Grid勉強会

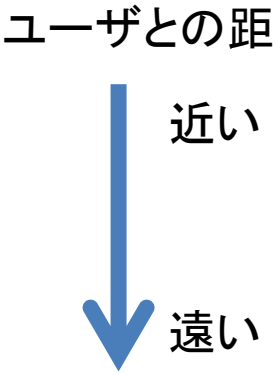
11/04/28

4.1節～4.2.3項

高瀬　亘

4.1.1 砂時計モデル

- 砂時計モデル
  - アプリケーション層
  - コレクティブ層
  - リソース層
  - コネクティビティ層
  - ファブリック層



- 昔はファブリック層とアプリケーション層だけであり、それらをつなぐ技術が不足していた。その部分がネックになり、組織を越えたリソースの利用ができないことから、くびれた砂時計を連想させた
  - ファブリック層にあるローカルリソースがアプリケーション層までうまく届かない
- くびれを解消するために、コネクティビティ層、リソース層、コレクティブ層が考え出された
  - これら3層がリソースを仮想化してユーザに提供する
  - グリッド・コンピューティング実現のためには、これらの層のプロトコル、サービスを開発する必要がある（4.1.7）

4.1.2 ファブリック層

- ローカルリソースをグリッド環境で利用するためのインタフェースを上位層に提供
  - ローカルリソース
    - CPU, メモリ, ストレージ, ネットワーク等のハード寄りのリソース
  - インタフェース
    - NFS等のプロトコルのことで、リソースにローカルアクセスするときのインタフェース
  - サービス
    - ローカル・リソースをローカル・レベルで効率的に運用するためのスケジューリングやファイル転送、ネットワーク負荷状況監視等のサービス。ローカルでリソースを使用するための前提条件。

4.1.3 コネクティビティ層

- ファブリック層のリソースにアクセスするために必要な通信（TCP/IP）と、ローカル・リソースにアクセスする際の認証のプロトコルを規定
- ファブリック層とリソース層の仲介役

- 必要なサービス
  - シングル・サインオン
    - 認証は1度だけ
  - デリゲーション
    - 権限委譲
  - 様々なローカル・セキュリティ・ソリューションへのマッピング
    - ローカル⇔グリッド間のセキュリティのすりあわせ
  - ユーザ・ベースの信頼関係
    - ユーザ中止にリソースの使い方を提供。ユーザ・オリエンテッド

4.1.4 リソース層

- 上位層にローカル・リソースを利用するためのプロトコル
- ユーザは、自分のコンピュータを操作するのに近い感覚で他者のリソースを利用できるようになる

- 以下の2種類のプロトコルがある
  - インフォメーション・プロトコル
    - リソースの構造や状態の情報の把握
  - マネジメント・プロトコル
    - リソースの操作のマネジメント

4.1.5 コレクティブ層

- リソース間の相互作用を補完するプロトコルとサービスを規定
  - 仮想化したリソースをユーザが効率よく使用できるように、ユーザを手助けするためのサービスが必要

- コンポーネントの例
  - ディレクトリ・サービス
  - スケジューリング：ジョブを流し方
  - ブローカーサービス：適切なリソースの割当
  - 監視、診断サービス
  - データ・レプリケーション・サービス：計算資源の近いところにデータを置く
  - ワークロード・マネジメントシステムとコラボレーション・フレームワーク
  - ソフトウェアのディスカバリ・サービス
  - Community Authorization Servers（CAS）：メンバ管理
  - アカウンティングと支払のサービス

4.1.6 アプリケーション層

- グリッド環境内で実行されるアプリケーションから成る
  - 他のあらゆる層のプロトコルやサービスを利用する
  - アプリケーションをグリッドに対応させることで、ユーザはグリッド環境を全く意識しなくてよくなる
    - アプリケーションが全て仲介をしてくれる

4.2 Globus Toolkit 2の概要

- Globus Toolkit
  - グリッド・コンピューティングの実現に必要な機能を実装したオープンソースのミドルウェア
  - グリッド・コンピューティングミドルウェアのデファクト・スタンダード
  - バージョン2
    - サイエンスやハイパフォーマンスコンピューティング分野で実績があるバーション
    - NAREGIはバージョン4を使用
    - 現在はバージョン5
- Globus Alliance
  - Globus Toolkitを作成した団体
  - 前身はGlobus Project
- Globus Grid Forum(GGF)
  - グリッド・コンピューティングの標準化団体
  - 後にGlobal Grid Forumになり、現在はOGF(Open Grid Forum)
- 産業技術総合研究所（産総研）
  - 日本におけるグリッド・コンピューティングの研究を牽引
  - Ninf-G（RPC）やGfarm（ファイル）といったミドルウェアを開発（Globus Toolkit2に実装）

4.2.2 Globus Toolkit2の全体構成

- 3本の柱とそれらを支える土台から成る
  - Grid Security Infrastructure（GSI）
    - 土台。グリッド・コンピューティングのシステム全体のセキュリティ担当
  - Resource Management
    - 柱。リソースとジョブの管理担当
  - Information Services
    - 柱。リソース情報の収集担当
  - Data Management
    - 柱。データの管理担当
- Globus Toolkitはあくまでツールキットであり、インストールするだけではグリッド環境は構築できない

4.2.3.1 GSI概要

- クライアント⇔サーバ間で相互認証が必要
  - クライアントが必要なもの
    - 自身のユーザ証明書
    - ユーザ証明書を署名したCAの証明書
  - サーバが必要なもの
    - サーバ自身のホスト証明書
    - サーバ証明書を署名したCAの証明書

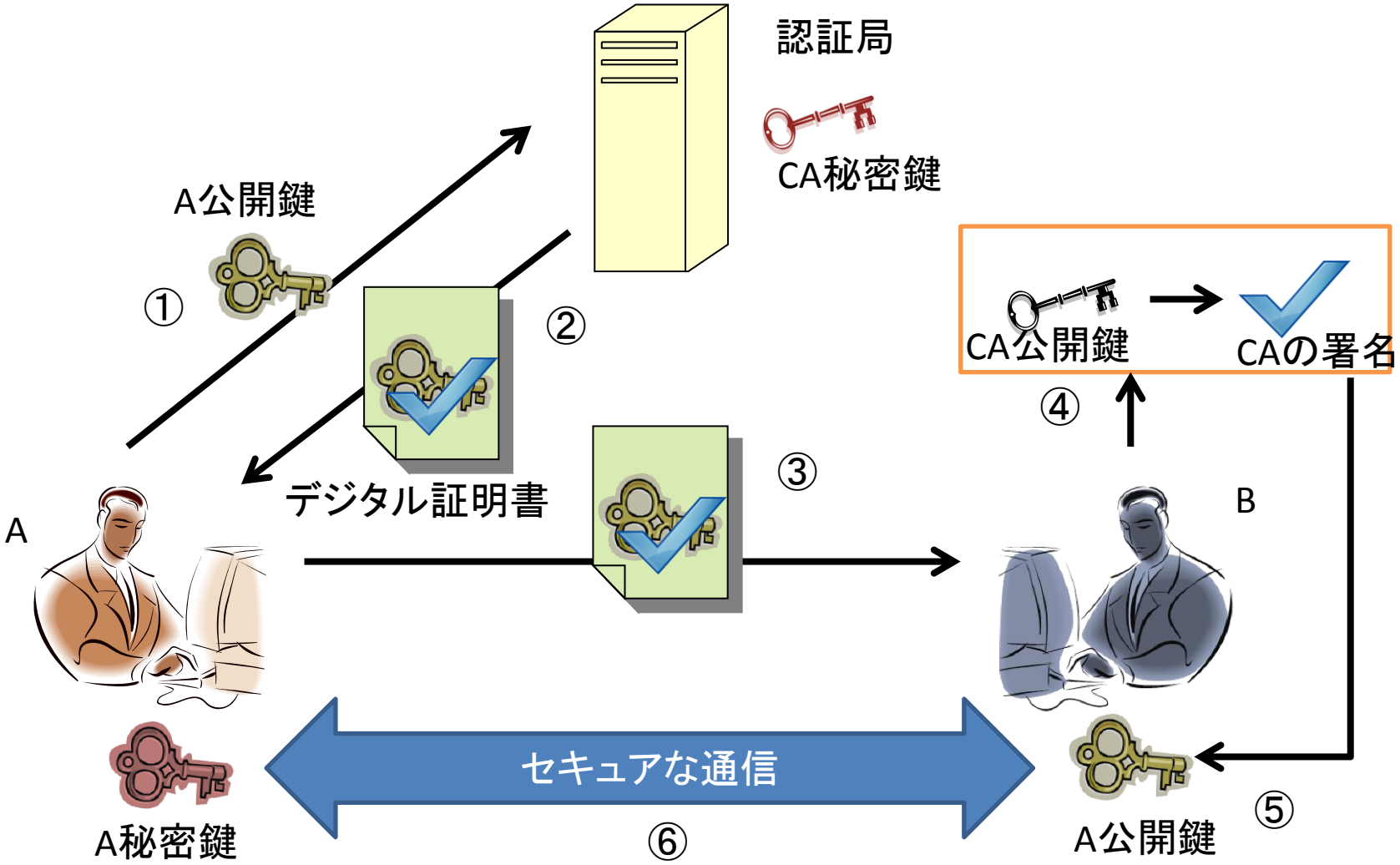
Public Key Infrastructure（PKI）

- デジタル証明書を安全に発行、管理、運用するためのシステム基盤
- 構成要素：証明書、認証機関、リポジトリ（証明書の保管場所）
- PKI構成技術：公開鍵暗号による電子署名

PKI

1. Aの公開鍵を認証機関に登録
2. 認証機関がAの公開鍵に自身（CA）の署名をつけて返信（返信されたもの＝デジタル証明書）
3. Bと通信を開始する前にAの「CAの署名付き公開鍵」をBに送信
4. Bは、Aの公開鍵についているCAの署名を、予め持っていたCAの公開鍵で復号して検証
5. 検証がOKならBは、信頼できるAの公開鍵を得たことになる
6. Aは秘密鍵を、BはAの公開鍵を持っているのでセキュアな通信ができる（秘密鍵による電子署名の作成と公開鍵による検証）
7. もし、CがAになりすまして偽のAの公開鍵をBに送信したとしてもCAの署名が無いので拒否される

PKI


認証局
CA秘密鍵
A公開鍵
①
②
CA公開鍵
CAの署名
④
デジタル証明書
③
A
B
セキュアな通信
A秘密鍵
⑥
A公開鍵
⑤

4.2.3.2 GSIの証明書

- X.509にて規格化されたデジタル証明書を用いた認証方式（PKIが前提）
- グリッド環境のすべてのユーザとサービスは、証明書を使用して本物であるかを識別

- ユーザ証明書、プロキシ証明書
  - クライアントユーザの認証に必要
- ホスト証明書
  - GRAM（Globus Resource Allocation Manager）、GridFTPの認証に必要
- LDAP証明書、Gfarm証明書等のサービスの証明書
  - MDS（Monitoring and Discovery Service）の認証に必要
  - サービスが本物かどうか
- CA証明書
  - 上記証明書が正しいものかを確認
- ユーザ、ホスト、LDAP証明書に含まれる情報
  - DN（Distinguished Name）：ユーザやホストを識別する
  - 公開鍵（秘密鍵は別に存在）
  - CAのDN
  - CAのデジタル署名：秘密鍵で暗号化されているので、公開鍵で復号して正当性を検証

4.2.3.3 プロキシ証明書

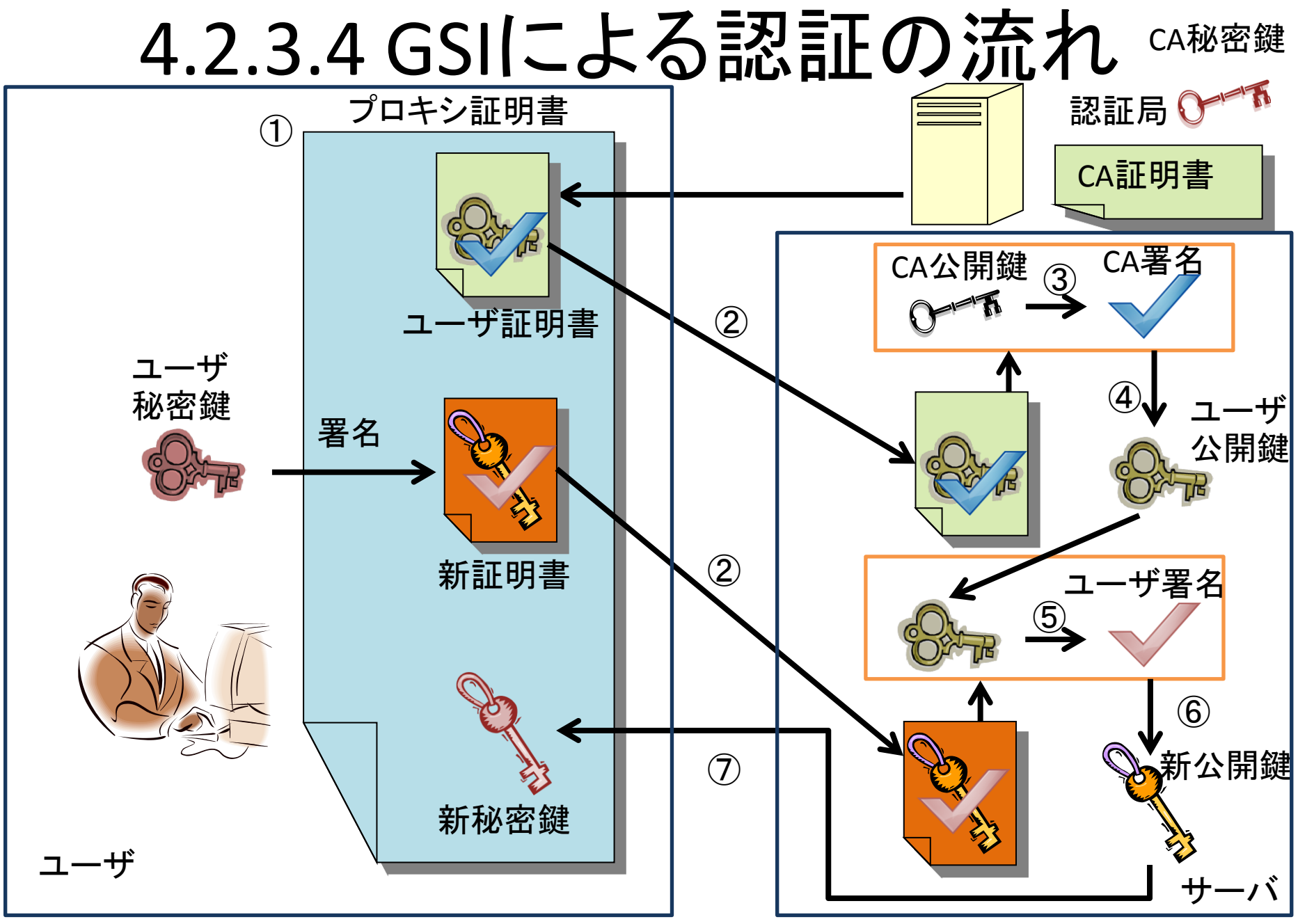
- ユーザ証明書とユーザ証明書のパスフレーズを使用して、プロキシ証明書を発行
  - プロキシ証明書のパスフレーズを設定するのはNAREGI特有
    - NAREGIはジョブの実行時間が長いものを流すことを想定して、有効期限を長く設定してある
    - 有効期限が長いので、プロキシ証明書を持っているだけで認証OKにしてしまうとセキュリティが甘くなってしまう
- プロキシ証明書の発行：グリッド環境へのログインに相当（シングルサインオン）
- グリッド環境におけるパスポート的な役割
- 他人に渡ると不正利用される可能性があるので有効期限がある

4.2.3.4 GSIによる認証の流れ

1. ユーザはユーザ証明書のパスフレーズを入力してプロキシ証明書を作成する
   - 新証明書、新秘密鍵、ユーザ証明書が含まれる
2. ユーザは、プロキシ証明書内の新証明書（ユーザの秘密鍵で署名）とユーザ証明書をサーバへ送る
3. サーバは、ユーザ証明書に対し、CAの署名を確認（サーバの持っているCA証明書を利用）
4. 確認できたら、ユーザ証明書からユーザの公開鍵を取り出す
5. 新証明書に対し、ユーザの署名を確認（ユーザの公開鍵を使用）
6. 確認できたら、新証明書から新公開鍵を取り出す
7. ユーザ側の新秘密鍵に対してチャレンジする
8. サーバ側で取り出した新公開鍵とユーザ側のプロキシ証明書内の新秘密鍵が正しい場合、サーバは通信相手が証明書に署名したユーザであることを確認できる（認証OK）
9. 以上で、ユーザの認証が終わり、その後サーバの認証がうまくいけば通信が確立される

- 認証が入り組んでいるのでレスポンスが悪い（仕方ない）

4.2.3.4 GSIによる認証の流れ



4.2.3.5 GSIによる認可

- サーバ上におけるユーザのアクセス制御には、grid-mapfileを使用
  - ユーザ証明書内のDNとサーバ上のローカルユーザを関連付ける（サーバ上に対応するアカウントが必要）

4.2.3.6 GSIによる権限委譲

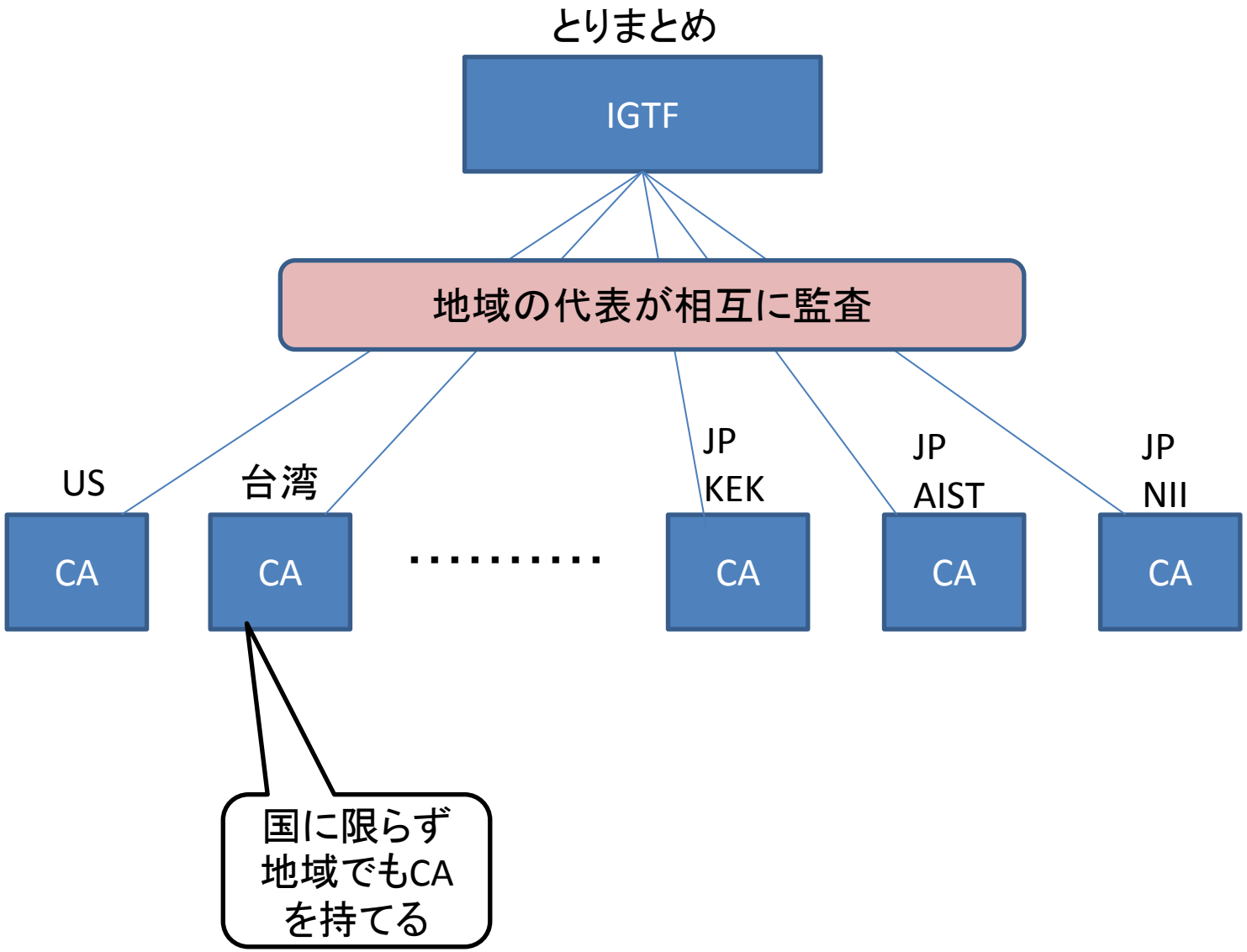
- プロキシ証明書を使用することで、一旦認証が終われば、グリッド環境内でサービスを利用する際は新たな認証が発生しない
  - ユーザがGRAMサーバ（ジョブ管理サーバ）間で認証が終わっていれば、その後、GRAMサーバ⇔GridFTPサーバ間で通信をする場合であっても、権限委譲したプロキシ証明書を使用してサーバ同士で認証を行ってくれる
    - 通常は制限付き委譲であり、委譲の委譲はできない

4.2.3.7 CAの選択

- Globus CA
  - Globus Allianceが管理
- Simple CA
  - テスト用のCA（自前で構築）
- OpenSSL
  - オープンソースのCA構築パッケージ（自前で構築）
- コマーシャルCA
  - その他組織によって提供されているCA
  - SECOM、ベリサイン等
  - 有料の場合が多い

- KEKはGlobus CA、もしくはOpenSSLを利用したNAREGI CAを使用

CA


とりまとめ
IGTF
地域の代表が相互に監査
US
台湾
JP
KEK
JP
AIST
JP
NII
CA
CA
CA
CA
CA
国に限らず
地域でもCA
を持てる

グリッド環境における証明書の取り扱い

- 証明書の中身を見るコマンド
  - $ grid-cert-info –f 証明書名

- CA証明書
  - /etc/grid-security/certificates/CA証明書
- ホスト証明書
  - /etc/grid-security/ホスト証明書
- ユーザ証明書
  - /home/ユーザ/.globus/ユーザ証明書

次回

- 11/05/10 13:30
- 4.2.4項～4.2章（一応4.3章まで読んでくる）