12라운드 DES 차분 공격 보고서

1. 이론적 배경

공격은 Differential Cryptanalysis of the Full 16-round DES(Eli Biham, Adi Shamir, 1993년) 논문을 바탕으로 구현되었다. 논문의 공격법은 2라운드의 차분 반복 구조를 이용하고, 16라운드의 경우 $O(2^{37})$ 이라는 시간 내에 키를 찾을 수 있다(엄밀히 말해서는 data collection에 $O(2^{57})$의 시간이 소요되고 Analysis에 $O(2^{37})$ 시간이 소요된다.). 이 보고서 내에서 시도한 공격은 12라운드 DES 공격이다. 4라운드가 줄어들게 되면 대략 $234^2 \approx 2^{15.7}$ 배 빠르게 키를 구할 수 있게 되고, 공격의 큰 흐름은 16라운드일 때와 동일하지만 유일하게 달라지는 부분이 논문 상에서 1 Round와 16 Round의 common bits를 이용해 left key를 복원하는 부분이다. 1 Round와 12 Round에서는 common bits가 달라지기 때문에 논문과 같이 Round 16의 S4부터 시작하는 것이 아니라 common bits가 많은 곳을 따로 찾아 거기서부터 left key를 복원해야 할 것이다. common bits table은 아래와 같다.

K1/K12	S1	S2	S3	S4	X	S5	S6	S7	S8	X
S1			12,25	6	15,18,2					
S2	4,7,22		1,16		11					
S3		24,27	5	20,13,9						
S4	17,	8,21,14,3		28						
X	10,26		19	23						
S5						48	38	42,32	53	56
S6						52,34	41	46,49		31
S7						45,40	54		50,29,35	
S8						30	51,33	37	43	47
X							44	55	39	26

이 표에 따라 left key는 Round1 S3, Round12 S4, Round12 S2, Round1 S2, Round12 S1, Round12 S3, Round1 S1 순서로 복원을 해나갔다. Round1 S4, Round12 S2의 common bits가 제일 많지만 Round1 S4의 입력 차분은 0이므로 아무런 의미가 없어 Round1 S4나 Round12 S2에서 시작하는 대신 Round1 S3에서 시작했다.

이외에는 논문의 흐름대로 따라갔으나 quartet method는 적용하지 못했다.

2. 구현 상의 특이사항 및 논문의 개선점

bit permutation이 많고 확률이 낮아 디버깅이 쉽지 않은 프로그램의 특성상 구현을 할 때 12라운드 차분 경로를 따라가는 평문, 키 쌍을 우선 구해놓고 구현을 하는게 나을 것이라고 판단, 우선 12라운드 차분 경로를 따라가는 평문, 키 쌍을 구하는 프로그램을 만들었다. (Diff_Find12RDiffPath) 이 프로그램을 통해 Key = A2 16 40 C2 4E 18 54 06, P1 = 2C 2F 45 E3 26 BC F9 A1, P2 = 2C 2E 84 E5 3F DC F9 A1 이라는 12라운드 차분 경로를 따라가는 평문, 키 쌍을 구했다. 10라운드 차분 경로는 대략 2-3분에 1번씩 찾아졌고 12라운드 차분 경로는 대략 10시간에 1번씩 찾아졌다.

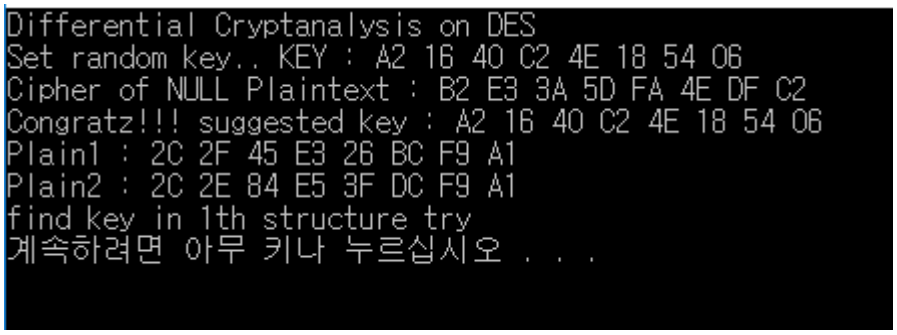
또한 논문 상에서는 임의로 택한 P에서 2^{24}개 plaintext를 가지는 structure를 1개 만들었을 때 올바른 S-box 입력 / 출력 차분을 가진 pair를 1.19개 정도 찾을 수 있을 것이라고 주장했지만 실제로 구현을 해본 결과 대략 100개의 structure에서 4개의 pair가 등장하는 빈도로 pair를 찾을 수 있었다. 논문의 계산법은 마지막 라운드의 input 차분 32비트 중에서 20비트가 반드시 0으로 고정됨을 간과하고 계산하여 pair의 등장 빈도를 실제보다 높게 계산했다.

key suggestion 또한 각 pair에서 left key는 1개 정도, right key는 16개 정도가 candidate key가 되어 결과적으로 각 pair마다 대략 16개씩 균등하게 제안될 것이라고 주장한 논문과 달리 실제로 구현을 해봤을 때에는 대부분의 pair에서 key가 제안되지 않는 반면 일부 pair에서는 key가 수천, 수만개씩 제안되는 현상이 벌어졌다. 이 부분에 대해서는 필터링 과정이 논문에서 가정한 것과 같이 서로 독립이 아니기 때문에 특정 경우에는 필터링을 과도하게 많이 통과한 것이 아닌가 하는 추론을 해보았으나 엄밀하게 증명해내지는 못했다.

3. 프로그램 실행 결과

프로그램의 코드는 https://github.com/blisstoner/DES-12Round-Differential-Attack 에서 확인할 수 있다.

우선 임의로 structure, key를 택하지 않고 아래와 같이 올바른 경로를 따라가는 Plain 쌍이 포함된 structure를 넣고 key를 복원하게 하는 경우 right key가 찾아지는 것을 확인할 수 있다.



그러나 임의의 12라운드 공격의 경우 경로를 올바르게 따라갈 확률은 $234^{-4} \approx 2^{-31.48}$ 이고 structure 1개당 대략 0.04개의 pair가 찾아지므로 structure가 right pair를 포함하고 있을 확률은 $0.04 * 2^{12} * 2^{-31.48} \approx 2^{-24.12}$ 가 되어 대략 1600만개의 structure에 대해 확인해야 right pair를 얻을 수 있는데 실제로 실행을 해보면 1분에 대략 2400개의 structure밖에 확인을 못하기 때문에 약 5일 가까이의 시간이 필요하게 된다. 그로 인해 실제 키를 랜덤으로 정한 후 이를 복원해내는 것은 확인하지 못했다. structure가 포함하는 pair의 개수가 적고 structure 내에서 key를 복구할 때 필요로 하는 연산이 생각보다 많다는 점 때문에 시간이 예상보다 굉장히 오래 걸리게 되었다. 만약 10라운드에 대해 공격을 진행했다면 대략 30분 내외로 키를 구할 수 있었을 것이다.