| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SignedObject.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/security/SignatureSpi.html)   [**NEXT CLASS**](http://docs.google.com/java/security/Signer.html) | [**FRAMES**](http://docs.google.com/index.html?java/security/SignedObject.html)    [**NO FRAMES**](http://docs.google.com/SignedObject.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#4d34og8) |

## **java.security**

Class SignedObject

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **java.security.SignedObject**

**All Implemented Interfaces:** [Serializable](http://docs.google.com/java/io/Serializable.html)

public final class **SignedObject**extends [Object](http://docs.google.com/java/lang/Object.html)implements [Serializable](http://docs.google.com/java/io/Serializable.html)

SignedObject is a class for the purpose of creating authentic runtime objects whose integrity cannot be compromised without being detected.

More specifically, a SignedObject contains another Serializable object, the (to-be-)signed object and its signature.

The signed object is a "deep copy" (in serialized form) of an original object. Once the copy is made, further manipulation of the original object has no side effect on the copy.

The underlying signing algorithm is designated by the Signature object passed to the constructor and the verify method. A typical usage for signing is the following:

Signature signingEngine = Signature.getInstance(algorithm,  
 provider);  
 SignedObject so = new SignedObject(myobject, signingKey,  
 signingEngine);

A typical usage for verification is the following (having received SignedObject so):

Signature verificationEngine =  
 Signature.getInstance(algorithm, provider);  
 if (so.verify(publickey, verificationEngine))  
 try {  
 Object myobj = so.getObject();  
 } catch (java.lang.ClassNotFoundException e) {};

Several points are worth noting. First, there is no need to initialize the signing or verification engine, as it will be re-initialized inside the constructor and the verify method. Secondly, for verification to succeed, the specified public key must be the public key corresponding to the private key used to generate the SignedObject.

More importantly, for flexibility reasons, the constructor and verify method allow for customized signature engines, which can implement signature algorithms that are not installed formally as part of a crypto provider. However, it is crucial that the programmer writing the verifier code be aware what Signature engine is being used, as its own implementation of the verify method is invoked to verify a signature. In other words, a malicious Signature may choose to always return true on verification in an attempt to bypass a security check.

The signature algorithm can be, among others, the NIST standard DSA, using DSA and SHA-1. The algorithm is specified using the same convention as that for signatures. The DSA algorithm using the SHA-1 message digest algorithm can be specified, for example, as "SHA/DSA" or "SHA-1/DSA" (they are equivalent). In the case of RSA, there are multiple choices for the message digest algorithm, so the signing algorithm could be specified as, for example, "MD2/RSA", "MD5/RSA" or "SHA-1/RSA". The algorithm name must be specified, as there is no default.

The name of the Cryptography Package Provider is designated also by the Signature parameter to the constructor and the verify method. If the provider is not specified, the default provider is used. Each installation can be configured to use a particular provider as default.

Potential applications of SignedObject include:

* It can be used internally to any Java runtime as an unforgeable authorization token -- one that can be passed around without the fear that the token can be maliciously modified without being detected.
* It can be used to sign and serialize data/object for storage outside the Java runtime (e.g., storing critical access control data on disk).
* Nested SignedObjects can be used to construct a logical sequence of signatures, resembling a chain of authorization and delegation.

**See Also:**[Signature](http://docs.google.com/java/security/Signature.html), [Serialized Form](http://docs.google.com/serialized-form.html#java.security.SignedObject)

| **Constructor Summary** | |
| --- | --- |
| [**SignedObject**](http://docs.google.com/java/security/SignedObject.html#SignedObject(java.io.Serializable,%20java.security.PrivateKey,%20java.security.Signature))([Serializable](http://docs.google.com/java/io/Serializable.html) object, [PrivateKey](http://docs.google.com/java/security/PrivateKey.html) signingKey, [Signature](http://docs.google.com/java/security/Signature.html) signingEngine)            Constructs a SignedObject from any Serializable object. |

| **Method Summary** | |
| --- | --- |
| [String](http://docs.google.com/java/lang/String.html) | [**getAlgorithm**](http://docs.google.com/java/security/SignedObject.html#getAlgorithm())()            Retrieves the name of the signature algorithm. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**getObject**](http://docs.google.com/java/security/SignedObject.html#getObject())()            Retrieves the encapsulated object. |
| byte[] | [**getSignature**](http://docs.google.com/java/security/SignedObject.html#getSignature())()            Retrieves the signature on the signed object, in the form of a byte array. |
| boolean | [**verify**](http://docs.google.com/java/security/SignedObject.html#verify(java.security.PublicKey,%20java.security.Signature))([PublicKey](http://docs.google.com/java/security/PublicKey.html) verificationKey, [Signature](http://docs.google.com/java/security/Signature.html) verificationEngine)            Verifies that the signature in this SignedObject is the valid signature for the object stored inside, with the given verification key, using the designated verification engine. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Constructor Detail** |
| --- |

### SignedObject

public **SignedObject**([Serializable](http://docs.google.com/java/io/Serializable.html) object,  
 [PrivateKey](http://docs.google.com/java/security/PrivateKey.html) signingKey,  
 [Signature](http://docs.google.com/java/security/Signature.html) signingEngine)  
 throws [IOException](http://docs.google.com/java/io/IOException.html),  
 [InvalidKeyException](http://docs.google.com/java/security/InvalidKeyException.html),  
 [SignatureException](http://docs.google.com/java/security/SignatureException.html)

Constructs a SignedObject from any Serializable object. The given object is signed with the given signing key, using the designated signature engine.

**Parameters:**object - the object to be signed.signingKey - the private key for signing.signingEngine - the signature signing engine. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during serialization [InvalidKeyException](http://docs.google.com/java/security/InvalidKeyException.html) - if the key is invalid. [SignatureException](http://docs.google.com/java/security/SignatureException.html) - if signing fails.

| **Method Detail** |
| --- |

### getObject

public [Object](http://docs.google.com/java/lang/Object.html) **getObject**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html),  
 [ClassNotFoundException](http://docs.google.com/java/lang/ClassNotFoundException.html)

Retrieves the encapsulated object. The encapsulated object is de-serialized before it is returned.

**Returns:**the encapsulated object. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurs during de-serialization [ClassNotFoundException](http://docs.google.com/java/lang/ClassNotFoundException.html) - if an error occurs during de-serialization

### getSignature

public byte[] **getSignature**()

Retrieves the signature on the signed object, in the form of a byte array.

**Returns:**the signature. Returns a new array each time this method is called.

### getAlgorithm

public [String](http://docs.google.com/java/lang/String.html) **getAlgorithm**()

Retrieves the name of the signature algorithm.

**Returns:**the signature algorithm name.

### verify

public boolean **verify**([PublicKey](http://docs.google.com/java/security/PublicKey.html) verificationKey,  
 [Signature](http://docs.google.com/java/security/Signature.html) verificationEngine)  
 throws [InvalidKeyException](http://docs.google.com/java/security/InvalidKeyException.html),  
 [SignatureException](http://docs.google.com/java/security/SignatureException.html)

Verifies that the signature in this SignedObject is the valid signature for the object stored inside, with the given verification key, using the designated verification engine.

**Parameters:**verificationKey - the public key for verification.verificationEngine - the signature verification engine. **Returns:**true if the signature is valid, false otherwise **Throws:** [SignatureException](http://docs.google.com/java/security/SignatureException.html) - if signature verification failed. [InvalidKeyException](http://docs.google.com/java/security/InvalidKeyException.html) - if the verification key is invalid.

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SignedObject.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/security/SignatureSpi.html)   [**NEXT CLASS**](http://docs.google.com/java/security/Signer.html) | [**FRAMES**](http://docs.google.com/index.html?java/security/SignedObject.html)    [**NO FRAMES**](http://docs.google.com/SignedObject.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#4d34og8) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).