Tehnici de criptare în comunicațiile mobile. Algoritmul ZUC.

Encryption in Mobile Communications. ZUC Algorithm.

Laura IANCU[1], Ioan BACIVAROV[2]
[1] Faculty of ETTI, University POLITEHNICA of Bucharest, Romania
laura_yln@yahoo.com
[2] EUROQUALROM Laboratory, Faculty of ETTI, University POLITEHNICA of Bucharest, Romania
bacivaro@euroqual.pub.ro

Abstract

ZUC is a data stream cipher, easy to implement, one of the fastest algorithms to encrypt messages in mobile communications. Because of the key and initialization vector large size (128-bits), ZUC provides high security and is enough resistant to many types of attacks: Weak Key Attacks, Guess-and-Determine Attacks, Algebraic Attacks, Timing Attacks, but not enough robust to withstand the DPA (Differential Power Analysis) type attack.

This article makes an analysis of ZUC algorithm and presents the encryption efficiency, and its vulnerabilities; also it is made a comparison with other algorithms used in telecommunications (SNOW 3G, Kasumi, DES/3DES and AES).

Index terms: encryption, ZUC, algorithm, stream cipher, security

1. Introducere

Până în vremurile moderne, termenul *criptografie* se referea aproape exclusiv la criptare, procesul de conversie a informației obișnuite (textul în clar) într-un text neinteligibil (textul cifrat). Decriptarea este procesul invers, de trecere de la textul cifrat, neinteligibil, la textul clar. Un *cifru* este o pereche de algoritmi care efectuează atât această criptare, cât și decriptarea. Modul de operare detaliat al unui cifru este controlat de algoritm și de o cheie. Această cheie este un parametru secret (în mod ideal, cunoscut doar celor care comunică) pentru contextul unui anume schimb de mesaje. Cheile sunt importante, iar cifrurile fără chei variabile sunt simplu de spart și deci mai puțin utile. De-a lungul istoriei, cifrurile erau adesea folosite direct pentru criptare și decriptare, fără proceduri adiționale, cum ar fi autentificarea sau testele de integritate.

În utilizarea populară, termenul "cod" este adesea folosit cu sensul de orice metodă de criptare sau de ascundere a înțelesului. Totuși, în criptografie cuvântul cod are un înțeles mai restrâns; acela de înlocuire a unei unități de text clar (un cuvânt sau o frază) cu un cuvânt

codat (de exemplu, plăcintă cu mere înlocuiește atac în zori). Codurile mai sunt folosite în criptografie doar uneori, pentru anumite lucruri cum ar fi desemnarea unităților (de exemplu, "zborul Bronco" sau Operațiunea Overlord) întrucât cifrurile alese corect sunt mai practice, mai sigure și în același timp mai bine adaptate calculatoarelor decât cele mai bune coduri.

Studiul modern al cifrurilor cu chei simetrice se leagă mai ales de studiul cifrurilor pe blocuri și al cifrurilor pe flux și al aplicațiilor acestora. Un *cifru pe blocuri* este, într-un fel, o formă modernă de cifru polialfabetic Alberti: cifrurile pe blocuri iau la intrare un bloc de text clar și o cheie, și produc la ieșire un bloc de text cifrat de aceeași dimensiune. Deoarece mesajele sunt aproape mereu mai lungi decât un singur bloc, este necesară o metodă de unire a blocurilor succesive. S-au dezvoltat câteva astfel de metode, unele cu securitate superioară într-un aspect sau altul decât alte cifruri. Acestea se numesc moduri de operare și trebuie luate în calcul cu grijă la folosirea unui cifru pe blocuri într-un cripto-sistem.

DES (*Data Encryption Standard*) și AES (*Advanced Encryption Standard*) sunt cifruri pe blocuri considerate standarde de criptografie de către Guvernul american (deși DES a fost în cele din urmă retras după adoptarea AES). În ciuda decăderii ca standard oficial, DES (mai ales în varianta triple-DES, mult mai sigură) rămâne încă popular; el este folosit într-o gamă largă de aplicații, de la criptarea ATM la securitatea e-mail-urilor și accesul la distanță securizat. Multe alte cifruri pe blocuri au fost elaborate și lansate, cu diverse calități, însă s-au găsit metode de spargere pentru o mare parte dintre acestea.

Cifrurile pe flux de date (*stream cipher*), cum este și cifrul ZUC prezentat in acest articol, în contrast cu cele pe blocuri, creează un flux arbitrar de material-cheie, care este combinat cu textul clar, bit cu bit sau caracter cu caracter. Într-un cifru pe flux de date, fluxul de ieșire este creat pe baza unei stări interne care se modifică pe parcursul operării cifrului. Această schimbare de stare este controlată de cheie, și, la unele cifruri, și de fluxul de text clar. RC4 este un exemplu binecunoscut de cifru pe flux.

Funcțiile hash criptografice (adesea numite *message digest*) nu folosesc neapărat chei, sunt o clasă importantă de algoritmi criptografici. Acestea primesc date de intrare (adesea un întreg mesaj) și produc un hash scurt, de lungime fixă, sub forma unei funcții neinversabile. Pentru hash-urile bune, coliziunile (două texte clare diferite care produc același hash) sunt extrem de dificil de găsit. În prezent este utilizată în protejarea unei mari varietăți de sisteme, precum e-commerce, rețele de telefonie mobilă și ATM-urile băncilor.

Criptarea poate fi folosită pentru a asigura discreția și/sau intimitatea, dar și alte tehnici sunt necesare pentru a face comunicațiile sigure, în mod particular verificarea integrității și autenticității unui mesaj; de exemplu, un cod de autentificare a mesajelor sau semnături digitale. Altă considerație este protecția împotriva analizei traficului.

Criptarea sau ascunderea codului de software este folosită în protecția copierii de software împotriva ingineriei inverse, analiza aplicațiilor neautorizate, crack-uri și pirateria software.

GSM (Global System for Mobile Communications) este un standard dezvoltat de către ETSI (European Telecommunications Standards Institute) pentru a descrie tehnologii pentru a 2 a generație (2G) de rețele celulare digitale. Inițial dezvoltat ca un înlocuitor al primei rețele celulare analogice, standardul GSM reprezenta o rețea bazată pe comutație de circuite care deservea servicii de voce full duplex. Standardul a fost dezvoltat în continuare pentru a putea face transfer de date de tip CS (*Circuits Switching*, comutație de circuite), ca mai târziu să permită transfer de date de tip PS (*Packet Switching*, comutație de pachete).

Rețeaua GSM oferă următoarele funcții de securitate:

- confidențialitatea IMSI;

- verificarea identității unui abonat pentru a proteja accesul la servicii;
- confidențialitatea (secretizarea) informațiilor utilizatorului;
- confidențialitatea informațiilor de semnalizare.

Sistemul GSM prevede controale de securitate. Operatorul de sistem vrea să se asigure că abonatul care solicită servicii este valid (autentificat). Pe de altă parte, abonatul dorește să aibă acces la servicii fără a-i fi compromisă confidențialitatea datelor personale.

Autentificarea este procesul prin care sunt schimbate informații între device-urile de comunicare (de obicei telefoane mobile) și rețeaua de comunicații, permițând operatorului de rețea să confirme valabilitatea identității. Acest proces previne folosirea frauduloasă a telefoanelor mobile.

Ce-a de-a treia generație (3G) de rețele digitale celulare a fost dezvoltată de către Third Generation Partnership Project (3GPP) și denumită Universal Mobile Telecommunications System (UMTS). Evoluția UMTS a adus un nou sistem de radio acces numit LTE (*Long Term Evolution*) și o nouă rețea core denumită SAE (*System Architecture Evolution*) și a introdus doi noi algoritmi de criptare și integritate a datelor: 128-EEA3 și 128-EIA3. Setul de algoritmi rezultat se bazează pe o metodă de generare a șirurilor numită **ZUC**, după Zu Chongzhi, om de știință chinez. Algoritmii au fost creați la centrul de cercetare Data Assurance and Communication Security (DACAS) al Academiei Chineze de Științe.

După adoptarea și standardizarea de către 3GPP (3rd Generation Partnership Project), 128-EIA3 va reprezenta al 3-lea algoritm de integritate din cadrul LTE. Algoritmul 128-EIA3 este o funcție hash universală bazată pe familia de funcții hash Carter-Wegman. Diferența constă în felul în care este generat atributul de masking; la 128-EIA3, acesta este dependent de lungimea mesajului și are la bază codorul ZUC. 128-EIA3 calculează un MAC de 32 de biți al unui mesaj de intrare de lungime cuprinsă între 1 si 20 000 de biți cu ajutorul unei chei de integritate (Integrity Key, IK) și al unui vector de inițializare (Initialisation Vector, IV) de 128 de biți.

Metoda de generare a șirului de criptare ZUC joacă un rol important in a 4-a generație de comunicații mobile, fiind inclusă în 3GPP LTE-Advanced. După un program de evaluare de lungă durată, algoritmul ZUC a fost declarat îndeajuns de robust încât să reziste multor criptanalize, dar nu îndeajuns de rezistent pentru DPA, unul dintre cele mai puternice amenințări ale SCAs (Side Channel Analysis). Până în prezent, pentru a combate DPA se folosesc metodele DES și AES, doi algoritmi de criptare care lucrează cu blocuri, nu cu un flux continuu precum ZUC.

2. Algoritmul ZUC

2.1. Structura generală a algoritmului

Nucleul algoritmului ZUC1 implementează cifrul de generare ZUC în acord cu Algoritmii de Integritate și Confidențialitate 3GPP 128-EEA3 și 128-EIA3 versiunea 1.6. Acesta produce fluxul care constă în blocuri de 32 de biți, folosind o cheie de 128 de biți.

Multiple configurații ale nucleului ZUC1 sunt disponibile. Numărul "32" din sintagma "ZUC1-32" indică troughput-ul în biți per clock, așadar versiunea ZUC1-32 este de 4 ori mai rapidă decât ZUC1-8. Versiunea mai nouă, E3, este disponibilă, aceasta oferind suport pentru algoritmii de confidențialitate și integritate EEA3 și EIA3. Nucleul ZUC1-2-E3 este foarte mic ca dimensiune (12.000 de porți).

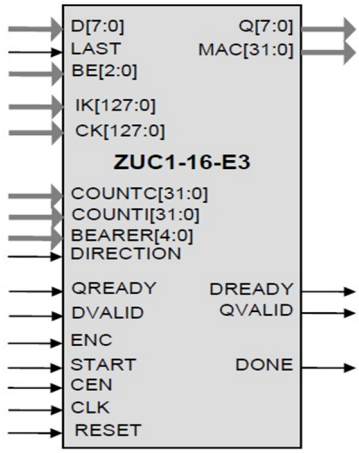

Fig. 1. Configurația ZUC1-16-E3

Descrierea pinilor:

Nume	**Tip**	**Descriere**
CLK	Input	Semnal de clock
RESET	Input	Semnal de reset
CEN	Input	Semnal sincron de enable. Când LOW, nucleul ignoră toate intrările și toate ieșirile trebuie ignorate
START	Input	Când HIGH, o operație criptografică este lansată
ENC	Input	HIGH corespunde operației de criptare, LOW corespunde decriptării. De obicei este fixat în HIGH sau LOW.
DREADY	Output	Când HIGH, nucleul așteaptă date de la magistrala D
QVALID	Output	Când HIGH, datele de ieșire sunt valide pe magistrala Q
QREADY	Input	Când HIGH, circuitul extern este pregătit să accepte datele pe magistrala Q
DVALID	Input	Când HIGH, datele sunt valide pe magistrala D
DONE	Output	Când HIGH, procesarea de mesaje este completă, datele de MAC sunt valide
CK[127:0]	Input	Cheie de criptare
IK[127:0]	Input	Cheie de integritate
COUNTC[31:0]	Input	Counter al criptării pachetelor
COUNT[31:0]	Input	Counter al integrității pachetelor
BEARER[4:0]	Input	Informația despre purtătoare
DIRECTION	Input	Direcția link-ului. HIGH pentru uplink, LOW pentru downlink. De obicei este fixat în HIGH sau LOW
Q[7:0]	Output	Date de ieșire
D[7:0]	Input	Date de intrare
LAST	Input	Atunci când HIGH, indică ultimul cuvânt al datelor de intrare pe magistrala D
BE[2:0]	Input	Numărul de biți valabili în ultimul cuvânt -1. Eșantionat de către nucleu în același timp în care LAST eșantionează HIGH.
MAC[31:0]	Output	Valori MAC calculate

2.2. Definirea algoritmului ZUC

În acest capitol se va prezenta pe scurt ultima versiune a algoritmului ZUC. Noul algoritm de criptare ZUC folosește o cheie secretă de 128 de biți și un vector de inițializare de 128 de biți ca intrare, având ca ieșire un șir de caractere pe 32 de biți, care este folosit pentru a cripta sau decripta datele. Conform specificațiilor oficiale privind algoritmul ZUC, acesta este compus din 3 niveluri logice. Nivelul superior este un registru de deplasare cu reacție liniară (LFSR - *Linear Feedback Shift Register*) compus din 16 celule, nivelul de mijloc produce operația de reorganizare a biților (BR - *Bit Reorganization*), iar nivelul inferior este o funcție neliniară F.

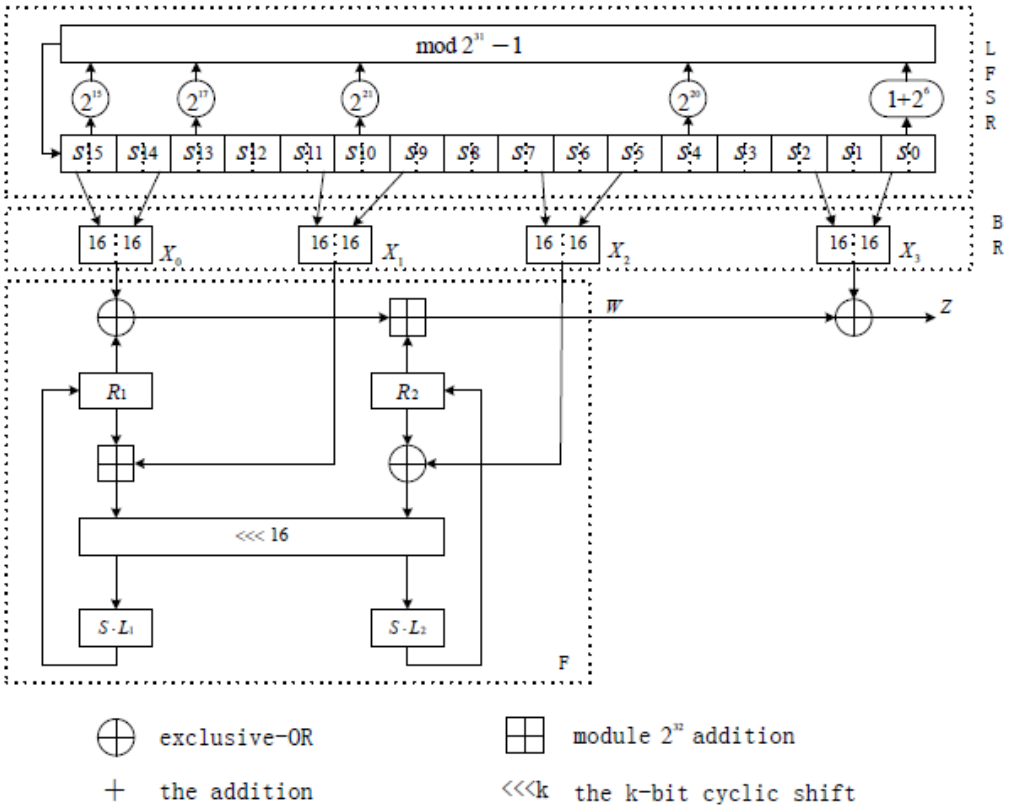

Fig. 2. Structura algoritmului ZUC

2.2.1. Registrul de deplasare cu reacție liniară LFSR

Registrul LFSR are 16 celule pe 31 de biți (S_0, S_1, …, S_{15}). Fiecare celulă de registru S_i ($0 \le i \le 15$) este restricționată în a lua valori din următorul set: {1, 2, 3, ..., 2^{31}-1}. Registrul LFSR are două moduri de operare: modul de inițializare și modul de execuție. Modul de inițializare funcționează după Algoritmul 1 prezentat în continuare.

Algoritmul 1.

LFSRWithInitialisationMode (u){

1. $v = 2^{15}s_{15} + 2^{17}s_{13} + 2^{21}s_{10} + 2^{20}s_4 + (1 + 2^8)mod(2^{31} - 1);$
2. $s_{16} = (u + v)mod(2^{31} - 1);$
3. $If\ s_{16} = 0, then\ set\ s_{16} = 2^{31} - 1;$
4. $(s_1, s_2, \dots, s_{15}, s_{16}) \rightarrow (s_0, s_1, \dots, s_{14}, s_{15}).$

}

În modul de execuție, registrul LFSR nu primește intrări și funcționează după Algoritmul 2 prezentat mai jos.

Algoritmul 2.
LFSRWithWorkMode(){
1. $v = 2^{15}s_{15} + 2^{17}s_{13} + 2^{21}s_{10} + 2^{20}s_4 + (1 + 2^8) mod (2^{31} - 1);$
2. $If\ s_{16} = 0, then\ set\ s_{16} = 2^{31} - 1;$
3. $(s_1, s_2, \dots, s_{15}, s_{16}) \rightarrow (s_0, s_1, \dots, s_{14}, s_{15}).$

}

2.2.2. Reorganizarea biților

Nivelul de mijloc este reprezentat prin procedura de reorganizare a biților.

Extrage 128 de biți din celulele LFSR și formează 4 cuvinte de câte 32 de biți fiecare, unde primele 3 cuvinte vor fi trimise următorului nivel, funcția neliniară F, iar ultimul cuvânt va fi folosit în inițializarea cifrului.

Presupunând că s_0, s_2, s_5, s_7, s_9, s_{11}, s_{14} și s_{15} sunt 8 celule ale LFSR, atunci reorganizarea biților formează 4 cuvinte de 32 de biți X_0, X_1, X_2, X_3 din celulele anterioare, după Algoritmul 3 prezentat în continuare.

Algoritmul 3.
BitReorganization(){
1. $X_0 = s_{15H} \parallel s_{14L};$
2. $X_1 = s_{11L} \parallel s_{9H};$
3. $X_2 = s_{7L} \parallel s_{5H};$
4. $X_3 = s_{2L} \parallel s_{0H}.$

}

2.2.3. Funcția neliniară F

Există două celule de memorie pe 32 de biți, R_1 și R_2 în funcția neliniară F. Intrările funcției sunt X_0, X_1, X_2, primele 3 cuvinte de ieșire ale procedurii BR. Ieșirea funcției F este un cuvânt pe 32 de biți. Prezentarea detaliată a funcției F este descrisă în Algoritmul 4, în care S este o matrice 32x32.

Algoritmul 4.
F(X_0, X_1, X_2){
1. $W = (X_0 \oplus X_1 \boxplus) R_{21};$
2. $W_1 = R_1 \boxplus X_1;$
3. $W_2 = R_2 \oplus X_2;$
4. $R_1 = S\big(L_1(W_{1L} \parallel W_{2H})\big);$
5. $R_2 = S\big(L_2(W_{2L} \parallel W_{1H})\big).$

}

Atât L_1, cât și L_2, sunt transformate liniare din cuvinte de 32 de biți și sunt definite astfel:

$$L_1(X) = X \oplus (X <<<_{32} 2) \oplus (X <<<_{32} 10) \oplus (X <<<_{32} 18) \oplus (X <<<_{32} 24)$$
$$L_2(X) = X \oplus (X <<<_{32} 8) \oplus (X <<<_{32} 14) \oplus (X <<<_{32} 22) \oplus (X <<<_{32} 30)$$

2.2.4. Procedura Key Loading

Procedura Key Loading va extinde cheia și vectorul de inițializare în 16 numere întregi pe 31 de biți, constituind faza inițială a LFSR.

Fie cheia de inițializare k pe 128 de biți și vectorul de inițializare iv pe 128 de biți să fie $k=k_0 \parallel k_1 \parallel k_2 \parallel \dots \parallel k_{15}$ și $iv = iv_0 \parallel iv_1 \parallel iv_2 \parallel \dots \parallel iv_{15}$, unde k_i și iv_i sunt încărcate în celulele S_i după formula $s_i=k_i \parallel d_i \parallel iv_i$, unde d_i este o constantă dată.

2.2.5. Executarea algoritmului ZUC

Executarea algoritmului ZUC este compusă din 2 stadii: stadiul de inițializare și cel de executare.

În timpul stadiului de inițializare, algoritmul rulează următoarele operații de 32 de ori:
1. BitReorganization();
2. W = F(X_0, X_1, X_2);
3. LFSRWithInitialisationMode(w>>1).

După faza de inițializare, algoritmul de criptare trece în faza de execuție. La începutul acesteia, algoritmul execută următoarele operații o singură dată și aruncă ieșirea W a funcției F:
1. BitReorganization();
2. F(X_0, X_1, X_2);
3. LFSRWithInitialisationMode().

În continuare algoritmul intră în faza de generare a cheii. Următoarele operații sunt executate o singură dată și un cuvânt Z pe 32 de biți este produs ca ieșire:

1. BitReorganization();
2. $Z = F(X_0, X_1, X_2) \oplus X3$;
3. LFSRWithInitialisationMode().

2.3. Implementarea hardware

Structura implementării hardware a algoritmului ZUC este ilustrată în figura anterioară.

Este în principiu constituită din următoarele părți (a se vedea figura 3):

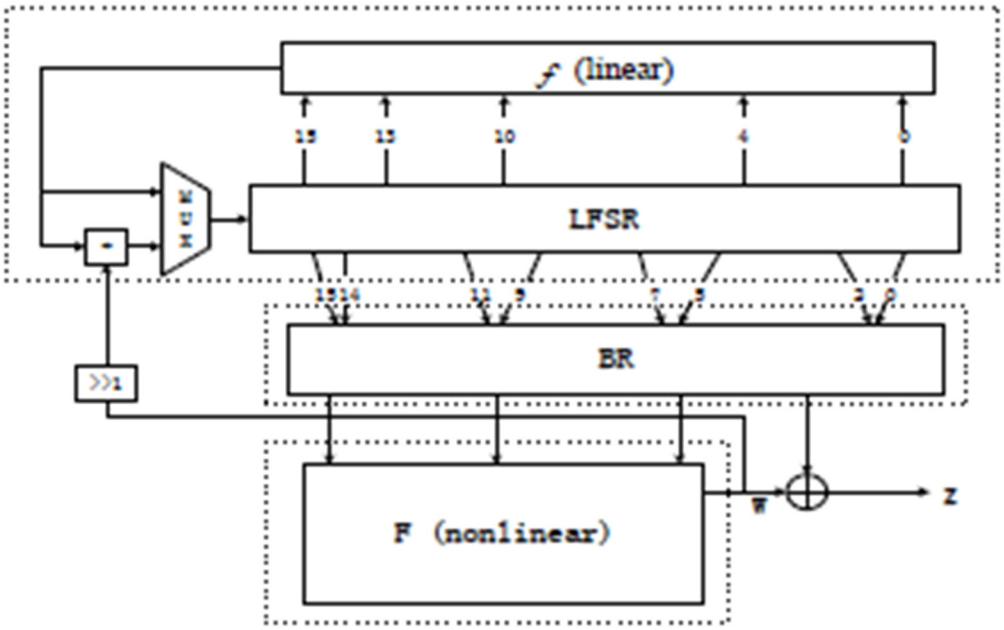

Fig. 3. Implementarea hardware a algoritmului ZUC

- Un registru LFSR din 16 celule s_0, s_1, …, s_{15};
- Un bloc logic combinațional BR care implementează operația de reorganizare a biților;
- Un bloc logic combinațional F care implementează funcția neliniară F;
- Un bloc logic combinațional f care implementează operația mod(2^1-1);
- Porți logice XOR și alte operații logice.

2.4. Algoritmii de confidențialitate și criptare 128-EIA3 și 128-EEA3

128-EIA3 este vulnerabilă la atacurile de tip *birthday forgery*. Aceste atacuri au la bază probabilitatea ca într-un set de oameni aleși aleator, o anumită pereche formată din acești oameni va avea aceeași zi de naștere. Într-un grup de cel puțin 23 de oameni este o probabilitate de peste 50% ca 2 dintre aceștia să fie născuți în aceeași zi. Probabilitatea atinge 100% într-un grup de 366 de oameni.

128-EIA3 poate fi împărțit în 2 componente, funcția universală H și operația XOR între rezultatul funcției hash și atributul de masking. Să considerăm perechea de mesaje (x, x') unde x și x' sunt mesaje distincte de aceeași lungime, cu:

$$h(x) = g(H_t)$$
$$h(x') = g(H'_t)$$

H_t este rezultatul funcției hash, iar g este XOR-ul funcției H cu atributul de masking.

Coliziunea h(x) = h(x') poate fi cauzată de 2 motive:

- dacă $H_t = H'_t$ (și atunci este denumită coliziune internă), sau
- dacă $H_t <> H'_t$ dar $g(H_t) = g(H'_t)$ (și este denumită coliziune externă).

În contextul algoritmului 128-EIA3, coliziunea internă se referă la coliziunea în interiorul funcției hash , iar coliziunea externă se referă la coliziunea datorată operației de XOR.

Standardul 128-EIA3 este susceptibil la coliziunile interne ale funcției sale universale hash pentru mesaje distincte de aceeași lungime și cu aceleași cheie și vector de inițializare IK, IV. Coliziunea internă a funcției H provoacă și coliziune externă pentru aceeași IK și IV. Pentru coliziune internă la 128-EIA3 sunt necesare 216 perechi de mesaje și un mesaj ales pentru a porni o verificare de mesaj fals cu probabilitatea de succes 1.

În cazul detectării unei coliziuni a perechii (x, x'), 128-EIA3 va genera același MAC pentru h(x||y) = h(x'||y) pentru orice bloc *y* cu aceleași IK și IV. Probabilitatea de a găsi coliziuni interne în 128-EIA3 crește odată cu înmulțirea mesajelor MAC. Acest lucru se poate vedea în tabelul de mai jos:

Tabel 1. Statistici coliziuni interne

Nr. crt.	**Numărul perechilor mesaj-MAC cunoscute**	**Numărul perechilor cu coliziuni identificate**
1	2^{16}	1
2	2^{18}	7
3	2^{20}	136
4	2^{22}	2010

Coliziunea externă se obține în urma operației de XOR între valoarea de masking și rezultatul funcției hash H. EIA3 este vulnerabil la acest tip de coliziune pentru mesaje diferite de aceeași lungime, dar cu IK și IV diferiți.

Cauzele pentru coliziune externă sunt enumerate mai jos:

- IK fix, iar variabila contor a vectorului de inițializare IV este incrementată pentru fiecare interogare a algoritmului de EIA3;
- IK generat aleatoriu, iar variabila contor a vectorului de inițializare IV este incrementată pentru fiecare interogare a algoritmului de EIA3.

Cazurile de mai sus au nevoie de cel puțin 216 perechi de mesaje MAC cunoscute pentru a detecta coliziunile externe. Probabilitatea de detecție crește dacă mărim interogările algoritmului de prezicere. Acest lucru se poate vedea în următorul tabel:

Tabel 2. Statistici coliziuni externe

Nr. crt.	Numărul perechilor mesaj-MAC cunoscute	Numărul perechilor cu coliziuni identificate
1	2^{16}	1
2	2^{18}	8
3	2^{20}	140
4	2^{22}	2081

Probabilitatea de succes a unui atacator care dorește să obțină o valoare corectă MAC este egală cu $\max(2^{-ik}, 2^{-m})$, unde *ik* este lungimea cheii de inițializare în biți, iar *m* este lungimea rezultatului MAC în biți. Probabilitatea ca un atacator să obțină un MAC corect este egală cu 1 pentru un număr de perechi de mesaje mai mare decât 2^{32}. Un atacator cu acces la procedura de verificare poate genera un mesaj fals. Acest lucru poate fi foarte dăunător când un centru GSM de autentificare AUC încearcă să autentifice abonații.

3. Eficiența de criptare. Vulnerabilități.

Pentru a asigura securitatea algoritmului de criptare ZUC, a fost implementat un program de evaluare complex, format din evaluarea de către comisia ETSI SAGE, de către echipe de experți academicieni și în final o perioadă de evaluare publică. După acest program îndelungat de evaluare s-a decis că algoritmul ZUC este îndeajuns de rezistent la multe tipuri de atacuri: Weak Key Attacks, Guess-and-Determine Attacks, Algebraic Attacks, Timing Attacks etc. Algoritmul ZUC nu a fost suficient de robust să reziste la atacuri de tip DPA. De menționat că putem face distincție între *Simple Power Analysis* (SPA) și metode de analiză mai puternice precum *Differential Power Analysis* (DPA) sau *Correlation Power Analysis* (CPA). Atacurile de acest tip speculează faptul că puterea consumată de un dispozitiv criptografic este dependentă de valorile intermediare obținute în timpul procesului de criptare.

Analiza puterii diferențiale (DPA) este un atac de tip canal aleator ce implică analiza statistică a măsurătorilor consumului de putere dintr-un criptosistem. Atacul speculează tendința consumului de putere al microprocesoarelor de a varia în timp ce efectuează operații cu chei secrete. Atacurile DPA au proprietăți de procesare de semnal și corectarea erorilor care extrag datele din rezultatele măsurătorilor care conțin prea mult zgomot pentru a putea fi analizate folosind o analiză de putere simplă (SPA). Prin intermediul DPA, un atacator poate obține chei secrete analizând măsurătorile consumului de putere rezultate din operațiile de criptografiere efectuate de către un dispozitiv vulnerabil.

Analiza diferențială de putere

În ultimul deceniu, prin analiza de canal adiacent s-a demonstrat că există o amenințare majoră asupra implementărilor criptografice, indiferent de versiunile de software sau hardware. DPA este una dintre cele mai

comune tehnici datorită eficienței sporite, aplicabilității bune și performanței ridicate comparativ cu alte metode. Aproape toate metodele de lucru pe DPA sunt codări bazate pe blocuri, există foarte puține metode de codare bazate pe flux continuu. Atacurile DPA întâmpină o serie de dificultăți dacă la bază avem un algoritm de generare de flux continuu. Cheia generată de un astfel de flux este independentă de textul și textul criptat și nu se pot folosi valorile de putere ale acestora (text și text criptat) și valorile ghicite (părți din codul ce trebuie recompus) pentru a determina rezultatul final.

DPA și fluxurile criptate

În aplicații reale, pentru a se păstra sincronizarea transmițător-receptor, fluxul criptat trebuie resincronizat frecvent. În această situație, starea inițială a fluxului este schimbată frecvent cu diferiți vectori de inițializare (IV), în timp ce cheia secretă (ce se dorește a fi aflată în timpul unui atac DPA) este aceeași. Atacurile DPA rămân posibile și pentru acest tip de metodă de criptare.

Atacurile DPA se desfășoară după cum urmează, dorindu-se recuperarea parolei byte cu byte:

- Stadiul colectării de date: se citește consumul de putere;
- Se folosește DPA în prima rundă de inițializare pentru a recupera k9 și k5. Valorile registrelor R1 și R2 și starea celor 16 celule LFSR din prima rundă sunt folosite în a doua etapă a stadiului de inițializare pentru a recupera k10 și k6;
- Se utilizează aceeași tactică pentru următoarele câteva etape și se determină informațiile corecte k11, k7, k12 și k13;
- În a 6-a etapă se face o căutare amănunțită pentru k15, k14, k4 și k0;
- k11 și k1 se determină ca în etapa a 6-a;
- Se determină ultimul byte al cheii secrete de inițializare, k2, în a 8-a etapă.

5. ZUC vs alți algoritmi de criptare în telecomunicații

Câteva dintre aspectele cheie ale algoritmului ZUC sunt prezentate în continuare:

- Generează șirul criptat cu algoritmul ZUC 1.6;
- Viteză foarte mare: până la 40Gbps în procesul de 65nm, 10Gbps în Altera Stratix III;
- Dimensiuni mici: cel puțin 7,5K porți logice;
- Îndeplinește specificațiile ETSI SAGE ZUC și EAE3/EIA3 (algoritmi de confidențialitate/integritate);
- Date de ieșire - blocuri de date de 32 biți;
- Folosește cheie și vector de inițializare de 128 biți;
- Nu necesită memorie externă;
- Cod complet funcțional disponibil în Verilog și inclus în librăriile ASIC.

Aplicații în care se folosește ZUC:

- Comunicații mobile securizate;
- Generare șiruri pentru algoritmii de confidențialitate și integritate 128-EEA3 și 128-EIA3 pentru LTE.

SNOW 3G

Câteva dintre aspectele cheie ale algoritmului SNOW 3G LTE sunt prezentate în continuare:

- Generează șirul criptat cu algoritmul SNOW 3G;
- Viteză mare: până la 7.5 Gbps în procesul de 65nm;
- Dimensiuni mici: cel puțin 7,5K porți logice;
- Îndeplinește specificațiile ETSI SAGE SNOW 3G și UEA2/UIA2 (algoritmi de confidențialitate/ integritate);
- Date de ieșire - blocuri de date de 32 biți;
- Folosește cheie și vector de inițializare de 128 biți;

- Nu necesită memorie externă;
- Cod complet funcțional disponibil în Verilog și inclus în librăriile ASIC.

Tipurile de aplicații în care se folosește SNOW 3G sunt:

- Comunicații mobile securizate;
- Generarea șirurilor pentru algoritmii de confidențialitate și integritate UEA2 și UIA2 pentru LTE.

Kasumi

Câteva dintre aspectele cheie ale algoritmului Kasumi sunt prezentate în continuare:

- Generează șirul criptat cu algoritmul Kasumi - cifru tip bloc de date;
- Execută doar criptare, nu și decriptare;
- Rata de criptare mare: până la 3 Gbps în procesul de 65nm;
- Dimensiuni mici: cel puțin 5,5K porți logice;
- Îndeplinește specificațiile ETSI SAGE KASUMI și 3GPP TS 35.202;
- Procesează blocuri de date de 64 de biți;
- Folosește cheie de 128 biți;
- Nu necesită memorie externă;
- Cod complet funcțional disponibil în Verilog și inclus în librăriile ASIC.

Printre aplicațiile în care se folosește Kasumi se pot aminti:

- Comunicațiile mobile securizate;
- Generarea șirurilor pentru algoritmii UMTS f8 și f9.

DES/3DES

Câteva dintre aspectele cheie ale algoritmului DES/3DES (*Data Encryption Standard*) sunt prezentate în continuare:

- Criptează și decriptează folosind algoritmul DES/3DES - un algoritm tip bloc de date;
- Viteză mare: până la 3 Gbps în procesul de 90 nm;
- Dimensiuni mici: cel puțin 3K porți logice pentru 3DES;
- Procesează blocuri de date de 64 de biți;
- Opțiune de verificare a parității;
- Folosește una, două sau trei chei de 56 de biți fiecare;
- Nu necesită memorie externă;
- Cod complet funcțional disponibil în Verilog și inclus în librăriile ASIC;
- DES simplu nu necesită licență.

Algoritmii DES/3DES se folosesc în aplicații de tipul:

- Comunicațiilor mobile securizate;
- Tranzacțiilor financiare securizate;
- Identificării securizate prin frecvență radio (RFID).

AES

Câteva dintre aspectele cheie ale algoritmului AES (*Advanced Encryption Standard*) sunt prezentate în continuare:

- Criptează și decriptează folosind algoritmul AES; nucleele de dimensiuni mici oferă doar criptare, implementările mai complexe oferă și decriptare;
- Viteză mare - 10 Gbps;
- Mai rezistent și mai rapid ca DES/3DES;
- Procesează blocuri de date de 128 de biți;
- Chei de criptare de 128, 192 sau 256 de biți;
- Opțiune de verificare a parității;
- Design simplu, sincron, reutilizabil;
- Nu necesită memorie externă;
- Cod complet funcțional disponibil în Verilog și inclus în librăriile ASIC.

6. Concluzii

ZUC este un cifru pe flux de date, ușor de implementat; este unul dintre cei mai rapizi algoritmi pentru criptarea mesajelor în comunicațiile mobile. Datorită cheii și vectorului de inițializare de mărime mare, 128

de biți, oferă o securitate ridicată. Este însă vulnerabil la atacuri de tip DPA (analiza diferențială a puterii). Atacurile de acest tip speculează faptul că puterea consumată de un dispozitiv criptografic este dependentă de valorile intermediare obținute în timpul procesului de criptare.

Bibliografie

[1]. M. Tang, P. Cheng and Z. Qiu, *Differential Power Analysis on ZUC Algorithm*, Cryptology ePrint Archive, 2012.

[2]. R. Z. Haider, *Birthday Forgery Attack on 128-EIA3* (Version 1.5), National University of Science and Technology, Pakistan, 2010.

[3]. G. Sekar , *The Stream Cipher Core of the 3GPP Encryption Standard 128-EEA3*: *Timing Attacks and Countermeasures*, Indian Statistical Institute, Chennai Centre, SETS Campus, MGR Knowledge City, CIT Campus, Taramani, Chennai 600113, India, 2009.

[4]. Hongjun Wu et al., Cryptanalysis of the Stream Cipher ZUC in the 3GPP Confidentiality & Integrity Algorithms 128-EEA3 & 128-EIA3, Nanyang Technological University, Singapore, Jun. 2010.

[5]. ZUC Cipher. (2013, Oct. 04) [Online] http://www.ipcores.com/ZUC_cipher_IP_core.htm

[6]. Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document2: ZUC Specification - ETSI/SAGE Specification.

[7]. Wikipedia. (2013, Oct. 18) [Online] http://en.wikipedia.org/wiki/Zuc_stream_cipher