ODF 1.3 Proposal: Protection-Key Enhancements

Proposal v1.05

11 June 2012

This proposal introduces two protection-key authentication techniques that mitigate (but do not eliminate) risks associated with use of hash-code digest algorithms when the derived protection-key values are in plain view in XML elements of ODF documents.

The fundamental defect mitigated by the enhancements is that protection-key values in ODF 1.2 documents are not secret; current protection-key values are amenable to malicious re-use, including but not limited to discovery of the associated password.

Specification of these provisions in an early Committee Specification Draft for ODF 1.3 secures an opportunity for experimental introduction in ODF 1.2 extended documents and confirmation of practical utility and reliability of the additional techniques. The proposed default method, AUTHZ160, can be applied immediately in ODF 1.2 and also ODF 1.0/1.1 conforming documents without any limitation (see section C. Deployment Considerations).

This proposal does not reply on any support beyond that already expected of ODF consumers and producers for SHA1, HMAC-SHA-1, PBKDF2, and the generation of cryptographically-random strings of bytes.

Table of Contents

1 Rationale

1.1 Vulnerability of Password Hash Values

The use of password hashes in easily-discovered XML element and attribute values is subject to compromise of the hashed password. Despite the use of increasingly-stronger digest algorithms to lengthen the time required for carrying out brute-force attempts to match the hash to a password, passwords must be assumed to be compromised when the hash values used become known. This is the case when passwords are used to derive protection-key hash values that are recorded in ODF documents.

Recent (June 2012) attacks against SHA1 password hashes confirm that discovery of passwords for SHA1 hashes has become commonplace. Attacks against discovered SHA1 hashes are now being crowd-sourced among hacker groups on the Internet. Once a password that has a given hash becomes known, the password and its hash become widely available on the internet, reducing the detection of the password to a database-look-up by members of hacker communities.

These considerations are reason enough to deprecate the carrying of password hash values within XML elements of ODF documents for any purpose whatsoever.

An additional difficulty is that ODF protection-key digest values are easily removed/replaced. An extracted digest value can be re-purposed where the same hash procedure is used without having to know the password. The digest value is all that is needed. Extracted digests can also be published in compilation of known digest values for use in exploits and for comparison with digests associated with already-known or brute-force-guessed passwords.

There are also "transitive attack vectors" enabled by exposed digest values. For example, there is a place in the encryption procedures for ODF 1.0-1.2 documents where a password's SHA1 digest is used as an intermediate value. This becomes a point where known SHA1 digest values can be inserted in an attack on the encryption. In this case, it is not necessary to discover the password. Once a successful decryption has been discovered, it is possible to counterfeit other encryptions using that hash as if the same password was used. It is highly desirable to arrange for the protection- key values in ODF documents to be useless in transitive attacks of this kind.

1.2 SHA1DK for Password-Based Protection-Key Values

The first algorithm, SHA1DK, uses a customized password-based key-derivation procedure to derive a unique protection-key value that does not correspond to any conventional hash. It is extremely costly but not impossible for a determined adversary to successfully attack an SHA1DK protection-key value. The key vulnerability is the tendency for continued use of weak passwords. In addition, the class of weak passwords is growing as attack methodologies improve.

Introduction of SHA1DK allows passwords to continue being used in deriving protection-key values; it is not invulnerable.

Using SHA1DK makes it extremely unlikely that the same protection-key value will ever be derived in a separate use of the same password by anyone. That makes it very difficult for a culprit to detect situations where the same password is used by inspection of protection-key values alone.

In addition, SHA1DK is designed so that the resulting protection-key value is not usable anywhere that a digest of other form is required in ODF documents, including in cryptographic procedures. SHA1DK protection key values are resistant to transitive attacks so long as SHA1DK usage is confined to derivation of protection-key values (in ODF documents).

The SHA1DK procedure is also designed so that SHA1 digests in existing protection-key values can be replaced by SHA1DK protection-key values without requiring the password to be known. It is not feasible to recover the SHA1 digest from an SHA1DK derived-key value.

Those are the primary benefits of SHA1DK. There is no mitigation of the use of the value as a protection key on a forged or altered ODF document.

[Note: An extension, SHA1DKX, allows, identification of the use case to be incorporated in the derivation of a digest, so that transitive use of the digest itself is impeded; different use-case identifications will lead to different SHA1DKX protection-key (or other password-authentication) codes. Substitution of SHA1DKX can be considered if that is seen to be important. Although this limits the context in which an SHA1DKX digest can be used for authentication of a password, there is no protection against either SHA1DKX or SHA1DK digests being created by use of a known SHA1 digest or, equivalently, a known password.]

1.3 AUTHZ160 for Password-Less Protection-Key Values

AUTHZ160 is the default. It is the recommended technique for deriving protection-key values. This method is to be understood when there is a protection-key attribute and an accompanying protection-key-digest-algorithm attribute is absent.

The only constraint on AUTHZ160 implementations is that there be no relationship whatsoever between a chosen password and the protection-key value. This also means that there is no interoperable means to "unlock" the associated protection simply by presentation of a password.

AUTHZ160 is not specified beyond requiring that a password is not used to derive the protection-key value. AUTHZ160 protection-key values cannot be directly attacked to discover a password used for its authentication: there is no such password.

The prospect that the protection-key value itself can be misused in some way by use on different or altered ODF documents is not eliminated.

The simplest approach to creation of an AUTHZ160 protection-key value is by generating a 160-bit cryptographically-random value. This satisfies the essential condition that there be no password association.

Any authentication of requests to remove the protection set by use of an AUTHZ160 protection-key value is implementation defined.

[Note: The first 160 bits of an SHA1DK (better, SHA1DKX) derived key satisfies the conditions for use as an AUTHZ160 protection-key value. The remaining 160 bits are all that is dependent on the password and the full SHA1DK (or SHA1DKX) derived key can be protected elsewhere by secret means.]

2 Proposed Changes

Section/ Subsection	Instruction
Front Matter	⟦*Add to end of Declared XML Namespaces*⟧ ⟦ ... ⟧ urn:oasis:names:tc:opendocument:xmlns:of:1.2 http://docs.oasis-open.org/ns/office/1.2/meta/odf# http://docs.oasis-open.org/ns/office/1.3/protection#
1.3	⟦*Insert in the Normative References immediately following the entry for [OWL]* ⟧ **[PKCS5]** *PKCS #5 v2.0: Password-Based Cryptography Standard*. RSA Laboratories. March 25, 1999. Available on the Internet at <http://www.rsa.com/rsalabs/node.asp?id=2127>. Also available as IETF Informational RFC 2898 at <http://www.ietf.org/rfc/rfc2898.txt>
19.697	⟦*Change the end of the table:protection-key section. Correct the data type in the schema as well.*⟧ ⟦ ... ⟧ ~~The~~Any password shall be provided as a sequence of bytes in UTF-8 encoding. Consumers shall not silently ignore the presence of a `table:protection-key` attribute. A consumer that does not recognize or support the specified `table:protection-key-digest-algorithm` (including the default) shall not accept any password for release of the protection key. Any other means for removal of a `table:protection-key`, and the associated protection is implementation-defined. [Note: Protection locks are neither password security, document security, nor document integrity provisions. There is no assurance of confidentiality of any password employed in creation of protection-key values. In addition, protection-key attributes themselves are easily defeated by removing/replacing them in the XML elements that contain them. The protection-key settings can also be extracted and employed for unintended purposes.] The `table:protection-key` attribute is usable with the following elements: `<office:spreadsheet>` 3.7 and `<table:table>` 9.1.2. The `table:protection-key` attribute has the data type ~~`string`~~`base64Binary` 18.2

Section/ Subsection	Instruction
19.698	⟦*Insert text in the table:protection-key-digest-algorithm section following the second paragaph.*⟧ ⟦ ... ⟧ The password shall be provided as a sequence of bytes in UTF-8 encoding. Algorithm AUTHZ160 is defined as follows: The value encoded in the protection-key attribute consists of exactly 20 octets (160 bits) of binary data not derived from an user-provided password. This value shall be different for every occurrence of a protection-key attribute in the document and the value shall be indistinguishable from a cryptographically-random value. Any means by which consumers recognize the producer of the protection-key value and authenticate a request to remove the protection is implementation-defined. [Note: The binary data can simply be a cryptographically-random value. Use of a protection-key value that is not derived from a password avoids compromise of passwords by exposing their digest values in ODF documents.] Algorithm SHA1DK is defined as follows: The value encoded in the protection-key attribute consists of exactly 40 octets (320 bits) of binary data. The first 20 octets are a cryptographically-random value, *salt*. The *salt* value shall be different for each occurrence of a protection-key attribute in the ODF document. The final 20 octets are a cryptographically-derived value, *dk* (for derived key). The *dk* value is derived from *salt* and the UTF-8 encoding, *password*. Removal of the protection lock is authorized on acceptance of a password for which SHA1DK produces a match to *dk*. *dk* = SHA1DK(*password*, *salt*) = PBKDF2(*PRF*, SHA1(*password*), *salt*, *c*, *dkLen*) where dkLen = 20, the number of octets to be produced for *dk*, *PRF* is the underlying pseudorandom function HMAC-SHA-1 for use by PBKDF2, *c* = F*(*26+*salt*[0]*)* is the iteration count for the PBKDF2 algorithm, *salt*[0] is the initial octet of *salt* interpreted as an unsigned binary integer, F(*n*) is the *n*-th Fibonacci number (with F(26) = 121393) and PBKDF2 is as defined in [PKCS5]. If the value of *c* exceeds an implementation-defined threshhold for a consumer's practical derivation of *dk*, *dk* shall not be derived and no *password* value shall be accepted. For a producer that implements SHA1DK, the value of *salt* shall be such that *c* is within an implementation-defined range for demanding but practical derivations of *dk* by that producer. [Note: The challenge is to have the *dk* derivation computationally lengthy enough to discourage repeated attacks on the password while being rapid enough to be acceptable in creation and in the intended use of SHA1DK.]

Section/ Subsection	Instruction
19.698	⟦*Change text in the table:protection-key-digest-algorithm section following the previous insertion.*⟧ ⟦ ... ⟧ Any other procedures, their identifying IRIs, and their application to derivation of `table:protection-key` values are implementation-defined. Consumers shall support the digest algorithms identified by IRIs ~~http://www.w3.org/2000/09/xmldsig#sha1~~`http://docs.oasis-open.org/ns/office/1.3/protection#authz160` (for AUTHZ160), ~~which is the default,~~ `http://docs.oasis-open.org/ns/office/1.3/protection#sha1dk` (for SHA1DK), `http://www.w3.org/2000/09/xmldsig#sha1`, and `http://www.w3.org/`~~2000/09/xmldsig~~`2001/04/xmlenc#sha256`. They may support other algorithms described in §5.7 of [xmlenc-core] or alternative procedures. ~~Producers should use http://www.w3.org/2000/09/xmldsig#sha256.~~ Producers that support `table:protection-key` shall support AUTHZ160 and should do so by omitting the associated `table:protection-key-digest-algorithm` attribute. Producers should support SHA1DK, using the explicit IRI `http://docs.oasis-open.org/ns/office/1.3/protection#sha1dk`. All digest algorithms of §5.7 in [xmlenc-core] are deprecated for use as `table:protection-key-digest-algorithm` methods. Producers should not employ those algorithms and their IRI values. [Note: Removal of all support for those methods is anticipated in a future version of this specification.] The default value for this attribute is ~~`http://www.w3.org/2000/09/xmldsig#sha1`~~`http://docs.oasis-open.org/ns/office/1.3/protection#authz160`

Section/ Subsection	Instruction
19.850	⟦*Change the end of the text:protection-key section. Correct the data type in the schema as well.*⟧ ⟦ ... ⟧ ~~The~~Any password shall be provided as a sequence of bytes in UTF-8 encoding. Consumers shall not silently ignore the presence of a `text:protection-key` attribute. A consumer that does not recognize or support the specified `text:protection-key-digest-algorithm` (including the default) shall not accept any password for release of the protection key. Any other means for removal of a `text:protection-key` and the associated protection is implementation-defined. [Note: Protection locks are neither password security, document security, nor document integrity provisions. There is no assurance of confidentiality of any password employed in creation of protection-key values. In addition, protection-key attributes themselves are easily defeated by removing/replacing them in the XML elements that contain them. The protection-key settings can also be extracted and employed for unintended purposes.] The `text:protection-key` attribute is usable with the following elements: `<text:alphabetical-index>` 8.8, `<text:bibliography>` 8.9, `<text:illustration-index>` 8.4, `<text:index-title>` 8.2.3, `<text:object-index>` 8.6, `<text:section>` 5.4, `<text:table-index>` 8.5, `<text:table-of-content>` 8.3 and `<text:user-index>` 8.7. The `text:protection-key` attribute has the data type ~~`string`~~`base64Binary` 18.2

Section/ Subsection	Instruction
19.851	⟦*Change text in the text:protection-key-digest-algorithm section following the second paragaph.*⟧ ⟦ ... ⟧ The password shall be provided as a sequence of bytes in UTF-8 encoding. The additional IRIs for the procedures AUTHZ160 and SHA1DK are used in accordance with the definition of those procedures in 19.698. Any other procedures, their identifying IRIs, and their application to derivation of `text:protection-key` values are implementation-defined. Consumers shall support the digest algorithms identified by IRIs ~~http://www.w3.org/2000/09/xmldsig#sha1~~`http://docs.oasis-open.org/ns/office/1.3/protection#authz160` (for AUTHZ160), ~~which is the default,~~ `http://docs.oasis-open.org/ns/office/1.3/protection#sha1dk` (for SHA1DK), `http://www.w3.org/2000/09/xmldsig#sha1`, and `http://www.w3.org/`~~`2000/09/xmldsig`~~`2001/04/xmlenc#sha256`. They may support other algorithms described in §5.7 of [xmlenc-core] or alternative procedures. ~~Producers should use http://www.w3.org/2000/09/xmldsig#sha256.~~ Producers that support `text:protection-key` shall support AUTHZ160 and should do so by omitting the associated `text:protection-key-digest-algorithm` attribute. Producers should support SHA1DK, using the explicit IRI `http://docs.oasis-open.org/ns/office/1.3/protection#sha1dk`. All digest algorithms of §5.7 in [xmlenc-core] are deprecated for use as `text:protection-key-digest-algorithm` methods. Producers should not employ those algorithms and their IRI values. [Note: Removal of all support for those methods is anticipated in a future version of this specification.] The default value for this attribute is ~~`http://www.w3.org/2000/09/xmldsig#sha1`~~`http://docs.oasis-open.org/ns/office/1.3/protection#authz160`

3 Deployment Considerations

The protection-key extensions are designed so that an implementation of any version of ODF that supports at least (or only) the default SHA1 method will not be disturbed when longer or different protection-key values are employed. Password authentication will fail, as intended.

3.1 Down-level Considerations

It is sufficient that arbitrarily-long base64Binary encoding of the protection-key value be tolerated and only the expected size be used from that value down-level.

ODF 1.2 consumers are already faced with the prospect of unexpected protection-key lengths and unknown digest algorithm names. ODF 1.0/1.1 consumers may have greater difficulty depending on the resiliency of their implementations.

The desired situation is that any user-interface attempt to remove a lock for which the protection-key is derived by an unknown method will simply fail to accept any offered password. This will be either because the protection-key value cannot be matched by an assumed digest algorithm or because the implementation does not recognize the identified digest algorithm or a specific protection-key value for a recognized algorithm.

3.2 Immediate Usability of AUTHZ160 for Default Protection Keys

In making AUTHZ160 the default protection-key method, the technique is immediately useable with down-level ODF 1.0/1.1/1.2 consumers. By omitting the protection-key-digest-attribute when AUTHZ160 is the method, down level consumers will attempt to authenticate the protection key with an SHA1 password hash. AUTHZ160 protection-key values have the correct format. However, authentication against an SHA1 password hash will fail. That is the intended result.

When producers employ AUTHZ160 for setting a protection lock, the user should be advised that the lock cannot be released by using and version of ODF consumer that expects a password-based authorization.

3.3 Confirmation of Resilient Down-Level Treatment

A series of resiliency tests are being developed in the repository of the OASIS ODF Interoperability and Conformance (OIC) TC so that consumers at all levels can be tested to confirm that these additional provisions and others of their kind already allowed for in ODF 1.2 and beyond will not break consumers and fail appropriately when a protection-unlock attempt fails.

3.4 Future-Proofing of Extended ODF 1.2 Consumers and Producers

The nature of AUTHZ160 usage as a default, with no use of a protection-key-digest-algorithm attribute, is indistinguishable to a down-level consumer and any future consumer that accepts a default 160-bit protection-key value.

For migration to use of the additional protection-key digest methods, with explicit protection-key-digest-algorithm values, it is desirable to have proof of concept and trial use in advance of ratification of new ODF 1.3 and versions beyond ODF 1.3.

For provisional use in ODF 1.2 consumers and producers, the IRIs for AUTHZ160 and SHA1DK specified in this proposal shall not be used. Producers should not use these IRIs except in conformance with approved OASIS Standard ODF specifications that provide for their use.

Instead, there can be agreement on provisional IRIs, not using an OASIS namespace, for implementation-defined use in *extended* ODF 1.2 documents. This is allowed in the ODF 1.2 specification. These extended-document protection-key-digest-algorithm IRIs can be implementation-defined to identify the use of AUTHZ160 and SHA1DK exactly as specified in this proposal.

A convenience for ODF 1.2 consumers that recognize such provisional IRIs is to also accept the OASIS-namespace IRIs proposed for AUTHZ160 and SHA1DK. However, ODF 1.2 producers cannot produce those IRIs in ODF 1.2 documents.

This arrangement allows the ratification of these IRIs and the associated methods in ODF 1.3 to be usable down-level in ODF 1.2 consumers by ODF 1.3 producers using the official IRIs, once an ODF 1.3 Standard is approved.

The one risk to having ODF 1.2 consumers anticipate the enhanced protection-key IRIs and associated methods is if ODF 1.3 specifies those IRIs but alters the methods.

This risk is avoided by the ODF TC agreeing that the IRIs specified in this proposal will not be employed in ODF 1.3 to identify any methods other than those proposed here. The adoption of the methods in this proposal is not required. Adoption of different methods, with different IRIs is also not prevented.

In the outside case, so long as future-proofing is maintained, some of the provisional implementations will continue to be for extended ODF 1.2 documents and are likely to be usable in *extended* ODF 1.3 documents as well.