

Re: Resolved cases in the known IV attack on WEP

Source: <http://www.derkeiler.com/Newsgroups/sci.crypt/2003-02/1597.html>

From: Scott Fluhrer (sfluhrer@ix.netcom.com)

Date: 02/02/03

From: "Scott Fluhrer" <sfluhrer@ix.netcom.com>

Date: Sat, 1 Feb 2003 21:22:40 -0800

Shill <shill@free.fr> wrote in message

news:b1bevh\$28ef\$1@norfair.nerim.net...

> *I have one more question. At the end of section 7.1, you state*
> *"Hence, by examining approximately 60 IVs with the above*
> *configuration, the attacker can rederive $K[A]$ with a probability*
> *of success greater than 0.5."*

>

> *How did you obtain the figure of 60 IVs? Was it through calculus*
> *or simulation? I have tried calculus but it gets very messy:*
Simulation.

>

> *Assume we collect X IVs. Assume the secret byte we are trying to*
> *guess is 0 (without any loss of generality). Let X_n be the number*
> *of times the attacker's algorithm guesses byte n .*

>

> *evidently, sum (for $i = 0$ to 255) $X_i = X$*

>

> *After we've collected X IVs, we choose the byte that occurs most*
> *frequently as our guess.*

>

> *The probability that, after X runs, we will have more occurrences*
> *of byte 0 than any other byte is:*

>

> *Sum for $N = 2$ to R*

> *$P(X_0 = N \text{ and } X_i < N \text{ for every } 1 \leq i \leq 255)$*

>

> *It is fairly easy to compute this probability when $X_0 > R/2$*
> *(absolute majority) but it gets hard, IMO, when we only have a*
> *relative majority...*

>

> *So I turned to simulation. I wrote a function that returns 0 with*
> *probability 0.05 and k ($1 \leq k \leq 255$) with probability $0.95/255$.*

>

> *I generated X numbers and looked for the most frequent occurrence*
> *(I repeated this experiment 2^{25} times).*

>

> When $X=31$, the probability to correctly guess 0 is 50.5%. When
> $X=64$, the probability to correctly guess 0 is 74.6%. This is why I
> am puzzled by your figure of 60...

We did exactly what you did, with two differences:

– Our function returned 0 with probability $0.05 + 0.95 * 1/256$. That is, we assumed that if the resolved state wasn't preserved, the first byte was effectively random, and could be "the right value for the wrong reason". Of course, this doesn't explain the discrepancy, as the difference that generates is in the wrong direction.

In retrospect, our assumption doesn't look quite right — if there was exactly one swap among the three permutation elements, and the element that was swapped wasn't $S[1]$, it wouldn't output the correct value. Hence, it is reasonable to suspect that, when the resolved condition isn't ultimately preserved, the probability of the first byte output is what it would be if resolved condition were preserved is less than $1/256$ (but more than the 0 that your simulation assumed).

– If you generate X IV's, and you see two (or more) values the same number of times, what do you do? There are three obvious strategies:

A) Since we don't have a clear winner, say we don't know

B) If the correct value occurs as many times as any other value, say we succeeded

C) Pick randomly among the highest occurring values, and if that one is the correct one, say we succeeded.

I have just simulated this, and with strategy (B) (and your probability distribution), that at $X=31$, the probability of success is 0.5061, and at $X=64$, probability=0.7468 — I assume that this is the strategy you used. We, in contrast, used strategy (A) (which, with our probability distribution, and at $X=54$, gives probability 0.5019 — I don't quite remember why we rounded it up to 60 — possibly out of the spirit of conservatism).

In retrospect, strategy (C) looks even more realistic. This gives (again, using your probability distribution, which are a bit pessimistic), probability 0.5020 at $X=52$.

However, going even further, this entire analysis is a bit crude. If we are willing to go to a bit of work on the post-analysis, we realize that we don't need for this to come up with the unique value of the key byte — it may be preferable to come up with (say) 4 values, one of which is right with probability > 0.9 . Then (to take it a bit simplistically here), if we have 13 unknown key bytes, and we have 4 possibilities for each, then with 4^{13} work, we can check all those possibilities. I say this is a bit simplistic, because this attack uses previous key bytes to obtain the next one (which is why we shied away from this approach in our work). However, working further with the attack, I have noticed that, as long as you get the sum of the previous key bytes correct (that is, you make a usually correct prediction on the intermediate j value), you often get the correct value for the next

sci.crypt: Re: Resolved cases in the known IV attack on WEP

key byte, even if you are guessed wrong on the exact values of the previous key bytes. This certainly looks like a fruitful area for more study.

--

poncho