Statistical Attack on RC4

Distinguishing WPA

Pouyan Sepehrdad, Serge Vaudenay, and Martin Vuagnoux

EPFL
CH-1015 Lausanne, Switzerland
http://lasecwww.epfl.ch

Abstract. In this paper we construct several tools for manipulating pools of biases in the analysis of RC4. Then, we show that optimized strategies can break WEP based on 4000 packets by assuming that the first bytes of plaintext are known for each packet. We describe similar attacks for WPA. Firstly, we describe a distinguisher for WPA of complexity 2^{43} and advantage 0.5 which uses 2^{40} packets. Then, based on several partial temporary key recovery attacks, we recover the full 128-bit temporary key by using 2^{38} packets. It works within a complexity of 2^{96}. So far, this is the best attack against WPA. We believe that our analysis brings further insights on the security of RC4.

1 Introduction

RC4 was designed by Rivest in 1987. It used to be a trade secret until it was anonymously posted in 1994. Nowadays, RC4 is widely used in SSL/TLS and Wi-Fi 802.11 wireless communications. 802.11 [1] used to be protected by WEP (Wired Equivalent Privacy) which is now being replaced by WPA (Wi-Fi Protected Access) due to security weaknesses.

WEP uses RC4 with a pre-shared key. Each packet is encrypted by a XOR to a keystream generated by RC4. The RC4 key is the pre-shared key prepended with a 3-byte nonce IV. The IV is sent in clear for self-synchronization. There have been several attempts to break the full RC4 algorithm but it has only been devastating so far in this scenario. Indeed, the adversary knows that the key is constant except the IV, which is known. An active adversary can even alter the IV. Nowadays, WEP is considered as being terribly weak since passive attacks can recover the full key easily by assuming that the first bytes of every plaintext frames are known. This happens to be the case due to protocol specifications.

In order to fix this problem, the Wi-Fi Alliance has replaced WEP by WPA [1]. Peer authentication is based on IEEE 802.1X which accommodates a simple authentication mode based on a pre-shared key (WPA-PSK). Authentication creates a Temporary Key (TK). The TK then goes through the temporary key integrity protocol (TKIP) to derive per-packet keys (PPK). The idea is that TK is derived into a TTAK key to be used for a number of frames limited to 2^{16}. Each frame applies a simple transformation to TTAK and a counter TSC to derive the RC4 per-packet key PPK. Again, the 3 first bytes of the RC4 key are known (they actually depend on the counter).

In addition to the key derivation, WPA provides a packet integrity protection scheme which prevents from replaying or altering the IV. Thus, only passive key recovery attacks can be considered.

Our contribution. In this paper, we construct tools for manipulating pools of biases. With our theory, we then analyze several statistical strategies for partial key recovery. We apply it to recover the 8 weak bits of the WPA key TK by using 2^{38} to 2^{40} packets. Incidentally, we apply our analysis to WEP and show that the best attacks so far can still be improved. We then transform our partial key recovery attack into a practical distinguisher for WPA. Finally, we build a full session key recovery with complexity 2^{96} and 2^{38} packets.

Related Work. We mention three approaches for the cryptanalysis of RC4: attacks based on the weaknesses of the Key Scheduling Algorithm (KSA) and attacks based on the weaknesses of the Pseudorandom Generator Algorithm (PRGA), and blackbox analysis.

As for the KSA, one of the first weaknesses published on RC4 was discovered by Roos [32] in 1995. This correlation binds the secret key bytes to the initial state S'_0. Roos [32] and Wagner [38] identified classes of weak keys which reveal the secret key if the first key bytes are known. This property has been largely exploited to break WEP (see [5,9,13,18,19,33,34,35,37]). Another class of result concerns the inversion problem of KSA: given the final state of the KSA, the problem is to recover the secret key [4,28].

Regarding PRGA, the analysis has been largely motivated by distinguishing attacks [8,11,22,24] or initial state reconstruction from the keystream bytes [10,17,25,36] with complexity of 2^{241} for the best state recovery attack. Relevant studies of the PRGA reveal biases in the keystream output bytes in [23,29]. Mironov recommends in [26] that the first 512 initial keystream bytes must be discarded to avoid these weaknesses.

Jenkins published in 1996 on his website [14] two biases in the PRGA of RC4. These biases have been generalized by Mantin in his Master Thesis [21]. Paul, Rathi and Maitra [30] discovered in 2008 a biased output index of the first keystream word generated by the PRGA. Another bias on the PRGA has been experimentally discovered by Maitra and Paul [20]. Finally, Sepehrdad, Vaudenay and Vuagnoux [33] discovered 48 new correlations in PRGA and 9 new correlations between the key bits and the key stream. This led to the fastest attack on WEP at the moment.

In practice, key recovery attacks on RC4 must bind KSA and PRGA weaknesses to correlate secret key words to keystream words. Some biases on the PRGA [16,30,20] have been successfully bound to the Roos correlation [32] to provide known plaintext attacks. Another approach is blackbox analysis, which does not require any binding. This was exploited in [33].

In 2004, Moen, Raddum and Hole [27] discovered that the recovery of at least two RC4 packet keys in WPA leads to a full recovery of the temporal key and the message integrity check key. Once from the same segment of 2^{16} consecutive packets, two keys are successfully recovered, the Moen, Raddum and Hole attack can be applied. This leads to a TK key recovery attack on WPA with complexity 2^{103} using 2 packets. Almost all known and new key recovery attacks on WEP could have been applied to WPA if there were several packets using the same RC4 key. Indeed, only the Fluhrer, Mantin and Shamir attack [9] is filtered. However, WPA uses a different secret key for every encrypted packet. In 2009, Tews and Beck [34] found a practical attack on WPA-PSK to inject data in encrypted communication. Note that this attack does not recover the encryption key and need some additional quality of services features (described by IEEE 802.11e) which are not activated by default.

Structure of the paper. We first present in Section 2 RC4, WEP, and WPA, known biases in RC4 and some tools to be able to manipulate a pool of biases for target key bytes. Then, we study key recovery attacks to be able to recover some “weak bits” of the temporary key in Section 3. We show applications to WEP in Section 4, then present a distinguisher for WPA and a full temporary key recovery for WPA in Section 5.

2 Preliminaries

2.1 Description of RC4 and Notations

The stream cipher RC4 consists of two algorithms: the Key Scheduling Algorithm (KSA) and the Pseudo Random Generator Algorithm (PRGA). The RC4 engine has a state defined by two registers (words) i and j and one array (of N words) S defining a permutation over $\mathbf{Z}_N$. The KSA generates an initial state for the PRGA from a random key K of L words as described on Fig. 1. It starts with an array $\{0,1,\dots,N-1\}$, where $N=2^8$ and swaps N pairs, depending on the value of the secret key K. At the end, we obtain the initial state $S'_0 = S_{N-1}$.

KSA

```
for $i = 0$ to $N-1$ do
    $S[i] \leftarrow i$
end for
$j \leftarrow 0$
for $i = 0$ to $N-1$ do
    $j \leftarrow j + S[i] + K[i \bmod L]$
    swap($S[i]$,$S[j]$)
end for
```

PRGA

```
$i \leftarrow 0$
$j \leftarrow 0$
loop
    $i \leftarrow i + 1$
    $j \leftarrow j + S[i]$
    swap($S[i]$,$S[j]$)
    output $z_i = S[S[i] + S[j]]$
end loop
```

Fig. 1. RC4 KSA and PRGA Algorithms

Once the initial state S'_0 is created, it is used by the second algorithm of RC4, the PRGA. Its role is to generate a keystream of words of $\log_2 N$ bits, which will be XORed with the plaintext to obtain the ciphertext. Thus, RC4 computes the loop of the PRGA each time a new keystream word z_i is needed, according to the algorithm on Fig. 1. Note that each time a word of the keystream is generated the internal state of RC4 is updated.

Notations. In this paper, we define all the operators such as addition, subtraction and multiplication in the group $\mathbf{Z}_N$ where $N = 256$ (i.e. *words* are *bytes*). Thus, $x+y$ should be read as $(x+y) \bmod N$.

Let $S_i[k]$ (resp. $S'_i[k]$) denote the value of the permutation defined by array S at index k, after round i in KSA (resp. PRGA). We also denote $S_{N-1} = S'_0$. Let j_i (resp. j'_i) be the value of j after round i of KSA (resp. PRGA) where the rounds are indexed with respect to i. Thus, the KSA has rounds $0, 1, \ldots, N-1$ and the PRGA has rounds $1, 2, \ldots$. The KSA and PRGA are defined by

KSA

$$\begin{aligned}
j_{-1} &= 0 \\
j_i &= j_{i-1} + S_{i-1}[i] + K[i \bmod L] \\
S_{-1}[k] &= k \\
S_i[k] &= \begin{cases} S_{i-1}[j_i] & \text{if } k = i \\ S_{i-1}[i] & \text{if } k = j_i \\ S_{i-1}[k] & \text{otherwise} \end{cases}
\end{aligned}$$

PRGA

$$\begin{aligned}
j'_0 &= 0 \\
j'_i &= j'_{i-1} + S'_{i-1}[i] \\
S'_0[k] &= S_{N-1}[k] \\
S'_i[k] &= \begin{cases} S'_{i-1}[j'_i] & \text{if } k = i \\ S'_{i-1}[i] & \text{if } k = j'_i \\ S'_{i-1}[k] & \text{otherwise} \end{cases} \\
z_i &= S'_i[S'_i[i] + S'_i[j'_i]]
\end{aligned}$$

2.2 Description of WEP

WEP [2] uses a 3-byte IV concatenated to a secret key of 40 or 104 bits (5 or 13 bytes) as an RC4 key. Thus, the RC4 key size is either 64 or 128 bits. In this paper, we do not consider the 40-bit key variant. So, $L = 16$. We have

$$K = K[0] \| K[1] \| K[2] \| K[3] \| \cdots \| K[15] = \mathsf{IV}_0 \| \mathsf{IV}_1 \| \mathsf{IV}_2 \| K[3] \| \cdots \| K[15]$$

where IV_i represents the $(i+1)$th byte of the IV and $K[3] \| \ldots \| K[15]$ the fixed secret part of the key. In theory, the value of the IV should be random but in practice, it is a counter, mostly in little-endian, and incremented by one each time a new 802.11*b* frame is encrypted. Sometimes, some particular values of IV are skipped to thwart specific attacks based on "weak IVs". Thus, each packet uses a slightly different key. RC4 then produces a keystream which is XORed to the plaintext to obtain the ciphertext.

It is well known [31] that a relevant portion of the plaintext is practically constant and that some other bytes can be predicted. They correspond to the LLC header and the SNAP header and some bytes of the TCP/IP encapsulated frame. For example, by XORing the first byte of the ciphertext with the constant value 0xAA, we obtain the first byte of the keystream. Thus, even if these attacks are called known plaintext attacks, they are ciphertext only in practice.

2.3 Description of WPA

WPA includes a key hash function [12] to defend against the Fluhrer-Mantin-Shamir attack [9], a Message Integrity Code (MIC) [7] and a key management scheme based on 802.1X [3] to avoid key reuse and to ease the key distribution.

The 128-bit Temporal Key (TK) is a per-session key. It is derived from the key management scheme during the authentication and is given as an input to the phase1 key hash function (key mixing algorithm) together with the 48-bit Transmitter Address (TA) and a 48-bit TKIP Sequence Counter (TSC) which is sometimes called IV. We will avoid this latter name to avoid confusion with the first 3 bytes of the RC4 key (which indeed only depend on TSC but are not equal).

TK can be used to encrypt up to 2^{48} packets. Every packet has a 48-bit index TSC which is split into IV32 and IV16. The IV32 counter is incremented every 2^{16} packets. The packet is encrypted using a 128-bit RC4KEY which is derived from TK, TSC, and some other parameters (e.g. device addresses) which can be assumed constant and known by the adversary for our purpose. As for WEP, the first three bytes of RC4KEY only depend on TSC so they are not secret. The derivation works in two phases. The first phase does not depend on IV16 and is done once every 2^{16} packets for efficiency reasons. It derives a 80-bit key TTAK, called TKIP-mixed Transmit Address and Key (TTAK) in the standard (but denoted P1K in the reference code).

$$\mathsf{TTAK} = \mathsf{phase1}(\mathsf{TK}, \mathsf{TA}, \mathsf{IV32})$$

The second phase uses TK and IV16 to derive a 96-bit key PPK which is then turned into RC4KEY:

$$\mathsf{RC4KEY} = \mathsf{phase2}(\mathsf{TK}, \mathsf{TTAK}, \mathsf{IV16})$$

The key derivation of WPA based on a pre-shared key is depicted on Fig. 2 (without protocol parameters such as TA).

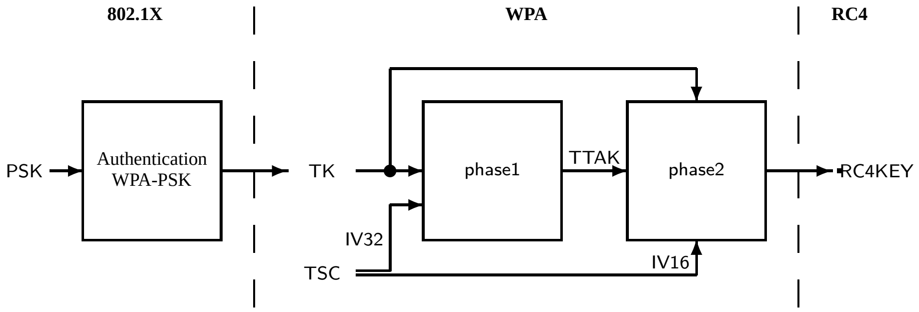

Fig. 2. WPA Key Derivation based on Pre-Shared Key Authentication Method

The RC4KEY is simply defined from PPK, TK, and IV16 by

$$\begin{array}{ll}
\mathsf{RC4KEY}[0] = \mathsf{high8}(\mathsf{IV16}) & \mathsf{RC4KEY}[1] = (\mathsf{high8}(\mathsf{IV16}) \text{ or } \mathsf{0x20}) \text{ and } \mathsf{0x7f} \\
\mathsf{RC4KEY}[2] = \mathsf{low8}(\mathsf{IV16}) & \mathsf{RC4KEY}[3] = \mathsf{low8}((\mathsf{PPK}[5] \oplus (\mathsf{TK}[1] \| \mathsf{TK}[0])) \gg 1) \\
\mathsf{RC4KEY}[4] = \mathsf{low8}(\mathsf{PPK}[0]) & \mathsf{RC4KEY}[5] = \mathsf{high8}(\mathsf{PPK}[0]) \\
\mathsf{RC4KEY}[6] = \mathsf{low8}(\mathsf{PPK}[1]) & \mathsf{RC4KEY}[7] = \mathsf{high8}(\mathsf{PPK}[1]) \\
\vdots & \vdots
\end{array}$$

In what follows, we denote $K[i] = \mathsf{RC4KEY}[i \bmod 16]$ and $\mathsf{IV} = K[0] \| K[1] \| K[2]$ to use the same notations as in WEP. By convention, TTAK and PPK are considered as vectors or 16-bit words. TK and $\mathsf{RC4KEY}$ are considered as vectors or 8-bit words. Vectors are numbered starting from 0.

Note that a filter avoids the use of some weak IV classes. Actually, only the weak IV class discovered by Fluhrer, Mantin, and Shamir [9].

2.4 Biases in RC4

Throughout this paper, we denote $\bar{K}[i] = K[0] + \cdots + K[i]$. We let z denote the keystream derived from K using RC4. The first bytes of a plaintext frame are often known (see [37]), as well as IV, the first 3 bytes of K. That is, we assume that the adversary can use z and IV in a known plaintext attack.

We let I_0 be a set of integers, which represent the key byte indices which are already known. We call an I_0-clue a value clue for all $\bar{K}$ bytes whose index are in I_0. To begin with, we have $I_0 = \{0,1,2\}$ and $\mathsf{clue} = \mathsf{IV}$.

Given a set of indices I_0 and an index i, we assume that we have a list $\mathsf{row}^{\mathsf{RC4}}_{i|I_0}$ of pairs $(\bar{f}_j, p_j)$ in which $\bar{f}_j$ is a function such that for any I_0-clue clue, we have

$$\Pr\left[\bar{K}[i] = \bar{f}_j(z, \mathsf{clue})\right] = p_j$$

For simplicity, we assume that for i, z, and clue given, all $\bar{f}_j(z, \mathsf{clue})$ are pairwise different. We further assume that the events $\bar{K}[i] = \bar{f}_j(z, \mathsf{clue})$ with different i's are independent. We will also assume that $\bar{f}_j$ is of form $\bar{f}_j(z, \mathsf{clue}) = f_j(h(z, \mathsf{clue}))$ where $\mu = h(z, \mathsf{clue})$ lies in a domain of size N_μ. h is just a function compressing data to the minimum necessary.

Table 1. Classes of Biases in RC4.

I_0 is the set of $\bar{K}$ indices which are already known, P_{MP}, P_{KI}, P_{008}, and P_{009} are defined in Appendix, t is the largest integer such that $0, 1, \ldots, t \in I_0$, and

$$\sigma_i = \sum_{j=0}^{t} S_{j-1}[j] + \sum_{j=t+1}^{i} S_t[j]$$

(e.g. $\sigma_i = \frac{i(i+1)}{2}$ and $t = -1$ when $0 \notin I_0$).

row	reference	$\bar{f}$	p	comment
$i \neq 1$	$\mathsf{MaitraPaul}(i, I_0)$	$\bar{K}[i] = z_{i+1} - \sigma_i$	$P_{\mathsf{MP}}(i,t)$	see [20]
i	$\mathsf{KleinImproved}(i, I_0)$	$\bar{K}[i] = -z_i + i - \sigma_i$	$P_{\mathsf{KI}}(i,t)$	see [37]
1	$\mathsf{SVV\_bb\_000}$	$\bar{K}[1] = z_1 - 1$	$1.04237/N$	see [33]
2	$\mathsf{SVV\_bb\_003}$	$\bar{K}[2] = z_2 - 3$	$0.65300/N$	see [33]
$i = 16i'$	$\mathsf{SVV\_008}(i, I_0)$	$\bar{K}[i] = z_i - i - \sigma_i$	$P_{008}(i,t)$	see [33]
$i = 16i'$	$\mathsf{SVV\_009}(i, I_0)$	$\bar{K}[i] = -z_i - i - \sigma_i$	$P_{009}(i,t)$	see [33]

We use a list of classes of biases from Table 1 (see [33]). More specifically, we use the rows $\mathsf{row}^{\mathsf{RC4}}_{i|I_0}$ in Table 2 taken from [33]. This table applies to RC4 in general but

Table 2. Table of Unconditional Biases in RC4 from known Key Bytes I_0

i	biases			
0	MaitraPaul(i,I_0)			
1	KleinImproved(i,I_0)	SVV_bb_000		
2	KleinImproved(i,I_0)	MaitraPaul(i,I_0)	SVV_bb_003	
3	KleinImproved(i,I_0)	MaitraPaul(i,I_0)		
⋮	⋮	⋮		
15	KleinImproved(i,I_0)	MaitraPaul(i,I_0)		
16	KleinImproved(i,I_0)	MaitraPaul(i,I_0)	SVV_008(i,I_0)	SVV_009(i,I_0)
17	KleinImproved(i,I_0)	MaitraPaul(i,I_0)		
⋮	⋮	⋮		
31	KleinImproved(i,I_0)	MaitraPaul(i,I_0)		
32	KleinImproved(i,I_0)	SVV_008(i,I_0)	SVV_009(i,I_0)	
33	KleinImproved(i,I_0)			
⋮	⋮			
47	KleinImproved(i,I_0)			

can be transformed for the WEP or WPA context due to $L = 16$. Indeed, we have $\bar{K}[i+16j] = \bar{K}[i] + j\bar{K}[15]$ for $0 \leq i \leq 15$ and $j = 0,1,2$. We define $\mathsf{deduce}(I)$ to be the set of all i's such that we can compute $\bar{K}[i]$ using this property, based on the values of $\bar{K}$ with indices in I. For instance, $\mathsf{deduce}(0,1,2,5) = \{0,1,2,5\}$ and $\mathsf{deduce}(0,1,2,5,15) = \{0,1,2,5,15,16,17,18,21,31,32,33,34,37,\ldots\}$. Next, we transform the above table by removing some rows for keys which can be deduced and by merging rows leading to the same key byte. Namely, we define $\mathsf{row}_{i|I_0}$ as follows: if $i \in \mathsf{deduce}(I_0)$, the row has a single "bias" $\bar{f}_1(z,\mathsf{clue}) = \bar{K}[i]$ with probability $p_1 = 1$ since $\bar{K}[i]$ can be computed from clue. Otherwise, the row is the concatenation of all $\mathsf{row}^{\mathsf{RC4}}_{i'|I_0}$ for i' in $\mathsf{deduce}(I_0 \cup \{i\}) - \mathsf{deduce}(I_0)$. For instance, $\mathsf{row}_{2|\{0,1,2\}}$ has a single bias, $\mathsf{row}_{5|\{0,1,2\}} = \mathsf{row}^{\mathsf{RC4}}_{5|\{0,1,2\}}$, and

$$\mathsf{row}_{5|\{0,1,2,15\}} = \mathsf{row}^{\mathsf{RC4}}_{5|\{0,1,2,15\}} \| \mathsf{row}^{\mathsf{RC4}}_{21|\{0,1,2,15\}} \| \mathsf{row}^{\mathsf{RC4}}_{37|\{0,1,2,15\}}$$

Given two lists of byte indices I_0 and $I = (i_1,\ldots,i_{\#I})$, we construct a new table $\Pi(I|I_0)$ in which the list of rows is $\mathsf{row}_{i_1|I_0}$, $\mathsf{row}_{i_2|I_0,i_1}$, ..., $\mathsf{row}_{i_{\#I}|I_0,i_1,i_2,\ldots,i_{\#I-1}}$. For instance, $I_0 = \{0,1,2\}$ and I is a list of key byte indices which are sequentially obtained using biases. We assume that I_0 is a minimal set in the sense that there is no strictly smaller set with same $\mathsf{deduce}(I_0)$. We further assume that I is a minimal set in the sense that there is no strictly smaller set with same $I \cap I_0$ and $\mathsf{deduce}(I \cup I_0)$. For instance, $I = (2,3,13,14,15)$ is minimal for $I_0 = \{0,1,2\}$, but $I = (2,3,13,14,15,16)$ is not. We define $\nu = (\bar{K}[i])_{i \in I}$ which belongs to a set of size $N_\nu(I) = N^{\#I}$. Given $i \in I$, we let $d_i^{\Pi(I|I_0)}$ be the length of row for $\bar{K}[i]$ in $\Pi(I|I_0)$. Given a tuple $(j_i)_{i \in I}$ such that $1 \leq j_i \leq d_i^{\Pi(I|I_0)}$ for all $i \in I$, by collecting together the j_ith bias of row i, we obtain an agglomerated bias to compute ν from z and an I_0-clue clue. Note that for technical reasons, we may have to keep elements of I_0 in I. This is why we may have rows for

$i \in I_0$ in $\Pi(I|I_0)$ with a single bias of probability 1. We let

$$k(I|I_0) = \prod_{i \in I} d_i^{\Pi(I|I_0)}$$

be the number of possible agglomerated biases. For convenience, we number the agglomerated biases with an index ℓ from 1 to $k(I|I_0)$ and each number defines a tuple $(j_i)_{i \in I}$. So, the ℓth bias is defined by $\nu = f_\ell(z, \mathsf{clue})$ with probability

$$p_\ell^{\Pi(I|I_0)} = \prod_{i \in I} p_{i,j_i}^{\Pi(I|I_0)}$$

where $p_{i,j}^{\Pi(I|I_0)}$ is the probability of the jth bias in the row corresponding to $\bar{K}[i]$ in $\Pi(I|I_0)$.

We let $N_\mu(\Pi(I))$ be N raised to the power of the number of z_i bytes and I_0 bytes appearing in any of the biased equations from $\Pi(I)$. E.g., $N_\mu(\Pi(3,13,14|0,1,2)) = N^8$ since biases for $\bar{K}[3]$ are based on z_3 and z_4, and biases for $\bar{K}[13]$ and $\bar{K}[14]$ are based on z_{13}, z_{14}, and z_{15}. We further need IV to compute S_t. So, we have 8 bytes in total: z_i for $i \in \{3,4,13,14,15\}$ and IV. Given a key stream z, we define $\mu = h^{\Pi(I)}(z, \mathsf{clue})$ as the vector of all z_i and clue bytes which are useful. We define $\nu = f_\ell^{\Pi(I)}(\mu)$.

For simplicity, we write Π, k, N_ν, N_μ, p_ℓ, h, and f_ℓ when I and I_0 will be made clear from context. That is, the range of h has size N_μ, and f_ℓ goes from a domain of N_μ elements to a range of N_ν elements.

In the following, we use

$$s_e = \sum_{\ell=1}^{k} p_\ell^e = \sum_{\substack{(j_i)_{i \in I} \\ 1 \le j_i \le d_i}} \prod_{i \in I} p_{i,j_i}^e = \prod_{i \in I} \sum_{j=1}^{d_i} p_{i,j}^e$$

for an integer e, and

$$\varepsilon_e = \left(\sum_{\ell=1}^{k} \left(p_\ell - \frac{1}{N_\nu} \right)^e \right)^{\frac{1}{e}} = \left(\sum_{i=0}^{e} \binom{e}{i} s_{e-i} (-N_\nu)^{-i} \right)^{\frac{1}{e}}$$

ε_e is called the *cumulated bias of order e*. The table below gives a few examples of cumulated biases.

I_0	I	N_ν	k	N_μ	ε_1	ε_2	ε_4
$\{0,1,2\}$	$(3,13,14)$	2^{24}	2^3	N^8	$2^{-21.37}$	$2^{-22.79}$	$2^{-23.40}$
$\{0,1,2\}$	$(15,3,14)$	2^{24}	$2^{8.81}$	N^{20}	$2^{-16.60}$	$2^{-20.79}$	$2^{-22.69}$
$\{0,1,2\}$	$(15,3,13,14)$	2^{32}	$2^{11.13}$	N^{23}	$2^{-21.82}$	$2^{-27.19}$	$2^{-29.69}$

2.5 Conditional Biases in RC4

We extend the notion of bias to the notion of conditional bias. We now assume that for each i we have d_i functions $\bar{f}_{i,j}$ and corresponding predicates $\bar{g}_{i,j}$ such that

$$\Pr\left[\bar{K}[i] = \bar{f}_j(z, \mathsf{clue}) | \bar{g}_j(z, \mathsf{clue})\right] = p_j$$

for some probability $p_j \neq \frac{1}{N}$. We further define

$$\Pr\left[\bar{g}_j(z,\mathsf{clue})\right] = q_j$$

and call q_j the *density* of the bias. For simplicity, we assume that for some given i, z, and clue, all suggested $\bar{f}_j(z,\mathsf{clue})$ when $\bar{g}_j(z,\mathsf{clue})$ holds, are pairwise distinct. We further assume that the events $\bar{K}[i] = \bar{f}_j(z,\mathsf{clue})$ with different i's are independent. We will also assume that $\bar{f}_j$ and $\bar{g}_j$ are of form $\bar{f}_j(z,\mathsf{clue}) = f_j(h(z,\mathsf{clue}))$ and $\bar{g}_j(z,\mathsf{clue}) = g_j(h(z,\mathsf{clue}))$ where $\mu = h(z,\mathsf{clue})$ lies in a domain of size N_μ.

We use the conditional biases in Table 5. All of them except SVV_db were taken from Korek [18] (they can be extracted from Aircrack, see [6,37]). We used some new formulas to compute their probabilities which are given in Appendix.

Given two minimal sets of byte indices I_0 and I as in the previous section, we also make a table $\Pi(I|I_0)$ and collect a list of ℓ agglomerated biases in which probabilities and densities are multiplied. We define

$$\bar{s}_e = \sum_{\ell=1}^{k} q_\ell p_\ell^e \quad , \quad \bar{s}_e^{(N_x)} = \sum_{\ell=1}^{k} \frac{q_\ell}{1-\frac{q_\ell}{N_x}} p_\ell^e$$

and

$$\begin{aligned} \bar{\varepsilon}_e &= \sqrt[e]{\sum_{\ell=1}^{k} q_\ell \left(p_\ell - \frac{1}{N_\mathsf{v}}\right)^e} = \sqrt[e]{\sum_{i=0}^{e} \binom{e}{i} \bar{s}_{e-i} (-N_\mathsf{v})^{-i}} \\ \bar{\varepsilon}_e^{(N_x)} &= \sqrt[e]{\sum_{\ell=1}^{k} \frac{q_\ell}{1-\frac{q_\ell}{N_x}} \left(p_\ell - \frac{1}{N_\mathsf{v}}\right)^e} = \sqrt[e]{\sum_{i=0}^{e} \binom{e}{i} \bar{s}_{e-i}^{(N_x)} (-N_\mathsf{v})^{-i}} \end{aligned}$$

$\bar{s}_e^{(N_x)}$ resp. $\bar{\varepsilon}_e^{(N_x)}$ is the regular $\bar{s}_e$ resp. $\bar{\varepsilon}_e$ with a special correcting factor depending on some value N_x. This correction may look arbitrary. It will appear in the analysis of Section 3. The $\bar{s}$ values can be computed easily by

$$\bar{s}_e = \sum_{(j_i)_{i\in I}} \prod_{i\in I} q_{i,j_i} p_{i,j_i}^e = \prod_{i\in I} \sum_{j=1}^{d_i} q_{i,j} p_{i,j}^e$$

In the sequel, when $q_\ell \neq 1$ we assume $q_\ell \ll 1$ to approximate $\frac{1}{1-\frac{q_\ell}{N_x}} \approx 1 + \frac{1}{N_x-1} 1_{q_\ell=1}$. So, we compute $\bar{s}_e^{(N_x)}$ like for $\bar{s}_e$ but add a fraction of the regular s_e term for unconditional biases.

$$\bar{s}_e^{(N_x)} = \sum_{(j_i)_{i\in I}} \prod_{i\in I} \frac{q_{i,j_i}}{1-\frac{q_{i,j_i}}{N_x}} p_{i,j_i}^e \approx \frac{s_e}{N_x-1} + \prod_{i\in I} \sum_{j=1}^{d_i} q_{i,j} p_{i,j}^e$$

The approximation is very useful to estimate $\bar{s}_e^{(N_x)}$ with low complexity. Namely, we can compute all useful $\bar{\varepsilon}_e$'s in time $O(ed)$ where d is the total number of biases, although the number of agglomerated biases k is of order $d^{\#I}$ which can be very large.

2.6 More Definitions

We denote

$$\varphi(\lambda) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\lambda} e^{-\frac{x^2}{2}}\,dx = \frac{1}{2}\mathsf{erfc}\left(-\frac{\lambda}{\sqrt{2}}\right)$$

In particular, $\varphi(-\lambda/\sqrt{2}) = \frac{1}{2}\mathsf{erfc}(\frac{\lambda}{2})$.

3 Attacking Weak Bits Based on Biases

There are 8 bits of TK that we call *weak* because they have a simple relation with bits in PPK. These bits consists of the 7 most significant bits of $\mathsf{TK}[0]$ and the least significant bit of $\mathsf{TK}[1]$. We will define statistical attacks using the following mappings:

$$\begin{tikzcd} z^m, \mathsf{IV}^m \arrow[r, "h"] & \mu \arrow[r, "f_\ell", "\text{if } g_\ell(\mu)"'] & \nu \arrow[r, "\pi"] & x \end{tikzcd}$$

Here, z^m is the mth keystream using IV^m, μ is a compressed information to compute ν, some RC4 key bytes which are used to compute x, some information about TK which we want to recover using statistics. We define N_x the number of possible values for x.

3.1 First Attack: Recovering some Weak Bits of TK

We use $I_0 = \{0,1,2\}$ and $I = (2,3,13,14)$. Given $\bar{K}[2], \bar{K}[3], \bar{K}[13], \bar{K}[14]$, the adversary can compute $K[3] = \bar{K}[3] - \bar{K}[2]$ and $K[14] = \bar{K}[14] - \bar{K}[13]$. We have

$$\begin{aligned} \mathsf{PPK}[5] &= K[15] \| K[14] \\ K[3] &= \mathsf{low8}((\mathsf{PPK}[5] \oplus (\mathsf{TK}[1] \| \mathsf{TK}[0])) \gg 1) \end{aligned}$$

So, given $\nu = (\bar{K}[2], \bar{K}[3], \bar{K}[13], \bar{K}[14])$ the adversary can compute $x = \mathsf{high7}(\mathsf{TK}[0])$ by

$$\pi(\nu) = \mathsf{low7}((\bar{K}[3] - \bar{K}[2]) \oplus ((\bar{K}[14] - \bar{K}[13]) \gg 1))$$

$N_\nu = 2^{32}$ is the total number of possible ν's and $N_x = 2^7$ is the total number of possible x's. We have $N_\mu = 2^{48}$, the total number of $\mu = h(z, \mathsf{IV})$.

We can recover the 7 weak bits as follows: for each candidate value x, each packet m, and each $\ell = 1, \ldots, k$ (corresponding to a tuple $(j_2, j_3, j_{13}, j_{14})$), if agglomerated condition $g_\ell(h(z^m, \mathsf{IV}^m))$ holds, we define $\nu = f_\ell(h(z^m, \mathsf{IV}^m))$ the value of RC4 key bytes suggested by bias ℓ on packet m, which is correct with probability p_ℓ. We let $x = \pi(\nu)$ the suggested value of x computed as explained. We let $X_{x,m,\ell}$ be some magic coefficient a_ℓ (to be optimized later) if $\pi(f_\ell(h(z^m, \mathsf{IV}^m))) = x$ and 0 otherwise. We let $Y_x = \sum_{m=1}^{n} \sum_{\ell=1}^{k} X_{x,m,\ell}$ where n is the total number of packets to be used. Clearly, the correct value for ν is suggested with probability p_ℓ and others are obtained randomly. We assume incorrect ones are suggested with the same probability $\frac{1-p_\ell}{N_\nu - 1}$.

If x is not the correct value, it is not suggested for sure when ν is correct. Since π is balanced, this incorrect x has $\frac{N_\nu}{N_x}$ values ν belonging to the set of $N_\nu - 1$ incorrect ones.

So, x is suggested with with probability $\frac{N_\nu}{N_x} \times \frac{1-p_\ell}{N_\nu - 1}$. So, the $X_{x,m,\ell}$ for incorrect x's are random variables (r.v.) with expected values

$$a_\ell q_\ell N_\nu \frac{1-p_\ell}{N_x(N_\nu - 1)}$$

if x is not the correct value.

If x is the correct value, it is suggested with probability p_ℓ for the correct ν and when ν is one of the $\frac{N_\nu - N_x}{N_x}$ (incorrect) preimages of x by π. That is, with overall probability $p_\ell + \frac{N_\nu - N_x}{N_x} \times \frac{1-p_\ell}{N_\nu - 1}$. So, the $X_{x,m,\ell}$ for the correct x are r.v. with expected values

$$a_\ell q_\ell N_\nu \frac{1-p_\ell}{N_x(N_\nu - 1)} + a_\ell q_\ell \frac{N_\nu p_\ell - 1}{N_\nu - 1}$$

The difference between these two expected values is important but it is not the same for the variance. Since every x is suggested with probability roughly $\frac{q_\ell}{N_x}$, we assume that the variance of all $X_{x,m,\ell}$ can be approximated by $\frac{q_\ell}{N_x}\left(1 - \frac{q_\ell}{N_x}\right) a_\ell^2$. Let Δ be the operator making the difference between distributions for a good x and a bad one. We have

$$\begin{aligned}
E(Y_{\mathsf{bad}}) &= \frac{n}{N_x\left(1 - \frac{1}{N_\nu}\right)} \sum_\ell a_\ell q_\ell (1 - p_\ell) \\
\Delta E(Y) &= \frac{n}{1 - \frac{1}{N_\nu}} \sum_\ell a_\ell q_\ell \left(p_\ell - \frac{1}{N_\nu}\right) \\
V(Y) &\approx n \sum_\ell a_\ell^2 \frac{q_\ell}{N_x}\left(1 - \frac{q_\ell}{N_x}\right)
\end{aligned}$$

Where $E(Y_{\mathsf{bad}})$ denotes the expected value of an Y_x variable for any bad x. Here, we removed the subscript x of Y_x in $\Delta E(Y)$ and $V(Y)$ since these do not depend on a specific value for x. Let λ be such that $\Delta E(Y) = \lambda\sqrt{V(Y)}$. The probability that the correct Y_x is lower than any wrong Y_x is $\rho = \varphi\left(-\frac{\lambda}{\sqrt{2}}\right)$. That is, the expected number of wrong x's with larger Y_x is

$$r = (N_x - 1)\varphi\left(-\frac{\lambda}{\sqrt{2}}\right) \tag{1}$$

So,

$$n = \lambda^2 \left(1 - \frac{1}{N_\nu}\right)^2 \frac{\sum_\ell a_\ell^2 \frac{q_\ell}{N_x}\left(1 - \frac{q_\ell}{N_x}\right)}{\left(\sum_\ell a_\ell q_\ell \left(p_\ell - \frac{1}{N_\nu}\right)\right)^2}$$

By derivating both terms of the fraction with respect to a_ℓ and equaling them, we obtain that the optimal value is reached for

$$a_\ell = \frac{N_x}{N_x - q_\ell}\left(p_\ell - \frac{1}{N_\nu}\right)$$

This leads us to

$$\begin{aligned} E(Y_{\mathsf{bad}}) &= \frac{n}{N_y}\left(\overline{\varepsilon}_1^{(N_x)} - \frac{1}{1-\frac{1}{N_\nu}}(\overline{\varepsilon}_2^{(N_x)})^2\right) \\ \Delta E(Y) &= \frac{n}{1-\frac{1}{N_\nu}}(\overline{\varepsilon}_2^{(N_x)})^2 \\ V(Y) &\approx \frac{n}{N_x}(\overline{\varepsilon}_2^{(N_x)})^2 \\ n &= \frac{\lambda^2}{N_x\left(\overline{\varepsilon}_2^{(N_x)}\right)^2}\left(1-\frac{1}{N_\nu}\right)^2 \end{aligned} \tag{2}$$

So we can see where the correction in $\overline{\varepsilon}_2^{(N_x)}$ appears.

The attack works as follows:

```
set $I = (2,3,13,14)$ and $I_0 = \{0,1,2\}$
initialize the $Y_x$ counters to 0
**for** $m = 1$ to $n$ **do**
    **for** $\ell = 1$ to $k$ **do**
        **if** $g_\ell(h(z^m, \mathsf{IV}^m))$ holds **then**
            compute $\nu = f_\ell(h(z^m, \mathsf{IV}^m))$, the suggested $(\overline{K}[2], \overline{K}[3], \overline{K}[13], \overline{K}[14])$
            compute $x = \pi(\nu)$
            increment $Y_x$ by $a_\ell = \frac{N_x}{N_x - q_\ell}\left(p_\ell - \frac{1}{N_\nu}\right)$
        **end if**
    **end for**
**end for**
output $x = \arg\max_x Y_x$
```

Clearly, the time complexity is nk. The complexity is measured in terms of number of times the **if** structure is executed. This should have a complexity which is essentially equivalent to executing the phase2 key derivation. The memory complexity has the order of magnitude of N_x. Here is a variant:

```
set $I = (2,3,13,14)$ and $I_0 = \{0,1,2\}$
initialize a table $y_x^\mu$ to 0
**for** $\ell = 1$ to $k$ **do**
    **for** all possible $\mu$ such that $g_\ell(\mu)$ holds **do**
        compute $x = \pi(f_\ell(\mu))$
        increment $y_x^\mu$ by $a_\ell = \frac{N_x}{N_x - q_\ell}\left(p_\ell - \frac{1}{N_\nu}\right)$
    **end for**
**end for**
initialize the $Y_x$ counters to 0
**for** $m = 1$ to $n$ **do**
    **for** all $x$ **do**
        compute $\mu = h(z^m, \mathsf{IV}^m)$
        increment $Y_x$ by $y_x^\mu$
    **end for**
```

```
end for
output $x = \arg\max_x Y_x$
```

Now, the time complexity is $N_\mu k + N_x n$ and the memory complexity is $N_\mu N_x$. So, let say that the complexity is

$$c = \min(nk, N_\mu k + N_x n) \tag{3}$$

The two complexity curves cross for $n = N_\mu \frac{k}{k-N_x} \approx N_\mu$ when $N_x \ll k$.

For $I = (2,3,13,14)$, we have $N_\nu = 2^{32}$, $N_\mu = 2^{48}$, and $N_x = 2^7$. The complexities with and without using conditional biases are summarized in Table 3. As we can see, when ignoring the conditional biases, we need about 30% more packets but the complexity is much lower because k is smaller. So, conditional biases do not seem useful in this case.

3.2 Second Attack

Let $I_0 = \{0,1,2\}$, $I = (15,2,3,14)$, and $x = \mathsf{low1}(\mathsf{TK}[1])$ be the last weak bit. Given IV and $\nu = (\bar{K}[2], \bar{K}[3], \bar{K}[14], \bar{K}[15])$, we deduce $x = \pi(\nu)$ by

$$\pi(\nu) = \mathsf{high1}((\bar{K}[3] - \bar{K}[2]) \oplus (\bar{K}[15] - \bar{K}[14]))$$

So, we apply the first attack with this I and $N_x = 2$. Since $15 \in I$ we have more biases. We have r, n, and c from Eq. (1), Eq. (2) and Eq. (3).

For $I = (15,2,3,14)$, we have $N_\nu = 2^{32}$, $N_\mu = 2^{48}$, and $N_x = 2$. The complexities are summarized in Table 3. Again, conditional biases are not very useful. We can also see that this choice of I leads to a much better attack than the one from Section 3.1 in terms of n but the complexity is slightly higher. This is due to a larger k.

3.3 Merging Attacks

Given two attacks with sets I^1 resp. I^2 for recovering independent x^1 resp. x^2 leading to characteristics Y_x^1 resp. Y_x^2, c^1 resp. c^2, n^1 resp. n^2, λ^1 resp. λ^2, one problem is to merge the sorted lists of x^1 and x^2. One can follow the approach by Junod-Vaudenay [15]. We sort pairs following their likelihood ratio, which is obtained by multiplying the likelihood ratio of both terms. We assume that all Y_x^i are independent, normally distributed with variance $V(Y^i)$, and expected value either $E(Y_{\mathsf{bad}}^i)$ or $E(Y_{\mathsf{bad}}^i) + \Delta E(Y^i)$. Given x^i, the ratio for x^i being the correct value based on the observation $Y_{x^i}^i$ is

$$\frac{\Pr[Y_{x^i}^i | x^i \text{ good}]}{\Pr[Y_{x^i}^i | x^i \text{ wrong}]} = \frac{\frac{1}{\sqrt{2\pi V(Y^i)}} e^{-\frac{\left(Y_{x^i}^i - E(Y_{\mathsf{bad}}^i) - \Delta E(Y^i)\right)^2}{2V(Y^i)}}}{\frac{1}{\sqrt{2\pi V(Y^i)}} e^{-\frac{\left(Y_{x^i}^i - E(Y_{\mathsf{bad}}^i)\right)^2}{2V(Y^i)}}}$$
$$= e^{Y_{x^i}^i \frac{\Delta E(Y^i)}{V(Y^i)} + \frac{\Delta E(Y^i)}{2V(Y^i)}(\Delta E(Y^i) - E(Y_{\mathsf{bad}}^i))}$$

So, when multiplying some terms of this form for pairs of values, sorting them by decreasing product is equivalent to sorting them by decreasing value of

$$Y_{x^1,x^2} = Y^1_{x^1}\frac{\Delta E(Y^1)}{V(Y^1)} + Y^2_{x^2}\frac{\Delta E(Y^2)}{V(Y^2)}$$

With same assumptions as in [15], we are back in the situation where Y_{x^1,x^2} is normally distributed. We have

$$\Delta E(Y) = V(Y) = \frac{(\Delta E(Y^1))^2}{V(Y^1)} + \frac{(\Delta E(Y^2))^2}{V(Y^2)} = (\lambda^1)^2 + (\lambda^2)^2$$

So, $\lambda = \sqrt{(\lambda^1)^2 + (\lambda^2)^2}$, and the average number of wrong (x^1, x^2) pair with higher score than the good one is $r = (N^1_x N^2_x - 1)\varphi(-\frac{\lambda}{\sqrt{2}})$. Overall, we can use

$$n = \frac{\lambda^2}{N_{x^1}\left(\frac{\bar{\varepsilon}_2^{(N_{x^1})}(1)}{1-\frac{1}{N^1_\nu}}\right)^2 + N_{x^2}\left(\frac{\bar{\varepsilon}_2^{(N_{x^2})}(2)}{1-\frac{1}{N^2_\nu}}\right)^2}$$

and $c = c^1 + c^2$ by using Eq. (3) for c^1 and c^2. We can use these merging rules to merge the two previous attacks. We obtain the results from Table 3.

Table 3 shows the complexity when merging the previous attacks to recover the 8 weak bits of TK. We compare it with the attack using a merged set I directly. As we can see, merging attacks with small I's is much better.

Table 3. Complexities of several attacks to recover $\log_2 N_x$ bits from TK. We compare them when including conditional biases and without. We provide the number of packets n, the running time complexity c, the expected number r of better wrong values, as well as parameters k, $\varepsilon = \bar{\varepsilon}_2^{(N_x)}$, λ, and N_ν. Except when $N_x = 2$ for which it would not make any sense, we target $r = \frac{1}{2}$ (that is, the correct value has the higher score in half of the cases, roughly). We used $I_0 = \{0,1,2\}$.

	reference	I	n	c	r	N_x	k	ε	λ	N_ν	N_μ	cond. biases
1u	Section 3.1	$(2,3,13,14)$	$2^{40.13}$	$2^{43.13}$	$\frac{1}{2}$	2^7	$2^{3.00}$	$2^{-21.65}$	3.76	2^{32}	N^8	without
1c	Section 3.1	$(2,3,13,14)$	$2^{39.70}$	$2^{51.87}$	$\frac{1}{2}$	2^7	$2^{12.17}$	$2^{-21.44}$	3.76	2^{32}	N^{10}	with
2u	Section 3.2	$(15,2,3,14)$	$2^{36.10}$	$2^{44.91}$	$\frac{1}{4}$	2	$2^{8.81}$	$2^{-18.62}$	0.95	2^{32}	N^{20}	without
2c	Section 3.2	$(15,2,3,14)$	$2^{35.98}$	$2^{54.35}$	$\frac{1}{4}$	2	$2^{18.37}$	$2^{-18.56}$	0.95	2^{32}	N^{22}	with
3u	merge 1u+2u		$2^{39.33}$	$2^{48.17}$	$\frac{1}{2}$	2^8			4.08			without
3c	merge 1c+2c		$2^{39.05}$	$2^{57.43}$	$\frac{1}{2}$	2^8			4.08			with
4u		$(15,2,3,13,14)$	$2^{47.67}$	$2^{58.81}$	$\frac{1}{2}$	2^8	$2^{11.14}$	$2^{-25.81}$	4.08	2^{40}	N^{23}	without
4c		$(15,2,3,13,14)$	$2^{47.36}$	$2^{71.37}$	$\frac{1}{2}$	2^8	$2^{24.01}$	$2^{-25.65}$	4.08	2^{40}	N^{25}	with

We may think that we could get better results by using the entire vector Y instead of Y_x only to compute the likelihood ratio of x. By redoing the computations, we obtain

$$\frac{\Pr[Y|x^i\ \mathsf{good}]}{\Pr[Y|x^i\ \mathsf{wrong}]} = \frac{\Pr[Y|x^i\ \mathsf{good}]}{\frac{1}{N^i_x-1}\sum_{x\neq x^i}\Pr[Y|x\ \mathsf{good}]} = \frac{N_x - 1}{\sum_{x'\neq x} e^{(Y_{x'}-Y_x)\frac{\Delta E(Y)}{V(Y)}}}$$

When x is good and x' is bad, the exponential in the sum is of order $e^{-\lambda}$. When x is bad and x' is good, it has order e^{λ}. When both are bad, it has order $e^{\pm\sqrt{\lambda}}$. So, we have to compare one ratio of order e^{λ} to others of order $\frac{N_x-1}{e^{\lambda}+(N_x-2)e^{\pm\sqrt{\lambda}}}$. We know that a wrong ratio is higher than the good one with probability $\varphi(-\lambda/\sqrt{2})$. When multiplying the independent likelihood ratios for x^1 and x^2, if we approximate $\sum_{x'_1\neq x^1} F(x'_1)\sum_{x'_2\neq x^2} G(x'_2) \approx \sum_{(x'_1,x'_2)\neq(x^1,x^2)} F(x'_1)G(x'_2)$, we obtain a likelihood ratio of same form based on Y_{x^1,x^2}. This validates the above rule of the thumb for sorting pairs following their Y_{x^1,x^2} score.

4 Attack on WEP

We apply the first attack with $x = \nu$: we only want to recover key bytes which are the same for all packets. This attack produces a ranking of possible x's in a form of a list $\mathcal{L}$ by decreasing order of likelihood.

We use the following attack:

```
compute the ranking $\mathcal{L}_{15}$ for $I = (15)$ and $I_0 = \{0,1,2\}$
**for** each $\bar{k}_{15}$ in $\mathcal{L}_{15}$ **do**
    run recursive attack on input $\bar{k}_{15}$
**end for**
stop: attack failed
**recursive attack with input** $(\bar{k}_{15},\bar{k}_3,\ldots,\bar{k}_{i-1})$**:**
**if** $i \leq i_{\max}$ **then**
    compute the ranking $\mathcal{L}_i$ for $I = (i)$ and $I_0 = \{0,\ldots,i-1,15\}$
    truncate $\mathcal{L}_i$ to its first $\rho_i$ terms
    **for** each $\bar{k}_i$ in $\mathcal{L}_i$ **do**
        run recursive attack on input $(\bar{k}_{15},\bar{k}_3,\ldots,\bar{k}_{i-1},\bar{k}_i)$
    **end for**
**else**
    **for** each $\bar{k}_{i_{\max}+1},\ldots,\bar{k}_{14}$ **do**
        test key $(k_3,\ldots,k_{14},k_{15})$ and stop if correct
    **end for**
**end if**
```

Let $\varepsilon_i = \bar{\varepsilon}_2^{(N_x)}(i|0,\ldots,i-1,15)$ for $i = 3,\ldots,i_{\max}$ and $\varepsilon_{15} = \bar{\varepsilon}_2^{(N_x)}(15|0,1,2)$ be the ε used by the attack on $\bar{K}[i]$. Similarly, let $N_x = N_\nu = N$, and r_i, k_i, λ_i, c_i be their parameters following Eq. (1,2,3). Let R_i be the rank of the correct k_i value in L_i. We know that $E(R_i) = r_i$. We can easily see that $V(R_i) = r_i\left(1-\frac{r_i}{N_x-1}\right)$. By using the law of large numbers, the probability that R_i is lower than ρ_i is $u_i = \varphi\left(\frac{\rho_i - r_i}{\sqrt{r_i\left(1-\frac{r_i}{N_x-1}\right)}}\right)$ so the success probability is $\prod_{i=3}^{i_{\max}} u_i$ and the complexity is

$$c = c_{15} + r_{15}\left(c_3 + \rho_3\left(c_4 + \rho_4\left(\cdots c_{i_{\max}} + \rho_{i_{\max}} N^{14-i_{\max}}\cdots\right)\right)\right)$$

To approximate the optimal choice of ρ's, we set $\rho_i = r_i + \alpha\sqrt{r_i\left(1-\frac{r_i}{N_x-1}\right)}$ for some α. The success probability is $\varphi(\alpha)^{i_{\max}-2}$. We can adjust $\alpha = \varphi^{-1}(2^{-\frac{1}{i_{\max}-2}})$ so that this becomes 50% and we obtain c in terms of n. Computation shows that figures are better for $i_{\max} = 14$. For this, we have $\alpha \approx 1.588$. We plotted $\log_2 c$ in terms of n on Fig. 3.

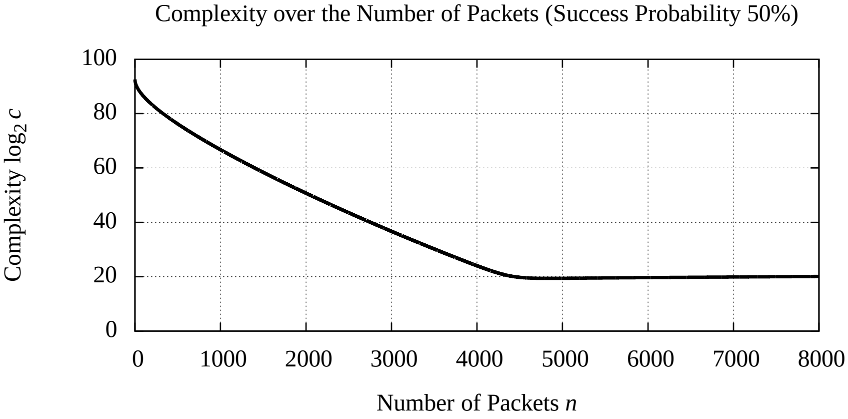

Fig. 3. Logarithmic complexity in terms of data complexity for breaking WEP

Note that this computation assumes real values for the ρ's. Since they must be integer, the real complexity may be slightly higher. For instance, with $n = 4\,000$, the plotted complexity is $2^{24.02}$. With integral values, we can try with $\rho_i = 5$ for $i = 3,5,6$ and $\rho_i = 4$ for $i = 4,7,8,\ldots,14$. We obtain $c = 2^{24.35}$ and a success rate of 51%.

Note that without the conditional biases, the same analysis with $4\,000$ gives a complexity of 2^{66}. So, these biases make a huge difference in this case.

5 Attack on WPA

5.1 Distinguishing WPA

The first attack can be turned into a distinguisher as follows. The expected value and variance of the correct Y_x are roughly $E(Y_{\mathsf{bad}}) + \lambda\sqrt{V(Y)}$ and $V(Y)$. The random variable Y_x is larger than $T = E(Y_{\mathsf{bad}}) + \lambda'\sqrt{V(Y)}$ with probability $\varphi(\lambda - \lambda')$. Now, if we replace the WPA packets by some random sequences, the counters all have expected value $E(Y_{\mathsf{bad}})$ and variance approximately $V(Y)$. The probability that a given counter exceeds T is $\varphi(-\lambda')$. The probability that any counter exceeds this is lower than $N_x\varphi(-\lambda')$. So, the condition $\max_x Y_x > T$ makes a distinguisher of same n and c as in the first attack, and with advantage larger than $\varphi(\lambda - \lambda') - N_x\varphi(-\lambda')$. We find the optimal $\lambda' = \frac{\lambda}{2} + \frac{\ln N_x}{\lambda}$. So, $\mathsf{Adv} \geq \beta$ with

$$\beta = \varphi\left(\frac{\lambda}{2} - \frac{\ln N_x}{\lambda}\right) - N_x\varphi\left(-\frac{\lambda}{2} - \frac{\ln N_x}{\lambda}\right) \tag{4}$$

We use the same values as before and target $\mathsf{Adv} \geq \frac{1}{2}$. We use Eq. (2) for n, Eq. (3) for c, and Eq. (4) for a lower bound β of the advantage. The performances of the distinguishers are summarized on Table 4. Again, the attack based on $I = (15,2,3,14)$ is better in terms of number of packets but not in terms of complexity. It works using 2^{38} packets and complexity 2^{47}. The one based on $I = (2,3,13,14)$ works with 30% more packets (2^{40}) with no conditional biases but with a much better complexity 2^{43}.

Table 4. Complexities of several distinguishers for WPA. We compare them when including conditional biases and without. We provide the number of packets n, the running time complexity c, the bound on the advantage β, as well as parameters k, $\varepsilon = \bar{\varepsilon}_2^{(N_x)}$ or ε_2, λ, and N_ν. We target $\beta = \frac{1}{2}$. We used $I_0 = \{0,1,2\}$.

	I	n	c	β	N_x	k	ε	λ	N_ν	N_μ	cond. biases
1u	$I = (2,3,13,14)$	$2^{39.85}$	$2^{42.85}$	0.5	2^7	$2^{3.00}$	$2^{-21.65}$	3.41	2^{32}	N^8	without
1c	$I = (2,3,13,14)$	$2^{39.42}$	$2^{51.59}$	0.5	2^7	$2^{12.17}$	$2^{-21.44}$	3.41	2^{32}	N^{10}	with
2u	$I = (15,2,3,14)$	$2^{37.94}$	$2^{46.76}$	0.5	2	$2^{8.81}$	$2^{-18.62}$	1.81	2^{32}	N^{20}	without
2c	$I = (15,2,3,14)$	$2^{37.82}$	$2^{56.19}$	0.5	2	$2^{18.37}$	$2^{-18.56}$	1.81	2^{32}	N^{22}	with

5.2 Temporary Key Recovery

The results from [27] lead to an "easy" attack on WPA: guess the 96-bit PPK and the 8 weak bits of TK within an average complexity of 2^{103} until it generates the correct keystream. Then, guess the 96-bit PPK of another packet in the same segment (with the weak bits already known). Then, apply the method of [27] to recover TK. We improve this attack by recovering the weak bits of TK separately: we know from Table 3 that we can recover the weak bits of TK by using 2^{38} packets. After having recovered the weak bits, we note that the 96-bit PPK is now enough to recalculate RC4KEY. So, we can do an exhaustive search on PPK for a given packet until we find the correct one generating the packet. This works with complexity 2^{95} on average. We do it twice to recover the PPK of two packets in the same segment. Given these two PPK sharing the same IV32, we recover TK by using the method of [27]. Therefore, we can recover the temporary key TK and decrypt all packets with complexity 2^{96}. The number of packets needed to recover the weak bits is 2^{38}.

6 Conclusion

We deployed a framework to handle pools of biases for RC4 which can be used to break WPA. In the case of the 8 weak bits of TK, we have shown a simple distinguisher and a partial key recovery attack working with 2^{38} packets and practical complexity. This can be used to improve the attack by Moen-Raddum-Hole [27] to mount a full temporary key recovery attack of complexity 2^{96} using 2^{38} packets. So far, this is the best temporal key recovery attack against WPA. In a future work we plan to study further key recovery attacks to recover more pieces of TK with complexity lower than 2^{96}.

We have shown that conditional biases are not very helpful for breaking WPA but they really are against WEP. Indeed, we recover keys with a success rate 50% by using 4000 packets and a complexity of 2^{26}.

References

1. ANSI/IEEE standard 802.11i, Amendment 6 Wireless LAN Medium Access Control (MAC) and Physical Layer (phy) Specifications, Draft 3. IEEE. 2003.
2. IEEE Std 802.11, Standards for Local and Metropolitan Area Networks: Wireless Lan Medium Access Control (MAC) and Physical Layer (PHY) Specifications. 1999.
3. IEEE 802.1 WG. 802.1x: Standards for Local and Metropolitan Area Networks: Port-Based Access Control. IEEE. 2001.
4. E. Biham, Y. Carmeli. Efficient Reconstruction of RC4 Keys from Internal States. In *Fast Software Encryption'08*, Lausanne, Lecture Notes in Computer Science 5086, pp. 270–288, Springer-Verlag, 2008.
5. A. Bittau. Additional Weak IV Classes for the FMS Attack. 2003.
http://www.netstumbler.org/showthread.php?postid=89036#pos%t89036
6. R. Chaabouni. Breaking WEP Faster with Statistical Analysis. Semester project. EPFL/LASEC. 2006.
7. N. Ferguson. Michael: an Improved MIC for 802.11 WEP. IEEE doc. 802.11-2/020r0. 2002.
8. S. R. Fluhrer, D. A. McGrew. Statistical Analysis of the Alleged RC4 Keystream Generator. In *Fast Software Encryption'00*, New York, NY, USA, Lecture Notes in Computer Science 1978, pp. 19–30, Springer-Verlag, 2001.
9. S. R. Fluhrer, I. Mantin, A. Shamir. Weaknesses in the Key Scheduling Algorithm of RC4. In *Selected Areas in Cryptography'01*, Toronto, Ontario, Canada, Lecture Notes in Computer Science 2259, pp. 1–24, Springer-Verlag, 2001.
10. J. Dj. Golic. Iterative Probabilistic Cryptanalysis of RC4 Keystream Generator. In *Information Security and Privacy: 5th Australian Conference (ACISP'00)*, Brisbane, Australia, Lecture Notes in Computer Science 1841, pp. 220–223, Springer-Verlag, 2000.
11. J. Dj. Golic. Linear Statistical Weakness of Alleged RC4 Keystream Generator. In *Advances in Cryptology EUROCRYPT'97*, Konstanz, Germany, Lecture Notes in Computer Science 1233, pp. 226–238, Springer-Verlag, 1997.
12. R. Housley, D. Whiting, N. Ferguson. Alternate Temporal Key Hash. IEEE doc. 802.11-02/282r2. 2002.
13. D. Hulton. Practical Exploitation of RC4 Weaknesses in WEP Environments. 2001.
http://www.dachb0den.com/projects/bsd-airtools/wepexp.txt
14. R. Jenkins. ISAAC and RC4. 1996.
http://burtleburtle.net/bob/rand/isaac.html
15. P. Junod, S. Vaudenay. Optimal Key Ranking Procedures in a Statistical Cryptanalysis. In *Fast Software Encryption'03*, Lund, Sweden, Lecture Notes in Computer Science 2887, pp. 235–246, Springer-Verlag, 2003.
16. A. Klein. Attacks on the RC4 Stream Cipher. *Design, Codes, and Cryptography*, vol. 48, pp. 269–286, 2008.
17. L. R. Knudsen, W. Meier, B. Preneel, V. Rijmen, S. Verdoolaege. Analysis Methods for (Alleged) RC4. In *Advances in Cryptology ASIACRYPT'98*, Beijing, China, Lecture Notes in Computer Science 1514, pp. 327–341, Springer-Verlag, 1998.

18. Korek. Next Generation of WEP Attacks? 2004. http://www.netstumbler.org/showpost.php?p=93942&postcount=%35
19. Korek. Need Security Pointers. 2004. http://www.netstumbler.org/showthread.php?postid=89036#pos%t89036
20. S. Maitra, G. Paul. New Form of Permutation Bias and Secret Key Leakage in Keystream Bytes of RC4. In *Fast Software Encryption'08*, Lausanne, Lecture Notes in Computer Science 5086, pp. 253–269, Springer-Verlag, 2008.
21. I. Mantin. Analysis of the Stream Cipher RC4. 2001. http://www.wisdom.weizmann.ac.il/∼itsik/RC4/rc4.html
22. I. Mantin. Predicting and Distinguishing Attacks on RC4 Keystream Generator. In *Advances in Cryptology EUROCRYPT'05*, Aarhus, Denmark, Lecture Notes in Computer Science 3494, pp. 491–506, Springer-Verlag, 2005.
23. I. Mantin, A. Shamir. A Practical Attack on Broadcast RC4. In *Fast Software Encryption'01*, Yokohama, Japan, Lecture Notes in Computer Science 2355, pp. 152–164, Springer-Verlag, 2002.
24. A. Maximov. Two Linear Distinguishing Attacks on VMPC and RC4A and Weakness. In *Fast Software Encryption'05*, Paris, France, Lecture Notes in Computer Science 3557, pp. 342–358, Springer-Verlag, 2005.
25. A. Maximov, D. Khovratovich. New State Recovery Attack on RC4. In *Advances in Cryptology CRYPTO'08*, Santa Barbara, California, U.S.A., Lecture Notes in Computer Science 5157, pp. 297–316, Springer-Verlag, 2008.
26. I. Mironov. Not So Random Shuffles of RC4. In *Advances in Cryptology CRYPTO'02*, Santa Barbara, California, U.S.A., Lecture Notes in Computer Science 2442, pp. 304–319, Springer-Verlag, 2002.
27. V. Moen, H. Raddum, K. J. Hole. Weaknesses in the Temporal Key Hash of WPA. *Mobile Computing and Communications Review*, vol. 8, pp. 76–83, 2004.
28. G. Paul, S. Maitra. Permutation After RC4 Key Scheduling Reveals the Secret. In *Selected Areas in Cryptography'07*, Ottawa, Ontario, Canada, Lecture Notes in Computer Science 4876, pp. 360–377, Springer-Verlag, 2007.
29. S. Paul, B. Preneel. A New Weakness in the RC4 Keystream Generator and an Approach. In *Fast Software Encryption'04*, Delhi, India, Lecture Notes in Computer Science 3017, pp. 245–259, Springer-Verlag, 2004.
30. G. Paul, S. Rathi, S. Maitra. On Non-Negligible Bias of the First Output Byte of RC4 towards the First Three Bytes of the Secret Key. *Design, Codes, and Cryptography*, vol. 49, pp. 123–134, 2008.
31. J. Postel, J. Reynolds. A Standard for the Transmission of IP Datagrams over IEEE 802 Networks. RFC 1042. 1988.
32. A. Roos. A Class of Weak Keys in RC4 Stream Cipher (sci.crypt). 1995. http://groups.google.com/group/sci.crypt.research/msg/078a%a9249d76eacc?dmode=source
33. P. Sepehrdad, S. Vaudenay, M. Vuagnoux. Discovery and Exploitation of New Biases in RC4. To appear in the proceedings of SAC'10.
34. E. Tews, M. Beck. Practical Attacks Against WEP and WPA. In *Proceedings of the Second ACM Conference on Wireless Network Security WISEC'09*, Zurich, Switzerland, pp. 79–86, ACM, 2009.
35. E. Tews, R.-P. Weinmann, A. Pyshkin. Breaking 104 Bit WEP in Less Than 60 Seconds. In *Information Security Applications, 8th International Workshop, WISA'07*, Jeju, Korea, Lecture Notes in Computer Science 4867, pp. 188–202, Springer-Verlag, 2007.
36. V. Tomasevic, S. Bojanic, O. Nieto-Taladriz. Finding an Internal State of RC4 Stream Cipher. *Information Sciences: an International Journal*, vol. 177, pp. 1715–1727, 2007.

37. S. Vaudenay, M. Vuagnoux. Passive-only Key Recovery Attacks on RC4. In *Selected Areas in Cryptography'07*, Ottawa, Ontario, Canada, Lecture Notes in Computer Science 4876, pp. 344–359, Springer-Verlag, 2007.
38. D. Wagner. Weak Keys in RC4 (sci.crypt). 1995.
http://www.cs.berkeley.edu/∼daw/my-posts/my-rc4-weak-keys

A Computation of Biases

Biases were computed using the following formulas:

$$\begin{aligned}
P_{\mathsf{KI}}(i,t) &= P_J P_C(i,t) + \frac{1-P_J}{N-1}\left(1-P_C(i,t)\right) \\
P_{\mathsf{MP}}(i,t) &= P_D(i) P_B(i,t) P_0 \left(1-\frac{1}{N}\right) + \frac{1}{N} \\
P_{008}(i,t) &= P_8 P_C(i,t) + \frac{1-P_8}{N-1}\left(1-P_C(i,t)\right) \\
P_{009}(i,t) &= P_9 P_C(i,t) + \frac{1-P_9}{N-1}\left(1-P_C(i,t)\right) \\
\mathsf{Kor}_c^b(i,t) &= r_c(t) P_E^b(i,t) + \frac{1-r_c(t)}{N-1}\left(1-P_E^b(i,t)\right) \\
P_{\mathsf{SVV10}}(i,t) &= P_{\mathsf{db}} P_C(i,t) + \frac{1-P_{\mathsf{db}}}{N-1}\left(1-P_C(i,t)\right)
\end{aligned}$$

where $P_J = \frac{2}{N}$, $P_0 = \left(\frac{N-1}{N}\right)^{N-2}$, $P_8 = \frac{1.05}{N}$, $P_9 = \frac{1.0338}{N}$, $P_{\mathsf{db}} = 0.038488$,

$$\begin{array}{rlrl}
P_A^b(i,t) &= \left(\frac{N-b}{N}\right)^{i-t-1} & r_1(t) &= \left(\frac{N-1}{N}\right)^{N-t} \\
P_B(i,t) &= \prod_{k=1}^{i-t-1} \frac{N-k}{N} & r_2(t) &= \left(\frac{N-2}{N}\right)^{N-t-1} \\
P_C(i,t) &= P_A^1(i,t) P_B(i,t) P_0 \left(1-\frac{1}{N}\right) + \frac{1}{N} & r_3(t) &= \left(\frac{N-2}{N}\right)\left(\frac{N-3}{N}\right)^{N-t-1} \\
P_D(i) &= \frac{(N-i-1)(N-i)}{N^3}\left(\frac{N-2}{N}\right)^{N-3+i}\left(\frac{N-1}{N}\right)^3 & r_4(t) &= \left(\frac{N-1}{N}\right)\left(\frac{N-2}{N}\right)^{N-t-1} \\
P_E^b(i,t) &= P_A^b(i,t) P_B(i,t)\left(1-\frac{1}{N}\right) + \frac{1}{N} & r_5(t) &= \left(\frac{N-4}{N}\right)^{N-t}
\end{array}$$

These formulas are new. Biases were originally provided with probabilities for $t = -1$. Except for the Korek biases, we have checked that the probabilities match with an error less than 4%. The accuracy of formulas for Korek biases are still unclear but orders of magnitude are correct. They were inspired by [6]. Details on how we have got all formulas are omitted due to lack of space.

Table 5. Conditional Biases for RC4

If $\bar{g}_i$ holds then $\bar{K}[i] = \bar{f}_i$ with probability p_i. All biases except SVV_db are from [18]. SVV_db is from [33].

row	reference	$\bar{f}$	$\bar{g}$	p
i	A_u15	$2 - \sigma_i$	$z_2 = 0,\ S_t[i] = 0$	Kor_1^0
i	A_s13	$S_t^{-1}[0] - \sigma_i$	$S_t[1] = i,\ z_1 = i$	Kor_4^1
i	A_u13_1	$S_t^{-1}[z_1] - \sigma_i$	$S_t[1] = i,\ z_1 = 1 - i$	Kor_2^1
i	A_u13_2	$1 - \sigma_i$	$S_t[i] = i,\ S_t[1] = 0,\ z_1 = i$	Kor_2^0
i	A_u13_3	$1 - \sigma_i$	$S_t[i] = i,\ S_t[1] = 1 - i,\ z_1 = S_t[1]$	Kor_2^0
i	A_s5_1	$S_t^{-1}[z_1] - \sigma_i$	$S_t[1] < i,\ S_t[1] + S_t[S_t[1]] = i,\ z_1 \neq S_t[1],\ z_1 \neq S_t[S_t[1]],\ S_t[1] \neq 1$	Kor_3^1
i	A_s5_2	$S_t^{-1}[S_t[1] - S_t[2]] - \sigma_i$	$S_t[1] > i,\ S_t[2] + S_t[1] = i,\ S_t[1] = z_2,\ S_t^{-1}[S_t[1] - S_t[2]] \neq 1,\ S_t^{-1}[S_t[1] - S_t[2]] \neq 2$	Kor_3^3
i	A_s5_3	$S_t^{-1}[2 - S_t[2]] - \sigma_i$	$S_t[1] > i,\ S_t[2] + S_t[1] = i,\ z_2 = 2 - S_t[2],\ S_t^{-1}[z_2] \neq 1, S_t^{-1}[z_2] \neq 2$	Kor_3^2
i	A_u5_1	$S_t^{-1}[S_t^{-1}[z_1] - i] - \sigma_i$	$S_t[1] = i,\ z_1 \neq i,\ S_t^{-1}[z_1] < i,\ S_t^{-1}[S_t^{-1}[z_1] - i] \neq 1,\ z_1 \neq 1 - i$	Kor_3^2
i	A_u5_2	$1 - \sigma_i$	$S_t^{-1}[z_1] = 2,\ S_t[i] = 1$	Kor_2^0
i	A_u5_3	$1 - \sigma_i$	$S_t[1] > -i,\ S_t[i] = i,\ S_t[1] = S_t^{-1}[z_1] - i,\ S_t^{-1}[z_1] \neq 1$	Kor_3^0
$i > 4$	A_u5_4	$S_t^{-1}[z_2] - \sigma_i$	$S_t[1] = 2,\ S_t[4] + 2 = i,\ S_t^{-1}[z_2] \neq 1,\ S_t^{-1}[z_2] \neq 4$	Kor_3^1
i	A_s3	$S_t^{-1}[z_2] - \sigma_i$	$S_t[1] \neq 2,\ S_t[2] \neq 0,\ S_t[2] + S_t[1] < i,\ S_t[2] + S_t[S_t[2] + S_t[1]] = i,\ S_t^{-1}[z_2] \neq 1,\ S_t^{-1}[z_2] \neq 2,\ S_t^{-1}[z_2] \neq S_t[1] + S_t[2]$	Kor_5^1
4	A_4_s13	$S_t^{-1}[0] - \sigma_4$	$S_t[1] = 2,\ z_2 = 0,\ S_t[4] \neq 0$	Kor_2^1
4	A_4_u5_1	$S_t^{-1}[N - 2] - \sigma_4$	$S_t[1] = 2,\ z_2 \neq 0,\ S_t^{-1}[z_2] = 0$	Kor_3^1
4	A_4_u5_2	$S_t^{-1}[N - 1] - \sigma_4$	$S_t[1] = 2,\ z_2 \neq 0,\ S_t^{-1}[z_2] = 2,\ K[0] + K[1] + S_0[1] = 2$	Kor_3^1
i	A_neg_1a	$1 - \sigma_i$	$S_t[2] = 0,\ S_t[1] = 2,\ z_1 = 2$	0
i	A_neg_1b	$2 - \sigma_i$	$S_t[2] = 0,\ S_t[1] = 2,\ z_1 = 2$	0
i	A_neg_2	$2 - \sigma_i$	$S_t[2] = 0,\ S_t[1] \neq 2,\ z_2 = 0$	0
i	A_neg_3a	$1 - \sigma_i$	$S_t[1] = 1,\ z_1 = S_t[2]$	0
i	A_neg_3b	$2 - \sigma_i$	$S_t[1] = 1,\ z_1 = S_t[2]$	0
i	A_neg_4a	$-\sigma_i$	$S_t[1] = 0,\ S_t[0] = 1,\ z_1 = 1$	0
i	A_neg_4b	$1 - \sigma_i$	$S_t[1] = 0,\ S_t[0] = 1,\ z_1 = 1$	0
16	SVV_db	$S_t^{-1}[0] - \sigma_i$	$z_i = -16$	$P_{\mathsf{SVV10}}(i,t)$