คู่มือการใช้งาน TeDA Sign API

[TeDA Sign Application Programming Interface]

ฉบับเรียนเล่นเป็นเร็ว 1st Edition

ETDA สพธอ
www.etda.or.th
สำนักงานพัฒนาธุรกรรมทางอิเล็กทรอนิกส์ (องค์การมหาชน)
Electronic Transactions Development Agency (Public Organization)

บทที่ 1
ภาพรวม

ปัจจุบันการทำธุรกรรมทางอิเล็กทรอนิกส์ได้มีปริมาณเพิ่มขึ้นอย่างมีนัยสำคัญ กระบวนการลงลายมือชื่ออิเล็กทรอนิกส์จึงเป็นกลไกที่สำคัญในการทำให้ธุรกรรมนั้นมีผลผูกพันทางกฎหมาย ตามพระราชบัญญัติว่าด้วยธุรกรรมทางอิเล็กทรอนิกส์ พ.ศ. 2544 ได้บัญญัติไว้ถึงการลงลายมือชื่ออิเล็กทรอนิกส์ 2 ประเภท กล่าวคือ

1) ลายมือชื่ออิเล็กทรอนิกส์ ตามมาตรา 9 เช่น การส่งอีเมล์ (e-mail) การทำธุรกรรมโดยใช้ชื่อบัญชี (User Name) และรหัสผ่าน เป็นต้น
2) ลายมือชื่ออิเล็กทรอนิกส์ที่เชื่อถือได้ ตามมาตรา 26 ซึ่งได้แก่การลงลายมือชื่ออิเล็กทรอนิกส์ด้วยเทคโนโลยีโครงสร้างพื้นฐานกุญแจสาธารณะ (Public Key Infrastructure: PKI)

ถึงแม้ว่าพระราชบัญญัติว่าด้วยธุรกรรมทางอิเล็กทรอนิกส์ได้รับรองให้การลงลายมือชื่อทั้งสองประเภทมีผลผูกพันทางกฎหมาย แต่ในกรณีที่มีการลงลายมือชื่ออิเล็กทรอนิกส์ตามมาตรา 9 อาจมีประเด็นคำถามหรือข้อพิพาทว่าการลงลายมือชื่ออิเล็กทรอนิกส์นั้นมีเหมาะสมกับธุรกรรมที่ทำหรือไม่ รวมถึงมีการแก้ไขข้อมูลภายหลังจากที่ได้ลงลายมือชื่ออิเล็กทรอนิกส์หรือไม่ ด้วยเหตุนี้ในการทำธุรกรรมทางอิเล็กทรอนิกส์ที่มีมูลค่าสูง หรือมีความสำคัญการลงลายมือชื่ออิเล็กทรอนิกส์ตามมาตรา 26 จึงเป็นกลไกที่สำคัญที่ทำให้เชื่อมั่นได้ เนื่องจาก ในการลงลายมือชื่อด้วยเทคโนโลยี PKI จะให้ความน่าเชื่อถือในการธุรกรรม 3 ประการ คือ

1) **ความครบถ้วนถูกต้อง (Integrity)** คือความสามารถที่จะตรวจสอบว่าข้อมูลที่ได้ทำธุรกรรมไปนั้นมีความครบถ้วน ถูกต้อง และไม่ถูกเปลี่ยนแปลงแก้ไขภายหลังจากที่ได้ลงลายมือชื่อไปแล้ว
2) **การยืนยันตัวบุคคล (Authentication)** คือ มีเพียงเจ้าของลายมือชื่อเท่านั้นที่สามารถลงลายมือชื่ออิเล็กทรอนิกส์ด้วยเทคโนโลยี PKI ได้ เนื่องจาก เจ้าของลายมือชื่อเป็นเพียงผู้เดียวที่ล่วงรู้ข้อมูลสำหรับลงลายมือชื่ออิเล็กทรอนิกส์ หรือที่เรียกว่า กุญแจส่วนตัว (Private Key)
3) **การห้ามปฏิเสธความรับผิด (Non-Repudiation)** เมื่อเจ้าของลายมือชื่อได้ทำการลงลายมือชื่อแล้วนั้น จะไม่สามารถปฏิเสธผลผูกพันในธุรกรรมที่ตนได้ทำลงไปได้เนื่องจาก มีเพียงเจ้าของเป็นเพียงผู้เดียวที่ล่วงรู้กุญแจส่วนตัว

บทที่ 2

กลไกในการลงลายมือชื่ออิเล็กทรอนิกส์ด้วยเทคโนโลยี PKI

การลงลายมือชื่ออิเล็กทรอนิกส์ด้วยเทคโนโลยี PKI เป็นวิธีการที่ช่วยให้คู่กรณี (Related Party) และผู้ที่เกี่ยวข้องกับธุรกรรม (Relying Party) เชื่อมั่นได้ว่าผู้ที่ลงลายมือชื่ออิเล็กทรอนิกส์ด้วยเทคโนโลยี PKI เป็นผู้ทำธุรกรรมทางอิเล็กทรอนิกส์ดังกล่าวจริงและไม่สามารถปฏิเสธความรับผิดได้ อีกทั้งยังสามารถตรวจสอบได้ว่าเอกสารดังกล่าวมิได้ถูกแก้ไขภายหลังจากที่มีการลงลายมือชื่อ ซึ่งจากคุณสมบัติดังที่กล่าวมาข้างต้นเกิดจากการประยุกต์ใช้เทคโนโลยีระบบรหัสแบบอสมมาตร (Asymmetric Key Cryptography) และเทคโนโลยี Public Key Infrastructure (PKI) โดยมีกลไกในสร้างและตรวจสอบลายมือชื่ออิเล็กทรอนิกส์ดังรายละเอียดที่จะกล่าวต่อไป

เทคโนโลยีระบบรหัสแบบอสมมาตร (Asymmetric Key Cryptography)

เทคโนโลยีระบบรหัสแบบอสมมาตรเป็นเทคโนโลยีการเข้ารหัสลับ (Encryption) ที่ใช้คู่กุญแจ (Key Pair) ซึ่งประกอบด้วยกุญแจส่วนตัว (Private Key) และกุญแจสาธารณะ (Public Key) ที่มีลักษณะเป็นตัวเลขสุ่มขนาดยาวตามอัลกอริธึมที่ใช้ ตัวอย่างเช่น กุญแจที่สร้างขึ้นด้วยอัลกอริธึม RSA ขนาดความยาว 2048 บิต

```
3082 010a 0282 0101 00aa 5655 eed9 2d8a 3132 603b a971 c896 ab46 c821 709c
dd0c ae34 6dbc 40ba 036d 32a8 e681 ab5f 55f6 6d08 fc7a c263 b72a 8ee0 3d83
4a58 20a0 6598 16c6 7c72 0c2a 5f2b a6e3 2659 5dac 6a9f a8cc f9f1 3f90 4478 9a02
52fa 9b16 fd1e 2ada 2b86 4053 80c6 fae2 aba8 2c41 909c 1bda a664 34d9 09da
e2a5 27ce 7925 e516 8503 6ef6 a067 46ff 2306 f83e 72a6 67ae ada9 7654 b95d 4613
38e3 64f6 915f 125f b650 bb40 927d 2527 353c 7eb6 d100 10cd d987 2001 8106
3e6b f2bd b5ba b688 0777 1e69 74a1 3ac0 29f8 99d6 efc9 76c4 b277 0661 1715
42de c51f 389e 066d d056 b9c6 bdba 6bba 898b 17d5 64da 1eb9 842a 11a4 11db
301a 8b42 69e1 52cc f1cd 1def 9851 4e1b b480 217c 4902 0301 0001
```

ตัวอย่างกุญแจสาธารณะที่สร้างขึ้นด้วยอัลกอริธึม RSA

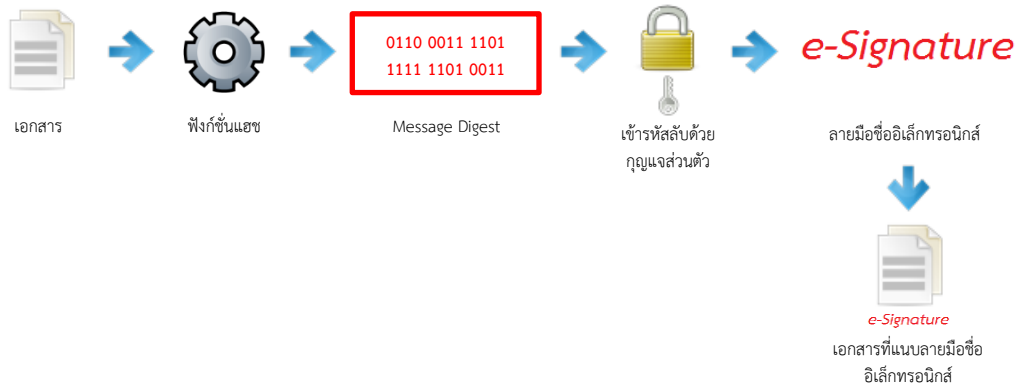
ซึ่งในการเข้ารหัสลับด้วยเทคโนโลยีระบบรหัสแบบอสมมาตรนั้น หากใช้กุญแจส่วนตัวในการเข้ารหัสลับ (Encryption) จำเป็นต้องใช้กุญแจสาธารณะที่ถูกสร้างขึ้นมาพร้อมกันในการถอดรหัสลับ (Decryption) ในทางกลับกันหากใช้กุญแจสาธารณะในการเข้ารหัสลับ (Encryption) จำเป็นต้องใช้กุญแจส่วนตัวที่ถูกสร้างขึ้นมาพร้อมกันในการถอดรหัสลับ (Decryption) ดังภาพ

การเข้ารหัสลับด้วยเทคโนโลยีระบบรหัสแบบอสมมาตร

จากกลไกการทำงานดังกล่าวข้างต้น หากข้อมูลถูกเข้ารหัสลับด้วยกุญแจส่วนตัวจะทำให้มั่นใจได้ว่าข้อมูลดังกล่าวถูกส่งหรือสร้างมาจากเจ้าของคู่กุญแจจริงเนื่องจาก กุญแจส่วนตัวจะถูกเก็บที่เจ้าของคนเดียวเท่านั้น ซึ่งเป็นคุณสมบัติของการยืนยันตัวบุคคล (Authentication) และการห้ามปฏิเสธความรับผิด (Non-Repudiation)

การลงลายมือชื่ออิเล็กทรอนิกส์

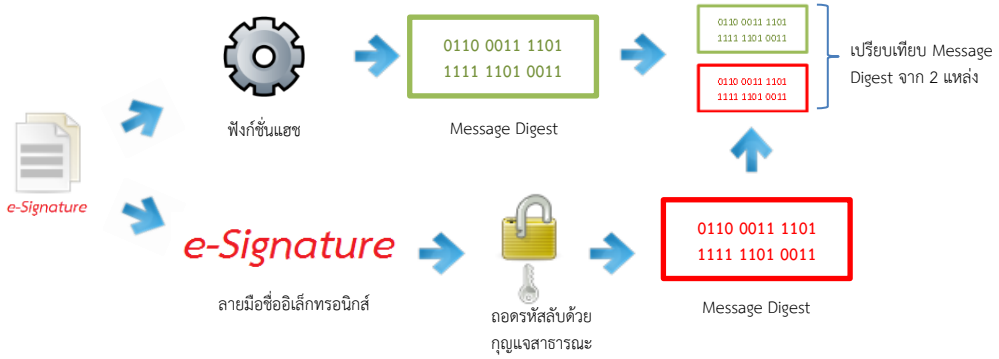
ในการลงลายมือชื่ออิเล็กทรอนิกส์นั้น เป็นการนำเทคนิคการเข้ารหัสลับด้วยกุญแจส่วนตัวมาเข้ารหัสลับ Message Digest ของไฟล์เอกสาร ซึ่ง Message Digest นี้เปรียบเสมือนลายพิมพ์นิ้วมือที่บ่งบอกถึงเอกลักษณ์ของข้อมูลต้นฉบับ โดยมีลักษณะเป็นบิตสตริงที่บุคคลอื่นๆ ไม่สามารถสร้างเลียนแบบขึ้นมาใหม่ได้ เนื่องจาก Message Digest ที่สร้างจากฟังก์ชั่นแฮช (Hash Function) นั้นจะให้ผลลัพธ์ที่แตกต่างกันออกไปตามเนื้อหาของเอกสาร ดังนั้นลายมือชื่ออิเล็กทรอนิกส์ของแต่ละไฟล์เอกสารจึงมีเอกลักษณ์เฉพาะแตกต่างกันตามข้อมูลในไฟล์เอกสาร ซึ่งกลไกการลงลายมือชื่ออิเล็กทรอนิกส์มีรายละเอียดดังภาพ



กลไกการลงลายมือชื่ออิเล็กทรอนิกส์

จากกลไกในการลงลายมือชื่ออิเล็กทรอนิกส์ดังกล่าวข้างต้น ลายมือชื่ออิเล็กทรอนิกส์จึงความแตกต่างกันตามไฟล์เอกสาร ทำให้สามารถตรวจสอบได้ว่าลายมือชื่ออิเล็กทรอนิกส์นั้นๆ มีผลผูกพันกับไฟล์เอกสารหรือไม่ และไม่สามารถปลอมแปลงได้เนื่องจากกุญแจส่วนตัวถูกสร้างจากตัวเลขสุ่มที่มีขนาดยาวจนเครื่องคอมพิวเตอร์ในปัจจุบันไม่สามารถสร้างค่าตัวเลขที่เหมือนกันกับกุญแจส่วนตัวได้

เมื่อผู้รับได้รับเอกสารที่มีลายมือชื่อลงลายมือชื่ออิเล็กทรอนิกส์ แอปพลิเคชั่นที่รองรับการตรวจสอบลายมือชื่ออิเล็กทรอนิกส์จะมีกระบวนการในการตรวจสอบความครบถ้วนถูกต้อง (Integrity) ของข้อมูล โดยการนำเอกสารที่ได้รับมาเข้าสู่ฟังก์ชั่นแฮชเพื่อสร้างเป็น Message Digest แล้วนำไปเปรียบเทียบกับ Message Digest ที่ได้จากการถอดรหัสลับด้วยกุญแจสาธารณะของผู้ที่ลงลายมือชื่อ หาก Message Digest ทั้งสองมีค่าเท่ากันหมายความว่าเอกสารดังกล่าวไม่ถูกเปลี่ยนแปลงแก้ไขภายหลังจากที่ได้ลงลายมือชื่อไปแล้วดังภาพ



กระบวนการตรวจสอบลายมือชื่ออิเล็กทรอนิกส์

เทคโนโลยีโครงสร้างพื้นฐานกุญแจสาธารณะ (Public Key Infrastructure: PKI)

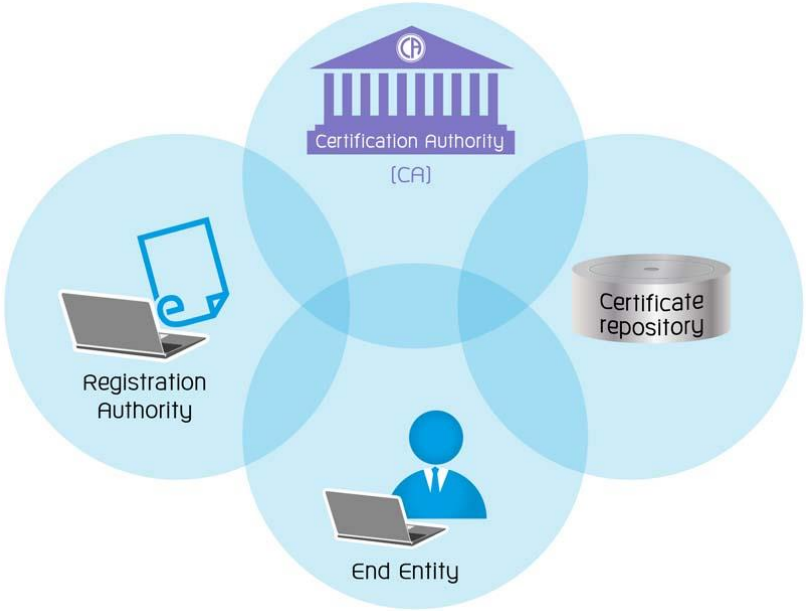
การลงลายมือชื่ออิเล็กทรอนิกส์ด้วยเทคโนโลยีระบบรหัสแบบอสมมาตรเพียงอย่างเดียวอาจไม่เพียงพอ เนื่องจาก ผู้รับเอกสารจะทราบได้อย่างไรว่ากุญแจสาธารณะที่นำไปใช้ในการตรวจสอบลายมือชื่ออิเล็กทรอนิกส์เป็นของผู้ลงลายมือชื่อจริง จึงจำเป็นต้องมีเทคโนโลยีโครงสร้างพื้นฐานกุญแจสาธารณะหรือ เทคโนโลยี PKI เข้ามาเพื่อรับรองว่า กุญแจสาธารณะที่ใช้ในการตรวจสอบลายมือชื่อเป็นของเจ้าของลายมือชื่อจริง เจ้าของลายมือชื่อนั้นมีตัวตนอยู่จริงและลายมือชื่ออิเล็กทรอนิกส์นั้นมีผลผูกพันทางกฎหมายไปถึงเจ้าของลายมือชื่ออิเล็กทรอนิกส์ อีกทั้งยังมีความสามารถในการตรวจสอบสถานะของกุญแจสาธารณะว่ายังมีความน่าเชื่อถืออยู่หรือไม่

จากหลักการดังกล่าวข้างต้นเทคโนโลยี PKI จึงประกอบด้วย 4 องค์ประกอบหลักคือ

1) ผู้ให้บริการออกใบรับรอง (Certification Authority: CA) เป็นบุคคลที่สามหรือหน่วยงานกลางที่ทำหน้าที่ในการออกใบรับรองอิเล็กทรอนิกส์ (Certificate) เพื่อรับรองว่ากุญแจสาธารณะเป็นของบุคคลที่กล่าวอ้างจริง อีกทั้งยังมีหน้าที่ในการแจ้งสถานะของใบรับรองอิเล็กทรอนิกส์

ว่ายังมีความน่าเชื่อถือหรือไม่ ด้วยการให้บริการเผยแพร่รายการเพิกถอนใบรับรอง (Certificate Revocation List: CRL) หรือให้บริการตรวจสอบสถานะใบรับรองอิเล็กทรอนิกส์แบบออนไลน์ (Online Certificate Status Protocol: OCSP)

2) เจ้าหน้าที่รับลงทะเบียน (Registration Authority: RA) ทำหน้าที่รับลงทะเบียนและตรวจสอบความมีตัวตนอยู่จริง เมื่อมีการยื่นขอใบรับรองอิเล็กทรอนิกส์ แจ้งเพิกถอนใบรับรองอิเล็กทรอนิกส์ หรือต่ออายุใบรับรองอิเล็กทรอนิกส์
3) ผู้ใช้บริการ (End Entity) ในที่นี้คือเจ้าของลายมือชื่อซึ่งเป็นบุคคลที่ประสงค์จะยื่นขอใช้บริการใบรับรองอิเล็กทรอนิกส์ผ่านทางเจ้าหน้าที่รับลงทะเบียน ทั้งนี้เจ้าของลายมือชื่อมีหน้าที่ในการแสดงหลักฐานต่างๆ รวมถึงปฏิบัติตามข้อตกลงที่ CA กำหนดไว้
4) แหล่งเก็บข้อมูล (Repository) เป็นระบบคอมพิวเตอร์ที่เปิดให้บุคคลที่เกี่ยวข้อง (Relying Parties) สามารถสืบค้นใบรับรองอิเล็กทรอนิกส์ของผู้ใช้บริการ รวมถึงรายการเพิกถอนใบรับรองได้จากแหล่งเก็บข้อมูลนี้ ทั้งนี้แหล่งเก็บข้อมูลอาจอยู่ในรูปของเว็บไซต์หรือระบบไดเรกทอรี (Directory) ขึ้นอยู่กับแนวปฏิบัติในการให้บริการของ CA



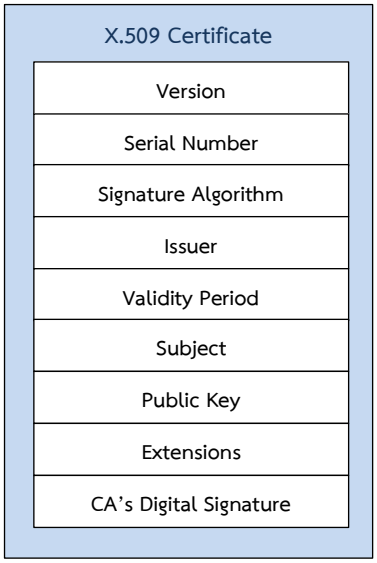
องค์ประกอบของเทคโนโลยีโครงสร้างพื้นฐานกุญแจสาธารณะ (PKI)

ใบรับรองอิเล็กทรอนิกส์ (Certificate)

เปรียบเทียบกับชีวิตประจำวัน ใบรับรองอิเล็กทรอนิกส์นี้ทำหน้าที่คล้ายบัตรประจำตัวประชาชนที่ออกโดยกระทรวงมหาดไทย สมมติบุคคลไปขอทำบัตรประจำตัวประชาชนครั้งแรก เจ้าหน้าที่จะทำการตรวจสอบสำเนาทะเบียนบ้านและสูติบัตร เพื่อพิสูจน์ตัวตนของบุคคลก่อนจะออกบัตรให้ เมื่อบุคคลนั้นทำนิติกรรมกับหน่วยงานใด หน่วยงานนั้นก็จะเรียกดูบัตรประจำตัวประชาชนเป็นหลัก เนื่องจาก

- มีข้อมูลชื่อ นามสกุล ที่อยู่ตามทะเบียนบ้าน
- มีรูปถ่ายของบุคคล
- มีวันออกบัตร วันบัตรหมดอายุ
- ปลอมแปลงได้ยาก
- ทางราชการสามารถระงับการใช้บัตรฯ ได้ หากบัตรถูกขโมยหรือใช้ในทางที่ผิด
- ออกโดยกระทรวงมหาดไทย ซึ่งเป็นหน่วยงานที่น่าเชื่อถือ

ในทำนองเดียวกัน ใบรับรองอิเล็กทรอนิกส์ก็ต้องมี CA ซึ่งเป็นหน่วยงานที่น่าเชื่อถือเป็นผู้ออกให้ สิ่งที่แตกต่างระหว่างบัตรประชาชนกับใบรับรองอิเล็กทรอนิกส์คือรูปแบบที่นำมาใช้งาน บัตรประชาชนอยู่ในรูปแบบเอกสารที่จับต้องได้ แต่ใบรับรองอิเล็กทรอนิกส์อยู่ในรูปของข้อมูลอิเล็กทรอนิกส์สำหรับบุคคลเดียว โดยมีรูปแบบข้อมูลตามมาตรฐาน X.509 Certificate เวอร์ชั่น 3 ที่กำหนดโดย ITU-T X.509 International Standard



ข้อมูลในใบรับรองอิเล็กทรอนิกส์ X.509 Certificate เวอร์ชั่น 3

การตรวจสอบความน่าเชื่อถือของใบรับรอง

จากความจำเป็นในการมีใบรับรองอิเล็กทรอนิกส์เพื่อรับรองว่ากุญแจสาธารณะที่ใช้ในการตรวจสอบลายมือชื่ออิเล็กทรอนิกส์นั้นมีความน่าเชื่อถือและเป็นของเจ้าของลายมือชื่อจริง ในทางเทคนิค แอปพลิเคชั่นจำเป็นต้องทำการตรวจสอบใบรับรองอิเล็กทรอนิกส์ทุกครั้งก่อนนำคู่กุญแจไปใช้งาน ตาม IETF RFC 5280 X.509 PKI Certificate ซึ่งมีขั้นตอนหลักๆ ดังนี้

1) ตรวจสอบว่าใบรับรองอิเล็กทรอนิกส์ออกโดย CA ที่กล่าวอ้างจริง และไม่ถูกแก้ไขภายหลังจากที่ได้รับการรับรองโดย CA แล้ว

2) ตรวจสอบว่าใบรับรองอิเล็กทรอนิกส์ของเจ้าของลายมือชื่อ รวมถึงใบรับรองอิเล็กทรอนิกส์ของ CA ยังไม่หมดอายุ โดยตรวจสอบได้วัน-เวลาที่หมดอายุได้จากข้อมูล Validity Period ในใบรับรองอิเล็กทรอนิกส์
3) ตรวจสอบว่าใบรับรองอิเล็กทรอนิกส์ของเจ้าของลายมือชื่อ รวมถึงใบรับรองอิเล็กทรอนิกส์ของ CA ไม่ถูกเพิกถอนในขณะที่ตรวจสอบใบรับรองอิเล็กทรอนิกส์ โดยสามารถตรวจสอบได้จากรายการเพิกถอนใบรับรอง (CRL) หรือให้บริการตรวจสอบสถานะใบรับรองอิเล็กทรอนิกส์แบบออนไลน์ (OCSP) ที่ CA ได้ประกาศไว้
4) ตรวจสอบว่ากุญแจส่วนตัวที่คู่กับใบรับรองอิเล็กทรอนิกส์ดังกล่าว ได้รับอนุญาตให้ใช้ในการลงลายมือชื่ออิเล็กทรอนิกส์ โดยตรวจสอบจากข้อมูล Key Usage ที่ระบุในใบรับรองอิเล็กทรอนิกส์

รูปแบบการลงลายมือชื่ออิเล็กทรอนิกส์

ลายมือชื่ออิเล็กทรอนิกส์ที่ถูกสร้างขึ้นด้วยเทคโนโลยี PKI สามารถสร้างให้อยู่ในรูปแบบต่างๆ ได้ ขึ้นอยู่กับการนำไปใช้งาน เช่น

- Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Standard

  PKCS#1 เป็นมาตรฐานที่ว่าด้วยการสร้างคู่กุญแจและการเข้ารหัสลับด้วยอัลกอริธึม RSA ซึ่งเป็นเทคโนโลยีระบบรหัสแบบอสมมาตร ผลลัพธ์ที่ได้จากการนำ Message Digest มาเข้ารหัสลับด้วยกุญแจส่วนตัวที่สร้างจากอัลกอริธึม RSA จึงเป็นลายมือชื่ออิเล็กทรอนิกส์พื้นฐานที่จะถูกนำไปประยุกต์ใช้ในรูปแบบอื่นๆ ต่อไป

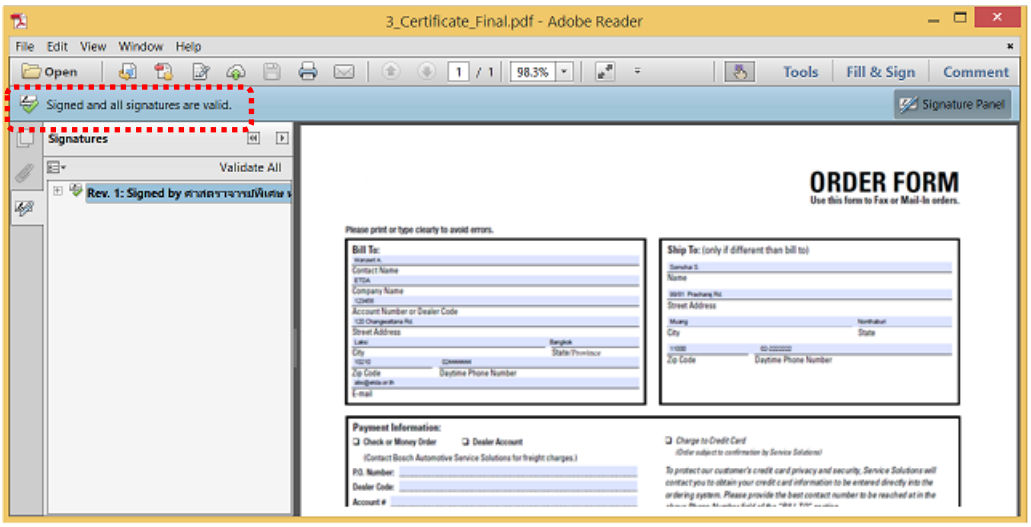
- Public-Key Cryptography Standards (PKCS) #7: Cryptographic Message Syntax Standard

  PKCS#7 เป็นมาตรฐานในการกำหนดโครงสร้างข้อมูลที่มีการลายมือชื่ออิเล็กทรอนิกส์กำกับและข้อมูลที่ถูกเข้ารหัสลับ ซึ่งโครงสร้างข้อมูลตามมาตรฐาน PKCS#7 นี้ถูกออกแบบให้สามารถนำผลลัพธ์ที่เกิดจากการเข้ารหัสลับตามมาตรฐาน PKCS#1 มารวมเป็นเนื้อเดียวกันกับเอกสาร และใบรับรองอิเล็กทรอนิกส์ได้ ซึ่งความสามารถในการจัดเก็บข้อมูลทั้งหมดเป็นเนื้อเดียวกันนี้จะช่วยให้สะดวกในการตรวจสอบความครบถ้วนถูกต้องในภายหลัง อีกทั้งยังง่ายต่อการจัดเก็บเอกสาร

  จากคุณสมบัติดังกล่าวข้างต้น PKCS#7 จึงถูกนำไปประยุกต์ใช้ต่อในอีกหลายรูปแบบ เช่น การรับส่งอีเมล์แบบปลอดภัย (Secure Multipurpose Internet Mail Extensions: S/MIME) การลงลายมือชื่ออิเล็กทรอนิกส์บนเอกสาร PDF เป็นต้น

- การลงลายมือชื่ออิเล็กทรอนิกส์บนเอกสาร PDF

การลงลายมือชื่อบนเอกสาร PDF เป็นการนำลายมือชื่ออิเล็กทรอนิกส์ในรูปแบบ PKCS#7 มาจัดเก็บในเนื้อเอกสาร PDF เพื่อให้ง่ายต่อการจัดเก็บและตรวจสอบความถูกต้องครบถ้วนในภายหลัง ซึ่งในการตรวจสอบลายมือชื่ออิเล็กทรอนิกส์บนเอกสาร PDF นี้สามารถทำได้ง่ายและสะดวกต่อผู้ใช้งานมาก เนื่องจากเมื่อผู้ใช้งานเปิดไฟล์เอกสาร PDF ด้วยโปรแกรม Adobe Reader โปรแกรมจะแสดงผลการตรวจสอบความถูกต้องครบถ้วนของเอกสาร และความน่าเชื่อถือของกุญแจที่ใช้ในการลงลายมือชื่ออิเล็กทรอนิกส์ในรูปแบบกราฟฟิกได้ทันที ดังตัวอย่างด้านล่าง



ผลการตรวจสอบลายมือชื่ออิเล็กทรอนิกส์บนเอกสาร PDF

บทที่ 4

TeDA Sign API

เนื่องจาก การลงลายมือชื่ออิเล็กทรอนิกส์ด้วยเทคโนโลยี PKI และตรวจสอบความถูกต้องของลายมือชื่ออิเล็กทรอนิกส์มีความซับซ้อน อีกทั้งมีรูปแบบการใช้งานที่หลากหลาย เช่น PKCS#1 PKCS#7 หรือ Portable Document Format (PDF) เป็นต้น ด้วยเหตุนี้สำนักงานพัฒนาธุรกรรมทางอิเล็กทรอนิกส์ (องค์การมหาชน) (สพธอ.) จึงได้ดำเนินการพัฒนาเครื่องมือหรือ TeDA Sign API ขึ้น เพื่อช่วยให้นักพัฒนาแอปพลิเคชันสามารถทำงานได้สะดวกรวดเร็ว และสอดคล้องตามแนวทางสากล

ความสามารถของ TeDA Sign API

1. การสร้างลงลายมือชื่ออิเล็กทรอนิกส์ในรูปแบบ PKCS#1, PKCS#7 และการลงลายมือชื่อบนเอกสาร PDF
2. การตรวจสอบความถูกต้องน่าเชื่อถือของลายมือชื่ออิเล็กทรอนิกส์ รวมถึงการตรวจสอบความถูกต้องของใบรับรองอิเล็กทรอนิกส์ ตามกระบวนการที่ประกาศไว้ใน IETF RFC 5280

คุณสมบัติเด่น

1. **ใช้งานง่ายและลดเวลาในการพัฒนาซอฟต์แวร์**

   TeDA Sign API จะช่วยให้ผู้พัฒนาซอฟต์แวร์มีความสะดวก รวมถึงลดเวลาที่ใช้ศึกษาและพัฒนาซอฟต์แวร์สำหรับสร้างและตรวจสอบลายมือชื่ออิเล็กทรอนิกส์ เนื่องจาก TeDA Sign API ได้นำกระบวนการที่มีความยุ่งยากซับซ้อนมารวมไว้เป็น Method ของ TeDA Sign API ทำให้ผู้พัฒนาซอฟต์แวร์ไม่ศึกษากระบวนการ PKI เชิงลึกและเรียนรู้ฟังก์ชั่นต่างๆ เพื่อสร้างและตรวจสอบลายมือชื่ออิเล็กทรอนิกส์ ซึ่งมาตรฐานที่ TeDA Sign API

2. **สอดคล้องกับแนวทางสากล**

   TeDA Sign API ได้รับการพัฒนาขึ้นโดยอ้างอิงแนวทางที่ระบุอยู่ในเอกสาร Internet Engineering Task Force (IETF) RFC อันเป็นที่ยอมรับในระดับสากล ทำให้เชื่อมั่นได้ว่าลายมือชื่อ

อิเล็กทรอนิกส์ที่สร้างโดย TeDA Sign API มีความน่าเชื่อถือตามแนวทางที่ทั่วโลกยอมรับซึ่งประกอบด้วย

- **Electronic Signature:** ลายมือชื่ออิเล็กทรอนิกส์ที่สร้างโดย TeDA Sign API เป็นลายมือชื่อที่สอดคล้องกับมาตรฐาน Public-Key Cryptography Standards (PKCS) ซึ่งก็คือ PKCS#1 (ปัจจุบันประกาศเป็น IETF RFC3447 Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1) และ PKCS#7 (ปัจจุบันประกาศเป็น IETF RFC 2315 PKCS #7: Cryptographic Message Syntax Version 1.5) นอกจากนี้ TeDA Sign API ยังรองรับการลงลายมือชื่อบนเอกสาร PDF ซึ่งปัจจุบัน PDF ได้รับการประกาศเป็นมาตรฐาน ISO 32000:1:2008
- **Certificate :** ใบรับรองอิเล็กทรอนิกส์ที่ TeDA Sign API รองรับเป็นใบรับรองอิเล็กทรอนิกส์ที่มีข้อมูล (Certificate Profile) สอดคล้องกับเนื้อหาที่ระบุใน IETF RFC 5280 X.509 PKI Certificate
- **Certificate Validation:** ในการตรวจสอบความน่าเชื่อถือของใบรับรองอิเล็กทรอนิกส์ ก่อนนำไปใช้ในการตรวจสอบลายมือชื่ออิเล็กทรอนิกส์นั้น TeDA Sign API ได้พัฒนาให้สอดคล้องกับกระบวนการที่กำหนดไว้ใน IETF RFC 5280 X.509 PKI Certificate

3. **รองรับการทำงานงานกับ Certificate Store หลายรูปแบบ**

เพื่อรองรับการทำงานร่วมกับระบบปฏิบัติการหรือสภาพแวดล้อมที่ติดตั้งแอปพลิเคชั่นที่หลากหลาย TeDA Sign API รองรับการทำงานร่วมกับฐานข้อมูลใบรับรองอิเล็กทรอนิกส์ (Certificate Store) ทั้ง Microsoft Certificate Store, Java Key Store, PKCS#7 Cryptographic Message Syntax Standard และ PKCS#12 Personal Information Exchange Syntax Standard

หนังสือเล่มนี้เหมาะกับใคร

เนื่องจากการลงลายมือชื่อด้วยเทคโนโลยี PKI มีความซับซ้อนและมีการใช้งานในหลายรูปแบบ เช่น PKCS#1 PKCS#7 หรือ Portable Document Format (PDF) เป็นต้น ด้วยเหตุนี้หนังสือคู่มือการใช้งาน TeDA Sign API ฉบับเรียนเล่นเป็นเร็วเล่มนี้จึงเหมาะสำหรับเจ้าหน้าที่ระดับปฏิบัติการ หรือนักพัฒนาซอฟต์แวร์ในองค์กรที่ต้องการประยุกต์ใช้ TeDA Sign API ในการลงลายมือชื่ออิเล็กทรอนิกส์บนเอกสารต่างๆ เช่น เอกสารกฎหมาย ใบเสนอราคา ใบกำกับภาษี เป็นต้น

บทที่ 3

การประยุกต์ใช้ TeDA Sign API ในแอปพลิเคชั่นขององค์กร

บริการ TeDA Time นำเสนอ API Toolkit ในภาษา Java ที่ช่วยให้นักพัฒนาซอฟต์แวร์ผนวกความสามารถของลายมือชื่ออิเล็กทรอนิกส์เข้ากับแอพพลิเคชันทางธุรกิจที่มีอยู่ในองค์กร หรือสร้างโปรแกรมใหม่ที่สามารถลงลายมือชื่อบนไฟล์เอกสารทุกฟอร์แมตได้โดยอัตโนมัติ

คุณสมบัติที่นักพัฒนาฯ พึงเรียกใช้จาก API Toolkit ได้แก่

- การสร้างลงลายมือชื่ออิเล็กทรอนิกส์ในรูปแบบ PKCS#1, PKCS#7 และการลงลายมือชื่อบนเอกสาร PDF
- การตรวจสอบความถูกต้องน่าเชื่อถือของลายมือชื่ออิเล็กทรอนิกส์ ตามกระบวนการที่ประกาศไว้ใน IETF RFC 5280

สำหรับเนื้อหาในบทนี้จะเป็นการอธิบายคลาสและเมธอด รายการ Error Code รวมทั้งนำเสนอตัวอย่างซอร์สโค้ดสำหรับประยุกต์ใช้งาน TeDA Sign API และแนวทางตรวจสอบความถูกต้องน่าเชื่อถือลายมือชื่ออิเล็กทรอนิกส์

ความต้องการทางระบบพื้นฐาน

- ระบบปฏิบัติการ Windows หรือ Mac OS
- Java Runtime Environment (JRE) และ Java Development Kit (JDK) เวอร์ชั่น 1.6 ขึ้นไป
- TeDA Sign API เวอร์ชั่น 1.0.0
- Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files

คลาสและเมธอด

1. PKCS1Signature: th.or.etda.teda.RFC3447.PKCS1Signature

Description : Class สำหรับสร้าง RSA Encryption ตาม RFC3447

Method :

1.1) Byte[] **sign** (

PrivateKey privateKey,	PrivateKey ที่ต้องการใช้ในการทำ Digital Signature
File inputFile,	ไฟล์ที่ต้องการทำ Digital Signature
String signingAlgorithm	Algorithm ที่จะใช้ในการทำ Digital Signature

)

1.2) Byte[] **sign** (

PrivateKey privateKey,	PrivateKey ที่ต้องการใช้ในการทำ Digital Signature
byte[] messageDigest,	ผลการทำ Hash ของ ไฟล์ที่จะทำ Digital Signature
String signingAlgorithm	Algorithm ที่จะใช้ในการทำ Digital Signature

)

1.3) void **verify** (

PublicKey publicKey,	PublicKey ที่จะใช้ตรวจสอบ SignatureFile
String b64SignatureValue,	ไฟล์ DigitalSignature ที่ต้องการจะตรวจสอบ
File verificationFile,	ไฟล์ที่ต้องการตรวจเทียบกันกับ Digital Signature
String signingAlgorithm	Algorithm ที่ใช้ในการทำ Digital Signature

)

1.4) void **verify** (

X509Certificate certificate,	Certificate ที่จะใช้ตรวจสอบ SignatureFile
String b64SignatureValue,	ไฟล์ DigitalSignature ที่ต้องการจะตรวจสอบ
File verificationFile,	ไฟล์ที่ต้องการตรวจเทียบกันกับ Digital Signature
String signingAlgorithm	Algorithm ที่ใช้ในการทำ Digital Signature

)

1.5) void **verify** (

PublicKey publicKey,	PublicKey ที่จะใช้ตรวจสอบ SignatureFile
byte[] signatureValue,	DigitalSignature ที่ต้องการจะตรวจสอบ
byte[] messageDigest,	ผลการทำ Hash ของ ไฟล์ที่ต้องการตรวจเทียบกันกับ Digital Signature
String signingAlgorithm	Algorithm ที่ใช้ในการทำ Digital Signature

)

1.6) void **verify** (

X509Certificate certificate,	Certificate ที่จะใช้ตรวจสอบ SignatureFile
byte[] signatureValue,	DigitalSignature ที่ต้องการจะตรวจสอบ
byte[] messageDigest,	ผลการทำ Hash ของ ไฟล์ที่ต้องการตรวจเทียบกันกับ Digital Signature
String signingAlgorithm,	Algorithm ที่ใช้ในการทำ Digital Signature

)

2. PKCS7Signature: th.or.etda.teda.RFC2315.PKCS7Signature

Description : Class สำหรับสร้าง RSA Encryption ตาม RFC2315

Method :

2.1) Byte[] **sign (**

PrivateKey privateKey,	PrivateKey ที่ต้องการใช้ในการทำ Digital Signature
X509Certificate[] chain,	Certificate Chain สำหรับทำ Digital Signature
File inputFile,	ไฟล์ที่ต้องการทำ Digital Signature
String signingAlgorithm	Algorithm ที่จะใช้ในการทำ Digital Signature

)

2.2) Byte[] **sign** (

PrivateKey privateKey,	PrivateKey ที่ต้องการใช้ในการทำ Digital Signature
X509Certificate[] chain,	Certificate Chain สำหรับทำ Digital Signature
byte[] messageDigest,	ผลการทำ Hash ของ ไฟล์ที่จะทำ Digital Signature
String signingAlgorithm	Algorithm ที่จะใช้ในการทำ Digital Signature

)

2.3) void **verify** (

File pkcs7File,	File PKCS#7 ที่ต้องการจะทำการตรวจสอบ
File verificationFile,	ไฟล์ที่ต้องการตรวจเทียบกันกับ Digital Signature
Date signDate	วันที่สร้าง PKCS#7 นี้ขึ้น

)

2.4) void **verify** (

PKCS7 pkcs7File,	PKCS#7 ที่ต้องการจะทำการตรวจสอบ
byte[] messageDigest,	ผลการทำ Hash ของ ไฟล์ที่ต้องการตรวจเทียบกันกับ Digital Signature
Date signDate	วันที่สร้าง PKCS#7 นี้ขึ้น

)

2.5) void **verify** (

PKCS7 pkcs7File,	PKCS#7 ที่ต้องการจะทำการตรวจสอบ
byte[] messageDigest,	ผลการทำ Hash ของ ไฟล์ที่ต้องการตรวจเทียบกันกับ Digital Signature
Date signDate,	วันที่สร้าง PKCS#7 นี้ขึ้น
File trustedKS	Key Store ที่ใช้เก็บ Trusted Root Certificate (Optional)
String password,	Password ของ Key Store ที่ใช้เก็บ Trusted Root Certificate (Optional)
String storeType,	ประเภทของ Certificate Store ได้แก่ JKS, PKCS7, PKCS12
X509CRL[] crls	CRL ที่จะใช้ประกอบการตรวจสอบ PKCS7 นี้

)

3. PAdES: th.or.etda.teda.ETSITS102778.PAdES

Description : Class สำหรับลงลายมือชื่อดิจิทัลบน PDF

Method :

3.1) String **sign** (

PrivateKey privateKey,	PrivateKey ที่ต้องการใช้ในการทำ Digital Signature
X509Certificate[] chain,	Certificate Chain สำหรับทำ Digital Signature
File inputFile,	ไฟล์ PDF Document ที่ต้องการทำ Digital Signature
File outputFile,	Path ของไฟล์ที่จะเกิดหลังจากทำ Digital Signature
String signingAlgorithm,	Algorithm ที่จะใช้ในการทำ Digital Signature
String signatureFieldName (Optional)	ชื่อของ Signature Field บน PDF ที่จะทำการ Sign

)

3.2) String **sign** (

PrivateKey privateKey,	PrivateKey ที่ต้องการใช้ในการทำ Digital Signature
X509Certificate[] chain,	Certificate Chain สำหรับทำ Digital Signature
InputStream inputStream,	Stream ของ Input PDF ที่ต้องการทำ Digital Signature
OutputStream outputStream,	Stream ของ PDF ที่ลงลายมือชื่อสำเร็จ
String signingAlgorithm,	Algorithm ที่จะใช้ในการทำ Digital Signature
String signatureFieldName, (Optional)	ชื่อของ Signature Field บน PDF ที่จะทำการ Sign
String tsaURL,	URL ของ TSA Server ที่จะใช้ในการทำ Timestamp
String tsaUsername,	Username ที่สำหรับเข้าใช้งาน TSA Service
File tsaPassword,	Passowrd ที่สำหรับเข้าใช้งาน TSA Service
String tsaPolicy	Policy OID ของ TSA Service ที่จะเรียกใช้งาน

)

3.3) String **sign** (

PrivateKey privateKey, PrivateKey ที่ต้องการใช้ในการทำ Digital Signature

X509Certificate[] chain, Certificate Chain สำหรับทำ Digital Signature

InputStream inputStream, Stream *ของ* Input PDF ที่ต้องการทำ Digital Signature

OutputStream outputStream, Stream ของ PDF ที่ลงลายมือชื่อสำเร็จ

String signingAlgorithm, Algorithm ที่จะใช้ในการทำ Digital Signature

PAdESSignAppearance signatureAppearance Signature Appearance Parameter *ที่ต้องการให้เกิดขึ้นกับการทำ* Digital Signature นี้

String tsaURL, URL ของ TSA Server ที่จะใช้ในการทำ Timestamp

String tsaUsername, Username ที่สำหรับเข้าใช้งาน TSA Service

File tsaPassword, Passowrd ที่สำหรับเข้าใช้งาน TSA Service

String tsaPolicy Policy OID ของ TSA Service ที่จะเรียกใช้งาน

)

3.4) void **verify** (

File verificationFile ไฟล์ PDF Document ที่ต้องการทำการ Verify

)

3.5) void **verify** (

InputStream inputStream, Stream ของ PDF ที่ต้องการตรวจสอบ

String signatureFieldName ชื่อของ Signature Field บน PDF ที่จะทำการ verify

File trustedKS Key Store ที่ใช้เก็บ Trusted Root Certificate (Optional)

String password, Password ของ Key Store ที่ใช้เก็บ Trusted Root Certificate (Optional)

String storeType ประเภทของ Certificate Store ได้แก่ JKS, PKCS7, PKCS12

)

Error Code

ต่อไปคือ Error Code ของเมธอด ตามที่ระบุไว้ใน th.teda.ErrorCode

Error Code	Alias	Description
1	DYNAMIC_XFA	ไฟล์อินพุตมีการใช้คุณสมบัติ Dynamic XFA ซึ่งเมธอดที่เรียกใช้งานอยู่ไม่รองรับ
2	FIELD_EXISTS	ฟิลด์ที่ระบุชื่อนั้นๆ มีอยู่จริง
3	FIELD_NOT_EXISTS	ฟิลด์ที่ระบุชื่อนั้นๆ ไม่มีอยู่จริง
4	SIGNATURE_ERROR	ค้นหาลายมือชื่อดิจิทัลไม่พบ หรือลายมือชื่อดิจิทัลเสียหาย
5	TIMESTAMP_MISSING	ไม่พบ Time-Stamp Token
6	CDP_PARSING	ไม่พบค่า CDP Extension หรือมีรูปแบบที่ผิดเพี้ยน

7	CRL_RETRIEVAL	เกิดข้อผิดพลาดในระหว่างกระบวนการดาวน์โหลด CRL
8	FILE_NOT_FOUND	หาไฟล์ไม่พบ หรือมีโปรแกรมอื่นกำลังใช้ไฟล์อยู่
9	UNKNOWN_HOST	หาเครื่องโฮสต์ไม่พบ หรือมีปัญหาด้านการเชื่อมต่อ
10	INVALID_PDF	ไฟล์อินพุตไม่ใช่เอกสาร PDF ที่มีรูปแบบถูกต้อง
11	MALFORMED_CDP_URL	URL ที่ระบุตัว CDP มีรูปแบบไม่ถูกต้อง
12	CRL_PARSING	กระบวนการ Parsing ของ CRL เกิดข้อผิดพลาด
13	PDF_FILE	ไฟล์นี้เป็นเอกสาร PDF ที่เข้ากันไม่ได้กับเมธอดที่เรียกใช้งาน
14	TOKEN_GENERATION	เกิดข้อผิดพลาดในกระบวนการสร้าง Time-Stamp Token
15	NO_SUCH_ALGORITHM	ไม่มีอัลกอริธึม Message Digest ตามที่ระบุ
16	FILE_READ	เกิดข้อผิดพลาดในการอ่านไฟล์

ตัวอย่างซอร์สโค้ด

1. PKCS1Signature

1.1) Sign

```
public void SignTheDocument() throws Exception{
	File keyPair = new File("path_to_file/KeyStore.p12");
	PrivateKey privateKey = null;

	try {
		// Initialize key store regard to keystoreType
		char[] pass = "password".toCharArray();

		// Load key store
		KeyStore ks = KeyStore.getInstance("pkcs12");
		ks.load(new FileInputStream(keyPair),pass);
		Enumeration<String> aliases = ks.aliases();
		while (aliases.hasMoreElements()){
			// Fetch accessing name from specified key store
			String alies = aliases.nextElement();
			if (ks.getKey(alies, pass)!=null) privateKey = (PrivateKey) ks.getKey(alies, pass);
		}
	} catch (Exception e) {
		e.printStackTrace();
	}

	byte[] sigVal = PKCS1Signature.sign(privateKey, new File("pathToContent/sample.txt"), "sha256withrsa");

	System.out.println("Complete");
	System.out.println("Signature Value = ".concat(new String(Base64.encode(sigVal))));
}
```

2. PKCS7Signature

2.1) Sign

```
public void SignTheDocument() {
		File inputFile = new File("path_to_file/file_to_sign.pdf");
		PrivateKey privateKey = null;

		try {
			String keystore_location = "path_to_file/KeyStore.p12";
			String password = "*******";
			KeyStore ks = KeyStore.getInstance("PKCS12");
			ks.load(new FileInputStream(keystore_location), password.toCharArray());

			String certificate_aliese = "prefered_cer";
			String certificate_password = "*******";
			privateKey = (PrivateKey) ks.getKey(certificate_aliese, certificate_password.toCharArray());
			String store_type = X509CertificateUtility.STORE_TYPE_PKCS12;
			X509Certificate[] certificate_chain = X509CertificateUtility.fetchCertificateFromKeyStore(
				new File(keystore_location), password, store_type);
			String algorithm = "sha256withrsa";

			byte[] pkcs7 = PKCS7Signature.sign(privateKey, certificate_chain, inputFile, algorithm);
		} catch (Exception e) {
			e.printStackTrace();
		}
}
```

2.2) verify

```
public void VerifyTheP7BSignature(){
		try {
			File pkcs7File = new File("path_to_file/signature.p7b");
			File verificationFile = new File("path_to_file/file_to_verified.pdf");
			PKCS7 pkcs7 = new PKCS7(ObjectReaderUtility.readFile(pkcs7File));

			byte[] messageDigest = MessageDigestUtility.digestFile(pkcs7.getDigestAlgorithmIds()[0].toString(), verificationFile);
			Date signDate = new Date();

			String store_location = "path_to_file/certificate_store.jks";
			File trustedKS = new File(store_location);
			String password = "******";
			String storeType = X509CertificateUtility.STORE_TYPE_JKS;

			X509Certificate[] chain = X509CertificateUtility.fetchCertificateFromKeyStore(trustedKS, password, storeType);
			X509CRL[] crls = CRLDistributionPoint.getCRLS(chain);
			PKCS7Signature.verify(pkcs7, messageDigest, signDate, trustedKS, password, storeType, crls);
		} catch (Exception e) {
			e.printStackTrace();
		}
}
```

3. PAdES

3.1) sign

```
public void SignTheDocument(){
	File inputFile = new File("path_to_file/file_to_sign.pdf");
	File outputFile = new File("path_to_file/signed_file.pdf");

	PrivateKey privateKey = null;
	try {
		String keystore_location = "path_to_file/KeyStore.p12";
		String password = "*******";
		KeyStore ks = KeyStore.getInstance("PKCS12");
		ks.load(new FileInputStream(keystore_location)
			,password.toCharArray());

		String certificate_aliese = "prefered_cer";
		String certificate_password = "*******";
		privateKey = (PrivateKey) ks.getKey(certificate_aliese, certificate_password.toCharArray());
		String store_type = X509CertificateUtility.STORE_TYPE_PKCS12;
		X509Certificate[] certificate_chain = X509CertificateUtility.fetchCertificateFromKeyStore (
			new File(keystore_location), password, store_type);
		String algorithm = "sha256withrsa";
		PAdES.sign(privateKey, certificate_chain, inputFile, outputFile, algorithm, null);
	} catch (Exception e) {
		e.printStackTrace();
	}
}
```

3.2) verify

```
public void VerifyTheSignedDoc(){
	try {
		File verificationFile = new File("path_to_file/signed_file.pdf");
		InputStream inputStream = new FileInputStream(verificationFile);

		String signatureField = "signature_field_in_pdf";
		String store_location = "path_to_file/certificate_store.jks";
		File trustedKS = new File(store_location);
		String password = "******";
		String storeType = X509CertificateUtility.STORE_TYPE_JKS;

		PAdES.verify(inputStream, signatureField, trustedKS, password, storeType);
	} catch (Exception e) {
		e.printStackTrace();
	}
}
```

บทที่ 4
อภิธานศัพท์และคำย่อ

คำศัพท์	ความหมาย
ASN.1 (Abstract Syntax Notation.One)	ภาษามาตรฐานสำหรับนิยามโครงสร้างข้อมูลที่ใช้สื่อสารข้ามฮาร์ดแวร์อินเทอร์เฟซ กำหนดโดย องค์การระหว่างประเทศว่าด้วยการมาตรฐาน (The International Organization for Standardization: ISO)
CA (Certification Authority)	หน่วยงานที่ออกใบรับรองอิเล็กทรอนิกส์
CMS (Cryptographic Message Syntax)	มาตรฐานการเข้ารหัสลับข้อความที่กำหนดโดย IETF ซึ่งทำงานโดยใช้ไวยากรณ์ของ PKCS#7
ETSI (European Telecommunication Standards Institute)	องค์กรที่ทำหน้าที่กำหนดมาตรฐานด้านโทรคมนาคมในยุโรป
FIPS (Federal Information Processing Standard)	มาตรฐานที่รัฐบาลสหรัฐอเมริกาประกาศ เพื่อให้เป็นมาตรฐานที่เกี่ยวข้องกับระบบคอมพิวเตอร์สำหรับหน่วยงานภาครัฐที่ไม่ใช่หน่วยงานทหาร และสำหรับคู่สัญญาที่ทำงานเกี่ยวกับองค์กรรัฐดังกล่าวด้วย
HSM (Hardware Security Module)	อุปกรณ์ที่ทำหน้าที่บริหารจัดการกุญแจดิจิทัล และเพิ่มประสิทธิภาพในการเข้ารหัส มีวัตถุประสงค์เพื่อ: 1. เพิ่มความมั่นคงปลอดภัยในการสร้างกุญแจ 2. เพิ่มความมั่นคงปลอดภัยในการจัดเก็บกุญแจ 3. ใช้เพื่อการเข้ารหัสข้อมูลที่มีความอ่อนไหวสูง 4. เพื่อแบ่งเบาภาระในการเข้าและถอดรหัสจากเครื่องแอพพลิเคชั่นเซิร์ฟเวอร์ เช่น TSA Server
IETF (Internet Engineering Task Force)	หน่วยงานที่พัฒนามาตรฐานโพรโตคอลสำหรับการทำงานของระบบเครือข่ายอินเทอร์เน็ต
ISO (International Standard Organization)	องค์การมาตรฐานสากล ที่ทำหน้าที่ออกมาตรฐานต่างๆ ที่เกี่ยวข้องกับธุรกิจและอุตสาหกรรม ประกอบด้วยสมาชิกจากหลายประเทศๆ ทั่วโลก
LTANS (Long-Term Archive and Notary Service)	บริการจัดเก็บและรับรองเอกสารทางอิเล็กทรอนิกส์ให้ใช้ได้ในระยะยาว
LTV (Long-Term Validation)	ความสามารถในการตรวจสอบความถูกต้องของลายมือชื่ออิเล็กทรอนิกส์ได้ไม่ว่าเวลาจะผ่านไปนานเท่าใด ไม่ว่าในภายภาคหน้าเอกสารรับรองความถูกต้องจะหมดอายุหรือสูญหาย
Nonce (Number Used Once)	ค่าที่ได้จากการสุ่มสร้างขึ้นมาสำหรับใช้ในกระบวนการยืนยันตัวตน โดยมีจุดประสงค์เพื่อป้องกันการโจมตีด้วยวิธีการส่งข้อมูลซ้ำ (Replay Attack)
NSA (National Security Agency)	สำนักงานความมั่นคงแห่งชาติของสหรัฐอเมริกา
NTP (Network Time	โพรโตคอลสำหรับปรับเทียบเวลาของระบบคอมพิวเตอร์ผ่านระบบเครือข่าย

Protocol)	ประกาศโดย IETF
OCSP (Online Certificate Status Protocol)	โพรโตคอลสำหรับตรวจสอบสถานะการเพิกถอนหรือยกเลิกใบรับรองอิเล็กทรอนิกส์ ประกาศโดย IETF
OID (Object Identifier)	ชุดตัวเลขที่บ่งบอกถึงความเป็นหนึ่งเดียวของออปเจ็กต์นั้น
PAdES (PDF Advanced Electronic Signature)	มาตรฐานในการลงลายมือชื่ออิเล็กทรอนิกส์บนเอกสาร PDF ประกาศโดย ETSI
PDF (Portable Data Format)	รูปแบบไฟล์เอกสารประเภทหนึ่งที่คิดค้นโดยบริษัท Adobe Systems
PKI (Public Key Infrastructure)	โครงสร้างพื้นฐานรหัสกุญแจสาธารณะ
PKCS (Public Key Cryptography Standard)	มาตรฐานเข้ารหัสลับที่ประกาศโดยบริษัท RSA Security มีหลายตัว เช่น PKCS#7 ใช้สำหรับการเข้ารหัสลับข้อความด้วย PKI และใช้กับการส่งใบรับรองอิเล็กทรอนิกส์ ซึ่งเป็นหลักการตั้งต้นของ S/MIME และมักใช้กับระบบ Single Sign On
RFC (Request For Comments)	เอกสารทางด้านเทคนิคหรือบันทึกที่เกี่ยวของกับการทำงานของระบบอินเทอร์เน็ตเช่น Protocol, Procedure, Program หรือบันทึกการประชุม ซึ่งบางส่วนจะได้รับการคัดเลือกเป็นมาตรฐานในที่สุด
RSA (Rivest, Shamir, Adleman)	อัลกอริธึมสำหรับการเข้ารหัสลับโดยใช้ Asymmetric Key โดยอาศัยแนวคิดการแยกตัวประกอบของเลขจำนวนเต็มขนาดใหญ่มากๆ จนไม่สามารถทำได้หากไม่รู้จำนวนเฉพาะที่เป็นส่วนหนึ่งของตัวประกอบนั้นเสียก่อน ทั้งนี้ เป็นอัลกอริธึมที่คิดค้นโดย Ron Rivest, Adi Shamir และ Leonard Adleman เมื่อปี ค.ศ. 1977
SHA (Secure Hash Algorithm)	อัลกอริธึมการเข้ารหัสข้อความหรือไฟล์ โดยอาศัยการคำนวณทางคณิตศาสตร์ที่พัฒนาโดย NSA
TeDA (Trusted e-Document Authority)	โครงการวิจัยที่พัฒนาโดย สำนักงานพัฒนาธุรกรรมทางอิเล็กทรอนิกส์ (องค์การมหาชน) หรือ สพธอ. เพื่อเป็นต้นแบบในการจัดเก็บเอกสารอิเล็กทรอนิกส์แบบถาวรที่มีความน่าเชื่อถือในประเทศไทย
TSA (Time-Stamping Authority)	ผู้ให้บริการซึ่งมีหน้าที่สร้างข้อมูลการประทับรับรองเวลาอิเล็กทรอนิกส์
TSP (Time-Stamping Protocol)	โพรโตคอลสำหรับติดต่อสื่อสารกับ TSA
TST (Time-Stamp Token)	ข้อมูลชุดหนึ่งที่แสดงความสัมพันธ์ระหว่างข้อมูลต้นฉบับกับเวลา ณ ขณะนั้น เพื่อใช้เป็นหลักฐานได้ว่าข้อมูลต้นฉบับนั้นมีอยู่ก่อนเวลาที่ระบุไว้
X.509 Certificate	ใบรับรองอิเล็กทรอนิกส์ที่มีรูปแบบตามมาตรฐาน X.509 ของ IETF ซึ่งมีข้อมูลบ่งบอกถึงบุคคลหรือหน่วยงานใดๆ ว่าเป็นเจ้าของกุญแจสาธารณะตัวจริง