Lecture Notes in Computer Science 1978

Bruce Schneier (Ed.)

Fast Software Encryption

7th International Workshop, FSE 2000
New York, NY, USA, April 2000
Proceedings

Springer

Bruce Schneier (Ed.)

Fast Software Encryption

7th International Workshop, FSE 2000
New York, NY, USA, April 10-12, 2000
Proceedings

Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editor

Bruce Schneier
Counterpane Internet Security, Inc.
3031 Tisch Way, Suite 100PE, San Jose, CA 95128, USA
E-mail: schneier@counterpane.com

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Fast software encryption : 7th international workshop ; proceedings /
FSE 2000, New York, NY, April 10 - 12, 2000. Bruce Schneider (ed.). -
Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ;
Milan ; Paris ; Singapore ; Tokyo : Springer, 2001
(Lecture notes in computer science ; Vol. 1978)
ISBN 3-540-41728-1

CR Subject Classification (1998): E.3, F.2.1, E.4, G.4

ISSN 0302-9743
ISBN 3-540-41728-1 Springer-Verlag Berlin Heidelberg New York

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

Printed in Germany

Typesetting: Camera-ready by author, data conversion by PTP Berlin, Stefan Sossna
Printed on acid-free paper SPIN 10781399 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 1978

Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Singapore
Tokyo

Lecture Notes in Computer Science

For information about Vols. 1–1920
please contact your bookseller or Springer-Verlag

Vol. 1921: S.W. Liddle, H.C. Mayr, B. Thalheim (Eds.), Conceptual Modeling for E-Business and the Web. Proceedings, 2000. X, 179 pages. 2000.

Vol. 1922: J. Crowcroft, J. Roberts, M.I. Smirnov (Eds.), Quality of Future Internet Services. Proceedings, 2000. XI, 368 pages. 2000.

Vol. 1923: J. Borbinha, T. Baker (Eds.), Research and Advanced Technology for Digital Libraries. Proceedings, 2000. XVII, 513 pages. 2000.

Vol. 1924: W. Taha (Ed.), Semantics, Applications, and Implementation of Program Generation. Proceedings, 2000. VIII, 231 pages. 2000.

Vol. 1925: J. Cussens, S. Džeroski (Eds.), Learning Language in Logic. X, 301 pages 2000. (Subseries LNAI).

Vol. 1926: M. Joseph (Ed.), Formal Techniques in Real-Time and Fault-Tolerant Systems. Proceedings, 2000. X, 305 pages. 2000.

Vol. 1927: P. Thomas, H.W. Gellersen, (Eds.), Handheld and Ubiquitous Computing. Proceedings, 2000. X, 249 pages. 2000.

Vol. 1928: U. Brandes, D. Wagner (Eds.), Graph-Theoretic Concepts in Computer Science. Proceedings, 2000. X, 315 pages. 2000.

Vol. 1929: R. Laurini (Ed.), Advances in Visual Information Systems. Proceedings, 2000. XII, 542 pages. 2000.

Vol. 1931: E. Horlait (Ed.), Mobile Agents for Telecommunication Applications. Proceedings, 2000. IX, 271 pages. 2000.

Vol. 1658: J. Baumann, Mobile Agents: Control Algorithms. XIX, 161 pages. 2000.

Vol. 1756: G. Ruhe, F. Bomarius (Eds.), Learning Software Organization. Proceedings, 1999. VIII, 226 pages. 2000.

Vol. 1766: M. Jazayeri, R.G.K. Loos, D.R. Musser (Eds.), Generic Programming. Proceedings, 1998. X, 269 pages. 2000.

Vol. 1791: D. Fensel, Problem-Solving Methods. XII, 153 pages. 2000. (Subseries LNAI).

Vol. 1799: K. Czarnecki, U.W. Eisenecker, Generative and Component-Based Software Engineering. Proceedings, 1999. VIII, 225 pages. 2000.

Vol. 1812: J. Wyatt, J. Demiris (Eds.), Advances in Robot Learning. Proceedings, 1999. VII, 165 pages. 2000. (Subseries LNAI).

Vol. 1932: Z.W. Raś, S. Ohsuga (Eds.), Foundations of Intelligent Systems. Proceedings, 2000. XII, 646 pages. (Subseries LNAI).

Vol. 1933: R.W. Brause, E. Hanisch (Eds.), Medical Data Analysis. Proceedings, 2000. XI, 316 pages. 2000.

Vol. 1934: J.S. White (Ed.), Envisioning Machine Translation in the Information Future. Proceedings, 2000. XV, 254 pages. 2000. (Subseries LNAI).

Vol. 1935: S.L. Delp, A.M. DiGioia, B. Jaramaz (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2000. Proceedings, 2000. XXV, 1250 pages. 2000.

Vol. 1936: P. Robertson, H. Shrobe, R. Laddaga (Eds.), Self-Adaptive Software. Proceedings, 2000. VIII, 249 pages. 2001.

Vol. 1937: R. Dieng, O. Corby (Eds.), Knowledge Engineering and Knowledge Management. Proceedings, 2000. XIII, 457 pages. 2000. (Subseries LNAI).

Vol. 1938: S. Rao, K.I. Sletta (Eds.), Next Generation Networks. Proceedings, 2000. XI, 392 pages. 2000.

Vol. 1939: A. Evans, S. Kent, B. Selic (Eds.), «UML» – The Unified Modeling Language. Proceedings, 2000. XIV, 572 pages. 2000.

Vol. 1940: M. Valero, K. Joe, M. Kitsuregawa, H. Tanaka (Eds.), High Performance Computing. Proceedings, 2000. XV, 595 pages. 2000.

Vol. 1941: A.K. Chhabra, D. Dori (Eds.), Graphics Recognition. Proceedings, 1999. XI, 346 pages. 2000.

Vol. 1942: H. Yasuda (Ed.), Active Networks. Proceedings, 2000. XI, 424 pages. 2000.

Vol. 1943: F. Koornneef, M. van der Meulen (Eds.), Computer Safety, Reliability and Security. Proceedings, 2000. X, 432 pages. 2000.

Vol. 1944: K.R. Dittrich, G. Guerrini, I. Merlo, M. Oliva, M.E. Rodriguez (Eds.), Objects and Databases. Proceedings, 2000. X, 199 pages. 2001.

Vol. 1945: W. Grieskamp, T. Santen, B. Stoddart (Eds.), Integrated Formal Methods. Proceedings, 2000. X, 441 pages. 2000.

Vol. 1946: P. Palanque, F. Paternò (Eds.), Interactive Systems. Proceedings, 2000. X, 251 pages. 2001.

Vol. 1947: T. Sørevik, F. Manne, R. Moe, A.H. Gebremedhin (Eds.), Applied Parallel Computing. Proceedings, 2000. XII, 400 pages. 2001.

Vol. 1948: T. Tan, Y. Shi, W. Gao (Eds.), Advances in Multimodal Interfaces – ICMI 2000. Proceedings, 2000. XVI, 678 pages. 2000.

Vol. 1949: R. Connor, A. Mendelzon (Eds.), Research Issues in Structured and Semistructured Database Programming. Proceedings, 1999. XII, 325 pages. 2000.

Vol. 1950: D. van Melkebeek, Randomness and Completeness in Computational Complexity. XV, 196 pages. 2000.

Vol. 1951: F. van der Linden (Ed.), Software Architectures for Product Families. Proceedings, 2000. VIII, 255 pages. 2000.

Vol. 1952: M.C. Monard, J. Simão Sichman (Eds.), Advances in Artificial Intelligence. Proceedings, 2000. XV, 498 pages. 2000. (Subseries LNAI).

Vol. 1953: G. Borgefors, I. Nyström, G. Sanniti di Baja (Eds.), Discrete Geometry for Computer Imagery. Proceedings, 2000. XI, 544 pages. 2000.

Vol. 1954: W.A. Hunt, Jr., S.D. Johnson (Eds.), Formal Methods in Computer-Aided Design. Proceedings, 2000. XI, 539 pages. 2000.

Vol. 1955: M. Parigot, A. Voronkov (Eds.), Logic for Programming and Automated Reasoning. Proceedings, 2000. XIII, 487 pages. 2000. (Subseries LNAI).

Vol. 1956: T. Coquand, P. Dybjer, B. Nordström, J. Smith (Eds.), Types for Proofs and Programs. Proceedings, 1999. VII, 195 pages. 2000.

Vol. 1957: P. Ciancarini, M. Wooldridge (Eds.), Agent-Oriented Software Engineering. Proceedings, 2000. X, 323 pages. 2001.

Vol. 1960: A. Ambler, S.B. Calo, G. Kar (Eds.), Services Management in Intelligent Networks. Proceedings, 2000. X, 259 pages. 2000.

Vol. 1961: J. He, M. Sato (Eds.), Advances in Computing Science – ASIAN 2000. Proceedings, 2000. X, 299 pages. 2000.

Vol. 1963: V. Hlaváč, K.G. Jeffery, J. Wiedermann (Eds.), SOFSEM 2000: Theory and Practice of Informatics. Proceedings, 2000. XI, 460 pages. 2000.

Vol. 1964: J. Malenfant, S. Moisan, A. Moreira (Eds.), Object-Oriented Technology. Proceedings, 2000. XI, 309 pages. 2000.

Vol. 1965: Ç. K. Koç, C. Paar (Eds.), Cryptographic Hardware and Embedded Systems – CHES 2000. Proceedings, 2000. XI, 355 pages. 2000.

Vol. 1966: S. Bhalla (Ed.), Databases in Networked Information Systems. Proceedings, 2000. VIII, 247 pages. 2000.

Vol. 1967: S. Arikawa, S. Morishita (Eds.), Discovery Science. Proceedings, 2000. XII, 332 pages. 2000. (Subseries LNAI).

Vol. 1968: H. Arimura, S. Jain, A. Sharma (Eds.), Algorithmic Learning Theory. Proceedings, 2000. XI, 335 pages. 2000. (Subseries LNAI).

Vol. 1969: D.T. Lee, S.-H. Teng (Eds.), Algorithms and Computation. Proceedings, 2000. XIV, 578 pages. 2000.

Vol. 1970: M. Valero, V.K. Prasanna, S. Vajapeyam (Eds.), High Performance Computing – HiPC 2000. Proceedings, 2000. XVIII, 568 pages. 2000.

Vol. 1971: R. Buyya, M. Baker (Eds.), Grid Computing – GRID 2000. Proceedings, 2000. XIV, 229 pages. 2000.

Vol. 1972: A. Omicini, R. Tolksdorf, F. Zambonelli (Eds.), Engineering Societies in the Agents World. Proceedings, 2000. IX, 143 pages. 2000. (Subseries LNAI).

Vol. 1973: J. Van den Bussche, V. Vianu (Eds.), Database Theory – ICDT 2001. Proceedings, 2001. X, 451 pages. 2001.

Vol. 1974: S. Kapoor, S. Prasad (Eds.), FST TCS 2000: Foundations of Software Technology and Theoretical Computer Science. Proceedings, 2000. XIII, 532 pages. 2000.

Vol. 1975: J. Pieprzyk, E. Okamoto, J. Seberry (Eds.), Information Security. Proceedings, 2000. X, 323 pages. 2000.

Vol. 1976: T. Okamoto (Ed.), Advances in Cryptology – ASIACRYPT 2000. Proceedings, 2000. XII, 630 pages. 2000.

Vol. 1977: B. Roy, E. Okamoto (Eds.), Progress in Cryptology – INDOCRYPT 2000. Proceedings, 2000. X, 295 pages. 2000.

Vol. 1978: B. Schneier (Ed.), Fast Software Encryption. Proceedings, 2000. VIII, 315 pages. 2001.

Vol. 1979: S. Moss, P. Davidsson (Eds.), Multi-Agent-Based Simulation. Proceedings, 2000. VIII, 267 pages. 2001. (Subseries LNAI).

Vol. 1983: K.S. Leung, L.-W. Chan, H. Meng (Eds.), Intelligent Data Engineering and Automated Learning – IDEAL 2000. Proceedings, 2000. XVI, 573 pages. 2000.

Vol. 1984: J. Marks (Ed.), Graph Drawing. Proceedings, 2001. XII, 419 pages. 2001.

Vol. 1987: K.-L. Tan, M.J. Franklin, J. C.-S. Lui (Eds.), Mobile Data Management. Proceedings, 2001. XIII, 289 pages. 2001.

Vol. 1989: M. Ajmone Marsan, A. Bianco (Eds.), Quality of Service in Multiservice IP Networks. Proceedings, 2001. XII, 440 pages. 2001.

Vol. 1991: F. Dignum, C. Sierra (Eds.), Agent Mediated Electronic Commerce. VIII, 241 pages. 2001. (Subseries LNAI).

Vol. 1992: K. Kim (Ed.), Public Key Cryptography. Proceedings, 2001. XI, 423 pages. 2001.

Vol. 1993: E. Zitzler, K. Deb, L. Thiele, C.A.Coello Coello, D. Corne (Eds.), Evolutionary Multi-Criterion Optimization. Proceedings, 2001. XIII, 712 pages. 2001.

Vol. 1995: M. Sloman, J. Lobo, E.C. Lupu (Eds.), Policies for Distributed Systems and Networks. Proceedings, 2001. X, 263 pages. 2001.

Vol. 1998: R. Klette, S. Peleg, G. Sommer (Eds.), Robot Vision. Proceedings, 2001. IX, 285 pages. 2001.

Vol. 2000: R. Wilhelm (Ed.), Informatics: 10 Years Back, 10 Years Ahead. IX, 369 pages. 2001.

Vol. 2003: F. Dignum, U. Cortés (Eds.), Agent Mediated Electronic Commerce III. XII, 193 pages. 2001. (Subseries LNAI).

Vol. 2004: A. Gelbukh (Ed.), Computational Linguistics and Intelligent Text Processing. Proceedings, 2001. XII, 528 pages. 2001.

Vol. 2006: R. Dunke, A. Abran (Eds.), New Approaches in Software Measurement. Proceedings, 2000. VIII, 245 pages. 2001.

Vol. 2009: H. Federrath (Ed.), Designing Privacy Enhancing Technologies. Proceedings, 2000. X, 231 pages. 2001.

Vol. 2010: A. Ferreira, H. Reichel (Eds.), STACS 2001. Proceedings, 2001. XV, 576 pages. 2001.

Vol. 2024: H. Kuchen, K. Ueda (Eds.), Functional and Logic Programming. Proceedings, 2001. X, 391 pages. 2001.

Preface

Since 1993, cryptographic algorithm research has centered around the Fast Software Encryption (FSE) workshop. First held at Cambridge University with 30 attendees, it has grown over the years and has achieved worldwide recognition as a premiere conference. It has been held in Belgium, Israel, France, Italy, and, most recently, New York.

FSE 2000 was the 7th international workshop, held in the United States for the first time. Two hundred attendees gathered at the Hilton New York on Sixth Avenue, to hear 21 papers presented over the course of three days: 10–12 April 2000. These proceedings constitute a collection of the papers presented during those days.

FSE concerns itself with research on classical encryption algorithms and related primitives, such as hash functions. This branch of cryptography has never been more in the public eye. Since 1997, NIST has been shepherding the Advanced Encryption Standard (AES) process, trying to select a replacement algorithm for DES. The first AES conference, held in California the week before Crypto 98, had over 250 attendees. The second conference, held in Rome two days before FSE 99, had just under 200 attendees. The third AES conference was held in conjunction with FSE 2000, during the two days following it, at the same hotel.

It was a great pleasure for me to organize and chair FSE 2000. We received 53 submissions covering the broad spectrum of classical encryption research. Each of those submissions was read by at least three committee members – more in some cases. The committee chose 21 papers to be presented at the workshop. Those papers were distributed to workshop attendees in a preproceedings volume. After the workshop, authors were encouraged to further improve their papers based on comments received. The final result is the proceedings volume you hold in your hand.

To conclude, I would like to thank all the authors who submitted papers to this conference, whether or not your papers were accepted. It is your continued research that makes this field a vibrant and interesting one. I would like to thank the other program committee members: Ross Anderson (Cambridge), Eli Biham (Technion), Don Coppersmith (IBM), Cunsheng Ding (Singapore), Dieter Gollmann (Microsoft), Lars Knudsen (Bergen), James Massey (Lund), Mitsuru Matsui (Mitsubishi), Bart Preneel (K.U.Leuven), and Serge Vaudenay (EPFL). They performed the hard – and too often thankless – task of selecting the program. I'd like to thank my assistant, Beth Friedman, who handled administrative matters for the conference. And I would like to thank the attendees for coming to listen, learn, share ideas, and participate in the community. I believe that FSE represents the most interesting subgenre within cryptography, and that this conference represents the best of what cryptography has to offer.

Enjoy the proceeedings, and I'll see everyone next year in Japan.

August 2000 Bruce Schneier

Table of Contents

General Stream-Cipher Cryptanalysis

AES Cryptanalysis 2

Block-Cipher Cryptanalysis 2

Theoretical Work

Real Time Cryptanalysis of A5/1 on a PC

Alex Biryukov[1], Adi Shamir[1], and David Wagner[2]

[1] Computer Science department, The Weizmann Institute, Rehovot 76100, Israel
[2] Computer Science department, University of California, Berkeley CA 94720, USA.

Abstract. A5/1 is the strong version of the encryption algorithm used by about 130 million GSM customers in Europe to protect the over-the-air privacy of their cellular voice and data communication. The best published attacks against it require between 2^{40} and 2^{45} steps. This level of security makes it vulnerable to hardware-based attacks by large organizations, but not to software-based attacks on multiple targets by hackers.

In this paper we describe new attacks on A5/1, which are based on subtle flaws in the tap structure of the registers, their noninvertible clocking mechanism, and their frequent resets. After a 2^{48} parallelizable data preparation stage (which has to be carried out only once), the actual attacks can be carried out in real time on a single PC.

The first attack requires the output of the A5/1 algorithm during the first two minutes of the conversation, and computes the key in about one second. The second attack requires the output of the A5/1 algorithm during about two seconds of the conversation, and computes the key in several minutes. The two attacks are related, but use different types of time-memory tradeoffs. The attacks were verified with actual implementations, except for the preprocessing stage which was extensively sampled rather than completely executed.

REMARK: We based our attack on the version of the algorithm which was derived by reverse engineering an actual GSM telephone and published at `http://www.scard.org`. We would like to thank the GSM organization for graciously confirming to us the correctness of this unofficial description. In addition, we would like to stress that this paper considers the narrow issue of the cryptographic strength of A5/1, and not the broader issue of the practical security of fielded GSM systems, about which we make no claims.

1 Introduction

The over-the-air privacy of GSM telephone conversations is protected by the A5 stream cipher. This algorithm has two main variants: The stronger A5/1 version is used by about 130 million customers in Europe, while the weaker A5/2 version is used by another 100 million customers in other markets. The approximate design of A5/1 was leaked in 1994, and the exact design of both A5/1 and A5/2 was reverse engineered by Briceno from an actual GSM telephone in 1999 (see [3]).

B. Schneier (Ed.): FSE 2000, LNCS 1978, pp. 1–18, 2001.

In this paper we develop two new cryptanalytic attacks on A5/1, in which a single PC can extract the conversation key in real time from a small amount of generated output. The attacks are related, but each one of them optimizes a different parameter: The first attack (called **the biased birthday attack**) requires two minutes of data and one second of processing time, whereas the second attack (called the **the random subgraph attack**) requires two seconds of data and several minutes of processing time. There are many possible choices of tradeoff parameters in these attacks, and three of them are summarized in Table 1.

Table 1. Three possible tradeoff points in the attacks on A5/1.

Attack Type	Preprocessing steps	Available data	Number of 73GB disks	Attack time
Biased Birthday attack (1)	2^{42}	2 minutes	4	1 second
Biased Birthday attack (2)	2^{48}	2 minutes	2	1 second
Random Subgraph attack	2^{48}	2 seconds	4	minutes

Many of the ideas in these two new attacks are applicable to other stream ciphers as well, and define new quantifiable measures of security.

The paper is organized in the following way: Section 2 contains a full description of the A5/1 algorithm. Previous attacks on A5/1 are surveyed in Section 3, and an informal description of the new attacks is contained in Section 4. Finally, Section 5 contains various implementation details and an analysis of the expected success rate of the attacks, based on large scale sampling with actual implementations.

2 Description of the A5/1 Stream Cipher

A GSM conversation is sent as a sequence of frames every 4.6 millisecond. Each frame contains 114 bits representing the digitized A to B communication, and 114 bits representing the digitized B to A communication. Each conversation can be encrypted by a new session key K. For each frame, K is mixed with a publicly known frame counter F_n, and the result serves as the initial state of a generator which produces 228 pseudo random bits. These bits are XOR'ed by the two parties with the 114+114 bits of the plaintext to produce the 114+114 bits of the ciphertext.

A5/1 is built from three short linear feedback shift registers (LFSR) of lengths 19, 22, and 23 bits, which are denoted by $R1, R2$ and $R3$ respectively. The rightmost bit in each register is labelled as bit zero. The taps of $R1$ are at bit positions 13,16,17,18; the taps of $R2$ are at bit positions 20,21; and the taps of $R3$ are at bit positions 7, 20,21,22 (see Figure 1). When a register is clocked,

its taps are XORed together, and the result is stored in the rightmost bit of the left-shifted register. The three registers are maximal length LFSR's with periods $2^{19}-1$, $2^{22}-1$, and $2^{23}-1$, respectively. They are clocked in a stop/go fashion using the following majority rule: Each register has a single "clocking" tap (bit 8 for $R1$, bit 10 for $R2$, and bit 10 for for $R3$); 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 actually clocked. Note that at each step either two or three registers are clocked, and that each register moves with probability 3/4 and stops with probability 1/4.

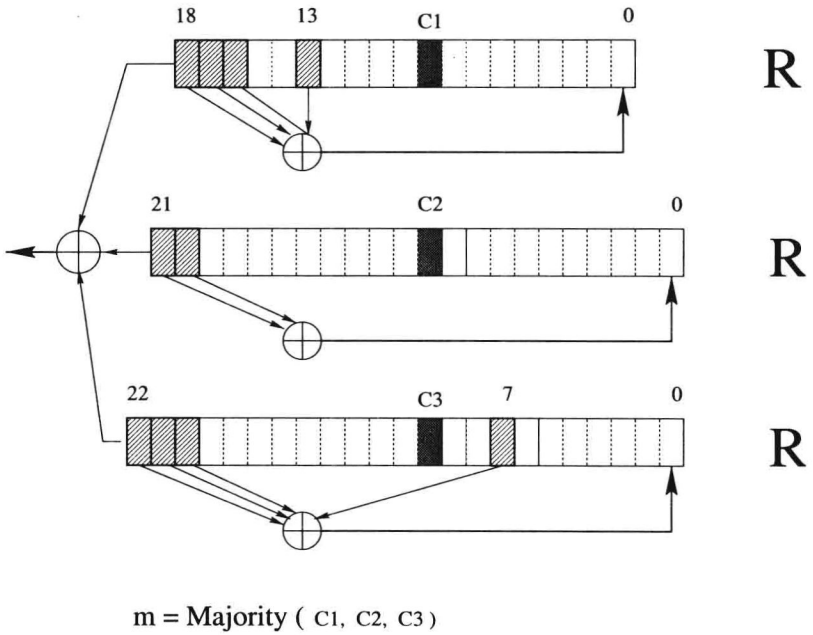

Fig. 1. The A5/1 stream cipher.

The process of generating pseudo random bits from the session key K and the frame counter F_n is carried out in four steps:

- The three registers are zeroed, and then clocked for 64 cycles (ignoring the stop/go clock control). During this period each bit of K (from lsb to msb) is XOR'ed in parallel into the lsb's of the three registers.
- The three registers are clocked for 22 additional cycles (ignoring the stop/go clock control). During this period the successive bits of F_n (from lsb to msb) are again XOR'ed in parallel into the lsb's of the three registers. The contents of the three registers at the end of this step is called the **initial state** of the frame.

- The three registers are clocked for 100 additional clock cycles with the stop/go clock control but without producing any outputs.
- The three registers are clocked for 228 additional clock cycles with the stop/go clock control in order to produce the 228 output bits. At each clock cycle, one output bit is produced as the XOR of the msb's of the three registers.

3 Previous Attacks

The attacker is assumed to know some pseudo random bits generated by A5/1 in some of the frames. This is the standard assumption in the cryptanalysis of stream ciphers, and we do not consider in this paper the crucial issue of how one can obtain these bits in fielded GSM systems. For the sake of simplicity, we assume that the attacker has complete knowledge of the outputs of the A5/1 algorithm during some initial period of the conversation, and his goal is to find the key in order to decrypt the remaining part of the conversation. Since GSM telephones send a new frame every 4.6 milliseconds, each second of the conversation contains about 2^8 frames.

At the rump session of Crypto 99, Ian Goldberg and David Wagner announced an attack on A5/2 which requires very few pseudo random bits and just $O(2^{16})$ steps. This demonstrated that the "export version" A5/2 is totally insecure.

The security of the A5/1 encryption algorithm was analyzed in several papers. Some of them are based on the early imprecise description of this algorithm, and thus their details have to be slightly modified. The known attacks can be summarized in the following way:

- Briceno[3] found out that in all the deployed versions of the A5/1 algorithm, the 10 least significant of the 64 key bits were always set to zero. The complexity of exhaustive search is thus reduced to $O(2^{54})$. [1]
- Anderson and Roe[1] proposed an attack based on guessing the 41 bits in the shorter R_1 and R_2 registers, and deriving the 23 bits of the longer R_3 register from the output. However, they occasionally have to guess additional bits to determine the majority-based clocking sequence, and thus the total complexity of the attack is about $O(2^{45})$. Assuming that a standard PC can test ten million guesses per second, this attack needs more than a month to find one key.
- Golic[4] described an improved attack which requires $O(2^{40})$ steps. However, each operation in this attack is much more complicated, since it is based on the solution of a system of linear equations. In practice, this algorithm is not likely to be faster than the previous attack on a PC.

[1] Our new attack is not based on this assumption, and is thus applicable to A5/1 implementations with full 64 bit keys. It is an interesting open problem whether we can speed it up by assuming that 10 key bits are zero.

- Golic[4] describes a general time-memory tradeoff attack on stream ciphers (which was independently discovered by Babbage [2] two years earlier), and concludes that it is possible to find the A5/1 key in 2^{22} probes into random locations in a precomputed table with 2^{42} 128 bit entries. Since such a table requires a 64 terabyte hard disk, the space requirement is unrealistic. Alternatively, it is possible to reduce the space requirement to 862 gigabytes, but then the number of probes increases to $O(2^{28})$. Since random access to the fastest commercially available PC disks requires about 6 milliseconds, the total probing time is almost three weeks. In addition, this tradeoff point can only be used to attack GSM phone conversations which last more than 3 hours, which again makes it unrealistic.

4 Informal Description of the New Attacks

We start with an executive summary of the key ideas of the two attacks. More technical descriptions of the various steps will be provided in the next section.

Key idea 1: Use the Golic time-memory tradeoff. The starting point for the new attacks is the time-memory tradeoff described in Golic[3], which is applicable to any cryptosystem with a relatively small number of internal states. A5/1 has this weakness, since it has $n = 2^{64}$ states defined by the 19+22+23 = 64 bits in its three shift registers. The basic idea of the Golic time-memory tradeoff is to keep a large set A of precomputed states on a hard disk, and to consider the large set B of states through which the algorithm progresses during the actual generation of output bits. Any intersection between A and B will enable us to identify an actual state of the algorithm from stored information.

Key idea 2: Identify states by prefixes of their output sequences. Each state defines an infinite sequence of output bits produced when we start clocking the algorithm from that state. In the other direction, states are usually uniquely defined by the first $log(n)$ bits in their output sequences, and thus we can look for equality between unknown states by comparing such prefixes of their output sequences. During precomputation, pick a subset A of states, compute their output prefixes, and store the (prefix, state) pairs sorted into increasing prefix values. Given actual outputs of the A5/1 algorithm, extract all their (partially overlapping) prefixes, and define B as the set of their corresponding (unknown) states. Searching for common states in A and B can be efficiently done by probing the sorted data A on the hard disk with prefix queries from B.

Key idea 3: A5/1 can be efficiently inverted. As observed by Golic, the state transition function of A5/1 is not uniquely invertible: The majority clock control rule implies that up to 4 states can converge to a common state in one clock cycle, and some states have no predecessors. We can run A5/1 backwards by exploring the tree of possible predecessor states, and backtracking from dead ends. The average number of predecessors of each node is 1, and thus the expected number of vertices in the first k levels of each tree grows only linearly in k (see[3]). As a result, if we find a common state in the disk and data,

we can obtain a small number of candidates for the initial state of the frame. The weakness we exploit here is that due to the frequent reinitializations there is a very short distance from intermediate states to initial states.

Key idea 4: The key can be extracted from the initial state of any frame. Here we exploit the weakness of the A5/1 key setup routine. Assume that we know the state of A5/1 immediately after the key and frame counter were used, and before the 100 mixing steps. By running backwards, we can eliminate the effect of the known frame counter in a unique way, and obtain 64 linear combinations of the 64 key bits. Since the tree exploration may suggest several keys, we can choose the correct one by mixing it with the next frame counter, running A5/1 forward for more than 100 steps, and comparing the results with the actual data in the next frame.

Key idea 5: The Golic attack on A5/1 is marginally impractical. By the well known birthday paradox, A and B are likely to have a common state when their sizes a and b satisfy $a * b \approx n$. We would like a to be bounded by the size of commercially available PC hard disks, and b to be bounded by the number of overlapping prefixes in a typical GSM telephone conversation. Reasonable bounds on these values (justified later in this paper) are $a \approx 2^{35}$ and $b \approx 2^{22}$. Their product is 2^{57}, which is about 100 times smaller than $n = 2^{64}$. To make the intersection likely, we either have to increase the storage requirement from 150 gigabytes to 15 terabytes, or to increase the length of the conversation from two minutes to three hours. Neither approach seems to be practical, but the gap is not huge and a relatively modest improvement by two orders of magnitude is all we need to make it practical.

Key idea 6: Use special states. An important consideration in implementing time-memory tradeoff attacks is that access to disk is about a million times slower than a computational step, and thus it is crucial to minimize the number of times we look for data on the hard disk. An old idea due to Ron Rivest is to keep on the disk only special states which are guaranteed to produce output bits starting with a particular pattern α of length k, and to access the disk only when we encounter such a prefix in the data. This reduces the number b of disk probes by a factor of about 2^k. The number of points a we have to memorize remains unchanged, since in the formula $a * b \approx n$ both b and n are reduced by the same factor 2^k. The downside is that we have to work 2^k times harder during the preprocessing stage, since only 2^{-k} of the random states we try produce outputs with such a k bit prefix. If we try to reduce the number of disk access steps in the time memory attack on A5/1 from 2^{22} to 2^6, we have to increase the preprocessing time by a factor of about 64,000, which makes it impractically long.

Key idea 7: Special states can be efficiently sampled in A5/1. A major weakness of A5/1 which we exploit in both attacks is that it is easy to generate all the states which produce output sequences that start with a particular k-bit pattern α with $k = 16$ without trying and discarding other states. This is due to a poor choice of the clocking taps, which makes the register bits that affect the clock control and the register bits that affect the output unrelated

for about 16 clock cycles, so we can choose them independently. This easy access to special states does not happen in good block ciphers, but can happen in stream ciphers due to their simpler transition functions. In fact, the maximal value of k for which special states can be sampled without trial and error can serve as a new security measure for stream ciphers, which we call its **sampling resistance**. As demonstrated in this paper, high values of k can have a big impact on the efficiency of time-memory tradeoff attacks on such cryptosystems.

Key idea 8: Use biased birthday attacks. The main idea of the first attack is to consider sets A and B which are not chosen with uniform probability distribution among all the possible states. Assume that each state s is chosen for A with probability $P_A(s)$, and is chosen for B with probability $P_B(s)$. If the means of these probability distributions are a/n and b/n, respectively, then the expected size of A is a, and the expected size of B is b.

The birthday threshold happens when $\sum_s P_A(s)P_B(s) \approx 1$. For independent uniform distributions, this evaluates to the standard condition $a*b \approx n$. However, in the new attack we choose states for the disk and states in the data with two non-uniform probability distributions which have strong positive correlation. This makes our time memory tradeoff much more efficient than the one used by Golic. This is made possible by the fact that in A5/1, the initial state of each new frame is rerandomized very frequently with different frame counters.

Key idea 9: Use Hellman's time-memory tradeoff on a subgraph of special states. The main idea of the second attack (called the random subgraph attack) is to make most of the special states accessible by simple computations from the subset of special states which are actually stored in the hard disk. The first occurrence of a special state in the data is likely to happen in the first two seconds of the conversation, and this single occurrence suffices in order to locate a related special state in the disk even though we are well below the threshold of either the normal or the biased birthday attack. The attack is based on a new function f which maps one special state into another special state in an easily computable way. This f can be viewed as a random function over the subspace of 2^{48} special states, and thus we can use Hellman's time-memory tradeoff[4] in order to invert it efficiently. The inverse function enables us to compute special states from output prefixes even when they are not actually stored on the hard disk, with various combinations of time T and memory M satisfying $M\sqrt{T} = 2^{48}$. If we choose $M = 2^{36}$, we get $T = 2^{24}$, and thus we can carry out the attack in a few minutes, after a 2^{48} preprocessing stage which explores the structure of this function f.

Key idea 10: A5/1 is very efficient on a PC. The A5/1 algorithm was designed to be efficient in hardware, and its straightforward software implementation is quite slow. To execute the preprocessing stage, we have to run it on a distributed network of PC's up to 2^{48} times, and thus we need an extremely efficient way to compute the effect of one clock cycle on the three registers.

We exploit the following weakness in the design of A5/1: Each one of the three shift registers is so small that we can precompute all its possible states, and keep them in RAM as three cyclic arrays, where successive locations in each