(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
07 November 2019 (07.11.2019)

WIPO | PCT

(10) International Publication Number
WO 2019/212580 A1

(51) International Patent Classification:
G06F 21/53 (2013.01) *H04L 29/06* (2006.01)
H04L 12/24 (2006.01) *G06Q 20/32* (2012.01)
G06F 21/74 (2013.01)

(21) International Application Number:
PCT/US2018/042684

(22) International Filing Date:
18 July 2018 (18.07.2018)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/664,463 30 April 2018 (30.04.2018) US

(71) Applicant: **GOOGLE LLC** [US/US]; 1600 Amphitheatre Parkway, Mountain View, CA 94043 (US).

(72) Inventors: **SAPEK, Anna**; 1600 Amphitheatre Parkway, Mountain View, CA 94043 (US). **SAVAGAONKAR, Uday**; 1600 Amphitheatre Parkway, Mountain View, CA 94043 (US). **ANDERSEN, Jeffrey, Thomas**; 1600 Amphitheatre Parkway, Mountain View, CA 94043 (US). **ROEDER, Thomas, Michael**; 1600 Amphitheatre Parkway, Mountain View, CA 94043 (US).

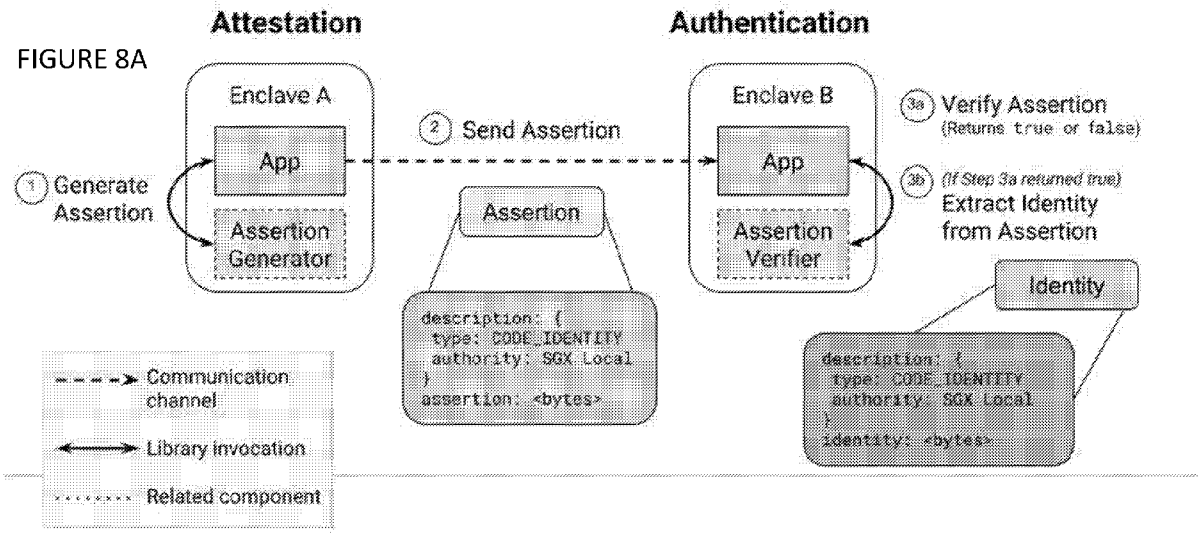
(74) Agent: **CACCIABEVE, Noelle, L.** et al.; Lerner, David, Littenberg, Krumholz & Mentlik, LLP, 600 South Avenue West, Westfield, NJ 07090 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM,

(54) Title: ENCLAVE INTERACTIONS

FIGURE 8A



(57) Abstract: Aspects of the disclosure provide various methods relating to enclaves. For instance, a method of authentication for an enclave entity with a second entity may include receiving, by one or more processors of a host computing device of the enclave entity, a request and an assertion of identity for the second entity, the assertion including identity information for the second identity; using an assertion verifier of the enclave entity to determine whether the assertion is valid; when the assertion is valid, extracting the identity information; authenticating the second entity using an access control list for the enclave entity to determine whether the identity information meets expectations of the access control list; when the identity information meets the expectations of the access control list, completing the request.

[Continued on next page]

WO 2019/212580 A1

TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

— *with international search report (Art. 21(3))*

ENCLAVE INTERACTIONS

CROSS-REFERENCE TO RELATED APPLICATION

[0001] The present application claims the benefit of the filing date of U.S. Provisional Application No. 62/664,463, filed April 30, 2018, the disclosure of which is hereby incorporated herein by reference.

BACKGROUND

[0002] Enclave technologies may enable software programmers to develop secure applications that are contained inside secure execution environments called enclaves. Enclaves can be identified by a code identity. An enclave's code identity encapsulates an enclave writer's expectation of the enclave's behavior, and typically comprises cryptographic information about the code that is running inside the enclave, the environment in which the code is run, and any additional parameters that affect code execution. Enclave code identity is fundamental to many of the secure operations performed by an enclave.

[0003] An application that runs inside an enclave typically has safeguards like memory and code integrity, and memory encryption. These safeguards protect the enclave from code that executes outside of it, like the operating system or hypervisor. Additionally, enclaves can provide verifiable attestations of the code that they are running. This is a key security property that enables trust within an enclave system. With these protections and security assurances, a programmer can write an application that first verifies that it is indeed running the expected code, performs some security-sensitive computations, and provisions or extracts any additional application secrets into/out of the enclave. A single-enclave system like this is sufficient for a simple application in which the application logic is self-contained and does not require communication with outside entities. However, this same security does not translate well into more complex systems that involve multiple enclave entities, both local and remote.

[0004] Some systems also provide for secret sealing which allows for a secret to be encrypted such that only an entity with specific identity can open it. The aforementioned enclave technologies may provide hardware support for provisioning keys that are bound to a specified subset of an enclave's identity. This may grant enclaves the ability to seal secrets such that they can only be unsealed by another enclave executing on the same machine whose identity matches the subset specified by the sealer machine.

SUMMARY

[0005] Aspects of the disclosure provide a method of authentication for an enclave entity with a second entity. The method includes receiving, by one or more processors of a host computing device of the enclave entity, a request and an assertion of identity for the second entity, the assertion including identity information for the second identity; using, by the one or more processors, an assertion verifier of the enclave entity to determine whether the assertion is valid; when the assertion is valid, extracting, by the one or more processors, the identity information; authenticating the second entity using, by the one or more processors, an access control list for the enclave entity to determine whether the identity information meets expectations of the access control list; and when the identity information meets the expectations of the access control list, completing, by the one or more processors, the request.

[0006] In one example, the second entity is a non-enclave entity. In another example, second entity is a second enclave entity. In another example, the assertion is not valid, denying the request. In another example, when the identity information does not meet the expectations of the access control list, denying the request. In another example, the method also includes using an assertion generator of the enclave entity to generate a second assertion including identity information for the enclave entity and sending the second assertion to the second entity for verification. In another example, the assertion verifier includes instructions for verifying assertions and extracting identities out of verified assertions. In another example, the assertion verifier provides a Boolean response that indicates whether the assertion is valid. In another example, the method also includes using an identity access control list evaluator to determine whether the identity information meets expectations of the access control list, and the identity access control list evaluator includes instructions for operating on a set of identities possessed by an entity and evaluating that set of identities against an access-control policy. In this example, the identity access control list evaluator provides a Boolean response that indicates whether the identity information meets the expectations of the access control list match. In addition, the access control list includes expressions of enclave identity expectations, and the identity expectations are used to determine whether the identity information meets expectations of the access control list. In addition or alternatively, the enclave identity expectations are configured as predicates that include two or more identity expectations via one or more logical operations. In addition or alternatively, each identity expectation includes a reference identity and a match

specification, and wherein at least one reference identity and at least one match specification are used to determine whether the identity information meets expectations of the access control list.

[0007] In another example, the enclave entity includes an enclave server including instructions configured to receive and complete the request and wherein the enclave server is used to complete the request. In another example, the identity information includes an identity description that classifies the identity of the assertion an identity type supported by the enclave entity and identifies an authority responsible for handling identities of that identity type. In this example, the authority is used to identity the assertion verifier in order to determine whether the assertion is valid. In addition or alternatively, the authority is used to identify a library in order to determine whether the identity information meets the expectations of the access control list. In addition or alternatively, the authority is used to identify a plugin including interfaces that in order to determine whether the assertion is valid and in order to determine whether the identity information meets the expectations of the access control list. In another example, the enclave entity and the second entity are located in local memory of the host computing device. In another example, the second entity is located in local memory of a second host computing device, the second host computing device being different from the host computing device.

[0008] Another aspect of the disclosure provides a method of establishing a communication channel between an enclave entity and a second entity. The method includes receiving, by one or more processors of a host computing device of the enclave entity, a request to initiate the communication channel from the second entity; negotiating, by the one or more processors, with the second entity a record protocol for the communication channel and a secret key for cryptographically protecting traffic sent over the communication channel; and after the negotiating is completed, communicating with the second entity using the communication channel using the record protocol and using the secret key to encrypt and authenticate data exchanged over the communication channel.

[0009] In one example, the method also includes conducting, by the one or more processors, an attestation and authentication process to verify an identity of the second entity, the attestation process including receiving an assertion of the second entity's identity and verifying the second entity's identity. In this example, the attestation and authentication process further includes sending an assertion of the enclave entity's identity to the second

entity. In another example, negotiating the record protocol occurs before conducting the attestation and authentication process. In this example, the method further includes negotiating one or more types of assertions of identity before conducting the attestation and authentication process such that the one or more types of assertions of identity are used in the attestation and authentication process. In this example, negotiating one or more types of assertions of identity includes identifying one or more types of assertions that the enclave entity is capable of making and verifying. In addition or alternatively, negotiating one or more types of assertions of identity includes receiving one or more types of assertions that the second entity is capable of making and verifying.

[0010] In another example, the request is received as a remote procedure call to the enclave entity. In another example, the record protocol is a message passing protocol that is used to send application-level data when communicating with the second entity using the communication channel. In another example, the method also includes conducting, by the one or more processors, an authentication and authorization process to verify an identity of the second entity, and at least part of the authentication and authentication process is conducted after the negotiating is complete. In this example, a first part of the authentication and authorization process includes verifying a received assertion of the second entity's identity. In another example, a second part of the authentication and authorization process includes authorizing the second entity using identity information extracted from the assertion. In this example, the first part is performed before the communicating using the communication channel and the second part is performed before the communicating using the communication channel. Alternatively, the method also includes aborting the communication channel when the enclave entity is unable to authenticate the second entity. In another example, the record protocol is negotiated before the secret key is negotiated. In another example, the second entity is a non-enclave entity. In another example, the second entity is a second enclave entity. In this example, the negotiating is conducted with an enclave server of the second enclave entity configured to an enclave server including instructions configured to receive and complete the request and wherein the enclave server is used to complete the request. In another example, the enclave entity and the second entity are located in local memory of the host computing device. In another example, In another example, the second entity is located in local memory of a second host computing device, the second host computing device being different from the host computing device.

[0011] Another aspect of the disclosure provides a method of sealing secrets in a first enclave entity of a host computing device. The method includes inputting, by one or more processors of the host computing device, to a sealing library of the first enclave entity a header and a secret, the header including an identity access control list for the secret; generating, by the one or more processors, using the sealing library, a key; sending, by the one or more processors, the key and the identity access control list to a second enclave entity; in response to the sending, receiving, by the one or more processors, from the second enclave entity a sealed version of the key; using, by the one or more processors, the sealing library to seal the secret; and appending the sealed version of the key and the identity access control list to the sealed secret.

[0012] In one example, the sealing library includes instructions for unsealing data by accepting sealed data and output the unsealed data. In another example, the method also includes establishing communications with the second enclave entity via a remote procedure call. In this example, the method also includes using the sealing library to identify a location of the second enclave entity in order to establish the communications. In another example, the received sealing key is independent of a host computing device of the second enclave entity. In another example, the method also includes using the identity access control list to authenticate the second enclave entity before sending the key. In another example, the first enclave entity does not have access to an unencrypted version of a master secret key used to generate the sealed version of the key such that the first enclave entity is unable to unseal the sealed version of the key. In another example, the method also includes sending the sealed secret and appended sealed version of the key to a third enclave entity. In this example, the secret is only able to be sent to the third enclave entity after the secret has been sealed. In another example, the identity access control list identifies one or more entities that are able to unseal the sealed secret. In another example, sealing the secret includes sealing the secret to a whitelist that allows any enclave entity whose identity fulfills at least one identity expectation in the identity access control list to unseal the sealed secret. In another example, sealing the secret includes sealing the secret to a whitelist that allows any enclave entity whose identity fulfills all identity expectations in the identity access control list to unseal the sealed secret.

[0013] In another example, the method also includes inputting to the sealing library of the first enclave entity a second sealed secret including an appended sealed version of a second

key; sending, by the one or more processors, a request to unseal the second sealed secret to a third enclave entity; receiving, by the one or more processors, an unsealed version of the second key; and using, by the one or more processors, the sealing library to unseal the second secret. In this example, the third enclave entity is one of a plurality of identical enclave instances all running identical code and each in possession of a shared master secret such that the first enclave entity is able to send the request to any of the plurality of identical enclave instances. In addition, if the third enclave entity is not available, sending the request to a fourth enclave entity. Also, the unsealed version of the second key is received from the fourth enclave entity, and the fourth enclave entity is one of the plurality of identical enclave instances. In addition or alternatively, the request is sent after a remote procedure call communication channel is established between the first enclave entity and the second enclave entity using an identity access control list for the second sealed secret. In addition or alternatively, the method also includes using the sealing library to identify a location of the third enclave entity in order to send the request to the third enclave entity. In another example, the second enclave entity is a remote sealing root, such that the second enclave entity is a different enclave from the first enclave entity. In another example, the first enclave entity and the second enclave entity are located in local memory of the host computing device. In another example, the second enclave entity is located in local memory of a second host computing device, the second host computing device being different from the host computing device.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] FIGURE 1 is a functional diagram of an example system in accordance with aspects of the disclosure.

[0015] FIGURE 2 is a functional diagram of aspects of the system of FIGURE 1.

[0016] FIGURES 3-7 are example representations of code in accordance with aspects of the disclosure.

[0017] FIGURES 8A-8B are an example flow of an attestation, authentication and authorization concepts in accordance with aspects of the disclosure.

[0018] FIGURE 9 is an example flow of a handshake protocol in accordance with aspects of the disclosure.

[0019] FIGURE 10 is an example configuration of a header in accordance with aspects of the disclosure.

[0020] FIGURE 11 is an example configuration of a sealed secret in accordance with aspects of the disclosure.

[0021] FIGURE 12 is an example flow of a secret sealing framework in accordance with aspects of the disclosure.

DETAILED DESCRIPTION

OVERVIEW

[0022] Aspects of the disclosure relate to identification and authentication of enclaves even where such enclave entities are backed by, or were developed using, different enclave technologies and exist on different machines. The identification and authentication features described herein may involve a plurality of different concepts, including attestation (where an enclave or non-enclave entity makes a claim about its identity), authentication (where one enclave can verify or confirm the properties of another enclave's or entity's identity), and authorization (where an enclave is provided access to information and/or operations based on that enclave's identity) as discussed in further detail below. As used herein, an identity may represent some unique characteristics of an entity. These characteristics may be used by other entities to draw inferences about the expected behavior of that entity. In a system involving multiple enclaves, an enclave can authenticate with an enclave peer by presenting an attestation of its code identity to that enclave. Attestation may be essential for establishing secure communication between two entities and also may enable higher-level systems to support authorization and access control.

[0023] A typical enclave should only have 1 type of code identity for attestation and authentication because the enclave is backed by 1 type of enclave technology. However, in more complex scenarios when a system includes multiple enclaves back by different types of technologies, there is a need to support other forms of attestation and, additionally, other types of identities. For instance, enclaves can possess other types of identities other than code identity such as certificates or tokens as well as other features unrelated to the code identity. Additionally, if an enclave system includes both non-enclave as well as enclave entities, the system may need to support identities other than code identities such as other cryptographic credentials, like tokens or certificates. In some instances, an enclave system may also include entities which may not possess any cryptographic credentials at all, and as such, the system may need to support the concept of an unauthenticated entity.. As some attestation mechanisms are CPU-specific and can only be used meaningfully within a

particular machine or computing device, there is also a need to provide alternative mechanisms of attesting code identity such that an attestation is meaningful to remote entities, or even non-enclave entities.

[0024] To enable the operations needed for implementing attestation, authentication, and authorization in an identity framework that supports many different types of identities, three roles or programming interfaces, including a generator, a verifier and an evaluator may be employed at each enclave in an enclave system. An Assertion Generator may include instructions for generating assertions on behalf of entities in the enclave system. In this regard, each Assertion Generator may provide an operation that generates an assertion that makes a claim of an identity. Assertions generated by these operations may be cryptographically-verifiable.

[0025] An Assertion Verifier may include instructions for verifying assertions, and extracting identities out of verified assertions. The Assertion Verifier may provide an operation that verifies an assertion of a specific type of identity. In that regard, an enclave may include different Assertion Verifiers for different types of entities.

[0026] An identity access control list (ACL) may include data defined by a user or programmer of the application that created the enclave in order to evaluate arbitrary logical expressions of enclave identity expectations. The identify ACLs provides a way for a user to specify exactly what properties are relevant for any given situation, relationship, or enclave. Each enclave may also include one or more Identity ACL Evaluators that include instructions for operating on a set of identities possessed by an entity and evaluating that set of identities against an access-control policy.

[0027] Each of the enclaves may also include an enclave client and an enclave server. The enclave clients may include instructions for generating remote procedure calls (RPCs) to other enclaves or entities within the enclave system. Similarly, the enclave server may include instructions for receiving and responding to requests generated by or from other enclaves or entities within the enclave system.

[0028] The identity framework may define common representations for identities, assertions, and identity expectations. For instance, the identity framework may include classes of identity or categories of identity supported by the identity framework as well as an authority designation which identifies or defines the entity responsible for handling a particular type of identity or assertion.

[0029] Different types of identities in the identity framework may share a common representation. For instance, each identity may have an Identity Description, which classifies the identity into one of the identity classes supported by the identity framework and also identifies the authority responsible for handling identities of that type. These common representations may be used when defining common operations on identities, assertions, and identity expectations. The operations for each of the aforementioned programing interfaces in the identity framework may also be defined in terms of these common representations rather than specific forms or instances of these constructs. By defining operations in terms of these common representations, it is possible to unite different identities, assertions, and identity expectations into a single, unified framework.

[0030] The identity framework described may be extensible in order to support various notions of enclave identity. Extensions to the framework may be provided in the form of plugins. A plugin may be a library that implements each of the three programming interfaces for a particular identity and assertion type. An application that leverages the identity framework may also make use of many plugins simultaneously. This may provide for a rich development environment that supports many types of enclave identity.

[0031] Communication within a system of multiple enclaves also poses a more challenging problem than a single-enclave application. A single-enclave application may only require exchanging information between the enclave and a local, untrusted caller. This problem is typically addressed by providing programmers with a simple message-passing mechanism as well as attestation primitives that can be used to develop a secure enclave application. In a multi-enclave system, it may be the case that the enclaves that need to communicate are on different machines and/or are backed by different enclave technologies. They must rely on untrusted communication mechanisms, such as UDP or TCP sockets to send and receive messages. To provide for additional security when establishing a communication channel, these enclaves may utilize a remote procedure call (RPC) security framework which provides for additional messages to enable the enclaves to perform mutual authentication/attestation, as well as agree on an encryption key that could be used to encrypt their communications.

[0032] One possible way of securing enclave communication is to modify the application-level protocol between the enclaves to incorporate enclave attestation in an ad-hoc way. For example, consider a protocol that enables Enclave A to retrieve a secret from Enclave B. The two enclaves could achieve this using a sequence of messages like the following:

Enclave A → Enclave B
Message 1: identifies the secret and requests a challenge

Enclave B → Enclave A
Message 2: contains a challenge

Enclave A → Enclave B
Message 3: provides a public key, and an attestation of Enclave A's identity that is bound to the challenge and the public key

Enclave B → Enclave A
Message 4: contains the actual secret encrypted with the public key

[0033] However, secure-protocol development is challenging to program, dangerously error-prone, and rests the majority of the security burden on the programmer. While there are some protocols available, these are limited in that they are specific to the technology on which an enclave is backed. For instance, Intel's Enhanced Privacy ID (EPID) protocol, which provides an authentication mechanism as well as key-exchange, only supports attestation for SGX enclaves. There are various other cryptographic handshake protocols that can be used to establish secure and authenticated channels, such as Transport Layer Security (TLS) or Application Layer Transport Security (ALTS). However, none of these protocols support authentication based on enclave code identity or more than one form of identity.

[0034] To address these limitations, the identity framework can be used to implement an RPC security system based on enclave identity by providing a handshake protocol that performs enclave-identity-based authentication during establishment of a communication channel. The handshake protocol may make use of various implementations of the Assertion Generator and Assertion Verifier to enable the exchange of assertions between two entities, thereby allowing the entities to mutually-authenticate. By using the identity framework described above, the handshake protocol may be able to enable a unified authentication mechanism that supports arbitrary types of enclave identities.

[0035] The RPC security system may be used wherein an RPC channel is secured based on enclave identities and RPCs are routed over that channel. In this regard, calls into or out of an enclave can be modelled as RPCs. This RPC security system can also be applied to calls between pair of enclaves or between enclaves and non-enclave entities that possess some cryptographic credentials.

[0036] To initiate the RPC channel, a handshake protocol may be performed during channel establishment. The handshake protocol may be used to achieve specific security guarantees on the subsequent established channel and may provide channel confidentiality, authentication using one or more enclave identities, and support for a higher level authorization system. The handshake protocol may be used during establishment of an RPC communication channel and may take place before the enclave client and enclave server exchange any application-level data. During the handshake protocol, the entities exchange and verify assertions, negotiate a record protocol, and negotiate a secret key that is used to cryptographically protect future traffic sent over the channel. After the handshake protocol completes, the enclave client and enclave server communicate using the record protocol and all data exchanged is encrypted with the secret key.

[0037] The programming interface that supports the RPC security system may be implemented as a library discussed above. The library may define the exact format and interpretation of the bytes of an assertion. Because of the use of a library, the format of the assertion, the assertion offer, and the assertion request need not be strictly specified by the handshake protocol, and the handshake protocol may operate on opaque blobs of bytes that represent assertions, assertion requests, and assertion offers. A simplified example of a possible protocol is described below.

[0038] The handshake protocol allows for various identities to be asserted during the exchange. The desired assertions could be configured on both the client and the server. These libraries could be registered with handshake through static registration, and all registered libraries could be added to the registration map and can be queried to check for the existence of a particular library. Using this registration map, the enclave client and enclave server (or an RPC server of a non-enclave entity) can generate and verify assertions depending on their configuration and on which libraries are available.

[0039] The handshake protocol supports authentication for various types of enclave identities and allows participants to exchange multiple identities during the authentication process. As with the identity framework, the RPC security system can be adapted to new identities by adding new Assertion Generators and Assertion Verifiers for those identities. After the handshake protocol has been completed, both the enclave client and enclave server (or an RPC server of a non-enclave entity) may access authentication properties of the connection in order to access the identity information of the other to proceed to authenticate one another.

[0040] As noted above, some systems also grant enclaves the ability to seal secrets such that they can only be unsealed by another enclave executing on the same machine whose identity matches the subset specified by the sealer. However, these features cannot be used to seal secrets to an enclave whose identity does not match the sealer's identity nor can such featuresbe used to seal secrets to enclaves executing on a different machine or computing device.

[0041] Thus, another possible application of the identity framework may include a secret sealing framework. The secret sealing framework may enable cryptographic-sealing of sensitive secrets within a multi-enclave system and disclosure of such secrets only to authorized entities in the system independent of the specific host machines or computing devices on which the entities reside.

[0042] The secret sealing frame work may involve a SecretSealer interface. A SecretSealer may accept a header, a secret, and additional authenticated data and output a sealed secret. This additional authenticated data may be encrypted during transit via the RPC communication channels described herein, but may be otherwise unencrypted. A SecretSealer may also include instructions for unsealing secrets by accepting a sealed secret and output a secret. The header may include metadata about the secret, as well as an identity ACL of enclave identities that are allowed to unseal the secret. This header may also be included in the sealed secret outputted by a SecretSealer and may be verified by a SecretSealer in order to output the secret.

[0043] The entity that actually implements the sealing and unsealing may be a sealing root and can either be a local or a remote entity. A local sealing root may be a root that executes in the same enclave as the SecretSealer. A remote sealing root may be a root that executes in a different enclave than the SecretSealer. The remote sealing root may function via the RPC communications channels and handshake protocols discussed herein. As such, a remote sealing root can make use of the Assertion Generator and Assertion Verifier interfaces to perform authentication with enclave clients that make requests to seal or unseal secrets. This also allows the remote root to cryptographically verify the identity of the enclave client, allowing the remote root to unseal secrets only when the requestor's identity matches the identity ACL present in the header of a sealed secret. In addition, an enclave client may leverage assertions by the remote sealing root and refuse to communicate with the remote root if its identity does not match the client's identity expectation(s).

[0044] The remote sealing root may include of a plurality of identical enclave instances all running identical code and each in possession of a shared master secret. This master secret is used to derive keys that seal individual secrets and may only be available in an unencrypted form within the remote sealing root enclave instances.

[0045] The identity ACL for a sealed secret may include of a set of enclave identity expectations as with the identity ACLs discussed above. Because the identity expectations that make up an ACL are grouped using an arbitrary nesting of logical operators, this may provide a SecretSealer with some flexibility when sealing a secret.

[0046] The features described here provide for complex multi-enclave systems that unite various notions of enclave identity and attestation into a single, technology-agnostic framework. These systems are especially useful for users that are looking to enclave technology for additional security for their applications or that want to alleviate concerns about running security-sensitive workflows in the cloud. The identity framework is also extensible and composable via plugins. For instance, the identity framework described herein can be expanded arbitrarily to add support for new kinds of identities. This extensibility is enabled via the framework's plugin model. As such, users can select which extensions to use in their applications, thereby enabling support for only the identities needed in their particular application.

[0047] In addition, the RPC security system features may allow for multi-enclave systems that are as secure as individual enclave applications. These features may be especially useful in situations in which an enclave does not know the location of the other entity (i.e. the recipient of the RPC) and scales very well to larger enclave systems making the RPC Security system especially useful for cloud-based computing. This is because the channel is secured so that only the entities involved in establishing the channel can read and write to the channel, the entities involved in establishing the channel exchange and verify each other's enclave identities, and both entities can set up authorization policies to control which calls can be made over the channel by authenticated entities. This RPC security system can be utilized in a multi-enclave system to enable secure and authenticated communication channels between two enclaves or enclaves and non-enclave entities in that system and also provide call-level authorization.

[0048] The secret sealing framework discussed herein also provides for sealing of secrets that is independent of the identity of the sealer and the hardware on which the secret is sealed. As

such, the secret sealing framework may enable cryptographic-sealing of sensitive secrets within a multi-enclave system and disclosure of such secrets only to authorized entities in the system independent of the specific host machines or computing devices on which the entities reside.

EXAMPLE SYSTEMS

[0049] FIGURE 1 includes an example enclave system 100 in which the features described herein may be implemented. It should not be considered as limiting the scope of the disclosure or usefulness of the features described herein. In this example, enclave system 100 can include computing devices 110, 120, 130 and storage system 140 connected via a network 150. Each computing device 110, 120, 130 can contain one or more processors 112, memory 114 and other components typically present in general purpose computing devices.

[0050] Although only a few computing devices and a storage systems are depicted in the system 100, the system may be expanded to any number of additional devices. In addition to a system including a plurality of computing devices and storage systems connected via a network, the features described herein may be equally applicable to other types of devices such as individual chips, including those incorporating System on Chip (Soc) or other chips with memory, that may include one or more enclaves.

[0051] Memory 114 of each of computing devices 110, 120, 130 can store information accessible by the one or more processors 112, including instructions that can be executed by the one or more processors. The memory can also include data that can be retrieved, manipulated or stored by the processor. The memory can be of any non-transitory type capable of storing information accessible by the processor, such as a hard-drive, memory card, ROM, RAM, DVD, CD-ROM, write-capable, and read-only memories.

[0052] The instructions can be any set of instructions to be executed directly, such as machine code, or indirectly, such as scripts, by the one or more processors. In that regard, the terms "instructions," "application," "steps," and "programs" can be used interchangeably herein. The instructions can be stored in object code format for direct processing by a processor, or in any other computing device language including scripts or collections of independent source code modules that are interpreted on demand or compiled in advance. Functions, methods, and routines of the instructions are explained in more detail below.

[0053] Data may be retrieved, stored or modified by the one or more processors 112 in accordance with the instructions. For instance, although the subject matter described herein

is not limited by any particular data structure, the data can be stored in computer registers, in a relational database as a table having many different fields and records, or XML documents. The data can also be formatted in any computing device-readable format such as, but not limited to, binary values, ASCII or Unicode. Moreover, the data can comprise any information sufficient to identify the relevant information, such as numbers, descriptive text, proprietary codes, pointers, references to data stored in other memories such as at other network locations, or information that is used by a function to calculate the relevant data.

[0054] The one or more processors 112 can be any conventional processors, such as a commercially available CPU. Alternatively, the processors can be dedicated components such as an application specific integrated circuit ("ASIC") or other hardware-based processor. Although not necessary, one or more of computing devices 110 may include specialized hardware components to perform specific computing processes, such as decoding video, matching video frames with images, distorting videos, encoding distorted videos, etc. faster or more efficiently.

[0055] Although Figure 1 functionally illustrates the processor, memory, and other elements of computing device 110 as being within the same block, the processor, computer, computing device, or memory can actually comprise multiple processors, computers, computing devices, or memories that may or may not be stored within the same physical housing. For example, the memory can be a hard drive or other storage media located in housings different from that of the computing devices 110. Accordingly, references to a processor, computer, computing device, or memory will be understood to include references to a collection of processors, computers, computing devices, or memories that may or may not operate in parallel. For example, the computing devices 110 may include server computing devices operating as a load-balanced server farm, distributed system, etc. Yet further, although some functions described below are indicated as taking place on a single computing device having a single processor, various aspects of the subject matter described herein can be implemented by a plurality of computing devices, for example, communicating information over network 150.

[0056] Each of the computing devices 110, 120, 130 can be at different nodes of a network 150 and capable of directly and indirectly communicating with other nodes of network 150. Although only a few computing devices are depicted in FIGURE 1, it should be appreciated that a typical system can include a large number of connected computing devices, with each different computing device being at a different node of the network 150. The network 150

and intervening nodes described herein can be interconnected using various protocols and systems, such that the network can be part of the Internet, World Wide Web, specific intranets, wide area networks, or local networks. The network can utilize standard communications protocols, such as Ethernet, WiFi and HTTP, protocols that are proprietary to one or more companies, and various combinations of the foregoing. Although certain advantages are obtained when information is transmitted or received as noted above, other aspects of the subject matter described herein are not limited to any particular manner of transmission of information.

[0057] Like the memory discussed above, the storage system 140 may also store information that can be accessed by the computing devices 110, 120, 130. However, in this case, the storage system 140 may store information that can be accessed over the network 150. As with the memory, the storage system can include any non-transitory type capable of storing information accessible by the processor, such as a hard-drive, memory card, ROM, RAM, DVD, CD-ROM, write-capable, and read-only memories.

[0058] In this example, the instructions of each of computing devices 110, 120, 130 may include one or more applications. These applications may define enclaves 160, 170, 180, 190 within memory, either locally at memory 114 or remotely at the storage system 140. Each enclave may be "hosted" by the hardware on which the enclave is stored. For instance, computing device 110 may be a host computing device for enclaves 160 and 170, and computing device 120 may be a host computing device of enclave 180. Each enclave can be used to store data and instructions while at the same time limit the use of such data and instructions by other applications. For instance the data may include sensitive information such as passwords, credit card data, social security numbers, or any other information that a user would want to keep confidential. And, as discussed further below, the instructions may be used to limit the access to such data. Although computing device 110 includes only two enclaves, computing device 120 includes only 1 enclave, computing device 130 includes no enclaves, and storage system 140 includes only 1 enclave, any number of enclaves may be defined with the memory of the computing devices 110, 120, storage system 140, or any other devices of system 100.

[0059] Each of these enclaves may be considered an "entity" of the enclave system 100. Similarly, applications of computing device 130 or the computing device 130 or other devices

or application, such fixed function security ASICs, smart cards, as well as remote services hosted in the cloud, may also be considered "non-enclave" entities.

[0060] As an entity, the enclaves may have or be used to generate identities and assertions that may enable attestation and authentication in an identity framework of the enclave system 100. An assertion may be a cryptographically-verifiable claim of an identity. However, to enable authorization, an identity expectation, which includes a base reference identity, and a match specification that specifies a subset of the reference identity that is relevant to the expectation may be used. In this regard, an identify expectation may represent an expectation of an identity or rather how an entity within the enclave system would state an expectation of what another entity's identity should be.

[0061] To enable the operations needed for implementing attestation, authentication, and authorization in the identity framework, three roles or programming interfaces, including a generator, a verifier and an evaluator may be employed at each enclave in the enclave system 100. For instance, turning to FIGURE 2, enclaves 160, 170, 180, 190 each include one or more Assertion Generators 210, 212, 214, 216. Each enclave may include different types of Assertion Generators for different types of assertions as each type of identity may have many different types of assertions. An Assertion Generator may include instructions for generating assertions on behalf of entities in the enclave system. In this regard, each Assertion Generator may provide an operation that generates an assertion that makes a claim of an identity. Assertions generated by these operations may be cryptographically-verifiable.

[0062] In addition, a verifier can verify the assertion to determine whether the claim of an assertion is valid. For instance, each enclave 160, 170, 180, 190 includes one or more Assertion Verifier(s) 220, 222, 224, 226. The Assertion Verifier may include instructions for verifying assertions, and extracting identities out of verified assertions. The Assertion Verifier may provide an operation that verifies an assertion of a specific type of identity. In that regard, an enclave may include different Assertion Verifiers for different types of identities. In operation, given an assertion, an Assertion Verifier may determine whether or not the claim of an identity in an assertion is valid. As an example, the Assertion Verifier may return a Boolean response (i.e. a yes/no answer) that indicates whether the assertion can be verified. If the assertion is valid and is verified, the Assertion Verifier may also extract the identity from the assertion.

[0063] Each enclave 160, 170, 180, 190 may also include an identity access control list (ACL) 230, 232, 234, 236. Each of these identity ACLs may include data defined by a user or programmer of the application that created the enclave in order to evaluate arbitrary logical expressions of enclave identity expectations. An enclave identity expectation consists of a reference identity and a match specification. As such, the identity ACLs provides a way for a user to specify exactly what properties are relevant for any given situation, relationship, or enclave. For instance, an enclave identity I would be considered to fulfill identity expectation E if the match specification subset of E's reference identity is identical to the match specification subset of I.

[0064] Each enclave 160, 170, 180, 190 may also include one or more Identity ACL Evaluators 240, 242, 244, 246. These Identity ACL Evaluators may include instructions for operating on a set of identities possessed by an entity and evaluating that set of identities against an access-control policy. The access-control policy may be specified as a predicate over one or more identity expectations. At an implementation level, such predicates can be specified as one or more identity expectations combined together via arbitrary logical operations like AND, OR, and NOT.

[0065] The Identity ACL Evaluators may also provide an operation that matches an identity against an identity ACL predicate or baseline identity expectation. In other words, given an identity and an identity ACL predicate identity expectation, this Identity ACL Evaluator checks whether the identity matches the expectation. For instance, if the two are incompatible, the Identity ACL Evaluator may return "false". If the two are compatible but do not match, the Identity ACL Evaluator may also return "false". Otherwise, if the match checks out, the Identity ACL Evaluator may return "true".

[0066] One or more of the Identity ACL Evaluators in the enclave system 100 can be implemented utilizing one or more sub-operations. For instance, an identity expectation matcher may provide an operation that matches an identity against an identity expectation. This operation may apply the match specification to the reference identity and then perform the match. This may provide a Boolean result (i.e. a yes/no answer). This operation can be used to implement evaluation of complex logical predicates. The Identity ACL Evaluator can thus implement predicate evaluation in terms of this identity expectation matcher. As an example, consider an ACL L that chains together identity expectations A, B, and C in the following predicate: ((A OR B) AND C). The Identity ACL Evaluator evaluates a list of

identities N against this ACL by delegating each individual matching operation (e.g. Match N against A, Match N against B, Match N against C) to the identity expectation matcher. The same identity expectation matcher may handle all these matching operations, or each match can be handled by a different identity expectation matcher. An identity expectation matcher may also delegate some matching operations to other matchers and summarize the results.

[0067] Each of the enclaves may also include an enclave client 250, 252, 254, 256 and an enclave server 260, 262, 264, 266. The enclave clients may include instructions for generating remote procedure calls (RPCs) to other enclaves or entities within the enclave system 100. Similarly, the enclave server may include instructions for receiving and responding to requests generated by or from other enclaves or entities within the enclave system.

[0068] The identity framework may define common representations for identities, assertions, and identity expectations. For instance, the identity framework may include classes of identity or categories of identity supported by the identity framework as well as an authority designation, for example defined in an Identity or Authority Description for an identity or an assertion as shown in FIGURES 4 and 6, which identifies or defines the entity responsible for handling a particular type of identity or assertion. As an example, the identity framework may support two (or many more) classes of identity: code identity and certificate-based identity. As such, a code identity may include an "ABC" code identity and a "123" code identity, where ABC code identity is handled by the ABC authority and 123 code identity is handled by the 123 authority. An entity that possesses either the ABC code identity or the 123 code identity can have its identity represented in the identity framework. Similarly, an assertion may include an "XYZ" type assertion and a "456" type assertion, where XYZ type assertion is handled by the XYZ authority and 456 type assertion is handled by the 456 authority. Of course, ABC, 123, XYZ, and 456 are merely abstract examples; any number of additional identities and authorities with different designations may also be used. FIGURE 3 provides an example defining of supported classes of identities with a protocol buffer enumerator.

[0069] Identities in the identity framework may share a common representation. For instance, each identity may have an Identity Description, which classifies the identity into one of the identity types or classes supported by the identity framework and also identifies the authority responsible for handling identities of that type, appended to an actual identity.

The Identity Description allows different types of identities to be represented in a consistent fashion. For example, the Identity Description with an authority value could be represented with a protocol-buffer message as shown in FIGURE 4.

[0070] The actual identity may be an opaque blob of bytes that utilizes any underlying encoding or data representation. By examining an identity's Identity Description, the identity class for the identity may be determined and the blob may be decoded to extract other details of the identity. An identity can, for example, be represented by a protocol-buffer message as shown in FIGURE 5.

[0071] Assertions in the identity framework may have a similar representation to identities. Like an identity, an assertion may have an Assertion Description. An Assertion Description may classify the identity that is asserted in the assertion into one of the identity classes supported by the identity framework and may also identify the authority that handles assertions of that type. FIGURE 6 provides an example defining of Assertion Description with an authority value with a protocol buffer enumerator.

[0072] An assertion includes a type of identity, an authority, and a blob of bytes containing the actual cryptographically-verifiable claim of identity. In this regard, the blob is the actual assertion of an identity. The type of identity and authority indicate how to interpret the blob so that the blob can be passed on to the correct Assertion Verifier for verification. An assertion can, for example, be represented with the protocol-buffer message as shown in FIGURE 7.

[0073] These common representations may be used when defining common operations on identities, assertions, and identity expectations. The operations for each of the aforementioned programing interfaces in the identity framework may also be defined in terms of these common representations rather than specific forms or instances of these constructs. By defining operations in terms of these common representations, it is possible to unite different identities, assertions, and identity expectations into a single, unified framework.

[0074] The identity framework described may be extensible in order to support various notions of enclave identity. Extensions to the framework may be provided in the form of plugins. A plugin may be a library that implements each of the three programming interfaces for a particular identity and assertion type. In this regard, different libraries may be used for different types of assertions or for different types of identities, although if two types of assertions are very similar, they may be handled by the same library. Alternatively, a single

library may be able to handle more than one type of assertion. The libraries may be plugged into the identity framework so that it can be used in conjunction with the entire identity framework. An application that leverages the identity framework may also make use of many plugins simultaneously. This may provide for a rich development environment that supports many types of enclave identity.

[0075] This identity framework may be implemented using any programming language that supports some notion of abstract classes (interfaces). For illustrative purposes, the examples herein relate to an identity framework implemented in a C++ ecosystem, where Protocol Buffers are used as the data representation format for identities, assertions, and identity expectations, though other languages and data representation formats may also be employed. Each of the programming interfaces described above, can be defined within as a Java class or a C++ class where each operation exists as a virtual class method. A component library may also be used to provide a concrete implementation of each programming interface. For instance, a component library can be utilized directly, such as to generate a specific assertion or to perform an identity match, or may be used in the context of a higher-level system that makes use of the abstract identity and assertion features described above. For instance, the identity matching operations may require access to various concrete implementations on demand.

[0076] One possible approach may utilize a combination of delegation and static registration. First, a concrete implementation of the interface may be used to handle all possible identity types. This class may be referred to as a delegator. Within a given program exactly one instance of this class may be instantiated such that the class is a singleton. Every other concrete implementation may also be instantiated once and then inserted into an execution-unit-wide registration map. In this regard, an instance of the registration map may exist inside of the Enclave, and another instance of the registration map may exist outside of the Enclave. These instances may be identical or with minor differences, for instance, given the different locations. This registration map could be populated via static registration of the available component libraries. In order to perform an identity match, the delegator's matching operation may be called with the inputs. The delegator may query the registration map with the Identity Description to find the appropriate concrete implementation of an identity matcher for that identity. If such an implementation exists, the delegator may delegate the

call to the matching operation on the respective object. If an implementation does not exist, the delegator may return an error that indicates that the identity type is not supported.

[0077] FIGURES 8A and 8B provide a visual representation of the attestation, authentication and authorization concepts in operation within the identity framework described above. Enclave A and Enclave B may represent two different enclaves running on the same (such as enclaves 160 and 170) or different computing devices (such as enclaves 160 and 180) backed by the same or different enclave technologies and operating within the enclave system. In this regard, the actual processing of such information may be performed by the processors 112 of the computing devices on which the enclaves reside or which otherwise operate on the enclaves (for instance, remotely).

[0078] Turning to FIGURE 8A, at step (1), an application (via the one or more processors 112) at the computing device of Enclave A uses Enclave A's Assertion Generator to generate an assertion of Enclave A's identity. At step (2), the assertion is sent by Enclave A via a communication channel, for instance locally or over network 150, to the Enclave B. At step (3a), an application (via the one or more processors 112) at the computing device of Enclave B utilizes Enclave B's Assertion Verifier to verify the received assertion. If Enclave B's Assertion Verifier returns "false" the authentication process ends as the assertion from Enclave A is not valid. If Enclave B's Assertion Verifier returns "true," at step (3b), the Assertion Verifier extracts the identity from the assertion.

[0079] Turning to FIGURE 8B, at step 4, after authentication, an enclave client (RPC client in FIGURE 8B) of Enclave A may be used to initiate an RPC to Enclave B via a communication channel, for instance locally or over network 150. At step 5a, an RPC server of Enclave B may be used to extract the client identity (via the one or more processors 112) and at step 5b may use Enclave B's Identity ACL Evaluator to determine whether the identity matches the authorization policy or the ACL for the RPC server. If Enclave B's Identity ACL Evaluator returns "false" the RPC is denied as the extracted identity is not valid or rather does not meet the expectations defined in the ACL. If Enclave B's Identity ACL Evaluator verifier returns "true," at step (6), then the extracted identity does meet the expectations defined in the ACL. As such, the RPC server of Enclave B may complete the RPC that was requested by the enclave client of Enclave A.

[0080] The identity framework described above may be used in various ways. For instance, the identity framework can be used to implement an RPC security system based on enclave

identity by providing a handshake protocol that performs enclave-identity-based authentication during establishment of a communication channel. The handshake protocol may make use of various implementations of the Assertion Generator and Assertion Verifier to enable the exchange of assertions between two entities, thereby allowing the entities to mutually-authenticate. By using the identity framework described above, the handshake protocol may be able to enable a unified authentication mechanism that supports arbitrary types of enclave identities.

[0081] The RPC security system may be used wherein an RPC channel is secured based on enclave identities and RPCs are routed over that channel. In this regard, calls into or out of an enclave can be modelled as RPCs. For instance, an untrusted entity could make an RPC into an enclave that offers an API to untrusted callers. Similarly, in the reverse situation, the enclave could make an outgoing RPC to the host or local computing device in order to request some sort of system resource or to write a log message. The RPC security system can also be applied to calls between enclave entities. An enclave that wants to make a call into another enclave can issue an RPC to an enclave server running in that enclave. Additionally, this RPC security system can also be applied to calls between enclaves and non-enclave entities that possess some cryptographic credentials.

[0082] As noted above, to initiate the RPC channel, a handshake protocol may be performed during channel establishment. The handshake protocol may be used to achieve specific security guarantees on the subsequent established channel and may provide channel confidentiality, authentication using one or more enclave identities, and support for a higher level authorization system.

[0083] FIGURE 9 is an example flow of messages between a "client" and a "server" according to the aforementioned handshake protocol. The entity which initiates the connection is the client. The client establishes a channel with the server. Either the client or the server may be an enclave entity, and it may be the case that both are enclave entities. In this regard, the client may represent an enclave client of a first enclave or entity, such as enclave 160 or a non-enclave entity such as an application of client computing device 110, 120, or 130, and the server may represent an enclave server of a second enclave or entity, such as enclaves 170, 180 or a non-enclave entity such as an application of client computing device 110, 120, or 130.

[0084] The handshake protocol may be used during establishment of an RPC communication channel and may take place before the client and server exchange any application-level data. During the handshake protocol, the entities exchange and verify assertions, as discussed above with regard to FIGURE 8A, negotiate a record protocol, and negotiate a secret key that is used to cryptographically protect future traffic sent over the channel. After the handshake protocol completes, the client and server communicate using the record protocol and all data exchanged is encrypted and/or authenticated with the secret key.

[0085] Referring to FIGURE 9, Message1, from the client to the server, and Message2, from the server to the client, may enable the participants (i.e. the server and the client) to negotiate the types of assertions, a handshake cipher suite, a record protocol and handshake protocol version. By allowing the participants to negotiate which types of assertions will be used, the handshake protocol is able to support authentication that meets the security expectations of both participants. This negotiation (Message1 and Message2) can take place at the beginning of the handshake before any identities are asserted. Thus, the participant that initiates the exchange adopts the "client" role by sending a message (Message1) containing its assertion requests and assertion offers to the second participant. The second participant adopts the role of the "server". The server receives the client's offers and requests, selects between them according to which identities the server is capable of asserting and verifying, and then sends back the server's own offers and requests (Message2).

[0086] The programming interface that supports the RPC security system may be implemented as a library discussed above. The library may define the exact format and interpretation of the bytes of an assertion. Because of the use of a library, the format of the assertion, the assertion offer, and the assertion request need not be strictly specified by the handshake protocol, and the handshake protocol may operate on opaque blobs of bytes that represent assertions, assertion requests, and assertion offers. The handshake protocol leaves the majority of the work to the libraries, but calls those interfaces in at specific times during the handshake protocol. A simplified example of a possible protocol is described below.

[0087] The handshake protocol allows for various identities to be asserted during the exchange. The desired assertions could be configured on both the client and the server. Each participant would use the appropriate library to either generate (using an Assertion Generator) and verify (using an Assertion Verifier) an assertion on demand. Again, as described above these libraries could be registered with a handshaker library through static registration, and all

registered libraries could be added to the aforementioned registration map and can be queried to check for the existence of a particular library. The handshaker library may be used to conduct the handshake protocol. In this regard, each of the client and server will have its own handshaker library. Using this registration map, the enclave client and enclave server (or an RPC server of a non-enclave entity) can generate and verify assertions depending on their configuration and on which libraries are available.

[0088] Returning to FIGURE 9, the record protocol may be a message-passing protocol that is used to send application-level data. For the enclave system 100, this may be used as the base protocol over which RPCs and all RPC metadata are sent. As an example, the RPC security system could employ gRPC, which uses HTTP for message framing, and the record protocol could use AES-GCM with 128-bit keys and an even-odd scheme for dividing the counter space. The HTTP frames would then be secured using this protocol once the handshake completed.

[0089] Message3 and Message4 allow for authentication between the client and the server. For instance, the server may generate, using the server's Assertion Generator, an assertion of the server's identity. The server may then send this assertion to the client (Message3). Thereafter, the client would verify the assertion using the client's Assertion Verifier. Similarly, the client may generate, using the client's Assertion Generator, an assertion of the client's identity. The client may then send this assertion to the server (Message4). Thereafter, the server would verify the assertion using the server's Assertion Verifier. In this regard, Message3 and Message4 may correspond to step 2 of FIGURE 2A. If the assertions are not verified or the Assertion Verifiers of the client and/or server return false, as in step 3A of FIGURE 8A, the client and/or server may abort the handshake.

[0090] The handshake protocol supports authentication for various types of enclave identities and allows participants to exchange multiple identities during the authentication process. As with the identity framework, the RPC security system can be adapted to new identities by adding new Assertion Generators and Assertion Verifiers for those identities. Thus, these identities can be arbitrarily extended as discussed in the ABC, 123, XYZ and 456 examples discussed above. In other words, the handshake protocol is independent of and need not be changed if new identities are added to the identity framework. This is because the handshake protocol does not need to understand the details of a specific identity or even how to operate on it.

[0091] Message5 and Message6 may enable channel confidentiality by allowing the handshake participants to agree on a handshake cipher suite and to negotiate a session encryption key. Channel confidentiality allows data transmitted on the channel to remain private to the channel participants. The session encryption key is only known to the participants involved in the handshake and is used to encrypt and/or authenticate further traffic on the channel. As such, no third party can eavesdrop on and/or tamper with the communication between these entities.

[0092] As an example, negotiation of a session encryption key can be achieved through a Diffie-Hellman key exchange. In this exchange, the participants, the client and the server may generate ephemeral Diffie-Hellman key pairs. The key pair includes a private key, which is kept secret by the participant, and a public key, which can be shared with the other participants. The participants exchange their Diffie-Hellman public keys and then, using elliptic-curve cryptography, they can compute a secret value. This secret value can then be used as an input to a key-derivation function (KDF), such as HKDF, to derive a session encryption key. The KDF can also take additional context information from the handshake, such a hash of the handshake transcript, as an input into the key derivation process. This would provide the property that the derived key is cryptographically tied to the identities exchanged and verified during the handshake, as well as any random values that were exchanged earlier in the handshake to establish session uniqueness.

[0093] After the handshake protocol has been completed, both the enclave client and enclave server (or an RPC server of a non-enclave entity) may access authentication properties of the connection in order to access the identity information of the other to proceed to authorize as in step (3b) of FIGURE 8A and steps (4)-(6b) in FIGURE 8B. This enables the client and server to establish and enforce access-control policies. In the context of an RPC security system, this means enforcing access control policies on particular RPCs defined in the RPC security system. For instance, authorization policies on the server side may be used to enforce that the client meets certain identity requirements. The RPC security system also supports client-side authorization decisions. This is fundamental for the enclave security model because clients themselves can possess secrets and must ensure that they only send these secrets to authorized entities.

[0094] After an RPC communication channel is established using the handshake protocol, the two entities may proceed to send their RPC traffic over the established secure channel. The

RPC communication channel is established if it meets the authentication requirements of both participants, but there can be additional authorization constraints layered on top of this. For instance, in the case of an enclave server communicating with remote enclave clients, the enclave server may accept RPCs generated by or from enclave clients with any 123 code identity but may have specific code-identity requirements for a set of highly-sensitive RPCs.

[0095] The authorization layer, which enables the callee to check whether the caller meets the identity expectations for a particular call, may be implemented in various ways. In one approach, the authorization mechanism may be built into the RPC system (e.g. gRPC). The enclave server and enclave client may be configured with their respective authorization policies when they are initially programmed, and the RPC security system then automatically enforces these policies. When a call is made, the authorization layer may access the underlying authentication properties of the RPC communication channel to determine whether the authorization policies are satisfied. In another approach, the authorization logic may be written into the enclave application logic, such as directly into the enclave client or enclave server. For instance, a programmer may access the underlying authentication properties of the channel and define the programmer's own authorization logic in order to allow an enclave to check whether the caller or callee meets the identity expectations for a particular call. This is especially useful where the RPC framework does not have built-in authorization and/or where a programmer needs to utilize more complex logic than what's expressible via the RPC authorization policy.

[0096] Both these approaches may require access to the underlying channel authentication properties that were extracted from the results of the handshake protocol discussed above. This can be encapsulated in an authentication context object on either of the channel. The enclave client's authentication context contains identity information about the enclave server and the enclave server's authentication context contains identity information about the enclave client. Because the authentication process is based on the aforementioned identity framework, it is possible to extract various types of enclave identities during the handshake protocol and store them in a common data representation, such as a Protocol Buffer.

[0097] The authorization policies themselves can be defined in terms of RPC names and enclave identity ACL (i.e., an arbitrary logical combination of enclave identity expectations). For instance, one example authorization policy would associate an RPC having a particular name with an enclave identity expectation. A more complex authorization policy would

associate an RPC with a logical predicate that strings together various enclave identity expectations into a more complex expectation. In this regard, the authorization process may proceed utilizing an Identity ACL Evaluators as shown in FIGURE 8B.

[0098] Another possible application of the identity framework may include a secret sealing framework. The secret sealing framework may enable cryptographic-sealing of sensitive secrets within a multi-enclave system and disclosure of such secrets only to authorized entities in the system independent of the specific host machines or computing devices on which the entities reside.

[0099] The secret sealing framework may involve a SecretSealer interface. For instance, as shown in FIGURE 2, each of the enclaves 160, 170, 180, 190 includes one or more SecretSealer(s) 280, 282, 284, 286. Each SecretSealer may be considered a "sealer library" that includes instructions for sealing secrets. For instance, a SecretSealer may accept a header, a secret, and additional authenticated data and output a sealed secret. This additional authenticated data may be encrypted during transit via the RPC communication channels described above, but may be otherwise unencrypted. A SecretSealer may also include instructions for unsealing secrets by accepting a sealed secret and output a secret.

[0100] The header may include metadata about the secret, as well as an identity ACL of enclave identities that are allowed to unseal the secret. This header may also be included in the sealed secret outputted by a SecretSealer and may be verified by a SecretSealer in order to output the secret. FIGURE 10 is representative of a configuration of a header, and FIGURE 11 is representative of a configuration of a sealed secret.

[0101] The entity that actually implements the sealing and unsealing may be a sealing root and can either be a local or a remote entity. A local sealing root may be a root that executes in the same enclave as the SecretSealer. Local sealing roots may directly leverage hardware support for key-provisioning to seal secrets to the identity of the enclave of the SecretSealer. For example, in some sealing technologies local sealing roots may bind secrets to either the code identity—a measurement over the code that implements the enclave—or the signer identity—a measurement over the entity that signed the enclave. However, such local sealing roots may be limited in that they may not be able to bind secrets to enclaves with a completely different identity than the enclave of the SecretSealer or enclaves running on a different host machine or computing device.

[0102] A remote sealing root may be a root that executes in a different enclave than the SecretSealer. Remote sealing roots may also run in a stand-alone enclave and receive requests for sealing and unsealing secrets from enclave clients. Remote sealing roots may also require communication via RPCs. In this regard, the remote sealing root may function via the RPC communications channels and handshake protocol discussed above. As such, a remote sealing root can make use of the Assertion Generator and Assertion Verifier interfaces to perform authentication with enclave clients that make requests to seal or unseal secrets. This also allows the remote root to cryptographically verify the identity of the enclave client, allowing the remote root to unseal secrets only when the requestor's identity matches the identity ACL present in the header of a sealed secret. In addition, an enclave client may leverage assertions by the remote sealing root and refuse to communicate with the remote root if its identity does not match the client's identity expectation(s).

[0103] Remote sealing roots do not need to leverage hardware support for provisioning sealing keys for sealed secrets. Doing so may lead to problematic situations. For instance, if the remote sealing root is damaged or lost, all secrets sealed against it will be lost as well. To address this, the remote sealing root may include of a plurality of identical enclave instances all running identical code and each in possession of a shared master secret. In this regard, if one of the enclaves of the remote sealing root is damaged or lost, others of the plurality may "fill in" and provide keys to unseal sealed secrets that were previously sealed by the damaged enclave.

[0104] This master secret is used to derive keys that seal individual secrets. The master secret may only be available in an unencrypted form within the remote sealing root enclave instances. For the sake of availability and reliability, an implementation may persist the master secret outside the remote-root enclaves in an encrypted form, using a local-root-based SecretSealer (protected by hardware-provisioned keys).

[0105] The identity ACL for a sealed secret may include of a set of enclave identity expectations as with the identity ACLs discussed above. Because the identity expectations that make up an ACL are grouped using an arbitrary nesting of logical operators, this may provide a SecretSealer with some flexibility when sealing a secret. For example, the secret can be sealed to a whitelist, allowing any enclave whose identity fulfills at least one identity expectation in the identity ACL to unseal the secret. In addition or alternatively, the secret

can be sealed such that it can only be unsealed by an enclave whose identity fulfills all identity expectations in the identity ACL for the sealed secret.

[0106] In addition to a header and encrypted data, a sealed secret also contains "additional authenticated data" and "bookkeeping info". These pieces of data are presented in a plain, unencrypted form, the authenticity of which can be established by unsealing the secret. "Additional authenticated data" may be provided by a SecretSealer and may allow unencrypted data to be cryptographically bound to the sealed secret. "Bookkeeping info" may be provided by the remote sealing root and may be treated as opaque data by a SecretSealer. This data can be used by the sealing root to store any internal, implementation-specific data related to the secret.

[0107] FIGURE 12 provide a visual representation of aspects of the secret sealing framework described above. Again, Enclave A and Enclave B may represent two different enclaves running on different computing devices (such as enclaves 160 and 180) backed by the same or different enclave technologies and operating within the enclave system. In this regard, the actual processing of such information may be performed by the processors 112 of the computing devices on which the enclaves reside or which otherwise operate on the enclaves.

[0108] The Remote Sealing Root of FIGURE 12 executes in a different enclave than the SecretSealers of Enclaves A and B. Thus, if Enclave A is enclave 160, the remote sealing root may be in enclave 170, enclave 180, or enclave 190. As such, for the purposes of demonstration, Machine 1 may refer to computing device 110, Enclave A may refer to enclave 160, Machine 2 may refer to computing device 120, Enclave B may refer to the enclave 180, and the Remote Sealing Root may be in enclave 190, for instance, operated by computing device 130 or some other computing device. Of course, other configurations are also possible. For instance, Machine 1 and Machine 2 could correspond to the same computing device, such as computing device 120, where Enclave A is enclave 160 and Enclave B is enclave 170.

[0109] Referring to FIGURE 10, at step (1), an application (via the one or more processors 112) at the computing device of Enclave A may input a request to seal a secret into Enclave A's SecretSealer (Sealing Library). This request includes a header, a secret, and additional authenticated data. As noted above, the header includes an identity ACL for the secret to be sealed. The SecretSealer generates a key (gen_key()) and sends a request (seal(key, acl)) at step (2) including the key generated by the SecretSealer and the identity ACL for the secret to

be sealed to the Remote Sealing Root. The location and/or identity of the remote sealer root may be selected by a programmer and defined in the SecretSealer. Of course, the request may be sent after Enclave A and the enclave of the Remote Sealing Root establish an RPC communication channel using the handshake protocol described above, and the Remote Sealing Root and Enclave A are able to authenticate one another using the identity ACL for the secret to be sealed as discussed above.

[0110] Once the Remote Sealing Root receives the request, the Remote Sealing Root may use a master secret key to seal the key (sealed_key). At step (3), the sealed key is sent back to the SecretSealer of Enclave A. The SecretSealer then uses the original key to encrypt (enc(secret, key)) the secret and appends the sealed key as well as the identity ACL to the sealed secret generated at step (4). Thereafter, the sealed secret is able to leave Enclave A. For instance, the sealed secret may be sent from Machine 1 to Machine 2.

[0111] In order to unseal the secret, the sealed secret may be passed to Enclave B's SecretSealer with a request to unseal the sealed secret. In response, Enclave B's SecretSealer may then read the sealed key from the sealed secret (read_sealed_key(secret)) and send a request (unseal(sealed_key)) including the sealed key at step (7). This request may be sent after Enclave B and the enclave of the Remote Sealing Root establish an RPC communication channel using the handshake protocol described above, and the Remote Sealing Root and Enclave B are able to authenticate one another using the identity ACL appended to the sealed secret as discussed above.

[0112] Once the Remote Sealing Root receives the request, the Remote Sealing Root may use the master secret key to unseal the sealed key (unsealed_key). At step (7), the unsealed key (which is the original key generated by the SecretSealer of Enclave A) is sent back to the SecretSealer of Enclave B. The SecretSealer then uses the unsealed key to decrypt (dec(secret, key) the sealed secret at step (8). Thereafter, the unsealed secret is able to be processed at Enclave B.

[0113] Remote sealing roots can be implemented in other ways as well. For instance, the remote root may generate both the sealed and unsealed key, and/or the client may send un-encrypted data (i.e. plaintext) to the remote sealing root for encryption.

[0114] Most of the foregoing alternative examples are not mutually exclusive, but may be implemented in various combinations to achieve unique advantages. As these and other variations and combinations of the features discussed above can be utilized without departing

from the subject matter defined by the claims, the foregoing description of the embodiments should be taken by way of illustration rather than by way of limitation of the subject matter defined by the claims. As an example, the preceding operations do not have to be performed in the precise order described above. Rather, various steps can be handled in a different order, such as reversed, or simultaneously. Steps can also be omitted unless otherwise stated. In addition, the provision of the examples described herein, as well as clauses phrased as "such as," "including" and the like, should not be interpreted as limiting the subject matter of the claims to the specific examples; rather, the examples are intended to illustrate only one of many possible embodiments. Further, the same reference numbers in different drawings can identify the same or similar elements.

CLAIMS

1. A method of authentication for an enclave entity with a second entity, the method comprising:

receiving, by one or more processors of a host computing device of the enclave entity, a request and an assertion of identity for the second entity, the assertion including identity information for the second identity;

using, by the one or more processors, an assertion verifier of the enclave entity to determine whether the assertion is valid;

when the assertion is valid, extracting, by the one or more processors, the identity information;

authenticating the second entity using, by the one or more processors, an access control list for the enclave entity to determine whether the identity information meets expectations of the access control list; and

when the identity information meets the expectations of the access control list, completing, by the one or more processors, the request.

2. The method of claim 1, wherein the second entity is a non-enclave entity.

3. The method of claim 1, wherein the second entity is a second enclave entity.

4. The method of claim 1, wherein when the assertion is not valid, denying the request.

5. The method of claim 1, wherein when the identity information does not meet the expectations of the access control list, denying the request.

6. The method of claim 1, further comprising:

using an assertion generator of the enclave entity to generate a second assertion including identity information for the enclave entity; and

sending the second assertion to the second entity for verification.

7. The method of claim 1, wherein the assertion verifier includes instructions for

verifying assertions and extracting identities out of verified assertions.

8. The method of claim 1, wherein the assertion verifier provides a Boolean response that indicates whether the assertion is valid.

9. The method of claim 1, further comprising, using an identity access control list evaluator to determine whether the identity information meets expectations of the access control list, wherein the identity access control list evaluator includes instructions for operating on a set of identities possessed by an entity and evaluating that set of identities against an access-control policy.

10. The method of claim 9, wherein the identity access control list evaluator provides a Boolean response that indicates whether the identity information meets the expectations of the access control list match.

11. The method of claim 10, wherein the access control list includes expressions of enclave identity expectations, and the identity expectations are used to determine whether the identity information meets expectations of the access control list.

12. The method of claim 10, where the enclave identity expectations are configured as predicates that include two or more identity expectations via one or more logical operations.

13. The method of claim 10, wherein each identity expectation includes a reference identity and a match specification, and wherein at least one reference identity and at least one match specification are used to determine whether the identity information meets expectations of the access control list.

14. The method of claim 1, wherein the enclave entity includes an enclave server including instructions configured to receive and complete the request and wherein the enclave server is used to complete the request.

15. The method of claim 1, wherein the identity information includes an identity

description that classifies the identity of the assertion an identity type supported by the enclave entity and identifies an authority responsible for handling identities of that identity type.

16. The method of claim 15, wherein the authority is used to identity the assertion verifier in order to determine whether the assertion is valid.

17. The method of claim 15, wherein the authority is used to identify a library in order to determine whether the identity information meets the expectations of the access control list.

18. The method of claim 15, wherein the authority is used to identify a plugin including interfaces that in order to determine whether the assertion is valid and in order to determine whether the identity information meets the expectations of the access control list.

19. The method of claim 1, wherein the enclave entity and the second entity are located in local memory of the host computing device.

20. The method of claim 1, wherein the second entity is located in local memory of a second host computing device, the second host computing device being different from the host computing device.

21. A method of establishing a communication channel between an enclave entity and a second entity, the method comprising:

receiving, by one or more processors of a host computing device of the enclave entity, a request to initiate the communication channel from the second entity;

negotiating, by the one or more processors, with the second entity a record protocol for the communication channel and a secret key for cryptographically protecting traffic sent over the communication channel; and

after the negotiating is completed, communicating with the second entity using the communication channel using the record protocol and using the secret key to encrypt and

authenticate data exchanged over the communication channel.

22. The method of claim 21, further comprising conducting, by the one or more processors, an attestation and authentication process to verify an identity of the second entity, wherein the process includes:

receiving an assertion of the second entity's identity; and

verifying the second entity's identity.

23. The method of claim 22, wherein the attestation and authentication process further includes sending an assertion of the enclave entity's identity to the second entity.

24. The method of claim 22, wherein negotiating the record protocol occurs before conducting the attestation and authentication process.

25. The method of claim 22, further comprising negotiating one or more types of assertions of identity before conducting the attestation and authentication process such that the one or more types of assertions of identity are used in the assertion and authentication process.

26. The method of claim 25, wherein negotiating one or more types of assertions of identity includes identifying one or more types of assertions that the enclave entity is capable of making and verifying.

27. The method of claim 25 wherein negotiating one or more types of assertions of identity includes receiving one or more types of assertions that the second entity is capable of making and verifying.

28. The method of claim 21, wherein the request is received as a remote procedure call to the enclave entity.

29. The method of claim 21, wherein the record protocol is a message passing protocol that is used to send application-level data when communicating with the second

entity using the communication channel.

30. The method of claim 21, further comprising conducting, by the one or more processors, an authentication and authorization process to verify an identity of the second entity, wherein at least part of the authentication and authentication process is conducted after the negotiating is complete.

31. The method of claim 30, wherein a first part of the authentication and authorization process includes and verifying a received assertion of the second entity's identity.

32. The method of claim 31, wherein a second part of the authentication and authorization process includes authorizing the second entity using identity information extracted from the assertion.

33. The method of claim 32, wherein the first part is performed before the communicating using the communication channel and the second part is performed before the communicating using the communication channel.

34. The method of claim 31, further comprising aborting the communication channel when the enclave entity is unable to authenticate the second entity.

35. The method of claim 21, wherein the record protocol is negotiated before the secret key is negotiated.

36. The method of claim 21, wherein the second entity is a non-enclave entity.

37. The method of claim 21, wherein the second entity is a second enclave entity.

38. The method of claim 37, wherein the negotiating is conducted with an enclave server of the second enclave entity configured to an enclave server including instructions configured to receive and complete the request and wherein the enclave server is used to

complete the request.

39. The method of claim 21, wherein the enclave entity and the second entity are located in local memory of the host computing device.

40. The method of claim 21, wherein the second entity is located in local memory of a second host computing device, the second host computing device being different from the host computing device.

41. A method of sealing secrets in a first enclave entity of a host computing device, the method comprising:

inputting, by one or more processors of the host computing device, to a sealing library of the first enclave entity a header and a secret, the header including an identity access control list for the secret;

generating, by the one or more processors, using the sealing library, a key;

sending, by the one or more processors, the key and the identity access control list to a second enclave entity;

in response to the sending, receiving, by the one or more processors, from the second enclave entity a sealed version of the key;

using, by the one or more processors, the sealing library to seal the secret; and

appending the sealed version of the key and the identity access control list to the sealed secret.

42. The method of claim 41, wherein the sealing library includes instructions for unsealing data by accepting sealed data and output the unsealed data.

43. The method of claim 41, further comprising, establishing communications with the second enclave entity via a remote procedure call.

44. The method of claim 43, further comprising, using the sealing library to identify a location of the second enclave entity in order to establish the communications.

45. The method of claim 41, wherein the received sealing key is independent of a host computing device of the second enclave entity.

46. The method of claim 41, further comprising using the identity access control list to authenticate the second enclave entity before sending the key.

47. The method of claim 41, wherein the first enclave entity does not have access to an unencrypted version of a master secret key used to generate the sealed version of the key such that the first enclave entity is unable to unseal the sealed version of the key.

48. The method of claim 41, further comprising sending the sealed secret and appended sealed version of the key to a third enclave entity.

49. The method of claim 48, wherein the secret is only able to be sent to the third enclave entity after the secret has been sealed.

50. The method of claim 41, wherein the identity access control list identifies one or more entities that are able to unseal the sealed secret.

51. The method of claim 41, wherein sealing the secret includes sealing the secret to a whitelist that allows any enclave entity whose identity fulfills at least one identity expectation in the identity access control list to unseal the sealed secret.

52. The method of claim 41, wherein sealing the secret includes sealing the secret to a whitelist that allows any enclave entity whose identity fulfills all identity expectations in the identity access control list to unseal the sealed secret.

53. The method of claim 41, further comprising:

inputting to the sealing library of the first enclave entity a second sealed secret including an appended sealed version of a second key;

sending, by the one or more processors, a request to unseal the second sealed secret to a third enclave entity;

receiving, by the one or more processors, an unsealed version of the second key; and

using, by the one or more processors, the sealing library to unseal the second secret.

54. The method of claim 53, wherein the third enclave entity is one of a plurality of identical enclave instances all running identical code and each in possession of a shared master secret such that the first enclave entity is able to send the request to any of the plurality of identical enclave instances.

55. The method of claim 54, wherein if the third enclave entity is not available, sending the request to a fourth enclave entity, and wherein the unsealed version of the second key is received from the fourth enclave entity, and wherein the fourth enclave entity is one of the plurality of identical enclave instances.

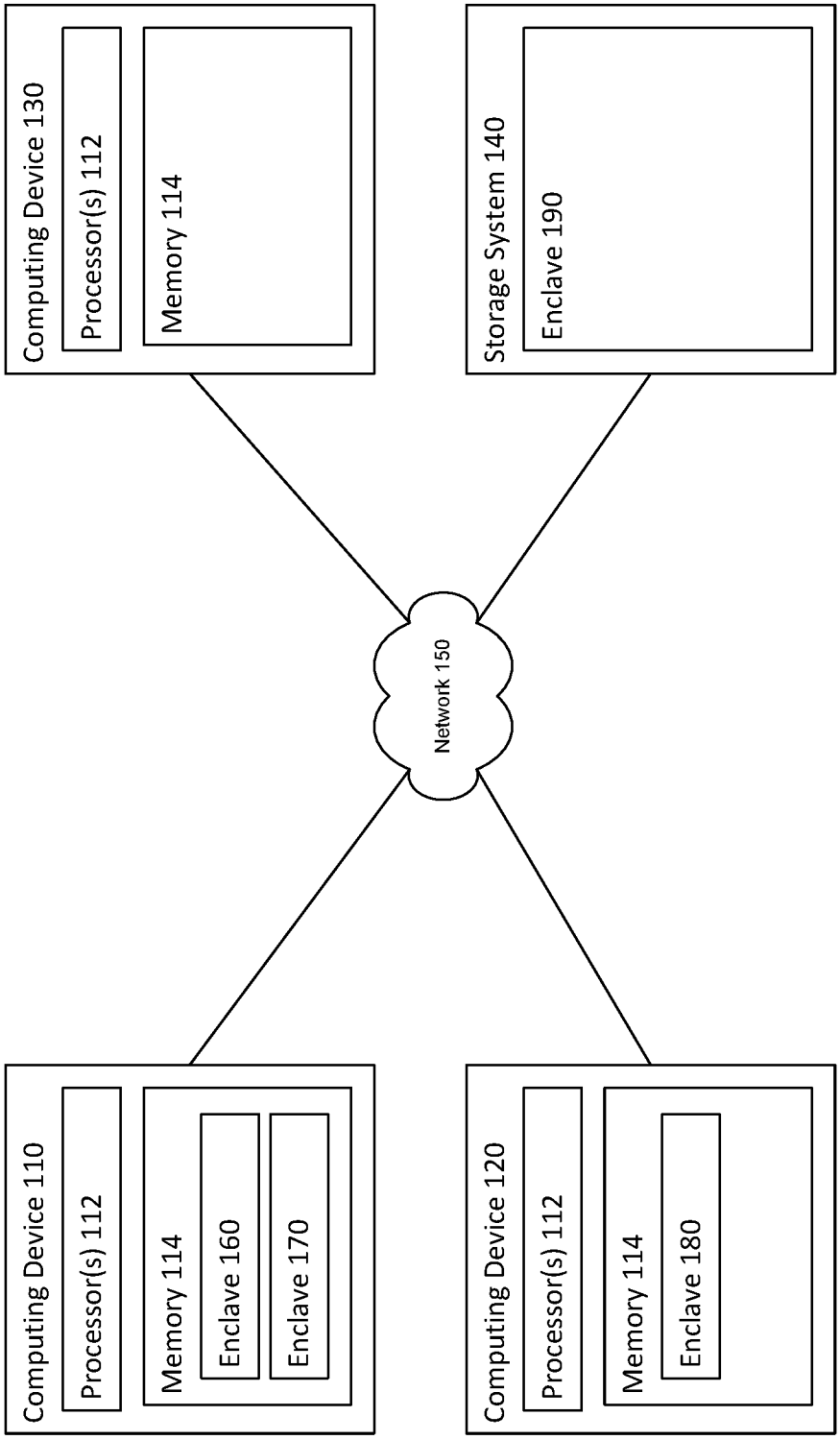
56. The method of claim 53, wherein the request is sent after a remote procedure call communication channel is established between the first enclave entity and the second enclave entity using an identity access control list for the second sealed secret.

57. The method of claim 53, further comprising, using the sealing library to identify a location of the third enclave entity in order to send the request to the third enclave entity.

58. The method of claim 41, wherein the second enclave entity is a remote sealing root, such that the second enclave entity is a different enclave from the first enclave entity.

59. The method of claim 41, wherein the first enclave entity and the second enclave entity are located in local memory of the host computing device.

60. The method of claim 41, wherein the second enclave entity is located in local memory of a second host computing device, the second host computing device being different from the host computing device.


Computing Device 130
Processor(s) 112
Memory 114
Storage System 140
Enclave 190
Network 150
Computing Device 110
Processor(s) 112
Memory 114
Enclave 160
Enclave 170
Computing Device 120
Processor(s) 112
Memory 114
Enclave 180

100

FIGURE 1

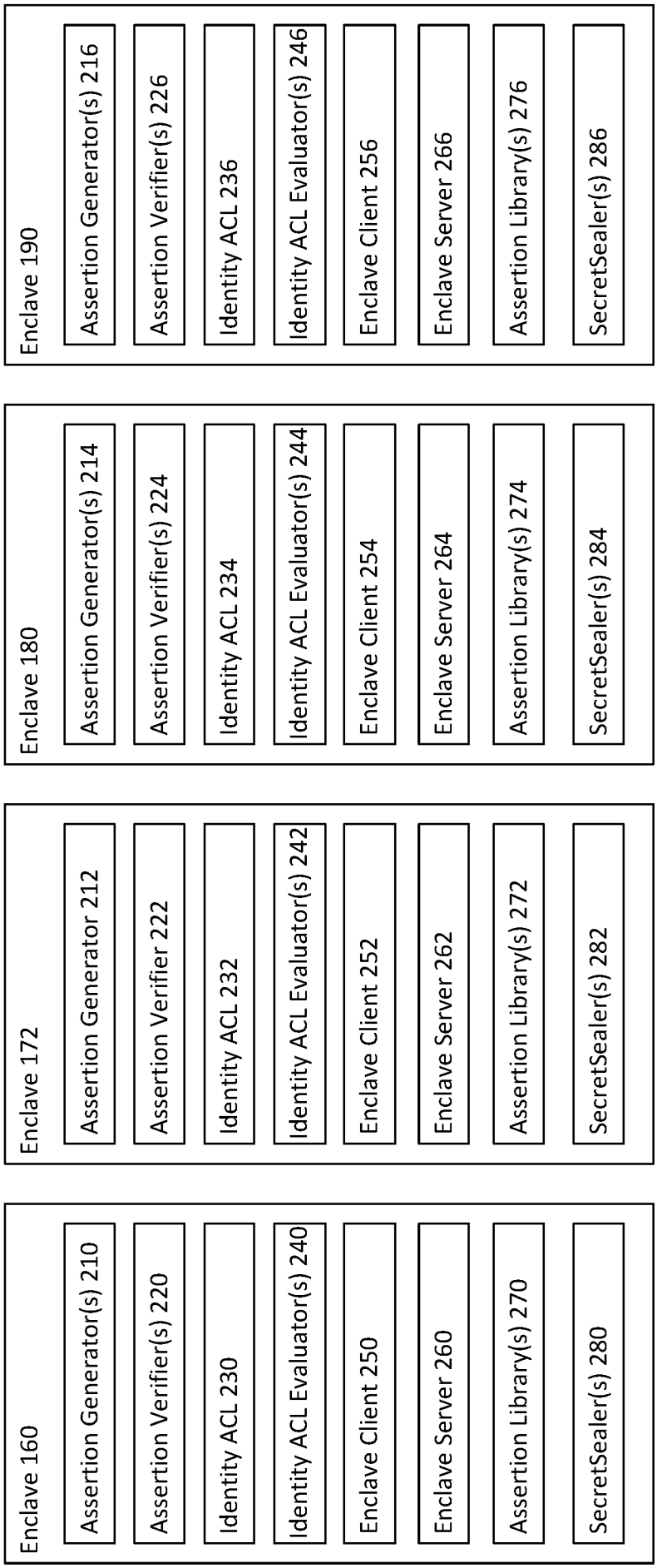

Enclave 160
Assertion Generator(s) 210
Assertion Verifier(s) 220
Identity ACL 230
Identity ACL Evaluator(s) 240
Enclave Client 250
Enclave Server 260
Assertion Library(s) 270
SecretSealer(s) 280
Enclave 172
Assertion Generator 212
Assertion Verifier 222
Identity ACL 232
Identity ACL Evaluator(s) 242
Enclave Client 252
Enclave Server 262
Assertion Library(s) 272
SecretSealer(s) 282
Enclave 180
Assertion Generator(s) 214
Assertion Verifier(s) 224
Identity ACL 234
Identity ACL Evaluator(s) 244
Enclave Client 254
Enclave Server 264
Assertion Library(s) 274
SecretSealer(s) 284
Enclave 190
Assertion Generator(s) 216
Assertion Verifier(s) 226
Identity ACL 236
Identity ACL Evaluator(s) 246
Enclave Client 256
Enclave Server 266
Assertion Library(s) 276
SecretSealer(s) 286

FIGURE 2

```
enum IdentityClass {
  UNKNOWN = 0,
  CODE_IDENTITY = 1,
  CERTIFICATE_IDENTITY = 2,
}
```

FIGURE 3

```
message IdentityDescription {
  optional IdentityClass identity_class = 1;
  optional string authority = 2;
}
```

FIGURE 4

```
message Identity {
  optional IdentityDescription description = 1;
  optional bytes identity = 2;
}
```

FIGURE 5

```
message AssertionDescription {
  optional IdentityClass identity_class = 1;
  optional string authority = 2;
}
```

FIGURE 6

```
message Assertion {
  optional AssertionDescription description = 1;
  optional string authority = 2;
}
```

FIGURE 7

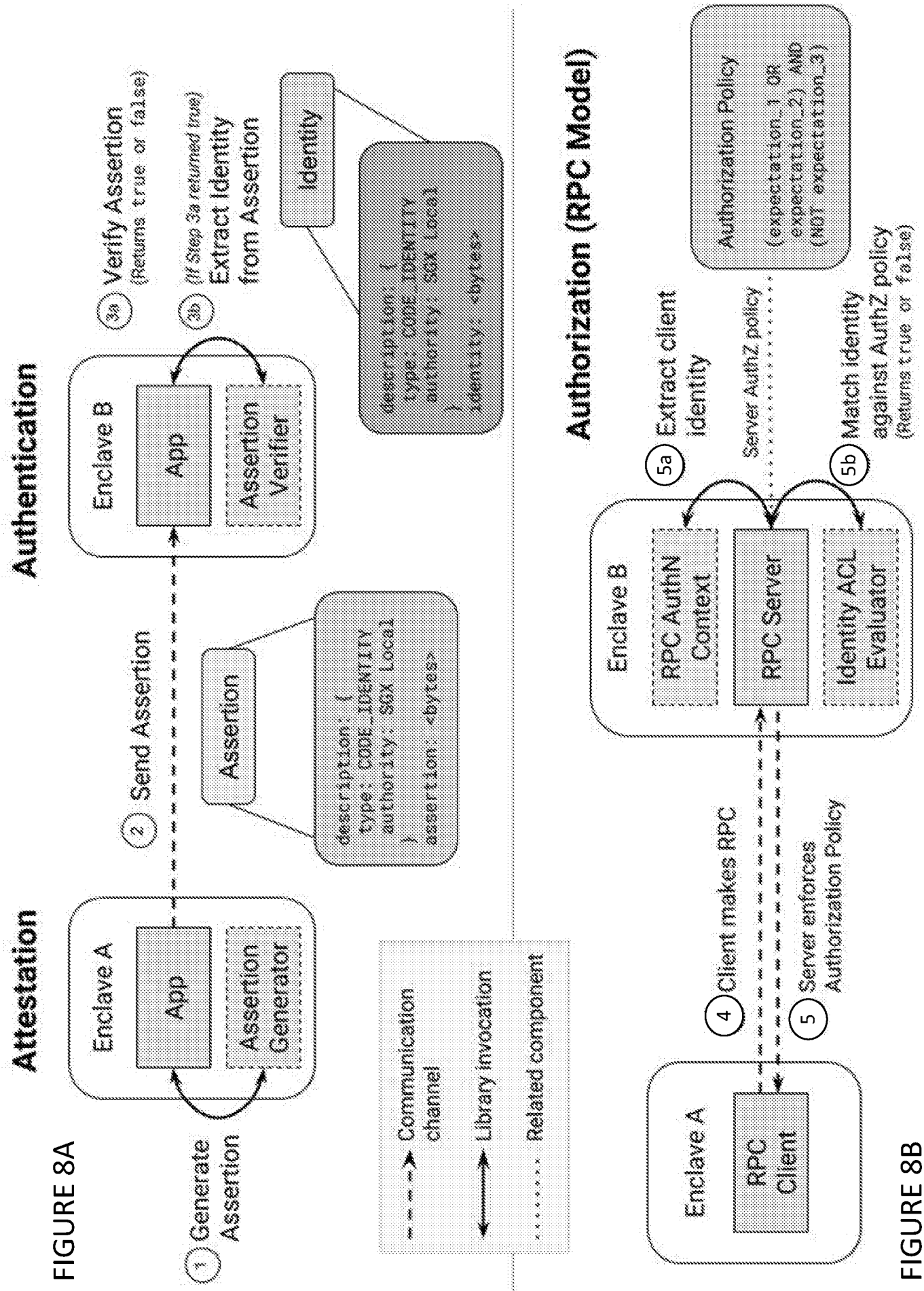


FIGURE 8A

FIGURE 8B

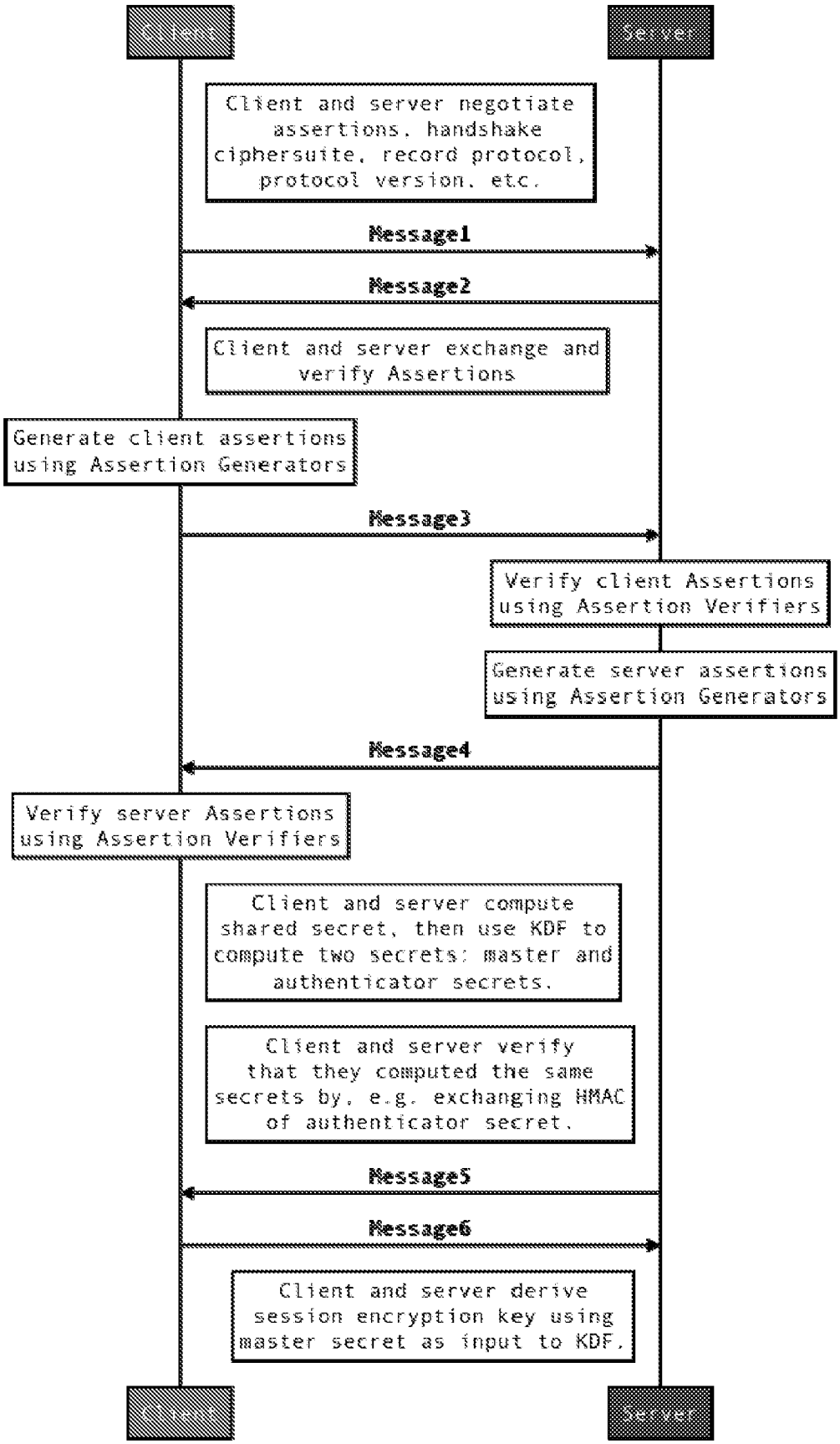

Client
Server
Client and server negotiate assertions, handshake ciphersuite, record protocol, protocol version, etc.
Message1
Message2
Client and server exchange and verify Assertions
Generate client assertions using Assertion Generators
Message3
Verify client Assertions using Assertion Verifiers
Generate server assertions using Assertion Generators
Message4
Verify server Assertions using Assertion Verifiers
Client and server compute shared secret, then use KDF to compute two secrets: master and authenticator secrets.
Client and server verify that they computed the same secrets by, e.g. exchanging HMAC of authenticator secret.
Message5
Message6
Client and server derive session encryption key using master secret as input to KDF.
Client
Server

FIGURE 9

```
message SealedSecretHeader {
  // Name of the secret. This is an arbitrary, user-defined string. The
  // SecretSealer does not associate any meaning with this value.
  //
  // Users of the SecretSealer interface are expected to populate this field.
  optional string secret_name = 1;

  // Version of the secret. This is an arbitrary user-defined string. The
  // SecretSealer does not associate any meaning with this value.
  //
  // Users of the SecretSealer interface are expected to populate this field.
  optional string secret_version = 2;

  // Purpose of the secret. This is an arbitrary, user-defined string. The
  // SecretSealer does not associate any meaning with this value.
  //
  // Users of the SecretSealer interface are expected to populate this field.
  optional string secret_purpose = 3;

  // Information about the sealing root.
  //
  // SecretSealer::SetDefaultHeader() must populate this field.
  optional SealingRootInformation root_info = 4;

  // An optional list of identities belonging to the author of the sealed
  // secret.
  //
  // The SecretSealer::Seal() and SecretSealer::Reseal() methods must populate
  // this field.
  repeated EnclaveIdentity author = 5;

  // ACL consisting of the enclave-identity expectations that are allowed to
  // access this secret.
  //
  // SecretSealer::SetDefaultHeader() must populate this field.
  optional IdentityAclPredicate client_acl = 6;

  // Policy that the client is expected to enforce on the unwrapped secret.
  // |secret_handling_policy| is an opaque field, and its interpretation is
  // specific to the client and/or secret.
  //
  // User of the SecretSealer interface is expected to populate this field.
  optional bytes secret_handling_policy = 7;
}
```

FIGURE 10

```
message SealedSecret {
  // Initialization vector used by the AEAD scheme used for encrypting the
  // secret. The size of the IV depends on the ciphersuite used for the
  // encryption (which may be included in the
  // SealingRootInformation::additional_info field).
  optional bytes iv = 1;

  // Serialized SealedSecretHeader. The header is included in its serialized
  // form to enable deterministic MAC computation.
  optional bytes sealed_secret_header = 2;

  // Data whose integrity and authenticity are verifiable.
  optional bytes additional_authenticated_data = 3;

  // Ciphertext as computed by an appropriate AEAD scheme.
  optional bytes secret_ciphertext = 4;

  // Bookkeeping information for the sealing root. This information is
  // strictly optional, and has no meaning for the client.
  optional bytes sealing_root_bookkeeping_info = 5;
}
```

FIGURE 11

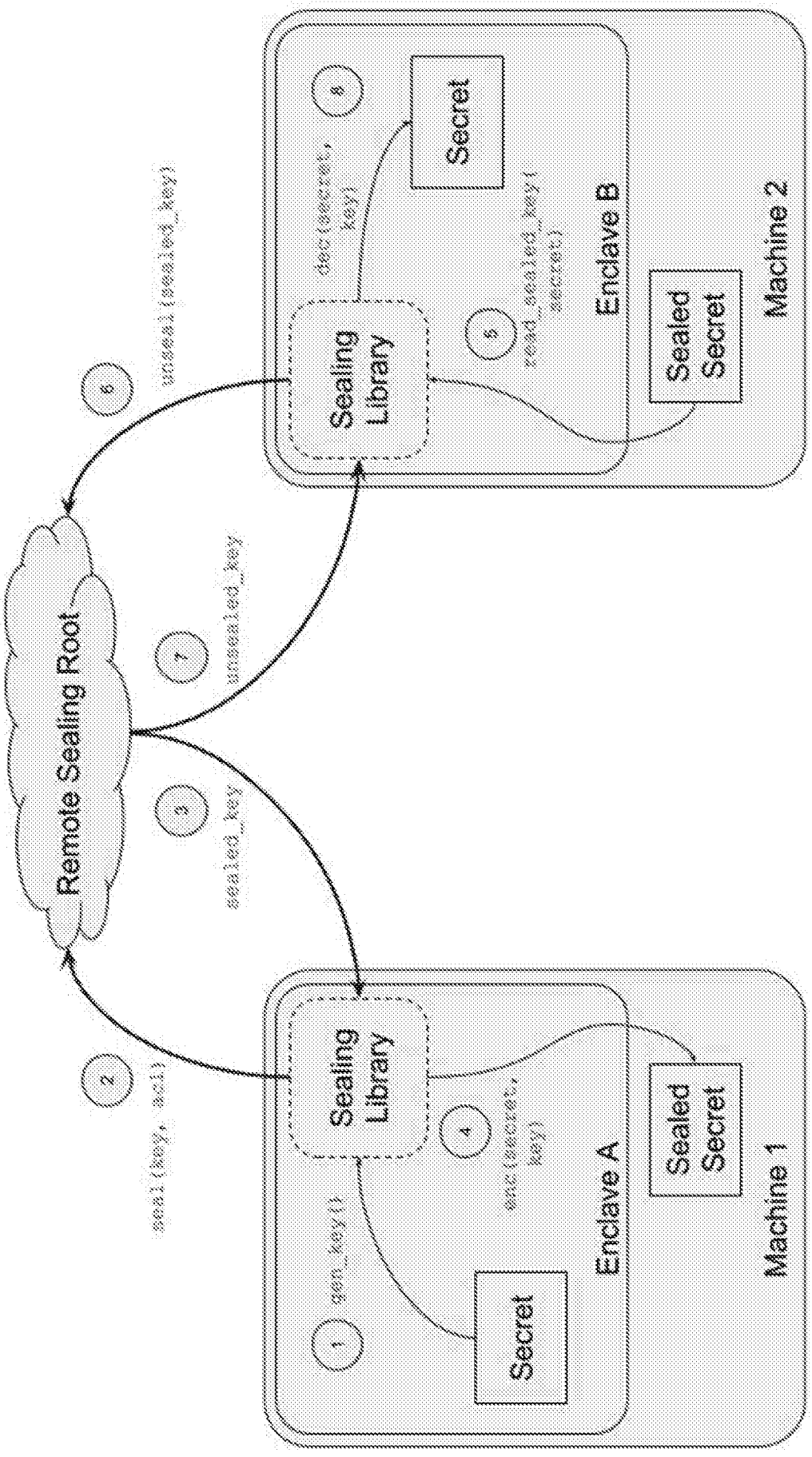

Machine 1
Enclave A
Sealing Library
Secret
Sealed Secret
Machine 2
Enclave B
Sealing Library
Secret
Sealed Secret
Remote Sealing Root
1
gen_key()
2
seal(key, aad)
3
sealed_key
4
enc(secret, key)
5
read_sealed_key()
6
unseal(sealed_key)
7
unsealed_key
8
dec(secret, key)

FIGURE 12

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2018/042684

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F21/53 H04L12/24 G06F21/74 H04L29/06 G06Q20/32
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06Q G06F H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	WO 2015/066028 A1 (APPLE INC [US]) 7 May 2015 (2015-05-07) paragraph [0024] - paragraph [0043]; figure 5	1-20

X	WO 2015/094261 A1 (INTEL CORP [US]) 25 June 2015 (2015-06-25) figures 3-7 paragraph [0011] - paragraph [0039] paragraph [0055] - paragraph [0088]	1-20

X	US 2017/201380 A1 (SCHAAP TRISTAN F [US] ET AL) 13 July 2017 (2017-07-13) paragraph [0025] - paragraph [0061]; figures 1,4A	1-20

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
1 October 2018	05/12/2018

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Veshi, Erzim

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2018/042684

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

see additional sheet

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying an additional fees, this Authority did not invite payment of additional fees.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☒ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

1-20

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.

☐ The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.

☐ No protest accompanied the payment of additional search fees.

FURTHER INFORMATION CONTINUED FROM PCT/ISA/ 210

This International Searching Authority found multiple (groups of) inventions in this international application, as follows:

1. claims: 1-20

   A method for authenticating another entity from an enclave entity. The enclave entity receives an assertion of identity, extracts identity and authenticates the other entity.

   ---

2. claims: 21-40

   Method for key and protocol negotiation between and enclave entity and another entity. The enclave entity receives a request to initiate communication, negotiates the protocol and the cryptographic key, and afterwards communicates with the second entity in an encrypted way.

   ---

3. claims: 41-60

   Method for sealing secrets between an enclave entity and a second entity. The second entity is also an enclave entity. In the first enclave, a header and a secret is input in a sealing library, a key is generated, and the key and an ACL is communicated to the second entity. The sealing library is used to seal the secret and a sealed version of the key is appended to the ACL.

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2018/042684

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 2015066028 A1	07-05-2015	AU 2014342529 A1	12-05-2016
		AU 2018202035 A1	19-04-2018
		CN 105684009 A	15-06-2016
		EP 3066627 A1	14-09-2016
		JP 6293886 B2	14-03-2018
		JP 2016537879 A	01-12-2016
		JP 2018092651 A	14-06-2018
		KR 20160082538 A	08-07-2016
		KR 20180019777 A	26-02-2018
		US 2015127549 A1	07-05-2015
		WO 2015066028 A1	07-05-2015
WO 2015094261 A1	25-06-2015	CN 105745661 A	06-07-2016
		EP 3084667 A1	26-10-2016
		KR 20160101108 A	24-08-2016
		US 2015347768 A1	03-12-2015
		WO 2015094261 A1	25-06-2015
US 2017201380 A1	13-07-2017	AU 2016385445 A1	16-08-2018
		CN 108476404 A	31-08-2018
		EP 3400730 A1	14-11-2018
		KR 20180096699 A	29-08-2018
		US 2017201380 A1	13-07-2017
		WO 2017120011 A1	13-07-2017