BAKI ALİ NEFT MƏKTƏBİ
BAKU HIGHER OIL SCHOOL

Mobile development and security

Session 18

Karim Karimov
Lecturer

SMS security

The Short Message Service, often referred to as SMS or texting, is a standardized form of communication that allows two mobile phone devices to exchange short text messages of up to 160 characters. The SMS specification was originally written into the set of GSM specifications in 1985; however, since then it has been incorporated into other competing technologies, such as **Code Division Multiple Access** (**CDMA**).
New features such as the **Multimedia Messaging Service** (**MMS**) allow users to send pictures, audio, and video recordings to each other by building new functionality on top of the SMS layer. Additionally, cellular carrier operators have added new ways to perform administrative functions such as sending voicemail notifications, pushing updated settings, and even pushing updates to the mobile phone operating system itself by using SMS as a delivery mechanism. Although these new capabilities have greatly expanded the usage of SMS, they have also expanded the attack surface that SMS presents in today’s modern mobile phones.

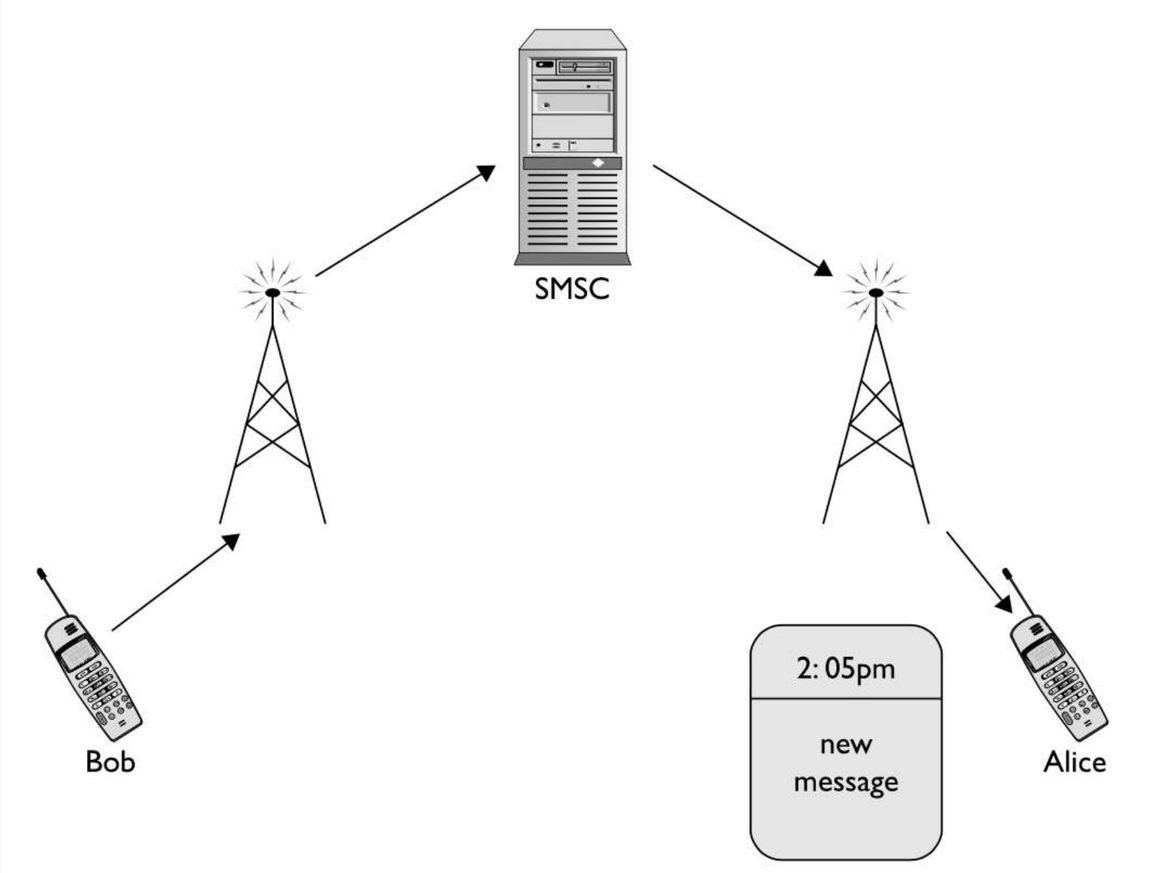
SMS security

For a significant portion of the time it has been deployed, there have been two reasons why SMS has been an incredibly difficult attack surface. First, mobile phones have traditionally not offered any easy way to obtain low-level control of the operating system of the device. An attacker wishing to try to compromise a victim’s mobile phone via SMS needs control over their own device first to be able to craft the carefully manipulated SMS messages used during the attack. Second, even if the attacker discovered a vulnerability in the mobile operating system of the victim’s phone, traditional mobile operating systems were closed-source proprietary systems that required an incredibly detailed understanding to be able to successfully execute malicious code on. However, both of these attack constraints have all but disappeared in recent years with the introduction of today’s modern mobile phones. Today’s phones allow attackers unprecedented control over the operating system.

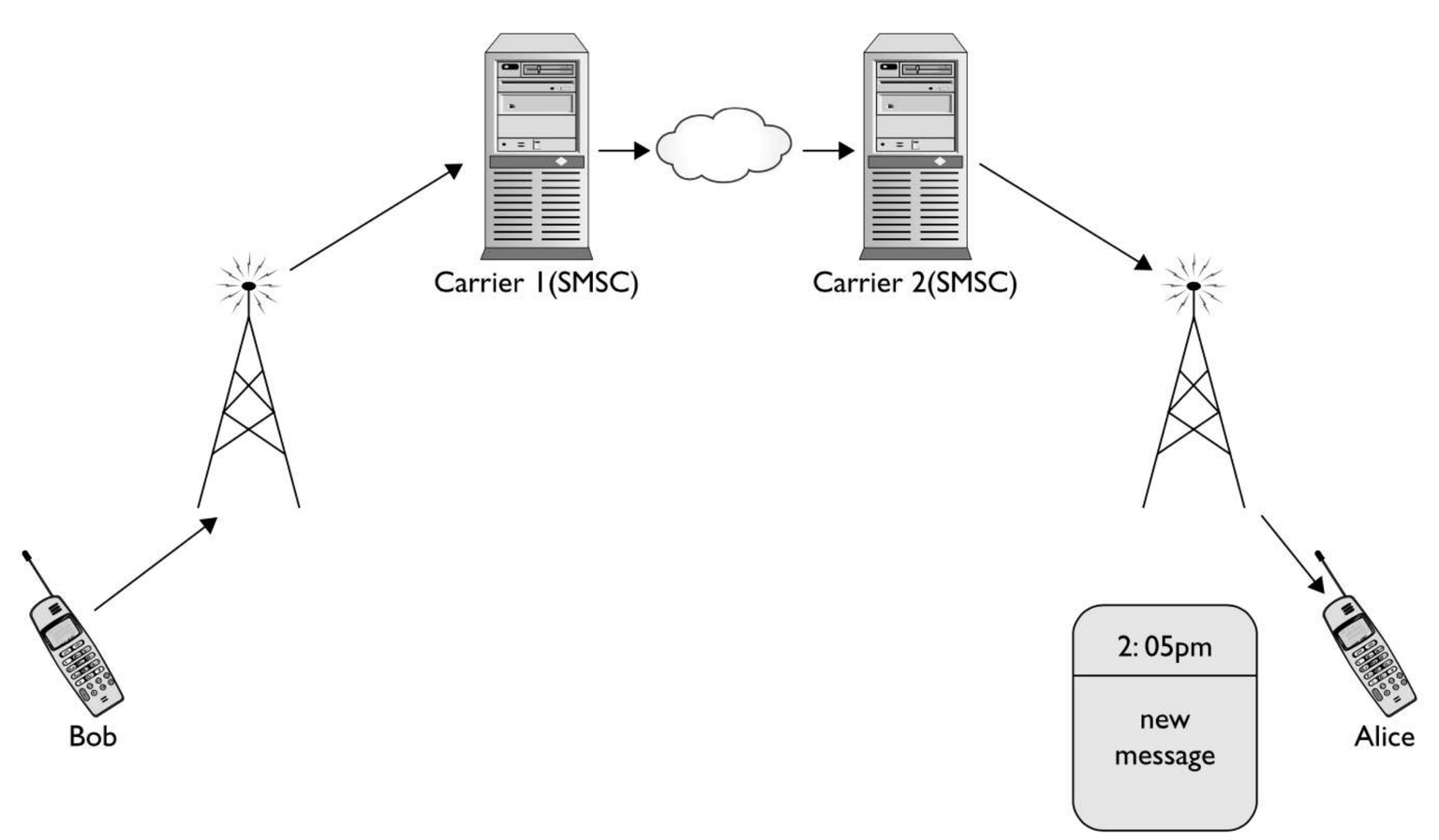
SMS overview

Short Message Service is a relatively straightforward system designed for one mobile subscriber to be able to send a short message to another mobile subscriber. The SMS system itself operates as a “store and forward” system within the carrier. This means that when mobile subscriber Bob wants to send a message to mobile subscriber Alice, the process involves a few (slightly simplified) steps. First, Bob composes the message on his mobile phone and then submits it to the carrier network. The server in the carrier network that handles the message is referred to as the short message service center, or SMSC. The SMSC receives the message from Bob and then checks to see if Alice is on the network and able to receive messages. If she is, the SMSC then forwards the message to Alice, who sees a new message appear on her mobile phone from Bob. The SMSC both stores incoming messages and determines when the messages can be forwarded on. This is what gives the system its name. This example has been slightly simplified, of course.

SMS message between phones using the same carrier



SMS message between phones on different carriers

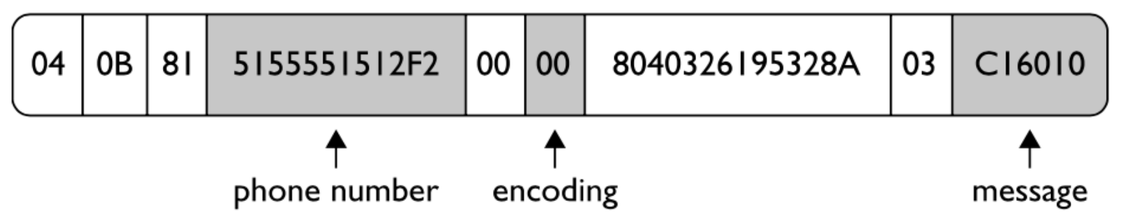


SMS payload

References to the contents of a message and bitmask header fields bring up an important question: ***What exactly does a raw SMS message look like?*** A raw SMS message is typically referred to as a protocol data unit, or PDU. A basic SMS PDU contains several header fields as well as the message contents. The header fields define a number of values that are essential in successfully delivering and understanding the message being sent. For example, one such value is the destination phone number of the message being sent. This value is what the SMSC uses to determine whom the message is being sent to. Another value is the message encoding type. SMS messages can typically be encoded in one of three encoding types: **GSM 7-bit**, **8-bit ASCII**, and **16-bit UCS2**. The GSM 7-bit encoding method is predominantly used in English language text messages. This encoding uses only seven bits per character instead of the normal ***ASCII eight bits***. Because a bit is shaved off from each character, a message can store extra characters, allowing the user to type more into a single message.

SMS payload

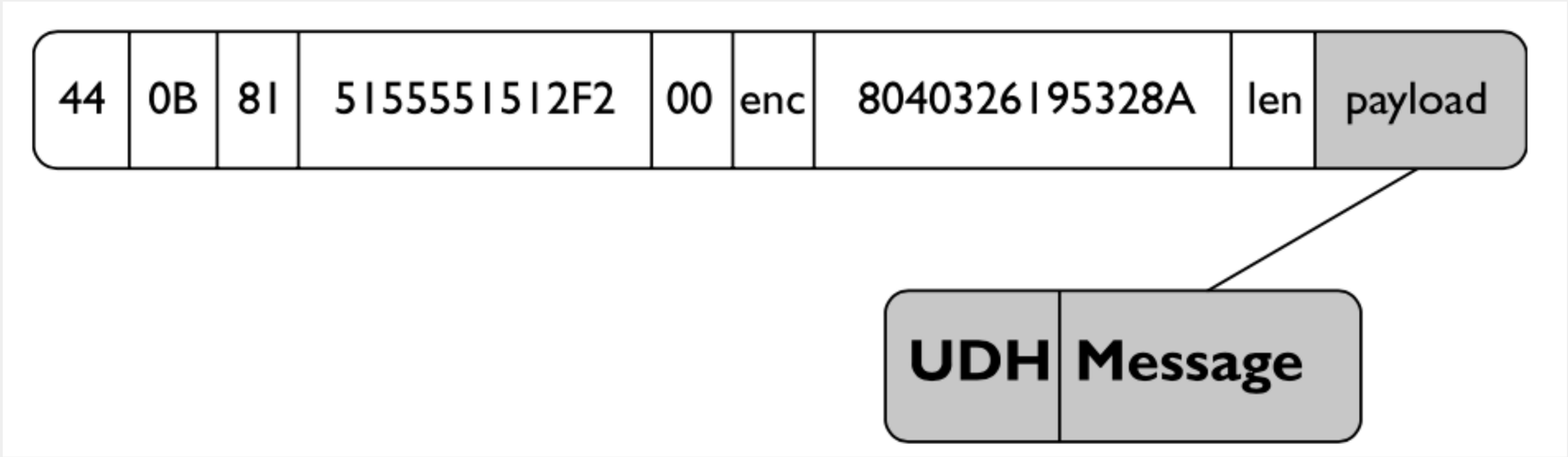
Figure below illustrates a basic SMS messages received from “1-555-555-1212” using GSM 7-bit encoding and containing the message “AAA.”



SMS payload

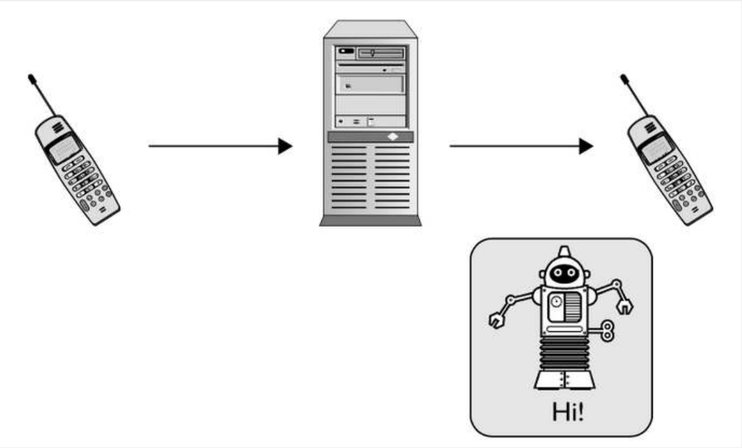
Arguably the most important field in an SMS PDU is the **User Data Header**, or **UDH**. The UDH allows an additional set of headers to be defined in the message contents portion of the SMS PDU. This is what has allowed such extensive functionality, such as multimedia messages, to be built on top of SMS. The ability for additional headers to be defined inside a message allows any amount of additional functionality can be built on top of SMS messages. This functionality does not always have to be as advanced as graphical messages, however. For example, one extremely common use of the User Data Header is to allow multipart SMS messages. When a user wishes to send a message that contains more characters than can be held by a single message, a User Data Header is defined that tells the receiving mobile phone that this message will be delivered in multiple parts. This multipart message UDH then contains all the information the receiving mobile phone needs to successfully reconstruct the different parts of the message into the one large message the user originally sent.

SMS payload


44
0B
81
515555151 2F2
00
enc
8040326195328A
len
payload
UDH
Message

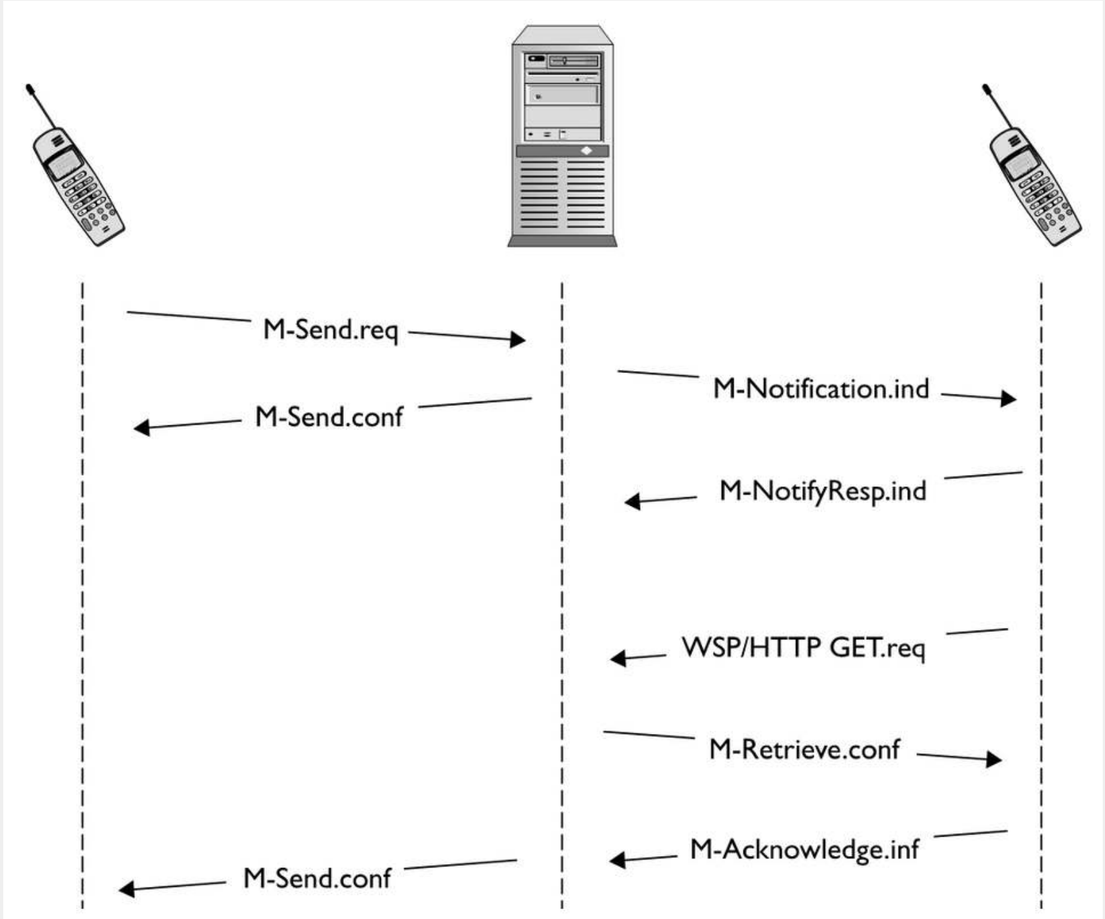
MMS overview

When SMS was first designed, it was only to send basic text content of a relatively small size. As with many technologies, however, it has since progressed far beyond its original design goals. Multimedia Messaging Service (MMS) is the next progression in the usage of SMS. MMS can send various types of images, audio, and video in addition to text. The demand for this functionality arose out of the changing nature of mobile phones themselves. As time progressed, mobile phones began to contain more and more functionality, such as the ability to record audio, take pictures, and even record video.



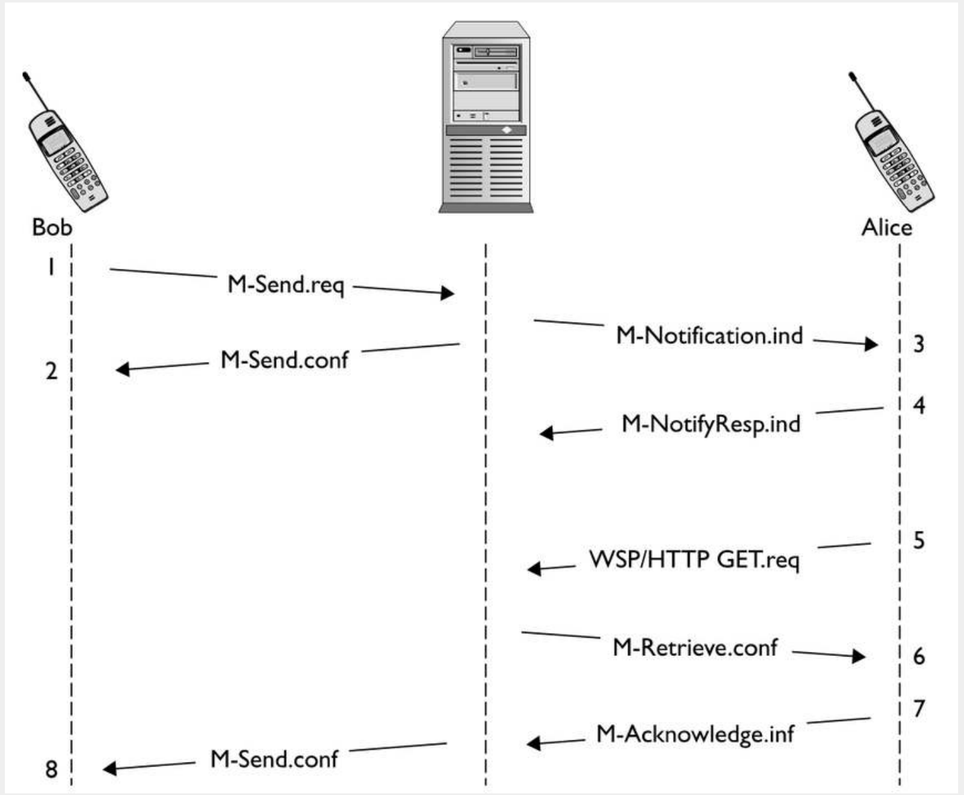
MMS overview

Although it may appear to the user that MMS is almost exactly like SMS, MMS is fundamentally different from SMS. From the mobile carrier perspective, MMS requires a far higher level of equipment and support. This is illustrated in Figure right, which shows the delivery of an MMS message with more details provided. In another example of its additional complexity over SMS, MMS does not use just one technology. Rather, several technologies are used throughout the creation and delivery of an MMS message.



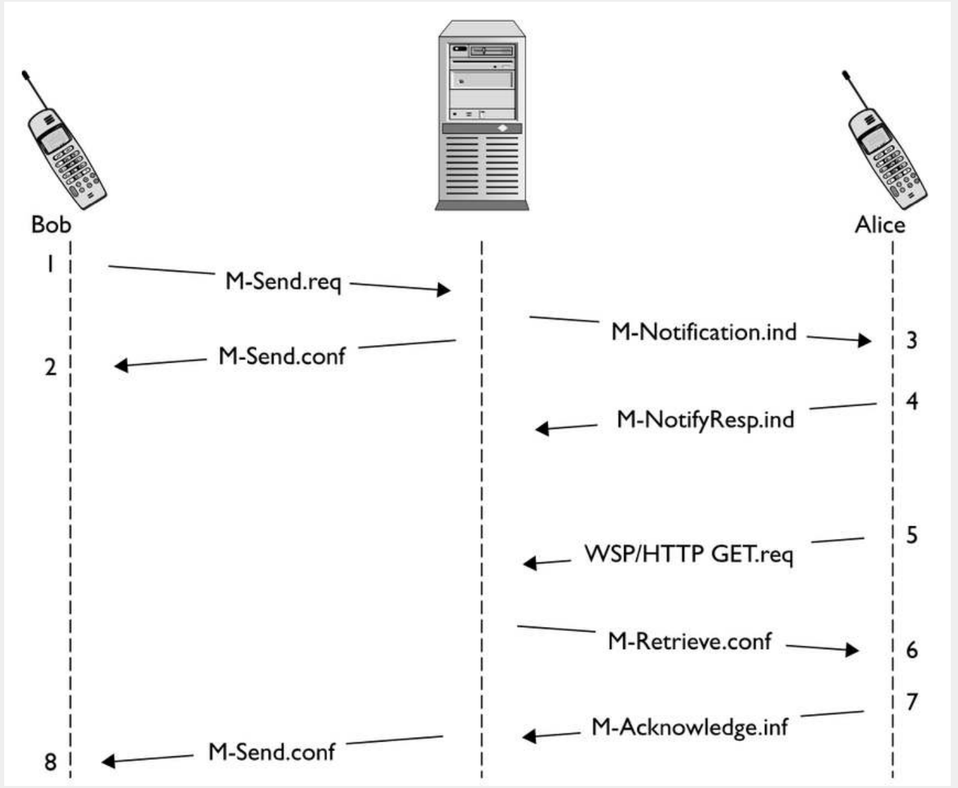
MMS Notification

1. Bob sends an MMS message. The underlying message is **M-Send-req**.
2. The Mulimedia Messaging Service Server (MMSC) confirms Bob's message and sends an **M-send-conf** message confirming the message.
3. The MMSC sends Alice a notification message, **M-Notification.ind**. The message contains an **URL**. Note that this URL is the key for various attacks.
4. Alice responds to the MMSC with an **M-NotifyResp.ind** message. This message is just a confirmation of message 3.



MMS Notification

5. Alice issues a **WSP** or **HTTP GET.req** message to the location provided in the notification message from message 3. This operation can happen after a time delay if user action is required. User action varies on different platforms.
6. The MMSC sends an **M-Retrieve.conf** message to Alice. The rest of the messages are background traffic, which isn't too relevant for attack purposes.
7. Alice sends the MMSC an **M.Acknowledge.inf** message. The message completes the MMS for Alice.
8. Finally, the MMSC sends Bob an **M-Send.conf** message.



Protocol Attacks

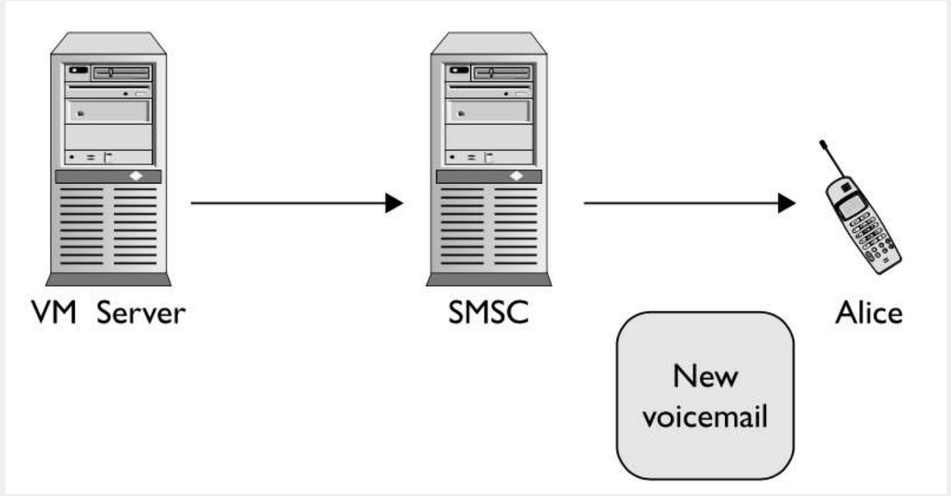
A number of attacks belong together under one logical heading of “protocol” attacks. However, these attacks can then be thought of as belonging to one of two subcategories.
The first of these subcategories is **abusing legitimate functionality**. These attacks do not exploit a previously undiscovered flaw in the implementation of software on a target mobile phone, but rather misuse legitimate functionality purposely built into the mobile phone.

Voicemail notifications occur after a voicemail has been left for a mobile subscriber. Once the carrier receives the voicemail, a notification must be generated to let the mobile subscriber know they have a voicemail waiting on the carrier’s voicemail server. This notification can be sent in one of several ways; however, the most commonly used method is to send the mobile subscriber a specially crafted SMS that informs the mobile phone that X number of voicemail messages are waiting. When the mobile phone receives this special message from the carrier, it displays a notification to the user, generally in the form of a pop-up or a graphic appearing on the phone.

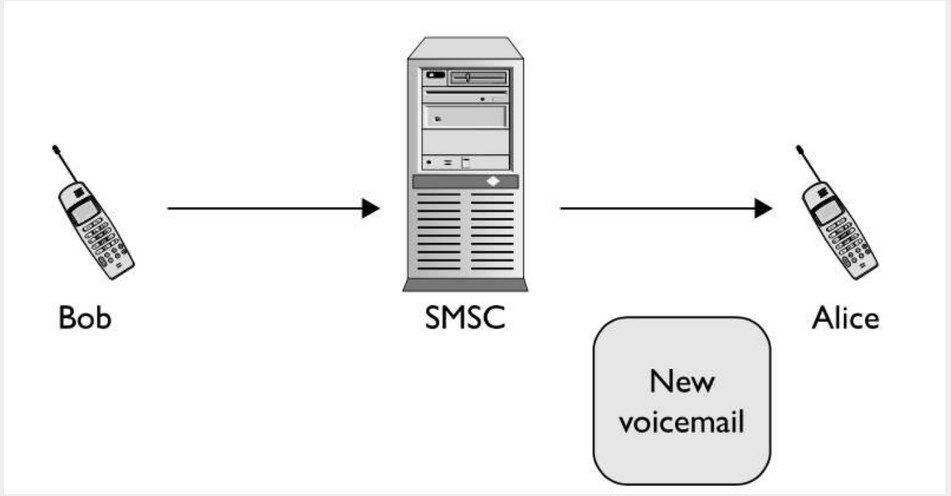
Abusing legitimate functionality

In this example, the voicemail notification is a special administrative message sent over SMS that is only supposed to be generated by the carrier's network. However, often there is nothing blocking an attacker from generating their own voicemail notification message and sending it to a victim. Although this could hardly be considered a serious attack, it serves to illustrate the point that there are legitimate administrative functions performed over SMS. These functions are meant to be sent from a carrier to a subscriber's mobile phone; however, typically nothing is in place to stop an attacker from spoofing these messages and sending them to victims' mobile phones.

Abusing legitimate functionality



Carrier-initiated voicemail notification



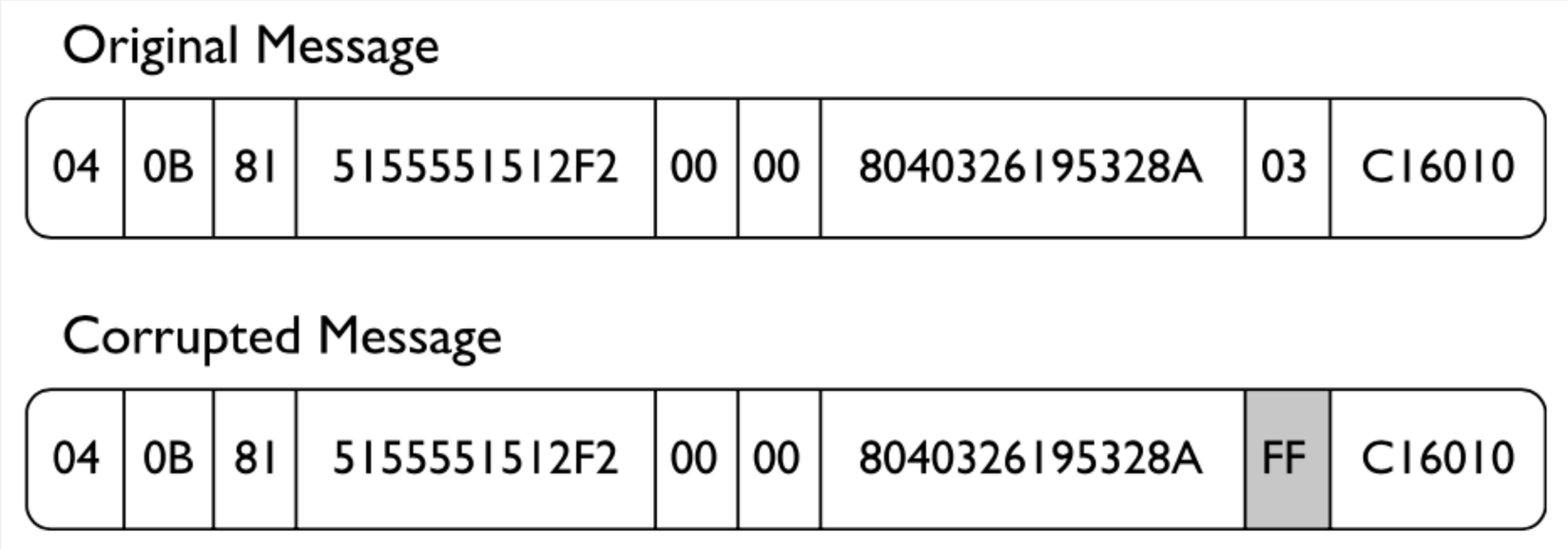
Spoofed voicemail notification

Vulnerabilities in the implementation

The second of these subcategories involves attempting to find **vulnerabilities in the implementations** of the popular SMS protocols. The goal of these attacks is to find a vulnerability in the SMS implementation of a mobile phone that would allow an attacker to send a corrupted message to a victim's phone that would result in the victim's phone running hostile code. Discovering these vulnerabilities often relies on corrupting, or fuzzing, otherwise valid SMS messages in such a way that triggers an error condition on the mobile phone. For example, in a basic SMS header, the length field tells the receiving mobile phone how many characters are stored in the incoming message. If an attacker can manipulate this length field to say there are a larger number of characters than there really are, an error condition could potentially be triggered.

Vulnerabilities in the implementation

Figure below illustrates a normal SMS with the length field first correctly set at 3 (03), followed by the same message with the length field manipulated to 255 (FF).

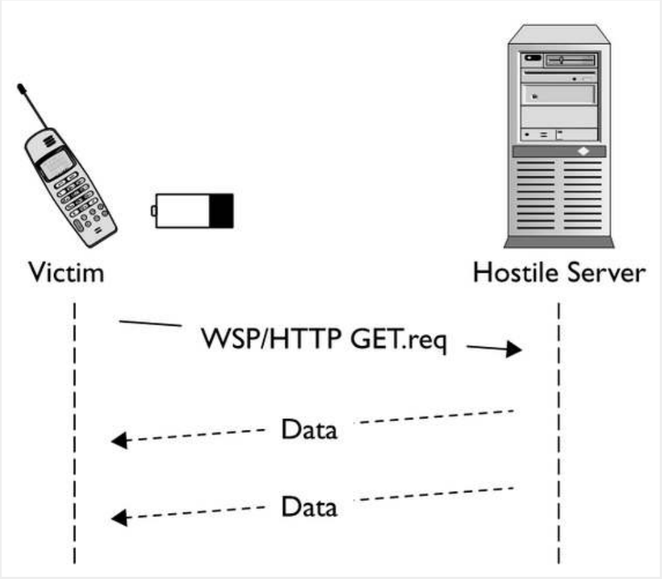


Battery-Draining Attack

One attack that builds off using MMS notifications is a battery-draining attack. The goal of a battery draining attack is to drain the battery of a victim’s phone without their knowledge in a manner that is far more rapid than normal usage, thereby knocking the victim’s, phone offline.
One of the easiest ways to perform this attack is by abusing the MMS notification functionality. To perform an attack via this method, an attacker crafts an MMS notification message that points to an attack server they control. In the case of a battery-draining attack, the bandwidth needed to perform the attack is negligible, which allows an attacker to use even a home DSL line to perform the attack.

Battery-Draining Attack

Once the MMS notification message has been constructed pointing to their hostile server, the attack sends the message to the victim. The victim's phone receives the message and automatically connects to the attack server to receive what it assumes will be a legitimate MMS message. Instead of returning a valid image file or video as part of a normal MMS, the attacker has instead configured their server to keep the victim's phone connected to the server indefinitely. The attacker can perform this in a number of ways, although the easiest method is by "pinging" the victim's phone with UDP packets. In this method, the attacker waits for the victim's phone to connect and then obtains the victim's IP address. The attacker then slowly sends UDP packets to the victim's IP address, which forces the victim's phone to stay online and keep the radio powered on.



Silent Billing Attack

Another attack that builds off the nature of MMS messages is the silent billing attack. This attack primarily targets mobile customers with prepaid mobile phones that depend on having a credit balance in their account. The goal of a silent billing attack is to silently drain the victims credit so that they are knocked offline and unable to perform further actions such as making or receiving calls.

Unlike text SMS messages, MMS messages have more overhead and involve several background messages to set up and confirm that an MMS has been successfully delivered. Because these background messages are only a small part of the process of an MMS message, mobile phones are programmed to not display messages of these types to the user. Instead, they are processed in the background as part of an MMS message, and if they refer to an invalid MMS message they are simply (from a user's perspective) silently ignored.

Silent Billing Attack

The silent billing attack takes advantage of two key facts about these types of messages: First, that these **messages are silently ignored** and not displayed to users. Second, that these messages are still perfectly **valid messages from the billing perspective** of the carrier's network. Therefore, an attacker can abuse these messages when they wish to deplete the balance of a victim who has a prepaid mobile phone account, unbeknownst to the victim. By bombarding the victim's phone with any of the background messages not displayed to the victim/user (for example, a ***Send Confirmation***), the attacker is able to rapidly wipe out the credit balance on the victim's account. The victim will not be aware that this attack has taken place because their phone didn't display any of the incoming messages. Thus, once the attack has been completed, the victim is unaware that their account is now empty and they will no longer receive legitimate incoming calls or text messages, in addition to being denied when attempting to place an outbound call or send a text message.

OTA Settings Attack

Over The Air (OTA) settings involve the ability of a carrier to push new settings to a customer's mobile phone on their network. Like SMS itself, the term OTA settings is actually a catchall that can refer to a number of different items. Everything from pushing new browser settings, to pushing firmware updates, to provisioning mobile phones for use on the carrier's network has been referred to as "***OTA settings.***"
One example for this kind of attack is pushing a new WAP browser settings to a target mobile phone. The goal of this attack is to install new settings into the browser configuration of the target mobile phone. If the attack is successful, the victim's browser will then route all traffic through a proxy that the attacker controls. The attacker is then able to sniff the connection to obtain personal information about the victim, as well as to perform **man-in-the-middle attacks** against the victim's traffic.

End of the session