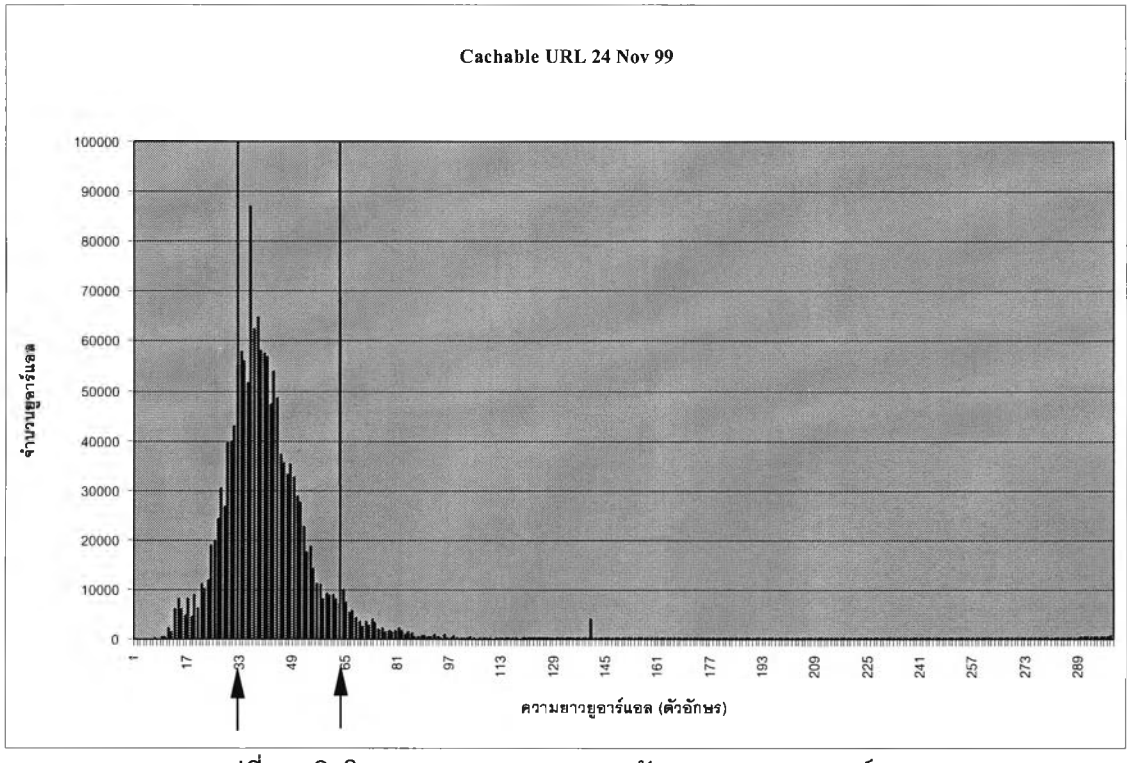
บทที่ 5
วิเคราะห์ผลการทดลอง

บทนี้จะเป็นวิเคราะห์ผลการทดลองที่แสดงในบทที่ 4 โดยทำการวิเคราะห์แต่ละพารามิเตอร์อย่างอิสระเพื่อให้เห็นความแตกต่างของแต่ละอัลกอริทึมอย่างชัดเจน จากนั้นจะนำแต่ละพารามิเตอร์มาแสดงความสัมพันธ์โดยพล็อตลงในกราฟเดียวกัน เพื่อใช้วิเคราะห์พิจารณาความเหมาะสมในการนำไปใช้งาน

5.1 ความยาวและลักษณะของยูอาร์แอล

จากตารางที่ 4.1 ในบทที่ 4 จะพบว่าจำนวนยูอาร์แอลที่สามารถแคชได้และไม่ซ้ำ แปรผันตรงกับจำนวนยูอาร์แอลที่สามารถแคชได้ และจำนวนยูอาร์แอลทั้งหมดในแต่ละวัน กล่าวคือถ้าหากจำนวนยูอาร์แอลมีจำนวนมากขึ้น จำนวนยูอาร์แอลที่สามารถแคชได้ และจำนวนยูอาร์แอลที่สามารถแคชได้และไม่ซ้ำ จะมีจำนวนมากขึ้นตามไปด้วย



รูปที่ 5.1 ฮิสโตแกรมแสดงการกระจายตัวของความยาวยูอาร์แอล

สำหรับค่าความยาวเฉลี่ยของยูอาร์แอล และค่าความยาวที่ยูอาร์แอลมีความยาวนั้นมากที่สุดแตกต่างกันอยู่เล็กน้อย โดยค่าความยาวยูอาร์แอลเฉลี่ยแต่ละวันอยู่ที่ประมาณ 49 ตัวอักษร และความยาวยูอาร์แอลที่มีจำนวนยูอาร์แอลซ้ำกันมากที่สุด คือ 44 ตัวอักษร และจากการพิจารณาฮิสโตแกรมของความยาวยูอาร์แอลของข้อมูลการเรียกขอของวันที่ 24 พ.ย. 2542 (ดังรูปที่ 5.1) จะพบ

ว่า ความยาวยูอาร์แอลจะกระจุกตัวอยู่ในช่วง 33.5 ถึง 64.44 ตัวอักษร (ใช้ค่าเฉลี่ย +/- ค่าเบี่ยงเบนมาตรฐาน ถ้าหากพิจารณาทั้ง 5 วัน พบว่าอยู่ในช่วง 32 - 68 ตัวอักษร) แสดงให้เห็นว่าความยาวยูอาร์แอลที่สามารถแคชได้มีค่าความยาวเพียงช่วงหนึ่งเท่านั้น ยูอาร์แอลที่มีความยาวต่ำกว่าหรือสูงกว่าในช่วงดังกล่าวมีปริมาณน้อยมาก และเนื่องจากยูอาร์แอลจะต้องขึ้นต้นด้วยสตริง "http://" หรือ "ftp://" ซึ่งมีความยาว 6-7 ตัวอักษร ดังนั้นยูอาร์แอลที่มีความยาวน้อยกว่า 6 ตัวอักษรจึงน่าจะเป็นความผิดพลาดในการเรียกขอ นอกจากนั้นลักษณะการกระจายตัวของความยาวยูอาร์แอลดังกล่าวยังสนับสนุนสมติฐานที่ว่ายูอาร์แอลที่ถูกเรียกขอมักจะเป็นยูอาร์แอลของเว็บที่ได้รับความนิยมอีกด้วย

จากค่าความยาวโดยทั่วไปของยูอาร์แอลที่ 44 ตัวอักษร จะนำไปใช้ในการกำหนดความยาวของรหัสในอัลกอริทึมที่ต้องมีการกำหนดความยาวรหัส โดยข้อกำหนดที่ว่า รหัสที่กำหนดขึ้นจะมีความยาวน้อยกว่าความยาวโดยทั่วไปของยูอาร์แอล อย่างน้อยประมาณ 50%

5.2 ความเร็วในการเข้ารหัส

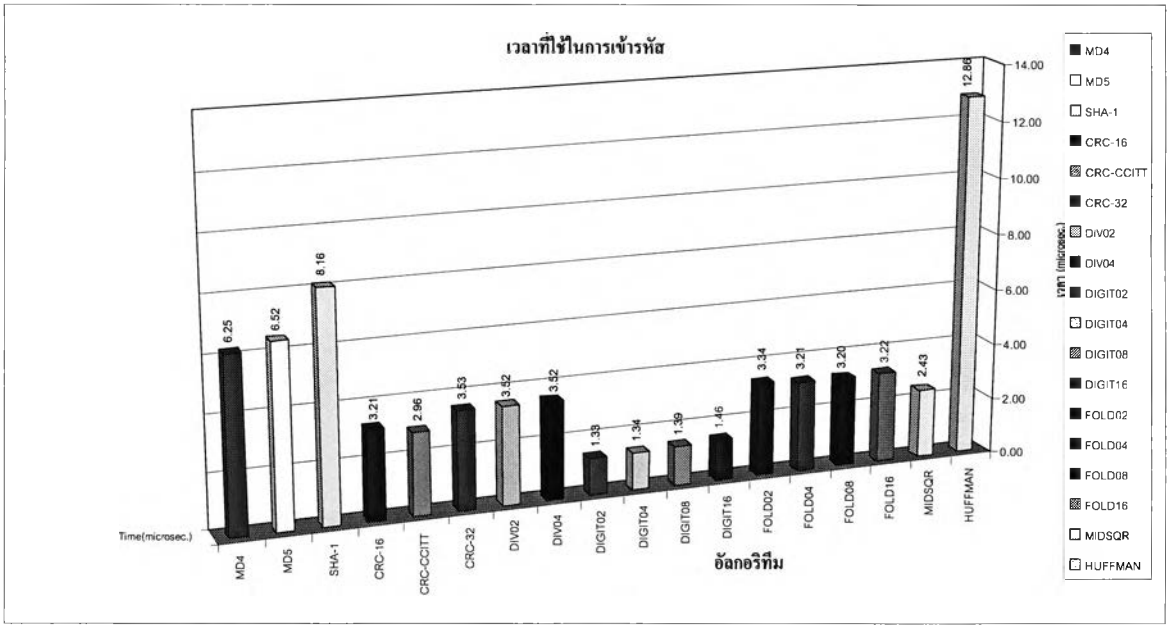
ในการพิจารณาเวลาที่ใช้ในการเข้ารหัสในแต่ละอัลกอริทึมจากตารางแสดงผลการทดลองในบทที่ 4 เพื่อให้เห็นภาพชัดเจนยิ่งขึ้นจะนำผลการทดลองซึ่งเป็นเวลาเฉลี่ยที่ใช้ต่อยูอาร์แอลหนึ่งยูอาร์แอลของแต่ละอัลกอริทึมมาเฉลี่ยอีกครั้ง (แสดงในตารางที่ 5.1) จากนั้นนำมาพล็อตกราฟเพื่อใช้เปรียบเทียบความเร็วของแต่ละอัลกอริทึมดังรูปที่ 5.2 โดยตัด MD2 ออกไม่แสดงผลในกราฟ เนื่องจาก MD2 ถูกออกแบบมาให้ใช้งานกับเครื่องคอมพิวเตอร์ที่มีสถาปัตยกรรม 8 บิต ทำให้ทำงานได้ช้ามากเมื่อเทียบกับอัลกอริทึมอื่นๆ ถ้าหากนำมาพิจารณาในกราฟเดียวกันจะเห็นความแตกต่างของอัลกอริทึมอื่นๆไม่ชัดเจน

ตารางที่ 5.1 ตารางแสดงการเปรียบเทียบเวลาที่ใช้ในการเข้ารหัสแต่ละอัลกอริทึม

กลุ่มอัลกอริทึม	อัลกอริทึม	เวลาที่ใช้ในการเข้ารหัสยูอาร์แอล 1 ยูอาร์แอล (ไมโครวินาที)	กลุ่มอัลกอริทึม	อัลกอริทึม	เวลาที่ใช้ในการเข้ารหัสยูอาร์แอล 1 ยูอาร์แอล (ไมโครวินาที)
MD	MD2	165.58	Simple Hash	DIGIT02	1.33
MD	MD4	6.25	Simple Hash	DIGIT04	1.34
MD	MD5	6.52	Simple Hash	DIGIT08	1.39
MD	SHA-1	8.16	Simple Hash	DIGIT16	1.46
CRC	CRC-16	3.721	Simple Hash	FOLD02	3.34
CRC	CRC-CCITT	2.96	Simple Hash	FOLD04	3.21
CRC	CRC-32	3.53	Simple Hash	FOLD08	3.20
Simple Hash	DIV02	3.52	Simple Hash	FOLD16	3.22
Simple Hash	DIV04	3.52	Simple Hash	MIDSQR	2.43
			Huffman	HUFFMAN	12.86

จากตารางที่ 5.1 และรูปที่ 5.2 จะพบว่า อัลกอริทึมที่ใช้เวลาในการเข้ารหัสต่อ 1 ยูอาร์แอลน้อยที่สุดคือ อัลกอริทึม Digit Analysis ทุกความยาวรหัส อัลกอริทึมที่ใช้เวลาในการเข้ารหัสต่อ 1 ยูอาร์แอลมากที่สุดคือ Huffman Coding และถ้าหากเรียงอัลกอริทึมตามลำดับเวลาที่ใช้ในการเข้ารหัสยูอาร์แอลจากน้อยไปมากได้ดังนี้ คือ Digit Analysis, Midsquare, CRC-CCITT, Folding 8 bytes, Folding 4 bytes, CRC-16, Folding 16 bytes, Folding 2 bytes, 4 bytes Division, 2 bytes Division, CRC-32, MD4, MD5, SHA-1 และ Huffman Coding ตามลำดับ

ถ้าหากพิจารณาตามกลุ่มของอัลกอริทึม พบว่ากลุ่มอัลกอริทึม Simple Hash Function และกลุ่ม CRC จะใช้เวลาในการเข้ารหัสต่อยูอาร์แอลค่อนข้างน้อย ส่วน Huffman และการเข้ารหัสในกลุ่ม MD นั้นใช้เวลาในการเข้ารหัสต่อยูอาร์แอลค่อนข้างมาก ผลการทดลองดังกล่าวสามารถอธิบายได้โดยพิจารณาจากการทำงานของแต่ละอัลกอริทึม (แสดงไว้ในบทที่ 2) การที่อัลกอริทึมในกลุ่ม Simple Hash Function สามารถทำงานได้อย่างรวดเร็วก็เนื่องมาจากความง่ายของอัลกอริทึม ไม่มีการคำนวณที่ซับซ้อน และเมื่อเปรียบเทียบเฉพาะในกลุ่ม Simple Hash Function ด้วยกันเองก็จะเห็นได้อย่างชัดเจนว่า อัลกอริทึม Digit Analysis ทำงานได้เร็วที่สุด เพราะการทำงานของอัลกอริทึมง่ายที่สุด ส่วนอัลกอริทึมอื่นๆ ในกลุ่มนี้จะใช้เวลาในการทำงานมากขึ้นเมื่ออัลกอริทึมมีการคำนวณที่ยุ่งยากมากขึ้นด้วย



รูปที่ 5.2 กราฟแสดงเวลาที่ใช้ในการเข้ารหัสยูอาร์แอล 1 ยูอาร์แอล

อัลกอริทึมในกลุ่ม CRC มีความเร็วเฉลี่ยในการทำงานที่ใกล้เคียงกัน เนื่องจากมีความแตกต่างเพียงตัวหารที่ใช้เท่านั้น และถึงแม้ว่าอัลกอริทึมจะมีความซับซ้อนมากกว่าอัลกอริทึมในกลุ่ม Simple Hash Function แต่ก็มีความเร็วที่ใกล้เคียงกัน เพราะอัลกอรทึม CRC ที่ใช้ในการทดลองนั้นเป็นอัลกอริทึมที่มีการปรับปรุงให้สามารถทำงานได้เร็วขึ้น ซึ่งมีการทำโอเปอเรชั่น XOR และ Shift bit

หลายครั้งทำให้เวลาที่ใช้มีค่าใกล้เคียงกับ Folding method ซึ่งอยู่ในกลุ่ม Simple Hash Function เนื่องจาก Folding method มีการคำนวณโอเปอเรชั่น XOR และ Shift bit เช่นเดียวกัน

อัลกอริทึม Huffman Coding และอัลกอริทึมในกลุ่ม MD ใช้เวลาในการเข้ารหัสที่ค่อนข้างมาก เนื่องมาจากการเข้ารหัสที่มีความซับซ้อน สำหรับอัลกอริทึม Huffman นั้น มีการทำงานแบ่งออกเป็น 2 ช่วง คือ ช่วงแรกจะเป็นการวิเคราะห์ตัวอักษรซึ่งมีการทำเพียงครั้งเดียวและใช้ผลการวิเคราะห์ในการเข้ารหัสในครั้งต่อๆไป เวลาที่แสดงเป็นผลการทดลองนั้นเป็นเวลาที่ใช้ในช่วงของการเข้ารหัสเท่านั้น ไม่นำเวลาในช่วงของการวิเคราะห์เข้ามาพิจารณา และนอกจากจะเป็นการเข้ารหัสแล้ว อัลกอริทึม Huffman Coding ยังมีลักษณะของการย่อข้อมูลอยู่ด้วย อัลกอริทึมจึงมีความซับซ้อนมากกว่าทุกกลุ่ม

ส่วนอัลกอริทึมในกลุ่ม MD นั้นใช้ในการเข้ารหัสเพื่อรักษาความปลอดภัยทำให้มีความซับซ้อนในการทำงานค่อนข้างมากและใช้เวลาในการเข้ารหัสมากขึ้นไปด้วย ถ้าหากเปรียบเทียบความเร็วในการทำงานเฉพาะในกลุ่มนี้ จะสามารถเรียงตามลำดับเวลาที่ใช้ในการเข้ารหัสจากมากไปหาน้อยได้ดังนี้ คือ SHA-1 MD5 และ MD4 จากผลการทดลองดังกล่าวสามารถอธิบายได้ดังนี้ คือ อัลกอริทึม MD2 ใช้เวลามากที่สุด (จึงได้ตัดออกไม่ได้นำมาพิจารณา) เนื่องจากเป็นอัลกอริทึมที่คิดค้นขึ้นมาเป็นอัลกอริทึมแรกในกลุ่ม และออกแบบให้ทำงานได้ดีกับเครื่องคอมพิวเตอร์ที่มีสถาปัตยกรรม 8 บิต ส่วน MD4 เป็นอัลกอริทึมที่เกิดขึ้นตามมาและมีการออกแบบให้ทำงานกับเครื่องคอมพิวเตอร์ที่มีสถาปัตยกรรม 32 บิต ทำให้มีความเร็วในการทำงานมากกว่า และ MD5 เป็นอัลกอริทึมที่เกิดขึ้นล่าสุด แต่มีความเร็วในการทำงานช้ากว่า MD4 อันเนื่องมาจากความซับซ้อนที่เพิ่มขึ้นในการปรับปรุงประสิทธิภาพด้านการรักษาความปลอดภัยทำให้ใช้เวลาในการทำงานมากขึ้นไปด้วย ส่วน SHA-1 เป็นอัลกอริทึมเดียวในกลุ่มที่แม้จะมีลักษณะการทำงานที่คล้ายกับ MD4 และ MD5 แต่ไม่ได้เป็นอัลกอริทึมที่พัฒนาต่อเนื่องกันทำให้มีความเร็วในการเข้ารหัสที่แตกต่างจากอัลกอริทึมอื่นๆในกลุ่ม แต่ก็ยังคงมีความใกล้เคียงกัน

นอกจากอัลกอริทึมที่แตกต่างกันใช้เวลาในการเข้ารหัสแตกต่างกันแล้ว ความยาวของรหัสที่กำหนดขึ้นล่วงหน้าก็มีผลต่อความเร็วด้วย ดังกรณีของ Digit Analysis และ Folding method

สำหรับ Digit Analysis method นั้น รหัสที่สั้นกว่าจะใช้เวลาน้อยกว่า เนื่องจากเป็นเพียงการเลือกช่วงของตัวอักษรมาใช้ ดังนั้นจึงเสียเวลาน้อยกว่าในการประมวลผลข้อมูลที่น้อยกว่า ส่วน Folding method นั้น รหัสที่สั้นกว่าไม่ได้มีความเร็วในการทำงานมากกว่า เช่น Folding 2 bytes ใช้เวลาในการเข้ารหัสมากกว่า Folding ที่ความยาวรหัสอื่นๆ ซึ่งสามารถอธิบายได้ดังนี้คือ ถึงแม้ว่ารหัสที่สั้นกว่ามีการประมวลผลกับข้อมูลน้อยกว่าก็ตาม แต่จากการทำงานของอัลกอริทึม ทำให้มีจำนวนรอบการประมวลผลมากขึ้นด้วย นอกจากนี้การประมวลผลข้อมูลครั้งละ 32 บิตจะทำงานได้เร็วกว่าการประมวลผลข้อมูลครั้งละ 16 บิต ทั้งนี้เป็นผลมาจากสถาปัตยกรรมของฮาร์ดแวร์และระบบปฏิบัติการ

5.3 ความยาวรหัสและประสิทธิภาพในการลดขนาดยูอาร์แอล

ความยาวรหัสที่ได้จากแต่ละอัลกอริทึมนั้น สามารถพิจารณาแยกออกได้เป็น 2 ประเภท คือ อัลกอริทึมที่ให้รหัสที่มีความยาวคงที่ และอัลกอริทึมที่ให้รหัสที่มีความยาวไม่คงที่ อัลกอริทึมที่ให้รหัสที่มีความยาวคงที่นั้น รหัสที่ได้จะมีความยาวเท่ากันหมดซึ่งความยาวรหัสอาจจะเกิดจากการกำหนดไว้ล่วงหน้า และเนื่องจากอัลกอริทึมเหล่านี้มีขนาดของผลลัพธ์ที่เท่ากัน ประสิทธิภาพในการลดขนาดของยูอาร์แอลจึงเท่ากันด้วย ดังนั้นเพื่อสะดวกในการพิจารณาจึงแบ่งอัลกอริทึมที่มีความยาวรหัสคงที่ ออกเป็นกลุ่มที่มีความยาวรหัสเท่ากันดังนี้

- *กลุ่มที่มีความยาวรหัส 2 ไบต์หรือ 16 บิต* ได้แก่ CRC-16, CRC-CCITT, 2 bytes Digit Analysis method และ 2 bytes Folding method
- *กลุ่มที่มีความยาวรหัส 4 ไบต์หรือ 32 บิต* ได้แก่ CRC-32, 4 bytes Digit Analysis และ 4 bytes Folding method
- *กลุ่มที่มีความยาวรหัส 8 ไบต์หรือ 64 บิต* ได้แก่ 8 bytes Digit Analysis และ 8 bytes Folding method
- *กลุ่มที่มีความยาวรหัส 16 ไบต์หรือ 128 บิต* ได้แก่ MD2, MD4, MD5, 16 bytes Digit Analysis และ 16 bytes Folding method
- *กลุ่มที่มีความยาวรหัส 160 บิต* ได้แก่ SHA-1 เพียงอัลกอริทึมเดียว

ส่วนอัลกอริทึมอีกประเภทหนึ่ง คือ อัลกอริทึมที่ให้ความยาวรหัสไม่คงที่ ได้แก่ Midsquare method, Division method และ Huffman Coding การพิจารณาความยาวรหัสที่ได้นั้นจะใช้ค่าเฉลี่ยความยาวรหัสจากค่าเฉลี่ยความยาวรหัสที่ได้จากยูอาร์แอลในแต่ละวันมาพิจารณา ส่วนการพิจารณาประสิทธิภาพในการลดขนาดยูอาร์แอลนั้นจะใช้ค่าอัตราส่วนในการลดขนาด (Size Reducing Ratio) และพิจารณาจะแยกแต่ละอัลกอริทึมออกจากกันเพื่อให้สามารถเปรียบเทียบความยาวกับอัลกอริทึมในกลุ่มอื่นๆ ที่กล่าวมาข้างต้นได้

อัตราส่วนในการลดขนาด (Size Reduction Ratio) ซึ่งใช้เปรียบเทียบประสิทธิภาพในการลดขนาดยูอาร์แอลจะคำนวณจากเปอร์เซ็นต์เฉลี่ยของขนาดที่สามารถลดได้ของแต่ละอัลกอริทึม ดังนี้

$$\textit{อัตราส่วนในการลดขนาด} = \frac{\textit{ความยาวยูอาร์แอลก่อนเข้ารหัส} - \textit{ความยาวรหัส}}{\textit{ความยาวยูอาร์แอลก่อนเข้ารหัส}} x100$$

โดยถ้าหากอัตราส่วนในการลดขนาดมีค่ามากแสดงว่ามีประสิทธิภาพในการลดขนาดมาก และในการหาประสิทธิภาพในการลดขนาดยูอาร์แอลนี้จะใช้ยูอาร์แอลซึ่งไม่ได้กำจัดยูอาร์แอลที่ซ้ำกันออกในการเข้ารหัสและคำนวณ

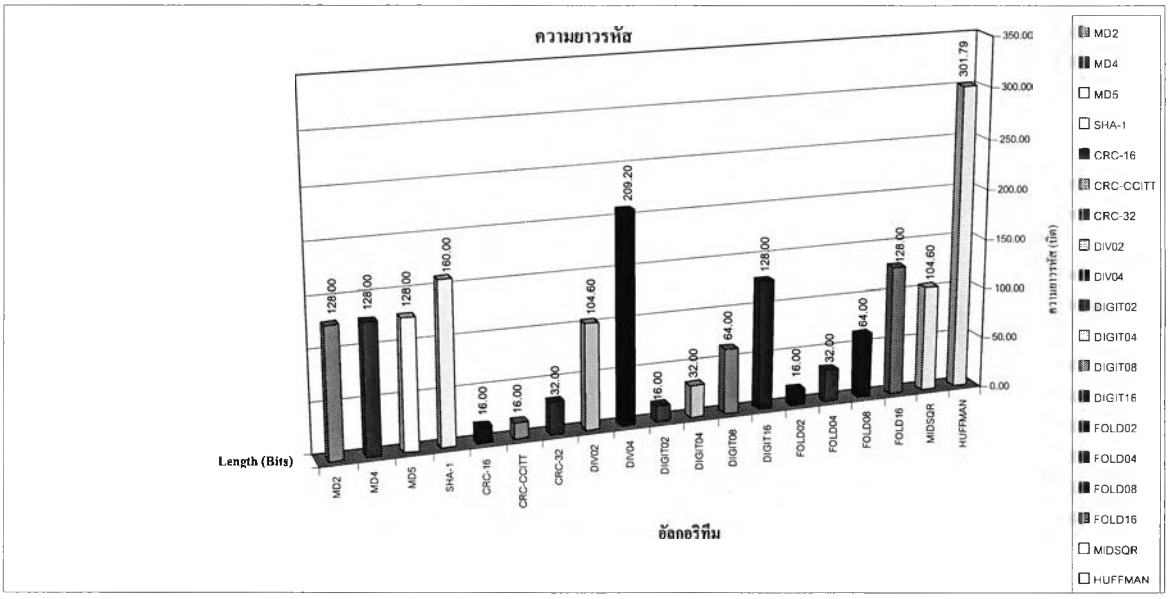
จากตารางแสดงความยาวรหัสในแต่ละอัลกอริทึมในบทที่ 4 สามารถสรุปมาเป็นตารางแสดงความยาวรหัสที่ได้จากแต่ละอัลกอริทึมในตารางที่ 5.2 และคำนวณอัตราส่วนในการลดขนาดยูอาร์แอลได้ดังแสดงในตารางที่ 5.3 และในการเปรียบเทียบความยาวรหัสและประสิทธิภาพในการลดขนาดยูอาร์แอลของแต่ละอัลกอริทึมแสดงเป็นกราฟ ดังในรูปที่ 5.3 และ 5.4 ซึ่งเป็นกราฟเปรียบเทียบความยาวของรหัสที่ได้จากแต่ละอัลกอริทึมและประสิทธิภาพในการลดขนาดยูอาร์แอลตามลำดับ

ตารางที่ 5.2 ตารางแสดงการเปรียบเทียบความยาวรหัสแต่ละอัลกอริทึม

กลุ่มอัลกอริทึม	Algorithm	Length (bits)	Size Reducing Ratio (%)
MD	MD2	128	64.35
MD	MD4	128	64.35
MD	MD5	128	64.35
MD	SHA	160	55.71
CRC	CRC16	16	94.60
CRC	CRC-CCITT	16	94.60
CRC	CRC32	32	90.28
Simple Hash	DIV2	104.60 (13.07 bytes)	73.64
Simple Hash	DIV4	209.20 (26.15 bytes)	47.36
Simple Hash	Digit 16	128 (16 bytes)	64.35
Simple Hash	Digit 08	64 (8 bytes)	81.64
Simple Hash	Digit 04	32 (4 bytes)	90.28
Simple Hash	Digit 02	16 (2 bytes)	94.60
Simple Hash	Fold16	128 (16 bytes)	64.35
Simple Hash	Fold08	64 (8 bytes)	81.64
Simple Hash	Fold04	32 (4 bytes)	90.28
Simple Hash	Fold02	16 (2 bytes)	94.60
Simple Hash	MSQ	104.60 (13.07 bytes)	73.68
Huffman	HUFF	256 (32.00 bytes)	29.69

ตารางที่ 5.3 ตารางแสดงการเปรียบเทียบประสิทธิภาพในการลดขนาดยูอาร์แอล

Key Length (Algorithms.)	Size Reducing Ratio (%)	100% - Size Reducing Ratio (%)
128 bits key length (MD2, MD4, MD5)	64.35	35.65
SHA-1 (160 bits) (SHA-1)	55.71	44.29
16 bits key length (CRC-16, CRC-CCITT,DIGIT02, FOLD02)	94.60	5.4
32 bits key length (CRC-32, DIGIT04, FOLD04)	90.28	9.72
64 bits key length (DIGIT08, FOLD08)	81.64	18.36
DIV02	73.64	26.36
DIV04	47.36	52.64
MIDSQR	73.68	26.32
HUFFMAN	29.69	70.31

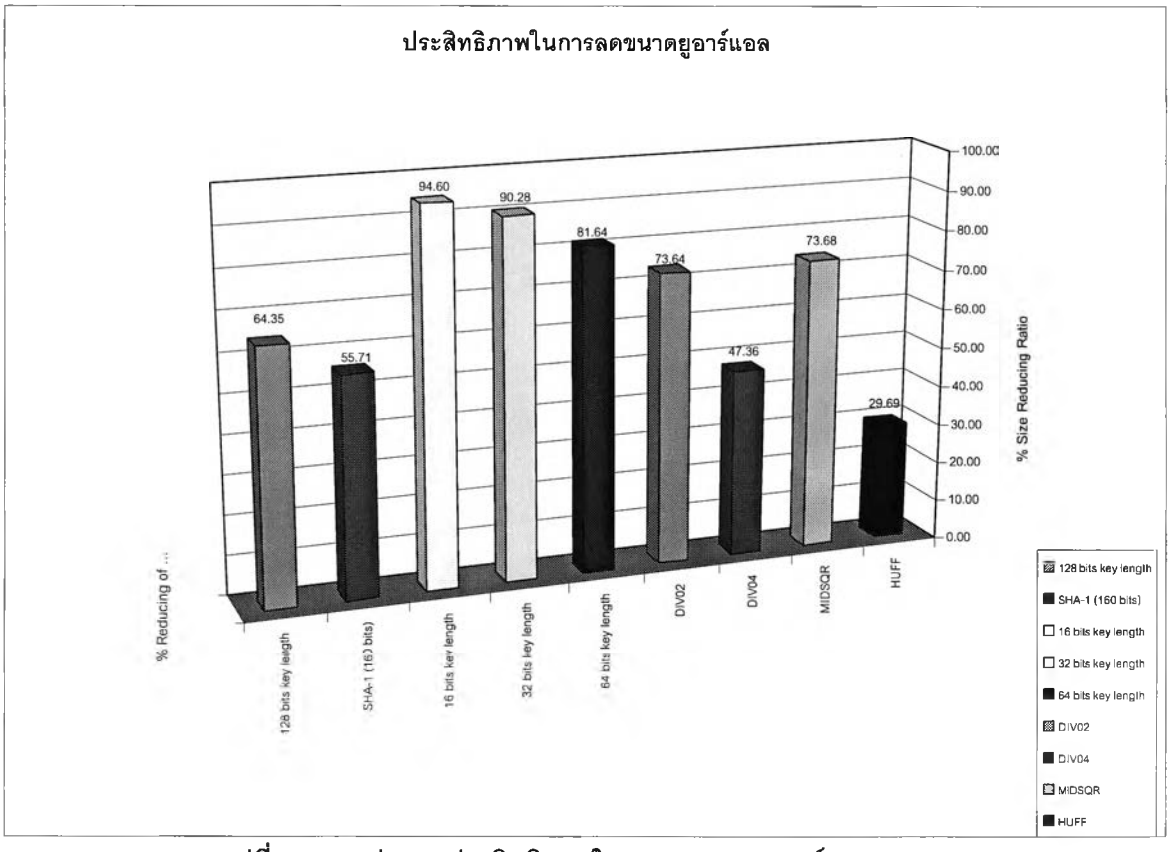


รูปที่ 5.3 กราฟแสดงความยาวรหัส

จากตารางที่ 5.2 แสดงความยาวรหัสที่ได้จากแต่ละอัลกอริทึม และกราฟในรูปที่ 5.3 ซึ่งแสดงการเปรียบเทียบความยาวรหัส พบว่าอัลกอริทึมที่ให้รหัสที่มีความยาวมากที่สุดคือ Huffman Coding ซึ่งมีความยาวรหัสเฉลี่ย 32 ไบต์ (หรือ 256 บิต) ส่วนขนาดรหัสที่สั้นที่สุดคือ 16 บิต ซึ่งได้แก่ CRC-16, CRC-CCITT, 2 bytes Digit Analysis และ 2 bytes Folding method

ข้อสังเกตประการหนึ่งก็คือ Division method ที่ใช้ตัวหารขนาด 2 ไบต์ และ Midsquare method จะมีความยาวรหัสที่เท่ากัน เนื่องจากการจำกัดค่าผลลัพธ์ที่ได้มีขนาดเท่ากัน จึงทำให้ได้ผลลัพธ์ที่มีขนาดเท่ากัน แต่อย่างไรก็ตาม ค่าของผลลัพธ์นั้นไม่มีส่วนสัมพันธ์กันเลย เพราะอัลกอริทึมในการคำนวณผลแตกต่างกัน

ในด้านประสิทธิภาพของการลดขนาดยูอาร์แอล จะพบว่ารหัสที่มีขนาดสั้นกว่าจะมีประสิทธิภาพในการลดขนาดยูอาร์แอลมากกว่า ดังจะเห็นได้จากกราฟ อัลกอริทึมที่มีความยาวรหัส 2 ไบต์หรือ 16 บิต จะมีประสิทธิภาพในการลดขนาดยูอาร์แอลมากที่สุด อัลกอริทึมที่มีความยาวรหัส 4 ไบต์ หรือ 32 บิต มีประสิทธิภาพในการลดขนาดยูอาร์แอลรองลงมา และอัลกอริทึม Huffman Coding ซึ่งมีความยาวรหัสมากที่สุดมีประสิทธิภาพในการลดขนาดยูอาร์แอลต่ำที่สุดด้วย แสดงให้เห็นว่าความยาวรหัสแปรผกผันกับประสิทธิภาพในการลดขนาดยูอาร์แอล กล่าวคือ อัลกอริทึมที่มีรหัสยาวจะมีประสิทธิภาพในการลดขนาดยูอาร์แอลต่ำ อัลกอริทึมที่มีรหัสสั้นจะมีประสิทธิภาพในการลดขนาดยูอาร์แอลสูง



รูปที่ 5.4 กราฟแสดงประสิทธิภาพในการลดขนาดยูอาร์แอล

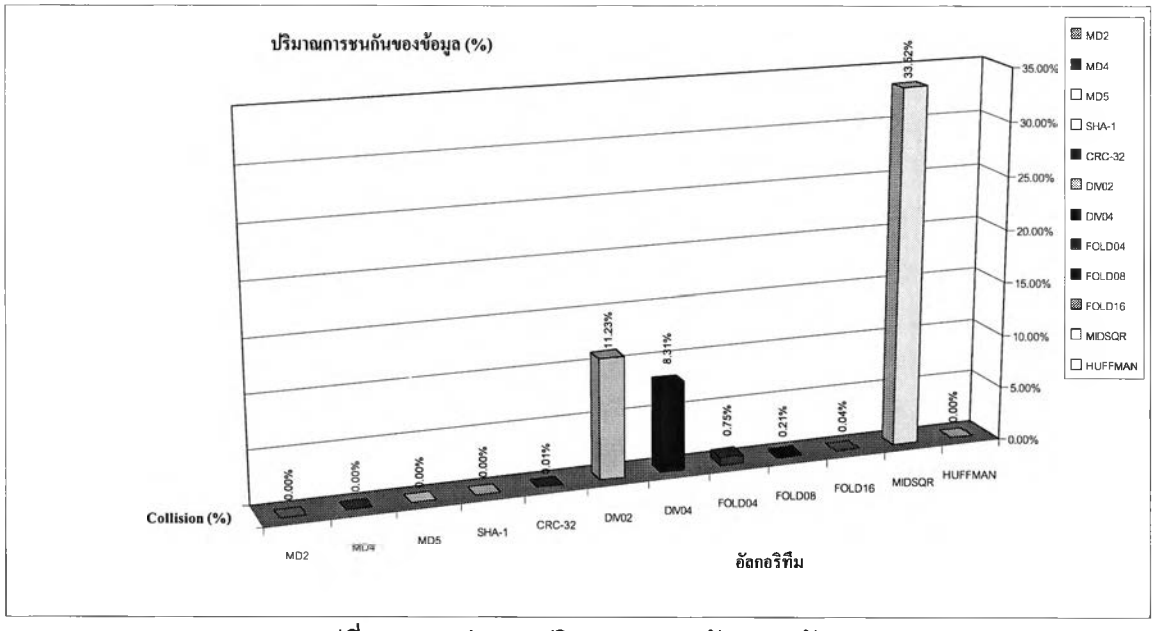
5.4 การชนกันของผลลัพธ์

ในการเปรียบเทียบปริมาณการชนกันของรหัสที่ได้จากแต่ละอัลอริทึม นำปริมาณการชนกันที่เกิดขึ้นของแต่ละอัลกอริทึมมาคำนวณเปอร์เซ็นต์ของจำนวนครั้งข้อมูลที่ชนต่อจำนวนยูอาร์แอลทั้งหมดของข้อมูลยูอาร์แอลในแต่ละวัน และนำค่าเฉลี่ยของเปอร์เซ็นต์มาใช้ในการเปรียบเทียบค่าเฉลี่ยดังกล่าวแสดงในตารางที่ 5.4

จากตารางเปรียบเทียบปริมาณการชนกันของรหัส พบว่าปริมาณการชนกันของรหัสที่ได้จากแต่ละอัลกอริทึมมีความแตกต่างกันมาก อัลกอริทึมเดียวกันและมีการกำหนดขนาดรหัสที่แตกต่างกันปริมาณการชนกันของรหัสที่ได้ไม่แตกต่างกันมากนัก อัลกอริทึมที่มีเปอร์เซ็นต์การชนกันของรหัสในปริมาณมากและค่อนข้างเด่นชัด คือ อัลกอริทึม Digit Analysis method ทุกขนาดความยาวรหัส อัลกอริทึมที่มีเปอร์เซ็นต์ปริมาณการชนกันของรหัสน้อยกว่า Digit Analysis method เล็กน้อย คือ อัลกอริทึม CRC-16, CRC-CCITT ซึ่งมีความยาวรหัส 16 บิต รวมทั้งอัลกอริทึม Folding method ที่ความยาวรหัส 16 บิตด้วย อัลกอริทึมเหล่านี้ถือว่าเป็นกลุ่มอัลกอริทึมที่ถือว่ามีปริมาณการชนกันของข้อมูลสูงมาก ดังนั้นเพื่อความชัดเจนในการเปรียบเทียบปริมาณการชนกัน จะตัดอัลกอริทึมเหล่านี้ซึ่งมีเปอร์เซ็นการชนกันของผลลัพธ์มากกว่า 50% ออกไม่นำมาพิจารณาในกราฟเปรียบเทียบปริมาณการชนกันของรหัส

ตารางที่ 5.4 ตารางแสดงการเปรียบเทียบปริมาณการชนกันของรหัสแต่ละอัลกอริทึม

กลุ่มอัลกอริทึม	Algorithm	% collision	กลุ่มอัลกอริทึม	Algorithm	% collision
MD	MD2	0.00%	Simple Hash	Digit 16	80.06%
MD	MD4	0.00%	Simple Hash	Digit 08	95.46%
MD	MD5	0.00%	Simple Hash	Digit 04	98.16%
MD	SHA	0.00%	Simple Hash	Digit 02	99.83%
CRC	CRC16	88.96%	Simple Hash	Fold16	0.04%
CRC	CRC32	0.01%	Simple Hash	Fold08	0.21%
CRC	CRC-CCITT	88.96%	Simple Hash	Fold04	0.75%
Simple Hash	DIV2	11.23%	Simple Hash	Fold02	97.22%
Simple Hash	DIV4	8.31%	Simple Hash	MSQ	33.52%
			Huffman	HUFF	0.00%



รูปที่ 5.5 กราฟแสดงปริมาณการชนกันของรหัส

อัลกอริทึมที่มีเปอร์เซ็นต์ปริมาณการชนกันของรหัสปานกลาง คือ อัลกอริทึม Division method และ Midsquare method ส่วนอัลกอริทึมกลุ่มที่มีเปอร์เซ็นปริมาณการชนกันของข้อมูลน้อยหรือไม่มีเลย คือ อัลกอริทึมในกลุ่ม MD, SHA-1, CRC-32, Folding method ที่มีความยาวรหัส 4, 8 และ 16 ไบต์ และ Huffman Coding อย่างไรก็ตามมีเพียง Huffman Coding เท่านั้นที่ปลอดการชนของรหัสอย่างแท้จริง เนื่องจากรหัสที่ได้ได้จากตัวอักษรแต่ละตัวที่กำหนดรหัสไม่ซ้ำกัน ส่วนอัลกอริทึมในกลุ่ม MD มีโอกาสเกิดการชนของรหัสน้อยมากๆเท่านั้น มิได้ปลอดการชนกันอย่างแท้จริง กราฟแสดงการเปรียบเทียบปริมาณการชนของรหัสแสดงในรูปที่ 5.5

5.5 สรุปผลการทดลอง

จากการวิเคราะห์ผลการทดลอง พบว่า ความเร็วในการเข้ารหัสมีส่วนสัมพันธ์กับความซับซ้อนของอัลกอริทึม โดยจากการพิจารณากราฟในรูปที่ 5.1 ซึ่งเปรียบเทียบความเร็วในการเข้ารหัส แสดงให้เห็นว่าอัลกอริทึมในกลุ่ม MD ซึ่งมีอัลกอริทึมที่ซับซ้อนใช้เวลามากกว่าอัลกอริทึมในกลุ่มอื่นๆ อัลกอริทึม Digit Analysis method ซึ่งมีความเร็วในการเข้ารหัสมากที่สุดนั้น มีการทำงานเพียงเลือกตัวอักษรช่วงหนึ่งมาใช้เป็นรหัสเท่านั้น อย่างไรก็ตาม จากการพิจารณาปริมาณการชนกันของรหัสจะพบว่า อัลกอริทึมที่ไม่ซับซ้อนมีแนวโน้มที่จะพบการชนของรหัสสูง เช่น กรณีของอัลกอริทึม Digit Analysis method พบการชนกันของรหัสที่ได้ในปริมาณสูงมาก

ในส่วนของความยาวของรหัสที่ได้จากแต่ละอัลกอริทึม พบว่า ความยาวของรหัสที่สั้นมีแนวโน้มที่จะมีปริมาณการชนกันของข้อมูลสูง ซึ่งจะเห็นได้จากอัลกอริทึมที่มีการกำหนดความยาวรหัสให้มีขนาด 2 ไบต์ (16 บิต) ไม่ว่าจะเป็น Digit Analysis method หรือ Folding method มีปริมาณการชนกันของรหัสสูงมาก และรวมทั้งอัลกอริทึม CRC-16 และ CRC-CCITT ซึ่งมีความยาวรหัส 16 บิต หรือ

2 ไบต์ ก็พบปริมาณการชนกันของรหัสสูงเช่นเดียวกัน นอกจากนั้น จากการพิจารณาประสิทธิภาพในการลดขนาดยูอาร์แอล พบว่า อลักอริทึมที่มีขนาดรหัสสั้นกว่ามีประสิทธิภาพในการลดขนาดยูอาร์แอลสูงกว่า และอัลกอริทึมที่มีความยาวรหัสยาวกว่ามีประสิทธิภาพในการลดขนาดยูอาร์แอลที่ต่ำกว่า

ในการพิจารณาเลือกใช้อัลกอริทึมให้ตรงกับความต้องการนั้น จะต้องพิจารณาจากความต้องการของแอปพลิเคชั่นเป็นหลัก โดยแบ่งความต้องการของแอปพลิเคชั่นออกเป็น 3 แบบ คือ

- แอปพลิเคชั่นที่ต้องการความเร็วในการเข้ารหัสและรหัสที่สั้นหรือประสิทธิภาพในการลดขนาดยูอาร์แอลเป็นหลัก เช่น การตัดสินใจในการเลือกเว็บพร็อกซี่เพื่อขอข้อมูล (ในการทำ Load Balancing) เป็นต้น แอปพลิเคชั่นลักษณะนี้จะต้องการให้เกิดการชนกันของข้อมูลที่เป็นปริมาณเท่าๆ กัน
- แอปพลิเคชั่นที่ต้องการความเร็วในการเข้ารหัสและปริมาณการชนกันของรหัสน้อยหรือไม่มีเลย เป็นหลัก เช่น การวิเคราะห์ล็อกไฟล์ การตรวจสอบข้อมูลในแคช เป็นต้น
- แอปพลิเคชั่นที่ต้องการรหัสที่สั้นหรือประสิทธิภาพในการลดขนาดยูอาร์แอลและปริมาณการชนกันของรหัสน้อยหรือไม่มีเลยเป็นหลัก เช่น การสอบถามข้อมูลระหว่างเว็บพร็อกซี่หรือใช้ในโปรโตคอลเพื่อสอบถามข้อมูลระหว่างแคช เป็นต้น

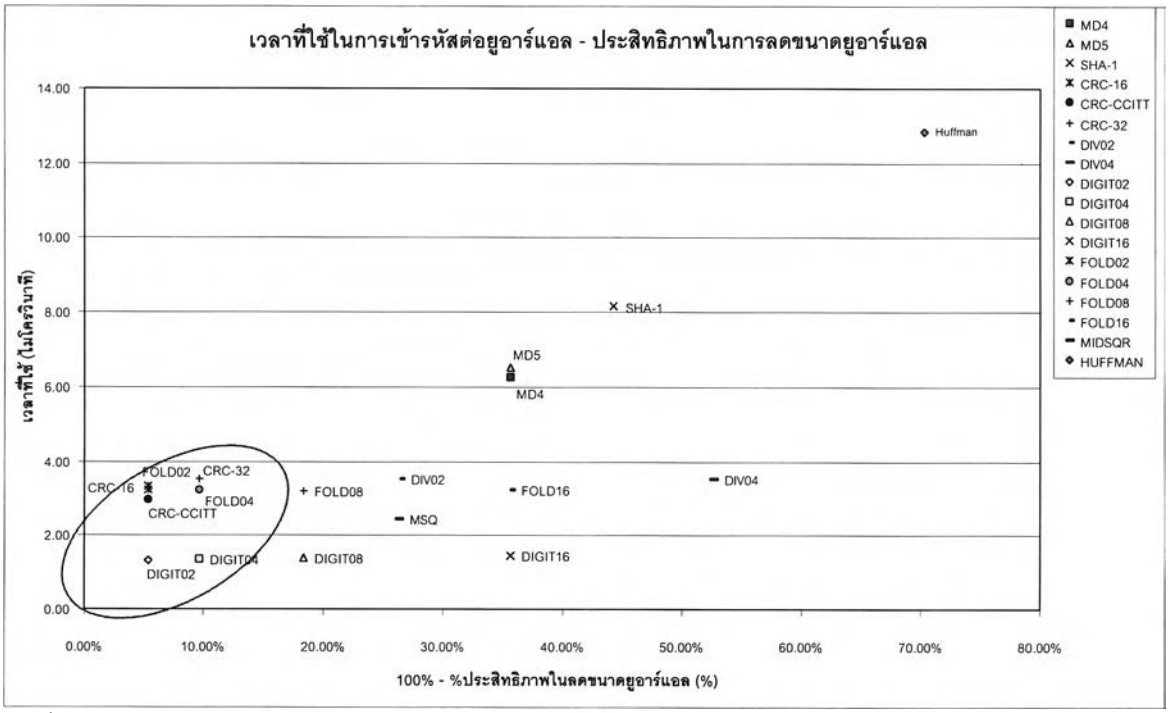
เพื่อให้สามารถพิจารณาได้ง่ายขึ้นจะนำข้อมูลในตารางที่ 5.1 แสดงความเร็วในการเข้ารหัสและตารางที่ 5.3 แสดงประสิทธิภาพในการลดขนาดยูอาร์แอลมาลงจุดในแผนภูมิแบบกระจายดังรูปที่ 5.6 เพื่อใช้ในการพิจารณาเลือกอัลกอริทึมตามความต้องการของแอปพลิเคชั่นที่ต้องการความเร็วในการเข้ารหัสและรหัสที่สั้นหรือประสิทธิภาพในการลดขนาดยูอาร์แอลเป็นหลัก นำข้อมูลในตารางที่ 5.1 และ 5.4 แสดงปริมาณการชนกันของรหัสมาลงจุดในแผนภูมิแบบกระจายดังรูปที่ 5.7 เพื่อใช้ในการพิจารณาเลือกอัลกอริทึมตามความต้องการของแอปพลิเคชั่นที่ต้องการความเร็วในการเข้ารหัสและปริมาณการชนกันน้อยหรือไม่มีเลยเป็นหลัก นำข้อมูลในตารางที่ 5.3 และ 5.4 มาลงจุดในแผนภูมิแบบกระจายดังรูปที่ 5.9 เพื่อใช้ในการพิจารณาเลือกอัลกอริทึมตามความต้องการของแอปพลิเคชั่นที่ต้องการรหัสที่สั้นหรือประสิทธิภาพในการลดขนาดยูอาร์แอลและปริมาณการชนกันน้อยหรือไม่มีเลยเป็นหลัก และเพื่อความสะดวกในการพิจารณาค่าประสิทธิภาพในแผนภูมิจะใช้ค่า 100% - ค่าประสิทธิภาพในการลดขนาดยูอาร์แอล ในการแสดงตำแหน่งในแผนภูมิแทนการใช้ค่าประสิทธิภาพลงในแผนภูมิโดยตรง และมีการตัดบางอัลกอริทึมออก ดังนี้

- ตัดอัลกอริทึม MD2 ออก ไม่นำมาพิจารณา เนื่องจาก MD2 ทำงานได้ช้ามากเมื่อเทียบกับอัลกอริทึมอื่นๆ เมื่อนำมาพล็อตกราฟแสดงความสัมพันธ์จะทำให้ไม่สามารถเห็นรายละเอียด และขาดความชัดเจนในการพิจารณาอัลกอริทึมอื่นๆ
- ตัดอัลกอริทึมที่มีปริมาณการชนกันของผลลัพธ์เกิน 50% ออก ในการพิจารณาเลือกอัลกอริทึมสำหรับแอปพลิเคชันที่เกี่ยวข้องกับปริมาณการชนกันของรหัสที่ได้ เนื่องจาก

ปริมาณการชนที่มากกว่า 50 เปอร์เซ็นต์นั้นมากเกินไปสำหรับแอปพลิเคชันที่ไม่ต้องการให้เกิดการชนกันของรหัส และทำให้พิจารณาอัลกอริทึมอื่นๆที่มีปริมาณการชนกันต่างกันมากด้วยความลำบาก และไม่ชัดเจน

จากแผนภูมิทั้งสาม อัลกอริทึมที่มีจุดอยู่ใกล้จุดกำเนิดมากที่สุดจะเป็นอัลกอริทึมที่สนองความต้องการแอปพลิเคชั่นที่มีแต่ละแบบได้ดีที่สุด อัลกอริทึมที่มีประสิทธิภาพในเกณฑ์พิจารณาเลือกจะอยู่ในวงกลม

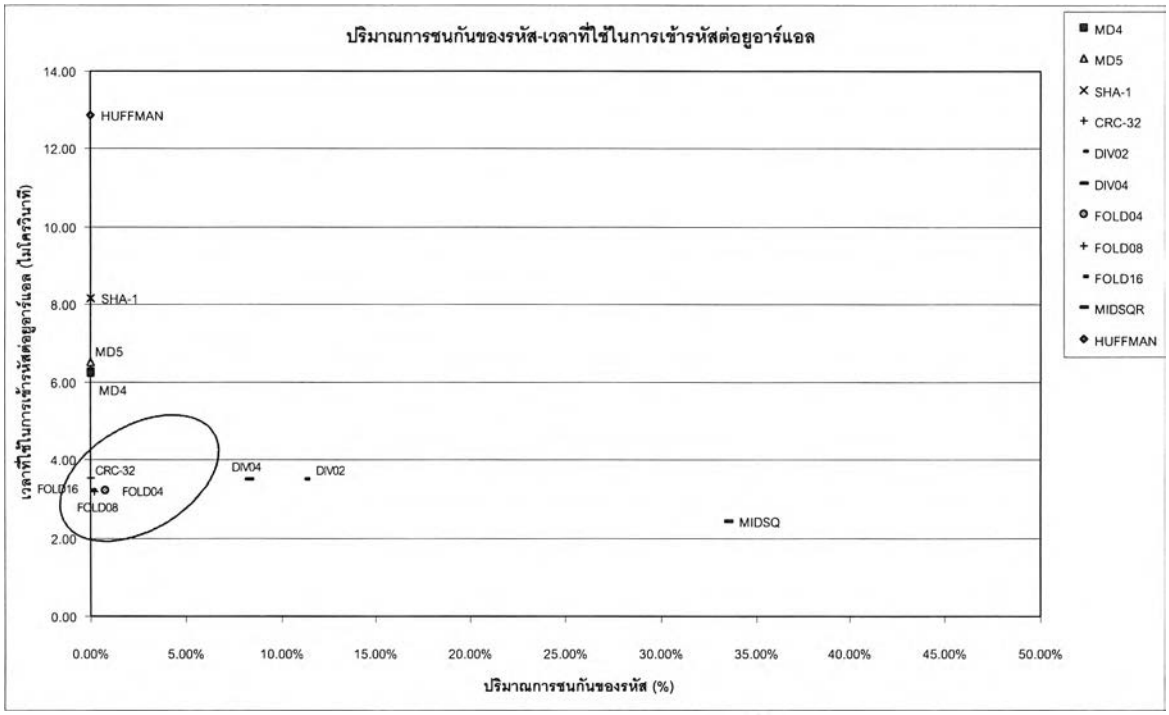
แผนภูมิในรูป 5.6 แสดงเวลาที่ใช้เข้ารหัสและประสิทธิภาพในการลดขนาดยูอาร์แอลของแต่ละอัลกอริทึมซึ่งแสดงให้เห็นว่าอัลกอริทึมที่เหมาะสมและใช้งานได้ดีกว่าในแง่เวลาที่ใช้ และประสิทธิภาพในการลดความยาวรหัสคือ Digit Analysis 2 (DIGIT02) และ 4 ไบต์ (DIGIT04), Folding 2 (FOLD02) และ 4 ไบต์ (FOLD04), CRC-16, CRC-CCITT และ CRC-32 และจากแผนภูมิอัลกอริทึมที่มีประสิทธิภาพในด้านความเร็วในการเข้ารหัสและประสิทธิภาพในการลดขนาดยูอาร์แอลดีที่สุด คือ Digit Analysis ขนาด 2 ไบต์ เนื่องจากมีตำแหน่งใกล้จุดกำเนิดของแผนภูมิมากที่สุด



รูปที่ 5.6 แผนภูมิแสดงเวลาที่ใช้ในการเข้ารหัส 1 ยูอาร์แอลและประสิทธิภาพในการลดขนาดยูอาร์แอล

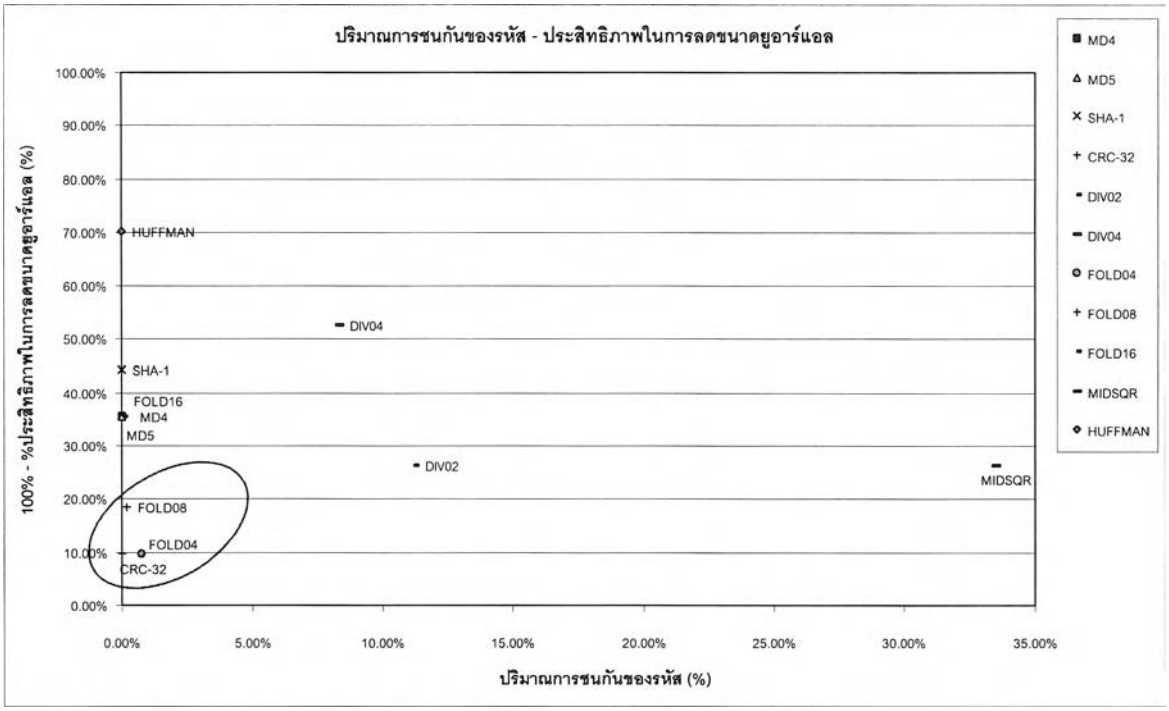
แผนภูมิในรูป 5.7 แสดงปริมาณการชนกันของรหัสและเวลาที่ใช้ในการเข้ารหัส 1 ยูอาร์แอลของแต่ละอัลกอริทึม โดยตัดอัลกอริทึมที่มีปริมาณการชนกันมากกว่า 50% ออกไม่นำมาพิจารณาด้วยเหตุผลดังที่ได้กล่าวไปแล้ว อัลกอริทึมที่เหมาะสมและใช้งานได้ดีกว่าในแง่ปริมาณการชนกันของรหัสและเวลาที่ใช้ในการเข้ารหัส คือ Folding method ขนาด 4 (FOLD04), 8 (FOLD08) และ 16 (FOLD16) ไบต์, และ CRC-32 อย่างไรก็ตาม ถึงแม้ว่าอัลกอริทึมที่กล่าวมานั้นจะมีปริมาณการชนกัน

ของรหัสน้อย แต่ก็ยังพบว่ามีการชนกัน ในแอปพลิเคชันบางประเภทไม่อนุญาตให้มีการชนกันของรหัสเกิดขึ้น ดังนั้นอัลกอริทึมดังที่กล่าวมาจึงไม่สามารถใช้ได้ กรณีเช่นนี้ อัลกอริทึมที่จะพิจารณาจะต้องเป็นอัลกอริทึมที่ไม่มีการชนกันของรหัสเกิดขึ้นเลย นั่นคือ อัลกอริทึมในกลุ่ม MD ได้แก่ MD2, MD4, MD5 และ SHA-1 และอัลกอริทึม Huffman Coding แต่เมื่อพิจารณาในเรื่องความเร็วในการเข้ารหัสร่วมด้วยแล้ว อัลกอริทึมที่มีความเหมาะสม คือ MD4 และ MD5 และเมื่อพิจารณาจากแผนภูมิแล้ว อัลกอริทึมที่มีประสิทธิภาพในแง่ของปริมาณการชนกันของรหัสและความเร็วในการเข้ารหัส คือ Folding method ขนาด 8 ไบต์ เนื่องจากมีตำแหน่งใกล้จุดกำเนิดของแผนภูมิมากที่สุด



รูปที่ 5.7 แผนภูมิแสดงปริมาณการชนกันของรหัสและเวลาที่ใช้ในการเข้ารหัส 1 ยูอาร์แอล

แผนภูมิในรูป 5.8 แสดงปริมาณการชนกันของรหัสและประสิทธิภาพในการลดขนาดยูอาร์แอลของแต่ละอัลกอริทึม จากแผนภูมิแสดงให้เห็นว่า อัลกอริทึมที่เหมาะสมและใช้งานได้ดีกว่าในแง่ปริมาณการชนกันของรหัสและประสิทธิภาพในการลดขนาดยูอาร์แอล คือ Folding method ขนาด 4 (FOLD04)ไบต์, 8 ไบต์ (FOLD08) และ CRC-32 และจากแผนภูมิจะพบว่าอัลกอริทึมที่มีประสิทธิภาพในแง่ของปริมาณการชนกันของรหัสและประสิทธิภาพในการลดขนาดยูอาร์แอลดีที่สุด คือ CRC-32 เนื่องจากมีตำแหน่งที่ใกล้จุดกำเนิดของแผนภูมิมากที่สุด



รูปที่ 5.8 แผนภูมิแสดงปริมาณการชนกันของรหัสและประสิทธิภาพในการลดขนาดยูอาร์แอล

เมื่อพิจารณาอัลกอริทึมแต่ละกลุ่ม สามารถสรุปลักษณะของอัลกอริทึมในกลุ่มต่างๆที่นำมาใช้ในการวิจัยได้ดังนี้

ในกลุ่ม MD อัลกอริทึมในกลุ่มนี้ ได้แก่ MD2, MD4, MD5 และ SHA-1 อัลกอริทึมในกลุ่มนี้มีข้อดี คือ ตามทฤษฎีแล้วจะไม่เกิดการชนกันของรหัสเลย แต่ในความเป็นจริง อาจพิจารณาได้ว่ามีโอกาสเกิดการชนกันของรหัสน้อยมาก ข้อเสียของอัลกอริทึมในกลุ่มนี้ก็คือ ใช้เวลาในการเข้ารหัสค่อนข้างมาก และได้รหัสที่ค่อนข้างยาวเมื่อเปรียบเทียบกับอัลกอริทึมอื่นๆ อย่างไรก็ตามประสิทธิภาพในการลดขนาดยูอาร์แอลอยู่ในระดับปานกลาง การที่อัลกอริทึมในกลุ่มนี้เกิดการชนกันของรหัสน้อยมากก็เนื่องมาจากอัลกอริทึมเหล่านี้ถูกสร้างขึ้นมาเพื่อใช้ในการรักษาความปลอดภัย และด้วยเหตุผลเดียวกันนี้จึงทำให้อัลกอริทึมมีความซับซ้อน ทำให้ใช้เวลาในการเข้ารหัสมากกว่าเมื่อเปรียบเทียบกับอัลกอริทึมในกลุ่มอื่นๆ อัลกอริทึมในกลุ่มนี้จึงเหมาะกับแอปพลิเคชันเว็บที่ไม่ต้องการความเร็วในการเข้ารหัสมากนัก และไม่ต้องการให้เกิดการชนกันของรหัสเลย

อัลกอริทึมในกลุ่ม CRC อัลกอริทึมในกลุ่มนี้ ได้แก่ CRC-16, CRC-CCITT และ CRC-32 อัลกอริทึมในกลุ่มนี้มีข้อดีคือ ใช้เวลาเข้ารหัสน้อย ขนาดรหัสที่ได้สั้นทำให้มีประสิทธิภาพในการลดขนาดยูอาร์แอลสูง แต่มีข้อเสียคือ มีปริมาณการชนกันของข้อมูลสูง ยกเว้น CRC-32 ซึ่งมีปริมาณการชนกันน้อยมาก อัลกอริทึมในกลุ่มนี้เหมาะสมกับแอปพลิเคชันเว็บที่อนุญาตให้มีการชนกันของรหัสได้หรือต้องการให้เกิดการชนกันของรหัสขึ้น เช่น การทำ load balancing เป็นต้น ยกเว้น CRC-32 ซึ่งมีคุณสมบัติที่ดีมาก และจะกล่าวโดยละเอียดอีกครั้งหนึ่ง

อัลกอริทึมในกลุ่ม Simple Hash Function ซึ่งได้แก่ Digit Analysis Method, Folding Method, Division Method และ Midsquare Method อัลกอริทึมในกลุ่มนี้มีข้อดีคือ ใช้เวลาในในการเข้ารหัสน้อยมาก และมีขนาดรหัสที่สั้น มีประสิทธิภาพในการลดขนาดยูอาร์แอลสูง แต่มีข้อเสียคือ มีปริมาณการชนกันของรหัสสูง ยกเว้นเพียงอัลกอริทึม Folding method ที่ความยาวรหัสมากกว่า 2 ไบต์ซึ่งมีปริมาณการชนที่ต่ำ โดยเฉพาะอย่างยิ่ง Folding method ที่ความยาวรหัส 8 และ 16 ไบต์ มีปริมาณการชนกันของรหัสต่ำมาก อัลกอริทึมในกลุ่มนี้จึงเหมาะสำหรับแอปพลิเคชั่นเว็บที่ต้องการให้เกิดการชนกันของรหัส เช่น การตัดสินใจเลือกเว็บพร็อกซี่ที่ให้บริการ หรือการทำ load balancing

อัลกอริทึมสุดท้ายก็คือ Huffman Coding ซึ่งพิจารณาจากทั้งทางด้านความเร็วในการเข้ารหัส ประสิทธิภาพในการลดขนาดยูอาร์แอล นั้นจะพบว่าเมื่อเปรียบเทียบกับอัลกอริทึมอื่นๆแล้ว Huffman Coding ไม่มีคุณสมบัติใดที่โดดเด่น หรือได้เปรียบกว่าอัลกอริทึมอื่นแต่อย่างใด ซึ่งถ้าหากจะพิจารณาเพื่อใช้ในการเข้ารหัสยูอาร์แอลแล้ว ถือว่าไม่มีความเหมาะสม

จากการวิเคราะห์ผลการทดลองที่กล่าวมาทั้งหมด จะพบว่าในการเลือกอัลกอริทึมในการเข้ารหัสที่มีความเหมาะสมสำหรับแอปพลิเคชั่นแต่ละชนิดนั้นจะต้องคำนึงถึงความต้องการของแอปพลิเคชั่นนั้นๆ ว่ามีความต้องการอย่างไร และไม่มีอัลกอริทึมใดที่เหมาะสมสำหรับทุกๆ แอปพลิเคชั่น อย่างไรก็ตามสำหรับในงานวิจัยนี้อัลกอริทึมที่มีประสิทธิภาพโดยรวมดีที่สุด คือ CRC-32 ทั้งนี้ เนื่องจาก CRC-32 เป็นอัลกอริทึมที่รู้จักกันดีในการตรวจสอบหาความผิดพลาดในการสื่อสารข้อมูล ใช้เวลาในการเข้ารหัสน้อย รหัสที่ได้สั้น (32 บิตหรือ 4 ไบต์) ทำให้มีประสิทธิภาพสูงในการลดขนาดยูอาร์แอล อีกทั้งปริมาณการชนกันของรหัสต่ำมาก เพียง 0.01% เท่านั้น ดังนั้นเมื่อพิจารณาประสิทธิภาพดังกล่าวเทียบกับอัลกอริทึมอื่นๆ แล้วจึงสรุปได้ว่า เป็นอัลกอริทึมที่มีประสิทธิภาพโดยรวมสูงที่สุดในจำนวนอัลกอริทึมทั้งหมดที่ใช้ในงานวิจัยนี้