SHASUB

*PROTOTYPE: Passphrase Secured **SHA**sum **SUB**set*

Author: Horvath, Attila

Version: 0.1

Date: 2022 / 06 / 16

The topic of this paper is a mechanism for embedding a functional equivalent SHASUM value in target file(s), termed SHASUB. This mechanism renders the contents of target file(s) to be reliably verifiable – comparable to current typical usage of SHASUM. SHASUB is as an alternate methodology of validating source files' contents.

NB: SHASUB, as described herein, is an embedded 'in-band' mechanism as opposed to SHASUM's out-of-band mechanism. As such the applicability of this prototype is limited to ASCII (textual) files whose content may be altered without affecting the significance and relevance of its payload content.

1 Introduction

The SHASUM mechanism is a common and ubiquitous methodology by which to reliably validate and confirm the 'finger prints' of the data contents of any file[1,2] – meaning the SHASUM of a file's contents can be used to determine whether the contents of a file has changed, or not[3], since the SHASUM was originally calculated.

SHASUB, as it relates to ASCII data files, addresses following drawbacks with the ubiquitous SHASUM usage for a subset of data types:

1. version control
2. discrete 'file pairing'
3. perpetual/recursive paradox. SHASUB addresses both of these issues.

1.1 Version Control

By way of example, Subversion [SVN] offers a feature termed 'keyword(s) substitution' – particularly useful in configuration management environments, typically software development, whereby the tool's feature permits versioning information to be automatically imprinted inside select files in user selected locations within respective files. This 'keyword(s) substitution' feature takes effect automatically when files enabled with this property are committed into the versioning repository/database.

As useful as SVN's 'keyword(s) substitution' feature is, due to SHASUM's inherent behavior it follows that the SHASUM value of a file cannot be calculated before files are committed into the versioning repository because the files are modified during the commit procedure rendering a pre-calculated SHASUM value moot. Therefore the SHASUM value of a committed file is required to be generated post-commit which means the SHASUM of a file inherently cannot be committed with the file – the SHASUM is required to be committed subsequently at a version greater/beyond that of the file itself.

SHASUB addresses this dilemma permitting the functional equivalent of SHASUM to be generated pre-commit.

1.2 Discrete 'File-Pairing'

When used as a coupled 'file pair' information construct, a file's contents and the corresponding calculated SHASUM must be kept as a 'file pair' – i.e. two discrete files (see section 1.4)[4]. If the 'file pair' is conveyed to a remote destination and the SHASUM is lost inadvertently, there may not be a reliable way of reconstituting the file's original SHASUM because the contents of the file may have changed subsequent to the loss of the SHASUM.

1.3 The Paradox

A practical way around the discrete 'file pair' mechanism is to merge the information by embedding the SHASUM of a file's contents within the file itself[5] alongside a file's contents without affecting the

[1] To include files with binary content.
[2] https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html
[3] In the event that SHASUM fails validation, it neither provides a mechanism with which to determine what has changed; nor does it provide a mechanism to revert to a SHASUM validated state.
[4] It is recognized that the 'file pair' may be packaged in archive file format.
[5] This instantiation of SHASUB works only for textual ASCII files and not for binary files.

significance of files' payload contents. Embedding a typical SHASUM, however, presents a perpetual/recursive paradox owing to the fact that once a file's SHASUM, when calculated in the usual manner across files' entire contents, is embedded within the file, it explicitly changes the file's content rendering the embedded SHASUM moot as it no longer can be used to validate the file's original payload contents because it includes itself.

1.4 Modus Operandi

Typical Sender/Receiver 'modus operandi' is for a sender to convey the validity of files' contents to a receiver as a 'file pair' comprised of the original file itself as well as its corresponding SHASUM value. The sender conveys this file pair to a receiver to validate on receipt to ensure the contents of the source file has not been compromised either inadvertently |OR| intentionally:

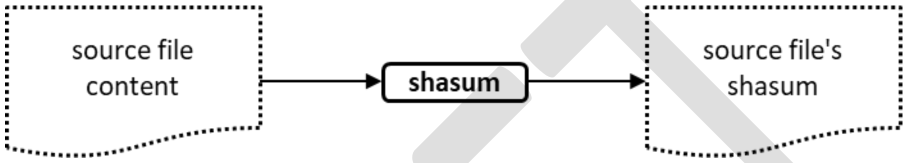


Fig. 1: Sender 'Modus Operandi'

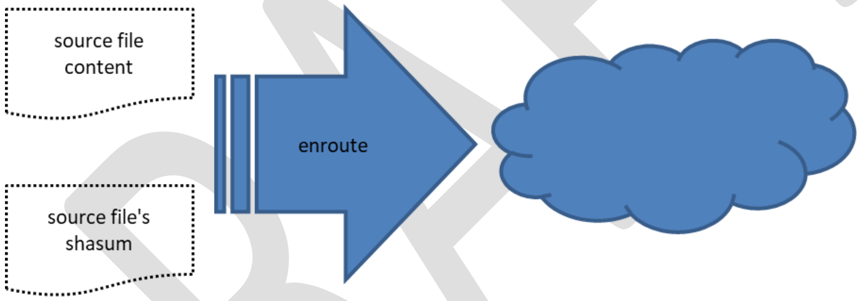


Fig. 2: File Pair In Transit

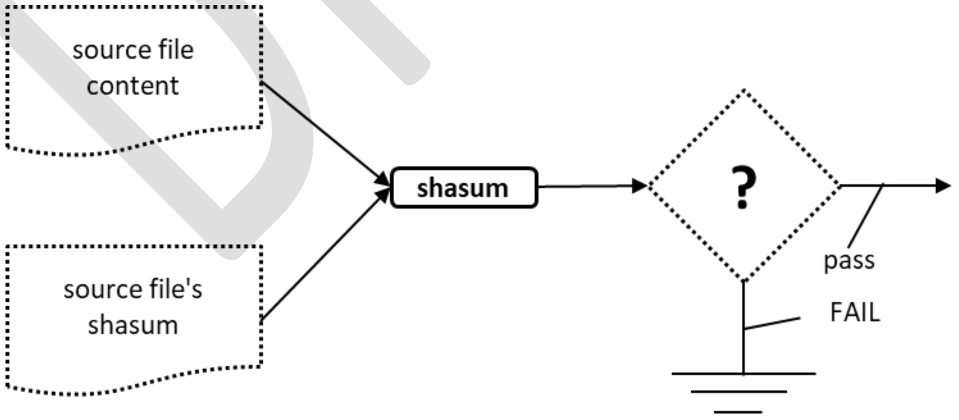


Fig. 3: Receiver 'Modus Operandi'

It is intuitively obvious from the Sender/Receiver 'modus operandi' depicted above, if a file encounters MIM [man in the middle] attack or a receiver corrupts or loses the SHASUM and/or corrupts a corresponding file’s payload data, only the sender at the originating end can reliably re-instantiate the file-pair.

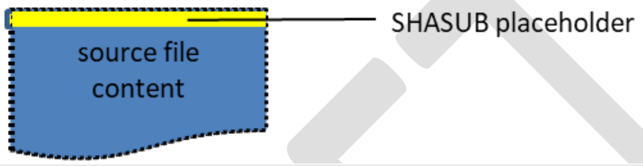
2 SHASUB

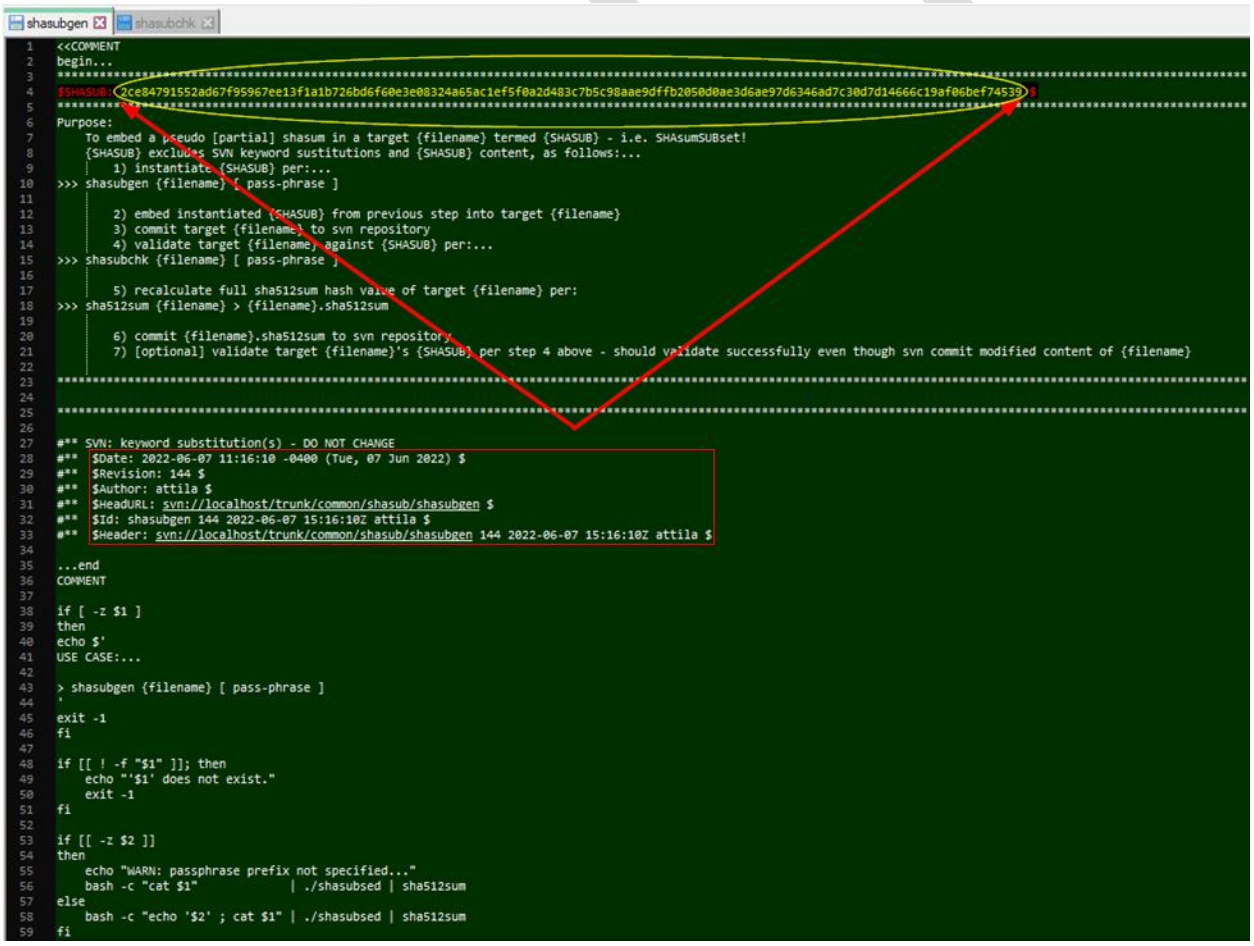
The term "SHASUB" means the "SHAsum of a data SUBset". Its usage is both a placeholder as well as a mechanism[6] for eliminating the 'file pair' and 'paradox' issues discussed above.

2.1 SHASUB Mechanism

2.1.1 SHASUB Placeholder

The term "SHASUB" when used as a placeholder is an unspecified arbitrary location in a file's content that identifies the location where a calculated 'partial SHASUM' value is manually stored so as not to corrupt the file's payload information[7] as illustrated below:





```
<<COMMENT
begin...
*****************************************************************************************************************************************
$SHASUB:2ce84791552ad67f95967ee13f1a1b726bd6f60e3e08324a65ac1ef5f0a2d483c7b5c98aae9dffb2050d0ae3d6ae97d6346ad7c30d7d14666c19af06bef74539$
*****************************************************************************************************************************************
Purpose:
    To embed a pseudo [partial] shasum in a target {filename} termed {SHASUB} - i.e. SHAsumSUBset!
    {SHASUB} excludes SVN keyword sustitutions and {SHASUB} content, as follows:...
    |    1) instantiate {SHASUB} per:...
>>> shasubgen {filename} [ pass-phrase ]
    |
    |    2) embed instantiated {SHASUB} from previous step into target {filename}
    |    3) commit target {filename} to svn repository
    |    4) validate target {filename} against {SHASUB} per:...
>>> shasubchk {filename} [ pass-phrase ]
    |
    |    5) recalculate full sha512sum hash value of target {filename} per:
>>> sha512sum {filename} > {filename}.sha512sum
    |
    |    6) commit {filename}.sha512sum to svn repository
    |    7) [optional] validate target {filename}'s {SHASUB} per step 4 above - should validate successfully even though svn commit modified content of {filename}
    |
*****************************************************************************************************************************************

*****************************************************************************************************************************************

#** SVN: keyword substitution(s) - DO NOT CHANGE
#**  $Date: 2022-06-07 11:16:10 -0400 (Tue, 07 Jun 2022) $
#**  $Revision: 144 $
#**  $Author: attila $
#**  $HeadURL: svn://localhost/trunk/common/shasub/shasubgen $
#**  $Id: shasubgen 144 2022-06-07 15:16:10Z attila $
#**  $Header: svn://localhost/trunk/common/shasub/shasubgen 144 2022-06-07 15:16:10Z attila $

...end
COMMENT

if [ -z $1 ]
then
echo $'
USE CASE:...

> shasubgen {filename} [ pass-phrase ]
'
exit -1
fi

if [[ ! -f "$1" ]]; then
    echo "'$1' does not exist."
    exit -1
fi

if [[ -z $2 ]]
then
    echo "WARN: passphrase prefix not specified..."
    bash -c "cat $1"              | ./shasubsed | sha512sum
else
    bash -c "echo '$2' ; cat $1" | ./shasubsed | sha512sum
fi
```

Fig. 4: SHASUB Placeholder

[6] Incorporates accompanying generation/validation tools.

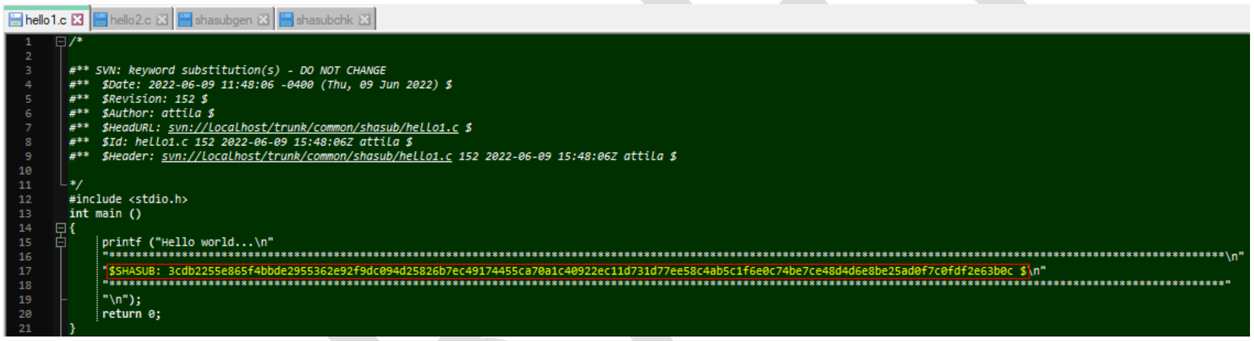
[7] Normally located within in a comment field, print statement, initialization field of a string variable, etc.

As indicated in RED above, the SHASUB value itself, indicated in YELLOW above, is delimited by the string '$SHASUB:ƀ' at the beginning and the string 'ƀ$' at the end where the symbol 'ƀ' is used to represent a blank/space character[8]. The delimiter enclosed SHASUB value itself is a 128 character SHASUM hexadecimal numerical value instantiated by the 'shasubgen' utility – see section 2.1.2 below 'SHASUB Instantiation/Generation'.

The SHASUB placeholder may appear anywhere in an ASCII file as long as [1]it is appropriately delimited and [2]it does not adversely alter the file's payload content. When encountered however, the content contained within delimiters is excluded from the SHASUB calculations – this is true for both generation and validation operations.

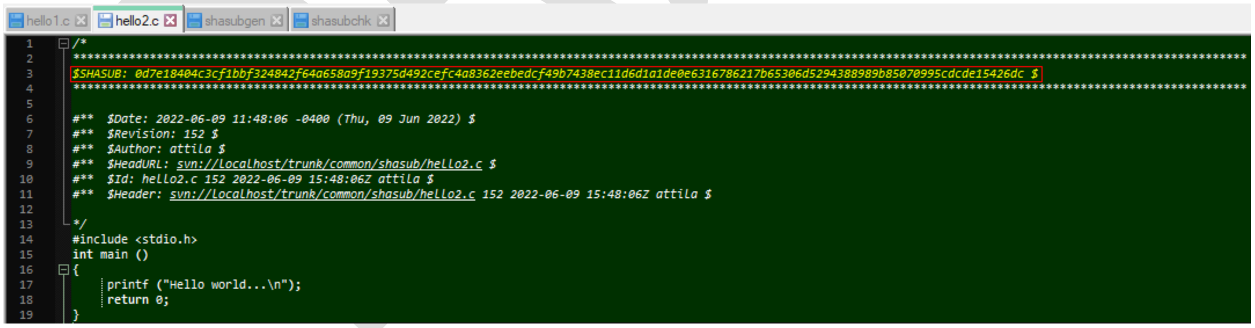
NB: SHASUB similarly addresses issues with Subversion keyword substitution feature per lines #28 thru #33 in Fig. 4: above. Files enabled for Subversion's keyword substitution feature require their SHASUM to be calculated after commits, not before – see 2.2.1 below for full discussion and 2.3 below for related recommendation.

2.1.1.1 Examples: Locating SHASUB Placeholders



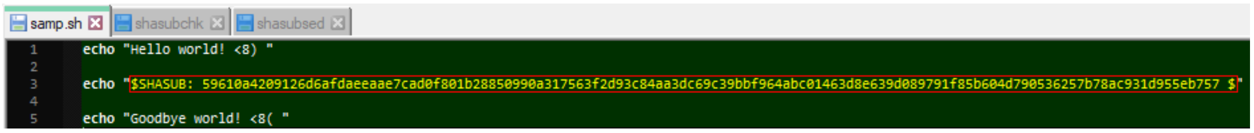
```
/*

#** SVN: keyword substitution(s) - DO NOT CHANGE
#**  $Date: 2022-06-09 11:48:06 -0400 (Thu, 09 Jun 2022) $
#**  $Revision: 152 $
#**  $Author: attila $
#**  $HeadURL: svn://localhost/trunk/common/shasub/hello1.c $
#**  $Id: hello1.c 152 2022-06-09 15:48:06Z attila $
#**  $Header: svn://localhost/trunk/common/shasub/hello1.c 152 2022-06-09 15:48:06Z attila $

*/
#include <stdio.h>
int main ()
{
    printf ("Hello world...\n"
    "****************************************************************************************************************************************************************\n"
    "$SHASUB: 3cdb2255e865f4bbde2955362e92f9dc094d25826b7ec49174455ca70a1c40922ec11d731d77ee58c4ab5c1f6e0c74be7ce48d4d6e8be25ad0f7c0fdf2e63b0c $\n"
    "****************************************************************************************************************************************************************"
    "\n");
    return 0;
}
```

Fig. 5: Program File – Printf() Format Statement



```
/*
*******************************************************************************************************************************************************************
$SHASUB: 0d7e18404c3cf1bbf324842f64a658a9f19375d492cefc4a8362eebedcf49b7438ec11d6d1a1de0e6316786217b65306d5294388989b85070995cdcde15426dc $
*******************************************************************************************************************************************************************

#**  $Date: 2022-06-09 11:48:06 -0400 (Thu, 09 Jun 2022) $
#**  $Revision: 152 $
#**  $Author: attila $
#**  $HeadURL: svn://localhost/trunk/common/shasub/hello2.c $
#**  $Id: hello2.c 152 2022-06-09 15:48:06Z attila $
#**  $Header: svn://localhost/trunk/common/shasub/hello2.c 152 2022-06-09 15:48:06Z attila $

*/
#include <stdio.h>
int main ()
{
    printf ("Hello world...\n");
    return 0;
}
```

Fig. 6: Program File – Comment Block



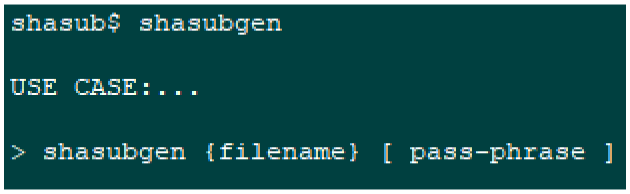
```
echo "Hello world! <8) "

echo "$SHASUB: 59610a4209126d6afdaeeaae7cad0f801b28850990a317563f2d93c84aa3dc69c39bbf964abc01463d8e639d089791f85b604d790536257b78ac931d955eb757 $"

echo "Goodbye world! <8( "
```

Fig. 7: BASH Script – Echo Block

[8] This presupposes that the SHASUB value's delimiters are unique character strings not otherwise found in files' payload contents.

2.1.2 SHASUB Instantiation/Generation

The instantiation of a file content's SHASUB entails the usage of the 'shasubgen' utility:



```
shasub$ shasubgen

USE CASE:...

> shasubgen {filename} [ pass-phrase ]
```

Fig. 8: 'shasubgen' Usage

Initial incorporation of SHASUB is a three step process. Subsequently however, only steps 2 and 3 below are required where an old/obsolete SHASUB value is replaced with a new/current SHASUB value:

1. Establishing the SHASUB placeholder
2. Instantiation of SHASUB value via 'shasubgen' utility
3. Embedding (incorporating) the SHASUB value in the SHASUB placeholder[9]

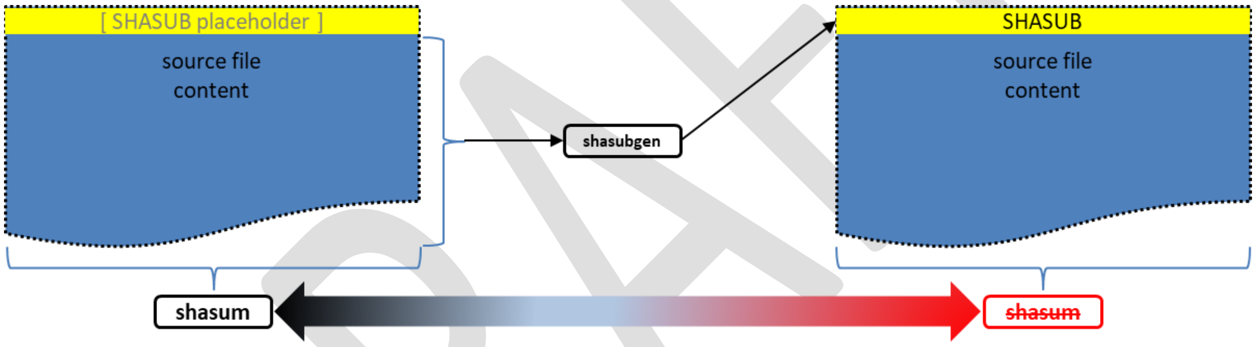
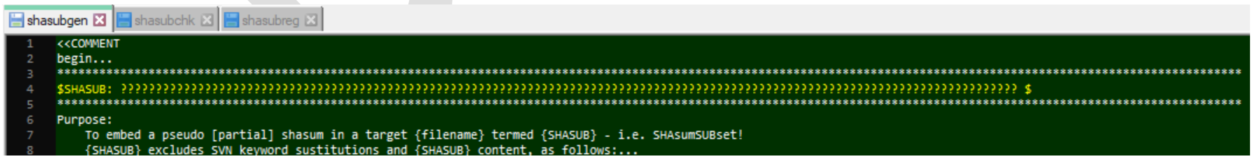


Fig. 9: SHASUB Incorporation

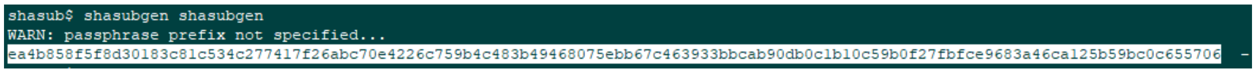
NOTE: The SHASUM value 'shasum' of a file's whole content calculated prior to embedding a SHASUB value will **not** match the SHASUM value '~~shasum~~' of the same file's whole content after the embedding procedure as depicted in Fig. 9: above.

Case in point, embedding 'shasubgen' utility's SHASUB value into the utility is as follows – see Fig. 10: through Fig. 12: below:



```
<<COMMENT
begin...
**************************************************************************************************************************
$SHASUB: ???????????????????????????????????????????????????????????????????????????????????????????????????????????????? $
**************************************************************************************************************************
Purpose:
    To embed a pseudo [partial] shasum in a target {filename} termed {SHASUB} - i.e. SHAsumSUBset!
    {SHASUB} excludes SVN keyword sustitutions and {SHASUB} content, as follows:...
```

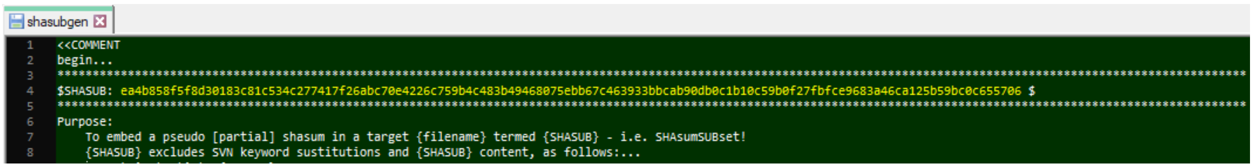
Fig. 10: Establishing SHASUB placeholder



```
shasub$ shasubgen shasubgen
WARN: passphrase prefix not specified...
ea4b858f5f8d30183c81c534c277417f26abc70e4226c759b4c483b49468075ebb67c463933bbcab90db0c1b10c59b0f27fbfce9683a46ca125b59bc0c655706  -
```

Fig. 11: Instantiate SHASUB value

[9] Embedding SHASUB is a manual procedure because it requires owner's knowledge of content to determine appropriate location that does not negatively impact the significance of the file's payload.

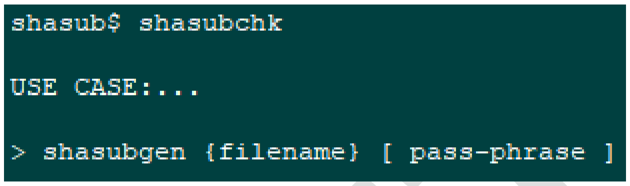

shasubgen

```
<<COMMENT
begin...
****************************************************************************************************************************************
$SHASUB: ea4b858f5f8d30183c81c534c277417f26abc70e4226c759b4c483b49468075ebb67c463933bbcab90db0c1b10c59b0f27fbfce9683a46ca125b59bc0c655706 $
****************************************************************************************************************************************
Purpose:
    To embed a pseudo [partial] shasum in a target {filename} termed {SHASUB} - i.e. SHAsumSUBset!
    {SHASUB} excludes SVN keyword sustitutions and {SHASUB} content, as follows:...
```

Fig. 12: Embed SHASUB value

2.1.3 SHASUB Validation

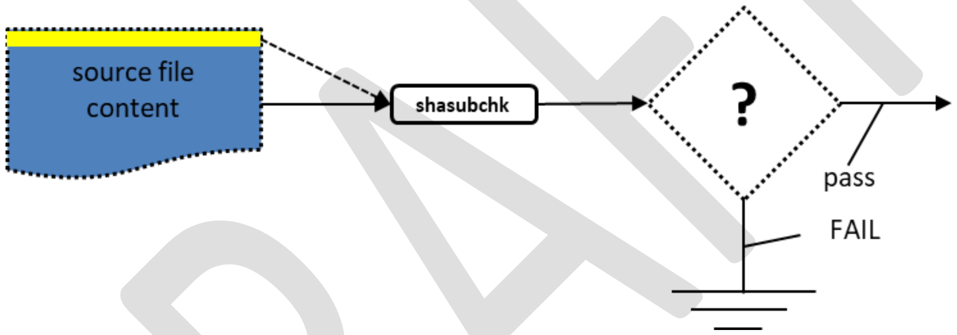
The validation of a file content via embedded SHASUB value entails the usage of the ‘shasubchk’ utility:



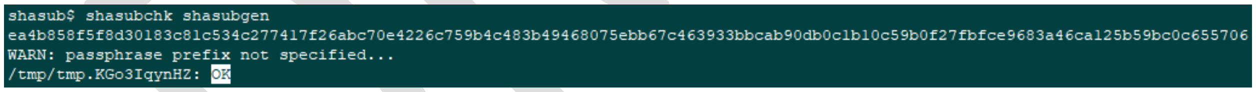
```
shasub$ shasubchk

USE CASE:...

> shasubgen {filename} [ pass-phrase ]
```

Validating the SHASUB of a file’s contents is a one step process:

1. Validate the SHASUB value via ‘shasubchk’ utility



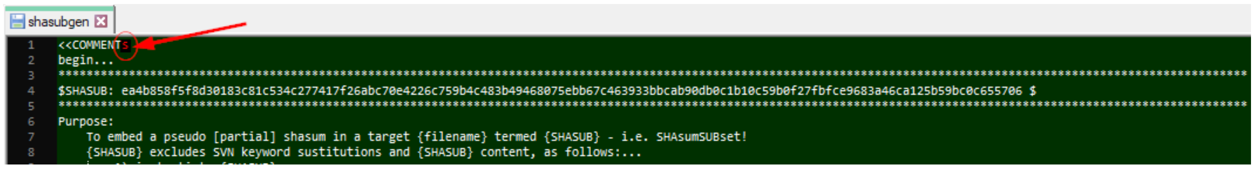
Case in point, invoking ‘shasubchk’ utility referencing {filename} locates the embedded SHASUB placeholder per specified delimiters (see 2.1.1 above ‘SHASUB Placeholder’) and validates the file’s corresponding SHASUB value:



```
shasub$ shasubchk shasubgen
ea4b858f5f8d30183c81c534c277417f26abc70e4226c759b4c483b49468075ebb67c463933bbcab90db0c1b10c59b0f27fbfce9683a46ca125b59bc0c655706
WARN: passphrase prefix not specified...
/tmp/tmp.KGo3IqynHZ: OK
```

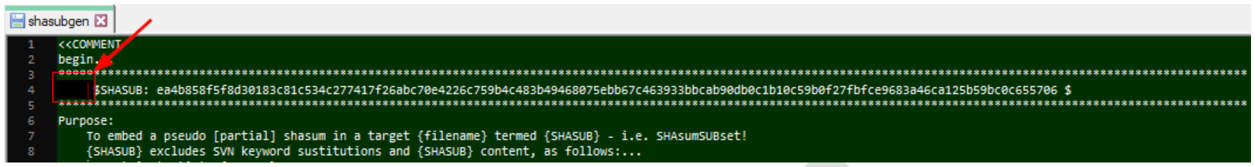
Fig. 13: Validate File Contents Per SHASUB value

NOTE: In the event that a file’s content is modified and/or corrupted, subsequent attempts to validate the file’s contents per embedded SHASUB value will fail – see Fig. 14: through Fig. 15: below.



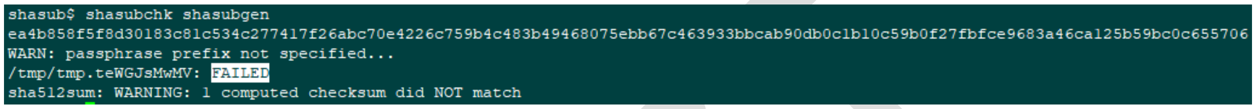
```
<<COMMENTs
begin...
*****************************************************************************************************************************
$SHASUB: ea4b858f5f8d30183c81c534c277417f26abc70e4226c759b4c483b49468075ebb67c463933bbcab90db0c1b10c59b0f27fbfce9683a46ca125b59bc0c655706 $
*****************************************************************************************************************************
Purpose:
    To embed a pseudo [partial] shasum in a target {filename} termed {SHASUB} - i.e. SHAsumSUBset!
    {SHASUB} excludes SVN keyword sustitutions and {SHASUB} content, as follows:...
```

/ OR /



```
<<COMMENT
begin.
*****************************************************************************************************************************
     $SHASUB: ea4b858f5f8d30183c81c534c277417f26abc70e4226c759b4c483b49468075ebb67c463933bbcab90db0c1b10c59b0f27fbfce9683a46ca125b59bc0c655706 $
*****************************************************************************************************************************
Purpose:
    To embed a pseudo [partial] shasum in a target {filename} termed {SHASUB} - i.e. SHAsumSUBset!
    {SHASUB} excludes SVN keyword sustitutions and {SHASUB} content, as follows:...
```

Fig. 14: File Payload Contents Corrupted

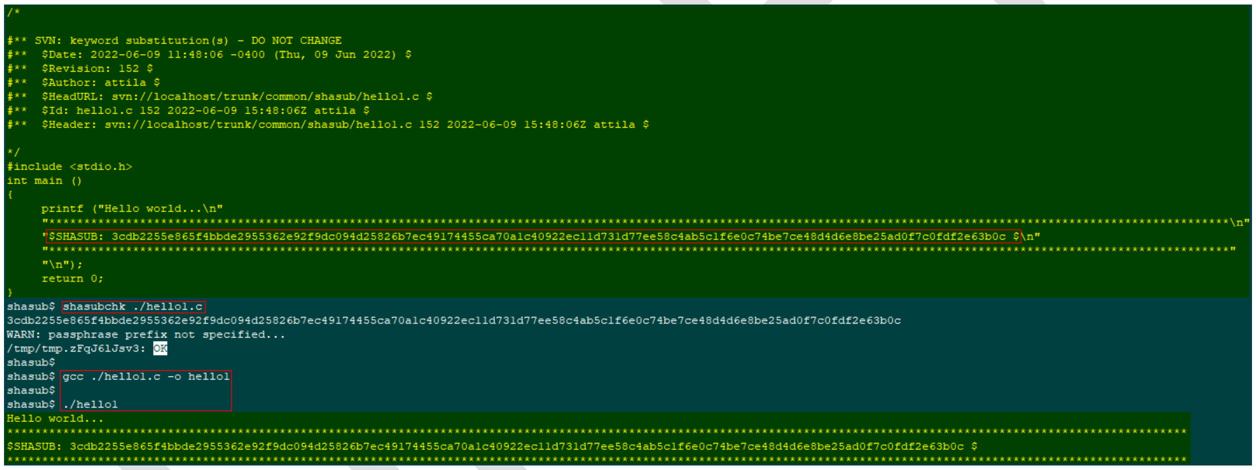


```
shasub$ shasubchk shasubgen
ea4b858f5f8d30183c81c534c277417f26abc70e4226c759b4c483b49468075ebb67c463933bbcab90db0c1b10c59b0f27fbfce9683a46ca125b59bc0c655706
WARN: passphrase prefix not specified...
/tmp/tmp.teWGJsMwMV: FAILED
sha512sum: WARNING: 1 computed checksum did NOT match
```

Fig. 15: SHASUB Validation Failure Per Embedded SHASUB Value

2.1.4 SHASUB: Use Cases

2.1.4.1 hello1.c

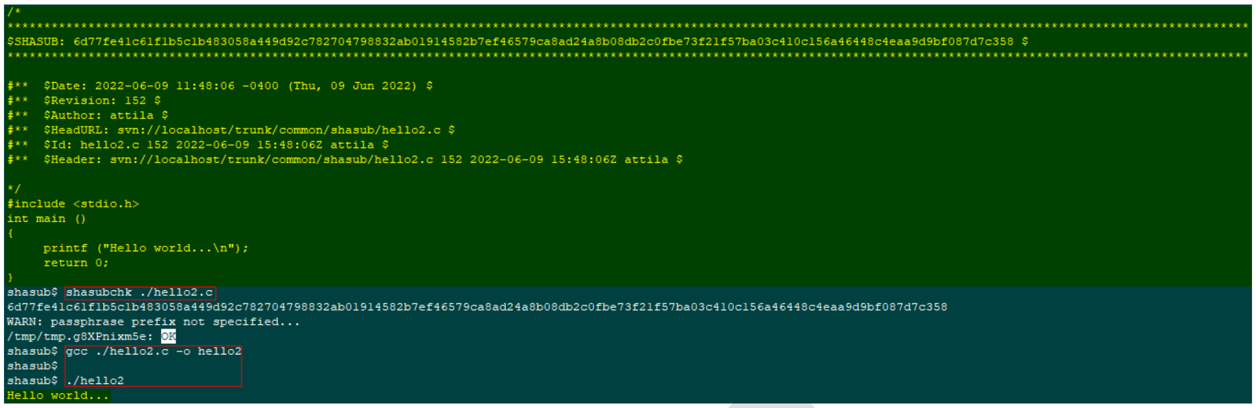


```
/*

#** SVN: keyword substitution(s) - DO NOT CHANGE
#**  $Date: 2022-06-09 11:48:06 -0400 (Thu, 09 Jun 2022) $
#**  $Revision: 152 $
#**  $Author: attila $
#**  $HeadURL: svn://localhost/trunk/common/shasub/hello1.c $
#**  $Id: hello1.c 152 2022-06-09 15:48:06Z attila $
#**  $Header: svn://localhost/trunk/common/shasub/hello1.c 152 2022-06-09 15:48:06Z attila $

*/
#include <stdio.h>
int main ()
{
     printf ("Hello world...\n"
     "****************************************************************************************************************************\n"
     "$SHASUB: 3cdb2255e865f4bbde2955362e92f9dc094d25826b7ec49174455ca70a1c40922ec11d731d77ee58c4ab5c1f6e0c74be7ce48d4d6e8be25ad0f7c0fdf2e63b0c $\n"
     "****************************************************************************************************************************"
     "\n");
     return 0;
}
shasub$ shasubchk ./hello1.c
3cdb2255e865f4bbde2955362e92f9dc094d25826b7ec49174455ca70a1c40922ec11d731d77ee58c4ab5c1f6e0c74be7ce48d4d6e8be25ad0f7c0fdf2e63b0c
WARN: passphrase prefix not specified...
/tmp/tmp.zFqJ6lJsv3: OK
shasub$
shasub$ gcc ./hello1.c -o hello1
shasub$
shasub$ ./hello1
Hello world...
****************************************************************************************************************************
$SHASUB: 3cdb2255e865f4bbde2955362e92f9dc094d25826b7ec49174455ca70a1c40922ec11d731d77ee58c4ab5c1f6e0c74be7ce48d4d6e8be25ad0f7c0fdf2e63b0c $
****************************************************************************************************************************
```

Fig. 16: ‘hello1.c’

Fig. 16: above illustrates SHASUB unintrusively embedded in printf() statement.

2.1.4.2 hello2.c

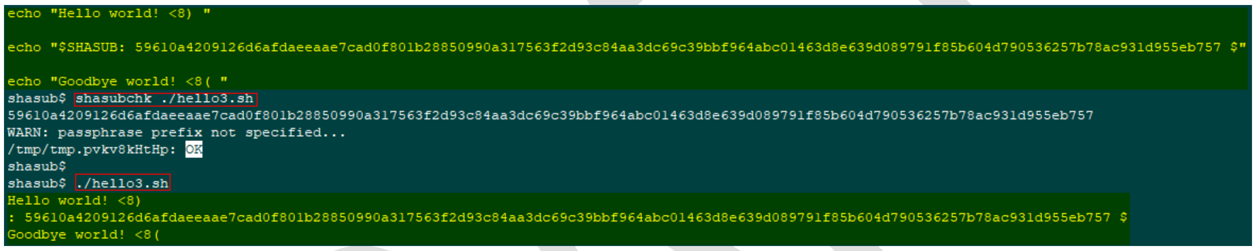


```
/*
*************************************************************************************************************************************************************************
$SHASUB: 6d77fe41c61f1b5c1b483058a449d92c782704798832ab01914582b7ef46579ca8ad24a8b08db2c0fbe73f21f57ba03c410c156a46448c4eaa9d9bf087d7c358 $
*************************************************************************************************************************************************************************

#**  $Date: 2022-06-09 11:48:06 -0400 (Thu, 09 Jun 2022) $
#**  $Revision: 152 $
#**  $Author: attila $
#**  $HeadURL: svn://localhost/trunk/common/shasub/hello2.c $
#**  $Id: hello2.c 152 2022-06-09 15:48:06Z attila $
#**  $Header: svn://localhost/trunk/common/shasub/hello2.c 152 2022-06-09 15:48:06Z attila $

*/
#include <stdio.h>
int main ()
{
     printf ("Hello world...\n");
     return 0;
}
shasub$ shasubchk ./hello2.c
6d77fe41c61f1b5c1b483058a449d92c782704798832ab01914582b7ef46579ca8ad24a8b08db2c0fbe73f21f57ba03c410c156a46448c4eaa9d9bf087d7c358
WARN: passphrase prefix not specified...
/tmp/tmp.g8XPnixm5e: OK
shasub$ gcc ./hello2.c -o hello2
shasub$
shasub$ ./hello2
Hello world...
```

Fig. 17: ‘hell02.c’

Fig. 17: above illustrates SHASUB unintrusively embedded in comment block

2.1.4.3 hello3.sh

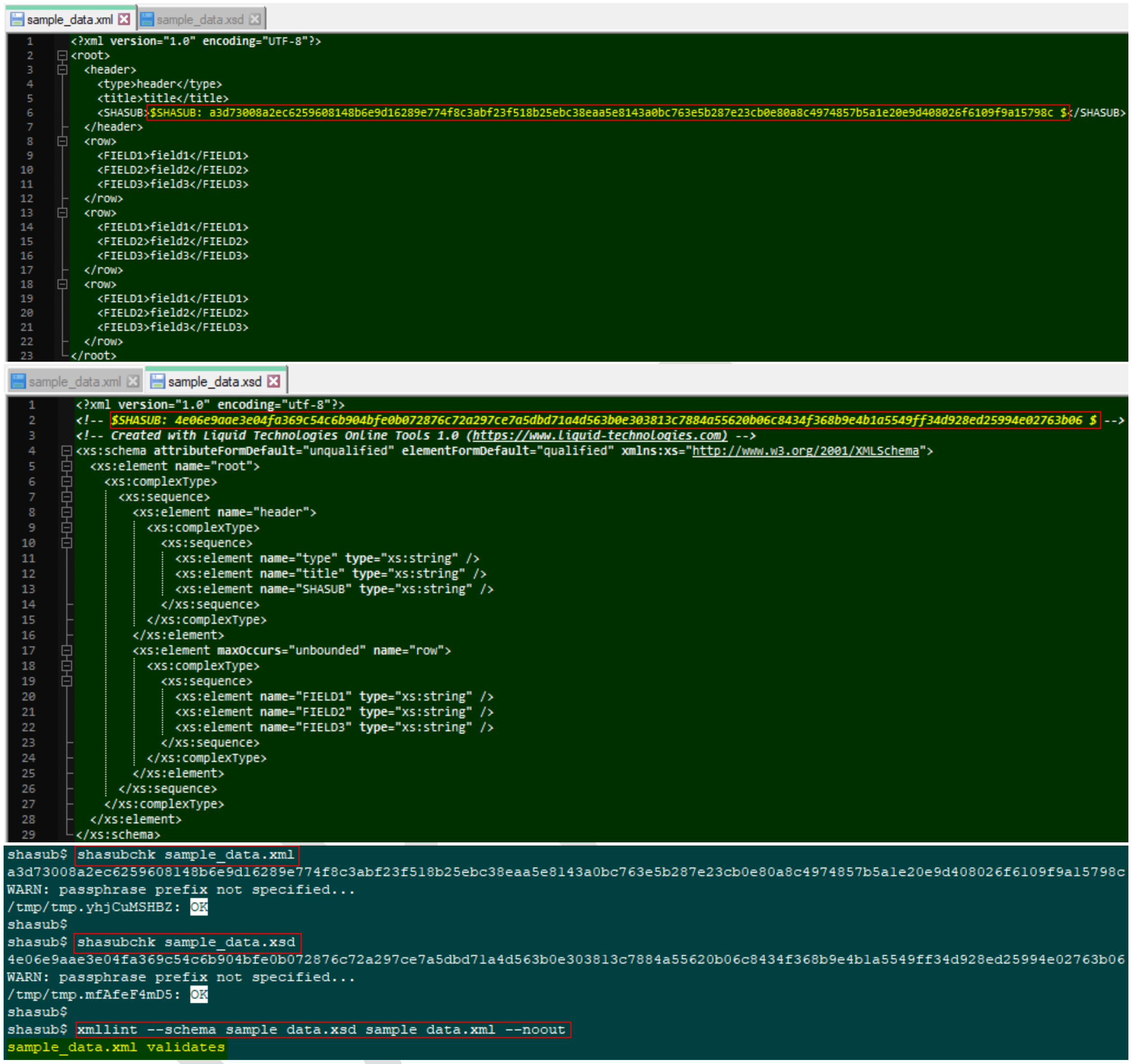


```
echo "Hello world! <8) "

echo "$SHASUB: 59610a4209126d6afdaeeaae7cad0f801b28850990a317563f2d93c84aa3dc69c39bbf964abc01463d8e639d089791f85b604d790536257b78ac931d955eb757 $"

echo "Goodbye world! <8( "
shasub$ shasubchk ./hello3.sh
59610a4209126d6afdaeeaae7cad0f801b28850990a317563f2d93c84aa3dc69c39bbf964abc01463d8e639d089791f85b604d790536257b78ac931d955eb757
WARN: passphrase prefix not specified...
/tmp/tmp.pvkv8kHtHp: OK
shasub$
shasub$ ./hello3.sh
Hello world! <8)
: 59610a4209126d6afdaeeaae7cad0f801b28850990a317563f2d93c84aa3dc69c39bbf964abc01463d8e639d089791f85b604d790536257b78ac931d955eb757 $
Goodbye world! <8(
```

Fig. 18: ‘hello3.sh’

Fig. 18: above illustrates SHASUB unintrusively embedded in ‘echo’ command.

2.1.4.4 *XML/XSD Sample Data*



sample_data.xml | sample_data.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <header>
    <type>header</type>
    <title>title</title>
    <SHASUB>$SHASUB: a3d73008a2ec6259608148b6e9d16289e774f8c3abf23f518b25ebc38eaa5e8143a0bc763e5b287e23cb0e80a8c4974857b5a1e20e9d408026f6109f9a15798c $</SHASUB>
  </header>
  <row>
    <FIELD1>field1</FIELD1>
    <FIELD2>field2</FIELD2>
    <FIELD3>field3</FIELD3>
  </row>
  <row>
    <FIELD1>field1</FIELD1>
    <FIELD2>field2</FIELD2>
    <FIELD3>field3</FIELD3>
  </row>
  <row>
    <FIELD1>field1</FIELD1>
    <FIELD2>field2</FIELD2>
    <FIELD3>field3</FIELD3>
  </row>
</root>
```

sample_data.xml | sample_data.xsd

```
<?xml version="1.0" encoding="utf-8"?>
<!-- $SHASUB: 4e06e9aae3e04fa369c54c6b904bfe0b072876c72a297ce7a5dbd71a4d563b0e303813c7884a55620b06c8434f368b9e4b1a5549ff34d928ed25994e02763b06 $ -->
<!-- Created with Liquid Technologies Online Tools 1.0 (https://www.liquid-technologies.com) -->
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="root">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="header">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="type" type="xs:string" />
              <xs:element name="title" type="xs:string" />
              <xs:element name="SHASUB" type="xs:string" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element maxOccurs="unbounded" name="row">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="FIELD1" type="xs:string" />
              <xs:element name="FIELD2" type="xs:string" />
              <xs:element name="FIELD3" type="xs:string" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```
shasub$ shasubchk sample_data.xml
a3d73008a2ec6259608148b6e9d16289e774f8c3abf23f518b25ebc38eaa5e8143a0bc763e5b287e23cb0e80a8c4974857b5a1e20e9d408026f6109f9a15798c
WARN: passphrase prefix not specified...
/tmp/tmp.yhjCuMSHBZ: OK
shasub$
shasub$ shasubchk sample_data.xsd
4e06e9aae3e04fa369c54c6b904bfe0b072876c72a297ce7a5dbd71a4d563b0e303813c7884a55620b06c8434f368b9e4b1a5549ff34d928ed25994e02763b06
WARN: passphrase prefix not specified...
/tmp/tmp.mfAfeF4mD5: OK
shasub$
shasub$ xmllint --schema sample_data.xsd sample_data.xml --noout
sample_data.xml validates
```

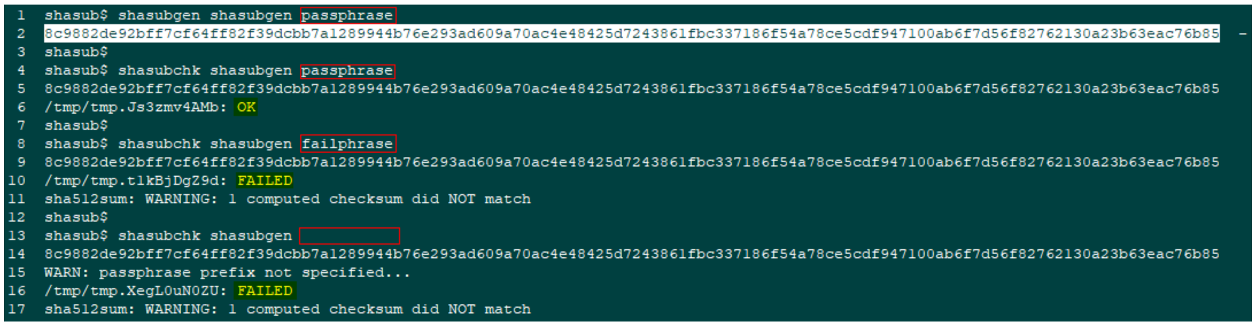
Fig. 19: XML/XSD Sample Data

Fig. 19: above illustrates SHASUB unintrusively embedded in XML/XSD file pair.

2.1.5 Secure SHASUB

The potential exists for the SHASUB mechanism is vulnerable to ‘tampering’. To address this valid concern, an optional ‘passphrase’ parameter has been added to the supporting utilities in order to make it ‘tamper-proof’.

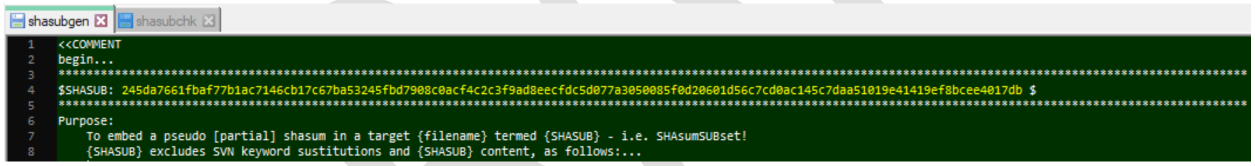
Referring to Fig. 20: below:...



```
 1  shasub$ shasubgen shasubgen passphrase
 2  8c9882de92bff7cf64ff82f39dcbb7a1289944b76e293ad609a70ac4e48425d7243861fbc337186f54a78ce5cdf947100ab6f7d56f82762130a23b63eac76b85  -
 3  shasub$
 4  shasub$ shasubchk shasubgen passphrase
 5  8c9882de92bff7cf64ff82f39dcbb7a1289944b76e293ad609a70ac4e48425d7243861fbc337186f54a78ce5cdf947100ab6f7d56f82762130a23b63eac76b85
 6  /tmp/tmp.Js3zmv4AMb: OK
 7  shasub$
 8  shasub$ shasubchk shasubgen failphrase
 9  8c9882de92bff7cf64ff82f39dcbb7a1289944b76e293ad609a70ac4e48425d7243861fbc337186f54a78ce5cdf947100ab6f7d56f82762130a23b63eac76b85
10  /tmp/tmp.t1kBjDgZ9d: FAILED
11  sha512sum: WARNING: 1 computed checksum did NOT match
12  shasub$
13  shasub$ shasubchk shasubgen
14  8c9882de92bff7cf64ff82f39dcbb7a1289944b76e293ad609a70ac4e48425d7243861fbc337186f54a78ce5cdf947100ab6f7d56f82762130a23b63eac76b85
15  WARN: passphrase prefix not specified...
16  /tmp/tmp.XegL0uN0ZU: FAILED
17  sha512sum: WARNING: 1 computed checksum did NOT match
```

Fig. 20: Passphrase SHASUB Usage

- Line #1 invokes 'shasubgen' with provided 'passphrase';
- Resultant generated SHASUB value on line #2 embedded in target file per Fig. 21: below;
- Line #4 invokes 'shasubchk' with provided valid 'passphrase';
- Embedded SHASUB value located in target file identified on line #5;
- Line #6 displays successful validation of file's contents;
- Line #8 invokes 'shasubchk' with provided invalid 'failphrase';
- Embedded SHASUB value located in target file identified on line #9;
- Line #10 displays unsuccessful validation of file's contents;
- Line #13 invokes 'shasubchk' without a passphrase;
- Embedded SHASUB value located in target file identified on line #14;
- Line #16 displays unsuccessful validation of file's contents;



```
<<COMMENT
begin...
*******************************************************************************************************************************************
$SHASUB: 245da7661fbaf77b1ac7146cb17c67ba53245fbd7908c0acf4c2c3f9ad8eecfdc5d077a3050085f0d20601d56c7cd0ac145c7daa51019e41419ef8bcee4017db $
*******************************************************************************************************************************************
Purpose:
    To embed a pseudo [partial] shasum in a target {filename} termed {SHASUB} - i.e. SHAsumSUBset!
    {SHASUB} excludes SVN keyword sustitutions and {SHASUB} content, as follows:...
```

Fig. 21: Passphrase SHASUB Value

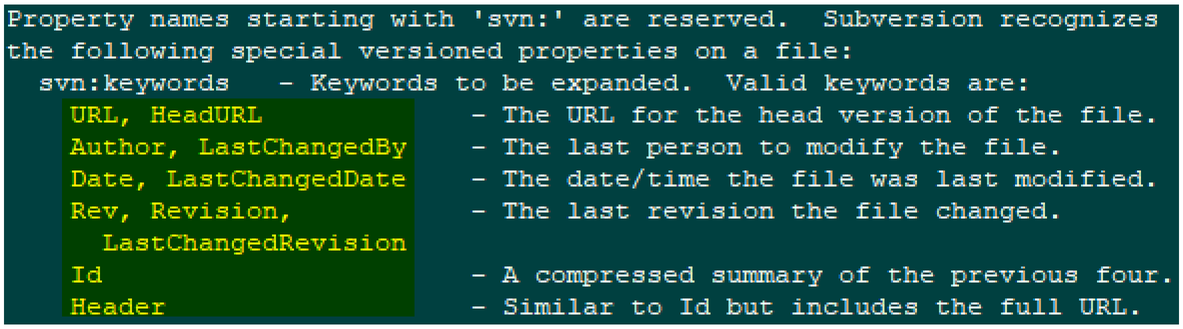
NOTE: The user supplied 'passphrase' on the command lines are transient. They are not embedded in the target file's contents. If the 'passphrase' is lost/forgotten, it is irretrievable by design due to SHASUB's implementation. In such circumstances, while the file's payload contents remain intact, nevertheless the contents are rendered questionable and cannot be validated. It is comparable to losing the SHASUM.

2.2 SubVersioN [SVN]

In addition to providing a mechanism for embedding a file's 'partial SHASUM', SHASUB also takes into account Subversion [SVN] – a sophisticated mainstream centralized file versioning system.

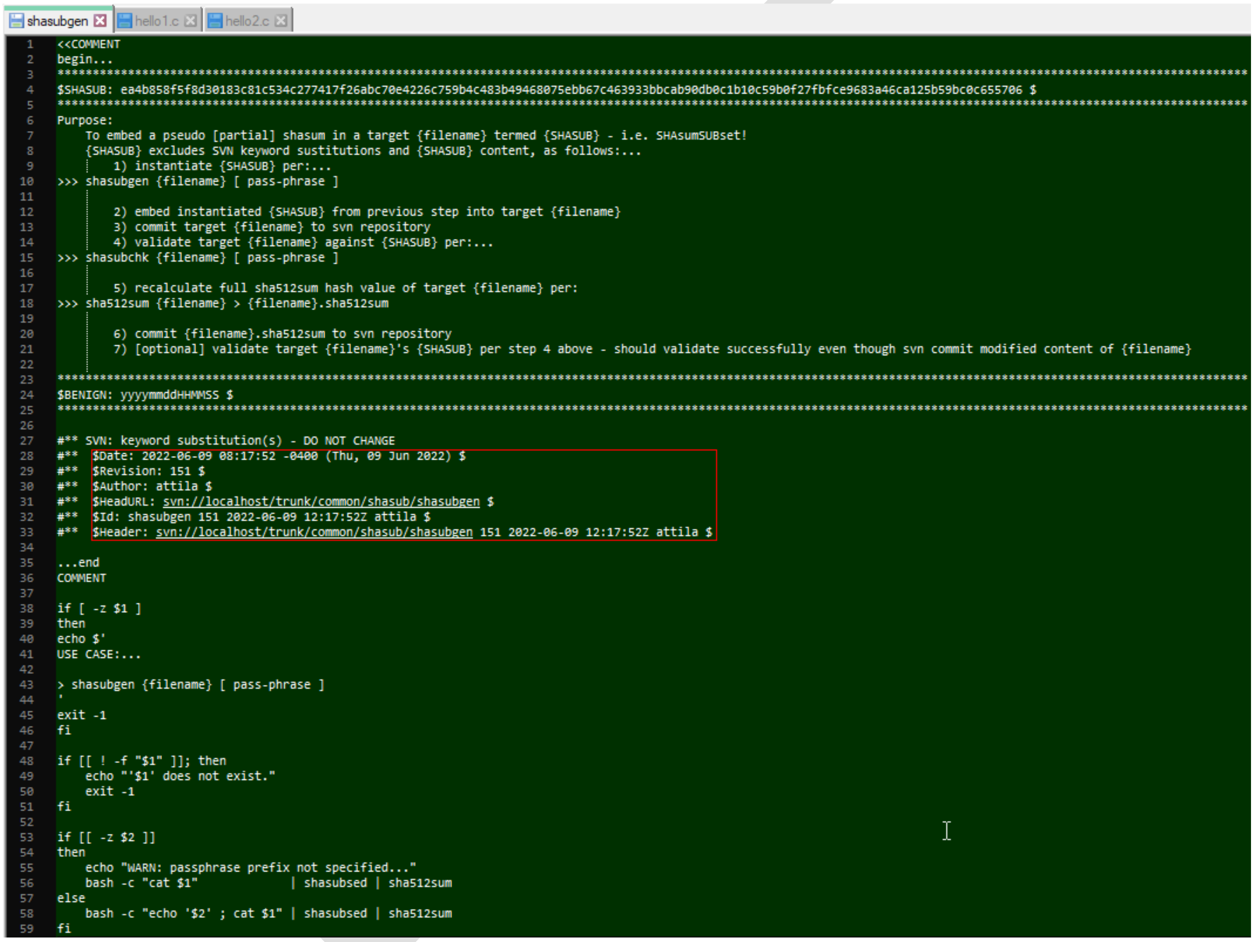
2.2.1 Keyword Substitution

Amongst other features, SVN provides the capability to automatically embed file revisioning properties [information] directly into benign section(S) of files' contents every time files are committed to the SVN's version control repository:



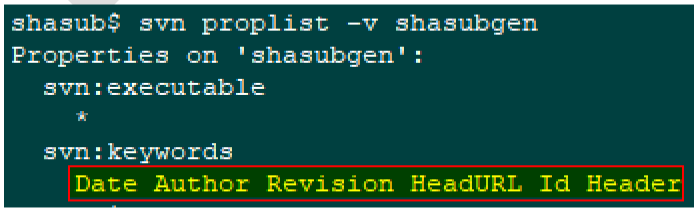
```
Property names starting with 'svn:' are reserved.  Subversion recognizes
the following special versioned properties on a file:
  svn:keywords   - Keywords to be expanded.  Valid keywords are:
    URL, HeadURL             - The URL for the head version of the file.
    Author, LastChangedBy    - The last person to modify the file.
    Date, LastChangedDate    - The date/time the file was last modified.
    Rev, Revision,           - The last revision the file changed.
      LastChangedRevision
    Id                       - A compressed summary of the previous four.
    Header                   - Similar to Id but includes the full URL.
```

:::



shasubgen hello1.c hello2.c

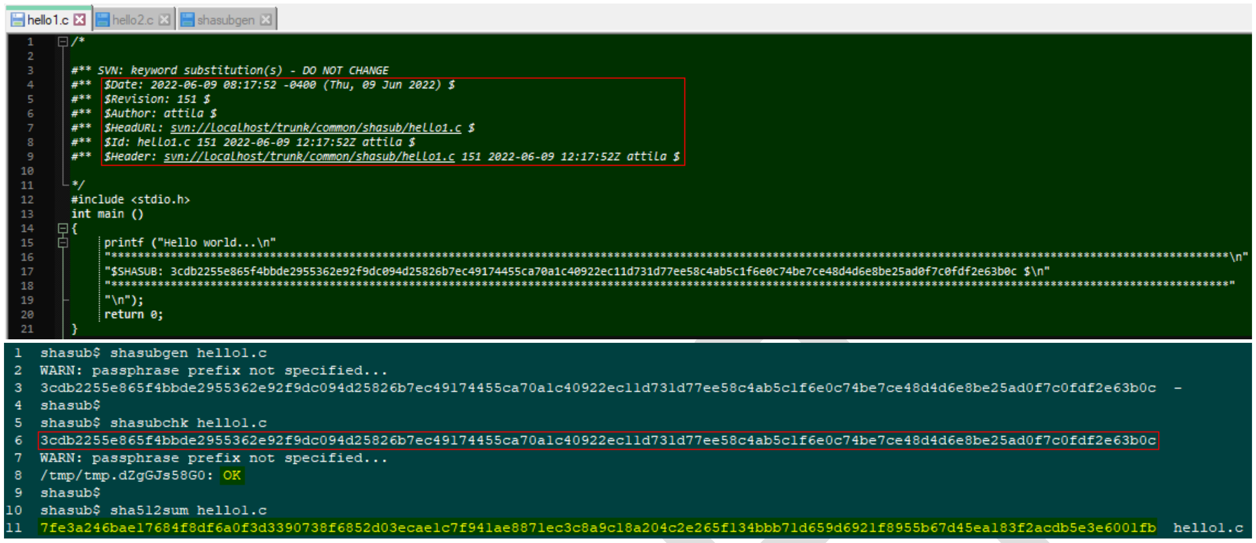
```
1   <<COMMENT
2   begin...
3   ****************************************************************************************************************************************************
4   $SHASUB: ea4b858f5f8d30183c81c534c277417f26abc70e4226c759b4c483b49468075ebb67c463933bbcab90db0c1b10c59b0f27fbfce9683a46ca125b59bc0c655706 $
5   ****************************************************************************************************************************************************
6   Purpose:
7       To embed a pseudo [partial] shasum in a target {filename} termed {SHASUB} - i.e. SHAsumSUBset!
8       {SHASUB} excludes SVN keyword sustitutions and {SHASUB} content, as follows:...
9           1) instantiate {SHASUB} per:...
10  >>> shasubgen {filename} [ pass-phrase ]
11
12          2) embed instantiated {SHASUB} from previous step into target {filename}
13          3) commit target {filename} to svn repository
14          4) validate target {filename} against {SHASUB} per:...
15  >>> shasubchk {filename} [ pass-phrase ]
16
17          5) recalculate full sha512sum hash value of target {filename} per:
18  >>> sha512sum {filename} > {filename}.sha512sum
19
20          6) commit {filename}.sha512sum to svn repository
21          7) [optional] validate target {filename}'s {SHASUB} per step 4 above - should validate successfully even though svn commit modified content of {filename}
22
23  ****************************************************************************************************************************************************
24  $BENIGN: yyyymmddHHMMSS $
25  ****************************************************************************************************************************************************
26
27  #** SVN: keyword substitution(s) - DO NOT CHANGE
28  #**  $Date: 2022-06-09 08:17:52 -0400 (Thu, 09 Jun 2022) $
29  #**  $Revision: 151 $
30  #**  $Author: attila $
31  #**  $HeadURL: svn://localhost/trunk/common/shasub/shasubgen $
32  #**  $Id: shasubgen 151 2022-06-09 12:17:52Z attila $
33  #**  $Header: svn://localhost/trunk/common/shasub/shasubgen 151 2022-06-09 12:17:52Z attila $
34
35  ...end
36  COMMENT
37
38  if [ -z $1 ]
39  then
40  echo $'
41  USE CASE:...
42
43  > shasubgen {filename} [ pass-phrase ]
44  '
45  exit -1
46  fi
47
48  if [[ ! -f "$1" ]]; then
49      echo "'$1' does not exist."
50      exit -1
51  fi
52
53  if [[ -z $2 ]]
54  then
55      echo "WARN: passphrase prefix not specified..."
56      bash -c "cat $1"             | shasubsed | sha512sum
57  else
58      bash -c "echo '$2' ; cat $1" | shasubsed | sha512sum
59  fi
```



```
shasub$ svn proplist -v shasubgen
Properties on 'shasubgen':
  svn:executable
    *
  svn:keywords
    Date Author Revision HeadURL Id Header
```

Fig. 22:	Revision Property Keywords

As lines #28 thru #33 illustrate in Fig. 22: above, files "propset" with specified revision property keywords are automatically updated/modified during SVN's commit by embedding revision information corresponding to latest commit. As stated earlier, in doing so the file's SHASUM value <u>before</u> the commit will not match the file's SHASUM value <u>after</u> the commit procedure. Fig. 23: and Fig. 24: below illustrates how SHASUB circumvents this issue.

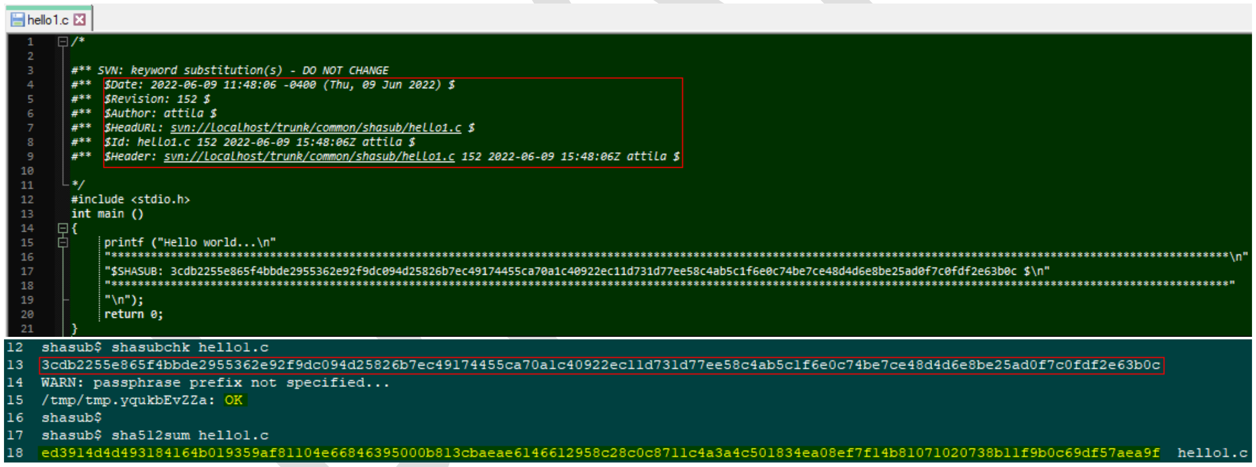


```
hello1.c  hello2.c  shasubgen

/*

#** SVN: keyword substitution(s) - DO NOT CHANGE
#**  $Date: 2022-06-09 08:17:52 -0400 (Thu, 09 Jun 2022) $
#**  $Revision: 151 $
#**  $Author: attila $
#**  $HeadURL: svn://localhost/trunk/common/shasub/hello1.c $
#**  $Id: hello1.c 151 2022-06-09 12:17:52Z attila $
#**  $Header: svn://localhost/trunk/common/shasub/hello1.c 151 2022-06-09 12:17:52Z attila $

*/
#include <stdio.h>
int main ()
{
    printf ("Hello world...\n"
    "*************************************************************************************************************************************************************************\n"
    "$SHASUB: 3cdb2255e865f4bbde2955362e92f9dc094d25826b7ec49174455ca70a1c40922ec11d731d77ee58c4ab5c1f6e0c74be7ce48d4d6e8be25ad0f7c0fdf2e63b0c $\n"
    "**************************************************************************************************************************************************************************"
    "\n");
    return 0;
}
```

```
1   shasub$ shasubgen hello1.c
2   WARN: passphrase prefix not specified...
3   3cdb2255e865f4bbde2955362e92f9dc094d25826b7ec49174455ca70a1c40922ec11d731d77ee58c4ab5c1f6e0c74be7ce48d4d6e8be25ad0f7c0fdf2e63b0c  -
4   shasub$
5   shasub$ shasubchk hello1.c
6   3cdb2255e865f4bbde2955362e92f9dc094d25826b7ec49174455ca70a1c40922ec11d731d77ee58c4ab5c1f6e0c74be7ce48d4d6e8be25ad0f7c0fdf2e63b0c
7   WARN: passphrase prefix not specified...
8   /tmp/tmp.dZgGJs58G0: OK
9   shasub$
10  shasub$ sha512sum hello1.c
11  7fe3a246bae17684f8df6a0f3d3390738f6852d03ecae1c7f941ae8871ec3c8a9c18a204c2e265f134bbb71d659d6921f8955b67d45ea183f2acdb5e3e6001fb  hello1.c
```

Fig. 23: Pre-SVN Commit

Fig. 23: illustrates the SHASUM value calculated on lines #10 thru #11 of file 'hello1.c' [$Revision: 151] <u>before</u> SVN commit.



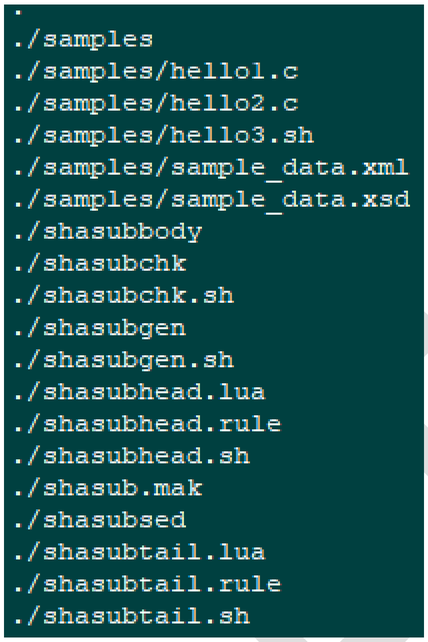
```
hello1.c

/*

#** SVN: keyword substitution(s) - DO NOT CHANGE
#**  $Date: 2022-06-09 11:48:06 -0400 (Thu, 09 Jun 2022) $
#**  $Revision: 152 $
#**  $Author: attila $
#**  $HeadURL: svn://localhost/trunk/common/shasub/hello1.c $
#**  $Id: hello1.c 152 2022-06-09 15:48:06Z attila $
#**  $Header: svn://localhost/trunk/common/shasub/hello1.c 152 2022-06-09 15:48:06Z attila $

*/
#include <stdio.h>
int main ()
{
    printf ("Hello world...\n"
    "*************************************************************************************************************************************************************************\n"
    "$SHASUB: 3cdb2255e865f4bbde2955362e92f9dc094d25826b7ec49174455ca70a1c40922ec11d731d77ee58c4ab5c1f6e0c74be7ce48d4d6e8be25ad0f7c0fdf2e63b0c $\n"
    "**************************************************************************************************************************************************************************"
    "\n");
    return 0;
}
```

```
12  shasub$ shasubchk hello1.c
13  3cdb2255e865f4bbde2955362e92f9dc094d25826b7ec49174455ca70a1c40922ec11d731d77ee58c4ab5c1f6e0c74be7ce48d4d6e8be25ad0f7c0fdf2e63b0c
14  WARN: passphrase prefix not specified...
15  /tmp/tmp.yqukbEvZZa: OK
16  shasub$
17  shasub$ sha512sum hello1.c
18  ed3914d4d493184164b019359af81104e66846395000b813cbaeae6146612958c28c0c8711c4a3a4c501834ea08ef7f14b81071020738b11f9b0c69df57aea9f  hello1.c
```

Fig. 24: Post-SVN Commit

Fig. 24: illustrates the SHASUM value calculated on lines #17 thru #18 of file 'hello1.c' [$Revision: 152] <u>after</u> SVN commit no longer matches. However, as seen on command line #12 thru #15, the file's SHASUB value still validates successfully.

2.3 SHASUB Constituents



```
.
./samples
./samples/hello1.c
./samples/hello2.c
./samples/hello3.sh
./samples/sample_data.xml
./samples/sample_data.xsd
./shasubbody
./shasubchk
./shasubchk.sh
./shasubgen
./shasubgen.sh
./shasubhead.lua
./shasubhead.rule
./shasubhead.sh
./shasub.mak
./shasubsed
./shasubtail.lua
./shasubtail.rule
./shasubtail.sh
```

Fig. 25: SHASUB Constituents

Fig. 25: above lists the constituents of SHASUB prototype. In its current rendition, it is implemented in 'bash' and 'sed' scripts.

If adopted for incorporation into 'coreutils', it is recommended to be implemented in a mainstream programming language – eg: C/C++.

Also, if adopted for incorporation, it is recommended SHASUB utilities not support SVN 'keyword substitution' feature. A recommendation to Apache® Subversion® support community should be approached to implement a new keyword to support SHASUB in accordance with SVN's 'keyword substitution' feature during commits.

3 Caveats/Limitations

- discuss the applicability to binary files – not just ASCII files

- one SHASUB placeholder per file

Acronyms

ASCII	American Standard Code for Information Interchange
SHA	Secure Hash Algorithms [https://en.wikipedia.org/wiki/Secure_Hash_Algorithms]
SHASUB	SHAsum SUBset – an abbreviation
SHASUM	Linux-based SHA sum utility with varying precision between 160 and 512 bits
SHAxSUM	Instance of SHASUM with specific precision – eg: 256, 512, etc.
Subversion	Apache Subversion [https://en.wikipedia.org/wiki/Apache_Subversion]
SVN	Subversion abbreviation
MIM	Man in the middle attack https://en.wikipedia.org/wiki/Man-in-the-middle_attack