5G Messaging: System Insecurity and Defenses

Jinghao Zhao*, Qianru Li*, Zengwen Yuan, Zhehui Zhang, Songwu Lu
University of California, Los Angeles
{*jzhao, qianru, zyuan, zhehui, slu*}*@cs.ucla.edu*

2022 IEEE Conference on Communications and Network Security (CNS) | 978-1-6654-6255-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/CNS56114.2022.9947238

***Abstract*—5G introduces Rich Communication Services (RCS) as the official messaging service. It replaces SMS with rich multimedia content over the chat session. RCS intends to provide "any network, any device" messaging services for a given user across various network (4G/5G or Wi-Fi) and devices (SIM-based phones or SIM-free tablets/gadgets). This work provides the first in-depth study of RCS system security. We find that although RCS is one of the mobile carrier services, it performs a weak cellular ID binding, which opens a door for attackers to hijack the victim's RCS service. Even with end-to-end encryption in place, impersonation and eavesdropping over chat messages are still feasible. The attacks could be persistent and stealthy. With abused RCS service, victims are vulnerable to various attacks of fraud, location tracking, unauthorized operations on business accounts, and spamming. We have empirically validated such attacks in 4 major US mobile carriers by following ethical requirements. We further propose and implement both long-term solutions and immediate remedies.**

I. Introduction

5G has standardized Rich Communication Services (RCS) as the official messaging system based on GSMA specifications [22]. It seeks to provide "any network, any device" messaging services across different mobile carriers [14]. It replaces text-based SMS with richer content in group chats, video, audio, and images for both users and businesses [12]. Different from the Instant Messaging (IM) applications such as WhatsApp that require user installation, RCS service is provided by carriers and has been deployed in the built-in message app on mobile OS such as Android [6]. It has rolled out with 1.2 billion active users from 90 mobile carriers in 60 countries to date [22]. RCS also promises to offer better user experiences and open up more opportunities for businesses [12], [13]. The business messaging service by RCS is projected to generate $8.3B revenue by 2024 [13].

System security is a primary goal for 5G RCS. From the security perspective, 5G RCS includes all necessary protection including user authentication, server authenticity, integrity check, and end-to-end encryption. Moreover, the system implementation is as excellent as Google Messenger [9], and its operations by carriers have no non-trivial slips except [21]. Therefore, the RCS design seems not to have significant security issues. Indeed, given the large-scale rollout since 2018, there have been no further reports on RCS insecurity.

However, our recent study shows negative results. We find that user authentication in 5G RCS can be breached in practice. As a mobile carrier service, RCS must be directly accessible to cellular subscribers (i.e., SIM holders). However, the current RCS binding with cellular IDs is weak and opens a door for hijacking the RCS service. By detailed inspection of the RCS signaling procedures, we devise the attack that can remotely hijack the victim's RCS service. The attack does not need any root access on the victim's device. It is also regardless of the victim's access network (4G/5G cellular or Wi-Fi) or device type. With hijacked accounts, other security functions of message confidentiality and integrity can also be compromised. Even with end-to-end encryption in place for RCS chat sessions, adversaries is still able to impersonate and eavesdrop on chat messages. We have validated such attacks with four US mobile carriers (with a total market share $> 98\%$) while following ethical requirements. We further validate that the attack could be persistent and stealthy in practice, which makes it hard to be detected and defended by victims.

Furthermore, enriched services provided by 5G RCS exacerbate the damage in the real world. We validate the insecurity and privacy intrusion in the current 5G RCS services. The attacker can launch fraud or location tracking by impersonating victims. The attacker could also perform business account invasion and heavy data spamming to cause financial losses to victims. We further conduct a user study from 679 participants covering a wide range of ages. The results show that the RCS service abuse could lead to financial losses and privacy leakage, with a high probability from 24% to 33.3% in practice.

To defend against such threats, we propose both long-term solutions and short-term fixes. In the long run, RCS must ensure the cellular IDs remain private to its app. For phones with SIMs, RCS should use SIM/eSIM-based keys that are exclusively tied to messaging apps. They are well tested by 4G/5G services like VoLTE. On SIM-free devices, RCS may leverage the assistance of the phone device and introduce an intermediate binding with the verified smartphone. We also propose quick remedies for immediate mitigation. Our gained insights on RCS and carrier-supported services may also shed light on securing other mobile applications. The lessons learned from RCS will further help to fix any authentication loophole there.

II. Background

A. System Architecture and Workflow

As shown in Figure 1, the RCS system consists of five components to offer enriched chat service: profile server, messaging server, content server, key server, and network-network interface (NNI). We illustrate their functions through the example of how user Alice logs on to RCS and starts chatting with her friend Bob. To start, Alice logs on to the

*Co-first authors

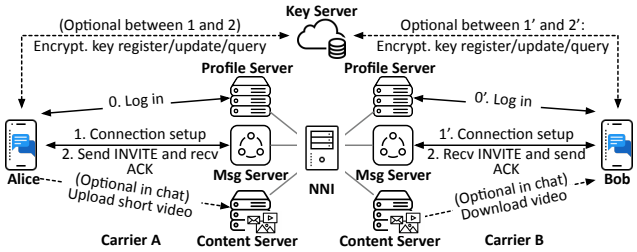
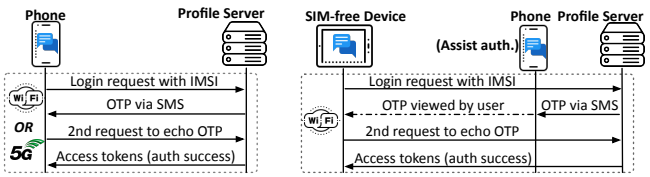


Fig. 1: RCS system architecture.



(a) Access from phones (b) Access from other devices

Fig. 2: Authentication in multiple use scenarios.

profile server and gets access tokens that authenticate all follow-up operations in the current logon session. Next, Alice connects with the messaging server using the access token and invites Bob to a live chat. If end-to-end encryption is supported, Alice will generate a set of keys, upload the public key to the key server, and query Bob's public key as well. Once Bob accepts the invitation, Alice sends messages (encrypted, say, with the e2e encryption scheme) to him. Optionally, Alice shares a file (e.g., picture/video) by uploading it to the content server and sending the URL to Bob. Finally, Alice ends chatting by sending Bob a BYE message. Note that Bob is reachable though served by a different carrier, given the inter-carrier connection provided by NNI.

B. RCS Security

RCS guarantees that any legitimate user can access services under any network (cellular 4G/5G or non-cellular Wi-Fi networks) and any device (phones or SIM-free tablets/gadgets). RCS system security is a well-informed design that seeks to meet the following security requirements:

User Authentication Supported by mobile carriers, 5G RCS is provisioned for mobile subscribers only. Therefore, RCS verifies the secure binding between the RCS account and cellular IDs (e.g., phone number). To support the access from any network any device, Figure 2 depicts the scenario-specific customization. For smartphones (Figure 2a), the account is bound to the phone number with OTP. International Mobile Subscriber Identity (IMSI) is also required to strengthen the binding. For SIM-free devices (non-cellular networks), RCS requires users to input OTP received from the phone with the corresponding SIM for authentication (Figure 2b).

Server Authentication RCS guarantees server authenticity by HTTPS/TLS with root certificates issued by trusted CA.

Integrity and confidentiality RCS adopts a series of standardized mechanisms to prevent eavesdropping and tampering of both signaling and chat messages. Service logon (which involves credential exchange) is protected by HTTPS; Chat messages are first protected by end-to-end encryption [7] and then encrypted by the server-client TLS connection.

III. Overview

A. Threat Model

In our study, victims could be anyone eligible for RCS service (i.e., cellular subscribers). We make two assumptions about the attacker's capabilities. First, the attacker owns a rooted device with 5G RCS enabled, and a malicious server with the network connection to launch attacks. Second, the attacker knows the victims' phone numbers by crawling profiles on social networks (e.g., LinkedIn, Facebook, etc.) The attacker does NOT require root access of the victim's device, or the privilege to control or manipulate RCS servers.

B. Attack overview

The current 5G RCS performs various protection, including ID binding, end-to-end encryption, and periodic re-authentication. We perform a detailed inspection of the security procedures of the 5G RCS and devise attacks as shown in Figure 3. First, we devise the account hijacking (§IV) by breaking the ID binding and launching the attack from a fake client remotely. Despite the end-to-end encryption, the attacker could still hijack the victim's account to initialize chat sessions. We enhance the hijacking attack to be persistent and stealthy to the victim. Furthermore, based on the new rich chat services with hijacked accounts, we explore various attacks in terms of impersonation, location tracking, unauthorized operations, and spam downloading (§V).

We validate vulnerabilities and attacks with victims using 4 U.S. mobile carriers (total market share $> 98\%$) and various Android phone models (Google Pixel 5/4a/2/1, OnePlus 7 Pro/8 Pro, and Mi MIX 2). Tested Android OS versions cover 9/10/11/12. Even users' phone models do not support 5G RCS (e.g., OnePlus 7 Pro and Mi MIX 2 in our experiment), they are still potential victims under attacks. This is because those legitimate SIM holders are eligible to use 5G RCS on a different device. Without specification, validation results of attacks and vulnerabilities apply to all tested operators/phones.

Ethical issues Our attacks do not incur damage to real users. The tested devices and SIM cards are only for experimental usage, but not associated with real users. Moreover, the attacks do not negatively impact the RCS service providers, given that only up to 3 devices are involved and little traffic is generated. Furthermore, the authors are reaching out to all four US carriers and working with them and the OS vendor to examine the uncovered vulnerabilities and proposed remedies.

IV. Component I: Hijacking 5G RCS

In this section, we introduce vulnerabilities in 5G RCS authentication. Attackers can exploit the weak ID bindings (§IV-A) to hack victims' accounts under any network and any device (§IV-B), and breaks the end-to-end encryption for chat

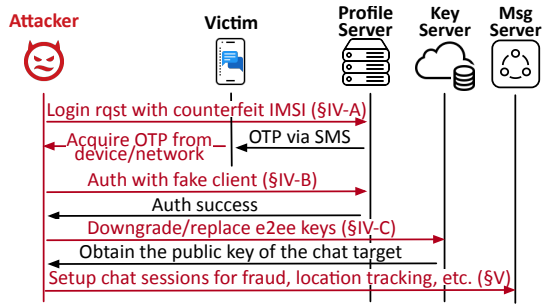


Fig. 3: Attack overview.

messages (§IV-C). We further show how to achieve a persistent and stealthy attack for the hijacking (§IV-D).

A. Weak ID Bindings

RCS takes different cellular IDs to match capabilities under different access networks and devices. Cellular IDs are verified by *confidential* information for security-wise strengths. RCS leverages two cellular IDs to bind the user: IMSI and phone number. We find that both adopted IDs introduce weak binding. All weaknesses break the initial authentication and open surfaces for account hijacking.

1) IMSI - Partial Binding: RCS server demands a cellular ID, IMSI, to claim the binding between an RCS account and the SIM holder. It takes advantage of IMSI being a private subscriber ID stored in a SIM. Although it is not accessible to non-system apps on Android OS (version 10 or above), the binding is weak considering two practical issues. First, it is not strictly enforced by service providers. In the current implementation, the user account is only bound to a fragment of IMSI referring to MCCMNC, a public identity code of the mobile carrier. Since attackers can acquire MCCMNC using the victim's phone number [5], they can easily counterfeit an IMSI to break the binding. On the other hand, even if the IMSI is thoroughly examined, it still suffers from leakage during cellular service attach (e.g., downgrade attack with IMSI catchers [36], [11]).

2) Phone Number - Verified by Public Information: RCS adopts one-time passwords (OTP) via SMS to verify the phone numbers. However, the attacker could eavesdrop on the victim's SMS from either the client or network side. From the client side, applications could read the SMS with APIs provided by mobile OS [28]. It has been validated that applications with SMS reading privilege could pass the application store's checking (e.g., Google Play) and be installed by victims [26]. From the network side, attackers could acquire the SMS by signaling attacks (SS7, diameter, etc.) [23], [29], [30]. To make things worse, 5G RCS adopts invisible SMS for OTP, which is hidden from users but automatically extracted by the built-in message app to complete verification. It targets better usability by getting rid of user operation or even attention during service access. However, invisibility facilitates the stealthiness of attacks. It makes victims lose the chance to realize the potential risk and stop it immediately.

Validation We validate the vulnerability for 4 US mobile carriers in our study. For all 4 operators, none of them provides sufficient examination on IMSI. Every carrier only checks that the carrier code (MCCMNC, first 6 digits in IMSI) and the URL in the request refer to the same mobile carrier. In our test, we concatenate the 6-digit code of any carrier and a random 9-digit number to create a fake IMSI like ⟨MCCMNC⟩123456789. With a fake IMSI, we make a successful request on behalf of the victim. We prototype an Android application to stole OTP with `READ_SMS` permission. Our results validated OTP leakage for all 4 US operators. OP-I/II/III mandate invisible SMS for smartphone registration. As OP-IV supports visible SMS, the application has one more choice to sniff OTP from the notifications.

B. Hacking Accounts under Any Network Any Device

The weak ID bindings enable the attacker to log on to the RCS service using the victim's cellular identity. Note that this hijacking attack could target *any* cellular service user, no matter whether they have existing RCS accounts or not. This is because attackers could create the RCS account first for the victim, hijack the target account, and do evil in her name.

To successfully launch the attack, we first learn the operation logic by collecting signaling/chat messages from a rooted device for various events: service request, login, message exchanging, and file sharing. After the detailed inspection of the signaling procedures, we deploy a fake client to counterfeit signaling messages for attack validation. The attacker feeds it with the victim's phone number to automatically communicate with RCS servers in the name of victims and intrude on their services. No matter what network the victim is under, the attacker could launch the attack remotely without accessing the victim's device. The implementation is composed of 3 major components, with 3000+ lines of code in Python.

- *Service logon.* This module hijacks the victim's RCS account to obtain his access tokens. It constructs spoofed requests using the victim's phone number and fake IMSIs, sends login requests to the 5G RCS profile server, and receives configuration files with access tokens. HTTPS connections are set up for communication.
- *OTP interception.* We deploy an application on the device side to acquire the OTP in experiments. An attacker could also acquire it from network-side SMS interceptions. Before hijacking starts, the fake client sets up a TCP connection with the app on the victim's phone. After sending a request to the 5G RCS server, the fake client pings the app to ask for OTP. The app then sniffs the OTP with OS APIs and share it with the logon module to complete verification.
- *Chat message exchanging.* We leverage this module for attacks in § V. We implement a simplified SIP [33] library to support basic message exchanging: REGISTER to set up a connection with the messaging server, INVITE to initiate the chat with another user, BYE to end the chat, to name a few. With the well-crafted signaling, the attacker could send and receive diverse messages supported by 5G RCS (text, locations, multimedia, etc.) with the victim's identity.

Validation Our validation shows that the attack is widely applicable. We have successfully hacked RCS accounts over 4

mobile carriers. Attackers need phone numbers only since they can counterfeit IMSI to pass the check. We validate that the attack could target a variety of phone models used by victims. We have launched successful attacks towards users with Google Pixel-5/4a/2 and OnePlus 7/8 Pro. The attack is not restricted to specific phone models; Moreover, attackers do not need to know the device model of victims. We issued an accepted request specifying Google Pixel-5 as the device model in the experiment, yet the victim is on a OnePlus phone. We also validate that attacks even apply to people who are not using RCS. Those users may disable RCS features, or their devices do not support RCS. We successfully created RCS accounts in the name of victims using Mi MIX 2 and OnePlus 7 Pro without RCS support.

C. Spoofing under End-to-end Encryption

The end-to-end encryption rolls out for the chat messages to further ensure the security of 5G RCS [7]. In addition to TLS encryption between clients and servers, RCS enhances the protection for user privacy since messages cannot be eavesdropped on or tampered with even by internal servers. End-to-end encryption is powered by a key server where users register, update and exchange public keys. Once a user logs on to RCS on a new device/app, the client will generate a pair of keys and share the public key with the server. If both users enable the function, plain messages between them will be encrypted using the keys [9]. In order to do spoofing, the attacker has to break the protection of end-to-end encryption after taking over the victim's account. Our study shows that one can bypass the challenge of end-to-end encryption with downgrade attacks or key replacement.

Downgrade attack A conversation will be downgraded to plain text for backward compatibility if either client does not support or enable end-to-end encryption. Therefore, one can bypass end-to-end encryption by claiming the incapability of clients. We deploy it within the Python-based attacker. Our experiments show that the attacker successfully skips end-to-end encryption and exchanges messages in plain text with the victim's contact.

Key replacement Another approach is to replace the original key of the victim after hijacking her account. Note that end-to-end encryption is set up after authentication. Since the attacker has already taken over the victim's account, he can update the key, retrieve the contacts' keys, and enable end-to-end encryption. However, it is still challenging to enable end-to-end encryption between the attacker (spoofing the user) and the contact by imitating the end-to-end encryption signaling. The application obfuscation prevents reverse engineering, and the key exchanging messages are in an unknown format.

Therefore, we exploited the implementation of messaging app to launch attacks. Specifically, we hooked the app on a rooted device to spoof the victim's account and replace the encryption key. Figure 4 illustrates the entire process. We used another rooted Android phone as the bot to launch the attack. We hooked the bot's messaging app using Frida [4].

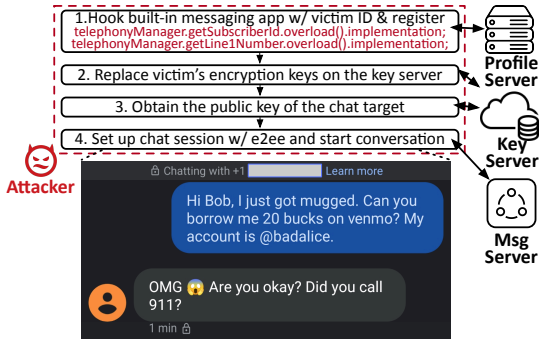


Fig. 4: Attacks end-to-end encryption via key replacement.

We fed the app with the victim's phone number and a fake IMSI. The hooked app first sent a logon request to the profile server and triggered SMS OTP during an attack. The bot passes verification and registers successfully by feeding the victim's OTP stolen from either the device or network side to the bot. After login in as a new user, the hooked app uploaded a new encryption key to the server. The new key is generated by the app thus could pass the key server checking. Before sending messages, the app also requested the public key of the target contact[1]. Until now, the attacker could set up a connection under end-to-end encryption with the victim's contact. Figure 4 shows the view of a successful attack where the lock sign under message bubbles indicates end-to-end encryption.

D. Persistent and Stealthy Attack

While our validation proves the feasibility of attacks, the current RCS system prevents persistent account hijacking. First, clients periodically re-connect to the server. If the RCS account is hijacked, the client could recover it no later than the subsequent reconnection. Second, the server enforces a rate limit on logon requests to prevent continuous attacks. We further devise a persistent and stealthy attack, making it hard to be perceived by victims.

Persistence of Attack We perform a dynamic reconnection in the attack to be persistent or long enough to cause damage. The attacker can still hack the account continuously if the rate limit is not as tight as the frequency of reconnection. We attempt a new attack immediately once detecting that the previous one is terminated by monitoring the incoming signaling message. If the new one succeeds, account hijacking could be persistent. In our experiments, persistent attack applies to OP-II, OP-III, and OP-IV. OP-I has a tight rate limit, and only one attack is permitted in a 10-hour window. However, there is sufficient time to intrude chat service and pose real damage in *one* attack. We conducted measurement over 4 carriers and Android OSes version 9 – 12; The validity period of one attack is approximately equal to the interval between consecutive connection refreshes, i.e., 2 – 3.5 hours.

Stealthy Connection Spoofing Furthermore, our attack is stealthy due to invisible SMS and lack of anomaly alert. With the hacked access token, the attacker can set up a

[1]According to our observation, this operation is performed once the user opens the dialog with the contact.

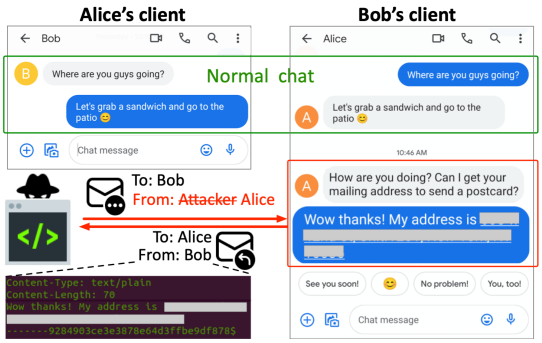


Fig. 5: Example of impersonation and fraud attack.

chat connection in the victim's name. As a result, the new connection will *replace* the original legitimate one, which is reasonable to accept only one phone per user. However, instead of dropping the old connection, RCS leaves it alive but unused. Consequently, the victim still assumes her chat connection is working well, while all messages targeting the victim are forwarded to the attacker only. We have validated the feasibility of spoofing chat connections stealthily over 4 U.S. carriers.

We attribute stealthy spoofing of chat connections to implementation flaws. To secure users in a better way, RCS should stay aware of potential threats and manage connections more wisely. Specifically, RCS should *explicitly* terminate the existing connection with the old device. Actually, it will benefit both security and usability. For security-wise, victims can get their accounts only if they are aware of the loss; For usability, it is even worse if the user stops receiving messages without being alerted a terminated service.

V. Component II: Exploit Rich Chat Services

In this section, we explore RCS's enriched service, such as location sharing, multimedia support, etc. We find that enriched services provided by RCS exacerbate the damage in real world. The attacker can launch impersonation-based fraud (§V-A), location tracking (§V-B), business account invasion (§V-C), and heavy data spamming (§V-D).

A. Text Chat: Impersonation-based Fraud

After hijacking the victim's RCS account, the attacker can impersonate the victim and send fraudulent messages to contacts. Figure 5 illustrates an example. The attacker breaks in the conversation between Alice and Bob and asks Bob for his mailing address. Bob's reply is forwarded to the attacker.

Unlike prevalent fraud from unknown numbers, it is much easier to obtain privacy or monetary gains by impersonating a friend or family. In case the recipients reply and ask for confirmation, the attacker will catch it and react accordingly. By handling such challenges, the attacker can further convince the recipients to take action as requested.

Moreover, the fraud is stealthy since the victim will never receive any message from her contacts, thus remaining unaware of underlying risks. The attacker can set up a new connection with hijacked accounts to replace the victim's original connection. Therefore, the messaging server will forward all messages

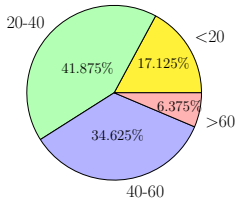


Fig. 6: Age distribution.

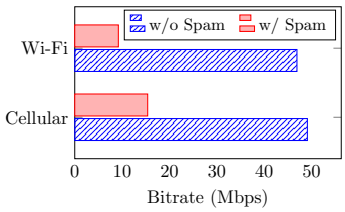


Fig. 7: Impact of spamming.

targeting the victim to the attacker. The attacker can further acknowledge those messages to make the sender believe in delivery to the right person. The entire attack is silent from the view of the victim.

Validation. We validate the practicality and effectiveness of impersonation-based fraud. We have successfully impersonated users from all 4 carriers. Unlike messaging spoofing, where attackers only send messages, we can further push it by reacting with the target. To evaluate the effectiveness of fraud, we conduct a user study by recruiting volunteers to take a questionnaire. We ask participants about their actions upon receiving short messages from a friend or unknown number asking for privacy or money. Those requests emulate common scenarios in our daily life to convince the recipients. We collect answers from 679 participants covering a wide range of ages, as shown in Figure 6. The fraud is very effective, according to the following findings.

- *Impersonating acquaintances greatly boosts the chance of successful fraud.* When receiving a short message asking for an address/money/gift from their family or friends, 11.5%/9.3%/4.6% of people will do it as requested immediately, as shown in Table I. Impersonation brings a considerable boost of 28.8×, 93×, and 11.5×, respectively, compared with fraud from unknown numbers. It also helps reduce the chance of failure (i.e., people take no action or make a phone call for confirmation).
- *The ability to interact with recipients via RCS live chat helps convince the recipients to take action as requested.* Impersonation-based fraud takes advantage of interaction with targets. The user study shows that 21.8%, 17.2%, and 19.4% of people need confirmation via live chat to request address, money, and gift, respectively. In total, the fraud will put 33.3%, 26.5%, and 24% of people at very high risk of privacy leakage, monetary loss in terms of money or gift.

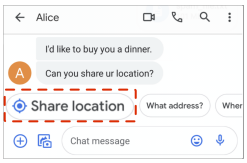
B. Location Sharing: Real-time Tracking

The attacker can exploit a new chat feature, location sharing, to track the victim's contacts in real-time. RCS clients support in-app retrieval of real-time location and sharing with `geolocationpush` message. The attacker can impersonate the victim as in §V-A to lure the victim's contacts to share real-time location. Moreover, location tracking is also stealthy since the spoofed connection has replaced the existing one between the victim and the victim's contact.

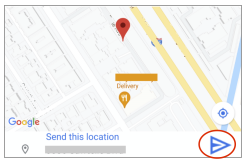
Validation. The feasibility is validated since this attack adopts similar techniques as impersonation-based fraud. In addition, we evaluate the effectiveness of location tracking.

TABLE I: Results of user study.

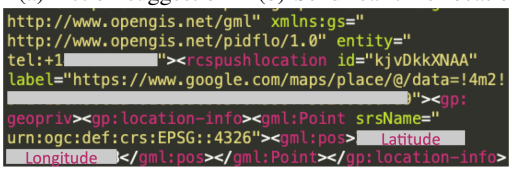
Requested item	From	Immediate action	Confirm via live chat	Confirm via phone call	No action
Home address	Friend/family	11.5%	21.8%	57.9%	8.8%
	Unknown number	0.4%	1.6%	6.0%	92.0%
Money	Friend/family	9.3%	17.2%	64.9%	8.6%
	Unknown number	0.1%	3.8%	10.6%	85.5%
Gift	Friend/family	4.6%	19.4%	61.9%	14.1%
	Unknown number	0.4%	1.8%	6.0%	91.8%
Real-time location	Friend/family	12.7%	22.5%	56.3%	8.5%
	Unknown number	0.6%	3.5%	6.8%	89.1%



(a) Action suggestion

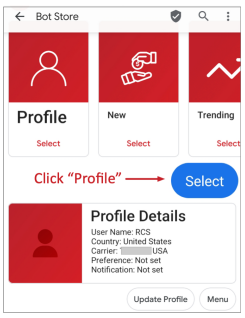


(b) Send real-time location

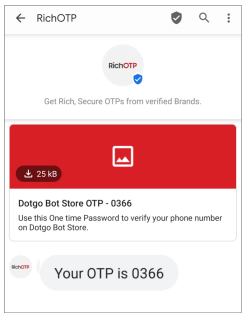


(c) GPS intercepted by attackers

Fig. 8: Location sharing via RCS.



(a) In-conversation actions



(b) OTP sent via RCS

Fig. 9: Exploit RCS chatbot functions.

The successful rate to track victims' contacts is pretty high. As shown in the user study (Table I), 12.7% of people would expose their locations immediately; 22.5% of people only need confirmation from live chat. Similarly, people tend to lower their guard when sharing real-time locations with contacts. Note that the location tracking is accurate as RCS uses GPS in plain text (Figure 8c).

C. Business Chatbots: Unauthorized Actions

More threats are emerging as RCS service providers are rolling out for business chatbots. The RCS chatbots directly connect users to a wide range of services, including merchants, restaurants, and banking, to name a few. In a conversation, users can take similar operations as on the webpage. Therefore, insecurities of RCS conversations would jeopardize the victim's accounts of other online services. Here we introduce two potential threats: unauthorized purchase/privacy retrieval and account take-over.

Unauthorized purchase/privacy retrieval Business chatbots enable quick actions in a conversation, like placing delivery orders, booking a flight, and viewing transaction history. Such an appealing feature makes it easier for users to access the services of a business and thus increases user engagement and sales. Pizza Hut and Booking.com have launched RCS chatbot in Europe [15]. However, the evil power to hijack RCS accounts would turn the new charm into a nightmare. Once the attacker intrudes on the victim's RCS service, he also acquires control of business accounts launched on RCS. Depending on the functionality of the business chatbot, the attacker might be able to place orders using the default payment method or retrieving account information.

We validated privacy retrieval on Bot Store [3], a platform providing chatbot service for businesses. It implements click buttons and menus to guide users. Figure 9a shows that the user can view his profile and initiate a new transaction by simple clicks in a conversation. In our experiment, the attacker exploited such convenience and successfully retrieved the victim's account profile.

Business account take-over Risks with chatbots are not limited to in-conversation actions. The attacker could completely take over the user's business account once he intrudes on the victim's RCS service. Along with the rolling out of business chat, OTP verification will be migrated from SMS to RCS (Figure 9b). Note that OTP is usually used as the *only* authentication factor for login or account recovery. Therefore, one can trigger new login or account recovery on behalf of the victim, intercept OTP from spoofed chat connection and then log onto the victim's business account. Note that the attacker could log in on websites and take complete control of the business account, not limited to chatbot-specific functions.

The risk is validated on Bot Store as well. First, the attacker intruded on the conversation between the victim and chatbot and waited for OTP. Then, he triggered authentication by clicking "forget password" on behalf of the victim. Immediately, the OTP through RCS was intercepted and used to log in to the business account.

D. File Sharing: Spam with Heavy Download

RCS supports file sharing in various forms, like photos and videos, which is a prominent appealing feature. RCS implements file transfer by media servers (Figure 1) with three major steps: (1) The sender uploads a file to the media server and obtains a URL. (2) The sender transmits a message, including the URL, to the receiver. (3) On receiving the message, the RCS client downloads the file from the media server.

However, one can abuse the implementation to launch a low-cost spamming attack with heavy download. While the uploaded file is accessible via the URL (generally 30 days), attackers can spam many users by repeatedly sending the URL. To make things worse, RCS clients automatically download large files without the user's consent. We find that system messaging app on Android 9 through 12 enables auto-downloading up to 105M for each file by default. In practice, the attacker can overwhelm the victim with parallel downloading tasks by sending messages at intensive frequency. To prolong the spamming attack, one probably hacks many RCS accounts to switching accounts before getting blocked by victims.

Validation In our test, we sent the URL of a 100MB file using the fake client and successfully triggered downloading on the target user's device. Then we repeatedly sent the URL to make the victim continuously busy with downloading. In our experiment, one URL message every 3 seconds is intensive enough to saturate the victim's network. Note that the cost of spamming is pretty low. Only one uploading from the attacker was sufficient throughout our test.

The attack would quickly use up mobile data and degrade the network performance for other applications. For data consumption, our investigation shows that most (58.4%) participants may suffer from data use-up or extra charge due to limited data plans; the threat is even more fatal for 30.6% of users whose monthly quota is no more than 5GB. To evaluate the impact on network performance, we play a 4K video on the test device under spamming attack for about 5 minutes and measure the average bitrate. Figure 7 shows that the spamming significantly hurt network performance for taking a large portion of bandwidth. The average bitrate of streaming decreases by 80.4% (from 46.9 to 9.2 Mbps) under Wi-Fi and 68.6% (from 49.1 to 15.4 Mbps) under cellular. Our experiments do not abuse or congest RCS servers, given only one user involved and low bandwidth consumed (below 50 Mbps).

VI. Solutions for RCS

We propose both long-term solutions and quick remedies to secure RCS based on the lessons from vulnerabilities and attacks. While an attacker has to complete three major components to pose real harm, hijacking an RCS account is the key. A long-term solution is switching to IDs confidential to the RCS app for secure binding (§ VI-A). Before it is launched, service providers should consider quick remedies to mitigate attacks temporarily (§ VI-B).

A. Long-term Solutions

We focus on user authentication to secure the entire system in the long run. While ID binding fits the goal of RCS authentication very well, currently adopted cellular IDs are prone to spoofing and counterfeiting because none of them are confidential to the 5G RCS app/users. Different from IM apps (e.g., Signal, WhatsApp, etc.), we leverage the unique features of RCS as a carrier service. We leverage built-in keys in the SIM for access from any device and any network. eSIM inherits SIM's functions and our solution works for both SIM and eSIM[2]. Our solutions still maintain good usability so that users can access the service with no memory-wise efforts and negligible physical operations.

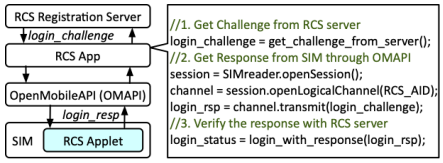
1) Authentication for smartphones: On smartphones, we leverage one advantage of RCS: the ability to access authentication resources of mobile carriers. As a carrier-based service, RCS has the option to adopt SIM-based ID, like what VoLTE does. An RCS-specific ID is issued by service providers and pre-installed onto the SIM. Thus, the secret ID is bound to the SIM holder. Those authentication functions can be built into the existing 4G/5G SIM applet. Since the mechanisms are used for authentication in 4G/5G and VoLTE, service providers can easily reuse the current infrastructure to RCS. In addition, SIM-based authentication can be extended to non-cellular networks like Wi-Fi. To protect communications via an untrusted network, we need to set up a secure channel with an authentic server. We can continue to use the existing RCS design mechanisms: enforcing TLS connection with public root certificate issued by recognized CA. Therefore, service providers could safely adopt this measure for user authentication under any network.

How does it resolve vulnerabilities? SIM-based cellular ID ensures the ID binding without hurting user experiences. First, the isolated hardware provides exclusive access to the ID for the target service. Specifically, the SIM applet strictly binds the cellular ID to the signature of RCS apps. Therefore, the cellular ID becomes a real "secret" to RCS service. The SIM only grants access to apps whose signatures are on a white list [1]. The app signature is created using the developer's private key and is hard to be abused by attackers [2]. More importantly, RCS users get authenticated automatically without any memory-wise or physical efforts. Thus, the usability maintains. In addition, the operators can quickly deploy the new applet over the air without hardware replacement.

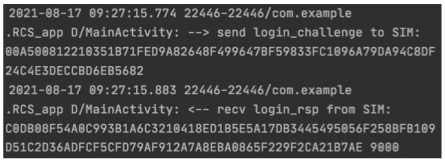
Implementation on clients We implement an RCS applet with Javacard SDK [8] on a programmable SIM card, together with a prototype RCS messaging app and RCS registration server to test the authentication procedure. As shown in Figure 10a, when the RCS app registers, it first retrieves the login challenge from the RCS server with the phone number. Then the app opens a session and forwards the challenge to the RCS applet through the OpenMobileAPI [10]. After the RCS applet calculates the response with the preset key and challenge, the app authenticates with the RCS server. We test the implementation on the Google Pixel 4a with Android 12. Figure 10b shows that we can successfully perform the SIM-based authentication for RCS registration.

2) Authentication for SIM-free devices: We propose to authenticate SIM-free devices by creating a binding to the verified, trusted smartphone. It is based on the RCS feature that access from SIM-free devices always relies on the assistance of the SIM holder's smartphone. We take the approach of pushing a confirmation dialog to the trusted smartphone and ask for the

[2]For simplicity, we use SIM to represent both SIM and eSIM later.



(a) Authenticate RCS client through SIM



(b) RCS-SIM authentication log

Fig. 10: Implementation of SIM-based authentication at client side.

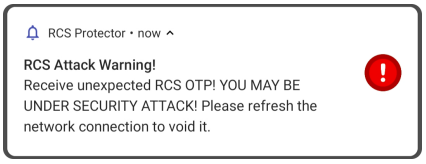


Fig. 11: Implementation of hijacking alert.

user's consent when attempting to access on a SIM-free device [20]. Moreover, we propose the management of those bindings on smartphones to control access from associated devices. On the smartphone, the user can see which devices are online and connected to servers. More importantly, the user can unbind any SIM-free device and immediately logs it out. To make the protection more efficient, RCS could support a blacklist of device IPs so that the user can always ignore the binding request from the same address.

How does it resolve vulnerabilities? The current verification binds with phone numbers with SMS OTP, which suffers from both device and network side attackers. Our confirmation way is secure for two reasons. First, the binding is only established with the authentic user's *physical* action. The confirmation message is not generated until the user clicks on "approve". Next, it is super hard to spoof the phone-issued approval. The binding enabled by SIM-based IDs (§ VI-A1) poses great challenges in impersonating the user on phone access. In addition, the connection between smartphone and server is secured, and thus, messages can hardly be tampered with or injected.

B. Quick Remedies

Considering severe damage caused by RCS insecurities, we propose the following quick remedies while deploying the fundamental solutions is necessary.

Enforce the binding with IMSI One solution is to enforce the binding between IMSI and RCS accounts. With Android version 10 or above, only system apps can retrieve IMSI from SIM. Therefore, attackers can hardly access it without root. It requires minimal modifications to the current design, given that the infrastructure already supports it. However, this solution still opens the attack surface since IMSI could be leaked by rogue base stations, etc. Attackers with more advanced technical support can still break into the victim's RCS account.

New login alert We propose a client-based alert to stop service intrusion after it happens. RCS account hijacking will trigger SMS OTP sent to the victim's device based on the current design. If it does not follow an RCS service request, we can probably infer the undergoing attack. We have implemented a prototype application for anomaly detection and alert. As shown in Figure 11, if the victim did not attempt to log on but received RCS OTP, the application will notify the user. The RCS messaging app does not need additional permissions to integrate anomaly detection and alert functions. Then the client could invalidate the attacker's login by refreshing the network connection, turning on and off airplane mode, or restarting the phone, etc.

Conservative configuration for downloading We suggest fixes for RCS clients to avoid spamming and keep the usability of file sharing. Generally, the client should adopt conservative default settings for file downloading. To reduce consumption of mobile data, the client could separate configurations for cellular networks and Wi-Fi. Upon receiving URLs of large files, the client asks for the user's approval and suggest him/her connecting to Wi-Fi if available. All remedies above are only concerned with RCS client implementation. The service provider could achieve better protection with a new software release.

VII. Discussion

RCS is different from regular mobile services, which rarely have substitutes better than OTP-based ID binding to balance security and usability. As one of the carrier services, 5G RCS did not make good utilization of carrier-powered security strengths, such as the secure SIM-based IDs in 4G/5G data access [16], VoLTE [27], etc. It overlooked such resources and thus caused insecurity.

Insights of RCS (in)security are not limited to carrier-based services but also can be generalized to mobile applications. First, the provided data for ID verification should be strictly confidential. While the weak bindings such as the SMS OTP are vulnerable, other authentication leveraging the in-SIM keys such as the Generic Bootstrapping Architecture (GBA) is much more secure [17]. Not restricted to mobile-carrier-based services, GBA is available to regular online applications and can provide a strictly private verification of phone numbers as account identity. Second, the authentication should apply multiple bindings in parallel with cross-checking. For example, RCS binds with phone numbers and IMSI but does not perform the cross-validation and fails to offer adequate protection. For general mobile applications, cross-checking such as Two-Factor Authentication [32] should be enforced for user authentication.

VIII. Related Work

To our best knowledge, this paper provides the first in-depth study of RCS system security. Several works related to RCS security focused on a specific system component, e.g., the messaging server [37], or operational slips [21]. Different from prior studies, we cover all the registration procedures and use scenarios corresponding to any network and any device support. In addition to analysis, our work validates the feasibility and real-world impacts of attacks. SMS security is well studied by a plethora of work [31], [25], [30], [34], [19]. Although the SMS and 5G RCS are both provided as carrier services, 5G RCS

security is a different problem with insufficient exploration, given completely different system architecture. There are many studies about ID binding security, including DNS spoofing [35], ARP spoofing [18], new attacks in SDN [24]. Our study starts on the weak ID bindings in 5G RCS and incites deeper thinking about how to apply ID binding to authentication for generic mobile services properly.

IX. Conclusion

As the official 5G messaging service, RCS is expected to become an essential platform for operators. In this work, we conduct an in-depth study on 5G RCS (in)security. We have uncovered several vulnerabilities and devised attacks by analyzing their standards and real-world implementation. The weak ID bindings allow for account hijacking. Even with the recent end-to-end encryption for RCS, adversaries may still breach the messaging service persistently and stealthy. The victims further suffer from impersonation-based fraud, location tracking, business account invasion, and heavy data spamming. We have validated such attacks with 4 US mobile carriers while following ethical requirements. We have designed and implemented both short-term remedies and long-term solutions. Lessons learned from our RCS study are also applicable to these apps and usage scenarios.

X. Acknowledgment

We greatly thank the reviewers for their insightful comments and constructive feedback. This work is partly supported by NSF CNS-1910150 and NSF CNS-2008026.

References

[1] Android access rule file (arf) support. https://source.android.com/devices/tech/config/uicc.
[2] Android application signing. https://source.android.com/security/apksigning.
[3] Dotgo bot store. https://dotgo.com/.
[4] Frida. https://frida.re/docs/android/.
[5] Get current mcc/mnc from mobile phone number. https://stackoverflow.com/questions/23888311/get-current-mcc-mnc-from-mobile-phone-number.
[6] Google completes global rollout of rcs for android. https://telecoms.com/507569/google-completes-global-rollout-of-rcs-for-android/.
[7] How end-to-end encryption in messages provides more security. https://support.google.com/messages/answer/10262381?hl=en.
[8] Java card platform, classic edition 3.0.5. https://docs.oracle.com/javacard/3.0.5/.
[9] Messages end-to-end encryption overview - technical paper. https://www.gstatic.com/messages/papers/messages_e2ee.pdf.
[10] Open mobile api. https://developer.android.com/guide/topics/connectivity/omapi.
[11] Protecting high-level personnel from imsi catchers. https://www.securitymagazine.com/articles/91767-protecting-high-level-personnel-from-imsi-catchers.
[12] Rcs set to revolutionize mobile messaging ecosystem. https://www.forest-interactive.com/newsroom/forest-interactive-rcs-set-to-revolutionize-mobile-messaging-ecosystem/.
[13] Rcs users to grow 294% by 2024, driving new revenue for mnos. https://www.interoptechnologies.com/news/2021/rcs-users-to-grow-294-by-2024-driving-new-revenue-for-mobile-operators.
[14] Why rcs? https://jibe.google.com/.
[15] Case study: Pizza hut delivery grabs a slice of rcs messaging. https://www.mobileindustryeye.com/intelligence/case-study-pizza-hut-delivery-grabs-a-slice-of-rcs-messaging-enabled-by-imimobile/, 2020.
[16] 3GPP. TS33.501: Security architecture and procedures for 5G System, Dec. 2020.
[17] 3GPP. 3GPP TS33.220: Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (GBA), 2021.
[18] Marco De Vivo, Gabriela O de Vivo, and Germinal Isern. Internet security attacks at the basic levels. *ACM SIGOPS operating systems review*, 32(2):4–15, 1998.
[19] Sarah Jane Delany, Mark Buckley, and Derek Greene. Sms spam filtering: Methods and data. *Expert Systems with Applications*, 39(10):9899–9908, 2012.
[20] Periwinkle Doerfler, Kurt Thomas, Maija Marincenko, Juri Ranieri, Yu Jiang, Angelika Moscicki, and Damon McCoy. Evaluating login challenges as adefense against account takeover. In *The World Wide Web Conference*, pages 372–382, 2019.
[21] BlackHat EU. Mobile network hacking – all-over-ip edition. https://i.blackhat.com/eu-19/Wednesday/eu-19-Yazdanmehr-Mobile-Network-Hacking-IP-Edition-2.pdf.
[22] GSMA. Rcs deployment. https://www.gsma.com/futurenetworks/rcs/.
[23] Silke Holtmanns and Ian Oliver. Sms and one-time-password interception in lte networks. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2017.
[24] Samuel Jero, William Koch, Richard Skowyra, Hamed Okhravi, Cristina Nita-Rotaru, and David Bigelow. Identifier binding attacks and defenses in software-defined networks. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 415–432, 2017.
[25] Nan Jiang, Yu Jin, Ann Skudlark, and Zhi-Li Zhang. Greystar: Fast and accurate detection of sms spam numbers in large cellular networks using gray phone space. In *22nd USENIX Security Symposium (USENIX Security 13)*, pages 1–16, 2013.
[26] Zeyu Lei, Yuhong Nan, Yanick Fratantonio, and Antonio Bianchi. On the insecurity of sms one-time password messages against local attackers in modern mobile devices. In *Proceedings of the 2021 Network and Distributed System Security (NDSS) Symposium*, 2021.
[27] Chi-Yu Li, Guan-Hua Tu, Chunyi Peng, Zengwen Yuan, Yuanjie Li, Songwu Lu, and Xinbing Wang. Insecurity of voice solution volte in lte mobile networks. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 316–327, 2015.
[28] Siqi Ma, Runhan Feng, Juanru Li, Yang Liu, Surya Nepal, Elisa Bertino, Robert H Deng, Zhuo Ma, and Sanjay Jha. An empirical study of sms one-time password authentication in android apps. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 339–354, 2019.
[29] Collin Mulliner, Ravishankar Borgaonkar, Patrick Stewin, and Jean-Pierre Seifert. Sms-based one-time passwords: attacks and defense. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 150–159. Springer, 2013.
[30] Collin Mulliner, Nico Golde, and Jean-Pierre Seifert. Sms of death: From analyzing to attacking mobile phones on a large scale. In *USENIX Security Symposium*, volume 168. San Francisco, CA, 2011.
[31] Bradley Reaves, Nolen Scaife, Dave Tian, Logan Blue, Patrick Traynor, and Kevin RB Butler. Sending out an sms: Characterizing the security of the sms ecosystem with public gateways. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 339–356. IEEE, 2016.
[32] Ken Reese, Trevor Smith, Jonathan Dutson, Jonathan Armknecht, Jacob Cameron, and Kent Seamons. A usability study of five {Two-Factor} authentication methods. In *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)*, pages 357–370, 2019.
[33] Jonathan Rosenberg, Henning Schulzrinne, Gonzalo Camarillo, Alan Johnston, Jon Peterson, Robert Sparks, Mark Handley, and Eve Schooler. Rfc3261: Sip: session initiation protocol, 2002.
[34] Yubo Song, Kan Zhou, and Xi Chen. Fake bts attacks of gsm system on software radio platform. *Journal of Networks*, 7(2):275, 2012.
[35] U Steinhoff, A Wiesmaier, and R Araújo. The state of the art in dns spoofing. In *Proc. 4th Intl. Conf. Applied Cryptography and Network Security (ACNS)*, 2006.
[36] Daehyun Strobel. Imsi catcher. *Chair for Communication Security, Ruhr-Universität Bochum*, 14, 2007.
[37] Guan-Hua Tu, Chi-Yu Li, Chunyi Peng, Yuanjie Li, and Songwu Lu. New security threats caused by ims-based sms service in 4g lte networks. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1118–1130, 2016.