# Implementation of a Brute Force Attack on A5/1 Cryptographical Algorithm in a GPU-based Volunteering Computing Project

Bulavintsev V., Semenov A., and Zaikin O.

Matrosov Institute for System Dynamics and Control Theory of Siberian Branch of Russian Academy of Sciences (ISDCT SB RAS)

**Abstract.** We present an advanced brute force attack on the A5/1 cryptoalgorithm, that is still broadly used in modern GSM networks. We use a well-known idea introduced by R.Anderson more than 20 years ago to greatly reduce the search space. The primary contribution of this article is the implementation of the Anderson's attack on a GPU platform with the bitslice technique. The preliminary estimates of the attack's speed showed that, with the use of a GPUs processing power, the attack could be conducted in the real time on a modern computer cluster or in a volunteer computing project. To verify our estimates with the use of BOINC technology we concieved the specialized volunteering computing project and executed our variant of Anderson's attack within it. In the 7 days the project lasted, 10 A5/1 cryptanalysis problems were solved. The results presented in this work provide yet another proof of A5/1's cryptographical weakness that make it totally unsuitable for transmission of any kind of sensitive data through modern GSM networks.

**Keywords:** A5/1 algorithm, brute-force cryptanalysis, GPU, volunteering computing

## 1 Introduction

A5/1 algorithm is a keystream generator with key length of 64 bits. It is used to encrypt voice and SMS traffic in 2nd generation ("2G") GSM networks. 3rd generation GSM networks accept 2G communication protocol for backward compatibility. A practice of sending voice traffic through 2G protocols in 3G networks to conserve bandwith increase service availability is widely adopted among mobile phone operators. A widely adopted practice among mobile operators is to still use 2G protocols to route voice traffic even in 3rd generation GSM networks.

One can name A5/1 as one of the most publicly recognized cryptographical algorithms, along with RSA and DES, it's discussion reaching far beyond the borders of professional cryptographers community. For example, article [**?**] publicly examines NSA's ability to efficiently decrypt A5/1. So, in this article we won't touch on the details of A5/1's creation history and the reasons of it's rise as the

world's 'de facto' mobile communication standard. Milestones in cryptanalysis of A5/1 would be briefly outlined in Related Works section.

Among all the differrent methods of A5/1 cryptanalysis we distinguish those which were realized in practice and allowed to reliably conduct the cryptanalysis procedure on an non-weakened variant of algorithm. Apparently, the first such attack was performed in 2008 with the help of the special FPGA-based computational platform COPACOBANA [**?**]. In 2009 distributed algorithms for boolean satisfiability (SAT) were used to solve several cryptanalysis instances of stock A5/1 in a specialized grid system 'BNB-Grid' [**?**]. These results were further improved in 2011 [**?**]. By the end of 2009 the 'A5/1 Cracking Project' had published rainbow tables [**?**] for A5/1. Provided with 8 bursts (912 bits) of keystream these tables allowed to find the secret key in less than a minute with more than 85% probabilty. Despite the huge (over 2 Tb) size, to this day these rainbow tables provide the most practical method of A5/1 cryptanalysis. It's main shortcoming is that the probability of success is significantly less than 100%. Meanwhile, the growth of computational power of GPUs and FPGAs made practical the attack based on reduction of search space from $2^64$ to $2^53$, which was described by R. Anderson in 1994. As was mentioned earlier, FPGA-based variant of the Anderson's attack was already performed in 2008 by COPACOBANA creators. So, the primary goal of our work is to demonstrate the viability of GPU-based variant of the attack. Let us note that GPUs are much easier to operate than FPGAs, and the former belong to the class of consumer-grade devices and could be found in any modern PC, while the latter belong to the class of specialized equipment. With the usage of the BOINC software platform [**?**], this qualities of GPUs allowed us to implement the attack in the form of a volunteering computing project using idle computational capabilites of the project members home PCs. Our estimates of attack's speed were based on our previous work [**?**].

Let's make a brief outline of the article's contents. In Section 2 we describe the A5/1 algorithm along with some advanced brute-force attacks on it. **??** introduces bit-slicing technique and goes through important details of implementing the Anderson's attack with it. **??** provides a look into the internal organization of the volunteering project we created for conducting the attack with GPUs. **??** contains the retrospective of A5/1 cryptanalysis works related to our study.

## 2 A5/1 algorithm and some attack on it

A5/1 keystream generator consists of 3 linear feedback shift registers (LFSRs) [**?**], defined by the following primitive polynomials:

$$X^{19} + X^{18} + X^{17} + X^{14} + 1, LFSR1;$$
$$X^{22} + X^{28} + 1, LFSR2;$$
$$X^{23} + X^{22} + X^{21} + X^{8} + 1, LFSR3.$$

The illustration of A5/1 generator's work scheme can be seen at **??**.

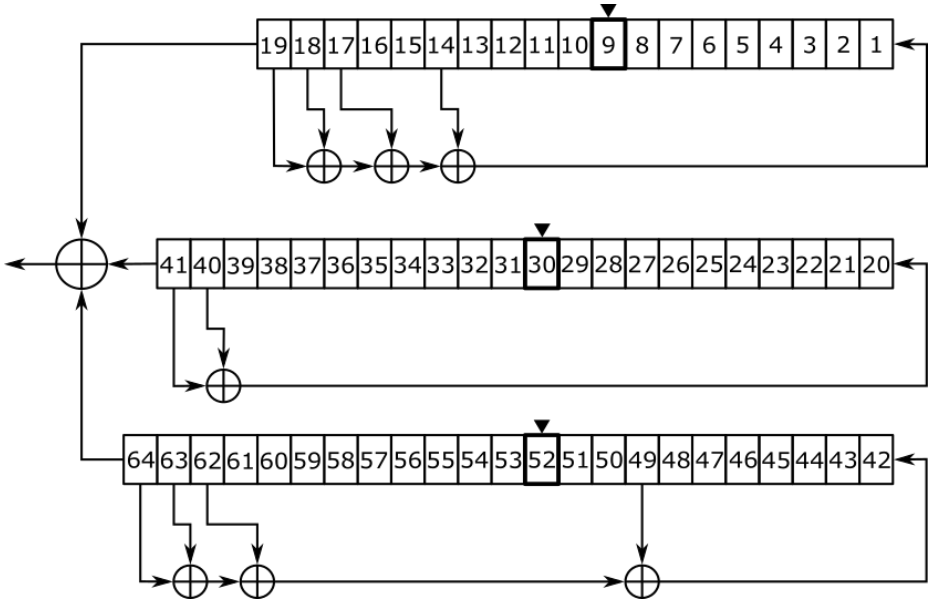


**Fig. 1.** A5/1 generator work scheme

The outputs of LFSRs are mixed with linear function, which provides the perfect corellation immunity. Non-linearity of the equations is achieved by clocking the registers asynchronously - for each clocking of the whole generator any of the 3 registers could be clocked or it could retain it's current state. Register with index $j \in \{1, 2, 3\}$ will be clocked if the following boolean function $\chi_j$ becomes 1:

$$\chi_j = (b_j \equiv majority(b_1, b_2, b_3));$$
$$majority(A, B, C) = (A \wedge B) \vee (A \wedge C) \vee (B \vee C).$$

Here $b_1, b_2, b_3$ denote clocking bits marked at **??** by black wedges. Conversely, if at some moment $\chi_j = 0$, LFSR $j$ won't clock (it will remain in it's last state).

Generator A5/1 is used in the GSM protocol for high-speed encryption of large volumes of information with the short secret key. The whole process is split into 'sessions' about 3,5 hours long. Each session uses it's own 'session key'. We won't touch on the topic of specialized protocols used in GSM networks to build and transfer the session key.

Along the message data, GSM protocol sends error correction data for the message. The message complete with error correction data constitutes a 456 bits long 'frame', which is further broken down into 4 'bursts' of 114 bits. The bursts are than encrypted and send over the air. To encrypt a burst the A5/1 generator is initialized with the 64-bit 'local key', that is built using session key and a natural number called 'the frame number' (FN). After the encryption of one burst is complete, the frame number is increased by 1. When FN overflows,

the session ends and a new session is initialized (hence the 3,5 session length). FN is always known from the open data transmitted over the network.

If the message length is less than 23 bytes, the message would be filled with fixed pattern padding to 23 bytes. Some GSM technical messages used during the voice call are fixed length and always padded. As padding is always the same, and it is encrypted by some local key, we got a typical plain text attack scenario at hand [**?**]. Indeed, the padding plays the role of the known plaintext, allowing the attacker to get the corresponding keystream fragment. This vulnerability allows the attacker to get no less than two frames (912 bits) of the keystream. It was demonstrated in [**?**] that the knowledge of even one local key and FN is enough to efficiently restore the session key, which makes the decryption of the whole session possible.

The described vulnerability of GSM protocol allows to build some successful attacks on it, based on the idea of 'advanced brute force'. In particular, the large-scale preprocessing made possible the creation of the rainbow tables, which, provided 8 bursts of keystream, allows to determine the session key with probability in less than a minute with $> 85\%$ probability. The tables take about 2Tb of disk space. This attack is presented in detail in [**?**], and still stands among the most practical ones. Instead, we will focus on the idea of an attack that was suggested by Ross Anderson in 1994 in a small essay on the A5/1 cryptographic strength [**?**]. Next we describe the essence of the Anderson's attack.

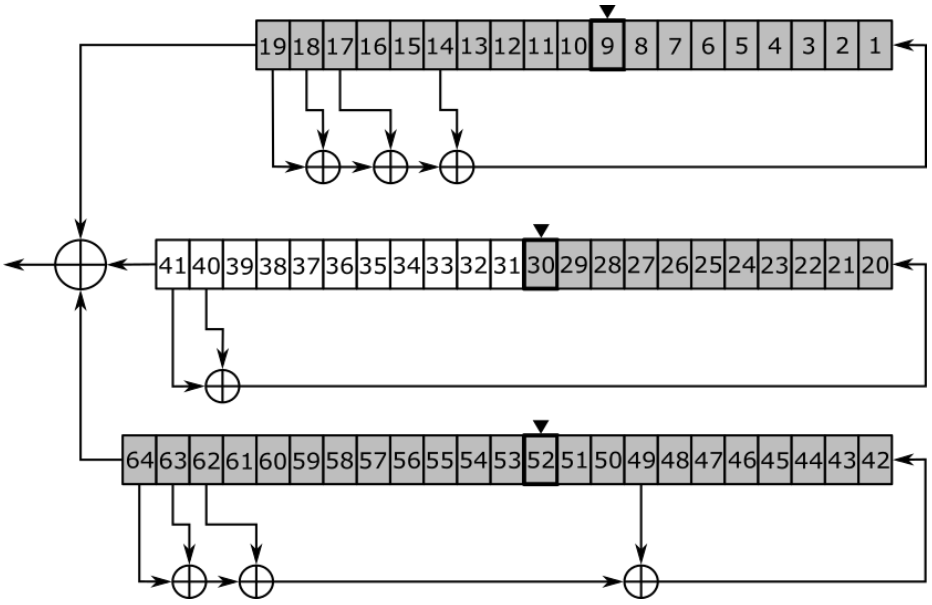


**Fig. 2.** The set of guessing bits used in Anderson's attack (greyed out).

Anderson's attack is a typical example of a 'guess and determine attack' (see [**?**]). Suppose we know the bits filling the 1st and the 3rd LFSR, and bits of the 2nd LFSR from the beginning of the register to the clocking bit (bits 31 to 41, see Fig. 2). Next suppose we know 64 bits of the keystream. Now, as was shown by R. Anderson, 11 unknown bits of the 2nd LFSR could be figured out without any additional guesses. This happens because the clocking bits are known (and so is the clocking schedule for the next 11 clockings of the 2nd LFSR), and known are 2 out of 3 XOR-ed LFSR output bits and the result of the XOR operation (from the keystream). Therefore, one can efficiently work out the unknown bits of the LFSR 2 one by one, by clocking the generator and applying XOR operation to corresponding keystream bits and output bits of LFSRs 1 and 3.

The description of the algorithm of determination of unknown 11 bits of LFSR 2 provided above makes obvious the possibility to mount a brute force attack on the A5/1 generator over the search space of $2^{53}$. The simplicity of the algorithm provides an opportunity to implement it on a specialized computational architecture. One such implementation was built with FPGAs by authors of [**?**]. In the following sections we describe our implementation of this attack for modern GPUs.

Winery [1] is graphical modeling tool.

Simple Figure

**Fig. 3.** Simple Figure

**Table 1.** Simple Table

Simple Table

cref Demonstration: Cref at beginning of sentence, cref in all other cases.

Figure 3 shows a simple fact, although Fig. 3 could also show something else.

Table 1 shows a simple fact, although Table 1 could also show something else.

Section 1 shows a simple fact, although Sect. 1 could also show something else.

Brackets work as designed: <test>

The symbol for powerset is now correct: $\wp$ and not a Weierstrass p ($\wp$).

1. All these items... 2. ...appear in one line 3. This is enabled by the paralist package.

"something in quotes" using plain tex or use "the enquote command".

## 3 Conclusion and Outlook

**Acknowledgments** ...

In the bibliography, use `\textsuperscript` for “st”, “nd”, ...: E.g., “The 2nd conference on examples”. When you use JabRef, you can use the clean up command to achieve that.

## References

1. Kopp, O., et al.: Winery - A Modeling Tool for TOSCA-based Cloud Applications. In: Proceedings of 11th International Conference on Service-Oriented Computing (ICSOC'13). LNCS, vol. 8274, pp. 700–704. Springer Berlin Heidelberg (2013)

All links were last followed on October 5, 2014.