ششمین کنفرانس بین المللی انجمن رمز ایران
۱۵ و ۱۶ مهر ماه ۱۳۸۸، دانشگاه اصفهان
انجمن رمز ایران
دانشگاه اصفهان

حمله بازیابی کلید روی الگوریتم دنباله‌ای Nofish

محمدعلی ارومیه چی‌ها، الهام شکور

صنایع الکترونیک زعیم
orumiehchi@zaeim.co.ir
shakour@zaeim.co.ir

چکیده

در این مقاله امنیت رمز دنباله ای Nofish با طول کلید مخفی ۵۱۲ بیت بررسی می‌گردد. ثابت می‌شود که الگوریتم در مقابل حمله تمایز کاملا آسیب پذیر است و همچنین حمله بازیابی کلید با پیچیدگی حداکثر $2^{28.2}$ پیشنهاد می‌شود که قادر است تمام بیت‌های کلید مخفی مولد را بطور کامل و با احتمال یک محاسبه نماید. بخشی از نتایج پیاده سازی حمله نیز در این مقاله بیان شده است.

واژه‌های کلیدی

رمز دنباله ای، تحلیل رمز، رمز دنباله ای Nofish، حمله بازیابی کلید، حمله تمایز.

۱- مقدمه

الگوریتم‌های رمز دنباله ای، نقش اساسی در رمزنگاری و محرمانگی داده، بخصوص برای کاربردهای با سرعت بالا ایفا می‌کنند. در رمزهای دنباله ای، یک رمز دنباله ای روی هر بیت متن اصلی به عنوان یک تابع متغیر با زمان عمل می‌کند. در این حالت با توجه به متغیر با زمان بودن تابع رمز کننده، مشکل یکسان بودن قطعات رمز شده‌ای که متن‌های اصلی یکسانی دارند، وجود ندارد. همچنین بر خلاف رمزهای قالبی، در مورد رمزهای دنباله ای می‌توان انتشار خطا را کاهش داده و حتی آن را در حد همان یک بیتی که دچار خطا شده است، نگه داشت. این مزیت در سامانه‌های مخابراتی از اهمیت بسزایی برخوردار است. علاوه بر این رمزهای دنباله‌ای در مقایسه با رمزهای قالبی، از نظر پیاده سازی، ساده تر، با سخت افزاری کم حجم تر و ارزان تر هستند. همچنین سرعت رمزنگاری و رمز گشایی آنها هم بطور معمول سریعتر از رمزهای قالبی است. با توجه به این مزیت‌ها و همچنین بالا بردن جنبه‌های نظری و کاربردی بیشتر در طراحی و تحلیل این سامانه‌ها، پروژه eSTREAM [۴] در سال ۲۰۰۴ اقدام به برگزاری فراخوانی جهت طراحی سامانه‌های رمز دنباله ای امن کرد. در همین راستا الگوریتم‌هایی حتی پس از اتمام فراخوان این پروژه بطور عمومی پیشنهاد شدند. از جمله این الگوریتم‌ها، الگوریتم رمز دنباله ای Nofish است. در این مقاله اثبات می‌کنیم که این الگوریتم در مقابل حمله تمایز بشدت ضعیف است و با توجه به روابط همبستگی محاسبه شده در این مقاله، خروجی الگوریتم از یک دنباله کاملا تصادفی قابل تمییز است. همچنین، با استفاده از حمله تمایز پیشنهادی، حمله بازیابی کلید در زمان واقعی را ارائه خواهیم داد.

این مقاله بصورت زیر سازماندهی شده است. بخش دوم شامل توصیف مختصر الگوریتم رمز دنباله ای Nofish است. در بخش سوم چگونگی حمله تمایز و حمله بازیابی کلید روی الگوریتم را بحث می‌کنیم و در نهایت به نتیجه گیری میپردازیم.

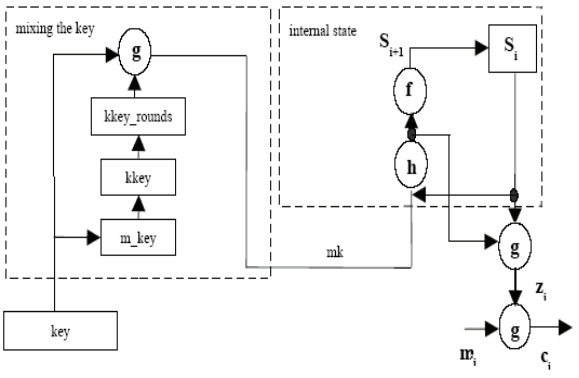
۲-۲ تشریح الگوریتم

این الگوریتم یک الگوریتم دنباله ای همزمان با طول کلید ۵۱۲ بیت است. ایده طراحی الگوریتم Nofish بر اساس الگوریتم دنباله ای Henkos است[۲و۳].

درحقیقت طبق ادعای طراح سعی شده است که بر اساس نقطه ضعفهای الگوریتم Henkos توابع بکار رفته در بخشهای مختلف الگوریتم Nofish بهبود یابد[۳].

در الگوریتم Nofish از کلید مخفی به طول ۶۴ بایت (۵۱۲ بیت) استفاده شده و بردار اولیه در آن بکار نرفته است. در ادامه این بخش به شرح مختصر الگوریتم می‌پردازیم. لازم بذکر است که در این مقاله از توصیف کامل الگوریتم به جهت رعایت اختصار

خودداری شده است و تنها بخش‌های اصلی و مورد نیاز برای فهم حمله پیشنهاد شده بین شده است. خواننده علاقمند می‌تواند به [۱] مراجعه کند. این الگوریتم شامل سه بخش مخلوط کننده کلید، بارگذاری کلید و مولد دنباله کلید است. و از سه آرایه Key، m_key، input با طول ۶۴ بایت برای تولید دنباله کلید اجرایی بهره می‌برد. در واقع بردار Key حاوی ۲۵۶ بیت از کلید اصلی مولد، بردار m_key به عنوان یک جایگشت تصادفی و بردار input به عنوان حالت داخلی مولد شناخته می‌شود. در ذیل هر سه بخش اصلی مولد تشریح می‌گردد. (برای سادگی فهم الگوریتم و همچنین درک سریع تر از حمله پیشنهادی، هر بخش از الگوریتم و حمله به الگوریتم بصورت شبه کد نیز معرفی می‌گردد.)



شکل ۱: نمای کلی الگوریتم دنباله ای Nofish

۲-۱ مخلوط کننده کلید

در بخش مخلوط کننده کلید، ابتدا متغیری به نام kkey با مقدار ۵۱۲ مقداردهی می‌شود و سپس بردار m_key که شامل ۶۴ بایت است بصورت زیر مقداردهی می‌شود:

$$For\ (i=0,\ \dots\ ,64) \quad M\_k[i]= i \qquad (۱)$$

در مرحله بعد برای مخلوط کردن مقادیر کلید از چهار دسته عدد S_1= {7,26} و S_2 ={5,22} و S_3 ={3,16,29} و S_4 ={2,13,24} بصورت زیر استفاده می‌شود.

```
kkey=512;
for(i=0, ... ,31)                                        (۲)
{
        if(Reminder(i,19)==7) kkey= kkey – (i⊕
        (key[i]+key[i+32]));
        else
        if(Reminder(i,17)==5)
        kkey=Reminder(kkey,(i⊕
        (key[i]+key[i+32])))+512;
                else
         if(Reminder(i,13)==3) kkey= kkey +( i⊕
        (key[i]+key[i+32]));
        else
         if(Reminder(i,11)==2) kkey= kkey ×i;
}
kkey=|kkey|;
```

که ⊕، +، * ، Reminder(A,B) و |A| به ترتیب معرف Xor، جمع ، ضرب ، باقیمانده تقسیم B بر A و قدر مطلق عدد A است. بعد از محاسبه مقدارkkey برای محاسبه بردار جدید m_k بصورت زیر عمل شده است:

```
for(j=0, ... , kkey)                                     (۳)
{
        sump=0;
        for(i=0, ... ,63)  sump=sump+m_key[i];
        for(i=0, ... ,63) m_key[i]=
        Reminder(sump,(67-i));
}
```

و کلید برای مقداردهی m_keyو در آخر این مرحله از مقادیر استفاده شده است. inputبایتهای

$$for(i=0,\ \dots\ ,63) \quad input[i]=m\_key[i] \oplus key[i]; \qquad (۴)$$

۲-۲- بارگذاری کلید

بعنوان تابع SWدر این بخش از الگوریتم از دو تابع اصلی بعنوان تابع مبتنی بر جمع استفاده شده است. با ADجابجایی و) با یکدیگر inputاستفاده از تابع اول بایتهای حالت داخلی (بردار جابجا می‌شوند و با استفاده از تابع دوم مقدار هر بایت با جمع خود می‌شود. برای بایت آخر نیز، مقدار آن بایت و بایت بعدی بهنگام می‌شود. به این ترتیب، این مرحله به این بایت با بایت اول جمع بار تکرار می‌شود. حاصل دور $T= (64 + kkey) \bmod 64$اندازه آخر این مرحله بعنوان حالت اولیه برای بخش مولد دنباله کلید می‌رود.بکار

```
for (j=0, ... ,T)                                        (۵)
{
        for(i=0, ... ,63)
        {
                x=input[m_key[i]];
                input[m_key[i]]=input[i];
                input[i]=x;
        }
        primul=input[0];
        for(i=0, ... ,62)
        input[i]=( Reminder (input[i],input[i+1]))
        ⊕ key[i];                                        (۶)
        input[63]=( Reminder (input[63],primul))
        ⊕ key[keylen-1];
}
```

۳-۲ مولد دنباله کلید

برای بدست آوردن دنباله کلید طبق شکل (۱)، از خروجی دور آخر مرحله بارگذاری استفاده می‌شود (S_i) و به این ترتیب بایتهای دنباله کلید اجرایی (output) بصورت زیر ساخته می‌شود:

```
for(i=0, ... , 63)                                        (۷)
{
        x=input[m_key[i]];
        input[m_key[i]]=input[i];
         input[i]=x;
}
primul=input[0];
for(i=0, ... , 62)
            input[i]=( Reminder
            (input[i],input[i+1])) ⊕ key[i];
                                                          (۸)
input[63]=( Reminder (input[63],primul) ⊕ key[63];
for(i=0, ... , 63)
            output[i] =(input[i] ⊕ (Reminder
            (input[i],input[i+1])) ⊕ key[i]);             (۹)
            output[63]= (input[63] ⊕ (Reminder
            (input[63],input[0])) ⊕ key[63]);
```

تحلیلهای انجام شده توسط طراح، حاکی از امنیت بالای الگوریتم است که [۱] بطور کامل به آن پرداخته است.

۳- تحلیل الگوریتم دنباله‌ای Nofish

در این مقاله دو حمله بر روی این الگوریتم پیشنهاد می‌کنیم و ثابت خواهیم کرد که این الگوریتم بصورت زمان واقعی بازیابی کلید خواهد شد. در بخش اول حمله تمایزی ارائه می‌شود که قادر است دنباله خروجی الگوریتم را با احتمال یک از دنباله کاملا تصادفی تشخیص دهد و در بخش دوم، با استفاده از ایده حمله اول، حمله بازیابی کلید روی الگوریتم انجام می‌شود. این حملات نشان می‌دهند که پیچیده نمودن الگوریتم الزاما به بالا بردن امنیت آن کمک نمی کند. اما قبل از تشریح حملات، به بررسی نقاط ضعف استفاده شده در حمله می‌پردازیم. با استفاده از این نقاط ضعف، قادر خواهیم بود براحتی حملات تمایز و کشف کلید را روی الگوریتم Nofish انجام دهیم.

۳-۱ نقاط ضعف الگوریتم Nofish

۱- وابسته بودن جایگشت m_k تنها به مقدار kkey

طبق الگوریتم طراحی شده، بایتهای m_k[i] ابتدا با مقادیر i=0 , 63 , ... پر می‌گردند و سپس به تعداد kkey مرتبه محتویات m_k بطور جایگشتی بهنگام می‌گردد. دراینصورت با دانستن مقدار kkey می‌توان بطور دقیق مقادیر m_k را محاسبه کرد.

۲- محدود بودن مقدار kkey

با توجه به رابطه (۲) مقدار درنظرگرفته شده برای kkey یک عدد ۳۲ بیتی است که این مقدار در جریان الگوریتم بیشتر از ۵۱۲ حالت نخواهد بود و بنابراین حدس kkey با پیچیدگی 2^9 انجام می‌شود. مقادیر دقیق kkey در ضمیمه مقاله آمده است.

۳- وجود رابطه خطی بیتهای کم ارزش مقادیر input[i], input[i+1], k[i]:

با توجه به روابط (۸و۹) بیت کم ارزش دنباله خروجی الگوریتم ترکیب خطی از بیتهای حالت داخلی الگوریتم و k[i] است. این رابطه خطی برای همه بایتهای خروجی با احتمال یک برقرار است. با استفاده از همین روابط خطی می‌توان هر دو حمله را طرح ریزی کرد.

۴- ضعف در بهنگام سازی آرایه input:

این آرایه در واقع نقش حالت داخلی الگوریتم را دارد و حمله کننده می‌تواند با محاسبه این مقدار به کلید دست یابد. همانطور که از رابطه (۸) مشخص است بیت صفر input[i] تنها تابعی از بیت صفر input[i+1] و k[i] است و همچنین بیت یک input[i] تابعی از بیتهای صفر و یک input[i] و input[i+1] و بیت یک کلید است. بعبارت دیگر در بهنگام سازی بیت jام input[i] تنها بیتهای کوچکتر مساوی j از بیتهای کلید، input[i] و input[i+1] -در زمان قبل- نقش دارند و بقیه بیتها در محاسبه این مقادیر کاملا بی اثر هستند. با استفاده از همین ضعف می‌توان بیتهای با ارزش کمتر را بدون اطلاع از بیتهای با ارزش بیشتر محاسبه نمود.

حال با استفاده از نقاط ضعف اشاره شده دو حمله تمایز و بازیابی کلید مخفی بر روی این الگوریتم را پیشنهاد می‌نماییم.

۳-۲ حمله تمایز روی الگوریتم دنباله‌ای Nofish

همانطور که در بخش ۲ اشاره شد، هسته اصلی این الگوریتم آرایه input با طول ۶۴ است که در هر مرحله اجرای الگوریتم با اعمال دو تابع SW, AD محتویات آن بهنگام می‌گردد. واضح است که در بهنگام سازی محتویات حالت داخلی آرایه input، حالت داخلی در زمان قبل و محتویات کلید مشارکت می‌کنند. اما با توجه به نقطه ضعف ۳، بیتهای کم ارزش این آرایه ترکیب خطی از کلید و محتویات کم ارزش در زمان قبل است. این روند برای بیتهای بعدی نیز بصورت تقریب خطی از بیتهای کلید و محتویات متناظر در زمان قبل قابل محاسبه است. در اینصورت با نوشتن روابط خطی روی بیت کم ارزش بایتهای خروجی الگوریتم در یک مرحله از تولید دنباله کلید، جمع این روابط همیشه برابر صفر خواهد بود و به این ترتیب یک رابطه متمایز کننده از خروجی الگوریتم Nofish بدست می‌آید. پیچیدگی داده جهت اعمال حمله تمایز بر این الگوریتم تنها ۶۴ بایت خروجی الگوریتم است.

در بخش بعد نشان خواهیم داد که با استفاده از این حمله تمایز، می‌توانیم حالت داخلی و کلید مخفی الگوریتم را بدست آوریم.

۳-۳ حمله بازیابی کلید روی الگوریتم دنباله‌ای Nofish

با توجه به نقاط ضعف ۱و۲ می‌توان الگوریتم حمله محاسبه کلید مخفی الگوریتم را طراحی کرد. با توجه به اینکه مقدار بردار m_k در ابتدا بطور معلوم مقداردهی می‌شود و تعداد جایگشتها برای تغییر حالت داخلی آن تنها وابسته به مقدار kkey می‌باشد، بنابراین می‌توان نتیجه گرفت که تعداد جایگشتهای متفاوت تولید شده، حداکثر برابر با مقدار حداکثر kkey و حداقل برابر مقدار حداقل kkey است. از طرف دیگر مقدار kkey فقط می‌تواند ۵۱۲ حالت داشته باشد. بنابراین تعداد کل جایگشتهای تولید شده توسط الگوریتم ۵۱۲ خواهد بود. پس kkey براحتی قابل حدس است و به ازای هر حدس kkey یک جایگشت معلوم و قطعی از آرایه m_key محاسبه می‌شود. مقدار حالت داخلی الگوریتم (آرایه input) که ترکیب کاملا خطی از کلید اصلی مولد و محتویات بردار m_key است، بصورت روابط خطی و تنها با متغیرهای کلید مخفی قابل محاسبه خواهد بود.

اما می‌دانیم که الگوریتم پس از انجام مرحله مخلوط کلید (mix-key) وارد مرحله تولید دنباله کلید اجرایی می‌گردد. در این بخش نیز محتویات آرایه input طی گذر از دو تابع SW و AD بصورت خطی و غیرخطی بهنگام می‌شود. حال بر اساس نقطه ضعف ۳ بیت کم ارزش دنباله کلید اجرایی الگوریتم، ترکیب خطی از حالت داخلی و کلید مخفی مولد است اما برای بیتهای بعدی دنباله کلید اجرایی، این ترکیب بصورت غیرخطی انجام شده است. روابط خطی بدست آمده به ازای یک مقدارkkey در ضمیمه آمده است. حال با تشکیل دستگاه معادلات از متغیرهای ورودی (کلید و حالت داخلی) سعی در حل این دستگاه خواهیم داشت.

فرض می‌کنیم ۶۴ بایت متوالی اول از خروجی را در اختیار داشته باشیم (می توان حمله را برای هر ۶۴ بایت متوالی مربوط به یک۱- دور مشخص را در نظر گرفت.) چنانچه مقدار kkey را حدس بزنیم، مقدار آرایه m_key طبق الگوریتم بسادگی بدست می‌آید و همچنین آرایه input بصورت تابع خطی از کلید اصلی مولد قابل محاسبه خواهد بود.

با این توصیف تا لحظه تولید دنباله کلید اجرایی، می‌توان تنها بیت کم ارزش آرایه input را بعنوان تابع خطی از بیتهای کلید درنظر گرفت. نوشتن روابط خطی بدست آمده حاکی از آن است که دستگاه معادلات با ۶۴ معادله (و ۶۴ متغیر) دارای مرتبه کامل است و بسادگی توسط یکی از روشهای حل معادله خطی در میدان GF(2) قابل حل است. پیچیدگی مورد انتظار برای حل این معادلات $۲^{۱۶.۲}=(۶۴)^{۲.۷}$ خواهد بود. تا این مرحله از الگوریتم بیت صفرام بایتهای کلید و بایتهای آرایه input محاسبه می‌شود. بیتهای بعدی خروجی توابع غیرخطی از محتویات حالت داخلی و کلید مخفی هستند که با روش فوق نمی توان بطور مستقیم معادلات آنها را حل کرد. برای حل معادلات بیتهای بعدی باید بصورت بازگشتی عمل کنیم. به این ترتیب که پس از محاسبه بیتهای کم ارزش، معادلات بیتهای مجاور آن از معادلات درجه دوم وبالاتر به معادلات درجه اول تبدیل می‌شوند و با حل دستگاه جدید مقادیر این بیتها نیز محاسبه می‌گردد. با تکرار این مراحل تمامی بیتهای حالت داخلی و بیتهای کلید مخفی محاسبه می‌شوند. بدین صورت بیتهای کلید با داشتن تنها ۶۴ بایت از خروجی محاسبه می‌شود. الگوریتم حمله بازیابی کلید را می‌توان در روابط زیر خلاصه کرد:

۴-۳ الگوریتم حمله

- مقدار kkey را حدس بزن. (۵۱۲ حالت)
- به ازای مقدار kkey حدس زده شده، مقدار بردار m_key را محاسبه کن.
- بازای m_key مقدار بردار input را بهنگام کن.
- مراحل زیر را برای محاسبه input و دنباله کلید اجرایی تکرار کن.
- مراحل زیر را از i=0 تا i=7 تکرار کن.

۱- بازای بیت i ام از هر بایت خروجی، رابطه خطی مبتنی بر کلید را محاسبه کن.

۲- دستگاه معادلات خطی مربوطه را حل کن.

$i=i+1$ -۳

بیتهای کلید، جواب دستگاههای حل شده در بند ۲ است.

با توجه به الگوریتم حمله، می‌توان پیچیدگی محاسباتی جهت بازیابی کلید را برابر $۲^{۲۸.۲}=۲^{۱۶.۲}\times۵۱۲\times۸$ درنظر گرفت که در آن ۵۱۲، پیچیدگی حدس kkey و $۲^{۱۶.۲}=۶۴^{۲.۷}$ پیچیدگی حل دستگاه با ۶۴ معادله و ۶۴ مجهول درنظر گرفته شده است.

۴- نتیجه گیری

در این مقاله ما بر اساس نقاط ضعف موجود در بخش بارگذاری و تابع بهنگام کننده حالت داخلی الگوریتم توانستیم دو حمله تمایز

و بازیابی کلید روی الگوریتم انجام دهیم. به این ترتیب حمله تمایز با داشتن ۶۴ بایت خروجی الگوریتم و حمله بازیابی کلید با پیچیدگی $2^{28.2}$ روی الگوریتم بطور کاملا عملی قابل انجام است.

مراجع

[1] Marius Oliver Gheorghita, Nofish : A new stream cipher, avalalbe at www.eprint.iacr.org/2009.

[2] Prasanth Kumar Thandra and S.A.V. Satya Murty, Weaknesses in HENKOS Stream Cipher, www.eprint.iacr.org/2008.

[3] Marius Oliver Gheorghita, HENKOS Stream Cipher, www.eprint.iacr.org/2008.

[۴] eSTREAM, the ECRYPT Stream Cipher Project. http://www.ecrypt.eu.org/stream/

ضمیمه

با پیاده سازی حمله بازیابی کلید بر الگوریتم Nofish به ازای kkey= 10251 به دستگاه معادلاتی به شکل زیر رسیدیم که این دستگاه با داشتن مقادیر خروجی و با روش حذفی گاوس یا مثلثی کردن ماتریس ضرایب قابل حل است. پس از حل معادلات خطی برای بیت صفر-ام هر بایت از کلید مخفی، می‌توان بیتهای بعدی را نیز طبق الگوریتم حمله محاسبه نمود. در ذیل به عنوان نمونه، دستگاه معادلات خطی مربوط به بیت کم ارزش خروجی را آورده ایم. در دستگاه $z_1, \ldots, z_{64}$ و k_1 ، k_{64}... به ترتیب بیت‌های کم ارزش خروجی و کلید مخفی از هر بایت مجزاهستند.

$1+k_{48}+k_1+k_{63}+k_2=z_1$
$k_{63}+k_2+k_{52}+k_3=z_2$
$k_{52}+k_3+k_{54}+k_4=z_3$
$k_{54}+k_4+k_{25}+k_5=z_4$
$k_{25}+k_5+k_{58}+k_6=z_5$
$k_{58}+k_6+k_{18}+k_7=z_6$
$k_{18}+k_7+k_{21}+k_8=z_7$
$1+k_{21}+k_8+k_{46}+k_9=z_8$
$1+k_{46}+k_9+k_{35}+k_{10}=z_9$
$k_{35}+k_{10}+k_{23}+k_{11}=z_{10}$
$1+k_{23}+k_{11}+k_4+k_{12}=z_{11}$
$k_4+k_{12}+k_{40}+k_{13}=z_{12}$
$k_{40}+k_{13}+1+k_{31}+k_{14}=z_{13}$
$1+k_{31}+k_{14}+k_{10}+k_{15}=z_{14}$
$1+k_{10}+k_{15}+k_{12}+k_{16}=z_{15}$
$k_{12}+k_{16}+k_{45}+k_{17}=z_{16}$
$k_{45}+k_{17}+k_{44}+k_{18}=z_{17}$
$k_{44}+k_{18}+k_{42}+k_{19}=z_{18}$
$k_{42}+k_{19}+k_{38}+k_{20}=z_{19}$
$k_{38}+k_{20}+k_{13}+k_{21}=z_{20}$
$1+k_{13}+k_{21}+k_{20}+k_{22}=z_{21}$
$1+k_{20}+k_{22}+k_7+k_{23}=z_{22}$
$k_7+k_{23}+k_{15}+k_{24}=z_{23}$
$k_{15}+k_{24}+k_1+k_{25}=z_{24}$
$k_1+k_{25}+k_{11}+k_{26}=z_{25}$
$k_{11}+k_{26}+k_6+k_{27}=z_{26}$
$k_6+k_{27}+k_{19}+k_{28}=z_{27}$
$k_{19}+k_{28}+k_5+k_{29}=z_{28}$
$k_5+k_{29}+k_{43}+k_{30}=z_{29}$
$k_{43}+k_{30}+k_{27}+k_{31}=z_{30}$
$k_{27}+k_{31}+k_{16}+k_{32}=z_{31}$
$k_{16}+k_{32}=z_{32}$
$k_{32}+k_{34}=z_{33}$
$k_{32}+k_{34}+k_{28}+k_{35}=z_{34}$
$k_{28}+k_{35}+k_{30}+k_{36}=z_{35}$
$k_{30}+k_{36}+k_{29}+k_{37}=z_{36}$
$k_{29}+k_{37}+k_3+k_{38}=z_{37}$
$1+k_3+k_{38}+k_{26}+k_{39}=z_{38}$
$1+k_{26}+k_{39}+k_{24}+k_{40}=z_{39}$
$k_{24}+k_{40}+k_{17}+k_{41}=z_{40}$
$k_{17}+k_{41}+k_9+k_{42}=z_{41}$
$k_9+k_{42}+k_{22}+k_{43}=z_{42}$
$k_{22}+k_{43}+k_{34}+k_{44}=z_{43}$
$1+k_{34}+k_{44}+k_{14}+k_{45}=z_{44}$
$1+k_{14}+k_{45}+k_{39}+k_{46}=z_{45}$
$1+k_{39}+k_{46}+k_8+k_{47}=z_{46}$
$1+k_8+k_{47}+k_{36}+k_{48}=z_{47}$
$1+k_{36}+k_{48}+k_2+k_{49}=z_{48}$
$1+k_2+k_{49}+k_{47}+k_{50}=z_{49}$
$k_{47}+k_{50}+k_{37}+k_{51}=z_{50}$
$k_{37}+k_{52}=z_{51}$
$k_{51}+k_{52}+k_{41}+k_{53}=z_{52}$
$k_{41}+k_{53}+k_{49}+k_{54}=z_{53}$
$k_{49}+k_{54}+k_{50}+k_{55}=z_{54}$
$k_{50}+k_{55}+k_{53}+k_{56}=z_{55}$
$k_{53}+k_{56}+k_{55}+k_{57}=z_{56}$
$k_{55}+k_{58}=z_{57}$
$k_{57}+k_{58}=z_{58}$
$k_{56}+k_{60}=z_{59}$
$k_{56}+k_{61}=z_{60}$
$k_{60}+k_{61}=z_{61}$
$1+k_{64}+k_{63}=z_{62}$
$1+k_{63}+k_{61}=z_{63}$
$k_{64}+1+k_{61}+k_{48}+k_1=z_{64}$