



IceWall SSO

Version 10.0

パスワード暗号化ライブラリ 開発者マニュアル

日本ヒューレット・パッカード株式会社
2010 年 8 月

Printed in Japan

HP Part No. B2873-97010

Rev.100819A

ご注意

本書に記載されました内容は、予告なしに変更することがあります。

本書は内容について細心の注意をもって作成いたしましたが、万一ご不審な点や誤り、記載もれなど、お気づきの点がございましたら当社までお知らせください。

当社は、本書に記載されている誤り、あるいは備品、パフォーマンス、または本書の使用に関連して発生する偶然または必然的な損傷について責任は負いかねます。

本書の著作権は Hewlett-Packard Development Company, L.P. に帰属します。本書の一部あるいは全部についての無断複製、転載を禁じます。

本製品パッケージとして提供した本マニュアル、CD-ROM 等の媒体は本製品用だけにお使いください。

IceWall はヒューレット・パッカー社社の商標です。

Adobe は Adobe Systems Incorporated の米国及びその他の国における登録商標または商標です。

Java は、米国 Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

Microsoft および Windows は、米国 Microsoft Corporation の、米国、日本およびその他の国における登録商標または商標です。

Oracle は、Oracle Corporation 及びその子会社、関連会社の米国及びその他の国における登録商標です。

目次

1 はじめに	1
2 パスワード暗号化ライブラリとは	2
3 開発の手順	3
3.1 暗号化・復号アルゴリズムの決定	3
3.2 コーディング・ビルド	3
3.3 テスト・デバッグ	3
4 ライブラリ仕様	5
IW_DB_Hash	6
IW_DB_PassCheck	7
5 注意事項	8
5.1 暗号化ライブラリの影響範囲	8
5.2 パフォーマンスの低下	8
5.3 システム運用中のライブラリ変更について	8
6 制限事項	9
7 参考	10
7.1 スケルトンソース (iwphash.c)	10
7.2 ヘッダーファイル (iwphash.h)	10
7.3 メイクファイル (HP-UX 64Bit 版 Itanium : Makefile)	11
7.4 メイクファイル (Linux 64Bit 版 : Makefile)	11

1 はじめに

本書には、認証 DB に保存されるユーザーのパスワードへの暗号化を行うライブラリの開発に必要な情報が記述しています。

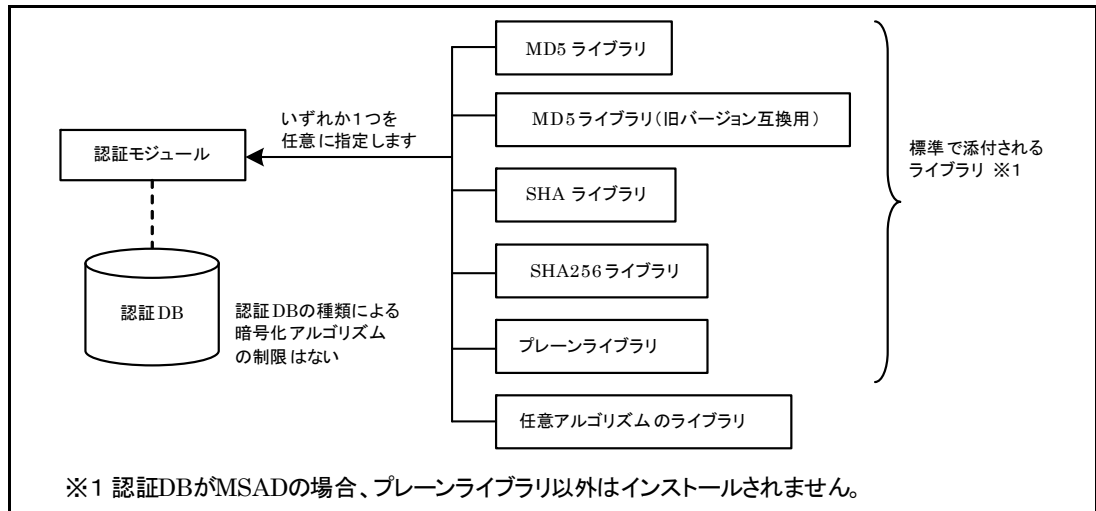
なお、本書では「暗号化」は「不可逆変換ハッシュアルゴリズム」を意味します。

ご注意

認証 DB として Microsoft Active Directory を使用する場合は、パスワード暗号化ライブラリの変更はできませんので、あらかじめご了承ください。

2 パスワード暗号化ライブラリとは

IceWall SSO は、パスワード暗号化アルゴリズムを任意に選択することができます。
暗号化アルゴリズムの選択は設定ファイルで指定します。



パスワード暗号化ライブラリは Shared Library として実装されています。標準暗号化ライブラリ以外のアルゴリズムを使用する場合は、任意のアルゴリズムを組み込んだ Shared Library を開発することとなります。

3 開発の手順

パスワード暗号化ライブラリを開発を行うための一般的な手順を説明します。

3.1 暗号化・復号アルゴリズムの決定

パスワードの暗号化および復号を行うアルゴリズムを決定してください。

3.2 コーディング・ビルド

決定したアルゴリズムに基づき、パスワード暗号化ライブラリのソースコードをコーディングします。コーディング後に **Shared Library** としてビルドします。

コーディングは、開発キットに含まれるスケルトンソースを利用してください。
ビルドは開発キットに含まれる **Makefile** で行います。

```
$ cd /opt/icewall-ssso/developkit/certd/PwdLib
$ make
```

3.3 テスト・デバッグ

テストプログラム等により、開発された **Shared Library** のテストおよびデバッグを行います。テストが終了した後に、ライブラリを認証モジュールへ組み込み、動作テストを行います。

インストールは以下の手順で行います。

- (1) 認証モジュールを停止します。

```
$ /opt/icewall-ssso/certd/bin/end-cert
```

- (2) 標準でインストールされるパスワード暗号化ライブラリのリンクを削除します。

```
$ rm /opt/icewall-ssso/lib/certd/libPH.sl
```

- (3) 開発したパスワード暗号化ライブラリへのリンクを設定します。

HP-UX 11i v3 (Itanium)、Linux の場合

```
$ ln -s /opt/icewall-ssso/developkit/certd/PwdLib/libiwPH.sl /
/opt/icewall-ssso/lib/certd/libPH.so
```

- (4) 認証モジュールを起動します。

```
$ /opt/icewall-ss0/certd/bin/start-cert
```

4 ライブラリ仕様

パスワード暗号化ライブラリでは、後述の定義に基づいた、2 つのインターフェイス関数を開発する必要があります。

- パスワードの暗号化機能 (IW_DB_Hash)
- 暗号化されたパスワードの比較機能 (IW_DB_PassCheck)

暗号化機能におけるパスワードの暗号化仕様は以下のようになります。

- (1) 暗号化されたデータの先頭にはアルゴリズム識別コードを「{ }」で括る。
(例) {MyHash} 暗号化されたパスワードデータ
- (2) 暗号化されたパスワードのデータ長は 128 バイト以内。(アルゴリズム識別コード部分を含む)

IW_DB_Hash

ユーザーがログインおよびパスワード変更時に入力したパスワードの暗号化を行うインターフェイス関数です。

書式 **int IW_DB_Hash(char* passwd, char* hashdata)**

引数 **passwd** : ユーザーが入力したパスワードが格納されたバッファのポインターです。このバッファサイズは 1 ～ 128 バイトとなります。

hashdata : 暗号化されたパスワードを格納するバッファのポインターです。このバッファサイズは 128 バイト固定となっています。

戻り値 以下の値を返します。
 PH_OK : 正常終了
 PH_NG : 異常終了

解説 このインターフェイス関数で基本的な処理の流れは以下のようになります。
 (1) 引数 **passwd** よりパスワードを取得
 (2) 取得したパスワードの暗号化
 (3) 暗号化したパスワードを引数 **hashdata** へ格納
 (暗号化したパスワードの終端にヌル文字 ('¥0') を追加してください。)

インターフェイス関数の引数として渡されるバッファは、認証モジュールで初期化された後に引数として渡されます。**passwd**、**hashdata** 共に最大 129 バイト (128+NULL 文字) となっているため、この領域への操作は十分に注意して行ってください。バッファサイズより大きいデータを格納した場合の動作は保証されません。

IW_DB_PassCheck

ユーザーがログインおよびパスワード変更時に入力したパスワードを暗号化したものと、認証 DB に保存されている暗号化されたパスワードとの比較を行うインターフェイス関数です。

書式 **int IW_DB_PassCheck(char* inpass, char* dbpass)**

引数 **inpass** : ユーザーが入力したパスワードが暗号化された状態で格納されているバッファのポインターです。

dbpass : 認証 DB に保存されたパスワードが暗号化された状態で格納されているバッファのポインターです。

戻り値 以下の値を返します。
 PH_OK : パスワードが一致する
 PH_NG : パスワードが一致しない

解説 このインターフェイス関数で基本的な処理の流れは以下となります。
 (1) 引数 **inpass** と引数 **dbpass** のパスワードを比較
 (2) 比較結果を戻り値とする

このインターフェイス関数の引数として渡されるバッファは、**inpass**、**dbpass** 共に最大 129 バイトで編集できません。(Read Only となります。) 編集を行った場合の動作は保証されません。

5 注意事項

パスワード暗号化ライブラリを開発する場合、以下の点に注意してください。

5.1 暗号化ライブラリの影響範囲

暗号化ライブラリ内で実行される処理は、認証モジュールの内部処理と密接な関係にあるため、ライブラリ内で体系的なエラーが発生した場合は、そのプロセスがダウンする可能性があります。開発されたライブラリのテストを十分に行った上で認証モジュールへ組み込んでください。

また、認証モジュールのプロセスに影響を与える処理は記述しないでください。

5.2 パフォーマンスの低下

暗号化ライブラリ内に処理を追加した場合、その処理を通常処理に追加で実行するためにパフォーマンスが低下します。このため、暗号化ライブラリ内では極力複雑な処理は行わないことを推奨します。

5.3 システム運用中のライブラリ変更について

ある暗号化ライブラリを使用してシステムを運用後に別の暗号化ライブラリに変更した場合、ライブラリ変更前に認証 DB に登録されているユーザーは暗号化されたパスワードが一致しなくなるため、ログインできなくなります。

6 制限事項

暗号化ライブラリを開発する場合には、以下の制限事項があることに注意してください。以下の制限内で開発されない場合には、プロセス自体が実行できない場合や動作が不安定になる場合があります。

- (1) リンクするライブラリは認証モジュールと同じく 64 ビットのライブラリとしてください。
- (2) 標準ライブラリ関数およびユーザー関数はスレッドセーフ版を使用してください。
認証モジュールは内部処理にスレッドを使用しています。暗号化ライブラリは、スレッド内のファンクションとして、認証モジュールよりコールされます。このため、ライブラリ内で使用するシステムコールは、スレッドセーフ版を使用してください。
- (3) バージョンの異なる標準ライブラリをアーカイブでリンクしないでください。
認証モジュールは、ダイナミックにライブラリを使用して動作するように、開発されています。
ただし、暗号化ライブラリにアーカイブライブラリをリンクしている場合には、暗号化ライブラリにリンクされているライブラリが利用されることになるため、バージョンの違いによる不整合を招く可能性があります。(OS への Patch を行っても解決しない、等のトラブルにもなる可能性があります)

注) 特に Oracle 等、生成されるライブラリ構成がインストール環境によって変わる (libcintlsh.sl の構成が変わる) ものを暗号化ライブラリで利用する場合にはリンクされているライブラリの状態について十分注意する必要があります。

7 参考

標準でインストールされるパスワード暗号化ライブラリ開発キットの内容です。
開発キットの構成は以下のディレクトリおよびファイルとなっています。

ディレクトリおよびファイル					内容
/opt/icewall-ss0	/developkit	/certd	/PwdLib	/iwphash.c	パスワード暗号化 ライブラリ開発キット
				/iwphash.h	
				/Makefile	

7.1 スケルトンソース (iwphash.c)

```
#include <stdio.h>
#include "iwphash.h"

int IW_DB_Hash( passwd, hashdata )
char *passwd;
char *hashdata;
{
    strcpy( hashdata, passwd );
    return( PH_OK );
}

int IW_DB_PassCheck( inpass, dbpass )
char *inpass;
char *dbpass;
{
    if( ( strcmp( inpass, dbpass ) ) != 0 ) {
        return( PH_NG );
    } else {
        return( PH_OK );
    }
}
```

7.2 ヘッダーファイル (iwphash.h)

```
/*-----*/
/* Password Hash Interface */
/*-----*/
#ifndef PHASH_H
#define PHASH_H

/*-----*/
/* Define */
```

```
/*-----*/
#define PH_OK      0          /* 成功          */
#define PH_NG      -1         /* 失敗          */

#endif /* #ifndef PHASH_H */
```

7.3 メイクファイル (HP-UX 64Bit 版 Itanium : Makefile)

```
CC          =   cc

CCFLAGS      =   -D_UNIX -D_HPUX_SOURCE -D_POSIX_C_SOURCE=199
506L -Aa +e -D_FILE_OFFSET_BITS=64 -z +DD64 +z

LIBS         =   -L./

MAKEFILE     =   Makefile

OBJS         =   iwphash.o

PROGRAM      =   iwPH

SRCS         =   iwphash.c

all:         $(PROGRAM)

$(PROGRAM):  $(OBJS)

             ld -b -z -o lib$(PROGRAM).sl $(OBJS)

.c.o:        $(CC) $(CCFLAGS) $(LIBS) -c $(SRCS)

clean:       rm -f $(OBJS) lib$(PROGRAM).sl core
```

7.4 メイクファイル (Linux 64Bit 版 : Makefile)

```
CCa          =   gcc

CCFLAGS      =   -m64 -fPIC -Dlinux -D_FILE_OFFSET_BITS=64 -
D_LARGEFILE_SOURCE -D_GNU_SOURCE

LIBS         =   -L./

MAKEFILE     =   Makefile
```

OBJS	=	iwphash.o
PROGRAM	=	iwPH
SRCS	=	iwphash.c
all:		\$(PROGRAM)
\$(PROGRAM):		\$(OBJS) gcc -shared -o lib\$(PROGRAM).sl \$(OBJS)
.c.o:		\$(CC) \$(CCFLAGS) \$(LIBS) -c \$(SRCS)
clean:		rm -f \$(OBJS) lib\$(PROGRAM).sl core