

インタラクティブ フィンガープリント

須賀 祐治

株式会社インターネットイニシアティブ
101-0051 東京都千代田区神田神保町 1-105
suga@iij.ad.jp

あらまし 公開鍵証明書やPGP鍵などをトラストアンカーとして用いる場合、通常SHA-1などの暗号学的ハッシュ関数を用いて生成されたフィンガープリントを用いて2つの異なるチャネルで得られた情報の確からしさについて検証する方式が一般的である。本稿ではこれまでの静的な検証方法ではなく、何らかのインタラクションを持つ検証方法について提案し、いくつかの課題について提示する。

Interactive fingerprint

Yuji Suga

Internet Initiative Japan Inc.

Jinbocho Mitsui Bldg., 1-105 Kanda Jinbo-cho, Chiyoda-ku, Tokyo 101-0051, Japan

suga@iij.ad.jp

Abstract When using public key certificates and PGP keys as trust anchors, a common practice is to verify fingerprints obtained in two different channels by using the cryptographic hash function such as SHA-1. This paper proposes several verifying methods with interaction rather than static verification methods and presents some issues.

1 はじめに

公開鍵認証基盤(PKI)は既に30年という長い歴史[1]を持ち、インターネットの利用において不可欠な存在となっている。PCだけでなく携帯電話といった身近なデバイスを通じたインターネットショッピング・バンキングなど日常生活基盤にも浸透し、公開鍵証明書を用いた認証(Authentication)が頻繁に行われている。PKIにおいて認証を行う際には公開鍵証明書が用いられる。公開鍵証明書は公開鍵とエンティティ(仮想世界における識別子/IDや現実世界に存

在する個人もしくは組織体)を結びつける役割を持つ。現在最も一般的な公開鍵証明書フォーマットはISO/IEC X.509 [2]で規定されている。X.509 公開鍵証明書はSSL/TLS, IPsec IKE, WS-Security, DNSSECなどのセキュリティプロトコル規定、さらにS/MIME, コードサイニング, XML Signature /Encryption, PKCSなどのセキュアデータフォーマット規定で広く利用されている。

公開鍵証明書は信頼のおける認証機関(CA)が発行を行う。証明書内に含まれる公開鍵とエンティティの結び付けの確からしさはRSAなど

の公開鍵暗号を用いたデジタル署名により検証が行われる。証明書は階層的に構成可能なため複数の中間 CA 証明書を介して最終的なエンドエンティティ向けの証明書が発行されることもある。このケースにおいて、証明書の検証を自己署名(ルート)証明書まで遡ることでトラストアンカーに行き着く、つまり信頼できるというロジックで証明書の正当性を検証していく。

このとき、自己署名証明書の確からしさ、つまりトラストアンカーをどのようにセッティングするかについて考える。現在一般的な方法は OS やブラウザにプレインストールされている証明書群をユーザが信頼するという仮定で SSL/TLS などのセキュリティプロトコルを利用するという手法である。ユーザはプレインストールされた証明書群から削除したり追加したりして自らのトラストアンカーを構築する。

後者については、証明書の確からしさを確認する場合には自己署名証明書が改ざんされていないことを確認する必要がある。

■ 政府認証基盤(GPKI)におけるフィンガープリント	
■ ブリッジ認証局の自己署名証明書のフィンガープリント	
ハッシュ関数	フィンガープリント
SHA-1	3E79 D78B FCD0 ACE2 CC29 1181 7E02 FBC3 D9C4 9704 有効期間:2011年 4月11日 10:00 ~ 2021年 4月11日 10:00
SHA-1	615C B855 772A 67BB D4DD 2826 F5CD DEDD A616 F386 有効期間:2006年 4月24日 10:00 ~ 2016年 4月24日 10:00
■ 官職認証局の自己署名証明書のフィンガープリント	
ハッシュ関数	フィンガープリント
SHA-1	260D F56A 91EA B3BA E6F7 DA57 EC64 9D27 39D1 6A78 有効期間:2007年 9月27日 0:00 ~ 2017年 9月27日 0:00
■ アプリケーション認証局の自己署名証明書のフィンガープリント	
ハッシュ関数	フィンガープリント
SHA-1	7F8A B0CF D051 876A 66F3 360F 47C8 8D6C D335 FC74 有効期間:2007年12月13日 0:00 ~ 2017年12月13日 0:00
■ 注意	
SHA-1により算出したフィンガープリントは、40桁の16進数であり、「0」～「9」及び「A」～「F」の文字の組合せで示されます。ただし、フィンガープリントを表示するソフトウェアの種類又はバージョンにより、大文字又は小文字の相違、「」又はスペース(空白)の付加等表示方法が異なることがあります。	
Copyright (c) Administrative Management Bureau, Ministry of Internal Affairs and Communications	

図 1 GPKI におけるフィンガープリント

政府共用認証局自己署名証明書のフィンガープリントについては官報[3] や電子政府の総合窓口 Web サイト[4] (図 1)等において公開されており、これらの情報からダウンロードした自己署名証明書のフィンガープリント(拇印)と比較して相違がないことを必ず確認する旨[5]が記載されている。例えば自己署名証明書[6]を目視して確認する画面例が図 2 である。

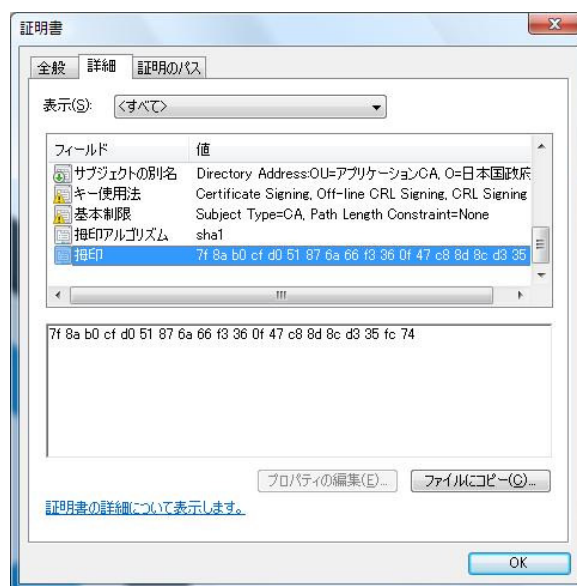


図 2 OS における確認画面の例

1.1 本稿で取り上げる課題

今年に入り認証機関へのハッキング行為により不正な証明書が発行されるケースが相次いでいる。3 月に起きた Comodo のケースでは複数の特定のエンドユーザ証明書をブラウザで受け入れない対策が行われた。一方で 8 月に起きた DigiNotar における事件[7][8]では、認証機関が持つ自己署名証明書自体をトラストアンカーとして扱わないという対策が取られている。このように認証機関の信頼性が揺らいでいることが分かる。一方で OS やブラウザベンダーから与えられたプレインストールされた証明書群を暗黙で信頼するのは同様の問題を孕んでいる。そのためユーザホワイトリストを作成するように受け入れるエンドユーザ証明書や CA 証明書を

またこの目視作業にはヒューマンエラーを誘導する問題が露呈されている。官報[3] や Web サイト[4] (図 1) および図 2 で紹介したフィンガープリントの表記は 16 進数表記 (Hex 表記) である点のみ統一されているが、

- 大文字・小文字の混在
- フォント
- セパレータの有無・位置
- 縦横表記

1.2 Textual near collision 攻撃

実証実験として OpenSSL0.9.8r コマンドラインから自己署名証明書を生成するスクリプトを実装して確認した。マシンスペックは Linux2.6.39

M1DrzCCApegAwIbAgIJA1.5mYdU1i09YMA0GCSqGS1b3dQEBBQUAMeMx CzA JBzB
 BAYTAkpQMRwwGgYDVQQKE: xNKYXBhbWVzZSBHb3Zm1cm5tZW50MRYwFAYDVQQLewI B
 cHBAsWNNhdG1vbkbNMB4XDTEyMDg2MTA2MD1xOV0XDTEyMDkxMTA2MD1xOV0wOzEL
 MAKGA1UEBhMCS1A: XHdaABgNVBA0TE0phcGFuZXN1IEdvdmVybml1bm9XZj1AUBGNV
 BAsTUDFwcGxpY2F0aW9uO0EwggE1MA0GCSqGS1b3dQEBAQUAA1BDwAwggEKAoIB
 AQCwbzJBF1MJkP8LCSyNgHrHG0+UBNMOB4D1m1cyL1kLGNqT0hY1XVOEGCJXu1ck
 EVWh5EEewX+tdb2j1MK6vFn6bjr1NoEhgCJAnfQXXy7W/AJ64ctNA5YVRVf1I1
 7tyLyP61Tj1X5p388mLpJpP2GLHhSTd2dEMTm61no/cnXkG601p4kwgY3ntR4jvB
 nQh9Wuz0172L3T0wG+py21QCQe0iURTxB0B0ncuBCL3E/M5Q80tsA2Wn6VVKH5u
 2/sgvHwrP0rBxBXP6yuYdYfayq1wtNhckD5N38fH1HHRjwdzPfxopM/C6wDU0a7R+
 RN1AbDdp3hjxnz69ebACwLwnAgMBAAEjgaUwga1wHQYDVRO0BBYEFJPdw0Jmt1E1
 kyJpH+32bhGT62NFMHMGAIUd1rMsRGAQFJPdw0Jmt1E1kyJpH+32bhGT62NFOUek
 RTBDMQswCQYDQGEwGEjKUDEcMBoGA1UECHMTSnFwYW51c2UgR292Z3JubWVudDEW
 MBQGA1UEC3MNQXBwbG1jYXRpb250Y1JAL5mYdU1i09YMAwGA1UdEwQFMAMBAfBw
 DQYJKoZIhvcNAQEFBQADggEBAKAt1FPWAjPhwcxgUSGMAULAVD0VYb7+S2+bgqJU
 Pm0aQkj33Ad1VuKKLXA7m1UOPRp68BX0VA0tKPLsZ8WpzmzqChgb+TjY0o1/V
 9TD1y32rYehWbXu9+63Z5vk0D5/OAw5zwLMJV1pai4KCoZX0TZGQFaAZXDR0Hir3
 DLvA/Z06afo9ys1+BQ1/vZWTPpbkP11te1kzG1bWanTAiR4spbACUNDL3cGvBz
 /4v8M11BvmJupmeAQ/6HQLdRwU0qc6G0DP+rggx6qeBN9/KSuUqV04o/OUA/9PQ
 Hr81BPz1RuBa1Ph1ePn9k1+r30cPHu4AFu729X591+iMk=

本稿の構成は以下の通りである。2 章にてモデルについて紹介し、3 章にて提案方式と評価、4 章にて課題と今後の研究の方向性について取り上げる。

2 モデル

物理的な媒体に印刷されるなどしたフィンガープリントとインターネット経由で取得した証明書を確認方法の一般化モデルが図3である。

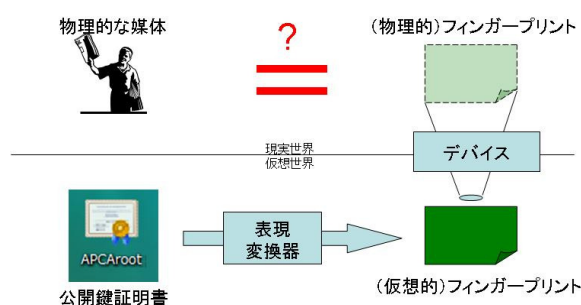


図3 フィンガープリント確認手法の一般化

仮想世界におけるデジタルデータとしてのフィンガープリントは公開鍵証明書から一方向性関数を通して、さらに表現変換器を通して仮想的(デジタル)フィンガープリントへと変換される。それをユーザが閲覧する際にはPCや携帯電話などのデバイスを通して物理的フィンガープリントとして表現される。ここでインターネットとは異なるチャネルを流通した物理的な媒体との比較が行われることとなる。

このときフィンガープリントの確認を行うケースには大きく分けてPassive modeとActive modeが存在する。前者は既存方式の目視で確認する方式が基本であり、後者はPCやその他のデバイスとのインタラクションが発生することを前提としている。特に後者の操作を「インタラクティブフィンガープリントを確認する」と呼ぶこととする。次章にてそれぞれのモードにおける新たな確認方式について提案する。

3 提案方式

3.1 Passive mode

Passive modeはデバイスから表現された物理的フィンガープリントをそのままの形式で確認する方法である。これまでのようなHex表現だけではなく、BASE32やBASE64[10]などの他のエンコーディング表現を行うことも可能である。SHA-1フィンガープリントを利用した場合、現在のHex表記では1文字辺り4ビットを確認するため40 digitsの確認が必要であるが、BASE32のケースでは32文字、BASE64のケースでは27文字の照合作業に削減できる。

特に漢字を利用できる環境においては、1漢字キャラクターあたりに埋め込み可能なデータ量はASCII文字よりも格段に増えることから確認作業の軽減が見込める。しかしUnicode文字符号化方式には割り当てのないコードワードが存在するためSHA-1フィンガープリントの先頭からすべて当てはめるというナイーブな方法ではうまく動作しない。1文字辺り24ビットと仮定した場合には、次のようにコードワード箇所をずらす方式が考えられる。先頭から24ビットがUnicodeビット空間で有効かどうかを確認しながら1ビットずつWindowをずらして7文字(=144ビット)を表現し、残り16ビットはHex表現などの既存方式を用いる方式である。課題として、同一表現になるケース、無効ケースの存在確率の大きさを計測する必要があることが挙げられる。ただし無効ケースの有無については1.2節で取り上げたように、比較的低い計算コストで確率的なデジタル署名を何度も試行することで回避できると考えられる。さらに証明書被発行者の望ましい文字(企業名の一部)を組み入れたり、ネガティブな文字(「死」など)を回避したりすることも可能となる。

さらにユーザが“読む”というアクションを拡張させる方式を考える。図4はデバイスが旋律を演奏し、ユーザは新聞などの物理的媒体に記載された楽譜との照合を行う事例である。

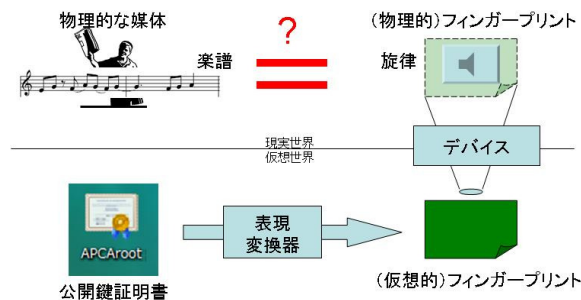


図4 楽譜と音声照合を行う事例

32音(3オクターブ弱+無音)と2/4音符単位で8パターンのリズムにして表現変換した場合、4/4で10小節程度の楽曲に抑えられるため十分実用的である。課題として難解な音階を排除して親しみやすいフレーズに変換するため前後のフレーズに制限を設けた場合には楽曲が長くなる点が挙げられる。

3.2 Active mode

デバイスは物理的フィンガープリントを投影や演奏などの表現を一方的に行うだけではなくユーザからの入力に応じることが可能である。例えばSHA-1フィンガープリントをHex/BASE32/BASE64表現で(物理的媒介に記載の文字を)入力させることが考えられる。このとき入力ごとに推奨候補文字を表記するなど選択を容易にすることで照合作業を容易にすることが可能である。さらに音声認識機能を持つデバイスにおいては、入力を音声で行うことも考えられる。このとき、意味のない文字列を入力(発音)するのではなく、意味のあるフレーズを入力させることでユーザの利便性を上げることも可能である。各単語とコードワードを多対1に関連付けておくことでSHA-1フィンガープリントから意味のある文章を生成することも可能である。

また楽譜/旋律事例においては、楽譜に記載された旋律をユーザが自ら歌うことで入力する方式が考えられる。その際には伴奏を演奏させることでユーザのインセンティブを引き出すなどの施策も検討すべきである。また、前節で指摘したように、各フレーズとコードワードを多対1に関連付けておくことで旋律は長くなる可能性が高いが、証明書被発行者の要望に応じる旋律になるように証明書を発行することも可能である。そのほかタッチパネルやデバイス自体を操作するなどゲーム感覚を持つ方式が望まれる。

文献[11]においてユーザのインセンティブをいかに引き出すかについてのひとつの方向性が示されているが、ユーザの属性や嗜好にも大きく依存するため、一概にどの方法がよいかについては選択的にすべきであると考えてられる。つまり同じSHA-1フィンガープリントに対して複数の確認手段を持つことが望ましい。

4 まとめ

本稿は公開鍵証明書をトラストアンカーとして用いる場合、通常SHA-1などの暗号学的ハッシュ関数を用いて生成されたフィンガープリントを用いて2つの異なるチャネルで得られた情報の確からしさについて検証する方式を一般化したモデルとして整理し、いくつかの方式について提案と課題をまとめた。静的な確認手法であるPassive modeでは確認作業の低減と効率化に着目して提案を行った。さらに何らかのインタラクションを持つ照合方法であるActive modeについてはユーザにデバイスを操作させることを前提にした方式を提案していた。

今後、いかにユーザがインセンティブを持つ仕掛け作りをするべきかを確認するため、さらに本稿で挙げた提案方式の有効性を示すために実証実験を行うことを検討したい。

参考文献

- [1] Christos Ellinides, “E-signatures: Vision and orientation of the european commission” IFIP TC11 24th International Conference on Information Security (SEC 2009), 2009.
- [2] ITU-T Recommendation X.509 (08/05) ISO/IEC 9594-8:2005, Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks. 2005.
- [3] 政府認証基盤を構成するブリッジ認証局システム, 官職認証局システム及びアプリケーション認証局システムの自己署名証明書のフィンガープリントの公示について(総務省), 官報平成 20 年 2 月 25 日(第 4774 号), http://kanpo.kanpo.net/product_info.php/cPath/83_350_368/products_id/979
- [4] 政府認証基盤(GPKI)におけるフィンガープリント, <https://www.e-gov.go.jp/fingerprint/gpki.html>
- [5] 法務省, 政府共用認証局自己署名証明書フィンガープリントの確認方法, http://shinsei.moj.go.jp/selfcert/fingerprint_houhou.html
- [6] アプリケーション認証局の自己署名証明書, <https://www.gpki.go.jp/apcaself/APCAroot.der>
- [7] Sophos, Falsely issued Google SSL certificate in the wild for more than 5 weeks, <http://nakedsecurity.sophos.com/2011/08/29/falsely-issued-google-ssl-certificate-in-the-wild-for-more-than-5-weeks/>
- [8] Electronic Frontier Foundation, Iranian Man-in-the-Middle Attack Against Google Demonstrates Dangerous Weakness of Certificate Authorities, <https://www.eff.org/deeplinks/2011/08/iranian-man-middle-attack-against-google>
- [9] Peter Eckersley, Jesse Burns, The (Decentralized) SSL Observatory, USENIX Security symposium '11, <http://www.usenix.org/events/sec11/tech/slides/eckersley.pdf>
- [10] S. Josefsson, The Base16, Base32, and Base64 Data Encodings, RFC4648, 2006.
- [11] Yuji Suga, Reconsideration of public key fingerprints, CRYPTO2010 Rump session, 2010.