Великотърновски университет „Св. Св. Кирил и Методий”
Педагогически факултет
Направление „Математика и Информатика”
Катедра “Информационни технологии”

ДИПЛОМНА РАБОТА

На тема:

Анализ, симулация и прилагане на криптография по елиптична крива при изпълнение в безжичните сензорни мрежи

Дипломант:
Душан Илия Биков
Спец. „Информатика.
Информационни Системи”
фак. №: МПИ – 10609р

Научен ръководител:
доц. дмн Стефка Буюклиева
Ръководител катедра:
доц. д-р Емилия Тодорова

Велико Търново
2011 г.

Съдържание

Резюме

В дипломната работа ще бъде представен анализ на сигурността и заплахите на безжичните сензорни мрежи и механизми за осигуряване на поверително и надеждна комуникация между възлите. Безжичните сензорни мрежи са специални мрежи, които се състоят от малки сензорни възли с ограничени ресурси и една или повече базови станции. Тези мрежи се използват за следене на определена цел, събиране на данни и изпращането на събраните данни до останалата част от системата. Сензорските възли обикновено са малки и евтини устройства, които се поставят в среди без надзор като нямат физическа защита, което ги прави лесно достъпни за злонамерни атаки. Днес, безжични сензорни мрежи все повече се използват за различни приложения, като наблюдение на животни, спешна медицинска грижа, следене на превозни средства, военни операции и др.

Безопасността или устойчивостта на атаки във всяка мрежа е от фундаментално значение. Във всяка безжична сензорна мрежа съществуват предизвикателства, като контрол на достъпа, надеждност, идентификация, цялост, които трябва да се гарантират. Поради ограниченията от живота на батерията, възлите най-често се намират в "състояние на сън" или са в състояние на обработка на данните. За тази цел са разработени редица криптографски системи, които са подходящи за употреба в среди с ограничени ресурси. Един от тях е криптография на основата на елиптична крива, която принадлежи към групата на криптографии с публичен ключ, базирани на елиптичните криви (ECC – Elliptic Curve Cryptography). Криптирането чрез елиптични криви предлагат независимо един от друг Neal Kobiltz и V.S.Miller. Публичният ключ е точка от елиптичната крива, докато частния ключ е случайно естествено число. Надеждността на ECC е в трудността да се реши Проблема за Дискретния Логаритъм (DLP - Discrete Logarithm Problem) върху елиптична крива, т.е. да се намери броят k, така че за дадени точки P и Q от елиптичната крива, Q = k*P = P+P+…+P. Казваме, че k е логаритъм от Q с основа P. Този проблем е практически нерешим за достатъчно големи стойности на k. Най-голямото предимство на ECC е, че използва много по-малки ключове. Така че, 160-битов ECC ключ предлага същото ниво на сигурност, както и 1024-битов RSA ключ, а 384-битов ECC ключ съответства на 7680-битов RSA ключ. Тези подобрения на производителността са от ключово значение в безжични среди, където мощността, паметта и живота на батерията са ограничени. Малките ключове водят до по-бързи изчисления и внедряване, съответно по-малкият брой операции води до по-малко използване на мощност, което пък от друга страна се постига с по-слаб процесор на хардуера.

Сензорните възли могат да бъдат далеч от мрежовата инфраструктура и далеч от хората. Поради големите цени за поддръжка на отдалечени възли е необходимо мрежите да бъдат самостоятелни, да са по-трайни и да запазват енергията за по-дълго. Възлите в

мрежата не знаят своите съседи, така че едно тривиално решение би било, когато всички възли биха имали един генерален ключ, за да се използват за сигурна комуникация от край-до-край. Затова безжичните сензорни мрежи въвеждат договаряне за ключа на симетрична криптосистема, използвана за криптиране и идентификация. С цел да се осигури идентификация, поверителност и сигурност са предложени повече съответни архитектури, платформи за безжични сензорни мрежи и операционни системи. За тази цел са разработени различни алгоритми, които се тестват на няколко сензорни платформи, като MICA2, MICAz, Imote2, TelsosB и Tmote Sky. За работа с безжична сензорна мрежа се използва вградена операционна система (embedded OS), чиито възможности са ограничени от хардуерните ресурси и изискванията на различните приложения, които се използват. Най-известната операционна система е TinyOS, след това Contiki, MANTIS, Nano-RK, LiteOS, SenOS и др.

Тук ще бъде разгледана математическата основа на елиптичните криви над крайни полета и операции с групите. Основен въпрос в този тип криптография е изборът на параметрите на кривата, тъй като техен правилен избор позволява по-голяма гъвкавост на устройствата с ограничени ресурси. Също така ще бъдат представени и различни алгоритми, които се използват за прилагане на аритметиката на крайните полета и аритметиката на точките, като събиране, умножение, инверсия и редукция. Ще стане дума за Проблема за Дискретния Логаритъм (или просто DLP) и нейното приложение в Diffie-Hellman протокола за обмен на ключовете (ECDH) и алгоритъма за дигитален подпис (DSA) според препоръките на NIST, по що ще се извършат съответни симулации.

След това ще бъдат приложени резултатите от изведените симулациите с различни параметри. Симулациите ще бъдат на база на сензорната платформа MICA2. Поради това че платформите имат ограничени ресурси, нужно е да се оптимизира кода за да се получи по-малко използване на ROM и RAM и енергия, което ще осигури повече памет за изпълнение на главните приложения и консумиране на по-малко енергия и мощност. Сервисите за сигурност, както управление с ключове и автентификация са критични за сигурна комуникация в безжичните сензорни мрежи. Поради това ще се изведе серия на симулации за да се оцени изпълнението на криптографията по елиптична крива и нужната енергия за установяване на ключ. Ще се извърши анализ и оценка при прилагане на криптография по елиптична крива, при използване на разни параметри и техники за оптимизация. Също така ще се сравнява приложение което използва елиптична крива с приложение което използва симетрична криптография, където ще се установи изискването на енергия.

Първа глава

Обзор на основните понятия за безжичните сензорни мрежи и заплахите за тяхната сигурност

1.1 Въведение в безжичните сензорни мрежи

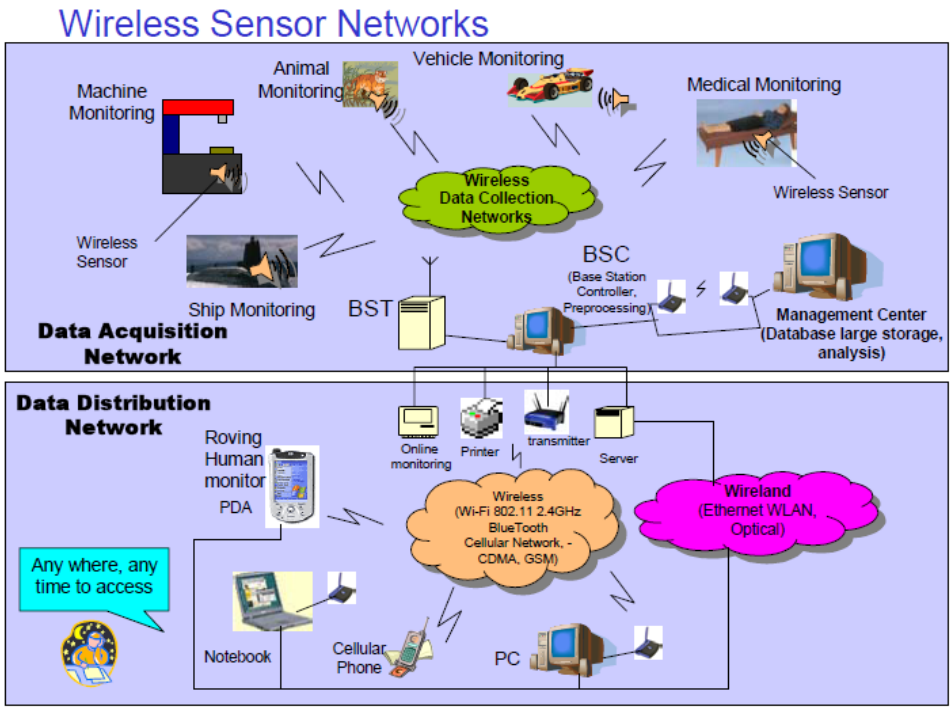
Структурния интегритет на сензорите механизми в околната среда, заедно с ефикасното изпращане на данните от сензорите може да осигури огромна полза за обществото. Могат да се използват както за военни така и за цивилни цели и това: локален наблюдение на състоянието и околностите, запазване на природни ресурси, подобряване на производството и продуктивността, както и за домашна сигурност [1]. Въпреки това ограничението за широко използване на сензорите остава поради тяхната технология, инсталация, разходите за поддръжка.

Безжичните сензорни мрежи съчиняват отреден брой на много - малки пространни автономни, компютри, комуникационни и чувствителни регистрационни устройства, с ограничена енергия, сметачки способности и памет, които заедно съдействат и са част от голяма и чувствителна задача.

Идеална безжична сензорна мрежа е приспособима, където компонентите консумират много малко електрическа енергия, използва малка памет, имат съответно интелигентен софтуер, способност за бързо сдобиване с данни, те са надеждни и прецизни в дълъг период от време, струват евтино за купуване и инсталиране и да не изиска истинска поддръжка.

Избора на оптимална сензорна безжична комуникационна връзка, изиска познание за дефинирания проблем и съответното приложение. Живота на батерията, стъпката на актуализация на сензора, големината и дизайна са главни въпроси за обсъждане. Примери на сензори с малка стъпка на пращане на данни включват сензори които измерват температура, влажност, и т.н. Примери на сензори с висока стъпка на пращане на данни включват сензори които измерват забързване, вибрации и т.н.

Най-новите достигана доведоха до способност за интегриране на сензори, радио връзка, битова електроника на една интегрирана платка (integrated circuit - IC). Тези възможност ни дава мрежа от сензори на много ниска цена с способност да комуникират един със друг използвайки безжично протоколи за изпращане на данни с много малко мощност. Безжичната сензорна мрежа (WSN) обикновено използва базисна станция (base station или “gateway”) коя може да комуникира с голям брой на безжични сензори през радио връзка. Данните се събират на безжичният сензорен възел, се компресират и се пращат кон базисната станция директно или се използва друг възел за препращане на данните на базисната станция. Целта на тези секция е да осигури кратко въвеждане в сензорните безжични мрежи.



Фигура 1.

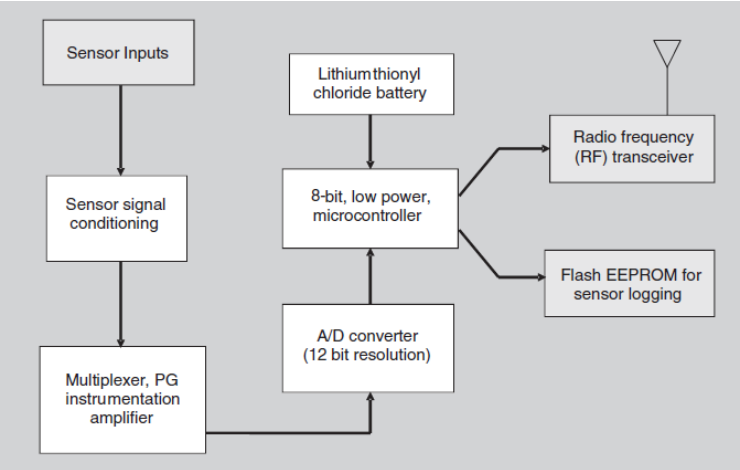
1.2. Индивидуална архитектура на безжичен сензорен възел

Функционалната блок диаграма на гъвкав безжичен сензорен възел е представена на фигура 2. Подхода с модулния дизайн осигурява гъвкава и многостранна платформа с коя може да се отговори на нуждите на широкия обсег на приложения [2]. На пример в зависимост от разпределението на сигнала на блока на състояния може да бъде препрограмиран или заменен. Това осигуря широк обсег на различни сензори който могат да се използват от безжичния възел. Също така може радио връзката да се промени по изискване на дадено приложение, или па по изискване на безжичния обсег и нуждата за двустранно комуникиране. Използването на flash паметта и осигуря на отдалечените възли да се сдобият с команди от базисната станция, или с събития (промяна на температура, влажност и т.н.) от един или повече входове на възела или пак с данни от други сензорни възли. Освен това вградения firmware може да се актуализира през безжичната мрежа. Микропроцесора има голям брой на функции включвайки:

1) ръководене с сбиране на данни от сензорите
2) регулиране на функциите и управление на мощността
3) свързване на данните на сензора с физическото ниво
4) управление с безжичният мрежов протокол

Ключова характеристика на всеки безжичен възел е минимизирането на мощността консумирана от система. Общо взето безжичната под система изиска най-много мощност. Поради това изгодно е пращане на данни през безжичната мрежа само когато е нужно. Този сензорно събитие водено от модела за събиране на данни, изиска алгоритъм кои ще

определя кога да праща данни кои се базират на събитие. Също така важно е да се минимизира консумирането на мощност в самия възел. Затова хардуера трябва да се дизайнера за да му позволява на микропроцесора да може да върши контрол на мощността на сигнала, сензора и сензорния сигнал.

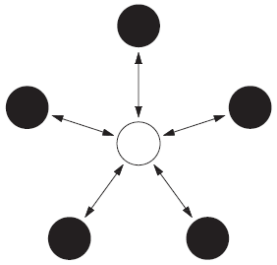


Фигура 2: Функционална блок диаграма на безжичен сензор

1.3. Топологии (topologies) на безжичните сензорни мрежи

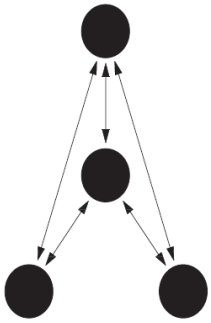
Съществуват голям брой на различни топологии (topologies) за радио комуникационните мрежи. Кратка дискусия за мрежовата топология кои важат за безжичните сензорни мрежи следва долу.

Star Network (Фигура 3.) е топология за комуникация къде една базова станция (base station) може да праща и /или приема съобщения до голям брой на отдалечени възли. Отдалечените възли може само да пращат или приемат съобщения от една базисна станция, при тях не е позволено да си изпращат съобщения помежду възлите. Предимство на тези тип на мрежи, е в тяхната простота и способност за задържане на консумирането на енергия на отдалечените възли на минимум. Тя също така позволява ниска комуникационна латентност между отдалечените възли и базисната станция. Недостатък на тази тип на мрежи е че базисната станция мора да бъде в радио обсега на всичките отдалечени индивидуални възли, и да е доволно по-силна от другите безжични мрежи (ако съществуват такива) поради това що управлението на мрежата зависи от базисната станция.


Фигура 3: Star мрежова топология

Mesh Network осигурява на всеки възел в мрежата да предава на кои да е друг възел в мрежата който е във радио предавателен обсег. Това осигурява комуникация известна като

multi-hop – това представлява възел които иска да прати съобщение до друг възел навън от радио комуникационния обсег, може да използва междинен възел кои ще го пренасочи съобщението до искания възел. Тези топология има предимства като redundancy и приспособимост (scalability).
Ако отделен възел се провали, възела може да комуникира с всеки друг работещ възел, и той може да го препрати съобщението до искания възел. Обсега на мрежата не е ограничен с обсега на отделните възли, и може да се разшири с додаване на нови възли в система. Недостатък на този тип на мрежи е в консумацията на енергия коя за възлите кои използват multi-hop комуникация е по-висока за разлика от възлите които не използват multi-hop, което го намалява живота на батерията. Когато броя на комуникационните скокове (hops) до дестинацията се увеличи времето изискано за предаване на съобщението се увеличава, особено ако е условие ниската мощност на работа на възела.



Фигура 3: Mesh network topology

Hybrid представлява топология помежду Star и Mesh Network. Hybrid Network осигуря стабилна, гъвкава и многостранна комуникационна мрежа, а поддържа състояние за минимално консумиране на енергия. При тези топология сензора с най-ниска мощност е неспособен да препрати съобщение. Това осигурява на разходи на минимална енергия да се поддържа. Определени възли в мрежата са способни за multi-hop, което осигуря предаване на съобщенията от възли с ниска мощност на останалите възли на мрежата. Общо взето възлите с multi-hop способност, са със по-висока мощност, и ако е възможно се включват на електрическата мрежа. Тези изпълнение на топология идва от мрежовия стандарт известен като ZigBee.

1.4. ISO/OSI моделът

Тези секция описва въвеждане в слоевете на ISO/OSI модела.

Физически слой (Physical Layer) го конвертира битовия поток от по високите слоеве, в подходящ сигнал за трансмисия. Честотата се избира в съгласие с лицензът. Безплатни честотни лицензи се наричат ISM (Industrial, Scientific and Medical) bands, в Европа е 868-868,6 MHz, а в Северна Америка и Австралия е 902-928 MHz, а 2400-2483,5 MHz е световно безплатен честотен band. Физическия слой дефинира разширение за типовете (FHSS, DSSS, CSS) и типови за използваната модулация, коя типично е BPSK или O-QPSK в безжичните сензорни мрежи.

Канален слой (Data Link Layer – DLL), главна отговорност е за мултиплексиране (multiplexing) на битовия поток, детекция на фреймовите на данните (date frame detection), medium access и контрола на грешки (error control). Също така е поделен на два под слоеве – higher Logical Link Control (LLC) и lower Medium Access Control (MAC). Подялбата на DLL на два под слоеве се изиска за приспособяване на логичните изискана за управление на пристъпа до споделения комуникационен медиум (въздуха). Критично за изпълнение на WSN е избора на MAC протокола. Съществуват много протоколи и не може да се посочи кой е най-добър, поради това че всеки се използва за различно приложение и има различни свойства. Приложенията може да имат различни критериума за изпълнения като закъснение (delay), fairness, throughput. Всеки WSN MAC протокол се опитва да осъществи една от най-важните нужди както що е съхранение на енергията.
Някои от MAC протоколите може да се разделят в три групи: low duty cycle, contention-based и schedule-based протокол. Sparse Topology and Energy Management (STEM), Sensor-MAC (S-MAC) и Mediation Device (MD) се протоколи кой припадат на първата група. В втората група припадат Carrier-Sense Multiplex Access (CSMA) и Power Aware Multiaccess with Signaling (PAMAS). Третата група се състои от следните протоколи Traffic-adaptive medium access protocol (TRAMA), Self-Organizing MAC for Sensors (SMACS).
Мрежов слой (Network Layer) е отговорен за доставянето на пакетите от край до край (от източника до получателят). На тези слой съществуват много протоколи за маршрутизация (routing) но само няколко от тях са приложими за безжичните сензорни мрежи. Могат да се поделят в три групи [6]: data centric, hierarchical и location-based. В първата група са: Sensor Protocols for Information via Negotiation (SPIN), Gradient-Based Routing (GBR), Constrained Anisotropic Diffusion Routing (CADR), COUGAR, Active Query forwarding In sensor Networks (ACQUIRE) и т.н. Втората група ги сдържи следните протоколи: Low Energy Adaptive Clustering Hierarchy (LEACH), Power-Efficient Gathering in Sensor Information Systems (PEGASIS), Threshold sensitive Energy Efficient sensor Network protocol (TEEN) и т.н. Трета група ги сдържи следните протоколи: Minimum Energy Communication Network (MECN), Geographic Adaptive Fidelity (GAF) и т.н.
Транспортен слой (Transport Layer), генерални има две главни функции: congestion control и loss recovery. Пример за тези протоколи са [3]: Congestion Detection and Avoidance (CODA), Event-to-Sink Reliable Transport (ESRT), Reliable Multisegment Transport (RMST), Pump Slowly, Fetch Quickly (PSFQ) GARUDA, Ad Hoc Transport Protocol (ATP).
Приложен слой (Application Layer) се дефинира и използва в безжичните сензорни мрежи. Някои от протоколите кои се използват на тези слой са: Sensor Management Protocol (SMP), Task Assignment and Data Advertisement Protocol (TADAP), Sensor Query и Data Dissemination Protocol (SQDDP). Тези протоколи за тяхна работа може да изискат протоколи от други слоеве.

1.5. Радио опции за физическия слой в безжичните сензорни мрежи

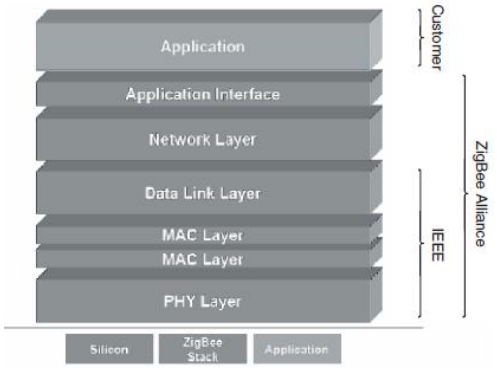
Физическия слой, определя оперативна честота, модулна схема и интерфейс на хардуера на радио системата. Съществуват много интегрирани платки с ниски радио мощности. Кон съответния избор за радио слоя на безжичните сензорни мрежи, можем да ги включим следните компании като Atmel, Microchip, Micrel, Melexis, и ChipCon. Ако е възможно полезно е използването на стандартен радио интерфейс. Това осигурява общи

характеристики (interoperability) между мрежи на различни компании. Най-известни радио стандарти са:
IEEE 802.11 е стандарт кой е за локални мрежи (local area networking) за релативно висок пренос на данни между компютъра или други устройства. Трансфера на данни се движи от 1 Mbps до над 50 Mbps. Типичния обсега е 300 стъпки с стандартна антена, обсега може значително да се увеличи с използване на насочена високо качествена антена. Две честоти и пряка секвенция с разширен спектъра на модулационна схема се на разполагане. Размера на трансфера на данни е висок за безжични сензорни апликации, поради изисканата на мощност, генерално се изключва неговото използване в сензорните безжични приложения.
Bluetooth (IEEE802.15.1 и .2). Bluetooth е частна мрежа (personal area network - PAN) стандарт кой е с по-малка мощност от 802.11. Първоначално е проектиран да служи на приложения за пренос на данни от компютъра до някои отделено устройство на пример GSM, или PDA. Bluetooth използва топология звезда (star) коя поддържа до седем отдалечени възли в комуникирането с една базисна станция. Докато някои компании имат изградено безжични сензори базирани на Bluetooth, тя не е широко прихваната поради ограниченията на протокола Bluetooth, кой включва:
1) Релативно висока мощност за къс преносен обсег.
2) Устройството изиска много време за да се синхронизира с мрежата, кога се връща от състояние sleep, с кое се увеличава консумирането на енергия.
3) Малак брой на възли в мрежата (<=7 възли).
4) Medium access controller (MAC) слоя е премного комплексен, когато се сравнява с изисканата за безжични сензорни приложения.
IEEE 802.15.4. стандарта е специално проектиран за изисканата на безжичните сензорни приложения. Стандарта е много гъвкав, специфициран за повече размери на пренос и повече честоти за трансмисия.
Изисканата на мощност се умерено ниски, хардуера е проектиран да осигури радио платката да отива в състояние на сън, с което се намалява изискането на мощност на минимум. Въпреки това когато възела се събужда от състоянието на сън, се постига бърза синхронизация с мрежата. Този осигуря много ниско напояване с енергия, поради това че радио платката е периодично изключена. Стандарта ги има следните характеристики:

1) Честота за трансмисия, 868 MHz/902-928 MHz/2.48-2.5 GHz.
2) Стъпки на пренос от 20 kbps (868 MHz Band) 40 Kbps (902 MHz Band) и 250 kbps (2.4 GHz Band).
3) Поддръжка на star and peer-to-peer (mesh) мрежови връзки.
4) Стандартна осигуря изборна опция за използване на AES-128 сигурност за шифроване на предаваните данни.
5) Индикация на качествена връзка коя се използва за multi-hop mesh мрежовия алгоритъм.
6) Използване на direct sequence spread spectrum (DSSS) за подробна комуникация с данни.

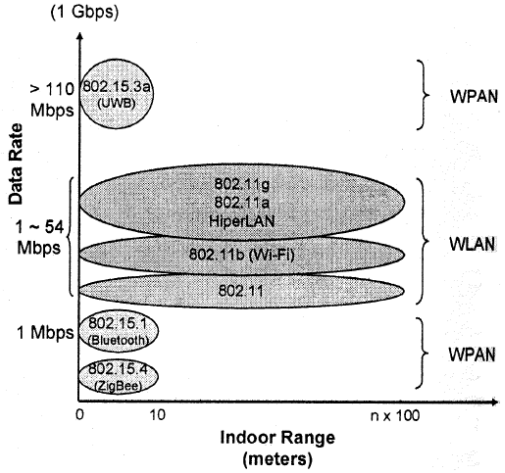
очаква се че от споменатите стандарти, IEEE 802.15.4 да стане общо приети за безжични сензори и приложения. 2.4-GHz band ще се използва широко, като основен световно известен безплатен лицензиран band.

ZigBee™ Alliance е асоциация на компании кои работят заедно за да се осигури сигурен, ефективен, ниско мощен система за безжично мрежово наглеждане следене и контрол на продукти на основание на отворен световен стандарт. ZigBee е специфицирано на IEEE 802.15.4 като физическия и MAC слоя се опитва да се стандартизират. Също така сервисите са съвместими с IEEE802.15.4 има черти както спецификациите на IEEE802.11. Мрежовата ZigBee спецификация е ратифицирана в 2004 година, поддържа звезда (srat) и Hybrid Star – Mesh Network. Като що може да се види на фигура 4, ZigBee поддържа IEEE802.15.4 спецификация и се разширява на мрежовата спецификация и приложният интерфейс.



Фигура 4: ZigBee stack

IEEE1451.5 стандарт е специфициран за комуникационна архитектура коя съответства с безжичните сензорни мрежи. В момента, IEEE802.15.4 физическия слой е избран за безжичен комуникационен интерфейс, се дефинира сензорния интерфейс.



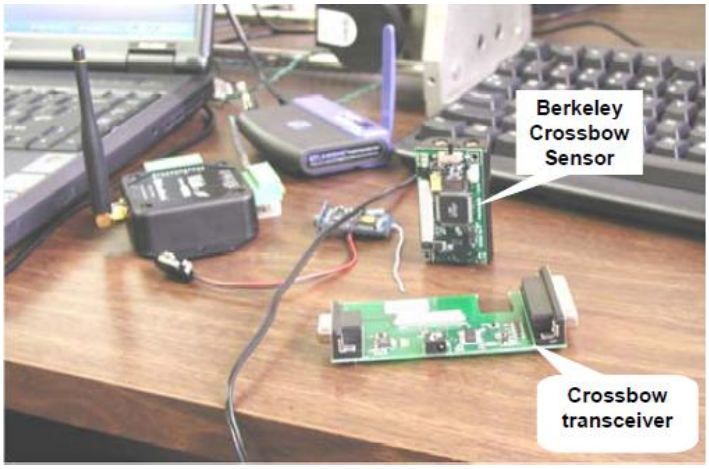
Фигура 5. Безжични мрежови стандарта

1.6. Комерсиално достъпни безжични сензорни системи

Съществуват много комерсиално достъпни безжични комуникационни възли включително Lynx Technologies, и различни Bluetooth устройства, включвайки и Casira устройство от Cambridge Silicon Radio, CSR.

Crossbow Berkeley Motes представляват гъвкави сензорни мрежови устройства на пазара които се използват с цел за проектиране на прототипи. Crossbow (http://www.xbow.com/)

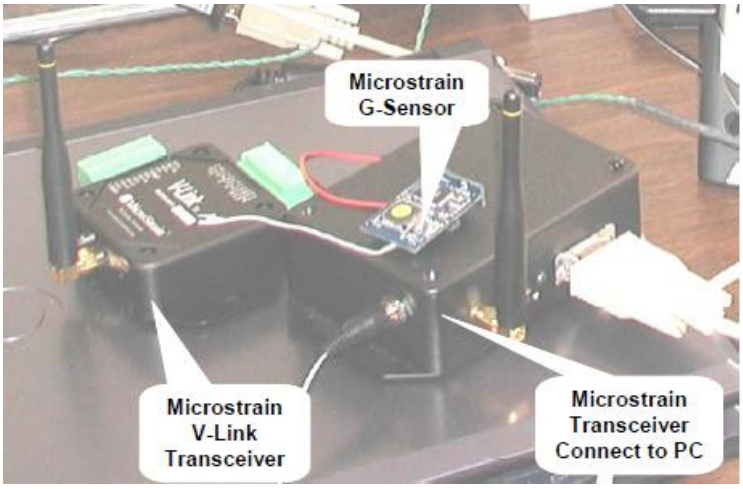
изработва три Mote processor radio module families – MICA [MPR300] (първата генерация), MICA2 [MPR400] и MICA2-DOT [MPR500] (втората генерация). Възлите идват с инсталирани сензора които може да измерва: температура, светлина, звук (Microphone), ускорение / Seismic и Magnetic. Тези сензори са особено подходящи за изграждане на мрежи за наблюдение на персонал, возила и т.н. Различни сензори могат да се инсталират ако се желае. Ниската мощност и малката физическа големина позволява да се поставят почти навсякъде. Бидейки всичките сензорни възли в мрежата може да действат като базисни станции, мрежата може самостоятелно да се конфигурира и тя има multi-hop routing опция. Оперативната честота е ISM band, 916Mhz или 433 MHz, с проток на данни от 40 KBits/sek. и обсег от 30 стъпки до 100 стъпки. Всеки възел има процесор с малка мощност с честота от 4MHz, flash памет със 128 Kbytes, и SRAM и EEPROM от 4KB. Оперативния система е Tiny-OS, а tiny micro-threading дистрибутивен операционна система развита от UC Berkeley, с NES-C (Nested C) език на изходния код (сходен с C). Инсталирането на тези типове устройства изиска доста програмиране.



Фигура 6. Crossbow Berkeley Motes

Microstrain's X-Link Measurement System (http://www.microstrain.com/) представлява един от най-лесните за използване, работа и програмиране. Честота коя се използва е на 916 MHz, коя в US е безплатна лицензa ISM band. Всеки сензорен възел е много канален със максимална подържа за 8 сензора, кой се поддържат от един безжичен възел. Съществуват три вида на сензорни възли - S-link (strain gauge), G-link (accelerometer) и V-link (поддържа всеки сензор кой генерира разлика на волтажа - supports any sensors generating voltage differences). Сензорния възел има pre-programmed EPROM, па голяма част от програмирането на потребителя не е нужно. Пространството за съхранение на данни е 2MB. Сензорните възли използват 3,6-Волт (lithium ion) вътрешна батерия (за пълнене се използва външна батерия от 9V коя се поддържа). Една базисна станция адресира повече възли. Всеки възел има уникална 16-битна адреса, така че е възможно да адресира максимум от 2^{16} възли. RF връзката между базисната станция и възлите е двупосочна и сензорния възел има програмирано съхранение на данни с малка честота. RF

връзката има 30 метри обсег с 19200 baud rate. Серийната връзка PC-232 между базисната станция и терминала (PC) е с baud rate 38400. LabVIEW интерфейса се подържа.



Фигура 7. Microstrain Wireless Sensors

1.7 Ограничени ресурси

Недостатък на сензорните мрежи е това че са с ограничени ресурси. Изпълнението на приложенията без разлика дали е за сигурна комуникация, основни действия в мрежата, следене, измерване и т.н. изискат известно количество от ресурси както памет за данни, програмна памет, ограничена мощност на процесора и енергия за съхраняване на сензора. Тези ресурси са ограничени и лимитирани в ”tiny” безжичните сензори.

- *Лимитирана програмна памет и памет за данни* (**Limited Memory and Storage Space**). Сензорните възли „tiny device” имат малка памет за данни и малка програмна памет. Кога се изиска да се изгради ефективен механизъм за сигурност, необходимо е да се лимитира големината на кода на алгоритъмът. На пример един тип на съвместим сензорен възел (TelosB) е 16-bit, 8 MHz RISC CPU със само 10K RAM, 48K програмна памет, и 1024K flash памет [7]. Поради тези ограничения, приложенията за сензори требва също да се лимитирани и малки. Тоталното пространство за TinyOS (стандартна операционна система за безжични сензорни мрежи), е приблизително 4K [8], а ядрото заема само 178 байта.
- *Лимитиране електрическа мощност* (**Power Limitation Energy)** представлява по-голямо ограничение за способностите на безжичните сензорни мрежи. Приема се за сензорен възел които е поставен в сензорната мрежа, не можем лесно да го сменим (висока оперативна цена) или презареждане. Поради това презареждане на батерията се извършва когато требва да се продължи живота на индивидуален сензор или на цялата сензорна мрежа. Когато се прилагат криптографски функции или протоколи в сензорните възли, енергийните нужди требва да се вземат в предвид. Когато включваме модул за сигурност в сензорния възел, ние се нуждаем тези модул да не влияе на живота на сензора относно на батерията. Консумирането на екстра енергия от сензорния възел се явява при модула за сигурност когато се

извършват функции за сигурност (шифроване (encryption), де шифроване (decryption), подписване на данните (signing data), верификация на подписа (verifying signatures), също така енергия се изискана за предаване на данните, вектора за инициализация кой е нужен за шифроване / де шифроване, изиска енергия за да се зачуват параметрите за сигурност.

Съществуват голям брой на стратегии кои могат да се използват за редуциране на просека на консумиране на енергия от радио връзката, кои включват:

■ Редуциране на размера на предаваните данни с помощ на компресия, редукция или избор.
■ Нисък приемно предавателно устройство с дълг цикъл и честота за пренос на данни.
■ Редуциране на frame overhead.
■ Осъществяване на строг механизъм за управление (power-down and sleep modes).
■ Осъществяване на стратегия за управление при осъществяване на някое събитие, само когато се случва някое събитие.

Стратегия за редуциране на консумация на мощността в сензора, включва:

■ Се включва сензора само когато се вземат мостри.
■ Се включва сигнала при условие когато се вземат мостри от сензора.
■ Се взема мостра когато ще се случи някое събитие.
■ Минимална стъпка на мострите от сензора за нуждите на приложението.

Други стратегии за редуциране на консумация на енергия се:

- изолиране и изключване на индивидуални елементи на устройството
- бързо събуждане и започване на обработката
- максимално оптимизация и минимизиране на действията
- завръщане в сън след извършване на някое действие.

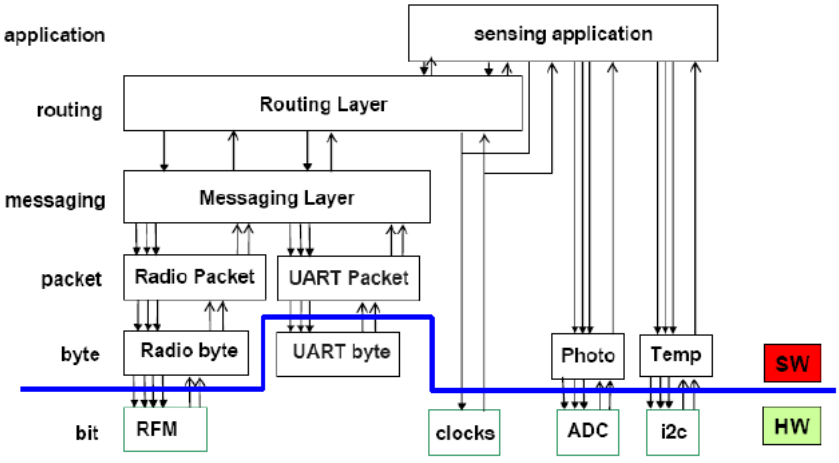
- *Лимитирана мощност на процесора*. Поради специфичността при проектиране на сензорни възли, а вземайки ги в предвид другите ограничения на ресурси, сензорния възел има на разположение процесор с малка мощност (от 8 MHz до 16 MHz). Поради това що процесора ги извършва всичките изчисления от него зависи нужното време за извършване на функциите в сензорния възел, той пряко влияе на консумирането на енергия.
- *Лимитиран радио обсег*. Друго ограничение е лимитирания радио обсег които зависи от стандарта които се използва на физически слой. От стандарта зависи и радио обсега които е между 30 до 100 стъпки, с използване на специални техники го увеличават обсега а с това и цената. В зависимост от нуждите за изграждане на безжични сензорни мрежи тези проблем се решава със осигурените техники, механизми кои се осигуряват в сензорните безжични мрежи.

1.8 TinyOS (Операционна система за безжични сензорни мрежи)

TinyOS е оперативна система за ниско мощни, гъвкави безжични вградени (embedded) сензорни мрежи за поддръжане и координация на оперативните нужди в сензорните мрежи, при което се изиска минимални хардуер (ресурси ROM, RAM). Две видни характеристики се това че е отворен код (open-source), и е модулно базирана рамка (framework). Поради това че е с отворен код, програмерот може да чете, распространява, и да го модифицира изворния код. Може да се подобрява, да се поправат грешките и да се иследва. Тяхна особина е че е компонентно – базирана рамка (framework) за модулна архитектура. Компонентите са статично свързани по що се осигурява бързо развитие и прилагане. Също така е повече кратно използваем код. Основните елементи са мрежова комуникация и управление на захранване.

TinyOS е компонентно базиран програмен модел, класифициран с езика NesC [24], диалект на C. TinyOS не е операционна система в класическия смисъл, тя е програмна рамка (framework) за вградени (embedded) системи и набор от компоненти, които позволяват изграждането на приложно-специфициран OS за всяко приложение. Типично приложение е околу 15K размер, на което базисната OS е около 400 байта, най-голямото приложение, е система за заявки с база данни, от около 64K байта.

TinyOs е операционна система, както е посочено по-горе, така че следващият въпрос е има ли ISO-OSI модел? Отговорът е да, и не. Да, защото има сходна структура и не защото не я следва структурата стрикно. Тези устройства, не разполагат с достатъчно памет, за да се приспособи на 7 слойния дизайн. На фигурата е покажан дизайна:



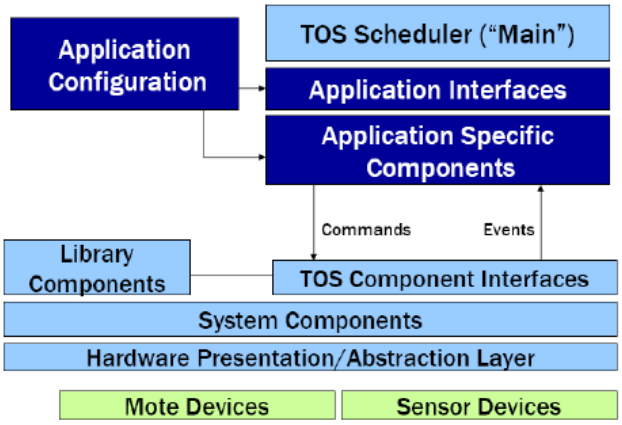
Фигура 8: TinyOs версия на ISO Модела

1.8.1 Архитектура:

Елементи на TinyOS са следните:

1. Единствен споделен стек
2. Нема ядро в традиционална смисла
3. Няма раководене с процесите / памета
4. Няма виртуална памет
5. Няма динамично моментално распределение на памета
6. Използват се глобални променливи за запазване на единството
7. Използва указатели за спестване на програмна памет

Съдържанието на структура е следното:



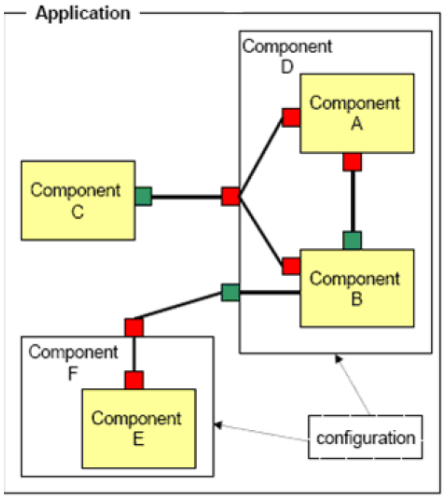
Фигура 9: TinyOs Структура на Слоевете

Модел на памета в TinyOs. TinyOs има моментално распределение на памета. Също така компонентите са статически свързани за време на компилирането, за да се определи изискания размер. Използването на локална променлива е лимитирано и кога се използва тя се записва в стекот. Повече се изпозват глобални променливи со що се избегнуват указателите с което се запазва памет.

Модулна структура на TinyOs:
Както бе споменато преди TinyOs има модулна структура. Има два вида компоненти:

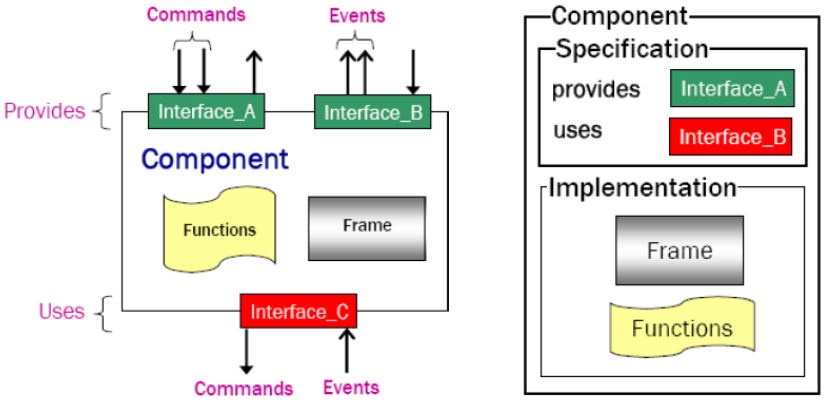
1. Модул: Компонент, написан с код
2. Конфигурация: Компонентно свързване

Компонентите са свързани заедно в кофигурациона единка което е образ на всяко приложене.



Фигура 10: Конфигурационно свързване

Зеления интерфейс (interface_A) осигуря възрзката, а црвения интерфейс (interface_C) използва тези връзка. Всеки компонент на TinyOs има рамка(frame), функции (function) и интерфейси (interfaces). Рамката поддържа стъпка на вътрешното / инициялно състояние. Състои се от глобалните променливи и други фиксирани структури. Функцията го носи кода, той има команди, събития и задачи, които са написани на диялект на езика C наречен NesC. Командите са входна точка, а събитията действат като извикваща функция. Параметрите на входната точка се предават на стека когато се връща статус. Интерфейса осигурява връзка. Се осигурява интерфейсна граница за компонентите. На следващата диаграма е посочено по-добре:



Фигура 11: NesC component

1.8.2 NesC

Езикът NesC е предназначена предимно за вградени системи (embedded systems), както сензорните мрежи. NesC подкрепя и е съвместим с TinyOS модел, както механизъм за структуриране, именуване и свързване на софтуерните компоненти в стабилна гъвкава вградена мрежова системи. Основната цел е да се позволи програмиране и проектиране на

компоненти които могат лесно да се съставът в пълен, съвпадаща система, коя осигуря широк контрол при компилирането.

NesC е разширение на програмния език C, проектиран да осъществи структурирана концепция и изпълнителен модел на TinyOS. Основната концепция на NesC е: отделяне на конструкцията и композицията, спецификация на отношението на набора на интерфейси.

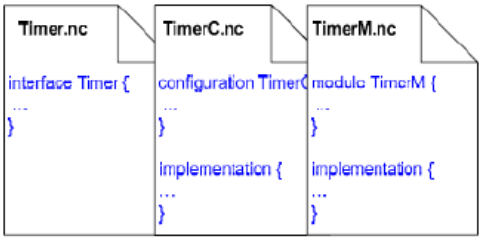
Приложението на NesC се състои от една или повече компоненти, свързани заедно да формират едно изпълнение. Тези компоненти осигурят и потребителски интерфейс. Тези интерфейс представлява точка на достъп до компонентите и са двупосочни. Интерфейса декларира набор от функции които се наричат команди които интерфейсния доставчик требва да ги осъществи и друг набор на функции наречени събития, които е потребителския интерфейс треба да ги осъществи. За да се извика команда в един интерфейс, треба да се осъществи събитие на тези интерфейс. Един компонент, може да използва или осигуря множество интерфейси и множество инстанции на един интерфейс. Интерфейсите могат да се осигурят или използват от компонентите. Осигурените интерфейси са предназначени да представляват функционалността на компонентите кои ги ползват потребителите, потребителския интерфейс я представлява функционалността на компонентите за извършване на нека работа.

Има два вида на компоненти в NesC: модули и конфигурации. Модула го осигуря кода на приложението, изпълнението на един или на повече интерфейси. Конфигурациите се използват сглобяване на други компоненти заедно, свързвайки интерфейса използван от компонентите с интерфейси представени от други. Всяко приложение на NesC е описано с високо ниво на конфигурация.

Повече за NesC. Кода е написан на NesC и изхода е ”C” програмен файл кой е компилиран и съединен със използване на gnu-gcc алатка за специфичен възел. Пред процеса (preprocessor) извършва приспособяване. Разширението на NesC файлът е —.nc . **C** се използва за конфигурация. **C** прави разлика между интерфейсите и компонентите кой ги осигурява.

Фигура 12: Конфигурация

M е за е за модулите и посочва дали един компонент сдържи модул и конфигурация.



Фигура 13:Конфигурация и модули

Програмиране с TinyOS. Приложенията за TinyOS са написани на NesC, диалект на езика за програмиране C, който е оптимизиран за ограничената памет на сензорните възли. Негови допълнителни средства основно са под форма на Java и шел (shell) скрипти. Приложенията за TinyOS са изградени от софтуерни компоненти, някои от които представляват хардуерни абстракции. Компонентите са свързани чрез интерфейси.

TinyOS предвижда интерфейси и компоненти за общи абстракции, като пакетна комуникация, маршрутизиране, отчитане, задействане и съхранение. TinyOS е напълно без блокиране, има един стек. Това позволява да поддържа висока съвместимост с единствения стек, което изиска записване на много малки събития. TinyOS предвижда задачи които са подобни на Deferred Procedure Call. Компонентите на TinyOS могат да назначат задача, която според списъка на OS може да се реализира по късно. Кода на TinyOS статично е свързан с програмния код, и се компилира цифрово, с използване на потребителския инструмент GNU. Асоциирано удобство е че услугите се осигурят от пълна платформа за развитие и работа с TinyOS.

1.8.3 TOSSIM

TinyOS е програмна рамка за вградени системи и е набор на компоненти кои осигуряват изграждане на специфични OS приложения. TOSSIM симулира TinyOS мрежа на битово ниво, като използва компоненти на TinyOS за реализация, почти идентични с mica 40Kbit RFM базиран стек [6]. TOSSIM предвижда две радио-модели: simple и lossy.

В TOSSIM мрежата, сигнала е единица или нула. Всички сигнали са с еднаква сила и колозията се моделира като логическо или - не е отказ. Това означава, че разстоянието не влияе върху силата на сигнала, а ако възела Б е много близо до възела А, сигнала от много по-далечния възела С е с съща сила. Това означава че интерференцията или радио смущенията в TOSSIM са много по лоши отколко в реалния свет.

Радио модела "simple" ги слага всички възли в една клетка. Всеки бит се приема без грешка. Въпреки че не е настанала грешка, два възли могат да предават по едно и също време, всеки възел в клетката ще го чуе дублирането на сигналите кое почти сигурно ще е корумпиран пакет. Въпреки това поради перфектната трансмисия в единична клетка, вероятността за предаване на два възли по също време е много малка и се дължи на

TinyOS CSMA протокола. Простия модел е полезен за изследване на алгоритми с единичен хоп (single-hop) и TinyOS компоненти за коректност. Приема на детерминирани пакети дава детерминирани резултати.

Радио модела "lossy" ги слага възлите в насочен (directed) график.

- Всеки ръб (a,b) в графа, сигнала му може да бъде изслушан от b.
- Всеки раб има стойност (0,1), което представлява вероятността за пращане на бит и дали ще е грешен (flipped) когато b го чува

На пример стойност 0,01 всеки предаден бит има вероятност от 1% за грешка, докато стойност 1,0 означава че всеки предаден бит ще е грешно предаден. Всеки бит се смета за независим.

1.9 Сигурност на безжичните сензорни мрежи

Използването на безжичните сензорни мрежи нараства и се прилага в много области и можем да бъдем сигурни че обсега на използване ще нараства и в други области където моментално не се използва. Безжичните сензорни мрежи се използват в широк обсег на приложения както що е наблюдение на околната среда, следене на обекти, наблюдение на здравето, в военните системи и т.н. [4].

Ако сигурността на мрежата е компрометирана (загрозена) може да доведе до сериозни последици, като губене на частни данни (кражби), не легално сдобиване с данни от безжичните сензорни системи както могат да бъдат данни от наблюдението, следи за обекти, данни от военни системи, също така, атаката може да се използва за монтиране на фалшиви данни, или па компрометиране на мрежата до мера когато няма да дава никакви данни. За състоянието да е по предизвикателни типовете на потенциални атаки се променят и заемат нови и различни форми.

Използването на сензорите в критичните системи както в заводите, самолетите, системите за нуждите на болниците, требва да осигурят автентичност (authenticity), цялост (integrity) и поверителност (confidentiality) на предадените данни. Но това е сравнително трудно в средата на WSN да се осъществи поради редуцираните ресурси, също така не може да се гарантира физическата сигурност. Безжичните сензори имат малка размер и ограничена памет, мощност на процесора и ограничен извор на енергията – батерия. Очевидно това представлява проблем за изпълнение на алгоритъм кой ще я използва наличната памет и които няма да я консумира цялата електрическа мощност. Безжичните сензорни мрежи го сдържат примата на ad hoc мрежа. Приложенията за сигурност на безжичните сензорни мрежи осигуряват някои способностите за събиране, отдалечена анализ на данни и възможността за откриване на атаки.

Моментално изследваната WSN са насочени на проектиране на нов протокол кой поддържа ефикасно използване на ресурсите главно се отнася на консумирането на енергията [5]. Тези протокол осигуря ефективно продължение на живота на сензорните мрежи. Въпреки ограничените ресурси сензорния възел е възможно да използва някои механизми за сигурност както що са криптографски алгоритми и message authentication codes (MAC) който осигурят базисни сервиси за сигурност както цялост (integrity), поверителност (confidentiality) и сервис за автентичност (authentication).

1.9.1 Атаки

WSN можем да ги поделим според някои функции. Някои модели са крайно едноставни: сензора извършва измерване и праща данни. Сложните модели включат изпълнение на сложени алгоритми за работа и обработка на данни. При дискусията за сигурност на WSN требва да изследваме от какво да се защитим а от какво не.

Има много форми и видови на атаки. Можем да ги поделим на: Пасивни атаки – за подслушване, наблюдение събиране на данни и т.н., Активни атаки включват фалшива външност (Masquerade), заместване, изменение, отказ на услуги (Denial of service) и т.н. От всичките атаката при която се проявява отказ на услуги е най-опасна, тъй като от останалите може да се защитава чрез автентификация и криптография.
Отказ на услуги представлява опасност за цялата система, което може да се извърши с ненасочена случайна атака, но вероятно е да бъде насочена поради това че в случая е трудно да се определи кой от участниците е и къде е в мрежата.
Атаките над WSN можем да ги поделим на няколко групи които следват:

Едноставно събиране (Simple Collection and Tranmittal):

• отказ на услуги – denial of service
• предаване на измислени данни - broadcasting spurious information
• физички атаки – physical attack
• атака с препращане на съобщения – replay attacks

Сензорния възел извършва периодични измервана и ги пращат на базисната станция при предпоставка че е достъпен и е в обсега. Тези типове на WSN е неустойчива на атаки които се отнасят на мрежово ниво. Атаката отказ на услуги (denial-of-service) състои се в блокиране (jamming) или интерференция на радио честоти, което инициира колизия. Също така са усетливи на измамнички (spoofing) атаки в които злонамерения атакува с изпращане на измислени данни (broadcasting spurious information), неустойчиви се на физички атаки (physical attack), както що е откradване унищожение и т.н. Атаките с повторно изпращане на вече пратено на предишно изготвено съобщение (replay attack) са присъни тук.

Препращане (Forwarding):

• черна дупка - Black Hole
• селективно препращане – Selective Forwarding
• корумпиране на данните – Data Corruption
• изтощаване на ресурсите - Resource Exhaustion

Сензора ги събира и ги праща данните на един от съседните сензори които се намира на пътя до базисната станция. Така посредничките сензори ги препращат съобщенията до базисната станция или до съседен сензор така че на край данните да стигнат до базисния сензор. При приемането на данни сензора (в първия случай) не ги обработва само ги предава нататък. Поради това WSN е неустойчива на атаката Black Hole, корумпиране на данните (Data Corruption) и изтощение на ресурсите (Resource Exhaustion). В атаката Black Hole възела които е отговорен за препращане на данните той ги отбива (унищожава). Атаката с корумпиране на данни я променя съдържанието на данните при предаване. Цел на тези атаки е превземане контрол над мрежата. Изтощение на ресурсите се осъществява на този начин че злонамерения праща голямо количество на данни кое представлява излишно трошене на енергийните ресурси (батерията). Когато пакета има експлицитно зададен път които требва да го помине това се нарича селективно препращане (Selective Forwarding).

Получаване и обработка на команди (Receive and Process Commands):

• пращане на измислени (фалшиви) команди

Сензорния възел приема команди от базисната станция която управлява пряко или през техен съсед (непряко) друг възел, при що се пращат команди за променя на състояние или действия. Възможността за обработка на командите е от голяма полза за намаляване на количеството данни в WSN. Командите могат да се пращат към всички възли (broadcast) или към един (unicast). При пращане към един възел нужно е да се осигури някакво адресиране на възлите. Тук идва атаката при която злонамерения възел се заема свойство на главен възел (базисна станция) и изпраща измислени (фалшиви) команди.

Само организация (Self-Organisation):

//atacks aginst routing protocol:
• spoofed, altered, or replayed routing information
• sinkhole attacks
• Sybil attacks
• wormholes
• HELLO flood attacks
• acknowledgement spoofing

Изпълнението на WSN се представлява като само организационна единица коя се учи и управлява с топологията. Информацията за топология на мрежата може да бъде позната само на базисната станция или да бъде споделена с някои друг второ степен главен възел

или пак с всички възли в мрежата. От възлите тогава се изиска да се отнасят като клъстери (cluster) за да могат да го решат проблема с колизията в мрежата. WSN, на кратко казано е чувствителна на всички атаки които се отнасят на маршрутните протоколи (routing protocols), тук ще ги споменем атаките Induced Routing Loops, Sinkholes, Wormholes, HELLO Flooding.
Когато могат да се преследват данните в самата структура на мрежата, злонамерения може да го променя данните, наново да ги праща същите или да измисли (spoofed, altered, or replayed routing information). Това я загрозява истинската фигура на мрежата каква я вижда базисната станция.
Атаката Sinkhole се темели на измислен (правилен) качествен и бърз път от намесения възел до базисната станция. Така че всички околни възли които искат да комуникират с базисната станция го използват намесения възел по що той указва голямо влияние в мрежата. Така що тези атака представлява селективно проследяване.
Атаката Sybil се извършава така че се слага един злонамерен възел в мрежата и той глуми като да е възел от мрежата. Поради това може да се намали ефикасността на някои алгоритми които мрежата ги ползва, на пример за намиране на най-къси пътища, разпределена обработка на данни, запазване на топологията или проследяване.
Wormhole представя атака при която злонамерения слага между два възли още един възел. Измежду два възли се създава директна връзка, помежду два поделени дела на мрежата с помощ на комуникационен спектър които мрежата не го ползва или с помощ на някое друго комуникационно средство. Пример за два лаптопа кои са по мощни от възлите това не е проблем. Злонамерения така може да ги увери възлите дека се близо и с това се създават скокове които пакета требва да ги направи (sinkhole).
Много протокола за определяне на състоянието на мрежата, в началото изискат всички възли да ги намерят своите съседи при що за тези цел се използва така наречени HELLO пакети. За добиване на карта на мрежата така че пакетите и обхождат всички възли, по що всеки възел ги знае своите първи съседи. Затова се изиска базисната станция да приема и да праща HELLO пакети. С това се предпоставя че приетите пакети са от доменна на възлите, при което ги памети като свои съседи. Тук злонамерения има добра прилика за атака. Базисната станция излъчва HELLO пакети с доста як сигнал (поради което има по-голяма и помощна антена от възлите) убеждавайки ги членовете на мрежата че им е съсед. Това довежда до конфузия в мрежата и се затруднява ефикасното проследяване.
Induced routing loops е атака на протокола за проследяване така че се създава точка къде е невъзможно протока на данни. Злонамерения създава или инициира нова връзка между възлите така че данните кръжат в мрежата търсейки я базисната станция.

1.9.2 Сервиси за сигурност на безжичните сензорни мрежи

Има няколко важни аспекти за сигурността на безжичните сензорни мрежи които се отразяват пряко върху изпълнението на шифроване и автентичност в системите. Те се свързани с поддържане на секретност, защита на данните и цялост на данните при работа на системата. Освен това, с помощ на сложни процедури за удостоверяване се гарантира че само съответни участници им е разрешено да участват в цялата система, независимо дали са активни или пасивни.

Access control – контрол на пристъпа. Това представлява контрол на пристъпа над безжична мрежа. Протокола на каналния слой треба да спре неупълномощен / непознат обект да участва в мрежовата комуникация. По разпоредбата сензора требва да е способен да детектора съобщение от неупълномощен възел и същото да го отбие.
Confidentiality – Поверителност. Това се отнася на защита срещу пасивни атаки, както на пример придобиване на данни с копиране и с анализа на трафика при оперативна анализа. Шифроването често се използва за защита срещу такива атаки. Това обаче не може да даде пълна гаранция за защита при всички използваеми схеми за шифроване.
Integrity – Интегритет. Поддръжането на цялост на системата, е свързано с гарантирането че съдържанието на съобщението не е променено или не се променило. Пълна цялост се отнася на целия поток на съобщението, при което се гарантира че няма съобщение или части от съобщението кои се изчезнали, са биле променени или са допълнени, което може да се изпълни при активна атака.
Authentication – Автентизиране. Атаките със маскиране (Masquerade) се осъществяват така че неупълномощен участник поема самоличност на лице което е упълномощено. Автентизирането представлява гаранция, че по всяко време има упълномощен достъп.
Non repudiation - няма отричане. Това е има практически голям интерес и ни гарантира че след изпращане на съобщение на една от страните, не са в състояние да отричат че са пратили, също така страната коя го приема съобщението може да докаже че първоначалната страна изпратила съобщение.

1.9.3 Моментални ограничения и податливост

Използването на автентификация и шифроване при симулациите или при други видове на реално временно свързани действия, се лимитира от допълнителното закъснение наметнато поради конструиране на съобщението, обмена, шифроване / де шифроване на всеки край на включените страни. С използването в реална среда, персонал, пускането на мобилната симулация се явява въпроса за използване на мобилните комуникационни устройства. Първо е въпроса за физическата сигурност на малките портативни цифрови устройства, PDA устройства, smart cards и т.н. Те лесно могат да се изгубят или откраднат. Тук идват и ограничените ресурси на тези устройства, които ограничават сложността и степента на сигурност коя е възможна в момента. На пример мощността на процесора и размера на памет я ограничава сложността на алгоритъма за шифроване / де шифроване. Освен това е ограничена честотата и обсега на безжичните мрежи което води до предаване с ограничение на скоростна и с ограничението на отдалеченост от базисната станция. На край основния проблем е несигурността на радио средата, при което с даден приемник можем да го получим, де шифроваме и тълкуваме съобщението.

Втора глава

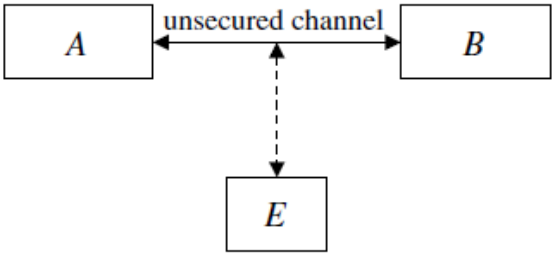
Обзор на основните понятия на криптографията по елиптична крива. Математическата основа на елиптичните криви над крайните полета и операциите с групи.

2.1 Основи на Криптографията

Криптографията се отнася на проектиране и анализ на математическите техники, кои осигуряват сигурност на комуникации в присъствие на злонамерени противници.

2.1.1 Основен комуникационен модел

На фигура 14, страните А (Alice) и Б (Bob) комуникират по един не сигурен (unsecured) канал. Предполагаме че всички комуникации са проведени в присъствие на Е (Eve) който е злонамерения, и чиято цел е да ги преодолее всички сервиси за сигурност с които се осигурени А и Б.



Фигура 14. Основен комуникационен модел.

На пример А и Б може да се двама души които общуват през клетъчна телефона мрежа и Е се опитва да и прислушва разговора им. Други пример е когато А използва браузъра и е на някоя индивидуален уеб сайт ~А и в процес на онлайн закупуване на продукт от онлайн магазин ~Б представен чрез техния уеб сайт Б. В тези сценарий комуникационния канал е интернетът. Злонамерения Е се опита да прочете трафика от точка А до точка Б при което да ги прочете и научи данните за кредитната картичка на ~А или да се опита фалшиво да се представи в транзакцията на А и Б в която ще има лична полза. Като трети пример ще я споменем ситуацията когато А изпраща емайл съобщение до Б през интернет. Злонамерения може да се опита да го прочете съобщението, да променя и изтрива избрани части от съобщението или да изпрати свое съобщение до Б. На край помислете за сценария, където А е една смарт карта, която е в процеса на установяване на автентичността на техния притежател ~А, на централния компютър Б кой е в някоя банка. Тук Е би можел да се опита да върши следене на комуникациите за да получи информация за потребителя ~А, или да се опита да тегли парични средства от сметката на потребителя

~A. От тези пример се вижда че комуникационния вход не е задължително осъществен от човек, може да бъде компютър, смарт карта или софтуерен модел които действат в името на физическо лице или организация, кое действие може да бъде в магазин, банка и т.н.

2.1.2 Цели на сигурността

Внимателно изследване на сценариите представени по-горе ги приказват следните основни изискания за сигурност на комуникациите:

1. Конфиденциалност (Confidentiality): съхранение на данни в секретност освен от упълномощените на кой се позволява да ги гледат съобщенията изпратени от А до Б и които не требва да ги чете Е.
2. Цялост на данните (Data integrity): да се гарантира, че данните не са били променени от неупълномощени лица – кое означава че В трябва да бъде в състояние да открие, когато данните, изпратени от А, са били променени от Е.
3. Автентификация на произхода на данните (Data origin authentication): потвърждаване на източника на данни - В трябва да бъде в състояние да удостовери че данните, изпратени от А наистина са с произход от А.
4. Удостоверяване за автентификация (Entity authentication): потвърждаване на самоличността на страната - В трябва да бъдат убедени в самоличността на другото лице с което комуникира.
5. Няма отричане (Non-repudiation): предотвратяване на лице да отрича предишните си ангажименти и действия - когато В получава съобщение, от А, не само че В е убеден, че съобщението е произхожда от А, също така В може да убеди неутрална трета страна за това, по този начин А не мога да отрича че е изпратило съобщение до В.

Някои приложения могат да имат и други сигурност ни цели, като анонимност при комуникирането или контрол на достъпа (ограничение до достъпа до ресурсите).

2.1.3 Неприятелски модел

За модела на реални заплахи към кои се изправени А и Б, ние генерално предполагаме че злонамерения Е има значителни възможности. Освен че е в състояние да ги чете всички данни предавани чрез канала, също така Е може да ги променя представените данни и да инжектира собствени фалшиви данни.

Освен това Е има значителни изчислителни ресурси на тяхно разположение. На край пълни описания на протоколи за комуникация и криптографски механизми (с изключение на секретните информации за ключовете) са познати на Е. Предизвикателство за криптографията е да се проектират механизми който ще я осигурят комуникацията и при такива мощни противници.

2.1.4 Симетрична / Асиметрична криптография

Посочени са основните понятия за криптографията, както за симетричната така и за асиметричната. Разглежда се използване на схеми с симетричен споделен ключ за шифроване / де шифроване, които схеми са свързани с проблема за разпределение и управление на ключовите, като и схемите с асиметричен (публичен) ключ.

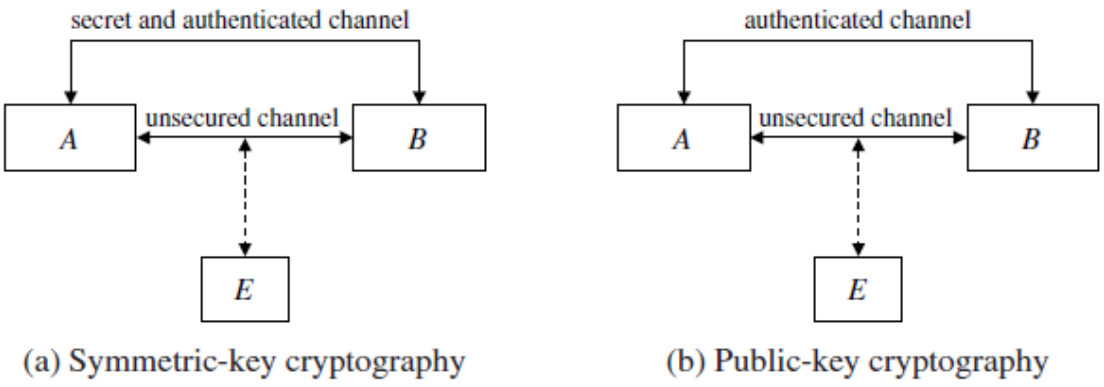
Шифроване – **ENCRYPTION.** Много методи биле използвани в текат на вековете и всичките имали различен степен на успех. Въпреки с навлизане на компютрите се улесни работата на злонамерения. Крипто анализата стана предмет на обсъждане и е достъпен на специалисти за сигурност, престъпниците, хакерите и т.н. Като резултат на това системите кои съществуват са под закана на сериозни атаки. Не съществува безусловна гаранция за сигурност в реалността. Реалните условия на компютърната сигурност би можело да я представим така че цената на разбиването на някои шифър е по-голяма от стойността на шифрираните данни, а времето за разбиване на шифърът е по-дълго от колко користния век на шифрираните данни [Stallings B. 1995]. Друго условие по принцип, е използване на временна отклонена мярка (за определено време система се обновява). Първия начин е по-лесен се разбиване, поради това що съществува задни вратички (backdoor), а и повече математически техники.

2.1.4.1 Криптография с симетричен ключ

Криптографските системи могат да бъдат до голяма степен разделени на два вида. На схеми со симетрични ключове, посочени на фигура 15.(а), първо комуникационно договаряне на ключовете което ще е секретно и те ще се автентични. В последствие те могат да използват схеми за шифроване с симетричен ключ както що се Data Encryption Standard (DES), RC4, или Advanced Encryption Standard (AES) за да се постигне конфиденциалност (confidentiality). Те също могат да използват и message authentication code (MAC) алгоритъм като HMAC за постигане на цялост на данните и автентизация на произхода на данните.

На пример, ако конфиденциалност е желателна и споделянето на секретен ключ за A и B е $\boldsymbol{к}$, тогава A ще шифрова текстово съобщение $\boldsymbol{m}$ с използване на шифрова функция $\boldsymbol{ENC}$ и ключа $\boldsymbol{к}$ и предаването ще бъде резултат шифрован текст c=$ENC_k(m)$ до B (c – шифрован текст). По получаването от B ще се използва функция за де шифроване DEC с същия ключ $\boldsymbol{к}$ и възстановяването на шифрираният текст ще бъде c=$DEC_k(c)$.

Ако се изиска цялост на данните и автентификация на произхода на данните, тогава A и B първо требва да се договорът за секретния ключ $\boldsymbol{к}$, по което A ще изчисли тагот за автентификация t=$MAC_k(m)$ което за текстовото съобщение използва MAC алгоритъм и ключ $\boldsymbol{к}$. И тогава ще ги изпрати m и t на B. При получаването на m и t, B ще го изпълни алгоритъма MAC и същия ключ $\boldsymbol{к}$ за изпълнение на t'=$MAC_k(m)$ на m и приемане на съобщението като произход на A, ако $t' = t$.



Фигура 15. Симетричната наспроти криптографията с публичен ключ.

Криптографията с симетричен ключ (известно още като криптография с частен ключ) дава възможност за поверителен и сигурен обмен на съобщения между две страни. Особено е важно данните да не се разкриват на трета страна. Цялост на данните може да бъде гарантиран с помощ на подходящ режим на работа с симетричен шифър. Автентичността без отричане може да се постигне с използване на криптография с симетричен ключ така що секретния ключ е познат само на двете страни.

Разпределение и управление на ключове. Главното предимство на криптографията с симетрични ключове е високата ефикасност, но има и значителни недостатъци в тези системи.

- Основен недостатък е така наречения проблем за разпределение на ключовете. Се изисква сигурно предаване на секретните ключове, преди започване на обмена на съобщенията.

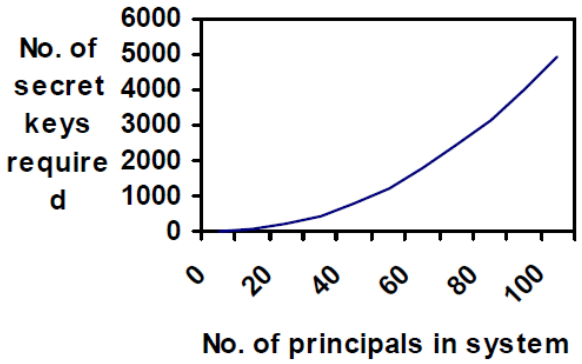
В някои приложения тази разпределяне може удобно да се направи с помощ с физически защитен канал кой ще ни представлява надежден куриер. Друг начин е използване на доверени онлайн сервиси, както трети страни които първоначално създават секретен ключ с всички субекти в мрежата и след това използват тези ключове за сигурна комуникация и пращане на ключовете при искане от същите. Решение както този може да бъде добро приспособено към места където има доверена централен авторитет, но абсолютно е неприложимо в приложения както емайл в интернет.

- Втория недостатък е проблема с управлението с ключовете. В мрежовата среда всека двойка потребители се нуждае от различни ключове кое в резултат в системата с n потребители се изиска ($\frac{n(n-1)}{2}$) двойка на ключове.

В мрежа с N потребители всеки потребител трябва да подържа ключове за N-1 потребители в мрежата. Този проблем може да бъде облекчен чрез използване онлайн доверие чрез което се разпределя ключове при искане, по този начин се намалява нуждите от съхранение на множество ключове. Обаче този решение не е особено практично в

някои сценария. На край когато се разпределят ключовете между двама (или повече) лица, криптосистемата с симетричен ключ не може да осигуря схеми за цифров подпис със които се осигурява не отхвърляне на услугите. Това е така защото не невъзможно да се направи разграничения между действията предприети от различните притежатели на секретния ключ.

Създаването и споделянето на тайната ръчно се оказва трудно поради това в разпредялбата са включват по-голям брой на лица отколко при традиционните системи. Също така се въвежда проблем за сигурността и съхранението на голям брой на двойки секретни ключови.



Фигура 16: Брой на ключове които се изискат в симетрична криптографска система

2.1.4.2 Криптография с публичен ключ (асиметрична криптография)

Идеята за криптография с публичен ключ, посочена на фигура 15.(b) е въведена от Diffie, Hellman и Merkle, [Diffie et al, 1976] така че в криптографията с публичен ключ ги няма горе посочените недостатъци на симетричната криптография. За разлика от схемите с симетричен ключ, схемите с публичен ключ изискат само ключ от страната с която комуникират при що ключа не е секретен но е автентичен. Всяка страна в комуникацията си избира двойка ключове (e, d) която се състои от публичен ключ ‘e’, и частен ключ ‘d’(този ключ се чува в секретност).

Съществуват голям брой на различни схеми, връз основа на двойка асиметрични ключове, е предложено да се архивират в отделни сервиси за сигурност кои се наведени в част 1 в книгата на Diffie и Helman. Съществуват три главни класи на схеми,

• Схеми за дигитален подпис
• Схеми за шифроване
• Схеми за разпределение на ключове

Въпреки че схемата с публичен ключ може да ни я осигури цялата функционалност необходима в съвременните протоколи за сигурност като SSL / TLS, тя е трудна за изпълнение поради високите изчислителни изискана. Дори и кога се прилага правилно всички PK схеми предложени до този момент са няколко пати по-бавни от най-известните схеми с частни ключ. На практика е въведено сместване на системи с симетричен ключ и с публичен ключ по при що получените системи се наричат хибридни криптографски системи. Алгоритмите с публичен ключ са избрани за установяване на ключове, а алгоритмите с симетричен ключ е избран за шифроване на данни. Като цяло на практика алгоритмите с публичен ключ може да ги разделим на три фамилии:

¥ Алгоритми базирани на математическо разлагане на множители на цели числа (*Integer Factorization Problem* - IFP): за дадено положително цяло число N, да се намерят прости множители. На пример, RSA, е най-популярния алгоритъм с публичен ключ кръстен на създавалите - Rivest, Shamir и Adelman

¥ Алгоритми базиран на DLP (*Discrete Logarithm Problem*): за дадени α и β да се намери положително число к така че $\beta = \alpha^{\kappa} mod\ p$. На пример, DH (Diffie-Hellman) ключов протокол за обмена [18] и алгоритъма DSA (Digital Signature Algorithm).

¥ Алгоритми базирани на *ECDLP* (*Elliptic Curve Discrete Logarithm Problem*): за точки P и Q на елиптични криви коя е определена в едно крайно поле, да се намери положително число к така, че Q = К * P. На пример ECDH (*Elliptic Curve Diffie-Hellman*) протокол за обмена на ключове и ECDSA (*Elliptic Curve Digital Signature Algorithm*).

Ще споменем че съществуват още много други схеми с публичен ключ като NTRU, или системи базирани на скрити области на уравнения които не са широко разпространени в употреба. Научната общност е на самото начало на изследването на сигурността на тези алгоритми.

Конфиденциалност. Когато страната A желае да изпрати текстово съобщение m, при що получава автентичен екземпляр от ~B на публичния ключ e_B и използва функция за шифриране ENC за извършаване на схемата на шифриране се изчислява с = $ECN_{e_B}(m)$. След това се предава на B която използва функция за де шифроване DEC и частния ключ d_B и за възстановяване на съобщението m = $DEC_{d_B}(c)$. Предполага се че злонамерения с съзнание на e_B (но не и на d_B) не може да го де шифрова с. Вижда се че няма изискана за секретност на e_B. От съществено значение единствено е A да получи истински екземпляр на e_B - в противен случай A би шифровала m с помощ на публичен ключ e_E от някоя страна E (E – злонамерения), коя претендира да е B, и m ще е възстановено от E.

Без отричане – цифров подпис (Non-repudiation Digital signature) схеми могат да се разработят за автентификация на произхода на данните (data origin authentication) и цялост на данните (data integrity) както и да се улесни и осигуря неотхвърляне на сервисите.

Страната A използва алгоритъм за генериране на цифров подпис SIGN които представлява схема за цифров подпис и частния им ключ d_A за да се извърши подписването се извършва s =$SIGN_{d_A}(m)$.

При получаване на m и s от страната B изиска автентичен екземпляр от A на публичния ключ на e_A по що използва алгоритъм за проверка и потвърждение на подписа че наистина е генериран от m и d_A. поради това че d_Aсе знае само от A, B може да се увери че съобщението е с произход от A. Освен това, като проверката се нуждае от не секретните елементи m и e_A, подписа s за m може също така да бъде верифициран от трета страна, която може да ги реши споровите, ако A отрича че е подписало съобщението m. За разлика от ръчните подписи, подписа s на A зависи от съобщението m кое се подписва, кое предотвратява получаване на подписа на друго съобщение m' и след това да твърди че е подписано m'.

Въпреки секретните изискана по отношение на публичния ключ e_A, важно е че при верификацията да се използва автентична копия e_A, когато се проверява подписа дали е генериран от A.

По тези начин криптографията с публичен ключ предвижда елегантни решения на три проблема които се срещат при симетричната криптография, които са разпределяна на ключове, управление с ключовете и осигуряване за неотхвърляне. Следи да се отбележи, че изисканата за секретен канал за разпределяне на ключове е елиминирано и се прилага PKI (public-key infrastructure) за разпределение и управление с публични ключове. Недостатък на криптографията с публични ключове е това че операциите с публичните ключове обикновено са значително по-бавни от колкото при симетричните ключове.

На край ще споменем че се проектирани и така наречени хибридни системи, които се по често използват. Хибридните системи я използват ефикасността на симетричните ключови алгоритми и функционалността на алгоритмите с публичен ключ.

2.2 Криптосистема с Елиптична крива – ECC (Elliptic Curve Cryptography)

Сравнително ново е използването на криптография с елиптична крива. Тя е представена независимо от Victor Miller [9] и Neil Koblitz [10], и от тогава тя представлява популярна област за изследване и е любимец на много компании и е стандартизиран от ANSI [11], IEEE [12], ISO [13] и NIST [14].
Криптографската система по елиптична крива се основа на добро познатия DLP (discrete logarithm problem) над група от точки на елиптична крива, кое е известно като ECDLP (*elliptic curve discrete logarithm problem),* като основа на схема с публичен ключ, така що ECDLP е трудно за изчисление, даже по-трудно от математическо разлагане на множители на целите числа (factoring). Поради това сигурността коя ни я прилагат тези схеми е по-голяма за по-малка дължина на ключове. Подобните схеми също така ще бъдат по гъвкави

при бъдещите подобрения на хардуера, под предпоставка че няма значително подобрение на ефикасността на алгоритъма за решаване на ECDLP.

Елиптичните криви, дефинирани над крайно поле осигурят групова структура коя се използва за прилагане на схеми в криптографията. Елементите на групата са смислени точки на елиптичната крива заедно с специална точка *O* (наречена ”точка на безкрайността”) отнасяйки се като идентичен елемент на групата. Груповата операция е събиране на точки, което се постига с помощ на аритметичните операции в съществуващото крайно поле. Основен главен изграждащ елемент на криптографските системи с елиптична крива е скаларното умножение на точка с операция в форма к * P, където к е положително цяло число а P е точка на елиптичната крива. Изчисляването на к * P означава добавяне на точка P точно к - 1 пъти в себе си, което води до друга точка Q върху елиптична крива. Обратната операция т.е. да се възстанови ***к*** когато са дадени точките P и Q = к * P = P+P+…+P, е известно като *Elliptic Curve Discrete Logarithm Problem* (ECDLP). До момента няма алгоритъм кой ще го реши ECDLP за под експоненциално (sub exponential) време в правилна избрана група на елиптичната крива [15]. Това я прави криптографията с Елиптична крива надежден клон на криптографията с публичен ключ, коя предлага подобна сигурност като и другите ”традиционно” DLP - базирани системи в употреба днес,с по-малка дължина на ключът при що и по-малки изискана на памет.
Много от новите протоколи за сигурност се отделят от избора на криптографски алгоритми от дизайна на протокола. Потребителя на протокол уговаря се за алгоритъма кои ще се използва за определена защитена сесия. Следователно, ECC базирани алгоритми могат лесно да се интегрират в съществуващите протоколи за постигане на сигурност и обратна съвместимост с по-малко ресурси. От което идва да по-нисък клас ограничени устройства да могат да използват такива протоколи, кои до неотдавна са сметани за неподходящи за тези системи.

2.2.1 Общо описание на ECDLP (elliptic curve discrete logarithm problem)

Елиптичните криви осигурят алтернативна основа за обмена и споделяне на тайни данни в среда къде е възможно преслушване, при което осигуря перфектна и напредна секретност. ECDLP, в кой ECC [18, 17] е базиран, типично въвлича възстановяване над Galois (*i.e.*, крайно) поле, F, от k ∈ F, зададен (за най-малко) k*G, G, и E, къде G е точка от елиптичната крива, E е крива от уравнението на Weierstrass

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

къде $a_i \in F$. От индивидуален интерес е F_p и F_{2p}, къде p е просто число, от където се вижда като уязвимост при под експоненциални (sub exponential) атаки [19]. Някога популярните, разширени полета над съставна (composite) степен над F_{2p} се показа уязвим при съкращения с Weil спускане [17] на ECDLP в DLP над хипер - елиптичните криви [20]. Но F_{2p}бинарни разширени полета, остават популярни в имплементацията на ECC,

особено в хардуера, които осигуряват алгоритми за специални пространства и времево ефикасни алгоритми.
За начало ще ги погледнем групите на елиптичните криви и ще покажем как могат да се използват в изпълнение на DLP (discrete logarithm *problem*).

Групи (Groups). Абелева *(Abelian)* група $(G,*)$ се състои от набор G с бинарна операция $* : G \times G \rightarrow G$ която ги задоволява следните условия:

(i) Асоциативност (*Associativity*) $a * (b * c) = (a * b) * c$ за всички $a,b, c \in G$.

(ii) Неутрален елемент – той е единствен (*Existence of an identity*). Съществува неутрален елемент $e \in G$ така че $a * e = e * a = a$ за всички $a \in G$.

(iii) Обратим елемент (*Existence of inverses*). За всеки $a \in G$, съществува обратим елемент $b \in G$, наречен *inverse* на a, така че $a * b = b * a = e$.

(iv) Комуникативност (*Commutativity*) $a * b = b * a$ за всеки $a,b \in G$.

Груповата операция обикновено се нарича адитивна (+) или мултипликативна (*).

Адитивна	Мултипликативна
(+)	(*)
e = 0 нулев елемент	e = 1 единичен елемент
$a' = -a$ обратим	$a' = a^{-1}$ обратим
$\underbrace{a + a + \cdots + a}_{n} = an$	$\underbrace{a * a * \ldots * a}_{n} = a^n$

Таблица 1.

В първата илюстрация групата се нарича адитивна (*additive*) група, (additive) идентичните елементи обикновено се означени с 0, а (additive) обратимия на a е означен с $-a$. В втората илюстрация групата се нарича мултипликативна (*multiplicative*) група, (multiplicative) идентичните елементи обикновено се означени с 1, а (multiplicative) обратимия на a е означен с a^{-1}. Групата е ограничена (*finite*) ако G *е* краен набор, за кой случай броя на елементите в G се нарича ред на G.

На пример, нека P е просто число, и нека F_p ={0,1,2, . . ., p−1} означува набор от цели числа по модул p (integers modulo p). Тогава $(F_p,+)$, къде е дефинирана операцията (+) събиране на цел брой модул p, е ограничена адитивна група от ред p с единствен елемент 0. Също така $(\boldsymbol{F^*_p}, *)$ когато се дефинира $\boldsymbol{F^*_p}$ означава не нулев елемент в F_p и е дефинирана операцията * умножение на цел брой с модул p, и е ограничена

мултипликативна група от ред $p-1$ с единствен елемент 1. (F_p, +, *) е крайно поле, означено кратко със F_p.

Ако G е ограничена мултипликативна група на ред $\boldsymbol{n}$ и g ∈ G. Най-малкото положително число t за което $g^t = 1$ (e = 1) ще нарича ред на g, като той винаги съществува и е делител на $\boldsymbol{n}$. Нека G е група и g ∈ G. Множеството от всички степени на g $\langle g \rangle = \{g^i : 0 \leq i \leq t-1\}$ в себе е група над съща операция, и ще я наричаме циклична подгрупа породена от g. Аналогичните твърдения са верни, ако G е адитивна. В този случай, порядъкът на g ∈ G е най-малкият положителен делител t на n, така че tg = 0, и <g> = {ig: 0 ≤ i ≤ t -1}. Тук, tg означават елемент, получени чрез добавяне на t копията на g. Ако G е елемент g от ред на n, тогава G се казва, че е циклична група и g се нарича генератор (*generator*) на G.

2.2.2 Генерализиран DLP (Generalized discrete logarithm problem)

Да предположим че *(G, *)* е мултипликативна циклична група от *n*-ти ред с генератор *g*. На пример принадлежащите параметри са *g* и *n,* частния ключ е цялото число *x* което се произволно избира от интервала [1, *n*−1], и публичния ключ е *y* = *gx*. Проблема е детерминирането на *x* при дадено *g*, *n* и *y* което представлява DLP в *G*.

За дискретен логаритъм проблем който се базира на *G* да бъде ефективен, с бърз алгоритъм трябва да се знае за изчисляване на груповите операции. По отношение на сигурността проблема в *G* трябва да е неподатлив.

Всеки две циклични групи от същи ред *n* по същество са същи, т.е. те имат съща структура, въпреки че елементите могат да и са записани различно. Различното представяна на групи от елементи могат доведат до алгоритми с различна бързина за изчисление на операциите с групи и за решение на DLP.

Най-известни групи за изпълнение на DLP са цикличните под групи на мултипликативната група на крайните полета и цикличните под групи на елиптичните криви който следват в продължение.

2.2.3 Групи на Елиптичните криви (Elliptic curve groups)

Нека *p* е просто число, и нека F_p означава област от цели числа модулно *p* (integers modulo *p)*. Елиптична крива *E* над F_p се дефинира със уравнение в форма

$$y^2 = x^3 + ax + b, \quad (1.4)$$

къде $a, b \in F_p$ удовлетворява $4a^3 + 27b^2 \neq 0 \pmod{p}$. двойка *(x, y)*, къде $x, y \in F_p$, е точка от кривата ако *(x, y)* го удовлетворява уравнението (1.4). Безкрайната точка се означава с ∞, също треба да е от кривата. Набора от всичките точки на *E* се означава с $E(F_p)$.

На пример ако E е елиптична крива над F_7 с определеното уравнение

$$y^2 = x^3 + 2x + 4$$

При това точките на E са

$E(F_7) = \{\infty, (0,2), (0,5), (1,0), (2,3), (2,4), (3,3), (3,4), (6,1), (6,6)\}$.

Един добре познат метод за събиране на две точки на елиптичната крива $(x1, y1)$ и $(x2, y2)$ за добиване на трета точка на елиптичната крива. Правилото за събиране изиска няколко аритметични операции (събиране, изваждане, умножение и inversion) в F_p с координатите $x1$, $y1$, $x2$, $y2$. С правилото за събиране, набора от точки $E(F_p)$ оформя (адитивна) абелева (abelian) група с елемент ∞ което обслужва роля на идентичен (единствен) елемент. Цикличната под група на тези група на елиптичната крива сега може да се използва за изпълнение на системи с дискретен логаритъм.

2.2.4 Генериране на ключове за елиптични криви

Нека E е елиптична крива дефинирана в крайно поле F_p. нека P е точка в $E(F_p)$, и се предполага че P е прост число от ред n. Тогава циклична подгрупа на $E(F_p)$ генерирана от P е

$$<P> = \{\infty, P, 2P, 3P, \ldots, (n-1)P\}.$$

Простото число p, уравнението на елиптичната крива E, и точката P и реда n, са публичните притежателни параметри. Частния ключ е цяло число k кое е избрано еднакво на случаен принцип от интервала $[1, n-1]$, както и съответния публичен ключ е $Q = kP$.

Проблема на определянето на k покрай зададените притежателни параметри и Q е ECDLP (*elliptic curve discrete logarithm problem*).

Алгоритъм 1. Генериране на двойка ключове на елиптична крива - Elliptic curve key pair generation

INPUT: Elliptic curve domain parameters (p, E, P, n).

OUTPUT: Public key Q and private key k.

1. Select $k \epsilon_R [1, n-1]$.

2. Compute $Q = kP$.

3. Return(Q,k).

2.2.5 Схема за шифриране с елиптична крива (Elliptic curve encryption scheme)

Представяме процедури за шифроване и де шифроване за елиптична крива аналогова, основана на схемата за шифроване на ElGamal посочена като Алгоритъм 1.1 и 1.2, съответно. Текст *m* първо е представен като точка *M*, и е шифроване със добавяне кон $k*Q$ къде *k* е случайно избрано цяло число, и Q е публичен ключ предвиден за получателя.

Подателя предава точките $C_1 = kP$ и $C_2 = M + kQ$ на получателя кой го използва своя частен ключ *d* за да го изчисли

$$dC_1 = d(kP) = k(dP) = kQ$$

И след това възстановява $\mathrm{M} = C_2 - kQ$. Злонамерения който желае да го възстанови *M* необходимо е да изчисли $k*Q$. Тези задача на изчисление $k*Q$ от притежателните параметри, *Q*, и $C_1 = kP$,е аналог на елиптичната крива на Diffie-Hellman проблема.

Алгоритъм 1.1 Основно шифриране по ElGamal с елиптична крива - Basic ElGamal elliptic curve encryption

INPUT: Elliptic curve domain parameters *(p, E, P,n),* public key *Q*, plaintext *m*.

OUTPUT: Ciphertext *(C*1,*C*2*)*.

1. Represent the message *m* as a point *M* in *E(*F*p)*.

2. Select $k \in R$ [1,$n-1$].

3. Compute $C1 = kP$.

4. Compute $C2 = M + kQ$.

5. Return(*C*1,*C*2).

Алгоритъм 1.2 Основно де шифриране по ElGamal с елиптична крива - Basic ElGamal elliptic curve decryption

INPUT: Domain parameters *(p, E, P,n),* private key *d*, ciphertext *(C*1,*C*2*)*.

OUTPUT: Plaintext *m*.

1. Compute $M = C2 - dC1$, and extract *m* from *M*.

2. Return(*m*).

2.3 Защо криптография по елиптична крива?

Има няколко критерий, който трябва да се вземат предвид при избора на схема с публични ключове за специфично приложение. Основни критерий са:

1. Функционалност (*Functionality).* Дали избрания метод криптография ни ги осигуря желаните способности?

2. Сигурност (*Security).* Какви гаранции са на разположение че протоколите са сигурни?

3. Ефективността (*Performance).* За желаното ниво на сигурност, дали протоколите отговарят на техническите цели?

Други фактори които могат да повлияят на решението включва наличие на най-добрите практични стандарти, разработени от акредитирани организации за стандартизация, наличието на търговски криптографски продукти, покритие на патенти както и степента за развой и съществуващите инсталации.

Семействата на RSA, DL и EC осигурят основна функционалност очаквана от публичните ключове, шифроване – де шифроване, дигитален подпис и разменна на ключове.

С тека на времето, изследователите са разработили техники за проектиране и доказване на сигурността на протоколите RSA, DL и EC при смислени предположения. Основния проблем е сигурността коя се основа на трудността на математическия проблем, който е необходим за сигурността на всички протоколи в фамилията на публични ключове — на пример проблема със математическото разлагане на множители (factorization) на целите числа за RSA система, DLP (discrete logarithm problem) за DL система, и ECDLP – (elliptic curve discrete logarithm problem) за EC системи. Трудността на тези проблеми пряко въздействат на ефективността тъй като по това се диктува размера на принадлежащите и ключовите параметри. Това от друга страна се отразява на основните аритметични операции.

Ще дадем оценка на големината на параметрите кой осигурят равностойна степен на сигурност за RSA, DL и EC системи. Тези сравнения показват че криптографията по елиптична крива е особена полезна за приложения кой изискат високо ниво на сигурност.

Ще започнем с въведение към някои основни понятия от алгоритъмов анализ.

Измерване на ефективността на алгоритмите (Measuring the efficiency of algorithms). Ефективността на един алгоритъм се измерва при ограничени ресурси кой ги консумира. Обикновено се използва мярка за време, но по някога и други мерки като пространство и броя на процеси също се считат. Логично е да се очаква че един алгоритъм консумира повече ресурси при по-голям вход, и ефикасността на един алгоритъм е описан като функция на входни размери. Размера се определя като брой на битове кой се недоходни за

представяне на вход с използване на разумно кодиране (encoding). На пример един алгоритъм за математическото разлагане на множители (factoring) на цел брой n има входен размер $l = [log_2 n] + 1\ bit$.

Изразите за време на изпълнение на алгоритъма са най-полезни ако са независими от особени платформи, използвани за изпълнение на алгоритъма. Това се постига чрез изчисляване на броя на изпълнените елементарни операции (на пример битови операции). В (най-лош случай) времето за изпълнение на алгоритъма е горна граница изразена като функция на входни размери, от броя на елементарни стъпки извършени от алгоритъма. На пример метода на пробното деление на кой фактор и е цялото число n със проверка на всички възможни фактори до $\sqrt{n}$ има изпълнение за време приблизително от $\sqrt{n} \approx 2^{2/l}$ стъпки за деление.

Трудно е да се получат точни изрази за времето на изпълнение на алгоритмите.

В тези ситуации удобно е да се използва "big-O" нотация: ако f и g се две функции които като стойности имат реални числа и при определени положителни цели числа, за кой записваме $f = O(g)$ и когато съществуват позитивни константи c и L така що $f(l) \leq cg(l)$ за всички $l \geq L$. Неформално това означава асимптотично (asymptotically), $f(l)$ не расте по-бързо от $g(l)$ в рамки на константни множители. Също полезна е "little-o" нотация. Записваме $f = o(g)$ ако за положителната константа c, съществува константа L така че $f(l) \leq cg(l)$ за $l \geq L$. Неформално това означава че $f(l)$ става незначително малко в сравнение с $g(l)$ за големи стойности на l.

Се приема идеята че ефективен алгоритъм е алгоритъм на който времето за изпълнение е ограничено от входния размер на полинома.

Определение 1. Нека A е алгоритъм за който имаме влез с битова дължина l.

(i) A е *polynomial-time* алгоритъм ако времето за изпълнение е $O(l^c)$ за някоя константа $c > 0$.

(ii) A е *exponential-time* алгоритъм ако времето за изпълнение не е в форма $O(l^c)$ за някоя константа $c > 0$.

(iii) A е *subexponential-time* алгоритъм ако времето за изпълнение $O(2^{0(l)})$, и A не е polynomial-time алгоритъм.

(iv) A е *fully-exponential-time* алгоритъм ако времето за изпълнение не е в форма $O(2^{0(l)})$.

Треба да се отбележи че *subexponential-time* алгоритъма, е също *exponential-time* алгоритъм, но не е polynomial-time алгоритъм. Въпреки това времето за изпълнение на алгоритмите с под експоненциално време расте по бавно от това на алгоритмите с напълно

експоненциално време. Под експоненциалните функции възникват когато възниква анализата за време на изпълнение на алгоритъм за разлагане на множители (factoring) на целите числа и намирането на DL (discrete logarithms).

Пример 1. (*subexponential-time* алгоритъм) Нека *A* да бъде алгоритъм чий то вход е цялото число *n* или малък набор от цели числа по модул *n* (така че на входа размера е $O(log_2 n)$). Ако времето за изпълнение на A е в форма на

$$L_n[\alpha, c] = O(e^{(c+O(1))(logn)^{\alpha}(log\ logn)^{1-\alpha}})$$

къде c е положителна константа а α е константа която задоволя $0 < \alpha < 1$, тогава A е *subexponential-time* алгоритъм. Ако $\alpha = 0$ тогава $L_n[0, c]$ тогава израза е полином в $log_2 n$ (така че A е *polynomial-time* алгоритъм), а ако $\alpha = 1$ тогава $L_n[1, c]$ е напълно експоненциален израз в $log_2 n$ (така че *A* е алгоритъм с *fully-exponential-time* алгоритъм). Параметъра α е добър показател за това че колко са близки *subexponential-time* алгоритъм за да бъде ефективно (polynomial-time) или неефективно (fully-exponential-time).

2.3.1 Разлагане на цели числа на множители (factorization) и DLP (discrete logarithm problems)

Следва на кратко описание на изследваната за разлагане на цели числа на множители, DLP и ECDLP (elliptic curve discrete logarithm problems).

Алгоритъм за проблема на целочисленото математическо разлагане на множители (integer factorization problem) подсещане че проблема на разлагане на цели числа е цялото число *n* което е продукт на два *l*/2-битни изрази, размера на входа е *O(l)* бита. Най-бързия известен алгоритъм за математическо разлагане на множители ако *n* е известен като *Number Field Sieve* (NFS) което очаквано време на изпълнение е под експоненциално

$$L_n\left[\frac{1}{3}, 1.923\right], \qquad (1.0)$$

NFS има два етапа: (*sieving stage*) етап на който са събрани някои отношения, и етап матрица (*matrix stage*) къде се решава голяма система на линейни уравнения. Sieving етапа е лесен за parallelize, и може да се изпълни с събиране от работните станции в интернет. Въпреки това за sieving да бъде ефективен, всяка работна станция трябва да има голяма памет. Етапа на матрицата не е лесен да се parallelize, поради това че отделните процеси често трябва да комуникират един с друг. Тези етапа е по-ефективна в изпълнението над една масова паралелна машина, от колкото в съединена мрежа от работни станции.

Алгоритъма за DLP (discrete logarithm problem) DLP ги има следните параметри *p* и *q* където *p* е *l*-бита а *q* е *t*-бита, и е делител на *p*−1, при входен размер *O(l)* бита. Най-бързия

известен алгоритъм за решение на DLP е *Number Field Sieve* (NFS) за което очаквано време на изпълнение е под експоненциално

$$L_{\mathrm{p}}\left[\frac{1}{3}, 1.923\right], \qquad (1.1)$$

докато *Pollard's rho algorithm* очаквано време на изпълнение

$$\sqrt{\frac{\pi q}{2}} \qquad (1.2)$$

В коментарите по-горе за NFS за целочисленото разлагане се прилага и NFS за изчисление на дискретен логаритъм. *Pollard's rho algorithm* може лесно да се parallelized така че не се налага отделните процеси да комуникират помежду си и само от време на време комуникират с централния процесор. Освен това при събиране алгоритъма изиска много малко памет за съхранение и малки ресурси от главната памет.

Избора на метода за решение на дадена илюстрация на DLP зависи от размера на параметрите p и q, които се определят от изразите (1.1) и (1.2). На практика, DL параметрите са избрани така че очакваното време на изпълнение в изразите (1.1) и (1.2) са приблизително равни.

Алгоритъм за ECDLP (elliptic curve discrete logarithm problem) припомняне че ECDLP изиска цяло число $d \in [1,n-1]$ така че $Q = d*P$, където n е t-бита, а P е точка от реда n на елиптичната крива определена над крайно поле F_p, и $Q \in <P>$. Ако приемем че $n \approx p$, което обикновено се случва на практика, тогава размера на входа е $O(t)$ бита. Най-бързия известен алгоритъм за решение на ECDLP е *Pollard's rho algorithm* където очаквано време на изпълнение е под експоненциално

$$\frac{\sqrt{\pi n}}{2} \qquad (1.3)$$

В коментарите по-горе за *Pollard's rho algorithm*, за изчисление на обикновен DLP също се прилага и за решение на ECDLP.

2.3.2 Сравнение на размерите на ключовете

Оценките са дадени за размера на параметрите който предоставят сравнителни нива на сигурност за RSA, DL, и EC системи, при допускане че горе споменатите алгоритми са наистина най-добрите който съществуват за целочисленото разлагане на множители, DL и ECDLP. По този начин не отчитаме фундаментални открития за бъдеще, като на пример откриването на значително по бързи алгоритми или изграждане на мащабен квантов компютър.

Ако времето е единствена мярка за използване и ефективност на алгоритъм, тогава размера на параметрите който предоставят сравнителни нива на сигурност за RSA, DL и EC системи може да получат с използване на изразите за време на изпълнение (1.0), (1.1), (1.2) и (1.3). Размера на параметрите, също така наречен размер на ключа, който осигуря сравнително ниво на сигурност за RSA, DL и EC системи при 80, 112, 128, 192 и 256-битови схеми за шифроване с симетричен ключ са изброени в табела 2. С ниво на сигурност от k бита може да означава че най-добрия алгоритъм известен за сриване на системата отнема приблизително 2^k стъпки. Избрани са пет специфични нива на сигурност, защото те представят необходим размер за изпълнение на схемите за шифроване с симетричен ключ, като SKIPJACK, Triple-DES, AES-Small, AES-Medium, и AES-Large, съответно.

Сравнението на размера на ключовете в табела 2 е до известна степен незадоволителна, доколко те се основат само на метриката време, необходима за NFS и Pollard's rho алгоритъм. Специфично NFS има няколко ограничени фактори включително ограничение на размера на паметта, необходима за

Security level (bits)					
	80 (SKIPJACK)	***112 (Triple-DES)***	***128 (AES-Small)***	***192 (AES-Medium)***	***256 (AES-Large)***
DL parameter q EC parameter n	160	224	256	384	512
RSA modulus n DL modulus p	1024	2048	3072	8192	15360

Таблица 2**.** RSA, DL и EC размер на ключовете за еквивалентни нива на сигурност. Битовата дължина са дадени за DL параметър q а за EC параметър n, и за RSA модул n и за DL модул p, съответно.

на етапа sieving, големината на матрицата, както и трудността на parallelizing на етапа на матрицата, тъй че тези фактори не са взети в предвид при анализа с *Pollard's rho* алгоритъма. Възможно е да се предостави стойностно еквивалентен (*cost-equivalent*) размер на ключа който вземат предвид цялата стойност на алгоритмите – това е времето за изпълнение, както и стойността за изграждане или придобиване на необходимия хардуер. Въпреки това стойностна е трудно да се оцени с разумна степен на точност. Последните изследвана показват че общата стойност на sieving и matrix етапа може значително да се редуцира чрез изграждане на потребителски хардуер. Поради това изгледа разумно консервативно да се подходи и да се използват времето като мярка на ефективност за NFS и Pollard's rho algorithms.

Сравнението в табела 2 ни показва че криптографията с елиптичната крива (ECC) използва от гледна точка на размера сравнително са по-малък за дадено ниво на сигурност от колко при RSA и DL системи. Разликата в размерите на параметри е особено изразена

при по-високите нива на сигурност. Предимствата които могат да се получат от системите с по-малки параметри включват скорост (по-бързи изчисления) и по-малки ключове и сертификати. По специфично е това че операциите с публичен ключ (като генериране на дигитален подпис и де шифроване) за ECC са много по-ефективни от колко при операциите с публичен ключ в RSA и DL. Също така операциите с публичен ключ (като верификация на дигитален подпис и шифриране) за ECC са много пати по ефективни от колко за DL системите.

Операциите с публичен ключ за RSA се очаква да бъдат малко по-бързи от операциите на ECC при малък показател за шифроване e (така че $e = 3$ или $e = 2^{16} + 1$) е избран за RSA. Предимството на ECC може да бъде важно в среди къде мощността на процесора, паметта, честотната лента (bandwidth), или консумирането на мощност е ограничена.

Следва още едно сравнение по дължини на ключовете. Сравнение с дължините на ключовете кои се изискат за ECC, Integer factorization (RSA) и традиционните схеми с симетричен ключ се посочени в табела 3, приспособено от [SEC 1, 2000].

Symmetric scheme	ECDLP based scheme	Integer factorisation based scheme
56	112	512
80	160	1024
112	224	2048
128	256	3078
192	384	7680
256	512	15360

Таблица 3: Сравнение на дължините на ключовете

Малката дължина на ключовете е полезно за системите къде ключовете трябва да разменят в мрежата и да се складират в устройства с ограничена памет. Въпреки това малките дължини на ключовете като резултат може да дойде до по бързо изпълнение на схемите, що е полезно в системите къде изпълнението в реално време представлява важен фактор, също така където важен фактор е консумирането на енергия (колко то е по-бързо изпълнението на схемата толко е по-малка консумацията на енергия).

2.4 Карта за изследване на елиптичните криви

Преди да насочим внимание на системите с елиптични криви треба да се обяснят следните понятия по отношение на крайно поле, елиптични криви и протоколи на криптографията:

1. Крайни поле, представляване на елементите на полето и алгоритмите за осъществяване на аритметиката на полета.

2. Елиптични криви, представляване на точките на елиптичната крива, и алгоритмите за осъществяване на аритметиката на елиптичната крива

3. Протоколи, и алгоритми за изпълнение на аритметиката на протоколите.

Има много фактори които могат да влияят на избора. Всички те трябва да се разглеждат едновременно, за да се стигне до най-доброто решение за конкретното приложение.

От практическо значение се включват следните фактори обмисляне на сигурността, прилагане на платформа (софтуер и хардуер), ограничения на специалната компютърна среда (на пример бързина на процесора, размер на кода (ROM), размер на паметта (RAM), брой на порти, консумиране на мощност, като и ограничения на специалната комуникационна среда (пр. честотна лента - bandwidth), време на реакция (response time).

Не изненадващо и особено трудно, а до известна степен и невъзможно, да се произнесем по единствен „най-добър" набор избрани параметри. На пример при оптимален избор на набор за приложение за работна станция може да е доста различно при оптимален избор за приложение за смарт карта.

2.5 Математическа основа

2.5.1 Въведение в теорията на крайните полета

Полета (Fields) са абстрактни алгебрични структури. Например множествата на рационалните числа $\mathbb{Q}$, реалните числа $\mathbb{R}$, и комплексните числа $\mathbb{C}$ с операциите събиране и умножение образуват числови полета. Поле $\mathbb{F}$ е множество от поне два елемента, в което са въведени две операции, събиране (означено с +) и умножение (означено с ·), които отговарят на основните аритметични операции:

(i) ($\mathbb{F}$,+) е адитивна абелева група с нулев елемент, означен с 0.
(ii) ($\mathbb{F} \setminus \{0\}$, ·) е мултипликативна абелева група с единичен елемент, означен с 1.
(iii) $(a+b) \cdot c = a \cdot c + b \cdot c$ за всеки $a, b, c \in \mathbb{F}$.

Поле $\mathbb{F}$, което съдържа краен брой елементи, се нарича крайно поле (finite field). В тази част представяме основните свойства на крайните полета.

2.5.2 Операции в поле (Field operations)

Освен двете операции събиране и умножение, в полето $\mathbb{F}$ можем да дефинираме още две операции – изваждане и деление. Ако a, b $\in \mathbb{F}$, то a $-$b = a $+(-$b) където $-$b е противоположния елемент на b в $\mathbb{F}$, така че b+($-$b) = 0. По подобен начин дефинираме и делението: ако a, b $\in$ F, b $\neq$ 0, a/b = a $\cdot$ b^{-1} където b^{-1} е обратния елемент на b в $\mathbb{F}$, т.е. b $\cdot$ b^{-1} = 1.

2.5.3 Съществуване и единственост (Existence and uniqueness)

Редът на крайното поле е броя на елементите на това поле. Съществуват крайни полета $\mathbb{F}$ от ред q тогава и само тогава, когато q е степен на просто число, т.е. q = p^m където p е просто число, наречено характеристика на $\mathbb{F}$, и m е положително цяло число. Ако m = 1,

тогава $\mathbb{F}$ се нарича просто поле. Ако m ≥ 2, тогава $\mathbb{F}$ се нарича съставно поле. За всяко q, по същество има само едно крайно поле от ред q. Неформално това означава, че всеки две крайни полета от ред q са структурно еднакви, само че използваното обозначението за представяне на елементите на полето са различни. Ние казваме, че всеки две полета от ред q са изоморфни. Крайно поле с q елемента означаваме с $\mathbb{F}_q$.

2.5.4 Просто поле (Prime fields)

Нека p е просто число. Тогава $\mathbb{F}_p$ = {0,1,2, . . ., p −1}, като операциите събиране и умножение се изпълняват по модул p. Например:
$\mathbb{F}_p$ (p=2 – тук ти го напиши) = {0,1}, 1+1=0
$\mathbb{F}_p$ (p=3 – тук ти го напиши) = {0,1,2}, 1+1=2, 1+2=3, 2+2=1, 2.2=1 (за 2+2 вместо 4 взимаме остатъка на 4 при деление на 3)

2.5.5 Двоични полета (Binary fields)

Крайни полета от ред 2^m се наричат двоични полета. Един от начините за конструиране на $\mathbb{F}_{2^m}$ е с използване на *polynomial basis representation*. Тук елементите на $\mathbb{F}_{2^m}$ са двоични полиноми (полиноми, чиито коефициенти са в поле $\mathbb{F}_2$ = {0,1}) от степен най-много *m* −1:

$$\mathbb{F}_{2^m} = \left\{a_{m-1}z^{m-1} + a_{m-2}z^{m-2} + \cdots + a_2z^2 + a_1z + a_0 : a_i \in \{0,1\}\right\}$$

2.5.6 Аритметика над прости полета (Prime field arithmetic)

При изграждане на приложения, в което се използва аритметика над прости полета, трябва да се обърне внимание на изграждането на алгоритмите. Използването на ефективни алгоритми в голяма степен ще увеличи ефективността на софтуера или съответно на приложението, в което се използва. Може да се ускори значително, ако числото *p* има подходяща стойност, с което се увеличава ефективността а с това и възможността за изпълнение във вградени системи. Предполагаме, че платформата за изпълнение има W- битова архитектура, където W е кратно на 8. Работните станции (Workstations) обикновено са 64- или с 32-битова архитектура. Ниско мощните платформи може и да имат по малко W, например вградените системи се 16-битови а смарт картите може да имат W = 8. Битовете на W- битовата дума U са номерирани от 0 до W −1, с най-десният бит на *U* назначен като бит 0.

Елементите на $\mathbb{F}_p$ са цели числа от 0 до $p - 1$. Нека $m = \lceil \log_2 p \rceil$ е битовата дължина на *p,* и $t = \lceil m/W \rceil$ е дължината на думата. Фигура 17. илюстрира случай, в който двоичното представяне на елемент *a* е съхранено в масив *A = (A*[*t* − 1], . . . *A*[2], *A*[1], *A*[0]*)* от *t W*-битова дума, където най-десният бит на *A*[0] е най-малко значещия бит.

A[*t* −1]	· · ·	*A*[2]	*A*[1]	*A*[0]

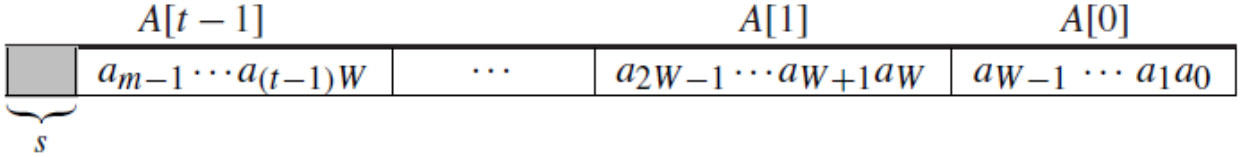
Фигура 17. Представянето на $a \in \mathbb{F}_p$ като масив *A* от *W*-битова дума. Като цяло число, $a = 2^{(t-1)w} A[t-1] + \cdots + 2^{2w} A[2] + 2^w A[1] + A[0]$.

2.5.7 Аритметика над двоичните полета (Binary field arithmetic)

При изграждане на приложение, в което има аритметика над двоични полета, трябва да се обърне внимание при изграждане на алгоритъма. Използването на алгоритъм, който ще е ефективен при извършването на аритметиката над двоични полета, в голяма степен ще увеличи ефективността на софтуера или съответно на приложението, в което се използва.
Предполага се, че се изпълняват за W-битова архитектура където W е кратно на 8. В бита на W-битовата дума U е номериран от 0 до $W-1$, с най-десният бит на U назначен като бит 0. Използват се следните стандартни означения за операциите на думите U и V:

$U \oplus V$ *изключващо или* - bitwise exclusive-or
U & V и - bitwise AND
$U \gg i$ *циклично преместване надясно* на U с i позиции
$U \ll i$ *циклично преместване наляво* на U *с* i позиции.

Нека $f(z)$ е неразложим двоичен полином от степен m, за който $f(z) = z^m + r(z)$. Разглеждаме елементите на $\mathbb{F}_{2^m}$ като двоични полиноми от степен най-много $m-1$.
Събирането на елементите на полето е събиране на двоични полиноми. Умножението се извършва по модул $f(z)$. Елементът на полето $a(z) = a_{m-1}z^{m-1} + a_{m-2}z^{m-2} + \cdots + a_2z^2 + a_1z + a_0$ е свързан с двоичния вектор $a = (a_{m-1}, a_{m-2}, \ldots, a_2, a_1, a_0)$ с дължина m. Нека $t = [m/W]$ и нека $s = Wt - m$. В приложението може да се съхранява в масив на t W-битовата дума:
$A = (A[t-1], \ldots, A[2], A[1], A[0])$, където най-десният бит на $A[0]$ е $a0$, а най-левият s бит на $A[t-1]$ не се използва (винаги треба да е 0).



Фигура 18. Представянето на $a \in \mathbb{F}_{2^m}$ като масив на A от W-битова дума. $s = tW - m$ най-високия бит на $A[t-1]$ остава неизползваем.

2.5.8 Оптимална аритметика за съставни полета (Optimal extension field arithmetic)

В предишните секции стана дума за аритметиката над поле $\mathbb{F}_{p^m}$ в случая $p = 2$ (двоично поле) и $m = 1$ (просто поле).
При хардуерни приложения двоичните полета са най-подходящи, тъй като операциите са свързани с битова смяна и събиране по модул 2. Простотата е атрактивна за имплементация върху приложения и за общо предназначени процесори, въпреки това умножението може да бъде много по-бавно от умножение в прости полета, ако е на разположение хардуерно умножение. От друга страна аритметиката за прости полета може да бъде по-трудно за ефективна имплементация, което се дължи на предаването на бит за пренос.
Основната идея на оптималната аритметика за разширени полета е да се изберат подходящи p *и* m, за да се получат полиноми, които отговаря на основните характеристики на хардуера.

Определение: Оптимално съставно поле *(Optimal extension field* - OEF) е крайно поле $\mathbb{F}_{p^m}$ такова че:

1. $p = 2^n - c$ за някое цяло число n и c със $\log_2 2\ |c| \leq n/2$
2. Съществува неразложим (irreducible) полином $f(z) = z^m - \omega$ над $\mathbb{F}_p$, който използваме за пораждащ полином на полето $\mathbb{F}_{p^m}$.

Ако $c \in \{\pm 1\}$, тогава OEF е от *Type I* (избраме *p да* е Мерсеново просто число (Mersenne prime) ако $c = 1$),

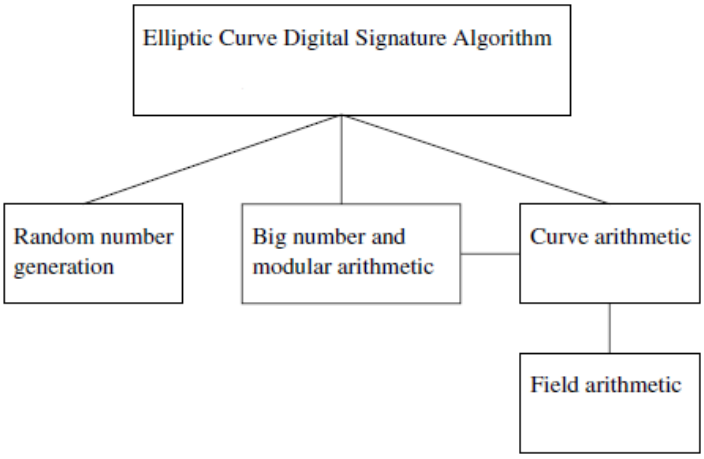
Ако $\omega = 2$, тогава OEF е от *Type II*.
Type I OEF имат особено проста аритметика над полето $\mathbb{F}_p$, докато Type II OEFs позволява опростяване на аритметиката в $\mathbb{F}_{p^m}$. Пример на OEFs е посочено в таблица 4.

p	f	parameters	Type
2^7+3	$z^{13}-5$	$n=7,\ c=-3,\ m=13,\ \omega=5$	—
$2^{13}-1$	$z^{13}-2$	$n=13,\ c=1,\ m=13,\ \omega=2$	I, II
$2^{31}-19$	z^6-2	$n=31,\ c=19,\ m=6,\ \omega=2$	II
$2^{31}-1$	z^6-7	$n=31,\ c=1,\ m=6,\ \omega=7$	I
$2^{32}-5$	z^5-2	$n=32,\ c=5,\ m=5,\ \omega=2$	II
$2^{57}-13$	z^3-2	$n=57,\ c=13,\ m=3,\ \omega=2$	II
$2^{61}-1$	z^3-37	$n=61,\ c=1,\ m=3,\ \omega=37$	I
$2^{89}-1$	z^2-3	$n=89,\ c=1,\ m=2,\ \omega=3$	I

Таблица 4. OEF примерни параметри.Тук, $p = 2^n - c$ е просто, а $f(z) = z^m - \omega \in \mathbb{F}_p[z]$ е неразложим над $\mathbb{F}_p$. Полето е $\mathbb{F}_{p^m} = \mathbb{F}_p[z]/(f)$ от ред приблизително 2^{mn}.

2.6 Аритметика на елиптичните криви (Elliptic Curve Arithmetic)

Криптографските механизми основани на елиптични криви зависят от включените аритметични операции над точките на кривата. Както що казах предходно, аритметиката на кривите е определена съществено за операции над полета където е от съществено значение ефективността. Ефективноста на операциите на кривите също така са от решаващо значение за критичните изпълнения.
Фигура 19 ни илюстрира модул от рамка за работа (framework) изискана за протокол като Elliptic Curve Digital Signature Algorithm (ECDSA). Аритметиката за кривата не че е изградена само от операции над полета, но в някои случаи също така се изхождат големи броеве и се използва модуларната (modular) аритметика (т.е., τ -adic операция в кривите на Koblitz се използва). ECDSA използва хеш (hash) функции, и определен брой на моулни операции, но при стъпките на комплексни изчисления са включени операции на криви.



Фигура 19. ECDSA съставни модули.

2.6.1 Представяне на елиптичните криви (Introduction to elliptic curves)

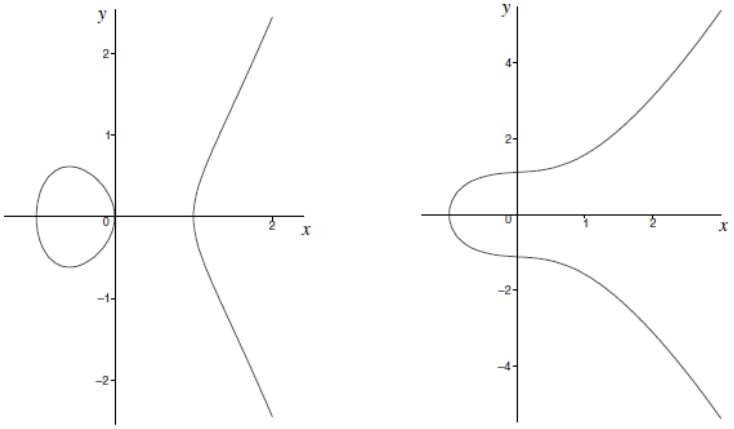
Определение: Елиптична крива E над поле K е дефинирана с уравнението

$E: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ (2.1)

където $a_1, a_3, a_2, a_4, a_6 \in K$ и $\Delta \neq 0$, където е дискриминантен (*discriminant)* на *E* и се определя на следния начин:

$$\left.\begin{aligned} \Delta = -d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6 \\ d_2 = a_1^2 + 4a_2 \\ d_4 = 2a_4 + a_1a_3 \\ d_6 = a_3^2 + 4a_6 \\ d_8 = a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2 \end{aligned}\right\}$$

Ако *L* е някое разширено поле на *K*, тогава набор на *L*-рационалните (rational) точки на *E* *са E(L)* = $\{(x, y) \in L \times L: E: y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6 = 0\} \cup \{\infty\}$ където ∞ е безкрайна точка.



(a) $E_1: y^2 = x^3 - x$ (b) $E_2: y^2 = x^3 + \frac{1}{4}x + \frac{5}{4}$

Фигура 20. елиптични криви над ℝ реални числа.

Забележка (*коментари по определението за елиптичните криви*)

(i) уравнението (2.1) се нарича уравнение на *Weierstrass*.

(ii) Казваме че *E* е определен върху *K* тъй като коефициентите a_1, a_3, a_2, a_4, a_6 от уравнението са елементи на *K*. По някога записваме *E*/K да изтъкнем че E е дефиниран над *K*, и *K* се нарича основно (underlying) поле. Ако *E* е дефинирано над *K*, тогава *E* е също дефинирано над всяко разширено поле на K.

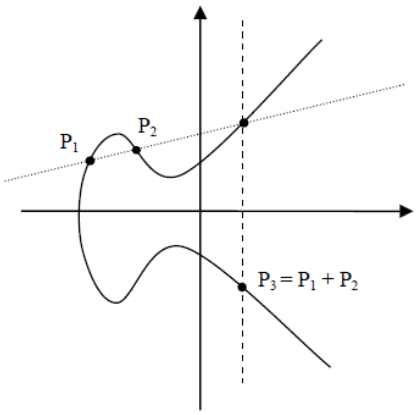
(iii) Условието $\Delta \neq 0$ гарантира че елиптичната крива е "гладка - smooth", няма точки и кривата има две или повече различни тангент (tangent) линии.

(iv) Точката ∞ е единствената точка от линията на безкрайност която отговаря на проекционната формата на уравнението на Weierstrass.

(v) *L*-рационалните (rational) точки на *E* са точките (x, y) които отговарят на уравнението на кривата и чиито координати x и y принадлежат на *L*. Безкрайната точка се разглежда в *L*-рационалните точки за всички разширени полета *L* на *K*.

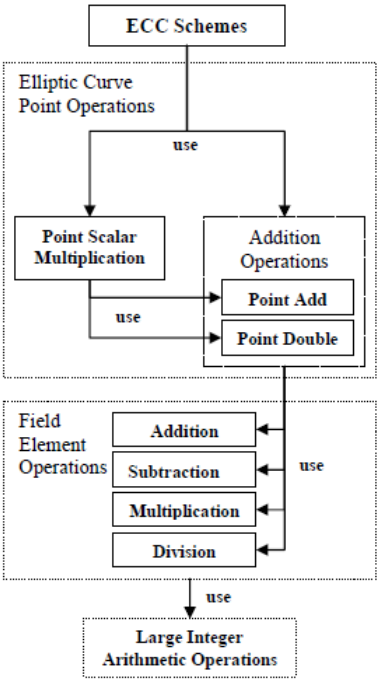
2.6.2 Операции изискани от ECC (Operations required by ECC)

Скаларното умножение, или многократното на събиране на точките на ЕС е главна операция изискана от схемите на ECC, също така може да се изискат и други операции както деление. За точно цяло число, при математическите аритметични операции за подразбираща се система за шифроване / де шифроване, те са бавни и може би и уникални за прилагането в тази област, откакто всичките околни схеми автоматично се изключват. Събирането на две ЕС точки, е илюстрирано на фигура 21.



Фигура 21: Илюстрация на събиране на две ЕС точки.

Операцията с скаларното умножение управляват с действителното изпълнение и нужно време за ECC съществуването е критично. Действителните математички операции зависят от избраната крива и зададеното поле, [Blake, I et al. 1999] както и да е постои ясна йерархия на основните математически операции, тя се показани на фигура 22.



Фигура 22: Йерархия на изискани основни математически операции

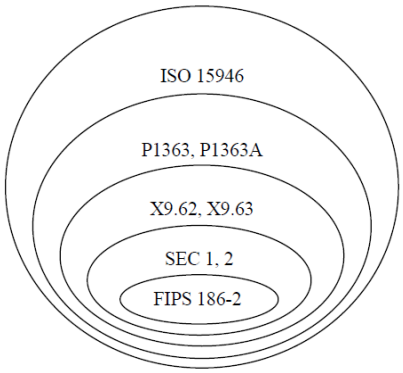
ЕС техниката за компресия на точка също така изиска квадратен корен изчислен в тази област. Всичките операции на полета изискат аритметични операции с цел брой, така че дължината на операторът значително е по-голяма от големината на компютърна дума. Ниското ниво на аритметичните операции с цели числа е може би най-доброто изпълнение достъпно за кодиране при рутината на асемблиране. Ясно се вижда че при изпълнението на ECC критичен е избора на оптимален алгоритъм на всяко ниво. Избора на алгоритъм зависи от избора на кривата и типът на крайни полета, и след това може да се оптимизира в зависимост от хардуерът. Ясно е че в дистрибуционните симулационни среди и замесените устройства може да имат много различни архитектури, изчислителни способности, ресурси и т.н.
Така че за оптималното изпълнение, за подразбиращ математически алгоритъм полезно би било да се конфигурират на ниво на устройството.

2.7 ECC стандарти (ECC Standards)

Напоследък голям брой на стандарти са публикувани, от кой за всеки стандарт се задават различни препоръки в отношение на използването на ECC технологиите. Обикновено са включени

- Стандартизирани версии на съвместими ECC схеми
- Предварително избран набор на EC параметри
- Стандартен формат за опис на ECC ключове, набор параметри, и други криптографски елементи.

В момента съществуват 5 главни стандарти, погледнете [Groβschadl, J. 2002] при неотдавнашните изследвана заключило се че те съществено се различават по техните препоръки. На фигура 23 е показано съвместимостта между стандартите.



Фигура 23: Interoperability между ECC стандартите

2.8 ECC имплементация на система (ECC SYSTEM IMPLEMENTATION ISSUES)

Има много работи които требва да се дискутират които от една страна са математичките основи на ECC схемите, а от друга страна практическата работа на ECC системите (подробно е описано в магистърската теза на MSc [Roberts P.H., 2004]). Следва листа която и покрай това че не е цялостна, представлява индикативен списък на релевантните въпроси. От това ще видим че треба да направим много решения на няколко нива.

EC Параметри (EC Parameters). ECC схемите действат на под групите от EC точките на елиптичната крива. Требва това да го оценим защото съществуват голям брой на възможни криви и групи на точки, които могат да се използват. Някои класи на криви съдържат математически структури които придават кон тях по ефективно математическо изпълнение, докато другите класи позволяват по ефективно изчисление на ECDLP и затова не са криптографски сигурни.

Въпроса за сигурността на кривата ще се дискутира покъсано, както и подредбата за устройване на ECC схемата. Необходимо е да се избере една група на точки, тоест всички потребители използват съща група точки на съща крива над също крайно поле.

Този група от точки може да се установи като EC набор на параметри, за който може да се дефинират като множество от 7-елементи,

(a, b, q, G, n, h, Fr)

къде, **a** и **b** дефинират характеристична елиптична крива, **q** идентифицира избрано основно крайно поле, **G** е основна точка коя генерира под група от точки на избраната крива. Тези точки ще се актуални точки които ще се използват от страна на схемата, **n** ни го дава броя на точки на специфицираната под група и **h** ни го дава общия брой на точки на избраната крива. **Fr** може да се използва за да ни даде индикация за представяне на използваните основни елементи на полето. Това е важно за характеристични две разширени полета, къде що съвкупност от различни основни елементи на полето се представени.

Избор на набор параметри (Parameter set selection*).* Избора на набор параметри може да бъде от публичната листа, но тези листа може да се проучена от злонамерения. Алтернатива тук е избора на частен набор на параметри. Това включва избор на крайни полета, генериране на крива, по що следва избора на подходяща под група и опция за

представяне на полета. Броя на точки на кривата може да се провери с използване на алгоритъма на Schoof's [Schoof, 1985].
Нататък следва използване на техника за генериране на случаен брой за избор на крива, над крайно поле, с използване на произволен носител на стринг, което също е включено в набора параметри за верификация. Това е опит да се избегне възможността за избиране на крива с скрити криптографски слабости, която ще го намалят нивото на ефективната сигурност.
Сигурност (Security). Нивото на сигурност предложено от ECC схемите, главно е определено с тежината на решаването на ECDLP над група от точки на EC използвана за тези схема. Това е проста подредба на циклична под група от точки на подходяща елиптична крива. Табела 5 [ANSI X9.62, 2001], ни приказва за разумни зададени дължини на n ни се изиска много дълг период за решаване на ECDLP, с използване на Rho алгоритъма [Pollard, 1987].
Съществуват голям брой на специални класи на криви къде се познати алгоритмите на под експоненциалната комплексност за решаване на ECDLP, погледни [Roberts. P.H 2004]. Тези криви трябва да се не се използват защото изискат по-малък период за изчисление.

Bit size of n	$\sqrt{(\pi / 4)}$	MIPS years
160	2^{80}	$8.5 * 10^{11}$
180	2^{93}	$7.0 * 10^{15}$
234	2^{117}	$1.2 * 10^{23}$
254	2^{177}	$1.3 * 10^{41}$
426	2^{213}	$9.2 * 10^{51}$

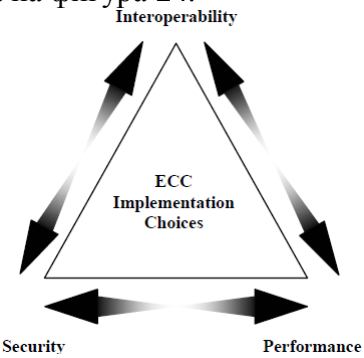
Таблица 5: Компютърна преценка за нужно време за решение на ECDLP при различни стойности на n

Треба да се отбележи че реда на основните точки от набора параметри **n**, го определя трудността за решаване на ECDLP и реда на полето **q** я определя големината на генерираните ключове. Ще спомена че в системите за сигурност се изиска повече от едно ниво на сигурност, отделни набори на параметри се изискат на всяко ниво и всеки потребител изиска отделна двойка ключове за всеки набор на параметри който се използва в системата.
Съвместимост (Interoperability). За да се постигне съвместимост (interoperability) на всички нива, на просторната система или вътрешната система, необходимо за всички потребители в схемата да има двойка ключове базирани на същия споделен набор EC параметри. На практика съществуват още две допълнителни изискания,

- •Използване на стандартизираните описи на криптографските елементи
- •Използване на стандартни версии на общите ECC схеми и протоколи за изпълнение. От сигурност ни причини възможно е редуциране на съвместимостта.

Изпълнение (Performance). Секцията на набор параметри, има голямо влияние над изпълнението, особено с отношение на алгоритмите за имплементация и изпълнение на основните математическите операции. Оптимизация с ефективно кодиране в асемблер или с имплементация на хардуера може да е необходимо за приложение в реално време (real-time). Ефективността на мрежата е зависима от съвместимите стандарти за форматиране, кодиране, също така и от ограничението на количеството предадени данни за определено време (bandwidth). На край винаги ще е трудно да се намери компромис между

съвместимостта (interoperability), изпълнението (performance) и сигурността (security), която зависимост е илюстрирана на фигура 24.



Фигура 24: ECC implementation tradeoffs

2.9 Подход за изпълнение (IMPLEMENTATION APPROACHES)

Оптимално изпълнение може да се получи чрез комплектно проектиране на цялата ECC система приспособена за нуждите на отделно приложение, но това е скъпо и се изиска време а и експертно познание. Втората алтернатива е подход на две нива, първото ниво е приложното, второто ниво е ниво на устройството / платформата на което ECC сервиса ще се използва:

2.9.1 Приложно ниво (Application Level Issues)

Се взема предвид следното:

- Изискано ниво на сигурност чрез избор на съответен набор на EC параметри
- Съответствие с стандартите
- Избора на криптографски схеми
- Форма за мрежов трансфер на ключове и други елементи на криптосистемата

2.9.2 Ниво на устройството (Device Level Issues)

Нивото на устройството се отнася за избора и оптимизацията на разположилите математически алгоритми, които се базират на избраният набор параметри на приложното ниво, за да ги осигури най-доброто изпълнение при зададено изчислителни ресурси на устройството за което се включват:

- Избора на външни елементи на полето и асоцииран математически алгоритъм.
- Избор на външни точки, и избор на оптимизация на алгоритъма за събиране на точки.
- Избор и оптимизиран алгоритъм за скаларно умножение, който се взема предвид при атаките на канал.
- Използване на специално предназначени полета, или използване на хардуер за целочислена аритметика в специфичното устройство.

2.9.3 Подход с комплект инструменти (Toolkit Approach)

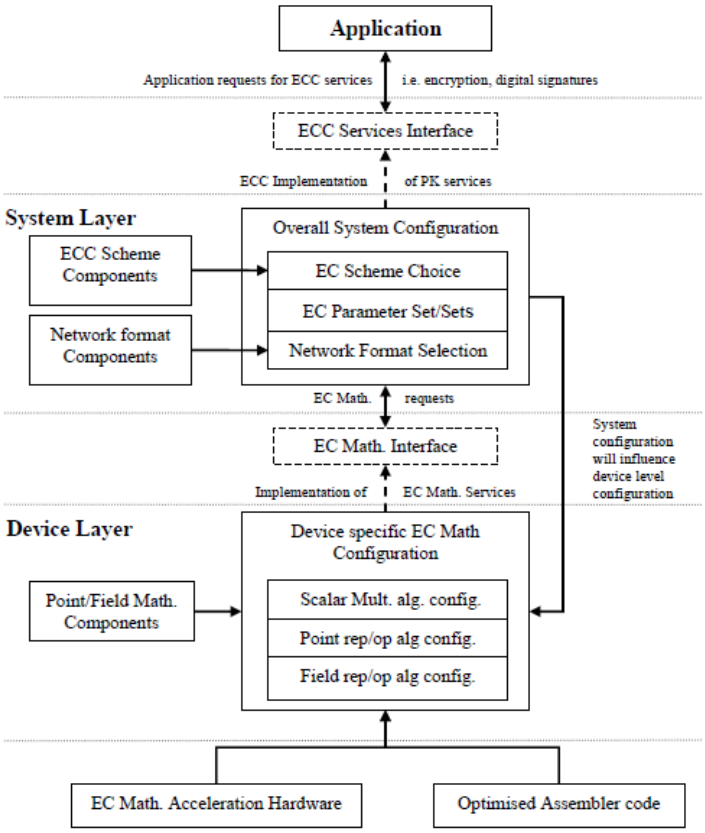
Новите ECC системи трябва да имат приспособим пристъп за изграждане на всяко ново приложение което изиска използване на ECC. Неизбежно е скъпо развоя на приложението.

Това в голяма част заема пристъпа за осигуряване на употребими средства за компонентите на ECC алгоритъма, които могат да се включат в единствена работна рамка (framework) за изграждане на потребителски ECC решения.

2.9.4 Предложение за ECC системна архитектура (ECC System Architecture Proposal)

Посочено е на фигура 25 архитектура на високо ново с две нива на пристъп.
Това включва два важни интерфейси:

- ECC интерфейс сервис между системното ниво и приложното.
- EC математички интерфейс между нивото на устройството и системното ниво (System Layer).



Фигура 25. Високо ниво на архитектура на две слоения пристъп с комплект инструменти

ECC интерфейсният сервис отеля приложните спецификации на системното ниво на което е осъществен избора от криптографските сервиси осигурени от конфигурацията на ECC системата. EC математичният интерфейс я отделя конфигурираната ECC система от избрания алгоритъм за изпълнение на основните математички операции на специфицираното устройство. Тези архитектура позволява разпределение на приложението на нива, кое се отнася на цялата система, с което се осигурява доминантния фактор, за провеждане на основните математички операции в рамките на устройството. Тези подялба представя полезна характеристика на системата с което се осигуря на потребителя да се концентрира на отредени важни въпроси.

Трета глава

Анализ на имплементация на ECC

3.1 Криптографски алгоритми

3.1.1 ECC

Опис: Double-and-Add Exponentiation in ECC

```
1       begin
2               y = y + y mod n /* double */
3               if e(j) == 1 then
4                       y = y + x mod n /* add */
5       end
```

Криптографския метод с елиптична крива е сравнително нов метод независимо открит от Koblitz и Miller в 1985. То се смята за важен пробив в тези област от страна на много изследователи. От 2005, U.S. National Security Agency (NSA) е отишла до толко далеч че го включили в Suite B набор от криптографски алгоритми, целта е да служи като основа при защита както на некласифицирана, така и на по-голямата част от класифицираната информация (http://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml). ECC е тип на Public Key Encryption (PKE), и подобно на първата генерация схеми за шифроване, се основа на варианта на Diffie-Hellman (DH). Докато стандартната PKC (Public Key Cryptography) се базира на факта че изчисляването на дискретен логаритъм (discrete log в форма на $\log_b: G \to Z_n$) е трудно. ECC зависи от идеята че е трудно да се намери и изчисли цел брой k, при дадени точки Q и P от елиптична крива така че Q = k*P. Това е операция която требва да се изпълни за да се получи текста (plain-text) от шифрирания текст (cyphertext) с използване само на публична информация (публичен ключ).

ECC се базира на елиптична крива в форма $y^2 = x^3 + ax + b$. Тези крива требва да се дефинира в голямо просто поле, както и координатите на кривата преминават чрез формираната група.

Съществуват препоръчани криви за използване в просто поле както 2^m. Фундаменталната операция в ECC е умножение на точка, където се изиска изчисление на Q (публичния ключ) в Q = k*P където k (частен ключ) а P е дадено. Докато атаката с грубата сила (brute force) е единствения начин да се извърши обратима операция, съществуват алгоритми които ще ви позволят ефективно изчисление на умножението на точка [21].

ECC се преференциални преди другите криптографски градивни блокови защото докато предните операции са значително лесни за изчисление със устройства с малка мощност, обратимите операции са значително по трудни, отколкото при стандартните PKE. На пример, покрай препоръките за големина на ключ за RSA от 2048 бита, за да се получи еквивалентно ниво на сигурност с използване на ECC требва да се използва ключ с размер от 224 бита. Компютърните системи стават се по-мощни, при което предимството на ECC ще става по очевидно.

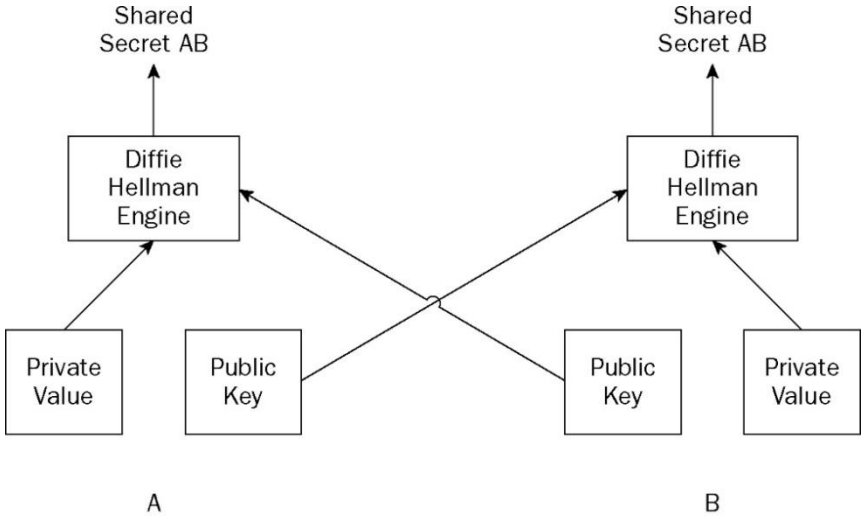
3.1.2 Elliptic Curve Diffie-Hellman (ECDH)

Първия протокол с публичен ключ е въведен от Diffie и Hellman. Сигурността на системата се базира на трудността на DLP.
Алгоритъма на Diffie-Hellman за разменна на ключовете може лесно да се изпълни с използване на елиптична крива.
Системата Diffie-Hellman работи на следния начин: Две страни който изискат да си изпращат шифровани съобщения един на друг, трябва да се споразумеят за симетричен ключ за шифроване с помощ на публично споразумение (ще е достъпно за всеки). Да предположим че имаме двама потребители Alice и Bob. Разпредялбата на ключа с използване на ECDH може да се изпълни с следната процедура (фигура 26):

• Alice иска да въведе общ ключ с Bob. Освен това всеки от двамата трябва да има двойка ключове, която се състои от частен и публичен ключ.
• Първо се избира крайно поле F_{p^m}
• Нека $\mathcal{E}$ е елиптична крива от крайно поле F_{p^m}
• След това се избира точка $P \in \mathcal{E}$
• Alice: избира „произволен" частен ключ – $d_A \in F_{p^m}$
  Изчислява публичен ключ - $Q_A = d_A * P$
• Bob: избира „произволен" частен ключ – $d_B \in F_{p^m}$
  Изчислява публичен ключ - $Q_B = d_B * P$
• Общ ключ: $K = d_A * d_B * P$
• Alice пресмята $d_A * Q_B$
• Bob пресмята $d_B * Q_A$

Лесно е да се покаже дека Alice и Bob споделуваат общ ключ (shared secret),

$$K = d_A * d_B * P$$



Diffie-Hellman Between Two Parties
Фигура 26.

Сигурността на ECDH се базира на K.
Elliptic Curve Diffie-Hellman Problem (ECDHP) можем да го определим на следния начин:
Определение: Дадени са G, Q ∈ Ɛ, така че: Q = k * G, където k е елемент от крайно поле. ECDHP е за намиране на k.
Изгледа неосъществимо изчисляването на K, d_A и d_B ако се знае само $P, d_A * P$ $и$ $d_B * P$. Функциите в ECDH се известни като необратими функции (one-way) [16].
Предимство на тези алгоритъм е в това че след генерирането на общ ключ, той може да се използва в симетрична криптосистема за период от време, до като не се генерира нов ключ.

3.1.3 ECDSA

ECDSA е форма на алгоритъм за дигитален подпис (Digital Signature) връз база на свойствата на елиптичните криви, по скоро като DLP. Дигиталният подпис представлява стойност изчислена от тайната стойност на съобщението, и представлява модерна замяна на саморъчен подпис.
ECDSA е асиметрична схема на подпис което осигурява устойчивост при атаки на съобщенията. Това означава че подписът и съобщението от което той произхожда са независими едно от друго [22].

Опис: Elliptic Curve Digital Signature Algorithm

```
1       begin
2               e = hash(m) /* Calculate the Hash of the message */
3       1:
4               Rand(k) mod n /* Choose a random k */
5               r = x1 mod n
6               if r == 0 then
7                       GOTO 1
8               s = k-1*(z + r*d) mod n mod n /* Exponentiation as in the previous examples */
9               if s == 0 then
10                      GOTO 1
11              return (r,s)
12      end
```

ECDSA вариант на DSA, който оперира с адитивната група на елиптична крива. В случая на TinyECC, домейн параметрите се генерирани на предварително по мощен уред (компютър), до като самия подпис се извършва на ниво на самия възел.
Този вариант използва по-малки ключове за същото ниво на секретност, както DSA. От друга страна, времето за изпълнение на основните операции и дължината на самия подпис са приблизително същите. Например, DSA с 1024-bit p и 160-bit q, и ECDSA над поле GF(p) за 160-bit просто p дават 320-битов подпис и се нуждаят само от няколко мили секунди за операции на 2 GHz Pentium.

3.2 Инсталиране на TinyOS-1.x

В тези секция се обяснява инсталацията на TinyOS 1.x. Съществуват повече начини и постъпки за инсталиране на TinyOS. До този момент съществуват TinyOS 1.x и TinyOS 2.x версии на оперативната система която е проектиране за ниско мощни сензорни устройства. Също така са пуснати версии TinyOS които могат да се използват под Windows и Linux. При процеса на инсталиране възможно е възникване на редица проблеми. Тук е представена постъпката низ няколко стъпки за инсталиране актуализиране на всичките необходими компоненти на TinyOS при инсталиране под Windows XP.

1. Инсталиране на tinyos-1.1.0-1is.exe

Изтегляне на tinyos-1.1.0-1is.exe от следната страна http://www.tinyos.net/windows-1_1_0.html, по изтегляне, се стартова файлът tinyos-1.1.0-1is.exe (требва да се избере "Complete" при стъпките за инсталиране).

2. Ъпгрейд на **Cygwin**

Изтегляне на cygwin-1.2a.tgz от следната страна http://www.tinyos.net/scoop/special/howto_upgrade_cygwin на самата страна са зададени инструкции за ъпгрейдуване на **Cygwin**

Cygwin представлява подобна Linux среда за Windows. Състои се от две части:

- DLL (cygwin1.dll) което се отнася като Linux API слой за емулация и осигуря реална Linux API функционалност.
- Набор от инструменти които ни я осигурят Linux среда.

Cygwin DLL моментално работи на всичките последни, комерсиално освободени x86 32 bit и 64 bit версии на Windows, с изключение на Windows CE.

3. Ъпгрейд на nesc

Изтегляне на nesc-1.1.2b-1.cygwin.i386.rpm от следната страна http://www.tinyos.net/dist-1.1.0/tinyos/windows/, в cygwin shell променете я директорията в директория където е зачуван файлът, инсталирането на новия nesc е с командата "rpm -ivh nesc-1.1.2b-1.cygwin.i386.rpm"

4. Рестартирайте компютъра

5. Ъпгрейд на tinyos-1.1.0 до **TinyOS 1.1.15 CVS Snapshots**

Изтегляне на tinyos-1.1.15Dec2005cvs-1.cygwin.noarch.rpm от следната страна http://www.tinyos.net/dist-1.1.0/snapshot-1.1.15Dec2005cvs/doc/install-snapshots.html, в cygwin shell променете я директорията в директория където е зачуван файлът, инсталирането на snapshot е с командата "rpm --force --ignoreos -Uvh tinyos-1.1.15Dec2005cvs-1.cygwin.noarch.rpm"

6. Проверка на инсталацията

В cygwin shell, "cd /opt/tinyos-1.x/tools/scripts/. Изпълнете я командата ./toscheck. Осигурите се дека нямате съобщения за грешки или предупреждаване (Може да не обръщате внимание на някои предупреждения свързани с msp430-gcc. Това е за telos motes). Постъпката е извършена

7. Проверка на PowerTossim

Посетете я следната страна http://www.eecs.harvard.edu/~shnayder/ptossim/install.html. Следете ги инструкциите "Basic instructions". Проверката е извършена ако ги извършите "Basic instructions" успешно.

Поради динамичната същественост на уеб пространството възможно е някои от горе наведените връзки да не се повече активни. Но това не трябва да представлява проблем защото с малко претърсване може да се намерят горе споменатите елементи които са нужни за напречена работа на TinyOS. Между другото съществуват и други постъпки за инсталиране на TinyOS, също така и постъпки за инсталиране на TinyOS на различни Linux дистрибуции.

3.2.1 Необходими модули за симулация и анализ

За симулация и анализ ще използваме вече споменатия TOSSIM които престава стандартна компонента от TinyOS. TOSSIM фокусира се на симулация на всеки бит на безжичната платформата mica. Разработените пакети които се използват за развой, разработка и изследване на приложения за платформата mica2.

Освен TOSSIM за нашето изследване ще използваме и PowerTossim и TinyViz.
Енергията е от решаващо значение и представлява ограничение за безжичните сензорни мрежи. Поради цената за разработване, прилагане и изграждане на една сензорна мрежа, е нужно да се направи изследвана на профила на мощността за дадено приложение, с цел да се спести цената и времето. С това ще сме на ясно за енергийните разходи нужни за работа на дадено приложение. PowerTOSSIM, представлява модулно разширение на TOSSIM. PowerTOSSIM ни представлява модел на консумация на мощност на приложенията за TinyOS. Тука е също включен подробен модел на консумация на мощност на Mica2 възли. PowerTOSSIM представлява стандартна компонента за версиите на TinyOS след 1.1.10.

Основни инструкции за PowerTOSSIM

1. компилиране на приложението
   ```
   $ make pc
   ```
2. в DBG се включва POWER. Ако не се нуждаем от други съобщения
   ```
   export DBG=power
   ```
3. стартиране на main.exe с включен флаг -p и записване на изхода в файл
   ```
   ./build/pc/main.exe -t=60 -p 10 > myapp.trace
   ```
4. стартиране на postprocess.py които ни дава резултати връз база на myapp.trace:

```
$TOSROOT/tools/scripts/PowerTOSSIM/postprocess.py --sb=0 –em
$TOSROOT/tools/scripts/PowerTOSSIM/mica2_energy_model.txt myapp.trace
```

Параметърът --sb определят дали възлите имат прикачено сензорна плочка (sensor board). Параметърът --em го определя енергийния модел. За подробности може да се използва postprocess.py --help при което дава описание за всички опции.

Представен е на изхода които се получава за CntToLedsAndRfm стартиран с `main.exe -t=60 -p 1`:

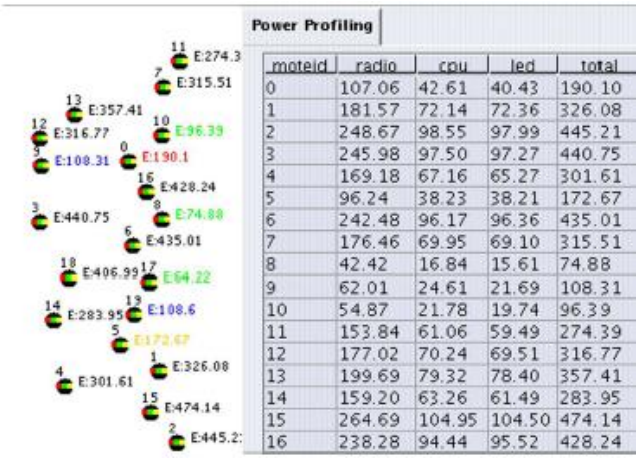
```
Mote 0, cpu total: 719.503906
Mote 0, radio total: 1235.255862
Mote 0, adc total: 0.000000
Mote 0, leds total: 571.570576
Mote 0, sensor total: 0.000000
Mote 0, eeprom total: 0.000000
Mote 0, cpu_cycle total: 0.000000
Mote 0, Total energy: 2526.330344
```

По подразбиране, TOSSIM използва стара версия на mica радио стек, който не поддържа управление на захранването и настройване на мощността на предаване. PowerTOSSIM включва порт на mica2 радио стека, така че сега е възможно да стартират програми, които се възползват от функциите на CC1000 за управление на захранването. За да се възползвате, от това се включва

PFLAGS + =-I% T/platform/pc/CC1000Radio

в файла Makefile от приложението.

TinyViz е графичен интерфейс на TOSSIM, който включва няколко плъгини за взаимодействие и анализ на състоянието на възли по време на изпълнение на приложението. TinyViz включва плъгин за анализ на мощноста (power profiling). Графичния интерфейс е представен на фигура 27.



Фигура 27. Графичен интерфейс на power profiling

Специфичното поведението на безжична връзка зависи от два елемента: радио и околната среда (канал), където са поставени. Следователно, за да се получат по-добри резултати от симулациите, трябва да се осигурят характеристиките на двата нужни елементи.

TinyViz ни дава възможност за тестване на сензорните мрежи при различна топология. Това са:

- GRID: възлите са разположени в квадратна grid топология. Броя на възлите трябва да бъде квадрат от цяло число. Се предполага дека сензорните възли са разположени според стандартна схема.
- UNIFORM: връз основа на броя на възлите (квадрат на цяло число), физически областта е поделена на брой на клетки. В рамките на всека клетка, възела се поставя случайно. Този топология може да се наблюдава като вариация на GRID.
- RANDOM: възлите са разположени случайно в рамките на физическата област.
- FILE: разположението на възлите се чете от файл дефиниран от потребителя.

PowerTOSSIM осигуря модул с която ги изчислява тактовите на CPU.

Погледнете [23] за подробно описание.
Симулацията на потребление на консумация на енергия от широко мащабно приложение за сензорни мрежи в голям процент дава реална фигура на реалната консумирана мощност от приложени за сензорни мрежи за платформа mica2 [24].

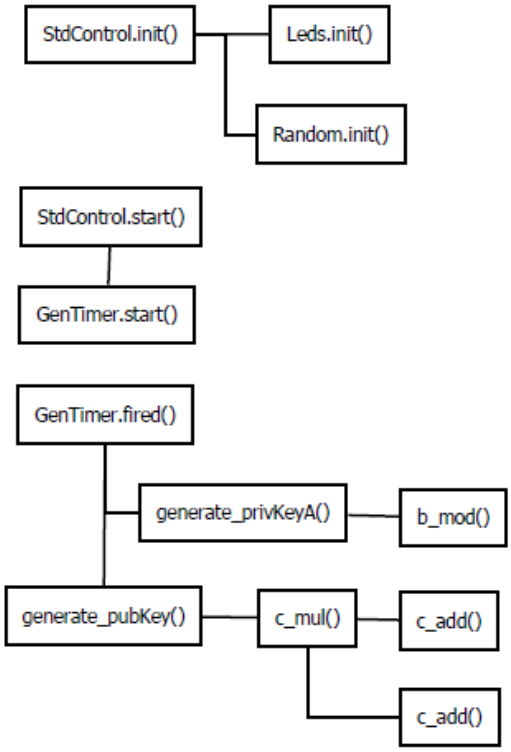
3.3 Основни характеристики на Mica2

Mica2 Mote е разработен от UC Berkeley е сензорен възел, който е с малка мощност. Той се базира на 8-bit Atmel AVR процесор 4MHz, 128KB за кодова памет, 512KB EEPROM, 4KB памет за данни SRAM, и ChipCon CC1000 радио с възможност за предаване с 38.4 Kbps (433, 868/916, or 310 MHz Multi-Channel Radio Transceiver) с обсег на предаване от близо 300m, 18 грама тегло, 2 AA батерии.

3.4 Анализ на EccM - EccM-2.0

Написано в 2004, EccM е първата известна имплементация на PKI която използва криптография по елиптична крива.
Намерение на авторите било ECC да го използват като метод за разпределение на 80 – битов симетричен ключ на TinySec.
Връз основа на дизайна на Dragongate Technologies Limited, Java-based jBorZoi 0.9 [25], EccM-2.0 е също така имплементация на ECC но с по-голям успех от EccM-1.0. Имплементацията на модулът за един възел е така че се избира случаен полином над F_{2^p}, за частен ключ след това с изчисления на кривата и базова точка препоръчана от NIST се изчислява публичният ключ за възела. В този версия умножението на точките се извършва с Алгоритъм IV.1 в [27]. Умножение на елементите в F_{2^p}, се реализира с Алгоритъма 2 в [26], докато инверсията се прилага с Алгоритъма 8 в [26].

Ключа тук е с дължина от 163-бита, EccM-2.0 оправя друг недостатък в 1.0. Тъй като 1.0 избира крива по случаен начин, се рискува (макар и с експоненциално малка вероятност) избора на крива коя да е уязвима на под- експоненциална атака с редукцията през MOV [29] с метода за изчисление на индекса [28]. EccM-2.0 по този начин се подчинява на препоръката на NIST за ECC над F_{2^p} [30].
По основното представяне на EccM ще обрнем внимание на модулите от кой е изграден. Приложението EccM, сдържи файл с име EccM.nc където Diffie-Hellman протоколът е описан за два възли за установяване на обща тайна с използване елиптична крива. Програмният поток е описан на фигура 28.



фигура 28: Логичен поток на EccM

Модулите и базовата точка се поставени в StdControl.start(). Настроен е таймер от три секунди, при които частният ключ е случайно генериран, и се използва за генериране на публичен ключ от степен (m^k). По връщането на експоненциалната функция, таймера се инициализира, и пакета с данните е подготвен и пратен. След като съобщението е пратено, и то е прието от комуникационния съученик, се генерира споделена тайна (generate_secret()).

Модула EccM.nc демонстрира жизнеспособността на ECC с изпълнение в MICA2 възел, с полином над поле GF_{2^p}. Параметрите на кода са следните:

Полином:

$$f(x) = x^{163} + x^7 + x^6 + x^3 + 1$$

Използва Koblitz Крива:

$$y^2 + xy = x^3 + x^2 + 1$$

Ред по отношение на кривата:

0x4000000000000000000020108a2e0cc0d99f8a5ef

Стартова точка:

(x, y) = (0x2fe13c0537bbc11acaa07d793de4e6d5e5c94eee8,

0x289070fb05d38ff58321f2e800536d538ccdaa3d9)

Модулът представлява универсално цяло число (multiprecision integer) над GF_{2^p} ("bints" кое означава "big ints") в big-endian формат, като foo[0] е bint's най-важният байт и foo[NUMWORDS-1] е bint's най-маловажният байт, при що се предлага foo е bint (т.е. масив от байови NUMWORDS).

И покрай това bints са представени с NUMWORDS байта и се подрежда за да подържа препълване през точката на мултипликация и покрай това че NUMWORDS /2 байта са необходими за представяне на модула от 163-битови елементи на полето.

В модула, функции за прилагане на ниско бинарно ниво имат имена кои започват с b_, функции които се отнасят за точките имат имена започващи с p_, функции които се занимават с кривата имат имена които започват с c_ , функции кои се занимават с елементите на полето имат имена които започват с f_.

През целия код възлите се наричат Alice и Bob. Алиса е един възел а Боб е друг възел.

Таймери се използват за планиране на генерирането на ключовете и предаването на ключовете, защото кода я уголемява активността на CPU. В работата с този модул се потребни относително дълги изчисления, извършване на задачи предизвикани от други асинхронни събития (на пример обработка на прекъсвания) които са склони да се подредят в опашката на TinyOS.

3.4.1 Симулация и анализ при прилагането на EccM

Преди започване на анализата ще споменем че ключа тук е с дължина от 163-бита, Относно параметрите на EccM като елиптичната крива, точките и т.н. са с 163 бита.

При симулация на EccM за два възела с помощ на TOSSIM се получава следния изход

```
$ DBG=am ./build/pc/main.exe 2
SIM: Random seed is 98382
```

```
1: Started up.
0: Started up.
1: Generating private key...
1: Generated private key.
1: privKeyA.s:
03 88 1a 2b 40 9f 21 5d a5 55 96 54 2d a6 b6 b0 b3 0f 73 87 f2
1: Generating public key...
1: Generated public key in 0 usec.
1: My public key:
x:
00 2f 41 21 96 1b 56 1f e3 b0 0e 43 a1 d7 9b 40 84 2d f5 0f 68
y:
02 39 68 b6 55 60 ca 69 da d9 38 24 a8 48 3d c0 54 ef 20 d4 62
0: Generating private key...
0: Generated private key.
0: privKeyA.s:
02 c2 03 8d 94 af d0 2e d2 2a c8 2a 9a a4 4b b6 d2 aa f2 0f dd
0: Generating public key...
0: Generated public key in 0 usec.
0: My public key:
x:
01 2b d6 ef ba 4f c9 94 20 dd 4e a2 42 a7 83 f2 89 ba 26 6a 2c
y:
00 e0 65 77 ca 58 e4 1e e4 75 c2 29 d0 1f bc 3a fe 4f 25 76 de
1: Sending x coordinate of public key...
00 2f 41 21 96 1b 56 1f e3 b0 0e 43 a1 d7 9b 40 84 2d f5 0f 68
1: Sending message: ffff, 83
        ff ff 83 7d 16 01 00 2f 41 21 96 1b 56 1f e3 b0 0e 43 a1 d7 9b
40 84 2d f5 0f 68 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0: AM_address = ffff, 83; counter:1
0: Received message:
        ff ff 83 7d 16 01 00 2f 41 21 96 1b 56 1f e3 b0 0e 43 a1 d7 9b
40 84 2d f5 0f 68 00 00 00 00 00 00 00 01 00 00 00 00 00 a5 cf 00 00
0: AM_type = 131
0: Received x coordinate of Bob's public key.
0: x coordinate received from Bob:
00 2f 41 21 96 1b 56 1f e3 b0 0e 43 a1 d7 9b 40 84 2d f5 0f 68
1: Sent x coordinate of public key.
1: Sending y coordinate of public key...
1: Sending message: ffff, 83
        ff ff 83 7d 16 00 02 39 68 b6 55 60 ca 69 da d9 38 24 a8 48 3d
c0 54 ef 20 d4 62 00 00 00 00 00 00 00 80 28 00 00 01 00 3a ac 00 00
0: AM_address = ffff, 83; counter:2
0: Received message:
        ff ff 83 7d 16 00 02 39 68 b6 55 60 ca 69 da d9 38 24 a8 48 3d
...
```

compiled Ecc to build/mica2/main.exe	
35428	**bytes in ROM**
1119	**bytes in RAM**

Таблица 6. ROM и RAM изискани от приложението за Mica2

Когато приложението се компилира в машинния език на Atmel AVR се получават 16228 редови машинен код.

Симулацията за консумация на енергия при топология RANDOM за период от 15 секунди (изчислена е средната стойност за възлите). Резултатите са представени в таблица 7 (енергията е в mJ).

Брой възли	4	9	16	25	36
Cpu total	124.941091	141.424757	161.686404	168.131451	180.559672
Radio total	190.110126	175.1506445	183.220009	186.70570524	207.2214743
Total energy	315.051217	316.5754015	344.906413	373.41141048	387.78114

Таблица 7.

Симулацията за консумация на енергия при топология RANDOM за период от 15 секунди, когато се възползват функциите на CC1000 за управление на захранването (изчислена е средната стойност на възлите). Резултатите са представени в Таблица 8 (енергията е в mJ).

Брой възли	4	9	16	25	36
Cpu total	144.7986607	153.37800	167.0382331	162.9878362	172.6190986
Radio total	151.752094	187.483841	215.9539929	194.8119082	210.8293001
Total energy	296.5507554	340.861841	382.992226	357.799744	383.4483987

Таблица 8.

Симулацията за консумация на енергия при топология GRID за период от 15 секунди (изчислена е средната стойност на възлите). Резултатите са представени в Таблица 9 (mJ).

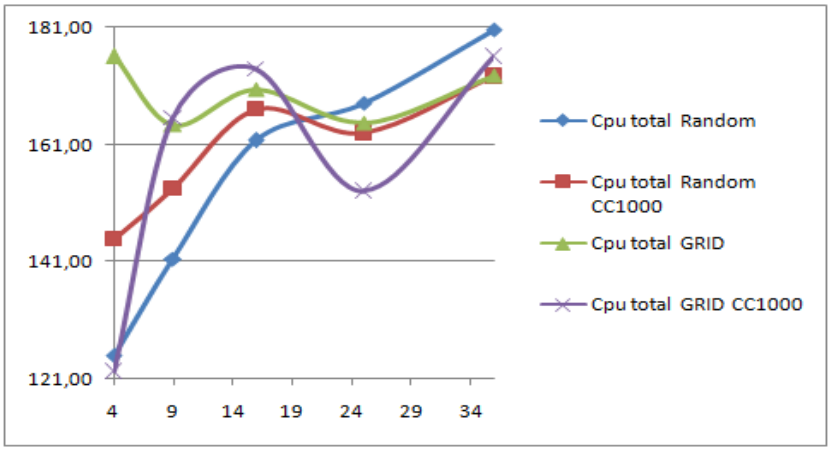
Брой възли	4	9	16	25	36
Cpu total	176.059134	164.346074	170,415981	164,7439727	172,8411306
Radio total	165.470957	199.2188286	204,3598061	202,6701889	196,5309241
Total energy	341.530091	337.102849	374,7757871	367,4141616	369,3720547

Таблица 9.

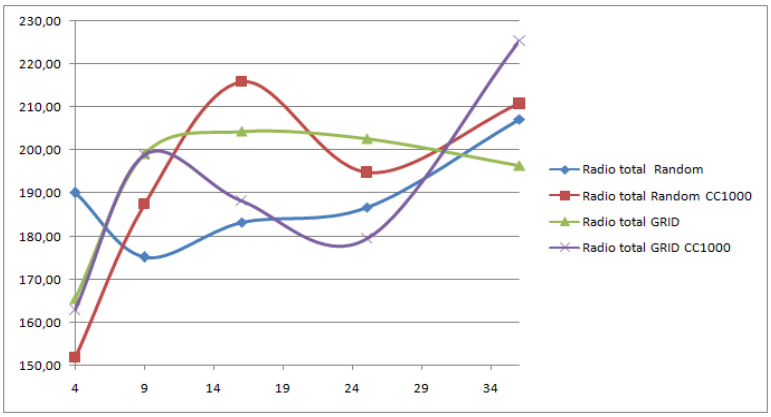
Симулацията за консумация на енергия при топология GRID за период от 15 секунди, когато се възползват функциите на CC1000 за управление на захранването (изчислена е средната стойност на възлите). Резултатите са представени в Таблица 10(mJ).

Брой възли	4	9	16	25	36
Cpu total	122.234584	165.428894	173.844406	153.06425756	176.0640
Radio total	162.967202	198.919796	188.178761	179.51128036	225.42413
Total energy	285.201786	364.348690	362.023167	332.57553792	401.48813

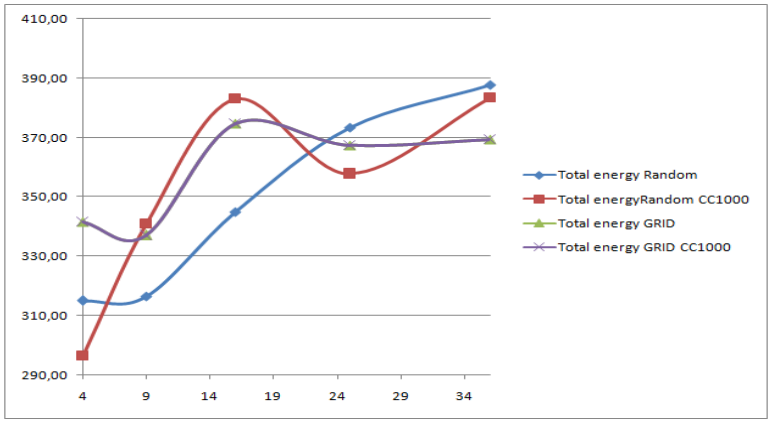
Таблица 10.



Фигура 29. Консумацията на енергия от страна на CPU



Фигура 30. Консумацията на енергия от страна на radio канала



Фигура 31.Средната стойност на съвкупната консумацията на енергия

От горните графици можем да забележим че консумацията на енергия варира за различен брой на възли в сензорната мрежа.

Частните ключове за Alice и Bob's са случайно избрани. Ако се предпоставя че съществува атака с повече ключове над едно съобщение, ние можем да го променим кода и да го фиксираме ключа.

In $TOSROOT/apps/EccM-2.0/EccM.nc, in void task generate_privKeyA():

```
--- for (i = NUMWORDS/2; i < NUMWORDS; i++)
---         {
---          privKeyA.s[i] = (word_t) call Random.rand();
---         }
+++ for (i = NUMWORDS/2; i < NUMWORDS; i++)
+++       {
+++       if (i%4==1)
+++       {
+++       privKeyA.s[i] = (word_t) 0x01;
+++       }
+++ else
+++       {
+++       privKeyA.s[i] = (word_t) 0x00;
+++       }
+++ }
```

При компилирането не се забележат особени разлики по заетата памет и т.н. Тук ще да дадем акцент на това дали фиксиране на ключа влияе на промяна на консумацията на енергия.

Тук симулацията ще се възползва с функциите на CC1000 за управление на захранването, и покрай това че не дава някои особени подобрения. Причината за използването на CC1000 е затова че е по-нова версия на радия стека.

Симулацията за консумация на енергия при топология RANDOM за период от 15 секунди, когато се възползват функциите на CC1000 за управление на захранването (изчислена е средната стойност). Резултатите са представени в Таблица 11. (енергията е в mJ).

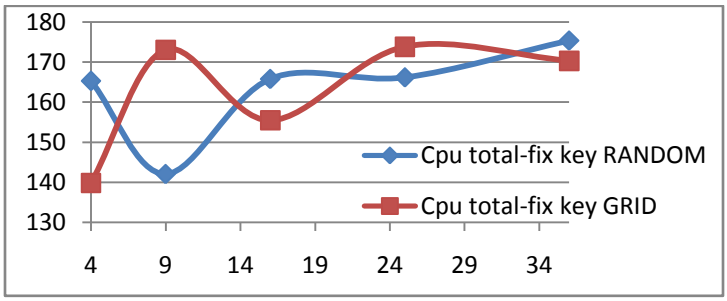
Брой възли	4	9	16	25	36
Cpu total	165.272437	142.08397	165.8037958	166.1737083	175.374120
Radio total	221.497821	207.6747582	199.7196878	196.3433886	186.045238
Total energy	386.770258	349.758728	365.5234836	362.5170969	361.419358

Таблица 11.

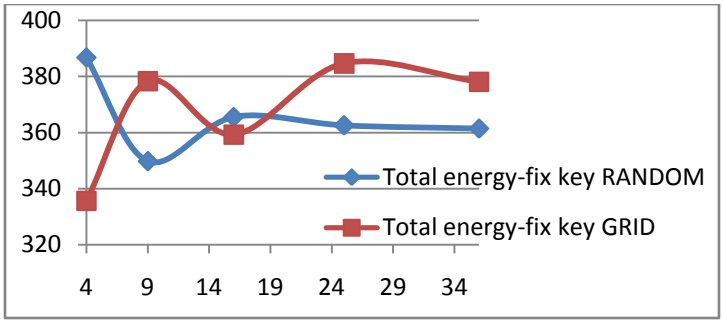
Симулацията за консумация на енергия при топология GRID за период от 15 секунди, когато се възползват функциите на CC1000 за управление на захранването (изчислена е средната стойност). Резултатите са представени в Таблица 12 (енергията е в mJ).

Брой възли	4	9	16	25	36
Cpu total	139.831697	173.073801	155.48262	173.824947	170.310630
Radio total	195.838567	205.245661	203.80611	210.825093	207.785208
Total energy	335.670264	378.319462	359.28873	384.650040	378.0958380

Таблица 12.

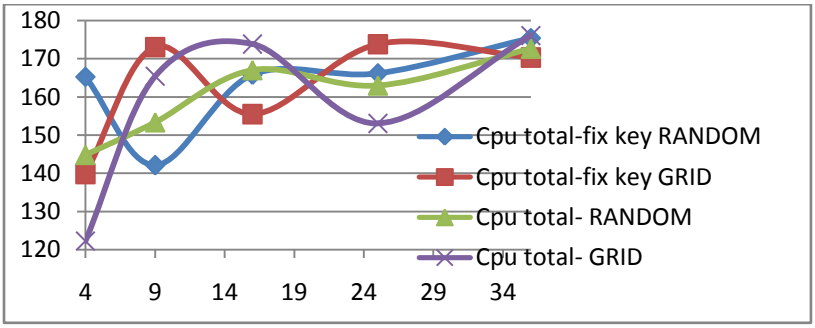


Фигура 32. Консумацията на енергия от страна на CPU когато ключа е фиксиран

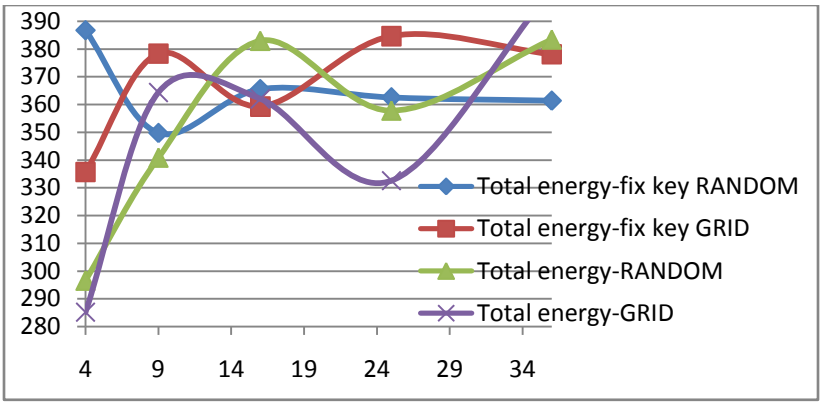


Фигура 33. Средната стойност на консумацията на енергия когато ключа е фиксиран

Анализ когато ключа се взема случайно и когато е фиксиран.



Фигура 34. Консумацията на енергия от страна на CPU



Фигура 35. средната стойност на консумацията на енергия когато ключа е фиксиран

От извършените симулации и съответно изчисления можем да видим че фиксирането на ключа не върши особено влияние при консумирането на енергия. Също тука се явяват вариации на енергията, няма константно увеличаване на консумацията на енергия при увеличаване на броя на възлите.

3.5 Анализ на TinyECC

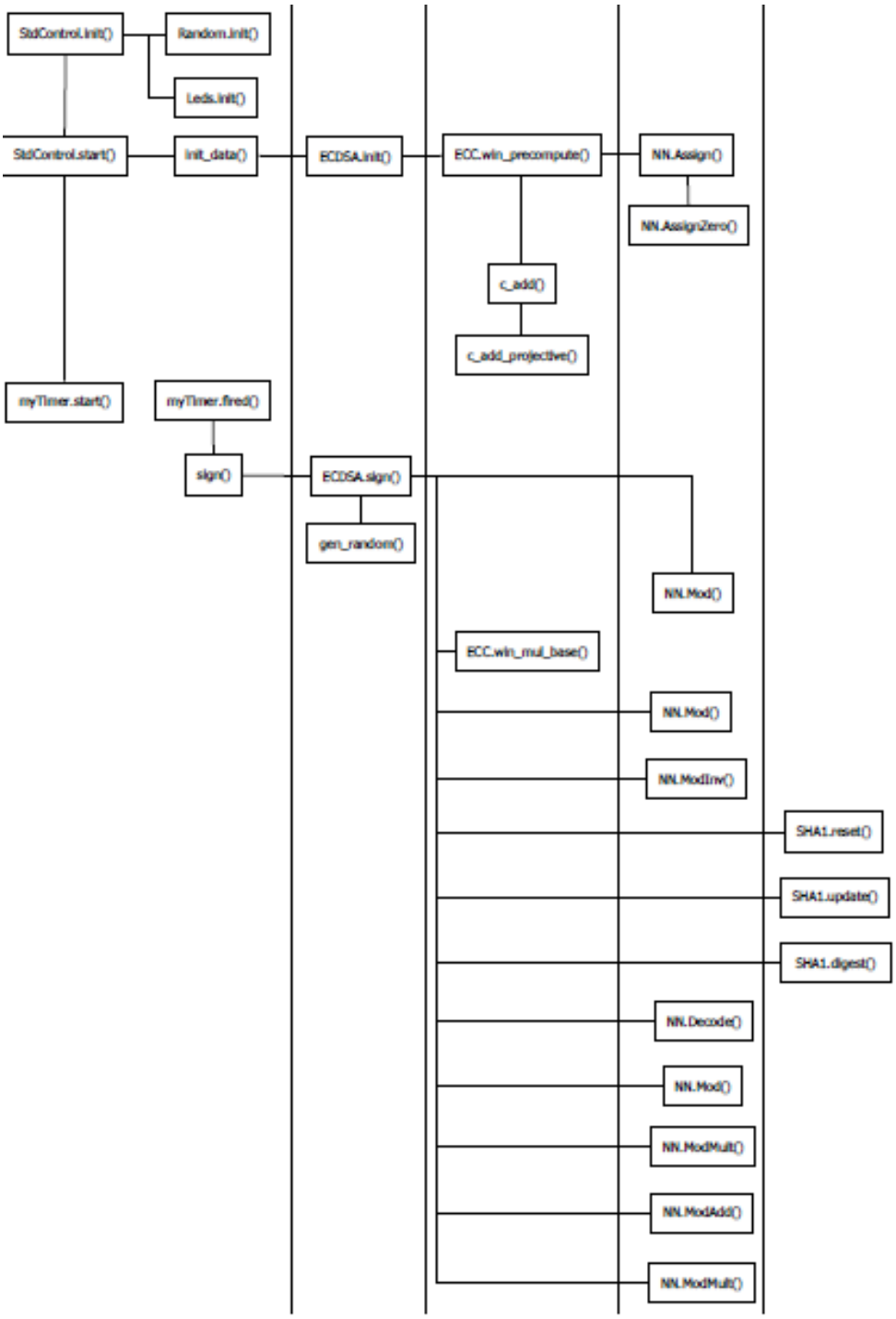
TinyECC [31] е софтуерен пакет кой осигуря ECC-базирана PKC операция коя има приспособима конфигурация и е интегрирана в апликациите за сензорните мрежи. Тя осигуря схема за дигитален подпис (ECDSA), протокол за разменна на ключове (ECDH), и схема за шифроване с публични ключове (ECIES). Осигуря определен брой превключватели за оптимизация, който предвиждат специфични мерки за оптимизация които може да ги използваме или да не ги използваме в зависимост от нуждите.
Някои от техниките за оптимизация се оптимизация с модиларна редукция с използване на pseudo-Mersenne прости числа, методът с плъзгащия прозорец за модулирано степенуване, използването на Jacobian координати за основните операции, inline assembly и hybrid умножение с което се постига изчислителна ефикасност. В действителност тези приложение е ефективно, имплементирани са повече трикове които могат да са много полезни, когато дойде време да се предлагат мерки за защита от различни атаки.

Моменталната версия на TinyECC подържа MICAz, TelosB/Tmote Sky и Imote2 motes. Тя подържа SECG препоръчани 128-bit, 160-bit и 192-bit параметри в доменни на елиптичните криви.

3.5.1 Анализа на кода на TinyECC

Частта от приложението TinyECC което е разгледано се състои от файловете NNM.nc, ECDSAM.nc. Elliptic Curve Digital Signature алгоритъма е описан така че използва техника за степенуване подобно на другите приложения.
Digital Signature алгоритъма с верификация генерално успешно защитава от канални атаки. Програмния поток е представен на Фигура 36.
Функцията за инициялизация, инициялизира случаен генериран брой. Когато се стартова приложението init_data(), изчиства всички съобщения, и поставя частен експонент

(exponent) и публична точка. Модула на ECDSA също така е инициялизиран преди извикване на таймера от 10 секунди, които ще ви позволи всички асинхронни задачи да ги изпълните. Когато таймера че измине, се извиква функцията sign(), където се генерира случайно съобщение, което е подписано, по което пакета се подготвя за евентуално изпращане.



Фигура 36: Логически поток на TinyECC за Alice

Дефиниране на интерфейсите:

1) NN.nc дефинира интерфейса NN, кой осигурява голям брой на естествени операции. NNM.nc го изпълнява тази интерфейс.

2) ECC.nc дефинира интерфейса ECC, кой ги осигуря основните операции на елиптичната крива и стартира операции на елиптичната крива за проектиране координатен система базиран на метод с плъзгащ прозорец. ECCM.nc го изпълнява тази интерфейс.

3) ECDSA.nc дефинира интерфейса ECDSA, кой осигуря ECDSA генериране и верифициране на дигитален подпис. ECDSAM.nc го изпълнява тази интерфейс.

4) ECIES.nc дефинира интерфейса ECIES, кой осигуря ECIES шифриране и де шифриране. ECIESM.nc го изпълнява тази интерфейс.

5) ECDH.nc дефинира интерфейса ECDH, кой осигуря ECDH разменна на ключове. ECDHM.nc го изпълнява тази интерфейс.

6) SHA1.nc дефинира интерфейса SHA1, кой осигуря SHA-1 функция. SHA1M.nc го изпълнява тази интерфейс.

7) CurveParam.nc дефинира интерфейса CurveParam, кой осигуря една функция за добиване на параметрите на елиптичната крива и друга функция за оптимизирано умножение с omega. secp128*.nc, secp160*.nc, secp192*.nc изпълняват тази интерфейс с което осигуряват параметри за SECG дефинирана елиптична крива. Сега е нужно да се определи името на кривата в makefile, ги избираме параметрите на елиптичните криви.

Inline Assembly код
Една част от файла NNM.nc е написан с inline assembly код за да се ускорят операциите с числата. Този inline assembly код е написан на AVR множеството инструкции и следователно са оптимизирани за XScale (MicaZ and Imote2). С коментиране на реда #define INLINE_ASM в файла NN.h се дава възможност за провеждане на симулация (за ползване на TinyECC на други 8-битови платформи). Функцията за степенуване е дефинирана в модула NNM.nc.
Като що е изнесено по горе параметрите ни представляват, елиптичната крива която ще се използва, точките на кривата, множество от точките, ключа и т.н.

3.5.2 Симулация и анализ при прилагането на TinyECC

3.5.2.1 Симулация и анализ при прилагането на ECDH

При симулация на ECDH за два възела с помощ на TOSSIM се получава следния изход

```
$ java show_ecdh micaz
```

```
start
[ time of EDH.init() is 2.4088542E-4 sec ]
Private key1:
e5af144de3325ee833da2ad22ed0af948d03c226
Public key1:
x: 1eac3bfa9da14d5b31e4278649139d8da3cc4e24
y: 92929fce45b5ba2fb416d5fcd673ddd71d066dfe
[ time of public key 1 generation is 0.040859375 sec ]
Private key2:
e52f546d737a7afaba9815f92d2d3edfd0155faf
Public key2:
x: 77d01db5e382508103f5ae0f6c62578397b2f3da
y: 41ad23591631dce11c11912cafc55aa0fe8a8476
[ time of public key 2 generation is 0.040859375 sec ]
established key1: 11b7b216c23a41f77a852d4c4b28040f7bc3c544
[ time of ECDH.key_agree() for 1 is 0.040859375 sec ]
established key2: 11b7b216c23a41f77a852d4c4b28040f7bc3c544
[ time of ECDH.key_agree() for 2 is 0.040859375 sec ]
Average timing result for 1 rounds
ECDH.init(): 2.4088542E-4
PK1: 0.040859375
PK2: 0.040859375
key_agree1: 0.040859375
key_agree2: 0.040859375
[ time of EDH.init() is 0.040859375 sec ]
Private key1:
3758eb365ce9b59708416e2d43b1fe3f806d93c5
Public key1:
x: d153580f5f8d5a72f5b08d4c18022555c188010
y: ce4a6ce9a02dc344c635b9f577392e4b4703a378
[ time of public key 1 generation is 0.040859375 sec ]
...
```

ECDH.init():	Key establish
0.00367	0.040859375

Таблица 13. Времето за изпълнение

Преди започнем с анализата за консумация на енергия ще споменем че параметрите на ЕС са 128-бита.

compiled testECDH to build/mica2/main.exe	
21910	**bytes in ROM**
1017	**bytes in RAM**

Таблица 14. ROM и RAM изискани от приложението с параметри за ЕС от 128-бита за Mica2

Когато приложението се компилира в машинния език на Atmel AVR се получават 9598 редови машинен код.

Симулацията за консумация на енергия при топология RANDOM за период от 15 секунди (изчислена е средната стойност). Резултатите са представени в Таблица 15 (mJ).

Брой възли	4	9	16	25	36
Cpu total	50.1572125	68.764630	100.1574555	112.32850064	113.7039075
Radio total	95.504825	87.151750	104.877636	89.435353454	98.2276666
Total energy	145.662037	155.91638	205.0350920	201.7638540	211.93157372

Таблица 15.

Преди започнем с анализата за консумация на енергия ще споменем че параметрите на ЕС са 160-бита.

compiled testECDH to build/mica2/main.exe	
22062	**bytes in ROM**
1077	**bytes in RAM**

Таблица 16. ROM и RAM изискани от приложението с параметри за ЕС от 160-бита за Mica2

Когато приложението се компилира в машинния език на Atmel AVR се получават 9642 редови машинен код.

Симулацията за консумация на енергия при топология RANDOM за период от 15 секунди (изчислена е средната стойност). Резултатите са представени в Таблица 17. (mJ).

Брой възли	4	9	16	25	36
Cpu total	62.3238485	86.9362892	106.084947	97.5055408	114.1586175
Radio total	78.6754517	94.684661	104.455775	94.90066148	92.1712088
Total energy	140.999300	181.620950	210.5407227	192.40620228	206.3298175

Таблица 17.

Симулацията за консумация на енергия при топология GRID за период от 15 секунди (изчислена е средната стойност). Резултатите са представени в Таблица 18 (mJ).

Брой възли	4	9	16	25	36
Cpu total	77.520925	101.655074	107.5624779	108.63463512	112.3279195
Radio total	62.81250525	90.268559	99.7459386	87.43793268	108.5692263
Total energy	140.33343025	191.923633	207.3084165	196.0725678	220.8971458

Таблица 18.

Преди започнем с анализата за консумация на енергия ще споменем че параметрите на ЕС са 192-бита.

compiled testECDH to build/mica2/main.exe	
22254	**bytes in ROM**
1137	**bytes in RAM**

Таблица 19. ROM и RAM изискани от приложението с параметри за ЕС от 192-бита за Mica2

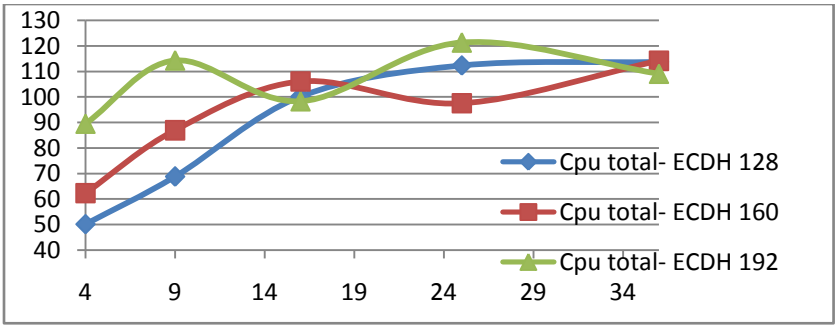
Когато приложението се компилира в машинния език на Atmel AVR се получават 9690 редови машинен код.

Симулацията за консумация на енергия при топология RANDOM за период от 15 секунди (изчислена е средната стойност). Резултатите са представени в Таблица 20 (mJ).

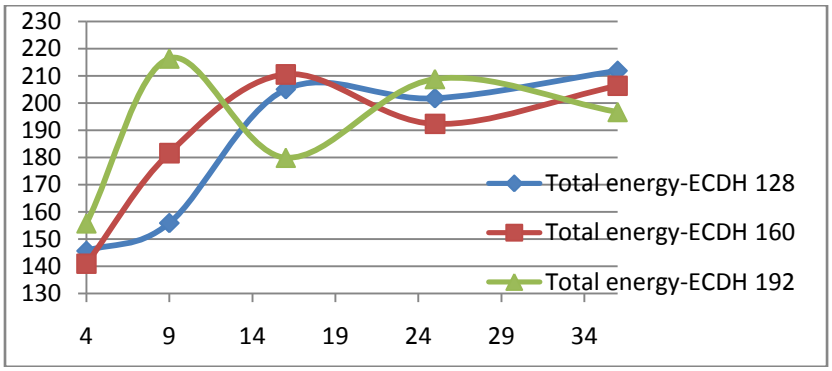
Брой възли	4	9	16	25	36
Cpu total	89.521294	114.2657	98.308898	121.287912	109.059409
Radio total	66.27305925	102.132115	81.6209153	87.54399796	87.6747934
Total energy	155.7943532	216.397815	179.9298133	208.83190996	196.7342024

Таблица 20.

Анализ при ECDH ще се върши за консумация на енергия при различните големини на параметрите, също така ще стане дума и за нужните ресурси които се изискат от приложението при различни големини на параметрите.

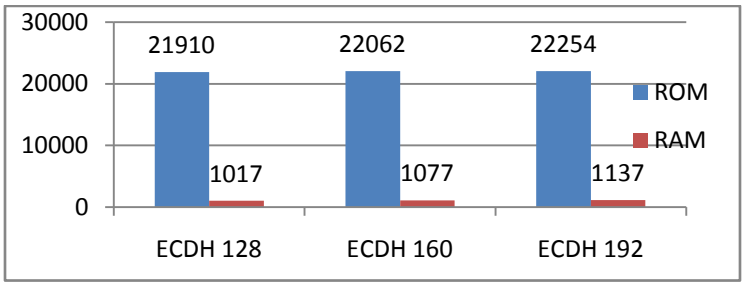


Фигура 37.Консумацията на енергия от страна на CPU



Фигура 38. средната стойност на съвкупната консумацията на енергия

От горе представените графи можем да видим че тук имаме привидно константно увеличаване на консумацията на енергия при увеличаване на броя на възли. Също така генерално можем да кажем че при прилагане на параметри с по малко битове разходите на енергия са по малки което е и логично.



Фигура 39. ROM и RAM изискани от приложението при различни параметри за EC.

Тук се вижда че при различни големина на параметрите за приложението заема различни ресурси. Можем да забележим че когато приложението използва параметри с повече битови изиска повече ресурси.

3.5.2.2 Симулация и анализ при прилагането на ECDSA дигитален потпис

При симулация на ECDSA за два възела с помощ на TOSSIM се получава следния изход

```
$ java show_ecdsa micaz
start
-------------------- round 0 ---------------------
Private key:
d: 6ef03f5c69f53758eb365ce9b5970843
[ time of ECC.init() is 0.040859375 sec ]
Public key:
x: 7cfc80dee917129b062563c8daba704d
y: d16a33f7282fa646d04df59d844f42f
[ time of public key generation is 0.040859375 sec ]
[ time of ECDSA.init() is 0.040859375 sec ]
content and signature
msg:
26c2038d94afd02ed22ada33e85e32e34d14afe54406828a9ab3ed5426c20a9ab
afa7a7a736d542fe5491cb6e24a13add93cf662
signature
r: 4b4750565aef7dd7f813eee616c9fe2d
s: 17e2b099c3ebd8109a3abcfa06b5cecb
[ time of signature generation is 0.040859375 sec ]
[ time of signature verification is 1584.152 sec ] (pass)
```

```
Average timing result
ECC.init(): 0.040859375
ECDSA.init(): 0.040859375
public key gen: 0.040859375
sign: 0.040859375
verify: 1584.152

-------------------- round 1 --------------------
Private key:
d: c1a19109455f6cf1b99d8b06c4a58f04
[ time of ECC.init() is 0.040859375 sec ]
Public key:
x: df2a61dbb929382ac0f198d7ed63a897
y: 19a07f922953c63830863f4f6f6e0f85
[ time of public key generation is 0.040859375 sec ]
[ time of ECDSA.init() is 0.040859375 sec ]
content and signature
msg:
aad328de32e34d14afe54911a1cc1fb0ee5223cd19bcff7067400e9bbdf97c7f7
06e5b38fe72634d19b1ec5622c30d99b1ec5f45
signature
r: e51852351c119d885ef61c083c913244
s: e052692232587ddf03a27ed4e511e9f2
[ time of signature generation is 0.040859375 sec ]
[ time of signature verification is 1584.152 sec ] (pass)
Average timing result
ECC.init(): 0.040859375
...
```

ECDSA.init():	sign:	verify:
0.040859375	0.040859375	1584.152

Таблица 21. Времето за изпълнение

Преди започнем с анализата за консумация на енергия ще споменем че параметрите на ЕС са 128-бита.

compiled testECDSA to build/mica2/main.exe	
23648	**bytes in ROM**
1022	**bytes in RAM**

Таблица 22. ROM и RAM изискани от приложението с параметри за ЕС от 128-бита за Mica2

Когато приложението се компилира в машинния език на Atmel AVR се получават 10473 редови машинен код.

Симулацията за консумация на енергия при топология RANDOM за период от 15 секунди (изчислена е средната стойност). Резултатите са представени в Таблица 23 (mJ).

Брой възли	4	9	16	25	36
Cpu total	127.387653	164.038994	161.6221412	155.568520	170.016688
Radio total	165.472878	198.601508	200.6124753	201.021701	209.110216
Total energy	292.860531	362.640502	362.2346165	356.590221	379.126904

Таблица 23.

Преди започнем с анализата за консумация на енергия ще споменем че параметрите на ЕС са 160-бита.

compiled testECDSA to build/mica2/main.exe	
23764	**bytes in ROM**
1070	**bytes in RAM**

Таблица 24. ROM и RAM изискани от приложението с параметри за ЕС от 160-бита за Mica2

Когато приложението се компилира в машинния език на Atmel AVR се получават 10531 редови машинен код.

Симулацията за консумация на енергия при топология RANDOM за период от 15 секунди (изчислена е средната стойност). Резултатите са представени в Таблица 25. (mJ).

Брой възли	4	9	16	25	36
Cpu total	133.7080975	158.7176521	163.92856	167.451212	174.83284
Radio total	157.872088	173.5971132	225.04748	201.984740	196.36631
Total energy	291.5801855	332.3147753	388.97604	369.435952	371.19915

Таблица 25.

Симулацията за консумация на енергия при топология GRID за период от 15 секунди (изчислена е средната стойност). Резултатите са представени в Таблица 26 (mJ).

Брой възли	4	9	16	25	36
Cpu total	122.997196	155.607147	154.064327	157.234528	172.346701
Radio total	164.4796695	193.612121	183.537657	191.786735	198.649925
Total energy	287.4768655	349.219268	337.601984	349.021263	370.996626

Таблица 26.

Преди започнем с анализата за консумация на енергия ще споменем че параметрите на ЕС са 192-бита.

compiled testECDSA to build/mica2/main.exe	
23810	**bytes in ROM**
1118	**bytes in RAM**

Таблица 27. ROM и RAM изискани от приложението с параметри за ЕС от 128-бита за Mica2

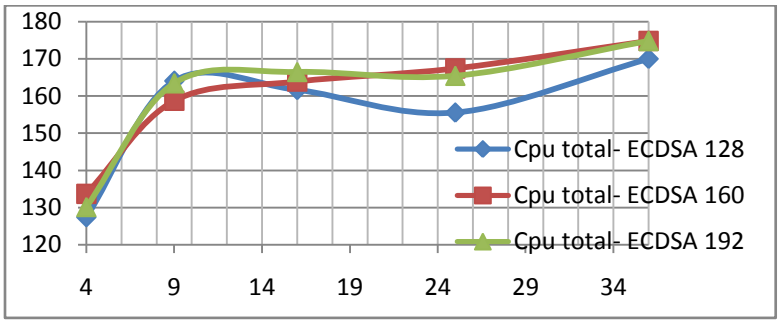
Когато приложението се компилира в машинния език на Atmel AVR се получават 10554 редови машинен код.

Симулацията за консумация на енергия при топология RANDOM за период от 15 секунди (изчислена е средната стойност). Резултатите са представени в Таблица 28 (mJ).

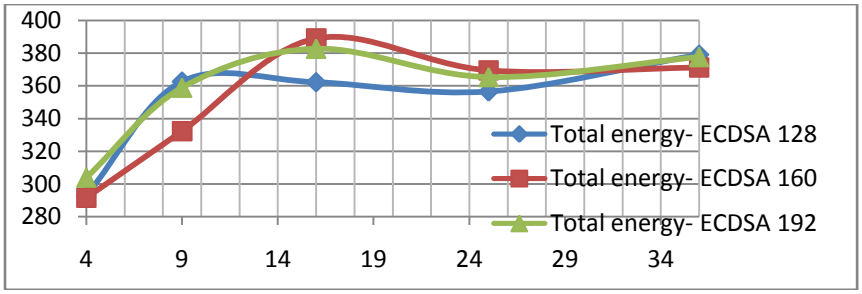
Брой възли	4	9	16	25	36
Cpu total	130.170740	163.427216	166.471818	165.470830	174.7802278
Radio total	173.752828	195.485354	216.220150	199.889158	202.8406199
Total energy	303.923568	358.912570	382.691968	365.359988	377.6208477

Таблица 28.

Анализ при ECDSA ще се върши за консумация на енергия при различни големини на параметрите, също така ще стане дума и за нужните ресурси които се изискат от приложението при различни големини на параметрите.

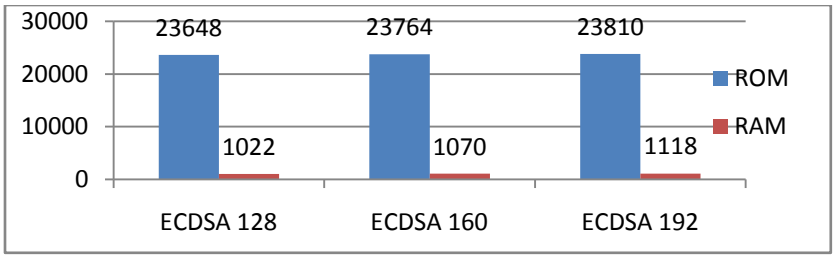


Фигура 40. Консумацията на енергия от страна на CPU



Фигура 41. Средната стойност на съвкупната консумацията на енергия

От горе представените графи можем да видим че тук имаме привидно константно увеличаване на консумацията на енергия при увеличаване на броя на възли. Също така генерално можем да кажем че когато се прилагат параметри с по-малко битове разходите на енергия се по малки което е и логично.



Фигура 42. ROM и RAM изискани от приложението при различни параметри за ЕС.

Тук се вижда че за различни големини на параметрите приложението заема различни ресурси. Можем да забележим че когато приложението използва параметри с повече битови изиска повече ресурси.

3.5.2.3 Симулация и анализ при прилагането на схема за шифроване с публични ключове (ECIES)

При симулация на ECIES за два възела с помощ на TOSSIM се получава следния изход

```
$ java show_ecies micaz
start
26c2038d94afd02ed22ada33e85e32e34d14afe54406828a9ab3ed5426c20a9abafa7a
7a736d542f
[ time of ECIES.init() is 4658.2236 sec ]
Private key:
d: e9b597084362f63cd9ad134ae2b61c49
Public key:
x: 8be65fbf4facc38771651404400679f5
y: bc012fdd7f35eced96bfc2fa2067d270
[ time of public key generation is 0.040859375 sec ]
0324bf803546ddbe20edd058baed4a987815f4930f6e45690c7665fd0f5e92c03f2604
851cd7793edf8e4c2e96f0538ece3c1a70b81d6604f3b051eb7f557e60a2e4dd557f3b
3af973ebfa8c8b
[ time of ECIES.encrypt() is 1584.152 sec ]
26c2038d94afd02ed22ada33e85e32e34d14afe54406828a9ab3ed5426c20a9abafa7a
7a736d542f
[ time of ECIES.decrypt() is 0.040859375 sec ]
Average timing result for 1 rounds
ECIES.init(): 4658.2236
public key gen: 0.040859375
encrypt: 1584.152
decrypt: 0.040859375
35e446028a93a8d720ce12aad328de32e34d14afe54911a1cc1fb0ee5223cd19bcff70
[ time of ECIES.init() is 2896.2285 sec ]
Private key:
d: c32256ecb1194d6372fe385b6e707f7c
Public key:
x: 1739398599bd83dfa2c135972df015cd
y: c50ad524ebb1f0f7d440ccd868a28ed2
```

...

ECIES.init():	encrypt:	decrypt:
3777.226	1584.152	0.040859375

Таблица 21. Времето за изпълнение

Преди започнем с анализата за консумация на енергия ще споменем че параметрите на ЕС са 128-бита.

compiled testECIES to build/mica2/main.exe	
25344	**bytes in ROM**
1113	**bytes in RAM**

Таблица 30. ROM и RAM изискани от приложението с параметри за ЕС от 128-бита за Mica2

Когато приложението се компилира в машинния език на Atmel AVR се получават 11249 редови машинен код.

Симулацията за консумация на енергия при топология RANDOM за период от 15 секунди (изчислена е средната стойност). Резултатите са представени в Таблица 31. (mJ).

Брой възли	4	9	16	25	36
Cpu total	70.04049975	62.7485373	94.19081462	101.631863	105.2247061
Radio total	87.039121	57.0020957	96.78557775	85.2534279	86.25014972
Total energy	153.4233142	119.750633	190.97639237	186.8852909	191.4748558

Таблица 31.

Преди започнем с анализата за консумация на енергия ще споменем че параметрите на ЕС са 160-бита.

compiled testECIES to build/mica2/main.exe	
25490	**bytes in ROM**
1161	**bytes in RAM**

Таблица 32. ROM и RAM изискани от приложението с параметри за ЕС от 160-бита за Mica2

Когато приложението се компилира в машинния език на Atmel AVR се получават 11290 редови машинен код.

Симулацията за консумация на енергия при топология RANDOM за период от 15 секунди (изчислена е средната стойност). Резултатите са представени в Таблица 33 (mJ).

Брой възли	4	9	16	25	36
Cpu total	94.18137375	86.6444034	106.2243269	95.0907869	113.218379
Radio total	91.291311	76.8818813	98.90589518	76.14700524	111.526220

Total energy	185.472684	163.526284	205.1302220	171.2377921	224.7445993

Таблица 33.

Преди започнем с анализата за консумация на енергия ще споменем че параметрите на EC са 192-бита.

compiled testECIES to build/mica2/main.exe	
25622	**bytes in ROM**
1209	**bytes in RAM**

Таблица 34. ROM и RAM изискани от приложението с параметри за EC от 192-бита за Mica2

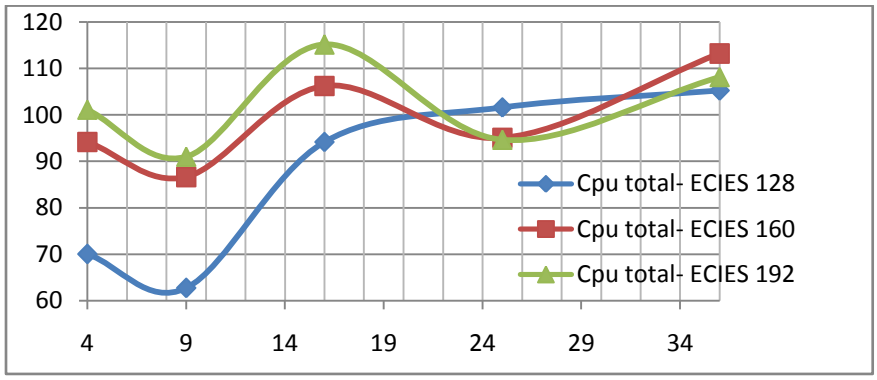
Когато приложението се компилира в машинния език на Atmel AVR се получават 11310 редови машинен код.

Симулацията за консумация на енергия при топология RANDOM за период от 15 секунди (изчислена е средната стойност). Резултатите са представени в Таблица 35 (mJ).

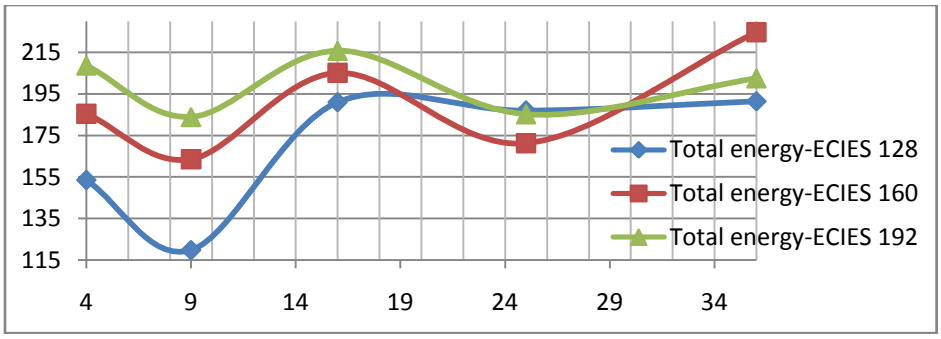
Брой възли	4	9	16	25	36
Cpu total	101.1258435	91.0041265	115.156102	94.6707226	108.17876344
Radio total	107.6333502	92.9064061	100.684059	90.553977	94.364872444
Total energy	208.75919375	183.910532	215.840161	185.2246996	202.54363588

Таблица 35.

Анализ при ECIES ще се върши за консумация на енергия при различни големини на параметрите, също така ще стане дума и за нужните ресурси които изиска приложението.

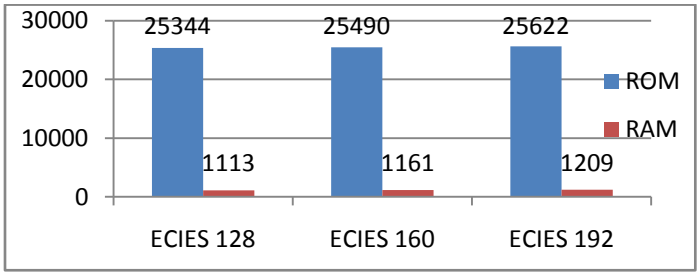


Фигура 43. Консумацията на енергия от страна на CPU



Фигура 44. Средната стойност на съвкупната консумацията на енергия

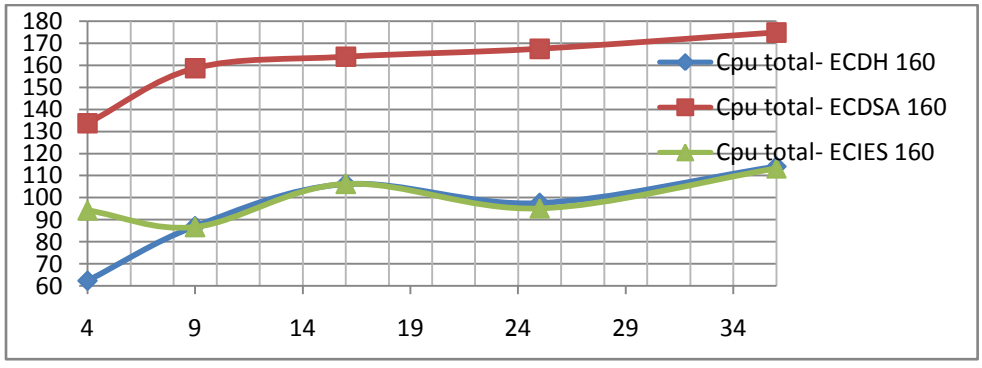
От горе представените графи можем да видим че тук имаме привидно константно увеличаване на консумацията на енергия при увеличаване на броя на възли. Също така генерално можем да кажем че когато се прилагат параметри с по-малко битове разходите на енергия се по малки което е и логично.



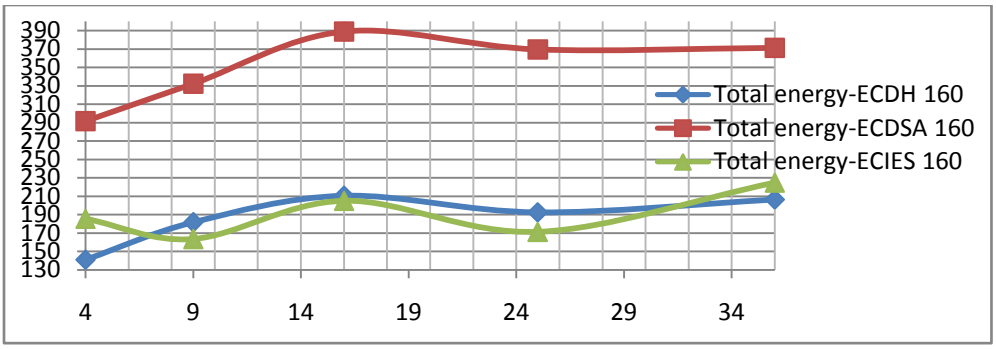
Фигура 45. ROM и RAM изискани от приложението при различни параметри за EC.

Тук се вижда че за различни параметри приложението заема различни ресурси. Можем да забележим че колкото са по-големи параметрите приложението изиска повече ресурси.

Анализата на ECIES, ECDSA, ECDH, техните нужни ресурси за 160 битови параметри и изисканата за консумация на енергия.



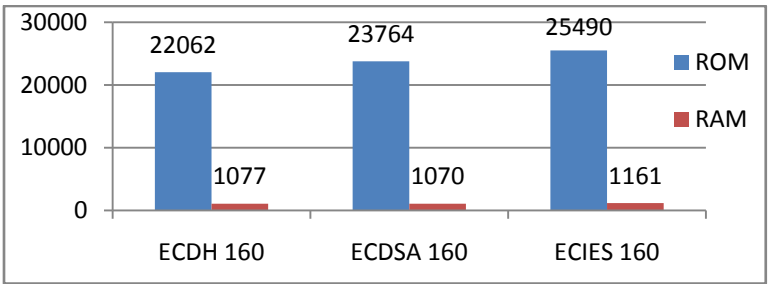
Фигура 46. Консумацията на енергия от страна на CPU, при ECIES, ECDSA и ECDH с 160 битови параметри



Фигура 47. Средната стойност на съвкупната консумацията на енергия, при ECIES, ECDSA и ECDH с 160 битови параметри.

От тези графи се вижда че разходите за енергия при ECIES и ECDH са почти същи до като ECIES изиска повече енергия, защото върши повече операции.

На следната фигура се дадени ресурсите които се изискат от ECIES, ECDSA и ECDH където са 160 битови параметри.



Фигура 48. ROM и RAM изискани от ECIES, ECDSA и ECDH (160 битови параметри).

Тук се вижда че за ECIES, ECDSA и ECDH за съща големина на параметри заемат различни ресурси. Можем да забележим че тук че ECDSA и въпреки това че изиска най-много енергия, той не заема най-много ресурси.

3.6 Оптимизация с TinyECC: Конфигурационна библиотека за ECC в безжичните сензорни мрежи [32]

Ще дадем кратък обзор на техниките за оптимизация, относно редукция на кода, нуждите за ROM и RAM, както влияе намаляване на времето за изпълнение на функциите по което логично идва и намаляване на консумирането на енергия.

Оптимизация за операции с големи цели числа

Barrett Reduction [34]: най-едноставен начин за извършване на модулни редукции на големи цели числа е да се използва деление [33]. Добър страничен ефект от такъв метод е това че го използва кода за деление така че се добива компактна (намалена) големина на кода. Barrett Reduction представлява алтернативен метод за модулна редукция (modular reduction) [34]. С този метод редукцията по модул на произволно цяло число се преминава в две умножения и няколко съкращения на модулите цели числа с форма 2^n. Когато се

използва за съкращение на едно число, Barrett reduction е по-бавно от основния алгоритъм с деление. Но, когато се ползва за съкращение на различни модулни числа и при което същото число се ползва повече пати, с повторно използване на същата стойност, Barrett reduction може да постигне по-голяма скорост от модулната редукция която се добива с деление.

HybridMultiplication и Hybrid Squaring [35]: Стандартните алгоритми за умножение на големи цели числа [33] ги слагат операндите и резултатните стойностите в низи. Кога такъв алгоритми се изпълняват в езиците на високо ниво като що е nesC, компилатора не може ефикасно да ги ползва регистрите на микропроцесора, при двоичния код е нужно повече пати да ги зарежда операндите от паметта в регистрите [35]. Gura et al. [35] предложила хибриден алгоритъм с умножение който е планиран за асемблерски код. Такъв алгоритъм може да го максимализира използваната на регистрите и да го намали броя на операциите с паметта.

Оптимизация за ECC операции

Projective Coordinate Systems [36]: (Проекционни координатни системи): Елиптичната крива се състои от безкрайна точка 0 и множество от точки в съответните координати (x,y), за $x,y \in Fp$ кои го задоволяват дефинирането на еднаквост (уравнение). Допълнително, една точка от елиптичната крива може да се представи на проекционна координатна система в форма на (x, y, z).

Основни и необходими операции в ECC се добавяне на точка и дублиране на точка. Тези операции в сроден координатна система изиска модулни обратими операции, които са много по трудни за изпълнение от другите операции като що са модулно умножение. С използване на проекционен координатен систем [36], модулната инверсия мога да се заменя с няколко модулни умножения. Како резултат на това, времето потребно за прибавяне и удвояване на точка при проекционния координатен систем може да биде много малко за разлика от сродна (основна) координатна система.

Sliding Window for Scalar Multiplications [36]: Плъзгащ прозорец за скаларни умножения: Скаларното умножение е основна операция коя се ползва при всички ECC схеми.

Тя е в форма $k*P$, къде k е цяло число а P е точка на елиптичната крива. При най-едноставния метод за изчисление на $k*P$, k се търси от най-важния бит до най-малко важния бит. Когато всеки бит е вече пребаран, нужно е алгоритъма да го изчисли удвояването на точката. Ако бита които се търси е "1", нужно е алгоритъмът да го изчислява удвояването на точката.

Метода с плъзгащ прозорец мога да го ускори скаларното умножение с търсене на w в също време. Всеки път когато се търси w бита, алгоритъмът требва да удвои w точки. При изчисление $2P$, $3P$, ..., и $(2w-1)P$, метода требва да прибави една точка за всеки w бита, и поради това се изиска повече време за изчисление.

Лесно може да се забележи дека тези метод с плъзгащ прозорец го увеличава използването на ROM (с прибавяне на допълнителен код) и RAM (за подредба на пред – изчислителните точки).

Shamir's Trick [36]: Трикът на Shamir. Оптимизацията се използва само за верификация на подпис ECDSA. Верификацията на подпис ECDSA изиска изчисление на формата $aP + bQ$, къде a, b се цели числа, а P, Q са две точки от елиптичната крива. Изпълнението

изиска два скаларни умножения и едно прибавяне на точка. Тези метод, позволява изчисление с разходи почти същи като изчисление на едно скаларно умножение. С предишното изчисление на *P+Q*, може да се търсят битовите a и b от най-важния до най-малко важния бит. Трикът на Shamir го увеличава използването на ROM (с прибавяне на допълнителен код) и RAM (за подредба на пред – изчислителните точки).

Curve Specific Optimization [19]: (Специфична оптимизация на крива [35]:) повечето елиптични криви специфицирани от NIST [36] и SECG [37] използват псевдо-Mersenne примитиви. псевдо-Mersenne примитивите се от форма $p = 2^n - c$, където $c \ll 2^n$. Тази метод може да се изведе с помощ на няколко модулни умножения и събиране без използване на операции за деление. Като резултат на това, времето потребно за модулна редукция може значително да се намали. На този начин използването на елиптични криви над псевдо-Mersenne примитиви може значително да ги подобри изпълнението.

3.6.1 Анализ на приложението при използване на TinyECC: Конфигурационна библиотека

Тук ще представим анализ и оценка на TinyECC, с помощ на конфигурационна библиотека за операции ECC. Уникална характеристика на TinyECC е неговото конфигуриране. Той предвижда редица мерки за оптимизация, с които можем да го конфигурираме TinyECC в зависимост от нуждите. Различни комбинации на оптимизации дават различни времето за изпълнение и изискат различно потребление на ресурсите, с което имаме голяма гъвкавост при интегрирането. Ще споменем че различните оптимизации различно се отнасят за различни платформи, също така тези оптимизации са по ефективни при реални условия. Тук ще направим няколко експерименти, за да направи оценка и анализ на изпълнението. Експериментите ще се отнасят на консумиране на енергия, заемането на ROM и RAM, от mica2.

Първия експеримент като го конфигурираме TinyECC да ги използва всичките оптимизации при което ще видим колко заема RAM и ROM за различни дължина на параметрите 128, 160 и 192 бита.

ECDH – mica2 - 128 byte key		**ECDH – mica2 - 160 byte key**		**ECDH – mica2 - 192 byte key**	
23392	**bytes in ROM**	**23566**	**bytes in ROM**	**23626**	**bytes in ROM**
2785	**bytes in RAM**	**2701**	**bytes in RAM**	**3069**	**bytes in RAM**

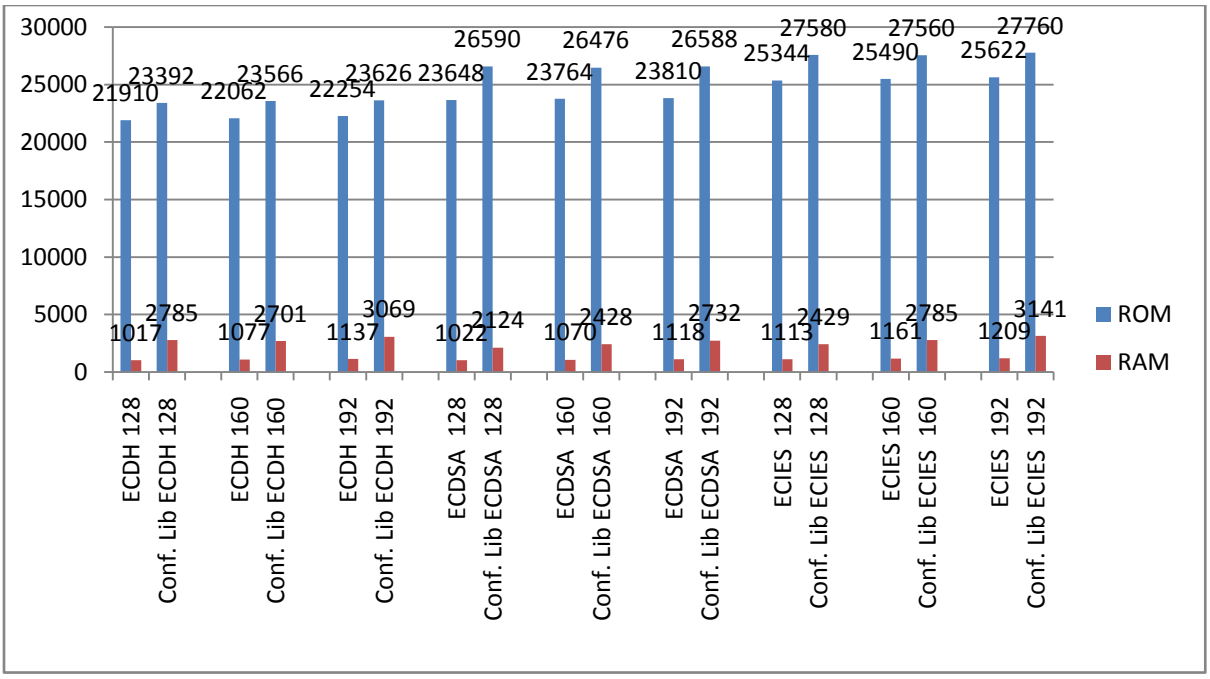
Таблица 36.

ECDSA – mica2 - 128 byte key		**ECDSA – mica2 - 160 byte key**		**ECDSA – mica2 - 192 byte key**	
26590	**bytes in ROM**	**26476**	**bytes in ROM**	**26588**	**bytes in ROM**
2124	**bytes in RAM**	**2428**	**bytes in RAM**	**2732**	**bytes in RAM**

Таблица 37.

ECIES– mica2 - 128 byte key		**ECIES – mica2 - 160 byte key**		**ECIES – mica2 - 192 byte key**	
27580	**bytes in ROM**	**27560**	**bytes in ROM**	**27760**	**bytes in ROM**
2429	**bytes in RAM**	**2785**	**bytes in RAM**	**3141**	**bytes in RAM**

Таблица 38.



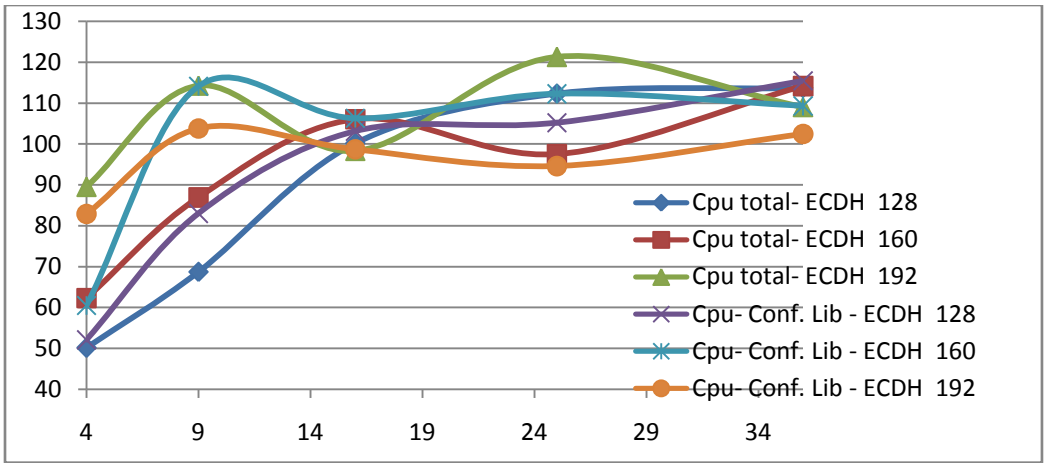
Фигура 49. ROM и RAM изискани от ECDH, ECDSA и ECIES когато не се използват оптимизациите от TinyECC: Конфигурационна библиотека и съответно при включени оптимизации (Conf. Lib означава включени оптимизации).

Тук можем да видим че с включване на оптимизациите изискванията за RAM и ROM нараства.

Втори експеримент като го конфигурираме TinyECC да ги използва всичките оптимизации при което да виждаме колко енергия консумира CPU (mica2) когато се симулира при топология RANDOM за период от 15 секунди (изчислена е средната стойност на възлите). Резултатите са представени в Таблица 39 (енергията е в mJ).

Брой възли	4	9	16	25	36
Cpu total - ECDH – mica2 - 128 byte key	52.12	83.01	103.12	105.21	115.45
Cpu total - ECDH – mica2 - 160 byte key	60.54	114.12	106.30	112.28	109.30
Cpu total - ECDH – mica2 - 192 byte key	82.89	103.77	98.70	94.60	102.48

Таблица 39.



Фигура 50. Консумацията на енергия от страна на CPU при ECDH за различни дължини на параметрите (Conf. Lib означава че се включени оптимизации, а числото е дължината на параметрите).

Тук можем да видим че с включване на оптимизациите не го значително увеличава изискването на енергия.

При използване на TinyECC: Конфигурационна библиотека трябва да внимаваме кои характеристики искаме да ги подобрим. Трябва да сме на ясно че при подобряване на някоя характеристика това се влияе на друга. Така че с включване на всички оптимизации от TinyECC: Конфигурационна библиотека нарастват нуждите за ресурси.

3.7 Симулация и анализ при прилагането на ECC-DH за разменна на ключове

ECC-DH представлява протокол за разменна на ключове, и е подобен на TinyECC - ECDH и EccM. Тука е включено оптимизация на кода [38, 39], промяна на операции, функции, алгоритми.

Често ускоряването на приложенията може да предизвика размера на кода да се увеличи. Това увеличаване на кода може да има неблагоприятен ефект, върху сложността и четливостта на дадена програма. Също така това може да бъде не приемливо за малки устройства, които имат ограничени ресурси.

Първа и най-важна част от оптимизирането на една програма е да намери къде да се оптимизира, коя част или модул от програмата се изпълнява бавно или използва огромна памет.

Трябва да се оптимизират тези части от програмата които се използват многократно, особено тези методи и функции които се извикват постоянно, при изпълнение на програмата.

Преди започнем с анализата за консумация на енергия ще споменем че параметрите на ЕС са 160-бита.

compiled testDH to build/mica2/main.exe	
22050	**bytes in ROM**
1077	**bytes in RAM**

Таблица 40. ROM и RAM изискани от приложението с параметри за ЕС от 160-бита за Mica2

Когато приложението се компилира в машинния език на Atmel AVR се получават 9635 редови машинен код.

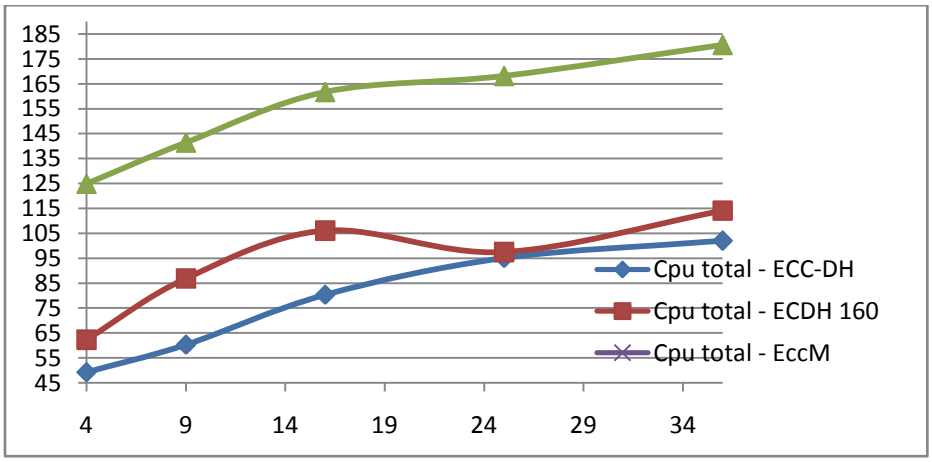
Симулацията за консумация на енергия при топология RANDOM за период от 15 секунди. Резултатите са представени в Таблица 41. (mJ).

Брой възли	4	9	16	25	36
Cpu total	49.2765555	60.360884	80.39449975	95.09078696	101.994509
Radio total	61.48450325	57.529935	84.72601987	80.39173724	98.2756495
Total energy	110.76105875	117.890819	165.1205196	175.4825242	200.270158

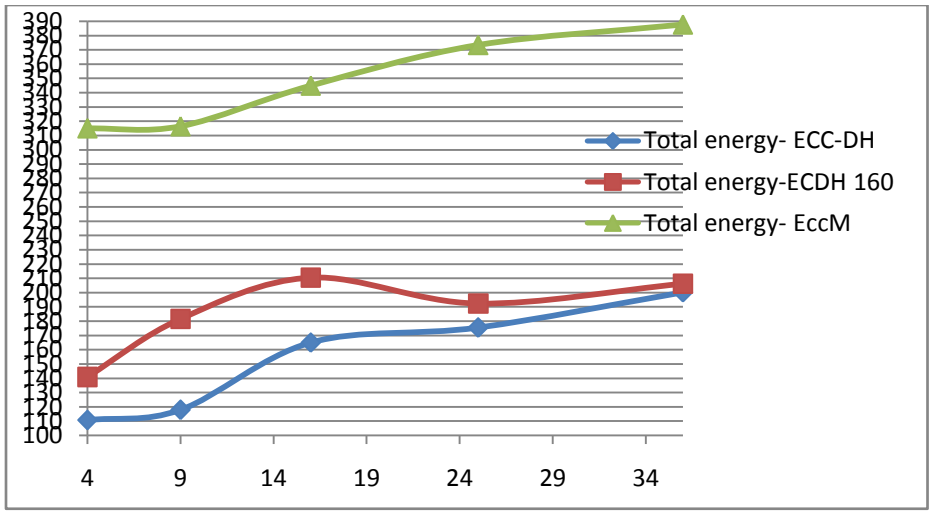
Таблица 41.

3.8 Анализ на ECC-DH, ECDH и EccM

Анализата ще се отнася за ECC-DH, ECDH и EccM поради това че тези приложения се отнасят за протокол за разменна на ключови по елиптична крива. Също така анализата ще се извършва при същи условия, брой на възли, време, топология. Параметрите са 160 бита на всичките приложения. Ще обърнем внимание на консумация на енергия и изисканата за ресурси (EccM 163 бита).

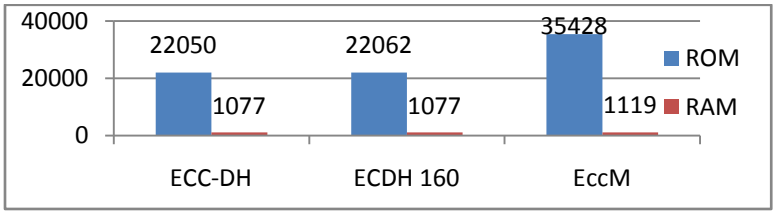


Фигура 51.Консумацията на енергия от страна на CPU при ECC-DH, ECDH и EccM (ECDH без използване на оптимизации).



Фигура 52. Средната стойност на съвкупната консумацията на енергия, при ECIES, ECDSA и ECDH с дължина на параметрите от 160 бита.

От графа можем да видим че е постигната целта и ECC-DH консумира по-малко енергия от ECDH. EccM консумира привидно доста повече енергия от ECC-DH, ECDH.



Фигура 53. ROM и RAM изискани от ECC-DH, ECDH и EccM където дължина на параметрите е 160 бита.

Тук EccM изиска най-много ресурси, което е и логично ако се вземе в предвид че троши най-много енергия. От тук идваме до заключение че EccM е не ефективно в сравнение с ECC-DH и ECDH. ECC-DH и покрай това че не се драстично различава в изисканата на ресурси то е за отреден процент по-добро, ако предвид се вземе трошенето на енергия.

3.9 TinySec описание и анализ

TinySec, е механизъм за шифроване на канално ниво, което означава че е първата част от решението за сигурност на tiny устройствата. Ядрото на TinySec представлява ефективен блок шифър. Обикновено използва единствен симетричен ключ кой е споделен между сензорните възли в мрежата. Преди предаване на пакетите, всеки възел ги шифрова данните и ги автентизира (authenticate) с Message Authentication Code (MAC).

Когато се използва TinySec, всеки възел може да комуникира с друг възел само ако е програмиран с същ ключ. Без допълнителни конфигурации в приложението, ще се използва файла за ключа (key-file) където се използва ключ по подразбиране:

Default key 6D524D67F24F178B0A69933FDD6C6F7B

При анализата ще използваме стойност на ключа по подразбиране.

compiled testTestTinySec to build/mica2/main.exe	
19960	**bytes in ROM**
707	**bytes in RAM**

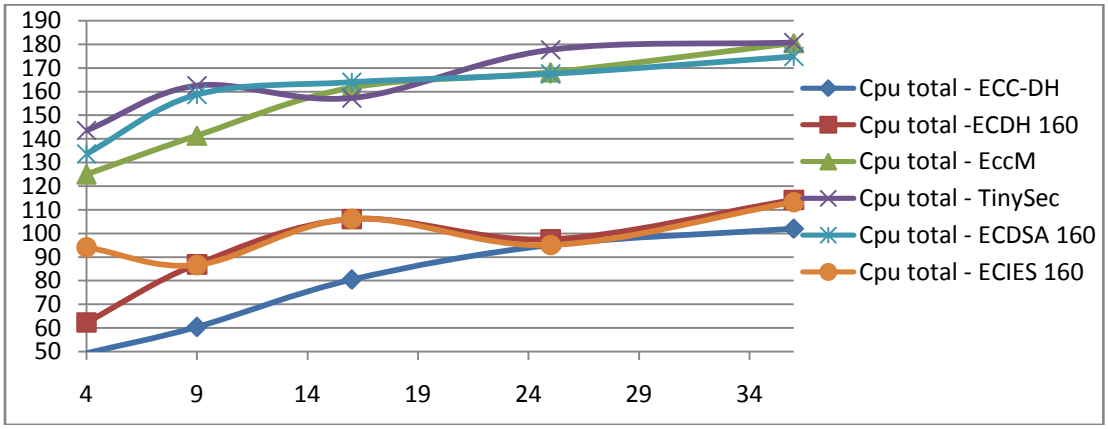
Таблица 41. ROM и RAM изискани от приложението

Когато приложението се компилира в машинния език на Atmel AVR се получават 8405 редови машинен код.

Симулацията за консумация на енергия при топология RANDOM за период от 15 секунди (изчислена е средната стойност). Резултатите са представени в Таблица 42. (mJ).

Брой възли	4	9	16	25	36
Cpu total	143.6109545	162.480233	157.250490	177.681979	180.7794262
Radio total	214.124906	208.93854	210.920670	234.985263	216.943338
Total energy	357.735860	371.418773	368.171160	412.667242	397.7227643

Таблица 42.



Фигура 54.Консумацията на енергия от страна на CPU

Фигура 54 ни представлява график на сравнение на консумацията на енергия на CPU от всички до сега споменати приложения. От тук можем да заключим че криптографията по елиптична крива може да се сравнява с симетрична криптосистема когато става дума за консумация на енергия.
Ако извършим сравнение на изисканата за ROM и RAM на TinySec с другите приложения ще забележим че за добър процент е в предимство този път.

Заключение

Идеята на тази дипломна работа е да се представи криптосистема която ще е приспособима на изисканата на безжичните сензорни мрежи (устройства с ограничени ресурси). В дипломната работа се зададени понятия за безжични сензорни мрежи, заплахите по тяхната сигурност, описани се нужните компоненти с кои се изгражда, прилага и симулира едно приложение за безжични сензорни мрежи. По запознаването с компонентите се обръща внимание на елиптичните криви. Обзор на основните понятия на криптографията по елиптична крива, математическата основа, стандарти, параметри, архитектура, приложимостта на елиптичната крива при изграждане на приложение за разменна на ключове, дигитален подпис и шифроване и де шифроване на данните, по което следва анализ на приложенията.

Можем да заключим че криптографията по елиптична осигуря по-голяма сигурност за малка дължина на ключа, което е един от главните недостатъци на асиметричната криптография. По-малката дължина на ключа осигуря по-бързо изпълнение по-малка консумация на енергия. Всичко това може да осигури приложимост на криптография по елиптична крива в устройства с ограничени ресурси като сензорните възли.

От извършената анализа ние можем да заключим че е осъществимо прилагането на ECC в безжични сензорни мрежи. Важно е да се спомене че е възможно използване на Inline Assembly код кой значително го ускорява извършването на операциите, също така тук може да се използват различни техники за оптимизация. При използването на техники за оптимизация требва да внимаваме коя характеристика искаме да е по ефективна, и като това ще се отрази на другите характеристики. Потребителя може да си го конфигурира приложението според своите нужди.

Използването на TinyOS е поради това че представлява OS за ниско мощни, гъвкави WSN за поддръжане и координация на оперативните нужди и поради две видни характеристики това че е отворен код (open-source), и е модулно базирана рамка (framework).

Като главни недостатъци можем да ги нагласим конфигурирането на TinyOS така че да работи с всички модули, друг недостатък е това че няма активна подържа, и трудно се намира отговор на проблемите на кои може да се натъкне. Също така за използване на отредени модули необходимо е конфигуриране. Прилагането на съществуващите приложения изискаше конфигуриране, и поправяне на възникналите грешки. Като трудност се прояви намирането на съответния хардуер и от този причина анализа се опира на резултатите получени при симулация. При симулация времето кое се получава при изпълнение на функциите е с голяма грешка, поради това анализа се базира на консумиране на енергия, изискането на ресурси, различните дължини на ключовете, параметри, като и използване на различните методи и техники за оптимизация.

При извършеното сравнение между приложения с ECC и приложението TinySec се вижда че ECC може да се сравнява с симетричната криптосистема, при прилагане в устройства с ограничени ресурси. С това се доказва че асиметричната криптографията може да се прилага в системи с ограничени ресурси, но въпреки получените резултати и по-нататък треба да се продължи с работа за усъвършенстване на приложенията а с това и продължаване на работния век на един сензорен възел а с това и на цялата безжична сензорна мрежа.

Използвана литература

[1] Lewis, F.L., “Wireless Sensor Networks,” Smart Environments: Technologies, Protocols, and Applications, ed. D.J. Cook and S.K. Das, John Wiley, New York, 2004.

[2] Townsend C.P, Hamel M.J., Arms S.W. (2001): Telemetered Sensors for Dynamic Activity & Structural Performance Monitoring, SPIE’s 8th Annual Int’l Conference on Smart Structures and Materials, Newport Beach, CA.

[3] SOHRABY, Kazem, MINOLI, Daniel, ZNATI, Taieb, Wireless sensor network: Technology, Protocols, and Applications.[s.l.]:Wiley,2006.307 s ISBN 978-0-471-74300-2.

[4] I. F. Akyildiz,W. Su, Y. Sankasubramaniam, and E. Cayirci. “Wireless Sensor Networks: A Survey”, *Computer Networks*, 38:393–422, 2002.

[5] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, “Energy-Efficient Communication Protocol for Wireless Microsensor Networks”, in *Proc. 33rd Hawaii International Conference on System Sciences*, Jan. 2000.

[6] AKKAYA, Kernal,YOUNIS, Mohamed. A Survey on Routing Protocol for Wireless Sensor Networks. In Ad Hoc Network Journal. [s.l.]: Elsevier. [2007].

[7] http://www.xbow.com/wireless home.aspx, 2006.

[8] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. Pister. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.

[9] V. S. Miller. Use of elliptic curves in cryptography. In H. C. Williams, editor, *Advances in cryptology | CRYPTO '85*, volume 218 of *Lecture Notes in Computer Science*, pages 417{426, Springer-Verlag Inc., Berlin, Germany, August 1986.

[10] N. Koblitz. Elliptic Curve Cryptosystems. *Mathematics of Computation*, 48(177):203{209, January 1987.

[11] *ANSI X9.62: Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*. American National Standards Institute, New York, USA, 1999.

[12] *IEEE P1363-2000: IEEE Standard Speci¯cations for Public-Key Cryptography*. IEEE Computer Society Press, Silver Spring, MD, USA, 2000.

[13] *ISO/IEC 15946: Information Technology | Security Techniques: Cryptographic Techniques based on Elliptic Curves*. International Organization for Standardization, Geneva, Switzerland, 2002.

[14] *FIPS 186-2: Digital Signature Standard (DSS). 186-2*. National Institute for Standards and Technology, Gaithersburg, MD, USA, February 2000. Available for download at http://csrc.nist.gov/encryption.

[15] A. M. Odlyzko. Discrete logarithms: the past and the future. *Designs, Codes, and Cryptography*, 19(2-3):129{145, March 2000.

[16] W. Diffie and M.Hellman, New Directions in Cryptography, IEEE Transactions on Information Theory, vol. 22, no. 6, pp. 644-654, Nov. 1976

[17] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[18] V. Miller. Uses of elliptic curves in cryptography. In *Lecture Notes in Computer Science 218: Advances in Crytology - CRYPTO ’85*, pages 417–426. Springer-Verlag, Berlin, 1986.

[19] P. Gaudry, F. Hess, and N. P. Smart. Constructive and destructive facets of weil descent on elliptic curves. Technical Report CSTR- 00-016, Department of Computer Science, University of Bristol, October 2000.
[20] G. Frey and H. Gangl. How to disguise an elliptic curve (weil descent). ECC '98, September 1998.
[21] HANKERSON, D. Guide to Elliptic Curve Cryptography. Springer, Berlin, 2004.
[22] JOHNSON, D. B., AND MENEZES, A. J. Elliptic curve dsa (ecsda): an enhanced dsa. In SSYM'98: Proceedings of the 7th conference on USENIX Security Symposium (Berkeley, CA, USA, 1998), USENIX Association, pp. 13–13.
[23] http://www.eecs.harvard.edu/~shnayder/ptossim/install.html
[24] Simulating the Power Consumption of LargeScale Sensor Network Applications, Victor Shnayder, Mark Hempstead, Borrong Chen, Geoff Werner Allen, and Matt Welsh Division of Engineering and Applied Sciences Harvard University
[25] D. T. Limited. jborzoi 0.9. http://dragongate-technologies.com/products.html,August 2003.
[26] D. Hankerson, J. L. Hernandez, and A. Menezes. Software implementation of elliptic curve cryptography over binary fields. *Lecture Notes in Computer Science*, 1965, 2001.
[27] I. Blake, G. Seroussi, and N. Smart. Elliptic curves in cryptography. *LMS Lecture Note Series*, 265, 1999.
[28] Silverman and Suzuki. Elliptic curve discrete logarithms and the index calculus. In *ASIACRYPT: Advances in Cryptology – ASIACRYPT: International Conference on the Theory and Application of Cryptology*. LNCS, Springer-Verlag, 1998.
[29] A. Menezes, S. Vanstone, and T. Okamoto. Reducing elliptic curve logarithms to logarithms in a finite field. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 80–89. ACM Press, 1991.
[30] N. I. of Standards and Technology. Recommended elliptic curves for federal government use. http://csrc.nist.gov/ CryptoToolkit/dss/ecdsa/NISTReCur.pdf, July 1999.
[31] LIU, A., AND NING, P. Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks. In 7th International Conference on Information Processing in Sensor Networks (IPSN 2008) (2008), pp. 245–256.
[32] An Liu, Peng Ning, "TinyECC: A configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks", *Proceeding of the 7th International Conference on Information Processing in Sensor Networks* (IPSN 2008), SPOTS Track, pp. 245-256, April 2008.
[33] D.E. Knuth. *The Art of Computer Programming*, volume 2: Seminumerical Algorithms. Addison Wesley, third edition, 1997. ISBN: 0-201-89684-2.
[34] A.J. Menezes, P. C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. ISBN: 0-8493-8523-7.
[35] N. Gura, A. Patel, and A. Wander. Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In *Proceedings of the 2004 Workshop on Cryptographic Hardware and Embedded Systems (CHES 2004)*, pages 119–132, August 2004.
[36] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to EllipticCurve Cryptography*. Springer, 2004.
[37] Certicom Research. Standards for efficient cryptography – SEC 2: Recommended elliptic curve domain parameters. http://www.secg.org/collateral/sec2_ final.pdf, September 2000.
[38]*http://www.codeproject.com/KB/cpp/C___Code_Optimization.aspx*
[39] Optimizing software in C++ An optimization guide for Windows, Linux and Mac platforms By Agner Fog. Copenhagen University College of Engineering. Copyright © 2004 - 2010. Last updated 2010 09-25.