組込み機器の暗号ソフトウェア実装に対する攻撃と対策 – Canon EOS と SONY PS3 の事例を踏まえた考察 –

大石 和臣　　松本 勉

横浜国立大学大学院 環境情報研究院
240-8501 横浜市保土ヶ谷区常盤台 79-7
{oishi,tsutomu}@ynu.ac.jp

あらまし PC の暗号ソフトウェア実装に対する攻撃は数多く知られている．特に，DVD あるいは Blu-ray Disc の再生アプリケーションや，暗号化コンテンツを配布して正規ライセンスを持つ者だけが視聴できるように制御するディジタル著作権管理（Digital Rights Management）システムについては，攻撃と対策のいたちごっこが続いている．最近は暗号ソフトウェア実装が搭載された組込み機器が増え，それらに対する攻撃も発表されるようになってきた．本稿では，そのような事例として Canon のディジタル一眼レフカメラ EOS と SONY のゲーム機 PS3 に対する攻撃を取り上げ，それらの事例を踏まえて対策を考察し，攻撃者による解析を検知する従来より耐性が強い方法を提案する．

A Study of Security for Cryptographic Embedded Software: Lessons Learned from the Canon EOS and SONY PS3 Cases

Kazuomi Oishi　　Tsutomu Matsumoto

Yokohama National University
Research Institute of Environment and Information Sciences
79-7 Tokiwadai, Hodogaya-ku, Yokohama 240-8501, JAPAN
{oishi,tsutomu}@ynu.ac.jp

Abstract Cryptographic software on personal computer has been extensively attacked. In particular, vicious circle of attack and defense has been continuing with respect to DVD or Blu-ray disc player applications and Digital Rights Management systems. Recently, many embedded devices tend to implement cryptographic software and some of the attacks against them are published. In this paper, we focus our attention on the Canon digital SLR camera EOS and SONY game console PS3 cases. Studying the attacks, we discuss the countermeasure and propose a method to detect attacker's analysis. The proposed method is more resistant than previous detection methods.

1　はじめに

パーソナルコンピュータ（以下，PC）や携帯電話，スマートフォーン，ディジタルカメラ，ゲーム機，インターネットが広く普及し，ディジタル情報を暗号を用いて保護することは一般的になった．PC ではユーザが自由にプログラムを作成して実行することができるのに対して，携帯電話やゲーム機などは，ユーザがプログラムを自由にインストールして実行することは通常はできず，あらかじめ用意された特定の機能だけを利用する．特定の機能を実現するためのコンピュータシステムを組み込まれた機器あるいはシステムを，組込み機器あるいは組込みシステムと呼ぶ．組込み機器は非常

に多い．炊飯器や洗濯機等の家電，携帯型音楽再生装置やカメラやゲーム機等の AV 娯楽機器，プリンタや複合機等の OA 機器，コンビニ等の POS 端末，自動車の制御を行う ECU 等が例としてあげられる．これらの組込み機器は，特定機能を実装したソフトウェア（ファームウェアと呼ぶ）によって制御される．

組込み機器が価値の高いデータを処理するようになり，その保護のために暗号を実装する例が増えている．最近は，組込み機器向け CPU の計算能力が高まり，メモリーの大容量化と低価格化も進んだため，公開鍵暗号のような計算量の多い暗号もソフトウェアで実装できるようになってきた．

PC の暗号ソフトウェア実装に対する攻撃は数多く知られている．最近は暗号ソフトウェア実装が搭載された組込み機器が増え，それらに対する攻撃も発表されるようになってきた．本稿では，そのような事例として Canon のディジタル一眼レフカメラ EOS と SONY のゲーム機 PS3 に対する攻撃を取り上げ，それらの事例を踏まえて対策を考察し，攻撃者による解析を検知する従来より耐性が強い方法を提案する．

以下，2 で攻撃事例を述べ，3 で従来研究されてきた対策について説明する．4 で考察を行い，対策方法を提案する．5 でまとめを述べる．

2 攻撃事例

2.1 DVD の CSS

一般消費者向けのエレクトロニクス機器に初めて暗号を搭載したとされている DVD（Digital Versatile Disc）は 1996 年に発売された．著作権付きコンテンツが記録された DVD が販売されるため，カジュアルコピーを防止することを目的として，独自の暗号化方式と復号の仕組みを採用したアクセス制御方式 CSS（Content Scramble System）が採用された．復号鍵を内蔵した再生装置あるいは PC の再生アプリケーションは，暗号化されたコンテンツを復号して再生できるが，復号したコンテンツを抜き出してコピーすることはできないように制御される．

1999 年に，PC 用のある DVD 再生ソフトウェアのオブジェクトプログラムに復号鍵がそのまま（暗号化されずに）格納されていたことがきっかけで，暗号化方式や復号鍵等の仕様が解析された．現在，PC を用いて DVD のカジュアルコピーを行うことは容易である．

2.2 Blu-ray Disc の AACS

AACS (Advanced Access Content System) は，Blu-ray Disc に記録されるコンテンツを制御するコンテンツ保護技術である．バージョンを自動更新する機能を持ち，あるバージョンが破られても新バージョンは再びコンテンツを保護できる特徴を持つ．

2007 年に，PC 用のある再生ソフトウェアが動作している間にメモリを監視することで，そこに現れる鍵データを見つけ出すことが可能となり，そのバージョンの AACS は破られた．その後，バージョンが更新されて破られたバージョンの鍵は無効化されたが，更新されたバージョンの AACS の無効化も行われ，いたちごっこは現在も続いている模様である．

2.3 Windows Media DRM

Windows Media DRM は， マルチメディアプレーヤー Windows Media Player で採用されているディジタル著作権管理（Digital Rights Management）システムである．マルチメディアファイルは暗号化され，復号鍵は暗号化されたライセンス内に格納され，ファイルとライセンスは別々に配布される．暗号化されたファイルを再生するとき，ユーザが対応するライセンスを取得するとライセンスに従った条件で再生される．暗号化されたファイルをコピーしても正当なライセンスを取得しなければ再生できない．

2006 年 8 月に，Windows Media DRM 10 と 11 を除去するソフトウェアが公開された．その開発者によると，プレーヤ動作時に秘密のデータが暗号化されずにスタックに置かれていたとされている[1]．

[1] R. Block, “The Engadget Interview: Viodentia, creator of FairUse4WM,” September 25th 2006. http://www.engadget.com/2006/09/25/the-engadget-interview-viodentia-creator-of-fairuse4wm/

2.4 Canon EOS

キヤノン株式会社（以下，Canon）は，2002 年 11 月にディジタル一眼レフカメラ EOS-1Ds とオリジナルデータ確認キット DVK-E1 を発売した．IC カードを USB 接続型リーダライタ経由で PC に接続して，専用アプリケーションを用いて，EOS-1Ds で撮影した画像がオリジナルから改ざんされたか否かを PC で検証できる機能（以下，オリジナル画像判定機能）を実現した．

2010 年 11 月に改ざん画像をオリジナルと判定させる方法が発表された [7]．この方法は，まず EOS のファームウェアをカメラ本体から読み出して，解析ツール（IDA Pro）でファームウェアを解析して，オリジナル画像判定に用いるデータのフォーマット，暗号アルゴリズム（HMAC），鍵（HMAC の生成鍵）を突き止めた．鍵は，obfuscated form でファームウェアに格納された値，未知の関数，BodyID，BoardID 等から生成され，カメラ毎に固有の値になるような処理がされていたが，カメラから抽出することができた．そして，カメラ本体の中で行われる HMAC 生成処理と同じ処理を任意の画像に対して行うことによって，オリジナルではない改ざん画像をオリジナルと判定させられること，およびその具体的な例を示した．

2.5 SONY PS3

株式会社ソニー・コンピュータエンタテインメント（以下，SONY）が 2006 年 11 月から販売している家庭用ゲーム機プレイステーション 3（以下，PS3）は，プロセッサやメモリ，光ディスクドライブなどのハードウェアをハイパーバイザが制御し，システムソフトウェア（OS）およびゲームやビデオ再生等のアプリケーションプログラムはハイパーバイザを介してリソースを利用して動作する．正当なファームウェアやプログラムには SONY のディジタル署名が付与され，PS3 が署名検証に成功するとそれらは実行されて動作することができる．以上を含む複数のセキュリティ機構のために，PS3 のリソースを自由に利用できるプログラムをユーザが作成して実行することは困難であった．

2010 年 1 月に，ハイパーバイザを介さずに PS3 のシステムメモリに対するアクセスをユーザが得る方法が発表された．この方法は，ハードウェアに対する glitch attack[2]によってハイパーバイザのメモリ管理処理を誤動作させ，ハイパーバイザ上で動作する Linux kernel からメモリへの直接アクセスを実現したとされている[3]．これをきっかけとして，メモリとハイパーバイザをユーザが制御できるようになった．

2010 年 12 月に，PS3 のディジタル署名（ECDSA）プログラムが解析され，ECDSA の実装が ECDSA の仕様を満たしていないこと[4]が明らかにされ，全ての PS3 に共通の秘密署名生成鍵（root key）を求める方法が発表された [8]．2011 年 1 月には鍵（metldr keys）の値，署名ツールが公開され，ユーザが作成したプログラムを PS3 で実行することは困難ではなくなった．

2011 年 3 月のシステムソフトウェアバージョン 3.60 アップデートでは，（metldr key を使わないように）従来とは異なるローダを採用して起動するような変更が行われ，それまでの攻撃は無効化された模様である．

2.6 攻撃の分類

攻撃をいくつかの種類に分類できる．

プログラム非実行型 いわゆる，ソフトウェアの静的解析による攻撃．DVD の CSS はこの攻撃で破られた．

プログラム実行型 いわゆる，ソフトウェアの動的解析による攻撃．AACS，Windows Media DRM，EOS はこの攻撃で破られた．

パッケージ非加工・アクティブ型 ハードウェアへの攻撃で，チップのパッケージや配線への加工は行わずにクロックや信号電圧などを変化させて非正常動作（故障）を誘因する攻撃．PS3 はこの攻撃とプログラム実行型攻撃の組み合わせによって破られた．

---

[2] ハイパーバイザがバッファをデアロケートするタイミングで 40nsec のパルスをメモリーバスに印加する．

[3] N. Lawson, “How the PS3 hypervisor was hacked,” root labs rdist, January 27, 2010. http://rdist.root.org/2010/01/27/how-the-ps3-hypervisor-was-hacked/

[4] 仕様では，メッセージ毎に異なる変数（乱数）を使って署名を生成するが，実装では，異なるメッセージに対して同一の定数を使って署名を生成していた．

3 対策

ソフトウェアに対する攻撃に着目し，今までに研究されている対策について述べる．

3.1 ソフトウェア保護の要件

暗号のソフトウェア実装を攻撃から保護するとき次の二つの性質が求められる．これらを満たすものを耐タンパーソフトウェアと呼ぶ [1].

秘密情報守秘性 ソフトウェアに実装されている秘密のデータやアルゴリズムを不正に読み取ることが困難であること．

機能改変困難性 ソフトウェアの仕様や製作者の意図に反する機能を実現できるようにソフトウェアを不正に改変することが困難であること．

以下ではこれらの性質をソフトウェアだけで実現する方法について述べる．

3.2 秘密情報守秘性に関する既存研究

3.2.1 コードを対象とする具体的な実現方法

難読化あるいは Code obfuscation

人間による解析を困難にすることを目的とする難読化は，解析者のスキルに応じて難読化の強度が変わるため評価が難しく，解析ツールへの耐性は明らかではない．

制御フローあるいはデータフローなどの具体的なプログラム（コード）を対象とする code obfuscation[6] は，その変換アルゴリズムを知られると逆変換するツールを作成されることにより効率的に解析される可能性が高い．アルゴリズム公開型で実装されたときにプログラム実行型解析に対して十分な強度を確保できる方法が存在するか否かは明らかではない．

テーブルネットワーク実装

共通鍵暗号 DES あるいは AES の暗号化アルゴリズムの構造的特徴を利用して，固定秘密鍵を異なる形式の異なる値に変換してプログラムに埋め込むテーブルネットワーク実装（ホワイトボックス実装）は，プログラム非実行型およびプログラム実行型の両方の攻撃に対して秘密情報守秘性を持ち得るアルゴリズム公開型の有望な難読化方法として期待された．しかし，今までに提案された全ての方式に対して，全数探索よりもはるかに少ない計算量で秘密鍵を抽出する攻撃が示された [13].

3.2.2 理論的な実現方法

Function Obfuscation

任意の関数を VBB（Virtual Black-Box）プロパティを満たして obfuscate できる obfuscator は存在しないことが理論的に示されている [2]．VBB プロパティとは，対象の関数に関して入力と出力の対応関係以外の情報を一切漏らさない性質である．一方，ポイント関数（ある特定の入力に対して 1 を出力し，それ以外の入力に対して 0 を出力する関数）を obfuscate することは可能であること，およびマルチビット出力のポイント関数を obfuscate することは可能であることが理論的に示されている [4].

Key Obfuscation

複数の公開鍵暗号を組み合わせた複合型暗号に関して，秘密鍵に関する obfuscation を実現する方式が提案されている．Re-encryption[10]（Alice の暗号鍵で暗号化したメッセージを入力して，同じメッセージを Bob の暗号鍵で暗号化した暗号文に変換して出力）と，Encrypted Signature[9]（メッセージを入力して，それに対して Alice のディジタル署名を作成し，その Alice の署名を Bob の公開鍵で暗号化した暗号文を出力する）については，アルゴリズム公開型で証明可能安全な方式が提案されている．ただし，[10] の方式は，復号のときに平文空間の中から平文候補を一つ一つ検証する処理を繰り返す．平文空間のサイズはセキュリティパラメータの多項式のオーダと定義されているので計算量理論的には効率的に実行可能であるが，実用性は少ない．

3.3 機能改変困難性に関する既存研究

3.3.1 コードを対象とする具体的な実現方法

ソフトウェアプロテクションシステム

機能改変困難性を付与するソフトウェアプロテクションシステム（ソフトウェアプログラム）は有償無償合わせて多数存在する．例えば，[5] に基づくと思われる Arxan Technologies, Inc. の製品，株式会社ハイパーテックの CrackProof，マルウェアに適用されるパッカーやクリプター等である．しかし，UPX のようなオープンソースのパッカーを除くと，仕様を公開しておらず，採用している実現方法は非公開（アルゴリズム秘密型）であるため強度は明らかではない．

自己インテグリティ検証

機能改変困難性を実現する基本的な方法として，自己インテグリティ検証（プログラムが実行時に自分のインテグリティを検証しながら処理を進める方法）が提案されている [1][5][11]．アルゴリズム公開型で，プログラム非実行型解析およびプログラム実行型解析の両方に耐性を持つ自己インテグリティ検証方式 [12] が提案されており，パラメータを変えることで人間および解析ツールに対する耐性を強めることが可能だが，その安全性は定性的に論証されており，数学的な安全性証明は与えられていない．

3.3.2 理論的な研究

機能改変困難性に関する理論的な研究は少ない．著者の知る範囲では，機能改変困難性の形式的な定義を試みた [3] が知られているのみである．

4 考察と提案

4.1 EOS と PS3 の事例に基づく考察

EOS の事例から，obfuscation はプログラム実行型解析に対して耐性を持つ必要がある．PS3 の事例から，ハイパーバイザによる仮想化は解析を困難にする効果があると考えられる．しかし，PS3 のハイパーバイザやローダの仕様は非公開（アルゴリズム秘密型）であったが製品寿命より短い期間内に秘密の署名生成鍵等が抽出されたので，PS3 の方法はアルゴリズム公開型には成り得ない方法だと思われる．いずれの事例でも，ファームウェアがプログラム実行型解析に対して十分な耐性を持たなかったため，ファームウェアに格納された秘密情報が守秘性を保つことができた期間は十分では無かった．

4.2 プラットフォーム検知方法の提案

以上の考察から，ファームウェアにプログラム実行型解析に対する耐性を持たせる方法を考える．ハイパーバイザを適用することが困難な組込み機器も多いので，ファームウェアを読み出された時でも有効な方法に着目する．ファームウェアを読み出されて解析ツール（例，IDA Pro）で解析されたとき，本当の組込み機器のプラットフォームで実行されているのか，解析ツールで解析されているのかを識別・検知することが可能で，後者の場合は正常動作しないようにする方法を考える．

4.2.1 Encrypted Signature に基づく時間計測プロトコル

組込み機器がネットワークに接続されていてネットワークと通信できるならば，ネットワークのノードとの間で暗号通信や認証プロトコルや通信にかかる時間の計測を実行して，プラットフォーム検知を行うことが可能である．

人間がプログラム実行型解析ツールを使ってブレークポイントを設定してラントレースを得ながら対話的に解析をする行為について，以下のような時間計測プロトコルでそれを検知できる．

前提：組込み機器のファームウェアは Encrypted Signature 方式 [9] を実装する．ネットワーク上の協力ノードも同じパラメータの方式を実装する．

1. ファームウェアは，時刻とファームウェアのシリアル ID を含むメッセージに対して Encrypted Signature を生成し協力ノードに送る．
2. 協力ノードは，受信した Encrypted Signature を復号検証する．検証成功の場合，受信時刻と前記シリアル ID を含むメッセージに対して

生成した Encrypted Signature と受信時刻のメッセージをファームウェアに送る．

3. ファームウェアは，受信した受信時刻のメッセージが許容時間差以内の値か否かを確認し，確認成功のときはその値とシリアル ID に対する Encrypted Signature を作成し，受信した Encrypted Signature との一致を確認する．確認失敗時は終了する．
4. 以上の処理を適切なインターバルで繰り返す．

提案プロトコルの応用として，協力ノードを組込み機器の CPU 内部に設けることが考えられる．この場合は，時刻の代わりに CPU カウンター値を用いる．CPU 内部に耐タンパーモジュールが必要だがネットワーク機能は不要なので，ネットワーク機能を持たない組込み機器にも適用可能である．

4.2.2 提案プロトコルの安全性と耐性

提案プロトコルでは，署名生成鍵に関する秘密情報守秘性を有する証明可能安全な複合型暗号（公開鍵暗号および署名）が実装されているため，ファームウェアを読み出されたとしても，暗号アルゴリズムを破ることおよび秘密の署名生成鍵を盗むことは困難である．従って，プロトコルに合格する偽のメッセージを攻撃者が生成することは困難であり，暗号を用いて外部との間で認証を行う従来の方法より強い耐性を持つ．時刻とシリアル ID が含まれるため，再送攻撃は防がれる．上記機能を実装したプログラムを機能改変困難性を付与する方法で耐タンパー化することにより，一致確認処理を回避するよう改変することも困難にできる．

5 まとめ

組込み機器の暗号ソフトウェア実装に対する攻撃について，実際の事例を踏まえて対策を考察し，攻撃者による解析を検知する従来より耐性が強い方法を提案した．

参考文献

[1] D. Aucsmith, “Tamper resistant software: an implementation,” Information Hiding 1996, pp.317-333, 1996.

[2] B.Barak, O.Goldreich, R.Impagliazzo, S.Rudich, A.Sahai, S.Vadhan and K.Yang, On the (im)possibility of obfuscating programs,” CRYPTO’01, pp.1-18, 2001.

[3] C. Basile, S. Di Carlo, T. Herlea, J. Nagra, and B. Wyseur, “Towards a formal model for software tamper resistance,” Second International Workshop on Remote Entrusting (ReTtust 2009), 16 pages, 2009.

[4] R.Canetti, R.R.Dadouk, “Obfuscating point functions with multibit output,” EUROCRYPT’08, pp.489–508, 2008.

[5] H. Chang and M.J.Atallah, “Protecting software code by guards,” DRM 2001, pp.160–175, 2002.

[6] C. Collberg, C. Thomborson and D. Low, “A taxonomy of obfuscating transformations,” Technical Report # 148, Department of Computer Science, The University of Auckland, July 1997.

[7] D. Sklyarov, “Forging Canon original decision data,” CONFidence 2.0, November 29th-30th, Prague, Czech Republic, 2010. (the slide is available at `http://www.elcomsoft.com/presentations/Forging_Canon_Original_Decision_Data.pdf` as of August 27, 2011.)

[8] fail0verflow “Console hacking 2010 PS3 epic fail,” 27th Chaos Communication Congress (27C3), December 29, 2010. (the slide is available via `http://psx-scene.com/forums/content/sony-s-ps3-security-epic-fail-videos-within-581/` as of August 27, 2011.)

[9] S. Hada, “Secure Obfuscation for Encrypted Signatures,” EUROCRYPT 2010, pp.92-112, 2010.

[10] S.Hohenberger, G.N.Rothblum, A.Shelat, V.Vaikuntanathan, “Securely Obfuscating Re-encryption,” TCC 2007, pp.233-252, 2007.

[11] B. Horne, L. Matheson, C. Sheehan, and R. Tarjan, “Dynamic self-checking techniques for improved tamper resistance,” DRM 2001, pp.141-159, 2002.

[12] 大石和臣，松本勉, “自己破壊的タンパー応答を発生する耐タンパーソフトウェア,” 電子情報通信学会論文誌. A, J94-A(3), pp.192-205, 2011.

[13] B. Wyseur, W. Michiels, P. Gorissen, and B. Preneel, “Cryptanalysis of White-Box DES Implementations with Arbitrary External Encodings,” Cryptology ePrint Archive, Report 2007/104, March 2007. (available at `http://eprint.iacr.org/2007/104` as of August 27, 2011.)