Don’t Try This at Home: A Layman’s Guide to Security Vulnerabilities of the Original 802.11 Standard

Dmitri “Dima” Varsanofiev
cooltech[AT]varsanofiev.com
www.varsanofiev.com/inside/802.11security.ppt

Key Points

- IANAC (I Am Not A Cryptographer)
  - All content of the presentation is work of other people (reference list is at the end)
- Just like the electrical grid changes at home, cryptography design is better left to professionals
  - At the very least, call an inspector afterwards - but before powering up the circuit!
  - Wisdom ignored by the original 802.11 group
- Professionals have already fixed the problems discussed here
  - Buy **WPA** (802.11i) gear

802.11 Security Story

- Basic 802.11 (1999) defines "wireless equivalent privacy" (WEP)
  - Uses RC4 cipher
  - Multiple holes found; group alerted by Jesse Walker
- WEP holes fixed in so called TKIP scheme
- A clean RSN encryption scheme is introduced that uses new AES cipher
- TKIP and RSN are defined in the 802.11i standard
- Industry body (WiFi) defined WPA security standard that mostly matches the 802.11i
- 802.11i and WPA are outside of scope of this talk

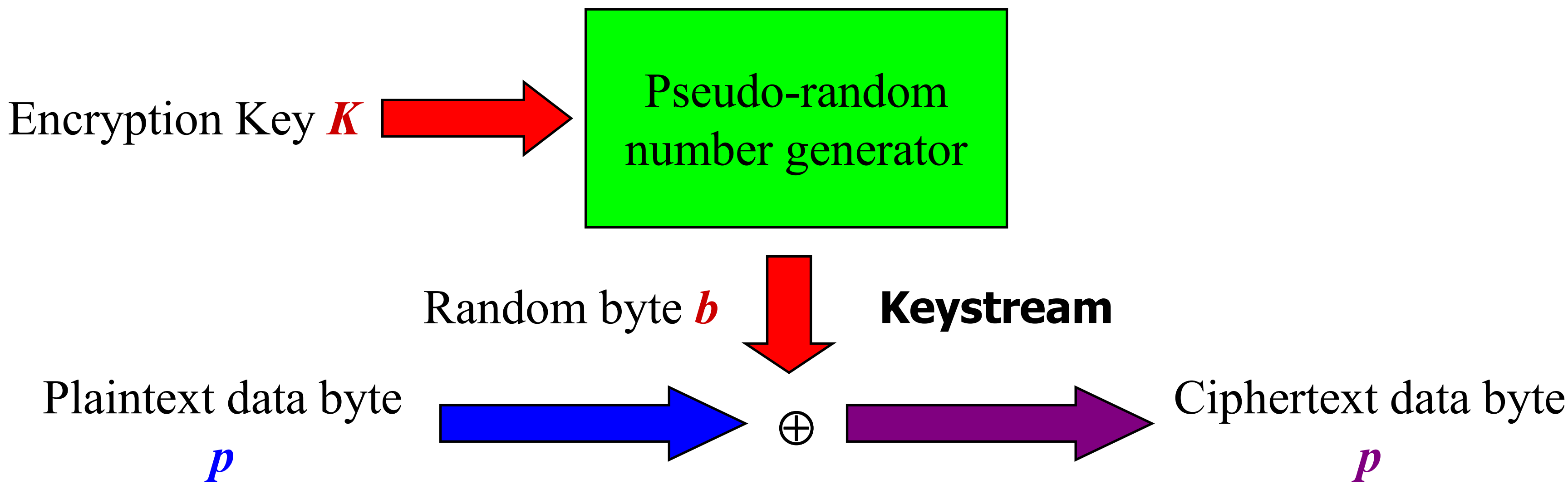
Basic Crypto Terminology

- Encryption turns Plaintext into Ciphertext using a secret Key
  - Decryption is the inverse transform
  - If the key is the same for encryption and decryption (“shared secret”, known to both sides), the algorithm is “symmetric”
- Authentication provides a way to check that the message has not been altered
  - Usually in the form of a cryptographic checksum - Message Authentication Code (MAC, called MIC in the 802.11i)

Basic Terminology Continued

- Crypto algorithm usually consists of a Cipher and Mode
  - Cipher is a way to randomize data. Examples: DES, AES, RC4, …
  - Mode is a way to apply the cipher to the plaintext
  - Simplest mode is Electronic Code Book, or ECB: plaintext is XORed with pseudo-random “keystream”

ECB Mode or “Vernam Cipher”



Decryption works the same way: $p = c \oplus b$

From an original slide by Jesse Walker

Application of ECB to Packet Networks

- Packets can be lost
  - Cipher has to be restarted for each packet
- Re-use of keystream is extremely bad!
  - If we can guess content of one packet, we can recover the keystream – and read all the packets with the same keystream
  - Simply XORing two packets with the same keystream yields a lot of information about the data
- Confucius says, "It is better to transmit packet unencrypted than to reuse the keystream"
  - In the former case, only the data in this packet is divulged
  - In the latter case, information about keys is divulged – this and a lot of other packets can be decrypted!

Initialization vector

- Unique data added to each packet and used to initialize pseudo-random generator to make the keystream unique
  - Called “initialization vector” (IV), “salt”
  - Must not be reused with the same key!
- To avoid the IV reuse certain mechanisms should be included into crypto exchange
  - Some way to avoid using the same IV with the same key
  - Some way to force a key change once IV space is exhausted

IV Flaw in WEP

- Key is shared by all stations in the network
  - No standard way to synchronously change the key
- IV contains only 24 bits
  - No IV synchronization techniques are defined across multiple stations
- Due to the "birthday paradox", with randomly chosen IVs, a "collision" (IV reuse) will very probably happen after about $O(2^{12}) = 5000$ packets
  - Many implementations use same IV sequence (usually 0, 1, 2, …) for all stations thus providing even more collisions ☹

More Basics: Replay Attack

- A replay attack is simply a re-transmission of the previously recorded message later in an attempt to harm the recipient
- How can it harm?
  - By divulging information about the system being attacked. For example, one can find out if some PC is powered on by re-sending messages to it and seeing if a response still comes
  - By exploiting the replay in a combination with some other vulnerability. For example, if A has established a secure channel with B, and we can replay to B A's disconnect message, we might have a chance of attack against A pretending to be B. For more spectacular opportunities, see WEP examples later.
- Typical prevention of this attack is a protected (encrypted or checksummed) timestamp of some kind

Flaw in the Join Exchange

- To join the 802.11 network, station has to "authenticate" itself
  - Open authentication
    - Station sends a message "I am a good guy"
    - Access point replies, "Yeah, you are a good guy" ☺
    - Think it is bad?
  - "Shared key" authentication is worse!
    - Station sends a message, "I am a good guy"
    - Access point replies, "Show me that you know the key – here is 128 bytes for you to encrypt"
    - Station sends an encrypted copy
    - Access point compares and says, "You are a good guy"
- What is wrong in this picture?

Keystream is easy to get

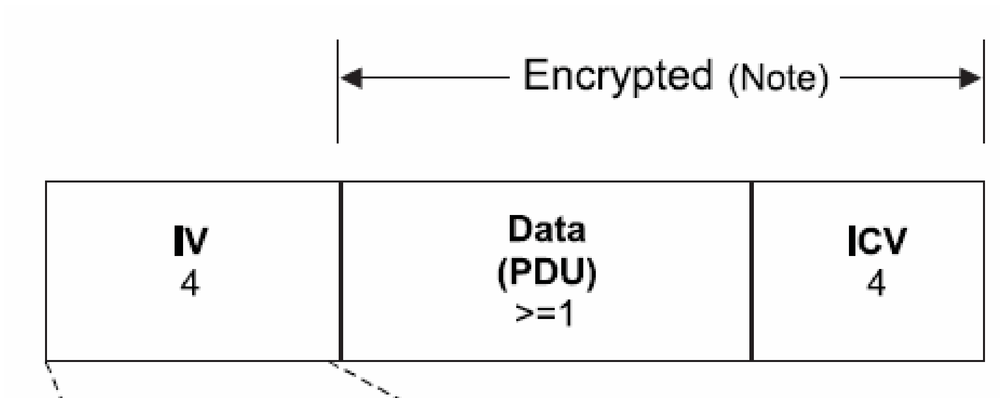
- If an attacker can listen on to this exchange, (s)he will see:
  - Plaintext in the second message
  - Corresponding ciphertext in the third
- As we know, in WEP ciphertext = plaintext XOR keystream
  - But … ciphertext XOR plaintext = keystream!
- We just got a way to encrypt some short frame and inject it into the system
  - Helps as a basis for inductive attacks (detailed below)
  - Useful for network study
    - Send a frame that will cause a reply (say, ARP), listen to reply
    - Replies are predictable = more keystreams
- **Never** enable the "shared key" authentication
  - Not useful for its original goal – as with one known keystream an attacker can always successfully authenticate anyhow ☺
  - On-demand keystream by spoofing the access point

Replay Flaw in WEP

- Original 802.11 only carried a timestamp as a 12-bit frame number in the unencrypted header
  - Not protected
  - Wide open for replay
- Combined with a weak design of the cryptographic checksum (next flaw), opens WEP up for elegant attacks

Crypto Checksum

- Encrypted frame in the original 802.11 WEP:



- Why do we need a checksum (ICV)?
  - To prevent bit-flipping attacks

More Basics: Bit-flipping

- Take an encrypted message, and change one bit. With ECB, decryption of this message will yield plaintext with the corresponding bit changed
- If an attacker knows an IP address in the frame, (s)he can change the address to his own, replay the frame, and receive a decrypted message from an AP over the wire ☺
- Prevented by using a crypto checksum
  - Checksum changes when bit changes
  - On the receiving side, if the checksum does not match, message should be discarded
  - Attacker now needs to change the desired bit(s) and the checksum bits simultaneously
  - With many bits to change simultaneously, attack becomes impractical
  - Only works if the checksum bit changes are not predictable

Weak Checksum in WEP

- WEP uses CRC-32 for crypto checksum
  - Perfectly good for data integrity
  - Unacceptable for cryptography
- CRC-32 is linear
  - CRC (X^Y) = CRC (X) ^ CRC (Y)
- If an attacker flips the bits in the message, it is easy to flip the corresponding bits in the checksum
  - NewICV = OldICV ^ ICV(modified bits)
  - Carries through the encryption stage
  - TCP checksum can be guessed with high probability
- Flip the address bits in the IP packet and receive it decrypted – No Key Knowledge Necessary ™

Inductive Attack (Bill Arbaugh)

- Take a short message with known keystream that solicits a response from the system
  - Basis is provided by the Join sequence flaw
  - Or look for a short message with a known plaintext (say, a DHCP request)
- Insert a byte into it just before the ICV
- Try all 256 byte values, adjust the ICV accordingly
  - One will yield a response
- The known keystream is extended by one more byte
- Repeat

Ultimate Flaw (Shamir et al.)

- Ultimate defeat of the encryption is to recover the key in a reasonable amount of time
- WEP uses RC4 as a cipher
- Mantin and Shamir in 2001 figured out a way to easily distinguish between the RC4-encrypted data stream and a random data stream
- Confucius says, “If your cipher is a bad random number generator, it will be broken soon”
- In few months, multiple effective attacks on RC4 were suggested
- The simplest one will be explained (touched?) here, based on an article by Fluhrer, Mantin, Shamir
  - Grossly simplified for presentation reasons. Actual attacks are similar in nature, but are significantly more complex

RC4

- Popular cipher still widely used
  - Invented by Rivest in 1987, trade secret of RSA
  - Leaked onto Internet in 1994 as “Alleged RC4”, or ARC4
- Convenient for implementation on a CPU
  - Only byte-level operations are used, no bit shuffling
- For randomization uses an internal “state” array S of size N and two indices into this array, i and j (sometimes referred to as x and y)
  - S is initially filled with a randomization of the key K of length L during so called “Key Scheduling Algorithm” (KSA)
  - S is updated during the random number generation phse (PRGA)

Details of the RC4

KSA(K)	PRGA(K)
Initialization:	Initialization:
For $i = 0 \ldots N-1$	$i = 0$
$S[i] = i$	$j = 0$
$j = 0$	Generation loop:
Scrambling:	$i = i + 1$
For $i = 0 \ldots N-1$	$j = j + S[i]$
$j = j + S[i] + K[i \bmod \ell]$	$\mathrm{Swap}(S[i], S[j])$
$Swap(S[i], S[j])$	Output $z = S[S[i] + S[j]]$

From an article by Fluhrer, Mantin, Shamir

Attack Assumptions

- First bytes of plaintext are known
  - True for 802.11, as practically all encrypted frames are prefixed by an “LLC header” that consists of known bytes
- Part of the key is known, so we can look only at the “easy” packets
  - True for 802.11 WEP, as the RC4 key (K) in WEP is made by concatenating the 3-byte IV and the secret key – and all the IV bits are known
  - For simplicity, we will assume that IV occupies first three bytes of K (in the actual WEP, the IV is appended to the key, making attack explanation way harder)
- One can observe multiple packets with different “easy” keys
  - True for 802.11 due to its wireless nature and varying IVs

Attack Idea

- First byte to be output (look at the PRGA):
  - S[S[1]+S[S[1]]]
- What would the S values be?
  - Let’s suppose we know A bytes of the secret key (initially A=0, of course)
  - Wait for the packets with IV of the values (A+3, N-1, X) with a variety of Xs
  - We can predict key setup behavior until step A+3
    - Or learn that it is became randomized (“resolved”)

Key Expansion

For $i = 0 \ldots N-1$

$j = j + S[i] + K[i \bmod \ell]$

$Swap(S[i], S[j])$

$A+3$	$N-1$	X	$K[3]$		$K[A+3]$	
0	1	2			$A+3$	
$A+3$	1	2			0	
i_0					j_0	

$A+3$	$N-1$	X	$K[3]$		$K[A+3]$	
0	1	2			$A+3$	
$A+3$	0	2			1	
	i_1				j_1	

$A+3$	$N-1$	X	$K[3]$		$K[A+3]$	
0	1	2			$A+3$	
$A+3$	0	$S[2]$			$S[j]$	
					i_{A+3}	

Getting the Key

- If the subsequent key setup steps will change any of the first values of S, we will be in a “resolved” situation with near-random output
- However, in the 5% cases these values will not be touched, so:
  - S[1] = 0, S[S[1]] = A+3
- The output, S[S[1]+S[S[1]]] = K[A+3]
  - Bingo – we have just learned an extra key byte!
- If we observe enough frames with this type of IV, we will see a peak in distribution
  - 60 frames give 50% probability
  - Note that things are not **that** bad – these are special 60 frames!
- In practice, few million frames are sufficient to crack the 40-bit WEP key
  - 128-bit key does not buy much extra protection – as the process has *linear complexity*

Practical Consequences

- Security of the "original" 802.11 (as defined in 1999) is not too bad if nobody is after you
- It is disastrous if somebody does want to get you!
  - Resources required to breach the security are freely available to an amateur
  - Time commitment is in minutes to hours
- OK for home use?
  - Protects from casual or accidental browsing of non-computer-savvy neighbors
  - In my home, connects to the Internet side only, not the internal network
- Unacceptable for large businesses
  - Use WPA (802.11i) gear

References

- Full bibliography is available at Bill Arbaugh's site: http://www.cs.umd.edu/~waa/wireless.html
- 802.11 study by Jesse Walker (2000), Unsafe at any key size; An Analysis of the WEP encapsulation (link embedded)
- Bill Arbaugh's inductive attack (2001), http://www.cs.umd.edu/~waa/attack/v3dcmnt.htm
- Key recovery by Scott Fluhrer, Itsik Mantin, and Adi Shamir http://www.drizzle.com/~aboba/IEEE/rc4_ksaproc.pdf