XÂY DỰNG CƠ SỞ DỮ LIỆU CÁC ĐA THỨC BẤT KHẢ QUY PHỤC VỤ CÁC ỨNG DỤNG THỰC TIỄN

***Trần Đức Dũng*[1]**
***Nguyễn Thị Ái Anh*[1]**
***Đặng Thanh Dũng*[2]**

TÓM TẮT

Đa thức bất khả quy có nhiều ứng dụng thực tiễn, đặc biệt là trong các lĩnh vực lý thuyết mã (coding theory), mã hóa mật mã (cryptography) và an toàn thông tin (information security). Bài báo này trình bày kết quả nghiên cứu và cài đặt thử nghiệm của quá trình xây dựng cơ sở dữ liệu các đa thức bất khả quy phục vụ các ứng dụng thực tiễn.

Từ khóa: *Đa thức bất khả quy,trường hữu hạn, lý thuyết mã, mã hóa mật mã, an toàn thông tin*

1. Giới thiệu

Đa thức bất khả quy có nhiều tính chất tương tự như số nguyên tố của vành số nguyên và có thể xem như là một sự mở rộng và thay thế tự nhiên của số nguyên tố trong nhiều ứng dụng thực tiễn, đặc biệt là trong lĩnh vực mã hóa mật mã và chữ ký số [1], [2], [3], [4]. Tuy nhiên, quá trình phát sinh đa thức bất khả quy tiêu tốn lượng thời gian đáng kể, đặc biệt là khi cần phát sinh đa thức bất khả quy bậc cao hay phát sinh với số lượng lớn.

Câu hỏi đặt ra là tại sao chúng ta không xây dựng sẵn một kho cơ sở dữ liệu các đa thức bất khả quy để khi cần có thể nhận được kết quả một cách nhanh chóng và tiện lợi? Chúng ta cũng có thể dễ dàng nhận được các đa thức bất khả quy thỏa mãn một số tiêu chí nào đó theo yêu cầu của ứng dụng. Quá trình xây dựng cơ sở dữ liệu các đa thức bất khả quy bao gồm quá trình phát sinh đa thức và chọn lựa cấu trúc lưu trữ thích hợp thuận tiện cho việc tìm kiếm. Một số công trình đưa ra một danh sách hạn chế các đa thức bất khả quy hoặc liệt kê một số lượng lớn các đa thức bất khả quy dạng đặc biệt [4], [5] chứng tỏ rằng quá trình xây dựng cơ sở dữ liệu các đa thức bất khả quy là thực sự cần thiết. Dựa vào các công trình nghiên cứu có liên quan chúng tôi đưa ra ba giải pháp phát sinh đa thức bất khả quy:

i.) Giải pháp thứ nhất là phát sinh ra đa thức bất kỳ bậc n rồi dựa vào giải thuật kiểm tra tính bất khả quy của đa thức (giải thuật IPT trong [6]) và thực hiện như sau: chọn ngẫu nhiên một đa thức rồi kiểm tra xem đa thức đó có phải là đa thức bất khả quy hay không và lặp lại quá trình này cho đến khi chọn được đa thức bất khả quy.

ii.) Giải pháp thứ hai dựa vào công thức tổng hợp: từ các đa thức bất khả quy đã biết chúng ta có thể tổng hợp nên các đa thức bất khả quy mới. Giải pháp này có thể tổng hợp ra các đa thức bất khả quy bậc cao mà không đòi hỏi nhiều thời gian.

iii.) Giải pháp thứ ba dựa vào kết quả của một định lý toán học (định lý 3.5, trang 84, [4]) mà theo đó chúng ta có thể tính được đa thức $I(q,n;x)$ là đa thức tích của tất cả các đa thức bất khả quy có bậc n trên trường F_q. Sau đó chỉ cần phân tích đa thức tích này ra thành các nhân tử bất khả quy là ta đã có thể

[1]Trường Đại học Đồng Nai
Email: aianhnguyen80@gmail.com
[2]Trường Đại học Sư phạm Kỹ thuật TP. HCM

tìm được tất cả các đa thức bất khả quy bậc n. Bài toán phân tích đa thức thành các nhân tử bất khả quy đã được nhiều tác giả tham gia nghiên cứu.

Quá trình xây dựng cơ sở dữ liệu các đa thức bất khả quy của chúng ta bao gồm hai giai đoạn. Ở giai đoạn đầu chúng ta sẽ xây dựng một cơ sở dữ liệu bao gồm đầy đủ các đa thức bất khả quy tới một ngưỡng ở bậc n nào đó (chẳng hạn $n = 25$). Ở giai đoạn sau chúng ta sẽ tổng hợp các đa thức bất khả quy bậc cao dựa vào kho cơ sở dữ liệu đã xây dựng được ở giai đoạn đầu.

Cấu trúc của bài báo như sau:

- Phần 2: khảo sát sự phân bố của các đa thức bất khả quy trên trường hữu hạn F_q, các công trình liên quan và một vài ví dụ ứng dụng điển hình.
- Phần 3: trình bày quá trình xây dựng cơ sở dữ liệu các đa thức bất khả quy bao gồm hai bước như đã nói ở trên.
- Phần 4: mô tả quá trình cài đặt thử nghiệm, so sánh kết quả và đánh giá hệ thống.
- Phần 5: kết luận và hướng phát triển.

2. Sự phân bố và ứng dụng của đa thức bất khả quy trên trường hữu hạn F_q

Trong phần này chúng tôi trình bày ý nghĩa thực tiễn của sự phân bố các đa thức bất khả quy trên trường hữu hạn F_q và nêu lên hai ứng dụng điển hình.

2.1. Sự phân bố của các đa thức bất khả quy trên trường hữu hạn F_q

Đa thức bậc n trên trường hữu hạn F_q là một biểu thức có dạng:

$$f(x) = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + \cdots + a_n x^n, \quad a_n \neq 0$$

trong đó $a_i \in F_q, i = 0, 1, \ldots, n$.

Đa thức bất khả quy là đa thức không thể phân tích thành tích của hai đa thức có bậc lớn hơn hay bằng 1. Một dạng đặc biệt của đa thức bất khả quy có nhiều ứng dụng trong thực tiễn là đa thức nguyên thủy. Đa thức nguyên thủy $f(x)$ là phần tử sinh của trường $F_q[x]/f(x)$ ($F_q[x]$ là tập tất cả các đa thức có hệ số lấy trong trường F_q).

Tập các đa thức bất khả quy là vô hạn. Ở mỗi bậc $n \geq 1$ đều có chứa một số lượng nhất định các đa thức bất khả quy [4]. Bảng 1(thực hiện tính toán theo công trình [4], Theorem 3.24, p. 84) cho biết sự phân bố của các đa thức bất khả quy trên trường nhị phân F_2 ($q = 2$) với các giá trị của bậc $n \leq 20$.

Bảng 1: *Sự phân bố theo bậc của các đa thức bất khả quy trên trường nhị phân F_2*

Bậc	ĐT BKQ	ĐT	Tỷ lệ (%)	Bậc	ĐT BKQ	ĐT	Tỷ lệ (%)
1	2	2	100	**11**	186	2048	9,08
2	1	4	25	**12**	335	4096	8,17
3	2	8	25	**13**	630	8192	7,69
4	3	16	18,75	**14**	1161	16384	7,08
5	6	32	18,75	**15**	2182	32768	6,65
6	9	64	14,06	**16**	4080	65536	6,22
7	18	128	14,06	**17**	7710	131072	5,88
8	30	256	11,71	**18**	14532	262144	5,54
9	56	512	10,93	**19**	27594	524288	5,26
10	99	1024	9,66	**20**	52377	1048576	4,99

Ta có nhận xét là khi bậc n càng cao thì tỷ lệ các đa thức bất khả quy trên tổng số các đa thức giảm, nhưng số lượng các đa thức bất khả quy tăng lên ngày càng lớn. Trên trường F_q bất kỳ ($q > 2$) các giá trị này sẽ lớn hơn nhiều. Số lượng lớn các đa thức bất khả quy là tiền đề cần thiết cho việc thực hiện mã hóa mà không sợ bị tấn công theo kiểu vét cạn (Brute-Force search). Hơn nữa, quá trình phát sinh khóa đòi hỏi phải chọn ngẫu nhiên một đa thức và kiểm tra xem đó có phải là đa thức bất khả quy hay không [3]. Số lượng lớn các đa thức bất khả quy đảm bảo cho quá trình này không phải lặp lại nhiều lần và do đó mới có giá trịứng dụng thực tiễn.

2.2. Một số công trình liên quan đến phân tích đa thức và tổng hợp đa thức

Bài toán phân tích đa thức thành các nhân tử bất khả quy đã được các nhà khoa học tham gia nghiên cứu từ rất sớm [7]. Có những giải pháp tổng quát và toàn diện, cũng có những giải pháp chỉ giải quyết một phần của vấn đề hoặc đối với một số dạng đa thức đặc biệt như Tonelli (1891) và Schoof (1985): chỉ phân tích các đa thức bậc 2 trên trường hữu hạn F_p. Arwin (1918) chỉ tách các thừa số có bậc khác nhau, tức là phân tích thành tích của các thừa số có cùng bậc, do đó chưa phải là thừa số bất khả quy. Berlekamp đề xuất hai giải thuật phân tích: một giải thuật tất định (deterministic) sử dụng công cụ đại số (1967) và một giải thuật ngẫu nhiên (1970) dựa theo phương pháp ngẫu nhiên của Collins và Knuth. Rabin giới thiệu một giải thuật ngẫu nhiên (1980) sau đó được cải tiến bởi Ben-Or (1981). Cantor và Zassenhaus (1981) đề xuất giải thuật ngẫu nhiên đầu tiên thực thi trong thời gian đa thức với không gian lưu trữ tuyến tính $O(n)$. Huang (1991) đưa ra giải thuật có thời gian thực thi tốt nhất nhưng lại phụ thuộc vào giả thuyết Riemann mở rộng (chưa được chứng minh). Chúng ta sẽ sử dụng các phương pháp phân tích của Berlekamp và Cantor-Zassenhaus trong quá trình cài đặtcủa bài báo này.

Bài toán tổng hợp các đa thức bất khả quy từ các đa thức bất khả quy đã biết đã được công bố trong nhiều công trình nghiên cứu của các tác giả hoặc được tổng hợp lại trong một số công trình liên quan [4], [5], [8], [9], [10]. Đa thức ban đầu dùng để tổng hợp cần thỏa mãn một số điều kiện tổng hợp nào đó, các đa thức được tổng hợp có thể ở cùng trường với đa thức ban đầu, hoặc có thể ở trong các trường mở rộng của nó. Chúng tôi chọn cài đặt thử nghiệm những công thức tổng hợp đơn giản, hiệu quả tránh không phải thực hiện tính toán phức tạp trên các trường mở rộng. Các phương pháp này được đề cập trong các công trình [4], [5], [9]. Về việc xây dựng cơ sở dữ liệu các đa thức bất khả quy, chúng tôi chưa tìm thấy công trình nào được công bố.

Việc tìm ra công thức dùng để tổng hợp các đa thức bất khả quy bậc cao từ các đa thức bất khả quy bậc thấp hơn đã biết có ý nghĩa thực tiễn to lớn. Nó giúp ta tìm được các đa thức bất khả quy bậc cao mà không đòi hỏi phải tốn nhiều thời gian để kiểm tra tính bất khả quy. Công thức tổng hợp đa thức dựa trên các định lý toán học được đề cập đến trong các công trình tham khảo có liên quan. Trong bài báo này chúng tôi chọn lựa cài đặt những công thức tổng hợp đơn giản, hiệu quả và hữu dụng,

tránh việc thực hiện tính toán trên các trường mở rộng vốn khá phức tạp. Sau đây là một số phương pháp tổng hợp được sử dụng.

i.) Phương pháp Order-Based:

Phương pháp Order-Based [5] (Theorem 3.9, p. 44) cần một đa thức bất khả quy thỏa điều kiện tổng hợp và có thể phát sinh ra vô số các đa thức bất khả quy bậc cao tùy ý. Phương pháp này được cài đặt bởi hàm O_PolyGen(Polly p, int k) và cho kết quả là một đa thức bất khả quy (có chứa tham số) tổng hợp được với tham số nhận giá trị k, hoặc thông báo đa thức bất khả quy p đã cho không thỏa mãn điều kiện tổng hợp. Ví dụ, từ đa thức bất khả quy $x^2 + x + 1$ ta tổng hợp được đa thức bất khả quy chứa tham số $k, k \geq 0$: $x^{2.3^k} + x^{3^k} + 1$. Do đó:

O_PolyGen$(x^2 + x + 1, 1) = x^6 + x^3 + 1$

O_PolyGen$(x^2 + x + 1, 2) = x^{18} + x^9 + 1$

…

ii.) Phương pháp Primitive-Based:

Phương pháp Primitive-Based [9] (Theorem 11, p. 16) cần một đa thức nguyên thủy thỏa mãn điều kiện tổng hợp. Phương pháp này được cài đặt bởi hàm P_PolyGen(Poly p) và cho kết quả là một đa thức bất khả quy bậc $n(2^n - 1)$, trong đó n là bậc của đa thức nguyên thủy p, hoặc thông báo đa thức p không thỏa mãn điều kiện tổng hợp.

P_PolyGen

$(x^5 + x^4 + x^2 + x + 1)=$

$$x^{155} + x^{154} + x^{152} + x^{147} + x^{144} + x^{142} + x^{141} + x^{138} + x^{137} + x^{136} + x^{135} + x^{134} + x^{130} + x^{129} + x^{127} + x^{126} + x^{125} + x^{121} + x^{120} + x^{119} + x^{118} + x^{117} + x^{113} + x^{112} + x^{110} + x^{109} + x^{108} + x^{106} + x^{104} + x^{99} + x^{96} + x^{94} + x^{93} + x^{90} + x^{89} + x^{88} + x^{87} + x^{86} + x^{82} + x^{81} + x^{79} + x^{78} + x^{77} + x^{75} + x^{73} + x^{68} + x^{65} + x^{63} + x^{62} + x^{58} + x^{55} + x^{53} + x^{52} + x^{49} + x^{48} + x^{47} + x^{46} + x^{45} + x^{41} + x^{40} + x^{38} + x^{37} + x^{36} + x^{34} + x^{32} + x^{26} + x^{23} + x^{21} + x^{20} + x^{17} + x^{16} + x^{15} + x^{14} + x^{13} + x^9 + x^8 + x^6 + x^5 + x^4 + x^2 + 1$$

Phương pháp Varshamov:

Phương pháp Varshamov [5] (Theorem 3.19, p. 49) cần một đa thức nguyên thủy thỏa mãn điều kiện tổng hợp và cho kết quả là một họ các đa thức bất khả quy được tính thông qua các công thức truy hồi. Phương pháp này được cài đặt thông qua hàm V_PolyGen(Poly p, int k) và trả về đa thức thứ k của họ đa thức tổng hợp được. Cũng giống như phương pháp Order-Based, phương pháp này cho kết quả là vô số đa thức bất khả quy với bậc lớn tùy ý, hoặc thông báo đa thức đã cho không thỏa mãn điều kiện tổng hợp.

V_PolyGen$(x^5 + x^2 + 1, 0) = x^{10} + x^9 + x^5 + x^4 + x^2 + x + 1$

V_PolyGen$(x^5 + x^2 + 1, 1) = x^{20} + x^{17} + x^{12} + x^{10} + x^6 + x^5 + x^2 + x + 1$

…

iii.) Phương pháp Kyuregyan

Phương pháp Kyuregyan [9] (Theorem 5, Corollary 2, p. 11) cần một đa thức bất khả quy bậc n và một đa thức nguyên thủy bậc m thỏa mãn các điều kiện tổng hợp và cho kết quả là một đa thức bất khả quy bậc $n(2^m - 1)$. Phương pháp này được cài đặt thông qua hàm K_PolyGen(Poly p, Poly q) và cho kết quả là một đa thức bất khả quy hoặc thông báo đa thức đã cho không thỏa mãn điều kiện tổng hợp.

K_PolyGen

$$(x^3+x+1, x^5+x^4+x^2+x+1) = x^{93}+x^{91}+x^{90}+x^{89}+x^{86}+x^{84}+x^{83}+x^{82}+x^{79}+x^{76}+x^{74}+x^{73}+x^{72}+x^{69}+x^{67}+x^{66}+x^{62}+x^{61}+x^{60}+x^{57}+x^{55}+x^{54}+x^{53}+x^{50}+x^{48}+x^{47}+x^{46}+x^{45}+x^{42}+x^{40}+x^{39}+x^{38}+x^{37}+x^{34}+x^{32}+x^{31}+x^{29}+x^{28}+x^{27}+x^{24}+x^{22}+x^{20}+x^{19}+x^{18}+x^{17}+x^{15}+x^{14}+x^{11}+x^8+x^7+x^5+x^4+x^3+x^2+1$$

So sánh giữa các phương pháp tổng hợp

- Phương pháp Order-Based và Varshamov (có tính chất đệ quy) cần một đa thức bất khả quy và phát sinh ra vô số đa thức bất khả quy có bậc cao tùy ý. Phương pháp Primitive-Based cần một đa thức nguyên thủy, phương pháp Kyuregyan cần một đa thức nguyên thủy và một đa thức bất khả quy. Tuy nhiên, cả hai phương pháp này đều cũng chỉ phát sinh (nếu thỏa điều kiện tổng hợp) ra một đa thức bất khả quy khác.
- Phương Primitive-Based không cần thêm điều kiện tổng hợp nào. Các điều kiện tổng hợp của hai phương pháp Order-Based và Varshamov tương đối dễ thỏa mãn. Điều kiện tổng hợp của phương pháp Kyuregyan là khó thỏa mãn nhất.

2.3. Một vài ứng dụng điển hình của các đa thức bất khả quy

Đa thức bất khả quy có nhiều ứng dụng thực tiễn [4]. Chúng tôi trình bày sơ lược hai ứng dụng điển hình: tạo ra bộ sinh số ngẫu nhiên từ đa thức nguyên thủy, mã hóa khóa công khai và chữ ký số dùng đa thức bất khả quy.

2.3.1. Tạo ra bộ sinh số ngẫu nhiên

Bộ sinh số ngẫu nhiên có vai trò quan trọng trong nhiều ứng dụng thực tiễn, chẳng hạn khóa trong các quá trình mã hóa đều được phát sinh một cách ngẫu nhiên.

Bộ sinh số ngẫu nhiên Tausworthe được xây dựng bằng cách tạo ra chuỗi số nhị phân ngẫu nhiên rồi chọn các khối l số nhị phân liên tiếp, mỗi khối được xem như là biểu diễn của một số nguyên $x \in [0, 2^l - 1]$. Sau đó ta tính kết quả $x/2^l$ của mỗi giá trị x để có được các số ngẫu nhiên trong khoảng $[0,1)$.

Đa thức nguyên thủy có tác dụng phát sinh ra chuỗi số nhị phân ngẫu nhiên, do đó có thể dùng để tạo ra bộ tạo số ngẫu nhiên Tausworthe. Để đơn giản hóa việc tính toán người ta thường chọn đa thức nguyên thủy dạng tam thức $x^p + x^q + 1$. Đa thức có bậc càng cao thì bộ sinh số ngẫu nhiên có quy luật càng giống với các quá trình ngẫu nhiên trong thực tế. Bộ sinh số ngẫu nhiên Tausworthe được xây dựng từ đa thức nguyên thủy được mô tả chi tiết trong công trình [9]. Chúng tôi trình bày tóm tắt qua một ví dụ cụ thể với đa thức nguyên thủy $f(x) = x^5 + x^2 + 1$ và chọn khối với giá trị $l = 5$.

Gọi $Z_2[x]$ là tập các đa thức có hệ số trên trường nhị phân Z_2, khi ấy vành thương $Z_2[x]/(f(x))$ là một trường bao gồm các phần tử là các đa thức có bậc $n < 5$. Trên trường này $f(x) \equiv 0 \ (mod\ f(x))$ nên ta có:

$x^5 + x^2 + 1 = 0 = 2x^5$(trên trường nhị phân $2t = 0$)

Do đó:

$$x^5 = x^2 + 1$$
$$x^6 = x.x^5 = x(x^2+1) = x^3 + x$$
$$x^7 = x.x^6 = x(x^3+x) = x^4 + x^2$$
$$x^8 = xx^7 = x(x^4+x^2) = x^5 + x^3 = x^3 + x^2 + 1$$

$x^9 = x.x^8 = x(x^3 + x^2 + 1) = x^4 + x^3 + x$

$x^{10} = x.x^9 = x(x^4 + x^3 + x) = x^5 + x^4 + x^2 = x^2 + 1 + x^4 + x^2 = x^4 + 1$

.

.

.

Nếu ta tính giá trị của các lũy thừa $x^0 = 1, x^1 = x, x^2, x^3, x^4, x^5, \ldots$ tại $x = 1$ rồi nhóm lại theo từng cụm $l = 5$ số (nhị phân) thì các chuỗi số nhị phân này là ngẫu nhiên và chúng được sử dụng để tạo ra bộ sinh số ngẫu nhiên Tausworthe như được mô tả trong [11].

2.3.2. Mã hóa khóa công khai và chữ ký số

Đa thức bất khả quy có thể dùng để thay thế cho số nguyên tố trong mã hóa khóa công khai và chữ ký số, quá trình ký và mã hóa được mô tả ngắn gọn thông qua một ví dụ cụ thể ở ngay bên dưới. Ramzi A. Haraty và các đồng nghiệp [3] đã làm thực nghiệm so sánh giữa hai phương pháp RSA cổ điển (dựa trên số nguyên tố) và RSA dựa trên đa thức và đã đưa ra một số kết luận như sau:

- Tấn công hệ thống RSA cổ điển rất dễ đối với các số nguyên tố nhỏ. Tuy nhiên, đối với các số nguyên tố có từ 100 chữ số trở lên thì hệ thống rất khó tấn công và cần phải sử dụng nhiều máy tính chạy song song.
- Tấn công hệ thống RSA dựa trên đa thức trở nên khó hơn khi kích thước của p tăng hoặc khi bậc của đa thức lớn.

Quá trình tạo khóa (của người gửi *A*) bao gồm các bước sau:

- Chọn ngẫu nhiên 1 số nguyên tố lẻ p và 2 đa thức bất khả quy $f(x)$ và $g(x)$ trên trường hữu hạn Z_p (rất tốn thời gian đối với đa thức bất khả quy bậc cao để đảm bảo tính an toàn của hệ thống).
- Tính $h(x) = f(x).g(x)$
- Tính $\phi(h(x)) = (p^r - 1)(p^s - 1)$ với r và s lần lượt là bậc của $f(x)$ và $g(x)$
- Chọn ngẫu nhiên số nguyên $e, 1 < e < \phi(h(x))$ sao cho $\gcd(e, \phi(h(x))) = 1$
- Sử dụng thuật toán chia Euclidean để xác định số nguyên $d, 1 < d < \phi(h(x))$ sao cho $ed \equiv 1 \ (mod\ \phi(h(x)))$ (d được xác định ở bước này là duy nhất)
- Công bố khóa công khai $(p, h(x), e)$ và giữ d là khóa bí mật

Quá trình ký (của người gửi *A*) bao gồm các bước sau:

- Nhận thông điệp $m(x)$ từ trong hệ thặng dư đầy đủ modulo $f(x)$ của $Z_p[x]$
- Tính $m_1(x) = R(m(x))$ là một đa thức trong hệ thặng dư đầy đủ modulo $h(x)$ của $Z_p[x]$
- Sử dụng khóa bí mật d để tính $s(x) = (m_1(x))^d \ (mod h(x))$
- Gửi thông điệp đã ký và mã hóa $s(x)$ của nguyên bản $m(x)$

Quá trình chứng thực (của người nhận *B*) bao gồm các bước sau:

- B nhận khóa công khai $(p, h(x), e)$ của A
- B tính $(s(x))^e$ $(=m_1(x))$
- B phục hồi nguyên bản $m(x)$ bằng cách tính $R^{-1}(m_1(x))$

Ví dụ với các bộ số cụ thể:

- A Chọn $p = 389$, $f(x) = x^2 + 376x + 43$, $g(x) = x^3 + 384x^2 + 3x + 10$ trên $Z_{389}[x]$, và tính:

$h(x) = f(x).g(x) = x^5 + 371x^4 + 111x^3 + 145x^2 + 388x + 41$,

$\phi(h(x)) = (389^3 - 1)(389^2 - 1) = 8907280505760$,

Sau đó A chọn $e = 95561135039$ và tính được $d = 5878808345759$.

Giả sử cần mã hóa thông điệp $m(x) = 1 + 3x + x^2$, A thực hiện tính toán sau:

$m_1(x) = R(m(x)) = m(x) = 1 + 3x + x^2$,

$s(x) = (m_1(x))^d = (1 + 3x + x^2)^{5878808345759} = 172x^4 + 86x^3 + 265x^2 + 59x + 177\ (modh(x))$,

A gửi cho B thông điệp đã được mã hóa và ký là $s(x)$.

- *B thực hiện quá trình xác thực và giải mã bằng cách tính:*

$[s(x)]^e = (172x^4 + 86x^3 + 265x^2 + 59x + 177)^{95561135039} = (1 + 3x + x^2)(modh(x)) = m_1(x)$,

$m(x) = R^{-1}(1 + 3x + x^2) = 1 + 3x + x^2$.

Để đơn giản trong cả hai quá trình ký và chứng thực ở trên chúng ta đều dùng hàm mã hóa $R(m)$ là hàm đồng nhất ($R(m) = m$), cũng tức là không có mã hóa và do đó $R^{-1}(m) = m$.

Tăng tính hiệu quả cho quá trình phát sinh khóa bằng cách sử dụng công thức tổng hợp đa thức:

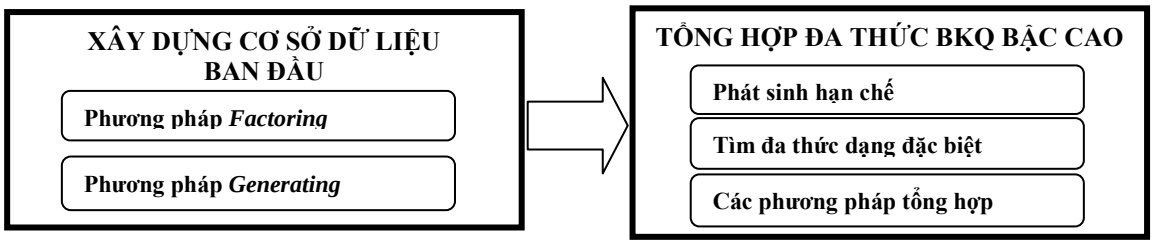
Với kho cơ sở dữ liệu các đa thức bất khả quy đã được xây dựng sẵn, chúng ta có thể tăng tính hiệu quả cho quá trình phát sinh khóa (chọn $f(x)$ và $g(x)$) trong quá trình tạo khóa bằng cách sau:

- Chọn ngẫu nhiên một đa thức bất khả quy trong kho cơ sở dữ liệu các đa thức bất khả quy (con số hữu hạn).
- Dùng công thức tổng hợp để tính ra một đa thức bất khả quy khác và dùng làm khóa (con số này có thể là vô hạn, nếu ta sử dụng các công thức tổng hợp có tính chất đệ quy).

Xét ví dụ sau: trong cơ sở dữ liệu có chứa đa thức bất khả quy trên trường nhị phân F_2: $x^2 + x + 1$. Dựa vào công thức tổng hợp ta biết các đa thức có dạng $f(x) = x^{2.3^k} + x^{3^k} + 1, k \geq 1$ cũng là các đa thức bất khả quy và số lượng các đa thức này là vô hạn. Do đó, chúng ta có thể chọn ngẫu nhiên một giá trị k và dùng $f(x)$ để làm khóa. Việc tính toán $f(x)$ không mất nhiều thời gian, không gian khóa là vô hạn, điểm yếu là khóa có cấu trúc khá đặc biệt có thể giúp cho các nhà phân tích mã có phương pháp tấn công đặc biệt nào đó đối với hệ mã này.

3. Xây dựng cơ sở dữ liệu các đa thức bất khả quy

Quá trình xây dựng cơ sở dữ liệu các đa thức bất khả quy bao gồm hai giai đoạn: giai đoạn xây dựng cơ sở dữ liệu ban đầu và giai đoạn tổng hợp đa thức như đã trình bày ở phần giới thiệu. Chúng ta chỉ cần xét các đa thức bất khả quy có hệ số đầu bằng 1 ($a_n = 1$). Hình 1 trình bày quá trình xây dựng cơ sở dữ liệu các đa thức bất khả quy.



Hình 1: *Quá trình xây dựng cơ sở dữ liệu các đa thức bất khả quy*

3.1. Xây dựng cơ sở dữ liệu ban đầu

Chúng ta sử dụng hai phương pháp để xây dựng cơ sở dữ liệu ban đầu: phương pháp Factoring và phương pháp Generating. Đối với phương pháp Factoring, ta không cần quan tâm đến các đa thức khả quy. Theo [4] (theorem 3.5, p. 84) ta có công thức tính tích của tất cả các đa thức bất khả quy bậc nvới hệ số $a_i \in F_q$. Để tìm tất cả các đa thức bất khả quy bậc n chúng ta chỉ việc phân tích đa thức tích $I(q,n;x)$ vừa tính được ra thành các nhân tử bất khả quy.Chúng tôi sử dụng hai phương pháp phân tích: phương pháp tất định Berlekamp [4] (p. 130) và phương pháp ngẫu nhiên Cantor-Zassenhaus [6] (p. 530). Ví dụ, để tìm tất cả các đa thức bất khả quy bậc 4 trên trường nhị phân F_2 chúng ta phân tích đa tích $I(2,4;x) = x^{12} + x^9 + x^6 + x^3 + 1$ ra thành tích các nhân tử bất khả quy (có tất cả ba đa thức bất khả quy bậc 4, phù hợp với công thức tính số lượng các đa thức bất khả quy bậc n trong [4]):

$$I(2,4;x) = x^{12} + x^9 + x^6 + x^3 + 1 = (x^4 + x^3 + x^2 + x + 1)(x^4 + x^3 + 1)(x^4 + x + 1)$$

Phương pháp Generating dựa vào giải thuật kiểm tra tính bất khả quy của đa thức: giải thuật IPT trong [6] (Theorem 19.10, p. 513).

Giải thuật IPT

Input: đa thức f có bậc n
Output: true hoặc false
$h \leftarrow x \bmod f$
for $i \leftarrow 1$ to $\lfloor n/2 \rfloor$ do
    $h \leftarrow h^q \bmod f$
    if $\gcd(h - x, f) \neq 1$
        return false
    endif
endfor
return true

Ta lần lượt phát sinh ra tất cả các đa thức bậc n rồi dùng giải thuật IPT để kiểm tra và loại bỏ các đa thức khả quy. Kết quả ta sẽ nhận được tất cả các đa thức bất khả quy bậc n. Khác với phương pháp Factoring phương pháp này phải xem xét tất cả các đa thức bậc n. Ví dụ, phát sinh tất cả các đa thức bậc 4 dạng $x^4 + a_3x^3 + a_2x^2 + a_1x + 1$ với $a_1, a_2, a_3 \in F_2$. Sau quá trình kiểm tra và loại bỏ kết quả thu được là ba đa thức bất khả quy như trong phương pháp Factoring ở trên.

3.2. Tổng hợp các đa thức bất khả quy bậc cao

Số lượng các đa thức bất khả quy tăng lên rất nhanh theo bậc, do đó việc tìm tất cả các đa thức bất khả quy ở bậc cao là bất khả thi và đôi khi không thực sự cần thiết. Chúng ta sẽ sử dụng các phương pháp sau đây để tìm các đa thức bất khả quy ở bậc cao:

- Phát sinh các đa thức bất khả quy với số lượng hạn chế.
- Tìm các đa thức bất khả quy dạng đặc biệt.

- Tổng hợp từ các đa thức bất khả quy bậc thấp hơn.

3.2.1. Phát sinh các đa thức bất khả quy bậc cao với số lượng hạn chế

Giả sử ta cần tìm k đa thức bất khả quy bậc n. Ta có thể thực hiện điều này bằng cách lần lượt phát sinh ngẫu nhiên một đa thức bậc n, dùng giải thuật IPT đã trình bày ở trên để kiểm tra xem đó có phải là đa thức bất khả quy chưa được phát sinh, nếu đúng thì sẽ thêm vào tập kết quả. Lặp lại quá trình này cho đến khi ta có đủ số lượng các đa thức bất khả quy cần thiết. Hàm R_PolyGen(n, k) phát sinh k đa thức bất khả quy bậc n.

3.2.2. Tìm các đa thức bất khả quy dạng đặc biệt

Đa thức bất khả quy dạng đặc biệt thường được nghiên cứu là nhị thức (binomial) (x^n+1) và tam thức (trinomial) (x^n+x^m+1). Trường nhị phân F_2 không có các đa thức bất khả quy dạng nhị thức. Thật vậy, khi n chẵn thì x^n+1 chia hết cho $x^{n/2}+1$ (do $x^n+1=(x^{\frac{n}{2}}+1)^2$), ngược lại khi n lẻ thì nó chia hết cho $x+1$. Do đó ta sẽ thực hiện việc tìm các đa thức bất khả quy dạng tam thức trên trường nhị phân F_2.

Về cơ bản chúng ta vẫn sẽ sử dụng giải thuật IPT ở trên, tuy nhiên chúng ta có thể giảm thiểu việc kiểm tra tính bất khả quy của các đa thức dạng tam thức bằng cách sử dụng phương pháp được mô tả trong [12] (Corollary 5, p. 1105) do Swan đề xuất. Định lý này giúp loại bỏ được phần lớn các trường hợp cần phải kiểm tra, điều này rất có ý nghĩa thực tiễn vì đối với đa thức bậc cao việc kiểm tra tính bất khả quy đòi hỏi một lượng thời gian đáng kể. Cũng theo Swanhai đa thức $T_{n,k}(x)$ và $T_{n,n-k}(x)$ có cùng tính chất bất khả quy, do đó trong trường hợp n và k đều lẻ thì việc khảo sát đa thức $T_{n,n-k}(x)$ sẽ quy về việc khảo sát đa thức $T_{n,k}(x)$. Từ đó ta chỉ cần xét tất cả các đa thức dạng $T_{n,k}(x)$ với $k \le n/2$ là đủ. Hàm T_PolyGen(n) phát sinh tất cả các tam thức bất khả quy bậc n.

3.2.3. Sử dụng công thức tổng hợp

Sau khi đã xây dựng được cơ sở dữ liệu ban đầu bao gồm đầy đủ tất cả các đa thức bất khả quy có bậc $n \le 25$ trên trường nhị phân F_2, chúng tôi thực hiện một vòng lặp và đối với mỗi đa thức lần lượt thực hiện các bước sau:

- Kiểm tra điều kiện tổng hợp của cả bốn phương pháp tổng hợp đã nêu ở mục 2.2 bên trên.
- Nếu thỏa điều kiện tổng hợp thì sẽ phát sinh các đa thức tổng hợp tương ứng và lưu trữ vào cơ sở dữ liệu.

Trong thực tế, quá trình tổng hợp này có thể thực hiện lặp lại nhiều lần và kết quả là kho cơ sở dữ liệu các đa thức bất khả quy ngày càng được mở rộng, cho đến khi đạt được yêu cầu của ứng dụng thực tiễn. Tuy nhiên, trong quá trình cài đặt thử nghiệm chúng tôi không thực hiện quá trình lặp này.

3.3. Cấu trúc lưu trữ cho các đa thức bất khả quy

Chúng ta cần chọn lựa cách thức lưu trữ các đa thức bất khả quy sao cho việc tìm kiếm có thể được thực hiện dễ dàng và nhanh chóng. Một trong những cách có hiệu quả nhất là sử dụng một hệ quản trị cơ sở dữ liệu chuyên nghiệp (SQL Server chẳng hạn) và lập chỉ mục tìm kiếm một cách thích hợp.

Nội dung cần lưu trữ là chuỗi nhị phân biểu diễn các hệ số của đa thức trên trường nhị phân F_2. Tuy nhiên, tìm kiếm trên chuỗi nhị phân này không

hiệu quả, do đó để tăng tốc độ cho quá trình tìm kiếm ta sử dụng hàm băm GetHashCode có các tính chất sau:

- Đối số của hàm là một chuỗi nhị phân, kết quả trả về là 1 số nguyên.
- Hai chuỗi nhị phân khác nhau có thể có cùng giá trị của hàm băm (trường hợp đụng độ hay tranh chấp, tuy nhiên điều này rất ít khi xảy ra do mục đích thiết kế của hàm băm).

Ngoài ra ta sử dụng thêm các thông tin khác nữa để lập chỉ mục tìm kiếm theo các yêu cầu khác nhau của ứng dụng thực tiễn cũng như để giải quyết trang chấp nếu có. Các thao tác cập nhật như thêm, xóa, sửa và tìm kiếm trên cơ sở dữ liệu được mô tả chi tiết bên dưới. Cấu trúc lưu trữ của một đa thức bất khả quy trong cơ sở dữ liệu được trình bày trong bảng 2.

Bảng 2: *Cấu trúc lưu trữ cho các đa thức bất khả quy*

ColumnName	Description	Primary Key
HashCode	Giá trị hàm băm của chuỗi nhị phân biểu diễn các hệ số của đa thức	Yes
Degree	Bậc của đa thức	No
SumBit	Tổng các hệ số của đa thức, cho biết số các hệ số có giá trị bằng 1 của đa thức	No
GroupOrder	Số thứ tự của đa thức trong nhóm các đa thức có cùng giá trị của HashCode và Degree, có tác dụng giải quyết tranh chấp nếu có	Yes
BitString	Chuỗi nhị phân biểu diễn các hệ số của đa thức	No
IsPrimitive	Đa thức có phải là đa thức nguyên thủy?	No

3.3.1. Thêm một đa thức bất khả quy vào cơ sở dữ liệu

Thao tác thêm một đa thức bất khả quy vào cơ sở dữ liệu được cài đặt thông qua hàm AddPoly(Poly `p`, bool *needCheck*) bao gồm các bước sau:

- Tính giá trị của hàm băm của chuỗi nhị phân biểu diễn các hệ số của đa thức.
- Xác định bậc của đa thức.

Dựa vào các giá trị vừa tính toán ở trên ta thực hiện tìm kiếm trong cơ sở dữ liệu. Nếu tìm không thấy ta sẽ thêm đa thức vào cơ sở dữ liệu với GroupOrder có giá trị bằng 0. Ngược lại ta có thể tìm thấy một nhóm các đa thức bất khả quy có cùng giá trị của hàm băm (danh sách này không nhiều, do tính chất hiếm khi đụng độ của hàm băm). Nếu đa thức cần thêm không có trong danh sách này thì ta thực hiện thêm đa thức với GroupOrder có giá trị là giá trị lớn nhất của GroupOrder trong nhóm cộng thêm 1. Trong trường hợp đa thức đã tồn tại trong cơ sở dữ liệu thì không cần phải thực hiện thao tác thêm nữa. Trong trường hợp chưa xác định được đa thức cần thêm có phải là đa thức bất khả quy hay không (*needCheck* bằng false), ta cần phải thực hiện thêm thao tác kiểm tra tính bất khả quy trước khi thêm vào cơ sở dữ liệu (đối với giải thuật phát sinh ra đa thức bất khả quy không cần phải kiểm tra nữa).

3.3.2. Xóa đi một đa thức trong cơ sở dữ liệu

Thao tác xóa đi một đa thức trong cơ sở dữ liệu được cài đặt thông qua

hàm DeletePoly(Poly p). Thao tác xóa thường được thực hiện sau thao tác tìm kiếm. Ta tính toán các thông tin HashCode, Degree, và GroupOrder của đa thức cần xóa để thực hiện thao tác xóa đa thức. Các thông tin này tạo thành khóa chính nên xác định duy nhất đa thức cần phải xóa. Trong thực tế thao tác xóa hiếm khi sử dụng.

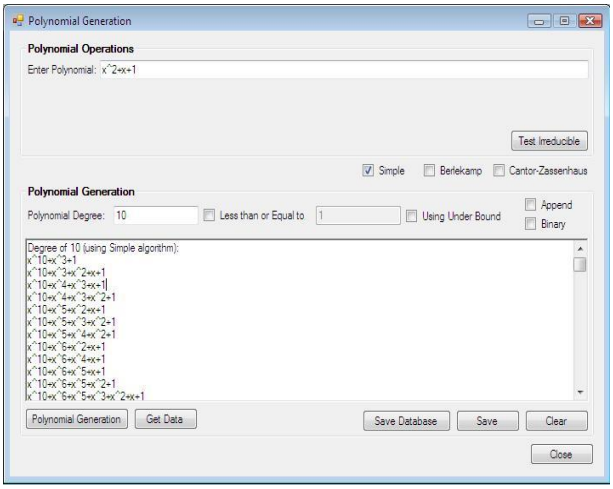
Vấn đề tìm kiếm đa thức khi xóa được thực hiện dựa trên các thuộc tính đã được lập chỉ mục. Hệ quản trị cơ sở dữ liệu SQL Server lưu trữ dữ liệu dưới dạng cấu trúc cây mở rộng (BTree++) nên thực hiện tìm kiếm sẽ nhanh chóng và hiệu quả hơn nhiều so với phương pháp tìm kiếm tuần tự, đặc biệt là đối với khối lượng dữ liệu lớn.

3.3.3. Cập nhật đa thức

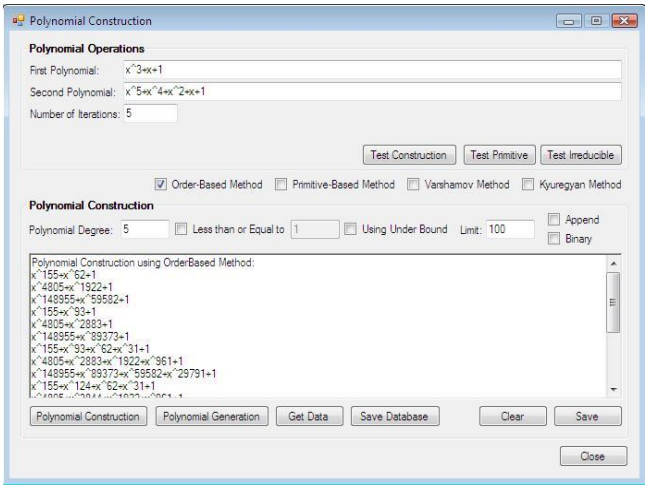
Khi thay đổi nội dung (chuỗi nhị phân biểu diễn các hệ số của đa thức) của một đa thức, các thông tin tương ứng như HashCode, Degree, SumBit cũng thay đổi theo. Do đó, thao tác cập nhật thật ra là thao tác thêm đa thức mới và xóa đi đa thức cũ. Giống như thao tác xóa, thao tác cập nhật cũng hiếm khi được sử dụng trong thực tế.

4. Kết quả cài đặt thử nghiệm

Cài đặt thử nghiệm được thực hiện trên môi trường Windows (hệ điều hành Vista) với công cụ Visual Studio 2008, sử dụng ngôn ngữ lập trình C# và hệ quản trị cơ sở dữ liệu SQL Server 2008. Máy tính cài đặt chương trình có cấu hình CPU Pentium Dual-Core, tốc độ 2GHz, bộ nhớ trong 1GB. Một số giao diện chính của chương trình như hình 2.



Polynomial Generation



Polynomial Construction

Hình 2: *Một số giao diện chính của chương trình*

4.1. Xây dựng cơ sở dữ liệu ban đầu

Cơ sở dữ liệu ban đầu bao gồm tất cả các đa thức bất khả quy trên trường nhị phân F_2 có bậc $n \leq 25$. Phương pháp Generating được cài đặt thông qua hàm SimplePolyGen(int n) với n là số bậc tối đa của đa thức bất khả quy cần phát sinh. Kết quả của hàm là tất cả các đa thức bất khả quy có bậc n.

Phương pháp Factoring được cài đặt thông qua hai hàm BerlekampPolyGen(int n) và Cantor-Zassenhaus(int n). Cả hai hàm này đều cho kết quả giống như hàm SimplePolyGen(int n). Hàm

BerlekampPolyGen sử dụng phương pháp phân tích Berlekamp trong khi hàm Cantor-Zassenhaus sử dụng phương pháp phân tích Cantor-Zassenhaus.

Bảng 3 so sánh hiệu quả của các phương pháp (ký hiệu "-" chỉ thời gian khá lâu, không ghi nhận). Phương pháp Berlekamp hiệu quả hơn phương pháp Cantor-Zassenhaus ở bậc n nhỏ. Phương pháp Simple đạt hiệu quả cao nhất. Điều này chứng tỏ rằng (cũng giống như bài toán phân tích số nguyên thành tích của các số nguyên tố), bài toán phân tích đa thức thành các nhân tử bất khả quy là một bài toán khó và đòi hỏi nhiều thời gian. Kết quả thực nghiệm này hoàn toàn phù hợp với lý thuyết.

4.2. Tổng hợp các đa thức bất khả quy bậc cao

Phương pháp thứ nhất dùng để tổng hợp các đa thức bất khả quy bậc cao là phát sinh đa thức bậc cao với số lượng hạn chế. Trong chương trình chúng tôi phát sinh 100 đa thức bất khả quy ứng với mỗi bậc $26 \le n \le 500$. Phương pháp thứ hai là tìm các đa thức bất khả quy bậc cao dạng đặc biệt (dạng tam thức $x^n + x^m + 1$). Hai phương pháp này thực sự không sử dụng công thức tổng hợp đa thức nhưng được thực hiện ở giai đoạn hai nên xếp vào mục tổng hợp cho tiện.

Phương pháp thứ ba sử dụng các phương pháp tổng hợp khác nhau. Chúng ta quét kho ban đầu và thực hiện công thức tổng hợp đa thức. Trong khuôn khổ bài báo này, chúng tôi chỉ sử dụng các đa thức có bậc $n \le 10$ để tổng hợp đa thức bất khả quy bậc cao.

Bảng 3: *So sánh hiệu quả của các giải thuật phân tích*

Bậc	Simple	Berlekamp	Cantor	Bậc	Simple	Berlekamp	Cantor
1	0h:0':0''	0h:0':0''	0h:0':0''	**14**	0h:0':1''	-	-
2	0h:0':0''	0h:0':0''	0h:0':0''	**15**	0h:0':3''	-	-
3	0h:0':0''	0h:0':0''	0h:0':0''	**16**	0h:0':7''	-	-
4	0h:0':0''	0h:0':0''	0h:0':0''	**17**	0h:0':15''	-	-
5	0h:0':0''	0h:0':0''	0h:0':0''	**18**	0h:0':34''	-	-
6	0h:0':0''	0h:0':0''	0h:0':1''	**19**	0h:1':12''	-	-
7	0h:0':0''	0h:0':0''	0h:0':5''	**20**	0h:2':44''	-	-
8	0h:0':0''	0h:0':0''	0h:0':56''	**21**	0h:5':47''	-	-
9	0h:0':0''	0h:0':1''	0h:6':1''	**22**	0h:12':55''	-	-
10	0h:0':0''	0h:0':12''	1h:28'21''	**23**	0h:28':5''	-	-
11	0h:0':0''	0h:1':39''	-	**24**	1h:3':4''	-	-
12	0h:0':0''	0h:16':28''	-	**25**	2h:13':17''	-	-
13	0h:0':1''	1h:35':10''	-				

Kết quả của quá trình nghiên cứu lý thuyết và cài đặt thử nghiệm là chúng tôi đã đưa ra được một quá trình xây dựng cơ sở dữ liệu các đa thức bất khả quy một cách có hệ thống và thực tế đã xây dựng được một kho cơ sở dữ liệu các đa thức bất khả quy tương đối lớn (có tới hơn bốn triệu đa thức bất khả quy) bao gồm:

- Tất cả các đa thức bất khả quy có bậc $n \le 25$ trên trường nhị phân $F2$.
- Với mỗi $26 \le n \le 500$: phát sinh (ngẫu nhiên) 100 đa thức bất khả quy ở mỗi bậc.

- Tất cả các đa thức bất khả quy dạng tam thức có bậc $n \leq 2000$.
- Các đa thức bất khả quy bậc cao được tổng hợp theo các phương pháp khác nhau. Đa thức bất khả quy có bậc cao nhất trong cơ sở dữ liệu là 668168.
- Tất cả các đa thức bất khả quy có bậc $n \leq 10$trên trường F_5.

Chúng tôi chưa tìm được một công trình xây dựng cơ sở dữ liệu các đa thức bất khả quy nào tương tự được công bố để so sánh kết quả. Sau đây chúng tôi đưa ra một số tiêu chí dùng để đánh giá và so sánh kết quả giữa các hệ thống (trên môi trường tương tự nhau):

- Độ lớn của kho cơ sở dữ liệu ban đầu, ngưỡng n càng lớn càng tốt.
- Độ lớn của toàn bộ cơ sở dữ liệu.
- Sự phong phú của các dạng thức, càng nhiều công thức tổng hợp càng tốt.
- Số lượng các đa thức bất khả quy bậc cao hơn một ngưỡng n nào đó.
- Hiệu quả của các thuật toán phát sinh đa thức và tìm kiếm trên cơ sở dữ liệu.

5. Kết luận và hướng phát triển

Trong bài báo các tác giả đã trình bày quá trình xây dựng kho cơ sở dữ liệu các đa thức bất khả quy phục vụ các ứng dụng thực tiễn một cách hệ thống bao gồm nhiều cách thức khác nhau và một cấu trúc lưu trữ cho phép thực hiện tìm kiếm hiệu quả. Các tác giả cũng đã xây dựng được một kho cơ sở dữ liệu tương đối lớn. Khi cần thiết, kho cơ sở dữ liệu này có thể mở rộng thêm với độ lớn tùy ý và có thể phát sinh ra đa thức bất khả quy có bậc cao tùy ý. Các phương pháp tổng hợp mới được bổ sung vào hệ thống một cách dễ dàng. Ở bước đầu xây dựng nên hệ thống cũng có một số hạn chế nhất định: chẳng hạn cài đặt chủ yếu được thực hiện ở trường nhị phân F_2, các ví dụ áp dụng chỉ mang tính minh họa (tính lý thuyết), ứng dụng thực tiễn đòi hỏi phải có quá trình khảo sát chuyên sâu và nghiên cứu kỹ lưỡng hơn, đặc biệt là môi trường và phạm vi ứng dụng cụ thể.

Trong tương lai chúng tôi sẽ khảo sát kỹ lưỡng điều kiện áp dụng của các ứng dụng thực tiễn và cách khai thác có hiệu quả kho cơ sở dữ liệu các đa thức bất khả quy đã xây dựng được, đặc biệt là trong lĩnh vực mã hóa mật mã và an toàn thông tin, chẳng hạn, liệu có thực sự xây dựng được một hệ thống mã hóa khóa công khai dựa trên sự tổng hợp đa thức hay không.

TÀI LIỆU THAM KHẢO

1. James T. Cross (1983), "The Euler φ-Function in the Gaussian Integers" (1983), The American Mathematical Monthly, Vol. 90, No. 8, pp. 518-528

2. A.N. El-Kassar, Ramzi Haraty, Y. A. Awad, N. C. Debnath (2005), "Modified RSA in the domains of Gaussian integers and polynomials over finite fields", Proceedings of the ISCA 18th International Conference on Computer Applications in Industry and Engineering

3. Ramzi A. Haraty (2004), "A Comparative Study of RSA Based Digital Signature Algorithms", Proceedings of the Sixth International Conference on Enterprise Information Systems (ICEIS 2004), vol. 3, pp. 79-84

4. R. Lidl, H. Niederreiter (1986), "Introduction to finite fields and their applications", Cambridge University Press, London

5. Alfred J. Menezes, I.F. Blake, X. Gao, R.C. Mullin, S.A. Vanstone and T. Yaghoobian (1993), "Applications of Finite Fields", Kluwer Academic Publishers, Boston Dordrecht-Lancaster,Netherlands

6. Victor Shoup (2008), "A Computational Introduction to Number Theory and Algebra", Cambridge University Press, London

7. Erich Kaltofen (1992), "Polynomial Factorization 1987-1991", Proc. Latin'92, I. Simon (Ed.), Springer Lect. Notes Comput. Sci., vol 583, pp. 294-313

8. I.F. Gao and R. J. Lambert (1994), "Constructive problems for irreducible polynomials over finite fields", in Information Theory and Applications (A. Gulliver and N. Secord, eds.), LNCS 793, Springer-Verlag, Berlin

9. Melsik K. Kyuregyan, Gohar M. Kyureghyan (2009), "Irreducible Compositions of Polynomials over Finite Fields", Otto-von-Guericke University, Armenia

10. M.K. Kyuregyan (2004), "Iterated constructions of irreducible polynomials over finite fields with linearly independent roots", Finite Fields Appl. Vol. 10, pp. 323-431

11. Lewis, T.G. and W.H. Payne (1973), "Generalized Feedback Shift Register Pseudorandom Number Algorithm", Journal of the ACM,vol 3, pp. 456-468

12. R.G. Swan (1962), "Factorization of polynomials over finite fields", Pac. J. Math., vol 12 (1962), pp. 1099-1106

BUILDING DATABASE OF IRREDUCIBLE POLYNOMIALS FOR PRACTICAL APPLICATIONS

ABSTRACT

Irreducible polynomials have many practical applications, especially in coding theory, cryptography and information security. This article presents the research results and an initial implementation of building database of irreducible polynomials for practical applications.

Keywords: *Irreducible polynomials, finite fields, coding theory, cryptography, information security*

(Received: 9/5/2018, Revised: 22/6/2018, Accepted for publication: 19/3/2019)