TOMÁS ALBUQUERQUE AZEVEDO

PROTOCOLO DE AUTENTICAÇÃO SEGURO DE UM SISTEMA BODYCOM™

São Paulo
2015

TOMÁS ALBUQUERQUE AZEVEDO

PROTOCOLO DE AUTENTICAÇÃO SEGURO DE UM SISTEMA BODYCOM™

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para graduação em Engenharia de Computação.

São Paulo
2015

TOMÁS ALBUQUERQUE AZEVEDO

PROTOCOLO DE AUTENTICAÇÃO SEGURO DE UM SISTEMA BODYCOM™

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para graduação em Engenharia de Computação.

Engenharia de Computação e Sistemas Digitais

Orientador: Prof. Dr. Marcos A. Simplicio Jr.
Coorientador: Prof. Dr. Paulo S. L. M. Barreto

São Paulo
2015

FICHA CATALOGRÁFICA

Azevedo, Tomás Albuquerque

Protocolo de autenticação seguro de um sistema BodyCom™ / T. A. Azevedo -- São Paulo, 2015.

61 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Criptografia 2.Tecnologia BodyCom™ 3.Cifra de bloco Speck 4.Aplicação de controle de acesso I.Universidade de São Paulo. Escola Politécnica.
Departamento de Engenharia de Computação e Sistemas Digitais II.t.

DEDICATÓRIA

Dedico este trabalho à minha mãe, Cristina, pelo apoio e compreensão em todas as minhas escolhas.

Ao meu pai, José Osório, que foi o parceiro que não tive no projeto e a pessoa que me inspirou a ser engenheiro.

À minha namorada e amiga, Mariana, pelo companheirismo e força dada sempre, e por toda sua dedicação no banner deste trabalho.

E aos meus amigos que me acompanharam e me apoiaram durante essa longa jornada que percorri na Poli

AGRADECIMENTOS

À minha família pelo apoio de sempre e pelas discussões sempre muito proveitosas que me tornaram a pessoa que sou.

Aos meus professores e orientadores, Marcos Simplicio e Paulo Barreto pela orientação, dedicação e disposição durante todos os momentos necessários.

Ao meu orientador durante meu intercâmbio na University of Birmingham, Flavio Garcia, que teve a ideia inicial para o projeto e que gentilmente me emprestou o kit de desenvolvimento para a continuação do projeto aqui no Brasil.

Finalmente, a todos os professores e colegas do curso de Engenharia de Computação da Escola Politécnica da USP, que me acompanharam e me guiaram durante todos esses anos.

RESUMO

Em 2013, a Microchip Technology Inc. lançou uma tecnologia inovadora que realiza a comunicação entre uma placa base e suas unidades-móveis que utiliza o corpo humano como meio de transmissão. Essa tecnologia pode ser aplicada a diversas situações de controle de acesso para facilitar sua usabilidade.

Acreditando no potencial dessa inovação e verificando que a aplicação modelo não realiza essa comunicação de modo seguro, esse projeto tem como objetivo estudar possíveis protocolos de segurança que sejam viáveis junto ao kit de desenvolvimento disponibilizado pela Microchip e incluir esse protocolo na aplicação modelo de forma a torná-la segura.

Ainda, o projeto propõe duas contribuições originais, a primeira refere-se a uma melhoria para plataformas leves no protocolo simétrico proposto por Lim e Lee e a segunda consiste na implementação para plataformas de 8 bits em linguagem C da cifra Speck, proposta em 2013 pela NSA.

Palavras-chave: Criptografia, Tecnologia BodyCom™, Cifra de bloco Speck, Aplicação de controle de acesso

ABSTRACT

In 2013, Microchip Technology Inc. launched an innovative technology which makes possible communication between a centralized controller and its mobile units utilizing the human body as the transmission medium. This technology can be applied to multiple access control scenarios improving their usability.

Believing in this innovation's potential and verifying that the demo application does not provide a secure communication, this project's objectives are to study security protocols that are viable when inserted into the development kit provided by Microchip and actually implement this protocol into the application so that it becomes secure.

Furthermore, the project proposes two original contributions: the first being an improvement for lightweight applications in the symmetric protocol proposed by Lim and Lee and the second an implementation for 8-bit platforms in C programming language of the Speck cipher, proposed in 2013 by the NSA,

Palavras-chave: Criptography, BodyCom™ Technology, Block chipher Speck, Access control application

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Objetivo

O sistema BodyCom™ da Microchip utiliza uma tecnologia inovadora de curto alcance que permite a comunicação entre um controle centralizado (base) e uma unidade-móvel sem-fio, utilizando o corpo humano como o meio de transmissão. A solução é inovadora pois possibilita a criação de sistemas cuja autenticação se dá através do corpo humano. Por exemplo, poderia-se desenvolver aplicações que destravam uma porta quando o usuário, com uma unidade-móvel em seu bolso, encosta na maçaneta, sendo que a comunicação entre a maçaneta e a unidade-móvel é estabelecida com sucesso pelo corpo do usuário, destravando a porta.

O objetivo do projeto é investigar o funcionamento do Kit de desenvolvimento da tecnologia BodyCom™ da Microchip, identificando e testando sua segurança. Ainda, o projeto tem o objetivo de criar um protocolo de autenticação seguro, entre a unidade-móvel e o controle centralizado, baseado nessa tecnologia.

1.2 Motivação

A tecnologia BodyCom™ pode ser utilizada em diversos tipos de aplicações já existentes no mercado com outros tipos de tecnologia, como aplicações de segurança pessoal, aplicações médicas, de gerenciamento de perfil de usuário e de controle de acesso.

Dentre as aplicações de segurança pessoal, podem ser ressaltadas as aplicações que requerem o acionamento de algum componente por um usuário de modo seguro, como ferramentas elétricas, computadores e armas de fogo. Nesse caso, apenas o usuário com a posse de uma unidade-móvel registrada na base conseguiria acionar o funcionamento do componente. A tecnologia poderia ser utilizada para, por exemplo, ligar uma furadeira, onde ela só

poderia ser ligada pela pessoa com posse da unidade-móvel, impossibilitando que pessoas não autorizadas, crianças no caso, liguem a furadeira. Outra aplicação sugerida seria colocar o controle centralizado em uma arma para garantir que apenas o policial que possui a unidade-móvel (acoplada, por exemplo, a seu relógio) consiga dispará-la (METZ, 2013). Além dessa aplicação, uma empresa italiana diz estar utilizando a tecnologia BodyCom™ para assegurar que os motociclistas não andem sem capacete, com uma unidade-móvel acoplada no capacete e o controle centralizado no guidão da motocicleta (METZ, 2013).

Em aplicações médicas, a tecnologia BodyCom™ pode ser utilizada para controlar o acesso aos quartos dos hospitais e ainda monitorar pacientes. Em aplicações de gerenciamento de perfil, a tecnologia pode ser utilizada para separar as informações referentes a cada uma das “unidades-móveis”. Nesse caso, se cada usuário possuir uma unidade-móvel a plataforma que possui o controle centralizado como, por exemplo, um vídeo game ou esteira de exercício, consegue identificar o usuário apenas pelo toque e gerenciar suas ações separadamente.

A tecnologia BodyCom™ pode ser utilizada para aplicações de controle de acesso como, por exemplo, o controle de acesso a carros e casas. Nesses caso, o acesso seria garantido ao usuário que possuir uma unidade-móvel cadastrada após o toque na maçaneta da porta. Dessa forma, como o sistema promete transmitir a informação entre o controle centralizado e a unidade-móvel de um modo seguro, exatamente por utilizar o corpo humano como meio de transmissão e porque “não há um canal mais seguro que o corpo humano”, a tecnologia BodyCom™ ajuda a prevenir o problema de ataque de retransmissão (*replay attack*), muito comum nos sistemas de segurança de entrada remota sem chave, como em automóveis (GOMEZ, 2013).

1.3 Justificativa

O projeto foi idealizado e iniciado como um projeto de pesquisa durante o intercâmbio do proponente na University of Birmingham, no Reino Unido, sob a

orientação do Prof. Dr. Flavio Garcia. O intercâmbio teve duração de um ano, mas o projeto de pesquisa ocorreu durante as férias de verão e, portanto, durante dois meses e meio. Assim, devido ao curto prazo, o projeto não pôde ser finalizado, de modo que o projeto de formatura fará a continuação e ampliação do projeto iniciado no exterior.

A tecnologia inovadora BodyCom™ permite que sua implementação não necessite de um design com RF antena nem cristais externos para estabilizar a frequência do canal de transmissão, pois a frequência utilizada é muito baixa. A tecnologia está de acordo com as radiações emitidas no corpo e não causa danos ao organismo, pois a corrente elétrica transmitida é mínima. Como a comunicação é feita pelo corpo humano, não é necessário um transceptor sem fio e isso, atrelado ao fato de não utilizar campos indutivos de alta tensão para realizar a transmissão faz com que a solução da Microchip tenha um consumo muito baixo de energia. Ainda, como a tecnologia utiliza uma autenticação bidirecional através do corpo humano, ela se torna muito mais segura que outras tecnologias que possibilitam os ataques de retransmissão.

No entanto, apesar do fato de utilizar o corpo humano como meio de transmissão trazer segurança para o sistema, ele por si só não é totalmente à prova de ataques, por isso, a tecnologia suporta a utilização de criptografia para adicionar ainda mais segurança às suas aplicações. No entanto, nenhum protocolo de segurança padrão é utilizado na aplicação *DEMO* do kit de desenvolvimento e as aplicações que utilizam esse código-fonte, apesar de estarem protegidas de ataques de retransmissão pela transmissão pelo corpo humano, não estão protegidas por ataques que forcem a transmissão de pacotes modificando o número de identificação. Nesse caso, é possível quebrar a segurança da aplicação *DEMO* com força bruta até encontrar algum número de identificação que esteja registrado no controle centralizado.

Portanto, a elaboração de um protocolo de segurança utilizando criptografia é extremamente importante para que a tecnologia se torne ainda mais segura e ganhe mais importância no mercado.

Um resultado importante do trabalho desenvolvido é uma contribuição original: melhorias de segurança e de implementação propostas para o protocolo de identificação Lim-Lee. Essa contribuição é descrita na seção 5.1.2.

1.4 Metodologia de trabalho

A metodologia de trabalho utilizada no projeto de formatura seguiu as seguintes etapas:

1. Levantamento das limitações de hardware e software do kit de desenvolvimento:

   Tratando-se de microcontroladores, que possuem limitações de hardware como baixa disponibilidade de memória e de software como uma linguagem C para dispositivos (Embedded C Language) que é ligeiramente diferente da linguagem C padrão, suas limitações são os maiores desafios do projeto. Como diversos os algoritmos de criptografia (em especial os assimétricos) são complexos e ocupam um espaço considerável na memória, os mesmos podem não caber na memória dos microprocessadores. Portanto, foi necessário levar em consideração os seguintes aspectos do kit de desenvolvimento:

   - Tamanhos das memórias de dados (ROM e RAM);
   - Tamanho da memória de programa;
   - Detalhes da linguagem C embutida (tamanho máximo dos inteiros, se engloba unsigned ou signed, etc...);
   - Detalhes do tamanho e formato dos pacotes do canal de transmissão;

2. Estudo da aplicação *DEMO* do kit de desenvolvimento BodyCom™:

   Foi estudado o código-fonte da aplicação *DEMO* do kit de desenvolvimento BodyCom™, que simula a comunicação através do

corpo humano, para verificar se não havia nenhum mecanismo de segurança já implementado. Constatado que não, o estudo da aplicação *DEMO* foi importante para a determinação do melhor modo de desenvolvimento do protocolo de segurança junto com a aplicação.

3. Estudo de viabilidade de protocolos de segurança:

   Nesta etapa estudou-se primeiramente a viabilidade de protocolos assimétricos de criptografia e após constatada sua inviabilidade com o kit de desenvolvimento, estudou-se a viabilidade de protocolos simétricos de criptografia.

4. Desenvolvimento e testes da aplicação

   Desenvolveu-se a aplicação segura utilizando a tecnologia BodyCom™ introduzindo um protocolo de segurança à aplicação *DEMO* do kit de desenvolvimento. Após o desenvolvimento foram realizados testes de integridade da segurança da nova aplicação.

5. Conclusões:

   Após o desenvolvimento do projeto, com base nas dificuldades e soluções encontradas, fez-se uma análise do trabalho executado, o que permitiu extrair ideias e conclusões sobre o desenvolvimento do sistema.

1.5 Organização

A seção 2 descreve os aspectos conceituais do projeto para melhor entendimento do funcionamento da tecnologia.

Na seção 3 é apresentada a especificação do projeto, com as limitações da tecnologia e da placa de desenvolvimento e seus impactos, os cenários de vulnerabilidade a serem solucionados pelo projeto e a especificação em alto nível do protocolo de segurança a ser desenvolvido.

Na seção 4 são avaliados esquemas de criptografia assimétrica e simétrica para identificação de entidades, e sua viabilidade tecnológica para a plataforma adotada neste projeto. Ainda na seção 4, são apresentadas melhorias originais para o protocolo de Lim-Lee e uma implementação original da cifra Speck em linguagem C para aplicações de 8 bits.

A seção 5 apresenta o desenvolvimento e os testes do projeto, com a implementação do protocolo de segurança junto à aplicação *DEMO* que utiliza a tecnologia BodyCom™.

Na seção 6 são apresentadas as conclusões gerais do trabalho.

2 ASPECTOS CONCEITUAIS

A tecnologia BodyCom™ utiliza uma frequência de 125 kHz para realizar a comunicação entre a base (controle centralizado) e a *wireless unit* (unidade-móvel) através do corpo humano, transformando-o em um emissor de baixa frequência, graças a sua alta permissividade a baixas frequências.

Ao tocar o sensor de toque da base, o usuário inicia o sistema de comunicação, eliminando a necessidade de iniciar o processo manualmente, o que é comum com sistemas de altas frequências (BAILEY, 2014).

Pequenas correntes são transmitidas através do corpo humano, gerando um campo eletromagnético na superfície da pele do usuário, possibilitando a comunicação com uma unidade-móvel quando a pele do usuário se aproxima a cerca de poucos centímetros ou a toca. Então, a *wireless unit* identifica o sinal recebido e envia outro sinal para a base identificá-la.

3 ESPECIFICAÇÃO DO PROJETO

A Microchip disponibiliza um kit de desenvolvimento para a tecnologia BodyCom™, como pode ser visto na Figura 1. O Kit consiste em (1) uma placa que representa o controle centralizado da tecnologia, composta principalmente pelo sensor de toque que ativa a comunicação, o microprocessador que torna a comunicação possível e uma tela de LCD para uma melhor interação com o usuário, e (2) por duas placas wireless menores que representam duas “unidades-móveis” da tecnologia. O kit vem pré-programado com um programa *DEMO* que simula as funcionalidades da tecnologia e ainda possui um modo *DEBUG*. O código-fonte do programa *DEMO* é disponibilizado para qualquer usuário que queira reprogramar o kit. É por meio desse kit que o projeto será desenvolvido, modificando o código-fonte para adicionar a funcionalidade de uma comunicação segura utilizando criptografia simétrica ou assimétrica.

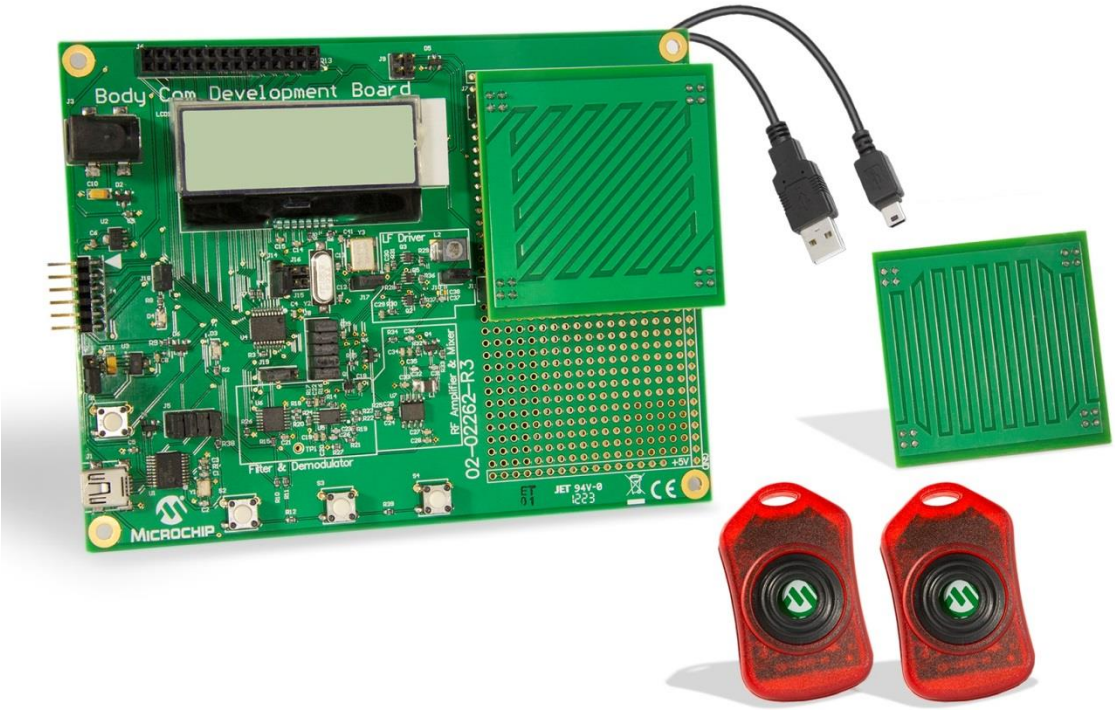

Figura 1 - Kit de desenvolvimento da tecnologia BodyCom™

3.1 Limitações

Com a *datasheet* dos microprocessadores da placa base (PIC16LF1829) e das unidades-móveis (PIC16LF1827) foi possível determinar o tamanho da

memória disponível para o projeto, como pode ser visto na Figura 2. O microprocessador da placa base suporta aproximadamente 8KB de memória ROM, enquanto o microprocessador das unidades-móveis possuem 4KB de memória ROM. Esse é um tamanho considerável para armazenar o código do programa, no entanto o tamanho da memória RAM também é muito importante para o desempenho de uma criptografia assimétrica, que requer muito mais memória RAM que uma criptografia simétrica, e o chip da placa base possui uma ocupação máxima de 1KB de RAM, enquanto o chip das unidades-móveis possuem apenas 384 bytes de RAM máxima, sem descontar o espaço de memória já ocupado pela aplicação *DEMO*.

Avaliações preliminares sugeriram que esquemas assimétricos de identificação poderiam exceder o espaço de RAM disponível na placa de desenvolvimento da BodyCom™, que é de apenas 1 KB na placa base e 384 bytes na placa das unidades-móveis. Por esse motivo, optou-se por investigar protocolos de autenticação não apenas usando criptografia assimétrica, mas também protocolos com criptografia simétrica e aumentar a flexibilidade de escolha.

PIC12(L)F1822/1840/PIC16(L)F182x/1847 Family Types

Device	Data Sheet Index	Program Memory Flash (words)	Data EEPROM (bytes)	Data SRAM (bytes)	I/O's(2)	10-bit ADC (ch)	CapSense (ch)	Comparators	Timers (8/16-bit)	EUSART	MSSP (I^2C™/SPI)
PIC12(L)F1822	(1)	2K	256	128	6	4	4	1	2/1	1	1
PIC12(L)F1840	(2)	4K	256	256	6	4	4	1	2/1	1	1
PIC16(L)F1823	(1)	2K	256	128	12	8	8	2	2/1	1	1
PIC16(L)F1824	(3)	4K	256	256	12	8	8	2	4/1	1	1
PIC16(L)F1825	(4)	8K	256	1024	12	8	8	2	4/1	1	1
PIC16(L)F1826	(5)	2K	256	256	16	12	12	2	2/1	1	1
PIC16(L)F1827	(5)	4K	256	384	16	12	12	2	4/1	1	2
PIC16(L)F1828	(3)	4K	256	256	18	12	12	2	4/1	1	1
PIC16(L)F1829	(4)	8K	256	1024	18	12	12	2	4/1	1	2
PIC16(L)F1847	(6)	8K	256	1024	16	12	12	2	4/1	1	2

Figura 2 - Especificação de memórias dos microprocessadores utilizados
(Tabela retirada da *datasheet* dos PICs e reduzida para caber na página)

Outro fator importante de limitação da placa de desenvolvimento da BodyCom™ é o tamanho máximo de cada pacote de dados de transmissão entre a placa base e cada unidade-móvel. O pacote de transmissão de dados da placa de desenvolvimento é composto por 1 byte que designa o tipo de comando realizado, 4 bytes que correspondem ao endereço da unidade-móvel, 1 byte correspondente ao tamanho dos dados transmitidos e o restante correspondente aos dados em si (Bailey, 2014). Após a análise do código-fonte, foi constatado, como mostra a Figura 3, que o tamanho máximo de dados que podem ser colocados em um pacote de transmissão é de 16 bytes. A escolha de cifras simétricas será norteada por essa métrica de espaço de memória do dispositivo.

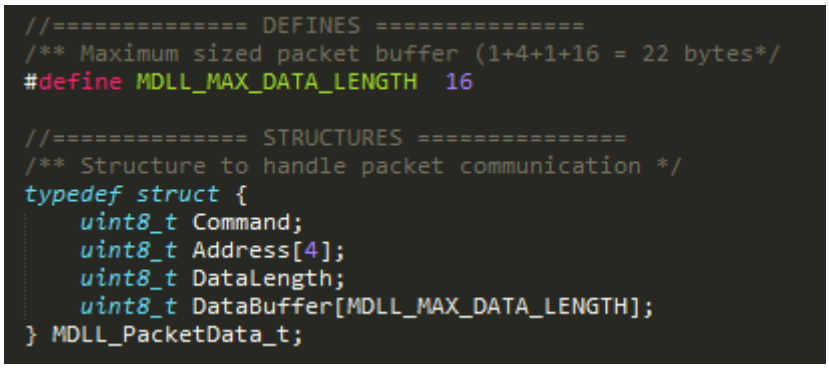

Figura 3 - Trecho do código-fonte com o tamanho máximo de dados em um pacote de transmissão.

A obtenção de uma licença PRO para o compilador XC8 da Microchip sem nenhum custo foi um avanço muito bem vindo no projeto, pois era algo que havia causado um empecilho previamente já que o código-fonte do kit de desenvolvimento da BodyCom™ só cabia na placa base com a compilação em modo PRO. No entanto, foi possível identificar que o código-fonte original ocupa 83,5% da capacidade de memória do microprocessador da placa base (6841 bytes dos 8KB totais) e isso pode causar algum empecilho ao inserir o código de autenticação. Portanto, na primeira etapa do projeto também foi inserida a tarefa de limpeza com código-fonte do kit de desenvolvimento para que todas as funcionalidades que não são relevantes ao desenvolvimento do projeto sejam removidas e apenas a funcionalidade de transmissão de dados pelo corpo humano permaneça no código. Só após a realização dessa etapa, e

consequentemente com a liberação da memória do microprocessador para a possibilidade de armazenar tanto o código de transmissão de dados quanto o código de criptografia e segurança, é que o projeto seguiu para a segunda etapa.

Além das limitações do kit de desenvolvimento que dificultam a elaboração do protocolo de segurança desenvolvido em software, também é necessário levar em consideração o consumo de energia do sistema. O sistema em si consome pouca energia pelas razões citadas anteriormente, mas a adição de um protocolo de autenticação pode aumentar um pouco esse consumo. Será necessário, então, quantificar esse aumento e verificar se ele causa grande modificação no consumo. É necessário que o sistema consuma o mínimo de energia possível para que ele possa ser o mais portátil (menores baterias) e dure o maior tempo possível sem precisar trocar suas baterias.

3.2 Protocolo de comunicação da aplicação *DEMO*

O protocolo original de comunicação da aplicação *DEMO* do kit BodyCom™ utiliza apenas o endereço da unidade-móvel para realizar a autenticação. Ao detectar um toque no painel, a base envia pacotes PING com o endereço de cada uma das unidades-móveis cadastradas na base. O comando PING é apenas um comando que possui o endereço da unidade-móvel no campo “Address” do pacote de transferência de dados e o campo de dados, “Data*Buffer*”, vazio. Caso o usuário que tocou no sensor da base tenha uma unidade-móvel cadastrada em sua posse, a unidade-móvel identifica o pacote recebido e, caso o endereço coincida com o seu endereço, ela envia um pacote semelhante ao pacote PING, com seu próprio endereço e com o campo de dados vazio, de volta para a base. A Figura 4 exemplifica esse simples protocolo, sendo a base (controle centralizado) representada pela letra A e uma unidade-móvel pela letra B.

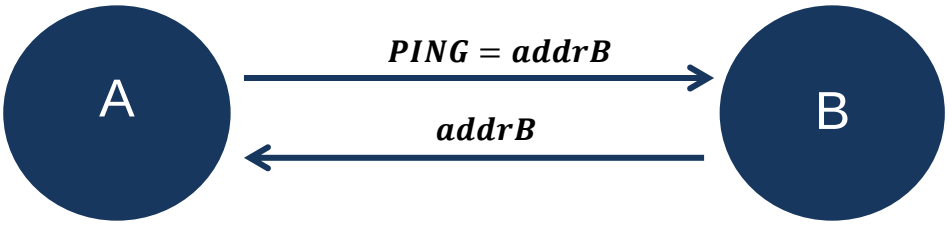

Figura 4 - Protocolo original da aplicação *DEMO*

O endereçamento dos pacotes na aplicação de demonstração é unilateral, ou seja, o campo de endereço do pacote é sempre preenchido com o endereço da unidade-móvel, independente da direção que o pacote percorre. Por isso, a placa de controle centralizado (letra A) trata o campo de endereço como destinatário ao enviar o pacote e como origem ao recebê-lo, e as unidades-móveis (letra B) tratam o campo de endereço de forma contrária, como origem ao enviar o pacote e como destinatário ao recebê-lo.

Percebe-se que o protocolo original não possui quase nenhuma segurança a não ser por não se comunicar com unidades-móveis que não tenham seu endereço pré-cadastrados na base. A base não envia pacotes para unidade-móveis que não estejam cadastradas em sua memória, no entanto, a base não verifica se o endereço de um pacote recebido está entre os endereços cadastrados.

O protocolo seguro desenvolvido nesse projeto utiliza o campo de dados do pacote de transferência para trocar informações criptografadas entre a base e a unidade-móvel para realizar a autenticação. O campo de endereço, “Address”, do pacote de transferência de dados não é criptografado, mas isso não afeta a segurança do protocolo.

3.3 Cenários de vulnerabilidade

A proposta desse projeto visa acrescentar segurança a uma tecnologia inovadora que já é relativamente segura por si só. No entanto, a tecnologia, apesar de ser segura contra ataques que interceptam o sinal de transmissão, já

que o sinal passa pelo corpo humano, não possui nenhuma segurança contra outros tipos de ataque.

Consideremos o cenário de controle de acesso de um usuário a sua casa. Nesse cenário, a aplicação *DEMO* recebe o pacote da unidade-móvel com o seu endereço de 4 bytes. O controle centralizado localizado na maçaneta da porta recebe o pacote e verifica o endereço para ver se é válido e, em caso positivo, o acesso é garantido. Assim, um ataque de força bruta obteria sucesso no controle de acesso, porque encontrar o endereço correto iterando entre os possíveis endereços de 4 bytes requer um processamento bem pequeno.

Além disso, por se tratar de um protocolo unilateral que não verifica se o endereço do pacote recebido é o mesmo do pacote enviado, um ataque por uma aplicação, que simula uma unidade-móvel que constantemente envia pacotes, poderia garantir acesso à placa base enviando um pacote com qualquer endereço, o que torna a aplicação de demonstração mais vulnerável.

Portanto, a elaboração de um protocolo de segurança com autenticação é muito importante para garantir que o sistema que incorpore a tecnologia seja mais seguro e para que a tecnologia se torne mais acessível.

3.4 Liberação de memória na aplicação *DEMO*

A aplicação *DEMO*, que veio com o kit de desenvolvimento BodyCom™, ocupa 6841 bytes dos 8192 bytes disponíveis de memória ROM e 490 bytes dos 1024 bytes disponíveis de memória RAM do chip PIC16LF1829 do controle centralizado do kit e ocupa 1763 bytes dos 4096 bytes de ROM e 175 dos 384 bytes de RAM do chip PIC16LF1827 das unidades-móveis. Esse é o principal limitador na inclusão e desenvolvimento de um protocolo de segurança que ocupe tão pouca memória disponível. O protocolo desenvolvido para a base do kit deve ter no máximo 1.3KB de memória ROM e 534 bytes de memória RAM e o protocolo desenvolvido para as unidades-móveis devem ter no máximo 2333 bytes de ROM e 209 bytes de RAM, um tamanho bem pequeno quando

se trata de segurança, onde geralmente um algoritmo mais complexo e mais seguro possui uma ocupação de memória maior. Por esse motivo, foi necessário remover as partes da aplicação *DEMO* que foram julgadas desnecessárias para o projeto e deixar apenas o código necessário para que a funcionalidade do BodyCom™ seja realizada.

A primeira parte do código a ser retirada foi a parte referente ao menu de *debug* da aplicação, um menu que informa ao usuário da aplicação diversas informações sobre ela, mas que apenas traz informações adicionais e desnecessárias para a comunicação em si. Ao se retirar essa parte, a aplicação diminuiu consideravelmente de tamanho e passou a ocupar 5022 bytes de memória ROM e 436 bytes de memória RAM, possibilitando ao protocolo de segurança ter no máximo 3181 bytes de ROM e 588 bytes de RAM.

Ao serem removidas as funcionalidades de utilização do display LCD, o código foi reduzido para uma ocupação de 3941 bytes de memória ROM e 342 bytes de memória RAM, possibilitando o protocolo de segurança ter no máximo 4251 bytes de ROM e 682 bytes de RAM. Como a funcionalidade de display LCD mostra-se útil para a demonstração da aplicação e como protocolos simétricos ainda cabem junto à aplicação *DEMO* mesmo com as funcionalidades de display, essa versão reduzida do código não foi utilizada.

4 PROCURA E VIABILIDADE DE PROTOCOLOS DE SEGURANÇA

4.1 Protocolos assimétricos

A criptografia simétrica produz algoritmos bem mais leves que a criptografia assimétrica. No entanto a segurança de um algoritmo simétrico é mais limitada que a de um algoritmo assimétrico, dado que se adota uma chave compartilhada que pode ser extraída de um dos componentes e utilizada nos demais, enquanto em um algoritmo assimétrico cada componente tem sua própria chave de uso exclusivo. Por esse motivo, foram avaliados vários algoritmos de criptografia assimétrica, procurando um que fosse leve o suficiente para rodar no kit BodyCom™, conforme relatado a seguir.

4.1.1 Criptografia de Curvas Elípticas

O aluno do IME Gustavo Zanon juntamente com seu orientador de iniciação científica e coorientador desse projeto, Dr. Paulo S. L. M. Barreto, obtiveram resultados promissores com criptografia assimétrica de curvas elípticas, conseguindo rodar uma autenticação com aproximadamente 800 bytes de memória RAM simulando em um computador, cerca de 200 bytes a menos que os 1024 bytes máximos do chip PIC16 do kit BodyCom™.

A criptografia de curvas elípticas (ECC) proporciona o desenvolvimento de um algoritmo com criptografia assimétrica, sendo, portanto, mais seguro que um algoritmo com criptografia simétrica. A ECC se baseia em utilizar curvas elípticas de Edwards, da forma $\pm x^2 + y^2 = 1 + dx^2y^2$ com coordenadas sobre um corpo finito, para criar esquemas de encriptação e assinatura. A aritmética com pontos de curvas desse tipo permite definir multiplicação de um ponto P da curva por um escalar inteiro k, de tal maneira que encontrar o valor de k a partir dos pontos P e kP é computacionalmente inviável, exigindo tempo exponencialmente grande no número de bits das coordenadas.

Os testes realizados no computador foram promissores, mas não foi possível compilar o código para o PIC16, pois a memória ROM ocupada ultrapassava a memória máxima do chip. Então, para testar o desenvolvimento do código, foi utilizado outro chip da Microchip, um PIC24FJ32GA002, um chip de 16-bits com 32KB de memória ROM (o quádruplo o PIC16) e 8KB de memória RAM (8 vezes mais que o PIC16).

Os testes do algoritmo de curvas elípticas, apresentados na Tabela 1 no PIC24 revelaram um tamanho aproximado de 11500 bytes de memória ROM e 1000 bytes de memória RAM, como pode ser observado na tabela abaixo. Apesar da ocupação de memória RAM estar dentro da capacidade do PIC16 em algumas versões, a ocupação de memória ROM é bem maior que os 8KB máximos do PIC16, sendo, portanto, impossível de realizar tais testes mesmo com o chip vazio, ainda mais incluindo o algoritmo junto com o código *demo* do kit BodyCom™. Portanto, o algoritmo de criptografia de curvas elípticas revelou-se inviável para o projeto e outros algoritmos assimétricos começaram a serem considerados.

Tabela 1 - Resultados dos testes do protocolo de curvas elípticas

	PIC24 - Otimização por espaço	
	RAM Size (Bytes)	**ROM Size (Bytes)**
v29.04.15	986	12786
v30.04.15	986	11499
v04.05.15	1020	11415
v05.05.15	1020	11499
V17.05.15	1020	11514

4.1.2 Protocolo de Identificação baseado em Polinômios Multivariados Quadráticos

Outro algoritmo assimétrico que foi considerado com potencial para rodar no PIC16 do kit BodyCom™ foi um protocolo de identificação baseado em polinômios multivariados quadráticos apresentado na dissertação de mestrado do Fábio Monteiro, do IME-USP, (MONTEIRO, 2012).

Definição

Esse protocolo “consiste em resolver um sistema de equações polinomiais multivariadas quadráticas sobre um corpo finito. Até hoje não se conhece algoritmo, nem mesmo quântico, de tempo polinomial que possa resolver esse problema, fazendo com que sistemas criptográficos baseados nessa primitiva mereçam ser investigados e desenvolvidos como reais candidatos a proverem nossa criptografia pós-quântica” (MONTEIRO, 2012).

Na tese de Monteiro, foi apresentada uma versão aprimorada do protocolo MQID-3, apresentado por Sakumoto, Shirai e Hiwatari em 2011, na qual é obtida uma redução de comunicação de aproximadamente 9%.

Testes

A Tabela 2 mostra os testes realizados tanto no PIC24 quanto no PIC 16. Os testes das primeiras versões só puderam ser realizados o PIC24, pois suas ocupações ultrapassavam as ocupações máximas do PIC16. No entanto, a terceira versão contou com melhorias no algoritmo que fizeram com que sua ocupação chegasse a 3330 bytes de memória ROM e 458 bytes de memória RAM no PIC24. Foi possível compilar essa versão para o PIC16 e os resultados de ocupação obtidos foram promissores: 4051 bytes de memória ROM e 624 bytes de memória RAM. Apesar de estarem apenas 3100 bytes livres na aplicação do kit BodyCom™ após a retirada da funcionalidade de *debug* da aplicação *DEMO*, talvez algumas melhorias poderiam ser aplicadas

ao protocolo de Sakumoto-Monteiro para que o código atingisse essa ocupação. No entanto, ao rodar o teste, o seu tempo de execução mostrou-se da ordem de um minuto e meio para o PIC24 e incríveis 20 minutos para o PIC16, sendo um tempo muito alto para a execução de aplicações com o kit BodyCom™ – um minuto e meio é um tempo extremamente grande e inviável para uma autenticação.

Tabela 2 - Resultados dos testes do protocolo de polinômios multivariados quadráticos

	PIC16 - Otimização por espaço			PIC24 - Otimização por espaço		
	RAM Size (Bytes)	ROM Size (Bytes)	Time (min)	RAM Size (Bytes)	ROM Size (Bytes)	Time (min)
v17.05.15				1382	15735	
v18.05.15				1314	13389	01:17
v19.05.15	624	4051	20:00	458	3330	01:16

Conclusão

Os tempos desfavoráveis de execução aliados à improbabilidade de melhorar esses tempos com quaisquer melhorias concebíveis no protocolo levaram à recomendação de descartar essa vertente, e investigar um último protocolo assimétrico para uso potencial no projeto: o esquema de identificação do Probema de Núcleos Permutados (Permuted Kernel Problem – PKP).

4.1.3 Problema de Núcleo Permutado

O problema de núcleo permutado (Permuted Kernel Problem – PKP) foi proposto primeiramente por Adi Shamir em 1989 (SHAMIR, 1989) e mostrou-se ser um protocolo assimétrico eficiente para estabelecer identidades de usuários a assinar mensagens digitalmente, utilizando chaves públicas e privadas compactas e provas de conhecimento-zero.

Definição

Em criptografia, uma prova de conhecimento-zero é um método utilizado para autenticação que possibilita uma parte provar para outra que uma declaração é

verdadeira sem revelar nada além da veracidade da declaração. Um exemplo famoso e simples que apresenta a ideia geral de uma prova de conhecimento-zero foi publicado primeiramente por Louis C. Guillou e Jean-Jacques Quisquater em 1988, e se baseia na situação em que um usuário i deseja provar para um usuário j que possui uma chave secreta, mas sem revelá-la.

Na história, há uma caverna em formato circular com a entrada por um lado e uma porta trancada do lado oposto (como é ilustrado na figura abaixo).

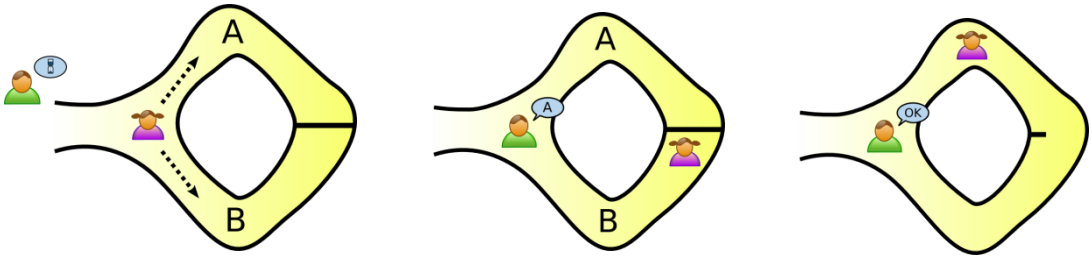

Figura 5 - Ilustração do exemplo de provas de conhecimento-zero
(Ilustrações retiradas de: https://en.wikipedia.org/wiki/Zero-knowledge_proof)

Os caminhos da caverna são rotulados A e B.

1. O usuário i entra na caverna, enquanto o usuário j espera do lado de fora, e escolhe um dos caminhos A ou B aleatoriamente.
2. O usuário j entra na caverna e escolhe aleatoriamente um caminho, A ou B, pelo qual ele deseje que o usuário i retorne.
3. Se o usuário i tiver posse da chave secreta que abre a porta, ele poderá retornar pelo caminho escolhido por j em 100% das vezes. Em caso contrário, como o caminho foi escolhido aleatoriamente no passo 2, o usuário i tem 50% de chance do caminho escolhido por ele ser o mesmo caminho escolhido por j.

A segurança do protocolo é estabelecida pelo número de vezes que ele é realizado. A chance do usuário i acertar o caminho escolhido por j em todas as vezes nas quais o protocolo é realizado diminui exponencialmente e, portanto, se o protocolo for executado 20 ou 30 vezes a segurança é grande o suficiente para ser aceita em protocolos computacionais, dependendo do nível de

segurança esperado. Assim, ao final de todas as execuções do protocolo, se o usuário i acertar todos os caminhos escolhidos por j, ele é autenticado; em caso contrário, a primeira vez que o usuário i errar o caminho escolhido, o protocolo é interrompido e a autenticação falha.

O problema de núcleo permutado de Shamir utiliza provas de conhecimento-zero mais complexas que a exemplificada para estabelecer autenticação entre dois usuários, mas a ideia básica é a mesma. Os usuários que participam do protocolo devem estabelecer previamente uma matriz A e um número primo p. Como o protocolo foi desenvolvido com base em permutações da matriz A, cada usuário escolhe uma permutação aleatória π da matriz (que serve como chave secreta) e um vetor aleatório V tal que uma permutação de V seja pertencente à A. Então, utilizando provas de conhecimento-zero, os usuários podem comprovar suas identidades um ao outro provando seu conhecimento da permutação secreta π sem que bisbilhoteiros e verificadores desonestos obtenham π.

Especificação

O protocolo do problema de núcleo permutado desenvolvido para esse projeto tem como base as especificações definidas pelo seu criador, Adi Shamir (SHAMIR, 1989), mas também considera os estudos da segurança do protocolo e as melhorias propostas por Thierry Baritaud e sua equipe (BARITAUD, et al., 1993) e por Guillaume Poupard (POUPARD, 1997).

O esquema opera da seguinte forma entre Alice e Beto. Inicialmente, Alice e Beto escolhem

- um número primo p,
- uma dimensão inteira n,
- uma matriz A de dimensão $n \times n$ e componentes inteiros módulo p,
- uma função de hash H.

Alice então escolhe como chave secreta uma permutação aleatória π sobre n valores, e como chave pública um vetor V de dimensão n e componentes

inteiros módulo p tal que $\pi(V) \in \ker(A)$. Numa instância do protocolo de identificação, em que Alice quer provar sua identidade a Beto, ambos repetem o seguinte protocolo:

1. Alice escolhe um vetor aleatório R de dimensão n e componentes inteiros módulo p, e também uma permutação aleatória σ. Alice envia para Beto os valores de hash $h_0 = H(\sigma, AR)$, $h_1 = H(\pi\sigma, \sigma(R))$.
2. Beto escolhe um valor aleatório c módulo p e pede a Alice o valor de $W = \sigma(R) + c * \pi\sigma(V)$.
3. Ao receber a resposta W de Alice, Beto escolhe um bit aleatório b e pede para Alice enviar σ se $b = 0$, ou então $\pi\sigma$ se $b = 1$. No primeiro caso, Beto confere se $H(\sigma, \sigma(A)W) = h_0$, e no segundo caso Beto confere se $H(\pi\sigma, W - c * \pi\sigma(V)) = h_1$.

Alice será sempre capaz de responder corretamente o valor de W. Um impostor que queira se fazer passar por Alice só terá 50% de chance de responder corretamente. Repetindo esse processo k vezes, a probabilidade de um falsário personificar Alice é de apenas 2^{-k}, um valor que pode ser tornado arbitrariamente pequeno.

Um aspecto interessante deste protocolo é que o falsário poderia tentar violar a segurança recuperando a chave privada π de Alice a partir de A e de V, uma vez que, por definição, $\pi(V) \in \ker(A)$., ou seja, $A(\pi(V)) = 0$. Contudo, conforme indicado por Shamir no artigo em que ele propõe o protocolo, esse problema é NP-difícil, e portanto intratável.

O protocolo acima faz uso de uma função de hash H. Para esse projeto, avaliaram-se para a função H o algoritmo Keccak (SHA-3) e construções baseadas nas cifras de bloco AES, Speck e Curupira2, todas operando no modo Matyas-Meyer-Oseas (MMO) devido à sua simplicidade e segurança.

Adotou-se um número de $k = 20$ repetições do protocolo para se obter um nível de segurança razoável para aplicações com a tecnologia do kit BodyCom™.

Testes

Os testes do protocolo foram realizados tanto no PIC24 quanto no PIC16 nas diversas versões do algoritmo. A Tabela 3 e a Tabela 4 mostram os testes de memória e tempo da versão do protocolo que obteve os melhores resultados.

Tabela 3 - Resultados dos testes do protocolo PKP no PIC16

	PIC16					
	PKP v27.05.15 (otimização de espaço)			PKP v27.05.15 (otimização de velocidade)		
	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)
Keccak	702	3439	23.1000	702	3877	19.6000
AES	512	3283	4.0325	512	3848	3.3650
Speck 128	429	2244	3.3490	429	2503	2.7860
Speck 96	417	2208	2.5610	417	2451	2.2540
Speck 80	411	2177	2.3385	411	2400	2.0740
Curupira2 (s/ tabelas)	421	2385	2.3560	421	2694	2.1185
Curupira2 (c/ TABS)	416	2576	1.5485	416	2895	1.3165
Curupira2 (c/ TABS, TAB0)	416	2601	1.5390	416	2920	1.3025
Curupira2 (c/ TABS, TAB0, TABX)	412	2836	1.4695	412	3146	1.2450

Tabela 4 - Resultados dos testes do protocolo PKP no PIC24

	PIC24					
	PKP v27.05.15 (otimização de espaço)			PKP v27.05.15 (otimização de velocidade)		
	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)
Keccak	550	3045	1.4200	550	3201	1.4169
AES	380	3168	0.3100	380	3303	0.3031
Speck 128	348	2493	0.3590	348	2613	0.3613
Speck 96	336	2439	0.3040	336	2556	0.3048
Speck 80	330	2421	0.2770	330	2538	0.2783
Curupira2 (s/ tabelas)	336	2787	0.5259	336	2985	0.5282
Curupira2 (c/ TABS)	336	2991	0.2190	336	3165	0.2155
Curupira2 (c/ TABS, TAB0)	336	3039	0.2170	336	3213	0.2160
Curupira2 (c/ TABS, TAB0, TABX)	336	3390	0.2116	336	3573	0.2086

Ao analisar os resultados dos testes, pode-se perceber que o desempenho no PIC24 é muito melhor do que no PIC16 em todos os tipos de cifra utilizados. Percebe-se também que a ocupação de memória ROM no PIC16 é um pouco melhor do que no PIC24 na maioria dos casos. Na questão da ocupação de memória ROM, acredita-se que essa divergência ocorre devido a utilização de uma compilação em modo PRO (já que há uma licença para o compilador do PIC16), que costuma gerar códigos até 40% menores do que uma compilação em modo FREE (utilizada no PIC24 pela falta de licença). Essa ocupação variou entre 2200 e 3500 bytes e como há apenas 2900 bytes livres na aplicação *DEMO* do kit BodyCom™ até o momento, nem todas as versões poderão ser utilizadas junto com o *demo*. Já na questão de desempenho, a enorme divergência de tempos de execução do algoritmo entre os dois PICs deve ter ocorrido devido a certas operações que não devem ser definidas no PIC16 e para aproximar esse desempenho do PIC24 seria necessário realizar tais operações em ASSEMBLY.

A ocupação de memória RAM da versão com Keccak é a única que ultrapassa a ocupação disponível de memória RAM (588 bytes), então essa versão não será considerada. A ocupação de memória ROM ficou bem abaixo da disponível em alguns casos, então esse critério foi contornado. O problema está no desempenho das versões no PIC16. Com os melhores tempos das versões no PIC24 seria possível incluir esse protocolo para diversas aplicações, pois um tempo de resposta entre 0,2 e 0,3 segundos é um tempo aceitável. No entanto, o melhor tempo no PIC16 é de 1,5 segundos, sendo o melhor tempo para uma versão que também caiba com folga no chip 2,3 segundos, o que são tempos inviáveis para algumas aplicações. Mesmo assim, há algumas aplicações, como o simples acesso à portas, que aceitam um tempo de resposta em torno de 1 e 2 segundos, portanto ainda é possível utilizar esse protocolo com o kit BodyCom™ e como essa é uma solução assimétrica, é uma solução mais segura do que uma simétrica.

4.2 Protocolos simétricos

4.2.1 Protocolo de Lim-Lee

Chae Hoon Lim e Pil Joong Lee do Departamento de Engenharia Elétrica da Universidade de Ciência e Tecnologia de Pohang, na Coreia do Sul, publicaram o artigo *Several practical protocols for authentication and key exchange* em 1995 com 5 protocolos de autenticação e troca de chaves para diferentes aplicações (LIM & LEE, 1995). No protocolo 1 é apresentada uma autenticação leve e muito eficiente para algoritmos simétricos e por isso esse foi o protocolo escolhido para exploração neste projeto.

Definição

No protocolo 1 de Lim-Lee define-se como K uma chave simétrica compartilhada pelos usuários e E uma cifra de bloco, portanto $E_K(X)$ denota uma encriptação da mensagem X com a chave secreta K. Ainda, define-se como R_i uma cadeia aleatória de dados escolhida pelo usuário i e ID_i como o identificador (endereço) do usuário i. Por fim, definem-se os operadores $\|$ e $\oplus$

como concatenação e a operação ou-exclusivo (XOR) entre cadeias, respectivamente.

O protocolo de Lim-Lee está exemplificado na Figura 6 e detalhado nos passos abaixo.

1. O usuário i escolhe uma cadeia aleatória R_i e envia ao usuário j essa cadeia encriptada com a chave K, $E_K(R_i)$.
2. O usuário j então recebe a mensagem, a decripta com a chave compartilhada K e recupera R_i. Ele então calcula uma nova chave $K_j \leftarrow K \oplus R_i$ pela concatenação de K e R_i, escolhe a cadeia aleatória R_j, e envia ao usuário i a mensagem $E_{K_j}(ID_i \parallel R_j)$, da concatenação do identificador de i com a cadeia aleatória de j encriptada com a nova chave K_j.
3. O usuário i calcula a chave $K_j \leftarrow K \oplus R_i$, com a chave simétrica compartilhada K e a cadeia aleatória R_i do passo 1 para decriptar a mensagem enviada por j. Ele então verifica se o identificador enviado ID_i equivale ao seu identificador. Em caso positivo, ele calcula a chave $K_i \leftarrow K \oplus R_j$ e envia ao usuário j a mensagem $E_{K_i}(R_i)$.
4. Por fim, o usuário j recebe a mensagem de i e calcula a chave $K_i \leftarrow K \oplus R_j$, com a chave simétrica compartilhada K e a cadeia aleatória R_j do passo 2 para decriptar a mensagem enviada por i. Com a mensagem decriptada, o usuário j verifica se a cadeia aleatória R_i extraída da decriptação coincide com a cadeia R_i escolhida no passo 2. Em caso positivo, a autenticação foi estabelecida com sucesso e os usuários podem trocar mensagens com segurança.

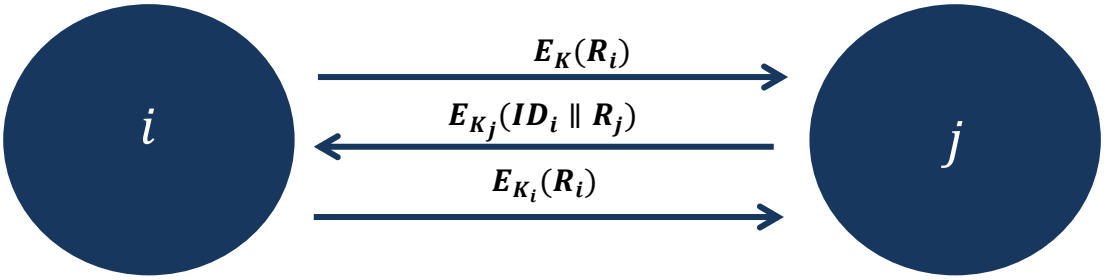

Figura 6 - Diagrama de exemplificação do protocolo Lim-Lee

Lim e Lee ainda analisam os ataques que poderiam ser tentados contra o protocolo e como a segurança do protocolo impede esses ataques. *"Um atacante se passando pelo usuário i poderia iniciar o protocolo apenas enviando uma cadeia aleatória de sua escolha ou enviar uma transmissão observada anteriormente, por outras comunicações pelo protocolo. Em ambos os casos, o atacante não consegue enviar uma resposta legítima no passo 3 por causa da nova cadeia aleatória enviada pelo usuário j de forma encriptada no passo 2"* (LIM & LEE, 1995). Ainda, a adição do identificador do recebedor no protocolo, o protege contra ataques paralelos, pois o torna direcionalmente dependente, impossibilitando a passagem de um atacante como um usuário, pois a reutilização de uma mensagem em outra comunicação se torna inútil.

Propostas de melhorias

Esta seção descreve uma contribuição original deste trabalho: melhorias de segurança e de implementação do protocolo de identificação Lim-Lee.

O protocolo de Lim-Lee realiza as operações $E_{K_j}(ID_i \parallel R_j)$ e $E_{K_i}(R_j)$, e com isso ele assume que a cifra utilizada é capaz de cifrar mensagens de tamanhos variáveis. Cifras desse tipo requerem uma complexidade maior e com isso uma ocupação de memória maior e em um projeto no qual a ocupação de memória é uma das maiores preocupações, o protocolo analisado apresenta uma falha determinante.

O protocolo também posterga a detecção de sucesso ou falha de decriptação até o último passo do protocolo, aumentando muito o tempo de detecção de uma falha no protocolo.

Ambos os problemas citados podem ser resolvidos com algumas modificações no protocolo. No primeiro passo, propõe-se invocar $E_K(ID_j \parallel R_i)$ ao invés de $E_K(R_i)$ e no terceiro passo propõe-se invocar $E_{K_i}(ID_j \parallel R_i)$ ao invés de $E_{K_i}(R_i)$ para que, desse modo, as cadeias a serem cifradas tenham sempre o mesmo tamanho. Assim, uma cifra que seja capaz de cifrar apenas mensagens de tamanhos fixos pode ser utilizada no protocolo, podendo ser uma cifra bem mais leve que uma cifra de tamanhos variáveis. Essas modificações também resolvem o segundo problema citado ao incluir a cadeia identificadora ID_j no primeiro passo do protocolo, garantindo assim que o protocolo prossiga apenas se o ID_j enviado por i coincidir com a cadeia identificadora de j, falhando já no passo 2 em caso contrário. Um esquema ilustrativo pode ser observado na Figura 7.

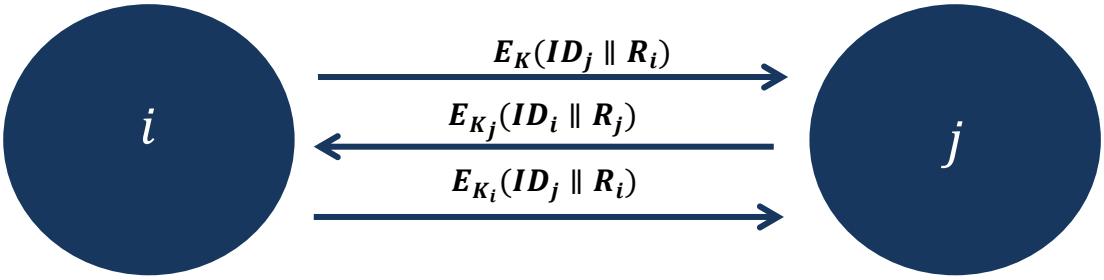

Figura 7 - Diagrama de exemplificação do protocolo Lim-Lee melhorado

Especificação do protocolo aprimorado

Notação:

- $K \in \{0,1\}^k$ é uma chave simétrica de k bits compartilhadas pelos usuários.
- $E: \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ é uma cifra de bloco, com chaves de k bits e blocos de n bits.

- $R_j \xleftarrow{\$} \{0,1\}^l$ denota a amostragem uniformemente aleatória de uma cadeia binária R_j de l bits.
- $ID_i \in \{0,1\}^m$ é um identificador, unívoco e de m bits, do usuário i.

Protocolo:

1. O usuário i escolhe $R_i \xleftarrow{\$} \{0,1\}^l$ e envia o criptograma $E_K(ID_j \parallel R_i)$ para o usuário j.
2. O usuário j recupera ID_j e R_i do criptograma acima, e verifica o identificador ID_j. Se essa verificação for bem sucedida, o usuário j calcula $K_j \leftarrow K \oplus (0^{k-l} \parallel R_i)$, escolhe $R_j \xleftarrow{\$} \{0,1\}^l$, e envia o criptograma $E_{K_j}(ID_i \parallel R_j)$ para o usuário i.
3. O usuário i verifica o identificador ID_i recuperado do criptograma acima. Se essa verificação for bem sucedida, o usuário i calcula $K_i \leftarrow K \oplus \left(0^{k-l} \parallel R_j\right)$, e envia o criptograma $E_{K_i}(ID_j \parallel R_i)$ para o usuário j.
4. O usuário j recupera ID_j e R_i do criptograma acima, e verifica se eles coincidem com os valores correspondentes recuperados no passo 2.

As relações entre os tamanhos devem ser $m + l = n$, de modo que $ID_j \parallel R_i \in \{0,1\}^n$ (e analogamente permutando i e j), e $l \leq k$ para possibilitar o mascaramento da chave K nas formas K_i e K_j. A concatenação de zeros $(0^{k-l} \parallel R_j)$ no cálculo das chaves K_i e K_j é realizada para que os operandos da operação ou-exclusivo sejam de mesmo tamanho.

No caso da aplicação desse protocolo no projeto, como o pacote de transferência de dados já é definido com um tamanho de 16 bytes, é possível utilizar uma cifra de no máximo 16 bytes, como por exemplo AES128 e Speck128 (com uma cifra de bloco e chave de 16 bytes) ou Curupira 2 (com uma cifra de bloco e chave de 12 bytes). Como o endereço (identificador) de cada unidade-móvel e da base já é definido na aplicação com um tamanho de 4 bytes, a cadeia aleatória deveria ter um tamanho de 12 bytes caso o

algoritmo utilizado fosse AES128 ou Speck128 e 8 bytes caso o algoritmo utilizado fosse o Curupira 2. Como parte da mensagem cifrada enviada contém o identificador do usuário, a segurança do protocolo depende do tamanho da cadeia aleatória que é concatenada ao identificador. Com isso, ao utilizar o Curupira 2 como cifra, a segurança do protocolo seria de 64 bits (8 bytes) e ao utilizar uma cifra de bloco de tamanho 16 bytes (AES128 ou Speck128), a segurança do protocolo seria de 96 bits (12 bytes).

Desenvolvimento do protocolo

Optou-se por utilizar uma cifra de 128 bits (AES ou Speck) pelo fato de sua segurança ser maior que uma cifra de 96 bits (Curupira 2). Primeiramente, o protocolo foi desenvolvido em linguagem C para ser testado inteiramente no microchip PIC16LF1829, chip da base do kit BodyCom™, e só após a certeza de viabilidade do protocolo, suas etapas foram divididas entre os chips PIC16LF1829 (base) e PIC16LF1827 (unidade-móvel) para que o protocolo fosse testado junto com a funcionalidade de transmissão pela pele humana do kit BodyCom™.

Primeiramente, foi necessário obter um código do algoritmo de criptografia, AES ou Speck, desenvolvido para rodar em plataformas de 8 bits e pequeno o suficiente para caber nas limitações de memória do kit. Foram encontradas diversas implementações do algoritmo AES128, mas todas demonstraram problemas ao serem executadas no chip do kit BodyCom™: ou o código continha apenas a funcionalidade de encriptação, mas não a de decriptação, ou o código mostrou-se muito grande para o chip, ou o código não executava com sucesso na aplicação. Por isso, seria necessário codificar a própria implementação do algoritmo AES128 para utilizá-lo.

Ao invés disso, optou-se por utilizar a cifra Speck, um algoritmo de criptografia desenvolvido pela NSA (National Security Agency) em 2013 especificamente para aplicações leves e para desempenho ótimo de software e hardware em microcontroladores (BEAULIEU, et al., 2013). Apesar de ser uma cifra muito recente, Speck já passou por algumas análises e Itai Dinur realizou diversos

ataques à cifra em seu trabalho *Improved Differential Cryptanalysis of Round-Reduced Speck* (DINUR, 2014) e a cifra não mostrou nenhuma vulnerabilidade até o momento. Beaulieu et al. obtiveram resultados muito bons de desempenho e ocupação na comparação do Speck128 com o AES128 em uma plataforma de 8 bits, como mostra a Figura 8 (retirada do artigo original (BEAULIEU, et al., 2013)). O código do Speck mostrou-se cerca de 52% menor em ocupação de memória flash que o AES e com uma taxa de transferência (throughput) 42% maior, e por esse motivo optou-se primeiramente pela implementação do algoritmo Speck em linguagem C para plataformas de 8 bits ao invés do algoritmo AES.

		hardware		software		
size	name	area (GE)	throughput (kbps)	flash (bytes)	SRAM (bytes)	throughput (kbps)
128/128	SIMON	1317	22.9	732	0	342
	SPECK	1396	12.1	396	0	768
	AES	2400	56.6	943	33	445

Figura 8 - Comparação de desempenho entre Simon, Speck e AES com tamanhos de chave e bloco de 128 bits.

Implementação da cifra Speck

A cifra Speck é baseada em rodadas de cálculos e o número de rodadas para a realização da cifração é dependente dos tamanhos do bloco e da chave de criptografia, de acordo com a Tabela 5.

Tabela 5 - Relação do tamanho de bloco, tamanho de chave e número de rodadas da cifra Speck

Tamanho do Bloco (bits)	Tamanho da Chave (bits)	Número de rodadas
32	64	22
48	72	22
	96	23
64	96	26
	128	27
96	96	28
	144	29
128	128	32
	192	33
	256	34

As operações utilizadas em cada rodada de cálculo da cifra são:

- OU-EXCLUSIVO (XOR) bit a bit, representado por $\oplus$.
- Adição em módulo 2^n, sendo n a metade do tamanho do bloco, representada por $+$.
- Rotações bit a bit para a esquerda e para a direita, representadas por S^j e S^{-j}, respectivamente, para j bits.

Na cifração do Speck, o bloco recebido é divido em duas partes x e y, sendo x a segunda parte do bloco e y a primeira parte, e as operações são realizadas entre essas partes e a chave k de acordo com os algoritmos de cifração abaixo, sendo R_k correspondente à encriptação e R_k^{-1} correspondente à decriptação (que utiliza a subtração modular ao invés da adição).

$$R_k(x,y) = \left((S^{-\alpha}x + y)\oplus k, S^{\beta}y \oplus (S^{-\alpha}x + y)\oplus k\right)$$

$$R_k^{-1}(x,y) = \left(S^{\alpha}\left((x\oplus k) - S^{-\beta}(x\oplus y)\right), S^{-\beta}(x\oplus y)\right)$$

As quantidades de rotações são $\alpha = 7$ e $\beta = 2$ se o tamanho do bloco for 32 bites e $\alpha = 8$ e $\beta = 3$ caso contrário.

Para que a implementação do código ficasse a mais compacta possível, em vista das limitações de hardware, foi descartado o bloco de 32 bits e também os casos em que o tamanho da chave é diferente do tamanho do bloco.

A cada rodada da cifra Speck o algoritmo de cifração é aplicado tanto à chave quanto à mensagem. No caso da chave, ela é dividida nas partes x e y e o número da rodada em questão é utilizado como k. No caso da mensagem, ela também é dividida em duas partes e a segunda parte da chave que acabou de ser atualizada é utilizada como k.

Beaulieu et al. realizaram uma implementação da cifra Speck em um microcontrolador AVR de 8 bits, mas a implementação foi realizada em linguagem assembly para a obtenção de desempenho ótimo (BEAULIEU, et al., 2014), então não foi possível utilizá-la. Song realizou uma implementação em linguagem C das cifras Simon e Speck e a disponibilizou como código aberto. Contudo, a implementação foi feita para plataformas de 32 bits e, além disso, a implementação de Speck só contava com a encriptação, faltando a decriptação. Portanto, esta também não pôde ser utilizada neste projeto.
Por esta razão, foi necessário realizar uma implementação própria da cifra Speck em linguagem C para uma plataforma de 8 bits, que também será disponibilizada como código aberto. A grande dificuldade da implementação deu-se em realizar as operações de adição e subtração modular maiores que 8 bits (módulo 64 no caso de um bloco de 128 bits) e as operações de rotação. Por fim, o código implementado é mais uma contribuição original do projeto e as ocupações de memória e tempos de execução podem ser vistas na Tabela 6.

Tabela 6 - Ocupações de memória e tempos de execução da implementação original da cifra Speck

	PIC16					
	Speck128 (otimização de espaço)			Speck128 (otimização de velocidade)		
	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)
Encrypt - 32MHz	117	557	0.0068	117	565	0.0062
Encrypt - 16MHz	117	557	0.0134	117	565	0.0122
Encrypt - 8MHz	117	557	0.0266	117	565	0.0243
Decrypt - 32MHz	117	736	0.0155	117	795	0.0130
Decrypt - 16MHz	117	736	0.0309	117	795	0.0258
Decrypt - 8MHz	117	736	0.0614	117	795	0.0509

Testes

Os testes do protocolo simétrico Lim-Lee com a cifra Speck no PIC 16 mostraram-se muito promissores, como pode ser observado na Tabela 7.

Tabela 7 - Resultados dos testes do protocolo Lim-Lee melhorado no PIC16

	PIC16					
	Lim-Lee (otimização de espaço)			Lim-Lee (otimização de velocidade)		
	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)
Speck128 – 32MHz	194	1683	0.0778	194	1826	0.0663
Speck128 – 8MHz	194	1683	0.3057	194	1826	0.2590

Pode-se observar que, para testes do protocolo completo na placa base do kit (PIC16LF1829), o protocolo simétrico Lim-Lee mostrou-se dentro de todas as limitações impostas pelo hardware do kit BodyCom™ nos três aspectos: 194 bytes de memória RAM dos 534 bytes disponíveis na base e dos 209 bytes disponíveis nas unidades-móvies, 1683 bytes de memória ROM dos 1351 bytes disponíveis na base e dos 2333 disponíveis nas unidades-móveis, e um tempo de execução de 0,07 segundos com um *clock* de 32MHz na plataforma, um

tempo pequeno o suficiente para as aplicações da tecnologia BodyCom™. No entanto, a aplicação *DEMO* fornecida funciona apenas com um *clock* de 8MHz, o que aumenta o tempo de execução significantemente para 0,26 segundos com otimização de velocidade. Mesmo assim, esse tempo de execução mostra-se viável para a maioria das aplicações possíveis com a tecnologia BodyCom™. Com isso, o protocolo Lim-Lee com a cifra Speck128 torna-se o principal candidato a protocolo do projeto.

5 DESENVOLVIMENTO DO PROJETO

O desenvolvimento do projeto consiste na inclusão do protocolo Lim-Lee melhorado, que foi o único protocolo estudado a se mostrar viável com as limitações impostas pelo kit de desenvolvimento da BodyCom™, na aplicação *DEMO* fornecida pela Microchip.

5.1 Reformulação da aplicação *DEMO*

Ao analisar o código-fonte da aplicação *DEMO*, observou-se que a aplicação não suporta o envio de múltiplos pacotes para uma única unidade-móvel e, por isso, foi necessário redesenhar a aplicação principal para que ela conseguisse suportar o protocolo Lim-Lee. O esquema de recepção e envio da aplicação reformulada desenvolvida para a placa base se encontra na Figura 9.

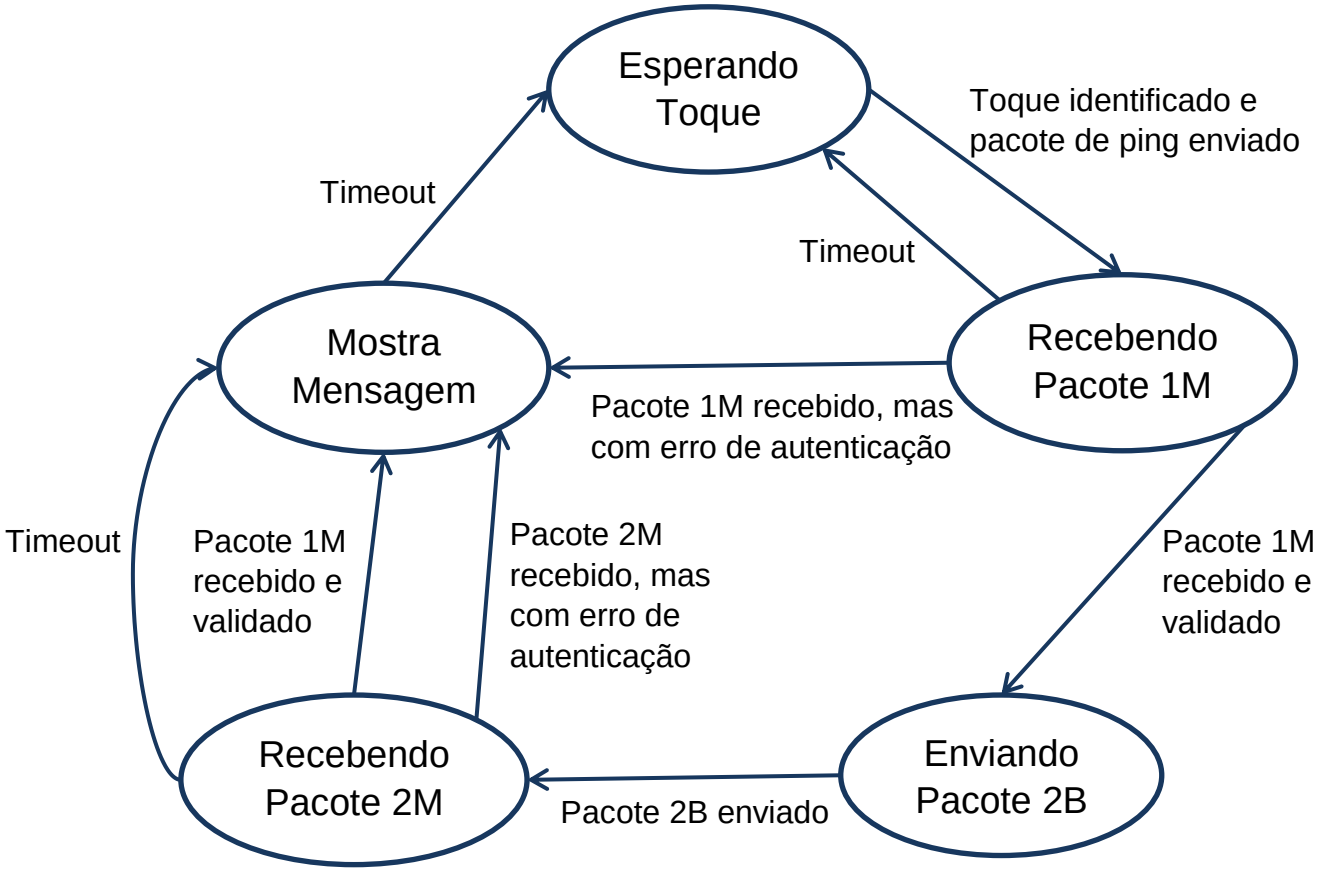

Figura 9 - Esquema principal da aplicação reformulada

O loop principal da aplicação reformulada chama infinitamente a função que implementa o esquema apresentado, dividido em 5 estados:

- **Esperando Toque:** Esse é o estado inicial da função principal e nele o programa checa se o sensor de toque foi pressionado. Em caso positivo, um pacote de *ping* é enviado para a primeira unidade-móvel registrada na placa base, um *timer* é estabelecido e o estado 2 é habilitado.
- **Recebendo Pacote 1M**: O segundo estado espera pelo recebimento do primeiro pacote da unidade-móvel que o estado 1 "*pingou*". Caso o pacote seja recebido e validado sem erros pelo processo de autenticação, o estado 3 é habilitado. Caso o pacote seja recebido, mas algum erro de autenticação seja encontrado, uma mensagem de erro é acionada e o estado 5 é habilitado, para que a mensagem seja mostrada. Caso nenhum pacote seja recebido dentro do tempo estabelecido, ocasionando um timeout, o estado 1 é habilitado novamente para que a próxima unidade-móvel cadastrada na placa base seja "*pingada*".
- **Enviando Pacote 2B:** O terceiro estado da aplicação reformulada apenas envia o segundo pacote da placa base à unidade-móvel, estabelece um *timer* para receber a resposta e habilita o estado 4. Esse estado é necessário, o segundo pacote não foi enviado logo após a recepção do pacote 1M, porque é necessário um tempo para que a unidade-móvel esteja pronta para receber um segundo pacote.
- **Recebendo Pacote 2M:** O quarto estado da aplicação espera pelo recebimento do segundo pacote vindo da unidade-móvel. Caso o pacote seja recebido e passe pela autenticação com sucesso, o acesso é garantido, uma mensagem de sucesso é acionada e o estado 5 é habilitado, para que a mensagem seja mostrada. Caso o pacote seja recebido, mas falhe na autenticação, uma mensagem de erro é acionada e o estado 5 é habilitado para mostrar a mensagem. Caso nenhum pacote seja recebido até o final do *timer*, uma mensagem de timeout é acionada e o estado 5 é habilitado para mostrar essa mensagem.

- **Mostra Mensagem:** O quinto e último estado apenas espera que o *timer* acionado no estado anterior seja atingido para que a mensagem seja mostrada por tempo suficiente e então habilita o estado inicial novamente para que a aplicação esteja pronta para o próximo acesso.

Além de realizar a reformulação da função principal da aplicação para que a integração com o protocolo de autenticação seguro seja possível, foi necessário corrigir alguns erros encontrados na aplicação de demonstração, como a contagem de timeout feita na interrupção de forma equivocada.

Também foi corrigido o erro da falta de verificação do endereço do pacote na aplicação *DEMO*. Como foi explicado anteriormente, a aplicação de demonstração não realiza a verificação do endereço do pacote enviado e do pacote recebido, que devem ser os mesmos, de forma a tornar a aplicação vulnerável a certos ataques. Portanto, na aplicação reformulada, todos os pacotes recebidos têm seu endereço verificado com o endereço do pacote anteriormente enviado e, caso não haja sucesso na verificação, o protocolo é terminado com um erro.

5.2 Geração de números pseudoaleatórios

O protocolo de Lim-Lee melhorado prevê a geração de números pseudoaleatórios para seu funcionamento. Como geradores de números pseudoaleatórios (*PRNG – Pseudo-Random Number Generator*) têm ocupação de memória e desempenho distintos dependendo de seu grau de segurança, assim como cifras, foi necessário realizar um estudo breve de escolha de um gerador que fosse viável junto às limitações e à segurança do projeto.

Foram considerados três *PRNG* para estudo: o gerador do compilador do kit de desenvolvimento XC8, um gerador desenvolvido por Robert Jenkins (JENKINS, [2009?]) e um gerador utilizando a própria cifra Speck implementada. A Tabela **8** mostra os testes de ocupação de memória e desempenho dos geradores de números pseudoaleatórios considerados.

Tabela 8 - Testes dos geradores de números pseudoaleatórios estudados

	PIC16LF1829 (base)					
	PRNG (otimização de espaço)			PRNG (otimização de velocidade)		
	RAM Size (Bytes)	ROM Size (Bytes)	Time (ms)	RAM Size (Bytes)	ROM Size (Bytes)	Time (ms)
Jenkins ranval()	48	560	2.2324	48	643	2.0713
XC8 rand()	23	293	2.8273	23	304	2.7606
Speck128	-	-	26.6100	-	-	24.2500

Os resultados dos testes realizados com os geradores consideram a geração de um número de 12 bytes, que é necessário no protocolo Lim-Lee.

O gerador do compilador XC8 foi considerado devido à sua baixa ocupação de memória, um fator determinante para o projeto, devido às limitações do kit de desenvolvimento. No entanto, não é possível saber se esse gerador passou em algum teste estatístico de segurança, então sua utilização não é aconselhável.

O gerador de Jenkins passou no teste DIEHARD, um teste razoável para fins criptográficos, porém não há indícios de que tenha passado no teste NIST, que é o teste determinante para garantir segurança criptográfica a um gerador de números pseudoaleatórios. No entanto, é o gerador de desempenho mais rápido encontrado e, caso a transmissão de dados seja muito lenta, esse gerador é uma opção a ser considerada.

O terceiro gerador analisado utiliza a cifra Speck para gerar um número aleatório, utilizando como chave de criptografia a semente a ser passada e como mensagem a ser encriptada, um contador. Ao utilizar uma cifra, esse gerador torna-se criptograficamente seguro e como a cifra já é utilizada no protocolo de Lim-Lee, sua ocupação de memória não precisa ser considerada. No entanto, seu tempo de execução é muito maior que o dos outros geradores, portanto, se esse é um fator crítico do projeto, talvez não seja interessante considerá-lo.

Em suma, se a prioridade do projeto for segurança, é aconselhável utilizar o gerador por Speck e, se a prioridade for desempenho, é aconselhável utilizar o gerador de Jenkins. Nesse projeto, o gerador a ser utilizado será determinado após os testes da aplicação, mas de qualquer forma, a semente a ser utilizada no gerador será um *timer* de 16 bits das placas, a ser iniciado junto à inicialização de hardware, e o gerador será iniciado assim que o primeiro toque na placa base ocorrer ou assim que a unidade-móvel acordar pela primeira vez, de forma a ter a semente mais aleatória possível.

5.3 Integração do protocolo de segurança com a aplicação

Ao realizar a integração do protocolo de Lim-Lee melhorado com a aplicação reformulada, mostrada na Figura 10, problemas foram encontrados. A placa base enviava o primeiro pacote à unidade-móvel, a unidade-móvel recebia o pacote e enviava seu primeiro pacote, porém a placa base não o recebia. Após uma análise superficial, foi observado que para atrasos maiores que 15 milissegundos, ou a unidade-móvel não consegue enviar o pacote ou a base não consegue recebê-lo. Já o oposto não ocorre, a placa base não tem problemas para enviar um pacote com qualquer atraso e a unidade-móvel consegue recebê-lo.

Esse erro na aplicação *DEMO* dificulta a introdução de um protocolo de autenticação pelo fato da criptografia necessitar de certo tempo para ser calculada. O projeto escolheu utilizar a cifra mais leve e rápida encontrada e, mesmo assim, uma encriptação de Speck128 no ambiente de desenvolvimento leva 25ms para ser calculada e uma decriptação leva cerca de 50ms, ambas gerando atrasos muito maiores do que a aplicação suporta.

Uma solução seria aumentar o *clock* das placas para 32MHz. No entanto, o propósito da unidade-móvel é ser um dispositivo pequeno e que tenha uma bateria duradoura e o aumento da velocidade do *clock* aumentaria muito o consumo de bateria, inviabilizando essa solução.

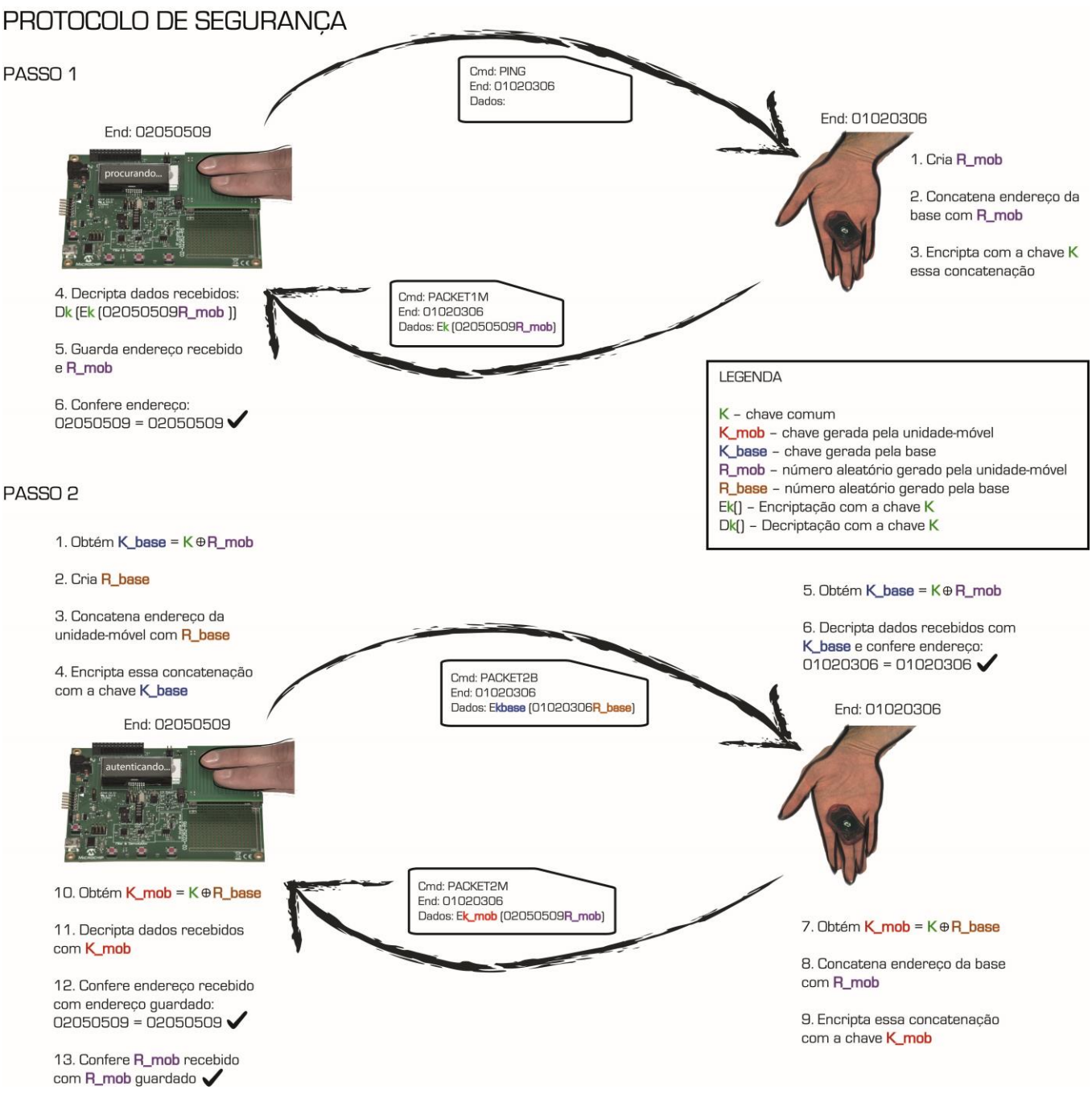

Figura 10 - Protocolo de segurança Lim-Lee melhorado integrado à aplicação BodyCom

Uma solução viável para o problema, porém não ideal, seria a redução do número de rodadas da cifra Speck de forma que a redução no tempo de cifração se torne aceitável para a aplicação. Essa solução ocasiona em uma redução significativa na segurança disponibilizada pela cifra Speck, pois a segurança está ligada diretamente ao número de rodadas e a todos os testes pelos quais a cifra foi submetida e passou. O número de rodadas estabelecido pelos criadores, no caso da Speck128, foi de 32 rodadas, mas para que as

rotinas de encriptação e decriptação da Speck128 atinjam tempos de execução aceitáveis para que a aplicação seja executada sem erro, o número de rodadas deve ser reduzido de 32 para 8.

Essa solução viável pode ser realizada para efeito de demonstração da aplicação funcionando no kit de desenvolvimento, mas a solução ideal para o problema seria analisar a fundo e em baixo nível as funções que realizam a comunicação entre a placa base e as unidades-móveis para encontrar o erro e corrigí-lo, de forma que o protocolo atinja sua segurança desejada.

5.4 Testes

5.4.1 Testes de desempenho

Para realizar os testes de desempenho foi escolhido utilizar um *timer* de 16 bits da placa base rodando com um *clock* de 256kHz e, para controlar o *overflow* do *timer*, foi utilizado um contador que era incrementado a cada interrupção de *overflow*. Dessa forma, os tempos de execução de cada parte do protocolo puderam ser medidos com maior precisão.

Para contornar o erro da aplicação com o atraso na montagem do pacote na unidade-móvel, o protocolo foi montado de forma particular. O protocolo foi montado completo do lado da placa base, com todos os passos do protocolo Lim-Lee e utilizando a cifra Speck128 com os 32 *rounds* pressupostos pela NSA. Como o erro ocorre do lado da unidade-móvel, para realizar os testes de desempenho, optou-se por substituir as rotinas que causam o atraso (encriptação e decriptação do Speck) por blocos fixos esperados, e no tempo de execução total foram somados os tempos individuais dessas rotinas. Dessa forma, foi possível prever o tempo de execução da aplicação reformulada por completo. Nesse caso, o gerador de números pseudoaleatórios escolhido foi o gerador por Speck, devido a uma segurança maior.

Outro teste de desempenho realizado foi com a aplicação reformulada reduzida, utilizando um número de rodadas da cifra Speck, de forma a não

ocasionar um atraso maior que 15 milissegundos na unidade-móvel e não acarretar em um erro na aplicação. Para esse teste foi necessário utilizar 8 *rounds* na cifra Speck e foi escolhido utilizar o gerador de números pseudoaleatórios Jenkins (JENKINS, [2009?]), por ser um gerador muito mais rápido e ter uma segurança razoável. Nesse teste é possível observar o desempenho da aplicação reformulada com segurança de ponta a ponta, apesar de a segurança ser diminuída pelo número de *rounds* reduzido.

A Tabela 9 mostra os resultados dos testes descritos acima e o teste de desempenho da aplicação original para comparação.

Tabela 9 - Testes de desempenho das aplicações BodyCom™

	Teste Desempenho (otimização de velocidade)			
	TOTAL (ms)	Manipulação dos Dados		Transmissão e Recepção (ms)
		Unidade-Móvel (ms)	Base (ms)	
Aplicação Reformulada Completa	1204.00	130.40	191.00	882.60
Aplicação Reformulada Reduzida	967.80	21.20	61.40	885.20
Aplicação Original	212.17	0.00	0.00	212.17

Os resultados apresentados na Tabela 9 mostram as médias de todos os resultados realizados. Pode-se observar que o tempo de execução da manipulação de dados da aplicação reformulada completa, 320ms, aproxima-se do tempo de execução do teste do protocolo Lim-Lee melhorado, 259ms, o que era esperado, pois o protocolo de Lim-Lee melhorado considera apenas a execução do protocolo sem a transmissão dos dados, o mesmo que a manipulação dos dados considera. O tempo da aplicação reformulada completa é um pouco maior, pois inclui o tempo de execução de geração do número pseudoaleatório, que seria da ordem de 50ms para ambos os lados, e ainda um tempo a mais de processamento das instruções que validam o comando e o endereço dos pacotes.

Ao analisar os tempos de execução da manipulação de dados para a aplicação reformulada reduzida, percebe-se que eles são bem menores do que os tempos da aplicação completa. Isso é esperado, devido à aplicação reduzida utilizar uma cifra Speck com quatro vezes menos rodadas e por utilizar um gerador de números pseudoaleatórios bem mais rápido.

Os tempos de execução médios das aplicações reformuladas mostraram-se maiores do que o esperado, pois tempos da ordem de 1 segundo são muito grandes para algumas aplicações, como é o caso da aplicação em uma arma policial que necessita de um tempo de reação muito baixo. No entanto, nota-se que o tempo de transmissão e recepção não diminui entre as aplicações reformuladas e faz parte 75% do tempo total de execução na aplicação completa e mais de 90% na aplicação reduzida, e é, portanto, o grande responsável pelo tempo da ordem de 1 segundo das aplicações.

A aplicação original não realiza nenhuma manipulação de dados pelo fato de não transmitir nenhum pacote com dados entre os dispositivos e, portanto, seu tempo total de execução é praticamente todo realizado pela transmissão e recepção. Ainda assim, transmitindo e recebendo apenas dois pacotes sem dados (um da base e outro da unidade-móvel), o tempo de execução é de 212ms, então os tempos de 880ms das aplicações reformuladas são esperados, já que elas enviam 1 pacote sem dados e 3 pacotes com 16 bytes de dados.

Em suma, o protocolo de segurança incluído junto à aplicação é executado em um tempo pequeno o suficiente para a maioria das aplicações de controle de acesso. Porém, a aplicação de demonstração realiza as transmissões e recepções de pacotes muito devagar e isso diminui o leque de aplicações que o protocolo de autenticação seguro proporciona.

5.4.2 Análise do consumo de energia

O consumo de energia é de suma importância para a unidade-móvel. Se a aplicação reformulada não realizar um consumo de energia controlado, a bateria da unidade-móvel acabará rapidamente e terá que ser trocada com

uma frequência inconveniente para o usuário. Dessa forma, foi necessário realizar a análise do consumo de energia da unidade-móvel com a nova aplicação para garantir que a utilização do protocolo de autenticação seguro seja viável com a tecnologia BodyCom™.

O consumo típico de energia da unidade-móvel pode ser observado na Figura 11, que foi retirada do documento de introdução à tecnologia BodyCom™ (BAILEY, 2014). Nela, observa-se que, durante o período de repouso, a unidade-móvel consome apenas 3μA de energia e isso permite que sua bateria dure um longo tempo, pois ela passa seu maior tempo de vida dormindo. Também é possível observar que, durante o período em que a unidade-móvel está acordada, são consumidos 27mA durante a transmissão de dados e 1.3mA durante o resto o tempo.

	Wait for Signal	**Receive Data**	**Decode Data**	**Send Answer**
AS3933	Wait for signal (2 μA)	Decode data (10 μA)	Wait for signal (2 μA)	
PIC18LF1827	Sleep (<1 μA)	PIC® MCU clock: 8 MHz (1.3 mA)	PIC® MCU clock: 8 MHz (1.3 mA)	
Tx Driver	OFF (0.1 μA)			ON (25 mA)
Time (for 1000 baud rate)	NA	5 ms + (1 ms * Nbytes)	< 1 ms	16 ms + (1 ms * Nbytes)
Power Consumption	3 μA	1.3 mA	1.3 mA	27 mA

Figura 11 - Consumo típico de energia da unidade-móvel.

Sendo assim, foi calculado que, para a nova aplicação reformulada, a unidade-móvel fica acordada por uma média de 1560 milissegundos. Desse tempo, 490ms são gastos com a transmissão de pacotes (245ms para cada pacote) e os restantes 1070ms são gastos com o recebimento e a manipulação de dados. Portanto, para uma transação típica da aplicação reformulada, a unidade-móvel consome 0,004mAh.

De acordo com a *datasheet* (HITACHI MAXWELL, LTD., 2008) da bateria utilizada pela unidade-móvel, CR2032, para uma tensão de até 2,0V a bateria aguenta um consumo de 220mAh. No entanto, foi constatado que a unidade-móvel para de transmitir pacotes com uma tensão de 2,9V, portanto seu consumo máximo seria em torno de 151mAh. Supondo um consumo médio de

4 transações por dia, seria uma média de 120 transações e um consumo de 0,48mAh por mês vindo das transações. Já o consumo em estado de repouso seriam de 0,003mA por 720 horas de um mês, ou seja, de 2,16mAh por mês. Isso totaliza um consumo de 2,64mAh por mês da unidade-móvel e significa que, considerando um consumo médio de 4 transações por dia da aplicação, a bateria da unidade-móvel duraria por 57 meses, ou seja, mais de 4 anos e meio. Supondo um consumo mais intenso de 10 transações diárias, o consumo não diminuiria tanto, pois a maior parte dele ainda seria consumida pela unidade-móvel em estado de repouso, e a bateria duraria 44 meses, ou seja, um pouco mais de 3 anos e meio.

Portanto, o consumo de energia na unidade-móvel continua aceitável para a aplicação reformulada e, mesmo para aplicações de uso mais intenso, não ocasionaria um período de troca de bateria inconveniente para o usuário.

5.4.3 Investigação do problema com a aplicação

A aplicação foi analisada mais a fundo com o objetivo de encontrar o motivo do problema encontrado na aplicação, em que um atraso maior que 15 milissegundos na criação do pacote na unidade-móvel impede o recebimento pela base.

Para realizar essa análise, foi construída uma rotina de rastreamento em todos os passos das rotinas de transmissão e recepção de pacote da aplicação. Dessa forma, um *buffer* era preenchido com diferentes caracteres dependendo do estado da transmissão e recepção que era executado. Assim que a transmissão ou a recepção terminavam, o *buffer* era exibido na tela LCD da placa base para que fossem avaliados se todos os estados esperados em uma transmissão ou recepção com sucesso eram executados em um cenário de erro. Para analisar a transmissão e a recepção na unidade-móvel, a mesma função de rastreamento foi utilizada, porém, o *buffer* foi enviado no campo de dados do pacote. Dessa forma, utilizando a comunicação serial EUSART disponível, foi possível visualizar o conteúdo do pacote em um computador.

No entanto, não foi possível identificar uma anormalidade nos passos da transmissão e recepção de nenhum dos dispositivos e, por esta razão, não foi possível identificar a causa do erro na aplicação.

Apesar disso, ao analisar mais a fundo a aplicação que faz possível a tecnologia BodyCom™, foi possível identificar alguns erros de programação de software cometidos:

- Rotina que controle o toque trata o *timeout* de forma equivocada, sendo necessária ser chamada inúmeras vezes para que um *timeout* seja acionado, além do cálculo de *jitter* também ser feito de forma equivocada.
- A tecnologia de transmissão utiliza um decodificador Manchester, mas a transmissão e a recepção são realizadas inteiramente por software bit a bit. Dessa forma, a chance da transmissão falhar aumenta, visto que o mesmo software contém outros erros e de fato falha, como o erro do atraso no pacote confirma.

Em suma, essa análise mostra que a tecnologia BodyCom™ ainda está em evolução, porque há alguns erros a serem corrigidos. Além disso, talvez a migração de uma transmissão feita por hardware ao invés de software seja uma opção para que o tempo gasto nas transmissões e recepções de pacotes diminua. Ainda assim, a tecnologia mostra um grande potencial para diversas aplicações mesmo com alguns erros e quando esses erros forem corrigidos o leque de opções aumentará ainda mais.

6 CONSIDERAÇÕES FINAIS

A tecnologia BodyCom™ é uma nova tecnologia, criada em 2013 pela Microchip, que possibilita uma abordagem diferenciada a aplicações de controle de acesso. No entanto, aplicações de controle de acesso precisam ser seguras e, ao analisar o kit de desenvolvimento que a Microchip disponibiliza, foi constatado que, apesar da tecnologia utilizar o corpo humano como meio de transmissão e este ser um meio muito seguro, a aplicação de demonstração não realizava nenhum protocolo de autenticação seguro e, portanto, estava exposta a ataques maliciosos. O projeto buscou explorar exatamente essas vulnerabilidades da aplicação de demonstração e propôs um protocolo de autenticação seguro para que a aplicação se tornasse viável a um ramo aind maior de aplicações.

Após verificar a inviabilidade de protocolos assimétricos serem utilizados, o projeto encontrou um protocolo simétrico que coubesse dentro das limitações impostas pelo kit de desenvolvimento, que possui capacidade de memória e processamento muito limitados. O projeto propôs melhorias originais ao protocolo simétrico escolhido, de forma a torná-lo mais adequado para aplicações muito leves, e ainda desenvolveu uma implementação original em linguagem C da cifra Speck, uma cifra desenvolvida em 2013 para performance ótima em plataformas pequenas.

Os testes na aplicação reformulada desenvolvida mostraram que um erro na implementação da tecnologia existe e que limita muito qualquer protocolo de segurança a ser associado a ela. Também, foi constatado que a implementação da tecnologia realiza a transmissão de pacotes inteiramente por software e provavelmente é isso que acarreta o grande tempo gasto durante essas transmissões.

Apesar de todas as complicações e limitações da tecnologia, foi possível desenvolver um protocolo seguro viável tanto em termos de desempenho quanto em termos de consumo de bateria por parte da unidade-móvel para

diversas aplicações. Além disso, pelo fato da tecnologia ser muito recente e ainda estar em evolução, seus erros devem ser corrigidos e seu hardware deve ser melhorado, e quando forem, o potencial da tecnologia crescerá muito e o protocolo desenvolvido neste projeto poderá ser utilizado para aumentá-lo ainda mais.

REFERÊNCIAS BIBLIOGRÁFICAS

BAILEY, B., 2014. AN1391: Introduction to the BodyCom Technology. *Microchip Technology Inc..*

BARITAUD, T., CAMPANA, M., CHAUVAUD, P. & GILBERT, H., 1993. On the Security of Permuted Kernel Identification Scheme. *Centre National d'Etudes des Télécommunications.*

BEAULIEU, R. et al., 2013. The Simon and Speck Families of Lightweight Block Ciphers. *National Security Agency, USA.*

BEAULIEU, R. et al., 2014. The Simon and Speck Block Ciphers on AVR 8-bit Microcontrollers. *National Security Agency, USA.*

DINUR, I., 2014. Improved Differential Cryptanalysis of Round-Reduced Speck. *Département d'Informatique, École Normale Supérieure.*

GOMEZ, K., 2013. *BodyCom Technology is world's first to use huma body as a low-power communication channel.* [Online]
Disponível em: http://www.pacetoday.com.au/news/bodycom-technology-is-world-s-first-to-use-human-b
[Acesso em 25 Outubro 2015].

HITACHI MAXWELL, LTD., 2008. *Maxwell Lithium Manganese Dioxide Battery CR2032 Datasheet.* s.l.:s.n.
JENKINS, R., [2009?]. *Bob Jenkin's Web Site.* [Online]
Disponível em: http://burtleburtle.net/bob/rand/smallprng.html
[Acesso em 24 Novembro 2015].

LIM, C. & LEE, P., 1995. Several practical protocols for authentication and key exchange. *Department of Electrical Engineering, Pohang University of Science and Technology.*

METZ, R., 2013. *Authentication System Would Use the Body to Secure Guns and Gadgets.* [Online]
Disponível em: http://www.technologyreview.com/news/512056/authentication-system-would-use-the-body-to-secure-guns-and-gadgets/
[Acesso em 25 Outubro 2015].

MICROCHIP TECHNOLOGY INC., 2013. *BodyCom Technology.* [Online]
Disponível em: http://www.microchip.com/pagehandler/en-us/technology/embeddedsecurity/technology/bodycom.html
[Acesso em 25 Outubro 2015].

MONTEIRO, F. S., 2012. Protocolo de Identificação baseado em Polinômios Multivariáveis Quadráticos. *Dissertação de Mestrado - Instituto de Matemática e Estatística, Universidade de São Paulo.*

POUPARD, G., 1997. A Realistic Security Analysis of Identification Schemes based on Combinatorial Problems. *Ecole Normale Superieure, Laboratoire d'Informatique.*

RILEY, M., 2013. *BodyCom Development Kit.* [Online]
Disponível em: http://www.drdobbs.com/security/bodycom-development-kit/240153458
[Acesso em 2015 Outubro 2015].

SHAMIR, A., 1989. An Efficient Identification Scheme Based on Permuted Kernels. *Applied Mathematics Department - The Weizmann Institute of Science.*
SONG, G., 2014. *Simon & Speck block cipher implementation open source code in C.* [Online]
Disponível em: https://github.com/GSongHashrate/SimonSpeck
[Acesso em 25 Outubro 2015].

APÊNDICES

A TESTES DO PROTOCOLO PKP

Testes das diferentes versões do algoritmo de Problema de Núcleos Permutados (PKP).

A.1 Versão de 20/05/2015

Tabela 10 - Resultados dos testes do protocolo PKP v20.05.15 no PIC16

	PIC16					
	PKP v20.05.15 (otimização de espaço)			PKP v20.05.15 (otimização de velocidade)		
	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)
Keccak	678	3313	35.3900	678	3731	30.3900

Tabela 11 - Resultados dos testes do protocolo PKP v20.05.15 no PIC24

	PIC24					
	PKP v20.05.15 (otimização de espaço)			PKP v20.05.15 (otimização de velocidade)		
	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)
Keccak	526	2895	2.2800	526	2955	2.1980

A.2 Versão de 24/05/2015

Tabela 12 - Resultados dos testes do protocolo PKP v24.05.15 no PIC16

	PIC16					
	PKP v24.05.15 (otimização de espaço)			PKP v24.05.15 (otimização de velocidade)		
	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)
Keccak	674	3294	34.6400	674	3708	29.7850
AES	487	3220	6.2040	487	3776	5.1410
Speck 128	402	2161	5.1340	402	2431	4.2460
Speck 96	390	2111	3.9390	390	2334	3.4510
Curupira2 (s/ tabelas)	396	2311	3.7400	396	2615	3.2880
Curupira2 (c/ TABS)	391	2484	2.2560	391	2794	1.9510
Curupira2 (c/ TABS, TAB0)	391	2509	2.2480	391	2891	1.9250
Curupira2 (c/ TABS, TAB0, TABX)	387	2748	2.2180	387	3052	1.8520

Tabela 13 - Resultados dos testes do protocolo PKP v24.05.15 no PIC24

	PIC24					
	PKP v24.05.15 (otimização de espaço)			PKP v24.05.15 (otimização de velocidade)		
	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)
Keccak	580	2706	2.2640	580	2784	2.2030
AES	410	2796	0.5580	410	2898	0.5470
Speck 128	378	2133	0.6243	378	2208	0.6185
Speck 96	366	2079	0.5451	366	2151	0.5358
Curupira2 (s/ tabelas)	366	2427	0.9016	366	2580	0.9038
Curupira2 (c/ TABS)	366	2631	0.3952	366	2760	0.3890
Curupira2 (c/ TABS, TAB0)	366	2679	0.3963	366	2808	0.3891
Curupira2 (c/ TABS, TAB0, TABX)	366	3030	0.3825	366	3168	0.3883

A.3 Versão de 25/05/2015

Tabela 14 - Resultados dos testes do protocolo PKP v25.05.15 no PIC16

	PIC16					
	PKP v25.05.15 (otimização de espaço)			PKP v25.05.15 (otimização de velocidade)		
	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)
Keccak	817	3403	23.2190	817	3825	19.9280
AES	627	3265	4.7260	627	3812	3.9720
Speck 128	544	2211	3.6740	544	2441	3.2370
Speck 96	532	2175	3.1520	532	2389	2.8000
Speck 80	527	2178	2.8530	527	2400	2.5510
Curupira2 (s/ tabelas)	538	2424	3.0580	538	2728	2.7270
Curupira2 (c/ TABS)	533	2592	1.9750	533	2903	1.7230
Curupira2 (c/ TABS, TAB0)	533	2617	1.9650	533	2928	1.7180
Curupira2 (c/ TABS, TAB0, TABX)	529	2850	1.9360	529	3154	1.6520

Tabela 15 - Resultados dos testes do protocolo PKP v25.05.15 no PIC24

	PIC24					
	PKP v25.05.15 (otimização de espaço)			PKP v25.05.15 (otimização de velocidade)		
	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)
Keccak	660	2913	1.5230	660	2994	1.4380
AES	490	3009	0.3241	490	3111	0.3184
Speck 128	458	2346	0.3793	458	2421	0.3783
Speck 96	458	2292	0.3291	458	2364	0.3248
Speck 80	458	2274	0.2889	458	2346	0.2950
Curupira2 (s/ tabelas)	446	2640	0.5373	446	2793	0.5378
Curupira2 (c/ TABS)	446	2844	0.2324	446	2973	0.2371
Curupira2 (c/ TABS, TAB0)	446	2892	0.2424	446	3021	0.2375
Curupira2 (c/ TABS, TAB0, TABX)	446	3243	0.2345	446	3381	0.2304

A.4 Versão de 27/05/2015

Tabela 16 - Resultados dos testes do protocolo PKP v27.05.15 no PIC16

	PIC16					
	PKP v27.05.15 (otimização de espaço)			PKP v27.05.15 (otimização de velocidade)		
	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)
Keccak	702	3439	23.1000	702	3877	19.6000
AES	512	3283	4.0325	512	3848	3.3650
Speck 128	429	2244	3.3490	429	2503	2.7860
Speck 96	417	2208	2.5610	417	2451	2.2540
Speck 80	411	2177	2.3385	411	2400	2.0740
Curupira2 (s/ tabelas)	421	2385	2.3560	421	2694	2.1185
Curupira2 (c/ TABS)	416	2576	1.5485	416	2895	1.3165
Curupira2 (c/ TABS, TAB0)	416	2601	1.5390	416	2920	1.3025
Curupira2 (c/ TABS, TAB0, TABX)	412	2836	1.4695	412	3146	1.2450

Tabela 17 - Resultados dos testes do protocolo PKP v27.05.15 no PIC24

	PIC24					
	PKP v27.05.15 (otimização de espaço)			PKP v27.05.15 (otimização de velocidade)		
	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)
Keccak	550	3045	1.4200	550	3201	1.4169
AES	380	3168	0.3100	380	3303	0.3031
Speck 128	348	2493	0.3590	348	2613	0.3613
Speck 96	336	2439	0.3040	336	2556	0.3048
Speck 80	330	2421	0.2770	330	2538	0.2783
Curupira2 (s/ tabelas)	336	2787	0.5259	336	2985	0.5282
Curupira2 (c/ TABS)	336	2991	0.2190	336	3165	0.2155
Curupira2 (c/ TABS, TAB0)	336	3039	0.2170	336	3213	0.2160
Curupira2 (c/ TABS, TAB0, TABX)	336	3390	0.2116	336	3573	0.2086

A.5 Versão de 09/06/2015

Tabela 18 - Resultados dos testes do protocolo PKP v09.06.15 no PIC16

	PIC16					
	PKP v09.06.15 (otimização de espaço)			PKP v09.06.15 (otimização de velocidade)		
	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)
Keccak	709	3516	23.0900	709	3956	19.7040
AES	519	3354	4.1000	519	3927	3.4300
Speck 128	435	2316	3.4020	435	2587	2.8580
Speck 96	424	2270	2.6200	424	2509	2.3200
Speck 80	418	2253	2.4010	418	2483	2.1240
Curupira2 (s/ tabelas)	428	2464	2.4760	428	2772	2.2520
Curupira2 (c/ TABS)	423	2641	2.4690	423	2960	1.3830
Curupira2 (c/ TABS, TAB0)	423	2666	1.5780	423	2985	1.3680
Curupira2 (c/ TABS, TAB0, TABX)	419	2901	1.5410	419	3217	1.3120

Tabela 19 - Resultados dos testes do protocolo PKP v09.06.15 no PIC24

	PIC24					
	PKP v09.06.15 (otimização de espaço)			PKP v09.06.15 (otimização de velocidade)		
	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)	RAM Size (Bytes)	ROM Size (Bytes)	Time (s)
Keccak	550	3237	1.4720	550	3363	1.4280
AES	380	3333	0.2665	380	3480	0.2627
Speck 128	348	2670	0.3166	348	2790	0.3133
Speck 96	348	2616	0.2658	348	2733	0.2631
Speck 80	348	2598	0.2455	348	2715	0.2628
Curupira2 (s/ tabelas)	336	2964	0.4539	336	3162	0.4535
Curupira2 (c/ TABS)	336	3168	0.1893	336	3342	0.1866
Curupira2 (c/ TABS, TAB0)	336	3216	0.1893	336	3390	0.1852
Curupira2 (c/ TABS, TAB0, TABX)	336	3567	0.1826	336	3750	0.1794