WebBedlam Simple Single Sign On

Purpose of this document

- Provide technical details of this solution for
  - Scrutiny by security experts
  - Implementation
  - Testing
- Provide some example implementations/algorithms

Technical Specifications

	Details	Example
Internal Format	This is the format of the data inside the encrypted token. The MIME type is `application/x-www-form-urlencoded` Must have: • "email" for uniquely identifying a user • "timestamp" for validity checking May have any number of other fields needed for the integration.	`fname=Michael&lname=Kenney&email=mkenney@webbedlam.com&timestamp=2007-12-10T22:01:57Z`
External Format	The format passed by the browser to the site to be authenticated with is encoded in `application/x-www-form-urlencoded` format as well (the default for HTTPS POST operations) Must have: • "token" the encrypted token with the initialization vector prepended, base 64 encoded	`token=tViT0CCe+56KSD1MdyfL7GvizYk0j7dmqCFFW9VYfjqrqgXazWTDf2P47qRqCK1ALiJQ7yZ7q0L5g0uY/livyI5ir4TcEw+O1wr4mp1SJbuoCYsOl9XkGu9CDz7hsDNrCEEUF8z1vhkB9cKrlezUBDNkucKusFwVp+QpVyhkNK334QNjqKM8vToWOjDI///c`

Encryption Standards	• Algorithm: AES (Rijndael)
	• Key size: 256 bit
	• Cipher mode: CBC (Cipher block chaining)
	• Padding PKCS#7,#5 (identical for 128/256bit ciphers) is acceptable but not required
	• Initialization Vector should be randomly generated and must be passed on all requests
		`AES-256-CBC`
		
		`AES256/CBC/PKCS5Padding`
		
		`Rijndael-256-CBC`

Hashing Standards	• Algorithm: SHA-256	`SHA256`
		
		`SHA-256`
		
		`SHA2-256`

Algorithms

Encryption

1. Get ‘email’ and ‘timestamp’.
   - ‘timestamp’ is the current UTC in ISO8601.
   - ‘email’ is a unique user Id in email address format uniquely identifies a user's account. It must be a valid email address.
     - Example:
       - email: `mkenney@webbedlam.com`
       - timestamp: `2007-12-10T22:01:57Z`

2. URL encode timestamp and email and build internal query string (www form encoding)
   - Example: `email=mkenney%40webbedlam.com.com&timestamp=2011-01-01T12%3A00%3A00Z`

3. Calculate SHA256 hash and append
   - SHA256: `É'_=_°¥Þ¥´ÏÌGôO%Õoùb .Ã1••Œ‘üÅŠ`

4. Generate random IV, 128 bits
   - Use secure random generator
   - Example: `µO¼rÙEðX1]‘dì&"`

5. Find the KEY for partner in configuration
   - Recommend randomly generated 1024 bit ASCII string
     - Example:
       `c4ca4238a0b923820dcc509a6f75849bc81e728d9d4c2f636f067f89cc14862ceccbc87e4b5ce2fe28308fd9f2a7baf3a87ff679a2f3e71d9181a67b7542122c`

6. Apply encryption
   - Example: `MBµ"„__ræ_~™Zkôj_÷Ú__t6f=âpÌ&RŠ¯_wØ}„êY ¿_£VC`ÌT•__ „™’%5uµ£®<_-(Øzq¯÷šÏä‘Ó{-Hý©öü|wâG_8»o_q We_ë‘JåÓBæî~ö—7œd÷'¼â^f_©__4H—‘+8óå$,> †°<ÒCø_`

7. Prepend IV, then base64 encode
   - Example:
     ```
     gXxom8nY9pbq7fdQEFe6BBIbTUK1nZOEERly5h1+mVpr9GoG99oaGHQ2Zj3icMwmUoqvHHfYfYTqWQm/HqNWQ2DMVJWQAgUNhJmSJTV1taOuPBetKNh6ca/3ms/kkdN7lkj9qfb8fHfiRw84u28IcQlXZR3rkUrl00Lm7n72lzecZPcnvOKPiGYOqRsaNEiXkSs48+Ukgj6ghro80kP4FQ==
     ```

8. Build form for browser
   - Example:
     ```
     <html><head><title>Login</title></head>
     <body onload="document.form.authform.submit()">
     If you are not logged in momentarily, please click the button
     <form name="authform" action="https://sso.thirdparty.com/lexmark">
        <input type="hidden" name="token"
     value="gXxom8nY9pbq7fdQEFe6BBIbTUK1nZOEERly5h1+mVpr9GoG99oaGHQ2Zj3icMwmUoqvHHfYfYTqWQm/HqNWQ2DMVJWQ
     AgUNhJmSJTV1taOuPBetKNh6ca/3ms/kkdN7lkj9qfb8fHfiRw84u28IcQlXZR3rkUrl00Lm7n72lzecZPcnvOKPiGYOqRsaNEi
     XkSs48+Ukgj6ghro80kP4FQ==" />
        <button onclick="document.form.authform.submit();">Login</button>
     <form>
     </body>
     </html>
     ```

Decryption

1. Take the current timestamp
   - Example: `Dec 10, 2007 10:02:00PM GMT`

2. Get token from post variables, remove url encoding (done automatically by most systems) and base64 decode
   - Encoded:
     `gXxom8nY9pbq7fdQEFe6BBIbTUK1nZOEERly5h1+mVpr9GoG99oaGHQ2Zj3icMwmUoqvHHfYfYTqWQm/HqNWQ2DMVJWQAgUNhJm SJTV1taOuPBetKNh6ca/3ms/kkdN7lkj9qfb8fHfiRw84u28IcQlXZR3rkUrl00Lm7n72lzecZPcnvOKPiGYOqRsaNEiXkSs48+ Ukgj6ghro80kP4FQ==`
   - After decode: `|h›ÉØö-êí÷P_W°OMBµ"„__ræ_~™Zkôj_÷Ú__t6f=âpÌ&RŠ¯_wØ}„êY ¿_£VC`ÌT•□__ „™'%5uµ£®<_-(Øzq¯÷šÏä'Ó{-Hý©öü|wâG_8»o_q We_ë'JåÓBæî~ö—7œd÷'¼â^f_©__4H—'+8óå$,> †°<ÒCø_`

3. Split the IV from the Encrypted portion. The IV will be the first 128bits
   - IV: `|h›ÉØö-êí÷P_W°O`
   - Encrypted: `MBµ□"„__ræ_~™Zkôj_÷Ú__t6f=âpÌ&RŠ¯_wØ}„êY ¿_£VC`ÌT•□__ „™'%5uµ£®<_-(Øzq¯÷šÏä'Ó{-Hý©öü|wâG_8»o_q We_ë'JåÓBæî~ö—7œd÷'¼â^f_©__4H—'+8óå$,> †°<ÒCø_`

4. Get the shared key
   - Example:
     `c4ca4238a0b923820dcc509a6f75849bc81e728d9d4c2f636f067f89cc14862ceccbc87e4b5ce2fe28308fd9f2a7baf3a87 ff679a2f3e71d9181a67b7542122c`

5. Decrypt the token using the IV and key from steps 3 and 4:
   - Example: `fname=Michael&lname=Kenney&email=mkenney%40webbedlam.com.com&timestamp=2011-01-01T12%3A00%3A00ZÀ‹ÚÚbªã¾aÓØ^®7¨:Ö…6ÏRÎ¦_Ó&Ã_=,Ó_`

6. Split the packet from the hash by removing the last 256bits as the hash
   - Packet: `fname=Michael&lname=Kenney&email=mkenney%40webbedlam.com.com&timestamp=2011-01-01T12%3A00%3A00Z`
   - Hash: `À‹ÚÚbªã¾aÓØ^®7¨:Ö…6ÏRÎ¦_Ó&Ã_=,Ó_`

7. Hash the packet with SHA256 and compare to the hash that was packaged with the packet:
   - Calculated Hash: `À‹ÚÚbªã¾aÓØ^®7¨:Ö…6ÏRÎ¦_Ó&Ã_=,Ó_`

8. Parse query string (split on &, then divide key/value pairs by splitting on =, then url decode values)
   - Example:
     - fname => `Michael`
     - lname => `Kenney`
     - email => `mkenney@webbedlam.com`
     - timestamp => `2011-01-01T12:00:00Z`

9. Parse time according to ISO8601 specification
   - Example: timestamp = January 01, 2011 12:00:00 AM GMT

10. Compare to first timestamp, if within 5 minutes, then accept
    - Example: original timestamp – current timestamp = 3 seconds
      - This is an acceptable offset

See attached code sample