A Practical Attack Against GSM

COMP 4019

Aaron Chen
100832149

GSM.

GSM or Global System for Mobile Communications is a second generation (2G) mobile cellular network standard. First rolled out in 1991, it was the first digital standard and added data transfer features such as SMS (Texting). What makes it relevant to this course is that it was also the first cellular network to use cryptography. The purpose of this cryptography was to authenticate users of the networks for billing purposes, user privacy being secondary. This design choice has ramification that will later be expanded on. While over a decade old, it is still in wide use, only partially phased out in western countries by newer standards, and powering the vast majority of networks everywhere else. It is estimated to currently in 2014 have over 5 billion subscribers, making it the most ubiquitous form of cryptography in the world.

Authentication.

GSM encryption begins with a 128-bit key (Ki) that is registered to the subscriber when they first sign up with a provider. This is the shared secret between the subscriber and provider and meant to be semi-permanent. When a subscriber connects to a station for the first time, a 128-bit random challenge is sent to subscriber. Using the Ki and challenge, a signed response and a 64-bit session key is generated using algorithms know as A3 and A8 respectively. When the provider receives the expected challenge response, the user is authenticated and encrypted communication begins.

Encryption.

Data transmission between the provider and subscriber is done using the stream cipher A5, which consists of 3 version A5/0,A5/1,and A5/2. A5/0 is a dummy cipher that provides no encryption while A5/2 is the intentionally weakened export version of the main A5/1 cipher. A5/2 has been shown to be

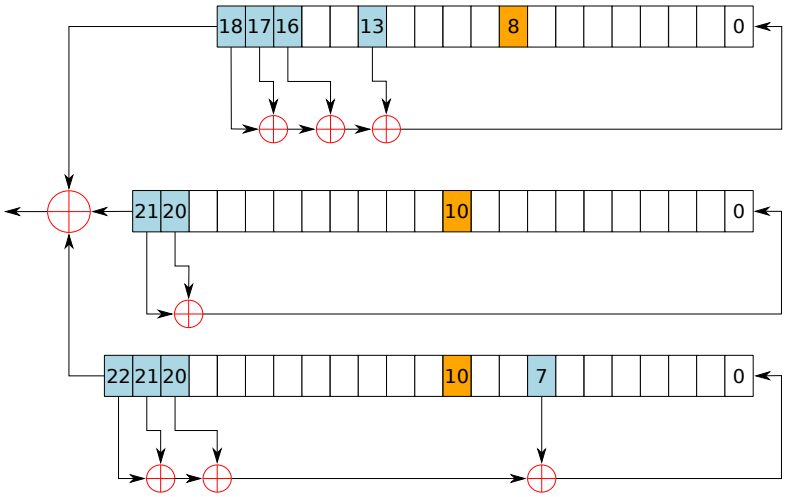
incredibly weak and is no longer recommended for use anywhere, which along with laxer cryptographic export laws, leaves A5/1 to be the default version in use. Encrypted data between the subscriber and provider is broken down into 2 114-bit “bursts” inside of a larger “frame”. The 64-bit key is input along with the 22-bit frame ID are used as inputs for the A5/1 stream cipher and 228 bits are generated which are XORed with the two bursts to form the encrypted data. Decrpytion is done using the exact same process, the session key and frame ID are inputted into the A5/1 stream cipher and a bit stream is generated. This is then XORed with cipher text to get the plaintext.

MITM.

As I previously noted, the design of GSM to prioritize user authentication for billing opens itself up to man in the middle attacks. As authentication is unidirectional, spoofing a provider station is trivial. When a user first connects, the spoofed station can simply send a random 128-bit challenge and when the response is returned, can simply reply that it's been authenticated without knowing the 128-bit Ki. It can then set the stream cipher to A5/0 and capture the unencrypted data.

A5/1.

Later sections require an understanding of A5/1 stream cipher in order to be meaningful.



The A5/1 Stream Cipher consists of 3 linear feedback shift registers of varying length. To generate a bit stream, the three MSBs of each register is XORed together, and then the cipher is clocked to get the next one. Clocking is done by shifting a register to the left and filling the LSB with XORed bit of various taps. Whether or not an individual shift register is clocked is determined by it's clocking bits, colored in orange above. A majority bit is calculated using the three clocking bits and if a clocking bit matches the majority bit, the individual register is clocked. At least two registers are clocked every cycle. To initialize the stream cipher, the shifts registers are zeroed and then clocked 64 times without taking into account the majority bits, while XORing the 64 bits of the session key into the LSB every cycle. Then the 22 bit frame is mixed in the same way over 22 cycles. Finally the entire cipher is clocked 100 times while taking the majority bits into consideration. The cipher is now ready and the bit stream is generated by clocking the cipher 228 times while considering the majority bits.

Attacking A5/1.

While various attacks on GSM have been published, none of them are practical, either requiring at least thousands of known plaintext and/or reducing the effective key size search space only by a small number of bits. While GSM does transmit some known plaintexts, a practical attack against A5/1 would require not knowing any unknown known plaintexts and be able to recover the key in a reasonable amount of time. Both of these criteria can be met by using a variation of brute-forcing.

Brute-forcing A5/1.

Designed for low cost complexity decades ago, the A5/1 cipher is fairly quick to calculate. Even taking that into account, brute-forcing the entire 2^64 keyspace using a single modern CPU would take on the order of a 1000 years. But, consumer graphics cards which consists of hundreds of large bit size processors, can be used to significantly reduce this time, down to months. While months of brute-forcing make breaking GSM feasible, the criteria for a practical attack have not be met. This can be done through a variety of modifications to simple brute-forcing.

Code Book Attack

The first modification to simple Brute-forcing to make GSM practical, is the realization, that by saving the results after calculating them, brute-forcing may only have to be done once and then saved. Cipher texts that can be looked up to find their corresponding key. While this seems ideal, It suffers from a number of pitfalls, the first one being that a GSM codebook would require 10^11 GB of space, an obviously unpractical amount of space.

Time-Memory Tradeoff.

Between the high computation, zero memory requirements of brute-forcing and the zero computation, high memory requirements of code books there exists an alternative that can be used to make attacking GSM practical. Chaining cryptographic calculations together by using it's output as an input and saving only the beginning and end into a dictionary table allows for reductions in memory requirements in exchange for computation.

$$\mathbf{aaaaaa} \underset{H}{\longrightarrow} 281\mathrm{DAF40} \underset{R}{\longrightarrow} \mathrm{sgfnyd} \underset{H}{\longrightarrow} 920\mathrm{ECF10} \underset{R}{\longrightarrow} \mathbf{kiebgt}$$

The amount of space used is proportional to the size of the chains, with increasing chain sizes decreasing memory used but also increasing computations proportionally. By choosing the chain size correctly, a balance between computation requirements and memory can be reached.

Distinguished Points.

A factor that can negatively affect the performance of dictionary tables is the assumption that memory access is instant. This is not reflective of contemporary hardware where RAM access is magnitudes longer than a CPU clock cycle. When doing a search for a key in simple dictionary table, a lookup is required after every chain step in order to determine if the current output is in the table, a massive bottleneck on contemporary hardware. This can be be fixed by ending chains when the output meets a certain criteria such as the number of trailing zeroes instead of fixed chains lengths. By using these “distinguished points”, tables lookup only needs to be done when the current search chain reaches a distinguished points.

Backclocking.

Because the A5/1 cipher is initialized using both a 64-bit key and 22-bit frame ID, it would seem that an implementation would need to treat the A5/1 cipher as having an 86-bit key, massively increasing required table size or computation required, making attacking A5/1 impractical in practice. This doesn't actually turn out to be true because of the ability to backclock the A5/1 cipher. Instead of the LSB being determined using the XOR of the taps, the MSB is determined by using the taps XORed with the LSB. This is slightly complicated in A5/1 by the irregular clocking determined by the majority bits, which causes multiple preceding states to possibly lead to the same end state. In practice, the first 64 bits outputted out of an initialized A5/1 cipher is used as the 64 bit internal state of the next A5/1 cipher and when a output stream match is found in the table, the state that generated it is found, the 22-bit frame of the output stream is backclocked out of it and the resulting state is the cipher initialized with just the 64-bit key. This state can then be clocked with any arbitrary frame ID to decrypt the entire GSM conversation.

Conclusion.

By combining everything discussed above, a practical GSM attack implementation is possible.