Универзитет у Београду

Математички факултет

Криптоанализа шифре E0

мастер рад

Студент:
Милош Мартиновић

Ментор:
Проф. Др Предраг Јаничић

Београд

2022.

Садржај

Предговор

Израда овог мастер рада је плод моје вишегодишње сарадње са проф. др. Миодрагом Живковићем без чијих савета и подршке не би ни било рада у овом облику. Проф. Живковић је суштински био ментор на изради ове мастер тезе, али је из административних разлога ту улогу преузео проф. Јаничић. Проф. Јаничићу и проф. Малкову се такође захваљујем на корисним сугестијама које су даље унапредиле мој мастер рад. Велику захвалност дугујем и својој породици на подршци у моментима кад није било лако превазићи све препреке које су се појављивале у процесу израде мастер рада.

1.Увод

E_0 је алгоритам који се користи за шифровање у оквиру стандарда Блутут (Bluetooh). Блутут је стандард за бежичну комуникацију разних уређаја, најчешће мобилних телефона, бежичних слушалица, штампача. E_0 је проточна шифра која се састоји од четири померачка регистра, различите дужине, са линеарном повратном спрегом, који се комбинују са коначним аутоматом. У раду се приказује алгоритам E_0 и архитектура безбедносног система у оквиру стандарда Блутут.

Алгоритам за шифровање E_0 из рада [1] је програмски имплементиран у програмском језику C, што је проверено помоћу тест вектора који су дати у спецификацији Блутут-а верзија 1.1 [4]. У раду се описује и реализује напад на E_0, при чему се због сложености напада као целине (2^{76} корака), реализација може практично извршити за умањену верзију алгоритма E_0. Претпоставка криптоанализе је да нападач поред шифрата поседује и одговарајући отворени текст, а задатак је да се реконстуише остатак отвореног текста који није познат. Разматрани напад на шифру E_0 је типа *претпостави и одреди* напад, где се претпоставља иницијално стање коначног аутомата и садржај померачког регистра најкраће дужине.

2. Блутут комуникација

2.1 Историја

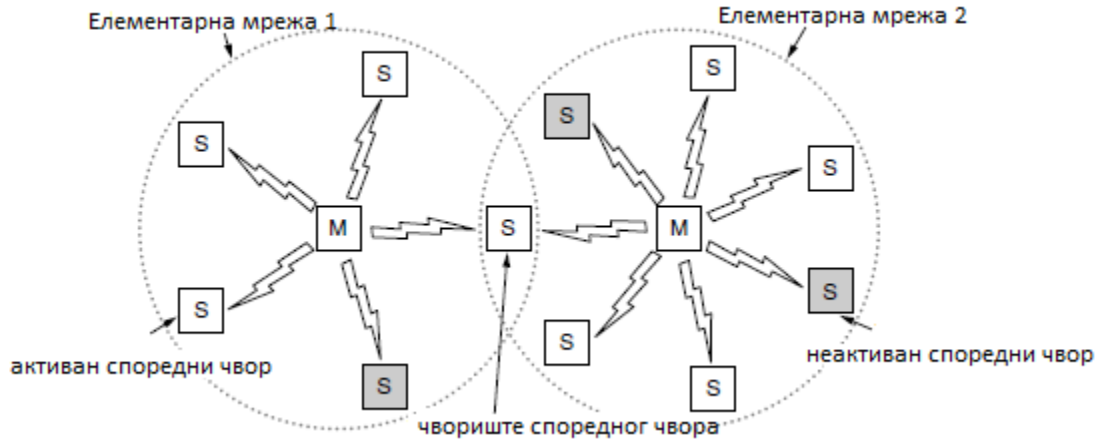
Почетком деведесетих година компанија Ериксон је кренула у решавање проблема повезивања без каблова својих мобилних телефона са другим електронским уређајима, пре свега са преносним рачунарима. Заједно са компанијама IBM, Intel, Toshiba и Nokia је 1998. године оформљена Специјална интересна група, SIG. Задатак тог конзорцијума је био да се стандардизује веза између рачунара и комуникационих уређаја, помоћу радио таласа кратког домета. Пројекат је именован Блутут по викиншком краљу Харланду Блаатанду. Блаатанд у буквалном преводу значи „плави зуб“, енгл. Bluetooth.

Прва верзија, Блутут 1.0 је објављена у јулу 1999. године и до данас Блутут користе практично цео опсег електронских уређаја, од мобилних телефона, рачунара, слушалица, играчких конзола, сатова итд. Верзија Блутут 3.0 из 2009. године омогућава повезивање уређаја са $WiFi$ мрежом чиме се убрзао пренос података. Актуелна верзија Блутут 5.2 је представљена у јануару 2020. на конференцији CES (Лас Вегас, САД).

2.2 Архитектура система Блутут

Основна јединица система Блутут је елементарна мрежа (енгл. piconet), која се састоји од једног главног чвора и највише 7 активних споредних чворова на раздаљини мањој од 10 метара. Више елементарних мрежа се може повезати преко чворишта, чиме се формира тзв. лабава мрежа (енгл. scatternet), слика 2.1.

Слика 2.1 Две елементарне мреже повезане у лабаву мрежу

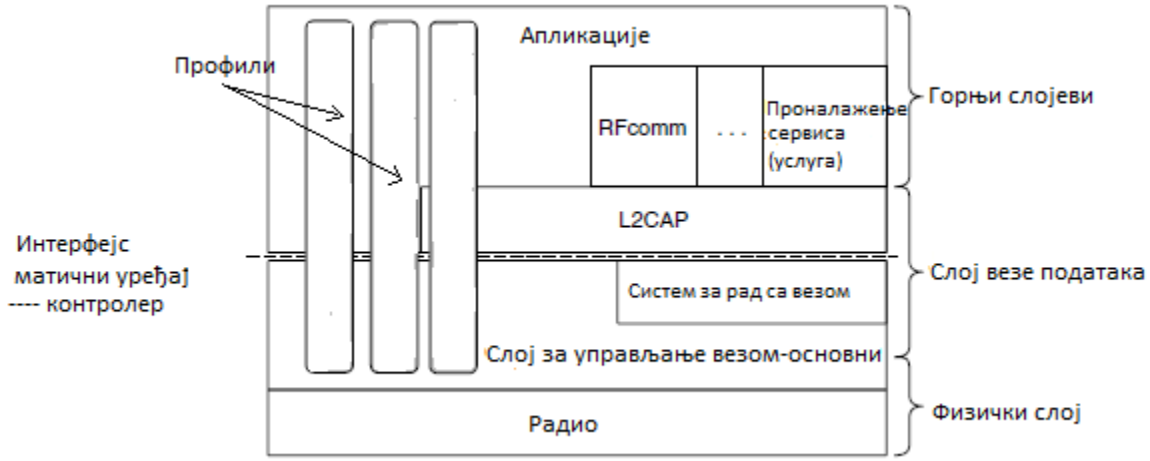


Главни разлог зашто се прави разлика између главних и споредних чворова је да се цена производње Блутут чипова спусти испод 5 долара. У елементарној мрежи главни чвор управља системским сатом (више у поглављу 2.6) и на тај начин одређује који од споредних чворова може да комуницира у одређеном интервалу. Сва комуникација је на релацији између главног и споредних чворова.

2.3 Скуп Блутут протокола

Блутут стандард садржи више протокола подељених у слојеве, слика 2.2.

Слика 2.2 Архитектура Блутут протокола



Архитектура протокола Блутут не личи на друге мрежне моделе попут TCP/IP, или мреже $WiFi$ (802.11). На самом дну протокола је физички радио-слој у коме се одвијају модулација и радио-пренос. Слој за управљање везом (основни, енгл. baseband) захвата и елементе физичког слоја. Ту се одређује начин на који главни чвор управља временским интервалима и како се временски интервали групишу у оквире [6].

Наредни слој је протокол за управљање везом (енгл. link manager). Систем за рад са везом прави логичке везе међу уређајима, контролише шифровање и управљање, као и квалитет услуге. Линија интерфејс матични уређај-контролер (енгл. Host-controller interface) раздваја две групе протокола и то тако што се протоколи испод линије инетрфејса реализују на Блутут чипу, а протоколи изнад линије реализују на Блутут уређају у ком се налази тај чип.

Изнад линије се налази протокол за LLC адаптацију (енгл. Logical Link Adaption Protocol, $L2CAP$) који склапа поруке променљиве дужине и обезбеђује поузданост. Велики број протокола користи $L2CAP$. Помоћу протокола Service Discovery се проналазе услуге на мрежи. Протокол $RFcomm$ (енгл. radio frequency communication) емулира стандардни серијски прикључак-порт PC рачунара, преко којег се са рачунаром повезују миш и тастатура.

У горњем слоју су смештене апликације. На слици 2.2 профили су представљени вертикалним кућицама, јер сваки од њих дефинише свој подскуп протокола намењен за одређену сврху. Помоћу мрежних протокола се успостављају канали између саговорника, а апликацијама се оставља да их користе на најбољи начин. SIG конзорцијум одређује конкретне примене и за сваку обезбеђује посебан скуп протокола. У једном моменту било је 25 таквих примена, тзв. профила.

За разне примене аудија и видеа постоје 6 профила. Профил $Intercom$ омогућава да два профила комуницирају као токи-воки уређаји. Профили $Headset$ и $Hands-free$ дозвољавају комуникацију између комплета за главу и његове базне станице, што омогућује разговор телефоном за време вожње. Профил $HeadSet$ не садржи никакве друге протоколе осим неопходних. Профили укључују $L2CAP$ ако имају пакете за слање, а изостављају $L2CAP$ ако имају сталан проток аудио узорака.

2.4 Радио слој Блутут везе

Сав саобраћај на релацији главни-споредни чворови иде кроз радио слој. Домет овог система, који ради на 2,4GH_Z , као и $WiFi$, је 10 метара. Подручје је издељено у 79 канала ширине 1MH_Z . Да би био могућ истовремени рад више мрежа примењује се фреквентно скакање са 1600 скокова у секунди и временом боравка на фреквенцији од 625μs. Сви споредни чворови у мрежи прелазе истовремено на следећу фреквенцију, уз поделу на временске интервале и псеудослучајну секвенцу скокова по диктату главног чвора.

У прошлости је долазило до међусобног ометања Блутут и $WiFi$ мреже што је онемогућавало њихов заједнички рад. У једном тренутку неке компаније су забраниле коришћење Блутута, али се касније дошло до задовољавајућег техничког решења. Одлучено је да Блутут прилагоди своју секвенцу скокова тако да избегне канале на којима већ постоје други RF сигнали и тако смањи штетну интерференцију. Овај поступак се зове адаптивно фреквентно скакање (енгл. adaptive frequency hopping).

2.5 Слојеви Блутут везе

Основни слој у систему Блутут (енгл. baseband, слика 2.2) сирове податке (битове) претвара у пакете и поставља неке основне формате. У једноставном случају, главни чвор елементарне мреже дефинише низ временских интервала од по 625μs. Парни интервали су резервисани за емитовање главног чвора, а непарни за емитовање споредних. Ово представља класично мултиплексирање поделом времена пола-пола, где један пакет може да заузме 1, 3 или 5 временских интервала. Унутар пакета кориснички подаци ради безбедности могу бити шифровани кључем који се бира приликом повезивања главног и споредног чвора. Фреквенција се никад не мења током преноса података, већ само између слања пакета. Из тог разлога, пакети од 3 или 5 временских интервала (енгл. slots) су ефикаснији од пакета који стају у један временски интервал, јер се за системске податке користи исти број битова, а шаље се 3 или 5 пута више корисничких података [6]. Између главног и споредног чвора сви пакети се крећу логичким каналом званим веза (енгл. link), који је постављен од стране протокола за управљање везом.

На почетку везе се проверава да ли два уређаја уопште могу да комуницирају, тј.обавља се поступак упаривања. Један од начина упаривања је да се оба уређаја конфигуришу са истим четвороцифреним бројем PIN (енгл. Personal Identification Number). Тај параметар је све што треба једном уређају да би се повезао са другим удаљеним партнером. Овакав начин повезивања није најбољи, јер учесници у комуникацији неретко користе једноставне PIN-ове, попут 0000 или 1234. Боља варијанта безбедног повезивања је да корисници само потврде да оба уређаја имају исти општи кључ (енгл. passkey), или да се на једном уређају очита његов кључ и пренесе на други уређај. Ова варијанта повезивања потпуно искључује коришћење PIN-а, и учесници у комуникацији треба само да потврде дужи кључ генерисан у уређају.

У моменту кад се процес упаривања заврши, комуникацију међу уређајима преузима протокол за управљање везом. Постоје два начина повезивања за пренос података међу корисницима. Први начин је синхроно повезивање уз креирање директне везе (енгл. Synchronous Connection Oriented, SCO). Сваком смеру у том начину повезивања се додељују тачно одређен временски интервал. Споредни чвор може бити повезан са највише три такве везе са

главним чвором. Како је време у SCO везама кључни фактор, пакети се шаљу само једном.

Други начин повезивања је асинхроно повезивање без успостављања директне везе (енгл. Async ConnectionLess, ACL), који се користи за повремени саобраћај комутирањем пакета. У овом начину повезивања се ништа не гарантује, јер се пакети могу изгубити, па се онда морају поново слати. Споредни и главни чвор могу имати једну такву везу.

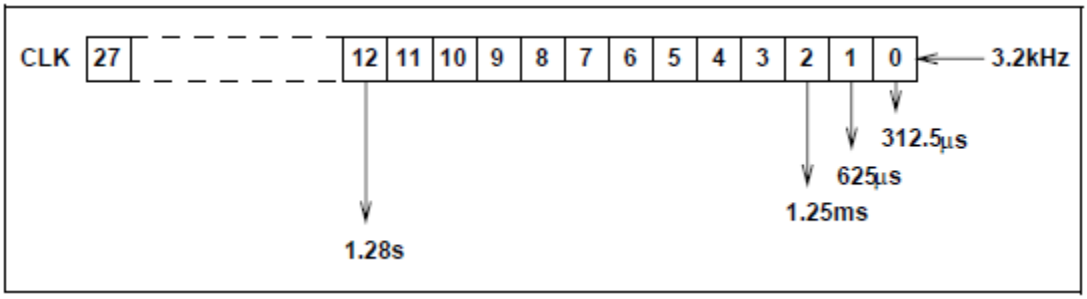
Подаци који се шаљу ACL везом долазе из $L2CAP$, слика 2.2. $L2CAP$ има четири функције. Он од виших слојева прихвата пакете до $64kb$ и распоређује их у оквиру за слање. На другом крају се од оквира поново склапају пакети. Тада $L2CAP$ одређује протокол из вишег слоја којем ће га послати. Затим се обезбеђује контрола грешака и поновно слање пакета за који није добијена потврда пријема. На крају слој $L2CAP$ обезбеђује тражени квалитет услуга за све везе.

2.6 Блутут клок

Сваки Блутут уређај има свој унутрашњи часовник који се никад не гаси и никад не подешава. Блутут клок (28-битни податак о времену) се добија очитавањем унутрашњег часовника. Усклађивање часовника са другим уређајима у мрежи се ради помоћу помераја (енгл. offsets) и унутрашњег сата. Сам Блутут часовник може бити иницијализован на било коју вредност и он представља кључни део Блутут примопредајника (енгл. transceiver). Часовник је заправо 28-битни бројач који има период од једног дана. На основу рада [4], најнижи бит бројача мења стање сваких 312.5μs, чему одговара брзина часовника од $3.2kHz$.

У оквиру пиконет мреже (мреже Блутут уређаја) која се успоставља, усклађивање низа промена фреквенције (Блутут користи систем хопинга, енгл. frequency hopping) на каналу везе одређује часовник главног (енгл. master) уређаја. Сваки споредни (енгл. slave) уређај у мрежи додаје померај свом часовнику да би се ускладио са часовником главног уређаја. Пошто сви часовници раде независно једни од других, помераји се морају редовно ажурирати.

Часовник одређује важне периоде у раду и окидач је за догађаје унутар Блутут примопредајника. У оквиру Блутут система постоје четири важна периода: 312.5μs, 625μs, 1.25ms и 1.28s. Ови периоди су повезани са битовима тајмера CLK_0, CLK_1, CLK_2 и CLK_{12} редом, слика 2.3.

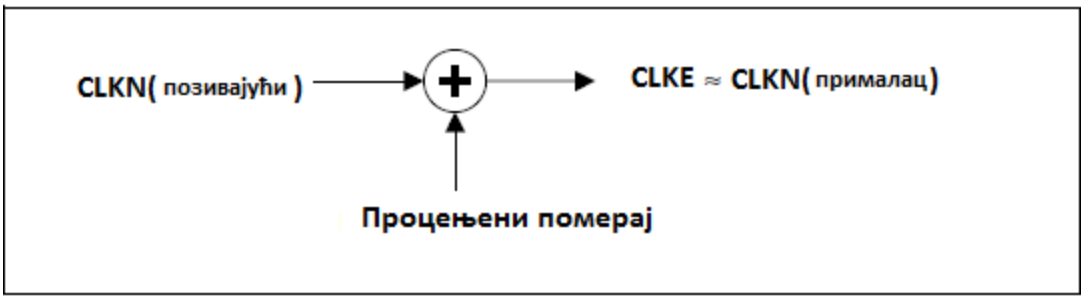


Слика 2.3. Бутут клок

Основни период у комуникацији (енгл. slot) траје 312,5μs . Комуникација на релацији главни-споредни уређај почиње у парним интервалима када CLK_0 и CLK_1 имају вредност нула. Блутут уређај може да се нађе у различитим режимима рада, па је са стањем часовника повезано неколико величина:

- $CLKN$ - унутрашњи клок
- $CLKE$ - претпостављени клок
- CLK - главни клок

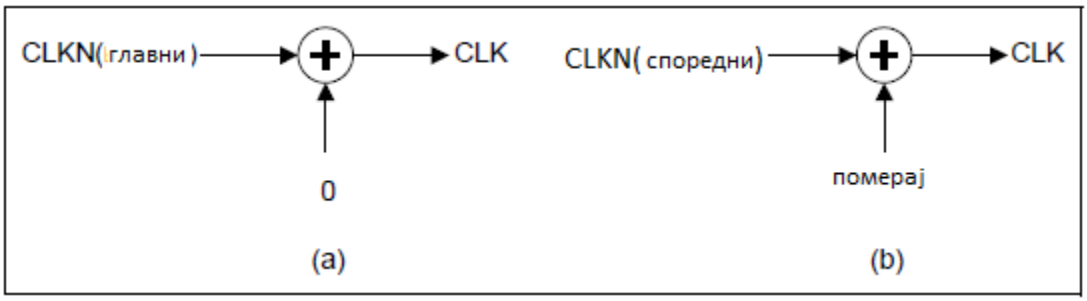
$CLKN$ је унутрашњи клок и он је референца за остале клокове. У стању високе активности, унутрашњи часовник се покреће кристалним осцилатором са тачношћу од ±20ppm (скраћено од engl. part per milion). У стањима ниже активности, као што су **standby**, **hold**, **park** и **sniff**, унутрашњи клок се одржава осцилатором мање снаге са умањеном тачношћу (± 250 ppm). Унутрашњи часовник (клок) има сваки уређај, и то је стање његовог бројача. Позивајући уређај (енгл. pager) и прималац (енгл. recipient) имају у општем случају различите вредности $CLKN$ (слика 2.4) .



Слика 2.4 Израчунавање $CLKN$ примаоца.

Прималац добија $CLKN$ позивајућег уређаја, утврђује за колико се то разликује од његовог $CLKN$ и одузимањем добија померај (енгл. offset). Описани процес је синхронизација.

Прималац сваки пут кад треба да израчуна CLK сабира свој тренутни $CLKN$ са тим израчунатим померајем (слика 2.5).



Слика 2.5 Израчунавање главног клока у главном и споредном уређају

Синхронизација мора повремено поново да се изврши, јер бројачи (часовници) позивајућег и примаоца у принципу не раде потпуно истом брзином.

CLK је часовник главног уређаја у пиконету. Он се користи за усклађивање активности унутар пиконет мреже. Сви Блутут уређаји такође користе CLK за усклађивање преноса и пријема података. CLK се изводи из унутрашњег сата $CLKN$ додајући померај, слика 2.5.

3. Опис алгоритма E_0

3.1 Улога алгоритма E_0 у оквиру Блутут комуникација

Протокол размене кључа унутар Блутут система

У овом делу текста излаже се начин успостављања Блутут везе, као и неке безбедносне поставке Блутут архитектуре. Протокол размене, тј. усаглашавања, кључа између Блутут уређаја је главни део безбедносне Блутут архитектуре. Претпоставимо да два уређаја, А и Б, желе да успоставе сигурну везу и да је уређај А тај који започиње комуникацију. Они покрећу протокол размене кључа да би одредили кључ везе (енг. link key). Кључ за шифровање је кључ који користи проточна шифра E_0 у процесу шифровања. Процес генерисања заједничког кључа зове се *упаривање* Блутут уређаја.

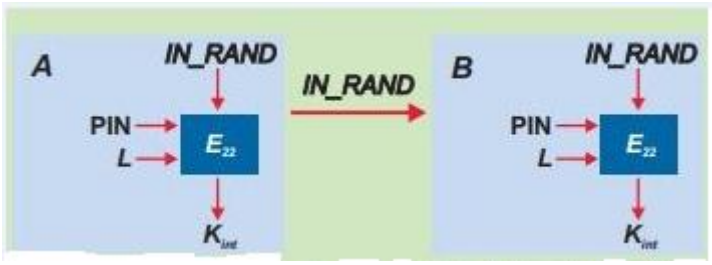
Генерисање кључа уређаја

Кад се Блутут уређај укључи по први пут, он рачуна кључ уређаја (енг. unit key). У питању је кључ који је јединствен за сваки уређај и готово се никад не мења и налази се у делу меморије који се чува и кад је искључено напајање. Овај кључ се користи само ако један од уређаја нема довољно велику меморију за смештање кључа сесије, што ће бити описано у делу генерисање кључа везе. Кључ уређаја се генерише на следећи начин: прво уређај рачуна случајни број $RAND$. На основу њега и Блутут хардверске адресе (јединствене фабрички уграђене вредности за сваку уређај понаособ), рачуна се кључ уређаја.

Генерисање кључа иницијализације

Да би се дошло до кључа сесије потребног ради комуникације међу Блутут уређајима, мора се проћи више корака. Прво треба креирати кључ иницијализације (енг. initialisation key). Овај привремени кључ је функција случајног броја IN_RAND (генерисан од стране уређаја A и послат уређају B), дељеног броја PIN и дужине L броја PIN, слика 3.1. PIN треба да се унесе на оба уређаја. Дужина PIN-а се креће од 8 до 128 битова. Најчешће се PIN састоји од четири децималне цифре. Уколико један од уређаја нема

могућност уношења података, користи се фиксирани PIN (претпостављена вредност је често 0000).

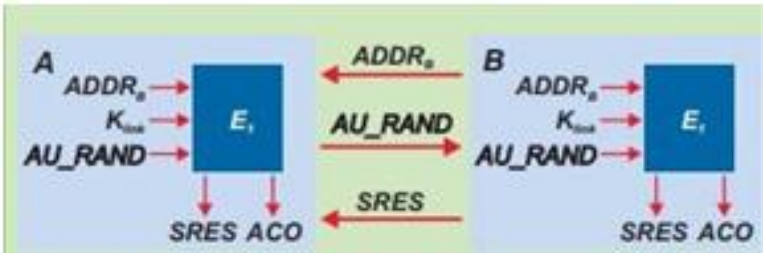


Слика 3.1 Генерисање кључа иницијализације

Као резултат добија се привремени дељени кључ, кључ иницијализације. Можемо приметити да је број PIN кључан за креирање кључа иницијализације, пошто за неког ко прати канал везе (енг. eavesdropper) можемо претпоставити да зна случајни број IN_RAND, слика 3.1.

Аутентикација уређаја који учествују у Блутут комуникацији

Сваки пут кад се направи нови дељени кључ иницијализације или кључ везе, оба уређаја покушавају да спроведу међусобну аутентикацију, која је базирана на протоколу изазов-одговор (енг. challenge-response).



Слика 3.2 Протокол међусобне аутентикације уређаја

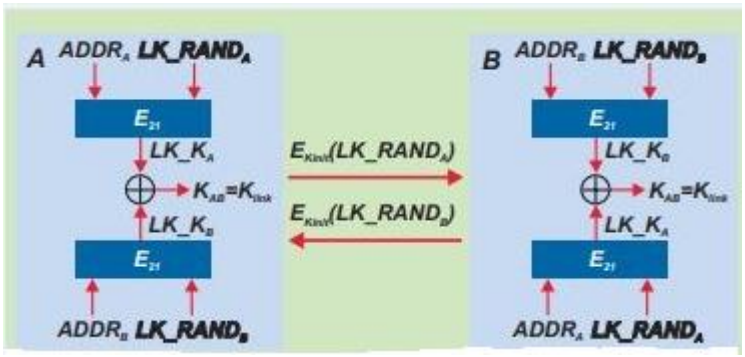
Овај протокол се извршава два пута. Прво се уређај B представља уређају A, слика 3.2. Уколико је ово представљање тј. аутентикација успешна, онда се уређај A представља уређају B. Аутентикација почиње тиме што уређај A генерише случајан број AU_RAND и шаље га уређају B [11]. Овај случајни број се назива *изазов* (енг. chalenge). Оба уређаја сад рачунају *одговор*:

$$SRES = E_1(\, ADDR_b, K_{link}, AU\_RAND\,),$$

где је $ADDR_b$ хардверска Блутут адреса уређаја B, а K_{link} је дељени кључ (кључ иницијализације или кључ везе). Затим уређај B враћа *одговор* уређају A. Уколико се *одговор* поклопи са вредношћу коју је израчунао сам уређај A, тада се може сматрати да је аутентикација успешно завршена. Наставља се са израчунавањем помоћног параметара ACO (енг. Authenticated Ciphering Offset) који се касније користи за рачунање кључа за шифровање, K_c. Алгоритам E_1, слика 3.2, је базиран на блок шифри $SAFER+$.

Генерисање кључа везе

Од овог тренутка оба уређаја имају заједнички кључ иницијализације. Из овог кључа изводи се нови практично трајни кључ, кључ везе (енг. link key). Кључ везе смешта се у оба уређаја ради даље комуникације. У зависности од меморијских капацитета два уређаја, кључ везе може бити кључ уређаја (енг. unit key) у уређају са малим капацитетом меморије или комбиновани кључ (енг. combination key) добијен из унетих вредности на оба уређаја, слика3.3.



Слика 3.3 Кључ везе је комбиновани кључ

Уколико је кључ уређаја A кључ везе, он се преноси шифрован од A до B. Шифровање се ради $XOR-$овањем кључа уређаја A са кључем иницијализације.

Уколико је кључ везе комбиновани кључ, онда оба уређаја генеришу прво случајан број LK_RAND. Ови случајни бројеви се шифрују кључем иницијализације алгоритмом E_{21}и шаљу другом уређају. Затим оба уређаја рачунају:

$$LK\_K_A = E_{21}(LK\_RAND_A\ ,\ ADDR_A),\ \text{и}$$

$$LK\_K_B = E_{21}(LK\_RAND_B,\ \ ADDR_B).$$

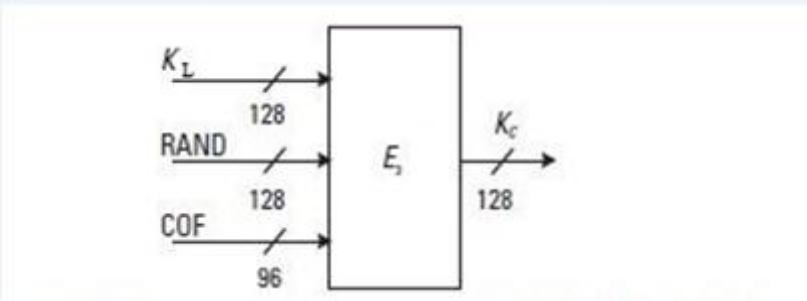
Комбиновани кључ K_{AB} се добија XOR −овањем LK_K_A и LK_K_B, слика 3.3. Алгоритам E_{21} је блок шифра $SAFER +$, са малим модификацијама. Пошто се изгенерише кључ везе, претходни кључ иницијализације се одбацује и почиње међусобна аутентикација уређаја са размењеним кључевима везе између уређаја, на управо описани начин.

Поступак описан на слици 3.3 се такође извршава приликом израчунавања новог кључа везе. Једина разлика је у томе што се случајни бројеви LK_RAND шифрују претходним кључем везе. Када се изгенерише нови кључ везе, стари се одбацује.

Након успешног креирања кључа везе и извршавања протокола узајамне аутентикације, може се прећи на генерисање кључа за шифровање.

3.2 Генерисање кључа за шифровање K_c'

Пре почетка процеса шифровања мора се израчунати кључ сесије. Уоквиру протокола Блутут кључ везе (енг. link key) се не узима директно за сам процес шифровања, већ се он у неколико корака доводи до потребног нивоа за механизам шифровања. Кључ K_c се може схватити као кључ за шифровање вишег нивоа из ког се изводе остали кључеви. Уређај A генерише случајан број EN_RAND_A и шаље га уређају B. Даље оба уређаја рачунају кључ шифровања K_C помоћу алгоритма E_3 уз улазне величине: K_L – тренутни кључ везе, 96-битни шифарски померај (COF), и 128 битни случајни број EN_RAND на основу израза $K_c = E_3(EN\_RAND_A, K_{LINK}, COF)$, слика 3.4.



Слика 3.4 Рачунање кључа за шифровање K_c

Вредност COF је заправо вредност ACO која се рачуна у оквиру протокола међусобне аутентикације. Ипак, уколико се кључ шифровања користи за слање онда је COF конкатенација (означена са $||$) хардверске Блутут адресе пошиљаоца са самом собом, тј. $COF{=}ADDR||ADDR$.

Унутар самог система Блутут долази до трансформације 128 битног кључа K_C у 128 битни кључ K_c' чија је реална дужина мања од 128 битова. Овде се под појмом реална дужина кључа подразумева логаритам за основу два броја могућих комбинација битова кључа. Као резултат се добија ограничени (енг. constraint) кључ K_C'. Ограничени кључ је уведен због извозно правних регулатива за криптографски хардвер. На основу рада [5] за дато L се K_c' рачуна се на следећи начин:

$$K_C'(x) = g_2^L(x)\{\, K_C(x)[\mathrm{mod} g_1^L(x)]\} \qquad (3.1)$$

где је

$$K_c = \left(K_{c,0}, \ldots, K_{c,127}\right), \qquad K_{c,i} \in \{0,1\}, \qquad K_{\mathrm{c}}(x) = \sum_{i=0}^{127} K_{\mathrm{c,i}}\, x^i \ ,$$

$$K_c' = \left(K_{c,0}', \ldots, K_{c,127}'\right), \qquad K_{c,i}' \in \{0,1\}, \qquad K'_{\mathrm{c}}(x) = \sum_{i=0}^{127} K'_{\mathrm{c,i}}\, x^i .$$

Овде су $g_1^L(x)$ и $g_2^L(x)$ полиноми над пољем $GF(2)$. Број L представља индекс ова два полинома, од 1 до 16, слика 3.5, и део је ознаке једног и другог полинома (није у питању степеновање на L). Другим речима, има 16 варијанти полинома $g_1(x)$ и $g_2(x)$ и у свакој варијанти је збир њихових степенова 128 или 127. Израчунавање на основу (3.1) се врши аритметичким операцијама над $GF(2)$. Операцијом свођења по модулу полинома $g_1(x)$ смањује се степен $K_c(x)$ на степен мањи од степена $g_1(x)$; нека је добијен полином $h(x)$. Тиме је реална дужина кључа $K'_{\mathrm{c}}(x)$ смањена и мања је или једнака од степена полинома $g_1(x)$. Множењем полинома $h(x)$ са $g_2(x)$ добија се полином степена мањег од 128.

L	Стп	g_1^L	Стп	g_2^L
1	8	0000000000000000000000000000011d	119	00e275a0abd218d4cf928b9bbf6cb08f
2	16	00000000000000000000000000001003f	112	0001e3f63d7659b37f18c258cff6efef
3	24	000000000000000000000000010100db	104	000001bef66c6c3ab1030a5a1919808b
4	32	00000000000000000000001000000af	96	000000016ab89969de17467fd3736ad9
5	40	0000000000000000000010000000039	88	000000000163063291da50ec55715247
6	48	000000000000000000010000000000291	77	0000000000002c9352aa6cc054468311
7	56	00000000000000000100000000000095	71	00000000000000b3f7fffce279f3a073
8	64	0000000000000001000000000000001b	63	0000000000000000a1ab815bc7ec8025
9	72	000000000000010000000000000000609	49	00000000000000000002c98011d8b04d
10	80	00000000001000000000000000000215	42	000000000000000000000058e24f9a4bb
11	88	0000000010000000000000000000013b	35	0000000000000000000000ca76024d7
12	96	000000010000000000000000000000dd	28	00000000000000000000000001c9c26d9
13	104	00000100000000000000000000000049d	21	0000000000000000000000000026d9e3
14	112	0001000000000000000000000000014f	14	00000000000000000000000000004377
15	120	0100000000000000000000000000000e7	7	00000000000000000000000000000089
16	128	10000000000000000000000000000000	0	00000000000000000000000000000001

Слика 3.5 Парови полинома g_1^L, g_2^L.

3.3 Иницијализација генератора E_0

Генератор E_0 се мора иницијализовати почетним вредностима за четири $LFSR$-а, укупно 128 битова, и четири бита, c_0 и c_{-1}, који одређују почетно стање коначног аутомата, слика 4.3. Ових 132 битова се добијају помоћу кључа K_c' дужине 128 битова, вредности BD_ADDR, слика 4.1, и времена CLK, вредности добијене од рада самог генератора E_0. Више о параметру CLK биће у делу о Блутут клоку, тачка 2.6.

На основу [5], иницијализација четири померачка регистра $LFSR$ се одвија на следећи начин:

1. Поред три улазне вредности $K_c^{'}$, адресе BD_ADDR, очитаног стања часовника CLK, додати и шестобитну константу 57 (слика 4.1).

а) Отворити све прекидаче на четири регистра, слика 3.6.

б) Сложити улазне битове на идентичан начин као на слици 3.6. Овде $ADR[i]$ означава битове BD_ADDR, а $CLK[i]$ означава одговарајуће битове

такта. Сви битови се убацују у регистре почев од бита најмање тежине (енг. least significant bit).

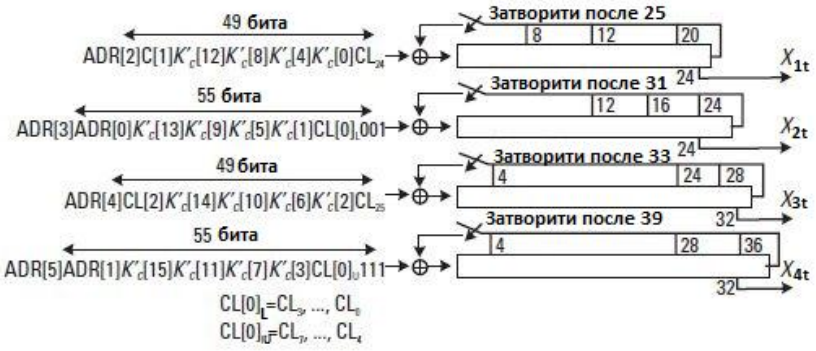
ц) На почетку процеса иницијализације, у моменту $t = 0$, поставити вредности регистра $LFSR$ на нулу.

д) Почети са убацивањем улазних бита, слика 3.6.

е) У моменту $t = 25$ затворити прекидач на првом регистру. Тада се улазни бит најкраћег регистра рачуна по формули из табеле 4.2. Излазни бит истог регистра се налази на 24-ом месту. Кад је $t = 31$ затворити други регистар. Кад је $t = 33$, тј. $t = 39$ затворити прекидаче на трећем, односно четвртом регистру.

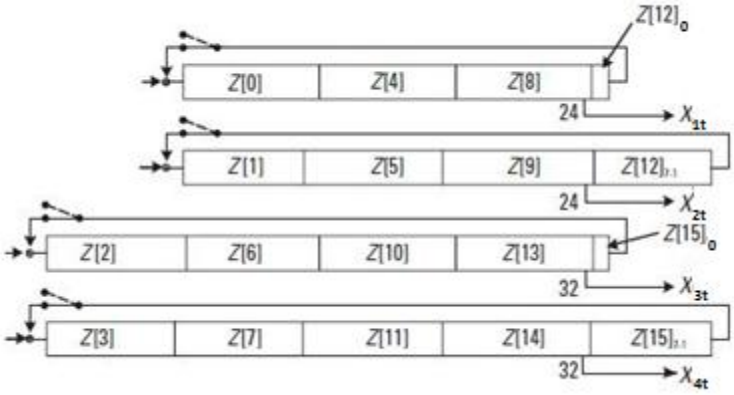
ф) У моменту $t = 39$ поставити битове c_{39} и c_{38} на нулу.

г) Пребацити у регистре преостале улазне битове.



Слика 3.6 Прва фаза иницијализације четири померачка регистра.

2. У моменту $t = 200$, када је генерисано 200 битова кључа, задржати тренутне вредности за c_t и c_{t-1}. Узети последњих 128 битова добијеног кључа. Нека су бајтови формирани од ових 128 битова означени са $z[0],...,z[15]$. Тим бајтовима се по други пут дефинише почетни садржај регистара, видети слику 3.7.



Слика 3.7 Друга фаза иницијализације четири померачка регистра.

3.4 Слабости протокола Блутут

Унутар технологије Блутут постоје одређене слабости, описане у тексту [7]. Као што је поменуто у тачки 3.1, кључ уређаја (енг. unit key) се користи само ако један актер у комуникацији нема довољно меморије да код себе држи сесијски кључ. Тај кључ се налази у непроменљивом делу меморије и готово никад се не мења. Такође, у делу генерисања кључа везе, одељак 3.1, кључ уређаја се шаље шифрован кључем иницијализације другом актеру у комуникацији. На основу [11], слабост система Блутут се огледа у следећем: ако је актер A послао свој кључ уређаја актеру B, онда B зна кључ уређаја A и може се представљати као A објекту C. Овакав напад са лажним представљањем (енг. impersonation attack) је немогуће избећи. Из тог разлога се препоручује да се кључеви уређаја не користе у комуникацији.

Постоје безбедносни проблеми у протоколу *изазов-одговор* који користи алгоритам E_1, слика 3.2, одељак 3.1. У раду [8] се износе слабости блок шифре $SAFER+$, која је у основи E_1. Откривен је проблем у поступку проширивања кључа шифре $SAFER+$ што омогућава реконструкцију кључа нешто брже него потпуном претрагом. Ова врста напада не угрожава реално Блутут, већ има више теоријски значај. Предлаже се да се уместо блок шифре $SAFER+$ користи AES, који је бржи и безбеднији.

Слабости у безбедносној архитектури алгоритма E_0 су описане у радовима [7,8,9]. О томе ће више речи бити у следећем поглављу.

Напад најмањег нивоа сложености је алгебарски напад [10]. Алгоритам E_0 је подложан овој врсти напада због могућности израчунавања иницијалног стања регистара генератора E_0 решавањем система нелинеарних једначина степена 4 над коначним пољем $GF(2)$. Тај систем се може трансформисати линеаризацијом у систем независних линеарних једначина са највише 2^{23} непознате. И оваква врста напада не угрожава систем Блутут јер захтева дугачак низ кључа током процеса иницијализације, а сама шифра E_0 унутар Блутута користи мале пакете за пренос података, чија дужина иде од 0 до највише 2745 битова [4].

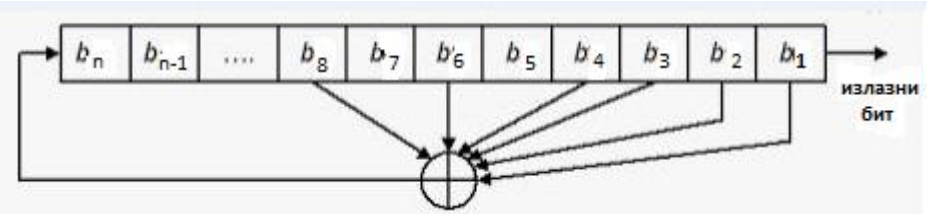
Постоје ипак и напади на E_0 који се могу практично извести. Многи од њих нису превише ефикасни, али такав није Воденејев (Serge Vaudenay) напад са познатим отвореним текстом [9]. Ово је један од најбржих напада на Блутут који користи мане у ресинхронизацији алгоритма E_0, као и условне корелације у коначном аутомату, унутар проточне шифре E_0. Воденејев напад проналази кључ шифровања за E_0 користећи прва 24 бита из $2^{23.8}$ блокова помоћу обраде 2^{38} варијанти.

3.5 Померачки регистар са линарном повратном спрегом и коначни аутомат с излазом

Померачки регистар с повратном спрегом (енг. feedback shift register) састоји се од померачког регистра чије су ћелије повезане функцијом повратне спреге (енг. feedback function). Сам померачки регистар је заправо низ битова и ако је регистар дугачак n битова, онда је то n-битни померачки регистар. Да би се добио један излазни бит, сви битови померачког регистра се померају за један бит удесно (видети слику 3.8). Бит који је пре померања био најдеснији је излазни бит. Нови бит на крајњој левој позицији се израчунава као функција свих битова у регистру. Период померачког регистра је дужина излазног низа пре него што исти крене да се понавља.

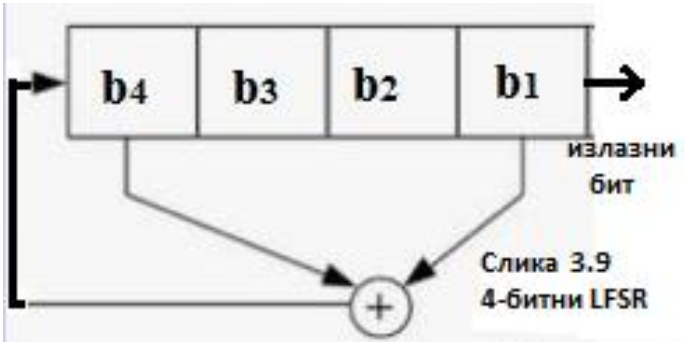
Најједноставнија варијанта померачких регистра с повратном спрегом, уједно и најчешће коришћена у криптографији, је померачки регистар са линеарном повратном спрегом (енг. linear feedback shift register, $LFSR$ слика 3.8). У овом случају померачки регистар има линеарну функцију повратне спреге, тј. та функција је примена само операције xor над одређеним битовима у регистру.

Листа позиција $t_1, t_2, \ldots t_m$ (претпоставља се да је $t_m = n$) над којим је примењена операција XOR се зове тап секвенца (енг. tap sequence).



Слика 3.8 Померачки регистар са линеарном повратном спрегом.

Једноставан 4-битни $LFSR$ чију тап секвенцу чине први и четврти бит је приказан на слици 3.9. Овде је број извода на регистру (дужина тап секвенце) $m = 2$, а тап секвенца је $t_1 = 1, t_2 = 4$.



Слика 3.9 Четворобитни $LFSR$.

Уколико је овај $LFSR$ на почетку иницијализован са 1110, он пролази кроз следећа унутрашња стања пре него она почну да се понављају:

1110
1111
0111
1011
0101
1010
1101
0110
0011
1001

0100
0010
0001
1000
1100

1110

Излазни низ битова овог регистра је низ битова најмање тежине: 0111101.... Померачки регистар дужине n може бити у једном од укупно $2^n - 1$ стања (ако то стање не чине све нуле, када излазни низ чине све нуле). То значи да $LFSR$ може да генерише псудослучајан низ са периодом дужине највише $2^n - 1$. LFS регистри са максималним периодом имају специјалне тап секвенце [10] и пролазе кроз сва унутрашња ненула стања. Тап секвенци $t_1, t_2, \dots t_m$ $LFSR$ дужине n може се придружити *полином повратне спреге*

$$1 + x^{t_1} + x^{t_2} + \cdots + x^{t_{m-1}} + x^n \qquad (3.2)$$

Неформално, полином се од функције повратне спреге добија тако што индекси постају експоненти. У претходном примеру, слика 3.9, x_{i+4} се представља као x^4, x_{i+3} је x^3 и x_i постаје x^0=1. Одговарајућа функција повратне спреге за полином повратне спреге из једначине (3.2) је

$$x_{i+n} = x_{i+t_1} \oplus x_{i+t_2} \oplus \dots \oplus x_{i+t_m}$$

Дужина померачког регистра једнака је степену полинома повратне спреге. Да би неки $LFSR$ имао максимални период, његов полином повратне спреге , једначина (3.2), треба да буде примитиван полином по модулу 2 (тј. треба да буде несводљив и најмање k за које полином дели $x^k + 1$ треба да буде једнако $2^n - 1$).

Функција повратне спреге за $LFSR$ са слике 3.9 је: $x_{i+4} = x_i \oplus x_{i+3}$, што може да се напише и као: $x_{i+4} \oplus x_i \oplus x_{i+3} = 0$. Одговарајући полином повратне спреге је $x^4 + x^3 + 1$.

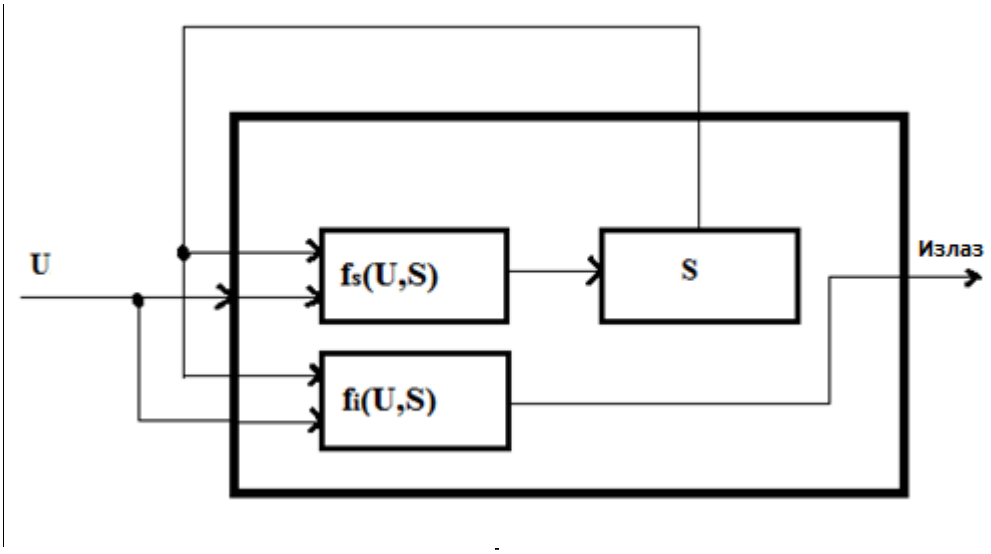
У оквиру напада на генератор E_0 померачки регистри $LFSR_2$, $LFSR_3$ и $LFSR_4$ имају непознат почетни садржај. Због тога се у сваком кораку t рачунају вектори коефицијената линеарних комбинација за сваку ћелију ових $LFSR$.

Почетно стање j-те ћелије регистра је вектор који на позицији j има јединицу, а на осталим позицијама нуле. Са векторима коефицијената линеарних комбинација се рачуна на исти начин као са битовима: вектори се

померају удесно за једну позицију, а на леви крај $LFSR$ уписује се вектор који се добија XOR -овањем вектора на одговарајућим позицијама $LFSR$.

У раду [3] наведени су неки примитивни полиноми различитих степенова. На пример, листи (32,7,6,2,0) одговара примитивни полином $x^{32}+x^{7}+x^{6}+x^{2}+1$. Због тога 32-битни померачки регистар са повратним спрегама са тридесет друге, седме, шесте и друге позиције је $LFSR$ са максималним периодом.

Коначни аутомат $KA=\{S,U,I,f_s,f_i,s_0\}$ се састоји од коначног скупа стања S, скупа улазних симбола U, скупа излазних симбола I, функције стања f_s која повезује пар (улаз, тренутно стање) са новим стањем коначног аутомата, функције излаза f_i која повезује пар (улаз, тренутно стање) са излазом из коначног аутомата, и иницијалног стања s_0, слика 3.10. Ако се коначни аутомат налази у стању $s\in S$ и на улазу добије симбол $u\in U$, онда он прелази у стање $f_s(u,s)$ и на излазу даје симбол $f_i(u,s)$.



Слика 3.10 Коначни аутомат

Један од начина за представљање рада коначног аутомата је преко табеле која за сваки пар $(u,s)\in U\times S$ садржи вредности функција $f_s(u,s)$ и $f_i(u,s)$. Други начин да опишемо рад коначног аутомата је преко дијаграма стања где је свако стање представљено кругом, а пар вредности (улаз, излаз) је означен на свакој стрелици прелаза између стања коначног аутомата. Табелама 3.11 и 3.12 представљене cu две табеле које описују начин рада коначног аутомата генератора E_0. Прва табела 3.11, даје вредност излазног бита коначног аутомата генератора E_0 на основу тренутног стања и улаза у аутомат, док

друга табела на основу истих улазних података као и прва, табела 3.12, даје следеће стање аутомата [4].

Улаз	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Стање																
0	0	1	1	0	1	0	0	1	1	0	0	1	0	1	1	0
1	0	1	1	0	1	0	0	1	1	0	0	1	0	1	1	0
2	0	1	1	0	1	0	0	1	1	0	0	1	0	1	1	0
3	0	1	1	0	1	0	0	1	1	0	0	1	0	1	1	0
4	1	0	0	1	0	1	1	0	0	1	1	0	1	0	0	1
5	1	0	0	1	0	1	1	0	0	1	1	0	1	0	0	1
6	1	0	0	1	0	1	1	0	0	1	1	0	1	0	0	1
7	1	0	0	1	0	1	1	0	0	1	1	0	1	0	0	1
8	0	1	1	0	1	0	0	1	1	0	0	1	0	1	1	0
9	0	1	1	0	1	0	0	1	1	0	0	1	0	1	1	0
10	0	1	1	0	1	0	0	1	1	0	0	1	0	1	1	0
11	0	1	1	0	1	0	0	1	1	0	0	1	0	1	1	0
12	1	0	0	1	0	1	1	0	0	1	1	0	1	0	0	1
13	1	0	0	1	0	1	1	0	0	1	1	0	1	0	0	1
14	1	0	0	1	0	1	1	0	0	1	1	0	1	0	0	1
15	1	0	0	1	0	1	1	0	0	1	1	0	1	0	0	1

Табела 3.11

Улаз	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Стање																
0	0	0	0	4	0	4	4	4	0	4	4	4	4	4	4	8
1	12	12	12	8	12	8	8	8	12	8	8	8	8	8	8	4
2	4	4	4	0	4	0	0	0	4	0	0	0	0	0	0	12
3	8	8	8	12	8	12	12	12	8	12	12	12	12	12	12	0
4	5	1	1	1	1	1	1	13	1	1	1	13	1	13	13	13
5	9	13	13	13	13	13	13	1	13	13	13	1	13	1	1	1
6	1	5	5	5	5	5	5	9	5	5	5	9	5	9	9	9
7	13	9	9	9	9	9	9	5	9	9	9	5	9	5	5	5
8	14	14	14	2	14	2	2	2	14	2	2	2	2	2	2	6
9	2	2	2	14	2	14	14	14	2	14	14	14	14	14	14	10
10	10	10	10	6	10	6	6	6	10	6	6	6	6	6	6	2
11	6	6	6	10	6	10	10	10	6	10	10	10	10	10	10	14
12	11	7	7	7	7	7	7	3	7	7	7	3	7	3	3	3
13	7	11	11	11	11	11	11	15	11	11	11	15	11	15	15	15
14	15	3	3	3	3	3	3	7	3	3	3	7	3	7	7	7
15	3	15	15	15	15	15	15	11	15	15	15	11	15	11	11	11

Табела 3.12

3.6 Поступак шифровања

При покушају комуникације два Блутут уређаја, прво долази до размене кључа, односно усаглашавања заједничког тајног кључа. Током овог протокола размене, најпре се усаглашава величина кључа који ће се користити. Сваки уређај има параметар L_{max} који одређује највећу дозвољену величину кључа у бајтовима, као и L_{min} који одређује најмању прихватљиву величину кључа. Циљ је да се приликом успостављања Блутут везе изабере највећа величина кључа подржана на оба уређаја.

Да би се смањила могућност криптографског напада у неким случајевима неће доћи до повезивања уређаја. Када је вредност L_{max} једног уређаја мања од минималне вредности дужине кључа L_{min} за други уређај, шифрована веза се неће успоставити.

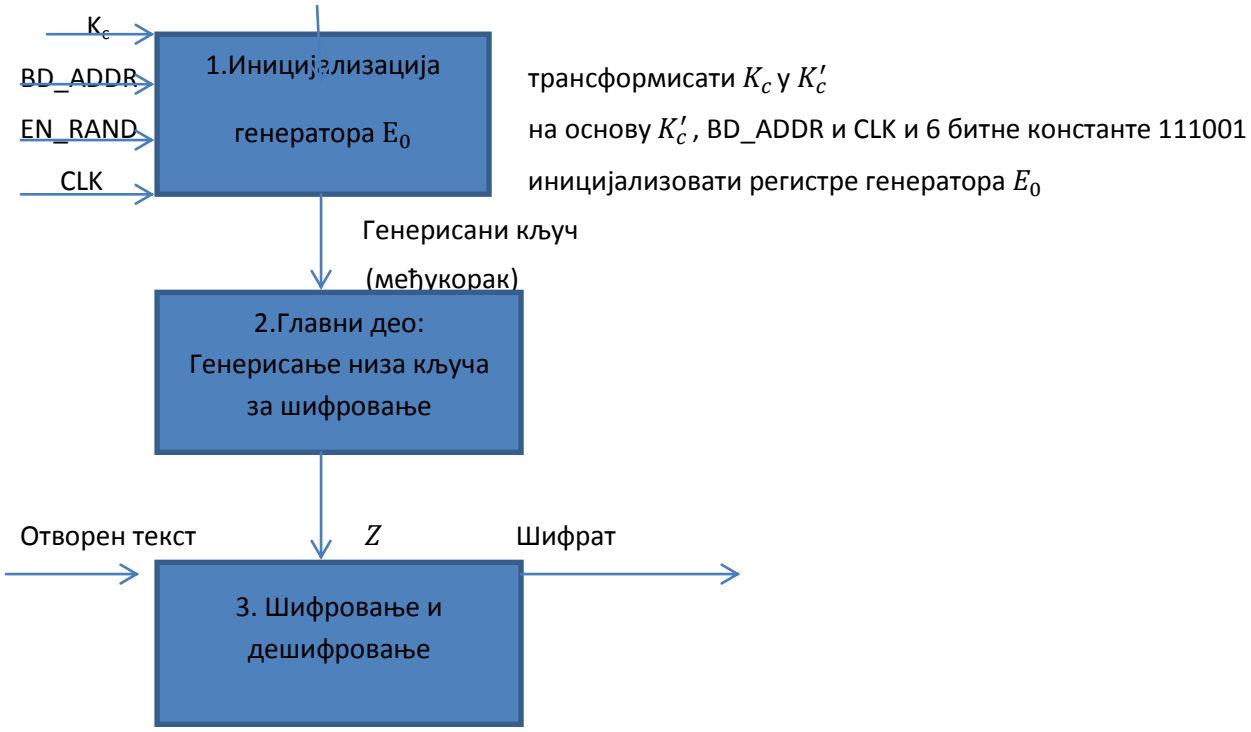
Сви учесници у мрежном саобраћају морају бити у могућности да прочитају заглавље пакета (енгл. packet header) да би видели да ли је порука намењена њима. У оквиру пакета шифром E_0 шифрују се само конкретни подаци (енгл. payload data).

Поступак шифровања алгоритмом E_0 се може поделити на три дела:

1) Иницијализација: генерисање иницијалног стања померачких регистара генератора E_0 (енгл. payload key generation). Овај део се састоји од убацивања улазних битова у одређеном распореду у линеарне регистре генератора E_0.

2) Покретање генератора E_0 и генерисање низа кључа за шифровање.

3) Шифровање, односно дешифровање сабирањем по модулу два, битова поруке са битовима низа кључа за шифровање.



Слика 4.1 Шифровање и дешифровање алгоритмом E_0

Уоквиру поглавља 3.1 поменуто је да сваки уређај који учествује у комуникацији има своју јединствену адресу и да један учесник који се повезује постаје главни (енгл. master) а остали споредни (енгл. slave).

Алгоритам E_0 добија као улаз 48 битова адресе главног Блутут уређаја BD_ADDR, 26 битова који одговарају тренутном стању часовника (енгл. real-time clock) CLK и кључ шифровања K_c. Параметар CLK је различит за сваки пакет и мења се сваких $625\mu s$, што резултује честим променама почетног стања генератора и повећањем отпорности на криптографске нападе. Детаљи рада Блутут часовник приказани су у одељку 2.6. Шифра је симетрична, па се дешифровање изводи на идентичан начин као и шифровање, користећи идентичан кључ.

Иницијализација шифре E_0 се одвија у две фазе. Прва фаза је почетна иницијализација, а друга генерише низ кључа шифре E_0.

Током првог нивоа иницијализације Блутут система, сесијски кључ K_c' у комбинацији са BD_ADDR и тактом CLK се користи за дефинисање иницијалног стања за четири померачка регистра са линеарном повратном спрегом ($LFSR$) укупне величине од 128 битова и четири бита стања коначног аутомата c_0 и c_{-1}. На завршетку прве фазе иницијализације креира се 200 битова проточне шифре, при чему се последњих 128 битова враћа назад у генератор E_0 као нове иницијалне вредности за четири померачка регистра, чиме почиње друга фаза генерисања низа кључа из генератора E_0. Пред почетак те фазе чувају се вредности регистара c_0 и c_{-1} са краја претходног дела иницијализације. Са затеченим вредностима стања померачких регистара генерише се максимална величина кључа за шифровање и генератор се реиницијализује да би се спречиле разне врсте криптогафских напада које би откриле иницијално стање Блутут система.

Као што је поменуто, стање генератора E_0 се састоји од четири померачка регистра укупне величине 128 битова, и четворобитног стања коначног аутомата. Пошто сами линеарни померачки регистри нису довољно криптографски сигурни, увођењем коначног аутомата постиже се нелинеарност шифре E_0. Померачки регистри се представљају полиномима повратне спреге са максималним периодом (примитивним бинарним полиномима), видети табелу 4.2.

Најмањи заједнички период сва четири $LFSR$ регистра је производ четири појединачна периода $P = (P_1 \times P_2 \times P_3 \times P_4)/7 \approx 2^{152.2}$. Производ се дели са 7 пошто је за P_3 и P_4 највећи заједнички делилац 7. На основу Блутут спецификације v1.2 [4], овај период генератор E_0 никад не достиже, јер се он сам поново иницијализује након генерисања највише 2745 битова.

LFSR	Дужина	Полином	Излазни бит	Дужина периода
$LFSR_1$	25	$t^{25}+t^{20}+t^{12}+t^8+1$	24	$2^{25}-1$
$LFSR_2$	31	$t^{31}+t^{24}+t^{16}+t^{12}+1$	24	$2^{31}-1$
$LFSR_3$	33	$t^{33}+t^{28}+t^{24}+t^4+1$	32	$2^{33}-1$
$LFSR_4$	39	$t^{39}+t^{36}+t^{28}+t^4+1$	32	$2^{39}-1$

Табела 4.2. Повратни полиноми четири линеарно померачка регистра

Излазни битови свих померачких регистара и излаз коначног аутомата се XOR-ују и тако се добија излазни бит генератора E_0. Затим се излазі из четири померачка регистра сабирају и добија се тробитни број. Горња два бита збира се користе као улаз у коначни аутомат.

Шифровање једног излазног бита одвија се на следећи начин:

а) Покрећу се четири померачка регистра и на излазу дају $x_{t,}^{i}\ i = 1,2,3,4$

б) Израчунава се наредни бит низа кључа $z_t = f_0\,(\,x_t\,, c_t\,)$, видети једначину (3.3)

ц) Израчунава се наредни бит шифрата $e_t = z_t \oplus m_t$, где је m_t одговарајући бит поруке.

д) Израчунава се s_{t+1}=$f_1(x_t,c_t)$, видети једначину (3.4)

е) Израчунава се наредно стање коначног аутомата c_{t+1}= $T(s_{t+1},c_t)$

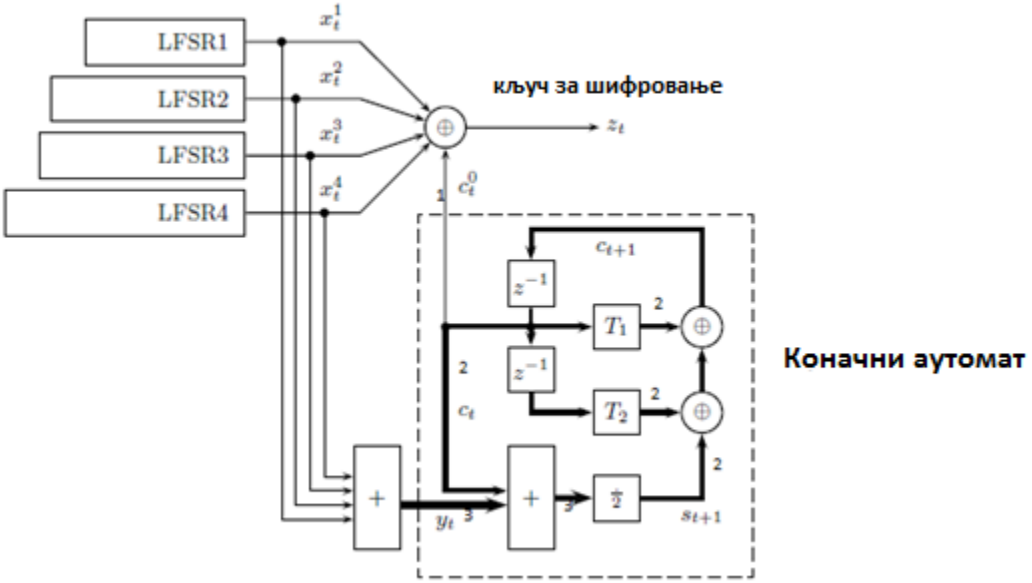
ф) $c_t = c_{t+1}$ – припрема меморијских битова за следећу итерацију коначног аутомата.

Поступак дешифровања шифрата је исти, изузев што у трећем кораку имамо $m_t = z_t \oplus e_t$, где је e_t одговарајући бит шифрата.

Схематски приказ рада генератора E_0 је приказан на слици 4.3, где је са z^{-1}означена меморијска ћелија која представља кашњење (енг.delay elements)

и садржи два бита. Мали бројеви испод линија означавају број битова који пролазе том путањом.

.



Слика 4.3 Проточна шифра E_0

Функција f_0 даје један бит излазног низа битова $z_1, z_2, ...$, где је $z_t \in GF(2)$. На основу [5], битови кључа за шифровање z_t се рачунају као сума по модулу 2 излазних битова померачких регистара x_t^i и најнижег бита c_t^0 излаза из коначног аутомата;

$$z_t = f_0(x_t, c_t^0) = x_t^1 \oplus x_t^2 \oplus x_t^3 \oplus x_t^4 \oplus (c_t^0 \bmod 2) \in \{0,1\} \quad (3.3)$$

Излазни битови регистра $LFSR_i$ у тренутку t, x_t^i, се добијају XOR-овањем својих одговарајућих битова за сваки од тих регистара понаособ (видети табелу 4.2).

Нелинеарна функција f_1 такође има као улазе вектор x_t и тренутно стање коначног аутомата c_t. Као излазну вредност функција f_1 даје двобитни вектор s_{t+1}. Ова функција је нелинеарна јер је целобројно сабирање нелинеарно у пољу $GF(2)$.

$$s_{t+1} = (s_{t+1}^1, s_{t+1}^0) = f_1(x_t, c_t) = [(y_t + 2 * c_t^1 + c_t^0)/2] \in \{0, 1, 2, 3\} \quad (3.4)$$

Овде је $y_t = x_t^1 + x_t^2 + x_t^3 + x_t^4 \in \{0, 1, 2, 3, 4\}$

Стање коначног аутомата одређују четири бита, који се чувају у две двобитне меморијске ћелије, z^{-1}. За сваки временски тренутак t, доњи меморијски елемент садржи претходну вредност горњег меморијског елемента. Зато ове две вредности можемо означити са c_t и c_{t+1}. Функција $\mathrm{T} = (\mathrm{T}_1, \mathrm{T}_2)$ се користи да измеша вредности ових битова, користећи четири бита из меморијских елемената и s_{t+1} као улаз. Као излазну вредност функција T враћа двобитни вектор c_{t+1} који се враћа на улаз коначног аутомата.

Нови садржај c_{t+1} горњег меморијског елемента се рачуна на основу следећег израза:

$$c_{t+1} = (c_{t+1}^1, c_{t+1}^0) = \mathrm{T}(s_{t+1}, c_t, c_{t-1}) = \mathrm{T}_0(s_{t+1}) \oplus \mathrm{T}_1(c_t) \oplus \mathrm{T}_2(c_{t-1})$$

Овде су T_0, T_1, и T_2 линеарне бијекције над пољем $GF(4)$, $(x_1, x_0) \to (y_1, y_0)$, дефинисане изразима

$$\mathrm{T}_0 = \mathrm{T}_1 \colon (x_1, x_0) \to (x_1, x_0),$$

$$\mathrm{T}_2 \colon (x_1, x_0) \to (x_0, x_1 \oplus x_0).$$

Овиме је описан процес рада проточне шифре E_0. Први програм који је имплементиран у програмском језику C, софтверски дефинише начин рада E_0, што је успешно тестирано на више тест вектора датих у Блутут спецификацији.

4. Опис реализованог напада

Приказује се напад са познатим отвореним текстом из рада [1], заснован на идеји да се за сваки претпостављени садржај регистра $LFSR_1$ и четири бита почетног стања коначног аутомата покуша са одређивањем почетног стања остала три регистра на основу расположивог одсечка излазног низа из генератора E_0.

Напад се састоји од петље у којој се пролазе све комбинације могућих почетних садржаја најкраћег регистра $LFSR_1$ и почетног стања коначног аутомата. За сваку претпоставку пролази се кроз познати део низа кључа z_t и успут одржава скуп S једначина по 103 непознате - почетна стања остала три померачка регистра: $x^2_{0...30}$, $x^3_{0..32}$ и $x^4_{0..38}$. При томе се прати, односно ажурира тренутно стање коначног аутомата (c_t и c_{t-1}), које зависи од целобројне суме непознатих излазних битова регистара $LFSR_2$, $LFSR_3$ и $LFSR_4$. Прецизније, напад се састоји од извршавања следеће процедуре за сваки претпостављени почетни садржај $LFSR_1$ и коначног аутомата:

1. Израчунати XOR тренутног излаза аутомата c^0_t, познатог излазног бита регистра $LFSR_1$ и тренутног бита низа кључа z_t. За тачну претпоставку добијени бит је једнак вредности XOR три излазна бита непознатих регистара $LFSR_2$,$LFSR_3$ и $LFSR_4$:

$$c^0_t \oplus x^1_t \oplus z_t = x^2_t \oplus x^3_t \oplus x^4_t$$

Тиме је одређена парност целобројне суме $x^2_t + x^3_t + x^4_t$, али то не одређује суму на јединствен начин. Ако је сума (видети табелу 4.4)

- **парна**, онда је вредност суме 0 или 2. У стабло претраге додају се две гране које одговарају овим двема варијантама и које се даље рекурзивно обрађују на исти начин:
  - ако је сума 0, онда се у систем S додају три линеарне једначине: $x^2_t = 0$, $x^3_t = 0$ и $x^t_4 = 0$.
  - у противном, ако је вредност суме 2, онда тројка (x^2_t, x^3_t, x^4_t) може да буде једнака (0,1,1), (1,0,1) или (1,1,0), што је еквивалентно са следећа два услова: XOR три бита је 0, а конјункција негације првог и негације другог бита је 0. Због тога се у систем S додаје линеарна једначина $x^2_t \oplus x^3_t \oplus x^4_t = 0$ и нелинеарна једначина $(1 \oplus x^2_t)(1 \oplus x^3_t) = 0$.

- **непарна**, онда је вредност суме 1 или 3. У стабло претраге додају се две гране које одговарају овим двема варијантама и које се даље рекурзивно обрађују на исти начин:

  - ако је сума 3, онда се у систем S додају три линеарне једначине: $x_t^2 = 1$, $x_t^3 = 1$ и $x_t^4 = 1$.

  - у противном, ако је вредност суме 1, онда тројка (x_t^2, x_t^3, x_t^4) може да буде једнака $(0,1,0)$, $(0,0,1)$, или $(1,0,0)$, што је еквивалентно са следећа два услова: XOR три бита је 1, а конјункција првог и другог бита је 0. Због тога се у систем S додаје линеарна једначина $x_t^2 \oplus x_t^3 \oplus x_t^4 = 1$ и нелинеарна једначина $x_t^2 x_t^3 = 0$.

  Вредности x_t^2, x_t^3 и x_t^4 у горњим изразима су линеарне комбинације непознатих - почетног стања одговарајућих регистара, видети део о једначинама које описују рад $LFSR$. Као што је речено, даље се изабере једна од две гране и додају се одговарајуће једначине у систем S.

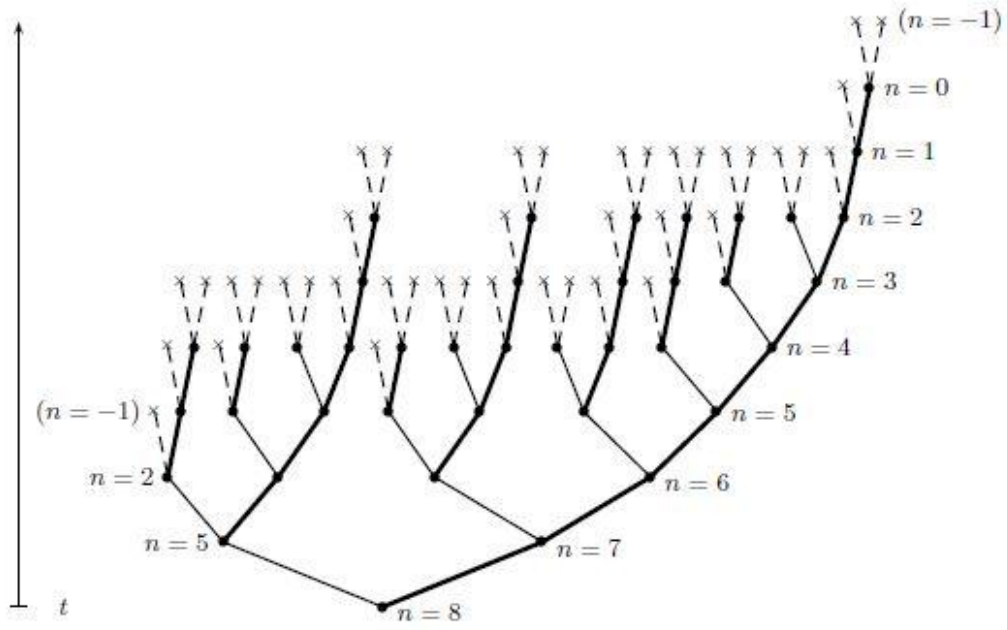
2. Проверава се конзистентност скупа S, па ако се добије контрадикција, враћамо се уназад (излази се из рекурзије) и разматра се следећа грана, односно наставља се са претходним рекурзивним позивом. Ако се прођу све гране у таквој претрази, претпоставка о почетном стању $LFSR_1$ и коначног аутомата се одбацује. Са друге стране, ако је систем S конзистентан и испуњен је услов $t > 132$, може се претпоставити да је одређено тачно иницијално стање $LFSR_1$ и коначног аутомата. Ово је на основу рада [1] оправдано претпоставити, јер се очекује да су 132 непозната битова почетног стања генератора E_0 одређени са 132 битова генерисаног низа кључа.

3. Ако систем S нема једнозначно решење, онда се користећи x_t^1 и претпостављену целобројну суму из корака 1, рачуна наредно стање коначног аутомата и наставља се рекурзивно са $(t+1)$-им битом излазног низа кључа.

сума	$x_t^2 x_t^3 x_t^4$	
Паран		Једначине
0	000	$x_t^2 = 0$ $x_t^3 = 0$ $x_t^4 = 0$
2	011 101 110	$x_t^2 \oplus x_t^3 \oplus x_t^4 = 0$ $(x_t^2 \oplus 1) \cdot (x_t^3 \oplus 1) = 0$
Непаран		
1	100 010 001	$x_t^2 \oplus x_t^3 \oplus x_t^4 = 1$ $x_t^2 \cdot x_t^3 = 0$
3	111	$x_t^2 = 1$ $x_t^3 = 1$ $x_t^4 = 1$

Табела 4.4 Формирање једначина скупа *S* приликом гранања у стаблу варијанти.

На слици 4.5 приказан је пример стабла претраге које се пролази у току напада. За пролазак кроз стабло користи се алгоритам претраживања са враћањем (енгл. backtracking). У сваком чвору стабла се додају нове једначине, тј. пут од корена до чвора представља систем једначина који је формиран до тог корака. Гране из чвора одговарају случајевима да ли у систем једначина иде једна или три једначине. Гранање се ради само из чворова којима одговара неодређени систем. Дакле, стабло претраге садржи варијанте система једначина.



Слика 4.5 Пример стабла претраге за дужину расположивог низа кључа $n = 8$

У поглављу 5 се описује програмска реализација напада.

Описани поступак личи на Флуереров алгоритам [2]. У питању је напад који унапређује неке претходне нападе на генератор E_0 и то тако што претпоставља садржај два краћа регистра и почетно стање коначног аутомата. Да би се одредио садржај преостала два $LFSR$-а, као и у нашој варијанти напада, прати се скуп линеарних једначина. Кад се добије некозистентан систем, та претпоставка се одбацује. Флуереров алгоритам смањује сложеност напада у односу на неке претходне нападе на 2^{85} корака. Међутим, овај напад је сложенији од описаног напада, јер се овде претпоставља почетно стање само једног $LFSR$. За разлику од Флуереровог напада, овде се користе и нелинеарне једначине система S (видети табелу 4.4).

5. Програмска реализација напада

Напад на шифру $\boldsymbol{E_0}$ је реализован у програмском језику $\boldsymbol{C}$ на укупно 797 линија кода, у развојном окружењу Code::Blocks 13.12. Функције `int proverisistem()` и `int solve()` су кључне у програму и биће описане у наставку програма.

5.1 Структура програма

Да би напад био могућ у реалном времену и због ефикаснијег развоја програма, смањене су оригиналне дужине регистара; нове дужине регистара су 5,7,9 и 11. Полиноми повратне спреге изабрани су из рада [3] тако да буду примитивни. Овим полиномима одговарају нове позиције извода за повратне спреге, видети табелу 5.1.

LFSR	Степен	Полином	Излазни бит	Дужина периода
$LFSR_1$	5	$t^5+t^3+t^2+t+1$	4	2^5-1
$LFSR_2$	7	$t^7+t^4+t^3+t^2+1$	4	2^7-1
$LFSR_3$	9	$t^9+t^6+t^5+t^3+1$	8	2^9-1
$LFSR_4$	11	$t^{11}+t^{10}+t^8+t+1$	8	$2^{11}-1$

Табела 5.1 Полиноми повратне спреге четири краћа регистра

Напад на умањену верзију алгоритма E_0 суштински је исти као напад на оригиналну верзију. Као што је речено, програмска реализација напада пролази кроз све могуће претпоставке о почетном стању регистра $LFSR_1$ и коначног аутомата st_0, укупно $5 + 4$ битова. За сваку претпоставку формира се стабло чији сваки пут од корена до неког листа одговара бинарном систему линеарних једначина по 27 непознатих. Када се приликом обиласка стабла дође до контрадикције, онда се одустаје од тренутног система, односно чвора у стаблу. Ако се приликом обиласка стабла установи да ни једном чвору не одговара конзистантан систем једначина, прелази се на нове вредности за $LFSR_1$ и st_0. Са поступком се наставља док се за неку претпоставку не пронађе чвор у стаблу коме одговара једнозначно решив

система са 27 непознатих. Решење система одређује почетна стања регистара $LFSR_2$, $LFSR_3$ и $LFSR_4$, а почетна претпоставка одређује почетно стање $LFSR_1$ и st_0, па је то тражено решење, тј. напад је успео. Пронађено решење се за сваки случај упоређује са задатим почетним стањем генератора.

У току рачунања се у сваком кораку t одређују вектори x_i, $i = 2,3,4$ са коефицијентима линеарних комбинација које изражавају садржаје регистара $LFSR_2$, $LFSR_3$ и $LFSR_4$ преко полазних садржаја тих регистара.

У главном програму (енг. main) на основу низа кључа добијеног симулацијом E_0 и почетног стања $LFSR_1$ и коначног аутомата, покреће се напад. Добијени почетни садржаји регистара 2, 3 и 4 треба да се сложе са оним који су убачени у симулацију E_0. Даље се ток програма који представља напад на E_0 предаје функцијама које решавају конкретне проблеме. Прва је функција **`int proverisistem()`**.

Псеудокод функције `int main()`

// t_{max} је дужина расположивог низа кључа

$x_2x[t_{max}, n_2]$; // коефицијенти линеарних комбинација

$x_3x[t_{max}, n_3]$;

$x_4x[t_{max}, n_4]$;

for $lfsr_1$=0 to $2^{n1} - 1$ **do** // пролаз кроз сва стања $lfsr_1$

  for st=0 to 15 **do** // пролаз кроз сва стања коначног аутомата

    ct[0]= st_0 ; // горња два бита коначног аутомата

    solve();

    if $ps_2 = 1$ then нађено је решење

    if $ps_2 \neq 1$ then решење није нађено

5.2 Функција `int proverisistem()`

Функција **`int proverisistem()`** решава систем линеарних једначина са бинарним коефицијентима и операцијом XOR (сабирање по модулу 2). Функција враћа 1 ако систем има јединствено решење, враћа 2 ако је систем неодређен и враћа 0 ако је систем контрадикторан. Ако функција враћа 1, она одређује и решење система, односно почетно стање регистара и коначног аутомата.

Функција добија систем једначина као матрицу са $n_2 + n_3 + n_4$ колоне, где су n_1, n_2, n_3 и n_4 дужине померачких регистара (у нашем случају 5, 7, 9 и 11 битова), које одговарају непознатима и додатном колоном за слободне коефицијенте. Свака колона одговара једном биту од $n_2 + n_3 + n_4$ непознатих битова почетног садржаја регистара $LFSR_2$, $LFSR_3$ и $LFSR_4$. Слободни коефицијенти су 0 или 1. Врста ове матрице коефицијената линеарних једначина добија се на начин описан у делу који описује рад $LFSR$-а и чине је коефицијенти линарних комбинација које изражавају $x_2[t]$, $x_3[t]$ и $x_4[t]$ преко непознатих битова који чине почетна стања $LFSR_2$, $LFSR_3$ и $LFSR_4$. Матрица може имати више врста, где свака врста одговара једној линеарној једначини добијеној при гранању стабла. Кад се изабере један од два случаја приликом гранања у алгоритму и дода се једна или више врста, онда се у том чвору стабла проверава да ли је дотадашњи систем једначина контрадикторан. За враћање и прелаз на нове случајеве приликом гранања коришћен је алгоритам претраге (енг. backtraking). У почетку, при мањем броју једначина, систем ће готово увек бити неодређен, па се онда у систем додају нове једначине.

Систем линеарних једначина решава се дијагонализацијом. На тај начин долази се до информације о каквом систему линеарних једначина је реч. Најпре се праве нуле испод главне дијагонале. Када се тако добије горња троугаона матрица, прелази се на прављење нула изнад главне дијагонале. У моменту када се испод и изнад главне дијагонале добију све нуле, долази се до информације да ли систем има јединствено решење, или је неодређен или је противуречан. Ако се у процесу дијагонализације добије врста у којој су сви елементи нуле а слободан коефицијент је један, онда је систем линеарних једначина противуречан. У случају да систем има мање једначина него

променљивих онда је он неодређен. У самом почетку претраге кад није прикупљено довољно једначина, систем је најчешће неодређен. У случају да систем има јединствено решење, у функцији **`int proverisistem()`** се конкретне вредности почетног стања регистара $LFSR_2$, $LFSR_3$ и $LFSR_4$ узимају из колоне слободних коефицијената.

5.3 Функција `int solve()`

Функција **`int solve()`** симулира пролазак кроз стабло, што је решено рекурзијом. Псеудокод алгоритма је дат у наставку поглавља. Стабло се даље грана кад је систем неодређен, и у том случају се додају нове једначине. Тражи се чвор у коме систем има јединствено решење, што се сваки пут проверава позивом функције **`int proverisistem()`**. Сваки чвор стабла одговара неком рекурзивном позиву.

У случају да је систем линарних једначина одређен, тј. када функција **`int proverisistem()`** врати 1, тада се у фукцији **`int solve()`** проверавају нелинеарне једначине. У оквиру фукције **`int solve()`** се нелинеарне једначине не решавају, него кад се реши систем линеарних једначина, проверава се да ли то решење задовољава и нелинеарне једначине. Даље је уведена и додатна провера слагања добијеног низа кључа из првог програма за првих $n_1 + n_2 + n_3 + n_4 + 4$ битова. На основу рада [1], уколико је скуп прикупљених једначина S конзистентан и $t > n_1 + n_2 + n_3 + n_4 + 4$, може се предпоставити да је добијено тачно иницијално стање генератора E_0. Ово се оправдава тиме да су $n_1 + n_2 + n_3 + n_4 + 4$ битова који чине унутрашње стање генератора E_0 једнозначно одређени излазним низом кључа генератора E_0 те дужине. Функција **`int solve()`** враћа 0 кад стабло не треба да се даље грана.

У случају кад је систем линеарних једначина неодређен, покрећу се нова гранања, односно рекурзивни позиви фукције **`int solve()`**. У сваком чвору стабла имамо једно гранање у зависности од парности, добијено из једначине 4.2, и још једно гранање по вишем биту парности, описано у кораку 1 основног напада. Тај други ниво гранања, тј. додатно гранање у оквиру једне вредности за парност, је означен у коду променљивом $sumabit[t]$. На основу вредности те променљиве ми знамо да ли се у систем линеарних једначина додају једна или три једначине, табела 4.4.

// ps - променљива која прати стање линерног дела система једначина

// ps1 - променљива која прати стање нелинерног дела система једначина

// ps2 - променљива која прати стање комплетног система једначина

```
proverisistem(); // провера претходног система једначина, резултат у ps
  if protivrečan sistem then return // крећемо се назад у стаблу претраге
  else if jednoznačno rešenje then
   provera nelinearnih jednačina
   provera da li generator E0 daje ubačeni niz ključa
   if tačno rešenje then
     ps2=1;
     izlaz iz svih rekurzija
   else return // крећемо се корак уназад у стаблу претраге
  else // претходни систем је неодређен, даље са гранањем у
       // стаблу претраге додавањем нових једначина у систем
    t1=t; // број употребљених чланова низа кључа
    bl1=bl; // тренутни број линеарних једначина у систему
    for sumabit=0 to 3 do // x2t+x3t+x4t,
                          // x2t, x3t и x4t су излазне вредности  из LFSR2-4
      dodati nove linearne jednačine u sistem
      ažurirati bl
      t= t+1;
      ažurirati novo stanje konačnog automata ct[t+1], ct[t]
      solve(); // рекурзивни позив функције, наставља се даље гранање
      if jedinstveno rešenje then  izlaz iz svih rekurzija
      else vraćanje starih vrednosti za t,bl
      return;
```

5.4 Тестирање програма

Исправност рада првог програма, који имплементира рад шифре E_0, проверена је помоћу тест вектора из текста [12]. На слици 5.1 је дат пример једног тест вектора, а на слици 5.2 исти тест вектор добијен првим програмом.

```
K'c4[0]  = 21   K'c4[1]  = 87   K'c4[2]  = F0   K'c4[3]  = 4A
K'c4[4]  = BA   K'c4[5]  = 90   K'c4[6]  = 31   K'c4[7]  = D0
K'c4[8]  = 78   K'c4[9]  = 0D   K'c4[10] = 4C   K'c4[11] = 53
K'c4[12] = E0   K'c4[13] = 15   K'c4[14] = 3A   K'c4[15] = 63

Addr4[0] = 2C   Addr4[1] = 7F   Addr4[2] = 94
Addr4[3] = 56   Addr4[4] = 0F   Addr4[5] = 1B

CL4[0]  = 5F   CL4[1]  = 1A   CL4[2]  = 00   CL4[3]  = 02
```

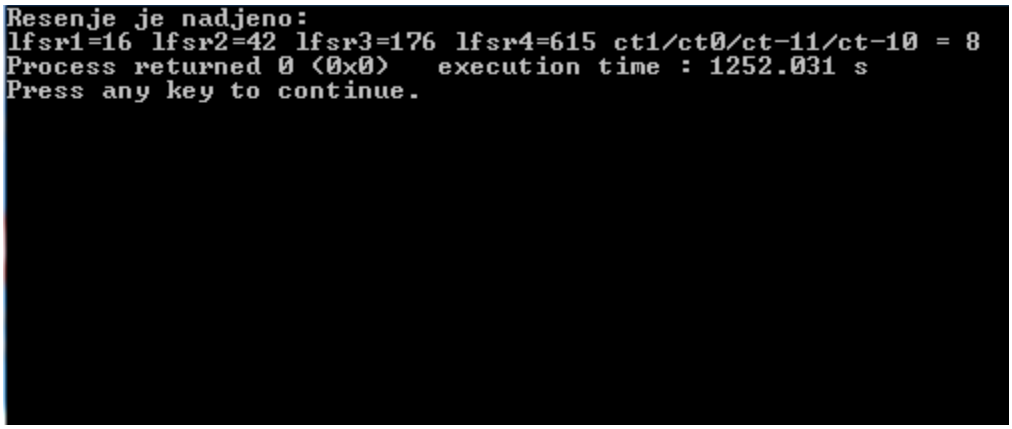
Слика 5.1 Пример тест вектора за шифру E_0.

```
i=209 lfsr1=2 lfsr2=11 lfsr3=fe lfsr4=13b x1t=0 x2t=0 x3t=1 x4t=0 zt=0 ct+1_ct=3
 ct_ct-1=14
i=210 lfsr1=5 lfsr2=22 lfsr3=1fd lfsr4=277 x1t=0 x2t=0 x3t=1 x4t=0 zt=1 ct+1_ct=
8 ct_ct-1=3
i=211 lfsr1=a lfsr2=45 lfsr3=1fa lfsr4=4ee x1t=1 x2t=0 x3t=1 x4t=1 zt=1 ct+1_ct=
2 ct_ct-1=8
z[0]=96
z[1]=42
z[2]=76
z[3]=15
z[4]=ba
z[5]=5a
z[6]=d3
z[7]=d6
z[8]=69
z[9]=a2
z[10]=cd
z[11]=50
z[12]=f5
z[13]=25
z[14]=d1
z[15]=bc
lfsr1=16 lfsr2=42 lfsr3=176 lfsr4=615 kljuc:
01110110001101101010101011010011001011010111111000001011101101010011011010101101
00111100100110111011001000001100001011010110 0Press any key to continue . . .
```

Слика 5.2 Резултат рада првог програма

Добијени кључ, као један од резултата рада првог програма, користимо као улаз за рад другог програма. Са скраћеним дужинама регистара успешан напад траје у просеку 30 минута. У једној од ранијих верзија напада на шифру E_0, која је садржала више позива функцији `printf()`, како би се пратиле вредности међурезултата, напад је у просеку трајао скоро 2 сата.

Слика 5.3 приказује резултат рада програма који имплементира напад на E_0.



Слика 5.3 Резултат рада другог програма

Приликом разматрања колико је напад скалабилан, тј. колико би се извршавао напад уколико би се дужине регистара повећале два пута, долазимо до одговора који је у вези са сложеношћу разбијања алгоритма. Сложеност алгоритма разбијања суштински зависи од броја варијанти које се проверавају, а то је $O(2^{|R_1|})$. Сложеност алгоритма даље зависи од сложености претраге за прикупљање једначина, коју је иначе тешко проценити. Сложеност алгоритма разбијања такође зависи од утврђивања да ли је систем неконзистентан или једнозначно решив. Сложеност утврђивања неконзистентности слична је сложености решавања система. Сваки пронађени систем линеарних једначина се решава алгоритмом сложености $O((|R_1| + |R_2| + |R_3|)^3)$.

6. Закључак

Програмери који се баве овом врстом проблематике труде се да пронађу слабости алгоритама који се користе за шифровање. Са друге стране, у циљу побољшања безбедности на мрежи (интернету) улажу се напори како би се креирали још бољи алгоритми за шифровање. У питању је надметање које ће увек постојати.

У овом раду је приказан и реализован напад на проточну шифру E_0 која се примењује у оквиру стандарда Блутут. Напад се извршава за умањену верзију алгоритма E_0.

Наведене су и коментарисане референце које указују на слабости Блутут стандарда као и различите нападе на E_0 [5]. Као што је често случај са криптографским материјалима, постоји одређен ниво тајновитости и мањак конкретних информација у истим, што је до извесне мере отежало реализацију програма написаног у програмском језику C.

Даљи рад би подразумевао модификације и оптимизацију напада на шифру E_0, како би расположиви ресурси били искоришћени на бољи начин. Развијени програм се може побољшати памћењем међурезултата рада методе дијагонализације матрице система линеарних једначина, како се дијагонализација не би радила сваки пут кад се промени систем линеарних једначина.

Литература

[1] C. De Cannie`re, T. Johansson, and B. Preneel, Cryptanalysis of the Bluetooth Stream Cipher, COSIC internal report, 15 pages, 2001.

[2] S. Fluhrer and S. Lucks. Analysis of the E0 Encryption System. In 8th Annual International Workshop of Selected Areas in Cryptography (SAC) 2001, Lecture Notes in Computer Science, LNCS 2259, pp. 38–48. Springer-Verlag, 2001.

[3] M. Živković, A Table of Primitive Binary Polynomials, Mathematics of Computation Vol. 63, No. 207 (Jul 1994), pp. 301-306

[4] Bluetooth Specification. https://www.bluetooth.org/spec/

[5] Christian Gehrmann , Joakim Persson, Ben Smeets, Bluetooth security, Artech House, *Computer Security Library*, 2004.

[6] A. S. Tanenbaum, D. J. Wetherall, Computer Networks, 5th ed. Prentice Hall, 2020.

[7] M. Jakobsson and S. Wetzel. Security Weaknesses in Bluetooth. In Proceedings of the Cryptographer's Track at the RSA Conference (CT–RSA '01), Lecture Notes in Computer Science, LNCS 2020, pages 176–191. Springer-Verlag, 2001.

[8] Lars R. Knudsen, A Key-schedule weakness in SAFER K-64, D. Coppersmith (Ed.): Advances in Cryptology - CRYPT0 '95, LNCS 963, 1995., pp. 274-286.

[9] Yi Lu, Serge Vaudenay, Faster Correlation Attack on Bluetooth Keystream Generator E0, Advances in Cryptology-CRYPTO 2004. pp. 407-425

[10] Bruce Schneier, Applied Cryptography, Second Edition: Protocols, Algorthms and Source code in C, *John Wiley & Sons, Inc*, 1996.

[11] Dave Singelée, Bart Preneel, Security Overview of Bluetooth, COSIC Internal Report, June, 2004.

[12] Specification of the Bluetooth System, Specification volume 1, Version 1.1, 2001.