บทที่ 4

การพัฒนาระบบ

การพัฒนาระบบประกอบด้วยการพัฒนาโปรแกรม 3 ส่วนหลัก คือ

- ส่วนให้บริการรหัสผ่านแบบใช้ครั้งเดียว
- ส่วนขอใช้บริการรหัสผ่านแบบใช้ครั้งเดียว
- ส่วนบำรุงรักษาฐานข้อมูล

ซึ่งทั้ง 3 ส่วนถูกพัฒนาขึ้นบนระบบปฏิบัติการลินุกซ์ โดยอาศัยเครื่องมือ เอพีไอ (API : Application Program Interface) ต่างๆ เช่น

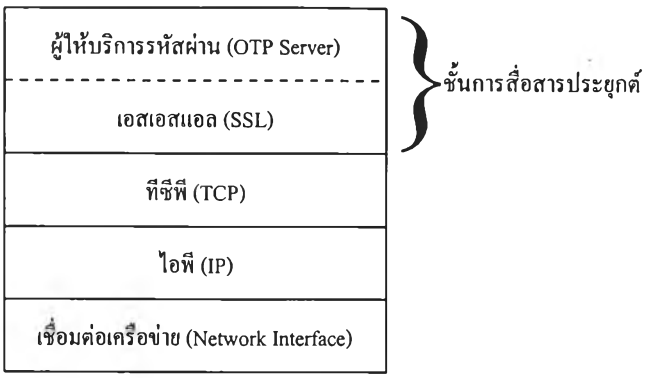
- ยูนิกซ์ซิสเต็มคอล (UNIX System call)
- เบอร์กเลย์ ซ็อกเก็ต (Berkeley Socket)
- เอสเอสแอลอีเอวาย (SSLeay)
- เอ็มเอสคิวแอล (mSQL)

4.1 หลักการพัฒนาโปรแกรม

4.1.1 การพัฒนาโปรแกรมส่วนให้บริการรหัสผ่านแบบใช้ครั้งเดียว

โปรแกรมในส่วนนี้ถูกพัฒนาขึ้นเพื่อให้บริการรหัสผ่านแบบใช้ครั้งเดียวผ่านเครือข่าย ทีซีพี/ไอพี โดยอาศัยเบอร์เลย์ซ็อกเก็ตไลบรารี ทำการเข้ารหัสข้อมูลที่รับส่งและพิสูจน์ตัวจริง โดยอาศัยไลบรารีของ เอสเอสแอลอีเอวาย

เนื่องจากมีการพิสูจน์ตัวจริงและการเข้ารหัสในโปรแกรมเป็นผลให้มีความต้องการใช้การติดต่อสื่อสารแบบเชื่อมถึงกัน (Connection Oriented) ซึ่งในที่นี้คือชั้นการสื่อสาร ทีซีพี และมีการซ้อนทับด้วยชั้นการสื่อสารแบบ เอสเอสแอล ส่วนโปรแกรมให้บริการรหัสผ่านแบบใช้ครั้งเดียวจะอยู่ที่ชั้นการสื่อสารประยุกต์เหนือชั้นการสื่อสาร เอสเอสแอล ดังแสดงในรูปที่ 4.1

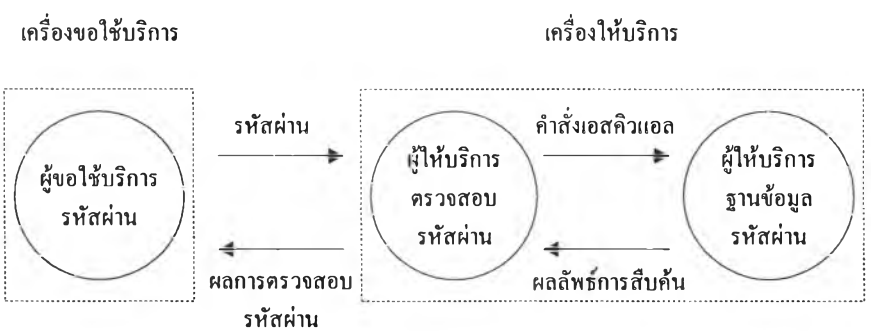


รูปที่ 4.1 ลำดับชั้นการสื่อสารของผู้ให้บริการรหัสผ่านแบบใช้ครั้งเดียว

ส่วนการจัดการข้อมูลต่างๆ ที่ใช้ในการบริการได้แก่ ข้อมูลบัญชีผู้ใช้ระบบรหัสผ่าน ข้อมูลรายละเอียดของเครื่องที่อยู่ในความดูแลของผู้ให้บริการและข้อมูลรหัสผ่าน ซึ่งข้อมูลเหล่านี้มีความสัมพันธ์ต่อกันจึงได้นำเอา เอ็มเอสคิวแอลไลบรารีมาใช้ในการสืบค้นฐานข้อมูลแบบเชิงสัมพันธ์ ซึ่งถูกให้บริการโดยผู้ให้บริการฐานข้อมูล (Database Server) ชื่อ "mSQL Engine" ทำให้สถาปัตยกรรมของส่วนให้บริการรหัสผ่านประกอบด้วยระบบรับ-ให้บริการ (Client-Server System) 2 ชุด คือ

- ระบบรับ-ให้บริการตรวจสอบรหัสผ่านแบบใช้ครั้งเดียว
- ระบบรับ-ให้บริการฐานข้อมูลระบบรหัสผ่าน

ดังแสดงในรูปที่ 4.2

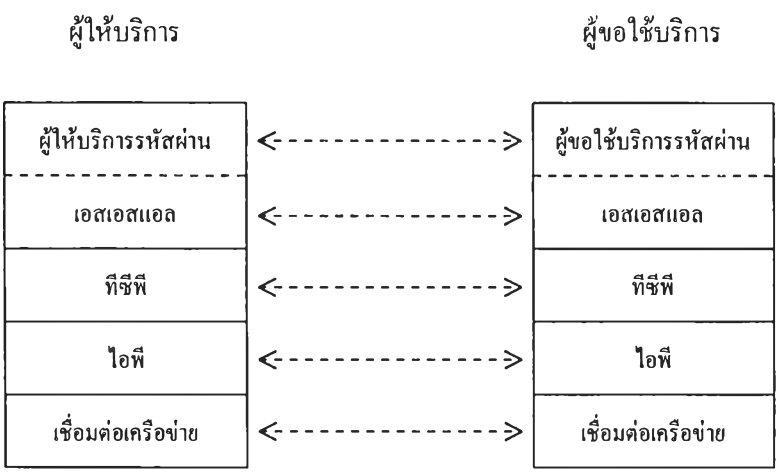


รูปที่ 4.2 สถาปัตยกรรมของส่วนให้บริการรหัสผ่านแบบใช้ครั้งเดียว

4.1.2 การพัฒนาโปรแกรมส่วนขอใช้บริการรหัสผ่านแบบใช้ครั้งเดียว

โปรแกรมนี้ถูกดัดแปลงมาจากโปรแกรมล็อกอิน (login) ของ เบอร์เลย์ ยูนิกซ์ (BSD UNIX) ซึ่งถูกใช้อยู่ในระบบปฏิบัติการลินุกซ์ โดยการแทรกโปรแกรมขอตรวจสอบรหัสผ่านแบบใช้ครั้งเดียวเข้าแทนการตรวจสอบรหัสผ่านแบบยูนิกซ์ เมื่อตรวจสอบพบว่าเขตข้อมูล “password” ของแฟ้มข้อมูล “/etc/passwd” มีค่าเป็น “*OTP*”

โปรแกรมนี้จะทำหน้าที่เป็นผู้ขอใช้บริการในระบบรับ-ให้บริการ ดังแสดงในรูปที่ 4.2 และมีลำดับชั้นการสื่อสาร ดังแสดงในรูปที่ 4.3



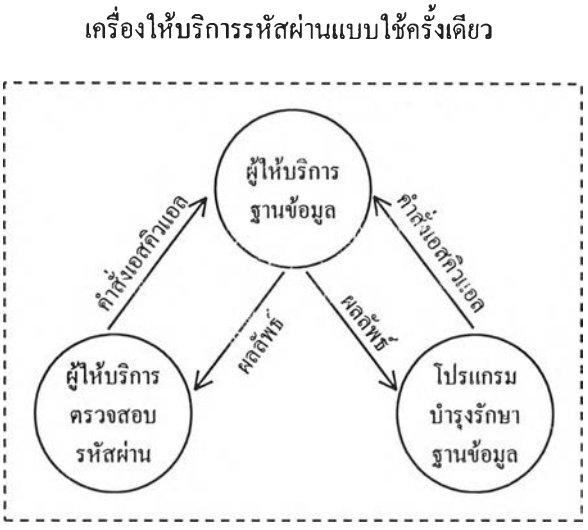
รูปที่ 4.3 ลำดับชั้นการสื่อสารและความสัมพันธ์ของทั้งผู้ให้บริการและผู้ขอใช้บริการ

4.1.3 การพัฒนาโปรแกรมบำรุงรักษาฐานข้อมูลรหัสผ่าน

โปรแกรมนี้ถูกพัฒนาขึ้นเพื่อจัดการกับระบบฐานข้อมูลรหัสผ่านที่ถูกใช้งานโดยผู้ให้บริการรหัสผ่าน และถูกให้บริการโดยผู้ให้บริการฐานข้อมูลชุดเดียวกัน ถึงแม้ว่าโปรแกรมจะทำงานอยู่บนเครื่องให้บริการตัวเดียวกัน แต่ยังคงมีสถาปัตยกรรมแบบระบบ

รับ-ให้บริการเช่นกัน ดังแสดงในรูปที่ 4.4 โดยมีขีดความสามารถในการจัดการฐานข้อมูล ดังนี้

- สามารถเพิ่ม แก้ไข และลบ รายการในตารางข้อมูลเครื่องขอใช้บริการที่อยู่ในความดูแลของผู้ให้บริการ “OTP_CLIENT”
- สามารถเพิ่ม แก้ไข และลบ รายการในตารางข้อมูลบัญชีผู้ใช้รหัสผ่านแบบใช้ครั้งเดียว “OTP_ACCOUNT”



รูปที่ 4.4 แสดงสถาปัตยกรรมของโปรแกรมทั้งหมดที่ทำงานอยู่ภายในเครื่องให้บริการรหัสผ่านแบบใช้ครั้งเดียว

4.2 เครื่องมือพัฒนาระบบ

เนื่องจากระบบถูกพัฒนาขึ้นโดยอาศัย เอพีไอไลบรารีดังกล่าวมาแล้ว จึงมีความจำเป็นต้องกล่าวถึงการใช้งาน เอพีไอ ดังต่อไปนี้

4.2.1 ยูนิกซ์ซิสเต็มคอล

โปรแกรมนี้ถูกพัฒนาขึ้นบนระบบปฏิบัติการลินุกซ์ ซึ่งเป็นระบบปฏิบัติการยูนิกซ์ประเภทหนึ่ง ดังนั้นทรัพยากรของเครื่องคอมพิวเตอร์จึงถูกควบคุมโดยระบบปฏิบัติการ

ทั้งหมด ทำให้มีความจำเป็นต้องอาศัยซิสเต็มคอล ซึ่งเป็นไลบรารีที่ใช้ในการขอใช้บริการและทรัพยากรของเครื่องคอมพิวเตอร์มาช่วนในการพัฒนา จึงขอกล่าวถึงซิสเต็มคอลที่ถูกใช้ในการพัฒนาระบบ ดังนี้

4.2.1.1 ซิสเต็มคอล fork

```
#include <sys/types.h>
#include <unistd.h>
pid_t fork (void);
```

ซิสเต็มคอลสั่งให้สร้างโพรเซสลูกโดยที่มีคุณสมบัติเหมือนโพรเซสพ่อ

ผลลัพธ์ที่ส่งกลับ

ในโพรเซสพ่อจะมีค่าเป็นหมายเลขของโพรเซสลูก

ในโพรเซสลูกจะมีค่าเป็น 0

ถ้ามีค่าเป็น –1 แสดงว่ามีความผิดพลาดเกิดขึ้น

4.2.1.2 ซิสเต็มคอล close

```
int close (int fd);
```

ซิสเต็มคอลที่ใช้ปิดไฟล์ที่ระบุหมายเลขไฟล์ (file descriptor) สามารถใช้ปิดซ็อกเก็ตได้เช่นกัน โดยระบุหมายเลขซ็อกเก็ต

ผลลัพธ์ที่ส่งกลับ

เป็นเลขจำนวนเต็ม มีค่าเป็น 0 เมื่อทำงานสำเร็จ และมีค่าเป็น –1 เมื่อเกิดข้อผิดพลาด

ค่าอาร์กิวเมนต์

fd หมายเลขของซ็อกเก็ตที่ต้องการปิด

4.2.2 เบอร์เลย์ซ็อกเก็ต

เป็นไลบรารีที่ใช้ในการพัฒนาโปรแกรมประยุกต์ที่ทำงานโดยอาศัยเครือข่ายทีซีพี/ไอพี ไลบรารีนี้ถูกรวมอยู่ในระบบปฏิบัติการลินุกซ์ ซึ่งในที่นี้จะขอกล่าวถึงซ็อกเก็ตซิสเต็มคอลเฉพาะที่ถูกใช้ในการจัดการกับชั้นสื่อสารทีซีพี

4.2.2.1 ซิสเต็มคอล socket

```
#include < sys/types.h >
#include < sys/socket.h >
int socket ( int family, int  type, int protocol );
```

ซิสเต็มคอลที่ใช้ในการสร้างซ็อกเก็ต (socket) ซึ่งใช้เป็นช่องทางติดต่อสื่อสารสำหรับโปรแกรมประยุกต์

ผลลัพธ์ที่ส่งกลับ

เลขจำนวนเต็มที่เป็นหมายเลขซ็อกเก็ต (socket descriptor) เพื่อใช้ในการอ้างถึงซ็อกเก็ตนี้ต่อไป

ค่าอาร์กิวเมนต์

family = AF_INET เนื่องจากใช้โพรโทคอลทีซีพี/ไอพี

type = SOCK_STEAM เนื่องจากใช้บริการแบบเชื่อมถึงกัน (Connection-Oriented) จึงกำหนดเป็น “TCP”

protocol = 0

4.2.2.2 ซิสเต็มคอล bind

```
#include < sys/types.h >
#include < sys/socket.h >
int bind ( int  sockfd, struct sockaddr * local_addr, int addrlen );
```

ซิสเต็มคอลที่ใช้ในการกำหนดแอดเดรสให้กับซ็อกเก็ต ซึ่งในที่นี้คือหมายเลขไอพีและหมายเลขพอร์ตสำหรับซ็อกเก็ตที่ระบุ

ผลลัพธ์ที่ส่งกลับ

เลขจำนวนเต็ม มีค่าเป็น 0 เมื่อทำงานสำเร็จ และมีค่าเป็น –1 เมื่อเกิดข้อผิดพลาด

ค่าอาร์กิวเมนต์

sockfd หมายเลขของซ็อกเก็ตที่ได้จากซิสเต็มคอล socket

local_addr ตัวชี้ (pointer) ของแอดเดรสของเครื่องให้บริการ (server) ที่เก็บอยู่ในโครงสร้างแบบ struct sockadd ในที่นี้มีค่าหมายเลขไอพีเป็น INADDR_ANY ซึ่งหมายถึง แอดเดรสใดๆ ของเครื่องผู้ให้บริการ และหมายเลขพอร์ต มีค่าเป็น 4444

addrlen ค่าความยาวของแอดเดรสที่ระบุในอาร์กิวเมนต์ local_addr

4.2.2.3 ซิสเต็มคอล listen

```
#include <sys/soocket.h>
int listen (int sockfd, int backlog);
```

ซิสเต็มคอลที่ใช้สั่งให้ซ็อกเก็ตบนเครื่องให้บริการทำงานแบบพาสซีฟ (passive) และกำหนดขนาดของคิว (queue) ที่เครื่องขอใช้บริการ (client) สามารถขอเข้าใช้ได้

ผลลัพธ์ส่งกลับ

เลขจำนวนเต็ม มีค่าเป็น 0 เมื่อทำงานสำเร็จ และมีค่าเป็น –1 เมื่อเกิดข้อผิดพลาด

ค่าอาร์กิวเมนต์

sockfd หมายเลขของซ็อกเก็ต

backlog ขนาดของคิวที่เครื่องขอให้บริการสามารถรอเข้าใช้ซ็อกเก็ตได้ ในที่นี้กำหนดให้เป็น 5

4.2.2.4 ซิสเต็มคอล accept

```
#include <sys/types.h>
#include <sys/socket.h>
int accept (int sockfd, struct sockadd *peer, int *addrlen);
```

ซิสเต็มคอลที่ใช้สั่งให้ซ็อกเก็ตแบบพาสซีพ (passive) ให้เข้าสู่ภาวะรอ (wait state) เพื่อรอการติดต่อเข้าใช้งาน เมื่อเครื่องขอใช้บริการติดต่อขอเข้าใช้ เครื่องให้บริการจะสร้างซ็อกเก็ตอันใหม่ที่มีค่าแอดเดรสเหมือนกับ sockfd และเปลี่ยนสภาวะเป็นภาวะเตรียมพร้อม (ready state) หลังจากนั้นการติดต่อระหว่างเครื่องให้บริการและเครื่องขอใช้บริการจะผ่านทางซ็อกเก็ตอันใหม่

ผลลัพธ์ที่ส่งกลับ

หมายเลขของซ็อกเก็ตที่สร้างขึ้นใหม่ ถ้าค่าเป็น –1 แสดงว่ามีข้อผิดพลาดเกิดขึ้น

ค่าอาร์กิวเมนต์

sockfd	หมายเลขซ็อกเก็ตที่รอการติดต่อจากผู้ขอใช้บริการ
peer	ตัวชี้แอดเดรสซ็อกเก็ตของเครื่องที่ติดต่อขอเข้าใช้บริการ
addrlen	ตัวชี้ค่าความยาวของแอดเดรสซ็อกเก็ต peer

4.2.2.5 ซิสเต็มคอล gethostname

```
#include <unistd.h>
int gethostname (char *name, int len);
```

ซิสเต็มคอลที่ใช้อ่านค่าชื่อเครื่องที่ซิสเต็มคอลที่ถูกเรียก

ผลลัพธ์ที่ส่งกลับ

เป็นเลขจำนวนเต็ม มีค่าเป็น 0 เมื่อทำงานสำเร็จ และมีค่าเป็น –1 เมื่อเกิดข้อผิดพลาด

ค่าอากิวเมนต์

name ตัวชี้ชื่อเครื่อง ซึ่งปกติค่าที่ได้จากระบบปฏิบัติการลินุกซ์จะมีค่าเป็นชื่อเครื่องตามด้วยชื่อโดเมน

len ความยาวของชื่อเครื่อง

4.2.2.6 ซิสเต็มคอล connect

```
#include <sys/types.h>
#include <sys/socket.h>
int connect (int sockfd, struct sockaddr * sockaddr, int addlen);
```

ซิสเต็มคอลที่ใช้ในฝั่งของผู้ขอใช้บริการ (client) เพื่อขอติดต่อไปยังผู้ให้บริการ (server) โดยปกติจะถูกใช้ในกรณีที่ซ็อกเก็ตถูกสร้างแบบ SOCK_STEAM

ผลลัพธ์ที่ส่งกลับ

เป็นเลขจำนวนเต็ม มีค่าเป็น 0 เมื่อทำงานสำเร็จ และมีค่าเป็น –1 เมื่อเกิดข้อผิดพลาด

ค่าอากิวเมนต์

sockfd หมายเลขซ็อกเก็ตของผู้ขอใช้บริการ

sockaddr ตัวชี้แอดเดรสของผู้ให้บริการที่ต้องการติดต่อ ซึ่งเก็บอยู่ในโครงสร้างแบบ struct sockaddr

addlen ความยาวของแอดเดรสที่ระบุใน sockaddr

4.2.3 เอสเอสแอลอีเอวาย ไลบรารี

เป็นไลบรารีที่ใช้ในการจัดการกับชั้นสื่อสาร เอสเอสแอล ซึ่งถูกใช้ในการสร้างช่องทางสื่อสารแบบเข้ารหัสและการพิสูจน์ตัวจริง ในการใช้งานไลบรารีจะต้องประกาศแฟ้มข้อมูลที่มีนามสกุล ".h" ดังต่อไปนี้

```
#include "rsa.h"
```

```
#include "crypto.h"
#include "x509.h"
#include "ssl.h"
#include "err.h"
```

จึงขอกล่าวถึงฟังก์ชันภาษาซีที่ถูกใช้ในการพัฒนาดังนี้

4.2.3.1 ฟังก์ชัน SSL_Load_error_strings

```
void  SSL_load_error_strings ( void );
```

ฟังก์ชันที่สั่งให้อ่านค่าของข้อความแจ้งข้อผิดพลาด (error string) ที่เก็บอยู่ในแฟ้ม “err.h” และ “ssl.h” มาใช้ในการแจ้งข้อผิดพลาด

4.2.3.2 ฟังก์ชัน SSL_CTX *SSL_CTX_new

```
SSL_CTX *SSL_CTX_new ( void );
```

เป็นฟังก์ชันที่ใช้ในการสร้าง เอสเอสแอลเซสชัน (SSL session) ซึ่งมีโครงสร้าง (structure) ข้อมูลแบบ SSL_CTX ที่ใช้ในการเก็บข้อมูลที่ใช้ในการทำงานของชั้น เอสเอสแอล เช่น สภาพแวดล้อม (environment) แคช (cashed) ของเอสเอสแอลเซสชัน (SSL session) และข้อมูลของใบรับรอง (certificate)

ผลลัพธ์ที่ส่งกลับ

ตัวชี้ของตัวแปรโครงสร้างแบบ SSL_CTX ที่ถูกสร้างขึ้น

ตัวแปรโครงสร้าง SSL_CTX มีองค์ประกอบดังนี้

```
SSL_CTX
    default cipher list
    session-id cache
```

certificate cache
default session-id timeout period
New session-id callback
Required session-id callback
session-id stats
Informational callback
Callback that is set, overrides the SSLeay
X509 certificate verification
The default Certificate/Private Key pair
Default read ahead mode.
Default verify mode and verify callback.
These are not used if the over ride callback
mentioned above is used.

4.2.3.3 ฟังก์ชัน SSL_CTX_use_RSAPrivateKey_file

int SSL_CTX_use_RSAPrivateKey_file(SSL_CTX *ctx, char *file, int type);

เป็นฟังก์ชันที่ใช้กำหนดค่าของคีย์ส่วนตัวสำหรับ เอสเอสแอลเซสชัน

ผลลัพธ์ที่ส่งกลับ

เลขจำนวนเต็ม มีค่าเป็น 0 เมื่อทำงานสำเร็จ และมีค่าเป็น –1 เมื่อเกิดข้อผิดพลาด

ค่าอาร์กิวเมนต์

ctx	ตัวชี้ เอสเอสแอลเซสชัน
file	ชื่อแฟ้มข้อมูลที่เก็บคีย์ส่วนตัวโดยระบุเป็นเส้นทางสมบูรณ์ (absolute path)
type	รูปแบบของการจัดเก็บแฟ้มข้อมูลคีย์ส่วนตัว ในที่นี้มีค่าเป็น SSL_FILETYPE_PEM

4.2.3.4 ฟังก์ชัน SSL_CTX_use_certificate_file

int SSL_CTX_use_certificate_file(SSL_CTX *ctx, char *file, int type);

เป็นฟังก์ชันที่ใช้กำหนดค่าของใบรับรอง (certificate) สำหรับ เอสเอสแอล เซสชัน

ผลลัพธ์ที่ส่งกลับ

เลขจำนวนเต็ม มีค่าเป็น 0 เมื่อทำงานสำเร็จ และมีค่าเป็น –1 เมื่อเกิดข้อผิดพลาด

ค่าอาร์กิวเมนต์

ctx ตัวชี้ เอสเอสแอลเซสชัน

file ชื่อแฟ้มข้อมูลที่เก็บใบรับรอง โดยระบุเป็นเส้นทางสมบูรณ์

type รูปแบบของการจัดเก็บแฟ้มข้อมูลใบรับรอง ในที่นี้มีค่าเป็น SSL_FILETYPE_PEM

4.2.3.5 ฟังก์ชัน SSL_load_verify_locations

```
SSL_load_verify_locations(ctx, CAfile, CApath);
#define SSL_load_verify_locations(ctx,CAfile,CApath) \
        X509_load_verify_locations((ctx)->cert,(CAfile),(CApath))
Int  X509_load_verify_locations ( CERTIFICATE_CTX *ctx,
        char *file_env, char *dir_env );
```

เป็นพรีโพรเซสเซอร์ (preprocessor) ที่ใช้กำหนดไดเรกทรอรีที่ใช้เก็บใบรับรองของผู้ออกใบรับรอง

ผลลัพธ์ที่ส่งกลับ

เลขจำนวนเต็ม มีค่าเป็น 0 เมื่อทำงานสำเร็จ และมีค่าเป็น –1 เมื่อเกิดข้อผิดพลาด

ค่าอาร์กิวเมนต์

ctx ตัวชี้ เอสเอสแอลเซสชัน

Cafile ชื่อของแฟ้มข้อมูลใบรับรองของผู้ออกใบรับรองโดยระบุเป็นเส้นทางสมบูรณ์

Capath เส้นทางสมบูรณ์ของไดเรกทรอรีที่เก็บข้อมูลของผู้ออกใบรับรอง

4.2.3.6 ฟังก์ชัน SSL_set_default_verify_paths

```
SSL_set_default_verify_paths(ctx);
#define SSL_set_default_verify_paths(ctx) \
        X509_set_default_verify_paths((ctx)->cert)
Int  X509_set_default_verify_paths ( CERTIFICATE_CTX *cts );
```

เป็นพรีโพรเซสเซอร์ที่กำหนดวิธีการพิสูจน์ใบรับรองให้เป็นไปตามค่าเบื้องต้น (default value) ที่กำหนดไว้ในแฟ้ม "/usr/local/ssl/lib/ssleay.cnf"

ผลลัพธ์ที่ส่งกลับ

เป็นเลขจำนวนเต็ม มีค่าเป็น 0 เมื่อทำงานสำเร็จ และมีค่าเป็น –1 เมื่อเกิดข้อผิดพลาด

ค่าอาร์กิวเมนต์

ctx ตัวชี้ เอสเอสแอลเซสชัน

4.2.3.7 ฟังก์ชัน SSL_CTX_set_cipher_list

```
Int  SSL_CTX_set_cipher_list ( SSL_CTX *ctx,char *str );
```

เป็นฟังก์ชันที่กำหนดรายการ (list) ของวิธีการเข้ารหัสที่เครื่องให้บริการหรือเครื่องขอใช้บริการสามารถทำงานได้ให้กับตัวแปรโครงสร้างแบบ SSL_CTX

ผลลัพธ์ที่ส่งกลับ

เลขจำนวนเต็ม มีค่าเป็น 0 เมื่อทำงานสำเร็จ และมีค่าเป็น –1 เมื่อเกิดข้อผิดพลาด

ค่าอาร์กิวเมนต์

ctx ตัวชี้ เอสเอสแอลเซสชัน

str รายการของวิธีการเข้ารหัส สามารถกำหนดได้หลายๆ วิธี โดยให้เครื่องหมาย colon คั่นระหว่างชื่อวิธีการเข้ารหัส

SSL_TXT_RC4_128_WITH_MD5 "RC4-MD5"

SSL_TXT_RC4_128_EXPORT40_WITH_MD5 "EXP-RC4-MD5"

SSL_TXT_IDEA_128_CBC_WITH_MD5 "CBC-IDEA-MD5"

SSL_TXT_DES_64_CBC_WITH_MD5 "CBC-DES-MD5"

SSL_TXT_DES_192_EDE3_CBC_WITH_MD5 "CBC3-DES-MD5"

SSL_TXT_DES_64_CFB64_WITH_MD5_1 "CFB-DES-M1"

SSL_TXT_DES_64_CFB64_WITH_NULL "CFB-DES-NULL"

4.2.3.8 ฟังก์ชัน SSL *SSL_new

SSL *SSL_new (SSL_CTX *ctx);

ฟังก์ชันที่ใช้สร้างช่องทางสื่อสาร เอสเอสแอล (SSL connection) ที่ใช้ในการรับส่งข้อมูลสำหรับชั้นสื่อสารประยุกต์

ผลลัพธ์ที่ส่งกลับ

ตัวชี้ช่องทางสื่อสาร เอสเอสแอล ซึ่งถูกอ้างถึงในการส่งข้อมูลต่อไป

ค่าอาร์กิวเมนต์

ctx ตัวชี้ เอสเอสแอลเซสชัน

4.2.3.9 ฟังก์ชัน SSL_clear

Void SSL_clear(SSL *ssl);

ฟังก์ชันที่ใช้ล้างบัฟเฟอร์ (buffer) ประจำช่องทางสื่อสาร เอสเอสแอลก่อนการใช้งาน

ค่าอาร์กิวเมนต์

ssl ตัวชี้ของช่องทางสื่อสารเอสเอสแอล

4.2.3.10ฟังก์ชัน SSL_set_fd

```
Int  SSL_set_fd ( SSL *ssl, int fd );
```

ฟังก์ชันที่ใช้เชื่อมช่องทางสื่อสาร เอสเอสแอลเข้ากับซ็อกเก็ต

ผลลัพธ์ที่ส่งกลับ

เป็นเลขจำนวนเต็ม มีค่าเป็น 0 เมื่อทำงานสำเร็จ และมีค่าเป็น –1 เมื่อเกิดข้อผิดพลาด

ค่าอาร์กิวเมนต์

ssl ตัวชี้ของช่องทางสื่อสาร เอสเอสแอล ในที่นี้คือค่าที่ได้จากซิสเต็มคอล SSL_new

fd หมายเลขซ็อกเก็ตที่ใช้ในการติดต่อจริง ในที่นี้คือ ค่าที่ได้จากซิสเต็มคอล accept

4.2.3.11ฟังก์ชัน SSL_accept

```
Int SSL_accept(SSL *ssl);
#define SSL_accept(a)             SSL2_accept(a)
int SSL2_accept ( SSL *s );
```

พรีโพรเซสเซอร์ที่ใช้สั่งให้รอการติดต่อขอใช้ช่องทางสื่อสาร เอสเอสแอล โดยการเข้าสู่ภาวะรอ และเมื่อมีการติดต่อขอใช้จะเข้าสู่ภาวะเตรียมพร้อม

ผลลัพธ์ที่ส่งกลับ

เลขจำนวนเต็ม มีค่าเป็น 0 เมื่อทำงานสำเร็จ และมีค่าเป็น –1 เมื่อเกิดข้อผิดพลาด

ค่าอาร์กิวเมนต์

ssl ตัวชี้ช่องทางสื่อสาร เอสเอสแอล

4.2.3.12 ฟังก์ชัน SSL_Connect

int SSL_connect(SSL *s);

ฟังก์ชันที่ใช้ในฝั่งผู้ขอใช้บริการ (client) ในการขอติดต่อกับชั้นสื่อสาร เอสเอสแอล ของฝั่งผู้ให้บริการ (server) ผ่านทางช่องทางสื่อสาร เอสเอสแอล

ผลลัพธ์ที่ส่งกลับ

เลขจำนวนเต็ม มีค่าเป็น 0 เมื่อทำงานสำเร็จ และมีค่าเป็น –1 เมื่อเกิดข้อผิดพลาด

ค่าอากิวเมนต์

s	ตัวชี้ช่องทางสื่อสาร เอสเอสแอล ที่ถูกเชื่อมกับชั้น ทีซีพี แล้ว

4.2.3.13 ฟังก์ชัน SSL_read

int SSL_read (SSL *s, char *buf, int len);

ฟังก์ชันที่ใช้อ่านข้อมูลตามจำนวนที่ระบุจากชั้นสื่อสารเอสเอสแอล มาเก็บไว้ในบัฟเฟอร์ ถ้ามีการเข้ารหัสข้อมูลที่ส่งผ่านชั้นเอสเอสแอล ฟังก์ชันจะทำการถอดรหัสเพื่อให้ได้ข้อมูลที่อ่านเข้าใจได้ (plain text) โดยอัตโนมัติ

ผลลัพธ์ที่ส่งกลับ

เลขจำนวนเต็ม มีค่าเป็น 0 เมื่อทำงานสำเร็จ และมีค่าเป็น –1 เมื่อเกิดข้อผิดพลาด

ค่าอากิวเมนต์

s	ตัวชี้ช่องทางสื่อสารเอสเอสแอลที่ต้องการอ่านข้อมูล
buf	ตัวชี้บัฟเฟอร์ที่ใช้เก็บข้อมูลที่อ่านได้จากชั้นสื่อสาร เอสเอสแอล
len	ขนาดของข้อมูลที่ต้องการอ่าน

4.2.3.14 ฟังก์ชัน SSL_write

```
int SSL_write (SSL *s, char * buf, int len);
```

ฟังก์ชันที่ใช้ส่งข้อมูลที่เก็บอยู่ในบัฟเฟอร์ตามจำนวนที่ระบุผ่านชั้นสื่อสารเอสเอสแอล ถ้ามีการกำหนดวิธีการเข้ารหัส ฟังก์ชันนี้จะทำการเข้ารหัสข้อมูลที่ส่งให้โดยอัตโนมัติ

ผลลัพธ์ที่ส่งกลับ

เลขจำนวนเต็ม มีค่าเป็น 0 เมื่อทำงานสำเร็จ และมีค่าเป็น –1 เมื่อเกิดข้อผิดพลาด

ค่าอากิวเมนต์

s	ตัวชี้ช่องทางสื่อสารเอสเอสแอลที่ต้องการส่งข้อมูล
buf	ตัวชี้บัฟเฟอร์ที่ใช้เก็บข้อมูลที่ต้องการส่งผ่านชั้นสื่อสารเอสเอสแอล
len	จำนวนข้อมุลที่ต้องการส่ง

4.2.3.15 ฟังก์ชัน SSL_get_verify_result

```
SSL_get_verify_result(s);
#define SSL_get_verify_result(s) ((s)->verify_result)
```

พรีโพรเซสเซอร์ที่ใช้อ้างถึงผลการพิสูจน์ตัวจริงของช่องทางสื่อสารเอสเอสแอล

4.2.3.16 ฟังก์ชัน *X509_cert_verify_error_string

```
char *X509_cert_verify_error_string ( int n );
```

ฟังก์ชันที่ใช้แปลงผลการพิสูจน์ตัวจริง จากเลขจำนวนเต็มมาเป็นข้อความ

ผลลัพธ์ที่ส่งกลับ

ตัวชี้ข้อความแสดงผลการพิสูจน์ตัวจริง

ค่าอาร์กิวเมนต์

n หมายเลขของข้อผิดพลาด

4.2.3.17 ฟังก์ชัน X509 *SSL_get_peer_certificate

X509 *SSL_get_peer_certificate(SSL *s);

ฟังก์ชันที่ใช้อ่านค่าของใบรับรอง (certificate) ของคู่สนทนา (peer) ตามช่องทางสื่อสารเอสเอสแอลที่ระบุ

ผลลัพธ์ที่ส่งกลับ

ตัวชี้ใบรับรองที่ถูกส่งมาในรูปแบบ X509

ค่าอาร์กิวเมนต์

s ตัวชี้ช่องทางสื่อสารเอสเอสแอล

4.2.3.18 ฟังก์ชัน PEM_write_X509

```
PEM_write_X509(stderr, peer);
#define PEM_write_X509(fp,x) \
PEM_ASN1_write((int (*)())i2d_X509,PEM_STRING_X509,fp, \
        (char *)x, NULL,NULL,0,NULL)
```

พรีโพรเซสเซอร์ที่ใช้แสดงค่าของใบรับรองในรูปแบบ X509 ออกทางอุปกรณ์แสดงผลข้อผิดพลาดมาตรฐาน (stdrerr)

4.2.3.19 ฟังก์ชัน X509_NAME *X509_get_subject_name

X509_NAME *X509_get_subject_name(X509 *a);

ฟังก์ชันที่ใช้อ่านชื่อผู้ถูกรับรอง (Subject Name) ในใบรับรอง

ผลลัพธ์ที่ส่งกลับ

ตัวชี้ชื่อผู้ถูกรับรอง

ค่าอาร์กิวเมนต์

a ตัวชี้ใบรับรอง

4.2.3.20 ฟังก์ชัน X509_NAME * X509_get_issuer_name

```
X509_NAME *  X509_get_issuer_name(X509 *a);
```

ฟังก์ชันที่ใช้อ่านชื่อผู้รับรอง (Issuer Name) ในใบรับรอง

ผลลัพธ์ที่ส่งกลับ

ตัวชี้ชื่อผู้รับรอง

ค่าอาร์กิวเมนต์

a ตัวชี้ใบรับรอง

4.2.3.21 ฟังก์ชัน char *X509_NAME_oneline

```
char *X509_NAME_oneline( X509_NAME *a );
```

ฟังก์ชันที่ใช้แปลงค่าที่เก็บอยู่ในโครงสร้างแบบ X509_NAME ให้อยู่ในรูปของข้อความ (string)

ผลลัพธ์ที่ส่งกลับ

ตัวชี้ของข้อความ

ค่าอาร์กิวเมนต์

a ตัวชี้ค่าที่เก็บในโครงสร้างแบบ X509_NAME

4.2.3.22 ฟังก์ชัน X509_free

```
void  X509_free( X509 *a );
```

ฟังก์ชันสั่งให้ยกเลิกการจองเนื้อที่ที่ใช้เก็บโครงสร้างแบบ X509

ค่าอาร์กิวเมนต์

a ตัวชี้ตัวแปรโครงสร้าง X509

4.2.3.23 ฟังก์ชัน *SSL_get_shared_ciphers

```
char *SSL_get_shared_ciphers( SSL *s, char *buf, int len );
```

ฟังก์ชันแสดงรายการวิธีการเข้ารหัสทั้งสองฝั่งสามารถทำงานร่วมกันได้

ค่าอาร์กิวเมนต์

s ตัวชี้ช่องทางสื่อสารเอสเอสแอล

buf ตัวชี้เนื้อที่ที่ใช้เก็บรายการวิธีการเข้ารหัส

len ขนาดของบัฟเฟอร์

4.2.3.24 ฟังก์ชัน *SSL_get_cipher

```
char *SSL_get_cipher( SSL *s );
```

ฟังก์ชันที่ใช้แสดงวิธีการเข้ารหัสที่ใช้งานอยู่ในช่องทางสื่อสารเอสเอสแอล ขณะที่กำลังทำงาน

ผลลัพธ์ที่ส่งกลับ

ตัวชี้ข้อความแสดงวิธีการเข้ารหัสที่เอสเอสแอลเซสชั่นใช้อยู่

ค่าอาร์กิวเมนต์

s ตัวชี้ช่องทางสื่อสารเอสเอสแอล

4.2.3.25 ฟังก์ชัน SSL_free

Void SSL_free (SSL *s);

ฟังก์ชันสั่งให้ยกเลิกการจองเนื้อที่หน่วยความจำของตัวแปรโครงสร้างแบบ SSL

ค่าอาร์กิวเมนต์

s ตัวชี้ตัวแปรโครงสร้างแบบ SSL ที่ใช้อ้างถึงช่องทางสื่อสารเอสเอสแอล

4.2.3.26 ฟังก์ชัน SSL_CTX_free

Void SSL_CTX_free (SSL_CTX * ctx);

ฟังก์ชันสั่งให้ยกเลิกการจองเนื้อที่หน่วยความจำของตัวแปรโครงสร้างแบบ SSL_CTX

ค่าอาร์กิวเมนต์

ctx ตัวชี้ตัวแปรโครงสร้างแบบ SSL_CTX ที่ใช้อ้างถึงเอสเอสแอลเซสชัน

4.2.4 เอ็มเอสคิวแอล ไลบรารี

เป็นฟังก์ชันภาษาซีที่ใช้ในการสืบค้นข้อมูลที่ถูกจัดการด้วยผู้ให้บริการฐานข้อมูล (Database Server) ชื่อ “mSQL Engine” ประกอบด้วยฟังก์ชันต่างๆ ที่ถูกใช้ในการพัฒนาดังนี้

4.2.4.1 ฟังก์ชัน msqlConnect

int msqlConnect(host)

char *host

ฟังก์ชันใช้ในการติดต่อกับผู้ให้บริการ เอ็มเอสคิวแอล (mSQL Engine)

ผลลัพธ์ที่ส่งกลับ

ถ้าสามารถขอใช้บริการจากผู้ให้บริการ เอ็มเอสคิวแอล ได้ จะให้ผลลัพธ์เป็นหมายเลขซ็อกเก็ต ซึ่งถูกใช้ในการอ้างถึงผู้ให้บริการ เอ็มเอสคิวแอล ต่อไป

ถ้าเกิดข้อผิดพลาด ผลลัพธ์จะมีค่าเป็น -1

ค่าอาร์กิวเมนต์

host	ชื่อเครื่องหรือหมายเลขไอพีของผู้ให้บริการ เอ็มเอสคิวแอล ถ้ามีค่าเป็น “Null” จะหมายถึง เป็นการติกต่อระหว่างผู้ให้บริการกับผู้ขอใช้บริการผ่านทางยูนิกซ์ซ็อกเก็ต (UNIX Socket) ซึ่งเหมาะสำหรับกรณีที่เครื่องผู้ให้บริการกับผู้ขอใช้ริการทำงานอยู่บนเครื่องเดียวกัน

4.2.4.2 ฟังก์ชัน msqlSelectDB

int msqlSelectDB(sock, dbName)

init sock;

char *dbname;

ฟังก์ชันใช้ในการเลือกฐานข้อมูล (database) ก่อนการสืบค้นข้อมูล เราสามารถใช้ฟังก์ชันนี้เพื่อการสลับไปมาระหว่างฐานข้อมูลที่อ้างถึงได้ โดยไม่จำเป็นต้องติดต่อกับผู้ให้บริการ เอ็มเอสคิวแอล ใหม่

ผลลัพธ์ที่ส่งกลับ

เป็นเลขจำนวนเต็ม มีค่าเป็น 0 เมื่อทำงานสำเร็จ และมีค่าเป็น –1 เมื่อเกิดข้อผิดพลาด

ค่าอาร์กิวเมนต์

sock	หมายเลขซ็อกเก็ตที่ได้จากฟังก์ชัน msqlConnect()
dbname	ชื่อของฐานข้อมูลที่ต้องการสืบค้น

4.2.4.3 ฟังก์ชัน msqlQuery

```
int msqlQuery(sock, query)
int    sock;
char *query;
```

ฟังก์ชันใช้ในการส่งคำสั่ง เอสคิวแอล (SQL command) ไปสืบค้นข้อมูลที่เครื่องให้บริการ เอ็มเอสคิวแอล ได้แก่คำสั่ง SELECT, DELETE และ UPDATE เป็นต้น

ผลลัพธ์จากการสืบค้นจะถูกเก็บอยู่ในบัฟเฟอร์ของ เอพีไอ ซึ่งโปรแกรมสามารถนำเอาผลลัพธ์มาใช้ได้ด้วยฟังก์ชัน msqlStoreResult() ดังนั้นก่อนการใช้ฟังก์ชัน msqlQuery ครั้งถัดไป จะต้องถ่ายข้อมูลออกจากบัฟเฟอร์ก่อน มิฉะนั้นข้อมูลที่อยู่ในบัฟเฟอร์จะถูกทับด้วยผลลัพธ์ของคำสั่งชุดใหม่

ผลลัพธ์ที่ส่งกลับ

เป็นเลขจำนวนเต็ม มีค่าเป็น 0 เมื่อทำงานสำเร็จ และมีค่าเป็น –1 เมื่อเกิดข้อผิดพลาด

ค่าอาร์กิวเมนต์

sock	หมายเลขซ็อกเก็ตที่ได้จากฟังก์ชัน msqlConnect()
query	คำสั่ง เอสคิวแอล ที่ต้องการส่งไปสืบค้น

4.2.4.4 ฟังก์ชัน msqlStoreResult

```
m_result  *msqlStoreResult()
```

ฟังก์ชันใช้ในการเก็บผลลัพธ์ของคำสั่ง เอสคิวแอล ในบัฟเฟอร์ของ เอพีไอ ลงในที่เก็บผลลัพธ์ที่มีโครงสร้างแบบ m_result

ผลลัพธ์ที่ส่งกลับ

ตัวชี้ที่เก็บผลลัพธ์ซึ่งมีโครงสร้างแบบ m_result

4.2.4.5 ฟังก์ชัน msqlFreeResult

```
Void msqlFreeResult(result)
    m_result  *result
```

ฟังก์ชันใช้ในการยกเลิกการจองเนื้อที่ที่ใช้เก็บผลลัพธ์ที่ได้จากฟังก์ชัน msqlStoreResult

ค่าอาร์กิวเมนต์

result ตัวชี้ที่เก็บผลลัพธ์โครงสร้างแบบ m_result

4.2.4.6 ฟังก์ชัน msqlFetchRow

```
m_row  msqlFetchRow(result)
m_result  *result
```

ฟังก์ชันใช้ในการดึงรายการที่ถูกเก็บอยู่ในที่เก็บผลลัพธ์โครงสร้างแบบ m_result มาเก็บไว้ในตัวแปรโครงสร้างแบบ m_row

ผลลัพธ์ที่ส่งกลับ

ตัวแปรแบบแถวลำดับ (array) ซึ่งแต่ละสมาชิกมีค่าเป็นตัวชี้แบบตัวอักษร (char *) ซึ่งชี้ไปยังแต่ละเขตข้อมูลที่ได้จากคำสั่ง เอสคิวแอล และมีค่าเป็น NULL เมื่อถึงจุดสุดท้ายของข้อมูล

ค่าอาร์กิวเมนต์

result ตัวชี้ที่เก็บผลลัพธ์โครงสร้างแบบ m_result

4.2.4.7 ฟังก์ชัน msqlNumRows

```
int msqlNumRows(result)
    m_result  *result;
```

ฟังก์ชันใช้แสดงจำนวนรายการ (record, row) ที่เก็บอยู่ในที่เก็บผลลัพธ์โครงสร้างแบบ m_result ที่ได้จากฟังก์ชัน msqlStoreResult()

ผลลัพธ์ที่ส่งกลับ

เลขจำนวนเต็มแสดงจำนวนรายการที่เก็บอยู่ที่เก็บผลลัพธ์ และมีค่าเป็น 0 เมื่อผลการสืบค้นไม่พบรายการที่ต้องการ

ค่าอาร์กิวเมนต์

result ตัวชี้ที่เก็บผลลัพธ์โครงสร้างแบบ m_result

4.2.4.8 ฟังก์ชัน msqlClose

```
int msqlClose(sock)
    int    sock;
```

ฟังก์ชันใช้สั่งปิดการติดต่อระหว่างผู้ขอใช้บริการกับผู้ให้บริการ เอ็มเอสคิวแอล ตามหมายเลขซ็อกเก็ตที่ระบุ

ผลลัพธ์ที่ส่งกลับ

เป็นเลขจำนวนเต็ม มีค่าเป็น 0 เมื่อทำงานสำเร็จ และมีค่าเป็น –1 เมื่อเกิดข้อผิดพลาด

ค่าอาร์กิวเมนต์

sock หมายเลขซ็อกเก็ตที่ได้จากฟังก์ชัน msqlConnect() ตอนที่ขอเปิดการติดต่อ

4.2.4.9 ฟังก์ชัน msqlDateToUnixTime

time_t msqlDateToUnixTime(msqldate)

ฟังก์ชันที่ใช้ในการแปลงข้อมูลวันที่ (date format) ของ เอ็มเอสคิวแอล มาเป็นค่าเวลาของยูนิกซ์ (UNIX time value)

ผลลัพธ์ที่ส่งกลับ

ค่าเวลาของยูนิกซ์ (time_t) มีหน่วยเป็นวินาที นับจากวันที่ 1 มกราคม ค.ศ.1970

ค่าอาร์กิวเมนต์

msqldatee	ข้อมูลวันที่ของ เอ็มเอสคิวแอล ซึ่งมีรูปแบบเป็น dd-mon-yyyy เช่น "12-JAN-1997"

4.2.4.10 ฟังก์ชัน msqlUnixTimeToDate

char * msqlUnixToDate(clock)
time_t clock

ฟังก์ชันใช้ในการแปลงค่าเวลาของยูนิกซ์ (UNIX time value) ไปเป็นข้อมูลวันที่ของ เอ็มเอสคิวแอล

ผลลัพธ์ที่ส่งกลับ

ตัวชี้ข้อมูลวันที่ของ เอ็มเอสคิวแอล

ค่าอาร์กิวเมนต์

clock	ค่าเวลาของยูนิกซ์

4.2.4.11 ฟังก์ชัน msqlDateOffset

```
char * msqlDateOffset(date, doff, moff, yoff)
    char * date;
    int    doff;
           moff;
           yoff;
```

ฟังก์ชันใช้คำนวณหาข้อมูลวันที่ของ เอ็มเอสคิวแอล จากวันที่ที่กำหนด และจำนวนวัน เดือน ปี ที่นับไปข้างหน้าหรือนับถอยหลังจากวันที่ที่กำหนด

ผลลัพธ์ที่ส่งกลับ

ข้อมูลวันที่ของ เอ็มเอสคิวแอล

ค่าอาร์กิวเมนต์

doff	จำนวนวันที่นับจากวันที่ที่กำหนด
moff	จำนวนเดือนที่นับจากวันที่ที่กำหนด
yoff	จำนวนปีที่นับจากวันที่ที่กำหนด