Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Informática

Especificação Baseada no Padrão UPnP para Autenticação e Autorização de Usuários em Ambientes de Computação Pervasiva

Thiago Bruno Melo de Sales

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Engenharia de Software

Nome do Orientador
Angelo Perkusich e Hyggo Almeida

Campina Grande, Paraíba, Brasil

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

S163e
2010 Sales, Thiago Bruno Melo de.
Especificação baseada no padrão UPnP para autenticação e autorização de usuários em ambientes de computação pervasiva / Thiago Bruno Melo de Sales. ─ Campina Grande, 2010.
113 f. : il.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.
Referências.
Orientadores: Prof. Dr. Ângelo Perkusich, Prof. Dr. Hyggo Almeida.

1. Redes de Acesso Local. 2. Padrão UPnP. 3. Computação Pervasiva. 4. Autenticação e Controle de Acesso. I. Título.

CDU – 004.725.5(043)

"ESPECIFICAÇÃO BASEADA NO PADRÃO UPnP PARA AUTENTICAÇÃO E AUTORIZAÇÃO DE USUÁRIOS EM AMBIENTES PERVASIVOS"

THIAGO BRUNO MELO DE SALES

DISSERTAÇÃO APROVADA EM 10.03.2010

HYGGO OLIVEIRA DE ALMEIDA, D.Sc
Orientador(a)

ANGELO PERKUSICH, D.Sc
Orientador(a)

LEANDRO DIAS DA SILVA, D.Sc
Examinador(a)

JOSÉ SÉRGIO DA ROCHA NETO, D.Sc
Examinador(a)

CAMPINA GRANDE - PB

Resumo

Impulsionado pelo barateamento dos dispositivos portáteis, dos avanços das tecnologias de redes sem-fio e das técnicas de desenvolvimento de softwares para dispositivos com recursos limitados, o paradigma da Computação Pervasiva vem aos poucos se tornando realidade, vislumbrando um mundo onde serviços computacionais integram-se de forma transparente à vida das pessoas.

O padrão de conectividade UPnP mostra-se como uma solução adequada para a descoberta e a provisão de serviços pervasivos. Entretanto, o padrão não disponibiliza mecanismos de autenticação e autorização de usuários, tarefas essenciais para a construção de serviços personalizados e o controle de acesso à recursos em ambientes pervasivos. Por consequência, desenvolvedores de serviços UPnP recorrem a soluções de terceiros, o que naturalmente dificulta a interoperabilidade entre diferentes soluções.

É proposta no presente trabalho uma especificação baseada no padrão UPnP para a autenticação e a autorização de usuários em ambientes pervasivos. A especificação contempla uma arquitetura de conectividade para a disponibilização de perfis de usuários e mecanismos para a definição de políticas de controle de acesso à serviços pervasivos. Para ilustrar a aplicabilidade da especificação, são apresentados dois estudos de casos baseados na recomendação de conteúdos multimídia e no controle de acesso a dispositivos de rede.

Abstract

The paradigm of Pervasive Computing is slowly becoming a reality motivated by different aspects such as cheaper mobile devices, the wide dissemination of wireless network technologies and the techniques of software development for devices with restricted resources. In addition, this paradigm embraces a world where computing services are seamlessly integrated to the people's everyday lives.

The UPnP standard is a good candidate for device discovery and pervasive service provisioning. However, the standard does not address user authentication and authorization mechanisms, important features for defining and building customized services and access control to network resources. Therefore, UPnP service developers must cope with third party user authentication and authorization solutions, which turns interoperability a difficult process to be accomplished.

This work proposes a UPnP-based user authentication and authorization specification for pervasive computing environments. Such specification provides a connectivity architecture for user profile availability service, also allowing the definition of access control policies for secure UPnP services. Within this context, this work presents two case studies: one for multimedia content recommendation and another for access control to network devices.

Agradecimentos

Agradeço primeiramente a Deus, por me proporcionar a vida, os desafios, as dificuldades e a força para superar mais um obstáculo. À Chiara Lubich, minha mãe espiritual, pelo dom da Obra de Maria.

Aos meus pais, Marcelo e Socorro, pelo apoio substancial por tudo que se possa imaginar. Meu amor por vocês será eterno e sem vocês este trabalho não teria sido realizado. Aos meus irmãos Leandro e Júnior. Leo, te agradeço por tudo e sem as suas contribuições técnicas este trabalho não teria sido concluído no tempo esperado. Júnior, apesar da distância, seu apoio e encorajamento foi muito importante para a concretização deste trabalho. Agradeço também a todos da minha família, tios, primos e agregados.

À minha noiva (e futura esposa =]) Lívia, pelo apoio incondicional e paciência perante meus "stresses" e momentos em que eu não pude estar ao seu lado. Te amo ao ponto de dar a minha vida por você, pequena. À minha segunda família, meus pais do coração Bartolomeu e Bernadete, meus cunhados Back (vulgo Lucas =]) e Clarinha, pela confiança e o apoio durante estes anos. Aos meus avós e tios do coração, de Arapiraca, obrigado por tudo.

A todos da Obra de Maria, em especial ao Movimento Gen, pelas contínuas orações e unidade construída. A Rodrigo Apolinário (ou, "manezin" =]), sua futura esposa e minha futura madrinha Carol (ou "mariazinha" =]), Bruno "Sapé", Tiago "Mossoró", Enoque, Eugênio, Arthur, Popô, Wendel e todos os/as Gen de João Pessoa e Campina Grande, obrigado pela unidade e pelo Jesus em Meio construído nos últimos meses. Aos focolarinos Giovanni e Simone.

Aos meus velhos amigos de Maceió, pelo apoio constante: Zé, Aninha, Alisson, e Wesley. Aos velhos companheiros de apartamento, Edu e Guto.

Aos meus orientadores Hyggo e Angelo, pela contribuição substancial para a concretização deste trabalho, atenção e paciência perante as minhas dificuldades.

Ao meu companheiro de apartamento e de trabalho, Ivo, pelas contribuições para a concretização deste trabalho. Aos meus companheiros do laboratório Embedded: Vicente, Danilo, Maurílio, Von, Taciana, Bublitz, e aos velhos companheiros Glauber, Marcos Fábio e Diego Bizerra.

À CAPES, pelo apoio financeiro.

Conteúdo

Lista de Símbolos

3G - *3ª Geração*

3DES - *Triple Data Encryption Standard*

ABAC - *Atribute-Based Access Control*

ACL - *Access Control List*

AES - *Advanced Encryption Standard*

CA - *Autoridade Certificadora*

CUPS - *C Unix Printing System*

DAC - *Discritionary Access Control*

DCP - *Device Control Protocol*

DES - *Data Encryption Standard*

DHCP - *Dynamic Host Configuration Protocol*

DIDL - *Digital Item Declaration Language*

DLNA - *Digital Living Network Alliance*

GENA - *General Event Notification Architecture*

GPS - *Global Positioning System*

HVAC - *Heating, Ventilation and Air-Conditioning*

IPP - *Internet Printing Protocol*

MAC - *Mandatory Access Control*

PDA - *Personal Device Assistance*

RBAC - *Role-Based Access Control*

SOA - *Service Oriented Architecture*

SOAP - *Simple Object Access Protocol*

SPKI - *Simple Public Key Infrastructure*

SSDP - *Simple Service Discovery Protocol*

SSL - *Secure Socket Layer*

TF-IDF - *Term Frequency, Inverse Document Frequency*

TLS - *Transport Layer Security*

UPnP - *Universal Plug and Play*

UUID - *Universally Unique IDentifier*

Wi-Fi - *Wireless Fidelity*

Wi-Max - *Worldwide Interoperability for Microwave Access*

XACML - *eXtensible Access Control Markup Language*

Lista de Figuras

Lista de Tabelas

Lista de Códigos Fonte

Capítulo 1

Introdução

"The most profound technologies are those that disappear." (Mark Weiser, 1991).

Com o crescente uso de dispositivos móveis, tais como PDAs, *Smart Phones* e *Internet Tablets*, e o extraordinário avanço das tecnologias de comunicação sem-fio (Wi-Fi, Wi-Max, GPS e 3G), pode-se observar uma ampla disseminação das aplicações móveis e embarcadas nos últimos anos. Este cenário é reflexo da clara migração da computação baseada em *desktop* para a computação móvel, o que tem proporcionado à comunidade científica vários desafios, dentre eles o gerenciamento de energia, memória e processamento dos dispositivos portáteis [9] [87].

A competitividade entre os fabricantes de dispositivos móveis e entre as operadoras de telecomunicação, aliada à esta revolução tecnológica, tem possibilitado o barateamento e a popularização destes dispositivos. Segundo resultados da Pesquisa Nacional por Amostras de Domicílios, divulgados em setembro de 2008 pelo Instituto Brasileiro de Geografia e Estatística (IBGE), entre 2006 e 2007 houve um crescimento de 17,8% no número de domicílios que possuem somente celulares, o que corresponde a 17,6 milhões de celulares. Este crescimento é significativo quando comparado aos 3,66 milhões de dispositivos em 2001[1]. Em março de 2009, a Agência Nacional de Telecomunicações[2] confirmou que o Brasil atingiu a marca de 152 milhões de celulares em uso, algo inimaginável quando os primeiros celulares chegaram ao país.

[1] Ano em que o Pnda passou a incluir os celulares na pesquisa.

[2] `http://www.anatel.gov.br/Portal/exibirPortalInternet.do`

Incentivada pelo avanço das tecnologias de mobilidade, a computação pervasiva [85] vem se tornando realidade, vislumbrando um mundo onde dispositivos móveis executam aplicações inteligentes e integram-se de forma transparente à vida das pessoas, auxiliando na execução de suas tarefas. Uma funcionalidade essencial de toda aplicação pervasiva é disponibilizar recursos (informações e serviços) aos usuários no lugar certo e na hora certa. Para isto, informações de contexto deverão estar disponíveis às aplicações para que estas possam tomar decisões de forma transparente aos usuários. Como exemplo, considere um turista que acaba de chegar a uma cidade desconhecida por ele. Este poderia então ser informado sobre os principais pontos turísticos da cidade, considerando suas preferências e condições atuais do ambiente. As aplicações que fazem uso de informações de contexto são denominadas *aplicações sensíveis ao contexto*, ou *aplicações cientes do contexto* [53].

Os avanços do paradigma da computação pervasiva têm ampliado os desafios na concepção, no desenvolvimento e na implantação de softwares para ambientes inteligentes. Para que a computação pervasiva se torne realidade, a agregação de três tipos de tecnologias deve ser alcançada [52]: dispositivos baratos e com baixo consumo de energia; uma infraestrutura de rede sem-fio para a comunicação entre estes dispositivos; e aplicações pervasivas. A disseminação dos recursos de hardware para conectividade, como redes Wi-Fi, e a crescente produção de dispositivos móveis com suporte a estas tecnologias, têm possibilitado o desenvolvimento de soluções para dar suporte a este novo paradigma. Entretanto, o desenvolvimento de aplicações voltadas para este domínio sofre um impacto significativo na forma em que estas são projetadas e desenvolvidas, pois novos requisitos de softwares anteriormente desnecessários são agora primordiais para contemplar as ideias iniciais deste paradigma. Adaptabilidade e sensibilidade ao contexto [31] são alguns exemplos destes requisitos.

Ambientes de computação pervasiva possuem um alto grau de heterogeneidade devido à diversidade de recursos inseridos na rede e à dinamicidade de entrada e saída de usuários no ambiente. Desta forma, é de real valia proporcionar uma infraestrutura de hardware e software para abstrair a complexidade de baixo nível presente em aplicações de computação pervasiva, tais como a conectividade entre dispositivos com diferentes interfaces de comunicação, a coleta das informações oriundas do ambiente e dos usuários e os requisitos de segurança e controle de acesso aos recursos disponíveis na rede. Neste ponto, os *middlewares* [64] [45] aparecem como soluções verticais que objetivam acelerar o processo de

desenvolvimento de requisitos de software, disponibilizando um conjunto de ferramentas pré-desenvolvidas e testadas.

É dentro deste contexto que diversas iniciativas para permitir comunicação de forma transparente entre diferentes dispositivos foram propostos nos últimos anos, provendo a base para a construção de serviços pervasivos. Exemplos de arquiteturas de conectividade são: *Jini* [35], JXTA [29] e *Bluetooth*. Entretanto, o padrão de conectividade UPnP (Universal Plug and Play) [20] faz uso de protocolos padronizados e já consolidados da Internet, tais como HTTP [79], SOAP [10] e XML [30], permitindo a interoperabilidade entre diferentes dispositivos numa rede baseada no modelo *Plug and Play*[3]. Basicamente, o padrão UPnP estende este modelo para facilitar a instalação e a configuração de dispositivos heterogêneos (computadores, impressoras, roteadores, TVs, câmeras, lâmpadas etc) numa rede IP, sem a necessidade de configurá-los para possibilitar os processos de obtenção de endereços IP e de descoberta e invocação de serviços de rede.

UPnP tem sido utilizado em larga escala tanto em aplicações proprietárias quanto em aplicações de código aberto, tais como o BRisa [27], o Coherence[4] e o GUPnP[5]. Devido à grande disseminação do padrão UPnP, as duas das principais instituições sobre padronização de tecnologias na Internet (a *International Organization for Standardization* e a *International Electrotechnical Commission*) considerou o UPnP como a "principal tecnologia para os processos de descoberta e controle de dispositivos e serviços de rede" [72]. Nos documentos da série ISO/IEC 29341-1 está definida a arquitetura de conectividade P2P entre os diferentes tipos de dispositivos e serviços de rede.

1.1 Problemática

O padrão UPnP oferece uma arquitetura de rede aberta e distribuída para conectividade *ad hoc* entre diferentes dispositivos com diferentes interfaces de comunicação. Não obstante, desde a sua concepção em meados dos anos 90, o padrão UPnP tem como objetivo fundamental formar uma rede *ad hoc* e *aberta* para proporcionar descoberta automática de dispositivos e serviços, desconsiderando quaisquer informações dos usuários inseridos no ambiente.

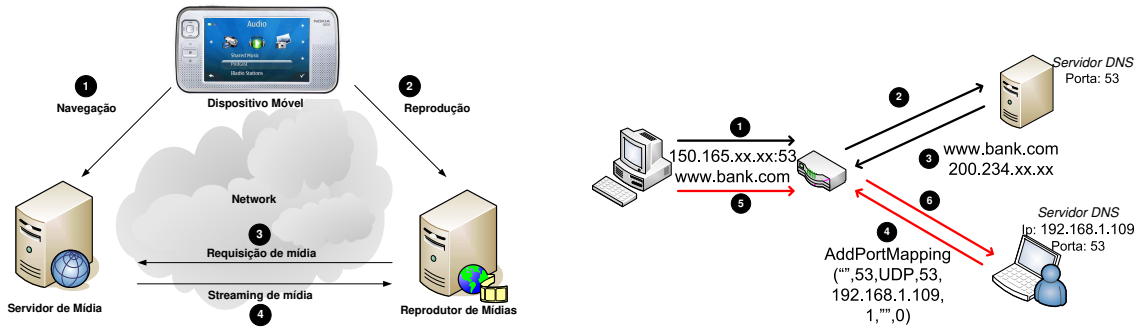
[3] Modelo para facilitar a instalação e a configuração de dispositivos periféricos.

[4] `http://coherence.beebits.net/`

[5] `http://www.gupnp.org/`

Como consequência, estas características do padrão UPnP trazem um impacto significativo em ambientes para a computação pervasiva. A sensibilidade ao contexto possibilita a autonomia dos serviços pervasivos baseando-se em informações de objetos inseridos no ambiente, e a heterogeneidade de usuários requer um mecanismo para controle de acesso destes serviços.

Como exemplo, considere o cenário ilustrado na Figura 1.1(a). Na especificação UPnP de áudio e vídeo [63] propõe-se uma abordagem para que dispositivos móveis possibilitem a navegação por conteúdos multimídia armazenados em um servidor de mídias (Etapa 1) e os reproduzem (Etapas 2, 3 e 4) num reprodutor de mídias. Todavia, o processo de navegação de mídias não faz uso de informações dos usuários e/ou do ambiente, o que permitiria, por exemplo, recomendar os conteúdos multimídia baseando-se nas preferências musicais dos usuários.



(a) Especificação UPnP de Áudio e Vídeo

(b) Ataque de *DNS Injection* na especificação UPnP IGD

Figura 1.1: Esquemas para aplicações baseadas no padrão UPnP.

A inexistência de informações dos usuários no padrão UPnP também traz alguns impactos com relação à segurança do padrão. Por exemplo, considere o cenário ilustrado na Figura 1.1(b), onde um usuário requisita uma *tradução de nomes* para o endereço www.bank.com (Etapas 1 e 2). O processo comum garante que o servidor DNS da rede privada responda com o endereço IP associado ao domínio solicitado (Etapa 3). Entretanto, um usuário com um programa malicioso pode requisitar um mapeamento de porta utilizando o serviço UPnP *addPortMapping* de um roteador com suporte a UPnP (*UPnP Internet Gateway Device*) [37], alterando todas as requisições com destino à porta 53 para um DNS que está sendo executado na máquina *192.168.1.109* (Etapa 4). Por conseguinte, todas as requisições subsequentes na porta 53 terão suas rotas modificadas, o que pode permitir diversos

tipos de ataques de rede, dentre eles o *DNS Injection* [50], retornando um endereço IP para uma máquina maliciosa na rede (Etapas 5 e 6).

Os cenários supracitados podem ser estendidos para outras especificações UPnP, tais como: adaptar automaticamente a intensidade de uma lâmpada [68] UPnP; ajustar a temperatura ambiente através dos dispositivos UPnP HVAC [74] (*Heating, Ventilation and Air-Conditioning*) de acordo com a preferência do usuário; controlar o acesso a impressoras [4] UPnP; entre outros.

De acordo com Vazquez [28], o padrão UPnP quando aplicado no contexto do paradigma da computação pervasiva carece de alguns fatores importantes. Dentre eles, destacam-se:

- *Modelos de Inferência*: não estão definidos, na atual arquitetura do padrão UPnP, métodos para possibilitar que dispositivos sejam capazes de realizar raciocínio na execução dos seus serviços e durante a fase de descoberta de serviços UPnP.

- *Sensibilidade ao Contexto*: dispositivos UPnP não fazem uso de informações de contexto. Proporcionar informações e serviços personalizados de acordo com o contexto limitam-se ao desenvolvimento de outras soluções proprietárias, tais como [62].

- *Autonomia*: como a arquitetura de conectividade UPnP define um modelo de requisição/resposta, dispositivos UPnP não podem proporcionar serviços com características de autonomia independentemente de uma requisição oriunda de outro dispositivo. Isto é, a execução de um serviço será realizada se, e somente se, outro dispositivo invocá-lo.

A diversidade de usuários que podem estar inseridos em ambientes de computação pervasiva também exige mecanismos de controle de acesso aos recursos disponíveis na rede. As especificações UPnP *Device Security* [21] e UPnP *Security Console* [22] podem ser utilizadas em conjunto para prover controle de acesso a serviços UPnP. Todavia, estas especificações não contemplam mecanismos para utilizar informações de contexto do usuário no processo de controle de acesso, dificultando a adoção de tais soluções em ambientes de computação pervasiva.

Finalmente, pode-se observar que o padrão UPnP carece de processos de autenticação e controle de acesso aos serviços que façam uso de informações dos usuários conectados na

rede. Estes mecanismos poderiam também dar suporte ao desenvolvimento de aplicações cientes de contexto, visto que os perfis dos usuários estariam disponibilizados na rede.

1.2 Objetivo

Este trabalho tem como objetivo definir, desenvolver e incorporar mecanismos de autenticação e autorização de usuários no nível de serviço no padrão UPnP para dar suporte à construção de serviços UPnP cientes de contexto. Estes mecanismos promoverão maior proatividade e segurança em aplicações UPnP a partir de informações dos usuários e do ambiente.

Mais especificamente, propõe-se a definição de uma especificação UPnP para dar suporte à identificação de usuários numa rede pervasiva, além de disponibilizar políticas de controle de acesso a serviços UPnP pervasivos. Denominada de UPnP-UP (Universal Plug and Play - User Profile), tal especificação propõe algumas mudanças para o padrão UPnP, porém, mantendo a compatibilidade com versões anteriores do referido padrão.

Para o processo de validação da especificação UPnP-UP, foi desenvolvido um sistema para recomendação de conteúdos multimídia e um mecanismo de controle de acesso a estes conteúdos. Como estudo de caso, utilizou-se o *framework* BRisa [27], por ser baseado em código-aberto e oferecer um conjunto de ferramentas para a construção de dispositivos e serviços baseados no padrão UPnP. Para que esta etapa fosse alcançada, fez-se necessário também realizar algumas modificações no referido *framework*, a fim de contemplar o gerenciamento de informações de autenticação dos usuários.

1.3 Relevância

Atualmente, os serviços disponibilizados através do padrão UPnP são totalmente abertos e não possuem nenhum mecanismo para identificar e reconhecer usuários conectados na rede. Na especificação do UPnP que trata sobre segurança, para os mecanismos de autenticação e autorização são consideradas apenas as informações dos dispositivos, o que não é suficiente para fornecer serviços UPnP personalizados e seguros em ambientes de computação pervasiva. Em tais ambientes, é centrado nas informações dos usuários que se espera disponibilizar serviços personalizados.

Desta forma, faz-se necessária a definição de uma arquitetura para proporcionar um modelo interoperável para a descrição e a aquisição de dados de usuários em redes UPnP, fornecendo assim um mecanismo para dar suporte à construção de serviços cientes de contexto e com suporte a controle de acesso. É importante também oferecer uma solução que mantenha a compatibilidade com os serviços UPnP já desenvolvidos e consolidados.

Portanto, este trabalho é relevante pois solucionará o problema de disponibilização de serviços UPnP com requisitos de controle de acesso e personalização para um usuário, de acordo com o perfil dele. Com efeito, novas aplicações baseadas no padrão UPnP poderão utilizar a extensão UPnP-UP como um mecanismo padrão para autenticação e autorização de usuários, proporcionando um modelo interoperável através da utilização das tecnologias já utilizadas no UPnP.

Finalmente, com a extensão UPnP-UP, pode-se vislumbrar novas pesquisas em diversas áreas de domínio da computação, tais como: privacidade em redes UPnP; definição do perfil de usuários para os diferentes domínios de dispositivos UPnP; personalização e proatividade de serviços UPnP; gerenciamento de conflitos de perfis de usuários em ambientes pervasivos e heterogêneos; especificação de pontos de controle UPnP com suporte a personalização e controle de acesso; entre outros.

1.4 Estrutura da Dissertação

O restante deste documento está estruturado da seguinte forma:

- No **Capítulo 2** estão descritos os conceitos envolvidos na solução deste trabalho. De forma mais precisa, são abordados os conceitos do padrão UPnP, apresentando os objetivos e as características para conectividade entre dispositivos de rede. Por fim, é feito um embasamento teórico dos principais conceitos de segurança e criptografia de dados, permitindo que o leitor possa compreender nos capítulos posteriores os requisitos utilizados para a solução do problema apresentado.

- No **Capítulo 3** é apresentada a especificação UPnP-UP, a qual visa possibilitar a concepção e o desenvolvimento de serviços UPnP seguros e com requisitos de personalização com base no perfil do usuário. É descrita a arquitetura de conectividade UPnP

para permitir a definição e a distribuição de políticas de controle de acesso, bem como o gerenciamento de perfis de usuários em redes UPnP.

- No **Capítulo 4** são discutidas algumas abordagens existentes, de forma geral, para autenticação e autorização em ambientes pervasivos. Em seguida, são apresentadas as soluções para autenticação e autorização em redes UPnP e discutidas as limitações de cada abordagem.

- No **Capítulo 5** são demonstrados os estudos de casos que implementam a especificação UPnP-UP. Desta forma, o leitor irá observar como a pesquisa pode ser concretizada e a sua relevância para o contexto de segurança e personalização de serviços em redes pervasivas utilizando o padrão de conectividade UPnP.

- No **Capítulo 6** são apresentadas as considerações finais e os trabalhos futuros, descrevendo as contribuições desta pesquisa, bem como os artigos publicados e o que se espera realizar para evoluir este trabalho.

Além dos capítulos descritos acima, nos Apêndices A e B estão descritas, com base no modelo do *UPnP Forum*, as especificações da arquitetura e do dispositivo propostos neste trabalho.

Capítulo 2

Fundamentação

Neste capítulo são apresentados os conceitos e as tecnologias utilizadas para a concepção da especificação UPnP-UP. Estes conceitos são considerados fundamentais para a compreensão do restante do trabalho.

2.1 Padrão de Conectividade UPnP

No padrão UPnP defini-se uma arquitetura para conectividade entre diferentes dispositivos com diferentes tipos de interfaces de comunicação. Foi projetado para proporcionar flexibilidade e simplicidade na descoberta de dispositivos e serviços em uma arquitetura de rede *ad hoc*, distribuída e aberta, a qual tem como base protocolos padrões da Internet, tais como o TCP/IP, HTTP e SOAP. O padrão foi inicialmente proposto pela Microsoft, em 1999, em conjunto com outras 28 empresas, incluindo Intel, Hewlett-Packard, Compaq e Samsung.

Atualmente, mais de 800 empresas fazem parte do *UPnP Forum*, o qual objetiva definir e publicar padrões, protocolos e modelos, permitindo a interoperabilidade entre diferentes dispositivos e serviços de rede em ambientes domésticos e corporativos. As empresas estão organizadas em comitês por áreas específicas, encarregadas de definir padrões para dispositivos e desenvolver implementações de testes que concretizem as especificações propostas. Os comitês existentes estão divididos nas seguintes áreas:

- Automação residencial e segurança
- Áudio e Vídeo

- Internet Gateway
- Dispositivos Móveis
- Eletrodomésticos

2.1.1 Componentes de uma rede UPnP

Estão definidos no padrão UPnP três componentes básicos: dispositivos, serviços e pontos de controle. Como ilustrado na Figura 2.1, os dispositivos baseados no padrão UPnP podem ser classificados em duas formas: *dispositivos controláveis* - aqueles que disponibilizam serviços UPnP - e os pontos de controle - aqueles que utilizam os serviços UPnP disponibilizados pelos dispositivos. No primeiro caso, o dispositivo exerce a função de um servidor, respondendo a requisições realizadas pelos pontos de controle. Atualmente, as principais especificações UPnP já desenvolvidas e consolidadas englobam desde dispositivos para *automação residencial* - aquecimento e climatização através de dispositivos HVAC (*Heating, Ventilation and Air-conditioner*) [74], iluminação de ambientes (*UPnP Lighting Controls*) [68] e monitoramento por câmeras de segurança [82] - a dispositivos para impressão [4] e *gateways* de rede.

Figura 2.1: Padrão de Conectividade UPnP.

Dispositivos UPnP

Um dispositivo UPnP pode disponibilizar serviços de rede e/ou compor outros dispositivos (*embedded devices*). Os dispositivos são agrupados de acordo com a semelhança dos serviços disponíveis. Como exemplo, um dispositivo relógio pode ser composto pelos

serviços *data_hora* e *cronômetro*. Um dispositivo rádio-relógio pode ser composto pelos serviços *sintonizar*, *data_hora* e *cronômetro* ou um serviço *sintonizador* e um dispositivo relógio (*embedded device*). O dispositivo que contém outros dispositivos aninhados e não está aninhado em outro dispositivo é denominado **dispositivo raíz**. Além disso, todo dispositivo UPnP disponibiliza um arquivo formatado em XML para fornecer as informações detalhadas para os pontos de controle, tais como fabricante, número serial e dispositivos aninhados.

Serviços UPnP

Os serviços UPnP representam as ações que um ponto de controle pode invocar. Os serviços são modelados através de variáveis de estados, os quais representam o estado atual do dispositivo. No exemplo do relógio, um serviço poderia ser modelado como uma variável de estado *current_time* e duas ações: *set_time* e *get_time*. Mudanças nos valores das variáveis de estados que modelam o dispositivo podem gerar eventos UPnP, os quais objetivam notificar outros dispositivos sobre a alteração destes valores. Como exemplo, o dispositivo relógio pode informar, a cada minuto passado, o valor atualizado do horário para os pontos de controle interessados neste evento. Todo dispositivo também fornece um arquivo formatado em XML para descrever os serviços e os respectivos parâmetros de entrada e saída, além das informações sobre as variáveis de estados que lançam eventos UPnP.

Ponto de Controle

Um ponto de controle é a entidade responsável por descobrir e invocar serviços UPnP. Os pontos de controle obtêm as informações dos dispositivos e dos respectivos serviços, envia solicitações de controle e inscrevem-se no servidor de eventos de um dispositivo UPnP.

2.1.2 Etapas de Conectividade UPnP

O mecanismo de conectividade UPnP é subdividido em seis etapas: *endereçamento, descoberta, descrição, controle, evento e apresentação*. Todas as etapas utilizam protocolos padronizados, amplamente utilizados e difundidos em redes locais e na Internet, o que garante a interoperabilidade entre diferentes dispositivos UPnP. Na Figura 2.2, ilustra-se a

pilha de protocolos utilizada no padrão UPnP.

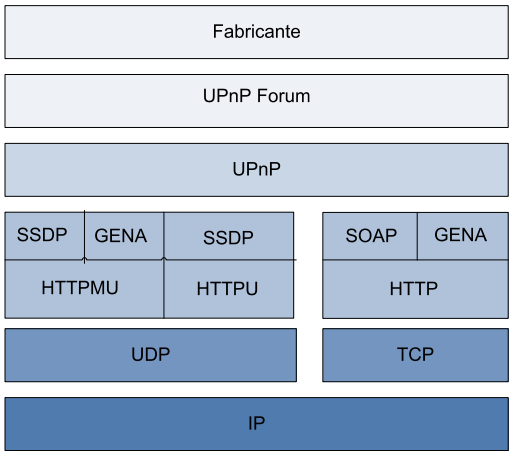


Figura 2.2: Pilha de protocolos UPnP.
Adaptada de [2]

Endereçamento

O primeiro passo do processo de conectividade UPnP (Etapa 0) está relacionado à maneira pela qual um dispositivo obtém um endereço IP. Cada dispositivo pode requisitar à rede um servidor DHCP (*Dynamic Host Configuration Protocol*) através do seu cliente DHCP, enviando mensagens de descoberta DHCP (*DHCP Discovery*). Se algum servidor DHCP estiver disponível na rede (enviando mensagens de descoberta DHCP - *DHCP Offer*), então a rede é *gerenciável* e o dispositivo deve usar o endereço IP atribuído pelo servidor DHCP. Caso contrário, a rede é *não gerenciável*, e o dispositivo deve selecionar um endereço IP através do mecanismo de Auto IP [39]. Nesta fase, um cliente DHCP faz uso do protocolo UDP para invocar a requisição de descoberta de um servidor DHCP.

Descoberta de dispositivos

Após obter um endereço IP, a Etapa 1 refere-se à descoberta de dispositivos em uma rede UPnP. Quando um dispositivo é adicionado à rede, o protocolo de descoberta UPnP permite que o dispositivo anuncie seus serviços a um ponto de controle. De forma similar, o protocolo de descoberta UPnP permite que um ponto de controle procure por dispositivos e os respectivos serviços disponíveis na rede. Basicamente, quando um dispositivo se conecta à rede, este envia mensagens de descoberta *multicast* (para o endereço e porta

239.255.255.250:1900) anunciando a sua presença na rede. Nesta fase, os protocolos SSDP (*Simple Service Discovery Protocol*) e HTTPMU são utilizados para a descoberta dos dispositivos UPnP.

Esta mensagem de descoberta contém informações tais como URL para a descrição detalhada sobre o dispositivo e o identificador do dispositivo, além de informação do tipo de dispositivo. O anúncio de descoberta segue um formato similar à Listagem 1:

```
1 NOTIFY * HTTP/1.1
2 HOST: 239.255.255.255.250:1900
3 CACHE-CONTROL: Tempo máximo de validade
4 LOCATION: URL para descrição do dispositivo
5 NTS: ssdp:alive
6 //outros campos foram omitidos
```

Listagem 1: Mensagem de notificação UPnP - Entrada do dispositivo.

Conforme a Listagem 1, a mensagem HTTP de anúncio está associada ao método *NOTIFY*, contendo ainda os seguintes campos de cabeçalho: *host*, o qual associa o endereço *multicast* reservado para o protocolo SSDP; *cache-control* refere-se ao tempo máximo (em segundos) em que aquele anúncio é válido na rede; e *location*, o qual armazena uma URL para a descrição mais detalhada sobre o dispositivo. Por outro lado, para descobrir outros dispositivos na rede, um ponto de controle deve enviar uma mensagem HTTP do tipo *M-SEARCH*. Ao receber esta notificação, dispositivos na rede devem retornar uma resposta UDP para o endereço e porta do dispositivo remetente, como descrito na Listagem 2.

```
1 HTTP/1.1 200 OK
2 CACHE-CONTROL: Tempo máximo de validade
3 LOCATION: URL da descricao do dispositivo
4 //outros campos foram omitidos
```

Listagem 2: Mensagem de resposta de descoberta M-SEARCH.

O anúncio de descoberta deve ser realizado pelo dispositivo raíz. Este envia 3 mensagens de descoberta, 2 mensagens para cada dispositivo aninhado e uma mensagem para cada serviço; ou seja, para um dado dispositivo raíz, com *d* dispositivos aninhados e *k* serviços distintos, são enviados $3 + 2d + k$ mensagens de anúncio UPnP.

Quando um dispositivo é removido da rede, o dispositivo deve enviar uma mensagem *NOTIFY multicast* do tipo *ssdp:byebye*. O formato da mensagem de remoção está descrito

na Listagem 3.

```
1 NOTIFY * HTTP/1.1
2 HOST: 239.255.255.255.250:1900
3 CACHE-CONTROL: Tempo máximo de validade
4 LOCATION: URL para descrição do dispositivo
5 NTS: ssdp:byebye
6 //outros campos foram omitidos
```

Listagem 3: Mensagem de notificação UPnP - Saída de um dispositivo.

Descrição

As informações adquiridas no processo de descoberta UPnP não são suficientes para que um ponto de controle utilize os serviços disponibilizados por outros dispositivos na rede. Desta forma, após a descoberta de dispositivos, na Etapa 2 um ponto de controle obtém a *descrição* do dispositivo e dos serviços que este disponibiliza. Esta descrição é adquirida a partir do campo *location* encapsulada nas mensagens de anúncio (*NOTIFY*) e resposta de descoberta UPnP (Listagens 1 e 2, respectivamente), formatada em XML.

A descrição UPnP para um dispositivo é subdividida em duas partes: *descrição do dispositivo* e *descrição dos serviços*. No primeiro caso, são fornecidas as informações específicas do fabricante do dispositivo tais como modelo, número de série, nome do fabricante, URLs para *websites* do fabricante, identificador universal do dispotivo (UDN), entre outros. Para cada serviço disponibilizado pelo dispositivo, pode-se listar as seguintes informações: tipo do serviço, nome, URL para descrição do serviço (que possibilita obter informações mais específicas do serviço em questão), URL para controle e URL para evento. A descrição é recuperada por um ponto de controle através de uma requisição *GET* do HTTP na URL fornecida na mensagem de descoberta UPnP. De forma resumida, na Listagem 4 está descrito o conteúdo XML de um dispositivo UPnP.

```
1 <root xmlns="urn:schemas-upnp-org:device-1-0">
2  <!-- basic UPnP description fields were omitted -->
3   <device>
4    <deviceType>
5     urn:schemas-upnp-org:device:Relogio:1
6    </deviceType>
7    <friendlyName>Relógio Genérico</friendlyName>
```

```
8    <manufacturer>Oriente</manufacturer>
9    <manufacturerURL>http://www.orientnet.com.br/<manufacturerURL>
10   <modelDescription>Relógio Oriente novo modelo.</modelDescription>
11   <modelNumber>134332765</modelNumber>
12   <UDN>uuid:005c70d3-bb6c-4b1d-b3bd-ff9cf6ffb385</UDN>
13   <!-- outras informações omitidas -->
14   <serviceList>
15    <service>
16     <serviceType>
17      urn:schemas-upnp-org:service:Time:1
18     </serviceType>
19     <serviceId>
20      urn:upnp-org:serviceId:11
21     </serviceId>
22     <SCPDURL>/Clock/clock-time.wsdl</SCPDURL>
23     <controlURL>/Clock/clock-time/control</controlURL>
24     <eventURL>/Clock/clock-time/event</eventURL>
25    </service>
26    <service>
27    </serviceList>
28    <presentationURL>URL para apresentação do dispositivo</
          presentationURL>
29  </device>
30 </root>
```

Listagem 4: Descrição do dispositivo Relógio.

Controle

Após obter a descrição detalhada do dispositivo e dos respectivos serviços, um ponto de controle faz uso dos serviços através de chamadas de métodos remotos via *Web Services* baseado no protocolo SOAP (*Simple Object Access Protocol*) [10]. Na Etapa 3, fase de controle, o processo de conectividade permite que um ponto de controle requisite remotamente os serviços de um dispositivo enviando mensagens de controle para as URLs dos serviços obtidos na etapa de descoberta. Para invocar um serviço UPnP, um ponto de controle deve enviar uma mensagem HTTP do tipo POST no formato descrito na Listagem 5:

```
1 POST url de controle HTTP/1.1
```

```
2 HOST: servidor:porta
3 CONTENT-LENGTH: bytes do corpo
4 CONTENT-TYPE: text/xml; charset="utf-8"
5 <!-- identificador do serviço e da ação invocada -->
6 SOAP-ACTION: urn:schemas:upnp-org:service:serviceType:v#actionName
7 <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
8             s:encodingStyle=" http://schemas.xmlsoap.org/soap/encoding/">
9         <s:Body>
10                <!-- Informações de parâmetros do serviço executado -->
11                <u:actionName xmlns:u="urn:schemas upnp org:service:
                      serviceType:v">
12                      <argumentName>valor</argumentName>
13                </u:actionName>
14        </s:Body>
15 </s:Envelope>
```

Listing 5: Mensagem de controle UPnP.

Na linha 6 da Listagem 5 está definido o serviço e o nome da ação que está sendo requisitado, e na linha 11 o parâmetro de entrada desta ação. A resposta desta requisição pode conter o valor de retorno do serviço ou informações sobre erros ocasionados por contextos diversos, como parâmetros inválidos ou ação inexistente.

Evento UPnP

A ação de invocar um serviço em uma rede UPnP poderá ocasionar mudanças nas variáveis que modelam o estado deste serviço em tempo de execução. Por exemplo, uma chamada remota ao serviço *"Play"* da especificação UPnP de áudio e vídeo modificaria o estado da mídia de *"pause"* para *"reproduzindo"*. Através de mensagens de evento (Etapa 4) realizadas pelos serviços, um ponto de controle pode ser informado sobre estas mudanças e, a partir destes eventos, atualizar as variáveis locais correlacionadas a este novo estado.

As notificações são realizadas através do protocolo GENA (*General Event Notification Architecture*) e os dados são expressos em XML. Pontos de controle que estiverem interessados em mensagens de eventos devem registrar-se ao dispositivo alvo através de mensagens HTTP do tipo *SUBSCRIBE*. Neste momento, o dispositivo envia mensagens de evento ao ponto de controle contendo os nomes e os valores de todas as variáveis de estados. Para

manter a inscrição ativa, pontos de controle devem renovar a inscrição antes que ela expire. Este mecanismo permite que se um ponto de controle desconectar da rede e não solicitar explicitamente o cancelamento do registro do evento através de mensagens *UNSUBSCRIBE*, a rede UPnP permaneça consistente.

Apresentação

Após a descoberta de dispositivos e serviços na rede, um ponto de controle pode utilizar a etapa de apresentação (Etapa 5) para acessar uma página *Web* do dispositivo alvo, sendo possível carregar via navegador Web os serviços oferecidos pelo dispositivo, além de controlá-lo e obter informações de *status* do dispositivo. O nível de interação depende unicamente das características da página, a qual deve ser compatível com HTML 3.0 ou posterior e deve levar em consideração as características dos *browsers* onde serão executadas.

2.1.3 Abordagens Existentes

Há diversas implementações do padrão UPnP, em diferentes linguagens de programação. As implementações descritas abaixo buscam abstrair os detalhes dos protocolos de rede apresentados na seção anterior, permitindo que desenvolvedores construam dispositivos UPnP focando apenas nos requisitos necessários às aplicações.

BRisa

O *framework* BRisa[27] provê suporte ao desenvolvimento de soluções baseadas no padrão UPnP e tem sido desenvolvido no Laboratório de Sistemas Embarcados e Computação Pervasiva[1] da Universidade Federal de Campina Grande. O *framework* possibilita a construção de dispositivos UPnP e pontos de controle, abstraindo do desenvolvedor a complexidade dos passos de conectividade UPnP. O *framework* oferece os seguintes requisitos:

- Definição e geração da descrição do dispositivo e dos serviços: o *framework* oferece uma API para definir as propriedades de um dispositivo, tais como nome e número serial, e os serviços com os respectivos parâmetros de entrada, saída e geração de eventos UPnP. Automaticamente, o BRisa constrói o conteúdo XML da descrição do

[1] http://embedded.ufcg.edu.br/

dispositivo e do serviço com base nas informações definidas através da linguagem Python.

- API para a construção de pontos de controle: o *framework* disponibiliza uma API para a construção de pontos de controle, oferecendo os métodos necessários para a descoberta e invocação de serviços UPnP, bem como os mecanismos para registrar-se a eventos UPnP de entrada e saída de dispositivos.

- BRisa Configuration Tool: esta ferramenta administrativa permite que desenvolvedores configurem o *framework* através de linha de comando.

Como estudos de caso, o *framework* disponibiliza também dois dispositivos UPnP. O *BRisa Media Server* e o *BRisa Media Renderer*, os quais implementam a especificação UPnP A/V. No primeiro caso, o dispositivo define uma arquitetura baseada em *plugins*, proporcionando extensibilidade na recuperação de conteúdos multimídia de diferentes fontes de dados, os quais podem ser reproduzidos no dispositivo *BRisa Media Renderer*.

No contexto deste trabalho, o *framework* BRisa foi utilizado para dar suporte na definição da especificação UPnP-UP e na construção dos estudos de caso apresentados no Capítulo 5.

Coherence

O outro exemplo é o *framework* Coherence, o qual é baseado no padrão DLNA (*Digital Living Network Alliance*) [2] que tem como base a especificação UPnP A/V. Coherence foi escrito em Python, é baseado em código-aberto e possui licença MIT. O *framework* também oferece um dispositivo Media Server e um dispositivo Media Renderer, fazendo uso de diversas APIs consolidadas para o processamento de conteúdos multimídia, tais como GStreamer e Rhythmbox e tem sido utilizado em diversos dispositivos já existentes no mercado, tais como o XBox 360, TV Sony Bravia e Nokia N800.

GUPnP

O *framework* GUPnP é baseado em código aberto e oferece uma API escrita em C (GObject) para a construção de dispositivos e pontos de controle UPnP. Apesar de não disponibi-

[2] http://www.dlna.org/home

lizar implementações de exemplo, o *framework* oferece um conjunto de APIs para o desenvolvimento de aplicações UPnP, como aquelas baseadas em áudio e vídeo. Além disso, o *framework* disponibiliza um conjunto de *widgets* baseado em GTK+ para a visualização dos dispositivos UPnP descobertos na rede pelo GUPnPControlPoint.

O *framework* é baseado no SDK *libupnp*[3], descontinuado desde 2006 e estendido por Michael Pfeiffer na definição da biblioteca *pupnp* (*Portable SDK for UPnP Devices*).

2.2 Segurança em Redes de Computadores

Nesta seção são apresentos os principais conceitos envolvendo os modelos e técnicas de criptografias e ataques de programas maliciosos em redes de computadores. Devido ao vasto conteúdo e o disseminado conhecimento nestas áreas, esta seção limita-se a discutir apenas os conceitos necessários para uma melhor compreensão do contexto de segurança aplicado ao padrão UPnP.

2.2.1 Criptografia de Dados

Criptografia é a ciência de utilizar técnicas de transformação de dados [66] para torná-los ilegíveis àquelas entidades que não fazem parte do canal de comunicação entre duas ou mais entidades. Através do uso de operações matemáticas, é possível cifrar os dados para ocultar o significado e decrifá-los para obter o significado novamente.

O uso das técnicas de criptografia de dados tem proporcionado diversos avanços em diferentes áreas da computação. Cenários de aplicações englobam desde a possibilidade de realizar transações bancárias de modo seguro e confortável a enviar documentos confidenciais nas áreas diplomáticas e militares entre governos. As técnicas de criptografia oferecem cinco tipos de serviços básicos:

[3] `http://upnp.sourceforge.net/`

Tabela 2.1: **Serviços oferecidos pela criptografia de dados.**

Serviço	Descrição
Integridade	Garante que a informação não foi alterada.
Controle de acesso	Garante que o acesso ao conteúdo da mensagem poderá ser realizada apenas por entidades autorizadas.
Autenticidade de origem	Garante a identidade do emissor da mensagem.
Não-repudiação	Previne que alguma entidade negue o envio/recebimento de uma mensagem.
Confidencialidade (ou sigilo)	Impede que entidades não autorizadas tenham acesso ao conteúdo original da mensagem, garantindo apenas que a origem e o destino possuam conhecimento deste conteúdo.

As características apresentadas acima são de suma importância em vários cenários de aplicações. No contexto de comércio eletrônico, por exemplo: a informação sobre o valor de uma transação não pode ser alterada (integridade); somente o cliente e a loja virtual podem ter acesso à transação (controle de acesso); deve-se garantir a identidade do cliente (autenticidade); o cliente tem como provar o pagamento e a loja virtual não tem como negar o recebimento da transação (não-repúdio); e finalmente, o conhecimento do conteúdo da transação fica restrito aos envolvidos (confidencialidade).

Existem diversos algoritmos de criptografia e os mais comuns utilizam o conceito de "chaves" para cifrar e decifrar uma mensagem. Segundo [70], existem várias técnicas criptográficas baseadas em chaves, divididas em duas categorias: criptografia simétrica (ou de chaves privadas) e criptografia assimétrica (ou de chaves públicas). A seguir, estão descritas cada uma destas categorias.

Criptografia Simétrica

O ciframento de uma mensagem baseia-se em dois componentes: um algoritmo e uma chave. Um algoritmo é uma transformação matemática responsável por converter uma mensagem em claro em uma mensagem cifrada e virce-versa. Já a chave é uma cadeia aleatória de bits

utilizada em conjunto com o algoritmo. A criptografia simétrica utiliza apenas uma única chave para cifrar e decifrar uma mensagem. Como ilustrado na Figura 2.3, um usuário *A* utiliza um algoritmo de ciframento e uma chave para transformar a mensagem em claro em texto cifrado e enviar ao usuário *B*. Ao receber o texto cifrado, o usuário *B* utiliza a mesma chave (e o mesmo algoritmo) para decifrar o conteúdo e obter o texto em claro novamente.

Figura 2.3: Uso da criptografia de chaves simétricas.

Este método é aplicado apenas para garantir a confidencialidade dos dados, visto que apenas os proprietários da chave poderão decifrá-los. No mundo real, este método de criptografia pode ser utilizado para cifrar dados de uma base de dados e/ou arquivos em um sistema com múltiplos usuários, garantindo que estas informações permanecerão confidenciais.

Apesar da simplicidade oferecida pela criptografia simétrica, este modelo apresenta algumas desvantagens:

- A criptografia simétrica não garante a identidade de quem enviou ou recebeu a mensagem (autenticidade e não-repudiação);
- A chave deve ser trocada entre as entidades comunicantes e ser mantida em sigilo, o que nem sempre pode ser garantido;
- Não há estabelecimento de um canal seguro[4] entre as entidades comunicantes para realizar a negociação da chave (apenas se duas pessoas se encontrarem pessoalmente, por exemplo).

Existem diversas implementações deste tipo de algoritmo, tais como: DES (Data Encryption Standard) [43], 3DES (Triple Data Encryption Standard) [41], AES (Advanced Encryption Standard) [16] e Blowfish [58]. Em [3] são apresentadas as diferenças entre os principais algoritmos de criptografia simétrica e um estudo de desempenho de cada algoritmo.

[4]Desconsiderando os métodos existentes para assegurar este procedimento.

Criptografia Assimétrica

Ao contrário da criptografia simétrica, a criptografia assimétrica [18][57] faz uso de duas chaves matematicamente relacionadas: uma chave privada ($K-$) e uma chave pública ($K+$). Qualquer uma das chaves pode ser utilizada para cifrar uma mensagem e a outra para decifrá-la. A chave privada deve ser mantida em sigilo e a chave pública pode ser disponibilizada livremente às entidades comunicantes. Como ilustrado na Figura 2.4, um usuário *A* criptografa a mensagem para o usuário *B* utilizando a chave pública de *B* (K_B+). Ao receber a mensagem, *B* obtém a mensagem em claro utilizando a chave privada correspondente (K_B-).

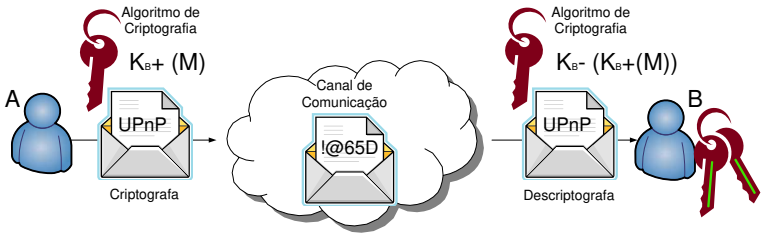


Figura 2.4: Uso da criptografia de chaves assimétricas.

Através desta abordagem, enquanto a chave privada estiver em sigilo, uma entidade poderá receber mensagens secretas sem o risco da confidencialidade desta mensagem ser violada. Por outro lado, algoritmos de criptografia assimétrica [47] normalmente demandam alto poder computacional se comparado às abordagens simétricas. Dentre os algoritmos de criptografia assimétrica mais conhecidos, encontram-se: RSA [65] e Curvas Elípticas [46].

No contexto deste trabalho, os algoritmos de criptografia apresentados são utilizados para trafegar informações confidenciais dos usuários (ver Seção 3.6). Em um modelo híbrido, algoritmos de criptografia assimétrica são utilizados para estabelecer um canal seguro de comunicação (*handshake*), permitindo o compartilhamento de chaves baseadas nos algoritmos de criptografia simétrica. Esta abordagem permite que dispositivos com recursos limitados, por exemplo, enviem mensagens criptografadas com chaves simétricas, a qual exige menor poder computacional. Este mesmo processo acontece com os principais protocolos de segurança da Internet, tais como SSL (Secure Socket Layer) e TLS (Transport Layer Security) [17].

2.2.2 Criptografia de Hash

Uma função de *hash* [34] H permite transformar uma mensagem *m* de qualquer tamanho em um identificador digital de tamanho fixo *m'*, chamado de valor de *hash*. Uma vez obtido o valor *m'* para uma mensagem *m*, é computacionalmente inviável fazer o processo inverso, ou seja, encontrar o valor *m* tal que $H(m) = m'$. Desta forma, uma função de *hash* tem a propriedade de ser uma função "one way". Além disso, este tipo de função deve oferecer as seguintes características:

- H é relativamente fácil de ser computado, para qualquer valor de *m*;
- H é livre de colisão. Ou seja, para quaisquer duas mensagens *m* e *M*, com $m \neq M$, temos $H(m) \neq H(M)$.

Algoritmos de *hash* são amplamente utilizados em diversos contextos dos sistemas de informação, tais como em indexação em tabelas de *hash* [81], detecção de dados duplicados e arquivos corrompidos etc. Os algoritmos de *hash* também podem ser utilizados em processos de autenticação, tais como assinaturas digitais, verificações de senhas e cadeias de *hash* (One-Way chains) [48]. Neste último caso, uma função de *hash* H é aplicada *n* vezes a uma cadeia de caracteres *m*. O valor inicial $m_0 = m$ é mantido em sigilo e é denominado *anchor*. Uma cadeia de *hash* de *n* elementos possui as seguintes propriedades:

- Para $1 \leq i \leq n$, $m_i = H_i(m)$ é o valor de *hash* da posição *i* da cadeia;
- $x = H_n(m_{n-1})$ representa o último valor da cadeia;
- Para provar a identidade de m_i, uma entidade precisa possuir m_{i-1}.

Como exemplo, para $n = 1000$, um servidor inicialmente armazena m_{1000} para um dado usuário. Na primeira autenticação, o usuário deve enviar m_{999} para o servidor. O servidor verifica se $H(m_{999}) = m_{1000}$. Caso afirmativo, o usuário está autenticado e o servidor armazena a última posição recebida pelo usuário, neste caso, m_{999}. Para as próximas requisições de autenticação, usuário envia m_{998}, repetindo o processo de verificação.

Neste trabalho, as cadeias de *hash* serão utilizadas como base para autenticação de usuários em um processo estabelecimento de sessão entre pontos de controle e dispositivos

UPnP (ver Seção 3.6.2). Como este modelo de autenticação apresenta uma maneira eficiente no processo de verificação, este mecanismo é amplamente utilizado em diversos domínios de aplicações, tais como em protocolos de segurança em redes de sensores sem-fio [59] [51] e melhorias no armazenamento e acesso a dados [11], [38], [71].

2.2.3 Assinatura e Certificação Digital

Uma assinatura digital [55] é um modelo de autenticação de informações digitais tipicamente análogo à assinatura física em papel. Através de uma assinatura digital, pode-se provar que uma mensagem foi realmente enviada pelo emissor (não-repúdio), o receptor pode confirmar que a assinatura foi feita pelo emissor (autenticidade) e que qualquer alteração da mensagem faz com que a assinatura digital não corresponda mais ao documento (integridade). Basicamente, uma assinatura digital é construída a partir da combinação dos algoritmos de criptografia assimétrica e das funções de *hash*.

De acordo com a Figura 2.5, para que um usuário *A* envie uma mensagem *m* assinada digitalmente ao usuário *B*, aplica-se uma função de *hash* à mensagem original *m*, gerando *m'* e, em seguida, cifra-se *m'* utilizando a chave privada do emissor, resultando na assinatura digital. A mensagem e a assinatura digital são enviadas ao receptor através de um canal de comunicação.

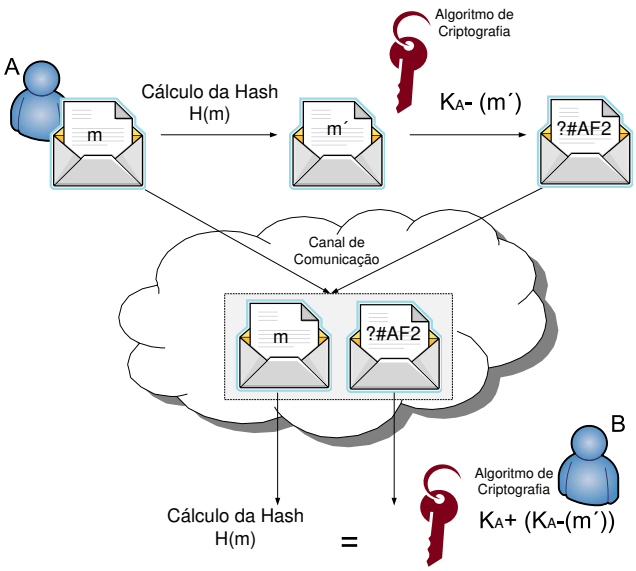


Figura 2.5: Geração de Assinaturas Digitais.

Para comprovar uma assinatura digital é necessário realizar dois procedimentos: calcular o valor de *hash* da mensagem *m* (gerando *m'*) e decifrar a assinatura com a chave pública do emissor (resultando em *m'*). Se forem iguais, a assinatura está correta, o que significa que foi gerada pela chave privada corresponde à chave pública utilizada na verificação e que o documento está íntegro. Caso sejam diferentes, a assinatura está incorreta, o que significa que pode ter havido alterações no documento ou na assinatura do documento.

O uso das funções de *hash* pode contribuir para que o processamento da assinatura digital não seja comprometido devido à lentidão dos algoritmos de criptografia assimétrica, consumindo um tempo baixo e uniforme, independente do tamanho do documento a ser assinado. Ademais, estas funções garantem a integridade da mensagem, visto que a alteração em um bit da mensagem irá gerar outro valor de *hash*. A criptografia assimétrica garante também a autenticidade e a não-repudiação do emissor, pois apenas ele é capaz de cifrar a mensagem com a chave privada correspondente à chave pública utilizada pelo receptor para checar a autenticidade da mensagem.

Os mecanismos oferecidos pela assinatura digital é uma forma eficaz de garantir a autoria de documentos eletrônicos. Em agosto de 2001, a Medida Provisória 2.200 [12] garantiu a validade jurídica de documentos eletrônicos e a utilização de certificados digitais para atribuir autenticidade e integridade aos documentos. Este fato tornou a assinatura digital um instrumento válido juridicamente.

Certificado Digital e Autoridades Certificadoras

Com a disseminação das assinaturas digitais, o gerenciamento das chaves de criptografia passa a ter dois procedimentos: primeiro, deve-se localizar a chave pública da entidade que se deseja comunicar e, segundo, deve-se obter uma garantia de que a chave pública obtida é, de fato, da entidade alvo.

Um certificado digital é um documento eletrônico assinado digitalmente e objetiva associar uma entidade a uma chave pública. Um certificado digital geralmente está no formato padrão ITU X.509 [24] e normalmente inclui informações como: nome da entidade a ser associada a chave pública, período de validade do certificado, chave pública, nome e assinatura da entidade que emitiu o certificado.

A entidade responsável pela emissão dos certificados digitais é denominada Autoridade

Certificadora (CA), entidade de confiança que funciona como cartório eletrônico. Desta forma, pela assinatura da chave pública e das informações sobre um usuário *A*, por exemplo, a CA garante que as informações sobre *A* está correta e que a chave pública em questão pertence à *A*. Um usuário *B* confere a assinatura da CA e então utiliza a chave pública disponível na certificação digital, seguro de que esta pertence à *A*.

Neste trabalho, os mecanismos de criptografias e assinaturas digitais permitem que usuários disponibilizem informações pessoais aos dispositivos UPnP assegurando as propriedades básicas apresentadas na Tabela 2.1. Ademais, certificações digitais permitem que dispositivos UPnP possam confiar nas informações recebidas por outros dispositivos registrados em CAs de confiança.

2.2.4 Autorização e Controle de Acesso

Um dos requisitos indispensáveis na área da segurança da informação é o processo de determinar se um sujeito (usuário, aplicação, processo etc) tem permissão de acessar um recurso (arquivo, serviço, um componente do sistema etc) com base em políticas de acesso predeterminadas. Define-se política como sendo o conjunto de regras que visa determinar o comportamento da rede ou de um sistema em diferentes condições [49]. O controle de acesso [5] é composto pelos processos de autenticação, autorização e auditoria (*accounting*). Após o estabelecimento da autenticação, o qual identifica quem acessa o recurso, a autorização determina o que o sujeito autenticado pode fazer e, posteriormente, a auditoria informa o que este sujeito realizou. O processo da auditoria está fora do escopo deste trabalhado. O leitor pode obter mais informações em [56].

Modelos de Controle de Acesso

Antes de apresentar os principais modelos de controle de acesso, é importante delinear as categorias que estes modelos estão conceitualmente inseridos [14]:

- Controle de Acesso Discricionário (DAC): é uma categoria de controle de acesso determinada pelo proprietário do recurso. O proprietário decide qual sujeito tem a permissão de acesso ao recurso e qual privilégio o sujeito possui. A Lista de Controle

de Acesso (ACL) [25] é o modelo mais comum da categoria DAC. Esta categoria é também utilizada nos principais sistemas operacionais, como Unix e Windows;

- Controle de Acesso Obrigatório (MAC): o controle de acesso é determinado pelo sistema e não pelo proprietário do recurso. Ou seja, os sujeitos não são considerados proprietários e, então, não podem gerenciar (atribuir permissões, por exemplo) os recursos disponíveis.

O modelo de controle de acesso mais conhecido pertencente à categoria DAC é a Lista de Controle de Acesso, normalmente utilizada em sistemas operacionais. Esta lista contém os identificadores dos usuários e as respectivas permissões de acesso, tais como "ler", "escrever" ou "executar". Apesar da facilidade de implementação, as ACLs possuem a desvantagem de escalabilidade enquanto o número de identificadores na lista e o número de usuários aumentam, tornando-se difícil mantê-las e gerenciá-las.

No modelo de controle de acesso baseado em papéis (RBAC) [15] um sujeito será comparado a um conjunto de regras predefinidas, permitindo o acesso de acordo com o papel que este representa no sistema. Conceitualmente, um papel pode ser um cargo ou uma função dentro de uma organização, que representa a autoridade necessária para conduzir as atividades associadas [23]. Como ilustrado na Figura 2.6, os sujeitos não estão relacionados diretamente com as permissões, o que facilita a definição das políticas de controle de acesso. Ao contrário da ACL, remover um sujeito do sistema não exige a remoção de todas as relações identificador do sujeito/permissão. Necessita-se apenas desassociá-lo do sistema. A flexibilidade também é percebida quando novos usuários são registrados e lhes são atribuídos os papéis no sistema, em vez de permissões diretas aos recursos disponíveis.

O modelo RBAC visa simplificar o gerenciamento dos sujeitos através de agrupamento com perfis similares, formando papéis e relacionando cada papel com as permissões de acesso. Associar uma permissão a um papel significa permitir ou negar o acesso a um conjunto de sujeitos associados a este papel. Apesar de ser amplamente utilizado, este modelo apresenta limitações importantes: (1) interoperabilidade, visto que o significado de um papel em um ambiente administrativo (casa, trabalho etc) pode ser completamente diferente em outros ambientes; e (2) "explosão de papéis", onde a quantidade de papéis pode ser muito grande ao ponto de ser maior que a quantidade de sujeitos envolvidos.

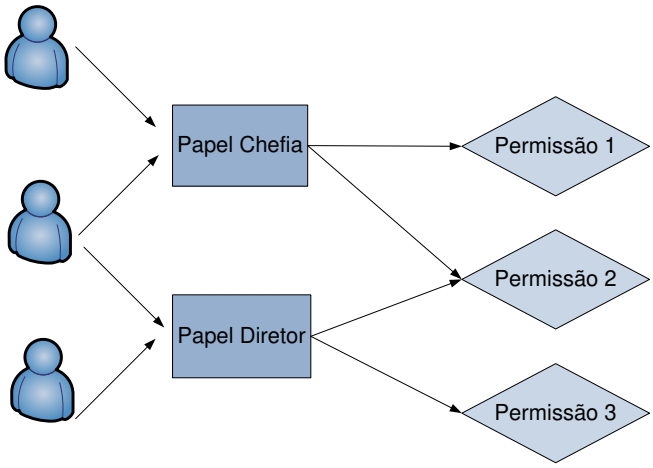


Figura 2.6: Diagrama representando o modelo RBAC.
Adaptada de [83]

O modelo de controle de acesso baseado em atributo (ABAC) [88], em contrapartida, faz uso das informações dos próprios sujeitos que pretendem realizar alguma operação, o que proporciona um modelo flexível e refinado para a definição de políticas de controle de acesso. Como descrito na Tabela 2.2, a ideia básica é modelar um sujeito que deseja realizar uma ação em um dado recurso e em um determinado contexto ou ambiente. Na Tabela 2.3, são descritos alguns exemplos da aplicabilidade das políticas baseadas neste modelo, o qual é a base para a definição de políticas de controle de acesso em diversos projetos, como os apresentados em [84][13][7].

Tabela 2.2: **Controle de Acesso Baseado em Atributos**

Sujeito	Ação	Recurso	Contexto ou Ambiente
Um sujeito...	...deseja realizar alguma ação...	...em um dado recurso...	...e em um dado contexto ou ambiente.

Tabela 2.3: **Exemplos de Controle de Acesso Baseado em Atributos**

Sujeito	Ação	Recurso	Contexto ou Ambiente
Um médico...	...deseja abrir e editar...	...o registro de um paciente...	...na sala de emergência do hospital às 8h.
Sr. Brown, pai...	...deseja acessar...	...o histórico escolar de suas filhas...	...de sua residência via Internet às 11h.
Um cliente do banco...	...deseja sacar R$500,00 reais...	...de uma conta...	...em um caixa eletrônico A na cidade B.

Apesar da variedade de opções para modelagem e definição de políticas de controle de acesso, a arquitetura da solução proposta neste documento faz uso dos modelos RBAC e ABAC. Justifica-se estas abordagens pela disponibilidade de tecnologias com suporte à descrição de políticas baseadas nestes modelos.

Capítulo 3

Autenticação e Autorização de Usuários em Ambientes de Computação Pervasiva

"Discovery consists in seeing what everyone else has seen and thinking what no one else has thought." (Albert Szent-Gyorgi).

Neste capítulo apresenta-se a especificação para permitir autenticação e autorização de usuários em redes UPnP. Inicialmente são introduzidas as diretrizes da solução, descrevendo os principais requisitos para contemplar serviços UPnP personalizados e seguros. Em seguida são apresentados os principais dispositivos envolvidos na solução cujas finalidades são provê autenticação e controle de acesso a usuários em ambientes pervasivos. Por fim, são descritos os mecanismos de autenticação e autorização de usuários, bem como as contribuições destes processos em ambientes pervasivos baseados no padrão UPnP.

3.1 Diretrizes da Solução

O desafio de lidar com uma rede dinâmica e heterogênea oferece diversas oportunidades de pesquisa no contexto de Computação Pervasiva. A dinamicidade de recursos, caracterizada pela entrada e saída de usuários, dispositivos e informações, demanda mecanismos de segurança e controle de acesso que deem suporte à definição de novas políticas conforme a presença de novos recursos na rede. Ademais, lidar com informações oriundas de diferentes fontes de dados (ex.: diferentes informações pertencentes a diferentes usuários) requer

soluções sistematizadas para proporcionar privacidade e integridade das informações bem como impedir a utilização de perfis de usuários resultantes de ataques de rede, tais como *user spoofing* [19].

A especificação UPnP-UP define uma arquitetura que contempla seguintes requisitos:

- **Autenticar usuários e controlar o acesso a serviços pervasivos**: o modelo de conectividade UPnP-UP permite identificar os usuários inseridos na rede e gerenciar a comunicação entre dispositivos dos usuários e entidades provedoras de serviços pervasivos. O modelo permite também que administradores de redes locais definam as políticas de controle de acesso aos serviços disponíveis na rede com base nas informações dos usuários que os acessam.

- **Adicionar diferentes níveis de segurança**: os níveis de segurança oferecem flexibilidade a depender do ambiente em que se pretende executar a solução. Administradores de rede podem definir em qual nível de segurança as informações serão trafegadas na rede e o usuário deve estar ciente e tomar a decisão sobre a disponibilidade das informações pessoais aos provedores de serviços;

- **Previnir ataques de programas maliciosos na rede**: dependendo do nível de segurança adotado, a arquitetura da especificação UPnP-UP buscou adicionar os requisitos necessários para "dificultar" ataques de programas maliciosos na rede, tais como os ataques de reprodução[26];

- **Oferecer um modelo de comunicação para a disponibilização de perfis de usuários**: a arquitetura da solução gerencia o compartilhamendo dos dados dos usuários com as entidades provedoras de serviços pervasivos, possibilitando a definição e a construção de serviços proativos.

Apesar do padrão UPnP ter sido o foco de estudo deste trabalho, a arquitetura proposta pode ser utilizada para quaisquer tecnologias de descoberta e provisão de serviços pervasivos devido a utilização de soluções disseminadas para autenticação e autorização em redes de computadores. A autenticação faz uso de assinaturas digitais (criptografias assimétricas, funções de hash) e o processo de autorização é baseado no modelo ABAC.

3.2 O Modelo de Conectividade UPnP-UP

O modelo de conectividade da especificação UPnP-UP permite definir e desenvolver serviços UPnP com suporte à personalização e controle de acesso para um usuário, proporcionando um ambiente proativo e seguro dos serviços disponíveis na rede. Para alcançar estes objetivos, o modelo de autenticação proposto identificará os usuários numa rede UPnP permitindo que o mecanismo de autorização realize controle de acesso aos serviços UPnP com base nas informações dos usuários. Está ilustrado na Figura 3.1 o cenário de conectividade UPnP-UP, apresentando o servidor de perfis para oferecer os serviços necessários de segurança e compartilhamento de perfis de usuários e uma entidade disponível na Internet (*upnp-up.org*) para proporcionar assinaturas digitais às entidades comunicantes numa rede UPnP.

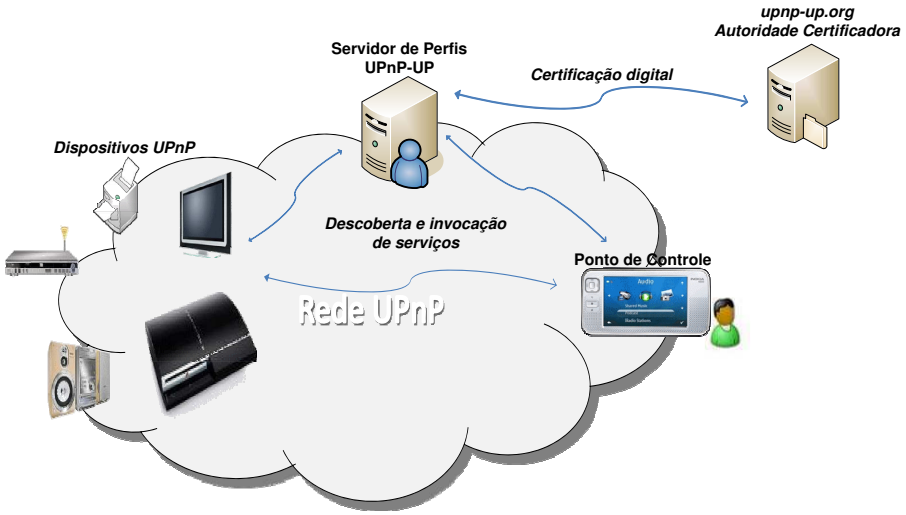


Figura 3.1: Modelo de Conectividade UPnP-UP.

Basicamente, o servidor de perfis oferece os serviços UPnP necessários para os mecanismos de autenticação de usuários, definição das políticas de controle de acesso aos serviços UPnP dos dispositivos disponíveis no ambiente e gerenciamento dos perfis dos usuários conectados à rede. Por outro lado, a autoridade certificadora garante a autenticidade dos usuários e do servidor de perfis, permitindo que pontos de controle e servidores de perfis possam confiar-se mutuamente numa rede UPnP. Além disso, esta entidade oferece as interfaces necessárias para a definição dos dados pessoais dos usuários.

3.3 Servidor de Perfis UPnP-UP

O servidor de perfis UPnP-UP é um dispositivo baseado no padrão UPnP responsável por armazenar dados de usuários, tais como nome completo, email e login, além de dados específicos inerentes às especificações UPnP, tais como UPnP Áudio/Vídeo, intensidade de lâmpadas, temperatura ambiente etc. Ademais, este dispositivo é capaz de provê os mecanismos necessários para identificar os usuários e especificar as políticas de controle de acesso aos serviços UPnP disponíveis na rede. Esta dissertação buscou definir uma especificação baseada no padrão UPnP para o servidor de perfis, proporcionando um modelo padronizado para a construção e a implantação de serviços UPnP seguros e cientes de contexto.

3.3.1 Descrição UPnP do Servidor de Perfis UPnP-UP

Estão disponíveis no padrão UPnP conteúdos formatados em XML que descrevem as características e os serviços providos por um dispositivo, como discutido na Seção 2.1. Na Listagem 6 é apresentada a descrição do servidor de perfis para o padrão UPnP utilizando o *XML Schema* definido pelo consórcio *UPnP Forum*.

```
1 <root xmlns="urn:schemas-upnp-org:upserver-1-0">
2   <!-- Informações ocultadas sobre a versão atual do dispositivo -->
3   <device>
4     <deviceType>
5       urn:schemas-upnp-org:device:UPServer:1
6     </deviceType>
7     <!-- Informações ocultadas sobre o dispositivo -->
8     <serviceList>
9       <service>
10        <serviceType>
11          urn:schemas-upnp-org:service:UPAuthentication:1
12        </serviceType>
13        <serviceId>
14          urn:upnp-org:serviceId:11
15        </serviceId>
16        <SCPDURL>/UPServices/up-authentication.wsdl</SCPDURL>
17        <controlURL>/UPServices/up-authentication/control</controlURL>
18        <eventURL>/UPServices/up-authentication/event</eventURL>
```

```
19        </service>
20        <service>
21          <serviceType>
22            urn:schemas-upnp-org:service:UPAuthorization:2
23          </serviceType>
24          <serviceId>
25            urn:upnp-org:serviceId:22
26          </serviceId>
27          <SCPDURL>/UPServices/up-authorization.wsdl</SCPDURL>
28          <controlURL>/UPServices/up-authorization/control</controlURL>
29          <eventURL>/UPServices/up-authorization/event</eventURL>
30        </service>
31        <service>
32          <serviceType>
33            urn:schemas-upnp-org:service:UPProfile:3
34          </serviceType>
35          <serviceId>
36            urn:upnp-org:serviceId:33
37          </serviceId>
38          <SCPDURL>/UPServices/up-profiles.wsdl</SCPDURL>
39          <controlURL>/UPServices/up-profiles/control</controlURL>
40          <eventURL>/UPServices/up-profiles/event</eventURL>
41        </service>
42      </serviceList>
43      <presentationURL>URL for presentation</presentationURL>
44    </device>
45 </root>
```

Listagem 6: Descrição UPnP do Servidor de Perfis UPnP-UP.

Além das informações básicas inerentes ao dispositivo, tais como o número de série e o endereço *Web* do fabricante, as Linhas 11, 22 e 33 da Listagem 6 definem, respectivamente:

- Os serviços de autenticação: identificar e gerenciar a entrada e saída de usuários na rede;
- Os serviços de autorização: definir e disponibilizar as políticas de controle de acesso aos serviços UPnP;

- Os serviços para personalização: recuperar dados dos usuários que englobam desde informações pessoais a gostos e preferências por gêneros musicais, temperatura e intensidade de iluminação ambiente.

3.3.2 Descrição dos Serviços UPnP do Servidor de Perfis UPnP-UP

Além da descrição do dispositivo, o padrão UPnP define um *XML Schema* para a descrição dos serviços disponibilizados pelo dispositivo em questão. As *tags* <SCPDURL> da Listagem 6 incluem as URLs para a obtenção de informações mais detalhadas dos serviços, os quais descrevem as variáveis de estado que os modelam. Na Listagem 7 está descrito o serviço de autenticação de usuários no servidor de perfis. O conteúdo completo com a descrição dos serviços obedecendo os requisitos exigidos pelo consórcio *UPnP Forum* encontra-se no Anexo B deste documento.

```
1 <?xml version="1.0"?>
2 <scpd>
3   <!-- Declarações de Namespaces ocultadas -->
4   <actionList>
5     <action>
6       <name>auth</name>
7         <argumentList>
8           <argument>
9             <name>encrypted-profile</name>
10            <direction>in</direction>
11            <relatedStateVariable>UUID</relatedStateVariable>
12          </argument>
13          <argument>
14            <name>profile</name>
15            <direction>in</direction>
16            <relatedStateVariable>UUID</relatedStateVariable>
17          </argument>
18          <argument>
19            <name>shareProfile</name>
20            <direction>in</direction>
21            <relatedStateVariable>profile</relatedStateVariable>
22          </argument>
```

```
23          <!-- Outras variáveis de entrada ocultadas -->
24          <argument>
25            <name>UUID</name>
26            <direction>out</direction>
27            <retval/>
28            <relatedStateVariable>username</relatedStateVariable>
29          </argument>
30        </argumentList>
31    </action>
32    <!-- Declarações de eventos ocultadas -->
33 </scpd>
```

Listagem 7: Descrição UPnP dos serviços disponibilizados pelo servidor de perfis UPnP-UP.

Basicamente, cada serviço está associado a um conjunto de *ações*. Na Listagem 7 está descrita a ação *Auth* (relacionada ao login do usuário nas Linhas 6 a 31), além das respectivas variáveis de entrada e saída representadas pelas tags <direction>. Na descrição dos serviços, também devem constar quais variáveis podem gerar eventos UPnP, tais como anunciar aos dispositivos interessados sobre a presença de novos usuários na rede quando estes autenticam-se no servidor de perfis. Todas as ações disponíveis pelo servidor de perfis estão descritas com detalhes no Apêndice B.

3.3.3 Mensagens de Eventos UPnP do Servidor de Perfis UPnP-UP

Como descrito na Seção 2.1, dispositivos UPnP podem lançar mensagens de eventos a rede de modo que eventuais dispositivos interessados na informação possam atualizar variáveis que modelam o estado de um serviço UPnP. No contexto do servidor de perfis, alguns eventos UPnP também podem ser enviados à rede com o objetivo de permitir a consistência das informações trafegadas e oferecer um mecanismo para a construção de serviços UPnP menos reativos e com um caráter de proatividade para ambientes pervasivos. Abaixo segue a lista dos eventos UPnP relacionados ao servidor de perfis.

- onUserEnter: evento informando o identificador do usuário que acaba de se autenticar no servidor de perfis.
  - Ex: “onUserEnter: 550e8400-e29b-41d4-a716-446655440000”;

  - Cenário de aplicação: personalização proativa de serviços: conteúdos multimídia, intensidade de iluminação e climatização de ambientes.

- onUserExit: evento informando o identificador do usuário que acaba de finalizar a sessão com o servidor de perfis.
  - Ex: "onUserExit: 550e8400-e29b-41d4-a716-446655440000";
  - Cenário de aplicação: remoção de dados de sessão do usuário em outros dispositivos UPnP.

- onChangeProfile: evento informando o identificador do usuário e o contexto (áudio/vídeo, temperatura ambiente etc) do perfil atualizado.
  - Ex: "onChangeProfile: 550e8400-e29b-41d4-a716-446655440000/AV".
  - Cenário de aplicação: dispositivos UPnP atualizam o perfil do usuário que pode estar armazenado localmente.

A descrição com mais detalhes sobre os eventos UPnP do servidor de perfis está disponível no Anexo A, no documento da especificação do servidor de perfis UPnP-UP.

3.4 Dispositivos UPnP com Suporte à Especificação UPnP-UP

Para contemplar serviços seguros e cientes de contexto, dispositivos UPnP devem obedecer a arquitetura da especificação UPnP-UP, considerando as condições de autenticação, autorização e personalização de serviços. Para alcançar este objetivo, a Tabela 3.1 apresenta os serviços UPnP que os dispositivos devem disponibilizar.

Tabela 3.1: **Serviços da especificação UPnP-UP para dispositivos UPnP**

Ações	Descrição	Parâmetros de Entrada (Obrigatórios)	Parâmetros de Entrada (Opcionais)	Parâmetros de Saída
initSession	Pontos de controle iniciam a sessão com um dispositivo UPnP. Utilizado apenas no nível 0 de segurança.	**UDN**: identificador do servidor de perfis que o usuário está autenticado. **UUID**: identificador do usuário no servidor de perfis. **profileP**: perfil do usuário com base no contexto do dispositivo UPnP.	Nenhum	**True/False**: confirma a abertura de sessão com o dispositivo UPnP.
openSession	Servidor de perfis requisita abertura de sessão entre um usuário e um dispositivo UPnP. Utilizado nos níveis de segurança 1 e 2.	**challenge**: número de desafio enviado através da ação *startSession* do servidor de perfis UPnP-UP. **UUID**: identificador do usuário no servidor de perfis.	Nenhum	**True/False**: confirma a abertura de sessão com o dispositivo UPnP. **UDN**: identificador deste dispositivo para o qual o usuário deseja iniciar uma sessão.
commitSession	Pontos de controle iniciam a sessão com um dispositivo UPnP. Utilizado nos níveis de segurança 1 e 2.	**UDN**: identificador do servidor de perfis que o usuário está logado. **UUID**: identificador do usuário que está iniciando a sessão com o dispositivo UPnP. **profileP**: perfil do usuário com base no contexto do dispositivo UPnP. **challenge**: número de desafio enviado através da ação *openSession* invocado pelo servidor de perfis. Utilizado também como número de sequência nas requisições subsequentes do ponto de controle ao dispositivo UPnP. **SymKey**: chave de criptografia simétrica para as requisições subsequentes criptografadas.	Nenhum	**True/False**: confirma a abertura de sessão com o dispositivo UPnP.

Continuação na próxima página

Tabela 3.1 – continuação da página anterior

Ações	Descrição	Parâmetros de Entrada (Obrigatórios)	Parâmetros de Entrada (Opcionais)	Parâmetros de Saída
logout	Finaliza a sessão do usuário com o dispositivo UPnP.	**UUID**: identificador de sessão do usuário com o servidor de perfis. **challenge**: número de sequência baseado no valor enviado pela ação *openSession*.	Nenhum	**True/False**: confirma o término da sessão.
getSupportedKeys	Pontos de controle obtêm um conteúdo formatado em XML com os valores das chaves públicas e dos algoritmos de criptografia que o dispositivo dar suporte para envio de requisições criptografadas.	Nenhum.	Nenhum.	Conteúdo formatado em XML informando os valores das chaves públicas e os algoritmos de criptografia que a implementação dar suporte.
updateProfile	Invocado por pontos de controle para atualizar o perfil do usuário.	**UUID**: identificador de sessão do usuário com o servidor de perfis. **challenge**: número de sequência baseado no valor enviado pelo ação *commitSession* **Context**: identificador do domínio ou contexto da aplicação. **File**: conteúdo atualizado, formatado em XML.	Nenhum.	Nenhum.

3.5 A Autoridade Certificadora upnp-up.org

Como ilustrado na Figura 3.2, a arquitetura da especificação UPnP-UP faz uso de uma autoridade certificadora disponível na Internet. O objetivo básico desta entidade é oferecer os mecanismos necessários para a emissão de certificados digitais para servidores de perfis, bem como permitir a mobilidade transparente dos usuários entre diferentes redes UPnP assegurando a integridade e a autenticidade do perfil do usuário.

Um das características essenciais para ambientes de computação pervasiva é o estabelecimento das relações de confiança entre as entidades comunicantes [73]. O uso da autoridade certificadora permite que pontos de controle e dispositivos UPnP estabeleçam relações de

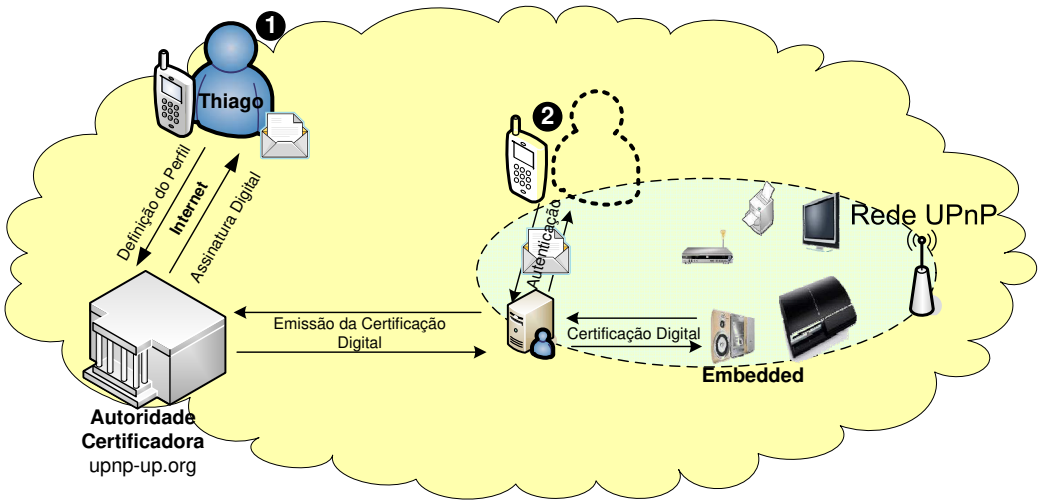


Figura 3.2: Obtenção e utilização de conteúdos assinados digitalmente através da Autoridade Certificadora upnp-up.org

confiança com servidores de perfis através de assinaturas digitais (ver Seção 3.6.2). Espera-se que a comunidade *UPnP Forum* seja a entidade responsável por manter e gerenciar a autoridade certificadora *upnp-up.org*.

3.6 Autenticação e Autorização de Usuários em Ambientes Pervasivos

Para proporcionar serviços personalizados e seguros, é importante identificar (autenticação) os usuários que desejam utilizar os serviços e, a partir deste procedimento, obter as informações necessárias para personalizar e realizar o controle de acesso (autorização) aos serviços de um dispositivo UPnP.

3.6.1 Perfis de Usuários na Especificação UPnP-UP

Diante do objetivo de proporcionar serviços personalizados e controle de acesso baseados no perfil do usuário, fez-se necessário definir uma arquitetura para o gerenciamento de perfis e a estrutura para permitir interoperabilidade entre serviços UPnP de diferentes domínios administrativos. O perfil do usuário divide-se em duas partes básicas: informações pessoais e preferências pessoais. No primeiro caso, estas informações são utilizadas para realizar con-

trole de acesso a serviços UPnP e, no segundo caso, proporcionar serviços personalizados.

Para permitir que um usuário seja identificado numa rede UPnP, este deve obter o perfil assinado digitalmente pela autoridade certificadora *upnp-up.org*. O perfil é estruturado em XML e assinado digitalmente através de *XML-Signature* [61], incluindo informações pessoais como nome, lista de emails, links para páginas pessoais, preferências por conteúdos multimídia (áudio e vídeo) etc. Para realizar controle de acesso, um dispositivo pode obter este perfil através dos servidores de perfis UPnP-UP e basear-se nas informações obtidas para permitir ou bloquear o acesso a um serviço UPnP. Neste caso, é importante salientar que um usuário pode exercer papéis diferentes em ambientes distintos, isto é, em um ambiente ele pode exercer o papel de "chefe" (de uma empresa) e em outro de "dono" (de uma residência). Desta forma, o perfil obtido através da autoridade certificadora também possui um mapeamento (referência) com perfis de outros domínios administrativos, como ilustrado na Figura 3.3.

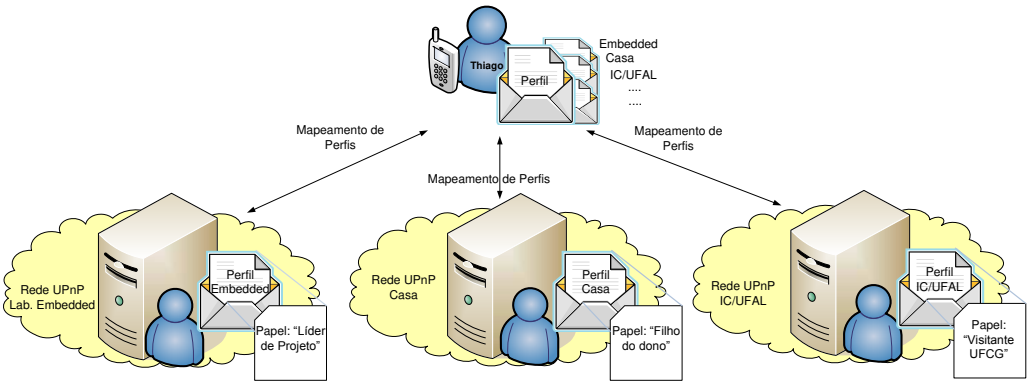


Figura 3.3: Diagrama do mapeamento entre perfis de um usuário.

Como exemplo, está descrito resumidamente na Listagem 8 o perfil de um usuário com base na especificação UPnP-UP. Além dos dados pessoais do usuário, nas Linhas 7 a 11 estão definidas as referências para outros perfis com base no nome e na posição geográfica do ambiente administrativo.

```
1 <userProfile up-id="TBMS7569">
2         <personal_data>
3                 <name>Thiago Bruno Melo de Sales</name>
4                 <username>thiagobrunoms</username>
5         </personal_data>
6         <!-- Outros dados pessoais ocultados -->
```

```
7         <external_profiles>
8                 <profile name="Embedded" latitude="-7.212367" longitude
                      ="-35.908268" />
9                 <profile name="Casa Campina Grande" latitude="-7.217444"
                      longitude="-35.906841" />
10                <profile name="IC/UFAL" latitude="9.554554" longitude
                      ="-35.774878" />
11        </external_profiles>
12        <!-- Informações sobre preferências ocultadas -->
13        <!-- Informações sobre a assinatura digital ocultadas -->
14 </userProfile>
```

Listing 8: Perfil resumido de um usuário descrevendo as referências para outros perfis externos.

O perfil assumindo as preferências nos diferentes contextos UPnP pode ser definido, por exemplo, em outras entidades computacionais, tal como no próprio dispositivo móvel ou obtidos através de redes sociais [40].

3.6.2 Autenticação da Especificação UPnP-UP

O modelo de autenticação da especificação UPnP-UP é dividido em 3 níveis de segurança e está ilustrado na Figura 3.4. Cada nível adiciona os requisitos de segurança do nível anterior, além dos requisitos para proporcionar maior grau de segurança na rede. Justifica-se esta separação para permitir maior flexibilidade entre diferentes necessidades de segurança e evitar a troca de mensagens desnecessárias em cenários onde os requisitos de autenticidade e confidencialidade destas mensagens não são necessários.

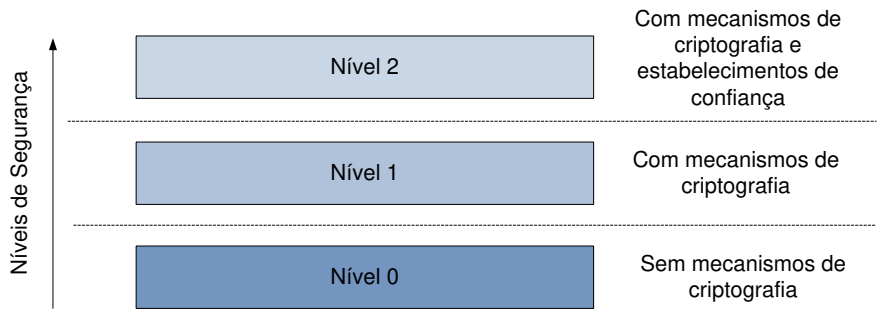


Figura 3.4: Níveis de segurança da especificação UPnP-UP.

Autenticação Nível 0

No nível 0 de autenticação da especificação UPnP-UP não há requisitos de confidencialidade das mensagens trafegadas nem processos para garantir a autenticidade destas mensagens. Um cenário seria uma residência, onde considera-se que todos os usuários conectados à rede são confiáveis (pessoas constituídas da mesma família, por exemplo). Desta forma, o processo de autenticação é simplificado, evitando a sobrecarga de dados criptografados na rede. O processo de autenticação é realizado através da ação UPnP *Auth* do servidor de perfis, detalhado no Apêndice B e ilustrado na Figura 3.5 (Etapa 1).

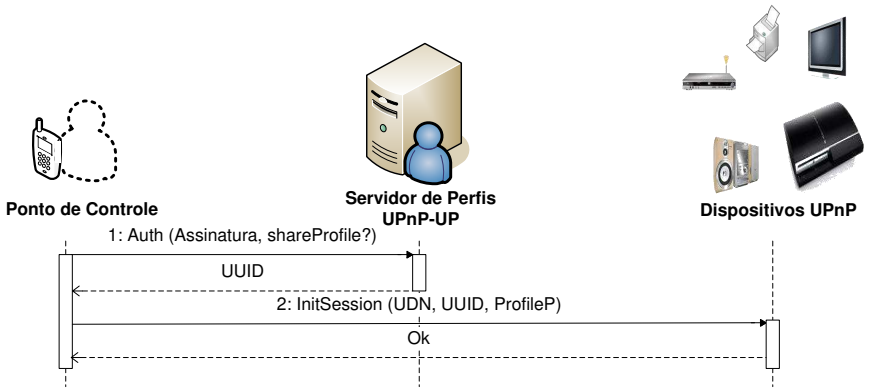


Figura 3.5: Autenticação e estabelecimento de sessão no nível 0 de segurança da especificação UPnP-UP.

Após requisitar a ação *Auth*, o ponto de controle recebe um UUID (*Universally Unique IDentifier*) para identificar o usuário na rede. Este UUID é único para cada usuário e válido durante toda a sessão da autenticação. Ou seja, enquanto o ponto de controle não invocar *Logout* ou caso expire a sessão do usuário. Neste último caso, a cada 300 segundos o ponto de controle do usuário deve invocar a ação *Ping* para notificar o servidor de perfis que o usuário está conectado à rede. Este mecanismo permite que, mesmo que o usuário desconecte da rede de forma inesperada (a energia da bateria do seu dispositivo chegou ao fim, por exemplo), suas informações de sessão não sejam mantidas no servidor de perfis e nos dispositivos UPnP. Para acessar quaisquer serviços dos dispositivos UPnP, o ponto de controle deve requisitar uma abertura de sessão com o dispositivo alvo através da ação UPnP *initSession* (Etapa 2), o qual confirma a abertura da sessão para o ponto de controle. Neste momento, o dispositivo pode personalizar e controlar o acesso aos serviços que o usuário

poderá posteriormente acessar.

Autenticação Níveis 1 e 2

O nível 0 de autenticação não possui mecanismos de criptografia nas requisições *Auth* e *initSession*. Por outro lado, a criptografia das informações não garante que uma requisição é autêntica, ou seja, se foi oriunda do emissor que construiu a requisição ou foi origem de um ataque de reprodução (*replay attack*) [75]. Neste tipo de ataque, uma requisição válida é obtida por outro usuário mal intencionado e, posteriormente, enviada novamente ao receptor.

Existem ambientes compartilhados (universidades, escritórios, aeroportos etc) em que a utilização dos mecanismos de criptografia é importante para assegurar a integridade e a confidencialidade das informações dos usuários. Além disso, faz-se necessário adicionar mecanismos que garantam a autenticidade da requisição, assegurando que estas sejam invalidadas se originadas de um ataque de reprodução.

Os níveis 1 e 2 adicionam mecanismos de criptografia para garantir a integridade e a confidencialidade das informações trafegadas na rede. Além disso, como a arquitetura UPnP permite a conectividade de um ou mais dispositivos, faz-se necessário adicionar mecanismos de confiança do servidor de perfis UPnP-UP, permitindo assim que pontos de controle e dispositivos UPnP possam confiar nas informações e nos serviços providos por um servidor de perfis UPnP-UP.

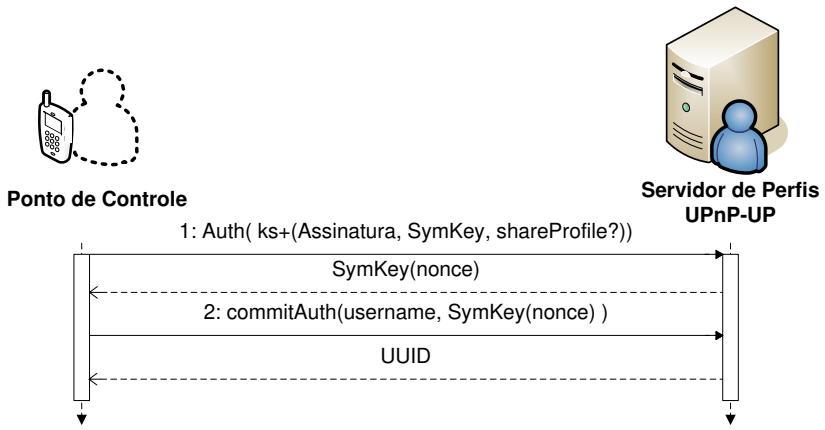


Figura 3.6: Autenticação de usuários utilizando os níveis 1 e 2 de segurança da especificação UPnP-UP.

O processo de identificação do usuário dos níveis 1 e 2 está ilustrado na Figura 3.6.

Assim como ocorre no nível 0, o ponto de controle do usuário invoca a ação *Auth* (Etapa 1) do servidor de perfis com os parâmetros descritos no Apêndice B. É importante notar que nestes níveis os parâmetros são criptografados com a chave pública do servidor de perfis (*Ks+*), o qual pode ser obtido através da ação UPnP *getSupportedKeys*. Ao receber uma requisição de autenticação, o servidor de perfis gera um valor aleatório (chamado de *nonce*) e o retorna, criptografado com a chave simétrica enviada na requisição *Auth* (*SymKey*), para o ponto de controle. O ponto de controle devolve o valor do *nonce* através da ação UPnP *commitAuth*, junto com o *login* do usuário (Etapa 2). Por fim, caso o valor de *nonce* recebido pelo servidor de perfis seja o mesmo recém enviado ao ponto de controle, o identificador do usuário (UUID) é gerado e entregue ao ponto de controle. O uso do *nonce* garante que o usuário está ao vivo, inviabilizando a realização de ataques de reprodução.

Assim como ocorre no nível 0 de autenticação, após a finalização do processo de identificação do usuário no servidor de perfis, o ponto de controle pode iniciar uma sessão com quaisquer dispositivos UPnP na rede. Entretanto, o modelo de inicialização de sessão apresentada para o nível 0 está também vulnerável a ataques de reprodução. Desta forma, a abertura de uma sessão realizada por um ponto de controle deve garantir que a requisição não é apenas válida, mas foi enviada pelo ponto de controle do usuário corrente. Está ilustrado na Figura 3.7 o processo de negociação para iniciar uma sessão entre um ponto de controle e um dispositivo UPnP. Este processo também envolve o servidor de perfis, o qual irá garantir a autenticidade do usuário que deseja acessar o dispositivo UPnP.

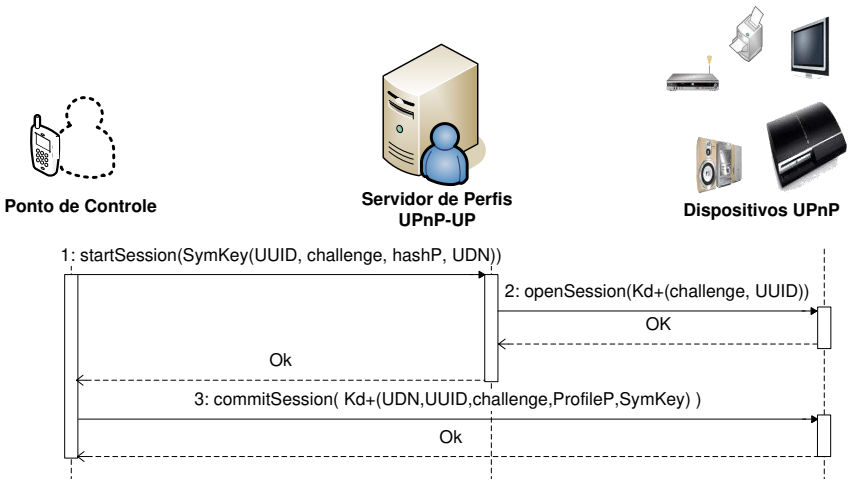


Figura 3.7: Abertura de sessão nos níveis de segurança 1 e 2 da especificação UPnP-UP.

Para acessar quaisquer dispositivos UPnP, um ponto de controle deve informar ao servidor de perfis o dispositivo UPnP que o usuário pretende iniciar uma sessão. Como ilustrado na Figura 3.7, o ponto de controle invoca a ação UPnP *startSession* (Etapa 1) do servidor de perfis criptografados com a chave simétrica negociada no processo de autenticação. Além do identificador do usuário (UUID) e do identificador do dispositivo que ele deseja utilizar (UDN), o ponto de controle deve enviar as seguintes informações:

- *challenge*: valor numérico aleatório que possibilitará um dispositivo UPnP associar o UUID do usuário autêntico que deseja acessar o dispositivo, a *posteriori*;
- *hashP*: valor de *hash* pertencente a uma *cadeia de hash*. A cadeia de *hash* de um usuário é baseada no valor do *nonce*, obtido no processo de autenticação. Este parâmetro busca garantir a autenticidade do usuário e da requisição.

O pedido de abertura de sessão é repassado para o dispositivo UPnP através da ação *openSession* (Etapa 2), com os dados criptografados com a chave pública do dispositivo alvo. Ao receber a requisição de abertura de sessão, o dispositivo armazena o valor *challenge* e confirma a inicialização da abertura da sessão. Caso autorizado, o ponto de controle deve invocar a ação *commitSession* (Etapa 3), informando os parâmetros apresentados na Tabela 3.1. Caso a abertura de sessão seja aceita, o ponto de controle poderá invocar quaisquer serviços do dispositivo UPnP, utilizando o UUID para identificar o usuário e o valor de *challenge* como número de sequência para previnir ataques de reprodução.

Ao operar no nível 1 de segurança, não há o estabelecimento de confiança entre dispositivos UPnP e pontos de controle com servidores de perfis. Entretanto, como a arquitetura de rede do padrão UPnP permite que um ou mais dispositivos do mesmo tipo sejam conectados à rede, existe a possibilidade de haver vários servidores de perfis conectados. Este cenário possibilitaria, por exemplo, a replicação dos dados de controle de acesso, oferecendo mecanismos de tolerância a falhas [67]. Por outro lado, ao permitir estes cenários de execução, qualquer usuário pode inserir e conectar um servidor de perfis na rede, definindo políticas de controle de acesso que não estão de acordo com a política administrativa da rede local. Desta forma, faz-se necessário um mecanismo de confiança do(s) servidor(es) de perfis UPnP-UP disponíveis na rede.

O nível 2 de segurança permite que servidores de perfis sejam registrados em autoridade certificadoras. A partir das CAs, pontos de controle podem se autenticar numa entidade confiável e dispositivos UPnP podem garantir que as requisições de abertura de sessão (através da ação *openSession*) são provenientes destas entidades. O nível 2 de segurança possui todos os requisitos do nível 1, incluindo mecanismos de certificação digital. Está ilustrado na Figura 3.8 o processo de disponibilização do certificado digital do servidor de perfis. Este processo ocorre em dois momentos distintos. Primeiro, antes de invocar a ação *Auth*, o ponto de controle deve obter o certificado digital do servidor de perfis a partir da ação *getSupportedKeys*. Segundo, a requisição da ação *openSession* deve incluir como parâmetro de entrada o certificado digital do servidor de perfis, possibilitando que dispositivos UPnP confirmem a autenticidade do servidor de perfis e permitam a abertura da sessão para o usuário alvo.

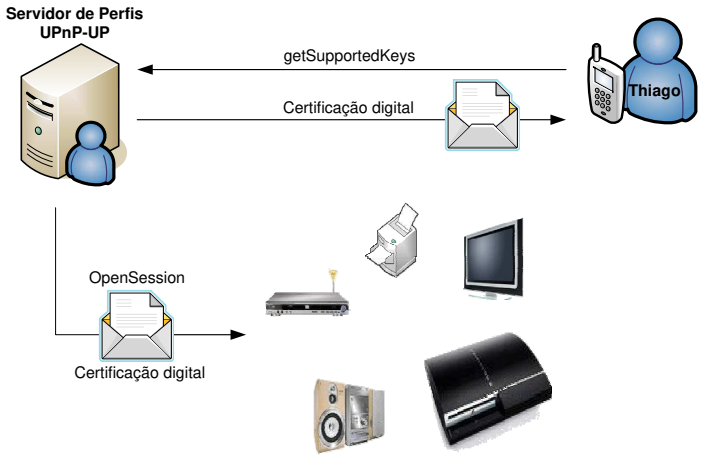


Figura 3.8: O certificação digital garante a autenticidade do servidor de perfis UPnP-UP aos pontos de controle e dispositivos UPnP.

3.6.3 Acesso aos Serviços UPnP com base na especificação UPnP-UP

Após o estabelecimento da sessão entre um ponto de controle do usuário e um dispositivo UPnP, as requisições aos serviços UPnP deste dispositivo devem também incluir o identificador UUID definido para o usuário, bem como o número de sequência da requisição (para os níveis 1 e 2 de autenticação). Estes parâmetros permitem que, para cada ação acessada, o dispositivo alvo possa identificar o usuário, garantir a autenticidade da requisição e efetuar personalização e controle de acesso baseando-se no seu perfil.

Desta forma, a especificação UPnP-UP permite que o identificador e o número de sequência sejam inseridos no cabeçalho da mensagem SOAP das requisições de controle UPnP, conforme descrito na Listagem 5. Esta opção evita que todos os serviços UPnP adicionem parâmetros de entrada e que os atuais serviços alterem suas interfaces de acesso, o que pode ocasionar numa quebra de interoperabilidade entre pontos de controle e serviços UPnP já consolidados. Como exemplo, está descrita na Listagem 9 a estrutura de uma mensagem SOAP onde estão inclusas as informações de sessão de um usuário. No campo <*s:Header*> (Linha 2) estão contidos os valores de sessão <*SeqNum*> (Linha 4) e <*UUID*> (Linha 5), representando o número de sequência da requisição e o identificador do usuário, respectivamente. Finalmente, no campo <*s:Body*> (Linha 8) estão definidos o nome da ação e do serviço sendo invocado, bem como o parâmetro de entrada desta ação.

```
1 <s:Envelope s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
2   <s:Header>
3     <Session>
4       <SeqNum>22883</SeqNum>
5       <UUID>8077e525-330a-4daa-b282-78beb6e6754b</UUID>
6     </Session>
7   </s:Header>
8   <s:Body>
9     <ns0:ServiceName xmlns:ns0="urn:schemas-brisaupnp-com:service:
        UserAuthentication:1">
10      <inputA>My input parameter</inputA>
11    </ns0:ServiceName>
12  </s:Body>
13 </s:Envelope>
```

Listagem 9: Requisição SOAP de controle à ação UPnP *NewService* com dados de sessão do usuário inseridos no cabeçalho da mensagem.

É importante salientar que o número de sequência deve ser trafegado de forma criptografada com a chave simétrica negociada durante o estabelecimento da sessão, evitando assim que o seu valor seja obtido por outrem. No contexto de um servidor de perfis, este número de sequência é baseado no valor *nonce* definido pelo servidor de perfis para um usuário no estabelecimento da sua autenticação. Como exemplo, o valor de *nonce* como

número de sequência é utilizado na ação *Ping*, discutida na Seção 3.6.2. Na perspectiva de quaisquer outros dispositivos UPnP, o número de sequência é baseado no parâmetro *challenge* definido na abertura da sessão com o dispositivo UPnP.

3.6.4 Autorização e Controle de Acesso

No contexto da especificação UPnP-UP, as políticas de controle de acesso estão estritamente relacionadas ao ambiente administrativo da rede local. Isto é, estas não são distribuídas na Internet e/ou compartilhadas entre diferentes domínios administrativos, mas cada rede UPnP realiza o gerenciamento das políticas de controle de acesso inerentes aos dispositivos localmente distribuídos.

O processo de definição e distribuição das políticas de controle de acesso da especificação UPnP-UP é realizado como ilustrado na Figura 3.9. As políticas de controle de acesso são armazenadas no servidor de perfis e definidas pelo administrador da rede administrativa utilizando uma aplicação interoperável (um ponto de controle ou uma interface gráfica executada no próprio servidor de perfis, por exemplo). Este modelo permite que tanto o servidor de perfis envie as políticas de controle de acesso a cada dispositivo da rede (*push-based*), quanto cada dispositivo adquira a lista de controle de acesso definida e armazenada no servidor de perfis (*pull-based*) baseando-se nos atributos do dispositivo (ex: nome, *UDN*, *URN*, tipo etc). Estas abordagens permitem distribuir as responsabilidades das interpretações das políticas entre os dispositivos, bem como possibilitar que dispositivos com recursos de processamento e armazenamento limitados deleguem a análise do controle de acesso ao servidor de perfis.

O cenário ilustrado na Figura 3.10 representa o processo de controle de acesso a um usuário com identificador *UUID 2*. Ao receber uma requisição deste usuário na rede (Etapa 1), um dispositivo UPnP obtém o perfil do usuário com informações pessoais a partir do servidor de perfis UPnP-UP (Etapas 2 e 3), permitindo ou negando o acesso a este serviço (Etapa 4). Esta abordagem possibilita que dispositivos UPnP armazenem localmente tanto as políticas quanto os perfis dos usuários que acessam o dispositivo em questão. Por outro lado, dispositivos com recursos limitados de armazenamento e processamento podem delegar a interpretação da política para o próprio servidor de perfis, obtendo apenas uma resposta de permissão ou bloqueio de acesso. A desvantagem desta última abordagem é que o processo

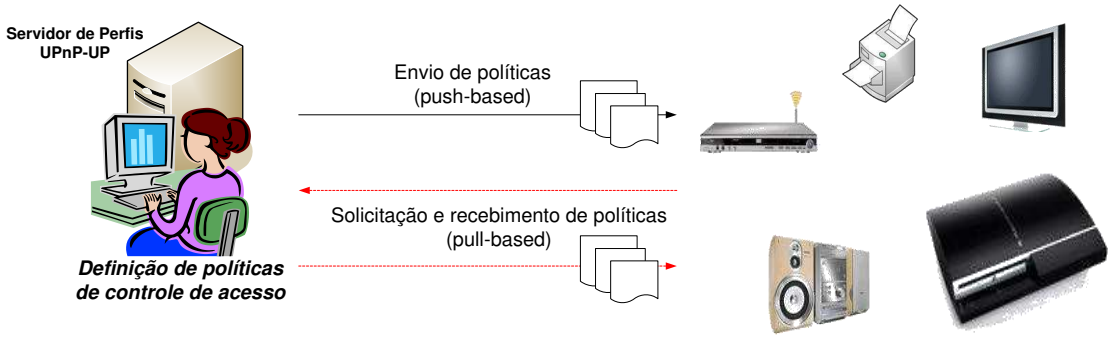


Figura 3.9: Cenário de definição e distribuição de políticas de controle de acesso em ambientes pervasivos baseados na especificação UPnP-UP.

para delegar deve ser repetido para toda requisição ao dispositivo UPnP.

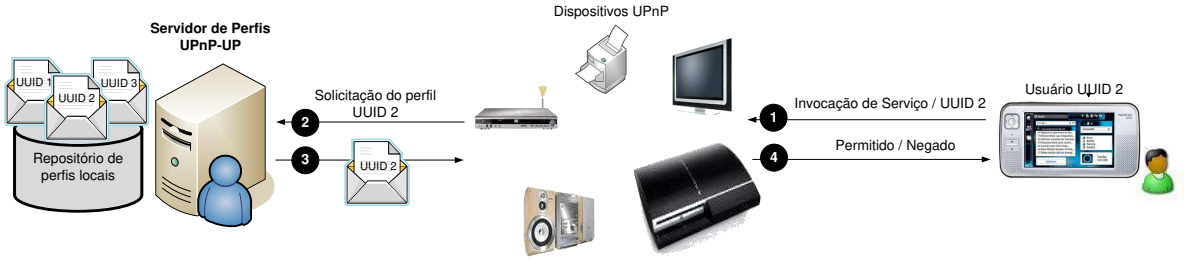


Figura 3.10: Processo de autorização baseado na especificação UPnP-UP.

3.7 Contribuições da Especificação UPnP-UP

Com os mecanismos de autenticação e autorização de usuários propostos neste trabalho, é possível vislumbrar um leque de aplicações nos contextos de segurança e desenvolvimento de serviços pervasivos. A seguir, são descritos os possíveis cenários de aplicação considerando os principais dispositivos UPnP atualmente padronizados pelo consórcio *UPnP Forum*.

- Monitoração e Vigilância: câmeras UPnP [82] têm como objetivo geral disponibilizar imagens de vídeo através de serviços para ambientes domésticos, permitindo que pontos de controle recebam o vídeo em tempo real e usuários realizem a monitoração do ambiente. É notória a necessidade de determinar os potenciais usuários que podem ter autorização para visualizar o conteúdo sendo transmitido. Portanto, a solução de autenticação e autorização de usuários proposta neste documento é realmente relevante

para ser integrada nestes ambientes, permitindo que tais imagens não sejam visualizadas por intrusos;

- Temperatura e Iluminação: determinar a temperatura do ambiente, a intensidade de iluminação e a posição das persianas de uma janela são alguns dos objetivos da especificação UPnP HVAC. Neste contexto, a especificação UPnP-UP permite que tais serviços sejam desenvolvidos com suporte a controle de acesso, bloqueando requisições oriundas de usuários que não possuem autorização para alterar a temperatura local ou a intensidade de iluminação atual, por exemplo. Além disso, é possível definir e construir serviços de ambientação com características de proatividade, determinando a climatização e a iluminação do ambiente com base em mensagens de eventos (realizados pelo servidor de perfis) de entrada e saída dos usuários;

- Dispositivos Multimídia: de acordo com a problemática discutida na Seção 1.1, dispositivos de mídia UPnP são capazes de disponibilizar e reproduzir conteúdos de áudio, vídeo e imagem, porém, não oferecem controle de acesso e/ou personalização de conteúdos. Conforme apresentado no Capítulo 5, a especificação UPnP-UP permite que desenvolvedores definam e construam serviços UPnP com requisitos de controle de acesso e personalização de conteúdos multimídia. Estes requisitos possibilitam controlar o acesso a conteúdos privados, como palestras, vídeos pessoais, bem como reproduzir proativamente conteúdos multimídia em ambientes pervasivos.

Por fim, os requisitos de personalização/proatividade de serviços e segurança, atualmente não presentes no padrão UPnP, são o cerne das contribuições deste trabalho. É possível agora definir serviços UPnP com características de proatividade e segurança, mantendo também os requisitos de interoperabilidade propostos pelo padrão.

Capítulo 4

Trabalhos Relacionados

Neste capítulo são apresentadas as abordagens existentes dentro do contexto de autenticação e autorização em ambientes pervasivos. Em especial, são discutidas também as propostas de segurança para o padrão UPnP, bem como as características e limitações de cada trabalho para proporcionar segurança em serviços baseados neste padrão.

4.1 Autenticação e Autorização em Ambientes Pervasivos

Segurança é um dos principais desafios em ambientes de computação pervasiva devido ao alto grau de heterogeneidade das entidades que podem estar presentes nestes ambientes [69] [36]. As soluções propostas pela comunidade científica no contexto de segurança para computação pervasiva são bastante heterogêneas, considerando diversos aspectos desde a localização geográfica das entidades envolvidas [32] ao estabelecimento de confiança entre estas entidades [42]. Em [33] é proposto um modelo de autorização dinâmico que utiliza como base o contexto atual do usuário para controlar o acesso aos serviços. A ideia básica é permitir que as alterações nas informações de contexto (horário, temperatura, posição geográfica etc) modifiquem dinamicamente o resultado das políticas de controle de acesso. Para alcançar este objetivo, cada usuário possui um conjunto de observadores (sensores, por exemplo) que são dinamicamente associados ao usuário e, ao mesmo tempo, confiáveis aos provedores de serviços. Tais observadores enviam continuamente mensagens de contexto aos provedores de serviços, permitindo que a validade das políticas de controle de acesso seja relativa ao tempo.

Já no trabalho descrito em [42], os autores partem da premissa de que os modelos existentes para autenticação e controle de acesso a usuários não são suficientes para serem empregados em ambientes de computação pervasiva, tendo em vista que os usuários não são conhecidos *a priori*. Desta forma, é proposto um modelo baseado no estabelecimento de confiança entre usuários para permitir que o direito de acessar um serviço seja delegado pelo proprietário do serviço através de credenciais baseados na arquitetura SPKI (*Simple Public Key Infrastructure*). Como exemplo, considere um usuário *A* que deseja acessar um serviço *S* gerenciado pelo usuário *B*. Através de credenciais, *B* emite para *A* um certificado digital (pois *B* confia em *A*) que deve ser enviado ao serviço *S*. Com efeito, como *B* é um usuário confiável por *S*, *A* pode acessar o serviço *S* com base no certificado digital emitido por *B*. Esta solução é interessante pois não exige a definição e a descrição de políticas de controle de acesso. Por outro lado, caso o proprietário do serviço não esteja disponível, existe a possibilidade do acesso ao serviço ficar comprometido.

As soluções de segurança para computação pervasiva são interessantes para serem integradas ao modelo de conectividade UPnP. Estas soluções buscam adicionar mecanismos de autenticação e autorização de usuários, permitindo que sejam integradas em quaisquer mecanismos de conectividade para ambientes pervasivos. Por outro lado, estas soluções dificultam a interoperabilidade com soluções UPnP já existentes, dificultando a sua adoção pela comunidade *UPnP Forum*. Desta forma, as duas próximas seções apresentam as soluções encontradas para permitir autenticação e autorização em redes UPnP.

4.2 As especificações UPnP *Device Security* e *Security Console*

A arquitetura básica para segurança em redes UPnP é apresentada nas especifições UPnP *Device Security* (DS) e UPnP *Security Console* (SC). Para oferecer serviços UPnP com suporte à controle de acesso, um dispositivo deve contemplar os requisitos básicos da especificação DS. Por outro lado, para que um usuário possa realizar tarefas de gerenciamento deste dispositivo, tais como definir e incorporar políticas de controle de acesso, ele deve utilizar um dispositivo que proporcione os serviços da especificação SC. Para possibilitar a realização destas tarefas, o SC deve oferecer ao usuário uma interface administrativa de alto nível.

Como ilustrado na Figura 4.1, um usuário obtém os direitos administrativos de um dispositivo DS através de um dispositivo SC (através de uma senha, por exemplo), adquirindo certificados digitais emitidos pelo DS. Através de uma interface administrativa, o usuário também pode definir e adicionar as políticas de controle de acesso, as quais são descritas em ACL com base nas informações dos pontos de controle que desejam acessar algum serviço do DS. Tais pontos de controle podem obter certificados de acesso através do dispositivo SC, apresentando estes certificados no momento em que acessar algum serviço disponibilizado pelo DS.

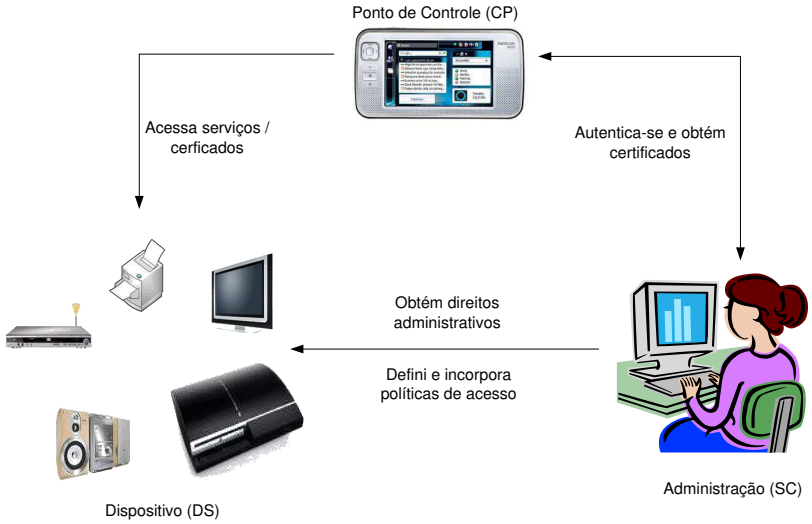


Figura 4.1: Arquitetura de comunicação UPnP *Device Security* e UPnP *Security Console*.

Apesar de oferecer uma solução com suporte à confidencialidade, integridade e prevenção contra ataques de reprodução, bem como ser padronizada pelo *UPnP Forum*, o controle de acesso é totalmente baseado nas informações dos pontos de controle, o que não é suficiente em ambientes de computação pervasiva, pois não há a integração com as informações dos usuários que acessam os dispositivos. Além disso, o modelo de segurança destas especificações não é flexível ao ponto de remover os mecanismos de criptografia, se necessários, pois entende-se que o ambiente pode ser, em sua essência, confiável[1]. Por outro lado, a arquitetura da especificação UPnP-UP oferece três níveis de segurança, proporcionando um modelo de autenticação mais flexível e permitindo que o controle de acesso seja baseado, entre outras informações, no perfil do usuário.

[1]Em ambientes domiciliares, por exemplo, onde o usuário possui controle dos usuários que podem ter acesso à rede.

Outras desvantagens desta solução são a utilização de uma linguagem não padronizada para a descrição das políticas de controle de acesso e a utilização de certificados digitais que não seguem o modelo X.509. Nestes casos, exige-se que desenvolvedores compreendam uma linguagem, de certo modo, proprietária, para que seja possível definir políticas de acesso e construir certificados digitais para permitir o acesso aos pontos de controle. Por outro lado, a especificação UPnP-UP propõe o uso da linguagem de marcação XACML no processo de autorização, e o modelo X.509 para a emissão de certificados digitais, facilitando o desenvolvimento de novos dispositivos UPnP interoperáveis também para os processos de autenticação e autorização.

4.3 Uma patente para Autenticação e Autorização de dispositivos em redes UPnP

Em [1] está definida uma patente para autenticação e autorização de pontos de controle em redes UPnP. Um ponto de controle está autorizado a acessar os serviços de um dispositivo UPnP se o endereço MAC (*Media Access Control*) do ponto de controle está registrado numa tabela de endereços MAC autenticados. A autenticação é realizada através de certificações digitais, emitidas por entidades certificadoras de confiança e armazenados no ponto de controle, enquanto a confidencialidade das mensagens é alcançada através de criptografia de chaves assimétricas.

Diversas desvantagens foram encontradas nesta proposta. Inicialmente, não há mecanismos de prevenção a ataques de reprodução e um endereço MAC existente na tabela de permissões pode ser falsificado e utilizado por outro ponto de controle através de um ataque de *MAC Spoofing* [89]. O impacto disso é um ponto de controle não autenticado acessar os serviços de um dispositivo UPnP. Ademais, o endereço MAC está localizado na camada de enlace da pilha de protocolos TCP/IP, enquanto o padrão UPnP visa abstrair os detalhes da rede através do uso da tecnologia *Web Services*. Finalmente, é inconcebível a utilização desta solução em ambientes de computação pervasiva pois também não há o conceito de contexto para garantir a autenticação, autorização, e provisão de serviços cientes de contexto em ambientes pervasivos. Através das informações de contexto, pode-se alcançar a dinamicidade proposta pelo padadigma da computação pervasiva.

4.4 Considerações sobre os Trabalhos Relacionados

Apesar da variedade de soluções para segurança no contexto da computação pervasiva, as propostas são complexas para serem implementadas e integradas ao modelo simplificado da arquitetura de conectividade UPnP. Além disso, as soluções para permitir autenticação e autorização em redes UPnP apresentam limitações importantes (ver Tabela 4.1) e não são suficientes para proporcionar mecanismos de segurança com base nas informações dos usuários conectados na rede. Por fim, a especificação proposta neste documento utiliza os mesmos conceitos do padrão UPnP, o que deve facilitar a sua integração à arquitetura do padrão.

Tabela 4.1: **Comparativo entre os trabalhos relacionados.**

Trabalhos/Requisitos	Autenticação	Autorização	Suporte a perfis de usuários	Vulnerável à ataques de rede
Implementing Access Control to People Location Information	Não	Sim	**Não**	Sim
Trust-Based Security in Pervasive Computing Environments	Sim	Sim	**Não**	Sim
UPnP Device Security (DS) e Security Console (SC)	Sim	Sim	**Não**	Sim
Patente para autenticação e autorização em redes UPnP	Sim	Sim	**Não**	Sim

Capítulo 5

Estudo de Caso

Os estudos de casos discutidos nesta seção visam validar a arquitetura da especificação UPnP-UP apresentada no Capítulo 3. Para alcançar este objetivo, são descritos os requisitos de implementação incorporados ao *framework* BRisa (versão Python) para possibilitar a construção de dispositivos UPnP com suporte à autenticação e autorização de usuários, bem como são apresentados os cenários e os detalhes de implementação utilizados para a concretização da especificação proposta.

5.1 Adaptações no BRisa

Como apresentado na Seção 2.1, o *framework* BRisa objetiva abstrair do desenvolvedor de software os detalhes para a construção de dispositivos baseados no padrão UPnP, como, por exemplo, a descoberta e a invocação de serviços. Através deste *framework* é possível construir um dispositivo UPnP, um conjunto de serviços UPnP associados ao dispositivo e, para cada serviço, um conjunto de ações. O *framework* proporciona automaticamente todas as etapas de conectividades UPnP descritas na Seção 2.1.

Resumidamente, a arquitetura de um dispositivo UPnP baseado neste *framework* está ilustrada na Figura 5.1. Ao receber uma requisição de controle UPnP oriunda de um ponto de controle, um servidor Web repassa a requisição ao módulo *Controlador* de serviços, o qual é responsável pelo tratamento da mensagem SOAP (extração dos dados de cabeçalho e dos parâmetros de entrada) através do submódulo *Interpretador SOAP*, bem como por obter a referência à ação requisitada, a invocação desta ação e o retorno do resultado desta invocação

para o servidor Web.

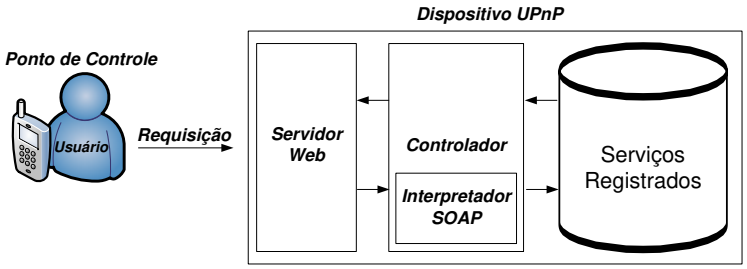


Figura 5.1: Arquitetura resumida de um dispositivo UPnP baseado no *framework* BRisa.

Para possibilitar o desenvolvimento dos estudos de casos, foram inicialmente realizadas algumas alterações no *framework* para contemplar requisitos da especificação UPnP-UP e possibilitar a construção de dispositivos UPnP com suporte à personalização e controle de acesso. Buscou-se manter a compatibilidade com versões anteriores dos dispositivos baseados no BRisa, permitindo que estas versões sejam compatíveis com a versão modificada da ferramenta. Estão descritos abaixo os principais requisitos adicionados ao BRisa.

1. Adicionar informações no cabeçalho de mensagens SOAP: no módulo *Interpretador SOAP* foram adicionados os requisitos necessários para incluir os dados de cabeçalhos SOAP, permitindo que as informações de sessões dos usuários (número de sequência, identificador UUID) sejam trafegadas nas ações invocadas por pontos de controle;

2. Determinar a autenticidade da ação UPnP e realizar controle de acesso: o módulo *Controlador* foi alterado para a realização das seguintes tarefas:

   (a) Verificar o nível de segurança UPnP-UP definido pelo servidor de perfis disponível na rede e, caso necessário, determinar a corretude do número de sequência proveniente do cabeçalho SOAP;

   (b) Delegar o mecanismo de controle de acesso ao módulo do serviço requisitado, mantendo também a sua coesão.

O requisito descrito no item 2a determina a condição necessária para garantir que a requisição não é oriunda de um ataque de reprodução. Esta etapa é realizada pelo módulo *Controlador*, o qual determina se o número de sequência para o usuário corrente é válido. Já o requisito definido no item 2b estabelece os mecanismos para a realização do controle

de acesso. Neste caso, após determinar a consistência do número de sequência, a invocação de quaisquer ações de um serviço UPnP é interceptada antes e depois da execução da ação em questão. No primeiro caso, o desenvolvedor do dispositivo pode determinar a análise das políticas de controle de acesso e, no segundo caso, pode-se realizar mecanismos de *log* para efetuar, posteriormente, tarefas de auditoria.

Está ilustrado na Figura 5.2 o diagrama de sequência dos requisitos supracitados. Ao receber quaisquer requisições para uma ação UPnP, o módulo *Controlador* do BRisa é responsável por determinar a consistência do número de sequência contido no cabeçalho SOAP da requisição (Etapa 2). Confirmado o número de sequência, o módulo *Controlador* invoca o método *before_call* do serviço UPnP para determinar se o usuário tem permissão para acessar a ação desejada (Etapa 3). Uma vez permitido o acesso (Etapa 4), o módulo *Controlador* invoca a ação solicitada (Etapas 5 e 6) bem como o método *after_call* do serviço UPnP para registrar as informações de acesso (Etapa 7), se necessário. Finalmente, é retornado ao ponto de controle o resultado da ação (Etapa 8).

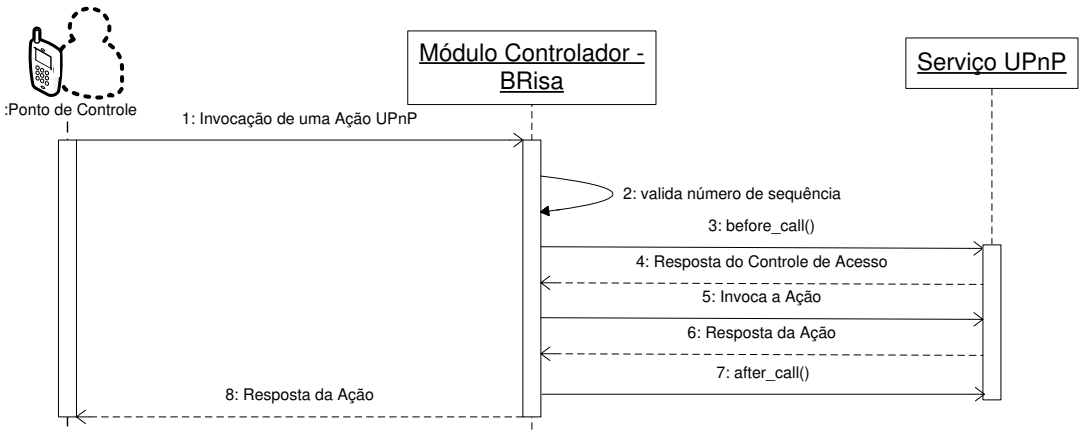


Figura 5.2: Diagrama de sequência: determinando número de sequência e controle de acesso no BRisa.

Caso o número de sequência seja inválido ou o usuário não possua permissão para acessar o serviço (Etapas 2 e 4, respectivamente), a invocação da ação não é realizada e códigos de erro UPnP definidos pela especificação UPnP-UP são retornados. Para mais detalhes sobre os códigos de erro, ver apêndice B.

5.2 Personalização e Controle de Acesso

Os cenários descritos focam a personalização de conteúdos multimídia com base na especificação *UPnP A/V* e o controle de acesso a impressoras com base na especificação *UPnP Printer Device*. Assim, espera-se demonstrar os diferentes contextos de implementação e os benefícios alcançados com a especificação UPnP-UP.

5.2.1 Personalização de Conteúdos Multimídia

O primeiro estudo de caso visa oferecer serviços UPnP personalizados com base no perfil do usuário. A recomendação e a personalização de conteúdos já estão bem consolidadas nos serviços aplicados na Web, em especial no contexto de comércio eletrônico, os quais estão contemplados em soluções como Ebay[1], Submarino[2] e Amazon[3]. As técnicas utilizadas em soluções baseadas na Web podem ser integradas também no contexto da computação pervasiva, pois fundamentam-se em bases matemáticas e, consequentemente, são independentes do domínio da aplicação. Não está no escopo deste trabalho discuti-las e o leitor pode obter informações adicionais em [44][60][8].

Como discutido na Seção 1.1 e ilustrado na Figura 5.3, a especificação *UPnP A/V* oferece uma abordagem para que pontos de controle possibilitem a navegação por arquivos multimídia (vídeo, áudio ou imagens) armazenados em um servidor de mídia (Etapas 1 e 2 através da ação UPnP *Browse*) e os reproduzam em um dispositivo denominado renderizador de mídias (Etapas 3, 4 e 5). O objetivo deste primeiro estudo de caso é possibilitar que a invocação a ação *Browse* retorne apenas os conteúdos multimídia com base nas preferências (musicais, por exemplo) do usuário, permitindo que o mesmo não seja "bombardeado" com informações que possivelmente não lhe agradem.

Neste contexto, para permitir a personalização de conteúdos multimídia em redes UPnP, a aplicação *BRisa Media Server* foi alterada para dar suporte aos seguintes requisitos:

- Autenticar usuários baseado-se na especificação UPnP-UP;
- Permitir a filtragem do conteúdo multimídia baseando-se no perfil dos usuários.

[1] www.ebay.com
[2] www.submarino.com.br
[3] www.amazon.com

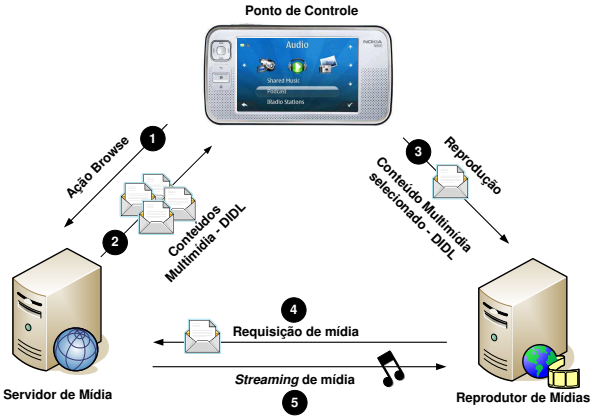


Figura 5.3: Navegação por conteúdos multimídia baseada no padrão UPnP.

No primeiro caso, os requisitos de autenticação propostos neste trabalho foram incorporados na aplicação, considerando principalmente os diferentes níveis de autenticação discutidos na Seção 3.6.2. No segundo caso, a ação UPnP *Browse* foi alterada para filtrar os conteúdos multimídia com base no perfil do usuário e retornar apenas aqueles conteúdos de que potencialmente o usuário poderá gostar.

Perfil do Usuário para Conteúdos Multimídia

A especificação *UPnP A/V* faz uso do padrão DIDL (*Digital Item Declaration Language*) [6] para descrever conteúdos multimídia e permitir, desta forma, a interoperabilidade entre diferentes aplicações. Com efeito, o perfil do usuário apresentado na Seção 3.6.1 permite descrever, além dos dados pessoais do usuário, as preferências nos diferentes domínios de dispositivos UPnP. Assim, a especificação UPnP-UP permite englobar as duas abordagens e acoplar as características da DIDL ao perfil do usuário para a descrição das preferências para conteúdos multimídia. Neste contexto, está descrito de forma resumida na Listagem 10 um exemplo do perfil do usuário incluindo também suas preferências para conteúdos multimídia baseado no padrão DIDL.

```
1 <?xml version="1.0" encoding="UTF-8">
2 <up:profile up-id="TBMS1302" xmlns:up="urn:schemas-upnp-up-org:up-1-0">
3
4   <up:personal_data id="personal_profile">
5     <up:name>Thiago Bruno Melo de Sales</up:name>
6     <up:username>thiagobrunoms</up:username>
```

```
7     <!-- Outras informações do perfil ocultadas -->
8   </up:personal_data>
9
10  <up:preferences_data>
11    <up:container_list>
12      <up:container id=0 name=AV>
13        <DIDL-Lite xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
14          xmlns:dc="http://purl.org/dc/elements/1.1/">
15          <item>
16            <dc:title>GenRosso - StreetLight - Brazilian Show</dc:title>
17            <dc:description>A musical based on the true story of
                  Charlies Moats.</dc:description>
18            <dc:format>video</dc:format>
19          </item>
20          <!-- Outros itens ocultados -->
21        </DIDL-Lite>
22      </up:container>
23    </up:container_list>
24  </up:preferences_data>
25 </up:profile>
```

Listagem 10: Estudo de Caso: perfil de um usuário para conteúdos multimídia.

De acordo com a Listagem 10, as linhas entre 4 e 8 definem os dados pessoais do usuário. As linhas entre 10 e 24 definem os dados inerentes as preferências dos usuários, divididas entre diferentes *containers*[4]. Entre as linhas 13 e 21 está definido o perfil do usuário para conteúdos multimídia, com base no padrão DIDL. Para cada *tag* <item> declarada, está definido um conteúdo multimídia de preferência do usuário, detalhando os outros atributos como o título, a descrição e o formato da mídia.

Recomendação de Conteúdos Multimídia em Redes UPnP

Neste trabalho utilizou-se uma técnica simples para filtragem de informação chamada TF-IDF (*Term Frequency, Inverse Document Frequency*) [86], a qual é baseada no modelo vetorial e objetiva mensurar a similaridade entre dois textos. No contexto deste trabalho, para

[4]Cada *container* descreve um contexto do padrão UPnP. Ex. Áudio e Vídeo, iluminação e temparatura ambiente etc. Para mais informações, ver Apêndice A.

cada conteúdo multimídia descrito na *tag* $< item >$ disponível no perfil do usuário, a técnica TF-IDF proporcionará a base para encontrar outros conteúdos multimídia ($< item >$) similares e que não estão presentes no perfil do usuário.

Basicamente, o algoritmo TF-IDF é dividido em dois componentes. O componente TF representa a frequência de um termo t_i em um documento d_j considerando também a quantidade total de termos presentes no documento. Desta forma, a frequência de um termo é definida como representada na equação 5.1:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{5.1}$$

onde $n_{i,j}$ é o número de ocorrências de um termo t_i em um documento d_j, normalizado pela soma do número de ocorrências $n_{k,j}$ de todos os termos t_k do documento d_j. O segundo componente utilizado no algoritmo é o IDF, o qual determina o quanto uma palavra é importante dentro de uma coleção de documentos e é dada pela equação 5.2:

$$idf(i) = \lg \frac{|D|}{|d : t_i \in d|} \tag{5.2}$$

onde $|D|$ é a quantidade total de documentos disponíveis no repositório e d é a quantidade de documentos em que o termo t_i está presente. Finalmente, o TF-IDF de um termo i para um documento j é dada pela equação 5.3:

$$(TF - IDF)_{i,j} = tf_{i,j} \times idf_i \tag{5.3}$$

Assim, para cada termo t_i de um documento d_j calcula-se o seu valor $(TF - IDF)_{i,j}$. O conjunto destes valores forma um vetor que representa o documento. Por fim, todos os documentos do repositório são representados por um vetor e, dado um vetor de entrada, calcula-se a similaridade entre o documento de entrada e cada documento do repositório através do cálculo do *cosseno* ou da distância euclidiana.

Está descrita na Listagem 11 a ação *Browse* da aplicação *BRisa Media Server*. Esta ação foi modificada para dar suporte à filtragem de conteúdos multimídia com base no perfil do usuário. Por razões de simplicidade, está descrito apenas o trecho de código necessário para a compreensão do processo de personalização dos contéudos multimídia.

De acordo com a Listagem 11, obtem-se inicialmente o perfil do usuário estruturado

em DIDL e representado pela variável *set_item_user* (linha 4). Posteriormente, calcula-se a similaridade de cada conteúdo multimídia disponível no perfil do usuário com os conteúdos multimídia disponíveis no repositório do servidor de mídias (linhas 7 e 8). O conjunto de conteúdos multimídia similares ao perfil do usuário são adicionados numa estrutura DIDL representada pela variável *didl* (linha 11) e retornada para o usuário requisitante (linha 13).

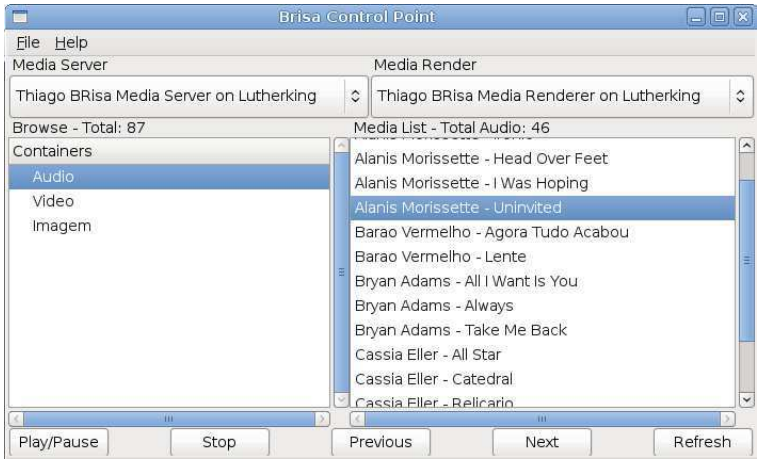
```
1 '''Implementação Browse do BRisa-Media-Server'''
2 def _soap_Browse(self, *args):
3         (object_id, browse_flag, filter, starting_index, requested_count,
4           sort_criteria, set_item_user) = args
5          #Outros trechos do código ocultados
6
7          for user_item in set_item_user:
8                  new_items = self.tfidf.calc(user_item, elements)
9                  #new_items é o conjunto de items similares a user_item
10                 for item in new_items:
11                         didl.add_item(item)
12
13         return didl.to_string()
```

Listagem 11: Ação *Browse* da aplicação BRisa Media Server com suporte a recomendação de conteúdos multimídia.

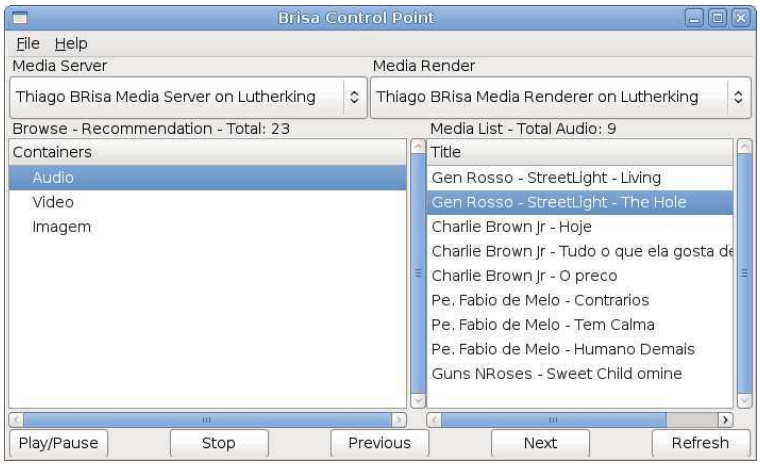
Estão ilustrados nas Figuras 5.4(a) e 5.4(b) os pontos de controle após a invocação da ação *Browse*. No primeiro caso, buscou-se invocar a ação considerando o mecanismo atualmente utilizado pelo padrão UPnP, ou seja, sem quaisquer processos de autenticação e personalização de serviços. Obviamente, todo o conteúdo disponível no servidor de mídias (87 conteúdos) foi retornado para o ponto de controle. Por outro lado, ao considerar o método *TF-IDF*, este número total foi reduzido em 3 vezes (23 conteúdos), o que mostra que além de oferecer conteúdos com base no perfil do usuário, também é possível evitar que o usuário seja sobrecarregado com dados, em tese, desnecessários.

5.2.2 Controle de Acesso a Dispositivos UPnP

O cenário discutido a seguir apresenta uma abordagem para realizar controle de acesso a dispositivos UPnP, mais precisamente a impressoras baseadas na especificação *UPnP Printer Device*. Devido à indisponibilidade de uma impressora com suporte ao padrão UPnP, o presente estudo de caso buscou definir um dispositivo UPnP de tal modo que mecanismos



(a) Requisição a ação Browse sem processos de filtragens de informações



(b) Requisição a ação Browse utilizando filtragens de informação

Figura 5.4: Navegação por conteúdos multimídia.

de impressão sejam disponibilizados através de serviços, porém, realizando-se requisições às impressoras reais através da API CUPS (*C Unix Printing System*).

Este cenário está ilustrado na Figura 5.5. Um servidor de impressão UPnP funciona basicamente como um *gateway* entre impressoras comuns e pontos de controle UPnP. Para permitir a impressão de conteúdos, o servidor comunica-se com impressoras comuns através do protocolo IPP (*Internet Printing Protocol*) por intermédio da API CUPS. Pontos de controle invocam ações UPnP de impressão disponíveis pelo servidor e que estão em conformidade com a especificação *UPnP Printer Device*, realizando controle de acesso com base nas políticas definidas.

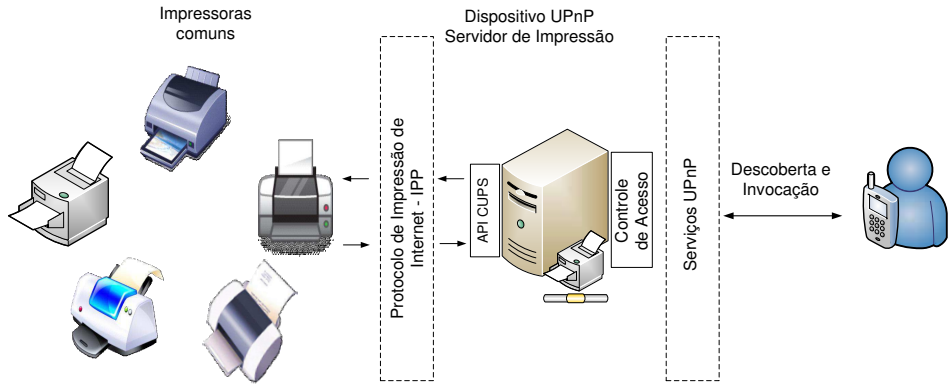


Figura 5.5: Cenário de estudo de caso: arquitetura de acesso e autorização à impressoras UPnP.

Definição da Política

Considere a seguinte política de acesso: *Usuários com atributos "role" e "supervised_by", cujos valores são "Mestrado" e "Hyggo" ou "Angelo", respectivamente, podem imprimir na impressora com UDN igual a "33220d90-655-4557-a207-572748f679" apenas entre 9:00 e 17:00*. Este estudo de caso utilizou o processo de definição e distribuição de políticas de controle de acesso discutidos na Seção 3.6.4.

Políticas de controle de acesso utilizadas na especificação UPnP-UP são definidas através da linguagem de marcação XACML (*eXtensible Access Control Markup Language*) [54], a qual possui extensões para os modelos de controle de acesso ABAC e RBAC. A política definida inicialmente está descrita nas Listagens 12, 13 e 14, subdivididas para facilitar a compreensão do leitor.

Inicialmente, a política define os potenciais usuários que podem acessar os recursos de uma impressora. Nas linhas 3 e 4 da Listagem 12 está definido o papel que os usuários devem exercer na rede para acessar o dispositivo. Já nas linhas 8 e 9 define-se o atributo que o usuário deve possuir.

```
1 <Subjects>
2    <Subject>
3       <AttributeValue
4 DataType="http://www.w3.org/2001/XMLSchema#string">Mestrado</
    AttributeValue>
5       <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml
            :2.0:attribute:role" DataType="http://www.w3.org/2001/XMLSchema#
```

```
        string"/>
6     </Subject>
7     <Subject>
8        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
             Angelo</AttributeValue>
9        <SubjectAttributeDesignator
10 AttributeId="urn:oasis:names:tc:xacml:2.0:attribute:supervised_by"
11 DataType="http://www.w3.org/2001/XMLSchema#string"/>
12    </Subject>
13    <!-- Subject ocultado para valor "Hyggo" -->
14 </Subjects>
```

Listagem 12: Política de controle de acesso: definição dos atributos de um usuário com permissão de acesso.

Uma vez definidos os atributos dos sujeitos da ação, estão descritos na política o recurso a ser controlado e a operação sobre o recurso. Nas linhas 3 e 4 da Listagem 13 está definido o identificador (UDN) da impressora UPnP e nas linhas 9 e 10 a ação UPnP a ser invocada (*createJob*).

```
1 <Resources>
2    <Resource>
3       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI
            ">33220d90-e655-4557-a207-572748f679</AttributeValue>
4       <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml
            :1.0:resource:UDN" DataType="http://www.w3.org/2001/XMLSchema#
            anyURI"/>
5    </Resource>
6 </Resources>
7 <Actions>
8    <Action>
9       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
            createJob</AttributeValue>
10      <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml
            :1.0:action:action-name" DataType="http://www.w3.org/2001/
            XMLSchema#string"/>
11   </Action>
12 </Actions>
```

Listagem 13: Política de controle de acesso: definição do recurso e da operação sobre o recurso.

Por fim, é importante também considerar as condições do ambiente para realizar o con-

trole de acesso. A política definida para este estudo de caso restringe as condições temporais de acesso ao recurso. Nos intervalos das linhas 3 a 6 e 7 a 10 estão definidos o horário inicial e final, respectivamente.

```
1 <!-- Only allow access from 9am to 5pm -->
2 <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
3   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-greater-
        than-or-equal">
4     <EnvironmentAttributeSelector DataType="http://www.w3.org/2001/
          XMLSchema#time" AttributeId="urn:oasis:names:tc:xacml:1.0:
          environment:current-time"/>
5     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time
          ">09:00:00</AttributeValue>
6   </Apply>
7   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-less-
        than-or-equal">
8     <EnvironmentAttributeSelector DataType="http://www.w3.org/2001/
          XMLSchema#time" AttributeId="urn:oasis:names:tc:xacml:1.0:
          environment:current-time"/>
9     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time
          ">17:00:00</AttributeValue>
10  </Apply>
11 </Condition>
```

Listagem 14: Política de controle de acesso: definição das condições de acesso ao recurso.

Os trechos das Listagens 12, 13 e 14 são partes de uma única política de controle de acesso e são organizados como descrito na Listagem 15. Além de definir o resultado final do processamento da política através do atributo *Effect*, a *tag* <*Rule*> contém a descrição da política, bem como os requisitos descritos nas Listagens supracitadas.

```
1 <Policy xmlns="urn:oasis:names:tc:xacml:1.0:policy">
2   <Rule RuleId="urn:oasis:names:tc:xacml:2.0:upnp-up:IIA003:rule" Effect
        ="Permit">
3     <Description>Usuários com atributos ‘‘role’’ e ‘‘supervised_by’’,
          cujos
4 valores são ‘‘Mestrado’’ e "Angelo" ou ‘‘Hyggo’’, respectivamente, podem
5 imprimir na impressora com UDN igual a ‘‘33220d90-655-4557-a207-572748
    f679’’
```

```
apenas entre 9:00 e 17:00.
      </Description>
      <Target>
         <Subjects> <!-- Definido na Listagem 12 --> </Subjects>
         <Resources> <!-- Definido na Listagem 13 --> </Resources>
         <Actions> <!-- Definido na Listagem 13 -->  </Actions>
      </Target>
      <Condition> <!-- Definido na Listagem 14 --> </Condition>
   </Rule>
</Policy>
```

Listagem 15: Política de controle de acesso a impressoras UPnP.

Servidor de Impressão UPnP

O servidor de impressão foi implementado utilizando o *framework* BRisa e considerando as alterações arquiteturais discutidas na Seção 5.1. Neste estudo de caso, duas ações UPnP foram implementadas neste dispositivo: *getAvailablePrinters* e *createJob*. A ação *getAvailablePrinters* retorna as impressoras disponíveis na rede e a ação *createJob* realiza a impressão propriamente dita.

O serviço UPnP *PrinterService*, descrito na Listagem 16, disponibiliza os métodos *before_call* e *after_call*, bem como a ação UPnP *createJob*. Entre as linhas 8 e 13 é realizado o controle de acesso proposto neste trabalho. A API *enterprise-xacml*[5] foi utilizada para possibilitar a interpretação das políticas de controle de acesso. Mais especificamente na linha 12, a API é invocada, interpretando a política definida anteriormente e retornando o resultado da inferência. Após a invocação do método *before_call*, realizado pelo módulo *Controlador* do BRisa, a ação requisitada pelo ponto de controle pode ser invocada.

Neste estudo de caso, não há políticas de controle de acesso para a ação *getAvailablePrinters*, o que permite que qualquer usuário possa obter a lista de impressoras disponíveis na rede. Por outro lado, a política definida na Listagem 15 para a ação UPnP *createJob* restringe o acesso a um grupo de usuários com determinados atributos. Desta forma, uma invocação à ação *createJob* é interceptada pelo método *before_call* e liberada apenas se o usuário assume estas características. Caso liberada, a ação *createJob* é invocada, a qual obtém a impres-

[5] http://www.google.code.com/enterprise-xacml

sora selecionada e o arquivo a ser impresso. Independentemente do resultado do controle de acesso, o método *after_call* é invocado pelo módulo *Controlador* para finalizar a requisição corrente do ponto de controle. Entre as linhas 16 e 22, verifica-se o resultado do controle de acesso e realiza-se mecanismos de *log*.

```
1  class PrinterService(Service):
2      '''Construtor'''
3      def __init__(self):
4          #Trechos do código omitidos
5          pass
6
7      '''Invocado pelo módulo Controlador antes de quaisquer ações do
           serviço PrinterService. O controle de acesso com base nos dados do
            usuário é realizado pelo método before_call.'''
8      def before_call(self, *args, **kwargs):
9          uuid = kwargs["UUID"]
10         action_name = kwargs["method_name"]
11         current_user = self.device.users[uuid]
12         access_result = self.authorization_manager.fire(current_user,
               action_name)
13         return access_result
14
15     '''Realiza tarefas pós-execução de quaisquer ação do serviço
           PrinterService.'''
16     def after_call(self, *args, **kwargs):
17         if kwargs["access_result"]:
18             access = "permited"
19         else:
20             access = "denied"
21         self.log.info("User:", self.device.users(kwargs["UUID"]).name)
22         self.log.info("Access to ", kwargs["method_name"], "was", access)
23
24     '''Ação createJob realiza a impressão numa impressora comum
           selecionada.'''
25     def createJob(self, *args, **kwargs):
26         file = kwargs["file"]
27         printer_name = kwargs["printer_name"]
```

```
28          print_result = cups.print(printer_name, file)
29          return {"printResult": print_result}
```

Listagem 16: Serviço UPnP do servidor de impressão UPnP.

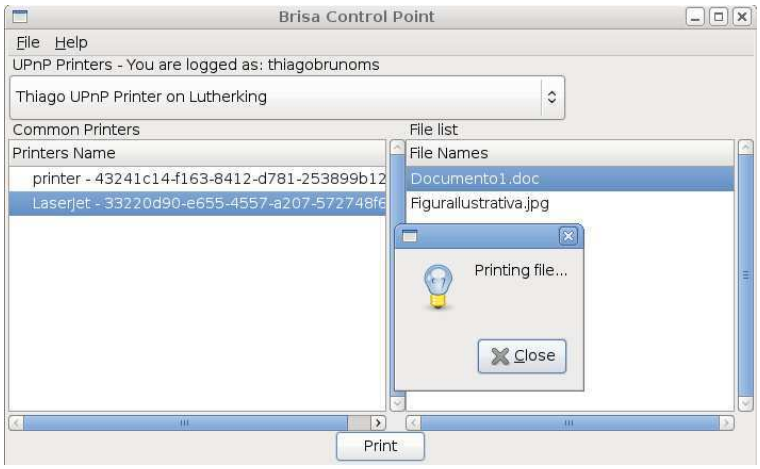
De forma resumida, o perfil do usuário utilizado para este estudo de caso está descrito na Listagem 17. As *tags* <embedded:roles> e <embedded:supervised_by> correspondem aos requisitos necessários que são analisados pela política de controle de acesso do ambiente "Embedded", o qual está definido como atributo da *tag* <up:external_profile>. Como descrito na Seção 3.6.1, o perfil abaixo corresponde a um subperfil inerente ao ambiente supracitado.

```
1 <?xml version="1.0" encoding="UTF-8">
2 <up:profile up-id="TBMS1302" xmlns:up="urn:schemas-upnp-up-org:up-1-0"
3 xmlns:embedded="urn:schemas-upnp-up-org/embedded:up-1-0">
4
5  <up:external_profile location="Embedded">
6   <up:personal_data id="personal_profile">
7     <up:username>thiagobrunoms</up:username>
8     <!-- Outras informações do perfil ocultadas -->
9
10    <embedded:roles>
11         <embedded:role>Mestrado<up:role>
12    <embedded:roles>
13
14    <embedded:supervised_by>
15         <embedded:supervise>Angelo</embedded:supervise>
16         <embedded:supervise>Hyggo</embedded:supervise>
17    <embedded:supervised_by>
18   </up:personal_data>
19  </up:external_profile>
20 </up:profile>
```

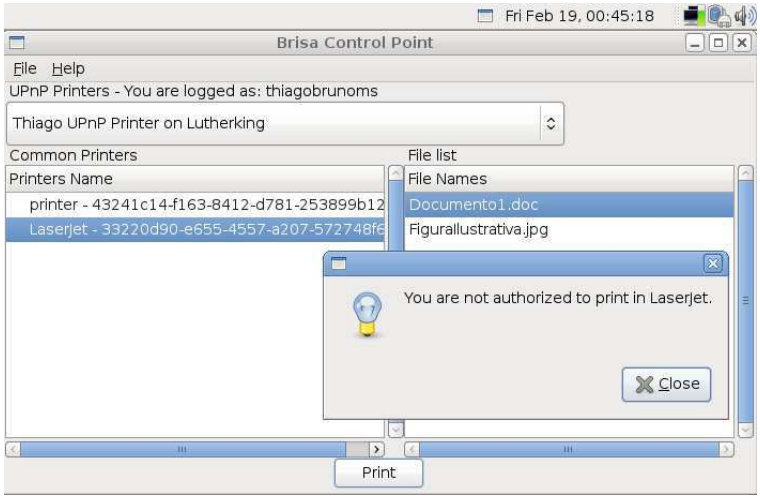
Listagem 17: Estudo de Caso: perfil de um usuário para ambiente "Embedded".

Nas Figuras 5.6(a) e 5.6(b) estão ilustrados os processos de seleção de dispositivos de impressão (descrevendo os nomes e os identificadores UDN de cada dispositivo) e as requisições de impressão realizadas pelo usuário com *login* "thiagobrunoms" para o servidor de impressão *Thiago UPnP Printer*. No primeiro cenário, o acesso à impressora foi permitido

ao usuário requisitante. Por outro lado, devido ao horário atual não constar entre o intervalo de tempo estabelecido pela política de controle de acesso e considerando que o usuário possui o valor da *tag* <supervised_by> diferente do esperado (definido como "Leandro") o usuário não obteve a permissão de acesso ao dispositivo com o UDN definido.



(a) Requisição permitida para a impressão (ação *createJob*)



(b) Requisição bloqueada para a impressão (ação *createJob*)

Figura 5.6: Solicitação de impressão utilizando a ação UPnP *createJob*.

5.3 Considerações sobre os Estudos de Casos

É importante salientar que o objetivo dos estudos de caso discutidos nesta seção foi apresentar a aplicabilidade do processo de filtragem de informações, e de definição de políticas de

controle de acesso. Buscou-se demonstrar que, através da proposta deste trabalho, é possível construir serviços UPnP personalizáveis e com suporte à controle de acesso com base em informações de contexto dos usuários e do ambiente.

A vulnerabilidade que pode ser explorada nos dispositivos *UPnP Gateway Device*, descrito na Seção 1.1, pode ser solucionada através dos mesmos procedimentos adotados no controle de acesso a dispositivos de impressão, considerando os atributos do usuário que solicita, por exemplo, um mapeamento de portas da Internet.

Capítulo 6

Considerações Finais

Neste trabalho propõe-se uma especificação para autenticação e autorização de usuários em ambientes de Computação Pervasiva, tendo como ponto de partida o padrão de conectividade UPnP. Com efeito, foi proposta uma arquitetura de conectividade para permitir a provisão de serviços UPnP personalizáveis e com suporte a controle de acesso, tendo como diferencial o foco nas informações dos usuários. A abordagem proposta mostra-se pouco invasiva com relação à atual versão do padrão UPnP graças aos esforços do consórcio *UPnP Forum* em proporcionar flexibilidade e extensibilidade na adição de novos dispositivos e serviços ao padrão.

Os objetivos traçados neste documento foram alcançados através da especificação UPnP-UP, a qual está disponível para ser integrada em soluções baseadas no padrão UPnP e é fruto de uma pesquisa com resultados publicados em congressos e conferências de âmbito nacional e internacional [78][77][76]. Tais publicações foram de real valia perante a comunidade científica para os avanços realizados durante o desenvolvimento deste trabalho. Através dos estudos de casos, foi possível também validar a solução proposta e apresentar uma abordagem de implementação em diferentes domínios de aplicação.

Por fim, espera-se que a especificação UPnP-UP seja a base para a concepção e o desenvolvimento de novas soluções para a personalização e a segurança de serviços pervasivos, contribuindo também para a disseminação do padrão UPnP e para futuras pesquisas no domínio da Computação Pervasiva.

6.1 Contribuições da Pesquisa

As contribuições deste trabalho podem ser constatadas em duas vertentes: especificação e implementação. No primeiro caso, tem-se a disponibilização das especificações UPnP-UP propriamente dita, as quais podem ser observadas nos Apêndices A e B. No segundo caso, tem-se alterações realizadas no *framework* BRisa para o desenvolvimento de soluções UPnP com suporte à personalização e controle de acesso, disponibilizando para desenvolvedores um *framework* com surpote aos requisitos inerentes à especificação UPnP-UP.

Arquiteturas e especificações baseadas no padrão UPnP devem, obrigatoriamente, disponibilizar um documento com os requisitos necessários para manter a interoperabilidade entre diferentes soluções. Este documento intitula-se DCP, acrônimo de *Device Control Protocol*, definido pelo *UPnP Forum*. Portanto, como resultado desta pesquisa, está disponível no Apêndice A o documento da arquitetura de conectividade UPnP-UP, detalhando os requisitos de autenticação e autorização discutidos neste trabalho. O DCP proposto provê as informações necessárias para o desenvolvimento de serviços UPnP com suporte à extensão UPnP-UP, o que naturalmente permite a interoperabilidade entre diferentes soluções.

Da mesma forma como especificada a arquitetura de conectividade UPnP-UP, foi escrita a especificação do servidor de perfis UPnP-UP. Estão disponíveis no Apêndice B os requisitos necessários para contemplar serviços de autenticação e autorização de usuários através do servidor de perfis UPnP-UP. São detalhados neste DCP os serviços e as respectivas ações UPnP que o dispositivo deve disponibilizar, bem como os eventos UPnP gerados pelo servidor de perfis. Este documento é uma opção a ser utilizada pelos desenvolvedores de servidores de perfis para obter mais detalhes dos requisitos de implementação, permitindo que pontos de controle e demais dispositivos UPnP sejam interoperáveis com o servidor de perfis.

Finalmente, a segunda contribuição deste trabalho é a implementação e a integração dos requisitos de autenticação e autorização propostos neste trabalho ao *framework* BRisa. As alterações realizadas na versão *BRisa Python* são compatíveis com as versões anteriores do referido *framework*. O BRisa é capaz de detectar se existem serviços inerentes à especificação UPnP-UP e, com base nestas informações, ativar os módulos necessários discutidos na Seção 5.1, abstraindo do desenvolvedor a responsabilidade de implementar requisitos como

a checagem dos números de sequência e descoberta de servidores de perfis.

6.2 Trabalhos Futuros

Um problema existente na maioria dos sistemas de computação distribuída é o rastreamento de comunicações para montar um perfil de preferências de um usuário específico. Esse rastreamento é uma violação explícita da privacidade, em especial se for realizada sem informar ao usuário [80]. Todavia, é inevitável disponibilizar serviços personalizados em ambientes de Computação Pervasiva sem fornecer informações de contexto, os quais podem naturalmente revelar o perfil das pessoas inseridas no ambiente.

É na perspectiva de privacidade de dados dos usuários que se espera evoluir este trabalho, permitindo que usuários disponibilizem suas informações com a garantia de que elas estarão protegidas contra entidades que não possuem autorização para acessar o conteúdo. Para garantir esta privacidade, é importante que em ambientes caracterizados por serviços pervasivos o usuário possa ter total controle sobre quais dados quer disponibilizar, para quais tarefas/situações, para quais usuários, por quanto tempo, e onde estarão armazenados. Neste contexto, apesar da especificação UPnP-UP permitir que o usuário determine o momento e o conteúdo do perfil que será disponibilizado aos dispositivos, ainda não é o suficiente para garantir que estas informações não serão violadas e utilizadas para comunicações indesejadas, como o envio de mala direta (*spam*) sem a permissão do usuário, por exemplo.

Bibliografia

[1] Klemets A., B. Da Costa, Walter Jr., and James T. UPnP Authentication and Authorization. `http://www.freepatentsonline.com/EP2078372A1.html`, Julho 2009.

[2] Presser A., Farrel Lee., and Lupton D. UPnP Device Architecture. `http://www.upnp.org/specs/arch/UPnP-DeviceArchitecture-v1.0-20060720.pdf`, 2008. Último acesso em, Maio 2008.

[3] Hadhoud Abdul Elminaam, Abdul Kader. Performance Evaluation of Symmetric Encryption Algorithms. *Communications of the IBIMA*, 8:280–286, 2009.

[4] Shivaun Albright, Tom Hastings, Harry Lewis, and Peter Zehler. Printer:1 Device Template. `http://upnp.org/standardizeddcps/documents/Printer_Definition_v1_20081015.pdf`, Agosto 2002.

[5] Steve Barker. The Next 700 Access Control Models or a Unifying Meta-model. In *SACMAT '09: Proceedings of the 14th ACM symposium on Access control models and technologies*, pages 187–196, New York, NY, USA, 2009. ACM.

[6] Jeroen Bekaert, Emiel De Kooning, and Herbert de Sompel. Representing Digital Assets Using MPEG-21 Digital Item Declaration. *Int. J. Digit. Libr.*, 6(2):159–173, 2006.

[7] Rakesh Bobba, Omid Fatemieh, Fariba Khan, Carl A. Gunter, and Himanshu Khurana. Using Attribute-Based Access Control to Enable Attribute-Based Messaging. In *Computer Security Applications Conference, 2006. ACSAC '06. 22nd Annual*, pages 403–413, Dezembro 2006.

[8] Robin Burke. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[9] C.-F. Chiasserini and R.R. Rao. Energy Efficient Battery Management. *Selected Areas in Communications, IEEE Journal on*, 19(7):1235–1245, Julho 2001.

[10] K. Chiu, M. Govindaraju, and R. Bramley. Investigating the Limits of SOAP Performance for Scientific Computing. *11th IEEE International Symposium on High Performance Distributed Computing, 2002. HPDC-11 2002. Proceedings.*, pages 246–254, 2002.

[11] Don Coppersmith and Markus Jakobsson. Almost optimal hash sequence traversal. In *Proceedings of the Fourth Conference on Financial Cryptography*. CSREA Press, 2002.

[12] Presidencia da Republica do Brasil. Medida Provisória nº 2.200-2, Agosto 2001.

[13] E. Damiani, S.D.C. di Vimercati, and P. Samarati. New Paradigms for Access Control in Open Environments. In *Signal Processing and Information Technology, 2005. Proceedings of the Fifth IEEE International Symposium on*, pages 540–545, Dec. 2005.

[14] Departamento de Defesa dos Estados Unidos. Trusted Computer System Evaluation Criteria (Orange Book). `http://csrc.nist.gov/publications/history/dod85.pdf`, 1983.

[15] M.A.C. Dekker, J. Crampton, and S. Etalle. RBAC Administration in Distributed Systems. In *SACMAT '08: Proceedings of the 13th ACM symposium on Access control models and technologies*, pages 93–102, New York, NY, USA, 2008. ACM.

[16] A.M. Deshpande, M.S. Deshpande, and D.N. Kayatanavar. FPGA Implementation of AES Encryption and Decryption. pages 1 –6, June 2009.

[17] T. Dierks and C. Allen. The TLS Protocol. http://www.ietf.org/rfc/rfc2246.txt. Último Acesso em Dezembro, 2009.

[18] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[19] Tamara Dinev. Why Spoofing is Serious Internet Fraud. *Commun. ACM*, 49(10):76–82, 2006.

[20] Didier Donsez. On-Demand Component Deployment in the UPnP Device Architecture. *4th IEEE Consumer Communications and Networking Conference, 2007. CCNC 2007.*, pages 920–924, Janeiro 2007.

[21] Carl Ellison. DeviceSecurity:1 Service Template. `http://www.upnp.org/standardizeddcps/documents/DeviceSecurity_1.0cc_001.pdf`, 2003. Último acesso em Dezembro, 2008.

[22] Carl Ellison. SecurityConsole:1 Service Template. `http://www.upnp.org/standardizeddcps/documents/SecurityConsole_1.0cc.pdf`, 2003. Último acesso em Dezembro, 2008.

[23] David F. Ferraiolo, John F. Barkley, and D. Richard Kuhn. A Role Based Access Control Model and Reference Implementation Within a Corporate Intranet. *ACM Transactions on Information and System Security*, 2:34–64, 1999.

[24] Warwick Ford. Advances in Public-Key Certificate Standards. *SIGSAC Rev.*, 13(3):9–15, 1995.

[25] Christian Friberg and Achim Held. Support for Discretionary Role Based Access Control in ACL-Oriented Operating Systems. In *RBAC '97: Proceedings of the second ACM workshop on Role-based access control*, pages 83–94, New York, NY, USA, 1997. ACM.

[26] Carles Garrigues, Nikos Migas, William Buchanan, Sergi Robles, and Joan Borrell. Protecting Mobile Agents from External Replay Attacks. *J. Syst. Softw.*, 82(2):197–206, 2009.

[27] A.L.V. Guedes, D.F.S. Santos, J.L. do Nascimento, L.M. Sales, A. Perkusich, and H.O. Almeida. BRisa UPnP A/V Framework. *International Conference on Consumer Electronics, 2008. ICCE 2008. Digest of Technical Papers.*, pages 1–2, Janeiro 2008.

[28] Juan Ignacio Vázquez Gómez. *A Reactive Behavioural Model For Context-Aware Semantic Devices*. PhD thesis, Universidad de Deusto, 2007.

[29] E. Halepovic and R. Deters. The Costs of Using JXTA. *Third International Conference on Peer-to-Peer Computing, 2003. (P2P 2003)*, pages 160–167, Setembro 2003.

[30] S. Harrusi, A. Averbuch, and A. Yehudai. XML Syntax Conscious Compression. *Proceedings on Data Compression Conference, 2006. DCC 2006.*, pages 10–411, Março 2006.

[31] Rui Jose Helder Pinto. Pervasive Location-Based Systems: The Fundamental Challenges between Vision and Reality. *International Journal of Pervasive Computing and Communications*, 1:7–12, 2005.

[32] Urs Hengartner and Peter Steenkiste. Implementing Access Control to People Location Information. In *SACMAT '04: Proceedings of the ninth ACM symposium on Access control models and technologies*, pages 11–20, New York, NY, USA, 2004. ACM.

[33] Vincent Hourdin, Jean-Yves Tigli, Stéphane Lavirotte, Gaëtan Rey, and Michel Riveill. Context-Sensitive Authorization in Interaction Patterns. In *Mobility '09: Proceedings of the 6th International Conference on Mobile Technology, Application; Systems*, pages 1–8, New York, NY, USA, 2009. ACM.

[34] Yih-Chun Hu, Markus Jakobsson, and Adrian Perrig. Efficient Constructions for One-Way Hash Chains. In *IN APPLIED CRYPTOGRAPHY AND NETWORK SECURITY (ACNS*, pages 423–441, 2005.

[35] Pan Hui, Onshun Chau, Xiaoshan Liu, and V.O.K. Li. A Peer-to-Peer Jini Architecture for Pervasive Multimedia. *Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th*, 5:3160–3164 Vol. 5, Setembro 2004.

[36] Dieter Hutter, Günter Müller, Werner Stephan, and Markus Ullmann, editors. *Security in Pervasive Computing, First International Conference, Boppard, Germany, March 12-14, 2003, Revised Papers*, volume 2802 of *Lecture Notes in Computer Science*. Springer, 2004.

[37] Prakash Iyer. InternetGatewayDevice:1 Device Template. `http://upnp.org/standardizeddcps/documents/UPnP_IGD_1.0.zip`, Novembro 2001.

[38] Markus Jakobsson. Fractal Hash Sequence Representation and Traversal. In *In IEEE International Symposium on Information Theory*, pages 437–444, 2002.

[39] Insu Jeong, Hyunjun Choi, and Joongsoo Ma. Study on Address Allocation in Ad-Hoc Networks. In *Proceedings of the Fourth Annual ACIS International Conference on Computer and Information Science*, pages 604–609. IEEE Computer Society, 2005.

[40] Tom Erik Julsrud and Per Morten Schiefloe. The Development, Distribution and Maintenance of Trust in Distributed Work Groups; A Social Network Approach. *Int. J. Netw. Virtual Organ.*, 4(4):351–368, 2007.

[41] Yang Jun, Li Na, and Ding Jun. A Design and Implementation of High-Speed 3DES Algorithm System. pages 175 –178, Dec. 2009.

[42] L. Kagal, T. Finin, and A. Joshi. Trust-Based Security in Pervasive Computing Environments. *Computer*, 34(12):154–157, Dec 2001.

[43] Raymond Kammer. Data Encryption Standard (DES). `http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf`, October 1999.

[44] Przemys Kazienko. Web-Based Recommender Systems and User Needs –The Comprehensive View. In *Proceeding of the 2008 conference on New Trends in Multimedia and Network Information Systems*, pages 243–258, Amsterdam, The Netherlands, The Netherlands, 2008. IOS Press.

[45] Kristian Ellebæk Kjær. A Survey of Context-Aware Middleware. In *SE'07: Proceedings of the 25th conference on IASTED International Multi-Conference*, pages 148–155, Anaheim, CA, USA, 2007. ACTA Press.

[46] Neal Koblitz. Elliptic Curve Cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[47] I. N. Kovalenko and A. I. Kochubinskii. Asymmetric Cryptographic Algorithms. *Cybernetics and Sys. Anal.*, 39(4):549–554, 2003.

[48] Leslie Lamport. Password Authentication with Insecure Communication. *ACM Communication*, 24(11):770–772, 1981.

[49] Deok Gyu Lee, Geon Woo Kim, Jong Wook Han, Young-Sik Jeong, and Doo-Soon Park. Smart Environment Authentication: Multi-Domain Authentication, Authorization, Security Policy for Pervasive Network. In *Ubiquitous Multimedia Computing, 2008. UMC '08. International Symposium on*, pages 99–104, Oct. 2008.

[50] Jin-Cherng Lin, Jan-Min Chen, and Cheng-Hsiung Liu. An Automatic Mechanism for Adjusting Validation Function. *Advanced Information Networking and Applications*, pages 602–607, 2008.

[51] D. Liu and P. Ning. Efficient Distribution of Key Chain Commitments for Broadcast Authentication in Distributed Sensor Networks. Technical report, Raleigh, NC, USA, 2002.

[52] Emerson Loureiro, Glauber Ferreira, Hyggo Almeida, and Angelo Perkusich. Pervasive Computing: What is it Anyway? *In M. Lytras and A. Naeve (Eds.), Ubiquitous and Pervasive Knowledge and Learning Management: Semantics, Social Networking and New Media to their full potential*, pages 1–34, 2007.

[53] Emerson Loureiro, Loreno Oliveira, and Hyggo Almeida. Improving Flexibility on Host Discovery for Pervasive Computing Middlewares. *MPAC '05: Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*, pages 1–8, 2005.

[54] Pietro Mazzoleni, Bruno Crispo, Swaminathan Sivasubramanian, and Elisa Bertino. XACML Policy Integration Algorithms. *ACM Trans. Inf. Syst. Secur.*, 11(1):1–29, 2008.

[55] Henk Meijer and Selim Akl. Digital Signature Schemes for Computer Communication Networks. In *SIGCOMM '81: Proceedings of the seventh symposium on Data communications*, pages 37–41, New York, NY, USA, 1981. ACM.

[56] C. Metz. AAA Protocols: Authentication, Authorization, and Accounting for the Internet. *Internet Computing, IEEE*, 3(6):75–79, Nov/Dec 1999.

[57] Pradosh Kumar Mohapatra. Public Key Cryptography. *Crossroads*, 7(1):14–22, 2000.

[58] A. Moussa. Data Encryption Performance Based on Blowfish. pages 131 –134, June 2005.

[59] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J. D. Tygar. SPINS: Security Protocols for Sensor Networks. In *Wireless Networks*, pages 189–199, 2001.

[60] Saverio Perugini, Marcos André Gonçalves, and Edward A..Fox. Recommender Systems Research: A Connection-Centric Survey. *Journal of Intelligent Information Systems*, 23(2):107–143, 2004.

[61] J. Reagle. XML Signature Requirements. `http://www.ietf.org/rfc/rfc2246.txt`, 2000.

[62] Kasim Rehman, Frank Stajano, and George Coulouris. An Architecture for Interactive Context-Aware Applications. *IEEE Pervasive Computing*, 6(1):73–80, 2007.

[63] John Ritchie. UPnP AV Architecture:1. `http://upnp.org/resources/upnpresources.zip`, Setembro 2008.

[64] Oriana Riva and Jaakko Kangasharju. Challenges and Lessons in Developing Middleware on Smart Phones. *Computer*, 41(10):23–31, 2008.

[65] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[66] Dorothy Elizabeth Robling Denning. *Cryptography and Data Security*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1982.

[67] Goutam Kumar Saha. Software Based Fault Tolerance: A Survey. *Ubiquity*, 5(Julho):1–1, 2006.

[68] Christoph Sahm and Hans J. Langels. DimmableLight:1 Device Template. `http://upnp.org/standardizeddcps/documents/DimmableLight1.0cc.pdf`, Novembro 2003.

[69] M. Satyanarayanan. Pervasive Computing: Vision and Challenges. *Personal Communications, IEEE*, 8(4):10–17, Aug 2001.

[70] Bruce Schneier. *Applied Cryptography*, volume 1. Wiley, October 1996.

[71] Yaron Sella. On the Computation-Storage Trade-Offs of Hash Chain Traversal. In *In Proceedings of Financial Cryptography*, 2003.

[72] Lisa Sherwin. UPnP Specifications Named International Standard for Device Interoperability for IP-Based Network Devices. `http://www.upnp.org/news/documents/UPnPForum_02052009.pdf`, Fevereiro 2009.

[73] Li Shiqun, Shane Balfe, Zhou Jianying, and Chen Kefei. Trust Based Pervasive Computing. *Wuhan University Journal of Natural Sciences*, 11:1477–1480, 2006.

[74] Larry Stickler. Hvac_System:1 Device Template. `http://www.upnp.org/download/standardizeddcps/hvac/HVAC_System_201.0.pdf`, Maio 2003.

[75] Paul Syverson. A Taxonomy of Replay Attacks. In *In Proceedings of the 7th IEEE Computer Security Foundations Workshop*, pages 187–191. Society Press, 1994.

[76] Sales T., Sales L., Almeida H., and Perkusich A. Enabling User Authentication and Authorization to Support Context-Aware UPnP Applications. In *Simpósio Brasileiro de Sistemas Multimídia e Web - WebMedia*, 2009.

[77] Sales T., Sales L., Almeida H., and Perkusich A. Towards to UPnP-UP: Developing Context-Aware UPnP Applications. In *XXIX Congresso da Sociedade Brasileira de Computaçao*, 2009.

[78] Sales T., Sales L., Pereira M., Almeida H., Perkusich A., Gorgonio K., and Sales M. Towards the UPnP-UP: Enabling User Profile to Support Customized Services in UPnP Networks. In *UBICOMM: The Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies.*, pages 206 –211, Outubro 2008.

[79] R. Fielding T. Berners-Lee. Hypertext Transfer Protocol – http/1.1. `http://www.ietf.org/rfc/rfc2616.txt`, Junho 1999.

[80] Andrew S. Tanennaum and Maarten Van Steen. *Sistemas Distribuídos - Princípios e Paradigmas*. Wiley, 2ª Edição edition, 2007.

[81] Yuzhe Tang, Shuigeng Zhou, and Jianliang Xu. LIGHT: A Query-Efficient Yet Low-Maintenance Indexing Scheme Over DHTs. *IEEE Trans. on Knowl. and Data Eng.*, 22(1):59–75, 2010.

[82] Joacim Tullberg. DigitalSecurityCamera:1 Device Template. `http://upnp.org/standardizeddcps/documents/Digital_Camera1.0_000.pdf`, 2005. Último acesso em Dezembro/2008.

[83] Márcio Araújo Varchavsky, Eduardo Martins Guerra, and Clóvis Torres Fernandes. Modelo de Controle de Acesso para uma Arquitetura Orientada a Serviços Visando a Integração de Aplicações de Comando e Controle. In *IX Simpósio Internacional de Guerra Eletrônica*, 2007.

[84] Lingyu Wang, Duminda Wijesekera, and Sushil Jajodia. A Logic-Based Framework for Attribute Based Access Control. In *FMSE '04: Proceedings of the 2004 ACM workshop on Formal methods in security engineering*, pages 45–55, New York, NY, USA, 2004. ACM.

[85] Mark Weiser. The Computer for the 21st Century. *Scientific American*, 265(3):66–75, Setembro 1991.

[86] Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. Interpreting TF-IDF Term Weights as Making Relevance Decisions. *ACM Trans. Inf. Syst.*, 26(3):1–37, 2008.

[87] J.L. Wuytack, S. da Silva, F. Catthoor, G. de Jong, and C. Ykman-Couvreur. Memory Management for Embedded Network Applications. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 18(5):533–544, Maio 1999.

[88] Eric Yuan and Jin Tong. Attributed Based Access Control (ABAC) for Web Services. In *ICWS '05: Proceedings of the IEEE International Conference on Web Services*, pages 561–569, Washington, DC, USA, 2005. IEEE Computer Society.

[89] Yasir Zahur and T. Andrew Yang. Wireless LAN Security and Laboratory Designs. *J. Comput. Small Coll.*, 19(3):44–60, 2004.

Apêndice A

DCP da Arquitetura UPnP-UP

UPnP User Profile Architecture:1

For UPnP™ Version *1.1*
Date: February 26, 2010

This UPnP User Profile Architecture (UPnP-UP) has been made available to UPnP Members as a proposal architecture for a new generation of electronic devices for pervasive computing environment. This architecture proposal has not been adopted as a Standardized DCP yet by the Steering Committee of the UPnP™ Forum. UPnP™ Forum Members have rights and licenses defined by Section 3 of the UPnP™ Forum Membership Agreement to use and reproduce the proposed architecture in UPnP™ Compliant Devices. All such use is subject to all of the provisions of the UPnP™ Forum Membership Agreement.

THE UPNP FORUM TAKES NO POSITION AS TO WHETHER ANY INTELLECTUAL PROPERTY RIGHTS EXIST IN THE PROPOSED DEVICE TEMPLATES, IMPLEMENTATIONS OR IN ANY ASSOCIATED TEST SUITES. THE PROPOSED ARCHITECTURE, IMPLEMENTATIONS AND ANY ASSOCIATED TEST SUITESARE PROVIDED "AS IS" AND "WITH ALL FAULTS". THE UPNP FORUM MAKES NO WARRANTIES, EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE UPnP-UP ARCHITECTURE, IMPLEMENTATIONS AND ASSOCIATED TEST SUITES INCLUDING BUT NOT LIMITED TO ALL IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE, OF REASONABLE CARE OR WORKMANLIKE EFFORT, OR RESULTS OR OF LACK OF NEGLIGENCE.

Authors	Company
Thiago Sales	Embedded Systems and Pervasive Computing Laboratory
Leandro Sales	Federal University of Alagoas – Computing Institute
Hyggo Almeida	Embedded Systems and Pervasive Computing Laboratory
Angelo Perkusich	Embedded Systems and Pervasive Computing Laboratory

1. Overview and Scope

1.1. Introduction

This document describes the overall UPnP User Profile Architecture (UPnP-UP), which forms the foundation for the UPnP-UP Server and UPnP-UP Device templates. The UPnP-UP Architecture defines the general interaction between UPnP Control Points, UPnP User Profile Server, and UPnP devices. The UPnP-UP architecture enables user authentication and authorization for UPnP devices and applications in order to support access control and context-aware UPnP services based on users information.

In computer security, access control includes, among other features, the authentication and the authorization mechanisms. Identification and authentication are the processes of checking something (or someone) as authentic. In short, authentication is the basic building block of security. In the electronic world, the authentication of an entity can be processed by using shared secrets (including passwords), public key cryptography schemes, biometrics, and so forth. User identification and authentication in pervasive environments is also important due to the range of devices and services that users have access to. Currently, many solutions use IP addressing identification and authentication, which are not enough in pervasive computing. This model works well for traditional deskbound PC, where the user usually works with a static IP address in the same machine. On the other hand, in a pervasive computing environment, users have different devices and connect to different networks.

An authorization process can be seen as the mechanism to allow or deny an access to a set of available resources of a computational entity (called objects). If a subject (a user, a device etc.) tries to get access to these resources, they will be allowed if they have some degrees of permissions. Access control models can be divided into two classes: those based on the capability and those based on access control list (ACL). The former is a non-falsifiable reference (or capability) that a subject gains from an object to access it, which is analogous to have a password of a physical safe. The ACL-based access control allows a subject to gain access to an object if its identity is on a list associated with the object. The access control in pervasive computing should also take into account the context of the objects in the environment, such as current time and user activities. These requirements bring new challenges in the computer security domain due to the high heterogeneity of such objects.

Computers are increasingly entering into our environments, ubiquitously embedded into devices and appliances available in people's everyday lives. In addition, the diversity of people in pervasive environment requires novel security solutions in order to protect available resources (e.g. files, devices, services etc) in the network. To protect the environment from illegal accesses and a variety of threats, researchers have proposed many frameworks and architectures that can be used in pervasive applications [8, 19, 12, 4]. However, these solutions provide non-standard technologies, which bring challenges to achieve interoperability among other solutions. In order to

safely deploy UPnP services and appliances based on user profiles, it is required to build an architecture that uses UPnP-based technologies to be easily integrated with UPnP networks.

1.2. Goals

The UPnP-UP architecture was explicitly defined to meet the following goals:

- To provide user authentication and authorization based on his/her information.
- To allow developers in defining and building context-aware UPnP applications.
- To enable the mechanisms necessary for strong authentication, authorization, replay prevention and privacy of user information and UPnP SOAP actions.
- To provide different levels of authentication, providing a flexible mechanism to authenticate users in UPnP network according to the network needs.
- To provide mechanisms for user information management among distributed UPnP devices.

1.3. Cryptography Notation

To help explain the cryptographic aspects of some actions, a compact notation is used. The letters S, C and D are used to indicate the UPServer, a Control Point and a UPnP device, respectively. The public key of a party *E* is indicated by P_E^+, the private key by P_E^-. Thus PD^+ is the public key of the device, while PD^- is its private key. Session encryption keys (symmetric keys) are represented by K subscripted with both parties; thus KCD would be a session key used between the control point and the device.

Encryption is represented by [], while signing is represented by {}. These operations are prefixed by the key used. So, for example, $PD^+[PC^-\{m\}]$ represents a message m, signed with the private key of control point C, and encrypted with the public key of device D.

2. Architectural Overview

The UPnP-UP defines the User Profile Server device (UPServer) and a set of UPnP services and events in order to build context-aware and secure UPnP services. The architecture provides different levels of authentication, allowing flexibility according to the network needs. A user defines his/her profile and gets a signed profile from a certification authority. In a UPnP network, the user´s control point authenticates in the UPServer by sending a signed user profile) and receives a user identification based on UUID. After the authentication, the user can open sessions with UPnP devices by sending user identification and a profile based on the context of the target UPnP device (profile for AV, HVAC etc).

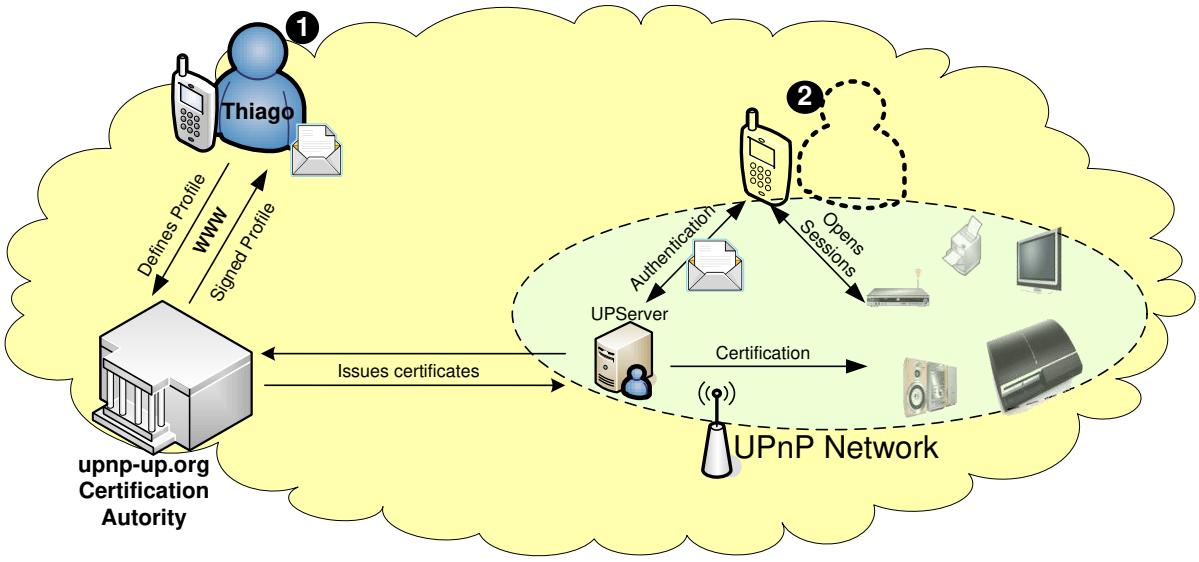


Figure 1: UPnP-UP Connectivity Model

2.1. User Profile

In order to provide secure and context-aware UPnP services based on users profiles, the UPnP-UP architecture provides a mechanism to manage the users profiles and how they are structured to allow interoperability among UPnP appliances. The user profile is divided into two parts: (1) personal information and (2) preferences information. The former can be used to provide access control, and the second to enable context-aware services.

The user profile is a structured, XML-based approach, also signed through XML-Signature. The profile includes the user name, list of email addresses, personal homepages, preferences for audio/video contents, HVAC, among other user features. To provide access control, a UPnP appliance can take into account the user personal information. Within this context, as a user is roaming through different UPnP networks, he/she can perform different roles for each environment. For instance, in a given environment, he/she can be a BOSS, and, in another environment, he/she can be the owner (of a house). So, the user profile also has a list of user profiles described as external links.

```
<root xmlns=”urn:schemas-upnp-up-org:up-1-0>
<userProfile id="TBMS7569">
      <personal_data>
            <name>Thiago Bruno Melo de Sales</name>
            <username>thiagobrunoms</username>
      </personal_data>
      <!--Other personal information were omitted-->

      <external_profiles>
            <profile name="Embedded" latitude="-7.212367" longitude="-
35.908268" />
            <profile name="Home at Campina Grande" latitude="-7.217444"
longitude="-35.906841" />
            <profile name="IC/UFAL" latitude="9.554554" longitude="-
35.774878" />
```

```
      </external_profiles>
      <!--Preferences information were omitted-->
      <!--Information about XML-Signature were omitted -->
</userProfile>
</root>
```

It is important to point out that the XML-Signature will only encrypt the personal information block. Users preferences information can be carried out without no encryption. However, the user can choose if he/she wants privacy of his/her information.

The user profile preferences are divided into containers. Each container has an id number and a name. The container is associated to a UPnP context, such as AV, HVAC etc.

```
<up:preferences xmlns="urn:schemas-upnp-up-org:up-1-0>
   <up:container_list>
      <up:container id=0 name="AV">
         //DIDL-based structure
      </up:container>
   </up:container_list>
</up:preferences>
```

The following table describes each container name and its identification and target context.

Table 1: Container information

Container Name	Id	Context
AV	*0*	*Audio, Video and Image*
HVAC	*1*	*Heating, ventilation, and Air-Conditioning*
DC	*2*	*Digital Camera*
QS	*3*	*Quality of Service*
PD	*4*	*Printer Device*
LP	*5*	*Low Power device*
Non-standard devices embedded by a UPnP vendor go here.	*TBD*	*TBD*

2.2. User Authentication

The UPnP-UP authentication model defines different levels in order to be used according to the network needs (Figure 2). There are 3 authentication levels and each of one includes the requirements of the less secure approach and adds the requirements to provide more security mechanisms for UPnP SOAP actions, such as authenticity, privacy and replay preventions. In short, the level 0 has no mechanisms for privacy or replay prevention. On the other hand, level 1 includes UPnP SOAP privacy and authenticity and Level 2 also adds trust relationships by using digital certificates.

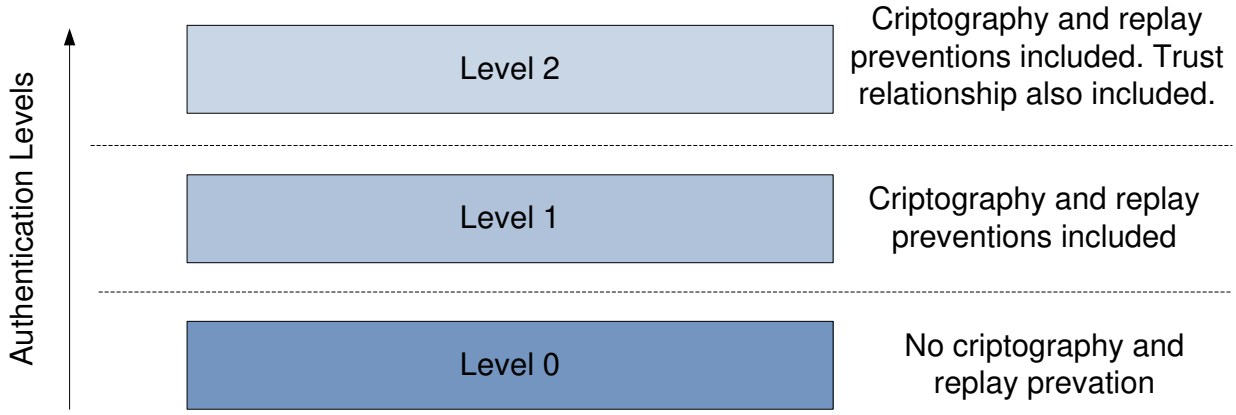


Figure 2: UPnP-UP authentication levels.

2.2.1. Authentication Level 0

The first authentication level has no mechanisms for privacy, confidentiality and replay preventions of the UPnP SOAP messages. The authentication process is simple and it is illustrated in Figure 3.

The authentication process is as follows: the control point authenticates the user by invoking the UPServer Auth action, sending the signature of the user profile and a Boolean value by specifying if the user wants to share his/her profile with the network (Step 1). That is, if True, any UPnP device can get the whole user profile by just invoking UPServer´s getProfile action. False otherwise. UPServer returns a user identification called UUID (Step 2), which will be used to open sessions with other UPnP appliances (Steps 3 and 4) by specifying the authenticated UPServer (UDN) and the users preferences (Profile_Preferences) based on the current device context.

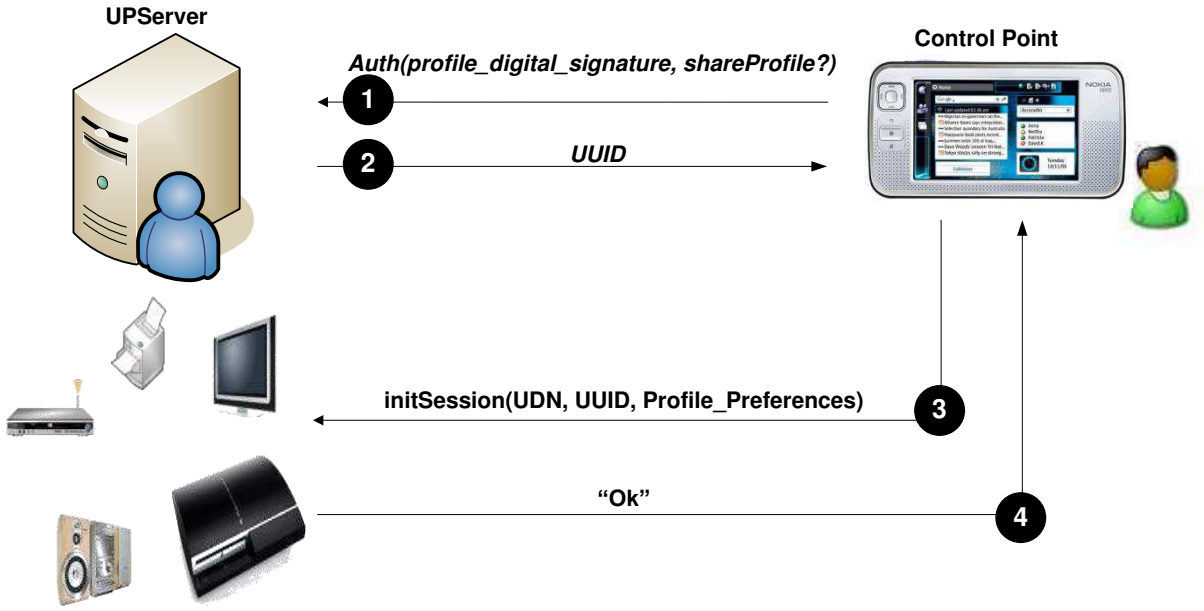


Figure 3: UPnP-UP User Authentication - Level 0

2.2.2. Authentication Levels 1 and 2

The second and third user authentication levels improve security and add cryptography mechanisms to ensure privacy, confidentiality and trust. In addition, replay prevention is also treated.

As illustrated in Figure 4, the authentication process is divided into 4 steps. The control point authenticates the user by invoking UPServer´s Auth action, sending the signed user profile, a symmetric key for encryption and a Boolean value that specifies if the user wishes to share his/her profile (Step 1). The UPServer returns a random value, called Nonce, encrypted by using the suggested symmetric key (Step 1´). The control point returns the user id (embedded into the profile) and the Nonce value, also encrypted using the symmetric key (Step 2). This process ensures that the Auth action is authenticity and it is not from a replay attack. Finally, the UPServer returns the user identification (UUID) to be used in the network.

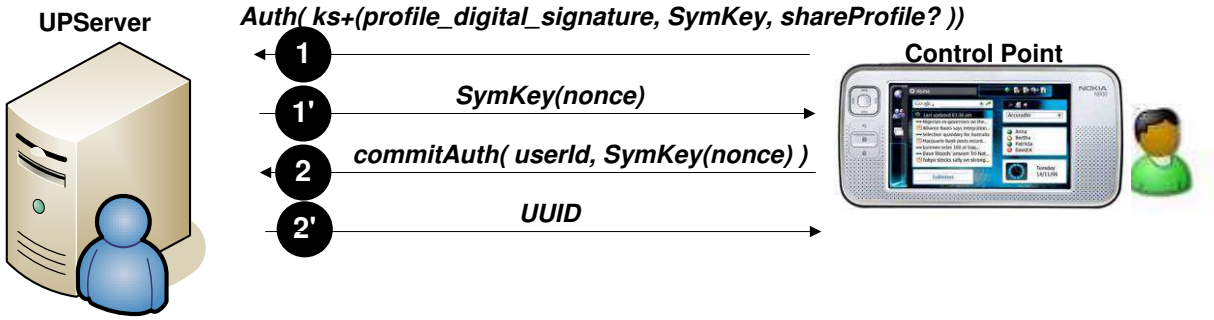


Figure 4: User Authentication - Level 1 and 2

After authenticating the user, the control point can open sessions with other UPnP appliances, as it is done with level 0. However, to ensure the authenticity of the user, the UPServer device must guarantee the authenticity of the user. The Figure 5 illustrates the process for opening a session between control points and UPnP appliances.

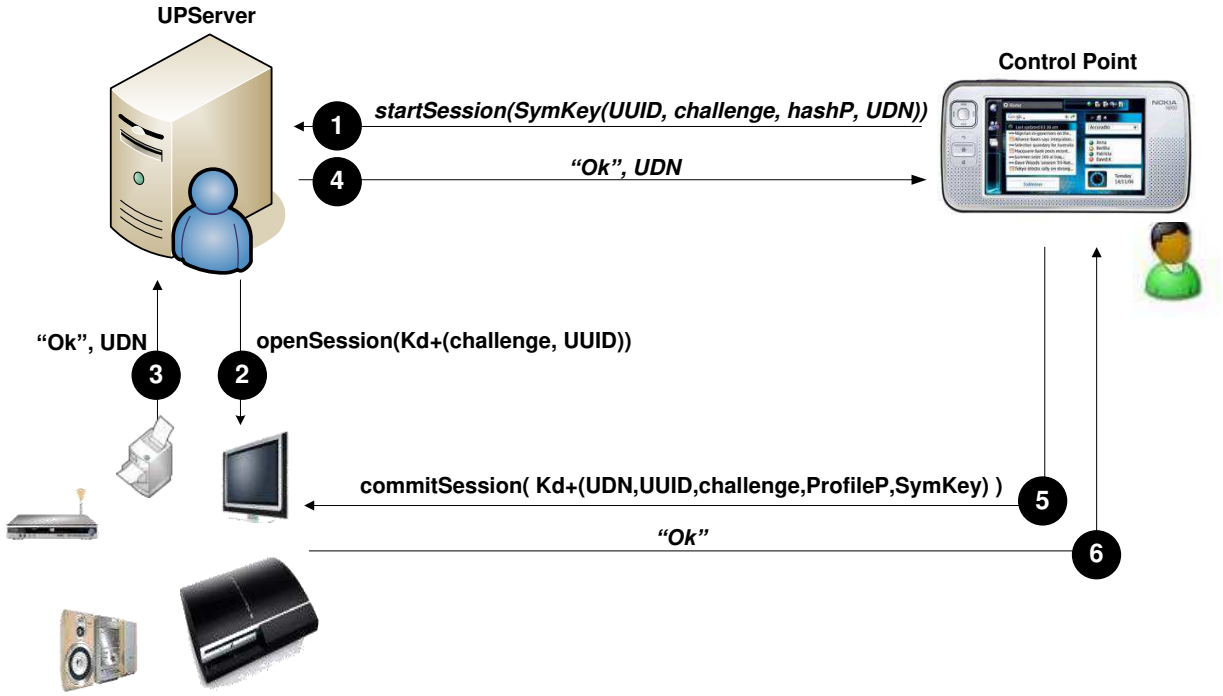


Figure 5: Session opening between Control Points and UPnP Appliances in authentication levels 1 and 2.

Consider in this document *Kd+* and *Kd-* as the public key and the private key of a given device *d*. To open a session with a device when running authentication levels 1 and 2, the control point must invoke the *startSession* action (Step 1) of the UPServer by sending the following parameters, encrypted with the symmetric key proposed in the *Auth* action: the user UUID, a random value called *challenge*, the last valid hash value of a given hash chain [2] and the target device´s UDN. For more information about their semantics, see *UPServer Device Template* [3]. The UPServer, once receiving the request, invokes the *openSession* action of the target device (Step 2), sending the proposed *challenge* value and the user UUID, both encrypted with the device´s *Kd+*. The target device can accept or deny the request, returning its confirmation (Step 3). Finally, the UPServer returns the confirmation and the control point requests the *commitSession* action (Step 5), encrypting the following parameters: UPServer´s UDN, the user UUID, the proposed *challenge* value, the user profile based on the context of the target device and a symmetric key value, to encrypt the next requests to the device. Finally, the target device returns a confirmation of the request (Step 6).

In spite of encrypting data and providing a suitable mechanism for confidentiality, the previously discussed scenario does not address trust relationships among control points and UPServers. That is, it is impossible to decide if the UPServer device is managed by the network administrator or it is a UPnP device offering authentication and authorization services, only. The relationship process proposed herein is based on digital certificates and is used in the authentication level 2. A

UPServer device offers its digital certificate, issued from the *upnp-up.org* certification authority, in two distinct ways and times. It is illustrated in Figure 6 how a control point and UPnP devices can trust in the available UPServer devices. First, before invoking *Auth*, a control point can invoke *getSupportedKeys*, receiving, among other information, the device´s digital certificate. Second, when dealing with session openings, the UPServer can, into the SOAP header, its digital certificate, granting that it is really a trusted entity that is trying to open a session on behalf of the user.

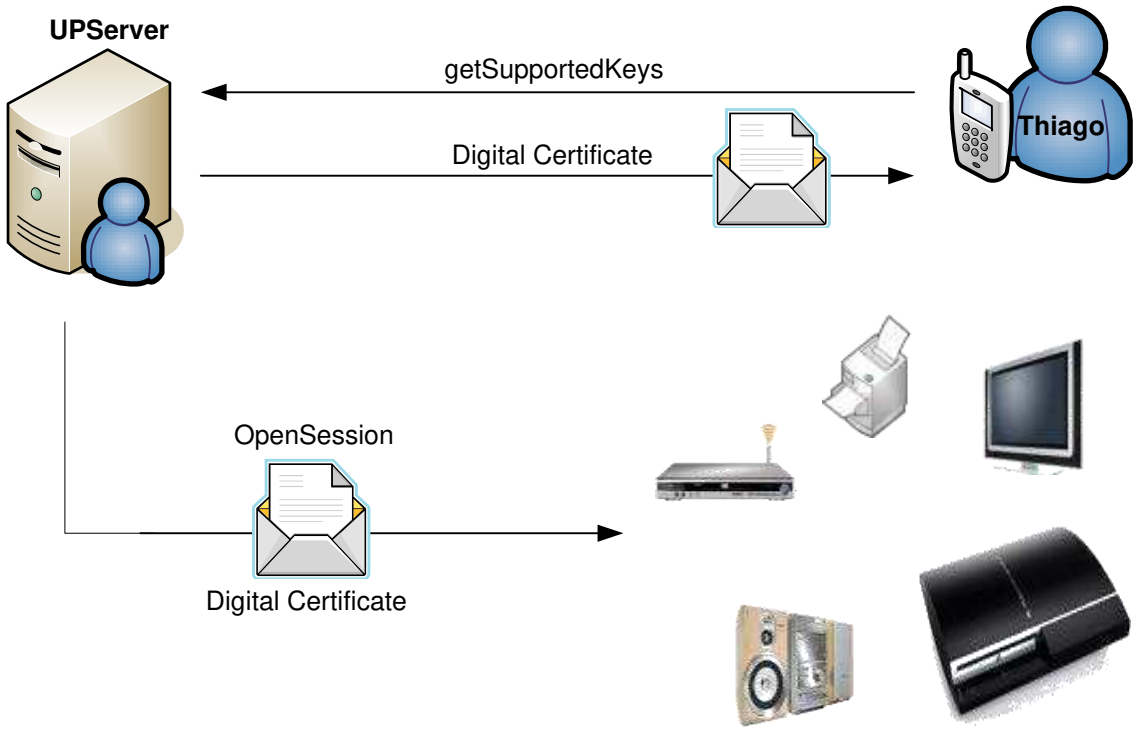


Figure 6: The use of certificates in order to UPnP devices to trust in UPServer devices.

2.3. User Authorization and Access Control

The process of defining and distributing access control policies is performed as illustrated in Figure 7. Access control policies are stored in the UPServer and defined by the network manager using a control point application that provides a graphical interface. After defining such policies, the UPnP-UP specification offers two modes in order to provide these policies to UPnP devices. First, a UPServer sends the access control policies to each UPnP device (push-based), who has the rights to define and send the policies to a given UPnP device. Second, each UPnP device acquires (based on the attributes of the device) the list of access control policies defined and stored in the UPServer (pull-based), keeping them locally, if possible.

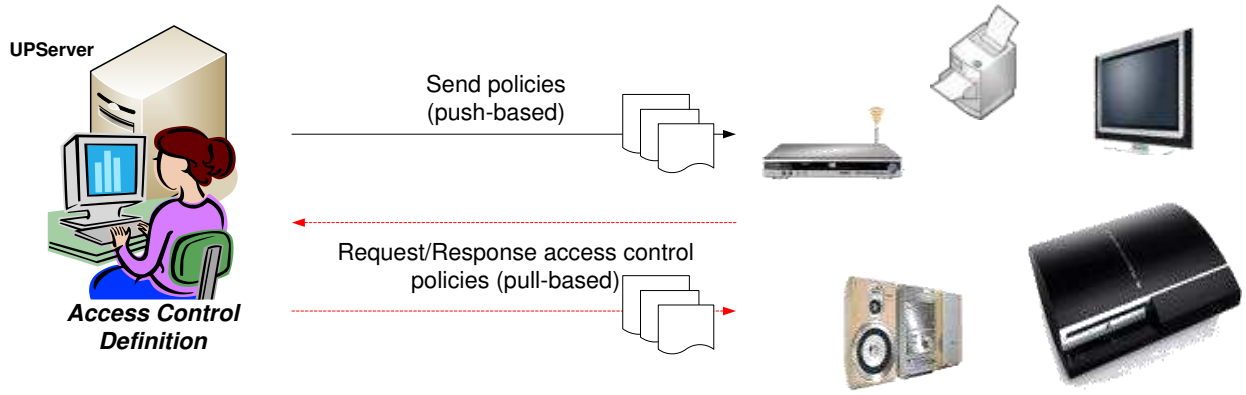


Figura 7: Definition and distribution of access control policies in UPnP-UP Spec.

This approach decouples the responsibilities of the interpretations of the access control policies between UPnP devices and UPServer device, which allows each UPnP device to grant access to a user for a specific UPnP service and action. In addition, there are devices that have limited storage and processing capabilities, enabling UPServer to execute the policies and to return the final access decision.

Apêndice B

DCP do Servidor de Perfis UPnP-UP

UPServer:1 *Device Template Version 1.01*

For UPnP™ Version *1.0*
Date: February 26, 2010

The UPnP User Profile Server device (UPServer) has been made available to UPnP Members as a proposal user profile server for a new generation of electronic devices for pervasive computing environment. This device follows the UPnP-UP Architecture proposal that has not been adopted as a Standardized DCP yet by the Steering Committee of the UPnP™ Forum. UPnP™ Forum Members have rights and licenses defined by Section 3 of the UPnP™ Forum Membership Agreement to use and reproduce the proposed architecture in UPnP™ Compliant Devices. All such use is subject to all of the provisions of the UPnP™ Forum Membership Agreement.

THE UPNP FORUM TAKES NO POSITION AS TO WHETHER ANY INTELLECTUAL PROPERTY RIGHTS EXIST IN THE PROPOSED DEVICE TEMPLATES, IMPLEMENTATIONS OR IN ANY ASSOCIATED TEST SUITES. THE *PROPOSED ARCHITECTURE, IMPLEMENTATIONS AND ANY ASSOCIATED TEST SUITES*ARE PROVIDED "AS IS" AND "WITH ALL FAULTS". THE UPNP FORUM MAKES NO WARRANTIES, EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE UPnP-UP ARCHITECTURE*, IMPLEMENTATIONS AND ASSOCIATED TEST SUITES* INCLUDING BUT NOT LIMITED TO ALL IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE, OF REASONABLE CARE OR WORKMANLIKE EFFORT, OR RESULTS OR OF LACK OF NEGLIGENCE.

Authors	Company
Thiago Sales	Embedded Systems and Pervasive Computing Laboratory
Leandro Sales	Federal University of Alagoas
Hyggo Almeida	Embedded Systems and Pervasive Computing Laboratory
Angelo Perkusich	Embedded Systems and Pervasive Computing Laboratory

1. Overview and Scope

1.1. Introduction

This device specification is compliant with the UPnP™ Device Architecture version 1.0. It defines a device type referred to herein as UPServer.

The UPServer specification defines a general-purpose device that can be used to instantiate user authentication and access control in UPnP network. It is based on the UPnP-UP Architecture Framework (described in another document). The UPServer exposes authentication services in order to guarantee users authenticity. In addition, authorization services provide the necessary requirements to enable access control for a new generation of UPnP services. As such, the UPServer can handle user profiles to provide them to UPnP appliances in order to allow customized services.

A full-featured UPServer device provides clients with the following capabilities:

- User authentication.
- Access control policies to UPnP appliances.
- A set of user profiles for customized services.
- Event generation for entry and exit of users to/from the UPnP network.

1.2. Notation

- In this document, features are described as Required, Recommended, or Optional as follows:
  The keywords "MUST," "MUST NOT," "REQUIRED," "SHALL," "SHALL NOT," "SHOULD," "SHOULD NOT," "RECOMMENDED," "MAY," and "OPTIONAL" in this specification are to be interpreted as described in [RFC 2119].

  In addition, the following keywords are used in this specification:
  PROHIBITED – The definition or behavior is prohibited by this specification. Opposite of REQUIRED.

  CONDITIONALLY REQUIRED – The definition or behavior depends on a condition. If the specified condition is met, then the definition or behavior is REQUIRED, otherwise it is PROHIBITED.

  CONDITIONALLY OPTIONAL – The definition or behavior depends on a condition. If the specified condition is met, then the definition or behavior is OPTIONAL, otherwise it is PROHIBITED.

  These keywords are thus capitalized when used to unambiguously specify requirements over protocol and application features and behavior that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

- Strings that are to be taken literally are enclosed in "double quotes".
- Words that are emphasized are printed in *italic*.
- Keywords that are defined by the UPnP Device Architecture specification are printed using the **arch** character style [DEVICE].

- A double colon delimiter, "::", signifies a hierarchical parent-child (parent::child) relationship between the two objects separated by the double colon. This delimiter is used in multiple contexts, for example: Service::Action(), Action()::Argument, parentProperty::childProperty.

1.2.1. Data Types

This specification uses data type definitions from two different sources. The UPnP Device Architecture defined data types are used to define state variable and action argument data types [DEVICE]. The XML Schema namespace is used to define property data types [XML SCHEMA-2].
For UPnP Device Architecture defined **boolean** data types, it is strongly RECOMMENDED to use the value "**0**" for false, and the value "**1**" for true. However, when used as input arguments, the values "**false**", "**no**", "**true**", "**yes**" may also be encountered and MUST be accepted. Nevertheless, it is strongly RECOMMENDED that all **boolean** state variables and output arguments be represented as "**0**" and "**1**".
For XML Schema defined Boolean data types, it is strongly RECOMMENDED to use the value "*0*" for false, and the value "*1*" for true. However, when used as input properties, the values "*false*", "*true*" may also be encountered and MUST be accepted. Nevertheless, it is strongly RECOMMENDED that all Boolean properties be represented as "*0*" and "*1*".

2. Device Definitions

2.1. Device Type

The following device type identifies a device that is compliant with this template:

urn:**schemas-upnp-org:device**:*UPServer:1*

2.2. Device Model

Products that expose devices of the type **urn:schemas-upnp-org:device:***UPServer:1* must implement minimum version numbers of all required embedded devices and services specified in the table below.

Table 2-1: Device Requirements

Device Type	Root	R/O[1]	Service Type	R/O	Service ID[2]
UPServer:1	*Root*	*R*	*UPAuthentication:1*	*R*	*UPAuthentication*
			UPAuthorization:1	*R*	*UPAuthorization*
			UPProfile:1	*R*	*UPProfile*
			Non-standard services embedded by a UPnP vendor go here.	*O*	*TBD*
Standard or Non-standard devices embedded by a UPnP vendor go here.	*Embedded*	*X*	*TBD*	*TBD*	*TBD*

1 *R* = REQUIRED, *O* = OPTIONAL, *X* = Non-standard.
2 Prefixed by urn:**upnp-org**:**serviceId**:

2.2.1. Description of Device Requirements

Any instance of a UPServer MUST have a UPAuthentication service, a UPAuthorization service and a UPProfile service. For a given instance (UPServer), there MUST only be one instance of these services. The semantics of additional standard UP (User Profile) services are not defined. Other standard services, such as UPnP QoS, MAY be added with semantics defined by the relevant specifications.

The UPAuthentication service allows control points to initiate an authentication session for a user in the UPServer device as well as open and a manage user sessions with UPnP appliances. The UPAuthorization service is used to provide access control policies against requests from control point on behalf of a user. Additionally, the UPAuthorization service is capable of grant access to allowed users when UPnP appliances have no processor and storage capacity. The UPProfile service is responsible for providing user profiles information to UPnP appliances and the management of user profiles updates.

2.2.2. Relationships Between Services

The *UPAuthentication::Auth()* action provides the trigger point for user authentication. If the security level defined by the UPnP-UP Architecture is 1 or 2, the *UPAuthentication::commitAuth()* action MUST be invoked in order to establish the authentication and avoid replay attacks.

The UPAuthentication::Ping() action MUST be invoked in a given time interval after a successful user authentication establishment provided by *UPAuthentication::Auth()*, allowing unexpected user exit.

3. Services and Actions

The UPServer exposes authentication-related actions, authorization-related actions and use profile related actions. It is described in this section the set of actions provided by the UPServer device.

3.1. Cryptography Notation for Selected Actions

To help explain the cryptographic aspects of some actions, a compact notation is used. The letters S, C and D are used to indicate the UPServer, a Control Point and a UPnP device, respectively. The public key of a party *E* is indicated by P_E^+, the private key by P_E^-. Thus P_D^+ is the public key of the device, while P_D^- is its private key. Session encryption keys (symmetric keys) are represented by K subscripted with both parties; thus K_{CD} would be a session key used between the control point and the device.

Encryption is represented by [], while signing is represented by {}. These operations are prefixed by the key used. So, for example, $P_D^+[P_C^-\{m\}]$ represents a message m, signed with the private key of control point C, and encrypted with the public key of device D.

3.2. Authentication Service and Actions

It is described in the following table the actions exposed by a device that implements the UPAuthentication service. Immediately following this table is detailed information about these actions, including short descriptions of the actions, the effects of the actions on state variables, and error codes defined by the actions.

Table 3: Actions

Name	Req. or Opt. [1]
Auth	*R*
commitAuth	*R*
startSession	*R*
logout	*R*
getSupportedKeys	*R*
ping	*R*
authoZ	*R*
updateProfile	*O*
getProfile	*R*
getPreferencesProfile	*O*
Non-standard actions implemented by an UPnP vendor go here.	X

[1] R = Required, O = Optional, X = Non-standard.

3.2.1. Auth

The Auth action is responsible for initializing an authentication session between a user (represented by his/her control point) and the UPServer.

3.2.1.1. Arguments

Table 3-1: Arguments for Auth

Argument	Direction	relatedStateVariable
UserProfile	*IN*	*on_user_enter*
Share	*IN*	
SymmetricKey	*IN*	
UUID	*OUT*	
NONCE	*OUT*	

All IN arguments MUST be encrypted as P_S^+[arguments].

The UserProfile argument is an XML-Signature based document generated by the upnp-up.org certification authority. That is $Ps^+[P_U^-\{profile\}]$. If the security level is "0", then the UUID output is used. Otherwise, NONCE output is used.

If Share is "1", then assumes that the user wants to automatically share his/her profile. Otherwise, "0" defines that the UPServer cannot provide the current user profile information to UPnP appliances.

The SymmetricKey argument specifies the K_{CS} between the control point and the UPServer. It is an XML structured document, such as:

```
<Keys>
   <Confidentiality>
```

```
      <Sym_Key>
            <Algorithm>AES</Algorith>
            <Value>AQABx931…</Exponent>
      </Sym_Key>
   </Confidentiality>
</Keys>
```

3.2.1.2. Effect on State (if any)

None.

3.2.1.3. Errors

Table 3-2: Error Codes for *Auth*

errorCode	errorDescription	Description
501	Action Failed	See UPnP Device Architecture section on Control
701	Signature Failure	The user profile XML signature is not valid
702	Symmetric Key Failure	The proposed symmetric key algorithm is not supported.
800-899	TBD	(Specified by a UPnP Vendor)

3.2.2. CommitAuth

This action is used when Security Level (defined in the UPnP-UP Architecture) is "1" or "2". It guarantee that the Auth action was not replayed.

3.2.2.1. Arguments

Table 3-3: Agruments for *CommitAuth*

Argument	Direction	relatedStateVariable
Username	*IN*	
NONCE	*IN*	
UUID	*OUT*	

The Username argument is the unique upnp-up users username. It cannot be encrypted in order to allow the UPServer to determine who has invoked Auth.

The NONCE argument guarantees that the control point is online, avoiding replay attacks (Auth action). It MUST be encrypted as P_S^+[nonce].

3.2.2.2. Effect on State

None.

3.2.2.3. Errors

Table 3-4: Error Codes for *commitAuth*

errorCode	errorDescription	Description
501	Action Failed	See the UPnP Device Architecture section on Control
701	Invalid NONCE	Received NONCE was not valid.
800-899	TBD	(Specified by a UPnP Vendor)

3.2.3. getSupportedKeys

This action is used to allow control points discover the algorithms and public key values supported key the UPServer device instance.

3.2.3.1. Arguments

Table 4-5: Agruments for *getSupportedKeys*

Argument	Direction	relatedStateVariable
Supported	*OUT*	

The Supported parameter is an escaped XML string giving public keys and algorithms supported by this device, in the format below. The list of all currently defined protocols IDs is given below.

```
<Supported>
      <HashAlgorithms>
            <algorithm>SHA1</algorithm>
      </HashAlgorithms>
      <EncryptionAlgorithms>
            <algorithm>
                  <RSAKeyValue>
                        <Modulus>xA7SEU+e0y…</Modulus>
                        <Exponent>AQAB</Exponent>
                  </RSAKeyValue>
            </algorithm>
            <algorithm>AES-128-CBC2</algorithm>
      </EncryptionAlgorithms>
      <SigningAlgorithms>
            <algorithm>RSA ...</algorithm>
            <algorithm>SHA1-HMAC</algorithm>
      </SigningAlgorithms>
</Supported>
```

3.2.3.2. Effect on State

None.

3.2.3.3. Errors

Table 3-6: Error Codes for *getSupportedkeys*

errorCode	errorDescription	Description
501	Action Failed	See the UPnP Device Architecture section on Control
800-899	TBD	(Specified by a UPnP Vendor)

3.2.4. startSession

This action is used to allow control points to initiate a session with a given UPnP appliance. It aims to guarantee the user authenticity to the target UPnP device.

3.2.4.1. Arguments

Table 3-7: Agruments for *startSession*

Argument	Direction	relatedStateVariable
UUID	*IN*	
Challenge	*IN*	
HashP	*IN*	
UDN	*IN*	
Confirmation	*OUT*	

All IN arguments MUST be encrypted as P_S^+[arguments]. Confirmation returns "1" for success and "0", otherwise.

The challenge parameter will be checked later by the target device with identification equals to UDN. The HashP is the lastest valid hash value of the control´s point hash chain.

3.2.4.2. Effect on State

None.

3.2.4.3. Errors

Table 3-8: Error Codes for *startSession*

errorCode	errorDescription	Description
501	Action Failed	See the UPnP Device Architecture section on Control
701	Invalid Hash Position	The lastest hash value of the hash chain is not valid.
702	Device UDN not found	The device with UDN was not found in the network.
800-899	TBD	(Specified by a UPnP Vendor)

3.2.5. logout

This action is used to allow control points to finalize a session with the UPServer instance.

3.2.5.1. Arguments

Table 3-1: Agruments for *logout*

Argument	Direction	relatedStateVariable
UUID	*IN*	
HashP	*IN*	
Confirmation	*OUT*	*on_user_exit*

HashP argument MUST be encrypted as P_S^+[hashP]. Confirmation returns "1" for success and "0", otherwise.

The UUID parameter announces the user that wants to finish his/her session. The HashP is the lastest valid hash value of the control´s point hash chain, which is used if security level is "1" or "2".

This action MUST fire an event announcing that the user has finished his/her session. All UPnP appliances SHOULD remove user-related information, if needed.

3.2.5.2. Effect on State

None.

3.2.5.3. Errors

Table 3-2: Error Codes for *logout*

errorCode	errorDescription	Description
501	Action Failed	See the UPnP Device Architecture section on Control
701	Invalid Hash Position	The lastest hash value of the hash chain is not valid.
800-899	TBD	(Specified by a UPnP Vendor)

3.2.6. ping

This action is used to allow control points announce that the user is still connected in the network. It avoids unexpected user disconnection due to situation such as battery lifetime, leaving unnecessary user information in the UPnP appliances.

3.2.6.1. Arguments

Table 3-11: Agruments for *ping*

Argument	Direction	relatedStateVariable
UUID	*IN*	
SequenceNumber	*IN*	
Confirmation	*OUT*	

SequenceNumber argument MUST be encrypted as P_S^+[SequenceNumber] and MUST be carried in the HEADER SOAP field.. Confirmation returns “1” for success and “0”, otherwise.

The UUID parameter announces the user that wants to finish his/her session. The HashP is the lastest valid hash value of the control´s point hash chain, which is used if security level is “1” or “2”.

3.2.6.2. Effect on State

None.

3.2.6.3. Errors

Table 3-12: Error Codes for *ping*

errorCode	errorDescription	Description
501	Action Failed	See the UPnP Device Architecture section on Control
701	Invalid Sequence Number	The sequence number is not valid.
800-899	TBD	(Specified by a UPnP Vendor)

3.3. Authorization Service and Actions

3.3.1. authoZ

This action is used to allow UPnP devices to retrieve the access control policies.

3.3.1.1. Arguments

Table 3-11: Agruments for *authoZ*

Argument	Direction	relatedStateVariable
DeviceUDN	*IN*	
DeviceType	*IN*	
ServiceName	*IN*	
ActionName	*IN*	
AccessControlPolicies	*OUT*	

DeviceUDN is the Device´s UDN from which the access control policies will be returned. DeviceType is used to specify the UPnP device type. For instance, ***urn:schemas-upnp-org:device:MediaServer:3*** returns access control policies regarding a Media Server device (version 3). The ServiceName specifies those policies that controls the access to this service. ActionName identifies those policies that controls the access to a specific action.

It is important to point out that the parameters value can be combined to produce the output (access control policies). If the UDN and action name are specified, only access control policies that control the access to this action in this device will be returned.

3.3.1.2. Effect on State

None.

3.3.1.3. Errors

Table 3-12: Error Codes for *authoZ*

errorCode	errorDescription	Description
501	Action Failed	See the UPnP Device Architecture section on Control
701	Invalid Device Type	The sequence number is not valid.
800-899	TBD	(Specified by a UPnP Vendor)

XML Device Description

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <URLBase>base URL for all relative URLs</URLBase>
  <device>
    <deviceType>urn:schemas-upnp-org:device:UPServer:1</deviceType>
    <friendlyName>short user-friendly title</friendlyName>
    <manufacturer>manufacturer name</manufacturer>
    <manufacturerURL>URL to manufacturer site</manufacturerURL>
    <modelDescription>long user-friendly title</modelDescription>
    <modelName>model name</modelName>
    <modelNumber>model number</modelNumber>
    <modelURL>URL to model site</modelURL>
    <serialNumber>manufacturer's serial number</serialNumber>
    <UDN>uuid:UUID</UDN>
    <UPC>Universal Product Code</UPC>
    <iconList>
      <icon>
        <mimetype>image/format</mimetype>
        <width>horizontal pixels</width>
        <height>vertical pixels</height>
        <depth>color depth</depth>
        <url>URL to icon</url>
      </icon>
      XML to declare other icons, if any, go here
    </iconList>
    <serviceList>
      <service>
        <serviceType>urn:schemas-upnp-
org:service:UPAuthentication:1</serviceType>
        <serviceId>urn:upnp-org:serviceId:UPAuthentication</serviceId>
        <SCPDURL>URL to service description</SCPDURL>
        <controlURL>URL for control</controlURL>
        <eventSubURL>URL for eventing</eventSubURL>
      </service>
      <service>
        <serviceType>urn:schemas-upnp-
org:service:UPAuthorization:1</serviceType>
        <serviceId>urn:upnp-org:serviceId:UPAuthorization</serviceId>
        <SCPDURL>URL to service description</SCPDURL>
        <controlURL>URL for control</controlURL>
        <eventSubURL>URL for eventing</eventSubURL>
      </service>
      <service>
        <serviceType>urn:schemas-upnp-org:service:UPProfile:1</serviceType>
        <serviceId>urn:upnp-org:serviceId:UPProfile</serviceId>
        <SCPDURL>URL to service description</SCPDURL>
        <controlURL>URL for control</controlURL>
        <eventSubURL>URL for eventing</eventSubURL>
```

```
      </service>

      Declarations for other services added by UPnP vendor (if any) go here
    </serviceList>
    <deviceList>
      Description of embedded devices added by UPnP vendor (if any) go here
    </deviceList>
    <presentationURL>URL for presentation</presentationURL>
  </device>
</root>
```