\kappa, \sigma, \rho, \pi) \wedge (b = b')$, where b' is the bit output by E.

Definition 3 (KEY-IND^f security)**.** Let Π be a (p, t, k)-protocol, let E be an adversary, and let f be a freshness predicate. Define the *KEY-IND^f-advantage* of E in attacking Π as

$$\mathbf{Adv}^{\text{key-ind}^f}_{\Pi}(E) = 2\Pr\left[\,\mathbf{Exp}^{\text{key-ind}^f}_{\Pi}(E) = 1\,\right] - 1\,.$$

Informally, we say that that Π is KEY-IND^f-secure if $\mathbf{Adv}^{\text{key-ind}^f}_{\Pi}(E)$ is small for all efficient E. □

4.1.3 Definition of Freshness. Our notion of freshness captures forward secrecy for our communication model initial configuration of assets. Per Section 3.1, we assume that S and D communicate strictly using SMS and do not initially share any information that can be used for authentication. We assume that S and C initially share a secret key (presumably the client has logged in to the server over an HTTPS connection). Finally, we will assume that D and C communicate over a SAS channel.

We first define a notion of partnering suitable for protocols in which two parties (S and D) authenticate one another with the assistance of a third party (C).

Definition 4 (Partnering)**.** Fix an adversary E and protocol Π with parties $\mathcal{I}$. Run E with Π as defined in Figure 4 and consider the values of σ, ρ when E halts. We say that sessions $(s, i), (t, j)$ are *partnered in σ, ρ via ℓ* if:

(i) *(SIDs match)* $\sigma^i_s = \sigma^j_t \neq \perp$;
(ii) *(SID unique)* $\sigma^i_s \neq \sigma[Z]$ for any $Z \notin \{(s, i), (t, j)\}$;
(iii) *(PIDs match)* $\rho^i_s = (j, \ell)$ and $\rho^j_t = (i, \ell)$.

We call ℓ the *intermediary* of the partnership. □

The **Test**-session is fresh if its private output and ephemeral key, or the private output and ephemeral key of its partner, have not been revealed to the adversary; and if the static key of neither the client nor the server was revealed prior to the **Test**-query. (This captures forward secrecy of the session key following compromise of the static key.)

Definition 5 (Freshness)**.** Fix an adversary E and protocol Π with parties $\mathcal{I} = \mathcal{S} \cup \mathcal{D} \cup \mathcal{C}$, where $\mathcal{S}, \mathcal{D}, \mathcal{C} \subseteq \{0,1\}^*$ are finite, non-empty, and pairwise-disjoint sets. Run E with Π as defined in Figure 4 and consider the values of T, σ, ρ when E halts. Let $(s, i) \in \mathbb{N} \times (\mathcal{S} \cup \mathcal{D})$. Let $(t, j) \in \mathbb{N} \times (\mathcal{S} \cup \mathcal{D})$ denote the partner of (s, i) in σ, ρ and let $\ell \in \mathcal{C}$ denote the intermediary; if no such t, j, ℓ exist, then set $t, j, \ell \leftarrow \perp$. We say that (s, i) *fresh in T, σ, ρ* if:

(i) $(\text{Rev}, s, i), (\text{Rev}, t, j) \notin T$;
(ii) $(\text{ERev}, s, i), (\text{ERev}, t, j) \notin T$; and
(iii) $\nexists x < y\ [T_y = (\text{Test}, s, i)] \wedge [(i \in \mathcal{S} \wedge T_x = (\text{SRev}, i)) \vee (j \in \mathcal{S} \wedge T_x = (\text{SRev}, j)) \vee (\ell \in \mathcal{C} \wedge T_x = (\text{SRev}, \ell))]$.

Define freshness predicate Fresh so that $\text{Fresh}_T(\kappa, \sigma, \pi, \rho) = 1$ iff either there is no **Test**-query in T or the session incident to the first **Test**-query is fresh in T, σ, ρ. □

4.2 Security of Registration

Our proof uses the GDH assumption [57], which is that the following problem is hard for the given group.

Definition 6 (The GDH problem)**.** Let $\mathbb{G} = (\mathcal{G}, \cdot)$ be a cyclic group with generator g. Define the GDH advantage of an adversary A attacking $\mathbb{G}$ as

$$\mathbf{Adv}^{\text{gdh}}_{\mathbb{G}}(A) = \Pr\left[\,x, y \twoheadleftarrow \mathbb{Z}_{|\mathcal{G}|} : A^{\mathbf{DDH}}(g^x, g^y) = g^{xy}\,\right],$$

where on input of $A, B, C \in \mathcal{G}$, oracle **DDH** returns $(\log_g A)(\log_g B) = \log_g C$. Informally, we say that the GDH assumption holds for $\mathbb{G}$ if advantage of any efficient adversary is small. □

Refer to the AKE protocol Π shown in Figure 3a. (We defer the formal specification to Appendix A.) Let k denote the key length of each key output by *kdf*; let k and h denote (resp.) the key length and output length of F; and let m be the OTP-length parameter. (Note that Π has a SAS-channel bandwidth of $2h$, since SAS1 has length $2h$ and SAS2 has length $\lceil \log m \rceil$; we stress that this is well within the capacity of our particular realization of the SAS channel (cf. Section 3.3.1).) Let $\mathbb{G} = (\mathcal{G}, \cdot)$ denote the DH group.

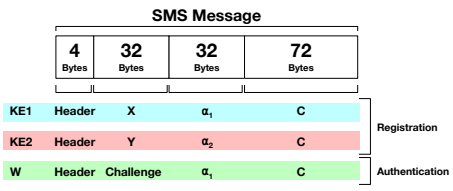


Figure 5: The SMS messages KE1, KE2, and W are composed of 140 bytes. The registration messages consist of a 4-byte header, a 32-byte key share, a 32-byte server identifier α, and 72 bytes for user information C. The authentication message contains a 4-byte header, a 32 byte challenge, a 32-byte server identifier α, and 72 bytes for user information C.

Theorem 1. *Let f = Fresh as in Definition 5. Let $n_{\mathcal{S}} = |\mathcal{S}|$, $n_{\mathcal{D}} = |\mathcal{D}|$, and $n_{\mathcal{C}} = |\mathcal{C}|$. For every t-time, KEY-IND^f-adversary E there exists a $O(t)$-time, GDH-adversary A for which*

$$\mathbf{Adv}^{\text{key-ind}\,f}_{\Pi}(E) \leq \Delta + \mu^2 \mathbf{Adv}^{\text{gdh}}_{\mathbb{G}}(A),$$

where F, kdf are modeled as random oracles; E makes $Q_{\mathbf{Init}}$, $Q_{\mathbf{Send}}$, and Q_F to **Init**, **Send**, *and F respectively; A makes at most Q_F queries to* **DDH**; *and*

$$\Delta = \mu \left[\frac{Q_{\mathbf{Send}}}{2^{\lfloor \log m \rfloor}} + \frac{Q_F + Q_{\mathbf{Init}}}{|\mathcal{G}|} + \frac{Q_F + Q_{\mathbf{Send}}}{2^{h-3}} \right] + \frac{n_{\mathcal{S}} n_{\mathcal{C}} Q_F}{2^k},$$

where $\mu = (n_{\mathcal{S}} + n_{\mathcal{D}}) Q_{\mathbf{Init}}$.

We defer the proof and a discussion of the bound's tightness to Appendix A.

5 METHODOLOGY

To validate our original hypothesis, we implemented our SOS protocol. Though our intention is for SOS is to be built into default messaging applications, for testing purposes we constructed a working app-based prototype of our proposed two-phase protocol. This decision was made specifically to allow us to effectively evaluate the functionality and usability of our protocols. With our prototype, we test deploy SOS on a small scale and conducted a study to evaluate the usability of SOS in comparison to traditional SMS-based 2FA. We note that our protoype app is subject to the same concerns as software tokens, which is why additional development is needed to incorporate SOS into device firmware before real world deployment is possible. We further elaborate on this future work in Section 7.

5.1 SOS Implementation

Our implementation consists of three components; A webserver, an android application, and a custom built 2FA server.

We implemented the device component of the SOS protocols as an Android application for our prototype that acts as a stand-in for the default messaging app. Building SOS as an Android app was done for testing purposes. The Android app was constructed for SDK 29 with a min SDK of 21. We chose for the app to communicate via SMS to maintain the similarity with traditional SMS-based 2FA. Because of this design decision, we chose to host the app via the Amazon App Store. Doing this was necessary for distribution as the Google Play Store does not allow apps that request SMS Permissions (unless it falls under the Default Message Handler) [10]. We elected

Figure 6: The main screen of the Android app we developed displays a token on the right, a website name with the username below it, and a timestamp under the token.

not to make a default message handler so that participants could uninstall our app without issue at the end of the study. Hosting the app on Amazon allowed us to use the SMS permissions. Although participants had an additional step of allowing apks from Amazon, this permitted participants to easily back out of the study if need be and remove SOS from their device after testing was complete. Additionally, this also prevented any possibility of access to private participant messaging data. By developing SOS as an app, we were able to attempt to adhere to the concept of least privilege.

On startup, the SOS app waits for an SMS message to be received by the device. The app inspects received messages and looks for the first two characters to see if they match the start sequence of "%@" for the registration phase or "%#" for the authentication phase. The contents of the SMS messages are divided into multiple sections based on the shortest possible SMS message length of 140 bytes. The first 4 bytes of all SOS messages are reserved for the message header. The next part of both registration messages, seen in Figure 5, is a 32-byte key share value. This is then followed by a 32 byte output of a SHA-256 hash of an identifier that the deploying entity would like to use to identify itself. We call this the "webID". This corresponds to α_1 and α_2 in the notation used in Section 3. (In particular, $\alpha_1 = \alpha_2$.) This typically would be a hash of a deploying website's URL. The last 72 bytes of the message are reserved for the username associated with the account set up of 2FA.

Once the exchange of registration messages is complete, the participant then verifies an HMAC (T_1, see Figure 5) via scanning a QR code, which we build into the app using Google Firebase. After verifying the HMAC the app then prompts the user with an OTP to complete the registration process, which we show in Figure 6.

The authentication message layout is similar to the registration messages but contains a challenge in place of the key shares. After the key confirmation during the registration phase is performed by the participant's device, the SOS app securely stores the key locally for use during authentication and associates the key with the webID and username contained within the SMS message.

The second part of our physical implementation was constructing our own server with SMS messaging capabilities. Though prebuilt IP-to-cellular gateways can be purchased, building our own allowed for increased functionality and control for precision testing and development. The server we built consisted of a Raspberry Pi 3 running RaspbianOS equipped with a 3G Adafruit cellular module. Though this is relatively low power, due to our limited messaging demands this would suffice.

The final part of our implementation was a website that would allow us to simulate the registration and authentication phases of SOS as if it were deployed. The website front end would represent the client machine, and provided the ability for participants to register an account and set up 2FA with our mobile application, as

well as login to an already registered account using SOS for 2FA. We developed the website backend using the Flask framework, which we wrapped in an Nginx reverse proxy. We deployed this using an Amazon Web Service's Lightsail instance and made it public via Google DNS registration. The website communicated with our SMS server via a secure IP connection which provided it with both sending and receiving capabilities via SMS.

The three components we devloped each play an integral role in our prototype implementation. First, the Android app implements our SOS protocol and simulates how OTP delivery would appear to users in a messaging application. Our app was also constructed with the ability to disable our SOS protocol and deliver the OTP via traditional SMS-based means. This feature was important for the user study which we discuss in the next subsection.

In order to evaulte the registration and authentication phases of SOS, we constructed a webserver that communicates with our prototype app and particiaptes in the SOS protocol as if it were deployed. In order for the webserver to communicate via SMS, we constructed a cellular gateway that interfaces with the our webserver and allowed it to send and receive SMS messages.

5.2 User Study Methodology

The goal of this user study is to evaluate the usability of SOS and determine if it will have a negative impact on the usability provided by standard SMS-based 2FA. In order to reach potential participants, we advertised our study on ResearchMatch [35], Facebook groups, and listserves. Interested participants reviewed the consent form and study instructions virtually. They then downloaded the SOS app on their Android phone and created an account on the accompanying website, SMS2FA.dev. It is important to note that the website and application were created as proof of the SOS concept. Participants were randomly put into one of two groups, which determined how they would register and if they received tokens through SOS or standard SMS-based 2FA. Our study evaluated the user experience for the case where the device and client are physically separated, where the SAS1 message is conveyed via a QR code. We choose to evaluate this case alone, as the process is identical outside of the QR code requirement when the client and device are co-located.

Participants were tasked with logging into the website three times over six days after being prompted by a push notification. Participants registered on day one and logged into the website on day three, five and for the last time on day seven. Each login ended with a survey about the participant's experience. At the study's conclusion, participants were compensated with a $10 Amazon e-gift card. This study was approved by the local IRB and followed a protocol similar to previously published studies [64, 76].

SOS SMS-based 2FA Group. Participants in the SOS SMS-based 2FA group used a QR code during registration. The SOS user study app was used to scan the QR code presented on the screen during account creation on the website. Once scanned, an OTP sent with SOS would appear in the app. This token would then be entered where requested on the website. These participants did not use the QR code scanner for the remainder of the study. Instead, after entering their username and password, participants received their token within the app automatically but through the SOS protocol.

Standard SMS-based 2FA Group. Participants in the Standard group received tokens without SOS. They do not use the QR code scanner for any part of the study.

5.3 Participants

A total of 74 participants completed this study. There were 27 participants in the SOS group and 47 in the Standard SMS-based 2FA group.[5] Most of the participants identified as women (53%) and White (75%). Participants also identified as Asian (18%), Black or African American (12%), American Indian (1%), Native Hawaiian(1%), Brazilian (1%), or preferred not to disclose (2%). Most of the participants were in their 30s (45%), while some were in their 20s (34%), 40s (14%), 50s (3%), 60s(3%), and 70s (1%). The majority of our participants had a college degree (70%), and all of them identified themselves as having average or above-average computer expertise.

6 RESULTS

The purpose of this study was to determine if the usability, cognitive load, and task difficulty of SMS-based 2FA were negatively impacted when SOS is implemented. To measure this, we asked each participant about their experience with each task.[6] The difficulty of each task was evaluated using a Single Ease Question (SEQ). The usability of the authentication process was evaluated using the System Usability Survey (SUS). The cognitive demand required for authentication was evaluated using the unweighted NASA Task Load Index (NASA-TLX). Using open-source statistical software called R [69], the Shapiro-Wilkes test was used to check the normality of the data and it determined that the data was non-parametric. Due to this, a one-tailed Mann-Whitney U test [78] was conducted to compare the feedback from the two groups in the study. A one-tailed test was used because this study is concerned with the possibility that SOS might have a negative impact on the Standard SMS-based 2FA user experience.

6.1 Task Difficulty

The *SEQ* is a single question that is answered on a seven-point Likert scale to indicate how difficult participants find a task [65]. A score of one is equivalent to "very difficult" and a score of seven means "very easy". This question was asked after registration and every login. We specifically looked at the SEQ score given by each participant after registration and the third login which happened on the last day of the study. We then used a one-tailed Mann-Whitney U test to analyze the results and test the following hypotheses:

$H1_o$: There is no significant difference between the median Login SEQ score for SOS and the Standard SMS-based 2FA.
$H1_a$: The median Registration SEQ score for SOS is significantly less than the Standard SMS-based 2FA.
$H2_o$: There is no significant difference between the median Login SEQ score for SOS and the Standard SMS-based 2FA.

[5] Imbalances in the groups can be attributed to minor technical difficulties with the OTP server code during the study. However, we were able to account for the imbalances through our analysis, which takes group size into account. Because of this, differences in group size do not have any impact on our results.
[6] We were unable to measure completion time and recognize this as a limitation of our preliminary study. However, unlike previous studies, our focus is on user perception of usability, which we measured in ways similar to previous studies. [26, 64, 82].

$H2_a$: The median Login SEQ score for SOS is significantly less than the Standard SMS-based 2FA.

These hypotheses were chosen because we are interested in identifying the negative impact of SOS on task difficulty. The overall average SEQ score for registration on our proof of concept system was 4.43 and increased to 5.95 by the third login, which was the last task of the study. Most participants (35%) gave the registration process a two and (42%) a seven for the third login. The Mann-Whitney U test showed that the median Login SEQ scores for the SOS group ($\widetilde{x}_1$=7) were not significantly less than (U=473, p>.05,z=-.01,r=.001, n_1=27, n_2=47) the Standard group ($\widetilde{x}_2$=6). The test also showed that the Registration SEQ scores for the SOS group ($\widetilde{x}_1$=3) were not significantly less (U=546, p>.05,z=-1.44,r=.17) than the Standard group ($\widetilde{x}_2$=5). Therefore, we fail to reject $H2_o$ and $H3_o$. This suggests that implementing SOS did not negatively impact the ease of the task.

6.2 Usability Evaluation

The SUS survey is comprised of ten questions and is used to measure the perception of the system usability [23]. Once calculated, the score will be a number out of 100, where a high score implies better usability. For this purpose of this study, we are not focused on comparing the usability of the system we built to others. Instead, we focus on the impact SOS had on the users' perception of usability. We test the following hypothesis:

$H3_o$: There is no significant difference between the median SUS score for SOS and the Standard SMS-based 2FA.

$H3_a$: The median SUS score for SOS is significantly less than the Standard SMS-based 2FA.

The average SUS score for SOS was 70.31 ($\widetilde{x}_1$=71.67) and 59.4 ($\widetilde{x}_2$=63.33) for Standard.The results from the Mann-Whitney U test showed that the median SUS score from the SOS group was not significantly less than the median SUS score for the Standard group (U=473, p>.05,z=-.04,r=.01, n_1=27, n_2=47). We, therefore, fail to reject $H4_o$. This also suggests that SOS will not negatively impact the perception of the system usability of SMS-based 2FA. [7]

6.3 Cognitive Load

The *NASA-TLX* survey is used to determine perceived workload by assessing participant perception of the mental demand needed, physical demand required, their personal performance, effort, temporal demand, and the level of frustration experienced [36]. In this survey, a lower score indicates a lower cognitive load. Therefore, we test the following hypothesis:

$H4_o$: There is no significant difference between the median NASA-TLX score for SOS and the Standard SMS-based 2FA.

$H4_a$: The median NASA-TLX score for SOS is significantly higher than the Standard SMS-based 2FA.

The results of the Mann-Whitney U test showed that the median NASA-TLX score for the SOS group ($\widetilde{x}_1$=10) was not significantly higher (U=429, p>.05, z=-.01, r=.001, n_1=27, n_2=47) than the median score for the Standard group ($\widetilde{x}_2$=19) . We, therefore, fail to reject $H4_o$. This suggests that SOS will not negatively impact cognitive load for users.

[7] We acknowledge that these SUS scores are low and should be improved in later work. However, the goal of this study was to measure the impact SOS has on usability.

7 DISCUSSION

Through our design of SOS we were able to meet all of our outlined security and design goals. In regards to objective D1, we require only one message per authentication and two per registration. Though ideally this might be reduced to a single message per phase, the strength of our adversary (described in Section 3.2) requires bidirectional communication in order to provide secure key agreement. Regardless, the two message requirement only occurs during registration, which happens once per account.

SOS operates entirely at end devices and requires only an IP-to-cellular gateway for servers, which entities using traditional SMS-based 2FA already have in accordance with objective D3. Servers do not require any additional hardware.

By designing SOS to run as part of the default messaging application, we were able to address objective D2. The only change to users is the QR scanning requirement during registration, and our user study (Section 5) showed no statistically significant difference in the usability of SOS compared to traditional SMS-based 2FA.

7.1 Variable OTP Size

SOS mitigates OTP-rerouting attacks using a shared secret key established during registration. This is the same approach as in the HOTP protocol [52], and in fact, our protocol enjoys roughly the same security properties; the novel part of SOS is the AKE protocol that forms the core of the registration phase. The protocol requires no provisioning of cryptographic assets on the device, and instead leverages a SAS channel [74] realized by user interaction. As one would expect, the security of registration depends crucially on the length of the OTP (SAS2, Figure 3a in Section 3). However, the bound we prove suggests that targeted attacks are infeasible in practice, even for modest OTP lengths (say, 6 digits).

7.2 Key Backup

In the event of a SIM Swap attack, an adversary is unable to derive the correct OTP without the authentication key. Defending against these attacks is an essential feature of SOS. However, the requirement to store a key means that, like other methods of 2FA (i.e., Google Authenticator, Authy, and so on), we need a way of migrating the authentication key to new mobile devices.

The most user-friendly recovery method capitalizes on full device backups via network storage. Both Android and Apple devices heavily encourage users to regularly perform full device backups to their respective cloud storage services. These processes have even become automated and a backup will occur daily (permitting that the device is connected to the Internet). Users can be given the option to include their SOS authentication keys as part of a network backup. The option to encrypt the backup of the SOS keys can be offered for users desiring enhanced security.

In the event that a migration is planned and the current device is still accessible, users could transfer authentication keys directly. To secure this transfer, both devices can perform a handshake and prompt the user with an OTP. The user can then confirm that the OTPs on each device match before transferring the SOS keys to secure the process. Though this method requires more user involvement than a cloud backup transfer, this method would allow

users to easily and securely transfer keys without the need for additional network storage services.

If a user loses or damages their mobile device, the keys are most likely lost and unrecoverable without a device backup. This is a complicated problem that affects all secure methods of 2FA by design. Currently, the re-registration process for secure methods of 2FA is uniquely determined by each deploying server. SOS is no exception to this, and recovery is dependent on server policy which may include single-use backup codes or other techniques. Moreover, in the event of numerous unsuccessful login attempts due to lost keys or an attack, websites can flag accounts to limit capabilities while a recovery process takes place.

7.3 Deployment Steps

Though initial deployment of an app version of SOS would allow early users to become familiar with the protocol, due to restrictions on SMS access put into place by Google and Apple this is not possible. Modifications to the default messaging application require us to seek the support of those companies and other OEM manufacturers. A benefit to our protocol is the ability to run alongside traditional SMS-based 2FA, allowing for gradual migration from the existing system to SOS. During the transition process, SOS would only be used when the appropriate SMS messages are received: otherwise, the device can still use traditional SMS-based 2FA as usual. This would also allow servers to gradually scale-up their usage of SOS.

7.4 Inclusive Computing

A wide array of 2FA products are available, and many enterprises provide their employees with strong methods to prevent account takeover (e.g., hardware tokens). While certainly a major deployment scenario, the model does not represent many scenarios in which 2FA is beneficial. For example, an employer-provided hardware token is unlikely to work in many small companies or those outside of the developed world. In such settings, one of the few assumptions about the availability of computing devices is that individuals carry with them a cellular phone. Indeed, much of the developing world performs the majority of their banking and digital payments using feature phones [63], and most such transactions are accomplished via SMS in GSM networks. SOS would bring significant security improvements to such a setting: these older networks are both susceptible to man in the middle attacks and are often configured to operate without encryption over the wireless portion of the network. As such, SOS has significant potential for deployment in places where other 2FA systems are simply unworkable.

8 RELATED WORK

Two-factor authentication is deployed by thousands of websites and systems [6, 24] and has been successful in thwarting unauthorized account access [25, 45]. These mechanisms can be in the form of biometrics, hardware tokens, additional knowledge factors, both current and historic location, software tokens, and phone numbers. Historically, the most widely adopted forms of 2FA have been hardware tokens and mobile device based 2FA methods, such as software tokens and SMS-based 2FA [24, 42].

Initially the most common form of 2FA [42], hardware tokens are a physical device that generate one-time passwords [68]. Traditionally, the OTP was derived from the current time and a shared secret, which expires after a fixed window of time. These OTPs can be exchanged either through manual entry, or by connecting the hardware token to the verifying machine. Though traditional hardware tokens have been largely phased out due to security concerns [22], more advanced versions are still used [14]. These can connect via USB (in some cases NFC and Bluetooth as well) [80] and provide strong security through adherence to the Universal 2nd Factor (U2F) standard [4] and public-key cryptography. Though modern hardware tokens provide improved security over their predecessors, they still exhibit similar usability issues associated with this method of 2FA [47]. Hardware tokens require users to front an initial investment and adopt a new technology that needs to remain on their person at all times, which can further inhibit the already slow adoption of 2FA by everyday users [58, 61, 67]. Additionally, there is a lack of support from websites and systems for modern hardware tokens [13]. Because of these issues, other forms of 2FA that can be incorporated into smartphones have shown wide popularity [42, 66].

The prevalence of smart phones has undoubtedly impacted the popularity of both software tokens and SMS-based 2FA. These two methods of 2FA have become the most widely offered and used at the time of writing [24, 66]. Software tokens can run across multiple devices and vary greatly from one another, such as the factor of authentication itself. Software tokens can use a separate account as the authentication factor such as an email address [33]. Software token commonly use OTP generation in the cases of Authy [2] and Google Authenticator [5], but these tokens are also beginning to eliminate OTPs in exchange for push notifications for increased security and usability. Methods of authentication via software tokens on mobile devices have also been proposed that use a device's microphone [39, 75, 81], camera [16], and accelerometer data [49]. However, despite the innovations and enhanced security of some software tokens, SMS remains the most preferred method for 2FA [26, 34]. Software tokens lack standardization and require users to download additional apps.

A majority of the efforts made in improving the security of 2FA focus on hardware and software tokens, and as a result, minimal developments have been made in security enhancements for SMS-based 2FA. The most similar work to ours is SecurePay developed by Konoth et. al [40], which provides secure OTP delivery for financial transactions through the use of ARM TrustZone. SecurePay focuses on OTP delivery via the traditional Internet, but also can be applied to SMS-based delivery as well. However, this system fundamentally differs from SOS. SecurePay focuses on addressing the concern of OTP interception via mobile malware, whereas SOS addresses completely different attacks rooted in vulnerabilities associated with cellular network infrastructure. Additionally, the details on how to apply SecurePay to SMS-based OTP delivery do not include essential parts of process, such as distribution of a shared secret prior to standard OTP delivery. Other deployment and usability issues exist with SecurePay as well, such as heavy resource requirements for cellular networks and exclusivity to only ARM TrustZone enabled devices. Because of this, SecurePay is unlikely to replace SMS-based 2FA or be a widely deployed alternative.

9 CONCLUSION

Text messaging serves as the most widely used medium for delivering OTPs used in two factor authentication. Unfortunately, this service is not secure, and adversaries have recently increased successful interception of such messages. Instead of attempting to push users to a new technology that is unfamiliar, less usable, and/or incapable of running on their device, we propose the SOS protocol. SOS provides a minor modification to existing default messaging clients, and hiding cryptographic operations while maintaining the user experience. In so doing, SOS strengthens text messaging and allows to be considered as a secure bearer of OTPs.

ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation under grant numbers CNS-1617474 and CNS-1562485. 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. We would also like to thank all of the individuals that participated in our user study.

REFERENCES

[1] 2017. *SMS SS7 Fraud.* Technical Report IR.70. GSMA.
[2] 2020. Authy. https://authy.com/.
[3] 2020. Duo U2F and Biometrics. https://duo.com/product/multi-factor-authentication-mfa/authentication-methods/u2f-and-biometrics.
[4] 2020. FIDO2 U2F Passwordless authentication. https://www.yubico.com/authentication-standards/fido2/.
[5] 2020. Google Authenticator. https://support.google.com/accounts/answer/1066447.
[6] 2020. List of websites and whether or not they support 2FA. https://twofactorauth.org.
[7] 2020. Ting Mobile Rates. https://ting.com/rates.
[8] 2020. Tips to complete account recovery steps. https://support.google.com/accounts/answer/7299973.
[9] 2020. "Two-factor authentication for Apple ID". https://support.apple.com/en-us/HT204915.
[10] 2020. Use of SMS or Call Log Permsissions. https://support.google.com/googleplay/android-developer/answer/9047303.
[11] 2020. What is Diameter Protocol? https://ribboncommunications.com/company/get-help/glossary/diameter-protocol.
[12] 2020. What is the Additional verification page? https://docs.microsoft.com/en-us/azure/active-directory/user-help/multi-factor-authentication-end-user-first-time.
[13] 2020. Works with YubiKey Catalog. https://www.yubico.com/works-with-yubikey/catalog/.
[14] 2020. Yubico. https://www.yubico.com/.
[15] Robert Abel. 2019. SS7 exploited to intercept 2FA bank confirmation codes to raid accounts. scmagazine.com. .
[16] Mozhgan Azimpourkivi, Umut Topkara, and Bogdan Carbunar. 2017. Camera based two factor authentication through mobile and wearable devices. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 3, 35.
[17] Paddy Baker. 2020. Crypto Exec's $1.8M SIM-Swap Lawsuit Has 'Critical Holes,' Says AT&T. https://www.coindesk.com/crypto-execs-1--8m-sim-swap-lawsuit-full-of-critical-holes-says-att.
[18] Sam Baker. 2019. Criminals hacking phone codes used by customers to verify bank transactions. https://www.telegraph.co.uk/money/consumer-affairs/banks-hit-mobile-phone-hackers/.
[19] Richard Barnes, Benjamin Beurdouche, Jon Millican, Emad Omara, Katriel Cohn-Gordon, and Raphael Robert. 2019. *The Messaging Layer Security (MLS) Protocol.* Internet-Draft draft-ietf-mls-protocol-08. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-ietf-mls-protocol-08 Work in Progress.
[20] Mihir Bellare and Phillip Rogaway. 1994. Entity Authentication and Key Distribution. In *Advances in Cryptology — CRYPTO' 93.* Springer Berlin Heidelberg, Berlin, Heidelberg, 232–249.
[21] Daniel J. Bernstein. 2006. Curve25519: New Diffie-Hellman Speed Records. In *PKC — Proceedings of the 9th International Conference on Theory and Practice of Public Key Cryptography.* 207–228.
[22] Peter Bright. 2011. RSA finally comes clean: SecurID is compromised. https://arstechnica.com/information-technology/2011/06/rsa-finally-comes-clean-securid-is-compromised/.
[23] John Brooke et al. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.
[24] Elie Bursztein. 2018. The bleak picture of two-factor authentication adoption in the wild. https://elie.net/blog/security/the-bleak-picture-of-two-factor-authentication-adoption-in-the-wild/#toc-4.
[25] Catalin Cimpanu. 2019. Microsoft: Using multi-factor authentication blocks 99.9% of account hacks. zdnet.com. .
[26] Stéphane Ciolino, Simon Parkin, and Paul Dunphy. 2019. Of Two Minds about Two-Factor: Understanding Everyday FIDO U2F Usability through Device Comparison and Experience Sampling. https://www.usenix.org/conference/soups2019/presentation/ciolino. In *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019).* USENIX Association, Santa Clara, CA.
[27] Jessica Colnago, Summer Devlin, Maggie Oates, Chelse Swoopes, Lujo Bauer, Lorrie Cranor, and Nicolas Christin. 2018. "It's not actually that horrible" Exploring Adoption of Two-Factor Authentication at a University. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems.* 1–11.
[28] Joseph Cox. 2017. Senator Demands Answers From Telecom Giants on Phone Spying. https://www.thedailybeast.com/senator-demands-answers-from-telecom-giants-on-phone-spying.
[29] Joseph Cox. 2019. Criminals Are Tapping into the Phone Network Backbone to Empty Bank Accounts. https://motherboard.vice.com/en_us/article/mbzvxv/criminals-hackers-ss7-uk-banks-metro-bank.
[30] D. DeFigueiredo. 2011. The Case for Mobile Two-Factor Authentication. *IEEE Security Privacy* 9, 5 (Sept. 2011), 81–85.
[31] Dominique Lazanski. 2016. Interconnect Security.
[32] Thomas Fox-Brewster. 2017. All That's Needed To Hack Gmail And Rob Bitcoin: A Name And A Phone Number. https://www.forbes.com/sites/thomasbrewster/2017/09/18/ss7-google-coinbase-bitcoin-hack/#484e841941a4.
[33] Kathleen Garska. 2018. Two-Factor Authentication (2FA) Explained: Email and SMS OTPs. https://blog.identityautomation.com/two-factor-authentication-2fa-explained-email-and-sms-otps.
[34] Conor Gilsenan. 2018. SMS: The most popular and least secure 2FA method. https://www.allthingsauth.com/2018/02/27/sms-the-most-popular-and-least-secure-2fa-method/.
[35] Paul A Harris, Kirstin W Scott, Laurie Lebo, NikNik Hassan, Chad Lighter, and Jill Pulley. 2012. ResearchMatch: a national registry to recruit volunteers for clinical research. *Academic medicine: journal of the Association of American Medical Colleges* 87, 1 (2012), 66.
[36] Sandra G Hart and Lowell E Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Advances in psychology.* Vol. 52. Elsevier, 139–183.
[37] Evolved Intelligence. [n.d.]. SS7 & Diameter Firewall. https://www.evolved-intelligence.com/products/fraud-and-security/signalling-firewall.
[38] Michael Kan. 2018. Hackers Beat Two-Factor Protection With Automated Phishing Attacks. pcmag. .
[39] Nikolaos Karapanos, Claudio Marforio, Claudio Soriente, and Srdjan Capkun. 2015. Sound-Proof: Usable Two-Factor Authentication Based on Ambient Sound. In *USENIX Security Symposium (USENIX Security 15).* USENIX Association. https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/karapanos
[40] Radhesh Krishnan Konoth, Björn Fischer, Wan Fokkink, Elias Athanasopoulos, Kaveh Razavi, and Herbert Bos. 2020. SecurePay: Strengthening Two-Factor Authentication for Arbitrary Transactions. In *5th IEEE European Symposium on Security and Privacy (EuroS&P 2020).*
[41] Brian Krebs. 2019. Why Phone Numbers Stink As Identity Proof. https://krebsonsecurity.com/tag/sim-swap/.
[42] Kyle Lady. 2015. Duolytics: Four Years with Four Factors. https://duo.com/blog/duolytics-four-years-with-four-factors.
[43] Brian LaMacchia, Kristin Lauter, and Anton Mityagin. 2007. Stronger Security of Authenticated Key Exchange. In *Provable Security,* Willy Susilo, Joseph K. Liu, and Yi Mu (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–16.
[44] Ponemon Institute LLC. 2019. The 2019 State of Password and Authentication Security Behaviors Report. https://www.yubico.com/wp-content/uploads/2019/01/Ponemon-Authentication-Report.pdf.
[45] Napier Lopez. 2019. Google data shows 2-factor authentication blocks 100% of automated bot hacks. https://thenextweb.com/google/2019/05/23/google-data-shows-2-factor-authentication-blocks-100-of-automated-bot-hack.
[46] G. Lorenz. 2001. Securing SS7 telecommunications networks. Proceedings of the Second Annual IEEE Systems, Man, and Cybernetics Information Assurance Workshop.
[47] Ronnie Manning. 2019. Yubico Releases the 2019 State of Password and Authentication Security Behaviors Report. https://www.yubico.com/2019/01/yubico-releases-the-2019-state-of-password-and-authentication-security-behaviors-report/.
[48] Venkadesan Marimuthu. 2019. Fraudulent subscriber identity module (SIM) swap detection. United States Patent 10178223.

[49] Rene Mayrhofer and Hans Gellersen. 2007. Shake Well Before Use: Authentication Based on Accelerometer Data. In *Pervasive Computing*, Anthony LaMarca, Marc Langheinrich, and Khai N. Truong (Eds.).

[50] T. Moore, T. Kosloff, J. Keller, G. Manes, and S. Shenoi. 2002. Signaling system 7 (SS7) network security. In *The 2002 45th Midwest Symposium on Circuits and Systems, 2002. MWSCAS-2002.*, Vol. 3. III–III. https://doi.org/10.1109/MWSCAS.2002.1187082

[51] Megan Morreale. 2017. Daily SMS Mobile Usage Statistics. https://www.smseagle.eu/2017/03/06/daily-sms-mobile-statistics/.

[52] D. M'Raihi, M. Bellare, F. Hoornaert, D. Naccache, and O. Ranen. 2005. *HOTP: An HMAC-Based One-Time Password Algorithm.* RFC 4226. RFC Editor.

[53] Collin Mulliner, Ravishankar Borgaonkar, Patrick Stewin, and Jean-Pierre Seifert. 2013. SMS-Based One-Time Passwords: Attacks and Defense. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, Konrad Rieck, Patrick Stewin, and Jean-Pierre Seifert (Eds.). Springer Berlin Heidelberg.

[54] Action Fraud News. 2014. Alert - how you can be scammed by a method called SIM Splitting.

[55] Newsbtc. 2017. Hackers Find Exploit Through SS7 SMS 2FA to Empty Bitcoin Wallets. https://www.newsbtc.com/2017/09/30/hackers-find-exploit-sms-two-factor-authentication-empty-bitcoin-wallets/.

[56] Sai Fong Ngan, Wai Ching Vincent Lok, and Kwok Hung Cheung. 2015. Wireless two-factor authentication, authorization and audit system with close proximity between mass storage device and communication device. US Patent App. 14/284,464.

[57] Tatsuaki Okamoto and David Pointcheval. 2001. The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. In *Public Key Cryptography*. Springer Berlin Heidelberg, Berlin, Heidelberg, 104–118.

[58] Thuy Ong. 2018. Over 90 percent of Gmail users still don't use two-factor authentication. https://www.theverge.com/2018/1/23/16922500/gmail-users-two-factor-authentication-google.

[59] Tom Parker. 2019. Shane Dawson, James Charles, King Bach, and other YouTubers hacked after alleged AT&T SIM swap. https://reclaimthenet.org/shane-dawson-james-charles-youtubers-hacked-att-sim-swap/.

[60] Christian Peeters, Hadi Abdullah, Nolen Scaife, Jasmine Bowers, Patrick Traynor, Bradley Reaves, and Kevin Butler. 2018. Sonar: Detecting SS7 Redirection Attacks with Audio-Based Distance Bounding. In *39th IEEE Symposium on Security and Privacy (IEEE Security and Privacy 18)*.

[61] Thanasis Petsas, Giorgos Tsirantonakis, Elias Athanasopoulos, and Sotiris Ioannidis. 2015. Two-Factor Authentication: Is the World Ready? Quantifying 2FA Adoption. In *Proceedings of the Eighth European Workshop on System Security (EuroSec '15)*. Association for Computing Machinery, New York, NY, USA.

[62] Nathaniel Popper. 2019. Hackers Hit Twitter C.E.O. Jack Dorsey in a 'SIM Swap.' You're at Risk, Too. https://www.nytimes.com/2019/09/05/technology/sim-swap-jack-dorsey-hack.html.

[63] Bradley Reaves, Nolen Scaife, Adam Bates, Patrick Traynor, and Kevin Butler. 2015. Mo(bile) Money, Mo(bile) Problems: Analysis of Branchless Banking Applications in the Developing World. In *Proceedings of the USENIX Security Symposium (SECURITY)*.

[64] Ken Reese, Trevor Smith, Jonathan Dutson, Jonathan Armknecht, Jacob Cameron, and Kent Seamons. 2019. A usability study of five two-factor authentication methods. In *Fifteenth Symposium on Usable Privacy and Security (SOUPS)*.

[65] J Sauro. 2012. 10 things to know about the Single Ease Question (SEQ). *Measuring U, 2012* (2012).

[66] Duo Security. 2017. Share of internet users in the United States who use two-factor authentication in 2010 and 2017, by method. https://www.statista.com/statistics/789942/us-use-of-two-factor-authentication/.

[67] Duo Security. 2017. Share of internet users in the United States who use two-factor authentication in 2013 and 2017. https://www.statista.com/statistics/789473/us-use-of-two-factor-authentication/.

[68] RSA Security. 2004. RSA SecurID Solution Named Best Third-Party Authentication Device by Windows IT Pro Magazine Readers' Choice 2004. https://web.archive.org/web/20100106232859/http://rsa.com/press_release.aspx?id=5028.

[69] R Core Team. 2013. R: A language and environment for statistical computing. (2013).

[70] Positive Technologies. 2018. Diameter vulnerabilities exposure report. https://www.ptsecurity.com/ww-en/analytics/diameter-2018/.

[71] Kurt Thomas, Damon McCoy, Chris Grier, Alek Kolcz, and Vern Paxson. 2013. Trafficking Fraudulent Accounts: The Role of the Underground Market in Twitter Spam and Abuse.. In *USENIX Security*. 195–210.

[72] Bill Toulas. 2019. SS7 Attacks Against Telecom Infrastructure Now on the Rise. https://www.technadu.com/telecom-infrastructure-ss7-attacks-rise/56704/.

[73] International Telecommunication Union. [n.d.]. ITU Standard Q.700 : Introduction to CCITT Signalling System No. 7. https://www.itu.int/rec/T-REC-Q.700/en.

[74] Serge Vaudenay. 2005. Secure Communications over Insecure Channels Based on Short Authenticated Strings. In *Advances in Cryptology – CRYPTO 2005*, Victor Shoup (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 309–326.

[75] Mingyue Wang, Wen-Tao Zhu, Shen Yan, and Qiongxiao Wang. 2018. SoundAuth: Secure Zero-Effort Two-Factor Authentication Based on Audio Signals. In *2018 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 1–9.

[76] Catherine S Weir, Gary Douglas, Martin Carruthers, and Mervyn Jack. 2009. User perceptions of security, convenience and usability for ebanking authentication tokens. *Computers & Security* 28, 1-2 (2009), 47–62.

[77] Kenneth P. Weiss. 1984. Method and apparatus for positively identifying an individual. U.S. Patent US4720860A,.

[78] Frank Wilcoxon. 1992. Individual comparisons by ranking methods. In *Breakthroughs in statistics*. Springer, 196–202.

[79] Ben Winck. 2019. One cryptocurrency investor reportedly lost $24 million worth of bitcoin in a SIM swap attack. https://markets.businessinsider.com/currencies/news/bitcoin-investor-loses-24-million-of-crypto-sim-swap-hackers-2019-11-1028677818.

[80] Yubico. 2018. YubiKey 5 Series: The Multi-Protocol Security Key. https://www.yubico.com/wp-content/uploads/2019/08/191238-YK5Series-Solution-Brief-r10.pdf.

[81] Dimitra Zarafeta, Christina Katsini, George E Raptis, and Nikolaos M Avouris. 2019. UltraSonic Watch: Seamless Two-Factor Authentication through Ultrasound. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, LBW2614.

[82] Verena Zimmermann and Nina Gerber. 2020. The password is dead, long live the password. A laboratory study on user perceptions of authentication schemes. *International Journal of Human-Computer Studies* 133 (2020), 26 – 44. https://doi.org/10.1016/j.ijhcs.2019.08.006 .

A SECURITY ANALYSIS FOR REGISTRATION

The GDH assumption is not essential to our argument, but is used to achieve a tighter bound. A reduction to the weaker computational DH (CDH) assumption is possible, but with a loss of a multiplicative factor of the number of kdf-queries made by E.

For parameters likely to be used in practice, the most important term in the bound is $(n_{\mathcal{S}} + n_{\mathcal{D}})Q_{\mathbf{Init}}Q_{\mathbf{Send}}/2^{\lfloor \log m \rfloor}$ because it depends crucially upon the length of the SAS2 message. This message encodes the OTP, a number between 0 and $m-1$ that is short enough for a user to type. If m is small, say $m = 10^6$ (OTPs are six digits), then $\lfloor \log m \rfloor$ is only 19 bits of security. As a result, the bound becomes vacuous after a fairly modest number runs of the protocol. However, if m is somewhat longer, say $m = 10^{16}$ (OTPs are sixteen digits), then $\lfloor \log m \rfloor$ is a respectable 53 bits of security.

In fact, this bound turns out to be reasonably tight, as exhibited by the following attack. Fix a server S, device D, and client C. The adversary's goal is to trick S into computing a session key it knows. It does so by replaying a SAS2 message sent by D.

(1) Step 1: Run the protocol with S, D, and C until D outputs the SAS2 message T_2 and record T_2 in a set $\tilde{\mathcal{Q}}$. Repeat this r times.
(2) Repeat the following procedure t times or until the attack succeeds.
   (a) Run the protocol until S outputs KE1 $\langle \alpha, X, C \rangle$. Choose $\tilde{\alpha} \in \{0,1\}^*$ and $\tilde{y} \twoheadleftarrow \mathbb{Z}_{|\mathcal{G}|}$ and send $\langle \tilde{\alpha}, \tilde{Y} \rangle$ to S, where $\tilde{Y} = g^{\tilde{y}}$. (This forges KE2 from the device.)
   (b) When S responds with AUTH1 $\langle \sigma, T_1, V_1, W \rangle$, compute $(K_{\text{kck}}, K) \leftarrow kdf(X^{\tilde{y}}, \sigma)$ and
   $$\tilde{T}_2 \leftarrow otp(F^2_{K_{\text{kck}}}(\sigma \,\|\, \langle S, D, C \rangle \,\|\, W), m)\,.$$
   (c) If $\tilde{T}_2 \in \tilde{\mathcal{Q}}$, then forward AUTH1 to C. When C responds with SAS1, ignore it and instead send $\tilde{T}_2$ to C. When C responds AUTH2, forward it to S. The attack has succeeded.

When the attack succeeds, the server S computes K_{kck} and K just as the adversary did. Hence, the server will deem $\tilde{T}_2$ to be valid and accept session key K. The attack succeeds if $\tilde{T}_2$ computed in step (2) is in the set $\tilde{\mathcal{Q}}$ computed in step (1). Because the adversary attempts step (2) t times, in the random oracle model for F, the success probability is roughly rt/m.

The significance of this attack is that its success probability is close to the proven upper-bound. This suggests that we cannot significantly improve the protocol's concrete security, at least not in our formal model. Indeed, the attack seems fairly unrealistic in a real world sense. On the positive side, the weakest term in the bound is purely a function of the adversary's online work and not its offline computation.

Security in the Standard Model. It is likely that KEY-IND^f security of our protocol could be proven in the standard model, but the proof would require stronger-than-usual assumptions. In particular, the standard PRF assumption of F would not suffice, because our adversary is empowered to reveal the static key used to authenticate the **Test**-session after the query is issued. (This captures forward secrecy of the session key after the static key is revealed.)

Another challenge would be to identify suitable assumptions on the function $otp(F(\cdot,\cdot,\cdot), m)$. In particular, even for fairly large m, collision resistance would would seem to be too strong of an assumption. However, such a strong assumption may not be necessary, since in the protocol, this function is evaluated on input of the pseudorandom challenge string W.

B SPECIFICATION OF REGISTRATION

This section of the Appendix has been omitted in efforts to meet the page requirement and can be found in the extended version of this paper made available on arXiv.

B.1 Proof of Theorem 4.7

This section of the Appendix has been omitted in efforts to meet the page requirement and can be found in the extended version of this paper made available on arXiv.