

ArmoredSoftware: Trust in the cloud

Annual Demonstration

Dr. Perry Alexander, Dr. Andrew Gill, Dr. Prasad Kulkarni,
Adam Petz, Paul Kline, Justin Dawson, Jason Gevargizian,
Leon Searl, Edward Komp

Information and Telecommunication Technology Center
Electrical Engineering and Computer Science
The University of Kansas
palexand@ku.edu, andygill@ku.edu, prasadk@ku.edu

January 15, 2015



Introduction and Project Goals

- Big Picture

- Implementation

Prototype demonstration and discussion

- Refine big picture to current demo

- Protocol Execution

- Appraisal

- Attestation Protocol Execution

- Measurement

- Communication

Short term goals and milestones

Questions and guidance



Trust in the Cloud

Provide new capabilities that establish and maintain trustworthy cloud-based application deployment

- ▶ Establish trust among cloud components
 - ▶ trust among cohorts of processes
 - ▶ trust among processes and environment
- ▶ Promote informed decision making
 - ▶ data confidentiality can be confirmed
 - ▶ execution and data integrity can be confirmed
- ▶ Autonomous run-time response and reconfiguration
 - ▶ responds to attack, failure, reconfiguration, and repair
 - ▶ response varies based on measurement



- ▶ Lightweight integration with existing cloud infrastructure
 - ▶ OpenStack cloud infrastructure
 - ▶ Xen+XSM VM infrastructure
 - ▶ Fedora, HotSpot JVM, GHC
- ▶ Trusted Computing Group standards compliant
 - ▶ Trusted Platform Module 1.2
 - ▶ TCG vTPM (in principle)
 - ▶ Trusted OS infrastructure
- ▶ Standard communication mechanisms
 - ▶ JSON structures for all exchanged data
 - ▶ *vchan* for on-platform communication
 - ▶ TCP/IP for off-platform communication



- ▶ Trustworthy protocol execution
 - ▶ executable protocol representation
 - ▶ protocol execution generates evidence of trustworthiness
 - ▶ highly focused protocols
 - ▶ strand space formal semantics
- ▶ Application specific measurement
 - ▶ managed and traditional execution environments
 - ▶ compile-time assistance for measurer synthesis
 - ▶ specialized measurement bundled with applications
- ▶ Attestation driven cloud application and data management
 - ▶ health monitoring
 - ▶ problem mitigation
 - ▶ application migration
 - ▶ access control



Research & Development Plan

- ▶ **Development and integrate measurement capabilities**
 - ▶ hosted languages (Java)
 - ▶ traditional compiled languages (C, C++)
 - ▶ integrate with environment measurers (Xen, OpenStack, OS)
- ▶ **Develop attestation capabilities**
 - ▶ flexible, user configurable protocol representation
 - ▶ measured protocol execution
 - ▶ protocol execution appraisal
- ▶ **Develop infrastructure trust argument**
 - ▶ develop lightweight vTPM infrastructure supporting mobility
 - ▶ launch from known roots of trust
 - ▶ maintain trust evidence at run time
 - ▶ maintain trust over migration



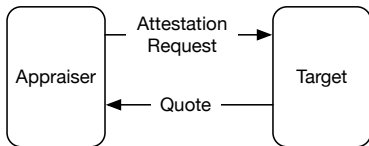
Research & Development Plan

- ▶ Automated synthesis and verification
 - ▶ measurer synthesis at application compile time
 - ▶ automated evidence appraisal from protocols
 - ▶ formal trust argument
- ▶ Demonstrations
 - ▶ initial simple infrastructure demonstrations
 - ▶ cloud-based “big data” environment demonstration
 - ▶ federated trust demonstration
 - ▶ *demonstrations as discovered/directed*
- ▶ Scale up and roll out
 - ▶ integration with Xen, OpenStack, Linux
 - ▶ installation management and packaging
 - ▶ effective web presence



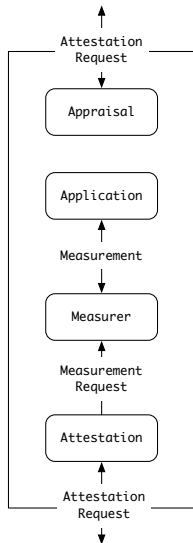
Semantic Remote Attestation

- ▶ Appraiser requests a quote
 - ▶ specifies needed information
 - ▶ provides a nonce
- ▶ Target gathers evidence
 - ▶ measures application
 - ▶ gathers evidence of trust
- ▶ Target generates quote
 - ▶ measurements and evidence
 - ▶ original nonce
 - ▶ cryptographic signature
- ▶ Appraiser assesses quote
 - ▶ good application behavior
 - ▶ infrastructure trustworthiness

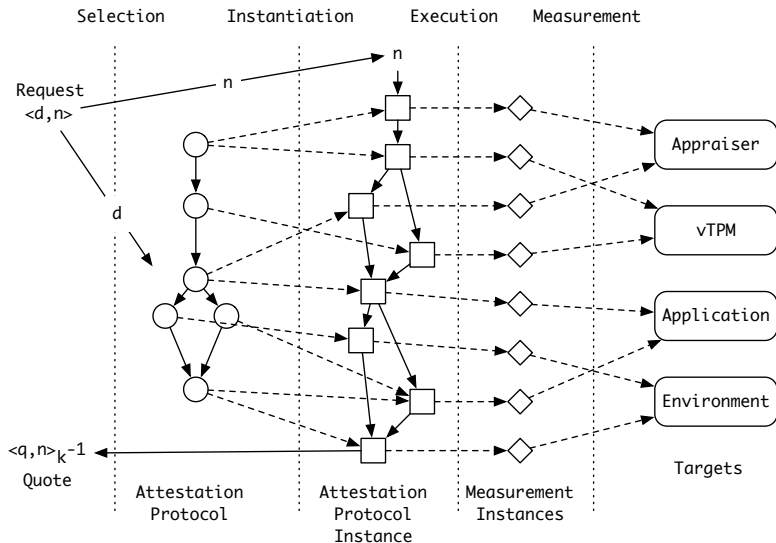


Armored Application Architecture

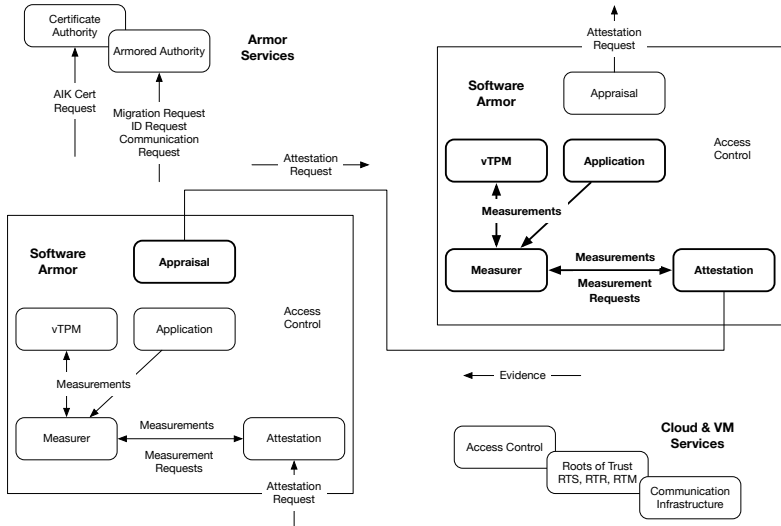
- ▶ Focus is user-space applications
- ▶ Assesses the cloud infrastructure and environment
- ▶ Attests to the state of its application
- ▶ High-assurance, lightweight infrastructure
- ▶ Influenced by the *Trusted Research Platform* and *Principles of Remote Attestation*



Measurement and Attestation



System-Level Architecture

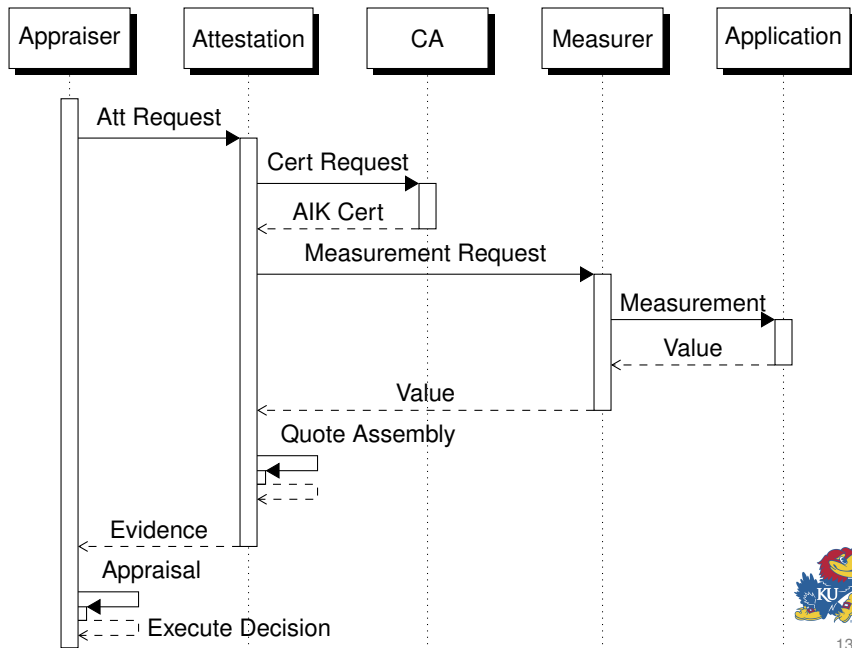


What We Are Demonstrating

- ▶ Execution of a CA-based Attestation Protocol
 - ▶ Attestation request
 - ▶ Protocol execution
 - ▶ Evidence appraisal
- ▶ Major architectural subsystems
 - ▶ Appraiser
 - ▶ Attestation Manager
 - ▶ Measurer
 - ▶ Instrumented JVM
 - ▶ vTPM and Certificate Authority
- ▶ Anomaly Detection
 - ▶ Bad signatures and PCRs
 - ▶ Bad CA certificates
 - ▶ Bad quotes and AIKs
 - ▶ Bad measurements



Abstract CA-Based Attestation Protocol



Message List Representation

App \rightarrow *Att* : d, N_{App}, PCR_m on C_{AppAtt}

Att \rightarrow *TPM* : *make_and_load_identity* on C_{AttTPM}

TPM \rightarrow *Att* : AIK^+, AIK_h on C_{TPMAtt}

Att \rightarrow *CA* : *Att*, AIK^+ on C_{AttCA}

CA \rightarrow *Att* : $\{K, |AIK|\}_{EK^+}, \{[AIK^+]_{CA-}\}_{K^+}$ on C_{CAAtt}

Att \rightarrow *TPM* : *activate_identity*($AIK_h, |AIK|$) on C_{AttTPM}

TPM \rightarrow *Att* : K on C_{TPMAtt}

Att \rightarrow *Meas* : d on $C_{AttMeas}$

Meas \rightarrow *Att* : e on $C_{MeasAtt}$

Att \rightarrow *TPM* : *quote*($AIK_h, PCR_m, |(e, N_{App}, [AIK^+]_{CA-})|$) on C_{AttTPM}

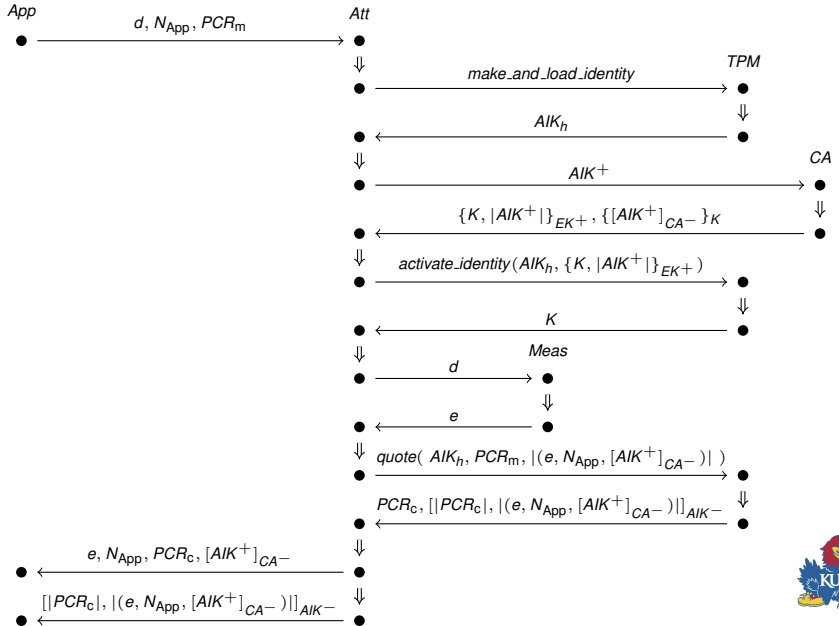
TPM \rightarrow *Att* : $PCR_c, [|PCR_c|, |(e, N_{App}, [AIK^+]_{CA-})|]_{AIK-}$ on C_{TPMAtt}

Att \rightarrow *App* : $e, N_{App}, PCR_c, [AIK^+]_{CA-}$ on C_{AttApp}

Att \rightarrow *App* : $[|PCR_c|, |(e, N_{App}, [AIK^+]_{CA-})|]_{AIK-}$ on C_{AttApp}



Strand Space Diagram Representation



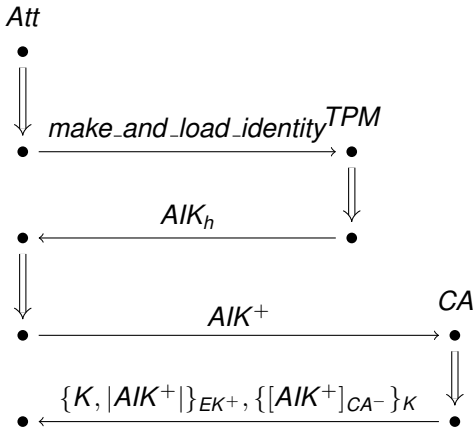


- ▶ Initiate with an attestation request
 - ▶ d abstractly defines desired evidence
 - ▶ N_{App} is the appraiser's nonce
 - ▶ PCR_m selects PCRs
- ▶ Attestation agent selects and executes protocol based on request

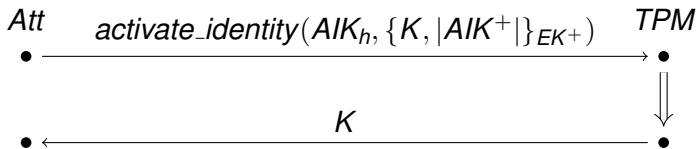


Generating and Certifying an AIK

- ▶ Request a new *AIK* from TPM (optional)
- ▶ Receive *AIK* handle
- ▶ Request AIK^+ signed by CA (*AIK* cert)
- ▶ Receive *AIK* cert encrypted with session key K
- ▶ Receive K encrypted with public EK

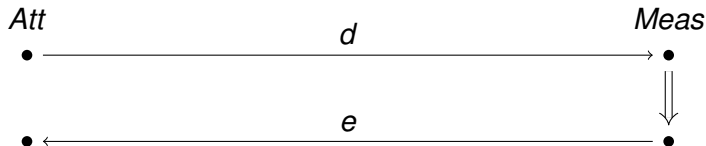


Activating the AIK



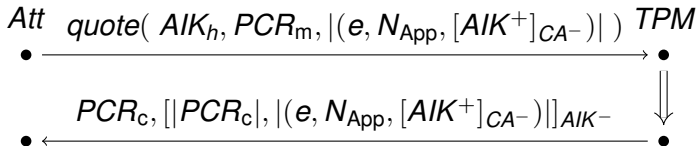
- ▶ Request TPM decryption of the *AIK* cert
- ▶ Receive K used to decrypt signed public *AIK*
- ▶ Only TPM can gain access to K
- ▶ Only TPM can obtain signed, public *AIK*
- ▶ Oddly, No manipulation of the *AIK* in this “activation” process





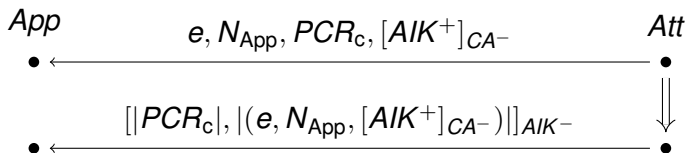
- ▶ Request information from measurer
- ▶ Receive evidence e from measurer
- ▶ d is abstract allowing protocol reuse
- ▶ Most protocols make many requests of the measurer





- ▶ Request a quote from the TPM
 - ▶ AIK identifies the signing AIK
 - ▶ PCR_m identifies desired PCRs
 - ▶ $|(e, N_{App}, [AIK^+]_{CA-})|$ guarantees integrity of returned evidence
- ▶ Receive quote from TPM
 - ▶ PCR_c is PCR composite built from requested PCRs
 - ▶ $[|PCR_c|, |(e, N_{App}, [AIK^+]_{CA-})|]_{AIK-}$ is the signed quote





- ▶ Receive evidence from the attestation manager
 - ▶ evidence
 - ▶ original nonce
 - ▶ PCR composite
 - ▶ signed AIK^+
- ▶ Receive TPM quote from the attestation manager
 - ▶ hash of all evidence
 - ▶ PCR composite
 - ▶ signed by AIK^-
- ▶ Evaluate evidence and quote



Demonstration detects failure of all aspects of attestation

$$e, N_{\text{App}}, PCR_C, [AIK^+]_{CA-}$$

- ▶ e – evidence gathered from running application
- ▶ N_{App} – prevents replay
- ▶ PCR_C – evidence in the form of PCR data from the vTPM
- ▶ $[AIK^+]_{CA-}$ – ensures validity of AIK^+

$$[|PCR_C|, |(e, N_{\text{App}}, [AIK^+]_{CA-})|]_{AIK-}$$

- ▶ PCR_C – hash ensures integrity of PCR data
- ▶ $|(e, N_{\text{App}}, [AIK^+]_{CA-})|$ – hash ensures integrity of evidence, nonce, and signed AIK^+



Attestation Protocol Execution



► Measurement Mechanisms

- Sampling: Measurer spawns asynchronous sampler thread to take measurements in parallel with application threads
- Instrumentation: Measurer inserts/toggles code instrumentation in the application threads to take measurements

► Measurement Components

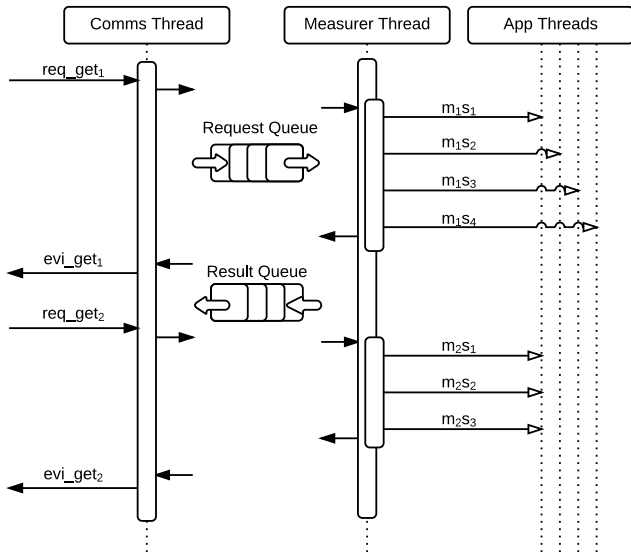
- Measurer Thread services individual measurements, dispatches sampler threads, and toggles measurement instrumentation
- Communications Thread communicates between the Attester (Vchan) and the Measurer Thread (within run-time)

► Measurement Request Types

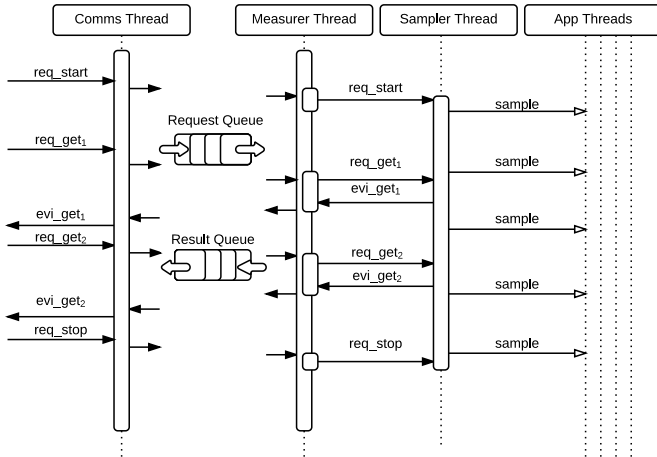
- individual measurements: get evidence
- continuous measurements: initiate, terminate, get evidence



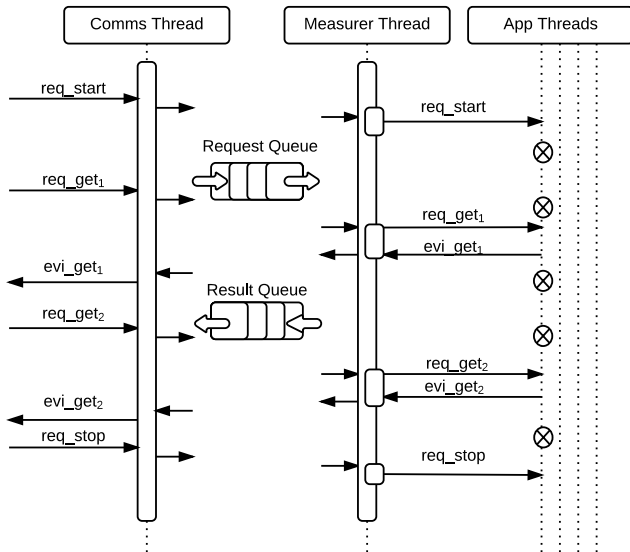
Measurer - Individual Sample



Measurer - Continuous Sampling



Measurer - Continuous Instrumentation



► Current Demo

- Individual measurement of user variable, specified externally
- Performed by the measurer thread by monitoring the stack or heap contents

► Static Analysis to define Application Expectations

- Use static analysis to generate "golden" expectations for applications. Run-time measurements should fall within these defined expectations.
- E.g. Generate a static callgraph and provide this as the "golden" bounds to the appraiser. Track the dynamic callgraph during execution. Send the dynamic callgraph to the appraiser when requested to ensure the program never breaks the static expectation.



2-3 Slides on Communication Mechanisms



Shared notion of AIKCertRequest, AIKCert, and CAResponse JSON structures.

Attester

- ▶ creates an AIKCertRequest (containing attester ID , AIK^+) and converts to JSON
- ▶ JSON sent as POST request to CA running as web server

Certificate Authority

- ▶ POST body bytes \rightarrow UTF8 \rightarrow JSON \rightarrow AIKCertRequest
- ▶ looks up EK^+ associated with ID in sql database
- ▶ $AIKCert = AIK^+$ signed with CA^-
- ▶ generates key K and encrypts with EK^+
- ▶ AIKCert encrypted with K
- ▶ both wrapped in a CAResponse, converted to JSON and sent as response.



Properties

- ▶ CA only responds to receiving an *AIKCertRequest*_{JSON}
- ▶ The CACert can *only* be decrypted by knowing K (and therefore EK^-)

Appraiser Knowledge after receiving Cert:

- ▶ signature on *AIK* ensures it was CA who generated signature
+
- ▶ only an entity knowing EK^- could decrypt and send the CACert
=
- ▶ **Attester is using a registered TPM**



Completed four demonstrations culminating in running an attestation protocol in response to an attestation request.

- ▶ **Attestation and Appraisal development**
 - ▶ CA-Based attestation protocol execution example
 - ▶ integration with Berlios TPM 1.2 emulator
 - ▶ simple dynamic appraisal of attestation results
- ▶ **Measurement development**
 - ▶ on demand Java program measurement
 - ▶ HotSpot-based Java VM run time measurements
 - ▶ standard mechanism for extending measurement capabilities
- ▶ **Communication infrastructure**
 - ▶ vchan, TCP/IP and socket communication infrastructure
 - ▶ language-based interface with TPM 1.2
 - ▶ JSON-based data exchange formats
 - ▶ initial certificate authority API



Goals and Milestones for 2015


- ▶ Push to the cloud
- ▶ Establish roots of trust and trust argument
- ▶ Executable protocol representation and protocol semantics
- ▶ Operational, integrated vTPM prototype
- ▶ Name Server / Certificate Authority prototype
- ▶ More capable measurement
- ▶ Downloadable demonstration



Questions and Guidance


- ▶ What problems are interesting?
- ▶ What problem would be a nice attention grabber?
- ▶ What should we be watching and integrating with?




-  Coker, G., Guttman, J., Loscocco, P., Herzog, A., Millen, J., O'Hanlon, B., Ramsdell, J., Segall, A., Sheehy, J., and Sniffen, B. (2011).

Principles of remote attestation.


International Journal of Information Security, 10(2):63–81.

-  Fábrega, F. J. T., Herzog, J. C., and Guttman, J. D. (1999).
Strand spaces: Proving security protocols correct.

Journal of computer security, 7(2):191–230.

-  Halder, V., Chandra, D., and Franz, M. (2004).
Semantic remote attestation – a virtual machine directed approach to trusted computing.

In *Proceedings of the Third Virtual Machine Research and Technology Symposium*, San Jose, CA.

-  Ryan, M. (2009).
Introduction to the tpm 1.2.
Draft Report.

