FLAT: Um Protocolo de Autenticação Federada para a Internet das Coisas

Maria L. B. A. Santos[1], Jéssica C. Carneiro[1], Antônio M. R. Franco[1], Fernando A. Teixeira[2], Marco A. Henriques[3], Leonardo B. Oliveira[1]

[1]UFMG, Belo Horizonte. [2]UFSJ, Ouro Branco. [3]Unicamp, Campinas.

{mburga,jessicarneiro,franco}@dcc.ufmg.br

teixeira@ufsj.edu.br,marco@dca.fee.unicamp.br,leob@dcc.ufmg.br

Abstract. *The expansion of the Internet of Things (IoT) creates a great challenge regarding the authentication of devices, especially when one considers the restriction of computational resources and the potential mobility of these devices between different authentication domains. Our solution, FLAT, is a federated authentication protocol specially suited for IoT, using symmetric cryptographic primitives when communicating with the resource constrained IoT Client. We present a prototype of the protocol and scenarios where it can be applied, as well as an evaluation of the computational and communication costs, whose results showed a reduction of around 36% in total communication costs.*

Resumo. *Com o crescimento da Internet das Coisas (IoT), a autenticação de dispositivos passa a ser um grande desafio, principalmente quando se leva em consideração a restrição de recursos computacionais e seu potencial de mobilidade entre diferentes domínios de autenticação. Nossa solução, FLAT, é um protocolo de autenticação federada especialmente modelado para IoT, com o uso de primitivas criptográficas simétricas na comunicação com o Cliente IoT. Apresentamos um protótipo do protocolo e cenários de uso da solução, além de uma avaliação de seus custos computacionais e de comunicação, que mostram ser possível uma redução de cerca de 36% nos custos totais de comunicação.*

1. Introdução

A Internet das Coisas – *Internet of Things* (IoT) [Atzori et al. 2010] está impactando nossa sociedade significativamente. Um exemplo da presença da IoT em nosso dia-a-dia são os dispositivos vestíveis [Wei 2014]. Existem incontáveis possibilidades de adoção destes dispositivos, que vão desde óculos que mostram informações ao longo do dia, a relógios que exibem alertas recebidos no *smartphone* e dispositivos que medem batimentos cardíacos durante atividades físicas [Atzori et al. 2010]. Estudos mostram que em poucos anos estaremos rodeados por bilhões de dispositivos IoT[1].

Igualmente, a área de Gestão de Identidade – *Identity Management* (IdM) é primordial para o uso efetivo de tais dispositivos [Abomhara and Køien 2014]. Precisamente, IdM refere-se à identificação de usuários em um dado sistema (e.x. uma rede, aplicativo ou serviço) e ao controle de seu acesso aos recursos daquele sistema. (Neste caso, o termo *identificação* se refere ao processo de autenticação da identidade de um usuário em um

[1]http://www.gartner.com/newsroom/id/2636073

sistema [Menezes et al. 2001].) Idealmente, IdM provê aos administradores de sistemas ferramentas para gerenciar o ciclo de vida da identidade dos usuários (e.x. criação, manutenção e exclusão) e, portanto, é vital para a segurança e produtividade das organizações.

A Gestão de Identidade Federada – *Federated Identity Management* (FIdM), por sua vez, torna possível a um domínio controlar o acesso de usuários externos aos seus recursos [Shim et al. 2005, Wangham et al. 2013, Nogueira et al. 2011] e viabiliza o compartilhamento de informações de identidade entre diferentes domínios [Sasso et al. 2014]. Por exemplo, permite que um usuário externo se autentique em um servidor local e utilize seus serviços sem a necessidade de criar uma nova identidade ou registrar-se naquele domínio. Neste caso, o processo de autenticação entre o Cliente (consumidor de um serviço) e o Provedor de Serviço – *Service Provider* (SP) é mediado por um Provedor de Identidade – *Identity Provider* (IdP) do domínio de origem do usuário (Cliente). FIdM (i) permite às aplicações a adoção do *Single-Sign-On* (SSO), (ii) aumenta a privacidade dos usuários restringindo a quantidade de informação compartilhada com o SP e (iii) melhora a experiência e a segurança para o usuário através da eliminação do registro de novas contas e da limitação do número de senhas que o usuário precisa guardar.

Existem muitas situações em que nós IoT em domínios diferentes precisam se comunicar de forma segura. Uma situação do mundo real que nos chamou a atenção por meio de uma das maiores empresas de tecnologia é a seguinte: suponha que um caminhão autônomo (sem motorista) que transporta minério de ferro tenha ficado fora de serviço no meio de uma mina [2]. A equipe técnica para o resgate precisa prover, no local, manutenção emergencial para consertar o caminhão. Para isso, idealmente, a equipe de resgate precisa de aparatos técnicos para comunicar com o maquinário do caminhão e prover assistência (Fig. 1). Perceba que nesta situação os dispositivos (caminhão e equipamento da equipe de resgate) pertencem a domínios diferentes (à companhia de mineração e à companhia de resgate técnico, respectivamente) que precisam interoperar o mais rápido possível (o caminhão parado pode levar a companhia a perder milhões de dólares por dia) e de forma autenticada (pois um caminhão autônomo como este poderia custar milhões de dólares).

Problema. Abordagens de IdM e FIdM amplamente adotadas são inadequadas para a IoT [Fremantle et al. 2014]. Em primeiro lugar, não existe IdM para dispositivos IoT. Em esquemas (F)IdM existentes, dispositivos IoT fazem uso das credenciais de seus usuários (humanos) para acessar e utilizar os serviços de um domínio. Essa prática é insegura e inapropriada pois, respectivamente, dispositivos não deveriam ter o mesmo nível de permissão que os usuários e alguns dispositivos não podem simplesmente ser associados a um usuário individual (e.x. a que usuário um sinal de trânsito em uma rodovia deveria ser associado?). Em segundo lugar, a natureza móvel e dinâmica dos dispositivos IoT carece de um nível maior de escalabilidade e interoperabilidade entre vários domínios diferentes se comparado a elementos de rede convencionais [Fremantle et al. 2014]. E, por último, o processo de autenticação em esquemas (F)IdM existentes utiliza o paradigma de *login* e senha, que é inadequado para dispositivos, ou utilizam de criptossistemas caros como *Rivest-Shamir-Adelman* (RSA) ou *Digital Signature Algorithm* (DSA) e, portanto, requerem um esforço computacional maior [Koppula and Muthukuru 2016]. Sendo assim, há a necessidade de um FIdM capaz de atender às demandas especiais dos dispositivos IoT.

[2] https://www.technologyreview.com/s/603170/mining-24-hours-a-day-with-robots

Solução. Como uma solução para este problema, nós propomos um esquema FIdM exclusivamente projetado para IoT. Em resumo, nós modelamos, implementamos e avaliamos uma prova de conceito dos estágios centrais de um sistema de IdM para IoT. Nossa solução, chamada **F**ederated **L**ightweight **A**uthentication of **T**hings (FLAT), substitui protocolos assimétricos pesados usados em uma *Public Key Infrastructure* (PKI) tradicional por outros mais adequados à IoT. Notavelmente, o FLAT combina criptossistemas simétricos e certificados implícitos [Brown et al. 2001] em sua série de ferramentas.

Organização. O restante deste artigo está estruturado da seguinte maneira: a Seção 2 apresenta o projeto do protocolo de autenticação federada e um cenário de uso do FLAT. A Seção 3 discute a implementação da prova de conceito do FLAT. A Seção 4 avalia a segurança e o uso de recursos computacionais do FLAT. Finalmente, nós discutimos os trabalhos relacionados na seção Seção 5 e concluímos na Seção 6.

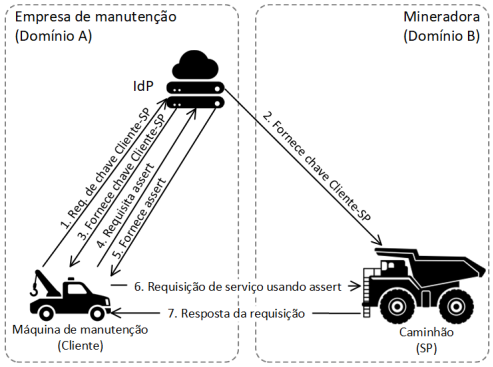


Figura 1: **Exemplo de uso de FIdM em uma mineradora**

2. Autenticação Federada para IoT

Nesta seção apresentamos nosso protocolo de autenticação federada para IoT, o FLAT. Descrevemos os processos e aspectos técnicos necessários ao funcionamento do protocolo: a etapa de pré-implantação, a descoberta de serviço, o estabelecimento de confiança entre as federações, formato do pacote e a especificação da operação do FLAT.

2.1. Modelo Federado Adotado e Premissas

Pré-implantação. O Cliente IoT gera uma chave secreta que é informada para o IdP através de um canal seguro no momento da configuração inicial do Cliente IoT. Para a geração da chave criptográfica do dispositivo assumimos o uso de *Physical Unclonable Function* (PUF), em que as características físicas de circuitos integrados podem ser utilizadas para derivação de chaves [Suh and Devadas 2007]. Uma resposta é gerada pelo PUF e passa por um processo de correção de erros. Em seguida, é aplicada uma função *hash* sobre a saída corrigida, sendo este *hash* utilizado como chave do dispositivo [Suh and Devadas 2007]. A chave produzida é transmitida do Cliente IoT para o IdP através de um canal seguro na fase de pré-implantação e, então, usada na comunicação entre eles.

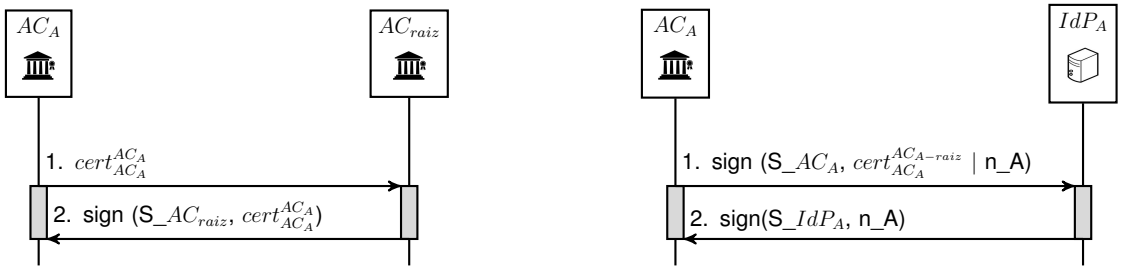
Descoberta de serviço. Em um cenário de IoT é razoável assumir que os dispositivos utilizem serviços fornecidos por outros dispositivos fisicamente próximos a eles [Ververidis and Polyzos 2008]. Em nossa proposta, assumimos que o Cliente IoT é capaz de descobrir os serviços fornecidos por dispositivos vizinhos, como por exemplo, pela recepção de

informações de serviços através de *beacons*, mensagens de *broadcast* enviadas periodicamente em um determinado raio de distância. Consideramos que, previamente à operação do FLAT, os SPs são capazes de enviar *beacons* informando periodicamente os serviços disponíveis no momento, bem como informações sobre seu *status* atual. Estas mensagens podem ser recebidas pelo Cliente IoT, que com os dados obtidos pode solicitar acesso ao serviço através da autenticação em seu IdP de origem.

Interfederação. Assumimos que as diferentes federações se comunicam utilizando um modelo baseado no eduGAIN [Lopez et al. 2006], serviço de interfederação que torna possível a conexão de federações de identidade acadêmicas e suas respectivas infraestruturas de autenticação e autorização. Para que uma relação de confiança seja estabelecida entre as federações, estas devem ter seus componentes registrados na *truststore*. Cada componente se comunica com a PKI, que adiciona a Autoridade Certificadora – *Certificate Authority* (AC) de cada federação, ou adiciona os IdPs e SPs da federação, obtendo certificados assinados pela AC da PKI da eduGAIN para cada um deles.

Tabela 1: **Nomenclatura utilizada na descrição do estabelecimento de confiança**

AC_{raiz}: AC raiz da PKI da interfederação	AC_x: AC da federação x
IdP_x: IdP da federação x	S_x: chave privada de x
n_x: *nonce* gerado por x	| : operação de concatenação
$cert_{AC_x}^{AC_{x-y}}$: certificado da AC_x assinado por AC_x e AC_y	



(a) Pré-operação 1 **(b) Pré-operação 2**

Figura 2: **Etapas de pré-operação**

Baseado no modelo de confiança da eduGAIN, assumimos que há uma PKI hierárquica com uma AC raiz e uma AC para cada uma das federações participantes, subordinadas à AC raiz. Esta AC raiz da PKI, assina os certificados das ACs de cada federação integrante da interfederação. Dessa forma, em vez de haver uma relação de confiança estabelecida par-a-par entre as ACs de cada federação, existe uma relação de confiança entre cada uma das ACs das federações e a AC raiz da PKI da interfederação.

Para que a confiança entre as federações seja estabelecida consideramos duas etapas antes da operação efetiva do FLAT: o estabelecimento de confiança entre as federações, com a assinatura do certificado da federação pela AC raiz da PKI da interfederação (pré-operação 1) e a distribuição aos componentes da federação dos novos certificados assinados (pré-operação 2). A nomenclatura utilizada na descrição destas etapas é mostrada na Tabela 1. As etapas de pré-operação são descritas na Fig. 2.

Na fase de pré-operação 1 é estabelecida a relação de confiança entre as partes, de forma que antes que esta etapa seja concluída, nem a AC raiz, nem a AC da federação

podem confiar em certificados auto-assinados enviados via rede uma para a outra. Assim, o processo descrito na pré-operação 1 não ocorre de forma automatizada via rede. Geralmente o estabelecimento de confiança é um processo manual ou feito por um canal seguro pré-estabelecido entre as partes que garanta a veracidade dos certificados envolvidos.

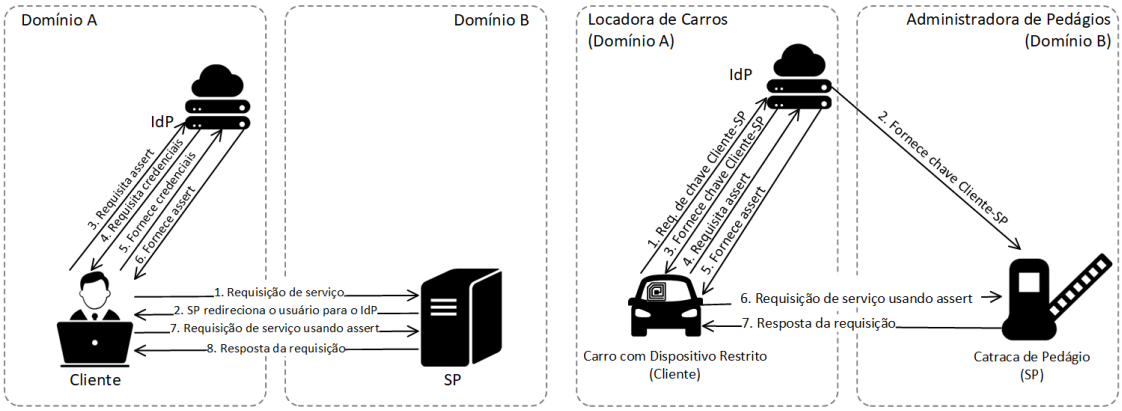
No passo 1 da Fig. 2a, a AC da federação A apresenta à AC raiz seu certificado auto-assinado contendo seus dados e informações de identificação (organização, nomes, filiação) juntamente com a chave pública utilizada pela AC da federação A para garantir os certificados por ela emitidos. No passo 2 da Fig. 2a, a AC raiz da interfederação retorna o certificado da AC da federação A, agora também assinado pela AC raiz.

Os novos certificados podem ser distribuídos antes da etapa de operação aos SPs e IdPs da federação recentemente integrada à interfederação, como mostra o exemplo da etapa de pré-operação 2 (Fig. 2b), que atualiza o certificado de um IdP da federação A.

Na pré-operação 2, a AC da federação A envia seu certificado agora também assinado pela AC raiz ao IdP da federação A (passo 1, Fig. 2b). Para confirmar que a operação ocorreu conforme esperado, no passo 2 da Fig. 2b o IdP da federação A envia o *nonce* n_A assinado, informando à AC de sua federação que o certificado foi recebido.

2.2. Protocolo FLAT

Visão Geral. Nossa proposta é um modelo de autenticação eficaz para dispositivos IoT com restrições de recursos que precisem ser autenticados em diferentes domínios. Por exemplo, na Fig. 3 é feita uma comparação entre o modelo tradicional de FIdM (Fig. 3a) e nossa proposta de FIdM para IoT (Fig. 3b), em que um veículo possui um Cliente IoT restrito embarcado que interage com um sistema de pagamento automático de pedágio.



(a) FIdM Tradicional **(b) Proposta de FIdM para IoT**

Figura 3: **Modelos de FIdM tradicional versus Proposta de Modelo FIdM para IoT**

O veículo precisa passar por praças de pedágio em diferentes estados, com diferentes empresas de cobrança. As diversas empresas de pagamento automático de pedágio formam uma federação dentro de um estado, tendo sua própria AC. Para que o Cliente IoT possa acessar os sistemas de pagamento automático de outros estados, é necessário o estabelecimento de uma federação de federações, ou interfederação, conforme descrito nas premissas de nossa solução. Neste exemplo, os sistemas de cobrança automática de pedágio, seriam mapeados como SPs. Os dispositivos dos veículos seriam os Clientes IoT que requisitam aos SPs o serviço de autorização para passar no pedágio e, por isso,

precisam ser autenticados (independentemente da identidade do dono ou motorista do veículo). O Cliente IoT solicita ao seu IdP uma chave simétrica para comunicação com o SP. O IdP informa a chave ao SP e ao Cliente IoT. Dessa forma, após receber o *assert* do IdP o Cliente IoT poderá realizar o pagamento automático de pedágio junto ao SP. Nesse contexto, os IdPs poderão ser gerenciados por uma associação de empresas de cobrança automática de pedágio ou órgãos do governo.

Tabela 2: **Nomenclatura utilizada na descrição do protocolo FLAT**

Client : Cliente IoT	$\|$: operação de concatenação
IdP: IdP do domínio de origem do Cliente IoT	$=$: operação de atribuição
SP : SP de domínio estrangeiro	n_x: *nonce* gerado por x
P_x: chave pública de x	S_x: chave privada de x
k_x/y: chave secreta k compartilhada entre x e y	cert_x: certificado de x
assert_req: requisição de *assert*	key_req: requisição de chave
serv: serviço propriamente dito	'serv ': descrição do serviço
function(k,s): função sobre a *string* s usando a chave k	

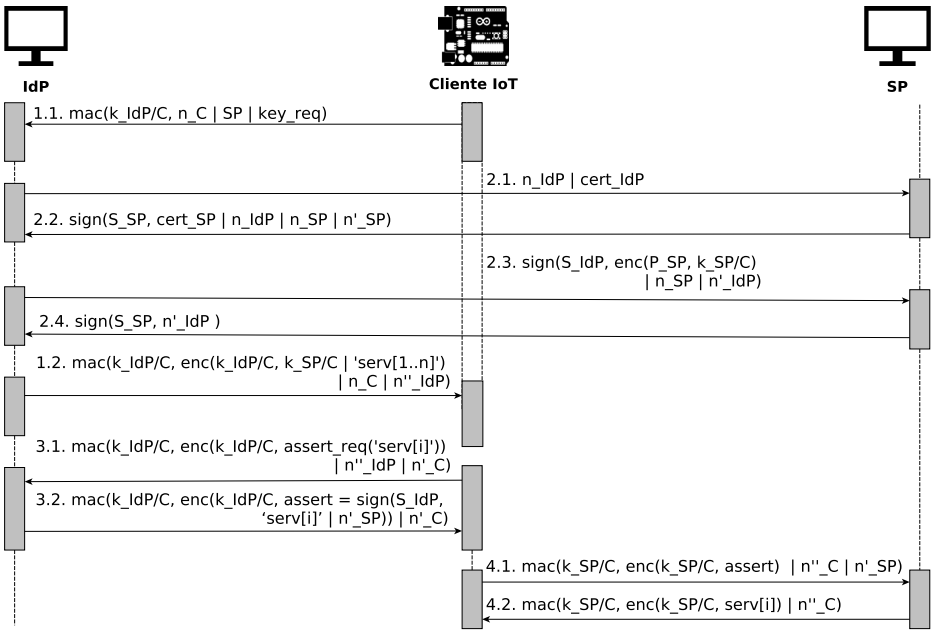


Figura 4: **Operação do FLAT**

Operação. O FLAT possui quatro etapas: (1) o Cliente IoT obtém uma chave com o IdP para utilizar na comunicação com o SP; (2) o IdP envia ao SP a chave que será usada na comunicação com o Cliente IoT; (3) o Cliente IoT obtém um *assert* de acesso ao serviço do SP; (4) o Cliente IoT acessa o serviço do SP.

Descrevemos, a seguir, cada um dos passos da etapa de operação do FLAT (Fig. 4) com suas respectivas primitivas criptográficas, utilizando a nomenclatura definida na Tabela 2. Note o uso de *Message Authentication Code* (MAC) para a garantia de autenticidade e integridade nas mensagens trocadas com o Cliente IoT. Para que o Cliente IoT utilize os serviços do SP ele precisa, primeiramente, obter junto ao IdP uma chave para se comunicar com o SP (k_SP/C). Dessa forma, o Cliente IoT se autentica utilizando a chave (k_IdP/C) compartilhada previamente entre ele e o IdP e solicita ao IdP uma chave para comunicação com o SP (passo 1.1, Fig. 4).

Em seguida, o IdP estabelece um canal seguro com o SP (passos 2.1 e 2.2, Fig. 4) utilizando assinatura digital e envia ao SP (passo 2.3, Fig. 4) a chave (k_SP/C) que será utilizada na comunicação do Cliente IoT com o SP. O SP então confirma o recebimento da chave ao IdP (passo 2.4, Fig. 4), concluindo a etapa 2 do protocolo. A etapa 1 do protocolo é concluída em seguida, com o envio da chave k_SP/C também ao Cliente IoT (passo 1.2, Fig. 4). O Cliente IoT, então, requisita ao IdP permissão de acesso ao serviço do SP – $sign(S\_IdP, \text{`}serv[i]\text{'} | \ n'\_SP)$ (passos 3.1 e 3.2, Fig. 4), por meio do *assert*, concluindo a etapa 3 do protocolo. Esse *assert* será utilizado pelo Cliente IoT para provar sua identidade junto ao SP.

De posse da chave e do *assert* do serviço, o Cliente IoT requisita o serviço ao SP (passo 4.1, Fig. 4). O SP então retorna uma mensagem de resposta com o serviço para o Cliente IoT (passo 4.2, Fig. 4). As mensagens recebem *nonces* (n''_C) para prevenir ataques de *replay* e são protegidas com a chave k_SP/C. Além disso, a mensagem é protegida por um MAC para garantir a integridade e autenticidade da requisição enviada.

Formato do Pacote. O formato do pacote utilizado no protocolo pode ser visto na Fig. 5. O primeiro *byte* é utilizado para identificar o tipo de mensagem que está sendo enviada. O FLAT possui dez tipos diferentes de mensagens, portanto um *byte* é suficiente. O segundo *byte* armazena o número de sequência do pacote. Os dois campos seguintes possuem 2 *bytes* cada e foram reservados para armazenar os identificadores de destino e origem da mensagem na camada de aplicação. Dessa forma, é possível identificar até pouco mais de 65 mil clientes IoT. Em seguida, há um campo de dois *bytes* para armazenar o tamanho do *payload*. Finalmente, definimos um *payload* de até 280 *bytes*, suficiente para armazenar as mensagens de envio de certificados utilizadas no FLAT, que constituem a maior sobrecarga de comunicação.

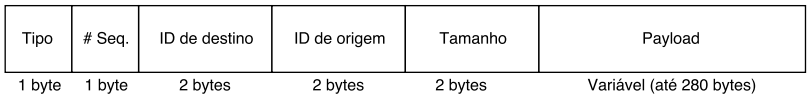


Figura 5: **Formato do Pacote FLAT**

3. Implementação e Protótipo

A implementação do protocolo, como protótipo, contemplou o desenvolvimento em C/C++ das funções de comunicação, além da integração da biblioteca criptográfica e do desenvolvimento dos módulos de criptografia necessários ao funcionamento do FLAT. Para a comunicação, utilizamos o *shield* Arduino WiFi[3]. Foi utilizada a biblioteca Arduino WiFi para programação do *shield* WiFi, além de um Makefile[4] adaptado para auxiliar na implementação, execução e depuração do código. Nas implementações dos módulos criptográficos foi utilizada a biblioteca Relic [Aranha and Gouvêa].

Para a execução da prova de conceito do FLAT utilizamos três dispositivos: dois Arduinos Due (Cliente IoT e SP) e um computador Intel Core i7 2.7 GHz, 8 GB RAM (IdP). O Arduino Due foi utilizado na execução do SP e Cliente IoT por ser um dispositivo restrito (processador Atmel de 84 MHz, 96 kB de SRAM e 512 kB de memória *flash*), mas ao mesmo tempo capaz de executar as primitivas criptográficas necessárias ao protocolo.

[3]https://www.arduino.cc/en/Reference/WiFi

[4]https://github.com/pauldreik/arduino-due-makefile

A arquitetura do FLAT é composta por três módulos: o do Cliente IoT, o do SP e o do IdP. O módulo Cliente IoT possui dois componentes criptográficos, o *Advanced Encryption Standard* (AES) para criptografia simétrica e o *Hash-based Message Authentication Code* (HMAC) para os MACs utilizados no protocolo. Os módulos do SP e do IdP possuem um maior número de componentes criptográficos, como o AES, o *Elliptic Curve Integrated Encryption Scheme* (ECIES), o *Elliptic Curve Digital Signature Algorithm* (ECDSA), o *Elliptic Curve Qu-Vanstone* (ECQV) e o HMAC, necessários para criptografia simétrica, criptografia assimétrica (cifração), criptografia assimétrica (assinatura), certificados implícitos e MACs, respectivamente. A adoção do ECQV no SP e IdP tem como objetivo diminuir os custos de comunicação e processamento, por proporcionar redução no tamanho dos certificados e no uso de recursos computacionais.

4. Resultados e Análise

Esta seção apresenta uma avaliação dos custos envolvidos no FLAT em relação ao consumo de memória, tempo de processamento e consumo de recursos de rede.

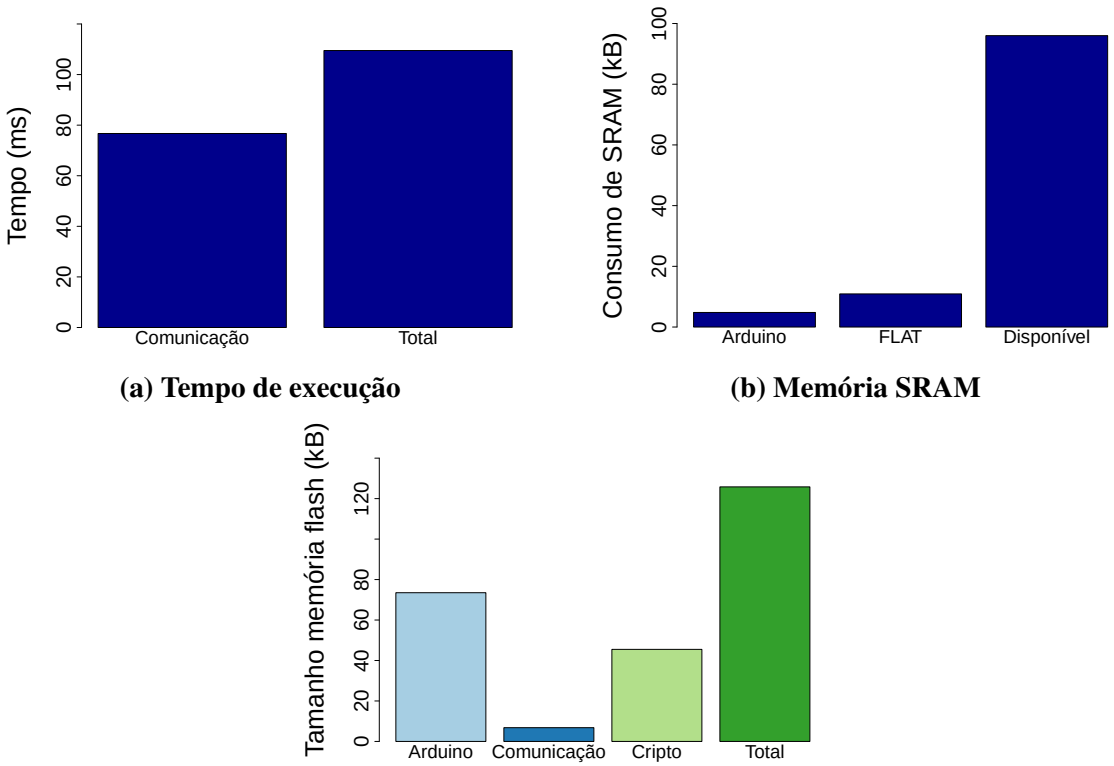
4.1. Uso de Recursos de Armazenamento e Processamento no Cliente IoT

Os experimentos descritos nesta subseção referem-se ao nó Cliente IoT, por este representar o dispositivo mais restrito em termos de recursos de memória *Static Random Access Memory* (SRAM) e de armazenamento.

Tempo de execução e memória SRAM. Para calcular o custo de tempo do protocolo, verificamos inicialmente o tempo necessário para a transmissão de mensagens e comunicação (Comunicação, Fig. 6a), obtendo o valor de 76,7 $\pm$ 7,5 ms com 85% de confiança. Posteriormente, calculamos o tempo de execução do protocolo por completo, executando todas as operações criptográficas e de comunicação (Total, Fig. 6a), obtendo o valor de 109,5 $\pm$ 19 ms com 85% de confiança.

Com o objetivo de calcular os custos de SRAM envolvidos no carregamento das bibliotecas nativas do Arduino, medimos o uso de memória embarcando no dispositivo uma aplicação vazia que apenas calculava o consumo de SRAM, obtendo o valor de 4,8 kB (Arduino, Fig. 6b). Em seguida, executamos o FLAT no Cliente IoT e calculamos o consumo de memória, desconsiderando a memória SRAM consumida no carregamento das bibliotecas nativas do Arduino, obtendo o valor de 10,9 kB (FLAT, Fig. 6b). Estes resultados foram comparados com a SRAM total disponível no Arduino Due, que é de 96 kB (Disponível, Fig. 6b). O consumo total de memória SRAM ocasionado pelo FLAT e pelas bibliotecas nativas do Arduino é de 16,4% do total disponível.

Tamanho da Aplicação. Em nossa análise do custo de armazenamento da aplicação, verificamos inicialmente o espaço na memória *flash* ocupado pelas bibliotecas nativas do Arduino, obtendo o valor de 73,5 kB (Arduino, Fig. 6c). Procedemos então para o custo de armazenamento relacionado às operações de comunicação do protocolo, que é de 6,8 kB (Comunicação, Fig. 6c). Avaliamos também o espaço ocupado pela biblioteca Relic juntamente com as funções criptográficas implementadas, obtendo o valor de 45,5 kB (Cripto, Fig. 6c). Por fim, calculamos o custo de armazenamento do protocolo completo embarcado no Cliente IoT, que é de 125,8 kB (Total, Fig. 6c). Desconsiderando o tamanho das bibliotecas nativas do Arduino, o maior custo de armazenamento do FLAT são as funções criptográficas, que correspondem a 36,1% do total da aplicação, enquanto



(a) Tempo de execução

(b) Memória SRAM

(c) Tamanho do FLAT e seus componentes

Figura 6: **Custos do Cliente IoT**

as funções de comunicação correspondem a 5,4% do total. Comparando o custo total de armazenamento do protocolo com a memória *flash* disponível no Arduino (512 kB), verificamos que a aplicação do FLAT ocupa 24,6% da memória *flash* disponível.

4.2. Custo de Comunicação

Para analisarmos o custo adicional de acesso à rede consumido pelo protocolo na fase de operação, comparamos o FLAT com um Baseline teórico baseado em um protocolo genérico de FIdM (fundamentado no OAuth) adaptado para IoT. Procedemos assim pois não seria adequado comparar uma solução exclusivamente projetada para dispositivos IoT com capacidade de processamento reduzida (FLAT) com uma solução tradicional ou que adapta um modelo tradicional para IoT utilizando dispositivos mais robustos (com capacidade de processamento de primitivas criptográficas assimétricas), como a proposta de Domenech *et al.* [Domenech et al. 2016]. Realizamos uma análise dos custos do Baseline e comparamos com os custos do FLAT com e sem o uso do ECQV.

Baseline. Na figura 7 apresentamos o esquema de autenticação genérico usado como Baseline. A notação utilizada é a mesma adotada para a descrição da operação do FLAT (Tabela 2). O Cliente, capaz de executar primitivas assimétricas, inicia a comunicação com o SP (passo 1, Fig. 7). O Cliente então recebe o certificado do SP (passo 2, Fig. 7). Neste ponto, o Cliente verifica as informações do certificado recebido e extrai a chave pública correspondente, cifrando uma outra chave que será compartilhada entre o Cliente e o SP (passo 3, Fig. 7). A garantia de que o SP conseguiu decifrar a mensagem é alcançada através do envio de um *nonce* utilizando como chave do MAC a chave simétrica compartilhada recebida do Cliente (passo 4, Fig. 7). No passo 5 da Fig. 7, o Cliente envia as

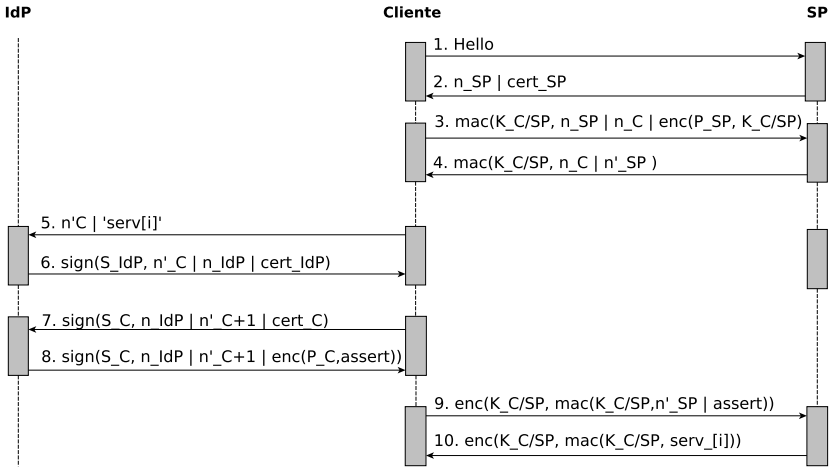


Figura 7: **Protocolo de identidade federada genérico usado como Baseline**

especificações do serviço que deseja acessar ao IdP, recebendo o certificado do IdP para verificação de sua identidade (passo 6, Fig. 7) e enviando seu certificado logo em seguida (passo 7, Fig. 7). Nos passos 8 e 9 da Fig. 7, respectivamente, o Cliente recebe o *assert* e o encaminha para o SP, que no passo 10 da Fig. 7 responde à requisição do Cliente.

Tabela 3: **Volume de comunicação do Cliente IoT**

Origem	**Destino**	**Baseline**	**FLAT**
Cliente IoT	IdP	279	90
IdP	Cliente IoT	599	182
Cliente IoT	SP	341	116
SP	Cliente IoT	293	50

Mensagens trocadas com o Cliente IoT. A quantidade de dados trocados com o Cliente IoT na fase de operação do FLAT é, em média, 71,6% menor do que no protocolo Baseline (Tabela 3). No Baseline é feito o uso de primitivas criptográficas assimétricas na comunicação com o Cliente IoT, o que gera uma sobrecarga na troca de mensagens. No FLAT, o Cliente IoT não realiza troca de certificados e nem utiliza criptografia assimétrica, o que se reflete positivamente no custo de comunicação.

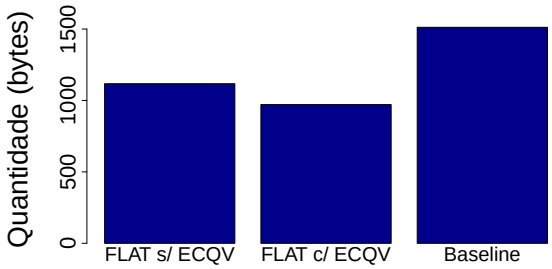


Figura 8: **Quantidade total de bytes trocados pelos protocolos**

Volume total de comunicação. Avaliamos o total em *bytes* trocados pelo protocolo Baseline e pela fase de operação do FLAT, considerando todas as mensagens trocadas entre Cliente IoT, IdP e SP (Fig. 8). O uso de primitivas simétricas na comunicação com o Cliente IoT, além do uso de *Elliptic Curve Cryptography* (ECC) na assinatura e cifração de

mensagens, permitiu uma redução de 26,1% na quantidade de *bytes* trocados pelo FLAT em sua versão sem ECQV (FLAT s/ ECQV, Fig. 8) quando comparado ao Baseline (Baseline, Fig. 8), evidenciando que nosso modelo de autenticação especialmente projetado para IoT e nossa escolha por primitivas criptográficas leves geram uma redução no volume de comunicação. A versão com o uso de ECQV (FLAT c/ ECQV, Fig. 8) proporcionou uma redução de 35,8% em relação ao Baseline. De fato, o uso do ECQV também diminui o volume de comunicação, uma vez que reduz o tamanho dos certificados trocados pelo IdP e SP, mantendo o mesmo nível de segurança utilizado pelo protocolo Baseline.

4.3. Avaliação de Segurança

Nesta subseção é feita uma análise de segurança da operação do FLAT, bem como da etapa de pré-operação.

Pré-operação. A etapa de pré-operação 1 (Fig. 2a) não ocorre de forma automatizada via rede e geralmente é realizada de forma manual. Portanto, esta etapa é considerada um processo formal e seguro de estabelecimento de confiança entre as partes. Já na etapa de pré-operação 2 (Fig. 2b), não são transmitidos dados sigilosos, e portanto, nenhuma das mensagens é cifrada. No entanto, para garantir a autenticidade, integridade, não-repúdio e *freshness*, as mensagens são assinadas e recebem *nonces*.

Operação. Assumindo que a chave secreta compartilhada entre IdP e Cliente IoT é gerada utilizando PUF e carregada através de um canal seguro, o FLAT oferece confidencialidade, integridade, autenticidade e *freshness* na autenticação de dispositivos IoT.

Na etapa de obtenção da chave para comunicação com o SP (passos 1.1 e 1.2, Fig. 4) todas as mensagens utilizam MAC de forma a assegurar sua autenticidade e integridade, uma vez que somente o Cliente IoT e o IdP conhecem a chave utilizada no MAC e ambos possuem o *hash* da mensagem. Além disso, o *freshness* das mensagens nesta etapa é garantido através do uso de *nonces*, assim como nas outras etapas do FLAT.

Para que o SP receba a chave para comunicação com o Cliente IoT (passos 2.1 a 2.4, Fig. 4), inicialmente o IdP envia seu certificado ao SP (passo 2.1, Fig. 4). Nesta mensagem não é necessário o uso de criptografia, uma vez que estão sendo enviados apenas dados não sigilosos: o certificado e um *nonce*. Da mesma maneira, no passo 2.2 da Fig. 4 não é necessário que a mensagem seja cifrada, pois também estão sendo enviados apenas *nonces* e o certificado do SP. No entanto, neste passo a mensagem é enviada assinada, de forma a garantir que é realmente o SP que está respondendo ao IdP e que a mensagem não foi modificada. No passo 2.3 da Fig. 4 é enviada a chave ao SP, de forma que para garantir sua confidencialidade, autenticidade, integridade e não-repúdio, esta é cifrada e assinada. O SP, para confirmar o recebimento da chave no passo 2.4 da Fig. 4, retorna o *nonce* recebido do IdP assinado, garantindo o *freshness* da mensagem.

Na etapa 3, em que o Cliente IoT recebe o *assert* do serviço (passos 3.1 e 3.2, Fig. 4) é utilizado o MAC, sendo mantida também a confidencialidade da requisição de *assert* e do próprio *assert*, que são cifrados pela chave compartilhada entre o Cliente IoT e o IdP. O *nonce* n'_SP, recebido pelo IdP no passo 2.2 da Fig. 4 é passado ao Cliente IoT no passo 3.2 da Fig. 4, de forma a garantir o *freshness* da mensagem e identificar que trata-se da mesma requisição de serviço feita nos passos anteriores. Este *nonce* será repassado ao SP no passo 4.1 da Fig. 4. Note que o *assert* consiste em uma assinatura, e portanto, garante que seu conteúdo foi realmente emitido pelo IdP e não foi alterado. Por

fim, na etapa 4, que também utiliza o MAC com a chave compartilhada entre SP e Cliente IoT, é encaminhado o *assert* cifrado ao SP. A confidencialidade do serviço fornecido ao Cliente IoT também é protegida, uma vez que este é cifrado no passo 4.2 da Fig. 4.

5. Trabalhos Relacionados

Há muitas propostas de FIdM para IoT que se baseiam nas soluções de FIdM para Internet (e.x. [Fremantle and Aziz 2016, Cirani et al. 2015, Domenech et al. 2016]). Essas propostas têm a vantagem de permitir a integração com sistemas tradicionais de FIdM, já que procuram adaptar para o contexto de IoT protocolos tradicionais de FIdM (como o OAuth[5]), reduzindo os custos computacionais do protocolo original [Fremantle and Aziz 2016]. Por exemplo, a proposta de Domenech *et al.* prevê uma solução de FIdM para IoT com suporte a SAML[6]. A solução está disponível no GIdLab [Wangham et al. 2013] e se aplica aos clientes IoT com recursos intermediários como BeagleBone ou Raspberry Pi. Porém, o custo da solução é alto se considerarmos clientes IoT com severas restrições de recursos como, por exemplo, o Arduino Due ou nós sensores. Silva e Silva também propõem uma arquitetura para autorização federada para IoT [Silva and Silva 2017]. Sua abordagem utiliza SAML para implementar um mecanismo de controle de acesso baseado no modelo *Role-based Access Control* (RBAC). Para endereçar o problema de dispositivos restritos, eles sugerem o uso de um controlador lógico com maior poder computacional para implementar os protocolos mais pesados. Nossa solução, o FLAT, adota uma estratégia diferente justamente para atender a dispositivos restritos e implementa protocolos leves para dispensar o uso de *gateways* externos.

Há também iniciativas de FIdM voltadas para cenários específicos de IoT (e.x. [Witkovski et al. 2015, Hong et al. 2016]). Witkovski *et al.* propõem uma solução que integra FIdM e IoT para fornecer SSO em serviços operados por técnicos de manutenção de fabricantes de produtos eletrônicos. Esta solução se baseia em criptografia simétrica e chaves de sessão para autenticar os técnicos e os dispositivos IoT, de forma que os fabricantes possam fornecer suporte aos dispositivos. Em um trabalho breve, Hong *et al.* afirmam que *Bluetooth Low Energy* (BLE) é uma alternativa viável para obtenção de controle de acesso em ambientes IoT domésticos. Para demonstrar sua afirmação, os autores desenvolveram um protótipo de controle de acesso utilizando BLE em dispositivos com recursos reduzidos. O trabalho de Hong *et al.*, assim como o de Witkovski *et al.*, propõem soluções interessantes para cenários IoT específicos. Nosso trabalho, no entanto, busca uma solução mais abrangente, que possa ser aplicada a diferentes ambientes IoT.

Entre os protocolos de autenticação com primitivas criptográficas simétricas, podemos destacar o Kerberos [Neuman et al. 2005]. Apesar de existir semelhanças entre o Kerberos e nossa proposta, como o fato de ambos permitirem a autenticação de usuários e/ou dispositivos de diferentes domínios, há diversos fatores que os diferenciam. Para começar, o Kerberos prevê, além do cliente, três entidades no processo de autenticação, enquanto o FLAT prevê duas: (i) o Servidor de Aplicação – *Application Server* (Server), equivalente ao SP; (ii) o Servidor de Autenticação – *Authentication Server* (AS), equivalente ao IdP; e o (iii) Servidor de Concessão de *Tickets* – *Ticket Granting Server* (TGS), para o qual não há uma entidade equivalente no FLAT, uma vez que a autenticação do

[5]https://tools.ietf.org/html/rfc6749
[6]https://tools.ietf.org/html/rfc7522

Cliente IoT e a emissão do *assert* para acesso ao serviço são ambas realizadas pelo IdP. Outra diferença é que enquanto no FLAT há a distribuição prévia de uma chave simétrica entre o Cliente IoT e o IdP, no Kerberos é necessária a distribuição prévia de uma chave simétrica entre o Server e o AS e a chave utilizada para cifrar as mensagens enviadas entre o AS e o cliente é uma chave simétrica derivada da senha do usuário.

6. Conclusão

Neste trabalho apresentamos o FLAT, um protocolo de autenticação federada especialmente modelado para dispositivos IoT com capacidade de processamento e armazenamento reduzidas. Discutimos cenários de aplicação da solução, em que a autenticação das coisas é mais relevante que a autenticação de seus usuários, e apresentamos um protótipo da solução na plataforma Arduino. Desenvolvemos uma avaliação experimental e analítica do FLAT, comparando os custos de memória SRAM, armazenamento e comunicação. Ainda, realizamos uma análise de segurança das etapas envolvidas no FLAT.

A avaliação dos custos de comunicação mostrou que a adoção de criptografia simétrica no Cliente IoT e do ECQV torna possível uma redução em média de 71,6% no volume de dados trocados com o Cliente IoT e de 35,8% no custo de comunicação total da fase de operação do FLAT, quando comparado com o protocolo Baseline. Em relação ao uso de recursos no Cliente IoT, o FLAT faz uso de 16,4% de SRAM e 24,6% de memória *flash* de uma plataforma Arduino, por exemplo. Como trabalhos futuros, pretendemos realizar testes com outros dispositivos IoT e aplicar de forma prática o protótipo a cenários reais de uso. Além disso, pretendemos medir o consumo energético da solução em diferentes dispositivos.

Agradecimentos

Agradecemos à RNP pelo apoio e financiamento deste projeto. Também agradecemos o apoio de CAPES, CNPq e FAPEMIG.

Referências

Abomhara, M. and Køien, G. M. (2014). Security and privacy in the internet of things: Current status and open issues. In *PRISMS 2014*, pages 1–8.

Aranha, D. F. and Gouvêa, C. P. L. RELIC is an Efficient LIbrary for Cryptography. `https://github.com/relic-toolkit/relic`.

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805.

Brown, D. R., Gallant, R., and Vanstone, S. A. (2001). Provably secure implicit certificate schemes. In *FC'01 Financial Cryptography*, pages 156–165. Springer.

Cirani, S., Picone, M., Gonizzi, P., Veltri, L., and Ferrari, G. (2015). IoT-OAS: An OAuth-based Authorization Service Architecture for Secure Services in IoT Scenarios. *IEEE Sensors Journal*, 15(2):1224–1234.

Domenech, M. C., Boukerche, A., and Wangham, M. S. (2016). An Authentication and Authorization Infrastructure for the Web of Things. In *Q2SWinet*. ACM.

Fremantle, P. and Aziz, B. (2016). Oauthing: privacy-enhancing federation for the internet of things. In *Cloudification of the Internet of Things (CIoT)*, pages 1–6. IEEE.

Fremantle, P., Aziz, B., Kopecký, J., and Scott, P. (2014). Federated identity and access management for the internet of things. In *SIoT'14*.

Hong, J., Levy, A., and Levis, P. (2016). Demo: Building Comprehensible Access Control for the Internet of Things Using Beetle. In *MobiSys'16*.

Koppula, S. and Muthukuru, J. (2016). Secure digital signature scheme based on elliptic curves for internet of things. *International Journal of Electrical and Computer Engineering*, 6(3):1002.

Lopez, D. R., Macias, J., Molina, M., Rauschenbach, J., Solberg, A., and Stanica, M. (2006). *Deliverable DJ5.2.3.1: Best Practice Guide – AAI Cookbook*. Géant 2.

Menezes, A. J., van Oorschot, P. C., and Vanstone, S. A. (2001). *Handbook of Applied Cryptography*. CRC Press.

Neuman, C., Yu, T., Hartman, S., and Raeburn, K. (2005). The kerberos network authentication service (v5). RFC 4120, RFC Editor.

Nogueira, H., Custodio, R. F., Moecke, C. T., and Wangham, M. S. (2011). Using notary based public key infrastructure in shibboleth. In *SBSeg'11*, pages 405–413. SBC.

Sasso, F. C., De Moraes, R. A. R., and Martina, J. E. (2014). A proposal for a unified identity card for use in an academic federation environment. In *Availability, Reliability and Security (ARES), 2014 Ninth International Conference on*, pages 265–272. IEEE.

Shim, S. S., Bhalla, G., and Pendyala, V. (2005). Federated identity management. *Computer*, 38(12):120–122.

Silva, C. E. and Silva, G. C. (2017). Uma proposta de arquitetura para autorização federada com internet das coisas. In *SBSeg'17*, pages 783–766. SBC.

Suh, G. E. and Devadas, S. (2007). Physical unclonable functions for device authentication and secret key generation. In *Proceedings of the 44th DAC*, pages 9–14. ACM.

Ververidis, C. N. and Polyzos, G. C. (2008). Service discovery for mobile ad hoc networks: a survey of issues and techniques. *IEEE Communications Surveys & Tutorials*, 10(3).

Wangham, M. S., Mello, E. R. d., Souza, M. C., and Coelho, H. (2013). Gidlab: Laboratório de experimentação em gestão de identidades. In *SBSeg'13*, pages 481–486. SBC.

Wei, J. (2014). How wearables intersect with the cloud and the internet of things : Considerations for the developers of wearables. *IEEE Consumer Electronics Magazine*, 3(3):53–56.

Witkovski, A., Santin, A., Abreu, V., and Marynowski, J. (2015). An IdM and Key-based Authentication Method for Providing Single Sign-On in IoT. In *GLOBECOM*. IEEE.