

# SWFP: Secure Web Feed Protocol

Frédéric Giasson  
*fred [at] fgiasson.com*

## Abstract

*SWFP ensures the secure broadcasting of web feeds' content over a local network or the Internet. The protocol is built to ensure a secure channel to enterprises that broadcast news and information to their employees through web feeds. The protocol also ensures the legitimacy of the readers.*

## 1. Introduction

Many companies use content syndication technologies to broadcast news or information to their employees. Presently, 5% of the Internet users use web feeds technologies to automatically gather news and other information as soon as it is posted online [5]. They predict that this number will grow significantly in the next years. Eventually businesses will need a secure channel to deliver this information to this growing community.

Content syndication markup languages such as RSS and Atom are a new way to broadcast information on the Internet. I will refer to them as "web feeds". A web feed is a XML document describing a resource. This resource could be a text, a piece of news, a URL, an image, etc. The web feed are distributed by a web server like any other web resources. Readers use special softwares called "feed readers" to gather and display the information broadcasted by the web feed.

Web feeds have been introduced and popularized by blogs in the past few years. The first purpose of web feeds was to help blog readers to easily know if one of the blogger they read posted new articles on their blog. Medias started to use them to broadcast news. Eventually, businesses also used them to broadcast news and information about them or their products to their customers. Recently they started using the technology to broadcast internal news and information to their employees.

The goal of web feeds is to broadcast information. Up to now, no security features have been implemented to encrypt the content of these feeds; the features would have been useless. However, the fact that businesses

start to use web feeds to send information to their employees raises security concerns.

"Knowledge is power" Francis Bacon once said. Crackers like Kevin Mitnick used any type of information on enterprises to help him crack their computer systems. In his book, *The Art of Deception*, he describes how he can use information, which could seem insensitive, about a business to help him infiltrate it. The art of using this knowledge to infiltrate a business is called social engineering. How social engineers proceed? They learn about the business procedures and environment. They try to usurp the identity of an employee. They try impersonating this employee and try to convince some other employee that they really are who they say they are. They do this with the previously gathered information; the power they acquired.

Today businesses need to take this fact into account. They need to take care of what they put in their trash bins and need to secure them. They need to control the information they broadcast. The web feed technology is another source of information that businesses need to try to control. It is exactly the purpose of the Secure Web Feed Protocol (SWFP): to control the information sent by a business to its employees via web feeds.

The purpose of this paper is to propose the SWFP (section 3). It also presents a security analysis of the protocol (section 4). Finally the protocol is compared with other solutions that people thought about to answer this new web feeds' need (section 5).

## 2. Key Idea

Here is the key idea behind SWFP I propose. The web feed's content should be encrypted with one or more symmetric encryption algorithm. Each feed message has a type. These types are used to restrain messages to a class of readers. Each reader could have access to one or more message types. A symmetric encryption key should be created for each known symmetric encryption algorithm and message type. These keys are shared between the server and the legitimate readers and should be distributed by the server to the readers with a protocol using a symmetric encryption algorithm. The protocol must take care of the

information it encrypts to prevent know-plaintext attacks to the symmetric algorithm.

### 3. Secure Web Feed Protocol

The Secure Web Feed Protocol will secure the communication between two actors:

1. **The Web Feed Server.** This server is devised in three parts:
  - a. It generates symmetric keys to encrypt specific content in the web feed.
  - b. It generates authentication certificates for readers
  - c. It publishes the encrypted feed. It is normally a traditional web server
2. **The Readers.** This is any legitimate reader who wants to read the encrypted web feed's content

We have some steps to perform before starting the protocol. The first step is the distribution of asymmetric key pair between the server and a reader. It is also the creation of the reader's authentication certificate. This certification will assure the server that the public key of a requesting client really belongs to him and that the reader can read a specific message type.

We will not detail the asymmetric key distribution protocol here. Depending of the context where SWFP is used, an asymmetric key distribution protocol could perform better than another. The most important thing you have to take care of is the authenticity of the reader requesting an asymmetric key pair and an authentication certificate. Otherwise you could generate them to an intruder with problems that this situation will engender. In a business context, I think that the best way to distribute the asymmetric key pair and the authentication certificate should be by hands. The reader should generate his asymmetric key pair and hand the public one to the administrator of the web feed server. Then the administrator should create the authentication certificate with the reader's ID, public key and the message types he will have access to and give it to him. Remember that the whole system is based on the assumption that the readers are genuine.

At any time the server can generate a symmetric key for each symmetric encryption algorithm and each messages type.

The protocol is built to handle the encryption of the feed with more than one symmetric encryption algorithm. This feature gives flexibility to the system's administrator and the feed reader's developer. If for some reason one of them does not like a specific

symmetric encryption algorithm he will have the possibility to choose another one. If a new symmetric encryption algorithm is created, the protocol will be able to take it into account immediately.

A type is a way to classify readers within the feed. It restrains them to have access to certain classes of messages. Their purposes are defined by the system administrator. The types accessible by a reader are indicated in his authentication certification.

For each handled symmetric encryption algorithm and for each defined type the server needs to generate a symmetric key. If he handles 3 symmetric encryption algorithms and has 3 types defined, he will need to generate 9 symmetric keys.

After that the web feed server has generated the symmetric keys he will encrypt given elements of the web feed with the newly generated symmetric keys.

This step is really important for one thing: the protocol should take care of the know-plaintext attacks. It is the reason why it only encrypts some elements and not the whole feed's content. Below is a RSS 0.91 file that I will take to demonstrate how the encryption of the feed is performed [2].

```
<?xml version='1.0'?>
<!DOCTYPE [...]>
<rss version='0.91'>
<channel>
<copyright>Copyright 2005 </copyright>
<pubDate>Wed, 24 Apr 2005 07:00:00</pubDate>
<description>A description</description>
<link> http://radio.weblogs.com/0140770/ </link>
<title>FredOnSomething</title>
<webMaster>fredg@videotron.ca(Fred)</webMaster>
<language>fr-ca</language>
<item>
<title> [encrypted data] </title>
<link> http://radio.weblogs.com/0140770/ </link>
<description>
[encrypted data]
</description>
</item>
</channel>
</rss>
```

The only elements that will be encrypted with the symmetric keys are the possible sensitive texts in the elements <title> ... </title> and <description> ... </description>. We will put us in the skin of an intruder and we will take into account that the whole RSS feed

above is encrypted with the symmetric keys. We know that the file encrypted is a RSS feed. We also know that a RSS feed is built with tags like <rss>, <item>, <channel>, etc. All these tags are known to the intruder and can be used to attack the symmetric encryption algorithm. Encrypting insensitive data like the copyright or the date could also cause problems because this information is easy to guess. Encrypting these unnecessary elements would open the symmetric encryption algorithm to know-plaintext attacks.

The server can regenerate the symmetric keys at any time. If you choose to regenerate them, the system will re-encrypt the web feed with the newly created symmetric keys.

The regeneration of the keys could be done manually by the system administrator or scheduled in the server. The purpose of this key regeneration is to give the possibility to system administrator to change the encryption keys when needed. Depending on the situation it could be useful the change them each hour or every year.

The reader checks if the feed contains new messages. If there are new messages, he will check the types of these new messages. If he can read the messages' types then he has two choices:

1. If he already has the encryption key, he can check if the encryption key is still the same.
2. In any case he can request the symmetric encryption key that encrypted this type of message to the server.

When these preliminary steps are done, a legitimate reader could then establish a communication with the web server to retrieve the encryption keys to decrypt the web feed's content. The protocol is described as:

1. **A**  $\longrightarrow$  **S**:  $N_a, \text{prefs}, \text{Cert}_a$
2. **S**  $\longrightarrow$  **A**:  $\{K\}_{K_a}, \{H(A, N_a, K)\}_{K_s^{-1}}, \{M\}_k$

At the first stage of the protocol the reader requests the symmetric encryption key that encrypted a message of a certain type that has been encrypted by a certain symmetric encryption algorithm.

**Note:** *The web feed will be available to anyone that has access to the web server. If the web server is connected on the Internet then anyone will be able to read the web feed's content. This is not a problem in itself because the sensitive information is encrypted. Only the readers that are registered to the web feed's*

*server with his public key will be able to request the symmetric key that encrypted the feed's content.*

- **A** is the reader
- **S** is the web feed's server
- $N_a$  is a nonce
- **Prefs** are the preferences of the protocol. In this preferences you have:
  - **T** is the type of the message
  - **Ea** is the desired symmetric encryption algorithm
- **Cert<sub>a</sub>** is the authentication certificate of **A**

The server will validate the request he received from the reader at stage 1. The authentication certificate **Cert<sub>a</sub>** is defined by:  $A^*, Ts, K_a, \{H(A^*, Ts, K_a)\}_{K_s^{-1}}$

Where:

- **A\*** is the identification number of the reader (e.g. his employee ID). The start (\*) tell us that the ID of the reader is optional. The reader is always legitimate but could be anonymous.
- **Ts** are the types available to the reader
- **K<sub>a</sub>** is the public key of the reader
- **H(...)** is the hash that ensure the integrity of the certificate
- $\{...\}_{K_s^{-1}}$  is the signature of the certificate by the server **S**

Below are the steps to perform to validate the received message. The validation is done in two steps. The first one is to validate the authentication certificate of the reader:

1. The server hashes **A\***, **Ts** and **K<sub>a</sub>** and saves the resulting hash:  $\text{hash} \longleftarrow H(A^*, Ts, K_a)$
2. The server decrypts the hash with its public key **K<sub>s</sub>**:  $\text{hash2} \longleftarrow \{H(A^*, Ts, K_a)\}_{K_s}$
3. The server compares the two resulting hashes **hash** and **hash2**
  - a. If **hash == hash2** then the certificate is valid
  - b. Else the certificate is considered falsified and discarded
    - i. The server ends the protocol with an error

The second step is to check if the requested symmetric encryption algorithm is known by the server and if the reader really has access to the requested type:

4. The server checks if the preference **Ea** is in his know symmetric encryption algorithm list
  - a. If yes, the server continues the protocol
  - b. Else, the server ends the protocol with an error

5. The server checks if the preference **T** belongs to **Ts**.
  - a. If yes, the server continues the protocol
  - b. Else, the reader does not have access to this message's type and the server ends the protocol with an error

It is important that the server ends the protocol if he suspects that the certificate was sent by an intruder. The server should send a notification to the system administrator to tell him about the possibility of an intruder in the system.

At the second stage of SWFP the server sends the requested symmetric key to the reader:

- $\{K\}_{k_a}$  is the encrypted symmetric key with the public key of the reader
- $H(\dots)$  is a hash to ensure the integrity of the message
- $\{\dots\}_{k_s^{-1}}$  is the signed hash by the server with its private key.

The reader will validate the request he received from the server. Below are the steps to perform in order to validate the received message:

1. The reader will decrypt the symmetric key with his private key:  $\{K\}_{k_a}^{-1}$
2. The reader hashes **A**, **N<sub>a</sub>** and **K** and saves the resulting hash:  $\text{hash} \leftarrow H(A, N_a, K)$
3. The reader decrypts the hash with the public key of the server  $K_s$ :  $\text{hash2} \leftarrow \{H(A, N_a, K)\}_{k_s^{-1}}$
4. The reader compares the two resulting hashes **hash** and **hash2**
  - c. If **hash** == **hash2** then the message is valid.
  - d. Else the message is considered falsified and discarded.
    - i. The reader ends the protocol with an error.

It is important that the reader ends the protocol if he suspects that the message was sent by an intruder. The protocol does not want that an intruder sends false information to our reader. Think about the consequences in a business environment where the readers (the employees) trust the messages they read coming from the feed.

When the reader has validated the message that came from the server he will decrypt the message with  $\{M\}_k^{-1}$ . Above I said that only some elements of the feed will be encrypted. However, nothing prevents the feed to be wholly encrypted. Then,  $\{M\}_k$  describe a feed partially or wholly encrypted.

Finally the unencrypted feed's content is displayed in the feed reader.

If there was more than one new message in the feed, the reader could perform the same steps to request the symmetric keys of the other new messages. Remember, all messages do not have the same type, then the symmetric keys is not the same.

## 4. Security Analysis

Some type of attacks could be performed to crack SWFP. This section is dedicated to some of them. This is not an exhaustive list but one that describes what could look like an attempt to crack the protocol.

The protocol tries to answer to different needs:

1. To send the symmetric key which encrypted a feed's message to a legitimate reader
2. To ensure the confidentiality of the sent symmetric key
3. To be sure that the messages sent by the reader or the server are fresh and not played back by an intruder

These properties are assured by some key features:

1. We are sure that the reader is legitimate when the server validates the authentication certificate sent by a reader. He knows if the reader is legitimate and if he is able to read the specific type of message he is requesting
2. We are sure that the confidentiality of the symmetric key because it is encrypted with the public key of **A** and only **A** could decrypt it with his private one
3. We are sure that the messages sent by the reader or the server are fresh because **A** generate a random number at each session.

### 4.1. Cracking ciphers

SWFP could use many symmetric and asymmetric encryption systems. If one of these systems is cracked, then the whole protocol is cracked. The security level of the whole encrypted feed is as strong as its weakest component.

### 4.2. Men in the middle attack

It seems that this type of attack is difficult to perform because SWFP uses authentication certificates.

This assertion is true as long as the reader's software is not corrupted by an intruder. If the reader's software has been compromised and the server's public key has been replaced by the one of the intruder then the intruder could pretend to be the server and send false information to the readers through the secure channel.

## 5. Related Work

Really few works have been done on the subject. As far as I know this is the first protocol that deals with this aspect of the web feeds. A discussion erupted two years ago on a blog where they raised the need for such a feature [1]. The solution suggested by the author was to use HTTP over SSL and HTTP Authentication to answer to the need. I will come back to this solution later.

Steven Garrity[1] asked the question: Why not using HTTP over SSL and HTTP Authentication to answer to this need? Some person, like Steven Garrity, uses already developed and accepted protocols like SSL to try to cope with the problem. However, I don't think this is the good strategy.

SSL and PGP could possibly be used to encrypt the web feeds' content, but they do not seem appropriate for the kind of problem we have. One of the main features of the SSL protocol is that the encryption symmetric key is changing at each new session. What does it mean? It means that each time a user establishes a SSL session with a server a symmetric encryption key is generated. It's even worse because the two parties participate to the creation of the session's symmetric encryption key. The goal of PGP is not the same. It's to encrypt email, files or any other resource once. It's not intended for broad distribution. In both protocols we don't have the notion of types that is central to SWFP. It's why these algorithms do not seem appropriate for this task.

The cost would be too high to implement one of these protocols to handle the need of the secure broadcasting of web feeds' content. If we wish to implement one of these protocols to distribute an encrypted feed we would need to encrypt the whole feed each time a reader would request to read the feed. The cost is just too high.

The strategy proposed by Steven Garrity is to use HTTP over SSL (HTTPS) with HTTP Authentication. HTTPS would provide the secure channel and HTTP Authentication would provide the authentication mechanism. A problem raised by Garrity is that some feed readers only implement HTTPS, others HTTP

Authentication and few implement both. An inconvenient I also see with this strategy is that who says HTTP Authentication also says login and password. In SWFP the authentication is performed with authentication certificates. The authentication steps of the reader to the server are transparent to him. I think that this transparency feature is an important one because it simplifies the process and brings non-expert users to use it. Only the things that appear simple are widely used.

Two types of feed readers are available: the web applications like Bloglines [3] or the standalone softwares like Omea Reader [4]. Both principles, HTTPS with HTTP Authentication and SWFP, could be implemented in standalone software and the implementation time, cost and difficulty are probably comparables. However, I think that SWFP would be much easier to implement in web applications. To use HTTPS with HTTP, the web applications would need to create the secure channel themselves with the feed's server. For example, Bloglines itself would need to create the secure channel with each private feed server. I do not think that it is imaginable. However, with SWFP nothing like that would be necessary because the encrypted feed is viewable by anyone who needs it, even web applications. From my personal experience If I check the FeedBurner stats of my personal blog: 30% of my readers use Bloglines. I think that it is considerable and that we need to take this fact into account.

Another aspect of the HTTP Authentication strategy is that it is not an optimal strategy to the problem. If a user subscribes to many private feeds then he will need to enter, each time, a login and password to check the feeds. All these elements of HTTP hinder the main purpose of using web feeds which is to make the wide distribution of information content easy and convenient. Personally I do not think this is viable. Think about the pain such a situation would engender, nobody would subscribe to such feeds.

## 6. Conclusion

We propose SWFP to secure web feed content for business-oriented applications. The inner workings of SWFP have been detailed and its security properties analyzed. We saw how SWFP works and analyzed his security properties. We saw the possible attacks that could be performed to crack the protocol. We finally compared SWFP with the few related works that have been written on the subject.

The first users I targeted were businesses. However, other types of users could be interested in such a protocol? I think it could be interesting for anyone who has a private blog viewable by their friends and families. It could be implemented as plug-ins in existing blog systems like WordPress, or added in services like Radio Userland or TypePad. Anyone who needs to confidentially broadcast information with web feeds could also be interested in SWFP.

It is what we will have to do as future work: implementing SWFP in a prototype system then in real systems like Type Pad or WordPress and in feed readers like Bloglines and Omea Reader.

## 7. References

[1] Private RSS Feeds: Support for security in aggregators, Steven Garrity, <http://labs.silverorange.com/archives/2003/july/PrivateRSS>

[2] RSS 0.91 Spec - revision 3, Dan Libby  
<http://my.netscape.com/publish/formats/rss-spec-0.91.html>

[3] Bloglines Home Page: <http://www.bloglines.com>

[4] Omea Reader Home Page:  
<http://www.jetbrains.com/omea/reader/>

[5] Trust MEdia: How Real People Are Finally Being Heard, by Edelman and Intelliseek, Spring 2005,  
[http://www.edelman.com/image/insights/content/ISwp\\_TrustMEdia\\_FINAL.pdf](http://www.edelman.com/image/insights/content/ISwp_TrustMEdia_FINAL.pdf)