#### modifyVerifier (Modification Verifier)

This element specifies the write protection settings which have been applied to a PresentationML document. Write protection refers to a mode in which the document's contents cannot be modified, and the document cannot be resaved using the same file name.

When [present](present.docx), the application shall require a password to enable modifications to the document. If the supplied password does not match the hash value in this attribute, then write protection shall be enabled. If this element is omitted, then no write protection shall be applied to the current document. Since this protection does not encrypt the document, malicious applications may circumvent its use.

The password supplied to the algorithm is to be a Unicode string; strings longer than 255 characters are truncated to 255 characters. The attributes of this element specify the algorithm to be used to verify the password provided by the user.

[Example: Consider a PresentationML document that can only be opened in a write protected state unless a password is provided, in which case the file would be opened in an editable state. This requirement would be specified using the following PresentationML:

<p:documentProtection …
 p:cryptAlgorithmClass="hash" p:cryptAlgorithmType="typeAny"
 p:cryptAlgorithmSid="1" p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" ... />

In order for the hosting application to enable edits to the document, the hosting application would have to be provided with a password that the hosting application would then hash using the algorithm specified by the algorithm attributes and compare to the value of the hashData attribute (9oN7nWkCAyEZib1RomSJTjmPpCY=). If the two values matched, the file would be opened in an editable state. end example]

|  |
| --- |
| Parent Elements |
| [presentation](presentation.docx) (§) |

|  |  |
| --- | --- |
| Attributes | Description |
| algIdExt (Cryptographic Algorithm Extensibility) | Specifies that a cryptographic algorithm which was not defined by this Office Open XML Standard has been used to generate the hash value stored with this document.This value, when [present](present.docx), shall be interpreted based on the value of the algIdExtSource attribute in order to determine the algorithm used, which shall be application-defined. [Rationale: This extensibility affords the fact that with exponentially increasing computing power, documents created in the future will likely need to utilize as yet undefined hashing algorithms in order to remain secure. end rationale]If this value is [present](present.docx), the cryptAlgorithmClass, cryptAlgorithmType, and cryptAlgorithmSid attribute values shall be ignored in favor of the algorithm defined by this attribute. [Example: Consider a PresentationML document with the following information stored in its protection element:<p:… p:algIdExt="0000000A"  p:algIdExtSource="futureCryptography" p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" />The algIdExt attribute value of 0000000A specifies that the algorithm with hex code A shall be used as defined by the futureCryptography application. end example]The possible values for this attribute are defined by the XML Schema unsignedInt datatype. |
| algIdExtSource (Algorithm Extensibility Source) | Specifies the application which defined the algorithm value specified by the algIdExt attribute.[Example: Consider a PresentationML document with the following information stored in one its protection element:<p:… p:algIdExt="0000000A"  p:algIdExtSource="futureCryptography" p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" />The algIdExtSource attribute value of futureCryptography specifies that the algorithm used here was published by the futureCryptography application. end example]The possible values for this attribute are defined by the XML Schema string datatype. |
| cryptAlgorithmClass (Cryptographic Algorithm Class) | Specifies the class of cryptographic algorithm used by this protection. [Note: The initial version of this Office Open XML Standard only supports a single version - hash - but future versions may expand this as necessary. end note][Example: Consider a PresentationML document with the following information stored in its protection element:<p:… p:cryptAlgorithmClass="hash"  p:cryptAlgorithmType="typeAny"  p:cryptAlgorithmSid="1" p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" />The cryptAlgorithmClass attribute value of hash specifies that the algorithm used for the password is a hashing algorithm. end example]The possible values for this attribute are defined by the [ST\_AlgClass](ST_AlgClass.docx) simple type (§). |
| cryptAlgorithmSid (Cryptographic Hashing Algorithm) | Specifies the specific cryptographic hashing algorithm which shall be used along with the saltData attribute and user-supplied password in order to compute a hash value for comparison.The possible values for this attribute shall be interpreted as follows:

|  |  |
| --- | --- |
| Value | Algorithm |
| 1 | MD2 |
| 2 | MD4 |
| 3 | MD5 |
| 4 | SHA-1 |
| 5 | MAC |
| 6 | RIPEMD |
| 7 | RIPEMD-160 |
| 8 | Undefined. Shall not be used. |
| 9 | HMAC |
| 10 | Undefined. Shall not be used. |
| 11 | Undefined. Shall not be used. |
| 12 | SHA-256 |
| 13 | SHA-384 |
| 14 | SHA-512 |
| Any other value | Undefined. Shall not be used. |

[Example: Consider a PresentationML document with the following information stored in its protection element:<p:… p:cryptAlgorithmClass="hash"  p:cryptAlgorithmType="typeAny"  p:cryptAlgorithmSid="1" p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" />The cryptAlgorithmSid attribute value of 1 specifies that the SHA-1 hashing algorithm shall be used to generate a hash from the user-defined password. end example]The possible values for this attribute are defined by the XML Schema unsignedInt datatype. |
| cryptAlgorithmType (Cryptographic Algorithm Type) | Specifies the type of cryptographic algorithm used by this protection. [Note: The initial version of this Office Open XML Standard only supports a single type - typeAny - but future versions may expand this as necessary. end note][Example: Consider a PresentationML document with the following information stored in its protection element:<p:… p:cryptAlgorithmClass="hash"  p:cryptAlgorithmType="typeAny"  p:cryptAlgorithmSid="1" p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" />The cryptAlgorithmType attribute value of typeAny specifies that any type of algorithm may have been used for the password. end example]The possible values for this attribute are defined by the [ST\_AlgType](ST_AlgType.docx) simple type (§). |
| cryptProvider (Cryptographic Provider) | Specifies the cryptographic provider which was used to generate the hash value stored in this document. If the user provided a cryptographic provider which was not the system's built-in provider, then that provider shall be stored here so it can subsequently be used if available.If this attribute is omitted, then the built-in cryptographic provider on the system shall be used.[Example: Consider a PresentationML document with the following information stored in its protection element:<p:… p:cryptProvider="Krista'sProvider" p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" />The cryptProvider attribute value of Krista'sProvider specifies that the cryptographic provider with name "Krista's Provider" shall be used if available. end example]The possible values for this attribute are defined by the XML Schema string datatype. |
| cryptProviderType (Cryptographic Provider Type) | Specifies the type of cryptographic provider to be used.[Example: Consider a PresentationML document with the following information stored in its protection element:<p:… p:cryptProviderType="rsaAES" p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" />The cryptProviderType attribute value of rsaAES specifies that the cryptographic provider type shall be an Advanced Encryption Standard provider. end example]The possible values for this attribute are defined by the [ST\_CryptProv](ST_CryptProv.docx) simple type (§). |
| cryptProviderTypeExt (Cryptographic Provider Type Extensibility) | Specifies that a cryptographic provider type which was not defined by this Office Open XML Standard has been used to generate the hash value stored with this document.This value, when [present](present.docx), shall be interpreted based on the value of the cryptProviderTypeExtSource attribute in order to determine the provider type used, which shall be application-defined. [Rationale: This extensibility affords the fact that with exponentially increasing computing power, documents created in the future will likely need to utilize as yet undefined cryptographic provider types in order to remain secure. end rationale]If this value is [present](present.docx), the cryptProviderType attribute value shall be ignored in favor of the provider type defined by this attribute. [Example: Consider a PresentationML document with the following information stored in its protection element:<p:… p:cryptProviderTypeExt="00A5691D"  p:cryptProvideTypeExtSource="futureCryptography" p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" />The cryptProviderTypeExt attribute value of 00A5691D specifies that the provider type associated with hex code A5691D shall be used as defined by the futureCryptography application. end example]The possible values for this attribute are defined by the XML Schema unsignedInt datatype. |
| cryptProviderTypeExtSource (Provider Type Extensibility Source) | Specifies the application which defined the provider type value specified by the cryptProviderTypeExt attribute.[Example: Consider a PresentationML document with the following information stored in its protection element:<p:… p:cryptProviderTypeExt="00A5691D"  p:cryptProvideTypeExtSource="futureCryptography" p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" />The cryptProvideTypeExtSource attribute value of futureCryptography specifies that the provider type used here was published by the futureCryptography application. end example]The possible values for this attribute are defined by the XML Schema string datatype. |
| hashData (Password Hash) | Specifies the hash value for the password stored with this document. This value shall be compared with the resulting hash value after hashing the user-supplied password using the algorithm specified by the preceding attributes and parent XML element, and if the two values match, the protection shall no longer be enforced.If this value is omitted, then no password shall be associated with the protection, and it may be turned off without supplying any password.[Example: Consider a PresentationML document with the following information stored in its protection element:<p:… p:cryptAlgorithmClass="hash"  p:cryptAlgorithmType="typeAny"  p:cryptAlgorithmSid="1" p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" />The hashData attribute value of 9oN7nWkCAyEZib1RomSJTjmPpCY= specifies that the user-supplied password shall be hashed using the pre-processing defined by the parent element (if any) followed by the SHA-1 algorithm (specified via the cryptAlgorithmSid attribute value of 1) and that the resulting has value must be 9oN7nWkCAyEZib1RomSJTjmPpCY= for the protection to be disabled. end example]The possible values for this attribute are defined by the XML Schema string datatype. |
| saltData (Salt for Password Verifier) | Specifies the salt which was prepended to the user-supplied password before it was hashed using the hashing algorithm defined by the preceding attribute values to generate the hashData attribute, and which shall also be prepended to the user-supplied password before attempting to generate a hash value for comparison. A salt is a [random](random.docx) string which is added to a user-supplied password before it is hashed in order to prevent a malicious party from pre-calculating all possible password/hash combinations and simply using those precalculated values (often referred to as a "dictionary attack").If this attribute is omitted, then no salt shall be prepended to the user-supplied password before it is hashed for comparison with the stored hash value.[Example: Consider a PresentationML document with the following information stored in its protection element:<p:… p:saltData="ZUdHa+D8F/OAKP3I7ssUnQ==" p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" />The saltData attribute value of ZUdHa+D8F/OAKP3I7ssUnQ== specifies that the user-supplied password shall have this value prepended before it is run through the specified hashing algorithm to generate a resulting hash value for comparison. end example]The possible values for this attribute are defined by the XML Schema string datatype. |
| spinCount (Iterations to Run Hashing Algorithm) | Specifies the number of times the hashing function shall be iteratively run (using each iteration's result as the input for the next iteration) when attempting to compare a user-supplied password with the value stored in the hashData attribute. [Rationale: Running the algorithm many times increases the cost of exhaustive search attacks correspondingly. Storing this value allows for the number of iterations to be increased over time to accommodate faster hardware (and hence the ability to run more iterations in less time). end rationale][Example: Consider a PresentationML document with the following information stored in its protection element:<p:… p:spinCount="100000" p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" />The spinCount attribute value of 100000 specifies that the hashing function shall be run one hundred thousand times to generate a hash value for comparison with the hash attribute. end example]The possible values for this attribute are defined by the XML Schema unsignedInt datatype. |

The following XML Schema fragment defines the contents of this element:

<complexType name="CT\_ModifyVerifier">

 <attribute name="cryptProviderType" type="[ST\_CryptProv](ST_CryptProv.docx)" use="required"/>

 <attribute name="cryptAlgorithmClass" type="[ST\_AlgClass](ST_AlgClass.docx)" use="required"/>

 <attribute name="cryptAlgorithmType" type="[ST\_AlgType](ST_AlgType.docx)" use="required"/>

 <attribute name="cryptAlgorithmSid" type="xsd:unsignedInt" use="required"/>

 <attribute name="spinCount" type="xsd:unsignedInt" use="required"/>

 <attribute name="saltData" type="xsd:string" use="required"/>

 <attribute name="hashData" type="xsd:string" use="required"/>

 <attribute name="cryptProvider" type="xsd:string" use="optional"/>

 <attribute name="algIdExt" type="xsd:unsignedInt" use="optional"/>

 <attribute name="algIdExtSource" type="xsd:string" use="optional"/>

 <attribute name="cryptProviderTypeExt" type="xsd:unsignedInt" use="optional"/>

 <attribute name="cryptProviderTypeExtSource" type="xsd:string" use="optional"/>

</complexType>