A Chunk Registration Request

For Enabling

The PNG Standard

To Support Digital Signatures

Specification, Technical Proposal & Request

Document History			
Version	**Date**	**Author**	**Remarks**
1.0	18.02.2008	Dialogika GmbH/tk	Initial Version
1.1	20.02.2008	Dialogika GmbH/tk	Minor Syntactical Corrections

Table of Contents

1 DISCLAIMER

This document proposes a new PNG chunk type and **requests its registration** in order to support PNG digital signatures in a generic and extensible manner.

Neither Dialogika GmbH nor LuxTrust S.A. claim any intellectual property rights or patents or protection of other rights concerning this proposal by virtue of authoring it.

In any event, if the W3C standardization group accepts registering the chunk type specified below, we would like the Internet community to benefit from the new feature and use it in an open and interoperable way so as to make networks and communications more secure.

LuxTrust S.A.

Based in Luxembourg, LuxTrust S.A. has been created in November 2005 by the Luxembourg Government and several important companies from the public and private sector in the Grand-Duchy of Luxembourg in order to face the increasing need for security and confidentiality in the electronic commerce and in the Internet-based relationships between citizens, administrations and companies.

LuxTrust S.A. is a certification authority delivering electronic certificates for people authentication and secure electronic signatures in Internet- and Intranet-transactions. As a trusted third party, LuxTrust furthermore guarantees highly secured electronic certification services.

For more information about LuxTrust please visit *www.luxtrust.lu*, section « *Qui sommes nous?* ».

DIaLOGIKa GmbH

is a German systems and software house founded in 1982 and conducts projects on behalf of industry, finance, and governmental and supranational clients such as the institutions of the European Union (EU).

DIaLOGIKa has contributed to the open source community by actively participating in the development of the W3C's http reference server. DIaLOGIKa has adopted Java since its very beginning for creating software and solutions, in addition to actively promoting the further development of Java by joining the JCP (Java Community Process) in 2001.

From the beginning DIaLOGIKa has focused on technically demanding projects in the field of multilingual text and data processing, security solutions, PKI consulting, specification and implementation.

2 PROBLEM STATEMENT

Portable Networks Graphics (PNG) is a powerful ISO/W3C standard for digitally storing images. In addition, PNG is patent-free and has consequently become very popular and also capable of replacing other formats like GIF, JPEG or TIFF. Finally, PNG is extensible by adding so-called **extension chunks**, with well-typed chunks being the atomic storage unit of PNG.

The more PNG is used for storing scanned documents, the more digital signatures will become an important common requirement for PNG. Unfortunately, the *PNG standard does not currently address digital signatures.*

3 SOLUTION OPTIONS

We discuss three possible scenarios for extending the PNG format with digital signature support:

3.1 Enveloping Signature

This approach attaches a digital signature by encapsulating the original document (image) in an envelope, with the envelope storing signature-related information, i.e. the signature itself and other items like signer certificates and timestamp signatures.

An adopted international standard exists, namely Cryptographic Message Syntax (CMS), for implementing such a format (cf. RFC 3852 and RFC 4853 for details). CMS supports envelopes for various purposes, with the **signed-data** content type (and optionally the **authenticated-data** content type) being used for the signature aspect.

3.1.1 Advantages

CMS is a well-defined **standard**, providing a generic format for storing digital signing data of any kind. Consequently, this approach could also be applied to a PNG data stream.

Existing CMS software like parsers, generators etc. could deal with this format and hence be able to understand and verify signed PNG data streams.

Since the format would rely on an adopted standard, it would be **robust and perennial** and also completely independent of future updates of the PNG format. In addition, the CMS format itself is designed to be extensible concerning future cryptographic algorithms without its syntax having to be modified.

Since the CMS and the PNG format are designed to support **streamed processing**, signature generation and verification does not require exhaustive processing and buffering capabilities. Consequently, even **huge images** could be processed on limited devices, e.g. handhelds.

3.1.2 Disadvantages

Using a container structure like an enveloping CMS signed-data format makes the original format **unreadable** for existing PNG processing tools (editors and/or viewers). Consequently, special handling would be required for making the signed content readable again, e.g. by additionally providing some kind of PNG wrapping/unwrapping tools.

However, processing PNG data streams in this way is not useful at all. In particular, it does not permit a digitally signed PNG to be published together with its signature on a website for processing by ordinary web browsers. Thus, an enveloping PNG signature would have a substantial **impact with regard to interoperability**.

Consequently, the enveloping signature approach would only be applicable to environments not requiring a high level of interoperability.

3.2 Detached Signature

This approach stores digital signatures separately of the signed data stream.

3.2.1 Advantages

This approach does not require any special format support for signing PNGs. A reasonable specification of the PNG-specific signing and verification processing would suffice to support digitally signed PNGs.

PNG signature information could be stored using any format, e.g. the CMS standard, which also provides for storing signed data objects separately of their signature information.

Existing tools and environments would not be impacted.

3.2.2 Disadvantages

Storing signed data separately of its signature information generally causes logistic problems over time when dealing with large amounts of data.

It would be additionally necessary to always link both elements involved and to set up a suitable infrastructure for signing, storing, exchanging and verifying PNG data streams.

Setting up such an infrastructure would increase processing overhead and also result in a major impact on overall PNG handling when dealing with signatures.

3.3 Enveloped Signature

This approach embeds a digital signature in the original document (image). In order to do this, the document (image) format in question has to provide a storage structure (sub-element) for digital signature information.

Unfortunately PNG has no defined standard for embedding digital signature information. Nevertheless, PNG is extendable and offers a simple way of adding a dedicated extension chunk in order to embed a digital signature in a PNG data stream (cf. sections below for technical details).

3.3.1 Advantages

Using the original document (image) format with an embedded signature provides compatibility with existing tools. In the case of PNG, decoders or viewers would discard a new, hence unknown extension of this type and treat an attached signature as if it weren't there. (cf. *error checking part* of the PNG standard: An unknown chunk type is **not** to be treated as an error unless it is a critical chunk.) As a consequence, digitally signed **PNGs** could **envelope their own signature** and at the same time be processed and/or viewed without impacting existing tools.

Example: the above-cited use case — which stores signed images on a website for optionally checking their signature — would be possible and still **interoperate with** existing **browsers**. By contrast, software aware of such an extension could check the

stored information and make the verification result transparent to the user, e.g. a special browser filter could filter a web page and exclude those images that are not properly signed.

Note that a significant feature of this approach, i.e. using PNG extensions, is that the viewable **content** of the image would **not** be **modified** when attaching a signature and consequently be **fully preserved in its original form**.

Other image signing approaches exist which integrate digital signatures in viewable content, e.g. by using unused alpha channels or other superposition methods. These approaches effectively modify the original content. Although the modification may not be visible, these approaches would cause subtle changes of the original document (image), thus potentially impacting legal validity.

3.3.2 Disadvantages

Embedded signatures (according to the proposal below) **cannot be used** for **recursively** signing a PNG in the sense that new signatures comprise the document **including** its previous signatures. This general restriction, however, is not important for most typical applications, because such scenarios are rarely used. In addition, there is a valid workaround for such purposes by embedding a PNG data stream with an enveloped signature into an enveloping signature (cf. section 3.1). By contrast, the "**multiple signers**" use case is **supported** by the enveloped signature approach.

Embedded signatures **rely on** their **container format**. Consequently, software developed for signing and verifying signatures is format-specific and cannot be re-used in a generic way.

4 SOLUTION PROPOSAL

After having pondered the implications of the above-discussed options, we **propose** using the **enveloped signature** approach for PNG. An important feature of this alternative is the sizable advantage of being interoperable with the large set of existing PNG viewers and editors.

In addition, we suggest using CMS (cf. RFC 3852 and RFC 4853 for details) as the embedded signature information container format. This method provides additional advantages:

- As already mentioned, CMS is a well-known, widely adopted international standard and robust against format changes even when new cryptographic algorithms are used in the future.
- Using an adopted standard for storing sensitive information avoids the possible risk of potential security issues which might be caused by using a proprietary storage format.
- This approach would still opt in favor of using enveloping, detached and enveloped signatures in any acceptable combination.

5 REGISTRATION REQUEST

In order to support enveloped digital signatures for the PNG standard, we request that the new PNG chunk type specified below be registered as an optional and non-visible part of the data format for storing signature information in a generic manner.

5.1 Format Specification

This proposal specifies support for **PNG digital signatures** in order to protect them against tampering, e.g. when storing PNGs on a website or in an archive. It also specifies support for PNG **message authentication codes**, thus addressing protection of PNG data streams during transmission between two or more parties. Although both options are addressed, the major focus is directed to digital signatures.

Note that tampering protection concerns an entire PNG data stream. There is a subtle aspect that PNG image chunks may be **re-ordered** under certain conditions and still result in the same visual representation. However, the tampering protection support proposed here would result in a signature being breached by re-ordering chunks in any way. This **strong approach** has been chosen due to the fact that the PNG standard does not define any equivalence relationship for chunk re-ordering of a given PNG.

Also note that this specification proposal supports **multiple signers in parallel** and/or **repeated signing** with optionally preserving existing signatures. However, it does not support use cases where a signed PNG is signed by another signer including the previous signature. For this purpose, an enveloping CMS signed-data content type of a signed PNG should be used (cf. RFC 3852 and RFC 4853 for details).

The above-cited technical limitation, which, however, doesn't impact typical use cases, is given by design, i.e. due to using an **enveloped signature approach** in contrast to an enveloping approach. Note that an important feature of the enveloped approach is its compatibility with existing editors and viewers, so that a signed PNG does not require any special processing for being visualized.

We propose the format (extension chunk) specified below, which is in accordance with the existing PNG standard and compatible with existing processing tools. (*Note that parts in italics supply explanatory hints only and do not specify additional rules.*)

5.1.1 Chunk Type

The new chunk type should be **ancillary**, as indicated by the lower-case beginning of its name.

Although the chunk content concerns the entire PNG data stream, this requirement exists due to the fact that digitally signing a PNG data stream should not be mandatory. Additionally, it provides compatibility with existing processing tools, i.e. editors and viewers, which should bypass optional chunks not understand by these tools.

5.1.2 Text Keyword

The new optional PNG extension chunk should be named **iSIG** ("*Signature Information*"), with the chunk type's precise hexadecimal coding being **0x69 0x53 0x49 0x47.**

*The chunk should be used for **optionally** storing digital signature information concerning the entire PNG data stream.*

5.1.3 Ordering and Multiplicity

The new **iSIG** chunk type should be the previous last chunk in a PNG. It may occur several times in a single PNG. In this latter case, no special **ordering** is imposed among the individual **iSIG** chunks, yet they should immediately follow one other without any different chunk type in-between.

This requirement opts for calculating a PNG digest with a minimum buffering overhead. Note that due to the PNG standard, the image trailer has be the very last chunk and hence requires some buffering, e.g. when verifying a digital PNG signature.

Consequently, this proposal supports streaming capability of PNG signature processing (signing and signature verification), which is an important aspect concerning processing performance and handling of sizable image data.

Multiple iSIG chunks allow for simultaneously attaching digital signatures and authentication codes to a PNG or also for repeatedly signing a PNG data stream, with multiple signing processes being reflected in the individual iSIG chunks.

5.1.4 Data Format

The overall format of an **iSIG** chunk should be exactly as specified in the PNG standard, i.e. comprising the type, length, data and checksum parts.

The data part of an **iSIG** chunk should be a **signed-data content type** or an **authenticated-data content type** according to the Cryptographic Message Syntax Specification (CMS) (cf. RFC 3852 and RFC 4853 for details), the above-cited container types being described in detail in the respective sections 5 and 9 of the aforementioned RFC 3852.

Note that this format is used for compatibility with adopted signature standards and hence compatibility with existing implementations capable of processing corresponding data and also compatibility with future evolutions of those standards, which are designed to be extensible without being broken by new algorithms.

Additionally, this format avoids possible security issues which might arise if a proprietary storage format were to be chosen instead. CMS, by contrast, is a popular, well-known, generic and extensible international standard which has been validated in the course of many years.

For both above-cited content types, the '**external signature**' variant should be chosen in line with section 5.2 of RFC 3852.

The **message digest** of a PNG must always be calculated over the entire original PNG, i.e. over each chunk including all length, type, data and CRC bytes in exactly the same order as they appear in the original PNG data stream, but **excluding** any **iSIG** chunk.

*Note that a **timestamp signature** can optionally be added (cf. **SigningTime** attribute in section 11.3 of RFC 3852).*

A PNG signature provided by this specification proposal on the basis of a CMS embedded in the new **iSIG** extension chunk should be named "**enveloped PNG signature**", in contrast to a possible enveloping PNG signature as described above.

5.2 Processing Notes

An enveloped PNG signature is designed to be generated and verified using streaming capabilities, as is the case for PNG data streams.

A signature is an optional feature of a PNG; by the same token an enveloped PNG signature processor is a supplementary processing device and consequently doesn't check the correctness of the PNG it is processing. Consequently, a PNG signature processor acts on top of other PNG processing, thus **delegating checking** PNG **format correctness** to the underlying PNG processing tools, i.e. viewers and/or editors. An enveloped PNG signature processor only considers correct signature handling, while respecting the entire PNG standard.

Note that this not a requirement but only a recommendation so as not to duplicate efforts when developing PNG signature processors and not to merge format and signature aspects, which also addresses performance aspects and flexibility for future PNG format extensions. By contrast, a signature processor provider may optionally combine signature generation/verification and PNG format validation.

Hence, there could be PNG data streams with a correct signature but in an incorrect PNG format, e.g. having incorrectly ordered chunks. However, such a situation is not a security issue but rather purely a format issue, which could be detected by an ordinary PNG decoder. Generally speaking, one can sign completely meaningless content using a valid digital signature and vice versa.

Also note that due to the strong approach, signing PNGs should take place after a PNG data stream has been generated. Similarly, signature verification should always be performed of the original PNG data stream before any viewer or editor re-orders chunks due to their own processing strategy.

5.2.1 Signature Generation

For this purpose, an enveloped PNG signature processor should act as a PNG data stream filter in the following manner:

- Read a PNG data stream as the input and calculate the PNG message digest over all bytes read in the order they appear:
    - Start processing with the leading PNG signature
      Note that the word "signature" here possesses the semantics specified in the PNG standard, not the semantics of a digital signature.
    - For each chunk, read the chunk's type and length bytes and continue reading "blindly" until reaching the end of the chunk (including the checksum).
      Note that chunks are read and processed byte by byte exactly in the order they appear in the input. In addition, length bytes are interpreted as specified in the PNG standard, e.g. network byte order has to be respected.

The chunk's checksum does not have to be verified by a signature processor (cf. above-cited note concerning separation of aspects).

- If the current chunk is not the image trailer, write each byte read to the output applying exactly the same order. Otherwise, buffer the image trailer after using it for digest calculation.

- A processor can either retain or discard an existing **iSIG** chunk, depending on its configuration options.
  In any event, the bytes of an existing **iSIG** chunk are never taken into account for calculating the digest.
  Note that questions on how to deal with previously existing signature information extend beyond the scope of this proposal. A PNG signature processing tool should decide whether to retain or delete previously existing signatures according to its processing policies.

- Calculate the signed-data or authenticated-data content type based on the calculated digest and private signing key(s) and other items, e.g. signed attributes. Store the content type (also comprising certificate(s) corresponding to the signing key(s), timestamp signature(s) etc.) to the output, correctly formatted as a new **iSIG** chunk.

- Write the buffered image trailer to the output.

  Note that when integrating PNG signature generation with PNG editors, the "you should only sign what you see" paradigm should be honored, e.g. by visualizing the PNG to be signed and making the signature generation process transparent to the user.

5.2.2 Signature Verification

For this purpose an enveloped PNG signature processor should act as a PNG data stream filter in a manner similar to that described in the previous section.

- However, instead of generating **iSIG** chunks, the respective chunk contents are analyzed and retained.

- When digest calculation is completed after the image trailer has been read, the available signed-data and/or authenticated-data content types are verified in the context of the newly calculated digest.

- If verification is successful, additional checks have to be performed according to the available signer information, e.g. PKIX check for certificates and timestamp verification.

- If all checks are successful, the overall process is successful with respect to the attached digital signature and/or authentication code(s). Otherwise, it has to fail.
  Note that PNG viewers might be made aware of the outcome of an integrated signature verification process so that they could optionally visualize verification results and associated information for the user's convenience.