**SecurityLevel**

Many cryptographic primitives and schemes have different security levels. For example, an encryption scheme can be CPA-secure (secure against chosen-plaintext attacks) or CCA-secure (secure against chosen-ciphertext attacks). The security level of a cryptographic entity is specified by making the implementing class of the entity declare that it implements a certain security level; for example, an encryption scheme that is CCA-secure will implement the Cca interface. Different primitives have different families that define their security levels (e.g., hash functions, MACs, encryption). It is often the case that different security levels of a given primitive form a hierarchy (e.g., any CCA-secure encryption scheme is also CPA-secure), and in this case they extend each other. Thus, it suffices to implement a Cca interface and this immediately implies that a Cpa interface is also implied.

All of the interfaces expressing a security level are marker interfaces that define types of security level and do not have any functionality.

**EncSecLevel**

This hierarchy specifies the security level of encryption schemes; it does not differentiate between symmetric and asymmetric encryption. There are two sub-hierarchies for encryption. The first relates to the adversarial power and includes Eav (eavesdropping adversary), CPA (chosen-plaintext attack), CCA1 (preprocessing chosen-ciphertext attack), and CCA2 (full chosen-ciphertext attack). The second relates to the aim of the attack and includes Indistinguishable (for the standard indistinguishability notion) and NonMalleable; note that non-malleability implies indistinguishability and thus the NonMalleable interface extends the Indistinguishable interface.

**Eav**

An encryption scheme that is only secure for eavesdropping adversaries (like a stream cipher) should implement this interface. It is also necessary to specify if such a scheme is Indistinguishable or NonMalleable.

**Cpa**

An encryption scheme that is secure in the presence of chosen-plaintext attacks should implement this interface. It is also necessary to specify if such a scheme is Indistinguishable or NonMalleable.

**Cca1**

An encryption scheme that is secure in the presence of preprocessing chosen-ciphertext attacks (meaning that the decryption oracle is available only before the challenge ciphertext is provided) should implement this interface. It is also necessary to specify if such a scheme is Indistinguishable or NonMalleable.

**Cca2**

An encryption scheme that is secure in the presence of (full) chosen-ciphertext attacks should implement this interface. Note that any Cca2 scheme is both Indistinguishable and NonMalleable. Thus, Cca2 extends both Cca1 and NonMalleable, and it suffices to have a CCA2-secure scheme implement only the Cca2 interface.

**Indistinguishable**

This interface should be used when the security level of the encryption scheme is according to the regular indistinguishability game that defines privacy.

**NonMalleable**

This interface should be used for encryption schemes that achieve non-malleability, meaning that it is infeasible for an adversary to generate a related ciphertext. Non-malleability always implies indistinguishability.

**ChannelSecLevel**

This interface is the root interface of the channel security level family. The different security levels of a channel can be: Plain, Encrypted, Authenticated and EncAuth (which means encrypted and authenticated).

**HashSecLevel**

This hierarchy specifies the security level of a cryptographic hash function. The levels in this hierarchy are TargetCollisionResistant and CollisionResistant.

**TargetCollisionResistant**

A hash function is target collision resistant if it is infeasible for an adversary to succeed in the following game: the adversary chooses a message x; next a random key K is chosen for the hash function and given to the adversary; finally the adversary outputs some y (not equal to x) such that H\_K(x)=H\_K(y).

Observe that this notion is of relevance for KEYED hash functions (note that the key is public, but randomly chosen).

**CollisionResistant**

A hash function H is collision resistant if it is infeasible to find two distinct values x and y such that H(x)=H(y).

**MacSigSecLevel**

This hierarchy specifies the security level of a message authentication code (MAC) or digital signature scheme. The hierarchy here only refers to the number of times that the MAC or signature scheme can be used; namely, OneTime or UnlimitedTimes. We do not currently have another interface for a bounded but not unlimited number of times; if necessary this can be added later. We also consider by default adaptive chosen-message attacks and so have not defined a separate hierarchy for adaptive/non-adaptive attacks and chosen versus random message attacks.

**OneTime**

Any MAC or signature scheme that is secure for one-time use only should implement this interface.

**UnlimitedTimes**

Any MAC or signature scheme that is secure for an unlimited number of uses should implement this interface. This is the security level of standard MAC and signature schemes.

**ProtocolSecLevel**

This interface is the root interface of the security level hierarchy for (secure computation) protocols. There are three different subhierarchies in this family. The first relates to the adversary’s capabilities and includes Semihonest, Malicious and Covert. The second relates to the question of composition and includes StandAlone and UC (universally composable). The third relates to the corruption strategy of the adversary and includes AdaptiveWithErasures and AdaptiveNoErasures (if no interface here is implemented then static security is assumed).

**SemiHonest**

Any protocol that is secure in the presence of semi-honest adversaries should implement this interface.

**Covert**

Any protocol that is secure in the presence of covert adversaries should implement this interface. We stress that the deterrent parameter is not guaranteed by the interface.

**Malicious**

Any protocol that is secure in the presence of malicious adversaries should implement this interface.

**StandAlone**

Any protocol that is proven secure in the stand-alone model (where secure protocols are run sequentially and not concurrently) should implement this interface.

**UC**

Any protocol that is universally composable (aka. UC-secure) should implement this interface.

**AdaptiveWithErasures**

Any protocol that is secure in the presence of an adaptive adversary, with the assumption that honest parties can securely erase data, should implement this interface.

**AdaptiveNoErasures**

Any protocol that is secure in the presence of an adaptive adversary, without assuming that honest parties can securely erase data, should implement this interface.

**Perfect**

This interface should be used for any primitive, scheme or protocol that achieves perfect security (e.g., a perfect MAC, one-time pad, perfectly-secure multiparty protocol, and so on). Note that by default, all schemes are computationally secure and so not implementing Perfect or Statistical should be interpreted as computational security.

**Statistical**

This interface should be used for any primitive, scheme or protocol that achieves statistical security (e.g., a perfect MAC, one-time pad, perfectly-secure multiparty protocol, and so on). Note that by default, all schemes are computationally secure and so not implementing Perfect or Statistical should be interpreted as computational security.

**DlogSecLevel**

This hierarchy specifies the security level of a cyclic group in which “discrete log” hardness is assumed to hold. The levels in this hierarchy are Dlog, CDH and DDH.

**Dlog**

A group in which the discrete log problem is assumed to hold should implement this interface.

**CDH**

A group in which the computational Diffie-Hellman problem is assumed to hold should implement this interface.

**DDH**

A group in which the decisional Diffie-Hellman problem is assumed to hold should implement this interface.