ĐẠI HỌC KHOA HỌC TỰ NHIÊN – ĐHQG_HCM
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO: BÀI TẬP 2

MÔN: NHẬP MÔN MÃ HÓA

Chủ đề: Square - Saturation - Integral - Multiset Attacks

Nhóm: 13

Gồm các thành viên:

Phạm Lê Quốc Khánh - 20120116

Trương Ngọc Huy – 20120109

Phạm Đức Huy – 20120107

I. Phân công:

Người thực hiện	Nội dung	Các nội dung chưa hoàn thành	Mức độ hoàn thành
Trương Ngọc Huy	Lý thuyết: II Square - Saturation - Integral - Multiset Attacks	không có	100%
Phạm Đức Huy	Lý thuyết: III Ví dụ Square Attack	không có	100%
Phạm Lê Quốc Khánh	Thực hành	Chưa thực hiện thêm chữ ký số vào hệ mã hóa	100%
Cả nhóm	viết báo cáo	Không có	100%

II. Square - Saturation - Integral - Multiset Attacks:

A. Tổng quan:[2]

- Multiset cryptanalysis lần đầu tiên được đề cập trong bài viết "The Block Cipher SQUARE" của J. Daemen, V. Rijmen, and L. Knudsen dưới dạng một kiểu tấn công dành cho thuật toán SQUARE và được gọi là **SQUARE attack**.
- Một kiểu tấn công tương tự SQUARE attack gọi là "**Saturation**" attack, được Lucks dùng để giải mật mã Rijndael/AES.
- "Collision" attack của Gilbert–Minier trên 7-rounds của mã Rijndael cũng như "**Integral**" cryptanalysis của Knudsen–Wagner trên 5-rounds của mã MISTY1 cũng sử dụng những ý tưởng tương tự những kiểu tấn công trên.
- Tất cả các kiểu tấn công trên được gọi chung là **multiset attack**.
- Multiset attack là một kiểu **chosen-plaintext attack**.
- Trái với cryptanalysis tuyến tính hay cryptanalysis vi phân, multiset attack chỉ có thể lấy thông tin từ một **nhóm** các plaintext, với mỗi cặp plaintext sẽ cho một vài mẫu thông tin nhỏ.

B. Nguyên tắc cơ bản và Biện pháp phòng chống:[3]

1. Nguyên tác cơ bản:

- Xét dữ liệu đầu vào của một S-box song ánh là một tập các plaintext.
- Giả định với một giá trị được lặp lại k lần trong các plaintext sẽ có một giá trị tương ứng lặp lại k lần trong giá trị đầu ra của S-box. Nếu đầu vào S-box là một hằng số thì đầu ra cũng là hằng số.
- Multiset attack vận dụng những đặc điểm trên của S-box để giải mã nhiều rounds nhất có thể.

2. Một số thuật ngữ:

- Một multiset là một tập các phần tử, mỗi phần tử có thể thể xuất hiện nhiều lần.
- Một **n-bit multiset** là một multiset bao gồm các phần tử thuộc tập $\{0,\ 1\}^n$.
- Một n-bit multiset với $\boldsymbol{k}.\ \boldsymbol{2^n}$ entries được gọi **active** hoặc **saturated** nếu bất kỳ giá trị nào trong $\{\boldsymbol{0},\ \boldsymbol{1}\}^{\boldsymbol{n}}$ được tìm thấy đúng $\boldsymbol{k}$ lần.
- Một multiset gọi là **passive** nếu nó chỉ chứa một giá trị bất biến.
- Một multiset gọi là **garbled** nếu nó không thuộc cả hai trường hợp trên.

3. Biện pháp phòng chống:

- Một số đặc trưng cụ thể của một số thành phần của mật mã, cụ thể là S-box, không ảnh hưởng đến độ hiệu quả của multiset attack.
- Vì multiset và integral attack nhắm đến tầng khuếch tán (diffusion layer) nên điều cần làm là tạo ra những tầng đủ mạnh để chống lại cách tấn công.

C. Ứng dụng:[3]

- Gilbert- Minier đã khai thác những sự đụng độ giữa các hàm thành phần trong bộ mã Rijndael để tấn công, được gọi là "collision attack".
- SASAS cryptanalysis của Biryukov và Shamir kết hợp nguyên tắc của multiset với một số đại số tuyến tính để tấn công hệ mã được xây dựng với cấu trúc S - A - S - A - S, với S là denoting một tầng trong S-boxes nghịch đảo song song và A là phép thay thế ánh xạ khả nghịch qua $GF(2^x)$.

III. Ví dụ Square attack:[1]

Các phép biến đổi được sử dụng trong cách tấn công được trình bày sau có ý nghĩa giống như các phép biến đổi được trình bày trong tài liệu [1].
Tập Λ là một tập hợp gồm 256 trạng thái, tất cả đều khác nhau ở một số byte trạng thái (active) và các byte trạng thái khác đều bằng nhau(passive). Đặt λ là tập chỉ số của các byte active. Chúng ta có

$$\forall x, \mathrm{y} \in A \begin{cases} x_i, j \neq y_i, j \; với \; (i,j) \in \lambda \\ x_i, j = y_i, j \; với \; (i,j) \notin \lambda \end{cases}$$

Áp dụng các phép biến đổi γ và $\sigma[k^t]$ trên các phần tử của Λ cho kết quả là một tập Λ (thường khác) có cùng tập chỉ số λ với tập Λ trước. Áp dụng π cho các byte active sinh ra kết quả là một tập hợp Λ. Áp dụng θ cho tập hợp Λ không nhất thiết cho kết quả là một tập hợp Λ. Tuy nhiên, vì mỗi byte đầu ra của γ là sự kết hợp tuyến tính (với hệ số nghịch đảo) của bốn byte đầu vào trong cùng một hàng, nên một hàng đầu vào có một byte active sẽ cho kết quả là 1 hàng gồm các bytes active.

Hãy xem xét một tập hợp Λ trong đó chỉ có một byte đang hoạt động. Bây giờ chúng ta sẽ theo dõi sự phát triển của vị trí của các byte hoạt động qua 3 vòng. Vòng đầu tiên không chứa θ, do đó vẫn chỉ có một byte hoạt động ở đầu vòng thứ hai. θ của vòng thứ 2 chuyển đổi hàng này thành một hàng byte hoạt động hoàn chỉnh, hàng này sau đó được chuyển đổi bởi π thành một hàng hoàn chỉnh cột. θ của vòng thứ 3 chuyển đổi giá trị này thành tập hợp Λ chỉ với các byte hoạt động.

Điều này vẫn xảy ra ở đầu vào vòng 4. Vì các byte của đầu ra của vòng thứ 3 (ký hiệu là a) nằm trong phạm vi trên tất cả các giá trị có thể và do đó cân bằng trên tập Λ¸, ta có

$$\bigoplus_{b=\theta(a),a\in\Lambda} b_{i,j} = \bigoplus_{a\in\Lambda} \bigoplus_{k} c_{j-k} a_{i,k} = \bigoplus_{l} c_l \bigoplus_{a\in\Lambda} a_{i,l+j} = \bigoplus_{l} c_l 0 = 0.$$

Do đó, các byte của đầu ra θ của vòng thứ tư được cân bằng. Cái này sự cân

$$a_{i,j} = S_\gamma[b_{j,i}] \oplus k^4_{i,j}.$$

bằng nói chung bị phá hủy bởi ứng dụng tiếp theo của γ. Một byte đầu ra của vòng thứ 4 (ký hiệu là a ở đây) có thể được biểu thị như một chức năng của trạng thái trung gian b ở trên

Bằng cách giả sử một giá trị cho $k_{i,j}^4$, giá trị của $b_{i,j}$cho tất cả các phần tử của tập hợp ∧ có thể được tính toán từ các bản mã. Nếu các giá trị của byte này không cân bằng trên ∧, giá trị giả định cho byte chính là sai. Cái này dự kiến sẽ loại bỏ tất cả trừ khoảng 1 giá trị chính. Điều này có thể là lặp lại cho các byte khác của k^4.

Chúng ta đã thực hiện cuộc tấn công và nhận thấy rằng hai tập hợp ∧ gồm 256 bản được chọn, mỗi bộ là đủ để xác định duy nhất khóa mật mã với xác suất thành công vượt trội.

Chúng ta đã tổng hợp thành bảng sau

Attack	#Plaintexts	Time	Memory
4-round	2^9	2^9	small
5-round type 1	2^{11}	2^{40}	small
5-round type 2	2^{32}	2^{40}	2^{32}
6-round	2^{32}	2^{72}	2^{32}

IV. Thực hành:

1. Đánh giá thuật toán:

- Ưu điểm: đảm bảo tạo được khóa có độ lớn bất kỳ
- Khuyết điểm: Do số được lưu dưới dạng chuỗi và bản thân số cũng phải rất lớn nên thời gian xử lý các số đều có thể mất rất nhiều thời gian.

2. Hướng dẫn chạy chương trình:

- Chương trình sẽ bắt đầu bằng việc hiện ra menu để người dùng lựa chọn thao tác mong muốn gồm: nhập khóa, sinh khóa tự động, mã hóa, giải mã và kết thúc chương trình.
- Nhập khóa sẽ yêu cầu người dùng nhập 2 số nguyên tố p và q để tính giá trị n và phiN. Nếu người dùng nhập sai kiểu dữ liệu hoặc nhập hợp số hoặc nhập 2 số nguyên tố giống nhau, chương trình sẽ thông báo lỗi nhập của người dùng và yêu cầu người dùng nhập lại. Sau đó, chương trình yêu cầu người dùng nhập số mũ công khai e, nếu người dùng nhập sai kiểu dữ liệu hoặc nhập số mũ không nguyên tố cùng nhau với phiN thì chương trình sẽ thông báo lỗi nhập của người dùng và yêu cầu người dùng nhập lại. Sau cùng, chương trình sẽ tính giá trị d là nghịch đảo của e.
- Sinh khóa sẽ yêu cầu người dùng nhập giá trị tối thiểu cho 2 số nguyên tố và độ chênh lệch giữa 2 số nguyên tố đó, chương trình sẽ sinh 2 số nguyên tố nhỏ nhất thỏa điều kiện và tính các giá trị n, phiN, e và d.
- thao tác mã hóa và giải mã sử dụng chung một hàm crypting. Hàm này lấy tham số là chuỗi "encrypt" hoặc "decrypt", chương trình sẽ yêu cầu người dùng nhập tên file cần mã hóa/giải mã và xác định giá trị số nguyên của toàn bộ nội dung file. Nếu giá trị này lớn hơn N thì file không được giải mã/mã

hóa. Ngược lại, chương trình sẽ tiếp tục thực hiện việc mã hóa/giải mã. Tùy vào tham số chuỗi nhập vào hàm mà chương trình sẽ chọn số mũ e hoặc d để mã hóa/giải mã nội dung file nhập. Chương trình sử dụng hàm powerMod để tính giá trị của đã qua mã hóa/giải mã và lưu giá trị vừa tính được dưới dạng chuỗi ký tự vào một file mới có tên là tên của file nhập dữ liệu cộng thêm chuỗi tham số nhập vào hàm crypting. Nếu file xuất kết quả không mở được thì chương trình sẽ báo lỗi và kết thúc quá trình chạy hàm

3. Các phương pháp tấn công:

- Hủy hoại gói tin bằng cách thêm, xóa hoặc thay đổi một phần hoặc toàn bộ nội dung trong file đã mã hóa khiến cho nó mất đi nội dung ban đầu hoặc không thể giải mã được.
- Giả danh cả 2 bên gửi và nhận gói tin để kiểm soát nội dung trao đổi giữa 2 bên.

4. Biện pháp phát hiện, phòng chống:

- Nếu gói tin bị thay đổi dẫn đến giá trị số nguyên của toàn bộ nội dung sẽ tăng lên lớn hơn N thì chương trình sẽ thông báo lỗi và không mã hóa thông điệp.
- Nếu gói tin bị hủy hoại thì kết quả giải mã sẽ trở nên vô nghĩa.
- Trường hợp gói tin bị thay đổi nội dung thành nội dung theo ý muốn của kẻ tấn công:
  + Bên gửi và bên nhận có thể thống nhất với nhau sử dụng thêm một cặp giá khác để ký tên (lũy thừa) vào thông điệp đã mã hóa trước đó trước khi gửi đi.
  + Cặp số mũ này là 2 số s và s’ với s’ là nghịch đảo của s trong modulo phiN, s sẽ được dùng để ký tên vào thông điệp đã mã hóa, s phải khác với d là nghịch đảo của e.
  + Bên gửi sẽ thực tính thông điệp mã hóa c là lũy thừa của thông điệp gốc trong modulo N, S là lũy thừa của c trong modulo N và s’ là s nghịch đảo của phiN đã được thống nhất với nhau. Sau đó 3 giá trị này được gửi đến bên nhận.
  + Bên nhận khi nhận được gói tin sẽ thực hiện so sánh c và S lũy thừa s’ trong modulo N. Nếu 2 giá trị bằng nhau thì ta khẳng định thông điệp là đúng do bên gửi mã hóa, ngược lại thì thông báo thông điệp không do bên gửi mã hóa.

V. Tài liệu tham khảo:

[1]: Daemaen, J, L.R. Knudsen, and V. Rijmen (1997). “The block cipher Squar.”

[2]: Multiset Attack | SpringerLink

[3]: Francois-Xavier Standaert, Gilles Piret, Jean-Jacques Quisquater(2003) . “Cryptanalysis of Block Ciphers: A Survey”

[4]:https://qlkh.humg.edu.vn/KhoaHocKhac/Download/2022?FileName=TRUNG_CHU_KY_SO.docx