

ブロック検証によるセキュアなインターネット起動

八木 豊志樹[†] 須崎 有康[†] 飯島 賢吾[†]
中村 めぐみ^{††} 宗藤 誠治^{††}

インターネット上から起動する OS では、ファイルシステムの改ざんなどの脅威が存在する。我々は、インターネット起動する OS “HTTP-FUSE KNOPPIX” において、ダウンロードされるファイルの検証を行うことによって、動作させるファイルシステムの完全性を保証した。本稿では、手法の詳細について解説をし、そのパフォーマンス、システムの限界およびその解決法について述べる。

Secure Internet Boot Based on Block Inspection

TOSHIKI YAGI,[†] KUNIYASU SUZAKI,[†] KENGO IJIMA,[†]
MEGUMI NAKAMURA^{††} and SEIJI MUNETOH^{††}

For Internet boot OS, there are menaces of manipulation for file system. We guaranteed legitimacy of file system to operate by inspecting a downloaded block file of “HTTP-FUSE KNOPPIX” which boot from Internet. In this paper, I report the details of technique and performance.

1. はじめに

われわれは、インターネットにつながっていればどこからでも起動する OS の研究を行っており、その一環として、インターネット上にある HTTP サーバからブロックファイルを取得し、それをクライアント側で再構成して仮想ループバックデバイスとする “HTTP-FUSE cloop” および、HTTP-FUSE cloop を用いて起動する “HTTP-FUSE KNOPPIX”^{3),4)} の開発を行っている。

HTTP-FUSE cloop は、ブロックファイルを HTTP サーバからダウンロードし、それを仮想ブロックデバイスに割り当てる。そのため、ブロックファイルの改ざん、HTTP サーバの成りすまし等の脅威が存在する。このため、使用している HTTP-FUSE cloop の内容が正しいものであるかどうかを確認する機構がなければ、安全にインターネットからの起動をすることはできない。

われわれは、ブロックファイル検証を行い、完全性を保証する HTTP-FUSE cloop を開発した。これに

より、ブロックファイルの改ざん、HTTP サーバ成りすましの脅威を防ぐ事が可能となり、セキュアなインターネットからのブートが行えるようになった。

2. HTTP-FUSE KNOPPIX

われわれは、インターネット上のファイルを読み込んで起動する KNOPPIX である “HTTP-FUSE KNOPPIX” を開発した。これは HTTP サーバに起動したいファイルシステムイメージのブロックファイルを設置すれば、そのブロックファイルを読み込んで起動する。本章では HTTP-FUSE KNOPPIX とその機構について解説する。

2.1 KNOPPIX

KNOPPIX^{1),2)} とは、ドイツの Knopper 氏が開発した 1CD ブート Linux システムである。KNOPPIX では、CD から起動し、X が立ち上がるまでのハードウェア検出機能に優れている。また、KNOPPIX では cloop と呼ばれる圧縮ループバックデバイスを用いることにより、2GB 程度の容量のファイルを 700MB 程度の CD から直接起動することが可能となっている。

2.1.1 cloop

KNOPPIX に用いられている圧縮ループバックデバイス “cloop” では、透過的に圧縮されたループバックファイルをマウントすることが出来る。ここではその仕組みについて解説する。

[†] 独立行政法人 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology

^{††} 日本アイ・ビー・エム株式会社 東京基礎研究所
Tokyo Research Laboratory, IBM Japan Ltd.

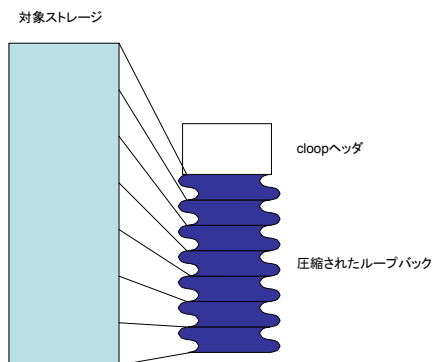


図 1 loop ファイルの構造

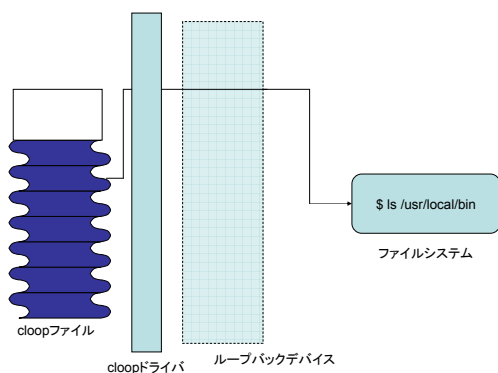


図 2 loop デバイスへのアクセス

loop では、対象となるブロックデバイス (HDD や CD など) のイメージを、一定サイズ (以下ブロックサイズ, 512byte 単位, 通常 64kByte) ごとに圧縮をかけ、その圧縮後の位置情報をヘッダに記録している。図 1 が loop ファイルのおおまかな構造である。圧縮されているブロックが詰めて配置されているため、一部分に変更を加えることが非常に難しくなっている。

loop デバイスドライバでは、図 2 のように、ループバックファイルにアクセスが生じると、ヘッダ情報を元に圧縮されたブロック単位で対応する圧縮イメージにアクセスを行い、展開したものをループバックデバイスとしてマウントを行う。

2.2 HTTP-FUSE loop

我々は、loop ループバックデバイスがブロックサイズ毎にアクセスされていることに着眼し、これをブロックファイルとして扱う HTTP-FUSE loop を開発した。

図 3 に HTTP-FUSE loop のデータ構造を示す。HTTP-FUSE loop では、圧縮されたブロックをブロックファイルという形でファイルとして保存し、ヘッダに相当する部分をインデックスファイルという形で保存したものを、HTTP サーバで公開することによ

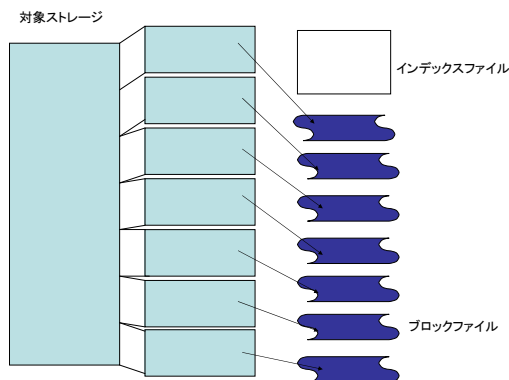


図 3 HTTP-FUSE loop のデータ構造

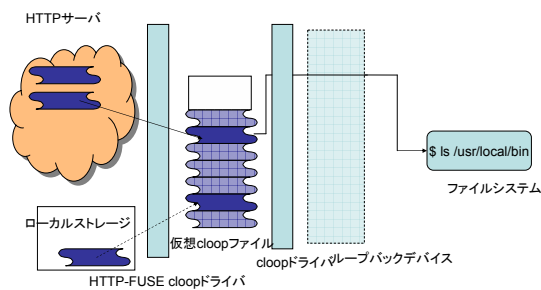


図 4 HTTP-FUSE loop の動作

て、HTTP-FUSE loop サーバとする。

HTTP-FUSE loop にてブロックファイル名として用いられているのは、ブロックの位置情報ではなく、ブロックの内容にて一意に MD5 ハッシュ値を用いている。基本的には Plan9 の Venti⁷⁾ と同じであるが、ブロックをファイルとすることにより、より柔軟に扱うことができる。

HTTP-FUSE loop クライアントでは、HTTP-FUSE loop サーバ上のインデックスファイルの URL を指定することにより、そのインデックス情報を元にクライアント上に仮想 loop ファイルを作成する。図 4 のように、仮想 loop ファイルにアクセスがあると、HTTP-FUSE loop クライアントは、対応するブロックファイルをオンデマンドでローカルまたは HTTP リクエストによって HTTP-FUSE loop サーバから取得し、仮想 loop ファイルへのアクセスを満足させる。

3. HTTP-FUSE loop に対する脅威

HTTP-FUSE loop では、インデックスファイルやブロックファイルは、HTTP によって取得される。そのため、HTTP サーバへの侵入、通信路でのデータ改ざん、HTTP サーバのなりすましなどによって、ブロックファイルを悪意のあるものに改ざんされる恐

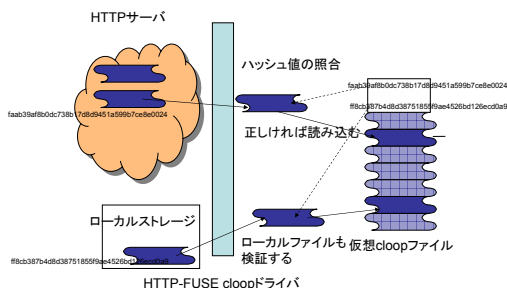


図 5 ブロックファイルの完全性検証

れがある。

このうち、ブロックファイルについてはハッシュ値の完全性を確認することで改ざんの脅威から守ることができた。しかしながら、現在 HTTP-FUSE cloop に用いているハッシュ値である MD5 には衝突を引き起こす方法が見つかったため、安全性に疑問が残る。

また、インデックスファイルについては改ざん検知法は確立されておらず、HTTP-FUSE cloop サーバを全面的に信頼するしかできなかった。

4. 手法の詳細

本章では、セキュアなインターネット起動を行うために、HTTP-FUSE cloop へおこなった拡張について述べる。

4.1 HTTP-FUSE cloop の完全性チェック

HTTP-FUSE cloop のブロックファイルは内容に対するハッシュ値がブロックファイル名となっているため、これをチェックすることによって、そのブロックの完全性を確認することが出来る。ブロックファイル名に用いていたハッシュ関数である MD5 には衝突が発見されているため、より衝突の起こりにくい SHA1 をハッシュ値として採用した。

ブロックファイルの完全性チェックは、図 5 のように、HTTP-FUSE cloop サーバからダウンロードしたものおよび、ローカルストレージに存在するキャッシュに対して行われる。ローカルストレージに対する完全性チェックは、HTTP-FUSE cloop ではローカルストレージに格納したキャッシュを次回起動時に再利用するため、ローカルキャッシュに対する改ざんを防ぐ為に行われる。

4.1.1 ブロックファイルへの攻撃

攻撃者が HTTP-FUSE cloop への攻撃を行う際に、ブロックファイルを改竄して悪意のあるコードを実行させるという方法が考えられる。本手法においては、悪意のあるコードを実行することを防ぐことが目的となっている。つまり、想定されていないブロックファ

イルを検知したらその時点で動作を止めてしまうことを目的としている。

HTTP-FUSE cloop のブロックファイルは、元となっているストレージを一定ブロックサイズごとに切り出して zlib による圧縮を行い、圧縮されたブロックファイルのハッシュ値をブロック識別子として用いている。また、インデックスファイルにはブロックファイルの識別子、ファイルサイズを格納している。

攻撃者が偽装されたブロックファイルを HTTP-FUSE cloop に読み込ませたときにクリアしなければならない点は以下のとおりとなる。

- ブロックファイルサイズとハッシュ値が同一でなければならない。サイズおよびハッシュ値が同一でなければ、HTTP-FUSE cloop ドライバのレイヤで不正なブロックであると検知される。
- zlib で展開を行うことができ、展開後のファイルサイズが同一でなければならない。zlib で展開を行うことができない、もしくは展開サイズがブロックサイズと同一でなければ、cloop デバイスドライバのレイヤで不正なブロックとして検知される。
- 展開されたブロックがファイルシステムの一部として矛盾しない必要がある。展開されたブロックが、ファイルシステムの一部として矛盾していれば、そのファイルシステムは破損していると扱われる。

この三点をクリアして、その上で悪意のあるデータを埋め込む必要がある。

一方向ハッシュ関数の特性上、ハッシュ値で表せる数を超える独立したデータが存在すればハッシュ値の衝突が生じる。

4.2 インデックスファイルの安全な配送

HTTP-FUSE cloop において、インデックスファイル名にはハッシュ値が用いられておらず、ブロックファイルの手法で改ざんされていないかを確認することはできない。また、たとえインデックスファイル自体への改ざんがなかったとしても、信頼できない「正しい」インデックスファイルへのアクセスが行われることに対しては無力である。このため、インデックスファイルが信頼の置ける発行者が発行したものであることを証明できなければならない。

そこで、我々は図 6 のようにインデックスファイルの位置とそのハッシュ値が書かれたファイルにデジタル署名をし、それを HTTPS サーバ上で公開する方式を取った。HTTP-FUSE cloop クライアントは、このファイルを読み込み、署名を検証することにより、

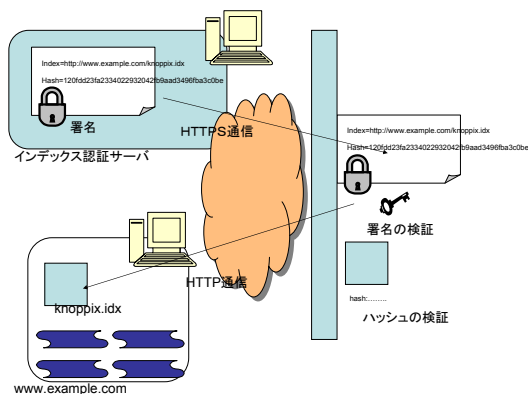


図 6 インデックスファイルの検証

ファイル発行者を確認する．正しいファイルであることが確認できたら，HTTP-FUSE cloop クライアントはその先にあるインデックスファイルを取得し，ハッシュ値を確認することでインデックスファイルの完全性を確認することができる．

4.3 完全性チェックに失敗した場合の対応

インデックスファイルやブロックファイルの完全性チェックに失敗した場合，基本的には HTTP-FUSE cloop が正しいものではないとみなされる．この場合，最低でも当該ブロックファイルへのアクセスは禁止されなければならない．ここで取られる方法としては，1 つはすでに HTTP-FUSE cloop が正しいものではないとみなし，以後 HTTP-FUSE cloop へのアクセス自体を禁止することが考えられる．また，インデックス情報が正しいと信じるならば，エラーのあったブロックへのアクセスのみを禁止する事も考えられる．ここで，仮に HTTP 転送もしくはキャッシュされたブロックに異常が起き (HTTP アクセスに失敗した等)，経路上でのエラーとみなし，HTTP-FUSE cloop クライアントが再度ブロックファイルへのアクセス要求を行うことも考えられる．この場合，ある程度の回数のみリトライを行い，全てのリトライにて失敗したら上記の方法を取ることが妥当であると考えられる．

4.4 本機能の実装

本機能は，既存の HTTP-FUSE cloop に，HTTPS 通信機能・署名検証機能・SHA1 ハッシュ生成機能を付与したものであり，OpenSSL⁵⁾ ライブラリの機能を利用している．

ブロックファイルの完全性チェック機構と完全性チェック失敗時の処理については，HTTP-FUSE cloop クライアントからできるだけ分離し，将来的にはユーザ定義した機能を外部から動的に読み込むことが出来るように計画を立てている．

5. 性能評価

ブロックファイルの完全性チェックが性能にどのような影響を及ぼすか，LAN 上および WAN 上を想定した場合にて性能評価を行った．WAN 上を想定した場合においては，FreeBSD の dummynet を利用したブリッジ型遅延発生器を用い，東京-沖縄間をの通信を想定した，片道 25msec の遅延を挿入した．性能評価に用いた環境は以下のとおりである．

- サーバ
 - CPU Intel(R) Xeon(TM) 2.80GHz
(FSB:400MHz, HT 無効)
 - メモリ 4GB
 - NIC Intel(R) PRO/1000 MT
 - OS Linux 2.6.15
 - HTTP サーバ Apache 2.0.55
- クライアント
 - CPU Intel(R) Pentium(R)4 2.40GHz
(FSB:800MHz, HT 無効)
 - メモリ 1GB
 - NIC Intel(R) PRO/1000 MT
 - OS Linux 2.6.15
- 遅延発生器
 - CPU Athlon64 3500+(Socket939)
 - メモリ 4GB
 - NIC Intel(R) PRO/1000 MT x 2
 - OS FreeBSD 6.0R(Kern-Developer)

5.1 ファイル読み込みテスト

HTTP-FUSE cloop を通した仮想 cloop ファイルの読み込みを行い，ブロックファイルの完全性チェックを行うことでパフォーマンスに与える影響を調査した．仮想 cloop ファイルのシーケンシャルアクセスを行うことによって，完全性チェックが与える影響を調査した．

調査に用いたブロックファイルは以下の通りである．
 対象ループバックファイルサイズ 5.3GB
 ブロックサイズ 256kbyte
 ブロックファイル数 10062
 平均圧縮ブロックファイルサイズ 190899byte

5.1.1 LAN 環境でのテスト

ブロックファイルの完全性チェックが与える影響を調査するため，LAN 環境にてスループット測定を行った．結果を図 7 に示す．

図 7 に示すように，ブロックファイルの完全性チェックを行うことにより，若干のスループットの劣化が確認されるが，その影響は小さいと考えられる．

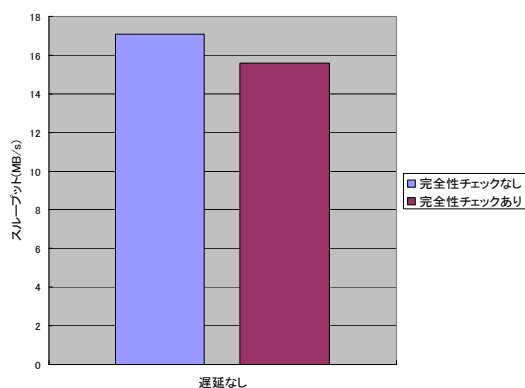


図 7 LAN 環境 (遅延なし) の場合のスループット

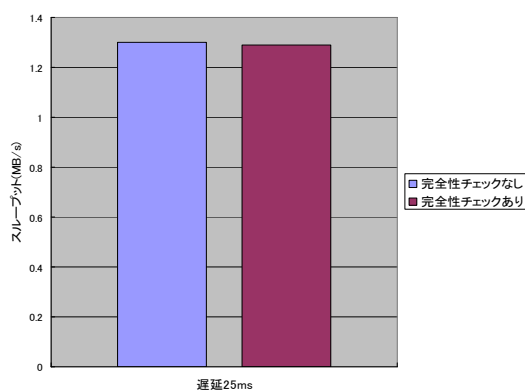


図 8 片道 25ms の遅延を発生させた場合のスループット

5.1.2 片道 25ms の遅延を発生させた環境でのテスト

次に、インターネットから用いられることを仮定し、片道 25ms の遅延を挿入した環境にて、同様の方法でスループット測定を行った。結果を図 8 に示す。

図 8 では、ネットワーク転送にかかるコストが増大したため、ブロックファイルの完全性チェックにかかるコストが無視できるほど小さくなった。これらの結果から、ブロックファイルの完全性チェックを行ってもコストとしてはさほど大きくないことがいえる。

6. 本手法の限界と解決法

本手法を用いることによって、HTTP-FUSE cloop サーバから取得するブロックが正しい配布者から配布されていることが保証された。しかしながら、HTTP-FUSE cloop クライアントを起動させるための最小システムが正しくなければ、それ以降の完全性は保証することが出来ない。システム全体の完全性を保証するためには、システム起動時からの完全性を確認する必要がある。

6.1 トラステッド・コンピューティング技術の適用

われわれは、システム起動時からの完全性を確認・検証する手段としてトラステッド・コンピューティング技術の適用を行った。

Trusted Computing Group(TCG)⁶⁾では、クライアント端末の完全性を検証するトラステッド・コンピューティングの規格を定めている。これは、対タンパ性を持つ TPM(Trusted Platform Module)を用いたプラットフォームの完全性測定および検証者への通知機能を用いて行われる。TPMには Platform Configuration Register(PCR)と呼ばれるレジスタを持ち、その内容は式 (1) に表される Extend と呼ばれる操作によってのみ更新される。

$$PCR(i) = SHA1(PCR(i-1) + Digest) \quad (1)$$

TCG によって定められているトラステッド・コンピューティングでは、CRTM (Core Root of Trust of Measurement) と呼ばれる BIOS 内にある書き換え不可領域が BIOS の完全性を測定し、Extend を行う。同様に BIOS が接続されている周辺機器の情報やブートローダの完全性を測定し、ブートローダがブートさせる OS の完全性を測定する。測定した結果は順次 Extend が行われるため、PCR にはブート時の情報が一意に記録される。ブート環境に変化が起きた場合には、Extend する値が変化するため、PCR の値に変化が生じる。この PCR の値を検証者に通知することによって、起動が正しく行われたか検証を行うことができる。

この手法を用いることにより、HTTP-FUSE cloop クライアント起動以前のシステムの完全性を検証することが出来るため、これにより全システムの完全性を検証することが可能となる。

トラステッド・コンピューティング技術を適用するため、我々はブートローダ GRUB にカーネルや初期 RAM イメージを計測する Trusted GRUB および、Trusted GRUB を用いて起動する KNOPPIX の開発を行っている。Trusted GRUB を KNOPPIX のブートローダとして用いることで、TCG に定めるトラステッド・コンピューティング技術を適用することができるようになった。

7. 今後の課題

インデックスファイルやブロックファイルの確認などの手法で解決できない問題がいくつか残っている。

7.1 一方方向ハッシュ関数の危殆化に対する対処

現状で安全であるとされているハッシュ関数であっても、攻撃法が発見されると危殆化する。実際に、現

在用いている SHA-1 では、ハッシュ値が同一である 2 つのメッセージを発見するための、誕生日攻撃よりも高速な手法が発見されている⁹⁾。完全に安全である一方ハッシュ関数は現在も発見されておらず、一方ハッシュ関数に対する攻撃法が発見されると、より安全なものに置き換えるということが続いている。HTTP-FUSE cloop においても、安全を確保するため利用できる一方ハッシュ関数の置き換えを容易に行うことができるのが望ましい。

この問題に関しては、インデックスファイルに利用する一方ハッシュ関数の情報を記述する方法を考えている。現在知られている一方ハッシュ関数から利用するものを選択する方法では、その全てが危殆化してしまうと使えなくなるため、プラグインプログラムとして自由に記述できる方法を検討している。

8. 関連研究

先行研究⁸⁾において、TCG の技術を用いた 1CD KNOPPIX の完全性測定を行っている。この研究では、起動に用いられるブートローダ、カーネル、初期 RAM イメージ、cloop の各ブロックのハッシュ値を計測する。cloop ファイルの完全性確認では、各ブロックのハッシュ値と、実際にアクセスされたブロックのハッシュ値との比較を行うことによって実現されている。ここでは、cloop の各ブロックのハッシュ値は、cloop ファイルの外に保存されている。

本研究では、HTTP-FUSE cloop の中にブロックの完全性チェックを入れることにより、ネットワーク上にある擬似ブロックデバイスの完全性を確認することができるようになった。

9. まとめ

HTTP-FUSE KNOPPIX は、インターネットに接続していれば HTTP 経由でどこからでもネットワーク起動出来る。しかし、既存の HTTP-FUSE KNOPPIX では、ファイル改ざんを検出することができず、悪意のあるコードを実行させられる危険性があった。

我々は、HTTP-FUSE cloop に対してブロックファイルの完全性検証およびインデックスファイルの配布者証明を行うことにより、HTTP-FUSE KNOPPIX をインターネット上からセキュアに動作させる方式を考案した。また、ブロックファイルの完全性チェックにかかるコストを計測し、それが無視できるほど小さいことを確認した。

HTTP-FUSE cloop の完全性検証が行うことが出来るようになり、それをベースにしているインターネッ

ト起動 OS, HTTP-FUSE KNOPPIX をセキュアに起動させることが出来るようになった。

また、TCG に定められたトラステッド・コンピューティングと組み合わせる事により、システム全体の完全性を保証することができ、HTTP-FUSE KNOPPIX のセキュア・シン・クライアントへの応用を行うことが可能になる。

参考文献

- 1) KNOPPIX
<http://www.knopper.net/knoppix/>
- 2) KNOPPIX 日本語版
<http://unit.aist.go.jp/itri/knoppix/>
- 3) HTTP-FUSE KNOPPIX
<http://unit.aist.go.jp/itri/knoppix/http-fuse/>
- 4) 須崎有康, 八木豊志樹, 飯島賢吾, 北川健司, 田代秀一, “ネットワークに対応した分割圧縮ループバックデバイス HTTP-FUSE-CLOOP とそれから起動する Linux,” インターネットコンファレンス 2005 (IC2005), 2005 年 10 月。
- 5) OpenSSL
<http://www.openssl.org/>
- 6) Trusted Computing Group,
<https://www.trustedcomputinggroup.org/>
- 7) Quinlan, S. and Dorward, D: Venti: a new approach to archival storage, the USENIX Conference on File and Storage Technologies, Monterey, CA, pp. 89-102 (2002)
- 8) 中村めぐみ, 宗藤誠治, 吉濱佐知子, “シンクライアントにおける完全性検証とその効率化”, Symposium on Cryptography and Information Security (SCIS2006), 2B1-1, 2006 年 1 月。
- 9) X. Wang, Y.L. Yin, and H. Yu.: Finding Collisions in the Full SHA-1, Advances in Cryptology – Crypto’05.