1.

a) Kb is the symmetric key between Bob and KDC, so only Bob can decrypt the ticket to Bob. Only KDC can create this.

b) The ticket to Bob can only be created by KDC, so including “Alice” in this message can let Bob know who will talk with.

c) SA is used for authentication and ensuring the confidentiality/integrity.

d) If it is sent directly to Bob before Alice initiates contact with Bob, Bob need to remember KAB for waiting. This is not statelessness.

2.

a) To begin the authentication process, a small amount of encrypted data that is issued by a server in the Kerberos authentication model is known as TICKET-GRANTING TICKET (TGT). They are used to obtain tickets that enable access to network resources

b) When Alice present her TGT to the KDC, the KDC can decrypt it, Alice can then use her TGT (credentials) to securely access network resources

c) Because K KDC is only known by KDC, so others can only get TGT, but cannot forge it

d) Extra work and no added security

3.

Insecurity:

1. Hash used for A3/A8 is COMP128 has been broken by 160,000 chosen plaintexts and with SIM, can get Ki in 2 to 10 hours.

2. Encryption between mobile and base station but no encryption from base station to base station controller this often transmitted over microwave link.

3. Encryption algorithm A5/1 has been broken with 2 seconds of known plaintext.

4. Attacks on SIM card: Optical Fault Induction¾can attack SIM with a flashbulb to recover Ki.

Partitioning Attacks¾using timing and power consumption, can recover Ki with only 8 adaptively chosen “plaintexts” With possession of SIM, attacker can recover Ki in seconds

5. Fake base station exploits two flaws: Encryption not automatic and Base station not authenticated.

6. Denial of service is possible: Jamming (always an issue in wireless)

7. Base station can replay triple (RAND, XRES, Kc):One compromised triple gives attacker a key Kc that is valid forever. No replay protection.

Modify: Mutual authentication

Integrity protect signaling (such as “start encryption” command)

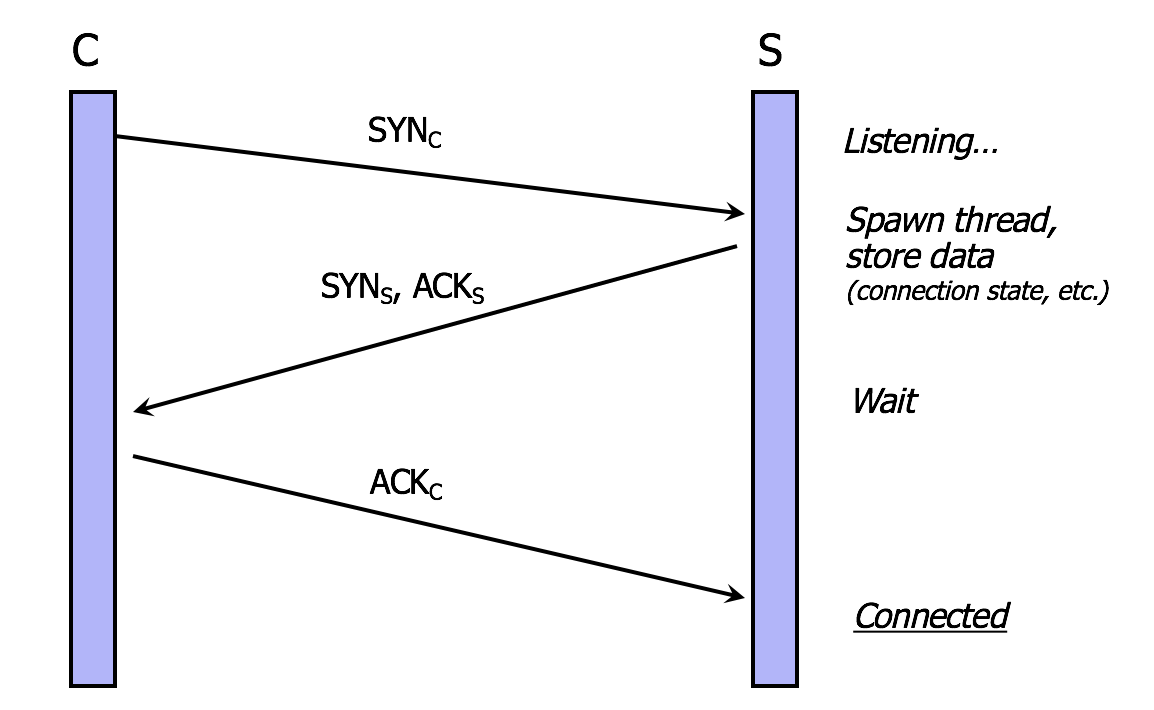
Keys (encryption/integrity) cannot be reused

Triples cannot be replayed

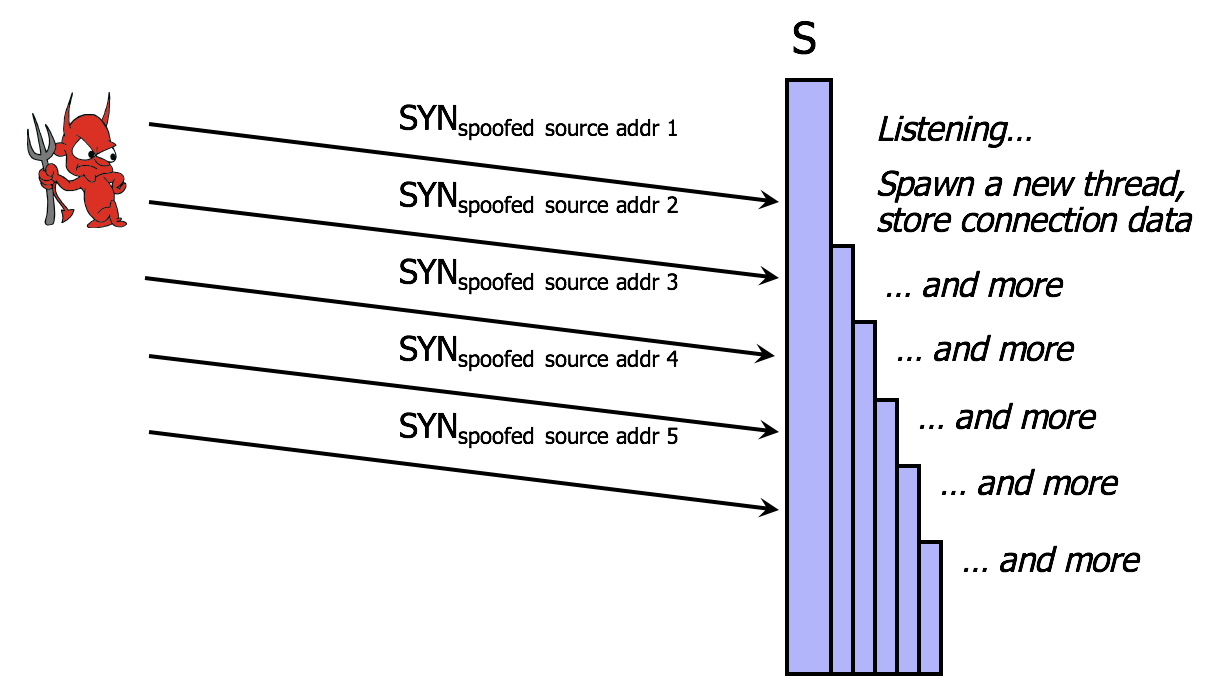
Strong encryption algorithm (KASUMI)

Encryption extended to base station controller

4. TCP Handshake figure:



SYN Flooding Attack figure:



Explanation:

1) Attacker sends many connection requests with spoofed source addresses

2) Victim allocates resources for each request

- New thread, connection state maintained until timeout

- Fixed bound on half-open connections

3) Once resources exhausted, requests from legitimate clients are denied

4) This is a classic denial of service pattern

- It costs nothing to TCP initiator to send a connection request, but TCP responder must spawn a thread for each request – asymmetry.

5.

The reflection attack uses normal behavior of network. In this scenario attacker sends packet with spoofed source address being that of target to a server•server response is directed at target. If send many requests to multiple servers, response can flood target. It includes various protocols such as UDP or TCP/SYN. In ideally they want response larger than request.

The general attack outline is as follows: The attacker initiates a connection to a target. The target attempts to authenticate the attacker by sending it a challenge. The attacker opens another connection to the target, and sends the target this challenge as its own. The target responds to the challenge. The attacker sends that response back to the target on the original connection. If the authentication protocol is not carefully designed, the target will accept that response as valid, thereby leaving the attacker with one fully authenticated channel connection.

This DDoS attack is a reflection-based volumetric distributed denial-of-service (DDoS) attack in which an attacker leverages the functionality of open DNS resolvers in order to overwhelm a target server or network with an amplified amount of traffic, rendering the server and its surrounding infrastructure inaccessible.

All amplification attacks exploit a disparity in bandwidth consumption between an attacker and the targeted web resource. When the disparity in cost is magnified across many requests, the resulting volume of traffic can disrupt network infrastructure. By sending small queries that result in large responses, the malicious user is able to get more from less. By multiplying this magnification by having each bot in a botnet make similar requests, the attacker is both obfuscated from detection and reaping the benefits of greatly increased attack traffic.

In a reflection DDoS attack, also called a DrDoS attack, there are three types of participants: the attacker, victim servers that act as unwitting accomplices, and the attacker’s target. The attacker sends a simple query to a service on a victim host. The attacker falsifies (spoofs) the query, so it appears to originate from the target. The victim responds to the spoofed address, sending unwanted network traffic to the attacker’s target. Attackers choose reflection DDoS attacks where the victim’s response is much larger than the attacker’s query, thus amplifying the attacker’s capabilities. The attacker sends hundreds or thousands of queries at high rates to a large list of victims by automated the process with an attack tool, thus causing them to unleash a flood of unwanted traffic and a denial of service outage at the target.7604  
A single bot in a DNS amplification attack can be thought of in the context of a malicious teenager calling a restaurant and saying “I’ll have one of everything, please call me back and tell me my whole order.” When the restaurant asks for a callback number, the number given is the targeted victim’s phone number. The target then receives a call from the restaurant with a lot of information that they didn’t request.  
As a result of each bot making requests to open-DNS resolvers with a spoofed IP address, which has been changed to the real source IP address of the targeted victim, the target then receives a response from the DNS resolvers. In order to create a large amount of traffic, the attacker structures the request in a way that generates as large a response from the DNS resolvers as possible. As a result, the target receives an amplification of the attacker’s initial traffic, and their network becomes clogged with the spurious traffic, causing a denial-of-service.

6.

a) Consider a server system with a table of 256 TCP (Transmission control protocol) connection requests and it sends the SYN-ACK packet five times if it fails to receive the ACK packet in response at 30 second intervals before the removal of request from its table.

- The number of SYN-ACK packet is 6(initial + 5 repeats = 6). Consider the following formula to calculate the rate at which the attacker continues to send the TCP connection requests.

{Rate of the attacker to send the TCP connection requests}

= {Number of SYN-ACK packets x Time interval taken to send the SYN-ACK packet}

Substitute the “Number of SYN-ACK packets” is “6”, and “Time interval taken to send the SYN-ACK packet” is “30 seconds” in the above formula.

{Rate of the attacker to send the TCP connection requests} = 6 x 30 = 180

Conversion of seconds to minute:

{Rate of the attacker to send the TCP connection requests} = 180/60 = 3 minutes

Number of requests that the attacker need to send through TCP connection:

Consider the following formula to calculate the number of requests that the attacker needs to send to ensure that the table remains full.

{Number of requests that the attack need to send through TCP connection} =

(number of connection requests in table} / {Rate of the attacker to send the TCP connection requests}

Substitute the “Number of connection requests in table” as “256” and “Rate of the attacker to send the TCP connection requests” in the above formula.

{Number of requests that the attacker need to send through TCP connection} = 256/3 = 85.333 = 86

Therefore, to make the table full, the attacker needs to send about 86 TCP connection requests per minute.

b) Required Bandwidth:

Assume that the TCP SYN packet is 40 bytes in size and use the following formula to calculate the required bandwidth for the attacker to continue the attack.

{Required bandwidth for the attacker to continue the TCP SYN spoofing attack} =

{Number of requests that the attacker need to send through TCP connection \* Size of TCP SYN packet in bytes \* Number of bits in each byte}/ {Number of seconds per minute}

Substitute the “Number of requests that the attacker needs to send through TCP connection” is “86”, “Size of TCP SYN packet in bytes” as “40”, “number of bits in each byte” as “8”, and “Number of seconds per minute” as “60”。

{Required bandwidth for the attacker to continue the TCP SYN spoofing attack} = 86\*40\*8/60 = 27520/60 = 458.66 bits/sec

Therefore, the required bandwidth for the attacker to continue the TCP SYN spoofing attack is 458.66 bits/sec which is a negligible amount.

7. Three lines of defense against DDoS are attack prevention and preemption, attack detection, mitigation, filtering and attack source traceback and identification block spoofed source addresses.

Attack prevention includes block spoofed source addresses such as on routers as close to source as possible but it is still far too rarely implemented. Rate controls in upstream distribution nets with on specific packets types, e.g. some ICMP, some UDP, TCP/SYN It also uses modified TCP connection handling and use SYN cookies when table full or selective or random drop when table full.

Attack Mitigation includes client puzzle

Let M = a SYN request message

R = value to be determined

T = current time

Sender must find R so that h(M, R,T) = (0 0…0,X), where N initial bits of hash value are all zero

Sender then sends (M, R,T)

Recipient (server) accepts the request, provided that h(M, R,T) begins with N zeros

8.

a) Over the past several months, we've seen major breaches exposing numerous usernames and passwords. The eBay and Adobe breaches impacted millions of accounts. Snapchat was compromised. With every password breach comes the inevitable question: Were the passwords stored securely? Unfortunately, this simple question is not simply answered.

Though hashing and encryption both provide valuable capabilities, for the vast majority of situations, there is only one right option for storing user passwords for an online application: hashing. This is a one-way function in which a hashed value cannot be reversed to obtain the original input value (i.e., the password). Symmetric encryption is based on the use of an encryption key and is a reversible operation. Anyone possessing the key can decrypt an encrypted value to obtain the original value. To store user passwords safely, it is critical to understand the differences between symmetric encryption and hashing. Algorithms such as PBKDF2, bcrypt, and scrypt all utilize per user salts and iterative hashing capabilities to store passwords securely. The Internet and web applications will become even more important parts of everyday life as they are trusted with increasingly sensitive information. It is imperative that developers and application owners ensure they are doing everything they can to secure user information. This starts with a proper understanding of fundamental security controls and the protection of user passwords using modern hashing algorithms.

b) This may be a bit counterintuitive, but older hashing algorithms are too fast. Attackers may not be able to reverse a hashed value to obtain the original input, but they could perform a brute force attack that hashes numerous possible password inputs and checks if they match any of the stolen hashes. To combat this threat, modern hashing algorithms can perform multiple iterations. This introduces a minor delay for a single hashing operation, but this small delay becomes massive if an attacker is performing a brute force attack. Consequently, iterative hashing algorithms make brute force attacks unrealistic, since they would require hundreds of years or more to complete By design, symmetric encryption is a reversible operation. This means that the encryption key must be accessible to the application and will be used for every password verification. If the encrypted passwords are stolen, the attackers only need to determine the symmetric key used by the application. Once that key becomes known, through a breach or through brute force attacks on a weak key, all passwords are instantly decrypted and accessible. This is not a good place to be.

c) In cryptography, a salt is random data that is used as an additional input to a one-way function that hashes a password or passphrase. The primary function of salts is to defend against dictionary attacks versus a list of password hashes and against pre-computed rainbow table attacks. The reason that salts are used is that people tend to choose the same passwords, and not at all randomly. Many used passwords out there are short real words, to make it easy to remember, but this also enables for an attack. Passwords are generally not stored in cleartext, but rather hashed. If you are unsure of the purpose of a hash-function, please read up on that first. Now, what the attackers can do is to simply generate a list of common passwords and their corresponding hashes. Comparing the hashes that a site has stored with the table will, if common passwords are being used, reveal the passwords to the attacker. A salt is simply added to make a common password uncommon. A salt value is generated at random and can be fairly small, the only purpose is to lower the probability that the hash-value will be found in any pre-calculated table. A common way to combine the salt and the password is to simply concatenate them, i.e. the stored hash value is Hash(salt||password). The common password password1 now magically becomes, e.g., 6$dK,3password1and is very unlikely to be found in a table. The salt can be stored completely in the clear in the database, next to the hashed value. Once the attacker has the database and wants to find the passwords, he needs to generate the pre-calculated table for each salt individually, a costly operation. Yet another way to defend against this kind of attack is to slow down the attacker. This can be archived by iterating the hash-function many times, i.e. storing Hash(Hash(Hash(Hash…(Hash(salt||password)))…). Also, a pepper can be used, which is another random value concatenated to the password, such that the stored value is Hash(pepper||salt||password). The pepper is then not stored at all and therefore all possible values of the pepper need to be tested when trying to log in. Using 8 bits for the pepper give 256 possible values, which is very fast when the true user tries to log in. However the attack will work 256 times slower since all pepper values need to be tested for each password.

When using per-user salts an attacker cannot simply review the stolen password hash database for duplicate hashes (which would indicate the same original password for both accounts). The introduction of a per-user salt ensures that even the same password will result in unique hashes.

An attacker cannot download a rainbow table and use it against the password hashes. A rainbow table is a large database of precomputed hashes for a variety of common passwords (or even all possible passwords of certain character sets and lengths). Without per-user salts an attacker could do a simple lookup of the stolen hash within the rainbow table to determine the original password. The introduction of per-user salts means the rainbow table is useless.

9.

1) here we have 26 character so that 4-character combination = 26^4 = 456,976 sec.

if we will think about the worst case, we will assume, you had to exhaust every combination until the end. so that in Average = 456,976 sec / 2 = 228,488 secs.

2) in the worst case total time = 26\*4 = 104 sec. Average case = 104 / 2 = 52 secs.

10.

a) More specifically, SunOS-4.0 introduces the file /etc/publickey that contains a user's network name A, the user's public key +Ka, and the corresponding private key -Ka. This private key is encrypted using DES with a key derived from the user's login password p. When A logs in, the login or rlogin command prompts for the password and decrypts {-Ka}p from A's entry to obtain -Ka; since +Ka and -Ka are mutual inverses, the value -Ka can be checked to validate that p was correctly supplied. Later the key pair (+Ka, -Ka) is used to establish temporary keys for remote procedure calls.

b) Since the file /etc/publickey is publicly readable, an attacker can guess p (say p') and compute -Ka' = Now he can choose an arbitrary message x and check that = x; if so, it is highly probable that p' = p. Moreover, -Ka = -Ka'. The attacker can conduct as many additional tests as needed by choosing new values for x to eliminate the chance of a false positive test. A possible solution is to restrict access to the file /etc/publickey as for the file /etc/passwd.

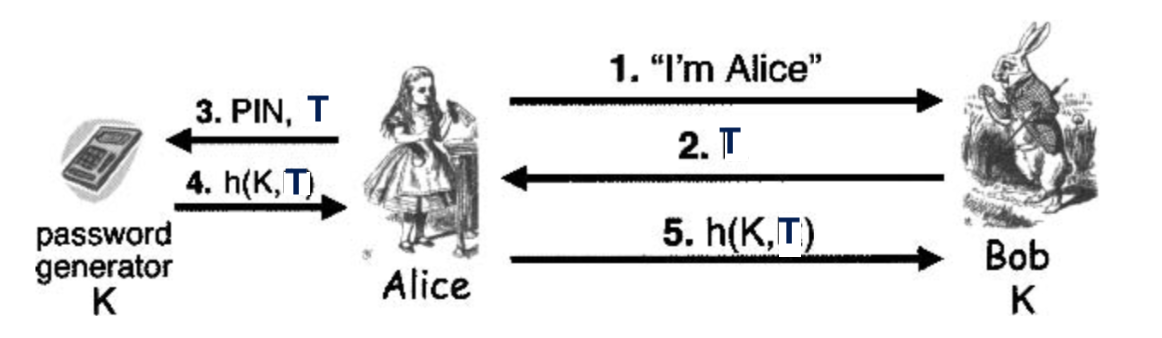
11.

a) We would expect to find 10^5 \* 10^7/ 10^10 = 100 false matches.

b) The probability is 10^7 / 10^10 = 1/1000

12.

a)



b) If there is no challenge T, then the response is always the same, h(K) , which would eliminate the need for Alice to actually posses the SecurlD device. So, it would not be a "something you have" form of authentication. Also, if Trudy observed one response, she would be able to replay that response forever to (mis)authenticate as Alice.

c) By using T, we save one message--there is no need to send the challenge. Note that T acts like a challenge R that Alice knows, so there is no need to send it. The downside to using T is that time becomes a security issue--if Alice and Bob's clocks are out of sync, by more than a minute, they cannot communicate. This opens up a new avenue of attack for Trudy.

d) Both are equally secure if implemented correctly.

13.

If the canary value is overwritten then there is likely a problem with the return address.

14.

a) The results are buff2 = 22222222 and buf2 = 11122222

b) Apparently, buffer 1 has overwritten the start of buffer 2.

c) Trudy might be able to overwrite some important data to, say, change a failed authentication into a successful authentication.

15.

a) If len is negative there is a problem. The test in the if will be passed, but then memcpy assumes that len is unsigned. So, a negative value for len is interpreted as a very large number, which would lead to a buffer overflow.

b) Explain how an integer overflow might be exploited by Trudy. Trudy can overflow an array and cause problems.