FLAT: Federated Lightweight Authentication of Things

Maria L. B. A. Santos[1], **Leonardo B. Oliveira**[1], **Marco A. A. Henriques**[2]

[1]UFMG, Belo Horizonte. [2]Unicamp, Campinas.

mburga@dcc.ufmg.br, leob@dcc.ufmg.br, marco@dca.fee.unicamp.br

Abstract. *The authentication of devices is one of the current challenges in IoT. In this sense, it is essential the development of an authentication model for IoT, especially considering the computational and storage restrictions of devices and their potential mobility between different domains. FLAT is a federated authentication protocol for IoT that aims to be a solution to this problem, associating symmetric cryptosystems in the Client side and implicit certificates. The results show that FLAT can reduce the data exchange overhead in around 31% when compared to the Baseline solution. FLAT's Client is also more efficient than the Baseline solution in terms of data transmitted and computation time, showing it can be an alternative to authentication in restricted devices.*

Resumo. *A autenticação de dispositivos é um dos atuais desafios em IoT. Neste sentido, é essencial um modelo de autenticação para IoT, especialmente quando consideramos as restrições de recursos dos dispositivos e sua potencial mobilidade. FLAT é um protocolo de autenticação federada para IoT que busca solucionar este problema, associando criptografia simétrica no Cliente e certificados implícitos. Os resultados mostram que FLAT é capaz de reduzir o total de dados transmitidos em cerca de 31% quando comparado com a solução Baseline. O dispositivo Cliente no FLAT é também mais eficiente do que a solução Baseline em total de dados trocados e em tempo de computação, mostrando que o FLAT pode ser uma alternativa para autenticação de dispositivos restritos.*

1. Introdução

A Internet das Coisas (*Internet of Things* – IoT) [Atzori et al. 2010] tem sido um tópico de pesquisa cada vez mais explorado, dadas suas inúmeras possibilidades de aplicação e o número crescente de dispositivos conectados. A utilização de tecnologias IoT tem modificado a vida diária das pessoas e também tem contribuído para a evolução de diversos processos na indústria, incluindo áreas como agricultura, mineração e pecuária.

Um dos desafios encontrados atualmente no desenvolvimento da IoT está relacionado à Gestão de Identidades (*Identity Management* – IdM). IdM é essencial no uso seguro de sistemas, fornecendo meios para controlar todo o ciclo de vida das identidades dos usuários e controlando o acesso aos recursos disponíveis no sistema [Windley 2005].

Dentre os modelos de IdM, temos a Gestão de Identidades Federada (*Federated Identity Management* – FIdM), que permite gerenciar o acesso de usuários externos aos recursos disponíveis em domínios diversos [Shim et al. 2005]. Em FIdM o acesso dos usuários aos recursos e serviços fornecidos por um determinado Provedor de Serviços (*Service Provider* – SP) passa pela autenticação no Provedor de Identidades (*Identity Provider* – IdP) local dos usuários. FIdM apresenta várias vantagens, como o aumento da

privacidade, uma vez que somente o IdP de origem do usuário mantém suas informações de identidade; permite o *Single-Sign-On* (SSO) [Maler and Reed 2008]; e evita a criação de múltiplas credenciais em cada domínio que o usuário utilizar um serviço.

Motivação. Na Internet tradicional existem soluções consolidadas em (F)IdM. Em IoT, entretanto, as soluções existentes não abordam todas as questões inerentes a este contexto. Em muitos cenários, os dispositivos utilizam as credenciais de seus usuários ao invés de utilizarem credenciais do próprio dispositivo. Esta não é uma prática ideal, pois um mesmo dispositivo pode ter diversos usuários (e então a qual identidade o dispositivo deve ser associado?) e na maioria dos casos o dispositivo não deveria ter as mesmas permissões de acesso que um usuário humano. As soluções existentes também são geralmente baseadas em criptossistemas pesados, como RSA/DSA, inadequados para dispositivos restritos frequentemente encontrados em cenários IoT.

FLAT, *Federated Lightweight Authentication of Things* é uma solução de autenticação federada para IoT que busca solucionar este problema. Entre as estratégias adotadas pelo FLAT para fornecer um modelo de autenticação leve para IoT estão: uso de apenas criptografia simétrica no dispositivo Cliente, uso de certificados implícitos [Brown et al. 2001] e substituição de criptossistemas pesados por outros mais adequados a IoT. Evita-se assim o uso da criptografia de chave pública no Cliente, e a consequente necessidade de processamento de primitivas assimétricas e longos certificados tradicionais.

Objetivos. O objetivo deste trabalho é fornecer um modelo de autenticação especialmente concebido para IoT, atingindo os seguintes objetivos específicos: (i) concepção de um protocolo de autenticação federado para IoT e suas respectivas premissas, (ii) implementação do protocolo de autenticação proposto, (iii) avaliação do protocolo de autenticação.

Produções Científicas. As seguintes produções científicas são associadas a este trabalho:

1. Artigo completo. *FLAT: Um Protocolo de Autenticação Federada para a Internet das Coisas*. Maria L. B. A. Santos, Jéssica C. Carneiro, Antônio M. R. Franco, Fernando A. Teixeira, Marco A. A. Henriques, Leonardo B. Oliveira. Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos – SBRC'18.

2. Artigo do Salão de Ferramentas. *Autenticação Federada para IoT Aplicada a um Sistema Automático de Estacionamento*. Maria L. B. A. Santos, Jéssica C. Carneiro, Antônio M. R. Franco, Fernando A. Teixeira, Marco A. A. Henriques, Leonardo B. Oliveira. Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos – SBRC'18. Prêmio de melhor artigo do Salão de Ferramentas SBRC'18.

3. Artigo da sessão de demonstrações. *Federated Authentication of Things: demo abstract*. Maria L. B. A. Santos, Jéssica C. Carneiro, Fernando A. Teixeira, Antônio M. R. Franco, Marco A. A. Henriques, Leonardo B. Oliveira. International Symposium on Information Processing in Sensor Networks – IPSN'18.

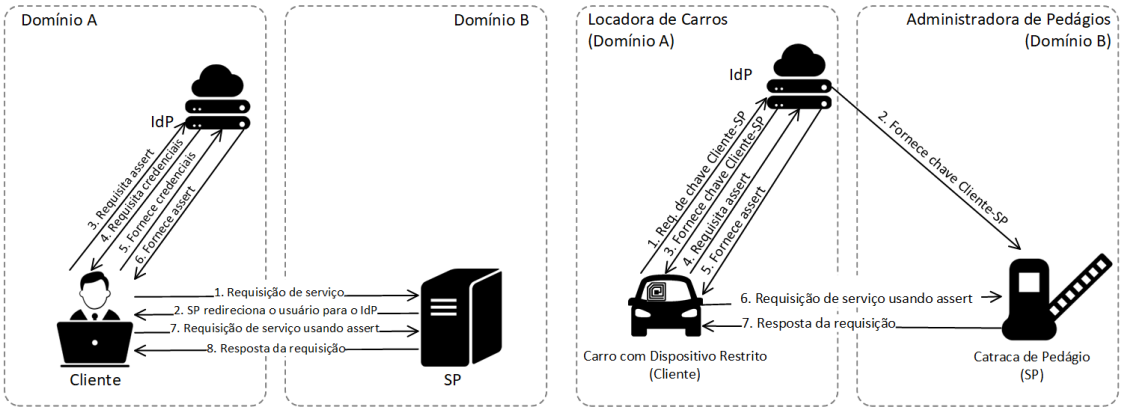
1.1. Organização

O restante deste artigo está organizado da seguinte forma. A Seção 2 descreve o protocolo de autenticação proposto e suas premissas. A Seção 3 fornece uma avaliação do FLAT considerando sua segurança, bem como seus custos de comunicação e computação. Os trabalhos relacionados são tratados na Seção 4. A Seção 5 apresenta uma conclusão, sumarizando os resultados obtidos.

2. FLAT: Autenticação Federada Leve para IoT

Esta seção descreve o funcionamento do FLAT, um protocolo de autenticação leve para IoT. Também são apresentadas as premissas necessárias ao funcionamento do FLAT.

A Fig. 1 mostra uma comparação entre um modelo FIdM tradicional e o modelo FIdM proposto pelo FLAT, aplicado a um cenário de pagamento automático de pedágio.



(a) FIdM tradicional. Baseado em [Birrell and Schneider 2013].

(b) FIdM especialmente concebido para IoT.

Figura 1: **FIdM tradicional (a) e FLAT (b)**.

Em um modelo FIdM tradicional, o Cliente solicita um serviço a um SP (passo 1, Fig. 1a), e então é redirecionado ao seu IdP de origem (passo 2, Fig. 1a) para que possa solicitar uma asserção que será repassada ao SP para obter acesso ao serviço (passo 3, Fig. 1a). Após fornecer suas credenciais (passos 4 e 5, Fig. 1a), o Cliente recebe a asserção (passo 6, Fig. 1a) e a encaminha para o SP (passo 7, Fig. 1a), que pode então fornecer o serviço (passo 8, Fig. 1a).

No FLAT, o Cliente solicita ao IdP uma chave para utilizar na comunicação com o SP (passo 1, Fig. 1b). O IdP então gera e envia para o SP a chave simétrica Cliente-SP (passo 2, Fig. 1b) e em seguida envia a mesma chave também para o Cliente (passo 3, Fig. 1b). O Cliente então solicita uma asserção ao IdP (passo 4, Fig. 1b) para enviar posteriormente ao SP de forma a obter acesso ao serviço. O Cliente então recebe a asserção (passo 5, Fig. 1b) e a encaminha juntamente com a requisição de serviço ao SP (passo 6, Fig. 1b), que provê o serviço ao Cliente (passo 7, Fig. 1b).

2.1. Premissas

Geração de Chaves. Uma vez que no FLAT o dispositivo Cliente utiliza apenas criptografia simétrica, assume-se que previamente à operação do protocolo ocorre a geração e distribuição de uma chave simétrica, de forma a possibilitar a comunicação $IdP \leftrightarrow Cliente$. Esta chave é gerada através de *Physical Unclonable Functions* (PUFs) de acordo com o descrito em [Suh and Devadas 2007], em que um *hash* aplicado à saída do PUF é utilizado como chave. O dispositivo Cliente gera a chave utilizando PUFs e envia ao IdP através de um canal seguro na etapa de pré-implantação, antes da operação do protocolo.

Descoberta de Serviço. Antes da operação do FLAT, é necessário o processo de descoberta de serviço, para que o Cliente tome conhecimento do SP e dos serviços sendo

ofertados por ele. Assume-se que previamente à operação do FLAT, *beacons* (mensagens de *broadcast* emitidas periodicamente em um determinado raio de distância) são utilizados pelo SP para enviar informações sobre seu *status* atual e serviços disponíveis.

Estabelecimento de Confiança. Antes da operação do FLAT é necessário que entidades (IdPs e SPs) de diferentes domínios estabeleçam uma relação de confiança entre si. Para tal, os certificados de cada entidade são assinados por uma Autoridade Certificadora (*Certificate Authority* – CA) comum a eles.

2.2. Formato da Mensagem

A Fig. 2 mostra o formato da mensagem utilizado no FLAT. O primeiro *byte* é utilizado para definir o tipo de mensagem sendo enviada (no FLAT são utilizados 9 tipos de mensagem). O segundo *byte* é utilizado para manter o número de sequência da mensagem. Os próximos 2 campos de 3 *bytes* cada são utilizados para manter as identificações de origem e destino na camada de aplicação. O quinto campo, de 2 bytes armazena o tamanho do *payload* da mensagem. O último campo de até 280 *bytes* armazena o *payload*.

Tipo	# Seq.	ID de destino	ID de origem	Tam.	Payload
1 byte	1 byte	3 bytes	3 bytes	2 bytes	Variável (até 280 bytes)

Figura 2: **Formato da mensagem.**

2.3. Operação do Protocolo

A Fig. 3 mostra a operação do FLAT, que é composta de 4 etapas: (i) Cliente obtém chave para comunicar-se com o SP, (ii) SP obtém chave para comunicar-se com o Cliente, (iii) Cliente obtém asserção e (iv) Cliente obtém o serviço utilizando a asserção.

Inicialmente, o Cliente envia uma requisição de chave ao IdP indicando o SP com o qual deseja se comunicar e obter um determinado serviço (passo 1.1, Fig. 3).

O IdP então envia ao SP seu certificado juntamente com um *nonce* (passo 2.1, Fig. 3), recebendo em seguida o certificado do SP (passo 2.2, Fig. 3). Após a troca de certificados, o IdP envia ao SP a chave simétrica que será utilizada na comunicação $Cliente \leftrightarrow SP$ (passo 2.3, Fig. 3). O SP conclui esta etapa da operação do protocolo enviando ao IdP um *nonce* assinado, indicando que a chave foi recebida (passo 2.4, Fig. 3).

Em seguida, a mesma chave simétrica enviada ao SP é também enviada ao Cliente, concluindo a etapa 1 da operação do protocolo iniciada no passo 1.1 (passo 1.2, Fig. 3).

O Cliente então solicita (passo 3.1, Fig. 3) e recebe uma asserção do IdP (passo 3.2, Fig. 3). Note que a asserção é uma mensagem assinada pelo IdP indicando o serviço solicitado pelo Cliente.

Na etapa 4 da operação do FLAT, o Cliente encaminha a asserção recebida do IdP para o SP (passo 4.1, Fig. 3), que então fornece o serviço ao Cliente (passo 4.2, Fig. 3).

OPERAÇÃO($Cliente\ C$, IdP, SP)

1.1. $C \rightarrow IdP$: MAC($k_{IdP,C}$, n_C | SP | key_req)
2.1. $IdP \rightarrow SP$: n_{IdP} | $cert_{IdP}$
2.2. $SP \rightarrow IdP$: SIGN(S_{SP}, $cert_{SP}$ | n_{IdP} | n_{SP} | n'_{SP})
2.3. $IdP \rightarrow SP$: SIGN(S_{IdP}, ENC(P_{SP}, $k_{SP,C}$) | n_{SP} | n'_{IdP})
2.4. $SP \rightarrow IdP$: SIGN(S_{SP}, n'_{IdP})
1.2. $IdP \rightarrow C$: MAC($k_{IdP,C}$, ENC($k_{IdP,C}$, $k_{SP,C}$) | 'serv[1...n]' | n_C | n''_{IdP})
3.1. $C \rightarrow IdP$: MAC($k_{IdP,C}$, ENC($k_{IdP,C}$, assert_req('serv[i]')) | n''_{IdP} | n'_C)
3.2. $IdP \rightarrow C$: MAC($k_{IdP,C}$, ENC($k_{IdP,C}$, assert = SIGN(S_{IdP},'serv[i]' | n'_{SP}) | n'_C)
4.1. $C \rightarrow SP$: MAC($k_{SP,C}$, ENC($k_{SP,C}$, assert) | n''_C | n'_{SP})
4.2. $SP \rightarrow C$: MAC($k_{SP,C}$, ENC($k_{SP,C}$, serv[i]) | n''_C)

Símbolos :

| :	concatenação
MAC(k, m) :	MAC sobre m utilizando a chave k
$k_{X,Y}$:	chave simétrica compartilhada por X e Y
n_X :	*nonce* gerado por X
key_req :	requisição de chave
assert_req :	requisição de asserção
$cert_X$:	certificado de X
S_X :	chave privada de X
P_X :	chave pública de X
SIGN(k, m) :	assinatura sobre m utilizando a chave k
ENC(k, m) :	cifração sobre m utilizando a chave k
'serv[1...n]':	lista de serviços
'serv[i]':	descrição do serviço i
serv[i]:	serviço i

Figura 3: **Operação do FLAT.**

2.4. Implementação e Protótipo

Como prova de conceito da solução proposta, foi implementado um protótipo do FLAT utilizando a biblioteca criptográfica Relic [Aranha and Gouvêa] e a biblioteca de comunicação Arduino Wi-Fi [1]. O protótipo foi desenvolvido majoritariamente na linguagem C/C++ com alguns *scripts* em Python, considerando o cenário de uma catraca de pedágio/estacionamento.

Neste modelo, o Cliente é representado por um Arduino Due (Atmel 84 MHz, 96 kB SRAM, 512 kB *flash*). O SP é representado por um Raspberry Pi Zero W (Broadcom BCM2385 ARM1176JZF-S 1GHz, 512 MB RAM, 8GB microSD). O IdP, por sua vez, é representado por um laptop (Intel Core i7 2,7 GHz, 8GB RAM, 1TB HD). De forma geral, considera-se o dispositivo Cliente como restrito (recursos reduzidos), o SP como intermediário e o IdP como um dispositivo mais robusto.

No dispositivo Cliente são utilizadas apenas primitivas criptográficas simétricas (HMAC, AES) enquanto no SP e IdP são utilizadas primitivas simétricas e assimétricas (HMAC, AES, ECIES, ECDSA, certificados implícitos). Vídeos, textos explicativos, e demais informações sobre o FLAT podem ser encontradas na página: `https://sites.google.com/view/flat`.

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

3. Avaliação

Esta seção fornece uma avaliação em termos de comunicação, tempo de computação e tempo total de execução do protocolo FLAT em relação a um protocolo Baseline FIdM baseado no amplamente adotado Shibboleth [2].

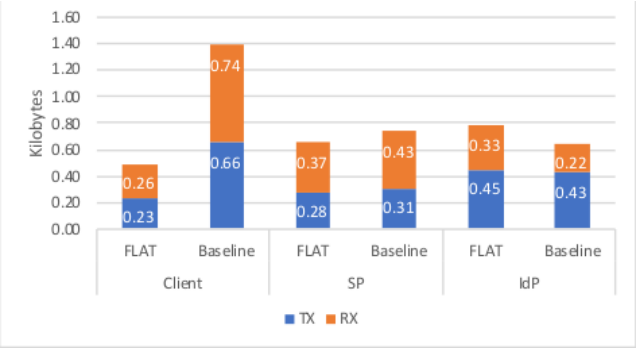


Figura 4: **Custos de comunicação.**

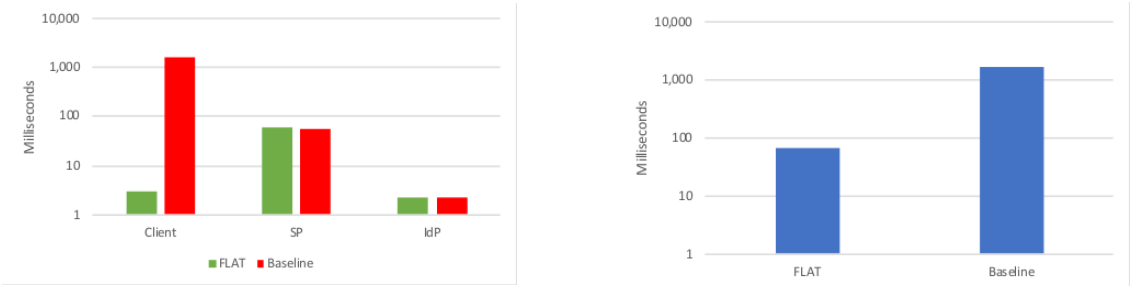
Comunicação. A Fig. 4 apresenta uma comparação dos custos de comunicação entre FLAT e Baseline, em que são mostrados o total de dados transmitidos (Tx) e recebidos (Rx) por cada entidade do protocolo (Cliente, SP e IdP). O Cliente FLAT é cerca de 65% mais eficiente em termos de dados transmitidos, recebidos e total de dados trocados. O SP do FLAT também é mais eficiente do que o SP da solução Baseline em termos de dados transmitidos (9%), recebidos (14%) e total de dados trocados (12%). Entretanto, considerando o IdP, a solução Baseline é cerca de 20% mais eficiente que o FLAT no total de dados transmitidos. De fato, para reduzir os custos de comunicação no Cliente e no SP, FLAT transfere a maior parte da carga de comunicação para o IdP, o que leva o IdP do FLAT a ser menos eficiente, mesmo com o uso de certificados implícitos. Apesar deste aspecto, no total de dados trocados o FLAT (considerando Cliente, IdP e SP) é cerca de 31% mais eficiente do que a solução Baseline.

Computação e Tempo Total de Execução. A Fig. 5a mostra os custos de computação relacionados a cada uma das entidades participantes do protocolo. O Cliente do FLAT é cerca de 520 vezes mais rápido do que o Cliente do Baseline. Entretanto, o IdP do Baseline é cerca de 2% mais rápido do que o do FLAT, enquanto o SP do Baseline também é mais rápido do que o SP do FLAT em cerca de 4%. Como mencionado anteriormente, FLAT delega mais atividades para IdP e SP de forma a reduzir a carga no dispositivo Cliente. Por outro lado, o Cliente do FLAT utiliza apenas primitivas simétricas, enquanto o Cliente do Baseline utiliza criptografia assimétrica, o que faz o Cliente do FLAT ser mais eficiente em termos de computação.

A Fig. 5b mostra uma comparação entre o tempo total de execução do FLAT e do Baseline. O tempo total para executar o FLAT é cerca de 4% do tempo de execução do Baseline. De fato, ao reduzir o tempo de computação no Cliente (dispositivo mais restrito da solução), FLAT também obteve uma redução no tempo de execução total do protocolo.

Segurança. FLAT foi concebido para fornecer autenticidade, confidencialidade, *liveness* e integridade na autenticação de dispositivos IoT. Em termos de autenticidade, são utilizados MACs (HMAC) e assinaturas digitais (ECDSA). Em termos de confidencialidade,

[2] `https://www.shibboleth.net/`



(a) Custos de computação. **(b) Tempo total de execução.**

Figura 5: **Custos de computação (a) e tempo total de execução (b).**

quando são enviadas informações sensíveis (como chaves, por exemplo), FLAT faz uso de cifração simétrica (AES) e assimétrica (ECIES). Em relação ao *liveness* são utilizados *nonces* de forma a evitar ataques de *replay*. Para garantir a integridade das mensagens são utilizados novamente MACs e assinaturas digitais, evitando-se ataques de modificação de mensagem. O uso de PUFs também protege contra ataques de manipulação em que há acesso físico aos dispositivos.

FLAT garante a autenticação das partes (IdP, SP e Cliente) por meio da verificação prévia das identidades através de certificados digitais e chaves simétricas, de forma a evitar ataques *Man-In-The-Middle* (MITM), que afetam a confidencialidade, autenticidade e potencialmente a integridade. Por ser um sistema FIdM, FLAT também protege a privacidade dos dados dos usuários/dispositivos, permitindo a limitação da quantidade de informação compartilhada e a redução do número de entidades com acesso aos dados.

4. Trabalhos Relacionados

Diferentemente da Internet tradicional, em que já existem soluções consolidadas em (F)IdM (SAML[3], Shibboleth, etc.), em IoT não há um consenso.

Hummen et al. propõem adaptações ao protocolo DTLS (como a pré-validação de certificados em um *gateway*) de forma a tornar possível o uso de certificados em dispositivos IoT restritos [Hummen et al. 2013]. O trabalho de Cirani et al. propõe o IoT-OAS, uma solução baseada em OAuth que transfere a lógica da autorização para um *gateway* externo [Cirani et al. 2015]. Domenech et al. propõem uma solução de autenticação e autorização baseada em SAML e XACML [Domenech et al. 2016]. O trabalho de Silva e Silva apresenta uma arquitetura com autorização baseada em XACML e RBAC/ABAC, e autenticação multi-fator utilizando SAML [Silva and Silva 2017]. Por outro lado, FLAT propõe uma solução especialmente modelada para IoT que possa ser executada mesmo em dispositivos restritos, sem o uso de *gateways* externos.

Witkovski et al. apresentam uma solução de autenticação para o contexto de uma *smart home* baseada em criptografia simétrica e chaves de sessão, de forma que técnicos de manutenção possam acessar dispositivos de forma segura [Witkovski et al. 2015]. Hong et al. também aborda um cenário específico de IoT, fornecendo uma solução de controle de acesso para IoT baseada em *Bluetooth Low Energy* (BLE) para *smart homes* e dispositivos pessoais [Hong et al. 2016]. FLAT, por sua vez, é uma solução concebida para ser mais abrangente, que possa ser aplicada a diferentes cenários IoT.

[3]https://tools.ietf.org/html/rfc7522

5. Conclusão

Este trabalho descreve o FLAT, um protocolo de autenticação federada especialmente modelado para IoT. Também é apresentada uma avaliação do protocolo proposto, em termos de comunicação, custos computacionais e segurança. Os resultados mostram que o dispositivo Cliente no FLAT é cerca de 520 vezes mais eficiente do que o protocolo Baseline em termos de tempo de computação. Além disso, em termos de total de dados trocados, FLAT é 31% mais eficiente do que o Baseline.

Este trabalho contempla uma parte essencial de um modelo (F)IdM para IoT: a autenticação. Os próximos passos são na direção de um modelo FIdM completo, abordando autenticação, autorização, descoberta de serviço e *tear-down*.

Referências

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.

Birrell, E. and Schneider, F. B. (2013). Federated Identity Management Systems: A Privacy-based Characterization. *IEEE Security & Privacy*, 11(5):36–48.

Brown, D. R., Gallant, R., and Vanstone, S. A. (2001). Provably Secure Implicit Certificate Schemes. In *FC'01*. 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.

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

Hummen, R., Ziegeldorf, J. H., Shafagh, H., Raza, S., and Wehrle, K. (2013). Towards Viable Certificate-based Authentication for the Internet of Things. In *Hot Topics on Wireless Network Security and Privacy*, pages 37–42. ACM.

Maler, E. and Reed, D. (2008). The Venn of Identity: Options and Issues in Federated Identity Management. *IEEE Security & Privacy*, 6(2).

Shim, S. S., Bhalla, G., and Pendyala, V. (2005). Federated Identity Management. *IEEE 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*. SBC.

Suh, G. E. and Devadas, S. (2007). Physical Unclonable Functions for Device Authentication and Secret Key Generation. In *44th DAC*. ACM.

Windley, P. J. (2005). *Digital Identity: Unmasking Identity Management Architecture (IMA)*. O'Reilly Media, Inc.

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.