Session Based Remote Attestation

Remote attestation is a concept where a (possibly) trusted platform signs data related to the platform and sends this data to a remote service, which then can use this information to assure itself that it can safely deploy various objects such as executable code and cryptographic keys to the attesting platform. There are currently two versions of this in the wild: 1) attestations associated with generated keys 2) attestations that apply to any operation. This paper describes a third approach using a "stateful" attestation system, which provides certain features not easily obtainable with the other schemes.

Introduction and Scope

It is obvious that a system that attests generated keys like in Android (as of version 8) is only usable for key generation and not for arbitrary objects including executable code. Due to that, this paper focuses on the second approach (attestation for *any* kind of operation). The target platform is supposed to be a Trusted Execution Environment (TEE). The exact nature of TEEs is out of scope for this paper but they are supposed to come with preinstalled attestation keys, certificates and trust anchors. Although platform attestations can be useful for any kind of trusted application *this paper only covers the provisioning of trusted data in a TEE from a Remote Provisioning Server (RPS)*.

Note: this is not really a scientific paper, but an *overview* of two attestation schemes. For details, please turn to the documents listed in the References section.

Static (Stateless) Remote Attestation

Static remote attestations as featured in IETF's TEEP require that each write or read access associated with TEE provisioning also is accompanied by fresh attestation data.

Session Based Remote Attestation

Session based remote attestation as briefly outlined in this paper is about *combining multiple operations into single unit*:

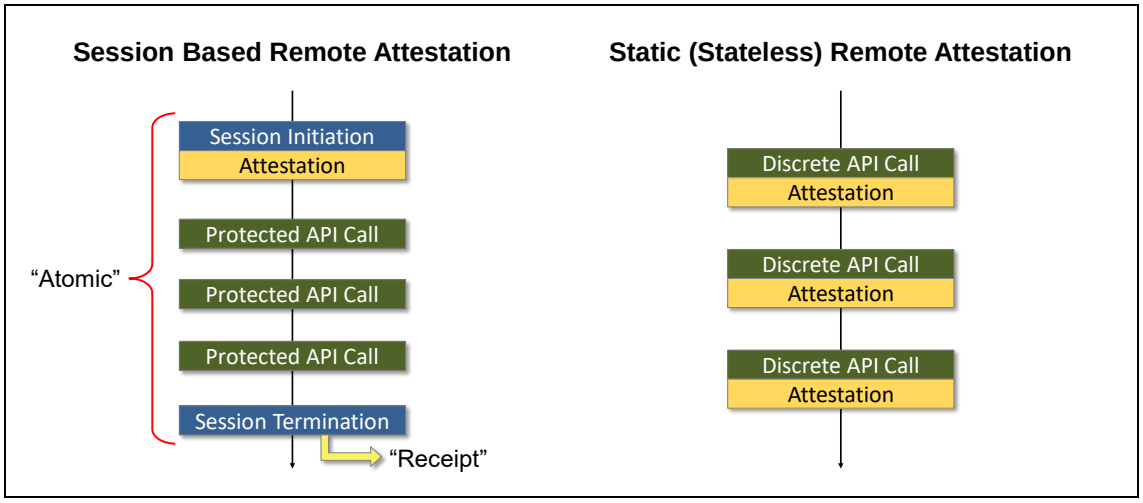
- Attestation of platform related data to the RPS
- Creation of a secret key shared between the TEE and the RPS
- Creation of a session inside of the TEE and reflected in the RPS

By combining attestation with shared key creation, subsequent operations performed in the associated session effectively "inherit" the qualities of the initial attestation. See Protected API.

A practical implementation of this scheme is the Secure Key Store (SKS), designed to support mobile phone based authentication and payment authorization systems.

The Attestation Concepts – Visualized

The state diagrams below illustrate the instruction flow from the RPS to the TEE:



The "Atomic" label refers to the ability performing a set of calls that either are committed or rolled back in its entirety. This feature is intended to support secure provisioning and replacements of virtual payment cards and similar where you do not want to the leave the user with a half-provisioned or broken wallet.

If all provisioning steps succeed, a "Receipt" signed by the session key is provided giving the RPS a clear signal that it succeed. If there is any kind of problem including software bugs, network errors, or if the user aborted the operation an error message will rather be generated. If the RPS lose contact with the TEE the session self-destructs after an RPS-defined timeout.

In the Static Remote Attestation model, each call is self-sufficient which is optimal for simple cases where there is limited need to build on results from previous operations.

Protected API

The protected API concept is a core feature of the Session Based Remote Attestation scheme. It is loosely based on TPM's authorized sessions but here combined with an initial remote attestation. Below is the generic form of the protected API:

Result Name (Session ID, {*Unencrypted Parameters*}, {*Encrypted Parameters*}, MAC)

Element	Description
Result	Result of the operation. If the result contains data for consumption by the RPS, a symmetric key derived by the shared session key vouches for this data through an HMAC operation which output also becomes a part of the result data
Name	Name of the API method
Session ID	Non-secret data holding a mutual TEE/RPS session identifier
Unencrypted Parameters	0..n parameters holding clear text data
Encrypted Parameters	0..n parameters holding data encrypted by a symmetric key derived from the shared session key
MAC	MAC holds the binary result of an HMAC operation (using a key derived from the shared session key) over the method *Name*, *Session ID*, the optional *Parameters* and an internal *Sequence counter*. The latter is incremented for each API call to enable verification of API call sequence order. Since the MAC is created by the RPS, the RPS needs to internally mimic the *Sequence counter* as well

Any errors found in the input data or MAC force session termination and rollback of provisioned data.

Other Features

There is a huge list of additional features offered by the Session Based Remote Attestation concept, many of them only obvious when you start using it. Below is a list of things exploited in the SKS/KeyGen2 proof-of-concept system:

- High-level human-readable MAC-protected protocol talking binary to primitive (resource constrained) TEEs
- Offloading JSON and BASE64 encoding/decoding, Networking and UI to the REE level
- Makes the TEE part independent of protocol language (XML/JSON/CBOR)
- API and response aggregation/pipe-lining
- User interaction inside of the secure protocol (like user PIN code setting according to RPS-defined policy)
- Provisioning of complex structures in a step-wise fashion
- Virtual namespace eliminates need for explicit key IDs
- Credential management with issuer isolation

Security Element Support

The Session Based Remote Attestation mechanism is potentially useful with a stateless Security Element (crypto processor) communicating with the TEE.

Acknowledgements

The Session Based Remote Attestation mechanism is very much influenced by experiences gathered at the Office of the CTO of RSA Security and with Nokia's ObC (On board Credential) system. Individuals who have directly or indirectly provided valuable input include Magnus Nyström (at that time at RSA), Jan-Erik Ekberg, N. Asokan, and Kari Kostiainen (all at that time at Nokia).

References

TEEP	Trusted Execution Environment Provisioning	https://tools.ietf.org/html/draft-ietf-teep-architecture-02 https://tools.ietf.org/html/draft-ietf-teep-opentrustprotocol-00
TPM	Trusted Platform Module	https://trustedcomputinggroup.org/resource/tpm-main-specification/
SKS	Secure Key Store	https://cyberphone.github.io/doc/security/sks-api-arch.pdf
KeyGen2	SKS provisioning protocol	https://cyberphone.github.io/doc/security/keygen2.html
Android	AndroidKeyStore attestation	https://developer.android.com/training/articles/security-key-attestation
ObC	Onboard Credentials	https://aaltodoc.aalto.fi/handle/123456789/3568
TEE/SE	TEE/Security Element "Combo" (outdated document)	https://github.com/cyberphone/openkeystore.old/blob/master/resources/docs/tee-se-combo.pdf

IPR Declaration

To simplify the adoption of Session Based Remote Attestation the core concept has been published as a "defensive publication" at: https://priorart.ip.com/IPCOM/000215433. Unfortunately, it is close to impossible to know if anybody else has IPR interests in this matter.

Author: Anders Rundgren, anders.rundgren.net@gmail.com
Version: 0.11