4
Serviço de privacidade de contexto

Neste capítulo, descrevemos as características e o funcionamento do serviço de privacidade de contexto - CoPS - que se adequa ao modelo e implementa os requisitos de privacidade discutidos no Capítulo 3. O CoPS oferece um gerenciamento de regras flexível através do qual o usuário define e gerencia a sua política de privacidade gradativa e interativamente, no decorrer do uso de aplicações sensíveis ao contexto.

Nas seções seguintes são descritas as características gerais da arquitetura e o padrão de interação entre as entidades e os serviços que fazem parte do sistema. Em seguida, são mostradas a estrutura geral das regras de privacidade e o algoritmo de especificidade que processa as requisições de acesso ao contexto de acordo com a política de privacidade dos usuários. Além disso, são discutidos também os principais aspectos envolvidos na implementação do CoPS, tais como a sua estrutura, o algoritmo de especificidade, mecanismos de *caching*, dentre outros.

4.1
Arquitetura do CoPS

A arquitetura do CoPS segue uma abordagem centralizada e engloba um conjunto de componentes que implementam as funcionalidades providas pelo serviço. Tais componentes, ilustrados na Figura 4.2, oferecem um controle flexível e de granularidade fina no processamento da política de privacidade dos usuários.

A abordagem centralizada (do controle de privacidade e da infra-estrutura de provisão de contexto) oferece ao desenvolvedor a flexibilidade de decidir se e como o serviço de privacidade deve ser incorporado à infra-estrutura de provisão de contexto. Ao invés de incorporar os mecanismos de controle de privacidade no serviço de contexto, nós concluímos que seria mais adequado tratar o controle de privacidade como um serviço à parte para tornar a solução mais genérica e independente da infra-estrutura de provisão de informação de contexto. No entanto, ao contrário da abordagem descentralizada, na qual a inferência do contexto e o controle de privacidade são feitas no

próprio dispositivo do usuário, a abordagem centralizada pode fazer com que o usuário se sinta mais vulnerável, dado que as suas informações são mantidas sob o domínio de terceiros, mesmo que supostamente confiáveis (e.g., Universidade, Bancos, Órgãos do governo, etc.). Em outras palavras, os usuários perdem o controle sobre a utilização e divulgação de suas informações. Além disso, a arquitetura centralizada representa um ponto único de falha e é menos escalável do que uma arquitetura distribuída.

Entretanto, a nossa proposta, conforme definido pelo modelo conceitual, está fundamentada na hipótese de que os serviços de provisão de contexto e de controle de privacidade estejam sendo administrados por uma entidade gerenciadora supostamente confiável em um ambiente organizacional. Nesse contexto, a relação de confiança entre os usuários e a organização é equivalente àquela mantida com outros serviços que seguem uma abordagem centralizada, como, por exemplo, serviços de correio eletrônico, serviços prestados por sítios web do governo, serviços bancários, etc. No entanto, vale ressaltar que utilizamos uma arquitetura centralizada mais pela factibilidade de implementação do que pela sua eficiência para tratar das questões de privacidade. Por ter adotado essa abordagem centralizada, não será necessário lidar com alguns problemas das arquiteturas distribuídas em redes *ad hoc* que implementam a provisão de contexto e o controle de privacidade de forma integrada, como, por exemplo, lidar com as limitações de recursos computacionais, descoberta e autenticação mútua dos dispositivos, disponibilidade e conectividade intermitente da rede, dentre outros.

Conforme discutido no Capítulo 1, existem brechas para ataques à privacidade em vários níveis das arquiteturas de rede envolvendo as mais diversas aplicações e protocolos de comunicação. No entanto, a arquitetura proposta trata somente das questões de privacidade relacionadas à política de divulgação da informação de contexto (i.e., localização) fornecida pela infra-estrutura de provisão de contexto, e não trata das possíveis violações de privacidade exploradas a partir das vulnerabilidades de outros serviços tais como: no serviço de coleta da informação de contexto, em algum protocolo de rede, em algum sistema intermediário na comunicação, etc.

A seguir, ilustramos o padrão de interação das entidades da arquitetura e, em seguida, descrevemos algumas questões gerais do projeto de alguns dos seus componentes como, por exemplo, o componente de gerenciamento de grupos e usuários (DUMAC), o componente de gerenciamento das regras de privacidade e algoritmo de avaliação de acesso (*Privacy Policy Specificity*), o mecanismo de notificações e geração de relatórios de acesso (*Report Manager*), dentre outros.

4.1.1 Interação entre as entidades do serviço

Em linhas gerais, o CoPS é formado por um servidor e duas APIs clientes. O servidor gerencia o acesso às informações de contexto e as APIs ocultam do desenvolvedor alguns detalhes envolvidos na comunicação com o CoPS. A API *Context Access Authorization (CAA)* é utilizada pelo serviço de contexto para enviar ao servidor CoPS requisições de autorização de acesso. A API *User and Policy Management (UPM)* é utilizada pelas aplicações clientes do Subject (e/ou PolicyMaker) e Requester para gerenciar grupos, analisar logs, solicitar acesso ao contexto, dentre outros.

Na Figura 4.1, é mostrado como os componentes do CoPS interagem entre si e com o serviço de provisão de contexto. No caso específico da MoCA, o LIS faria o papel do *Context Service*.

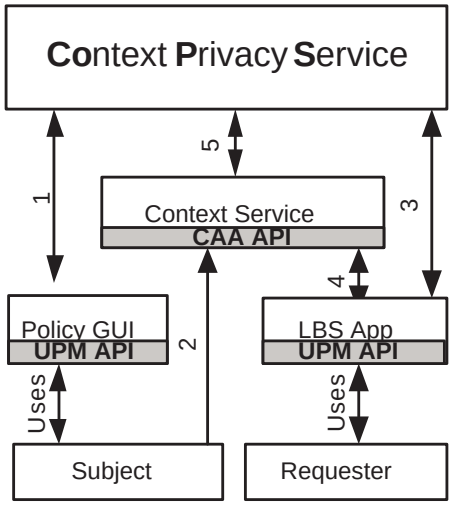

Figura 4.1: Interação entre cliente e servidor

A interação entre as entidades do serviço é similar ao descrito no padrão de interação do serviço de privacidade na Seção 3.2. As principais diferenças consistem do uso das APIs CAA e UPM e a implementação da autenticação do usuário no próprio serviço de privacidade. Na interação, o Requester autentica-se junto ao CoPS para validar a sua identidade (3). Esta autenticação gera um *token* de sessão que é usado para criar um **U**ser **I**dentification **T**oken (UIT) para futuras requisições. O UIT é o *hash* do *token* de sessão que é enviado pela aplicação LBS junto com suas requisições para atestar que o usuário requisitante já foi autenticado. Tanto o *token* como a requisição são passados pelo serviço de contexto para o CoPS para requisitar a autorização de acesso à localização (5). A implementação da geração e distribuição dos *tokens* de sessão do usuário é discutida em mais detalhes na Seção 4.2. Se o Requester possuir os devidos direitos de acesso (de acordo com a política do Subject),

o CoPS responde ao serviço de contexto com uma mensagem "Grant", caso contrário, com uma mensagem "Deny" ou "Not Available".

4.1.2 Visão geral dos componentes da arquitetura

Uma visão geral dos componentes da arquitetura é ilustrada na Figura 4.2. O componente *Communication Level* provê interfaces para comunicação síncrona e assíncrona com transmissão/recepção de dados não encriptados ou encriptados via SSL (*Secure Socket Layer*). Esse componente implementa os servidores de comunicação que interagem com as APIs UPM e CAA usadas pelas aplicações para realizar determinadas tarefas, tais como autenticação, processamento da autorização de acesso à localização, etc. As requisições recebidas pelo *Communication Level* são repassadas para o componente *Controller*, que por sua vez, interpreta o tipo de requisição recebida e interage com os demais componentes da arquitetura para executar a ação solicitada. Para processar eficientemente requisições concorrentes, o *Controller* gerencia um *pool* de *threads*.

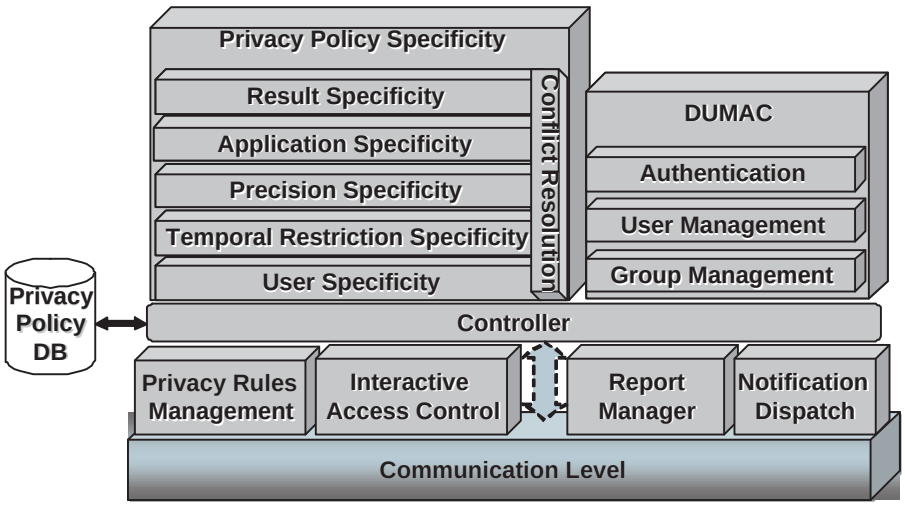

Figura 4.2: Arquitetura geral do CoPS

As requisições de acesso aos relatórios são repassadas pelo *Controler* para o componente *Report Manager*. Este componente gera relatórios com estatísticas de acesso representando-os hierarquicamente pelo ano, mês, semana e dia. Para cada nível da hierarquia, o *Report Manager* calcula as estatísticas das tentativas de acesso bem e mal sucedidas de indivíduos específicos ou grupos de usuários. Considerando que o Requester pode pertencer a vários grupos (e.g., Coworker, Friends), as estatísticas de acesso são contabilizadas de acordo com a regra de privacidade selecionada para avaliar a requisição do usuário em questão. Ou seja, se a regra selecionada está controlando o

acesso de um requisitante específico (e.g., requisições de João), este dado será contabilizado nas estatísticas de acesso desse requisitante, mas se a regra selecionada contém no campo Requester o nome de um grupo (e.g., Coworker), o *Report Manager* contabilizará o acesso para esse grupo.

Para otimizar o processamento de tais relatórios, o *Report Manager* implementa uma consolidação das estatísticas de acesso em uma hora pré-programada (e.g., de madrugada). Diariamente, o *Report Manager* consolida as estatísticas de acesso de todos os usuários e, em seguida, usa essas informações para consolidar/calcular as estatísticas de acesso da semana e, a partir dos dados estatísticos de cada semana, calcula as estatísticas de acesso do mês, e assim por diante.

Os componentes *Notification Dispatch* e *Interactive Access Control* são utilizados pelo componente *Privacy Policy Specificity* para enviar notificações e consultas de autorização de acesso ao Subject. Quando um Subject tiver escolhido uma política de controle de acesso tipo "Sob-Demanda", ou quando a ação associada à regra selecionada para avaliar uma requisição for "Ask Me", o componente *Interactive Access Control* interage com o Subject para consultá-lo sobre o resultado a ser aplicado às requisições recebidas do serviço de contexto. Quando consultado, o Subject deve informar o resultado (e.g., *Grant/Deny*) a ser aplicado à requisição e, além disso, na resposta, ele pode informar se a avaliação deferida deve ou não ser armazenada em sua política de privacidade para evitar futuras consultas desnecessárias.

Para interagir com o Subject e obter o resultado a ser aplicado às requisições recebidas, o componente *Interactive Access Control* configura um canal de troca de eventos (através da API publish/subscribe ECI da MoCA) com a API UPM usada pela aplicação cliente LBS do Subject. Através desse canal de eventos, esse componente envia uma requisição de consulta ao Subject e aguarda uma resposta por um determinado intervalo de tempo pré-configurado. Se for recebida uma resposta dentro do intervalo de tempo esperado, o CoPS avalia a requisição de acesso ao contexto de acordo com o resultado (e.g., *Grant, Deny* ou *Not Available*) informado na resposta do Subject. Se não for recebida uma resposta, o CoPS envia uma mensagem de "Not Available" para o serviço de contexto, que por sua vez, a repassa para o Requester.

O *DUMAC (Dynamic User Management and Access Control)* é utilizado para autenticar e administrar usuários e grupos. Antes de repassar um comando para os demais componentes, o *Controller* autentica o requisitante de uma dada tarefa a partir do UIT encapsulado em cada requisição. Dentre as operações implementadas pelo *DUMAC* para gerenciar usuários e grupos podemos

descrever: adicionar, remover, listar usuários e grupos, executar operações de *Join* e *Leave* de usuários para grupos, dentre outros.

O componente *Privacy Rules Management* é utilizado para processar as requisições de gerenciamento das regras ou alterações na configuração da política de privacidade de acordo com as necessidades específicas de cada usuário. Dentre as suas funcionalidades, esse componente é responsável por: identificar e gerenciar as regras de privacidade sensíveis ao contexto, implementar e gerenciar as políticas de privacidade hierárquicas, gerenciar a configuração do Modo Invisível, administrar regras de privacidade temporárias e gerenciar as operações básicas de administração de regras (e.g., adicionar, remover, ...).

4.1.3 Estrutura das regras de privacidade

A regra de privacidade no CoPS é composta por vários campos. Estes também estão presentes nas requisições de acesso ao contexto. Toda regra de privacidade está associada a uma política de acesso padrão (e.g., Reservado, Liberal ou Sob-Demanda). Tal associação é feita pelo Policy Maker e determinará o funcionamento do algoritmo básico de avaliação para cada requisição. Os campos das regras de privacidade e seus significados são descritos como segue.

- *Policy Maker*: Usuário que define/configura a regra de privacidade (pode ou não ser o próprio Subject);
- *Subject*: Usuário ou entidade cuja informação de contexto (e.g., localização) é controlada pela regra de privacidade;
- *Requester*: Usuário ou componente de software que requisita acesso à informação de contexto do Subject;
- *Context Variable*: Informação de contexto requisitada pelo Requester (e.g., localização do Subject);
- *Application*: Lista de nomes das aplicações que podem ser utilizadas pelo Requester para acessar a variável de contexto. O símbolo "coringa" '*' representa qualquer aplicação;
- *Precision*: Especifica a precisão, ou granularidade, do valor da variável de contexto a ser divulgada (e.g., para informação de localização, este atributo poderia ter valores Prédio, Andar, Sala, etc.);
- *Temporal Restriction*: Restrições de hora e data para divulgar a informação de contexto (e.g., dias da semana, das 9:00 às 14:00);

- *Freshness*: Especifica quão recente deve ser a informação de contexto a ser divulgada para um dado Requester (e.g., revelar somente a localização inferida há 15 minutos atrás, ou revelar a localização corrente);
- *Timestamp*: Registra o horário em que a regra de privacidade foi criada ou atualizada. Este campo é usado para resolver possíveis conflitos entre as regras com resultados contraditórios;
- *AccessPolicy*: Representa a política de acesso (Reservado, Liberal ou Sob-Demanda) com a qual a regra de privacidade está associada;
- *Policy Level*: Nível da hierarquia da regra de privacidade. O CoPS provê suporte às seguintes hierarquias: "Organization", "Individual" ou "Default" (com este grau de precedência);
- *Result*: Resultado a ser aplicado à requisição de acesso à informação de contexto. Os possíveis valores são: "Not Available", "Ask Me", "Grant" e "Deny";
- *Notify Me*: Tipo de notificação a ser enviada para o Subject quando uma requisição é avaliada pela regra. Por exemplo, "NoNotification", "E-Mail", "MSN" ou "SMS".

4.1.4 Definição de grupos

Conforme descrito na pesquisa realizada por (10), o conceito de grupos de usuários facilita a configuração da política de privacidade. Utilizando grupos, os Policy Makers podem adicionar uma única regra para tratar requisições de um grupo de requisitantes (e.g., *MyFamily*). Por isso, no CoPS, o *Subject* ou *Requester* de uma regra de privacidade podem ser usuários ou grupos.

Existem duas categorias gerais de grupos: grupos definidos pelo administrador e grupos definidos pelo usuário. Os primeiros podem ser estruturados hierarquicamente para refletir categorias de usuários em uma estrutura organizacional e, assim, definir as possíveis funções dos usuários na organização. Grupos no nível mais alto da hierarquia (categorias mais genéricas) contêm todos os grupos descendentes, por exemplo, o grupo "puc.employee" contém o grupo "puc.employee.prof", que por sua vez, contém o grupo "puc.employee.prof.cs". Os grupos definidos pelos usuários não refletem uma hierarquia por causa da dificuldade de avaliação e manutenção de membros dos mesmos.

Inicialmente, todos os usuários no CoPS pertencem ao grupo "Anonymous" (grupo criado pelo CoPS cujo gerenciamento é de responsabilidade do administrador). Isto facilita a especificação das regras de acesso para usuários desconhecidos. Além disso, este grupo também pode ser usado para garantir

o anonimato nas requisições, i.e., usuários podem enviar uma requisição como um usuário anônimo se ele deseja ocultar sua identidade. No entanto, esses usuários deverão estar sujeitos à política de acesso definida para esse nível de anonimidade, geralmente mais restritivo.

4.1.5 Algoritmo de especificidade da política de privacidade

A avaliação da política de privacidade é uma das principais tarefas realizadas pelo CoPS para oferecer um controle de acesso efetivo e de granularidade fina na avaliação das requisições. Para tanto, o *Controller* usa o componente *Privacy Policy Specificity* para avaliar as requisições de acesso à informação de contexto. Esse componente implementa o algoritmo de especificidade responsável por avaliar as requisições de acordo com as regras de privacidade relativas ao Subject. Primeiro, ele seleciona as regras da política de controle de acesso corrente escolhida pelo PolicyMaker, e, em seguida, compara a especificidade das regras correspondentes com o propósito de selecionar a regra mais específica dentre aquelas que "casam" com a requisição recebida. Durante este processo, mais de uma regra pode ser selecionada para avaliar uma dada requisição. Por exemplo, quando o Requester pertence a vários grupos mencionados no campo "Requester" de algumas regras (e.g., "Maria" pode pertencer aos grupos "Coworker" e "MyFriend" usados nas regras de João). Com base no conjunto de regras selecionado, o algoritmo de especificidade identifica e resolve possíveis conflitos a fim de escolher uma única regra que determinará o resultado final a ser aplicado ("Not Available", "Ask Me", "Grant" ou "Deny").

Em linhas gerais, o algoritmo de especificidade funciona da seguinte forma. A partir de um conjunto de regras selecionado previamente para avaliar uma requisição, o algoritmo identifica a regra mais específica desse conjunto comparando sucessivamente os campos das regras na seguinte ordem de prioridade: *Subject*, *Requester*, *Temporal Restriction*, *Precision*, *Application* e *Result*. Ao comparar as regras com relação a um determinado campo, somente aquelas com o valor mais específico neste campo são selecionadas para prosseguir a análise de especificidade dos demais campos, enquanto que as demais regras serão desconsideradas. Desta forma, mesmo se duas ou mais regras possuem diferentes especificidades relativas (i.e., elas diferem em dois ou mais campos), o algoritmo determina a regra mais específica analisando esses campos de acordo com suas prioridades. Para todos os campos, o coringa '*' representa o valor menos específico.

Para a especificidade dos campos "*Subject*" e "*Requester*", as regras de privacidade que contêm o nome de um usuário específico (e.g., "Alice") são

mais específicas do que as regras contendo um grupo definido pelo usuário (e.g., "MyFriend"), que por sua vez, são mais específicas do que as regras com grupos criados por administradores. A especificidade destes últimos segue a interpretação de uma hierarquia: grupos no nível mais baixo da hierarquia são mais específicos do que grupos no nível mais alto (e.g., "puc.employee.prof.cs" é mais específico do que o grupo "puc.employee.prof").

O mesmo critério de especificidade aplicado aos grupos hierárquicos definidos pelo administrador é utilizado também para o campo *Precision*. Na comparação das regras relacionadas à informação de localização, as regras mais específicas são aquelas onde o campo *Precision* contém o nível mais baixo na hierarquia dessa informação, por exemplo, a hierarquia "campus.predio.andar.sala" (nível 4) é mais específica do que "campus.predio.andar" (nível 3). Duas ou mais regras de privacidade podem estar no nível mais alto de especificidade com relação ao campo *Precision* se elas contêm o valor mais específico e estão no mesmo nível na hierarquia. Quando isto acontece, o próximo campo (de acordo com a ordem de prioridade de avaliação) dessas regras é comparado para identificar a regra mais específica. O desenvolvedor do serviço de contexto/localização tem que definir a sintaxe (e.g., campus.predio.andar.sala) da hierarquia de nomes que pode ser utilizada no referido campo para que o CoPS possa exportá-la para o Policy Maker durante a configuração da política de privacidade.

O campo *Temporal Restriction* representa o intervalo de tempo e data nos quais o Requester tem acesso permitido ou negado à informação de contexto, dependendo da abordagem da política de acesso utilizada (Liberal ou Reservado). Esse campo é muito útil quando os usuários querem restringir o acesso durante alguns períodos de tempo, como, por exemplo, no horário de almoço, no horário de trabalho, no fim de semana, etc. A especificidade para esse campo é processada pelo componente *Privacy Policy Specificity* em três fases: (1) seleciona as regras que combinam com o horário e data da requisição; (2) identifica a regra com o maior intervalo de tempo e verifica se os intervalos de tempo das demais regras selecionadas são subconjunto próprio[1] deste (e.g., a restrição temporal "Feb 5, 10:30am-2:00pm" é um subconjunto próprio da restrição "Feb 5, 10:00am-6:00pm"). As regras estarão no mesmo nível de especificidade em relação a esse campo se elas tiverem intervalos de tempo idênticos, ou se o intervalo de tempo de uma delas não for um subconjunto próprio da regra selecionada com o maior intervalo de tempo; (3) seleciona a regra com o menor intervalo de tempo, caso as regras selecionadas não estejam

[1]O conjunto S2 é um subconjunto próprio do conjunto S1 se todo elemento de S2 está em S1 e S1 possui algum elemento que não está em S2.

no mesmo nível de especificidade.

Com relação ao campo *Application*, a especificidade tem somente dois níveis possíveis: qualquer aplicação (representada por '*') e uma lista de aplicações. Finalmente, se todos os campos considerados anteriormente estão no mesmo nível de especificidade, o campo *Result* é utilizado para selecionar a regra mais específica para avaliar a requisição. Os possíveis valores para este campo são: "Not Available", "Ask Me" e "Grant" (ou "Deny"). O resultado "Not Available" tem uma maior precedência ou especificidade do que "Ask Me", que por sua vez, tem maior precedência do que os outros resultados. A razão é que "Not Available" implicitamente significa "Negar acesso" e "não deixar o Requester ter ciência disso", enquanto o "Ask Me" pode resultar em "Deny" ou "Grant", dependendo da intenção do usuário no momento da consulta. Um conflito é detectado quando existe mais de uma regra com o resultado "Not Available" ou "Ask Me", ou quando todas as regras têm um resultado "Grant" ou "Deny". Neste caso, a regra mais recente criada pelo Policy Maker é selecionada. Julgamos que seria necessário implementar uma escolha determinística para essas situações porque as regras conflitantes podem ter diferentes tipos de notificação.

Exemplo da avaliação da política de privacidade

Nesta seção, através de alguns cenários, mostramos um exemplo do funcionamento do algoritmo de especificidade para algumas possíveis regras de privacidade para um usuário hipotético chamado João, assumindo que a política de acesso *Reservado* tenha sido escolhida (ou seja, se nenhuma regra for selecionada para avaliar a requisição, o pedido de autorização de acesso será negado). As regras, mostradas na Tabela 4.1, determinam como, quando e para quem a localização de João pode ser revelada. Neste exemplo, nós também assumimos a existência de alguns grupos definidos pelo usuário e pelo administrador da organização (os grupos de João e da PUC são mostrados na Tabela 4.2) que são mencionados em algumas regras.

Regras	Subject	Requester	Temporal Restriction	Precision	Application	Result	Freshness	Policy Level	Notification
R1	Puc.Aluno	Puc.Adm	09:00 às 18:00	campus	Ap1	G	0	O	e-mail
R2	João	Puc.Aluno	09:00 às 18:00	*	*	G	5	U	ICQ
R3	João	Amigos	09:30 às 12:30	campus.predio	*	G	0	U	ICQ
R4	João	ColTrab	12:00 às 14:00	campus.predio	*	NA	0	U	Ausente
R5	João	Alice	09:00 às 13:30	campus.predio	*	G	10	U	MSN
R6	João	Alice	13:00 às 16:00	campus.predio. andar.sala	*	G	0	U	e-mail

Tabela 4.1: Regras de exemplo

Grupos Existentes		
Criador do grupo	**Nome do Grupo**	**Membros**
João	Amigos	Alice, Pedro
	ColTrab ⇒ Colegas de Trabalho	Alice, Maria, Pedro
Administrador	Puc.Aluno	João, Alice, Maria, Pedro
	Puc.Adm	Maria, Paulo

Tabela 4.2: Hipóteses de grupos do usuário e da organização

Como já discutido, a requisição sempre é avaliada pela regra mais específica da política de privacidade do Subject, e a comparação da especificidade das regras é implementada avaliando os seus campos na ordem: *Subject*, *Requester*, *Temporal Restriction*, *Precision*, *Application* e *Result*. Sendo assim, o algoritmo compara os valores das colunas (da esquerda para direita) da Tabela 4.1 com o objetivo de encontrar as regras que têm o valor mais específico no campo sendo analisado. As regras selecionadas são candidatas para as avaliações seguintes.

Cenário 1: Se Maria requisitar a localização de João às 13:00h usando a aplicação Ap1, R1 e R4 são selecionadas, pois Maria ∈ ColTrab e Maria ∈ Puc.Adm. Entretanto, a requisição seria permitida devido a R1, porque esta "casa" com os parâmetros da requisição (e.g., Horário em que a requisição foi recebida, aplicação utilizada pelo Requester) e pertence a um nível mais alto da hierarquia de regras do que R4. Regras da organização têm maior precedência do que regras do usuário. Nesse caso, R4 avaliaria a requisição se e somente se nenhuma regra no nível da organização fosse selecionada para avaliá-la.

Cenário 2: Para tratar uma requisição de Pedro recebida às 12:15, as regras R2, R3 e R4 seriam selecionadas. Porém, dentre essas, as regras R3 e R4 são escolhidas para a próxima fase da análise de especificidade, porque os grupos criados pelo usuário, mencionados nessas regras, são mais específicos do que o grupo criado pelo administrador, utilizado em R2. Finalmente, a requisição será avaliada por R4 porque, apesar dos campos *Requester*, *Temporal Restriction*, *Precision* e *Application* dessas duas regras terem o mesmo nível de especificidade, o valor dos campos *Result* são diferentes, e "Not Available" tem maior precedência que "Grant".

Cenário 3: Considerando uma requisição de "Alice" recebida às 13:15, as regras R4, R5 e R6 devem ser usadas. Dentre essas, R5 e R6 têm maior precedência do que R4 porque elas referenciam no campo *Requester* um usuário específico, "Alice", ao invés de um grupo, como em R4. Embora R5 e R6 estejam no mesmo nível de especificidade no campo *Temporal Restriction*, R6 é mais específica do que R5 no campo *Precision*, e, portanto, será selecionada para avaliar a requisição.

Requisitos	Status de implementação
Hierarquia das políticas de privacidade	Implementado
Políticas de controle de acesso	Implementado
Plausible Deniability	Implementado
Regras temporárias	Pendente
Grupos de usuários	Implementado
Notificações de acesso ao contexto	Implementado
Relatório de estatísticas de acesso	Parcialmente
Controle de acesso interativo	Pendente
Ajuste de granularidade da localização	Implementado
Restrição temporal	Implementado
Precisão (Freshness)	Implementado
Política de privacidade sensível ao contexto	Pendente
Modo invisível	Implementado
Contrato de uso de informações de contexto	Pendente
Algoritmo de especificidade	Implementado

Tabela 4.3: Status de implementação dos requisitos de privacidade

4.2 Implementação

O CoPS e as APIs *UPM* e *CAA* foram implementados em Java e estão disponíveis para *download* em (91). A Tabela 4.3 mostra o *status* da implementação dos requisitos discutidos na Seção 3.5. Dentre esses, destacam-se aqueles implementados pelos componentes *Privacy Policy Specifity* e *Privacy Rules Management* utilizados na avaliação e gerenciamento das regras de privacidade, respectivamente.

4.2.1 Visão geral da implementação

O CoPS segue o paradigma cliente/servidor, no qual a interação pode ser feita através de uma comunicação síncrona ou assíncrona, encriptada ou não encriptada. As aplicações do Subject e Requester utilizam a API User and Policy Management (UPM) para autenticar o usuário, adicionar/remover/consultar grupos, usuários e regras de privacidade. Já o serviço de contexto utiliza a API Context Access Authorization (CAA) para consultar o CoPS se um dado Requester está autorizado a acessar a informação de localização requisitada.

Inicialmente, o usuário se autentica com o CoPS através da API UPM, que por sua vez, estabelece um canal seguro de comunicação para transmitir os dados de autenticação. Esta autenticação produzirá um token de sessão que será usado para criar um **U**ser **I**dentification **T**oken (UIT) para futuras requisições. O UIT é o hash[2] do token de sessão compartilhado entre o cliente e o CoPS durante a autenticação. A partir de então, o CoPS implementa uma autenticação simétrica encriptando/decriptando o token compartilhado com o cliente. Para garantir a autenticidade do cliente, em cada requisição, o CoPS e o cliente incrementam o token de sessão e gera um novo *hash* do mesmo.

[2]Correntemente, estamos usando o SHA-1 para gerar o *hash* do token.

O protocolo de troca e geração de token trata da sincronização dos tokens trocados entre o cliente e o servidor.

O processo de autenticação é implementado pelo *DUMAC*, que é responsável por gerenciar todas as operações de gerenciamento (adicionar/remover/consultar) de usuários e grupos. Para otimizar a implementação de tais operações, o *DUMAC* armazena as informações em memória (em *hash tables*) e em um banco de dados, mantendo esses repositórios sincronizados.

O componente *Privacy Policy Specificity* utiliza o *DUMAC* para identificar os grupos do Subject aos quais um dado Requester pode pertencer. Esta informação é necessária para verificar quais regras combinam com a requisição de acesso ao contexto, conforme explicado na Seção 4.1.5. Para avaliar uma requisição, o componente *Privacy Policy Specificity* invoca o método `runPrivacyPolicySpecificity()`, que por sua vez, retorna a regra mais específica que combina com a requisição recebida ou `NULL` se nenhuma das regras se aplica. O pseudo-código do algoritmo utilizado para avaliar a especificidade das regras é ilustrado no Algoritmo 1, onde as variáveis $Xi, i = 2, .., 4$ denotam grupos de usuários, e $Ni, i = 1, .., 5$ são subconjuntos das regras resultantes das diferentes fases da análise de especificidade.

4.2.2 Implementação do algoritmo de especificidade

Conforme ilustrado no Algoritmo 1, após receber uma requisição com os argumentos *Subject, Requester, Context Variable, Precision e Application,* o algoritmo de especificidade avalia as regras que se aplicam à requisição e retorna um conjunto vazio ou um conjunto com a regra mais específica. Um conjunto vazio significa que nenhuma regra se aplica à requisição e a política de acesso padrão (Reservado, Liberal ou Sob-Demanada) do Subject deve ser usada para deferir a autorização de acesso da requisição. Se alguma regra é retornada, a requisição deve ser avaliada de acordo com o valor especificado no campo *Result* desta regra.

Inicialmente (passo 2 do Algoritmo 1) a especificidade das regras é analisada nos três níveis de hierarquia da política na seguinte ordem: Organização, Usuário e Padrão. Em cada nível da hierarquia, o algoritmo processa a especificidade das regras considerando cinco possíveis associações entre os campos Subject e Requester (presentes nas regras). Por exemplo, no passo 2.1 o método `evaluateRulesSpecificity()` avalia a especificidade das regras que contêm o nome do Subject e do Requester; no passo 2.3 este mesmo método avalia a especificidade das regras que mencionam no campo Subject o nome do usuário e, no campo Requester, grupos do Subject dos quais o Requester faz parte.

Avaliações semelhantes são feitas nos passos 2.5, 2.7 e 2.8.

O método `evaluateRulesSpecificity()` implementa a análise de especificidade baseada no conjunto de identificadores do Subject e Requester, e com base nos demais parâmetros recebidos como argumento. No passo 4, ele faz uma consulta SQL para obter as regras de privacidade que se aplicam a uma dada requisição. Este passo implementa a especificidade dos usuários, da variável de contexto e da política de acesso selecionando para as análises seguintes todas as regras que:

- o identificador do subject em *SubjectSet* é idêntico ao campo *Subject*;
- o identificador do requester em *RequesterSet* é idêntico ao campo *Requester*;
- a variável de contexto 'C' (e.g., localização) é idêntica ao campo *Context Variable*; e
- a política de acesso 'AP' é idêntica ao campo *AccessPolicy*.

Nos passos seguintes, esse método desempenha a seleção das regras de acordo com especificidade da restrição temporal, precisão, aplicação e resultado. Todas as fases de especificidade retornam um conjunto vazio se nenhuma regra é selecionada para as análises de especificidade posteriores. Como mencionado na Seção 4.1.5, a análise de especificidade do campo *Result* é necessária quando todas as outras fases do algoritmo de especificidade já foram processadas e ainda restam duas ou mais regras no mesmo nível de especificidade.

Algoritmo 1 Algoritmo da Especificidade das Regras de Privacidade

runPrivacyPolicySpecificity(Subject S, Requester R,
Context Variable C, Precision P, Application A)

1. Set X1, X2, X3 and X4 = Ø
1.1. Let AP = Subject's Current Access Policy

2. Loop over the 3 policy levels (Organization, Individual and Default),
starting at the highest level (Organization):

2.1. X1 = evaluateRulesSpecificity (S, R, C,P, A, AP)
if (X1 != Ø) return X1

2.2. Let X2 = subject's groups which the requester belongs to.
2.3. X1 = evaluateRulesSpecificity (S, X2, C,P, A,AP)
if (X1 != Ø) return X1

2.4. Let X3 = administrator's groups which the requester belongs to.
2.5. X1 = evaluateRulesSpecificity (S, X3, C,P, A,AP)
if (X1 != Ø) return X1

2.6. Let X4 = administrator's groups which the subject belongs to.
2.7. X1 = evaluateRulesSpecificity (X4, R, C,P,A,AP)
if (X1 != Ø) return X1

2.8. X1 = evaluateRulesSpecificity (X3, X4, C,P,A,AP)
2.9. if (X1 = Ø & level != Default) go back to Step 2
else return X1

evaluateRulesSpecificity(Set SubjectSet, Set RequesterSet,
ContextVariable C, Precision P, Application A, AccessPolicy AP)

3. Set N1,N2, N3, N4 and N5 = Ø

4. Let N1 = all privacy rules for which the subject(s) in SubjectSet is
identical to the Subject field; And for which the requester(s) in RequesterSet
is identical to the Requester field; And the Context Variable C is identical
to the ContextVariable field; And the AccessPolicy AP is identical
to the AccessPolicy Field.
4.1. if (N1 == Ø) return Ø

5. N2 = runTemporalRestrictionSpecificity(N1)
if (N2 == Ø) return Ø

6. N3 = runPrecisionSpecificity(N2, P)
if (N3 == Ø) return Ø

7. N4 = runApplicationSpecificity(N3, A)
if (N4 == Ø) return Ø

8. N5 = runResultSpecificity(N4)
9. return N5 containing one (the most specific) or none rule

4.2.3 Caching das autorizações de acesso

Para reduzir o tempo de resposta da requisição de autorização de acesso ao contexto, implementamos uma *cache* na API *CAA* que armazena o resultado (e.g., Grant, Deny ou Not Available) das requisições enviadas para o CoPS. Desta forma, uma vez que a requisição do requisitante R, usando a aplicação A, para um dado Subject S, em relação a uma variável de contexto específica C (e.g., localização) com precisão P foi processada, o serviço de contexto pode obter o resultado para consultas subseqüentes em relação (R, A, S, C, P) a partir da *cache* local.

A *cache* local gerenciada pela CAA é opcional, mas quando utilizada, é completamente transparente para o serviço de contexto. Para tratar das inconsistências entre os resultados das respostas armazenadadas em *cache*, e as alterações na política de privacidade do Subject, a CAA registra-se (usando a API *Publish/Subscribe ECI da* MoCA) no servidor de eventos do CoPS como um *subscriber* interessado em ser notificado sempre que o resultado de avaliação (e.g., Grant, Deny ou Not Available) de alguma das requisições armazenadas em sua *cache* mudar. Para tanto, a CAA envia junto com a mensagem de registro (também conhecida como *subscription*) a lista de requisições processadas e armazenadas em sua *cache*. A CAA envia uma nova *subscription* sempre que processar uma requisição do serviço de contexto que não está armazenada na *cache*. Dessa forma, sempre que ocorrer uma mudança na política de privacidade de um Subject, o CoPS avalia novamente as requisições de acesso ao contexto (registradas por cada subscriber) relacionadas a esse Subject e verifica se tal alteração gera um resultado de avaliação diferente para cada uma delas. Se tal atualização mudar o resultado de avaliação de alguma requisição, o servidor envia uma notificação de atualização de *cache* para o(s) *subscriber(s)* correspondente(s).

Depois que o Subject definiu a sua política de privacidade, provavelmente as regras de privacidade só serão atualizadas esporadicamente, e conseqüentemente, o número de notificações de atualizações de *cache* será pequeno. A partir dos resultados dos testes preliminares, percebemos que o uso da *cache* reduz consideravelmente o número e o tempo de resposta das requisições de autorização de acesso ao contexto.

O principal problema em utilizar a *cache* para armazenar os resultados das autorizações de acesso, é que existe um curto intervalo de tempo entre a atualização da política de privacidade e a entrega da notificação de atualização da *cache*.[3] Durante esse intervalo, existe um potencial risco da autorização de

[3]Na maioria dos casos, a latência da rede é menor que 3 segundos.

um acesso, obtida a partir da *cache*, não estar em conformidade com a política atual do Subject. Cabe ao desenvolvedor do serviço de contexto analisar se o ganho de desempenho com o uso da *cache* pode compensar o eventual risco de um resultado de acesso incorreto.

4.3 Integração com aplicações

Para integrar as aplicações ao CoPS, o desenvolvedor deverá, em uma fase inicial, modificar a implementação da aplicação cliente sensível ao contexto e do serviço provedor de contexto incorporando as APIs UPM e CAA, respectivamente. Para amenizar o impacto de tais mudanças, tentamos abstrair e facilitar ao máximo os detalhes de comunicação e interação com o CoPS através das referidas APIs. Para estar em conformidade com o modelo proposto, a aplicação cliente LBS e o serviço de contexto devem seguir o modelo de interação ilustrado na Figura 4.1. A seguir, descrevemos os requisitos requeridos para a integração de tais aplicações ao serviço de privacidade.

4.3.1 Integração com as aplicações clientes

Para o CoPS, existem duas categorias de aplicações clientes, a aplicação sensível ao contexto, por exemplo, a aplicação LBS utilizada pelo *Requester* para acessar a localização do *Subject*, e a aplicação de gerenciamento da política de privacidade utilizada pelo *Policy Maker* para administrar as regras, usuários, grupos, dentre outros. Seguindo o modelo de interação do serviço de privacidade, essas aplicações devem autenticar o usuário para, então, estarem aptas a enviar suas requisições para o serviço de contexto ou para o CoPS, respectivamente.

Essas aplicações utilizam a API UPM para autenticar o usuário e interagir com o CoPS. A autenticação é implementada através do método `authenticateUser(String user, String password, ...)` disponível na classe `Authenticator`. Para atestar que um dado usuário já foi autenticado, essas aplicações devem encapsular o *UIT*, obtido como valor de retorno do método de autenticação, em suas requisições.

A API UPM também oferece para a aplicação cliente um gerenciador de eventos através do qual ela pode receber e processar as consultas de controle de acesso interativo (requerido em regras com resultado "Ask Me" ou Política de acesso "Sob-Demanda") implementadas pelo componente *Interactive Access Control*, discutido na Seção 4.1.2. Para tanto, a aplicação deve adicionar um "escutador" de eventos (i.e., *event listener*) que será chamado/invocado sempre

que uma notificação de consulta ao Subject for recebida. A aplicação deve interagir com o Subject e, em seguida, publicar um evento (através da interface `publishEvent(...)`) com a resposta da autorização de acesso ao contexto correspondente.

As principais mudanças a serem realizadas nas aplicações clientes estão relacionadas ao encapsulamento de informações de identificação do usuário nas requisições enviadas para o CoPS ou para o serviço de contexto. Por exemplo, a aplicação LBS deve encapsular em suas requisições um objeto `AccessRequest` que contém as seguintes informações: *UIT*, nome do *Subject*, nome do *Requester*, variável de contexto (e.g., localização) e precisão requerida (e.g., campus.predio.andar.sala). Durante o processamento de uma requisição, o serviço de contexto repassa esse objeto para o CoPS com intuito de obter a autorização de acesso. Em seguida, o serviço de contexto interpreta a resposta do serviço de privacidade para, então, decidir o que deve ser enviado para a aplicação cliente (e.g., o contexto requerido ou simplesmente uma mensagem de "Not Available"/"Deny"). Já a aplicação de gerenciamento de regras encapsula em suas requisições as seguintes informações para interagir com o CoPS: o *UIT*, nome do usuário e o tipo de comando a ser processado (e.g., adicionar/remover/listar regras, usuários, grupos, etc).

4.3.2 Integração com o serviço de contexto

O serviço de contexto usa a API CAA para solicitar ao CoPS as autorizações de acesso ao contexto do *Subject*. Essas solicitações podem ser feitas serialmente ou em grupo encapsuladas em uma lista de requisições. A forma de consulta ao CoPS, normalmente, é definida pelo tipo de comunicação da aplicação LBS com o serviço de contexto. Essa comunicação pode ser síncrona ou assíncrona.

A interação síncrona baseia-se no padrão de comunicação requisição/resposta e a comunicação assíncrona é baseada em eventos para os quais a aplicação registra-se como interessada[4]. Na comunicação síncrona, o serviço de contexto processa a requisição da aplicação LBS, repassa para o CoPS (através do método `checkAccessAuthorization()` disponível na classe `ContextAccessManager` na API CAA) as informações necessárias (encapsuladas em um objeto `AccessRequest`) para obter a autorização de acesso ao contexto, interpreta a resposta do serviço de privacidade e decide o que deve ser enviado como resposta para a aplicação LBS: o contexto com a granularidade especificada pela política de privacidade do *Subject* ou uma mensagem de

[4]A implementação do LIS provê suporte a essas duas formas de comunicação.

controle "Not Available"/"Deny" (tal decisão fica a critério do desenvolvedor do serviço de contexto).

Na comunicação assíncrona, as aplicações LBS registram-se como interessadas em eventos publicados pelo serviço de contexto. Este, por sua vez, infere a localização do usuário e obtém a lista de identificadores das aplicações LBS a serem notificadas sobre a mesma (tais aplicações também são conhecidas como *subscribers* no paradigma de comunicação *publish/subscribe*). Ao invés de despachar o evento de imediato, o serviço de contexto deve interagir com o CoPS para obter a autorização de acesso de cada *subscriber* a ser notificado. Para tanto, o serviço de contexto, como na interação síncrona, deve utilizar o método `checkAccessAuthorization()` disponível na classe `ContextAccessManager` na API CAA. O serviço de contexto pode solicitar a autorização de acesso dos *subscribers* serialmente ou em grupo. Na consulta serial/seqüêncial, o serviço de contexto requisita a autorização de acesso através do objeto `AccessRequest` encapsulado na identificação de cada *subscriber*. Na consulta da autorização de acesso de um grupo de *subscribers*, o serviço de contexto envia uma lista de objetos `AccessRequest` obtidos a partir do identificador de cada *Subscriber* a ser notificado. Se for enviada uma lista de `AccessRequest` (cada um representando um Subscriber), o CoPS devolverá uma lista de objetos `AccessReply`, sendo que, o objeto `AccessReply` *i* representa uma resposta da autorização de acesso requisitada no `AccessRequest` *i* equivalente. Semelhante à consulta síncrona, na interação assíncrona o serviço de contexto interpreta a resposta recebida e despacha um evento (com o contexto solicitado) que condiz com a política de privacidade do *Subject*.

As principais modificações necessárias para a integração das aplicações podem ser descritas como segue.

- A aplicação LBS (i.e., a aplicação sensível ao contexto) deve adicionar um "escutador" de eventos no gerenciador de eventos da API UPM para receber e processar as consultas de controle de acesso interativo;
- A aplicação LBS deve autenticar-se no CoPS para obter o *UIT*;
- A aplicação LBS (síncrona) deve encapsular um objeto `AccessRequest` em suas requisições;
- A API publish/subscribe (i.e., pub/sub), usada pelas aplicações assíncronas, também deve encapsular na *Subscription*[5] um `AccessRequest`;
- O *Publisher* da *API publish/subscribe* deve oferecer uma interface a partir da qual seja possível obter a lista de *Subscribers* interessados em um dado

[5]A *subscription* representa o registro de interesse da aplicação em um dado evento.

evento. Isso é necessário para que o serviço de contexto possa solicitar ao CoPS as autorizações de acesso ao contexto antes de publicar o evento.

4.4 Discussões

A implementação do CoPS contempla a maioria dos requisitos de projeto discutidos no Capítulo 3, e oferece um controle de acesso flexível e de granularidade fina através dos componentes descritos em sua arquitetura tais como o componente *Privacy Policy Specificity*, *Notification Dispatch*, *Interactive Access Control*, dentre outros.

Apesar dos exemplos da aplicabilidade de alguns componentes estarem relacionados à informação de localização, a implementação atual também pode ser utilizada para tratar das questões de privacidade de outras informações de contexto.

O CoPS foi implementado em Java (com aproximadamente 27000 linhas de código) e está disponível em (91). A princípio, ele pode ser integrado a uma infra-estrutura de provisão de contexto que atenda aos requisitos definidos na Seção 4.3. Para reduzir o esforço de integração, o serviço de contexto deve utilizar a API CAA para usufruir das facilidades de interação com o CoPS tais como interfaces de comunicação síncrona e assíncrona, consultas de autorização de acesso ao contexto, *caching* para otimizar o tempo de resposta, dentre outras.

Para amenizar a complexidade de gerenciamento da política de privacidade, o CoPS oferece uma série de ferramentas e mecanismos que permitem o usuário definir e refinar, gradualmente, a sua política. Tal complexidade é uma conseqüência direta da dinamicidade das interações nas redes móveis, das necessidades de privacidade do usuário em diferentes contexto/atividades do dia a dia, dentre outros. Além das ferramentas oferecidas para facilitar o gerenciamento da política de privacidade (tais como perfis de privacidade, relatórios e notificações de acesso) é estritamente necessário que a interface gráfica utilizada pelo *Policy Maker* para esse propósito seja cuidadosamente projetada para explorar as funcionalidades providas pelo serviço.

No próximo capítulo, apresentamos a metodologia e os resultados da avaliação qualitativa e de desempenho do CoPS.