BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **14**, No 3

Sofia • 2014 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2014-0032

A Prevention Model for Session Hijack Attacks in Wireless Networks Using Strong and Encrypted Session ID

S. S. Manivannan, E. Sathiyamoorthy

School of Information Technology and Engineering, VIT University, Velllore, Tamil Nadu, India
Emails: manivannan.ss@vit.ac.in esathiyamoorthy@vit.ac.in

Abstract: *Most of the web applications are establishing the web session with the client. It is very important to protect the wireless networks against session hijacking attack. Session Hijack attack is easy to execute and difficult to detect. Wireless networks do not have specific boundary regions for the packets to be transferred. As the data packets are transferred in air, the chances of sniffing the network packets by the hackers or attackers are high by using the network sniffing tools. In this paper, we have proposed the Strong and Encrypted Session ID to prevent the session hijack attacks in web applications. Session ID is generated and the generated Session ID is encrypted, using a Secret Key Sharing algorithm and decrypted at the client side. We have tested the integrity of the session ID of length 32, 92 and 212 characters in a web application. Attacks are executed to capture the session ID of a web application. Our experimental results proved that 212 characters encrypted session ID completely prevents the session hijack attacks in web applications of wireless networks.*

Keywords: *Wireless networks, session Hijack attacks, network sniffing, encrypted session ID, SKS algorithm.*

1. Introduction

Wireless networks have emerged in the areas of education, information technology and communication, entertainment and commercial applications. Wireless networks are weakly secured against variety of attacks, such as Denial of Service, brute force attack and session Hijack attacks. Session Hijack attack is a severe threat to the wireless networks. Most of the web applications are involved in creating the session with the client. HTTP is the default protocol responsible for establishing the session at the application layer. The web session is the data transfer and communication between the client and the web server for the specific time period. The server side web sessions cannot handle the congestion perfectly. In client side web sessions, session cookies are used to maintain the state of the web applications. A Session Identifier is a unique ID assigned by the web server to each web session when a session is established between the client and the server. HTTP is a stateless protocol. Each request is independent. HTTP does not monitor the requests. Session attributes are used to maintain the state of the web applications. Session IDs are used to maintain the state of the sessions in web applications. Cookies are used to store the session IDs. There are several types of cookies available to maintain the state of the web applications that are listed in Table 1.

Table 1. Types of cookies

No	Cookie	Functionality and behaviour
1	Session cookie	Session cookies get deleted from the browser when the user closes the browser
2	Persistent cookie	It has a field “expire”. The persistent cookies get deleted after the time period is expired.
3	Secure cookie	Cookies are encrypted when it was transmitted
4	HTTP only cookie	Cookie will be used only for http or https protocol
5	Third party cookies	Third party cookies are set by multiple domain names
6	Super cookie	To track the technology that does not rely on HTTP cookies
7	Zombie cookie	The cookie are automatically recreated

There are several vulnerabilities that attack the current web application and they are listed in Table 2.

Table 2. Web application vulnerabilities

No	Vulnerability	Description
1	Session sniffing	Unauthorized way of viewing the session’s data during data transmission
2	HTTP packet sniffing	Sniffing the http packet of a web application session established between a client and a server
3	Session prediction	Predicting the session ID of a web session by using a brute force attack
4	Session fixing	Session ID is fixed by the attacker before the client establishes the session with the server
5	Session Hijacking	Session ID is sniffed and the session is hijacked after the client has established the session with the server

Based on the survey conducted in 50 web applications that belong to national and international companies, the percentage of web application vulnerabilities are analyzed and listed in Table 3.

Table 3. Percentage of Web application vulnerabilities

No	Vulnerabilities	Percentage
1	SQL injection	30 %
2	Session Hijacking	28 %
3	Cross site scripting	18 %
4	Distributed DoS attack	8 %
5	Phishing attack	8 %
6	Cloning attack	4 %
7	Others	4 %

2. Related works

Alex, Jason, Huany and Mohamed used the Keyed Hash Message Authentication Code (KHMAC) to verify the authentication [1] of the client and also to defend the replay attacks and volume attacks. Chomsiri has presented the HTTPS hacking protection [2] using ARP table, ARP watch and Anti sniff. Antony has discussed the disclosure of the online cookie use and its effects on consumer's trust and anticipated patronage [3] using three different studies

Ben Adida has presented the method of securing web sessions against Eavesdropping [4] using the secret token. The secret token is transferred over SSL stream and thus prevents the web session from eaves dropping attacks. Collin Jackson and Adam Barth have discussed protecting the high security websites from network attacks [5] using the Force Non https stream converted to https stream by a force https cookie. Juels, Markus and Tom have narrated the cache cookies for authenticating the web browsers [6] using an identifier tree and the Rolling Pseudonym scheme.

Nenad, Christopher and Kirda have presented the static analysis tool [7] called pixy to detect the web application vulnerabilities, such as cross site scripting, SQL injection and command injection. Richard Ford and Michael have presented the man in the middle attack to attack the https protocol [8]. Hacker injects the malicious code into the certificate and sends the fake certificate in the name server to the client. Shirley Gaw and E.W. Felten have presented various methods of storing the passwords and different methods of managing the online user passwords [9].

Paul Ritchie has discussed the list of security risks [10] that affects the web applications which are designed using Asynchronous Java script and XML (AJAX). The cross site scripting attack can be prevented by validating the client side user inputs to the web applications. Adam Barth, Collin Jackson and Jhon Mitchell have discussed Cross Site Request Forgery attacks and their defense methods [11].

Ben Adina has presented the method of securing the web application session against eavesdropping and session hijacking attacks [12] using the Session Lock protocol. F. Wang and Y. Zhang have presented the Secure Authentication and Key Agreement (SAKA) method [13] which provides mutual authentication and secure key management for session initiation protocol.

Roberto, Davide Ariu, Prahlad, Giacinto and Wenke Lee have presented the multiple classifier system for anomaly detection [14] that has a high detection rate against shell code attacks, polymorphic attacks and generic attacks. Ori Eisen has discussed the method of catching the man-in-the-middle and man-in-the-browser [15]. Yi pin Liao and S. S. Wang have presented the Self Certified public keys (SCPKs) which are more secure than the traditional HTTP digest authentication protocol for Session Initiation Protocol [16].

Cichon, Golebiewski and Miroslaw have discussed the advantages of key redistribution [17] over key pre-distribution in ad hoc networks. Armando, Roberto, Luca, Jorge, Giancarlo and Sorniotti have presented the flaw in the authentication of a single sign on protocols [18]. Self signed client certificate is the suggested solution to overcome the authentication flaw. Natallia Bielova has discussed the survey of java script security policies in the web browser [19]. He has compared the existing security policies with the current security policies for web developers. Y. Xiang, X. Shi, J. Wu, Z. Wang have presented the fast secure BGP routing protocol [20]. Traditional Internet routing protocols, such as Inter Domain Routing protocol, Border Gateway protocol are weak against malicious attacks. The presented FS-BGP is able to secure the AS paths and also prevent the prefix hijacking and routing attacks.

Nikolay Dokev and Ivan Blagoev have presented the signer and sender [21] by using HTTP method to transmit the authenticated data over networks. Evelina Pencheva and Ivaylo Atanasov have discussed the open service access and CAMEL application part protocol [22] to control the session in mobile networks.

3. Proposed system

The proposed system architecture is shown in Fig. 1. The web server generates the Session ID of required length using a Session ID generation algorithm. The generated session ID is encrypted at the server side and decrypted at the client side using the Secret Key Sharing (SKS) algorithm. When the client receiving the encrypted session ID, attacks are executed to capture the session ID and the results are analyzed and recorded.

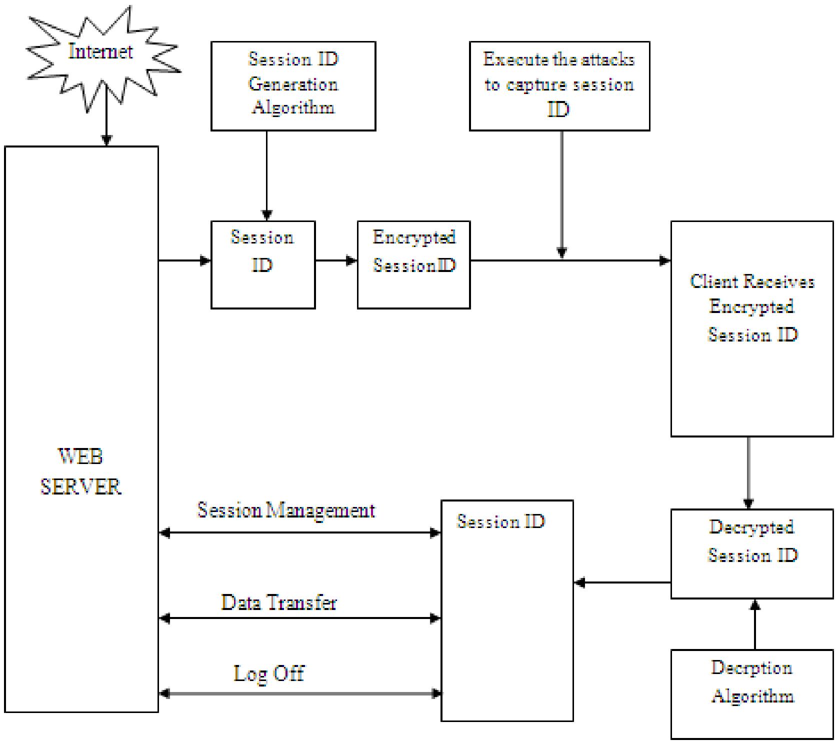


Fig. 1. Proposed system architecture

3.1. Proposed approaches

We have presented strong and Encrypted Session ID for the sessions in three different cases, such as 32,92 and 212 characters.

Case 1. 32 characters Session ID with encryption.
Case 2. 92 characters Session ID with encryption.
Case 3. 212 characters Session ID with encryption.

3.2. Session establishment

The client establishes the session with the server. The client is authenticated by the server by its login credentials.

3.3. Session ID generation

The web server generates the session ID using the following algorithm.

Algorithm 1

```
1. Initialize the following variables
        result, random_no, tempbuffer, resut_byte_length;
3. While result_byte_length < sessionIdLength then
       Generate the random_no using Message Digest
4.   For ( i= 0 ; i<randnum.length and result_byte_length < sessionIdLength ; i++)
              byte b1 = (byte) ((randnum[i] & 0xf0) >> 4);
              byte b2 = (byte) (randnum[i] & 0x0f);
              if (b1 < 10)
                 tempbuff.append((char) ('0' + b1));
              else
                 tempbuff.append((char) ('A' + (b1 – 10)));
              if (b2 < 10)
                 tempbuff.append((char) ('0' + b2));
              else
                 tempbuff.append((char) ('A' + (b2 – 10)));
              resultLenBytes++;
     End for
5.   if (jvmRoute != null)
            {
            tempbuff.append('.').append(jvmRoute);
            }
        result = tempbuff.toString();
   End while
```

3.4. Encryption and decryption of a session ID

The generated session ID is encrypted at the server side and decrypted at the client side using a Secret key sharing algorithm.

Algorithm 2. Secret key sharing

Step 1. The client establishes the session with the server using a login password.

Step 2. The client requests a RSA Public key from the server.

Step 3. The client encrypts the login password with RSA Public key.

Step 4. The server decrypts the login password and stores it in the session.

Step 5. The server encrypts the generated Session ID with AES and sends it to the client.

Step 6. The client decrypts the Session ID using AES with the login Password.

Step 7. Both the client and the server have now the same "secret key" which is used for communication.

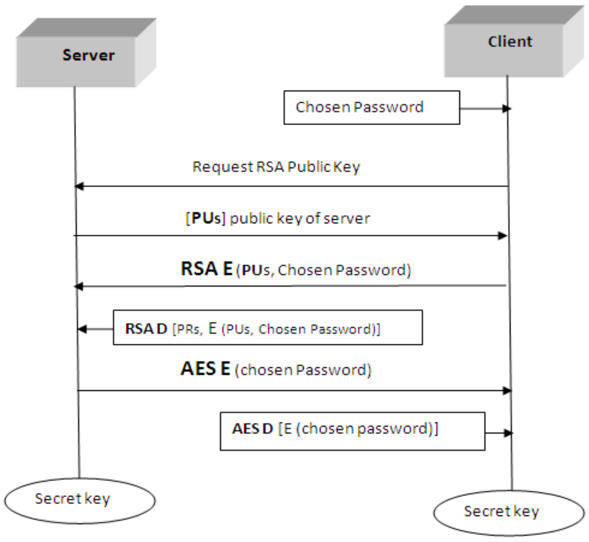

Server
Client
Chosen Password
Request RSA Public Key
[PUs] public key of server
RSA E (PUs, Chosen Password)
RSA D [PRs, E (PUs, Chosen Password)]
AES E (chosen Password)
AES D [E (chosen password)]
Secret key
Secret key

Fig. 2. Secret key sharing

Case 1. 32 Character Session ID encrypted with SKS

In this case the web server generates the 32 characters Session ID and encrypts the 32 character Session ID, using SKS algorithm and assigns it to the client.

```
(i)   Generate session ID =32 chars
(ii)  $SID_{new} \longleftarrow$ Encrypt { $SID_{32\ chars}$ }
(iii) Client receives the encrypted session ID
(iv)  While client receiving $SID_{new}$
          do
            capture session ID ( )
              execute  packet sniffing attack ( )
              execute man-in the middle attack ( )
          end
(v)   $SID_{attacked} \longleftarrow$ number of session IDs hijacked
(vii) $SID_{prevented} \longleftarrow$ number of session IDs not hijacked
(viii) $SID \longleftarrow$ decrypt($SID_{new}$)
```

Case 2. 92 Character Session ID encrypted with SKS

In this case the web server generates the 92 characters Session ID and encrypts the 92 character Session ID using SKS algorithm and assigns it to the client.

```
(i)    Generate session ID =92 chars
(ii)   $SID_{new} \longleftarrow$ Encrypt { $SID_{92\ chars}$}
(iii)  Client receives the encrypted session ID
(iv)   While client receiving $SID_{new}$
           do
             capture session ID ( )
               execute  packet sniffing attack ( )
               execute man-in the middle attack ( )
           end
(v)    $SID_{attacked} \longleftarrow$ number of session IDs hijacked
(vii)  $SID_{prevented} \longleftarrow$ number of session IDs not hijacked
(viii) $SID \longleftarrow$ decrypt($SID_{new}$)
```

Case 3. 212 Character Session ID encrypted with SKS

In this case the web server generates the 212 characters Session ID and encrypts the 212 character Session ID using SKS algorithm and assigns it to the client.

```
(i)    Generate session ID =212 chars
(ii)   $Sin_{new} \longleftarrow$ Encrypt { $SID_{212\ chars}$}
(iii)  Client receives the encrypted session ID
(iv)   While client receiving $SID_{new}$
           do
             capture session ID ( )
               execute  packet sniffing attack ( )
               execute man-in the middle attack ( )
           end
(v)    $SID_{attacked} \longleftarrow$ number of session IDs hijacked
(vii)  $SID_{prevented} \longleftarrow$ number of session IDs not hijacked
(viii) $SID \longleftarrow$ decrypt($SID_{new}$)
```

4. Experimental analysis

4.1. Experimental setup

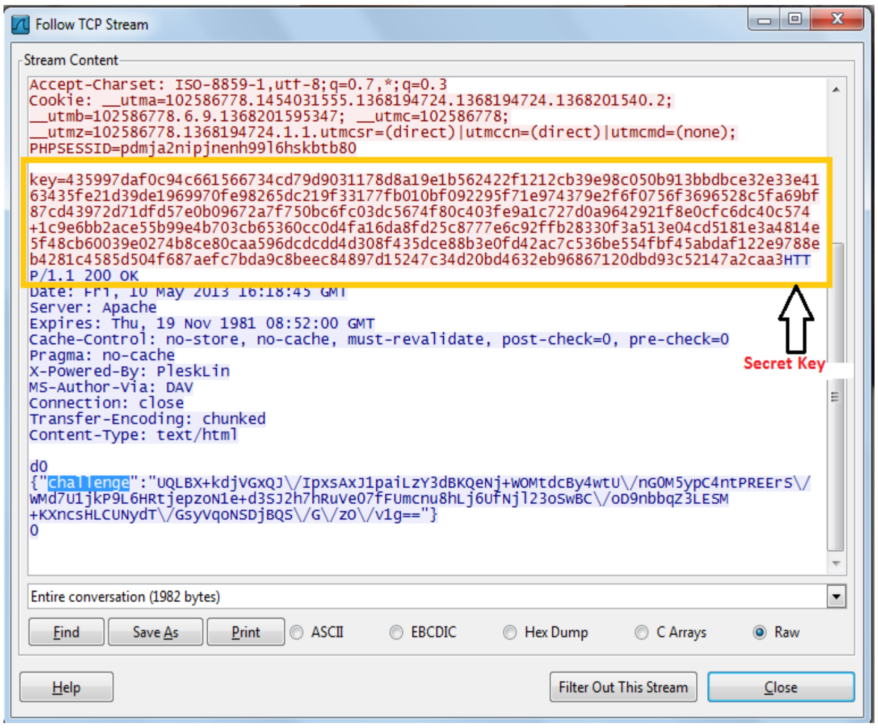
The web application **www.nationalrailways.com** is designed, using Java and Apache Tomcat Server. The client is authenticated using the login credentials, such as user name and password. The client is logged in to the web server by establishing the web session with the server. The server assigns the unique session ID for each time the client logs in to the server.

4.2. Experimental results

RSA keys used in Encryption and Decryption for the following cases. For example, each time these values will also change per session. The values of **n, e, d, p, q** are given below.

'n' =>
'164379245410857557641543769408676439267473563141667230709748498167801774396077697951590761314957322298258693612001197190127717413680981711035447584919018807603717766956585325149910624585910354666172378463673608292455847279835127487292100790379732190893633938061116797523645882456878244862202061503957629180613',
 'e' => '65537',
'd' =>
'636779267053855329570092194904783166779470958548704953424644837493957832730765917816651089958975311428382701896865342365833420769722868559709468435528640132441367320820511935048312965406554127380275979024950091807400669246394667237433894452201102526102493640922402781307495419230133451652749743522805035982225'
 'p' =>
'133125582549961745599169700863629452446216655565329677958818749503007431630337423661240129102196594278989483276005259649883259288825479844254230333 63239507'
 'q' =>
'123476827114853284795575770325752207511106752998046692826847985680819714062893000689159575591058916066782134213417962104069319370099392126733376830 74657159'

Fig. 3 shows the generated secret key using the secret key sharing algorithm. The generated key is shared between the client and the server.



```
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
Cookie: __utma=102586778.1454031555.1368194724.1368194724.1368201540.2;
__utmb=102586778.6.9.1368201595347; __utmc=102586778;
__utmz=102586778.1368194724.1.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(none);
PHPSESSID=pdmja2nipjnenh99l6hskbtb80

key=435997daf0c94c661566734cd79d9031178d8a19e1b562422f1212cb39e98c050b913bbdbce32e33e41
63435fe21d39de1969970fe98265dc219f33177fb010bf092295f71e974379e2f6f0756f3696528c5fa69bf
87cd43972d71dfd57e0b09672a7f750bc6fc03dc5674f80c403fe9a1c727d0a9642921f8e0cfc6dc40c574
+1c9e6bb2ace55b99e4b703cb65360cc0d4fa16da8fd25c8777e6c92ffb28330f3a513e04cd5181e3a4814e
5f48cb60039e0274b8ce80caa596dcdcdd4d308f435dce88b3e0fd42ac7c536be554fbf45abdaf122e9788e
b4281c4585d504f687aefc7bda9c8beec84897d15247c34d20bd4632eb96867120dbd93c52147a2caa3HTT
P/1.1 200 OK
Date: Fri, 10 May 2013 16:18:45 GMT
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
X-Powered-By: PleskLin
MS-Author-Via: DAV
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html

d0
{"challenge":"UQLBX+kdjVGxQJ\/IpxsAxJ1paiLzY3dBKQeNj+WOMtdcBy4wtU\/nGOM5ypC4ntPREErS\/
WMd7U1jkP9L6HRtjepzoN1e+d3SJ2h7hRuVe07fFUmcnu8hLj6UfNjl23oSwBC\/oD9nbbqZ3LESM
+KXncsHLCUNydT\/GsyVqoNSDjBQS\/G\/zO\/v1g=="}
0
```

Fig. 3. Generated secret key

Case 1. 32 Characters encrypted session ID

The server generates the 32 characters session ID and encrypts the 32 characters session ID using SKS algorithm. The encrypted session ID is assigned to the web client. At each session the server assigns the unique session ID to the client. The encrypted session ID, assigned to the 4 sessions are given in Table 4.

Table 4. 32 character plain session ID and encrypted session ID

Session ID (Length=32)	Encrypted session ID
61BBF1C93852828924718B CA037854F0	iQHBPW6HDFKejs7QlOnmUVgzCPNJz1oyCF4S0x/+Ahlvqp NuS9IJLg==
DDAF120DB46480BD6F2F3 3DA89C7EF81	tQIfNJqHDFJmqTakxTMDWC0rCKjODO5RdRNGecol1U6 W8WBm+t4DUA==
945A6E5AB65C1203B78B1 CEC54C6E22F	rAIGmrSHDFIv+gvuRmqTb1WAvUVlWAhjTR34zXh5N84i Oxrj+FglLg==
0A35043F669179D7D22999 3FD0519C17	4wN2FsKHDFI48ycrLQPfa4GDDi/GAfaymvbalYA0itjCHSop /qosmw==

The attacks are executed to capture the Session ID. The integrity of the session ID is tested by creating 10 sessions, 20 sessions and 30 sessions between the client and the server in the web application. The observed results for each session are given in Table 5.

Table 5. Results of 32 characters encrypted session ID

No	Metrics	50 chars encrypted Session ID		
		10 sessions	20 sessions	30 sessions
1	Number of the unique session IDs generated	10	20	30
2	Number of the session IDs attacked	1	2	2
3	Number of the session IDs prevented	9	18	28
4	Session Hijack prevention rate	90 %	90 %	94 %

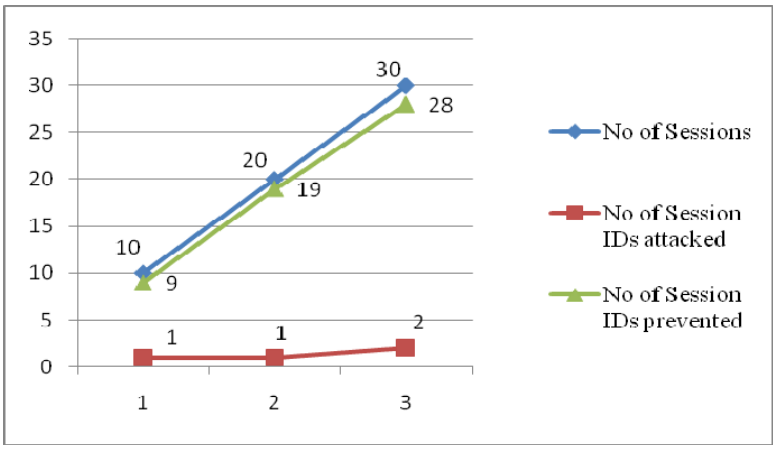


Fig. 4. No of Session IDs prevented for 10, 20, 30 sessions

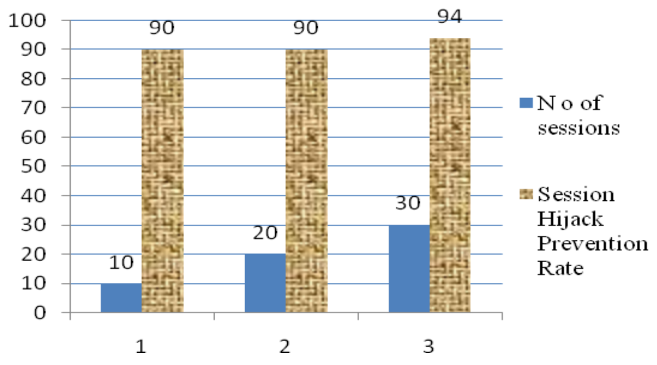


Fig. 5. Session Hijack prevention rate for 32 chars encrypted session ID

The results show that 32 characters encrypted session ID has a session hijack prevention rate of 90 % for 10 sessions, 90 % for 20 sessions and 4 % for 30 sessions.

Case 2. 92 character encrypted session ID

The server generates the 92 characters session ID and encrypts the 92 characters session ID using SKS algorithm. The encrypted session ID is assigned to the web client. At each session, the server assigns the unique session ID to the client. The encrypted session ID, assigned to the 4 sessions are given in Table 6.

Table 6. 92 character plain session ID and encrypted session ID

Session ID (Length=92)	**Encrypted Session ID**
13B1F75CD8026ACB057E037471A93C699B7E3738107E71F109D9888B14593f12e1cf999f58dede5db9bd87c578ec	QgA7Xe6IDFI6pFqFxC11Zz3LmYd9x0IKyChFk27tYDyrSj59QwcTv76A+y9NSRCJWIhP6j2fUOYThjog2HpScaUNjltaLr2PFsQ4G0gixR27f0FS0IzPiGIS/gV6bOGgKGQ3Mw==
38C8FFD13BF55A33E2DDEE751F04601D89AFB06DA2DD8819C149354B792657a11398b5b340b55465d451a74af342	2gIvbhKJDFJRWfZ3ESEtBbF5pnGwbahRc0oH8xKVV2XnWG8WTGuv9ElaJUwAPu7V5vPbaAaMJ38coRUvUXwFVSZZ/tlt97LCvcVHPbjYcqwBDAn3tZLT0xrAHtL7V1yIQ704Uw==
685170F3AE1E52434739E2587256F5C9F4BF4B912BD5606DBDA716A75A4B9f4a276b054fc84518e4586a8f790d10	cAPlLyKJDFKm52dBdIUj5MePw/0VN11k+WEfkImirEGHFuEgAC/qn8s6uN0dWKKF6ClC94Xg9OBBakyPGC/hsUlL5/9+1S9naBJKeraUeSTlSBrdgfO3iUsbBM1G2mqFrIxCmA==
1F6F15E9D28E5149B05294FD65897CE28E98FB8D0C7442C781EACE1FBB9265c530f4ffd3843da112cab046118799	MwNL50OJDFK4mhYesSvBbhV3HVJ6n6BfFLgL2CCWN9qzwyVSJoKxfreomGQvA/v6IxHBxRjt+gECrRqieVowVfzz/QWG5Wwl+YEg4CCxa/eP2ECbuV

The attacks are executed to capture the Session ID. The integrity of the session ID is tested by creating 10 numbers of sessions, 20 numbers of sessions and 30 numbers of sessions in the web application between the client and server. The observed results for each session are given in Table. 7.

Table 7. Results of 92 characters encrypted session ID

No	Metrics	50 chars encrypted Session ID		
		10 sessions	20 sessions	30 sessions
1	Number of unique session IDs generated	10	20	30
2	Number of session IDs attacked	0	1	1
3	Number of session IDs prevented	10	19	29
4	Session Hijack prevention rate	100 %	95 %	97 %

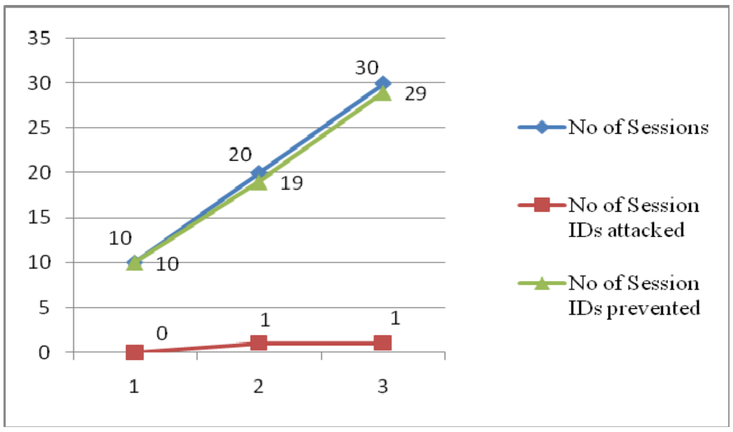


Fig. 6. No of session IDs prevented for 10, 20, 30 sessions

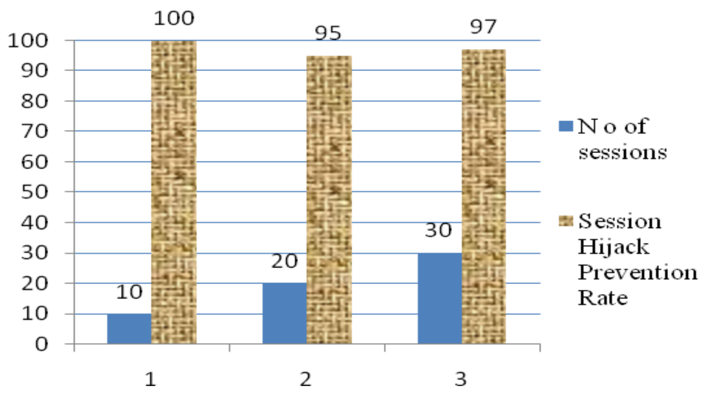


Fig. 7. Session Hijack prevention rate for 92 chars encrypted session ID

The results shows that 92 characters encrypted session ID has a session hijack prevention rate of 100 % for 10 sessions, 95 % for 20 sessions and 97 % for 30 sessions.

Case 3. 212 character encrypted session ID

The server generates the 212 characters session ID and encrypts the 212 characters session ID. The encrypted session ID is assigned to the web client. At each session, the server assigns the unique session ID to the client. The encrypted session IDs assigned to the 4 sessions are given in Table 8.

Table 8. 212 character plain session ID and encrypted session ID

Session ID (Length=212)	Encrypted session ID
5E3602C889EA285A8E4C107ABAAF726B858D6B6E31C0D331D7BEC19EAF79E7755CA729C7B0C7FAD9FEFFBD0DF078DABC31C25982861DCFA24FA6495BD9AE4F5D3DAEF8324509A3B9700F997E11FFD5C16DBFFF4CD1025CFCCC1Ac4fbbc05a93e3518151277c6b2e09bd0	RQMQBJqJDFJa+V9tyRkR+i8DDJV520lkbYyZM8ghRbDtvX3B6tW6W7wyJ5dy6RGPjXwZpvJt2bTSLTNU0zCAzataQu3SeYx5eYxgURx/Xk4CO0HW752lze9mF2kmeWSg42QuT6gWdmmLr4GVmL0SzJ0TrGpUQeFEhksFoZPmzOkdaygNp5awq5bOLkDgliYNX2LHrPiivOjygqzw67EyNmQXRQN8Vuypm+I5uQqS335L9QmrRevfqbNpCIGugUNDwfGrQBjngG81jhZw2qCq0FDgdMG2h9Q0EBqiTg==
7C3096D9EEADB2917F003202AED4CDBF52B8D0644E5EAA3238C66249A277269696FC106346D8A4316ACBBE1326596134F4E5ED8FA1EBF017DC70B1DCC82C6ED2704E8E204DDFACCB760B3D26C4B9781BB24C3B2A795145A4C9ADcd7942012e8b304760f752e11d74f4af	5QH7fbKJDFIvyR7oj7yDXLWx8pOUFpO78u79d4KpWdiYG0YIZY9HTLtOW/Xo4g6jfFTWaDHbQ2Zyw6kU1Cm9RlenHdyGOGI60GeN4w78JCIsxxfKvBW1bRevkENs7nyhvVpf9PjDFvK/UxWLSvS6xDFr3ZxiGv/eUcAAcNINXsdYzKWfyEc07YzAn81QGCxWeH/cTK7fObmv0iEYxPlpt9IS3QD9mhkqDNs/Jb6SknjQXd31dpCCpdiYbcUFRR9s2OV2cA+2dIabm/Ugk/j93QU5T/9ezIL+gHbflw==
E7BEA07AE7E335297EEAB1F53FBD78896500D7CF1C3FF276775688F9A048FDF857C7799ADACA56BB0700EA2EE3FDFF18B3DA6F12EDEE35C671F93C8980F8F3154C3FA063A7581AA95BDFD7B7A2E5379CC672E265617743EEA42E3a2a3b156a16f19f6867dff42b8d3f4a	HAIk8cKJDFLSK1e3r+OVlC3XwUJljxCwDXuyQpL9Sg507NpeG+/whEFlVAbiA3P2ffBnx+UjBXFoEYnNxpD91CyzQb9HlVkwVLszYo6JE3bW3dCMV7+6cg2Mg5lMeXmHsAMAArMnDE7ehIxV6R3gy6aaYNBDBt7ttTRSj22FGBaLUVt9yAS8Bl/x2kZiNFiyu0NBJ8Lm83qefoDtifidH1AE7mAr+GGKvoOFR9qsMsuO344VgfyU1AgYMrV9LHr+/g/tno+5usH30SNoiFdhbjYmmXNs/3KHNQKpMA==
361295DE73EEB0D9CDF763BB6A53728DEC9BAA08E3E7EE2D2A47DCFD9A889A90AE6E24A0841A9FEBD70EA75074309C1D6498F6547A3E19F70A240DC73BB5E0651FB071BA2FA98DBBD4B6D56303E38D480FF8A4E9312DD9A3D990b666817e9c1ba30bc2fc46983061229a	swK8O9WJDFJ/NE1+P4jtpjenwKIJoWgpe+tH6TKj+gmsTM2FQDrI9QLB7P6wP/jz07TyqdqYqbH/Ec7/kf+ufanCiVlB+hCTDEBIJyqqh7JMxx/rqxl9oV05qVA7MkFUq7QIsm9A72yTTSOwMDm9Vf8R8Rg7nzjB29NIfZrcYn3sMPI4sSpu/KwpkWWjmhxyFA1B2ihHD0mrTBOkuvv+QJ201a3odaF2T6cCK1ycYgXGcYyPwnm1RXimr9jegCO8+jIbb8CN0VMvruHFEBONWVKoTcIpslM1ap52Tw==

The attacks are executed to capture the session ID. The integrity of the session ID is tested by creating 10 numbers of sessions, 20 sessions and 30 sessions in the web application between the client and server. The observed results for each session are given in Table 9.

Table 9. Results of 212 characters encrypted session ID

No	Metrics	50 chars encrypted Session ID		
		10 sessions	20 sessions	30 sessions
1	Number of unique session IDs generated	10	20	30
2	Number of session IDs attacked	0	0	0
3	Number of session IDs prevented	10	20	30
4	Session Hijack prevention rate	100 %	100 %	100 %

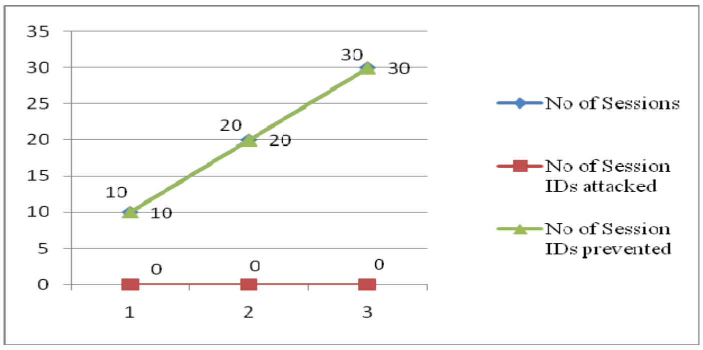


Fig. 8. No of session IDs prevented for 10, 20, 30 sessions

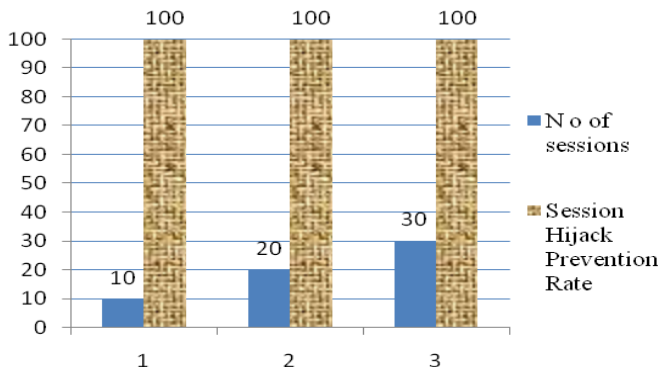


Fig. 9. Session Hijack prevention rate for 212 chars encrypted session ID

The results show that 212 characters encrypted session ID has a prevention rate of 100 % for 10 sessions, 100 % for 20 sessions and 100 % for 30 sessions.

4.3. Comparison of the session Hijack prevention rate

The session Hijack prevention rate is the ratio between the number of session IDs prevented to the total number of session IDs generated. Table 10 presents the session Hijack prevention rate.

Table 10. Session Hijack attack prevention rate

No	Approaches	Session Hijack attack prevention rate		
		10 sessions	20 sessions	30 sessions
1	32 characters encrypted session ID	90 %	90 %	94 %
2	92 characters encrypted session ID	100 %	95 %	97 %
3	212 characters encrypted session ID	100 %	100 %	100 %

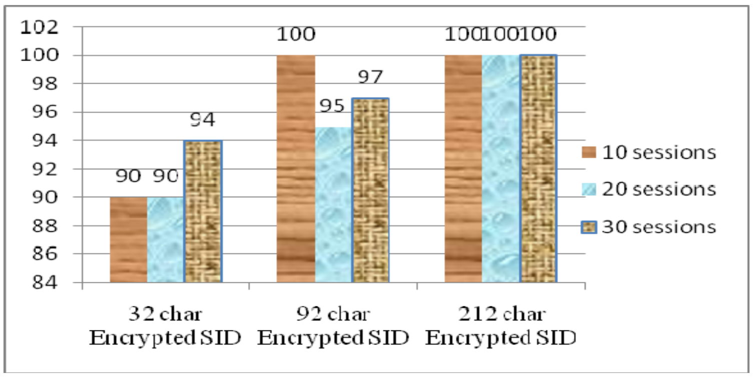


Fig. 10. Session Hijack prevention rate for 32, 92 and 212 chars encrypted session ID

Our experimental results proved that 212 characters of the encrypted Session ID has a session hijack prevention rate of 100 % for 10 sessions, 100 % for 20 sessions and 100 % for 30 sessions. So the 212 characters encrypted session ID completely prevents the session hijack attacks in wireless networks (Fig. 10).

5. Conclusion

Web application security becomes more important recently for the systems that are connected to wireless networks. The current web applications are weakly secured against session hijack attacks. In this paper we have proposed a strong and encrypted session ID to prevent the session hijack attacks. We have presented our approach by analysis of three different cases of encrypted session IDs of length 32 characters, 62 characters and 212 characters. We have tested the integrity of the session ID in a web application by establishing 10 sessions, 20 sessions and 30 sessions between the client and the server. The experimental results show that 212 characters of encrypted session ID completely prevents the session hijack attacks in wireless networks.

R e f e r e n c e s

1. A l e x, C h i n H u a n g, M o h a m e d. A Secure Cookie Protocol. – In: Proc. of IEEE Conference on Network Security, 2007, 333-338.
2. C h o m s i r i. HTTPS Hacking Protection. – In: Proc. of IEEE International Conference on Advanced Information Networking and Applications, 2007, 42-47.
3. M i y a z a k i, A. D. Online Privacy and the Disclosure of Cookie Use: Effects on Consumer Trust and Anticipated Patronage. – American Marketing Association, Vol. **27**, 2008, No 1, 19-33.
4. A d i d a, B. Session Lock: Securing Web Sessions Against Eavesdropping. – In: Proc. of International Conference on Web Client Security, China, 2008, 517-524.
5. J a c k s o n, C., A. B a r t h. Force HTTPS: Protecting High Security Websites from Network Attacks. – In: Proc. of International Conference on Web Client Security, China, 2008, 536-552.
6. J u e l s, A., T. M a r k u s. Cache Cookies for Browser Authentication. – In: Proc. of IEEE International Conference on Security and Privacy, 2008.

7. C h r i s t o p h e r, K i r d a. Pixy: A Static Analysis Tool for Detecting Web Application Vulnerabilities. – In: Proc. of IEEE International Conference on Security, 2009.
8. H o w a r d, M. Man-in-the Middle Attack to the HTTPS Protocol. – In: Proc. of IEEE Securiry and Privacy, 2009, 78-81.
9. G a w, S., E. W. F e l t e n. Password Management Strategies for Online Accounts. – In: Proc. of International Symposium on Usable Privacy and Security (SOUPS), Pittsburgh, USA, 2006, 44-55.
10. R i t c h i e, P. The Security Risks of AJAX / Web 2.0 Applications. Network Security, Secure Test, Ltd., UK, 2007.
11. B a r t h, A., C. J a c k s o n, J. M i t c h e l l. Robust Defenses for Cross Sire Request Forgery. – In: Proc. of ACM International Conference on CCS'08, Virginia, USA, 2008, 75-87.
12. A d i d a, B. Session Lock:Securing Web Sessions Against Eavesdropping. – In: Proc. of International World Wide Web Conference Committee (IW3C2), ACM, Beijing, China, 2008, 517-524.
13. W a n g, F., Y. Z h a n g. A New Provably Secure Authentication and Key Agreement Mechanisms for SIP Using Certificate Less Public Key Cryptography. 2008, 1-15.
14. P e r d i s c i, R., D. A r i u, P. G i a c i n t o, W. L e e. McPAD: A Multiple Classifier System for Accurate Payload Based Anomaly Detection. – Journal of Computer Networks, Vol. **53**, 2009, No 6, 864-881.
15. E i s e n, O. Catching the Fraudlent Man-in-the-Middle and Main-in-the-Browser. – Network Security, 2010, No 4, 11-12.
16. L i a o, Y i-P i n, S. S. W a n g. A New Secure Password Authenticated Key Agreement Scheme for SIP Using Self Certified Public Keys on Elliptic Curves. – Journal of Computer Communications, Vol. **33**, 2010, No 3, 372-380.
17. C i c h o n, J., Z. G o l e b i e w s k i, M. K u t y l o w s k i. From Key Pre-Distribution to Key Redistribution. – Journal of Theoretical Computer Science, Vol. **45**, 2012, No 3, 75-87.
18. C a r b o n e, A. R., L. C o m p a g n a, J. G i n c a r l o, S o r n i o t t i. An Authenication Flaw in Browser Based Single Sign on Protocols: Impact and Remediations. – Journal of Computers and Security, Vol. **33**, 2012, 41-58.
19. B i e l o v a, N. Survey on Java Script Security Policies and Their Enforcement Mechanisms in a Web Browser. – Journal of Logic and Algebraic Programming, Available Online from May 2013.
20. X i a n g, Y., X. S h i, J. W u, Z. W a n g, X. Y i n. Sign What You Really Care about Secure BGP AS-Paths Efficiently. **–** Journal of Computer Networks, Vol. **57**, 2013, No 10, 2250-2265.
21. D o k e v, N., I. B l a g o e v. An Approach for Automatic Transmission of Authenticated Data over Computer Networks. – Cybernetics and Information Technologies, Vol. **11**, 2011, No 2, 65-82.
22. P e n c h e v a, E., I. A t a n a s o v. Open Access to Call and Session Control in Mobile Networks. – Cybernetics and Information Technologies, Vol. **10**, 2010, No 1, 49-63.