社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－DPS－134 (15)
2008－CSEC－ 40 (15)
2008／3／6

仮想マシンを用いた暗号トークンの実装

鶴岡 行雄　　菊地 能直

NTT 情報流通プラットフォーム研究所

概要: 仮想マシンを用いた暗号トークンの構成法を提案し, その実装について報告する. 提案法は PC ハードウェア上の２つの仮想マシン上でそれぞれアプリケーションと暗号トークンモジュールを分離して実行し, 仮想マシン間の限定された手段により通信を行うことで安全性を高めたものである. 仮想化を用いないアプリケーションレベルのソフトウェア実装に比べるとより安全であり, ハードウェア実装と比べると低コストで実現できる.

Implementation of Cryptographic Token using Virtual Machine

Yukio Tsuruoka　　Yoshinao Kikuchi

NTT Information Sharing Platform Laboratories

Abstract: this paper proposes a virtual machine based construction of cryptographic token for PC clients, and shows the result of its implementation. The implementation involves two separate virtual machines for each application program and cryptographic token module in order to separate these two execution environments to protect against malwares. This virtual machine based implementation of cryptographic token is more secure than application-level software implementations, and can be realized with low cost than hardware implementations.

Keywords: authentication, virtualization, virtual appliance, cryptographic token, credential container

1. はじめに

インターネットの普及により電子商取引をはじめとするネットワーク上の価値の流通が急増している [1]. たとえばオンラインショッピングにおいてユーザはパスワードやクレジットカード番号等の購買に必要な情報を PC 端末から入力する. PC 端末はユーザの個人情報を扱う重要な役割を担うが, 一方でフィッシング, ウィルス, ボット等の PC 端末への脅威の存在がネットワーク上サービスの成長に対する潜在的な課題として問題視されている[2]. サービス提供者の観点からも, ユーザにサービスを提供するための根拠となるユーザ認証を確実に行う必要がある. ユーザ認証を安全に行うためには用いる認証アルゴリズムが安全であること, 認証アルゴリズムを PC 端末に正しく実装すること, 実装したモジュールを安全な環境で実行することが必要である. 実装するモジュールは, 認証に必要なクレデンシャル（秘密鍵等）を安全に保管する機能, 保管したクレデンシャルを用いて認証に必要なレスポンスを計算する機能を備える. このような暗号トークンモジュールの実装としては, 大きくソフトウェア実装とハードウェア実装に分けられる. ソフトウェア実装の例として Windows OS が管理する鍵ストアや Firefox 等のアプリケーションプログラムが独自に保持, 管理する鍵ストアなどがある. ソフトウェア実装による場合,簡易に利用できる反面, 不正に特権を取得したマルウェアにアクセスされ鍵を読み出されるリスクがある. 一方, ハードウェア実装 [3][4]では, 上記のマルウェアによる攻撃は防げるが, 製造コストや保守コストがかかるため高額のサービスでしか利用できないといった制約があった. ところで, 近年の PC 端末における仮想化技術[5][46]の進展に伴い, 仮想アプライアンス[6][7][43]が利用されている. 仮想アプライアンスとは PC 端末上の仮想マシン上でソ

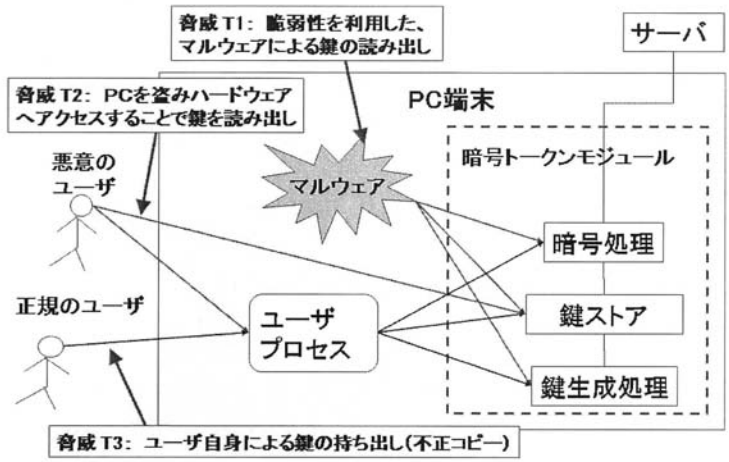

図1. ユーザ認証における脅威

フトウェアを実行することでアプライアンス機能を実現するものである. 既存の PC 端末ハードウェアを流用するため, 専用ハードウェアとして実装された従来のアプライアンスに比べ低コストで実現できる利点がある. このため本稿では仮想マシンを用いた安価かつ安全な暗号トークン機能（以下, 仮想クレデンシャルコンテナ）の実現について検討し, 実装結果を報告する.

2. PC 端末の安全性

本稿ではネットワークにアクセスする端末として PC 端末を仮定する. オープンな市場が形成された PC 端末では, クローズドな環境で開発される携帯電話機やゲーム機端末と比べアプリケーションプログラムに許される自由度が大きい. すなわち様々な API を利用できるが同時に脆弱性の入り込む余地が大きくなっていると言える. 日々新種が生み出されるウィルスやボットを検出/駆除するためにデータベースを更新し続けなくてはならないのが現状である. このように PC 端末(クライアント)はサーバに比して動作環境が多様であり, クライアントサーバ型システムの弱点になりうる. よってより高い安全性が求められている. 以下では PC 端末におけるセキュリティ上の脅威についてまとめ, それらの脅威に対する従来の対策技術を概括する.

2.1. PC 端末におけるセキュリティ上の脅威

PC 端末を用いてユーザ認証を行なう際の脅威モデルを図１に示す. ユーザ認証アルゴリズムとしては公開鍵暗号系を仮定する. 端末はネットワーク上のサーバから認証要求(通常チャレンジを伴う) を受信し, その応答として認証レスポンスをサーバに送信する. 端末は, この認証レスポンスを, 秘密鍵を使って計算する機能(暗号処理), 認証に用いる秘密鍵を端末上で生成する機能(鍵生成), 生成した秘密鍵を安全に保持する機能(鍵ストア)を持つものとする. なおここで考える認証の安全性とは悪意の第三者によってクレデンシャル(秘密鍵)が読みだせるか否かを問題とする. 具体的には以下の攻撃が想定される.

T1: ソフトウェアによる攻撃

ウィルスやボット等のマルウェアが鍵ストアから鍵を読みだす攻撃. このためには鍵ストアの構造の解析, マルウェアの送り込み, マルウェアによる特権の取得などが必要となる. 鍵ストアの構造の解析はリバースエンジニアリングや, ソースコードがある場合にはそれを参照して行われる. マルウェアの送り込みや特権の取得は, OS やアプリケーションの脆弱性を利用して行われる.

T2: PC ハードウェアへのアクセス

盗難などにより PC を入手した悪意の攻撃者が, PC のハードウェアにアクセスして鍵を読みだす攻撃. たとえばハードディスクドライブ(HDD)を抜き出して内容を読み出すなど.

T3: 鍵の不正コピー

ユーザ自身が利用ポリシーに反して鍵をコピーし多重利用などの不正行為を行うなど. たとえば一人分のアカウントを複数人で使いまわすなど.

以上の攻撃を防ぐための対策を以下に整理する.

2.2. マルウェア対策

一般に, 事前確認, 被害の最少化, 監視等の観点から対策が必要である. 事前確認としては起動時のインテグリティ確認, 被害の最少化としてはドメインの分離, 監視としてはブラックリストに基づくスキャン, フィルタリング, あるいはホワイトリストに基づく動的なインテグリティ確認などの要素技術がある.

C1-1: インテグリティ確認

マルウェアが混入していないことを保証するため, システム起動時にメモリ上にロードするモジュール（BIOS や OS 等）の改ざんの有無を確認する．具体的にはモジュールのハッシュ値を予め記憶しておき, ロード時に計算したモジュールのハッシュ値を, 記憶済みの値との比較するもの(Verified Launch) [8] [9] [10][11]や、ロードするモジュールに付加された電子署名を検証するもの[12]などがある．これらにより安全なTCB(Trusted Computing Base)が構築できる．なお上記に加えてブート時にロードされるモジュールの依存関係の管理も必要となる．

C1-2 ドメイン分離

様々なアプリケーションが共存するユーザ環境には常にマルウェアが混入するリスクがある．仮にマルウェアが混入した場合でも, 重要な処理を行う環境に影響を及ぼさないよう被害を最小化する必要がある．このためにはユーザ環境と, 重要な処理(暗号トークン処理など)を行う環境を分離する．その手段としては, 外部ハードウェアや PC に内蔵される専用チップの利用, 仮想化を用いたソフトウェアによる分離手法に分類できる．暗号トークン処理を例にあげると, PC 外部のハードウェアを利用するものとして IC カード[13], SIM[14], USB キー[3]や[15]などがある．また PC に内蔵される専用チップを利用するものとしては, TPM[16]や通信モジュール[17]などがある．仮想化を用いてドメイン分離を実現する方法としては TXT[18] [44]や [45]などがある．PC 端末における仮想化技術については３章で概説する．

C1-3: マルウェアの検出と除去
ファイルのスキャンやトランザクションの監視によりマルウェアを検出し除去するソフトウェアが広く利用されている．市販パッケージソフトや GW サービスの形態がある．

2.3. ハードウェアアクセス対策

PC 端末の停止時には, 鍵ストアの内容をハードディスクドライブ(以下, HDD)等のストレージに保存しておく必要がある．PC ハードウェアが盗難にあった場合にはストレージの内容は読み出される可能性があるため, ストレージ上で鍵ストアの秘匿性を確保する必要がある．このための市中技術として, 暗号化機能を持ち起動時にパスワードを要求する HDD [19]や,また PC に内蔵された IC チップである TPM [16], OS レベルの暗号機能 BitLocker [20]などがある．

2.4. 鍵の不正コピー対策

鍵ストアに対する適切なアクセス制御を行うことで鍵のエキスポートを制限し, ユーザの不正な読み出しを禁止することができる．

3 PC 端末における仮想化

コンピュータハードウェアの仮想化については従来から多くの研究や市中技術を含む実装がある．近年特に PC 端末における仮想化技術の進歩が著しい[47]．1999 年に VMware 社が VMware Workstation を発売して以来, 多くの仮想化ソフトウェアが市販されており, サーバ同様, 安価なエンドユーザ向け PC でも利用できる．当初はハードウェア抽象化のために多くのコードが必要であったが, プロセッサベンダが仮想化支援技術 VTX[21], AMD-V[22]をプロセッサに組み込んだことで仮想化ソフトウェアの実装がより容易になった．たとえばソースコード規模で 3K ステップ程度の仮想化ソフトウェア TVMM [23]が報告されている．これは一般的な OS のコード規模と比較すると, 相当に小さい TCB と言える．また今後の I/O 仮想化技術[24][25][26]の進歩によりさらなる実装の容易化が期待される．PC 端末における仮想化の利用形態のひとつに仮想アプライアンスがある．これは従来専用ハードウェアで実装されていたアプライアンスを PC 端末上の仮想マシン上で実行されるソフトウェアとして実現するものであり, 対応端末[27 や, さまざまな応用が提案されている．仮想化の一つの機能としてドメイン分離がある．２つのソフトウェアモジュールを別の仮想マシン上でそれぞれ実行することで, 相互の独立性が高まる．ただし仮想化の実現形態によってセキュリティ上の分離度は異なる．たとえば Xen[28]等, 仮想マシンを管理する仮想マシンマネジャーの管理下で OS が動作する形態ではより分離度が高いといえる．一方 Java VM [29]等, 抽象度の高い中間コードを実行する仮想マシンでは, 仮想マシン上の悪意のコードから, 仮想マシンの外界の情報を保護することはできるが(砂箱モデル)、逆に仮想マシンの外界の悪意のコードから仮想マシン上のコードを保護できない．このように一方向の保護であるためドメイン分離としては限定的である．

4．仮想クレデンシャルコンテナ

4.1. 機能構成

本節では, ２章で説明した脅威を防ぎ安全に認証を行うための PC 端末の構成について検討する．

ソフトウェア攻撃 T1 に対しては, マルウェアの検出・除去手法としてのウィルス対策ソフトが広く普及している（C1-3）．ただしゼロディ攻

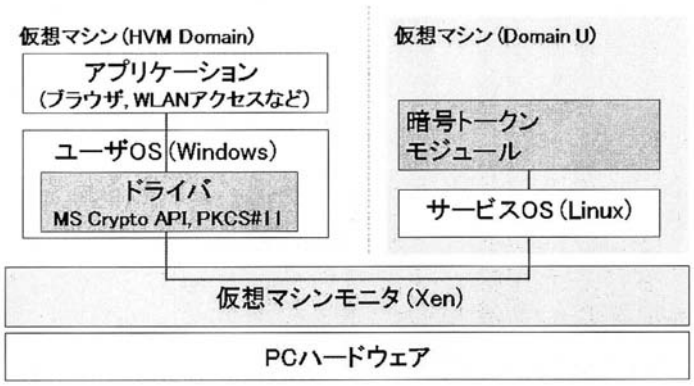

図2. 仮想クレデンシャルコンテナの構成

撃等を考慮すると対策 C1-3 のみでは一定のリスクが残る.このため提供サービスの重要度によってはドメイン分離(C1-2)やインテグリティ検証(C1-1)との組み合わせが必要となる. ここでは仮想化技術によりドメイン分離を実現することとし, 仮想マシンモニタとしてインテグリティ検証(Verified launch)に実績のある Xen ハイパバイザ(以下, Xen)を用いることとした.

なおアプリケーションを実行するユーザ OS と暗号トークンモジュールを動作させるサービス OS それぞれに仮想マシンを割り当てる.起動時のインテグリティ確認は BIOS, ブートローダ, Xen, サービス OS, 暗号トークンモジュール, ユーザ OS 上のドライバなどに対して行うことを想定する. ユーザ OS 環境はモジュール数が多くまた変更が頻繁に行われるため, 仮にインテグリティ確認を行おうとすると, 基準となるハッシュ値の管理は煩雑になる. このため重要なコンポーネントのみ検証対象とすることは運用上有効と考えられる.

ハードウェアアクセス攻撃 T2 に対しては,鍵ストア（具体的には認証に必要な秘密鍵等のクレデンシャルを格納したファイルなど）を暗号化して保存することで対応する. 実行時にこのファイルを復号してメモリ上にロードし認証処理を行う. ただし鍵ストアを復号するためのストレージ鍵を安全に保管しておく必要があり, このための領域が必要になる. さらに鍵の利用にあたっては PC 端末のユーザの確認が必要となる. たとえば前述した暗号化 HDD[19]や TPM[16]などが利用できる.

なおこれらの既存技術を利用する場合でも, ユーザ OS 環境で不当に特権を取得しうるマルウェアからストレージ鍵を保護する必要がある. 具体的には HDD や TPM などの I/O デバイスの仮想化（すなわち各仮想マシンへの, 論理的に分離独立した I/O デバイスの割り当て）が実現されていなければならない[26][30]. また上記のデバイスを持たない PC 端末については以下の方式を提案する. まず暗号トークンモジュールは起動時に, ネットワーク上の鍵サーバにアクセスし SSL[31]などにより保護チャネル１を確立する. またキーボード等の入力デバイスとの間で保護チャネル２を確立する. 保護チャネル２を通してユーザから PIN (Personal Identification Number) 等を入力し, その PIN を鍵サーバに送信する. PIN を確認した鍵サーバは,対応するストレージ鍵を暗号トークンモジュールに送信し, 暗号トークンモジュールは受信したストレージ鍵を用いて, 暗号化された鍵ストアを復号する. そしてストレージ鍵を消去する. 入力デバイスと暗号トークンモジュール間および暗号トークンモジュールと鍵サーバ間のチャネルはそれぞれ保護されているため安全とする. また暗号トークンモジュールの実行環境, および PIN 入力やネットワークアクセスに用いる I/O デバイスは, 仮想マシンモニタにより, ユーザＯＳが動作する仮想マシンとは独立した仮想マシンに割り当てられ動作するため安全である. また起動時のインテグリティが確認されるので安全である. 加えて実行中に動的にインテグリティ確認を行ってもよい.

鍵の不正コピー対策については上述したように暗号トークンモジュールが管理する認可設定により行う. 認可設定を記述したファイルにはネットワーク上の管理サーバによるディジタル署名を付加し改ざんを防止する. または鍵のエキスポート要求に際して,保護チャネルを通して上記管理サーバに問い合わせを行い, 認可設定を確認することも可能とする. 以上により, ユーザ自身による認可設定の改ざんを防止できる.

4.2 実装

4.1 節で検討したＰＣ端末の構成の一部について実装を行った. 今回実装したのは主にソフトウェア攻撃対策の一部についてであり, 残りは今後実装を予定している.

アプリケーションを実行させるユーザ OS はアプリケーション数およびユーザ数の多い

Windows とした．Xen 上で Windows を改変することなく実行するためには，プロセッサの仮想化支援機能[21][22]が必要であり，同機能を持つ PC 端末を前提とした．

暗号トークンモジュールはアプリケーションとは別の仮想マシン上で実行する必要がある．また暗号トークンモジュールを実行させるサービス OS は最小限の機能でよい．ここでは仮想マシンとして Xen の Domain U を，サービス OS として Linux を用いた．このほか Xen を管理するため Linux ベースの domain 0 が存在する．Windows 上のアプリケーションはセキュリティ API によりドライバを呼び出す．なおセキュリティ API は Microsoft Crypto API[32]と PKCS#11[33]をサポートする．ドライバは TCP/IP 上のソケットにより Domain U 上の暗号トークンモジュールと通信する．

5．アプリケーション

クライアント認証を必要とするアプリケーションの例として Web ブラウザとネットワークアクセスをとりあげ，実装した仮想クレデンシャルコンテナの動作を確認した．

Web ブラウザ(Firefox)と暗号トークンモジュールはドライバを介して PKCS#11 API により通信を行う．また証明書管理も暗号トークンモジュール側で行う．

ネットワークアクセスについては Windows のネットワーク接続機能を用い，ネットワークアクセス認証は 802.1x[34]を用いた．ネットワーク接続機能はドライバ(CSP)を介して Crypto API により暗号トークンモジュールと通信する．

処理の流れは，ユーザ OS 上のアプリケーションが認証処理を行う際に，ドライバを介して認証処理の要求が暗号トークンモジュールに送信され，処理結果がドライバ経由でアプリケーションに返されるというものである．

6．考察

上記実装および２つのアプリケーションへの適用を通して，提案技術のユーザビリティを確認した．アプリケーションの使用感については，通常のソフトウェアベースのトークン実装と比べて違いなく利用できた．ただし現実装ではディスプレイへの出力が Domain 0 の Linux 経由になるため使いにくい点があった．今後は Xen を I/O パススルーで利用することで改善されると考えられる．起動についてはユーザ OS の起動に加え仮想マシンモニタ，Domain 0 および Domain U の Linux、暗号トークンモジュールをそれぞれ起動しなければならないため，起動時間のオーバヘッドがある．これについても今後短縮を図る予定である．

また初期設定については，既存環境からのマイグレーションが可能であることを確認した．具体的には仮想化を利用していない PC 端末に対して，必要なコンポーネントを後からインストールし再起動することで，本環境に移行できる．ただし移行作業はまだ手動で行う部分があるので今後は容易化を進めたい．

仮想マシンの安全性の利点については上で述べたとおりだが，一方で仮想化を利用したマルウェアが報告されている[35] [36] [37] [38]．これらについては今後より詳細な検証が必要である．

7．まとめ

本稿では仮想マシンを使った安全で低コストな暗号トークン機能の実現に向けた実装方式の検討を行い,実装結果を報告した．今後は安全性の検証や利便性向上，導入の容易化，ID 管理機能[39][40]の統合や，プロビジョニング機能 [41][42]の統合，アプリケーション対応の強化などに取り組む．

謝辞：著者らは本研究の遂行にあたりサポートいただいた高橋健司氏，坂本泰久氏，コメントいただいた深沢友雄氏，実証にあたり協力いただいた佐藤正宏氏に感謝する．

参考文献

[1] 平成 18 年度電子商取引に関する市場調査報告書，平成 19 年 3 月，経済産業省
[2] 国内におけるコンピュータウィルス被害状況調査報告書 IPA, 2003 情財第 0314 号, 4 月, 2004.
[3]http://www.cyprotect-dl.de/safenet/iKey/iKey_4000_datasheet.pdf
[4] RSA SecurID Authentications, RSA Inc. http://www.rsa.com/products/securid/datasheets/SID_DS_0307.pdf
[5] James E. Smith, Ravi Nair, Virtual Machine, Elsevier
[6] Virtual Appliance Marketplace http://www.vmware.com/appliances/
[7] Miwa, S., Ohno, H., “A Development of Experimental Environments “SIOS” and “VMNebura” for Reproducing Internet Security Incidents,” Journal of the National Institute of Information and Communications Technology , Vol.52 No.1/2, pp.23-34 (2005).
[8] Garfinkel, T., Pfaff, B., Chow, J., Rosenblum, M., Boneh, D., ”Terra: A Virtual Machine-Based Platform for Trusted Computing,” in Proc. of Symp. on Operating Systems Principles (2003)
[9] Reiner Sailer, Xiaolan Zhang, Trent Jaeger, Leendert van Doorn., “Design and Implementation of

a TCG-based Integrity Measurement Architecture," 13th Usenix Security Symposium, Aug. 2004.
[10] Suzaki, K., Yagi, T., Iijima, K., Tsukamoto, J., Nakamura, M., Munetoh, S., "OS Circulation Environment "Trusted HTTP-FUSE Xenoppix," Virtualization Miniconf at LinuxConfAu07.
[11] Junhong Jiang, Kevin Tian, Chris Wright, Don Dugger, "Updating Xen for the Client Environment," Intel Open Source Technology Center
[12] www.verisign.co.jp/codesign/authcodeWP.html
[13] Toji, R., Wada, Y., Hirata, S., Suzuki, K., "Smart Card Information Sharing Platform. NICE. A Network-based Platform for Multi-application Smart Cards.," NTT Review, Vol 14, No.1, pp.13-19, 2002.
[14] 南本, 保谷: UIM バージョン 3 の開発, NTT DoCoMo テクニカルジャーナル, Vol.15, No.1, Apr (2007).
[15] http://www.ubiquitous-forum.jp/documents/sympo20040624/ubi-sympo_ntt.pdf
[16] Trusted Computing Group, "TCG TPM Specification Version 1.2"
[17] 増井,竹内,平間,原田: FOMA ユビキタスモジュールの開発とネットワークへの機能追加, DoCoMo テクニカルジャーナル, Vol.14, No.1, Apr (2006).
[18] David, Grawrock, The Intel Safer Computing Initiative, Intel Press
[19] DriveTrust Technology : A Technical Overview, Technology Paper, Seagate Technology LLC.
[20]http://www.microsoft.com/japan/technet/windowsvista/library/c61f2a12-8ae6-4957-b031-97b4d762cf31.mspx
[21] Intel Corporation, "IA-32 Intel Architecture Software Developer's Manual Vol.3B: System Programming Guide, Part 2," 2006.
[22] 岡野浩史, 仮想化を支えるハードウェア技術 (AMD), 情報処理 Vol.49 No.1 Jan 2008
[23] 金田憲二, 単一システムイメージを提供するため VMM と AMD-V を対象とする軽量 VMM, 第 6 回仮想化実装技術勉強会
[24] AMD, "AMD I/O Virtualization Technology (IOMMU) Specification," 34434 Rev 1.20
[25] Intel, "Intel Virtualization Technology (Intel VT for Directed I/O (Intel VT-d)," Technology@Intel Magazine, pp.19--22, May. 2007.
[26] Single Root I/O Virtualization 1.0 Specification, PCI-SIG
[27] Intel, "Intel Centrino with vPro Technology and Intel Core2 Processor with vPro Technology," white paper
[28] Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Pratt, I., Warfield, A., Barham, P. and Neugebauer, R.: Xen and the Art of Virtualization, in Proc. of Symp. on Operating Systems Principles (2003).
[29] Lindholm, T., Yellin, F., Java Virtual Machine Specification - 1999 - Addison-Wesley Longman
[30] Berger, S., Caceres, R., Goldman, K.A., Perez, R., Sailer, R., Doorn, L.v.; vTPM: Virtualizing the Trusted Platform Module, IBM Research Report RC23879 (W0602-126) Feb. 14, 2006.
[31] Dierks, T., Allen, C., RFC2246 The TLS Protocol Version 1.0, Jan. 1999.
[32] セキュリティ API に関する技術調査 –part 3- .NET Crypto API: 機能と利用法, IPA 15 情経第 1516 号, 2004 年 2 月
[33] RSA Laboratories, "PKCS #11: Cryptographic Token Interface Standard," Version 2.01, Dec. 1997
[34] IEEE Std 802.1X-2004, Port-Based Network Access Control, IEEE Computer Society, 13 Dec. 2004
[35] http://bluepillproject.org/
[36]Dino Dai Zovi, Hardware Virtualization-Based Rootkits, BlackHat
[37] Samuel T. King, Peter M. Chen, Yi-Min Wang, Chad Verbowski, Helen J. Wang, Jacob R. Lorch, "SubVirt: Implementing malware with virtual machines", in Proc. IEEE Symp. on Security and Privacy, Oakland, May 2006.
[38] Garfinkel, T., Rosenblum, M., When Virtual is Harder than Real: Security Challenges in Virtual Machine Based Computing Environments,
[39] Orihara, S., Tsuruoka, Y., Takahashi, K., Certificate-less User Authentication with Consent, Proc. of the third ACM Workshop on Digital Identity Management (2007).
[40] Orihara, S., Tsuruoka, Y., Takahashi, K., Trusted-link: web-link enhancement for integrity and trustworthyness, Proc. of the second ACM workshop on Digital Identity Management (2006).
[41] OSGi, "OSGi Service Platform Release 4 Version 4.1"
[42] Liberty Alliance ID-WSF Advanced Client 1.0 Specifications
[43] Kazuhiko Kato, Modeling and Virtualization for Secure Computing Environments. 12th Annual Asian Computing Science Conference (ASIAN'07)
[44] Intel Trusted Execution Technology, Preliminary Architecture Specification – Preliminary Architecture Specification and Enabling Considerations, Aug. 2007,
[45] Open and Secure Terminal Initiative (OSTI) Architecture Specification Revision 1.00
[46] Popek, G.J., Goldberg, R.P., "Formal Requirements for Virtualizable Third Generation Architecture," Comm. of the ACM, Vol. 17, No. 7, pp.412-421 (1974).
[47] Adams, K., Agesen, O., A Comparison of Software and Hardware Techniques for x86 Virtualization. Proc. of ASPLOS Oct., 2006.