Security in the GSM network

Marcin Olawski

olawskim@gmail.com

Abstract

The GSM network is the biggest IT network on the Earth. Most of their users are connected to this network 24h a day but not many knows anything abut GSM security, how it works and how good it is. Most people blindly trust GSM security and send by the network not only theirs very private conversations and text messages but also their current location. This paper will describe how that information is guarded in 2G networks and how much of it an attacker can access without our permission or knowledge.

Introduction

As of June 30, 2010, 1.967 billion[1] people use the Internet, according to Internet World Stats. Most of them heard at least once some terms related to internet security, like anti-virus, worm, firewall, spyware and so forth. In comparison, in July 2010 GSM Association announced that the number of global mobile connections has surpassed the 5 billion mark. Because of multiple SIM ownership there is always a lag between connections and subscribers, but it does not change the fact that far more people use GSM then the Internet. At the same time, how many of those subscribers know anything about GSM security? Have you ever hear terms like *Ki* number, temporary mobile subscriber identity or the A5 algorithm? For example, any IT specialist knows that it is easy to forge an e-mail, but not everyone is aware that spoofing someone's phone number is even easier. Any IT expert knows that even in correctly protected WLAN when an IP package reaches the Internet it will be travelling unencrypted. You will be surprised but it is the same in GSM networks. Connections are encrypted between mobile phone and a Base Transceiver Station (BTS), but for a further distance they travel unencrypted. You can ask if this changes anything for you? Can anybody with a walkie-talkie eavesdrop on your calls? Of course not, the point is, that GSM should be treated as any other wireless network. Users should know its strong and weak points, and how an attacker can use these against them.

This article is organized as follows: In the 3 next sections a short description of the GSM security model will be given. In further sections various active attacks on GSM protocols will be presented, and in the last 2 sections several passive attacks on GSM encryption will be described.

The principles of GSM security

When dealing with the principles of GSM security it is important to keep in mind the time when the system was designed. Work on pan-European mobile communication technology started in 1982. The basic parameters of the GSM standard were agreed in 1987 and more detailed specifications were completed in 1988. In the year 1989 the European Telecommunications Standards Institute defined the GSM standard as the internationally accepted digital cellular telephony standard. Because of the political situation in those days, designers decided against using

1 In this document the short scale is used, billion is 10^9

strong cryptography algorithms in GSM networks. In 1994 Ross J. Anderson, well know researcher, writer, and industry consultant in security engineering described the situation:

> *"My spies inform me that there was a terrific row between the NATO signals agencies in the mid 1980's over whether GSM encryption should be strong or not. The Germans said it should be, as they shared a long border with the Evil Empire; but the other countries didn't feel this way. And the algorithm as now fielded is a* [weak] *French design."*

In addition, planners decided that encryption would be applied on a radio patch only and there was no attempt to provide security on the fixed network part of the GSM. Also, "active attacks", like man-in-the-middle attacks, were not considered as a real thread. Constructing a BTS in those times was simply too expensive to make those attacks cost-effective and therefore there is no BTS-to-MS (Mobile Station) authentication. Conversely, designers focus on MS to BTS authorization to prevent phone cloning or call spoofing. Additionally, the MS to BTS authorization gives the system a convenient way to share encryption keys between the MS and the BTS and makes unauthorized tracking of a specific phone harder. It was also decided that customers should be able to easily change their phones but, at the same time, should not know their authentication keys. This problem was solved by SIM cards. The SIM card could be considered as an easy way to transfer customer identity from one mobile phone to another. To summarise, the most important principles of GSM security are: subscriber identity authentication and the ciphering of radio transmissions.

Subscriber authentication

In GSM networks the subscriber is identified by their SIM card, therefore the SIM card contains all of the details necessary to obtain access to a particular account. The two most important pieces of information hosted by the SIM are:

- The International Mobile Subscriber Identity (*IMSI*) – a unique number for every subscriber in the world. The number can be read from the SIM easily.
- The *Ki* number, the root encryption key. This is a randomly generated 128-bit number allocated to a particular subscriber. The *Ki* is highly protected, and is only known to the SIM card and the network's Authentication Centre. The phone itself never learns of the *Ki*. Every operations that requires knowledge of the *Ki* is performed inside the SIM, which is possible because the SIM card is an intelligent device with a microprocessor, storage and even an operating system.

The MS to Public Land Mobile Network (PLMN) authorization is performed by a challenge-response mechanism, as follows:

1. The phone connects to the network.
2. The phone submits its identity by sending a special identification number, called the *TMSI* (see below).
3. The network generates a 128-bit random number know as the *RAND* and sends it to the MS. The MS encrypts the *RAND* using the *Ki* and the authentication algorithm A3 implemented within the SIM:

$$A3_{Ki}(RAND) = SRES$$

4. The computed 32-bit value *SRES* (signed response) is sent back to the network.
5. The network performs the same operation to get an *XRES* (expected response) and compare it to *SRES*:

$$\mathrm{A3}_{Ki}(RAND) = XRES$$
$$SRES = XRES\ ?$$

6. If both values are equal the phone proved knowledge of the *Ki* and is thus authenticated.
7. The MS and the network compute a session cipher key *Kc* using A8 algorithm:

$$Kc = \mathrm{A8}_{Ki}(RAND)$$

8. The network chooses the encryption algorithm according to a list of supported algorithms reported by a cell phone and the encryption is activated. The *Kc* is used as the session cipher key.

Because encryption is activated after authorization to prevent an unauthorized user tracking the network avoids using the *IMSI*, instead it uses the Temporary Mobile Subscriber Identity (*TMSI*). The *TMSI* is generated when the user for the first time connects to the new network as follows:

1. A MS use its *IMSI* to identify itself to the network.
2. The network activates encryption and sends a new *TMSI* to the MS.

To provide even more security, the *TMSI* is frequently changed.

Most mobile operators implement the A3 and the A8 in fact as one algorithm, namely COMP128 (version 1, 2 or 3). The first version of the algorithm was leaked and is now publicly known, the designs of versions 2 and 3 are still kept secret.

The COMP128-1 is a hash function. It takes a 16 byte key *Ki*, and 16 byte *RAND*, to output a 12 byte hash as follows: ("||" means concatenation of byte arrays):

$$\mathrm{COMP128}(\ Ki||RAND\ ) = hash$$
$$SRES = hash[0...3]$$
$$Kc = hash[4...11]$$

More precisely the algorithm first loads *Ki* and *RAND* into a 32 byte vector *X*. *Ki* is stored in *X*[0..15] and *RAND* is stored in *X*[16..31]. Then, compression on *X* is applied. The compression consists of lookups to 5 tables T_0, T_1, T_2, T_3, T_4 and a butterfly-structure B_{A3}. Each T_j contains only (8-*j*)-bit values, thus, compression results in 32 4-bit values, that are further permuted by permutation P_{A3} and copied into *X*[16..31]. Then *Ki* is loaded into *X*[0..15] and a new iteration begins. The resulting 128 bits after the eight iterations are further compressed to 12 bytes by permutation/selection function F_{A3}, which forms the output of the algorithm. The pseudo-code of the compression in the COMP128-1 is listed below:

```
for j=0 to 4 do{
    for k=0 to $2^j-1$ do{
         for l=0 to $2^{(4-j)}-1$ do{
              m = $l + k\cdot 2^{(5-j)}$;
              n = $m + 2^{(4-j)}$;
              y = $(X[m] + 2\cdot X[n]) \bmod 2^{(9-j)}$;
              z = $(2\cdot X[m] + X[n]) \bmod 2^{(9-j)}$;
              X[m] = $T_j[y]$;
              X[n] = $T_j[z]$;
         }
    }
}
```

This complicated loop can by easy depict in human-readable form:

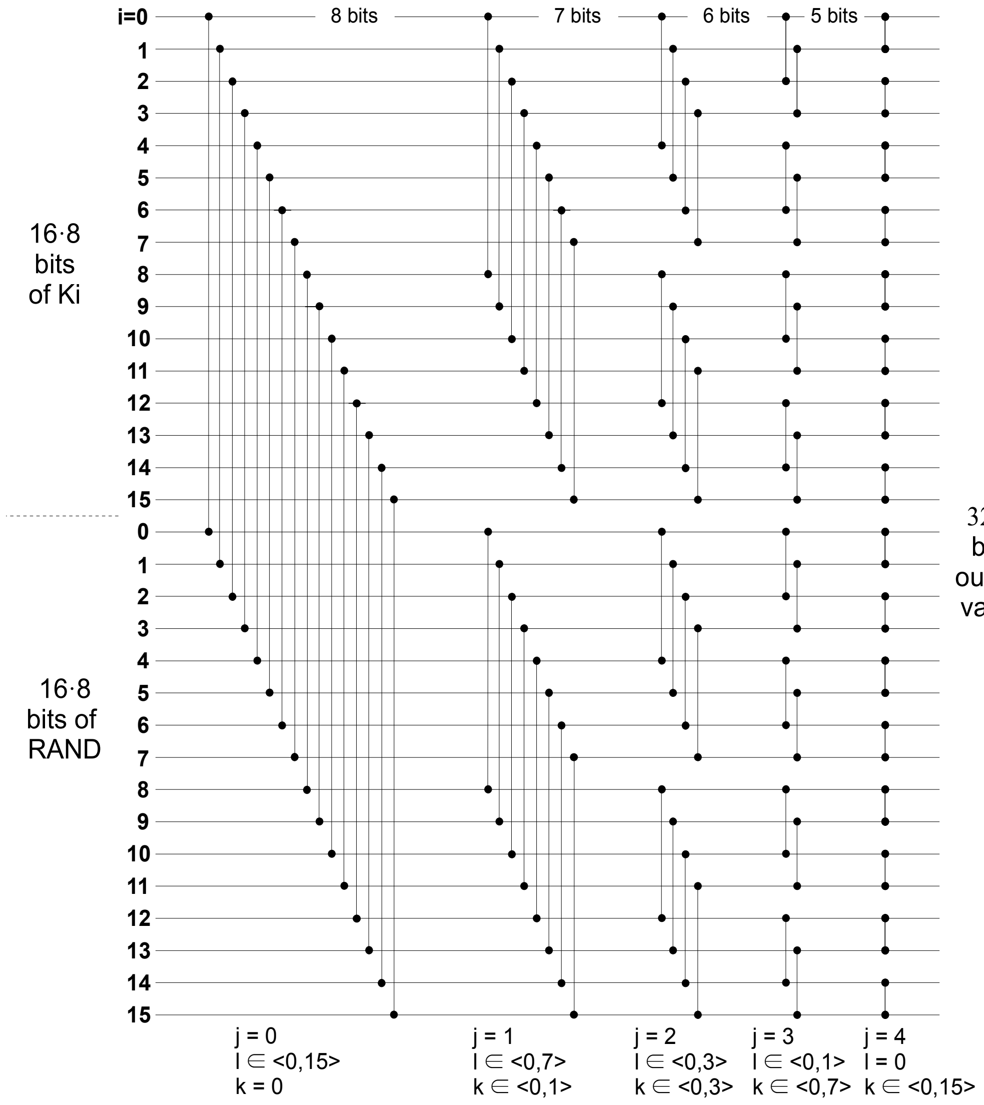

Fig 1. The butterfly-structure in the COMP128-1 (source 4)

In each layer there are 16 combining operations, each taking a pair of inputs to a pair of outputs:

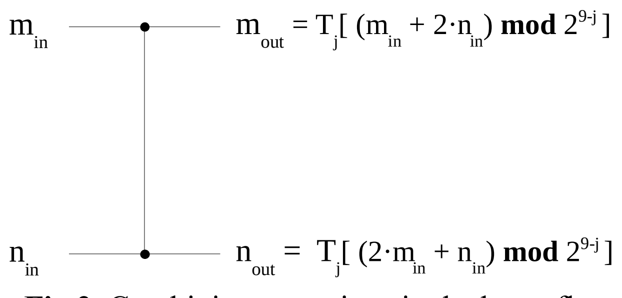
m_{in}
$m_{out} = T_j[\ (m_{in} + 2\cdot n_{in})\ \mathbf{mod}\ 2^{9-j}\ ]$
n_{in}
$n_{out} = T_j[\ (2\cdot m_{in} + n_{in})\ \mathbf{mod}\ 2^{9-j}\ ]$

Fig 2. Combining operations in the butterfly-structure

Description of P_{A3} and F_{A3} will be omitted and the reader could find a detailed description in [3].

Traffic encryption

The traffic encryption is activated after a MS authorized itself to the network. Encryption of the whole radio link is performed by the MS because it demands substantial computation power and cannot be perform by the 8-bit SIM card. Because the MS does not know the *Ki* it cannot be used as an encryption key. Instead of this, the MS and the network negotiate a new encryption key *Kc* as described in the previous section.

A GSM transmission is organized as sequences of bursts. One burst contains 114 bits and is sent every 120/26 ms. The ciphering works by generating a stream of bits, (a cipher block), which is XOR-ed with the plaintext, to produce the ciphered text. The data is decrypted on the other end by XOR-ing the received data with the identical cipher block (see Figure 3).

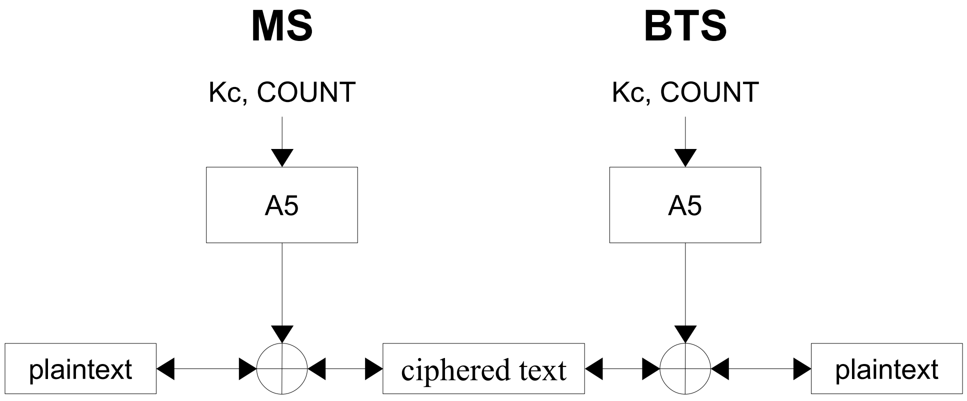

Fig 3. Data encryption and decryption in the GSM

This method has one serious drawback. Two of the same plaintexts give the same ciphertexts. This seemingly irrelevant feature is widely explored in a cryptanalysis, for example, the Enigma was cracked, among other things, because of this feature. To prevent this, the algorithm is also "seeded" by *COUNT*, the number of which is based on the TDMA frame number:

$$T1 = FN / 1326$$
$$T2 = FN \bmod 26$$
$$T3 = FN \bmod 51$$
$$COUNT = T1 || T3 || T2$$
$$\mathrm{A5}(Ki || COUNT) = cipher\_block$$

Where *FN* is a TDMA frame number. In GSM the Time Division Multiple Access is a method that allows serving the same physical channel on up to eight different phones. It is done by allocating the physical channel to different phones through a round-robin, where each phone transmits in a time slot that lasts 15/26 ms.

Currently there are 4 stream ciphers defined: A5/0, A5/1, A5/2 and A5/3. The A5/0 is the dummy cipher definition that the communication is unprotected. The A5/1 is a real cipher that was introduced in 1989, and due to encryption export laws could be used only by members of European Conference of Postal and Telecommunications Administration (CEPT) and the Coordinating Committee for Multilateral Export Controls (CoCom). Four years later, when it was apparent that GSM was going to be used globally and not just in Europe, a second, weaker algorithm, called A5/2, was developed for other countries. Later, towards the end of the 90s, because the global political situation changed, (CoCom ceased to function, the Russian Federation became a member of the CEPT), the need for the A5/2 disappeared and general use of the A5/1 was encouraged. The last algorithm, called KASUMI, was in 2002 added to the A5 family under the name of A5/3. It was designed by the Security Algorithms Group of Experts, part of the European standards body the European Telecommunications Standards Institute. KASUMI was not built from scratch, but it is a slightly optimized version of the MISTY1, an algorithm designed in 1995 by Mitsubishi Electric.

The A5/1 and the A5/2 are both stream ciphers that use Linear Feedback Shift Registers (LFSR). A shift register is a hardware register that can shift all its bits concurrently by one position and fill the "empty" bit with a given value delivered by some linear feedback functions (most often XOR). The positions of the bits that "participate" in the feedback function are referred to as a tap sequence. A5 stream ciphers are made by combining the output of several LFSR. The A5/1 uses three LFSRs with lengths of 19, 22, and 23 bits (64 in total). Figure 4 shows tap sequences and other details of the A5/1.

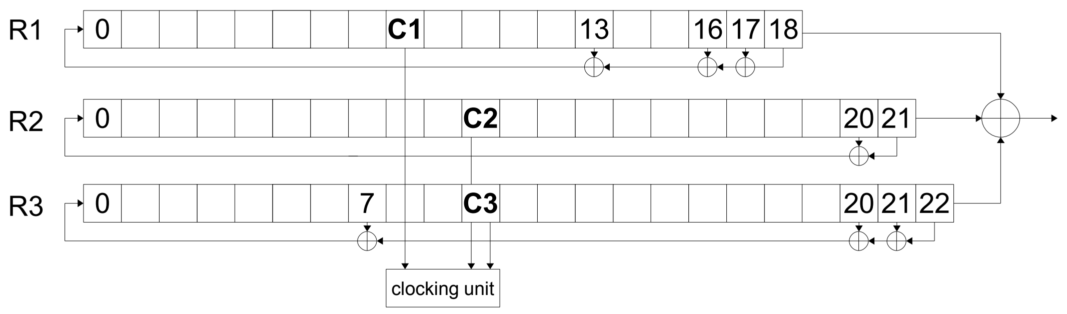

Fig 4. The internal structure of the A5/1

Registers $R1$, $R2$ and $R3$ are clocked in a stop/go fashion using the following rule: each register has a single clocking tap ($C1$, $C2$, $C3$); on each clock cycle, the majority function of the clocking taps is calculated and only those registers whose clocking taps agree with the majority bit are clocked. Before every use the A5/1 is initiated as follows:

```
1. $R1 = R2 = R3 = 0$
2. For $i = 0$ to 63 do
        $R1[0] = R1[0] \oplus Kc[i]$
        $R2[0] = R2[0] \oplus Kc[i]$
        $R3[0] = R3[0] \oplus Kc[i]$
        Clock all three registers ignoring the stop/go clocking control
3. For $i = 0$ to 21 do
        $R1[0] = R1[0] \oplus COUNT[i]$
        $R2[0] = R2[0] \oplus COUNT[i]$
        $R3[0] = R3[0] \oplus COUNT[i]$
        Clock all three registers ignoring the stop/go clocking control
4. For $i = 0$ to 99 do
        Clock the cipher by its regular clocking control, and discard the output
```

After initialization, 228 bits of output stream are being computed. 114 bits are used to encrypt data from the network to the mobile phone, and the other 114 bits are used to encrypt data from the phone to the network.

The A5/2 is very similar to its "older brother". It also uses three 19, 22, and 23 bits registers to produce output, but its stop/go mechanism is different. Instead of clocking taps it has an additional register $R4$ which is use only to control clocking of $R1$, $R2$ and $R3$. Figure 5 shows construction of the A5/2.

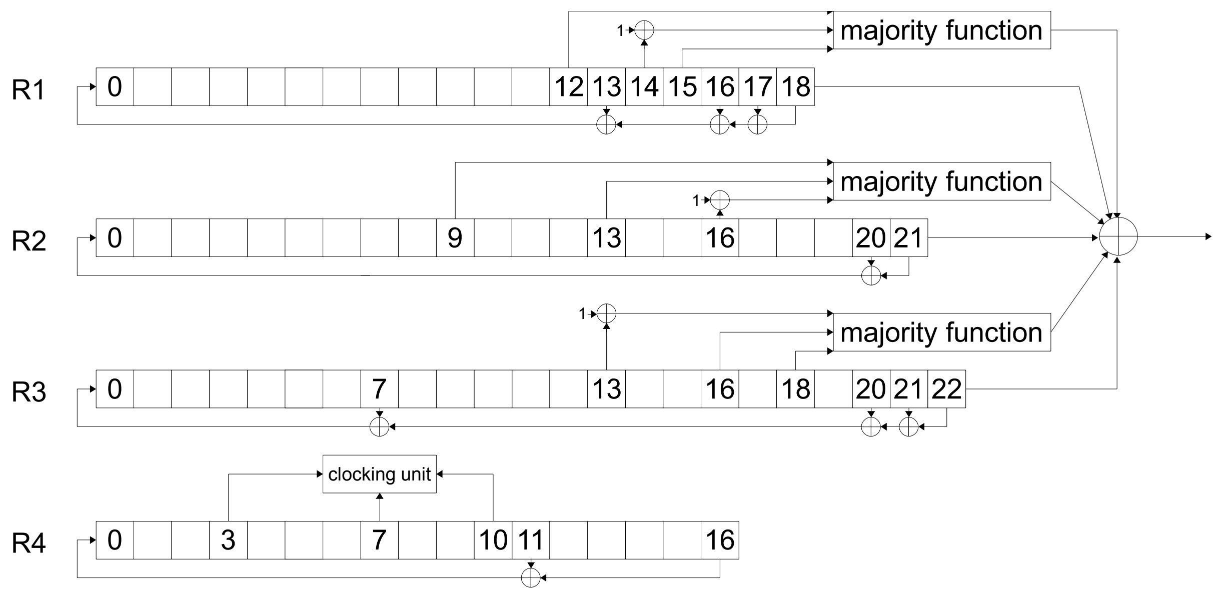

Fig 5. The internal structure of the A5/2

The clocking control of the A5/2 can be described as follows:

```
$m$ = Majority( $R4[3]$, $R4[7]$, $R4[10]$ )
if $R4[10] = m$ then clock the $R1$
if $R4[3] = m$ then clock the $R2$
if $R4[7] = m$ then clock the $R3$
```

The initialization of the A5/2 is similar to the initialization of the A5/1 and looks like this:

```
1. $R1 = R2 = R3 = R4 = 0$
2. For $i = 0$ to 63 do
       $R1[0] = R1[0] \oplus Kc[i]$
       $R2[0] = R2[0] \oplus Kc[i]$
       $R3[0] = R3[0] \oplus Kc[i]$
       $R4[0] = R4[0] \oplus Kc[i]$
       Clock all three registers ignoring the stop/go clocking control
3. For $i = 0$ to 21 do
       $R1[0] = R1[0] \oplus COUNT[i]$
       $R2[0] = R2[0] \oplus COUNT[i]$
       $R3[0] = R3[0] \oplus COUNT[i]$
       $R4[0] = R4[0] \oplus COUNT[i]$
       Clock all three registers ignoring the stop/go clocking control
4. $R1[15] = R2[16] = R3[18] = R4[10] = 1$
5. For $i = 0$ to 98 do
       Clock the cipher by its regular clocking control, and discard the output
```

The A5/1 and the A5/2 are similar to saved hardware in phones, on the contrary A5/3 is totally different construction. It has a block size of 64 bits and a key size of 128 bits. It is a Feistel cipher with eight rounds. Because the KASUMI is still seldom used in 2G GSM networks it will not be described in this paper. Description of the KASUMI could be found in [5].

Apart from cryptography, frequency hopping is used as another layer of security. In GSM the transceiver changes a physical carrier every frame (120/26 ms). The hopping sequence is defined by two parameters – the Mobile Allocation Index Offset (MAIO), which takes a value from zero to the number of frequencies in the list minus one, and the Hopping Sequence Number (HSN), which takes a value from 0 to 63. There are 2 modes of hopping - cycling hopping and non cyclic hopping. If the HSN is 0, cyclic hopping is used where the mobile station simply steps through a set of frequencies (Figure 6).

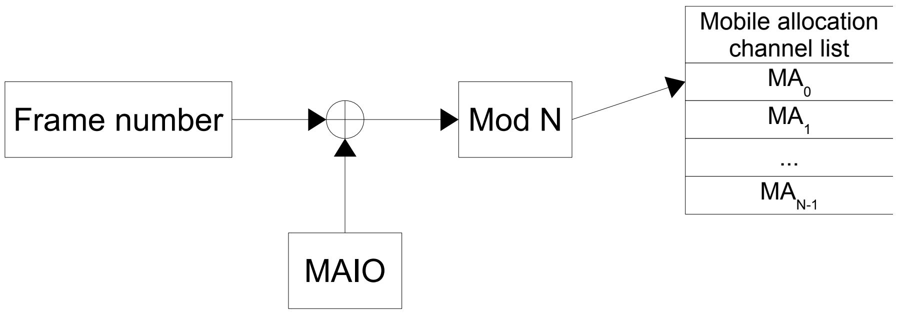

Fig 6. Cycling frequency hopping in the GSM

In non-cyclic hopping, the frame number and the HSN is used to seed a more complex hopping algorithm (Figure 7).

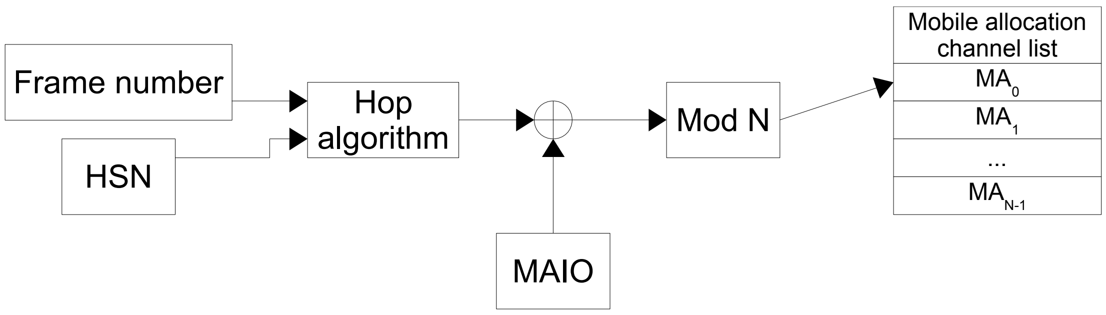

Fig 7. Non-cyclic frequency hopping in the GSM

Usually, traffic channels in the same cell bear the same HSN and different MAIOs. After a traffic channel is assigned, the mobile and the network compute the frequency for each burst according to the above information given at the time of assignment, and according to the TDMA frame number.

GSM Weakness

As was mentioned before the network does not authenticate itself to a phone. This is the most serious fault in GSM security, which allows a man-in-the-middle attack. This weakness was known for GSM constructors at the time of the GSM design, but it was expected that building a false BTS would be to expensive and it would be difficult to make those attacks cost effective. However, after 20 years the situation changed significantly. Today there are companies that product short range BTS, so an attacker can simply buy a BTS at a reasonable price. Also, there is a OpenBTS project. The OpenBTS is a project developed by Kestrel Signal Processing and the GNU Radio community aimed at constructing an open-source Unix application to present a GSM air interface. Apart from application one can find a description on how to build a BTS based on an inexpensive hardware on the OpenBTS web page. So, how much should everything cost? In 2008 the Kestrel team constructed a hybrid GSM-SIP base station to perform field tests. Radio and power gear for the base station costs about $4500 and its service radius of was roughly 2.4 km (~1.5 miles). The BTS was supposed to serve test users only but according to the Kestrel web site:

> *"On Saturday afternoon, we realized that unprovisioned users were making outgoing trunk calls. All they had to do was dial 1 at the beginning of the number to get routed to our VoIP carrier. We did some analysis and research and determined that this security hole was due to a combination of two bugs, one in our Asterisk configuration and another in our GSM control layer. (...) A post-test analysis of the CDRs and logs of the GSM stack showed that we successfully connected about 120 phone calls to 95 different numbers in area codes all over North America, most of them in the last 24 hours of the test. About half of these calls lasted over a minute and the longest was over 15 minutes."*

This story may look like a man-in-the-middle attack but to set a real man-in-the-middle an attacker has to solve one serious problem. The man-in-the-middle attack can only be effective when the attacker can impersonate each endpoint to the satisfaction of the other. In other words, if A is calling to B, B must see A as a caller. Because the attacker does not know his victim's *Ki*, he could not authenticate himself to the victim's network. He could use his own cell phone to call B but, as was mentioned before, B must see A's number on her/his cell phone. This problem could be sorted out in two ways. The easiest method is to spoof A's number. The disadvantage of this method is that the attack could be easily discovered by analysing A's Call Detail Record.

A more improved attack was proposed in 2003 by Elad Barkan, Eli Biham and Nathan Keller. They explored the fact that the network chooses the encryption algorithm according to list of supported algorithms reported by a cell phone in a message called class-mark. Because the class-mark message is unencrypted, the false BTS retransmits the victim's conversation to legitimate BTS with a fake class-mark message that contains only a A5/2 algorithm. The A5/0 would be better but any legitimate GSM network is expected to deny services to any MS that support only A5/0. The attacker can then listen in to the conversation through the cryptanalysis of the weaker A5/2 cipher. In addition, because there is no key separation, (the key-agreement protocol is independent of the encryption algorithm that is used), the attacker can use the A5/2 in the MS to falsify the BTS connection and the A5/1 in connection to victim's network.

Another serious vulnerability of the GSM is the lack of proper Caller ID or Sender ID verification. In other words, the caller number or SMS sender number could be spoofed.

Caller ID spoofing is not a specific GSM problem as it is also related to other types of telephony. The most popular ways of spoofing Caller ID are through the use of PRI lines or Voice over IP. The Primary Rate Interface is an access interface to the ISDN. It was designed for medium to large enterprises with digital PBXs to provide them access to the Public Switched Telephone Network. PRI lines consist of B and D channels. The B channels are primary data or voice communication channels while D channels are for control and signalling pieces of information. PRI lines are vulnerable to spoofing because the voice is sent through the B channel while the Caller ID is send through the D channel. In other words, Caller ID in the D channel is additional information to voice transmission in the B channel, and setting it to a bogus value does not disturb traffic in the voice channel. This fact is typically used by enterprises to display one main telephone number on all outgoing calls. Because of its high prices, PRI lines were out of reach of private people and spoofing was used mainly by businesses. The situation changed with the increase in popularity of VoIP technology. VoIP is easily vulnerable to spoofing because the voice is sent over an IP package and setting the Caller ID to a false value does not influence IP routing. In 2003, phone phreak

Lucky225 discovered a flaw with the VoIP provider Vonage that allowed users to set a fake Caller ID. His method was very simple. An attacker could apply a request to Vonage to port his phone number to their network, but instead of using his own number he could supply any valid telephone number. At the same time other phone phreaks figured out that Asterisk, an open source PBX software, allowed users to set their Caller ID freely within the application and then pass the spoofed Caller ID number to their VoIP provider. But the real spoofing boom began in 2004. In September 2004 an entrepreneur named Jason Jepson started up a Caller ID spoofing site, named Star38.com. Star38 gained attention from mainstream media around the world that initiated a crop of similar services. The following months brought sites like CallNotes.net, Camophone.com, CIDSpoof.com, CovertCall.com, PhoneGangster.com, SecretCalls.net, SpoofCard.com, SpoofTel.com, StayUnknown.com and others. Most of these web pages disappeared quickly and today the market is dominated by the SpoofCard. Despite the fact that the number of operators declined, the market is still growing and became a real threat not only for individuals but also for systems that use Caller ID for users authentication. For example, in 2006 SpoofCard announced that it had terminated the accounts of more than 50 customers, including Paris Hilton, for allegedly breaking into unauthorized voice mail boxes.

SMS Sender ID spoofing is similar to Caller ID Spoofing. Fraudsters need to find a SMSC that does not check the Sender ID during SMS sending and use it to an send SMS to a subscriber in another SMSC. The attack could be performed because many operators do not validate Sender ID when an SMS comes from different networks. For example, spoofing is possible with most European operators but it is not possible in the USA and Canada. The attacks became popular few years ago with the increase in popularity of online SMS gateways. Bulk SMS providers often allowed their clients to use anything as Sender ID provided that they pay bills regularly. A good example of such a provider was Clickatell, one of the world's largest bulk SMS providers. Any Clickatell customer could use a small application called SmsDumper to set his/her Sender ID to any phone number or short text. Nowadays however most bulk text messaging carriers feel the need to police their services and restrict their customers from changing Sender ID freely. For example, the aforementioned Clickatell verifies any new Sender ID before it can be used. Although respectable bulk SMS providers do not allow SMS spoofing there are many providers specialized in this field, like hoaxMail.co.uk or FakeMyText.com. Like Caller ID spoofing, Sender ID spoofing could be use not only to cheat individuals but also IT systems. In 2007 security researcher Nitesh Dhanjani described methods to impersonate users of Twitter.com by using FakeMyText.com. Dhanjani could perform his tricks because Twitter authorized users only by their phone number.

Another weakness of GSM security consists of encryption. Apart from the encryption algorithms weakness that will be described in the next two sections, data in GSM networks is encrypted only in the air interface. At the time of GSM design, it was expected that the ground parts of GSM networks would be using fixed, safe lines and therefore encryption would not be required. Today's situation changed. Operators use point to point microwave links, leased line, and sometimes even the Internet to interconnect various pieces of their networks. Another drawback of this situation is that operators' employees could have access to subscribers' data. From time to time the media covers stories about operators' workers who are eavesdropping on client's calls. For example, in 2005 Vodafone Greece discovered that someone with access to the company switches had been bugging more than 100 high-ranking government officials. The media suspected one of the Vodafone engineers who was found dead in his apartment shortly after the incident was discovered. Targets chosen suggest that some foreign intelligence agency was involved in this case, but there is no need to be a James Bond to read someone else's text messages. In 2002 a 21-year-old student from the UK who suspected his girlfriend of infidelity persuaded two friends, employees at O_2, to intercept her text messages and pass them on to him.

Apart from stories about dishonest workers, it is an open secret that eavesdropping on subscribers' conversations is sometimes practised during the handling of consumer complaints. For example, when a subscriber report crackles on the line, employees sometimes simply listen to a few minutes of the customer's conversation to check the legitimacy of the complaints. That case brings to one's mind a question of more serious threats to consider in terms of access to supposedly inaccessible information.

Another weakness attackers can exploit is vulnerability in the *IMSI* protection mechanism. As mentioned before, networks use *TMSI* to protect *IMSI* but if the network somehow loses track of a particular *TMSI* it must then ask the subscriber their *IMSI* over a radio link. The connection cannot be ciphered because the network does not know the identity of the user, and thus the *IMSI* is sent in plain text. The attacker can thus check whether a particular user (*IMSI*) is in the vicinity. He accomplishes this by imitating a legitimate BTS, and paging that subscriber by their *IMSI*. The subscriber's phone will then establish a radio connection, and the attacker can send the subscriber the identity request message, and the phone will respond with the *IMSI*.

Finally, to be able to intercept GSM communication an attacker must defeat frequency hopping. In order to do this he must learn the hopping sequence, as without it he would have to sample the entire bandwidth (tens of megahertz). The main problem with the hopping sequence is that the hopping sequence parameters on a particular BTS are usually fairly static and the typical hopping parameters of that BTS can be learned quickly. Thus, hopping does not give more security; it simply adds just another layer of complexity for the attacker.

Weakness of the A3/A8 algorithms

The A3 and the A8 relied on security through obscurity model but algorithms specifications were partially leaked and missing pieces were filled in by reverse-engineering of a working SIM card by Marc Briceno, Ian Goldberg, and David Wagner in 1998. It turned out that both the A3 and the A8 were in fact one algorithm, namely COMP128-1. Another huge surprise was that even though *Ki* is 64-bit number, its 10 last bits are always equal to zero. After reconstruction of the COMP128-1 Briceno, Goldberg and Wagner analysed it and proposed an attack on the algorithm that allows an attacker to extract a secret key (*Ki*) from a SIM card. It was a very serious bridge in GSM security. Because all GSM traffic is encrypted only by *Ki* and unencrypted *RAND,* if an attacker extracts the *Ki*, he can eavesdrop on all SIM owner phone conversations or SMS messages. Moreover, to perform the attack the attacker does not need physical access to the SIM card. As a result of this, the GSM Association Security Group issued a warning to operators that COMP123-1 was an example only, and they should have developed their own algorithm.

To extract *Ki* the attack used the birthday paradox. In probability theory, the birthday paradox pertains to the probability that in a set of randomly chosen people some pair of them will have the same birthday. The question is: how many people are needed before the probability exceeds 50%? Many people think of 366/2 = 183 or something along that line, but the correct answer is: only 23. It is because in a group of 23 people, there are ½·22·23 = 253 pairs, so that the probability for said duplicity is much higher (50.7%). In cryptography the mathematics behind the birthday paradox is used to assess the probability that for given hash function f we will find pair $x_1 \neq x_2$ such that $f(x_1) = f(x_2)$. Such a pair is called a collision.

Briceno, Goldberg and Wagner discovered that collisions in the COMP128-1 can be used to extract *Ki* from SIM. As mentioned before, COMP128-1 uses the "butterfly" network B_{A3}. The attack relies on the fact that collisions can be made to occur on the second layer of the first round:

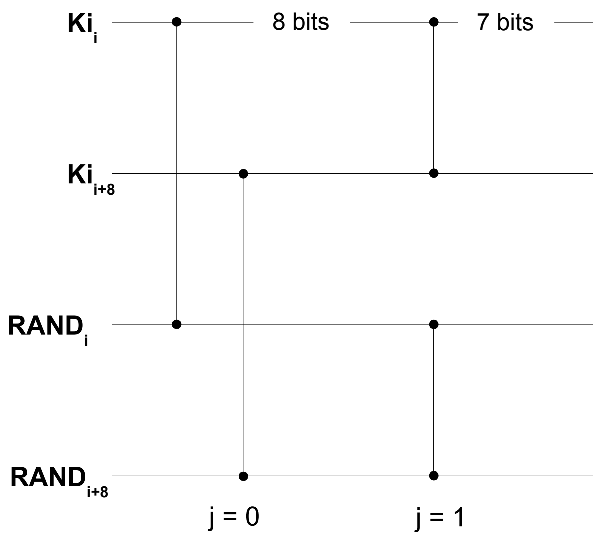

Fig 8. Collisions in the COMP128-1

As the diagram indicates, *Ki* can be attacked piecemeal, since for a given *i* at the second layer (*j*=1) the $4 \cdot 7 = 28$ bits of output only depend on the pair of bytes (Ki_i , Ki_{i+8}) from the key and the pair ($RAND_i$, $RAND_{i+8}$) from the challenge. The attacker iterates through the values of ($RAND_i$, $RAND_{i+8}$) while keeping the other bytes of the *RAND* challenge the same until a collision happens in the 96 bit output of the COMP128-1 function. Because 28 bits of output is small in comparison to the 2^{16} inputs and because of the aforementioned birthday paradox this is overwhelmingly likely to happen. Having such a collision, the attacker now has two pairs ($RAND^1_i$, $RAND^1_{i+8}$), ($RAND^2_i$, $RAND^2_{i+8}$) which produce the same output, almost certainly because of a collision at second layer. The attacker can now iterate through the values of (Ki_i , Ki_{i+8}) in his own computer, searching for the key bytes which would also produce the collision for those same values of ($RAND^1_i$, $RAND^1_{i+8}$), ($RAND^2_i$, $RAND^2_{i+8}$). By repeating this process 8 times all 16 bytes of *Ki* can be extracted. The expected number of challenges before this attack succeeds is around 150,000. But the number of queries could be reduced. Once the second layer has been attacked the attack could be mounted on further layers, and whole process may be reduced to around 20,000 queries.

The SIM card can be challenged in two ways. If attacker has physical access to the SIM he can simply use a smart card reader. At 6.25 challenges a second, this takes about 1 hour. This time can be shortened further, for instance by over-clocking a reader. The second method is the over-the-air attack. In this scenario the attacker is imitating a legitimate GSM network and uses an authentication procedure many times to extract the *Ki*. In perfect conditions he would need 235 ms to send an authentication request and 235 ms to receive the response (at the same time he could send the next request), thus the expected time to recover the *Ki* would be around 85 minutes.

In 2002 Josyula R. Rao, Pankaj Rohatgi, Helmut Scherzer and Stephane Tinguely proposed a side-channel attack on COMP128-1. In cryptography, a side-channel attack is any attack based on information gained from the physical implementation of a cryptosystem, rather than brute force or theoretical weaknesses in the algorithms. The aforementioned researchers identified that instantaneous power consumption and electromagnetic emission is correlated with the challenge in several implementations of COMP128-1. The anomaly is probably caused by the fact that COMP128-1 requires some 9-bit operations, while SIM cards are 8-bits chips. As can be observed from the specification of COMP128-1, when *j*=0:

$$X[m] = T_0[y]\ ,\ y = (X[m] + 2 \cdot X[n])\ \boldsymbol{mod\ 2^9}$$
$$X[n] = T_0[z]\ ,\ z = (2 \cdot X[m] + X[n])\ \boldsymbol{mod\ 2^9}$$

As it can be seen the T_0 table must have a 9-bit index. Since it is not possible to directly address such a table on an 8-bit addressing architecture, it is highly likely that programmers split T_0 into two tables, T_{00} and T_{01}, of size 256 each. Furthermore, the easiest way to split it is to store the first 256 elements of T_0 in T_{00} and the last 256 elements in T_{01}. This hypothesis explains the observed anomaly. Looking up a random element of T_{00} will result in a somewhat different power signal than looking up a random element of T_{01}. Moreover, there were 32 regions of a SIM card where these anomalies were observed. The researchers came to the conclusion that this is because of the fact that the first level of compression in COMP128-1 requires two table lookups into T_0, with indices y and z, for each of the 16 bytes of input. These observations allow users to mount a very efficient attack against COMP128-1. Notice that in the first layer (j=0):

$$m = i$$
$$n = m+16$$
$$X[m] = Ki[i]$$
$$X[n] = RAND[i]$$
$$y = (Ki[i] + 2 \cdot RAND[i]) \bmod 512$$
$$z = (2 \cdot Ki[i] + RAND[i]) \bmod 512$$

The following example will be used to explain how to compute *Ki*. Let's assume that an attacker has a SIM card with unknown *Ki*. He observed that all table lookups of $T_0[y]$ fell in T_{01} with *RAND*[0] in the range [27,...,154]. The transition when RAND[0] goes from 26 to 27 has to be caused by the value of y crossing 256. Similarly the transition when *RAND*[0] goes from 154 to 155 must be caused by the value of y crossing 512. From this it follows that the *Ki*[0] can only be 202 or 203. Next, the same classification is performed with z. The attacker observed that all the table lookups of $T_0[z]$ fell in T_{00} with *RAND*[0] in the range [0...105]. Since *Ki*[0] is either 202 or 203, the transition when *RAND*[0] goes from 105 to 106 only occurs for *Ki*[0]=203 and hence the attacker obtains the first byte of the key. To extract the whole key the attacker performs similar analysis on the remaining values of i.

To defeat the known weaknesses of the COMP128-1 two newer versions of the COMP128 were designed, namely COMP128-2 and COMP128-3. The specifications of the COMP128-2 and the COMP128-3 have not yet leaked so the SIM cards that use them are still safe.

Known attacks on A5/1 and A5/2 algorithms

The A5/1 and the A5/2 algorithms relied on the security through obscurity model. Therefore, independent researchers could not asses both algorithms and there was uncertainty about their strength. The situation changed after the reverse engineering of both in 1998. When cryptologists began to analys both algorithms it became obvious that both do not provide an adequate level of security. Furthermore, facts such as that the 10 last bits of *Ki* are always equal zero suggest that this is not by chance.

Most of the attacks, which will be described further, are known-plaintext attacks, i.e. to decode the encryption key an attacker must know not only the encrypted frames, but also their plaintext content. This assumption may look unrealistic, but researchers found methodology on how to extract some amount of plaintext from encrypted GSM frames. For example, every traffic channel between the handset and the network is accompanied by a slower control channel, namely Slow Associated Control Channel (SACCH). The network uses this channel to send some system messages to the mobile, as well as to control the power and timing of the current conversation. The content of the SACCH can be somehow predicted by observing the victim's phone behaviour. Additionally, some contents of the SACCH are cyclically transmitted. These messages can be obtained, for example, from eavesdropping on the beginning of a call when transmission is not encrypted, or from the attacker's phone.

A5/2 was analysed immediately after reverse engineering by using linear cryptanalysis.
The linear cryptanalysis was developed by Mitsuru Matsui in 1993 and today it is one of the most modern cryptanalytic methods. It consists on constructing systems of linear equations that approximate the action of a cipher and use these linear equations in conjunction with known plaintext-ciphertext pairs to compute the key.
In classical algebra, a linear expression in variables, $x_1, \dots , x_n$, has the form:

$$y = a_1 x_1 + a_2 x_2 + ... + a_n x_n$$

where a_i are real number constants. On the contrary, in the linear cryptanalysis all variables are binary variables and all classical algebraic operation are replaced by an appropriate binary operation, namely, addition is replaced by the XOR operation and a multiplication is represented as an AND operation.

binary addition, XOR	**binary multiplication, AND**
$0 \oplus 0 = 0$ $1 \oplus 0 = 0 \oplus 1 = 1$ $1 \oplus 1 = 0$	$0 \cdot 0 = 1 \cdot 0 = 0 \cdot 1 = 0$ $1 \cdot 1 = 1$

Table 1. Binary replacements for algebraic operations

The usual arithmetic rules for real numbers apply here, too, though they may seem a bit odd.

In 2003 Ian Goldberg, David Wagner and Lucky Green used the aforementioned technique to mount an attack on the A5/2. The attack is based on observation that $R4[10]$ is set to 1 after every key setup, hence $R4$ has the same value after initialization regardless of whether the bit $COUNT[10]$ is 0 or 1. Let f_i and $f_{i+\alpha}$ be the respective $COUNT$ values for frames i and $i+\alpha$. If f_i different from $f_{i+\alpha}$ only in $f[10]$ ($f_i \oplus f_{i+\alpha} = 0000000000010000000000_b$) then $R4_i = R4_{i+\alpha}$. Because of chosen permutation between the TDMA frame number and f, it happens for any two frames which are exactly 1326 TDMA frames apart ($\alpha = 1326$). Let Z_i and $Z_{i+\alpha}$ be the keystream values for the aforementioned frames, Goldberg, Wagner and Green observed that $Z_i \oplus Z_{i+\alpha}$ is linear in $R1_i$, $R2_i$, and $R3_i$. Therefore, for given $Z_i \oplus Z_{i+\alpha}$, the initial internal state of $R1_i$, $R2_i$, and $R3_i$ can be recovered by solving a linear systems of equations. Because the initial internal state of $R1_i$, $R2_i$, and $R3_i$ is 61 bits (three bits of $R1$, $R2$, and $R3$ are set to 1) only 61 bits of $Z_i \oplus Z_{i+\alpha}$ are required to solve these equations. Since $R4$ is not known, the attacker needs to guess all possible 2^{16} values of $R4$, and for each value solve the resulting linear equation, until a consistent solution is found.

In 2003 Goldberg, Wagner and Green's attack was improved by Elad Barkan, Eli Biham and Nathan Keller. They presented an attack that requires the keystream of any four frames. They described a way to write every output bit - even if on different frames - as a quadratic term of $R1_1$, $R2_1$, and $R3_1$ of the first frame. Given the output bits of four frames, they construct a system of quadratic equations for each of the 2^{16} possible values for $R4_1$ and solved it, until they found a consistent solution. Thus, they could recover the initial value of $R1_1$, $R2_1$, and $R3_1$ and by reversing the key setup, also the session key. Moreover, in the same paper Goldberg, Wagner and Green presented a ciphertext-only attack on the A5/2. To mount the attack they explored the fact that in SACCH channel error correction is applied before encryption. The error correction can be modelled as a multiplication of the plaintext message (denoted by P) by a constant matrix (denote by G), and XORed to a constant vector (denote by g), $M = (G \cdot P) \oplus g$. Now, let H be the parity-check matrix, i.e. $H \cdot (M \oplus g) = 0$ and C an encrypted message, $C = M \oplus Z$. Goldberg, Wagner and Green observed that:

$$H \cdot (C \oplus g) = H \cdot (M \oplus Z \oplus g) = H \cdot (M \oplus g) \oplus H \cdot Z = 0 \oplus H \cdot Z = H \cdot Z$$

Since H, C and g is known, they could convert the plaintext attack, described before, to the ciphertext-only attack.

The first attack on the A5/1 was proposed by Ross J. Anderson in 1994 when the A5/1 was not completely known. Anderson's basic idea was to guess the complete content of the registers $R1$ and $R2$ and about half of the register $R3$. In this way the clocking of all three registers is determined and the second half of $R3$ can be derived given 64 bits of keystream. In 1997 Jovan Dj. Golić, based on Anderson's description of the A5/1, suggested that the A5/1 should by prone to two types of attack, namely divide and conquer attack and time-memory trade-off attack. The divide and conquer attack is based on a divide and conquer algorithm design paradigm. The divide and conquer algorithm works by breaking down a problem into two or more sub-problems, that become simple enough to be solved. This behaviour could be use to break LFSRs. If the structure of the generator is known, and the secret key is the LFSR initial states, for a keystream generator consisting of n LFSRs, the total number of keys to be searched is $\prod_{i=1}^{n} (2^{L_i} - 1)$, where L_i is the length of i^{th} LFSR. Using a divide and conquer attack that determines individual LFSR states sequentially reduces the total number of keys to be search to $\sum_{i=1}^{n} (2^{L_i} - 1)$. The key idea of Golic's attack is to guess the lower half of each register (these bits determine the register clocking in the first few clock-cycles) and clock the cipher until the guessed bits "run-out". Each output bit immediately yields a linear equation in terms of the internal state bits belonging to the upper halves of three registers. Then guessing the clocking sequence is continued yielding again other linear equations that describe the output of the majority function. Whenever 64 linearly independent equations are obtained in this way the system is solved using Gaussian elimination. Golić predicts that about 20 bits should be guessed.

The second attack proposed by Golić was the time-memory trade-off attack. Time-memory trade-off attack is a practical method to decrease the time for key search. This type of attack can be applied, if the cipher has a small state size. Usually, in time-memory trade-off attacks, an attacker produces a number of output bits from certain states of the cipher and then keeps these cipher states and their corresponding outputs in pairs, in a sorted list. Then he searches to find a match between a received keystream sequence and the stored output sequences. If this occurs, the corresponding cipher state is obtained and from this state the key can be recovered.

Working time-memory trade-off attack against A5/1 was for the first time presented in 2000 by Alex Biryukov, Adi Shamir and David Wagner. They described two attacks. The first required a 300 GB table, two minutes of keystream and about one second of processing time on an average PC whereas the second attack requires 300GB table, two seconds of keystream and several minutes of processing time on an average PC.

Another line of attacks on the A5/1 started in 2001, when ideas from the correlation attacks were applied on the A5/1. Correlation attacks on LFSR-based keystream generators are based on statistical dependencies between observed keystream sequences and underlying shift register sequences. To recover an A5/1 key using a correlation attack an attacker is trying to discover how registers $R1$, $R2$ and $R3$ looked just before the key initialization. To achieve that the attacker tests by trial the encryption of many triplets $(R1_1,R2_1,R3_1)...(R1_n,R2_n,R3_n)$ until he finds proper one. But because he knows some statistical correlation between registers $R1$, $R2$, $R3$ and a cipher keystream he does not need to try all possible triplets, he could try only those which are correlated with the given (intercepted) keystream *Z*. In other words he uses the correlation for each triplet to asses the probability that the ith triplet ($R1_i$,$R2_i$,$R3_i$) will generate *Z*. The trial encryption is done only for triplets with the highest probability.

The first correlation attack on the A5/1 was published in 2003 by Patrik Ekdahl and Thomas Johansson. They observed that due to the linearity of the key setup, the initial internal value of *R*1, *R*2 and *R*3 at frame *j* is given by:

$$R1 = S_1 \oplus F_1^j$$
$$R2 = S_2 \oplus F_2^j$$
$$R3 = S_3 \oplus F_3^j$$

Where *S*1, *S*2, and *S*3 are the initial internal state of registers *R*1, *R*2 and *R*3 after the key setup using the correct key *K*, where the frame number is chosen to be zero, i.e.:

$$(S_1, S_2, S_3) = keysetup(K, 0)$$

Similarly, let F_1^j, F_2^j and F_3^j are the initial internal state of the registers *R*1, *R*2, and *R*3 after a key setup using all zeros as the key, but with frame number *j*, i.e.:

$$(F_1^j, F_2^j, F_3^j) = keysetup(0, j)$$

Let $S_i(l_i)$ and $F_i(l_i)$ denote the output bit of registers S_i and F_i after they were clocked l_i times from its initial state until the end of cycle *t*. The idea behind the attack is to observe that:

$$S_1(l_1) \oplus S_2(l_2) \oplus S_3(l_3) = Z(t) \oplus F_1^j(l_1) \oplus F_2^j(l_2) \oplus F_3^j(l_3) \qquad (1)$$

The equation (1) holds in two cases:

1. The LFSRs are really clocked l_1 , l_2 , l_3 at time *t*. If so, the expression will be true with probability 1.
2. If the condition in 1) is not fulfilled, the expression will still be true with probability ½ (i.e. by pure chance).

Therefore, equation (1) holds with probability:

$$p = \frac{1}{2} + \frac{1}{2}\,(\,Pr\{(l_1, l_2, l_3)\,at\,time\,t\})$$

Where Pr $\{(l_1 , l_2 , l_3)$ at time $t\}$ is the probability that at time *t* the LFSRs were clocked exactly l_1, l_2, l_3 times, respectively. The probability that at time *t* the LFSRs were clocked l_1, l_2, l_3 times is:

$$Pr\{(l_1, l_2, l_3)\,at\,time\,t\} = \frac{\binom{t}{t-l_1}\binom{t-(t-l_1)}{t-l_2}\binom{t-(t-l_1)-(t-l_2)}{t-l_3}}{4^t}$$

This means that the relation (1) is biased ($p > ½$). The probability *p* gives the estimation of the corresponding linear combination for one frame *j*.

In 2004 Alexander Maximov, Thomas Johansson and Steve Babbage observed that the aforementioned bias could be improved. Assume that at time *t* the LFSRs are clocked l_1, l_2, and l_3

times, respectively. Then we also assume that at time $t + 1$ the third LFSR is not clocked. Under these two assumptions, $R3$ contributes the same bit to output bits t and $t+1$. Thus, $R3$'s contribution is eliminated from the difference of these two output bits, and the following equation holds:

$$S_1'(l_1) \oplus S_2'(l_2) = Z'(t) \oplus F'^{j}_1(l_1) \oplus F'^{j}_2(l_2) \oplus F'^{j}_3(l_3) \tag{2}$$

When $S'(l) = S(l) \oplus S(l{+}1)$ and so on. Assuming the values in the clocking taps are uniformly distributed, the assumption that at time $t+1$ the third LFSR is not clocked holds with probability ¼ and now:

$$p=\frac{1}{2}+\frac{1}{2}\left(\frac{1}{4}\,Pr\{(l_1,l_2)\,at\,time\,t\}\right)=\frac{1}{2}+\frac{1}{8}Pr\{(l_1,l_2)\,at\,time\,t\}$$

When:

$$Pr\{(l_1,l_2)\,at\,time\,t\}=\frac{\binom{t}{t-l_1}\binom{l_1}{t-l_2}}{2^{3t-(l_1+l_2)}}$$

Note, that ¼ Pr$\{(l_1, l_2)$ at time $t\}$ > Pr$\{(l_1, l_2, l_3)$ at time $t\}$ so this gives us a larger bias when estimating the value of linear combinations of $S_i(l_i)$'s. Below is a comparison of these probabilities.

(l_1, l_2, l_3), t	Pr$\{(l_1, l_2, l_3)$ at time $t\}$ $\cdot 10^4$	¼ Pr$\{(l_1, l_2)$ at time $t\}$ $\cdot 10^4$
(76, 76, 76), 101	9.7434	22.1207
(79, 79, 79), 105	9.2012	21.2840
(80, 80, 80), 105	6.6388	19.3778
(79, 80, 81), 106	8.3858	20.8899
(82, 82, 82), 109	8.7076	20.5083

Table 2. Comparison of Ekdahl, Johansson and Maximov, Johansson, Babbage biases

One year later Elad Barkan and Eli Biham observe that the bias of the correlation can be further improved by checking the value of clocking taps $C1$ and $C2$. The value of clocking taps at time t could be simply found:

$$C1 = S_1(l_1+10) \oplus F^{j}_1(l_1+10)$$
$$C2 = S_2(l_2+11) \oplus F^{j}_2(l_2+11)$$

Now we have two distinct cases. The first of the two cases is when $C1 \neq C2$. Due to the clocking mechanism, $R3$ is always clocked in this case. However, in the second case, when $C1 = C2$, we gain a factor two increase in the bias. In this case, both $R1$ and $R2$ are clocked, and $R3$ is clocked with probability ½. Therefore, when $C1{=}C2$, the equation (2) holds with probability $\frac{1}{2}+\frac{1}{4}Pr\{(l_1,l_2)\,at\,time\,t\}$ compared to probability $\frac{1}{2}+\frac{1}{8}Pr\{(l_1,l_2)\,at\,time\,t\}$.

The table below presents a comparison of described attacks.

Attack	**Required know keystream frames**	**Average computation time**	**Success ratio**
Ekdahl, Johansson *(Pentium 4, 1.8GHz)*	70000 (322 s)	5 min	76%
	50000 (230 s)	4 min	33%
	30000 (138 s)	3 min	3%
Maximov, Johansson, Babbage *(Pentium 4, 2.4GHz)*	10000 (46 s)	10 min	99.99%
	5000 (23 s)	10 min	85%
	2000 (9.2 s)	10 min	5%
Barkan, Biham *(Pentium 4, 1.8GHz)*	2000 (9.2 s)	133 s	91%
	1500 (6.9 s)	7.2 min	54%

Table 3. Comparison of correlation attacks

The last group of attacks are hardware-assisted attacks. In 2001 Jörg Keller and Birgit Seitz mount an attack based on a FPGA chips. The FPGA (Field-Programmable Gate Array) chip is a semiconductor device that can be programmed by the customer after manufacturing. They implemented the attack on a Xilinx XC4062 FPGA. The attack is based on a simple guess-and-determines attack proposed by Anderson in 1994 where the shorter registers $R1$ and $R2$ are guessed and the longer register $R3$ is to be determined. To mount the attack only 64 bits of a known keystream are required. The attack time is about 236 days. To speed up the process Keller and Seitz proposed to exclude early recognized state candidates that for sure would be wrong. Unfortunately the algorithm excludes many valid state candidates and therefore the success probability of the attack is only 18%.

In 2008 the attack was refined by Timo Gendrullis, Martin Novotný and Andy Rupp. The attack was implemented on a special-purpose hardware device, called COPACOBANA. The COPACOBANA (Cost-Optimized Parallel Code Breaker) machine is a low-cost (about $10.000) cluster consisting of 120 Xilinx Spartan3-XC3S1000 FPGAs. The machine was the result of the joint work of the two universities: Ruhr-Universität Bochum and Christian-Albrechts-Universität zu Kiel. Because the attack excludes less state candidates then Keller's and Seitz's attack it's success probability is 100%. The flowchart of the attack is given in Figure 9.

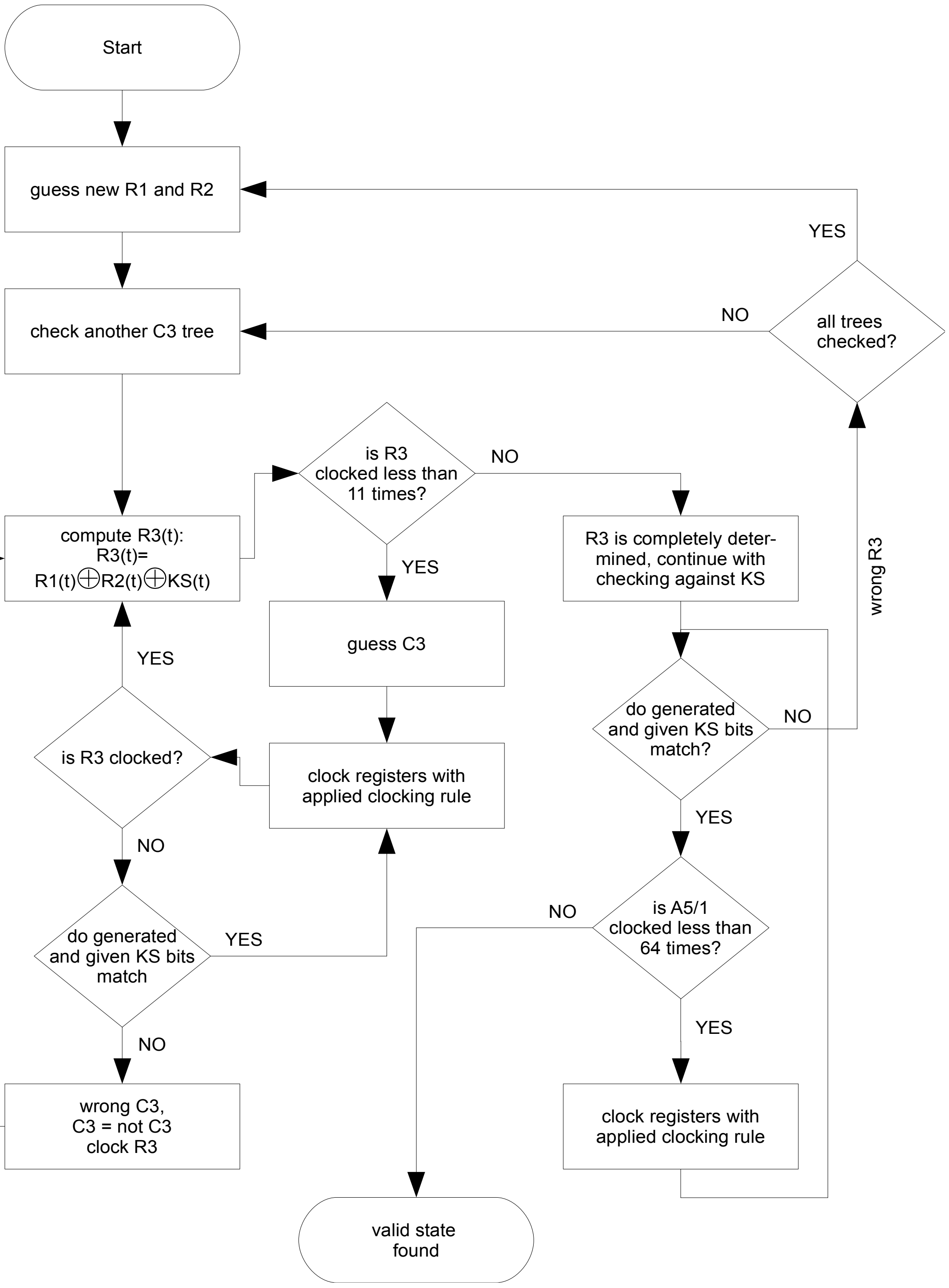
Start
guess new R1 and R2
check another C3 tree
YES
NO
all trees checked?
is R3 clocked less than 11 times?
NO
compute R3(t): R3(t)= R1(t)⊕R2(t)⊕KS(t)
YES
R3 is completely determined, continue with checking against KS
wrong R3
guess C3
YES
do generated and given KS bits match?
NO
is R3 clocked?
clock registers with applied clocking rule
YES
NO
NO
is A5/1 clocked less than 64 times?
do generated and given KS bits match
YES
NO
YES
wrong C3, C3 = not C3 clock R3
clock registers with applied clocking rule
valid state found

Fig 9. Flowchart of the Gendrullis, Novotný and Rupp's attack

To find a wrong state candidate Gendrullis, Novotný and Rupp use simply strategy. Fore every cipher cycle *t* botch value of the *C*3 must be checked. This leads to a tree presented on Figure 10.

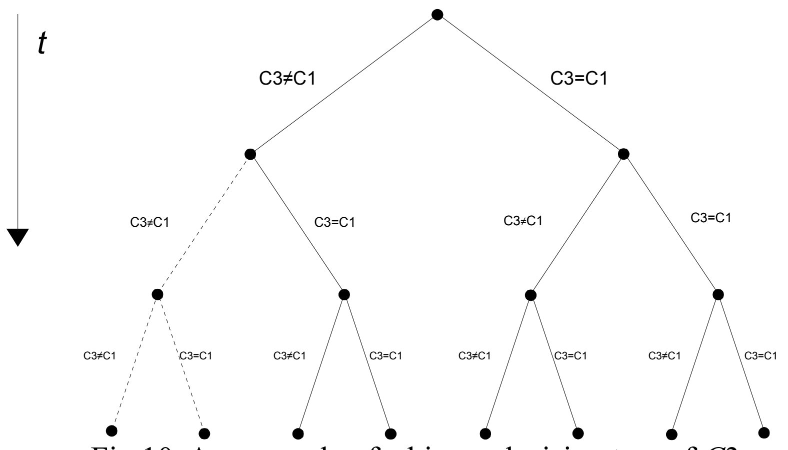

Fig 10. An example of a binary decision tree of *C*3

Let for some *t* registers *R*1 and *R*2 were clocked but *R*3 stood still. Because at this stage all output bits are known (*R*1(*t*), *R*2(*t*) were guessed, *R*3(*t*) determined in the previous iteration) a consistency check could be performed:

$$R1(t) \oplus R2(t) \oplus R3(t) = \text{keystream}(t)$$

If a given equation is not fulfilled all subtrees generated from this state could be safely discarded (dotted line on Figure 10).
The average attack time on the COPACOBANA machine is about 7 hours.

Conclusion

After careful analysis of GSM security it can be seen that although GSM had security in mind when drafting the original specifications, the GSM fails to deliver solid security for its users. GSM designers underestimated public cryptography progress, and computers speed progress while at the same time they overestimated their ability to keep technology in secret. GSM's faults result mainly from a combination of designing algorithms in security through obscurity model and a deliberate weakening of the system.

Fortunately for most GSM users however, the concerns are not great. Serious threats such as phone call eavesdropping are still not easily carried out and easier attacks like Caller ID spoofing have not become very popular amongst fraudsters, so that the casual telephone user can still feel relatively safe. However, those using GSM for highly sensitive information should be absolutely sure to use encrypted cell phones. There are many solutions on the market, beginning with expensive crypto phones for Government or military and ending with relatively cheap encryption software for Symbian or Windows Mobile phones, so that everyone should be able to find a solution that will suit his needs.

Moreover, system architects and security experts should keep in mind that aforementioned GSM weaknesses could be used against IT systems as well. Today risk is especially high because more and more services allow users to put content through text messages, moreover cell phone companies, credit card companies, and even banks rely on Caller ID information to authenticate their customers' identity. The solutions to those problems are quite simple, but come with the expense of usability, so companies should weigh the security risk of this issue against the impact on the ease of use of their service.

It is heartening, however, that GSM designers learnt their lesson and security in third-generation mobile telephony is significantly improved. In spite of the fact that the UMTS security was built on the GSM security, the 3G tries to correct the problems with 2G by adding new features such as BTS-to-MS authentication and by fixing know weaknesses, for example changing the A5/2 to the KASUMI. But, nonetheless the security of 3G telephony or its successor will be better and better, it is important to always remember that there is only one electrical communication system that will always remain 100% eavesdrop proof, that being electrochemical signals between our brain cells.

References

1. Internet World Stats, *Internet usage statistics*, www.internetworldstats.com, 2011
2. GSM Association, www.gsmworld.com, 2011
3. Marc Briceno, Ian Goldberg, David Wagner, *An implementation of the GSM A3A8 algorithm*, www.gsm-security.net/papers/a3a8.shtml, 1998
4. Stuart Wray, *COMP128: A Birthday Surprise*, www.stuartwray.net, 2003
5. European Telecommunications Standards Institute (ETSI), *Universal Mobile Telecommunications System (UMTS); Specification of the 3GPP confidentiality and integrity algorithms; Document 2: Kasumi specification*, www.etsi.org, 2007
6. The OpenBTS Project, openbts.sourceforge.net, 2010
7. Nitesh Dhanjani, *Twitter and Jott Vulnerable to SMS and Caller ID Spoofing*, www.oreillynet.com/onlamp/blog, 2007
8. Vassilis Prevelakis, Diomidis Spinellis, *The Athens Affair*, IEEE Spectrum, July 2007
9. John Leyden, *SMS security risks highlighted by Friends Reunited hacking case*, www.theregister.co.uk, 2002
10. Josyula R. Rao, Pankaj Rohatgi, Helmut Scherzer, Stephane Tinguely, *Partitioning Attacks: Or How to Rapidly Clone Some GSM Cards*, www.research.ibm.com, 2002
11. Reinhard Wobst, *Cryptology Unlocked*, Wiley, 2007
12. Ian Goldberg, David Wagner, Lucky Green, *The (Real-Time) Cryptanalysis of A5/2*, Lecture Notes in Computer Science 2729, Springer Berlin / Heidelberg, 2003
13. Elad Barkan, Eli Biham, Nathan Keller, *Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication*, Technion - Computer Science Department - Technical Report CS-2006-07–2006, www.cs.technion.ac.il, 2006
14. Ross J. Anderson, *A5 (was: Hacking digital phones),* Newsgroup Message from sci.crypt, 1994
15. Jovan Dj. Golić, *Cryptanalysis of Alleged A5 Stream Cipher*, Lecture Notes in Computer Science 1233, Springer Berlin / Heidelberg, 1997
16. K.Y. Lam, I. Shparlinski,H. Wang, C. Xing, *Cryptography and Computational Number Theory*, Birkhäuser, 2001
17. I.C. Göknar, L. Sevgi, *Complex Computing-Networks*, Springer-Verlag, 2006
18. Alex Biryukov, Adi Shamir, David Wagner, *Real Time Cryptanalysis of A5/1 on a PC*, www.cryptome.org, 2000
19. Patrik Ekdahl, Thomas Johansson, *Another attack on A5/1*, IEEE Transactions on Information Theory 49, 2003
20. Alexander Maximov, Thomas Johansson, Steve Babbage, *An Improved Correlation Attack on A5/1*, Selected Areas in Cryptography 2004, 2004
21. Jörg Keller, Birgit Seitz, *A Hardware-Based Attack on the A5/1 Stream Cipher*, fernuni-hagen.de, 2001
22. Timo Gendrullis, Martin Novotný, Andy Rupp, *A Real-World Attack Breaking A5/1 within Hours*, Lecture Notes in Computer Science 5154, Springer Berlin / Heidelberg, 2008
23. Friedhelm Hillebrand, *GSM and UMTS: the creation of global mobile communication*, Wiley, 2002
24. SMSspoofing, www.smsspoofing.com

25. Pacharawit Topark-Ngarm, Panupat Poocharoen, *GSM security Vulnerability*, Oregon State University, islab.oregonstate.edu, 2009
26. Paulo S. Pagliusi, *A Contemporary Foreword on GSM Security*, Infrastructure security: international conference, InfraSec 2002, Springer, 2002
27. Jeremy Quirke, *Security in the GSM system,* www.ausmobile.com, 2004
28. Man Young Rhee, *Mobile Communication Systems and Security*, Wiley, 2009