1. はじめに

インターネットのようなオープンネットワーク上での新しい認証技術として証明書が注目を浴び、証明書の利用を前提とする通信プロトコルの実装が増えてきている[10][13]。

通信プロトコルによっては、通信当事者間で必要となる証明書(ユーザ、プロセスおよび証明書発行局の証明書)や CRL の入手方法を特に定めていない場合がある。このような場合、通信プロトコルとは独立に、ユーザやプロセスが証明書を検索する機能や、証明書の有効性を確認する機能が必要になる。

証明書発行局の運用環境やポリシは様々であり、証明書発行局は組織のファイアウォール内で運用されている可能性もある。このような状況では、異なる証明書発行局から証明書の発行を受けたユーザやプロセス同士が証明書を利用した通信を行おうとした場合、全ての証明書発行局が全てのユーザやプロセスと通信できると仮定するのは現実的ではない。

そこで、われわれは証明書発行局間の通信を利用して証明書発行局間で証明書情報を共有し、組織外のエンティティに対して間接的に証明書情報を提供する機構を提案する。

本稿では、まず、複数の証明書発行局の存在を前提とした一般的な情報共有モデルを示す。さらに、平均応答時間を中心にモデルの考察を行ない、証明書の検索と証明書の有効性確認という2つの観点から、適したモデルを選択し設計を行なう。

提案した機構の一部は、われわれが開発した証明書発行局パッケージ(ICAP)の機能として実装中であり、その方針について簡単に述べる。

2. 証明書情報および証明書発行局の概要

本稿における証明書とは、公開鍵暗号における秘密鍵と公開鍵の対を、それを利用するエンティティ(アプリケーション、ユーザ等)と対応づけるためのものである。証明を行なうのは第三者機関である証明書発行局(以下、CA)であり、有効期限つきで署名を行ない、証明書として発行し配布する。有効期限内に、エンティティの属性や公開鍵の変更が生じた場合は、証明書を取り消す必要があり、CA は有効期限内に取り消された証明書のリストを CRL(Certificate Revocation Lists)として定期的に発行し配布する。

本節では、X.509[2] で定義されている、証明書や CRL の形式、および CA の一般的な機能概要についてまとめる。

2.1 証明書情報とは

ここでは、証明書および CRL をまとめて証明書情報と定義する。

2.1.1 証明書の形式

証明書は、あるエンティティによる署名を受け取って検証したり、あるエンティティだけに配送したい情報を暗号化する際に必要になる。

図1に X.509 バージョン3で定義されている証明書の形式のうち、オプショナルとして定義されている拡張フィールドを除いた基本的な部分を示す。

「サブジェクト」は、証明対象となるエンティティの名前であり、X.500[7] で定義されている構造的な名前形式、DN(Distinguished Name) を用いる。バージョン3では、拡張フィールドを利用することによって、DN よりも一般的に利用されている電子メールアドレスを証明書中に含めることが可能となっている。

証明書の拡張フィールドについては、バージョン3 で定義されている以外にも、PKIX[8] などによる独自拡張があり、例えば、証明書に関する問い合わせ先を示すフィールドの定義がある。

2.1.2 CRL の形式

CRL は、証明書が有効期限内に切れていないかを確認する際に必要になる。

図2に、X.509 バージョン2で定義されている CRL の形式を示す。バージョン3 ではこれらのフィールドに加えてオプショナルな拡張フィールドが定義されている。

CRL の発行間隔は、CA がサポートするアプリケーションの性質を考慮した上で、CA のポリシとして決められる。CRL の発行間

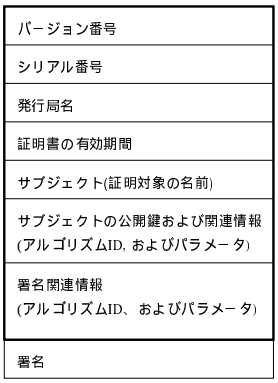

図 1: 証明書に含まれるデータ

隔が長いと、次の CRL が発行されるまでに証明書が取り消されている可能性がある。

CRL のサイズは有効期間内に取り消された証明書の数に比例して大きくなる。CRL のサイズが大きくなると、配送時の通信コストや CRL からシリアル番号を検索する時間が増大するといった問題がある。解決策としては、最初に CRL 全体を配送した後は、CRL の差分のみを配送して通信量を抑えるという方法があり、バージョン 3 では差分 CRL 形式が定義されている。

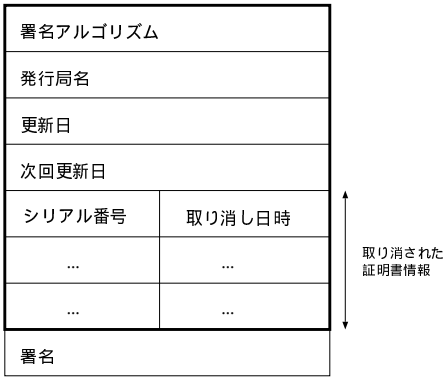

図 2: CRL に含まれるデータ

2.2 証明書発行局の概要

2.2.1 証明書発行局の機能

一般に CA の機能は、大きくは 3 つに分類される。

- エンティティ (ユーザ、プロセスなど) の認証機能
  エンティティからの証明書発行要求または証明書取り消し要求を受け付け、そのエンティティが本物かどうかを確認する機能
- 証明書情報の発行機能
  - エンティティと公開鍵の対応を、証明書として発行する機能
  - 発行した証明書が有効期間内に取り消された場合、定期的に CRL として発行する機能
- 証明書情報の配布機能
  発行した証明書および CRL を必要に応じてエンティティや他の CA に配布する機能。X.509 では X.500 ディレクトリの利用を前提にしている。

2.2.2 証明書発行局の階層モデル

インターネットのような広域分散環境では、単一の CA で全てのエンティティの証明書を管理するのはスケーラビリティに問題があり、複数の CA を分散配置させるのが現実的である。一般的には、CA は図 3にあるような階層構造をとり、CA の公開鍵はそれよりも上位の CA が証明する [6]。階層の最上位の CA と公開鍵の対応については信用するものとし、自己署名による証明書を何らかの方法であらかじめ入手することが前提となる。

階層構造では、エンティティの証明書に付加された署名を正しいと確認するまでに、そのエンティティの証明書を発行した CA から最上位の CA まで全ての証明書が必要になる。

2.3 単一証明書発行局における証明書情報配布機構

あるエンティティが通信相手の証明書を入手するには、通信相手から直接入手する場合と第三者から入手する場合がある。後者の場合、項 2.2.1で述べたように CA の証明書配布機能を利用することになる。

また、エンティティが通信相手の証明書を既に入手している場合に、その証明書が正しいかどうかを確認するには、証明書の階層上にある CA の証明書を入手する必要がある。

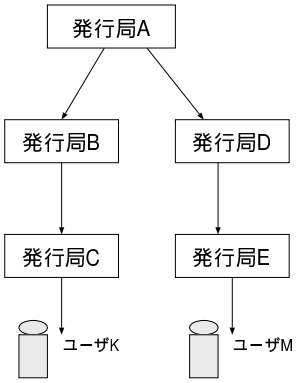

図 3: 証明書発行局の階層構造

さらに、あるエンティティが通信相手の証明書を既に入手している場合に、過去に利用されたその証明書が現在も有効かどうかを確認したい場合がある。

本項では、階層のない単一 CA における証明書情報配布機構を、証明書の検索機能と証明書の有効性確認機能の 2 つに分けてまとめる。

2.3.1 証明書の検索機能

証明書の検索は、通信相手の証明書を持っていない場合に利用する。X.500 ディレクトリを利用する場合、証明書の検索時の入力は、証明書に含まれるサブジェクト (DN) であり、出力は証明書そのものである。

既存の実装においては、X.500 ディレクトリ以外のツールを利用したり、DN 以外の入力形式を利用している場合など様々である [15]。DN 以外の入力形式を利用する例としては、項 2.1.1で述べたように、証明書中に電子メールアドレスを含めた上で、検索時の入力を電子メールアドレスとする方法がある。

一度入手した証明書をエンティティ側で 1 週間分キャッシュすることによって、証明書の検索による負荷が 2/5 程度に軽減されることが示されており [15]、このようなキャッシュ機能を前提にすれば検索頻度は通信相手数程度と考えられる。

2.3.2 証明書の有効性確認機能

証明書の有効性確認機能は、既に証明書を入手した状態で利用する。確認対象の証明書を入力とし、その証明書を発行した CA が管理する CRL を利用して有効性を確認することになる。出力は、有効か否かの有効性判定情報か、CRL の 2 通りがある。

(1) 出力が CRL の場合

有効性確認は各エンティティが行なう。例えば、X.509 では X.500 ディレクトリの CA のエントリに CRL を登録し、各エンティティが CRL を入手する方法が考えられている。

(2) 出力が有効性判定情報の場合

有効性確認は CA が行ない、判定情報を配送する。判定情報の要求および応答フォーマットについては新たに定義する必要がある [11][12]。エンティティには CRL は見えないため、エンティティは判定情報を返す CA を認証する必要がある。また、CA とエンティティ間の通信途中で判定情報が改ざんされないようにする必要もある。したがって、判定情報は CA の署名つきである。

一般に、証明書の有効性確認頻度は、証明書を利用するアプリケーションの要件によって決まり、これに関係して CRL の発行間隔も影響を受ける。

利用する度に証明書の有効性確認を行なう必要のあるアプリケーションでは、CRL の発行間隔は短い。CRL のサイズが大きい場合は CRL ではなく有効性判定情報を出力として受け取った方が通信量が少なくてすむ。

一方、利用する度に証明書の有効性確認を行なう必要のないアプリケーションでは、CRL の発行間隔が長くても構わない。CRL の発行間隔が長い場合、エンティティ側で次の発行日までキャッシュすれば CA に対する有効性確認頻度は低くなる。

3. 複数の証明書発行局における証明書情報共有機構の設計

CA が複数になると、異なる CA から証明書の発行を受けているエンティティ同士が証明書を利用した通信を行なう可能性が出てくる。例えば、PEM[4], S/MIME[3], MOSS[5] などセキュリティ強化電子メールを異なる組織のユーザ間でやりとりする場合である。この場合、エンティティは自分の知らない CA が発行した証明書を検索したり、証明書の有

効性確認を行なう必要がある。

本節ではまず、各エンティティが証明書情報を入手する際の、CA 間通信の必要性について述べる。それから、エンティティは自分の証明書を発行した CA に任意の証明書情報の問い合わせを行ない、CA 間の通信によって自動的に結果が得られるような機構の設計を行なう。

3.1 証明書発行局間通信の必要性

3.1.1 証明書の検索機能の場合

エンティティが通信相手の証明書を検索したい場合、通信相手の DN や メールアドレスを元にして、通信相手の証明書を発行している CA のアクセス先を見つける必要がある。そのためには、CA のアクセス先と、その CA が発行対象としている (エンティティの) DN やメールアドレスのドメインリストの対応表が管理される必要がある。CA が増えた場合の管理などを考慮すると、この対応表の管理をエンティティが行なうのは繁雑である。そこで、CA 間で連携して対応表を管理し、各エンティティは自分の証明書を発行した CA に問い合わせれば他の CA が発行した証明書を入手できるという機構について考える必要がある。

3.1.2 証明書の有効性確認機能の場合

エンティティが通信相手の証明書の有効性確認を行ないたい場合、その証明書中にある CA の名前を元にして、有効性確認機能へのアクセス先を見つける必要がある。そのためには、CA の名前とアクセス先の対応表が管理される必要がある。証明書中に有効性確認のためのアクセス先が含まれていれば [8]、対応表を管理する必要がなく、エンティティは直接有効性確認の問い合わせ先を調べることが可能である。この場合、CA 間の連携は必要がないように見える。

現実には、全てのエンティティが全ての CA に対してアクセス可能な環境、かつ全ての CA が全てのエンティティからのアクセスを許可できる環境とは限らない。例えば、組織毎に CA が立ち上げられていて、CA が組織のファイアウォール内に設置されている場合である。この場合、組織外のエンティティと証明書を使った通信を行なえるようにするには、CA の証明書情報配布機能を組織外と通信できるように設定する必要がある。CA の証明書情報配布機能をファイアウォール内で運用し続けるならば、組織外からのアクセスを限定できる方がよい (図 4)。そこで、CA 間で連携して証明書の有効性確認情報の問い合わせを中継し、組織外のエンティティからの CA へのアクセスを限定する方法が考えられる。

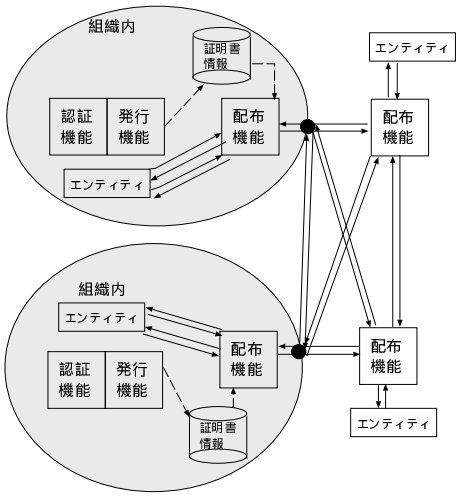

図 4: ファイアウォールがある場合の証明書情報共有例

3.2 設計上の前提条件

ここでは、機構の設計にあたり、CA、証明書、エンティティに関する前提条件をあげる。

- 証明書情報の要求から応答までは 1 プロセスで実現する
  例えば、結果を後から電子メール (SMTP) で受け取る場合は 2 プロセスとなり、本稿では範囲外である。
- エンティティは、任意の証明書の検索および証明書の有効性確認を、自分の証明書を発行した CA に対して行なう
- CA は少なくとも他の 1 つの CA からの通信要求を許可する
- CA 間に階層構造がある場合
  - 最下層の CA のみがエンティティの証明書を発行する

– 各 CA は自分の親 CA へのポインタを知っている

- エンティティは、自分の証明書を発行した CA (階層がある場合は階層上の全ての CA) の証明書を入手済である、あるいは入手する手段を知っている

3.3 証明書情報の共有機構モデルとその比較

ここではまず、各 CA をデータベースサーバ、証明書情報をデータと捉えた場合に考えられるサーバ間の共有機構モデルを列挙する。

モデル **1** 各 CA が他の全ての CA の証明書情報をあらかじめコピーし、ローカルに管理する (図 5)

モデル **2** CA 間で直接通信を行なって証明書情報を収集する (図 6)

モデル **3** CA 間の階層構造を利用し、自分の親 CA とだけ通信し、親 CA が証明書情報を収集する (図 7)

モデル **4** CA 間の階層構造を利用し、自分の親 CA と最初に通信し、親 CA が収集して返すポインタ情報を得た後、該当する CA と直接通信して証明書情報を収集する (図 8)

モデル **5** CA 間の階層構造を利用し、自分が階層上の親 CA と直接通信し、該当する CA のポインタ情報を得た後、該当する CA と直接通信して証明書情報を収集する (図 9)

モデル 3 ~ 5 は階層構造を前提としているが、階層構造のない場合でも、親 CA にあたる CA を決めることによって、同様のモデルとして考えられる。

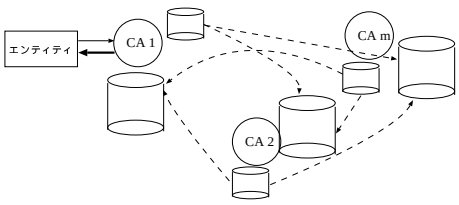

図 5: モデル 1

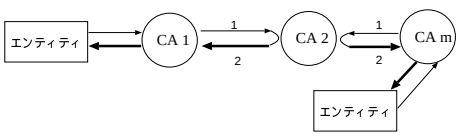

図 6: モデル 2

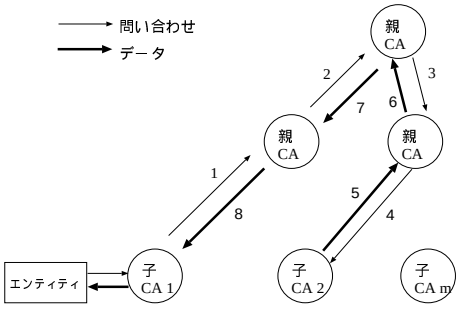

図 7: モデル 3

データの要求から応答までを 1 プロセスで実現するという条件を満たすためには平均応答時間ができるだけ小さくなるように設計する必要がある。そこで、各モデルの平均応答時間の特徴について調べることにする。

各モデルについて、表 1で示す定義および表 2で示す仮定を行ない、CA_i へのアクセス率 (λ_i) および一定時間内の CA_i でのサービス率 (μ_i) をモデル毎に計算した。結果を表 3に示す。通信時間の計算時、エンティティと CA 間の通信時間を省略している。

さらに表 4 に示す環境を想定し、M/M/1 待ち行列モデルを利用して、CA 数を変化させたときの CA 間の平均応答時間 $(T_i = 1/(\mu_i - \lambda_i))$ を求めた結果を図 10に示す。デー

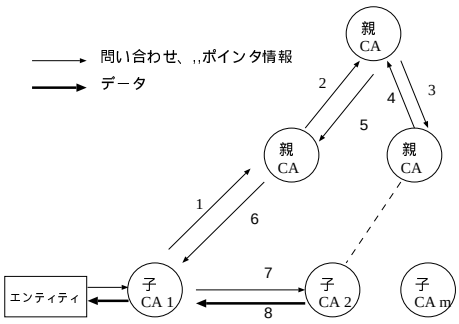

図 8: モデル 4

表 3: 各モデルの比較

	モデル 1	モデル 2	モデル 3(l=1)	モデル 4,5 (l=1)
CA_i へのアクセス率 (λ_i)	$n_i p$	$n_i p + \frac{2p(N-n_i)}{cm}$	$n_i p + \sum_{j\neq i}^{m} \frac{n_j p(m-1)}{cm}$	$n_i p + \sum_{j\neq i}^{m} \frac{2n_j p(m-1)}{3cm}$
CA_i でのサービス率 (μ_i)	$\frac{1}{aN}$	$\frac{m}{an_i+(an_i+c)(m-1)}$	$\frac{m}{an_i+(an_i+2c)(m-1)}$	$\frac{m}{an_i+(an_i+2c)(m-1)}$

	モデル 3(l=1)	モデル 4,5 (l=1)
親 CA へのアクセス率 (λ_i)	$\sum_{j=1}^{m} \frac{2n_j p(m-1)}{cm}$	$\sum_{j=1}^{m} \frac{2n_j p(m-1)}{cm}$
親 CA でのサービス率 (μ_i)	$\frac{1}{an_i}$	$\frac{1}{\alpha m}$

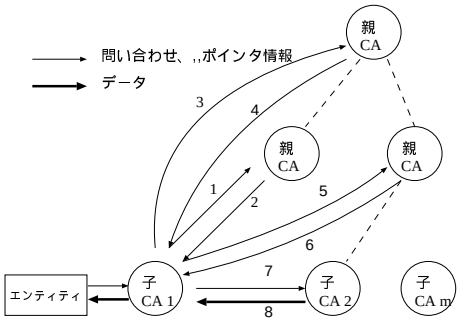

図 9: モデル 5

タベースの 1 回の検索処理時間の比例定数 a は、同一規模の単一 CA の平均応答時間 (T) が約 2 秒になるような性能を仮定して決めた。図 10の “model0” では T を示している。

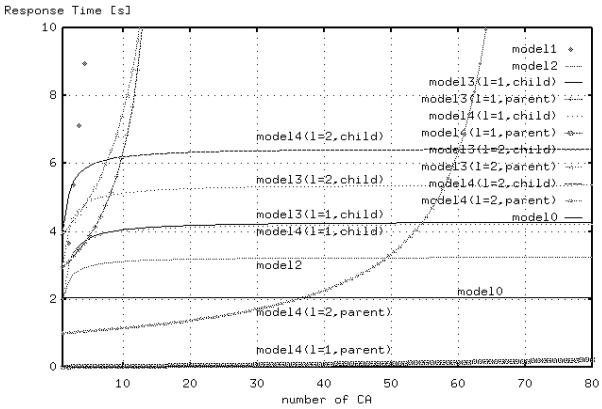

図 10: 各モデルの CA 間平均応答時間

モデル 1 は CA 数に比例してデータベースの大きさが増加するため、平均応答時間は大きくなる。CA が追加された場合は、他の CA のデータベースを全てコピーしなければならず、同じデータを CA 数 (m) 回送ることになる。また、データが追加された場合には、他の全ての CA に追加データを送信しなければならず、データの要求頻度よりもデータの更新頻度が高い場合には結局通信量が増えることになる。

表 1: パラメータの定義

m [個]	CA 数 (階層構造では最下層 CA 数)
n_i[個]	$CA_i(i = 1 \cdots m)$ のエンティティ数 (但し、 $N = \sum_{i=1}^{m} n_i$)
p_i[個/s]	1 エンティティの CA へのアクセス率
ν_i	あるアクセスが CA_i の情報に関する確率
$c_{ij}[s]$	CA_i と CA_j間の通信応答時間(RTT)
l	(階層構造の場合)CA の階層の段数

モデル 2 は最も CA 数のスケーラビリティに優れている。しかし、モデル 1 と同様に、CA が追加された場合には、他の全ての CA に追加情報を通知しなければならない。CA 数が次第に増えていく場合には管理が困難になる。

モデル 3 は 親 CA の負荷が大きく、CA 数のスケーラビリティには問題がある。しかし、CA の個数に係わらず、階層上直接関係している CA 間のみ (つまり信頼する CA 間のみ) で通信が行なわれるという特徴がある。つまり、CA 間の連携は最も緩やかである。

表 2: パラメータに関する仮定

$c_{ij} = c$
$\nu_i = 1/m$
$p_i = p$
CA_i のデータベースの大きさは n_i に比例
データベースの 1 回の検索処理時間は、データの大きさ、つまりエンティティ数に比例 (an_i)
(階層構造で) 親 CA は次の中継点を決めるための CA_i 名 とアクセス先の対応表を持つ
対応表の 1 回の検索処理時間は、対応表の大きさ、つまり CA の個数 n_i に比例 $(\alpha m\ )$
データベースのデータとアクセス先対応表の検索効率は等しい $\alpha = \frac{\text{対応表1エントリの大きさ}}{\text{データの大きさ}} a$

モデル 4 は 親 CA の負荷がモデル 3 より小さい。また、モデル 2 と比較すると、各 CA は個々の CA に対するアクセス先を知る必要はなく、CA 数が増加したときに管理が容易である。

モデル 5 は 階層の段数 (l) が 1 の場合にはモデル 4 と同一である。階層の段数が大きくなると、子 CA についてはアクセス率が高くなり、モデル 4 より平均応答時間が長くなる。親 CA については、他の 親 CA に問い合わせをして待つ必要がないため、サービス率が高くなり、平均応答時間が短くなる。モデル 5 では親 CA が任意の子 CA からの通信を許可する必要がある。

次項では各モデルの特徴を踏まえ、具体的に証明書情報の共有機構の設計について述べる。

表 4: モデル比較のための想定環境

n_i	100	既存の実験 CA 規模
p	0.0001	1 週間で 65 人と通信
c	1	[s]
データ	1000	[Byte/ 個](証明書相当)
対応表	100	[Byte/ エントリ]
a	0.02	データベース全体を 2 秒で処理
α	0.002	(表 2参照)

3.4 証明書の検索機能の設計

CA 数に対する平均応答時間と、CA 数が次第に増える場合の管理の容易さを考慮すると、モデル 4 かモデル 5 が適している。モデル 4 では、親 CA は階層上関係のある CA とだけ通信を行なうため、親 CA の通信を制限すべき状況ではモデル 4 の方が適している。

エンティティが検索キーとするのは、他のエンティティの DN や 電子メールアドレスである。これらと各 CA のアクセス先の対応表についてはあらかじめ 親 CA が 子 CA から収集しておく。

モデル 4 における検索機能の処理手順を図 11に示す。

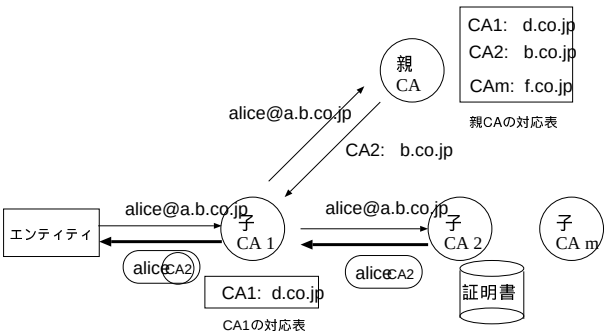

図 11: CA 間通信を利用した証明書の検索処理手順

3.5 証明書の有効性確認機能の設計

エンティティが入力する証明書中にで述べたように、アクセス先の情報が含まれていれば (項 2.1.1参照)、CA 名とアクセス先の情報は自動的に得られ、モデル 2 の欠点が解消される。

このような条件のもとでは、CA 数に対する平均応答時間の最も優れたモデル 2 が適している。この方法では、エンティティと CA 間の通信に制限がなければ、エンティティから直接該当する CA にアクセスすることも可能で平均応答時間はさらによくなる。

有効性確認機能の処理手順は図12のようになる。

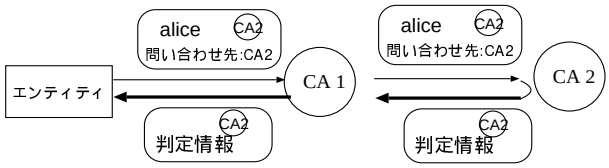

図 12: CA 間通信を利用した証明書の有効性確認処理手順

3.6 CA 間通信に制限がある場合

エンティティと CA 間の通信に制限があるように、CA 間の通信にも制限があるとすると、階層構造上直接関係する CA 間のみが通信を行なうモデル 3 のみが可能性として残される。

この場合、親 CA の負荷が高いため、親 CA の配布機能サーバを複数用意するか、CA の構造の一部で利用するか、補助的な手段として利用頻度を低くするといった条件で利用することになる。

3.7 キャッシュに関する考察

モデル 3,4,5 で親 CA の負荷を軽減するためには、子 CA と親 CA 間の通信を軽減させる必要があり、CA にキャッシュ機能を持たせることが考えられる。ここでは、各機能でキャッシュを導入する際の問題点について考察する。

3.7.1 証明書の検索機能

証明書の検索機能では、結果として得られる証明書は最新であることが求められる。つまり、証明書を入手した直後に有効性確認を行ったときには、有効と判定される必要がある。

証明書をキャッシュする場合には、常に有効な証明書をキャッシュしつづける必要があるため、CRL を入手してキャッシュから不要な証明書を取り除く作業が発生する。これは、キャッシュが利用されなくても必要な作業である。CRL のサイズが大きいと、キャッシュ容量の多くを CRL が占めることになり、証明書自体のキャッシュ容量は小さくなってしまう。CRL の更新頻度が高い場合、親 CA との通信が頻繁になって結局親 CA の負荷は軽減されない。

モデル 4,5 で CA とアクセス先の対応表をキャッシュする場合には、親 CA と子 CA の通信が減る。用意するキャッシュのサイズは証明書のキャッシュと比較して少量でよく、全体の処理時間への影響は少ないと考えられる。

3.7.2 証明書の有効性確認機能

(1) 出力が CRL である場合

CRL をキャッシュする場合には、CRL 中には次の発行日が含まれるため、データの一貫性を保つのは比較的容易である。親 CA の負荷を軽減するために、子 CA でキャッシュを行なう。キャッシュサイズに制限がなければ、子 CA は次第にモデル 1 に近付く。CRL の 1 エントリのサイズは証明書 1 個と比較して 5% ~ 10% 程度であることを考えると、検索処理時間の増加率は低くなり、親 CA の限界を抑えた運用が可能となる。

さらに、親 CA が子 CA の CRL をコピーして持つ場合には、証明書の有効性確認時の CA 間の最大通信回数を 1 回分減らし、子 CA の平均待ち時間を下げる効果が期待できる。

(2) 出力が 有効性判定情報である場合

有効性確認情報は、本来は確認対象となる証明書を発行した CA が作成するべきである。

CRL をキャッシュする場合には、有効性判定情報を作成し署名するのはエンティティの最寄りの CA となり、確認対象の証明書の署名と異なるという問題が生じる。証明書情報機構としては、エンティティが有効性確認を行なう場合に、厳密な確認が必要かどうかを指定できるようにすることが考えられる。一般には、有効性確認を頻繁に行なう必要があるアプリケーションほど厳密な確認を必要とすると予想され、その場合 CRL のキャッシュによる効果は期待できない。

有効性判定情報をキャッシュする場合には、有効性判定情報に期限が含まれていれば効果がある。

4. 発行局パッケージ(ICAP)における実装状況

われわれが開発した発行局パッケージ(ICAP)では、エンティティとCA間はHTTPを利用して通信を行なっており、ICAP単体での証明書検索機能と有効性確認については一部実装済である。今回の設計をもとにして、現在、単体の場合と同一のインターフェイスでCA間通信を利用した拡張をおこなっている。

CA間通信はWWWサーバ上で動作するCGIプログラムからHTTPを使ってリモートのCGIプログラムを呼び出す方針である。この方法で実際に、NEC EWS4800/320(CPU R4000, メモリ32MB)を使って、ローカルにCRL(約800byte)を転送したところ、平均応答時間は4.9(s)、そのうちの通信時間の平均が2.2(s)となった。通信に入る前に平均2.6(s)かかっていることから、CGIの呼び出しに予想以上に時間がかかることがわかった。さらに実験を行ない、HTTPとCGIだけで運用可能かどうか検討する必要がある。

5. まとめ

CAがファイアウォール内で運用される環境を考慮し、証明書情報をCA間の連携によって共有可能とするための機構の設計を行った。方針として、エンティティが証明書情報の要求から応答を得るまで1プロセスで実現できることとし、平均応答時間を評価の対象として5つのモデルの検討を行なった。さらに、証明書検索と有効性確認に適したモデルの選択と設計について述べた。CAでのキャッシュ機能は平均応答時間の軽減には効果的でない場合がある。特に、CA間の通信に制限のある場合はモデル3にキャッシュ機能を導入するよりも、親CAのサーバを複数にして負荷分散したり、CAの一部に適用したり、補助的な証明書情報共有手段として制限して利用するのが望ましい。

6. おわりに

現在のCAの運用動向としては、階層構造のあるCAよりも単体のCAが複数立ち上がっている状況である。今後、証明書を利用するサービスが増えてくると、CA間の相互認証を利用して、証明書情報を互いに共有したいという要求が高まると予想される。今回は、CA全体で1つのモデルを選択することを検討したが、いくつかのモデルが混在した状況についても考慮する必要があるだろう。

今後は、今回設計した方式の実装をおこない実用性について検討した後、大規模な環境での利用に適していると言われている証明書の機構の検証を進めていく予定である。

謝辞

本稿は、認証実用化実験協議会(ICAT)の広域認証技術タスクフォースでの定例会議において活発に行なわれた議論を中心に作成しました。本稿をまとめるにあたり、様々なアドバイスをいただいたタスクフォースメンバにあらためて深く感謝いたします。また、検討に参加して下さったWIDEプロジェクトのメンバ、ならびに日本電気株式会社ネットワーキング技術研究所インタネット技術部のグループメンバに感謝いたします。

参考文献

[1] ISO/IEC JTC 1/SC 21: “Information Technology - Open Systems Interconnection, Data Management and Open Distributed Processing”, Final Text of the 1993 Edition of ISO/IEC 9594-8—ITU-T Rec X.509 (Jun. 1994)

[2] ISO/IEC JTC 1/SC 21/WG 4 and ITU-T Q15/7: “Information Technology - Open Systems Interconnection – The Directory: Authentication Framework, AMENDMENT 1: Certificate Extensions”, Proposed Draft Amendment 1 to ITU X.509 (Jul. 1995).

[3] http://www.rsa.com/rsa/S-MIME

[4] J. Linn: “Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures”, RFC 1421 (Feb. 1993).

[5] S. Crocker, N. Freed, J.Galvin, S.Murphy: “MIME Object Security Services”, RFC1848, (Oct. 1995).

[6] S. Kent: “Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management”, RFC 1422 (Feb. 1993).

[7] CCITT: “The Directory: Overview of Concepts, Models and Service.”, CCITT Recommendation X.500 (1988).

[8] Internet-Draft: “Internet Public Key Infrastructure PartI: X.509 Certificate and

CRL Profile", draft-ietf-pkix-ipki-part1-03.txt (Dec. 1996).

[9] Hiroaki Kikuchi: "Internet Public Key Infrastructure Web-based Certificate and CRL Repository",(working draft)

[10] http://www.icat.or.jp/icap-doc/PEMCAT.txt

[11] 認証実用化実験協議会: "証明書の同報発信手順等に関する調査研究", 暗号認証技術を利用した鍵管理システムの調査研究 pp55–80 (Mar. 1996).

[12] 鮫島 吉喜, 菊池 浩明: "X.509 認証フレームワークの問題点と解決案", SCIS97 (Jan. 1997).

[13] 山崎 直洋, 菊池 浩明, 中西 祥八郎: "暗号化サーバによる電子メールのプライバシ強化の提案", SCIS97 (Jan. 1997).

[14] 服部 裕之, 櫻井 三子, 小林 良至, 菊池 浩明: "オンライン証明書発行局パッケージ (ICAP) の実装と評価", SCIS97 (Jan. 1997).

[15] 菊池 浩明, 黒田 康嗣, 永井 武: "プライバシ強化メール PEM における証明書配布局の実装と評価", 情報処理学会論文誌, vol.36 No.8, pp2063–2070 (Aug. 1995).

[16] 櫻井三子, 佐野晋: "PEM 公開運用実験への参加と課題", 信学会, 通信の快適性・安全性・信頼性ワークショップ (Mar. 1995).

[17] 櫻井三子, 佐野晋: "公開鍵暗号系を利用した証明書の変更を考慮した管理方式の設計", 情報処理学会, マルチメディア通信と分散処理ワークショップ pp93-100 (Oct. 1995).