Google DeepMind

Attested Noise Protocol for Low-TCB Trusted Execution Environments

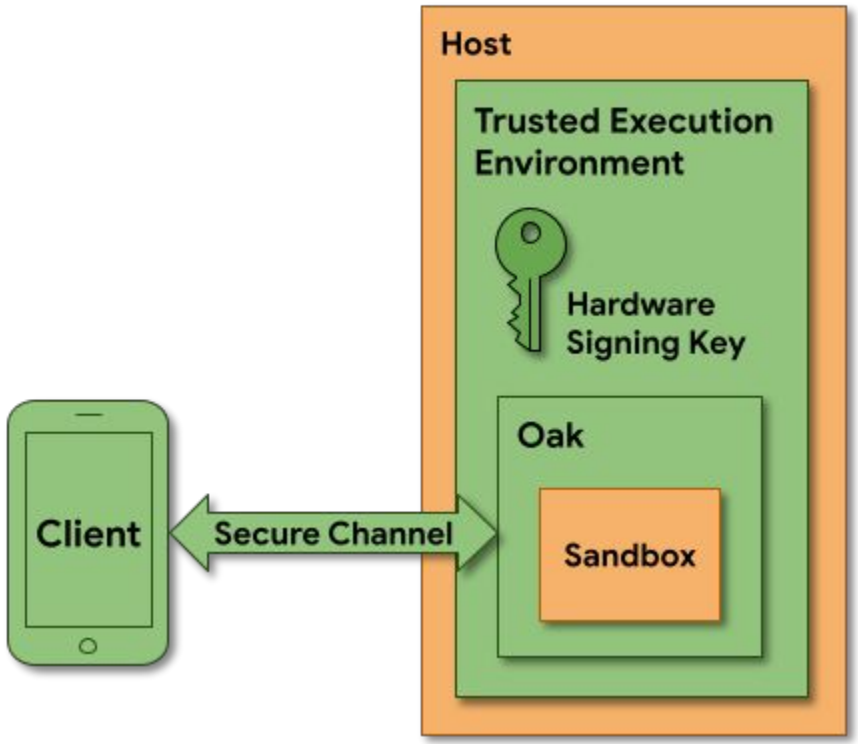
Ivan Petrov
ivanpetrov@google.com

Katsiaryna Naliuka
katsiaryna@google.com

Agenda

1. Project Oak
2. Noise Protocol
3. Remote Attestation

Project Oak



github.com/project-oak/oak

Research project aiming to make it possible for users to reason about **how their data will be used** by the server in ways verifiable by external reviewers

Oak Building Blocks

Trusted Execution Environments

- Minimize the Trusted Computing Base (TCB)
- Use restricted environments and sandboxing

Remote Attestation

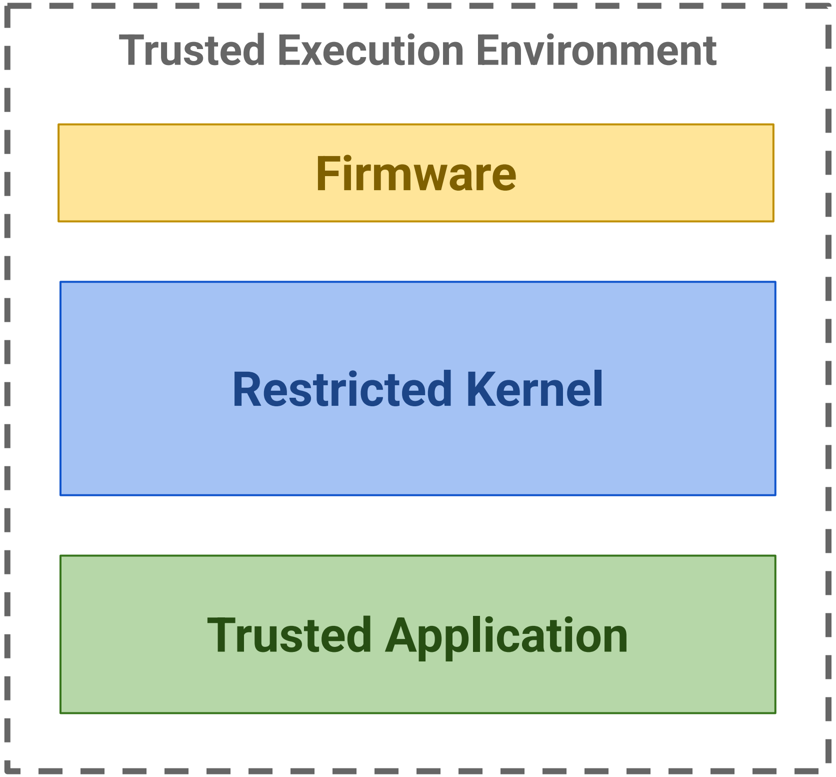
- Provide complete view of the workload

Transparency

- Open-source code
- Reproducible builds
- Verifiable Logs

Google

Restricted Environment



Firmware

- vBIOS + Bootloader

Restricted Kernel

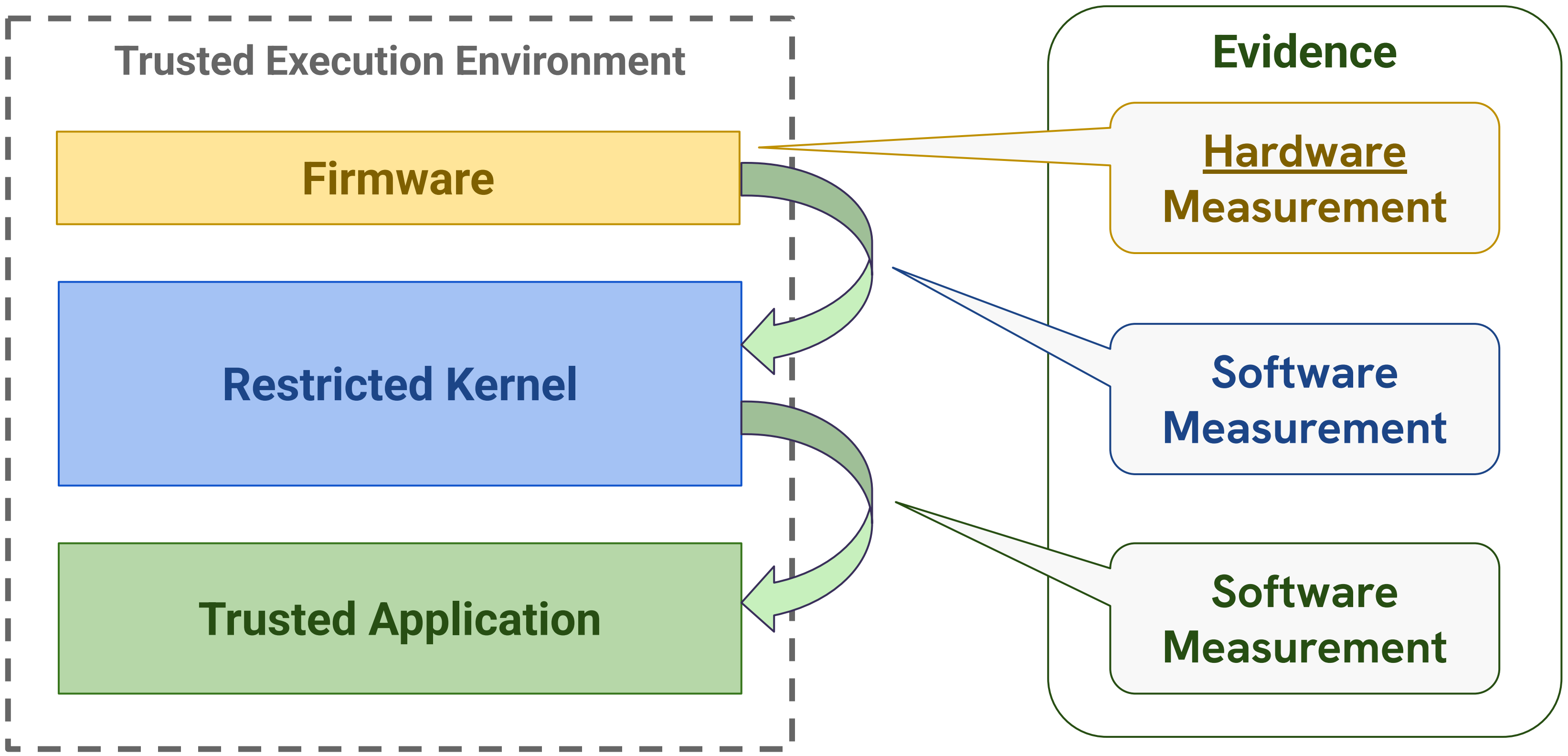
- Minimal syscall interface
- Single process, single-threaded
- No unattested executable pages

Features

- Minimal TCB
- Written in Rust
- Attestation stays valid after boot

Google

Device Identifier Composition Engine (DICE)



Goal

Use a Minimalistic Crypto Protocol

- Bind encrypted channel with remote attestation
- Don’t need PKI
- Don’t need certificates
- Minimize the amount of parsers
- Rust-only implementation

Goal

Use a Minimalistic Crypto Protocol

- Bind encrypted channel with remote attestation
- Don’t need PKI
- Don’t need certificates
- Minimize the amount of parsers
- Rust-only implementation

Noise Protocol Framework

www.noiseprotocol.org

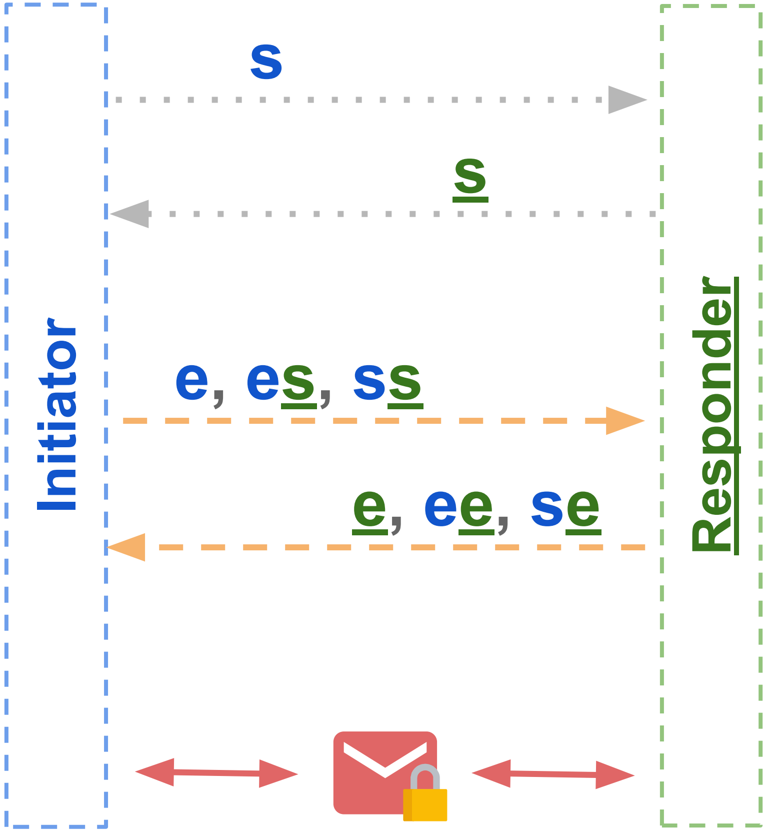
Noise Protocol

- Framework for building simple crypto protocols
- Directly based on Diffie-Hellman key agreement
  - *No certificates/certificate authorities*
- Doesn’t restrict the wire format
  - Protocol provides bytes
- Authentication is optional

Noise Protocol: patterns

- Noise patterns are based on the keys used in the handshake
  - *Ephemeral keys*
  - *Static keys, e.g., long term identity key*
    - *Pre-shared with the other party*
    - *Exchanged during the handshake*
- Formal proofs for confidentiality and authentication security guarantees
- Handshake pattern analysis tool: noiseexplorer.com

Noise Protocol



Notation

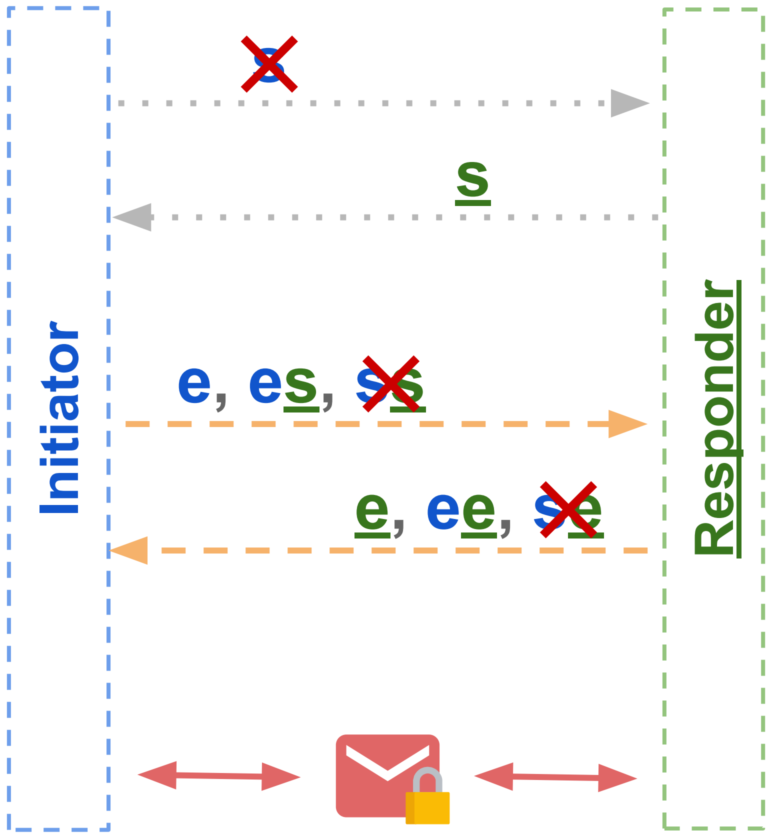
- **s** - *static key*
- **e** - *ephemeral key*
- **es, ee, ...** - *Diffie-Hellman*

Key agreement

- Rules for updating the local state
- Used to produce 2 symmetric keys (encryption/decryption)

Google

Noise Protocol



Notation

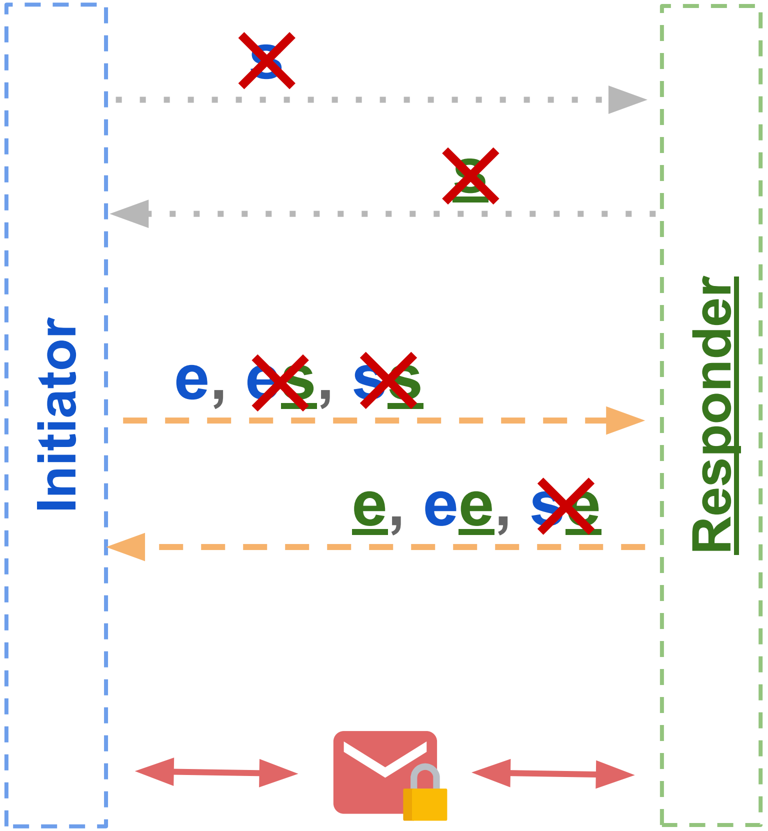
- **s** - *static key*
- **e** - *ephemeral key*
- **es, ee, …** - *Diffie-Hellman*

Key agreement

- Rules for updating the local state
- Used to produce 2 symmetric keys (encryption/decryption)

Google

Noise Protocol



Notation

- **s** - *static key*
- **e** - *ephemeral key*
- **es**, **ee**, ... - *Diffie-Hellman*

Key agreement

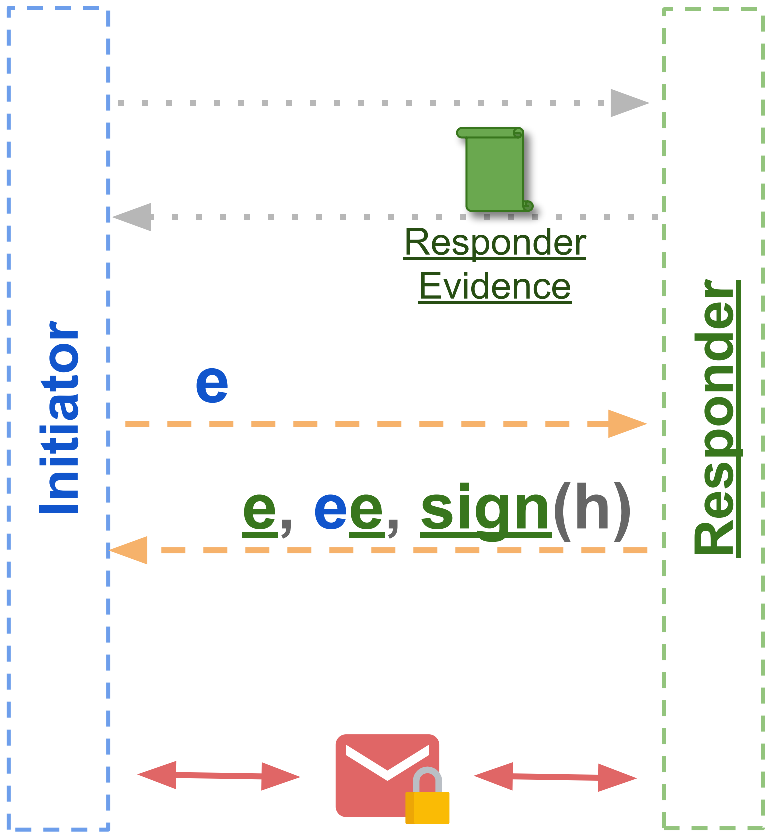
- Rules for updating the local state
- Used to produce 2 symmetric keys (encryption/decryption)

Google

Noise Attestation

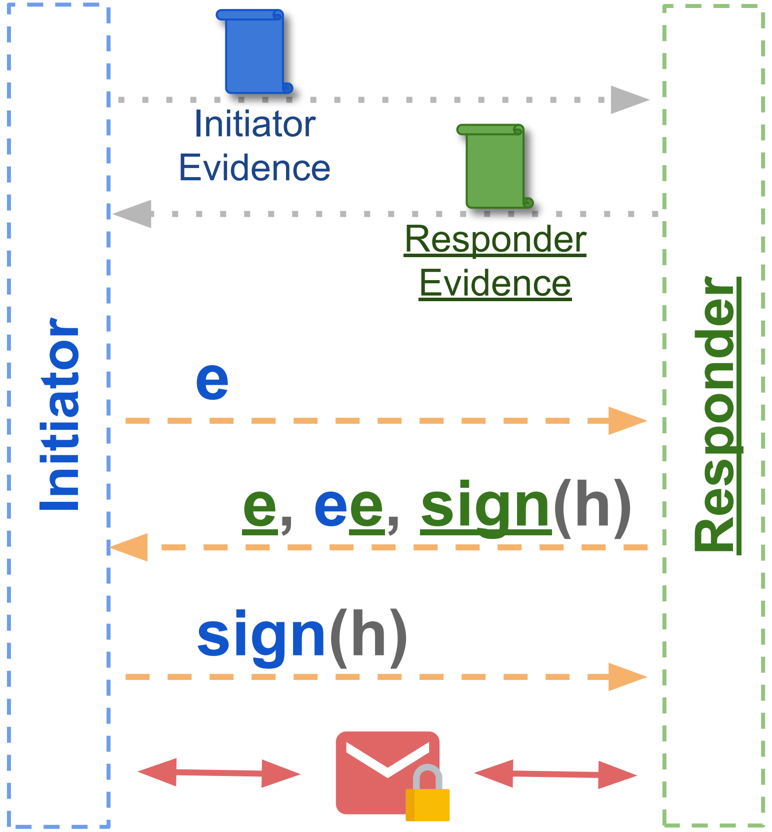
- Bind attestation to the Noise handshake
  - *Allows making it a <u>separate</u> step*
- Use Noise without modifications
  - *Retains security formal proofs*
- Supports bidirectional attestation
- Supports multiple attestations

Noise Attestation



- Responder provides attestation evidence
- Evidence contains a binding key
- Binding is done by signing the handshake transcript **h**
  - *Includes a usage string*

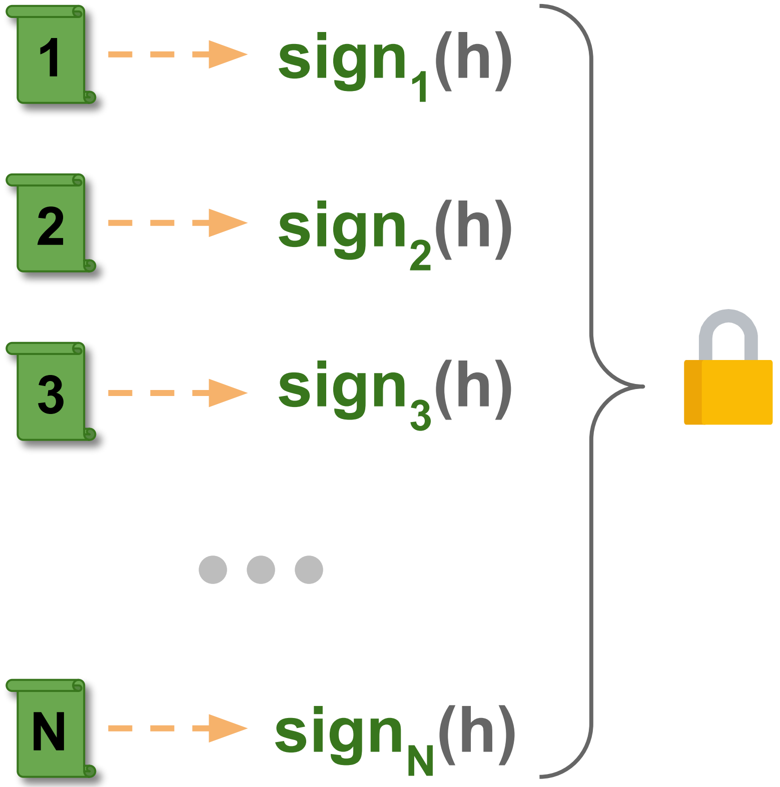
Bidirectional Noise Attestation



- The same approach can be applied to attest both parties

Google

Multiple Attestations



- This approach also allows us to bind multiple attestations to the channel
  - *By signing the handshake with individual binding keys*
- This feature can be useful, if the system has multiple attestable components

Google

Noise

- Small implementation:
    - 0.9K LOC Noise implementation
    - 2.5K SDK for attestation binding
    - Small subset of Rust Crypto
- Doesn’t need additional parsers
- Provides patterns that don’t require PKI

But:

- Custom solution

TLS

- Standard well accepted solution
- Wide variety of features for authentication support

But:

- BoringSSL
    - Threading
    - Standard library for C++ bindings
    - 1.6M LOC
        - but it’s *not* a fair comparison

Conclusion

- Use-case which minimizes the TCB
- Need for a minimalistic crypto protocol
- Use Noise Protocol Framework
- Bind end-to-end encrypted channel to remote attestation

Links

- Project Oak: github.com/project-oak/oak
- Noise Implementation:
  github.com/project-oak/oak/tree/main/oak_crypto/src/noise_handshake
- Attestation SDK: github.com/project-oak/oak/tree/main/oak_session

Google