**2-way Certificate Authentication & Data Encryptyion with   
iOS SDK 5 and   
Internet Information Service (IIS) 7.5**

1. **บทนำ**

ในปัจจุบัน มีการใช้งานอุปกรณ์พกพาแพร่หลายขึ้น ไม่ว่าจะเป็นโทรศัพท์เคลื่อนที่หรืออุปกรณ์กลุ่ม Tablet แนวโน้มที่กลุ่มบริษัทองค์กรภาคธุรกิจจะนำอุปกรณ์เหล่านี้มาใช้ก็มีมากขึ้นเรื่อยๆ เพื่อพัฒนาการทำงาน การขายและการให้บริการต่อเทคโนโลยีใหม่ๆ เหล่านั้น สิ่งหนึ่งที่หลีกเลี่ยงไปไม่พ้นก็คือเรื่องของ Security ที่ฝ่าย IT และนักพัฒนาจะต้องคิดว่า ทำอย่างไรจึงจะป้องกันไม่ให้ผู้ที่ไม่ได้รับอนุญาติเข้าถึงข้อมูลและบริการต่างๆ เหล่านั้นได้

เรื่องของระบบรักษาความปลอดภัยที่เกี่ยวกับการนำอุปกรณ์พกพานั้นค่อนข้างกว้างและมีหัวข้อที่เกี่ยวข้องมากมาย และอุปกรณ์พกพาเองก็มีหลากหลาย platform จากผู้ผลิตหลายรายด้วย ในเอกสารฉบับนี้จึงเน้นไปที่แนวทางการพัฒนา Mobile (Native) Application บน iOS Platform เพื่อรับส่งข้อมูลกับ Internet Information Service (IIS) 7.5 บน Windows Server Platform โดยใช้เทคนิคการ encrypt ข้อมูลและ authenticate แบบ 2-way certificate

1. **System Architecture**

เมื่อนำอุปกรณ์พกพาเข้ามาใช้เป็นส่วนหนึ่งของระบบงาน วิศวกรระบบ (System Architect) จะต้องคำนึงถึงคุณสมบัติของตัวอุปกรณ์ ข้อจำกัดของอุปกรณ์ ข้อจำกัดของ development platform รวมไปถึงรูปแบบและวิธีการในการรับส่งข้อมูลกันระหว่างอุปกรณ์และ server ด้วย

* 1. **System Environment and Requirement**

คุณสมบัติของอุปกรณ์แบบพกพาทั่วๆ ไป คือ

* สามารถพกพาไปทำงานนอกสถานที่ได้ มีน้ำหนกัเบากว่า Laptop ซึ่งสะดวกกว่าในงานบางประเภท
* สามารถเปิดใช้งานได้อย่างรวดเร็ว ไม่ต้อง boot เครื่อง และทำงานได้นานกว่า Laptop
* จะ standby เครื่อง (ปิดโปรแกรม, lock หน้าจอ, ปิดหน้าจอ, อื่นๆ) ตอนไหนก็ได้

แต่อุปกรณ์แบบพกพาเหล่านั้นก็มีข้อจำกัดเช่นกัน คือ

* ในบางสถานที่อาจไม่มีสัญญาณ Application จึงจำเป็นต้อง save ข้อมูลที่ยังไม่ได้ส่งไปยัง server ได้
* Bandwidth ในการติดต่อกับ server อาจมีไม่สูงนัก การรับส่งข้อมูลจึงจำเป็นต้องจำกัดเท่าที่จำเป็นจริงๆ

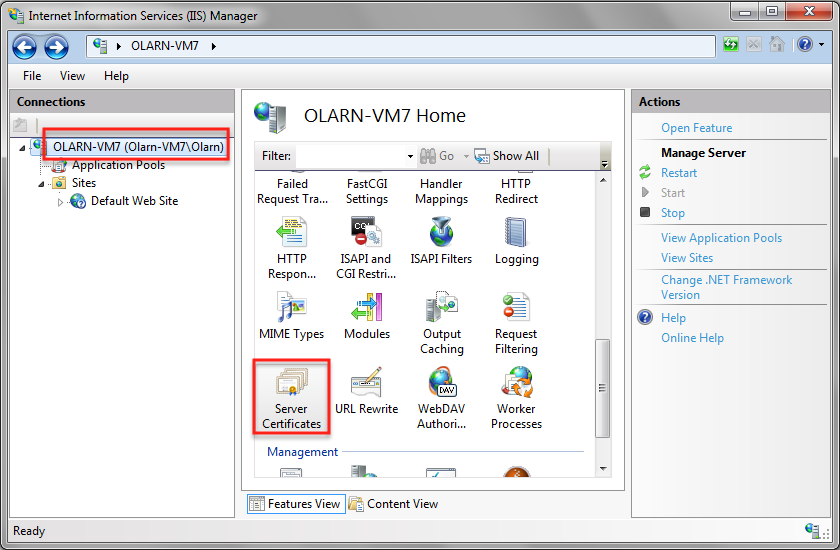
นอกจากนั้น ก็คือเรื่องของ Security ในระดับ application และข้อมูล คือ

* ทำอย่างไรจึงจะป้องกันไม่ให้บุคคลภายนอกสามารถอ่านเนื้อหาของข้อมูลที่ส่งผ่านไปใน internet
* ทำอย่างไรจึงจะป้องกันไม่ให้บุคคลภาบนอกเข้าถึงข้อมูลและบริการได้โดยไม่ได้รับอนุญาต
  1. **แนวทางในการ implement ระบบ**
     1. เมื่อผู้ใช้สามารถปิดอุปกรณ์พกพาตอนไหนก็ได้ และเมื่อเปิดมาก็จะต้องสามารถทำงานต่อได้ทันที ก็หมายความว่า application จะต้องสามารถ save state ต่างๆ ตอนไหนก็ได้ และเมื่อไม่มีสัญญาณในการเชื่อต่อกับ server ตัว application จะต้องสามารถ save state ไว้ได้ตลอดเวลาเช่นกัน ซึ่งแนวทางในปัจจุบันยังคงต้องพัฒนา **native application** ก่อน เพราะในปัจจุบัน web browser ยังทำงานด้านนี้ไม่สมบูรณ์นัก
     2. เมื่อไม่สามารถ connect server ได้ตลอดเวลา การรับส่งข้อมูลต้องเป็นแบบ disconnected แบบ web technology และเมื่อ bandwidth มีไม่สูงนัก การรับส่งข้อมูลจึงต้องกระชับที่สุด แต่เนื่องจาก iOS และ IIS (รวมถึง web server ทั่วไป) ไม่ได้ถูกพัฒนาโดย vendor รายเดียวกัน การรับส่งข้อมูลจึงต้องใช้รูปแบบที่เป็นมาตรฐานกลาง เทคโนโลยีที่เอกสารฉบับนี้เลือกใช้คือ **Restful Web Service** เนื่องจาก Restful นั้นเป็น SOA แบบหนึ่งที่รับส่งข้อมูลด้วย format แบบ JSON หรือ JavaScript Object Notation มีรูปแบบกระชับและง่ายกว่า Web Services ปรกติที่ส่งขอมูลด้วย XML และเมื่อใช้ JSON การ implement server ก็จะง่ายขึ้น เพราะการรับส่งจะใช้ HTTP Protocol ธรรมดาและรับส่งข้อมูลผ่านทาง method GET และ POST เท่านั้น
     3. การ encrypt ข้อมูล เมื่อใช้ Web Server การ encrypt ข้อมูลก็ใช้ SSL Encryption ด้วย protocol HTTPS ซึ่ง web server ในปัจจุบันก็รองรับอยู่แล้ว
     4. การ authentication จาก client ในเอกสารฉบับนี้จะใช้ Client Certificate โดยที่ Client ที่จะ connect มายัง server ได้ จะต้องมี certificate เพื่อยืนยันตัวตนของ Client เองว่าเป็น Client ตัวจริงที่สามารถใช้งาน service นั้นได้ ซึ่งบน IIS จะเรียก feature นี้ว่า **Client Certificate Mapping Authentication** ซึ่งเป็น feature หนึ่งของ IIS

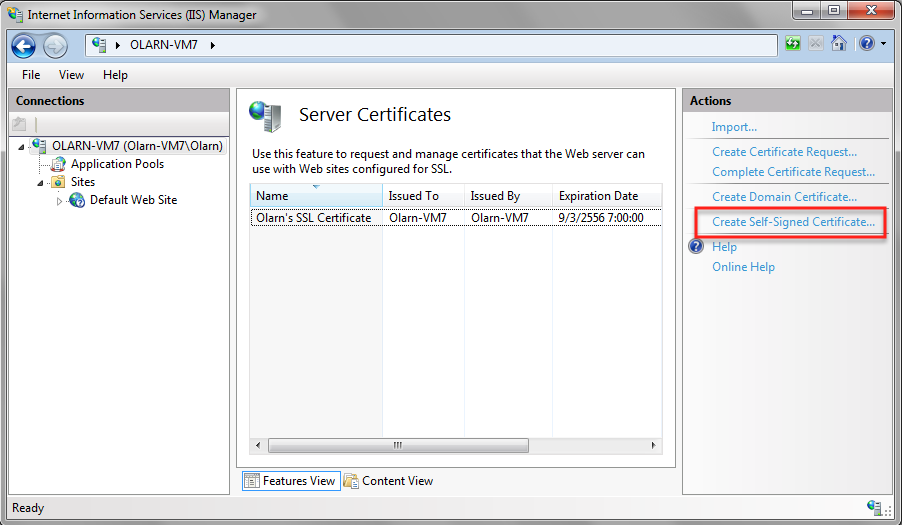
เนื่องจากเอกสารฉบับนี้เน้นไปที่การ implement 2-way certificate บน IIS เพราะฉะนั้นจึงไม่ได้เน้นการ implement Restful Web Service และ JSON แต่เนื่องจาก Restful เป็น Web Service ที่ทำงานผ่าน http/https protocol และ host application บน IIS การรับส่งข้อมูลจึงใช้การ return html ธรรมดา เพราะ JSON ก็เป็น text ที่ได้จาก web server เท่านั้น แต่เราจะ encrypt และ authenticate ด้วย certificate

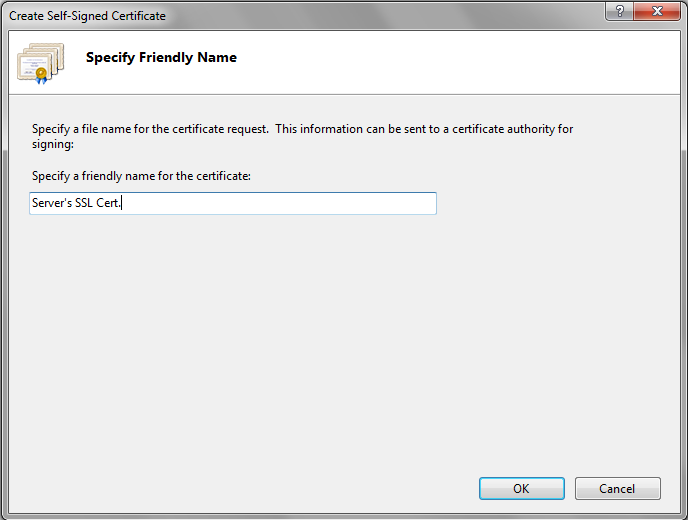
เพราะฉะนั้นภาพรวมของการ implement ในเอกสารฉบับนี้ คือ

1. ใช้ IIS เป็น Web Server / Web Application ในการ host Restful Web Service โดยที่ Windows เครื่องนั้นไม่ได้เป็นสมาชิกบน Domain
2. บน IIS จะใช้ Server Self-Signed Certificate เพื่อให้สามารถเชื่อมต่อแบบ https ได้
3. ทำการสร้าง user บน Windows
4. ที่ Client ทำการสร้าง Self-Signed Certificate จาก OS X (เพราะ iOS ต้องพัฒนาบนเครื่อง Mac)
5. นำ certificate ที่ได้ไป map บน IIS แบบ one-to-one กับ user บน Windows และ IIS
6. ตั้งค่า IIS ให้รับ connection เฉพาะ Client ที่มี Certificate เท่านั้น
7. สร้าง iOS Application เพื่อทดสอบ concept โดยทำการ Bundle certificate ใน application และส่ง certificate ไปเพื่อขอ authenticate และใช้งาน service เมื่อ web server ร้องขอ
8. **Understand Digital Certificate**
   1. Certificate
   2. Self-Signed Certificate
   3. Client Certificate
9. **Implementation**
   1. **Enable Server SSL** 
      1. สร้าง Self-Signed Certifitate บน IIS โดยเปิด Internet Information Service (IIS) Manager แล้วเลือก root node ใน Connections ด้านซ้ายมือ แล้ว double click ที่ Server Certificates

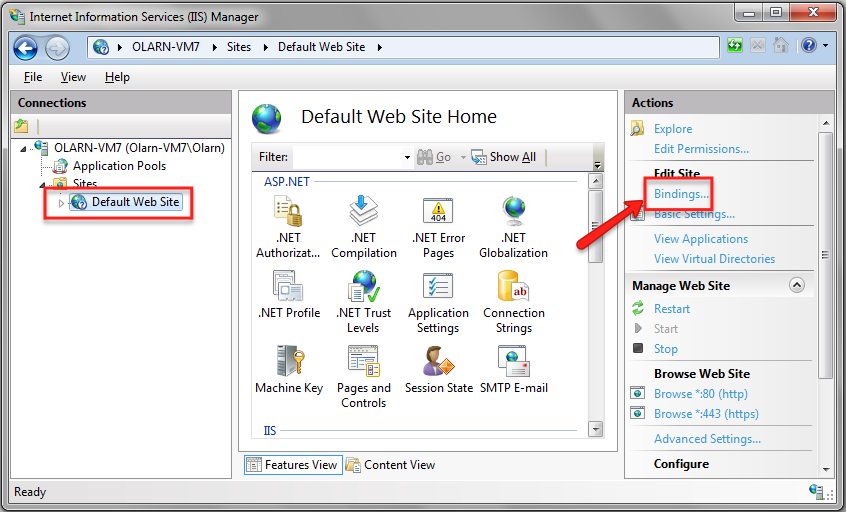


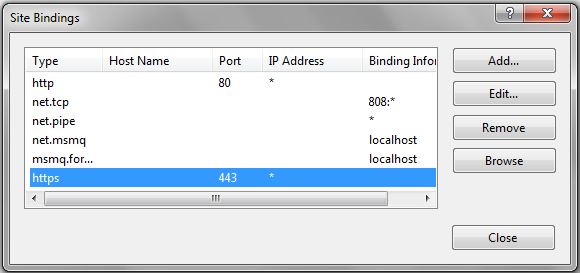
* + 1. ในหน้า Server Certificates ให้ click link “Create Self-Signed Certificate…” ที่อยู่ด้านขวามือ เมื่อมี dialog แสดงขึ้นมา ให้ตั้งชื่อ Certificate แล้ว click OK



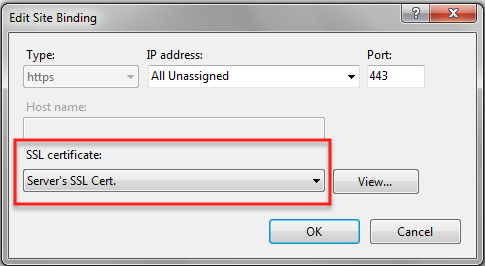


* + 1. เลือก Web Site หรือ Web Application ที่เราต้องการให้ link นั้นเป็น SSL (ตัวอย่างในเอกสารนี้จะเลือกที่ Default Web Site) จากนั้น click ที่ link “Bindings…” ใน Actions ที่อยู่ด้านขวามือ โปรแกรมจะแสดงหน้า Site Bindings ขึ้นมา

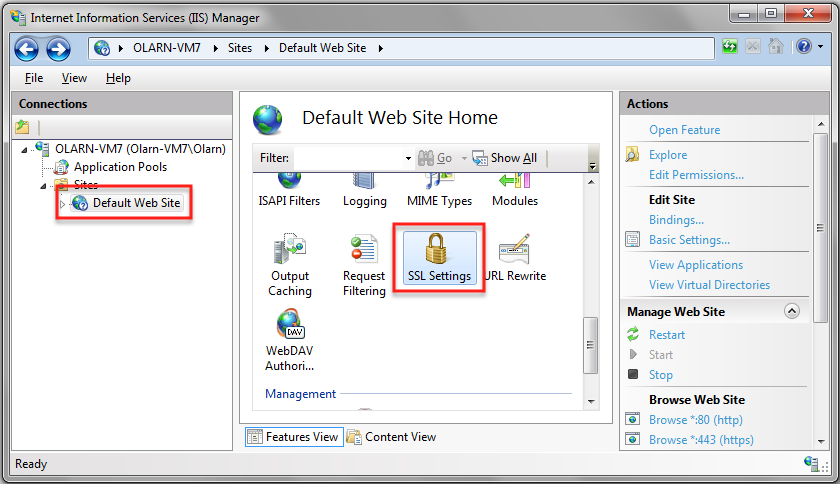


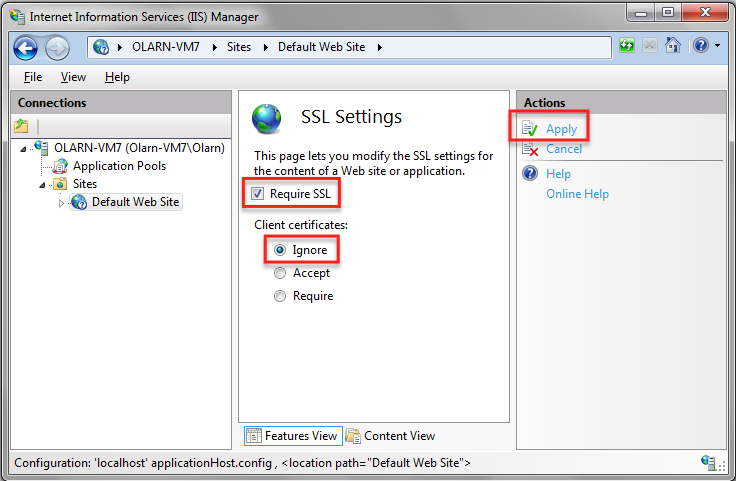


* + 1. ใน Site Bindings ให้เลือกรายการ “https” แล้ว click ปุ่ม edit (ในกรณีที่ไม่มี ให้ click ปุ่ม Add… แล้วเพิ่ม https เป็น port 443 เข้ามาก่อน) และเลือก certificate ที่สร้างขึ้นมาในข้อ 4.2 ในช่อง SSL certificate: แล้ว Click ปุ่ม “OK”

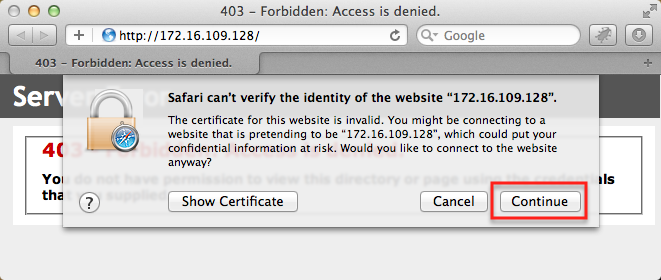
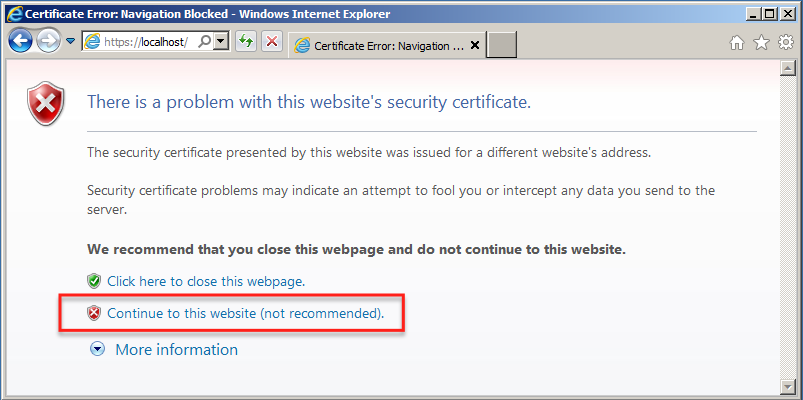


* + 1. ตั้งค่าให้ IIS บังคับให้ client ต้อง connect ผ่าน SSL เท่านั้น โดย double click ที่ SSL Settings จากนั้นในหน้า SSL Settings เลือก “Require SSL” และเลือก Client certificate เป็น Ignore แล้ว click “Apply” ใน Action ด้านขวามือ





* + 1. ทดสอบ browse webpage จากเครื่อง Server เองด้วย URL <https://localhost> (Browser จะเตือนว่าCertificate ไม่ได้รับการรองรับจาก CA ซึ่งอาจจะไม่ใช่ website ที่ต้องการเข้าถึงจริง นั่นเป็นเพราะเราไม่ได้เอา certificate ไป sign กับ CA) ให้ click “Continue to this website” กรณีที่เป็น IE หรือ click ปุ่ม “Continue” บน Safari ใน Mac

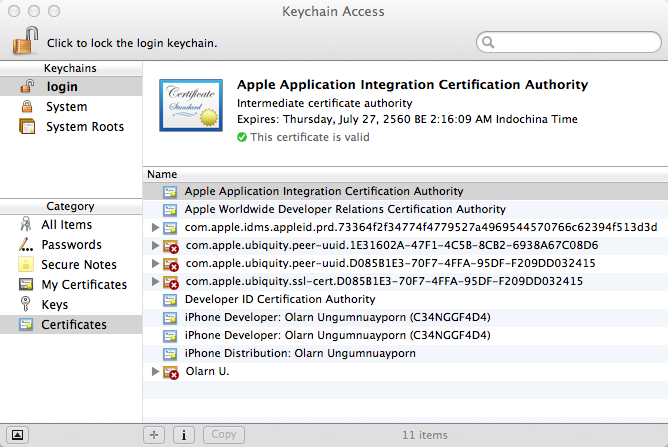


* 1. **Create Self-Signed Certificate on Mac OS X**

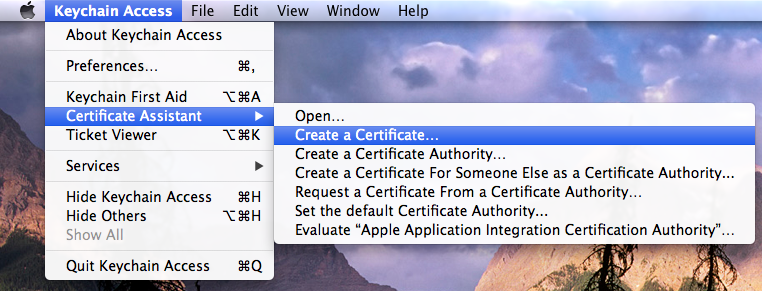
การกำหนดให้กรอก username และ password นั้นช่วยป้องกันให้ผู้ที่ไม่มี username สามารถเข้าถึง website ได้ แต่ถ้ารู้ว่า username และ password เป็นอะไร ใครๆ ก็สามารถเข้าถึงได้เช่นกัน ขั้นตอนต่อไปก็คือ กำหนดให้ผู้ที่จะเข้าถึง website ได้นั้น จะต้องมี certificate เพื่อยืนยันตัวตนที่ออกโดยฝั่ง client เอง และ certificate นั้นจะต้องถูก install ที่ server แล้วเท่านั้นจึงจะเข้าใช้งานได้ เมื่อผนวกกับการใช้ username และ password ก็จะยิ่งทำให้การเข้าถึงโดยบุคคลอื่นยากขึ้นไปอีก

ในขั้นตอนนี้จะเป็นการสร้าง Self-Signed Public Key Certificate ที่ฝั่ง client โดยใช้โปรแกรที่ชื่อว่า Keychain Access ซึ่งเป็นเครื่องมือที่มีอยู่แล้วใน Mac OS X ซึ่ง public key ที่ได้นี้จะถูกนำไปติดตั้งบน IIS

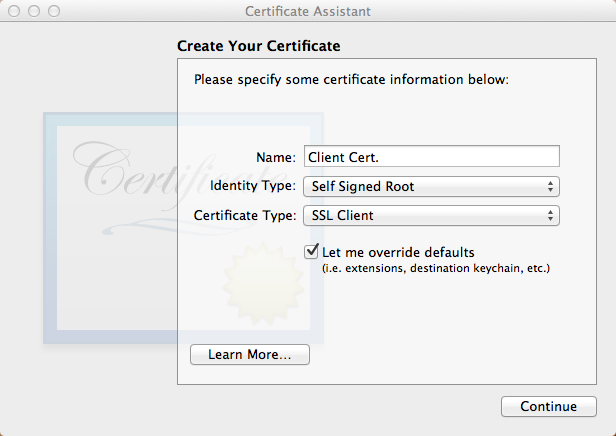
* + 1. เปิด Keychain Access จาก /Applications/Utilities/Keychain Access.app



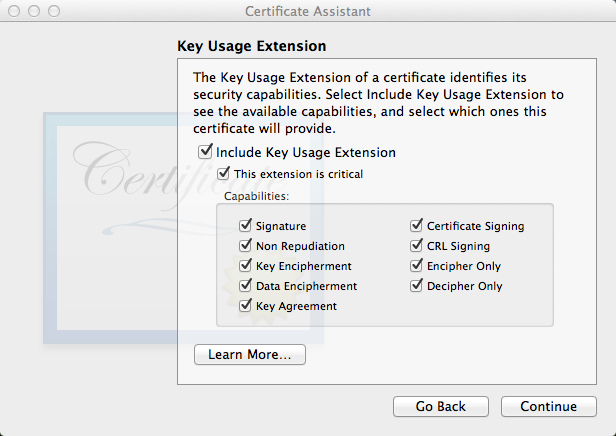
* + 1. สร้าง certificate ใหม่ จากเมนู Keychain Access > Certificate Assistant > Create Certificate …



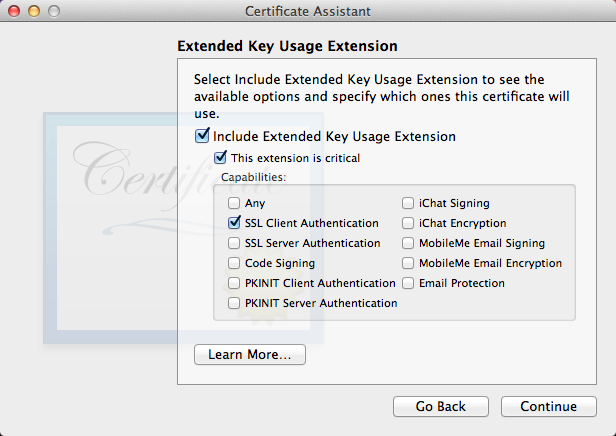
* + 1. ตั้งชื่อ Certificate, กำหนด Identity Type เป็น Self Signed Root, กำหนด Certificate Type เป็น SSL Client และเลือก “Let me override defaults” จากนั้น click ปุ่ม Continue



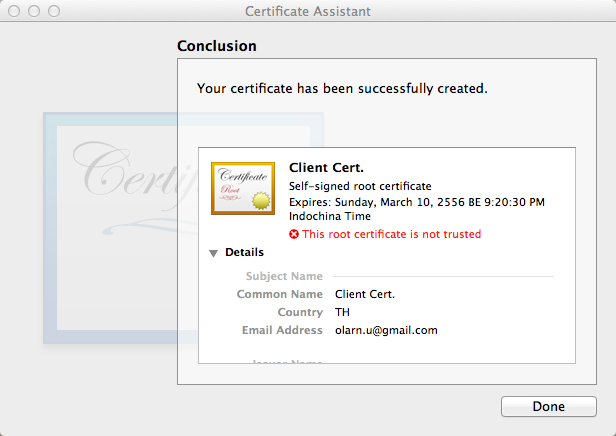
* + 1. ในหน้า Certificate Information ไม่ต้องแก้อะไร, click ปุ่ม Continue เพื่อทำรายการต่อ
    2. ในหน้าถัดมา เป็นข้อมูลเจ้าของ certificate ให้กรอกข้อมูลต่างๆ เพื่อใช้เป็นข้อมูลอ้างอิงในฝั่ง server จากนั้น click ปุ่ม Continue เพื่อทำรายการต่อ
    3. ในหน้า Key Pair Information ไม่ต้องแก้อะไร จากนั้น click ปุ่ม Continue เพื่อทำรายการต่อ
    4. ในหน้า Key Usage Extension ให้ click เลือกทั้งหมด จากนั้น click ปุ่ม Continue เพื่อทำรายการต่อ



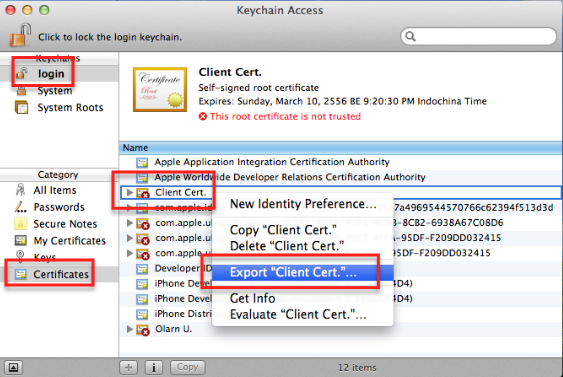
* + 1. ในหน้า Extended Key Extension ให้เลือก SSL Client Authentication (เลือกอยู่แล้วโดย default) click ปุ่ม Continue เพื่อทำรายการต่อ



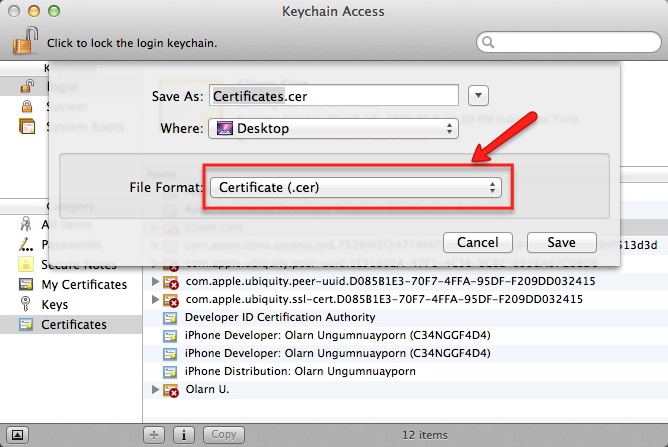
* + 1. ในหน้า Basic Constraints Extension ไม่ต้องแก้อะไร click ปุ่ม Continue เพื่อทำรายการต่อ
    2. ในหน้า Subject Alternate Name Extension ไม่ต้องแก้อะไร click ปุ่ม Continue เพื่อทำรายการต่อ
    3. ในหน้า Specify a Location For The Certificate จะเป็นส่วนที่กำหนดว่าให้เก็บ Certificate ที่ถูกสร้างขึ้นมานั้นไว้ที่ไหน ให้เลือก “Login” (เป็นค่าโดย default) แล้ว click ปุ่ม Create เราจะได้ Certificate ดังรูป จากนั้น click ปุ่ม Done



* + 1. ทำการ Export Public Key เพื่อนำ key นี้ไปใช้ที่ server โดย click ขวาที่ Certificate ที่เพิ่งสร้างขึ้นมา (สังเกตุจากชื่อที่เราตั้งในข้อ 4.3.3) แล้วเลือก Export “xxx” (xxx เป็นชื่อ Cert. ที่ตั้งขึ้นมา)



* + 1. ในหน้า Save Certificate ให้เปลี่ยน File Format ไปเป็น “Certificate (.cer)” ซึ่ง file ที่ได้จะเป็น public key แล้วที่ใช้บน Server



* + 1. Copy file ที่ได้ไปไว้บน IIS เพื่อนำไป import และ map เข้ากับ user บน Windows

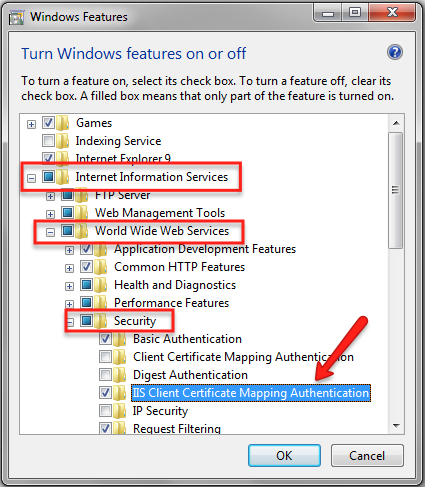
Certificate ที่ได้ในขั้นตอนนี้ จะเป็น Certificate แบบ Public Key ซึ่งเอาไว้ถอดรหัสที่ encrypt เอาไว้โดย Private Key อีกทีหนึ่งจากฝั่ง client เราสามารถเอา Public Key นี้ไปวางไว้ที่ server ใดก็ได้ที่ต้องการ authenticate ด้วย client certificate แต่จะเก็บ Private Key เอาไว้เพื่อสร้างรหัส ซึ่งจะเป็นขั้นตอนต่อไป

* 1. **Enable IIS Client Certificate**

ใน IIS6 นั้นจะมี user interface สำหรับตั้งค่า Client Certificate แต่ใน IIS7 นั้น user interface ถูกถอดออกไป เราสามารถติดตั้ง add-on ที่ 3rd พัฒนาขึ้นมาเพิ่มได้ แต่ add-on นั้นจะ set ค่าได้เฉพาะการ mapping แบบ one-to-one เท่านั้น ไม่สามารถตั้งค่าการ mapping แบบ many-to-one ได้ และในเอกสารฉบับนี้จะพูดถึงเฉพาะการ mapping แบบ one-to-one เท่านั้น

* + 1. เพิ่ม Feature Client Certificate บน IIS (จาก Add/Remove Program หรือ Server Manager บน Windows 2008 Server) โดยไปที่ “Internet Information Services > World Wide Web Services > Security” จากนั้น เลือก IIS Client Certificate Mapping Authentication แล้ว click “OK”

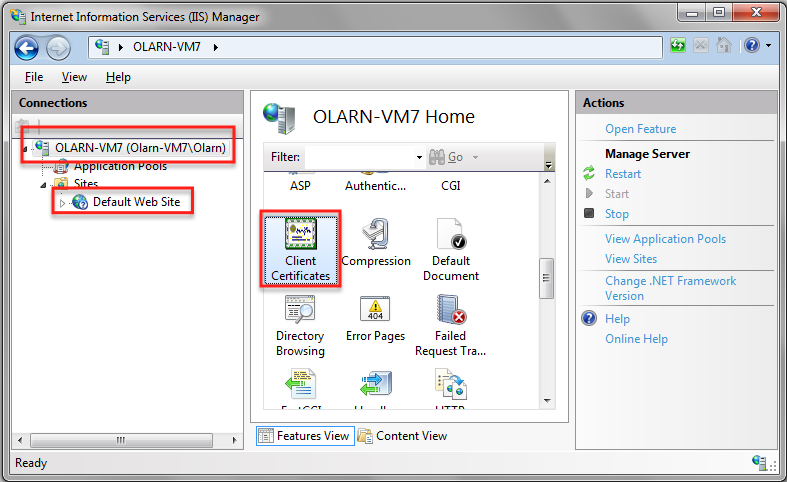
หมายเหตุ จากรูปตัวอย่าง เป็นการเพิ่ม feature บน Windows 7 สำหรับ Windows version อื่น ดูรายละเอียดเพิ่มเติมที่ <http://www.iis.net/ConfigReference/system.webServer/security/authentication/iisClientCertificateMappingAuthentication>



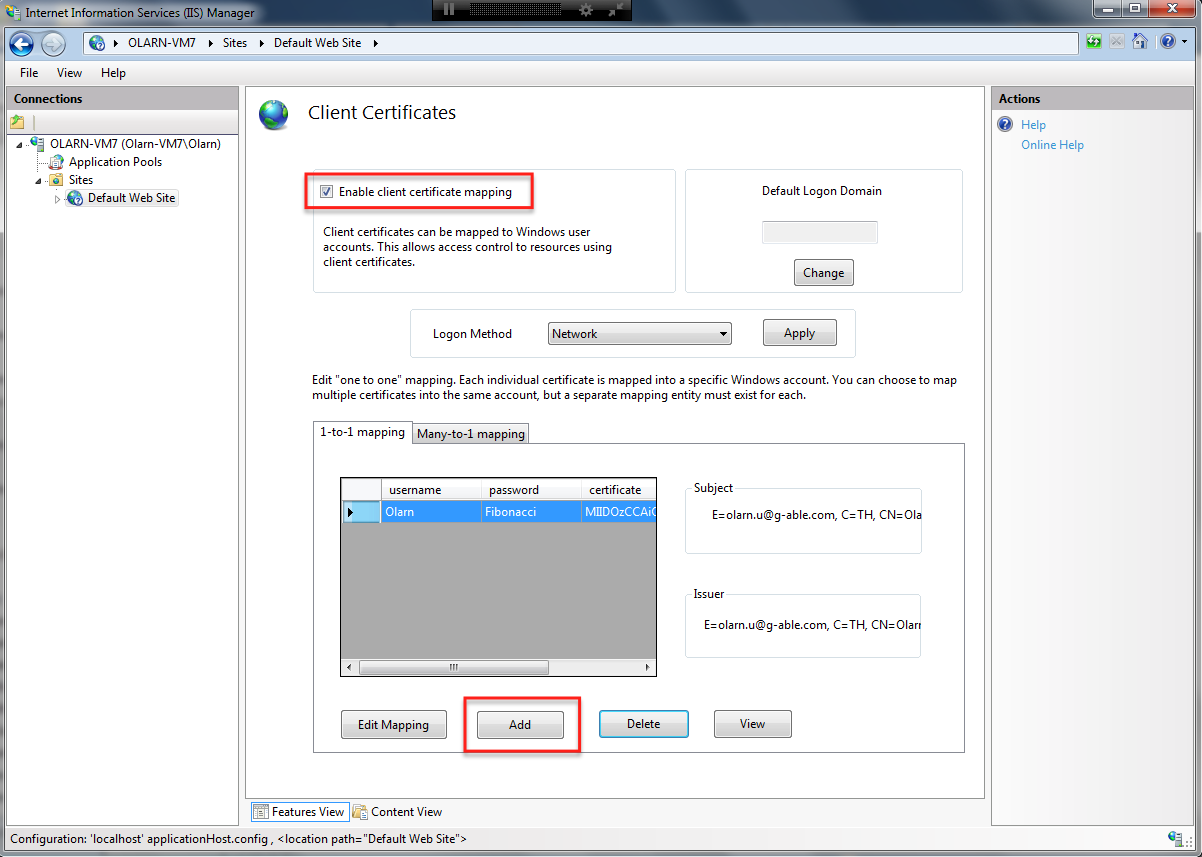
* + 1. Download plug-in ของ IIS7 เพื่อใช้ configure ค่าของ Client Certificate ง่ายขึ้น จาก URL <http://blogs.msdn.com/b/puneetgupta/archive/2009/08/03/where-is-the-client-certificates-ui-in-iis-7-0.aspx> จากนั้นทำการติดตั้ง โดยเลือกไฟล์ให้ตรงกับ OS ว่าเป็น 32bit หรือ 64bit (ใน zip file จะมีให้ทั้ง 2 edition)

หมายเหตุ ในกรณีที่ Client Mapping แบบ UI ใช้งานไม่ได้ เราสามารถ configure แบบ manual ได้ โดยดูวิธีทำเพิ่มเติมที่ <http://learn.iis.net/page.aspx/478/configuring-one-to-one-client-certificate-mappings/>

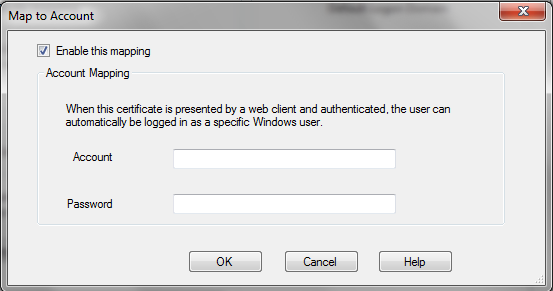
* + 1. เมื่อติดตั้งเสร็จแล้ว ให้เปิด IIS Management ขึ้นมาใหม่ จะพบว่า ที่ Server และ Default Website จะมี icon “Client Certificate” ปรากฎขึ้นมา



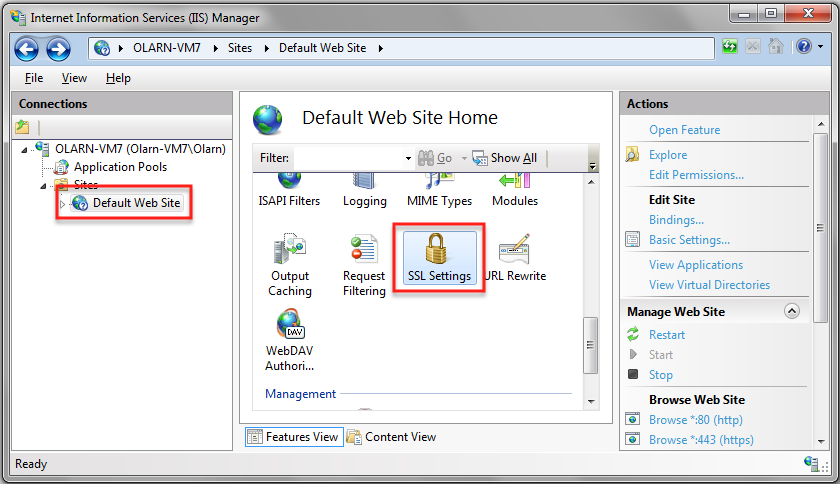
* + 1. เลือก website หรือ web application ที่ต้องการ แล้ว double click “Client Certificates” จากนั้นในหน้า Client Certificates ให้ click “Enable client certificate mapping” แล้วเพิ่ม certificate เข้ามาโดย click ปุ่ม “Add” ที่อยู่ด้านล่าง เลือกไฟล์ certificate ที่ได้จาก client (ที่สร้างจาก Mac)

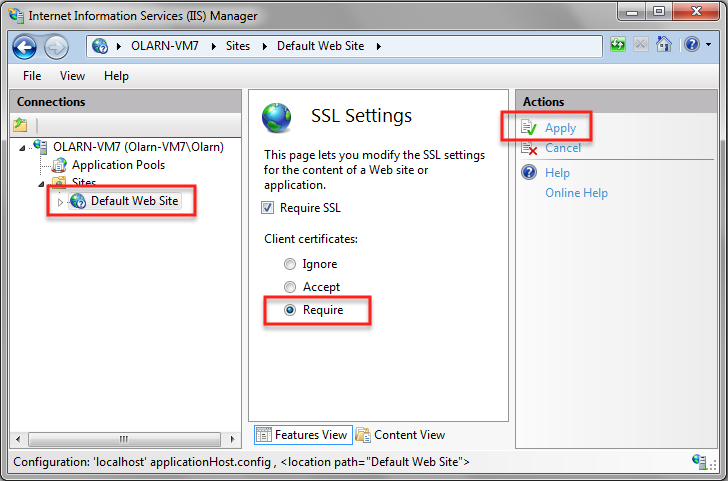


* + 1. ระบุ Account ที่ต้องการผูก certificate โดยป้อน username และ password ของ user ที่สร้างไว้ใน Windows แล้ว click OK, พิมพ์ password เพื่อยืนยันอีกครั้งแล้ว click OK



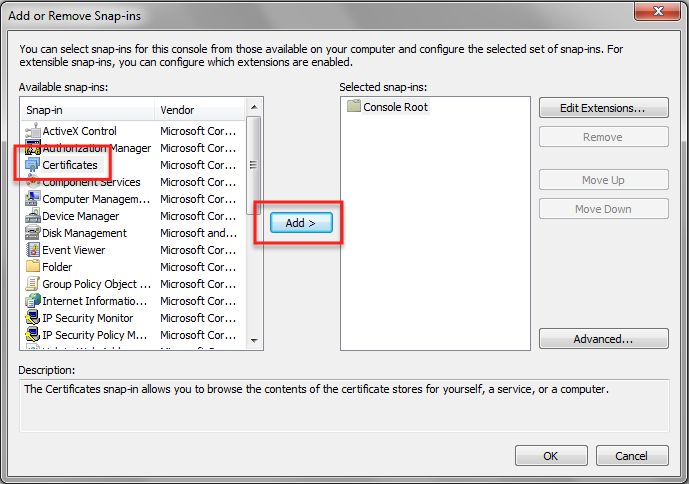
* + 1. จากนั้น ตั้งค่าให้ IIS บังคับให้ client ต้อง connect โดยใช้ Client Certificate เท่านั้น โดย double click ที่ SSL Settings ของ website หรือ web application ที่ต้องการ จากนั้นในหน้า SSL Settings ให้เลือก Client certificate เป็น “Require” แล้ว click “Apply” ใน Action ด้านขวามือ

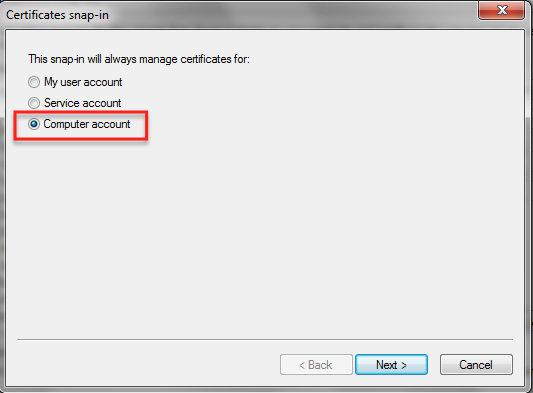
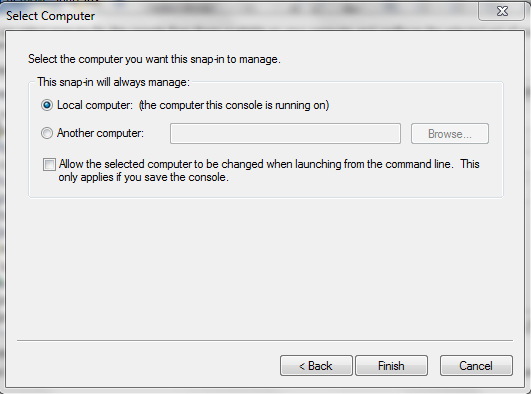




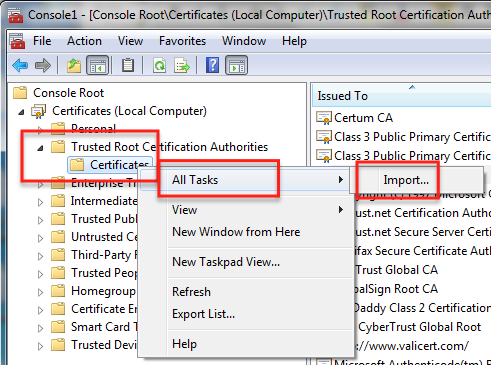
ทดลอง browse website จากเครื่อง Mac สังเกตุว่า Safari จะ popup ให้เราเลือก Certificate แต่ถึงแม้ว่าเราจะเลือก client certificate ที่ถูกต้องแล้วก็ตาม ก็ยังเข้าไม่ได้ นั่นเป็นเพราะ client certificate ที่เราสร้างขึ้นมานั้น ไม่ได้ถูกรับรองโดย CA เพราะฉะนั้น web server จะไม่ยอมให้ client ผ่านเข้ามาใช้ website ได้แม้ว่าเราจะ mapping certificate ในขั้นตอนที่แล้วก็ตาม เพราะฉะนั้นเราจะต้อง add certificate นี้เข้าไปใน trusted root ของ Windows เพื่อให้ web server เชื่อถือ certificate ตัวนี้ในขั้นตอนถัดไป

* 1. **Add Certificate as Trusted Certificate**
     1. เปิด Certificate Manager โดยใน Windows 7 หรือ Windows 2008 / 2008R2 ให้ click ปุ่ม Start แล้ว search คำว่า mmc จะเห็น mmc.exe ก็กด enter หรือถ้าเป็น Windows version ต่ำกว่านั้น ให้ click ปุ่ม Start > Run แล้วพิมพ์ mmc.exe
     2. ไปที่เมนู File > Add / Remove Snap-in…
     3. ในหน้า Add or Remove Sanp-ins เลือก “Certificates” ทางด้านซ้ายมือ แล้ว click ปุ่ม “Add >” โปรแกรมจะแสดงหน้าจอ Certificates Snap-ins ขึ้นมา ให้เลือก “Computer Account” แล้ว click ปุ่ม “Next >” และ click “Finish” ในหน้าถัดไป โปรแกรมจะแสดงหน้า mmc console อีกครั้ง ให้ click “OK”

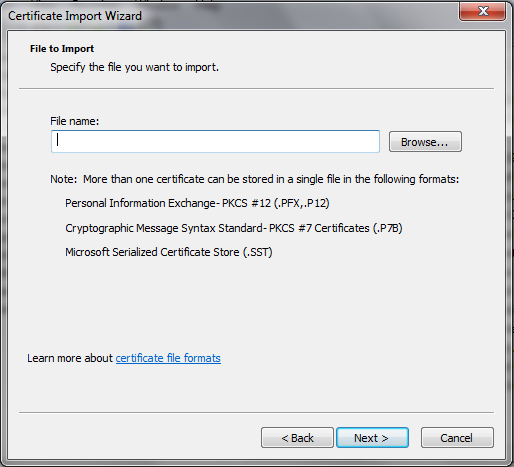


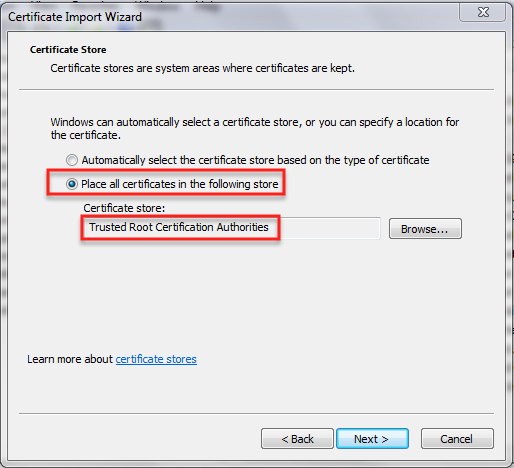
* + 1. จาก tree view ด้านซ้าย ไปที่ Trusted Root Certificate Authorities > Certificates แล้ว click ขวา เลือก “All Tasks > Import…” แล้ว click Next

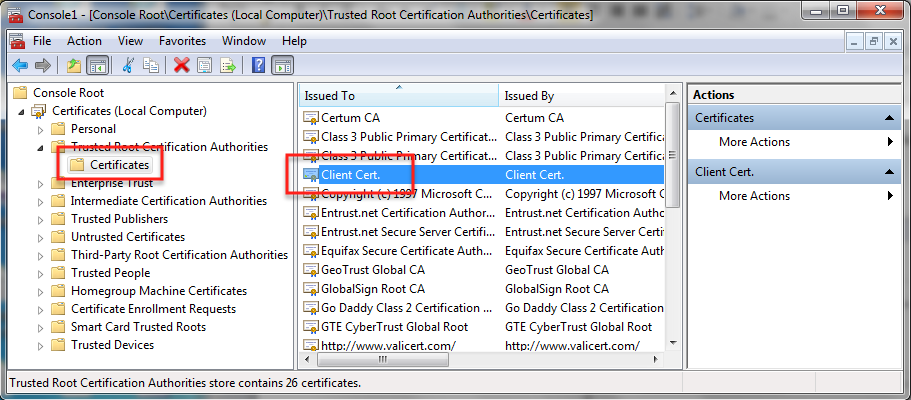


* + 1. ในหน้า File to Import ของ Certificate Import Wizard ให้ click ปุ่ม “Browse…” แล้วเลือกไฟล์ Certificate ที่ได้จาก client (เครื่อง Mac) แล้ว click ปุ่ม “Open” จากนั้น click “Next”



* + 1. ในหน้า Certificate Store ของ Certificate Import Wizard ให้เลือก “Place all certificates in the following store” และ Browse ไปที่ “Trusted Root Certification Authorities” (โดย default ควรจะเลือกไว้แล้ว) จากนั้น click “Next” และ click “Finish” สังเกตุว่า Certificate จาก client ควรจะปรากฎใน Certificates ของ Trusted Root Certification Authorities



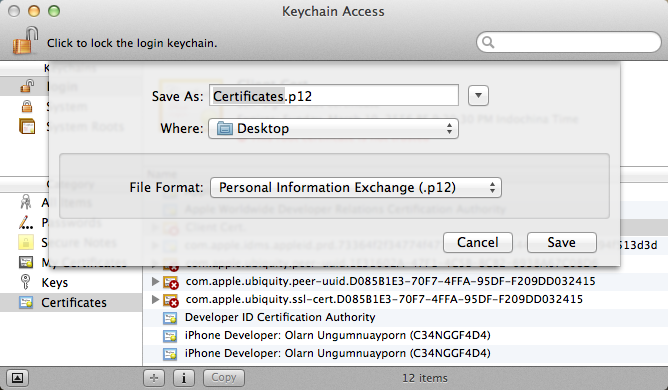


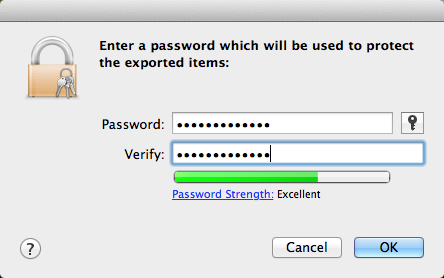
* + 1. ทอลอง Browse จากเครื่อง Mac คราวนี้เราจะสามารถเข้าถึง website ที่กำหนดได้แล้ว
  1. **Sign Private Key and Export Client Certificate**

ในการสร้าง client certificate ตามขั้นตอนที่ผ่านมานั้น เป็นการสร้าง Public Key ซึ่งถ้า certificate นั้นถูกเก็บอยู่ใน Keychain ของ Mac เราก็ไม่จำเป็นจะต้องสร้าง Private Key อีก ก็สามารถใช้ได้เลย แต่เมื่อเราต้องการนำ certificate ตัวนี้ไปใช้ที่อื่น เช่น bundle ไปกับ application เราจำเป็นที่จะต้องใส่ Private Key ลงไปใน certificate ด้วย เพราะถ้าไม่มี Private Key ใครก็ตามที่ได้ certificate ตัวนี้ไป ก็จะสามารถ login เข้าไปใช้ website ที่เราจำกัดไว้ได้ทันที เป็นการป้องกันอีกชั้นหนึ่งตามมาตรฐานของการ authentication และ encryption ด้วย certificate

ขั้นตอนในการ Sign Private Key และ Export จาก Keychain ของ Mac มีดังนี้

* + 1. ทำการ export client certificate เพื่อเตรียมนำไปใช้ใน iOS application โดยเปิด Keychain Access แล้ว click ขวาที่ certificate ที่เราสร้างเอาไว้ในข้อ 4.3 แล้วเลือก Export “xxx”… (xxx คือชื่อ certificate ที่สร้างเอาไว้)
    2. ในหน้า Save As นั้น สังเกตุว่า File Format จะต้องเป็น “Personal Information Exchange (.p12)” ให้ click ปุ่ม Save โปรแกรมจะให้ป้อน password ซึ่ง password นี้จะถูกนำไปสร้าง private key ที่ถูกฝังไปใน certificate ที่เรากำลังจะสร้างขึ้น

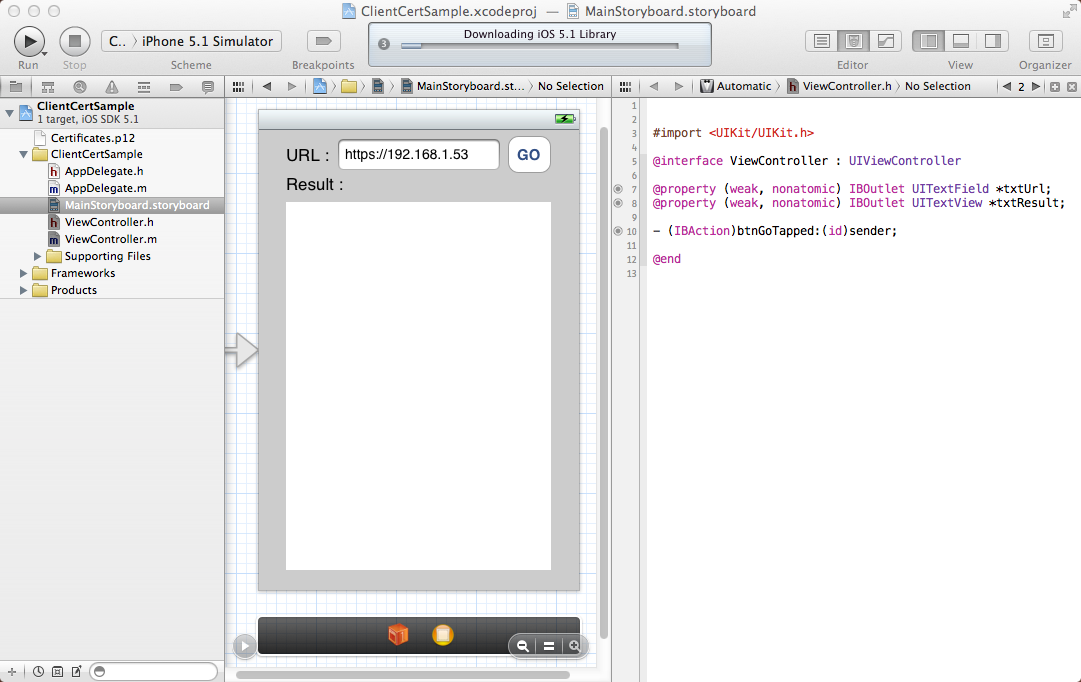




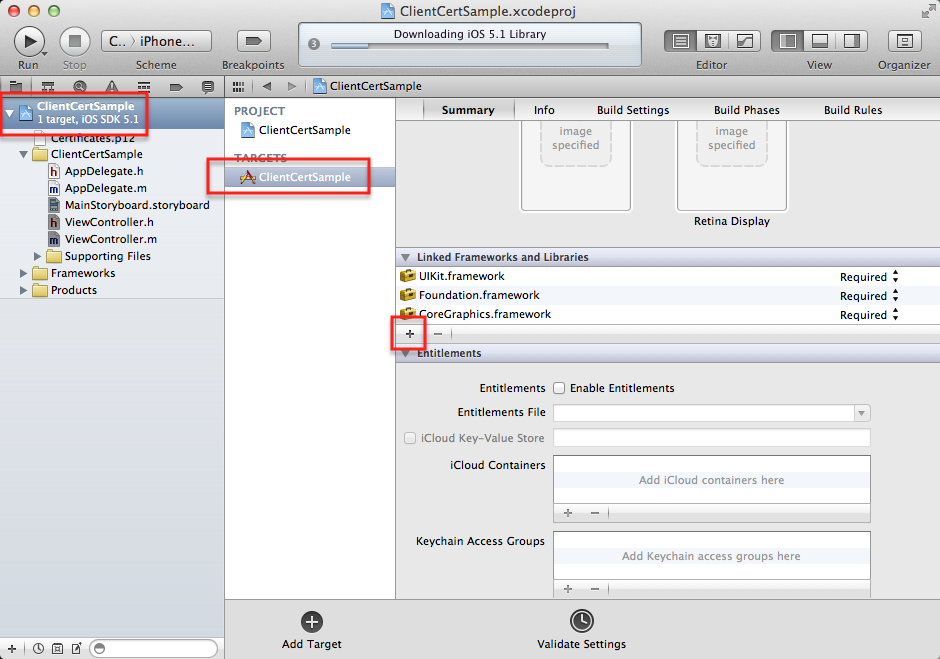
* + 1. ป้อน password แล้ว click OK เพื่อ export certificate ไปใช้ต่อใน iOS application
  1. **iOS Client Certificate Authentication**

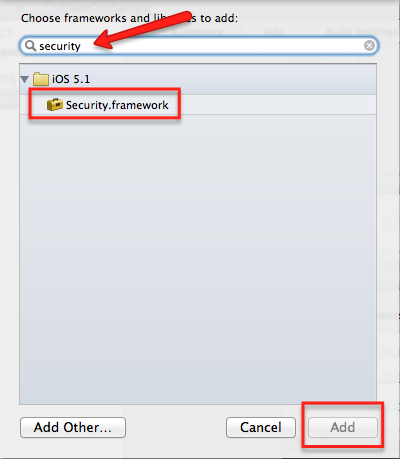
เมื่อเราสร้าง certificate ทั้ง 2 ฝั่งเรียบร้อยแล้ว *username* ที่ถูกกำหนดที่ server และ *client certificate* ที่สร้างใน client จะต้องถูก bundle ไปกับ application เพื่อใช้ในการ authenticate ไปยัง server มีขั้นตอนดังนี้

* + 1. สร้าง project iPhone Application ใน Xcode
    2. Drag certificate.p12 ที่ export จากข้อ 5.6 แล้ว drop ลงใน project
    3. เพิ่ม control ต่างๆ ใน view และผูก outlet และ action ตามรูป



* + 1. เพิ่ม Security.Framework เข้ามาใน project โดย click ที่ root node ใน Project Navigator แล้วเลือก TARGET แล้ว scroll ลงมาใน block “Linked Frameworks and Libraries” click ที่ปุ่ม “+” จากนั้น พิมพ์ security ในช่อง search แล้วเลือก “Security.Framework” แล้ว click ปุ่ม Add





* + 1. เพิ่ม code ใน ViewController.h

#import <UIKit/UIKit.h>

#import <Security/Security.h> //1

@interface ViewController : UIViewController<NSURLConnectionDelegate> //2

@property (strong, nonatomic) NSURLConnection \*connection; //3

@property (strong, atomic) NSMutableData \*returnedData; //3

@property (weak, nonatomic) IBOutlet UITextField \*txtUrl;

@property (weak, nonatomic) IBOutlet UITextView \*txtResult;

- (IBAction)btnGoTapped:(id)sender;

OSStatus extractIdentityAndTrust(CFDataRef inP12data, //4

SecIdentityRef \*identity,

SecTrustRef \*trust);

@end

//1. ทำการ Import package Security เข้ามาใน class

//2. กำหนดให้ ViewController ทำการ handling NSURLConnectionDelegate

//3. เพิ่ม property อีก 2 ตัว คือ NSURLConnection เพื่อใช้ connect ไปยัง web server และ   
 NSMutableData เพื่อใช้รับข้อมูลที่ส่งกลับมา server แบบ asynchronouss

//4. Header ของ method สำหรับ extract identity ออกมาจาก certificate

* + 1. เพิ่ม code @synthesize ใน ViewController.m (ใน comment //1 และ //2)

@implementation ViewController

@synthesize connection; //1

@synthesize returnedData; //2

@synthesize txtUrl;

@synthesize txtResult;

* + 1. เพิ่ม code ใน method “btnGoTapped” เพื่อสร้าง connection ไปยัง server

- (IBAction)btnGoTapped:(id)sender

{

returnedData = [[NSMutableData alloc] init]; //1

NSURL \*url = [[NSURL alloc] initWithString:@"https://192.168.1.53"]; //2

NSURLRequest \*request = [NSURLRequest requestWithURL:url]; //3

self.connection = [[NSURLConnection alloc] initWithRequest:request //4

delegate:self startImmediately:NO];

[self.connection start]; //5

}

//1. Initial object returnedData สำหรับรับข้อมูลจาก server แบบ asynchronous

//2. สร้าง object ของ NSURL และระบุ url ของ server

//3. สร้าง object ของ NSURLRequest เพื่อเตรียม request ไปยัง server

//4. สร้าง object ของ NSURLConnection เพื่อ connect ไปยัง server

//5. สั่ง connection Start เพื่อเริ่ม connect และ download ข้อมูล

* + 1. เพิ่ม method “extractIdentityAndTrust” เพื่อ extract identity จาก certificate

OSStatus extractIdentityAndTrust(CFDataRef inP12data,

SecIdentityRef \*identity,

SecTrustRef \*trust)

{

OSStatus securityError = errSecSuccess;

CFStringRef password = CFSTR("pass@word1"); //1

const void \*keys[] = { kSecImportExportPassphrase };

const void \*values[] = { password };

CFDictionaryRef options = CFDictionaryCreate(NULL, keys, values, 1, NULL, NULL);

CFArrayRef items = CFArrayCreate(NULL, 0, 0, NULL);

securityError = SecPKCS12Import(inP12data, options, &items);

if (securityError == 0) {

CFDictionaryRef myIdentityAndTrust = CFArrayGetValueAtIndex(items, 0);

const void \*tempIdentity = NULL;

tempIdentity = CFDictionaryGetValue(myIdentityAndTrust, kSecImportItemIdentity);

\*identity = (SecIdentityRef)tempIdentity;

const void \*tempTrust = NULL;

tempTrust = CFDictionaryGetValue(myIdentityAndTrust, kSecImportItemTrust);

\*trust = (SecTrustRef)tempTrust;

}

if (options) {

CFRelease(options);

}

return securityError;

}

//1. ระบุ Private Key ที่นี่

* + 1. เพิ่ม method “connection:willSendRequestForAuthenticationChallenge:” เพื่อ load certificate

- (void)connection:(NSURLConnection \*)connection willSendRequestForAuthenticationChallenge:(NSURLAuthenticationChallenge \*)challenge

{

// load certificate when server require

NSString \*path = [[NSBundle mainBundle] pathForResource:@"Certificates" ofType:@"p12"];

NSData \*p12data = [NSData dataWithContentsOfFile:path];

CFDataRef inP12data = (\_\_bridge CFDataRef)p12data;

SecIdentityRef myIdentity;

SecTrustRef myTrust;

extractIdentityAndTrust(inP12data, &myIdentity, &myTrust);

SecCertificateRef myCertificate;

SecIdentityCopyCertificate(myIdentity, &myCertificate);

const void \*certs[] = { myCertificate };

CFArrayRef certsArray = CFArrayCreate(NULL, certs, 1, NULL);

NSURLCredential \*credential = [NSURLCredential credentialWithIdentity:myIdentity

certificates:(\_\_bridge NSArray\*)certsArray

persistence:NSURLCredentialPersistencePermanent];

[[challenge sender] useCredential:credential forAuthenticationChallenge:challenge];

}

* + 1. เพิ่ม method “connection:didReceiveData:” เพื่อนำ data ที่ได้ใน session นั้นมา appent กับ returnedData ที่ได้ใน thread ก่อนหน้านั้น

- (void)connection:(NSURLConnection \*)connection didReceiveData:(NSData \*)data

{

// append returned data from server

[returnedData appendData:data];

}

* + 1. เพิ่ม method “connectionDidFinishLoading:” เพื่อนำ data ที่ได้ใน returnedData มาแสดงผล ซึ่งควรจะอยู่ในรูปของ html หรือ JSON ตามที่ web server ส่งผลลัพธ์กลับมา

- (void)connectionDidFinishLoading:(NSURLConnection \*)connection

{

if (returnedData) {

txtResult.text = [NSString stringWithUTF8String:[returnedData bytes]];

}

}

1. **Conclusion**
2. **References**