PharmaSUG 2012 - Paper TA01

Checksum Please: A Way to Ensure Data Integrity

Carey Smoak, Roche Molecular Systems, Inc., Pleasanton, CA

Mario Widel, Roche Molecular Systems, Inc., Pleasanton, CA

Sy Truong, Meta-Xceed, Inc., Fremont, CA

ABSTRACT

Want to ensure data integrity? Want to know if a file has been altered? Checksum, please! Checksums have a variety of applications, such as verifying that an application has been installed correctly and providing a method of verifying whether or not a file has been altered. For example, checksums can be used in a SAS program to verify that a comma delimited (CSV) file has not been altered before importing the CSV file into a SAS dataset. SAS by itself does not make use of checksums, but fortunately all operating systems have some utility available to do it. This paper will provide some background on their uses and concentrate on a particular example.

INTRODUCTION

A bit of history

Sending a message only readable by the intended receiver and unreadable by anyone else has been a quest since ancient times. The following examples can be found among the numerous attempts that have been made starting with hieroglyphics which were modified to make them unreadable for unintended audiences; from around 1900 BC, this example from ancient Egypt is the oldest known cryptographic text.

- The Bible in the Old Testament contains ciphers, although not necessarily intended to keep secrecy a letter substitution was described.
- Romans used the Caesar's cipher which was just a simple substitution of letters, every letter of a message was shifted the same number of positions in the alphabet.
- Ancient Greece has also their system to send a message a strip of leather was rolled around a cylinder and the message written there. The message strip was send and the receiver has to use another cylinder of exactly the same diameter of the sender to read the message.
- During the Dark Ages arts and sciences disappeared together with cryptography, however throughout the Middle Ages, ciphers were used by monks as a form of entertainment.
- In the 600s Arab science become the best in the world, words like "cipher" or "algebra" come from that time. The Arabs were the first to systematically compile all the knowledge about cryptography thus creating cryptology.
- In the 1200s Roger Bacon, an English Franciscan friar, who thought of microscopes and telescopes as well as horseless carriages and flying machines, wrote a book containing seven methods to cipher text.
- In 1379 a collection of Gabriele de Lavinde's ciphers was produced becoming the first European manual on cryptography.
- In 1474, Sicco Simonetta published *Regulae ad extrahendum litteras zifferatas sine exemplo,* a short work which stressed "methods of decipherment and afforded considerable statistical data" .
- In 1585 Blaise de Vigenère published a book on cryptology , describing the polyalphabetic substitution cipher, considered unbreakable for centuries to come. However this method had been described as early as 1467 by Leon Batista Alberti.
- In 1860 Wheatstone developed the Wheastone disc based on the 1817 Wadsworth's invention. It consisted on two concentric wheels used to generate a polyalphabetic cipher .
- By the end of WW I Germany invented the Enigma machine which was heavily used by German U-boats

during WW II.

All of these examples mentioned above have a common characteristic which is a big weakness: the decoding key. All of them use the same key to code and decode a message, that is, a symmetric key. They require a secure transmission of the key which has some of the same problems that occurs during message transmission.

It was not until 1976 when Diffie and Hellman publish a description of the public key method. There is a public key to encode messages and a different private key to decode. No secure key transmission is needed. The theoretical concept was already mentioned by William Jevons back in 1874 and proposed as a puzzle by Ralph Merkle in 1974. It was only in 1997 that the UK government disclosed that they had independently developed the same system. The generalization and commercial products still used today is known as RSA (Rivest, Shamir and Adleman) algorithm. The public key is the product of two large primes, the decoding scheme used the factors. The secrecy is based on the (current) difficulty to factor a large number.

Correctness

Data that is considered correct is when its contents are transferred and when it is received; that both are identical. We have two cases to consider when correctness is potentially compromised.

1. We are concerned only for transmission errors which would be somehow erratic.
2. Our concern is regarding intentional message modification.

In the example of SAS, this can occur when a SAS program imports ASCII files into SAS datasets. In this case, the ASCII files were downloaded from an FTP server. The transmission could introduce corruption to the ASCII files. In the second example, another user could have modified the ASCII files prior to the use of the SAS program. The intended ASCII file has been modified with good intentions such as making column headers more clear. This modification may or may not be malicious, but the transmission is no longer identical to the source which makes it incorrect.

Error correction

In addition to recognizing that the data is not correct, we need to identify what the specific errors or modification has been applied. The following are four example schemes to identifying these modifications:

1. Error detection like parity bits will detect if one or an odd number of bits in a message is not as expected but are unable to identify which one.
2. Another scheme like repeating messages must be regarded as very inefficient.
3. A more sensitive scheme is the Cyclic Redundancy Check (CRC) where a binary polynomial is applied as a divisor of the original message in binary format and taking the remainder as a result.
4. Error Correction or rather error-correcting codes, will also add redundant data but in a way the can detect errors and identify the faulty bit, for instance Hamming (7,4).

These four examples illustrate schemes for data correction methods. This paper will not demonstrate any SAS examples of error corrections, since it is not an optimal use of SAS and is outside the scope.

Error Detection

This paper will focus on the error detection of data during transmission. If we must know that a message has been accidentally modified in any way, we must use a checksum.

For example, the MD5 of the previous sentence is displayed below:

```
86fa7014d5c52f166c907d3777d30c61
```

This checksum was obtained on a Windows PC using the FCIV function which is available from Microsoft. As you can see from the example above, a checksum is typically a number obtained by using a checksum procedure similar to the FCIV function. Checksum may also be known as hash functions, fingerprint, etc...

Practical examples of checksum include:

- Parity bits in memory bytes
- Check digits in Social Security Numbers
- ISBN book numbers
- International Bank Account Numbers

- International Standard Serial Number
- International Standard Music Number
- UPC
- Encoding passwords
- Many others …

As you can see from the above list, there are many examples of checksums (e.g., MD5, SHA1, SHA2, etc.) which are ubiquitous without us knowing about it! The types of checksums will be explained in this paper.

Paper Scope

One of the main goals of obtaining a validated system is to verify that the software is installed according to the intentions of the software vendor. In the example of SAS, we need to ensure that the digital files installed on our server, matches with the benchmark files form SAS Institute. Checksum is a perfect tool for this since it can capture a digital signature between files on different systems and easily verify if they are identical. The digital signature of the files is a unique identifier analogous to a finger print of an individual person. If this were to be altered or different in any way, checksum will catch these accidental errors that can be introduction during the installation and configuration of a SAS system. Although checksum is a general purpose tool that can be used on any software, this paper will explore its uses for the SAS system and related data and program macros.

USES OF CHECKSUMS IN HEALTHCARE INDUSTRY

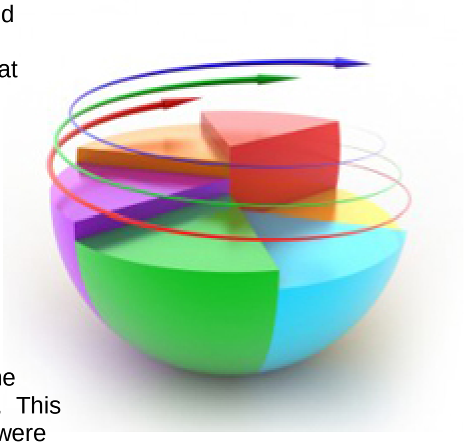
There are a variety of uses of checksums within the healthcare, including eCTD submissions to the FDA, SAS/Software installation, migrating files from one server to another, data transfers, and handling electronic files generated by lab instruments. Each of these scenarios will be discussed in this paper.

ECTD

Checksums should be provided for each file in an eCTD (electronic Common Technical Document) submissions to the FDA. The checksums allow the end-user to verify that the files were not altered during transmission to the FDA (Wang 2004; Bairnsfather 2003). The verification that the files have not been altered is done by the end-user comparing their own checksum for each file with the checksum provided by the initial user. If the two checksums agree, then the end-user can be confident that the files have not been altered.

SAS/Software Installation

SAS provides checksums and installation/operational qualification programs to verify that SAS has been installed and operates correctly. The user can run the installation/operational qualification (IQ/OQ) programs provided by SAS. The resulting output (a PDF file) will show the checksums provided by SAS and those generated when installing the product agree (Helton, McNealy, Halley 2003; Truong, Smoak 2010). Thus the user installing SAS can easily verify that SAS has been installed and operates correctly. SAS also provides notes identifying known instances where the checksum will not agree (Csont, Proskin 2007). Thus the user will be prompted to then verify these failures (when checksums disagree) using another method, or explain the failure in their documentation. This verification process of comparing checksums can be used with software products other than SAS (Baucom 2009).



File Migration (SAS Server)

Another use of checksums is verifying the migration of files from one server to another (Truong, Smoak 2010). Checksums are performed for each file being migrated from the old server. If the two checksums for each file agree, then the user can be confident that the files have not been altered during the migration. This is not a simple task. Two of the authors on this paper (Truong, Smoak 2010) were involved in such a project. It took 22 hours to migrate all of the files and 13 hours to compare all of the checksums.

Data Transfers

Checksums are also useful in verifying that data transfers have been performed correctly, e.g., the file has not been corrupted during transmission of the file (Grasse, Nelson 2006; Mehler 2004; Rausch 2006; Maddox, Fulenwider 1990; Kent 1986). Thus the integrity of data transfers can be verified using checksums.

Instrument Data

A special case of data transfers is described in this paper. Data from a laboratory instrument stored on a hard-drive can be exported as an Extensible Markup Language (XML) file. When the XML file is generated, a checksum accompanies it. A validated tool can then convert data in the XML file to a CSV file. The validated tool first compares the checksum of the XML file with its own checksum on the same file. If the two checksums agree then the tool parses data from the XML file into a CSV file. When the CSV file is generated, a new checksum accompanies it.

Then a SAS programs is run to import the CSV file into a SAS dataset. First the SAS program generates a checksum on the CSV file and compares it to the checksum generated by the validated tool that also generated the CSV file. If the two checksums agree then the SAS program creates a SAS dataset from the CSV file.

Thus the user can be confident that the data in the SAS dataset is the same as the original data on the hard-drive of the laboratory instrument. At each step in the process, checksums are used to verify data integrity. This process is illustrated in the figure below.

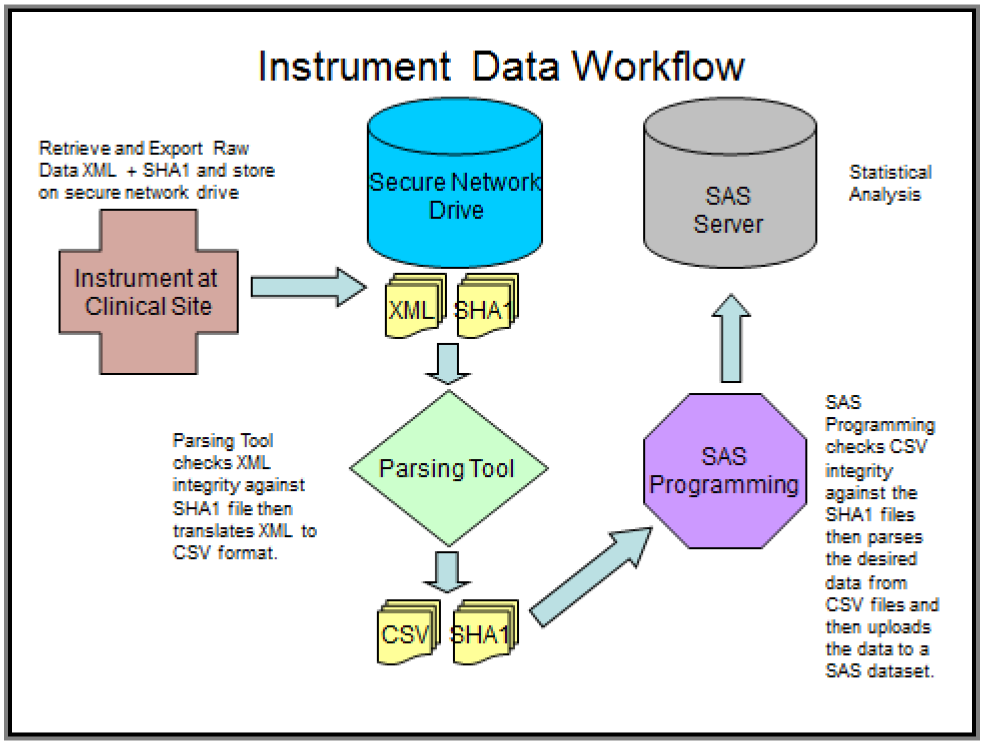

Instrument Data Workflow
Retrieve and Export Raw Data XML + SHA1 and store on secure network drive
Instrument at Clinical Site
Secure Network Drive
XML
SHA1
SAS Server
Statistical Analysis
Parsing Tool checks XML integrity against SHA1 file then translates XML to CSV format.
Parsing Tool
SAS Programming
SAS Programming checks CSV integrity against the SHA1 files then parses the desired data from CSV files and then uploads the data to a SAS dataset.
CSV
SHA1

Figure 1. Data workflow

Password Protection

Passwords are not only used in the healthcare industry, but in our everyday transactions. A recent article in Consumer Reports (January, 2012), discusses the topic of hack-proofing passwords. The article points out that passwords are not stored, rather their checksum. This is to make the passwords more secure because the original password is never written into the system and the checksum itself does not allow you to find the original password. However, certain types of checksums are more secure or robust than others (see Wang, Yu 2005 for more details). For example, SHA1 (Secure Hash Algorithm 1) came as a response to theoretical attacks on MD5 in the 1990's. Further along the road, SHA1 showed some vulnerabilities and SHA2, (which consists of a family of four messages digests: SHA224, SHA256, SHA384, SHA512) were developed. More recently NSA started developing an even more secured version: SHA3.

How are passwords hacked? Very simple, you just need to find any string that will replicate the checksum of the password that you want to hack! Of course, you need to know the checksum of the password and you need to know what checksum type (MD5 or SHA1) has been applied to the password. Any public checksum utility can be used to compute checksums of arbitrary strings. After a few billion tries you might be successful in hacking a password!

Another approach to hacking passwords is using a dictionary where instead of trying billions of arbitrary strings, you use common words from the dictionary. Thus special characters and numbers imbedded in passwords protects against this kind of attack. However, we should not get carried away with creating passwords which are so convoluted that they are hard to type or difficult to remember. Users who write down convoluted passwords, defeat the purpose of having a password at all.

Users who want more information on how the algorithms for checksums are computed, see the paper by Rivest (1992).

CONCLUSION

The most significant vulnerabilities in maintaining data integrity occurs when it is being transferred. This can happen when the binaries of a software is being installed from the intended vendor to a user's machine, or when lab data in CSV format is transferred to SAS datasets for analysis or when a clinical trial is submitted to the FDA for approval. These examples highlight an inflection point in the integrity of the data as it is transferred from its original form, which is correct, to a new environment that introduce unknown changes to the data. You can ensure that the data is correct by comparing it from its original source to its final transformed location. This paper illustrates the use of checksum which provide an effective way of performing this comparison and confirms if the data's integrity is intact. The SAS data used in the examples of this paper illustrates how varied and complex transformations to the original data can be. Regardless of the different environments that the SAS system and its data are transformed, the use of checksum can be applied to consistently verify its correctness and integrity. The core concepts of checksum have roots in cryptography that dates back hundreds of years. This is a testament to how checksum is relevant and effective in today's' SAS implementations as it stands up over time.

REFERENCES

"Hack-Proof Your Passwords." Consumer Reports. January, 2012, pp 19-21.

Bairnsfather S. "Progress Toward Standardization of Submissions with the Electronic Common Technical Document and the Evolving Standardization of Clinical Data." *Proceedings of the Pharmaceutical SAS Users Group,* 2003. www.lexjansen.com/pharmasug/2003/fdacompliance/fda087.pdf

Baucom J. "Pharmacogenomics: JMP Right In!" *Proceedings of the Pharmaceutical SAS Users Group,* 2009 www.lexjansen.com/pharmasug/2009/pr/pr02.pdf.

Blair E. "Using the SAS· System for Data Collection under the VMS Operating System." *Proceedings of the 15th SAS Users Group International*, 1990 http://www.sascommunity.org/sugi/SUGI90/Sugi-90-148%20Blair.pdf.

Csont WC, Proskin HM. "Taking a Walk on the Wildside: Use of the PROC CDISC-SDTM 3.1 Format." *Proceedings of the Pharmaceutical SAS Users Group*, 2007 www.lexjansen.com/pharmasug/2007/cc/cc23.pdf.

Grasse D, Nelson GS. "Base SAS vs. Data Integration Studio: Understanding ETL and the SAS Tools Used to Support It." *Proceedings of the Pharmaceutical SAS Users Group,* 2006 www.lexjnsen.com/pharmasug/2006/datamanagement/dm02.pdf; *Proceedings of the Pharmaceutical Users Software Exchange*, 2006 www.lexjansen.com/phuse/2006/dm/dm01.pdf; *Proceedings of the 31st SAS Users Group International*, 2006 http://www2.sas.com/proceedings/sugi31/099-31.pdf.

Helton E, McNealy J, Halley, PB. "SAS® Validation in the Pharmaceutical Industry." *Proceedings of the Pharmaceutical SAS Users Group,* 2003 www.lexjansen.com/pharmasug/2003/sasinstitute/sas137.pdf.

Jacobs CA. "Installing and Maintaining Supervisor and Product Bundles in LPA/ELPA for MVS/XA and MVS/ESA Sites." *Proceedings of the 16th SAS Users Group International*, 1991 http://www.sascommunity.org/sugi/SUGI91/Sugi-91-162%20Jacobs.pdf.

Dan Kaminsky: "MD5 To Be Considered Harmful Someday", *Cryptology ePrint Archive*, Report 2004/357, http://eprint.iacr.org/2004/357, 6 December 2004

Kent P. "Microcomputers and SYSTEM 2000 DBMS." *Proceedings of the 11th SAS Users Group International*, 1986 http://www.sascommunity.org/sugi/SUGI86/Sugi-11-110%20Kent.pdf.

Vlastimil Klima: "Finding MD5 Collisions – a Toy For a Notebook", *Cryptology ePrint Archive* Report 2005/075, 5 March 2005. http://cryptography.hyperlink.cz/md5/MD5_collisions.pdf

Kolb D. "MICRO TO HOST LINK: PERFORMANCE AND SETUP." *Proceedings of the 12th SAS Users Group International*, 1987 http://www.sascommunity.org/sugi/SUGI87/Sugi-12-139%20Kolb.pdf.

Laing J. "Client/Server Setup and Implementation: Web and non-Web Environments." *Proceedings of the 25th SAS Users Group International*, 2000 http://www2.sas.com/proceedings/sugi25/25/aa/25p019.pdf.

LeSueur J. "Minimizing Development Risk While Maximizing Warehouse Performance on UNIX® Systems." *Proceedings of the 25th SAS Users Group International*, 2000 http://www2.sas.com/proceedings/sugi25/25/dw/25p120.pdf.

Maddox, JA, Fulenwider DO. "USING THE SAS SYSTEM TO ANALYZE QUALITY CONTROL DATA FROM DATA COLLECTION DEVICES." *Proceedings of the 15th SAS Users Group International*, 1990 http://www.sascommunity.org/sugi/SUGI90/Sugi-90-75%20Maddox%20Fulenwider.pdf.

Mangold A. "Regulatory Compliant Validation and Deployment of SAS Programs with the Colibri Server." *Proceedings of the Pharmaceutical Users Software Exchange*, 2005 www.lexjansen.com/phuse/2005/rc/rc01.pdf.

Ilya Mironov, "Hash functions: Theory, attacks, and applications". November 14, 2005. http://research.microsoft.com/pubs/64588/hash_survey.pdf

Mehler G. "Next Generation Data Warehousing with SAS 9." *Proceedings of the 29th SAS Users Group International*, 2004 www2.sas.com/proceedings/sugi29/114-29.pdf; *Proceedings of the Western Users of SAS Software,* 2004 http://www.lexjansen.com/wuss/2004/data_warehousing/s_dwdb_next_generation_data_.pdf.

Rausch N. "Stars and Models: How to Build and Maintain Star Schemas Using SAS® Data Integration Server in SAS®9." *Proceedings of the 31st SAS Users Group International*, 2006 www2.sas.com/proceedings/sugi31/096-31.pdf

Rivest, Ronald L. "The MD5 message-digest algorithm," IETF RFC 1321, 1992. http://www.ietf.org/rfc/rfc1321.txt

Shannon, Claude E. " "The Communication Theory of Secrecy Systems". Bell Systems Technical Journal, Vol. xxx, No. 1 (Jan. 1949). http://dm.ing.unibs.it/giuzzi/corsi/Support/papers-cryptography/Communication_Theory_of_Secrecy_Systems.pdf

Truong S, Smoak C. “Validating and Migrating SAS® 9.13 to 9.2.” *Proceedings of the Western Users of SAS Software* http://www.lexjansen.com/wuss/2010/coders/2962_3_COD-Truong2.pdf.

Wang S. “eCTD -- A New Standard for FDA Electronic Submission.” *Proceedings of the Pharmaceutical SAS Users Group,* 2004 www.lexjansen.com/pharmasug/2004/fdacompliance/fc10.pdf.

Xiaoyun Wang and Hongbo Yu, “How to Break MD5 and Other Hash Functions”, EUROCRYPT 2005: 24th annual International.

X.Y. Wang, F.D. Guo, X.J. Lai, H.B. Yu, “Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD”, rump session of Crypto’04, E-print, 2004.

RECOMMENDED READING

- Kahn, David. The Codebreakers: The Story of Secret Writing. New York: Macmillan, 1967.
- Singh, Simon. The Code Book; The Science of Secrecy from Ancient Egypt to Quantum Cryptography. Anchor, 2000.
- Schneier, Bruce. Applied Cryptography, Protocols, Algorithms and Source Code in C. John Wiley & Sons, Inc., 1996

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Carey Smoak
Senior Manager, SAS Programming
Roche Molecular Systems, Inc.
4300 Hacienda Drive
Pleasanton, CA 94588
E-mail: carey.smoak@roche.com

Mario Widel
Manager, SAS Programming
Roche Molecular Systems, Inc.
4300 Hacienda Drive
Pleasanton, CA 94588
E-mail: mario.widel@roche.com

Sy Truong
MetaXceed, Inc. (MXI)
42978 Osgood Rd
Fremont, CA 94539-5627
Tel: (510) 979-9333
Fax: (510) 440-8301
E-mail: sy.truong@meta-x.com
Web: www.meta-x.com