Brief yet Scintillating Article about WEP and WPA2

Ashley Oudenne
The University of Texas at Austin
aoudenne@cs.utexas.edu

Jerremy Adams
The University of Texas at Austin
ja7872@cs.utexas.edu

December 16, 2012

Abstract

Interesting vague things about our paper. 200 words or less, and should say something like: In this paper, we present the WEP and WPA Four-Way Handshake protocols in detail and formally model them using the ProVerif Cryptographic Verifier.

1 Introduction

Wireless communication is a staple of 21st century daily life, and security is a serious concern for individuals, businesses, and organizations looking to communicate quickly and securely. To ensure this, different standards have been proposed that define protocols for secure communication.

One such standard was the 802.11 standard ratified in September of 1999, which intended to provide data confidentiality through Wired Equivalent Privacy, or WEP [1]. WEP uses a 40- or 104-bit encryption key that is manually entered into access points and devices. This key never changes, so if it is compromised, all future messages on the network are compromised until every device is manually rekeyed. The intention of WEP was to provide the same level of confidentiality as that of a traditional wired network.

Unfortunately, WEP relies on the RC4 stream cipher, which was thought to be secure but which is actually vulnerable to attacks. An attacker can recover plaintexts of messages and send fake messages if the same initialization vector is ever used in RC4. Additionally, the Cyclic Redundancy Check (CRC) checksum algorithm used by WEP does not provide a strong enough integrity guarantee, because it permits the guessing of individual bytes of a packet [8]. Since the CRC is simply a linear function of the message, an attacker can modify an encrypted message and fix the checksum so that the message appears not to have been modified.

To remedy this, IEEE released WPA in 2003 as a temporary remedy for WEP until a new standard could be ratified. WPA was designed to work on devices that were currently using WEP until new hardware was created, so it also uses the RC4 cipher and the CRC checksum mechanism [2]. However, it adds the Temporal Key Integrity Protocol, which implements a key mixing function that combines the root key with an initialization vector *before* passing it to RC4, instead of just concatenating the initialization vector and the root key as in WEP. This prevents repeated keystream attacks, to which WEP is

susceptible. TKIP also enforces rekeying and sequence counters to thwart replay attacks. It also includes an additional message integrity check called Michael that increases security.

Although WPA provides greater security than WEP, it is still susceptible to some of the same attacks as WEP, because of the insecurity of the RC4 cipher and the CRC checksum algorithm. As a result, the 802.11i standard was ratified in June of 2004 to replace the use of the TKIP protocol (which uses RC4) with AES-based CCMP encryption[3]. This provides strong security for key generation.

Both WPA and WPA2 rely on the computation of a secure key to encrypt data. To avoid the insecurity of WEP (in which data sent over the network is a direct function of the secret key), both parties begin with the same Pairwise Master Key (PMK) and use this key to compute the Pairwise Transient Key (PTK) which is actually used to encrypt data. Neither of these keys are ever sent over the network. Instead, the wireless access point and the supplicant station engage in the Four-Way Handshake protocol in order to transmit the data necessary to calculate the PTK.

While the Four-Way Handshake is immune to most attacks, it is vulnerable to a Denial-of-Service (DOS) attack [10]. This attack prevents the station and the access point from ever fully authenticating with one another. While an attack of this nature is not particularly devastating, in that it does not leak keys or permit the attacker to corrupt data, the availability of a network should not be influenced by an attacker.

In this paper, we present the WEP and WPA Four-Way Handshake protocols in detail and formally model them using the ProVerif Cryptographic Verifier. In Section 2, we present work by other researchers on the security of WEP and 802.11i. We particularly focus on the security of the Four-Way Handshake protocol. We explain the details of the protocols in Section 3. We also describe ProVerif and our attacker model. In Section 4, we explain how we model the protocols in ProVerif. We show the possible attacks on the WEP protocol and what particular aspects of the protocol lead to these attacks. We will then explain how these aspects have been eliminated in WPA/WPA2 due to the Four-Way Handshake Protocol, and then demonstrate how a Denial-Of-Service attack is still possible. Finally, in Section 5, we summarize our conclusions and suggest future work that could be done on proving the correctness of wireless security.

2 Related Work

There has been much work, both with automatic verifiers and without, on proving the insecurity of WEP. In [7], a large number of insecurities in WEP are discussed without the aid of an automatic verifier. The authors highlight possible attacks resulting from the risk of keystream reuse, due to the fact that encrypting two messages under the same keystream can reveal information about both messages. Since the initialization vectors used to compute the keystreams are re-initialized every time a wireless card is reinserted into a device, there are many opportunities for an attack of this nature.

The authors also discuss the risk of message modification in WEP due to the CRC checksum being a linear function of the message, which means that it distributes over XOR. An attacker can arbitrarily modify even messages he hasn't decrypted by simply XORing the ciphertext with some bitstream and the checksum of the bitstream. Messages can be injected into a

network because CRC is not dependent on the keystream used to encode a message. Once an attacker learns an initialization vector and its corresponding keystream, the keystream can be reused indefinitely to insert new messages into the network, because initialization vectors are never checked for freshness. This also allows an attacker to authenticate himself to the network.

The security of 802.11i with respect to confidentialty and authentication has also been verified without the use of automatic verifiers. In [10], He and Mitchell consider each stage of the protocol and argue its security from a number of possible threats, including malicious access points, session hijacking, eavesdropping, and message deletion. They conclude that it provides effective confidentiality and integrity when the CCMP protocol is used, and that it may provide satisfactory mutual authentication and key management. However, they identify several possible Denial of Service attacks, since the protocol is not designed to ensure liveness.

One of these Denial of Service attacks, identified in [9] and [13], deals with the Four-Way Handshake protocol that is responsible for establishing the Pairwise Transient Key (PTK). Using automatic verification tools, both groups of researchers prove that it is possible to launch a Denial of Service attack on a supplicant in which the supplicant is continually forced to regenerate a new but incorrect PTK, preventing it from ever communicating with the server. [9] solves this problem by suggesting the reuse of the supplicant's nonce until after the PTK has been established. However, this solution is susceptible to a Denial of Service attack against CPU resources. [13] proposes a different solution intended to avoid this. They suggest the addition of a large random number nonce that is sent along with traditional participant's nonce. However, both of these nonces are encrypted with the Pairwise Master Key, which is assumed to be unknown to the attacker. This ensures that attackers cannot flood the network with nonces in an attempt to trick the supplicant into continuously recomputing the PTK.

Automatic verification has also been applied to WEP. Lafourcade et al. found an attack similar to [7] described above using the ProVerif Cryptographic Verifier [12]. By placing multiple messages on the channel encrypted with the same keystream, the attacker is able to recover the contents of encrypted messages. To prevent this attack, they then implemented a version of WEP in which all intialization vectors were guaranteed to be unique. This protocol was considered secure by ProVerif. Unfortunately it is impossible to ensure that initialization vectors will always be unique in the real world, which is why WEP is no longer considered to be secure.

3 Modeling of Protocols

3.1 WEP Protocol Description

WEP, described in [7], uses a one-message protocol to transmit data between two parties that relies on a secret key k that has previously been shared between them. The intention of this protocol is to provide the same confidentiality as that of a wired network [1]. Before a message is sent, an integrity checksum $C(m)$ is computed on the message so that the recipient can verify that the message has not been altered in transit. The message is concatenated to this checksum to form the plaintext P.

The plaintext is now encrypted using the RC4 cipher. The sender chooses an initialization vector IV and uses this vector along with the secret key k to generate an arbitrary-length sequence

of pseudorandom bits known as a keystream [14]. The sender then uses exclusive-or (or XOR, denoted by $\oplus$) to XOR P and the keystream to generate ciphertext C. The complete encryption process can be represented by:

$$C = P \oplus RC4(IV, k)$$

The message, which consists of the initialization vector and the ciphertext, is then ready to be transmitted from the sender to the receiver. We will represent this transmission symbolically as:

$$A \rightarrow B : IV, (P \oplus RC4(IV, k)),$$

$$\text{where} P = \langle m, C(m) \rangle$$

Notice that the initialization vector IV is sent in the clear, meaning that anyone can read the value of the initialization vector. This should not matter because an attacker would need both the IV and the secret key k to recover the keystream used to decrypt the message, but we will show later in this paper that this is enough to break WEP.

Decryption works by reversing the encryption process described above. The recipient first regenerates the keystream used to encode the message with the secret key k and the IV sent along with the encrypted message. He can XOR the ciphertext with the keystream to recover the plaintext P:

$$\begin{aligned} P &= C \oplus RC4(IV, k) \\ &= (P \oplus RC4(IV, k)) \oplus RC4(IV, k) \\ &= P \end{aligned}$$

The recipient can then verify the integrity of the message by splitting P into $\langle m, C(m) \rangle$ and recomputing the checksum of m. If the computed checksum matches the sent checksum, then it was incorrectly assumed that the message has not been tampered with. However, as we discuss in Section 2, the checksum is not enough to provide integrity.

3.2 WPA/WPA2 Four-Way Handshake Protocol Description

The Four-Way Handshake protocol is used in both WPA and WPA2 to authenticate a station to an access point, to compute a pairwise transient key (PTK) to be used in future communication between these parties, and to distribute a group transient key (GTK) to be used by the station to communicate with other devices connected to the access point [13]. Once generated, the PTK is broken up into five different keys, but for the purposes of modeling the protocol it is sufficient to think of it as a single key. We can assume that both the station and the access point begin by knowing the Pairwise Master Key (PMK), which will be used to compute the PTK. The PMK is either computed by both parties in enterprise mode, or known ahead of time in pre-shared key mode (used for personal networks).

The Four-Way Handshake proceeds as follows. An access point and a wireless station each compute their own fresh random nonce, N_A and N_S, respectively. To begin the protocol, an access point sends a wireless station N_A in the clear with no guarantee of integrity. Once the station receives this nonce, it actually has all the information it needs to construct the PTK. It does so by concatenating five values: the PMK, the access point's nonce, it's own fresh random nonce, the MAC address of the access point, and the station's own MAC address. This concatenated value is then passed through a cryptographic hash function to derive the PTK.

Next, the station sends its nonce back to the

access point along with a Message Authentication and Integrity code (MAIC) created by running a MAC algorithm on the nonce using the PTK as the secret key. Once the access point has received the station's nonce and the corresponding MAIC, it can construct the PTK for itself. Once it has the PTK, the access point can run the same MAC algorithm on the station's nonce to ensure both parties have a consistent PTK. Notice that if the first message (the access point's nonce) was tampered with, the station and access point would not have a consistent PTK, and thus the integrity of the first message is guaranteed here. At this point, the access point can reason that the station is legitimate as its PTK was derived from the shared secret PMK.

To authenticate the access point to the station, the access point sends the GTK encrypted by the PTK along with another MAIC, also derived from the PTK. Upon receipt of this message by the station, the station can reason about the identity of the access point as above. That is, it can calculate the MAIC for the encrypted GTK and see that the access point is legitimate since it has constructed the PTK from the shared secret PMK.

Finally, the station merely sends back an acknowledgement to the access point, and both the access point and station install the keys for their communication session.

This protocol can be summarized symbolically as:

1. $AP \rightarrow STA : N_A$

   (STA calculates PTK)

2. $STA \rightarrow AP : N_S, MAIC(N_S)$

   (AP calculates PTK)

   (AP authenticates STA by verifying $MAIC(N_S)$)

3. $AP \rightarrow STA : \{GTK\}_{PTK}, MAIC(\{GTK\}_{PTK})$

   (STA authenticates AP by verifying $MAIC(\{GTK\}_{PTK})$)

4. $STA \rightarrow AP : ACK$

   (STA, AP install PTK for use in this session)

3.3 ProVerif

To formally prove the correctness of WEP and the Four-Way Handshake, we use the ProVerif Cryptographic Verifier created by Bruno Blanchet. ProVerif uses prolog rules to encode the protocol and abstracts away fresh values and the number of steps in the protocol [5]. It treats each fresh value as a function of other messages in the protocol, meaning that different values are used for each pair of protocol participants. Each step in the protocol can be completed any number of times, and past steps can be re-executed arbitrarily. This permits ProVerif to execute an unlimited number of runs of a protocol.

A protocol is proved to be secure with respect to some invariant by querying whether ProVerif can generate the inverse of that invariant. For example, a confidentiality invariant is proven by querying whether the attacker can learn a secret (e.g. a private key, or the contents of an encrypted message). Since we assume a Dolev-Yao attacker model, we assume that all messages on a channel c are owned by the attacker. The attacker can read, modify, generate, or delete any message he desires (though he cannot learn the contents of an encrypted message without the corresponding key). Since the attacker has unlimited access to all messages on c, an invariant is proved if the attacker cannot invert that invariant despite access to all the messages. This verifier has been used to successfully prove the insecurity of protocols such as the Diffe-Hellman

key exchange protocol, Initial Key Agreement, and the Needham-Schroeder symmetric-key protocol [12, 4].

3.3.1 Horn Clauses

ProVerif can take protocols in either horn clauses or pi-calculus. In our protocol implementations, we chose to use typed horn clauses. An untyped horn clause is a disjunction of literals with at most one positive literal (ex. $\neg p \vee \neg q \vee t$) [6]. They can be written as implications (ex. $(p \wedge q) \rightarrow t$), as they are in Prolog, on which ProVerif is based. Typed horn clauses merely allow the user to add a type system to the program for clarity and convenience, but the underlying logic resolution algorithm is the same as that of untyped horn clauses.

3.3.2 Attacker Model

In ProVerif, it is not necessary to explicitly model the attacker. Rather, the user constructs a list of clauses detailing what anyone, including the attacker, can do with messages that are put on the channel c. Such abilities include separating a message $c(a, b)$ into its component parts and placing those parts on the channel ($(c(a), c(b))$, decrypting messages if the encryption key of the message is known, encrypting messages using a known key, XOR-ing messages, and computing checksums of messages. We use the Dolev-Yao attacker model, in which the attacker can intercept, overhear, and create new messages, because it a very strong model. Since the attacker has complete control of the message, protecting a wireless protocol against a Dolev-Yao attacker should provide realistic security guarantees for real-world attackers.

4 Analysis of Protocol Models

4.1 ProVerif Model of WEP

In order to implement WEP in ProVerif, we first need to model the message that principal A sends to principal B. As described in section 3.1, A sends the message and its checksum ($\langle M, checksum(M)\rangle$) XORed with the keystream computed by using RC4 on a shared key K_{AB} and an initialization vector v.

```
forall p:principal;
    c(xor((m[p], checkSum(m[p])),
        rc4(v[], Kab[])));
```

We next identify an invariant that must hold true if the protocol is to be considered secure. Since an attacker can use XOR to modify messages and decrypt messages encrypted using the same keystream, we chose as one invariant the property that if an attacker XORs two different messages on the channel that were encrypted using the same keystream, then a secret has been learned. We then query ProVerif to determine whether the secret can ever be learned in all possible sequences of messages. Additionally, if the plaintext message m is ever put on the channel, there is a security breach. These two invariants (represented as secret[]) are represented in ProVerif by the following:

forall p:principal; c(m[p]) -¿ c(secret[]);

```
forall p:principal, q:principal;
    c(xor((m[p],checkSum(m[p])),
        (m[q],checkSum(m[q]))))
            & p <> q -> c(secret[]).
```

Finally, the attacker must have some knowledge of what to do with messages that are placed on the channel. For WEP, an attacker knows

how to encrypt and decrypt messages, XOR messages together, and compute the checksum of messages. Most importntly, according to the properties of XOR, given two encrypted messages which have the same keystream (because the IV was reused), the attacker can XOR the two messages together to remove the encryption on both of them. Though the attacker wouldn't yet have recovered the plaintext of a message, the resulting XOR would be quite vulnerable to statistical attacks or the exploitation of regular structure in messages. Please see Appendix A for the complete ProVerif implementation of WEP.

4.2 Attack on WEP

As [12] and [7] discovered, WEP is highly vulnerable to keystream reuse attacks. In our model, Proverif identified the following keystream reuse attack that allows the attacker to learn information about two messages he overhears on the channel (P represents the $\langle m, checksum(m)\rangle$ pair):

1. $A \rightarrow B : IV, (P_1 \oplus RC4(IV, k))$
2. $A \rightarrow B : IV, (P_2 \oplus RC4(IV, k))$
3. Attacker: $(P_1 \oplus RC4(IV, k)) \oplus (P_2 \oplus RC4(IV, k))$
4. Attacker: $P_1 \oplus P_2$

This is a very serious vulnerability. If the attacker knows either P_1 or P_2, he can recover the other message easily. Once a plaintext message is known, the keystream used to encrypt the message can be recovered by XORing the plaintext and the ciphertext together. Since WEP does nothing to prevent the reuse of old IVs, an attacker can now circumvent WEP and use the recovered keystream and IV indefinitely [7]. Clearly, this protocol is not secure enough for the transmission of data.

4.3 ProVerif Model of Four-Way Handshake

Implementing the Four-Way Handshake in ProVerif is more complicated than implementing WEP. We begin by representing the protocol as a series of messages. While the messages can be sent in any order, a message can't be sent until its predecessor message has been placed on the channel (ex. once message 2 is put on the channel, message 3 can then be put on the channel any number of times, in any order). As we describe in Section 3.2, the Four-Way handshake is a series of nonce exchanges (represented as nSTA and nAP for the station and the access point, respectively), verifications using a message authentication and integrity code (represented as maic()), and MAC addresses that are used to calculate the PTK (represented as makePTK()). In our implementation, we send the "MAC" address of the sender of each message (which we simply model as the principal's ID). We also identify each message using a wrapper (ex. msg1()) for convenience in ProVerif. The protocol is represented as follows:

```
1. forall p:principal; c(msg1(nAP[p],AP[]));

2. forall p:principal, n:bitstring;
     c(msg1(n,p))->c(msg2(nSTA[STA[],p],
       maic(nSTA[STA[], p],
         makePTK(n, nSTA[STA[],p],
           p, STA[], PMK[]))));

3. forall p:principal, n:bitstring;
     c(msg2(n,maic(n, makePTK(nAP[p],
       n, AP[],p, PMK[])))) ->
```

```
        c(msg3(encrypt(GTK[],
          makePTK(nAP[p],n,AP[],
            p,PMK[]))));
```

4.
```
forall p:principal, n:bitstring;
    c(msg3(encrypt(GTK[], makePTK(n,
      nSTA[STA[],p],p, STA[], PMK[]))))
        ->c(msg4(encrypt(nSTA[STA[],p],
            makePTK(n,nSTA[STA[],p],
              p,STA[],PMK[]))));
```

We then define invariants that must hold if this protocol is to be considered secure. Security is leaked if the PMK, GTK, or PTK is ever placed on the channel (this would represent the attacker being able to calculate these secret keys). Authentication is violated if message four is sent and key installation would then happen between any principals other than the station and the access point. Liveness is guaranteed as long as the station never generates inconsistent PTKs. If an attack is generated that causes the station to have inconsistent PTKs, as described in Section 2 and in [9], then the station will be unable to communicate with the access point. After a period of time, the access point will try to reinitiate the protocol with the station. The attacker can then repeat his attack, causing the station to again have inconsistent PTKs. This repeated process causes a Denial of Service attack. In our implementation of the Four-Way Handshake, we say that the liveness invariant is violated if a station's PTKs are inconsistant once, since if the attack can be mounted once, it can be mounted again to cause a Denial of Service attack. These three kinds of invariants (represented as dos[], authFail[], and secret[]) can be modeled as follows:

```
(* Liveness *)
forall p:principal, q:principal,
   nap:bitstring, nsta:bitstring,
      natt:bitstring; c(maic(nsta,
         makePTK(nap,nsta,p,q,PMK[])))  &
            c(maic(nsta,makePTK(natt,nsta,
               p, q,PMK[]))) &
                  nap <> natt -> c(dos[]);

(* Confidentiality *)
c(PMK[]) -> c(secret[]);

forall p:principal, q:principal,
   n1:bitstring, n2:bitstring;
      c(makePTK(n1, n2, p, q, PMK[]))
         -> c(secret[]);

c(GTK[]) -> c(secret[]);

(* Authentication *)
forall p:principal, q:principal,
   n1:bitstring, n2:bitstring;
      c(msg4(encrypt(nSTA[p,q],
         makePTK(n1,n2,p,q,PMK[]))))
            & p <> AP[] &
               q <> STA[] -> c(authFail[]).
```

We define the capabilities of the attacker to be those of the Dolev-Yao attacker model. For the Four-Way Handshake, the attacker can generate fake bitstring, encrypt and decrypt the GTK, generate any of the messages in the protocol using fake data, calculate the PTK, and generate MAICs. Please see Appendix B for the complete ProVerif implementation of the Four-Way Handshake.

4.4 Attack on Four-Way Handshake

Utilizing the ProVerif resolution engine, we determined that confidentiality and authentication properties hold for the Four-Way Handshake

protocol as intended, even against the very powerful Dolev-Yao attacker model. However, as previously seen in [9], the protocol is vulnerable to a very simple denial-of-service (DoS) attack. By definition, a Dolev-Yao attacker controls the network, and is thus always capable of mounting a DoS. Unfortuately, the attack could feasibly be generated by a much less powerful attacker. All that is necessary is the ability to inject messages into the network. We consider this DoS attack to be a significantly lower-threat attack as no secrets are leaked and no impersonation can take place. However, since the modern world relies so heavily on the availability of web-based services, DoS attacks can no longer be tolerated, and liveness is now an important property to guarantee in a web protocol.

The Four-Way Handshake DoS attack is mounted by sending a new nonce, distinct from the Access Point's nonce, to the Station in response to each of the Access Point's legitimate attempts to initiate a session. Each time the station receives a nonce, it attempts to construct the corresponding PTK. Since this leads to an inconsistent set of PTKs, when the Access Point continues with the protocol, the Station now has a different view of the PTK than the Access Point, and authentication fails.

A graphical depiction of the attack is presented as:

1. $AP \rightarrow STA : N_{AP}$

   (STA calculates PTK)

2. $STA \rightarrow AP : N_S, MAIC(N_S)$

   (AP calculates PTK)

3. $Attacker \rightarrow STA : N_{Att}$

   (STA re-calculates PTK)

4. $STA \rightarrow Att : N_S, MAIC(N_S)$

   (STA continues protocol, believing it to be legitimate)

5. $AP \rightarrow STA : \{GTK\}_{PTK}, MAIC(\{GTK\}_{PTK})$

   (AP continues protocol, no idea about the attacker's message)

   (STA has different PTK than AP and thus fails authentication)

5 Conclusions and Future Work

Confidentiality and authentication are two very important properties that must hold in order for an internet protocol to be considered secure. WEP, unfortunately, was unable to provide them due to its use of the insecure RC4 cipher and the fact that initialization vectors were frequently reused. 802.11i, however, guarantees both authentication and security, and automatic correctness verifiers can find no attack that would violate either principle.

Unfortunately, it is not enough to only provide authentication and confidentiality. Much of today's business transactions are conducted electronically, and companies cannot afford a Denial of Service attack. For this reason, 802.11i needs to be modified. Changhua He et al. claim that their solution of nonce reuse has been adopted by the 802.11i standards committee [11].

References

[1] IEEE Standard for Information Technology- Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11-1997*, pages i–445, 1997.

[2] Wi-fi protected access: Strong, standards-based, interoperable security for today's Wi-Fi networks. White paper, Wi-Fi Alliance, 2003.

[3] IEEE Standard for Information Technology- Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Medium Access Control (MAC) Security Enhancements. *IEEE Std 802.11i-2004*, pages 1 –175, 2004.

[4] Martín Abadi. Security protocols: Principles and calculi tutorial notes.

[5] Bruno Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *14th IEEE Computer Security Foundations Workshop (CSFW-14)*, pages 82–96. IEEE Computer Society, 2001.

[6] Bruno Blanchet. Using Horn clauses for analyzing security protocols. In Véronique Cortier and Steve Kremer, editors, *Formal Models and Techniques for Analyzing Security Protocols*, volume 5 of *Cryptology and Information Security Series*, pages 86–111. IOS Press, 2011.

[7] Nikita Borisov, Ian Goldberg, and David Wagner. Intercepting mobile communications: The insecurity of 802.11. In *MobiCom '01: Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, pages 180–189, 2001.

[8] H.I. Bulbul, I. Batmaz, and M. Ozel. Wireless network security: Comparison of WEP (wired equivalent privacy) mechanism, WPA (wi-fi protected access) and RSN (robust security network) security protocols. In *Proceedings of the 1st international conference on Forensic applications and techniques in telecommunications, information, and multimedia and workshop*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[9] Changhua He and John C. Mitchell. Analysis of the 802.11i 4-way handshake. In *Proceeding of the Third ACM International Workshop on Wireless Security (WiSe '04*, pages 43–50, 2004.

[10] Changhua He and John C. Mitchell. Security Analysis and Improvements for IEEE 802.11i. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS '05)*, 2005.

[11] Changhua He, Mukund Sundararajan, Anupam Datta, Ante Derek, and John C. Mitchell. A modular correctness proof of ieee 802.11i and tls. In *Proceedings of the 12th ACM conference on Computer and communications security*, CCS '05, pages 2–15. ACM, 2005.

[12] Pascal Lafourcade, Vanessa Terrade, and Sylvain Vigier. Comparison of cryptographic verification tools dealing with algebraic properties. In *FAST '09: Proceedings of the 6th International Conference on Formal Aspects in Security and Trust*, 2010.

[13] Jong Liu, Jun Zhang, and Jun Li. Security Verification of 802.11i 4-Way Handshake Protocol. In *ICC '08: IEEE Inter-*

national Conference on Communications, 2008, pages 1642–1647, 2008.

[14] SPORE: Security Protocols Open Repository. Wired equivalent privacy protocol. http://www.lsv.ens-cachan.fr/Software/spore/wep.html.

A ProVerif Implementation of WEP

B ProVerif Implementation of Four-Way Handshake