Using the TPM: Machine Authentication and Attestation

Ariel Segall
ariels@alum.mit.edu

Day 2

Approved for Public Release: 12-2749.
Distribution unlimited

License

All materials are licensed under a Creative Commons "Share Alike" license.

- http://creativecommons.org/licenses/by-sa/3.0

You are free:

to Share — to copy, distribute and transmit the work

to Remix — to adapt the work

to make commercial use of the work

Under the following conditions:

Attribution — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

Share Alike — If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.

What We'll Cover

- Review and deep dive: PCRs and Locality
- Attestation
- Machine authentication

PCR Review

- Series of 20-byte registers (length of a SHA-1 hash)
- Most modern TPMs have 24; older ones have 16
- Addressed by number: PCR-0, PCR-1, etc.
- Used primarily to store system measurements
- Reset to known value on every boot
- Can never be freely overwritten; use special extend, reset operations
- Easy to check; computationally infeasible to forge

Digging a Little Deeper: PCR Extend

The only way to *add* data to a PCR is with `TPM_Extend`

- Current value of a PCR is X. (Say, 0x0000....0000.)
- We extend the PCR with some data Y.
  - Y must be 160 bit (20 byte) value
  - 20 bytes = SHA1 hash, allowing longer data
- TPM calculates hash(Y,X)=Z; changes value in PCR to Z.
- We can update further:
  - Extend with A: value is hash(A,Z)=hash(A, hash(Y,X))
  - Extend with B: PCR value is hash(B, hash(A,Z))
  - ...etc.
- Verifiers who know the values extended into the PCRs can easily verify
  - Perform the same hash chain themselves
- Computationally infeasible to forge (must break SHA-1)
  - Given PCR state N and desired state M, adversary would need to find X such that hash(X,N)=M; violates one-way assumption

Digging a Little Deeper: PCR Reset

Some (but not all) PCRs are *resettable*.

This means they can be reset to a known state by executing the `TPM_PCR_Reset` command.

- Whether a given PCR is resettable or not is defined in platform spec
  - All PC client TPMs have the same settings
  - Server or virtual TPMs could differ; specs do not exist yet
- Reset requires appropriate permissions
  - Usually based on *locality*, which we'll discuss next
- Sets PCR value back to default, erasing all data currently present
  - Either 0x000...000 or 0xFFF...FFF, depending on PCR & machine state

Locality

What is locality?

- Primitive caller-based access control for TPM
- Based on CPU state, flags on memory pages
  - If you're familiar with OS rings, similar concept
- PCRs provide state check; locality says "who sent this command?"
  - Even if my OS is in approved state, random apps shouldn't be able to use OS' keys
- Used to regulate access to PCRs
- Locality checked whenever PCR state checked
- Not utilized much today outside DRTM

Locality in Theory

Locality	Meaning
4	Trusted Hardware/DRTM
3	Auxiliary Components/DRTM
2	Trusted OS
1	Trusted Applications
0	Static RTM/Legacy

Locality in Practice

Locality	Meaning
4	Trusted Hardware/DRTM
3	Software launched by DRTM
2	Controlled by OS/TPM Driver
1	Controlled by OS/TPM Driver
0	SRTM; Default

PCR Usage

PCR Index	Alias	pcrReset	pcrResetLocal for Locality 4, 3, 2, 1, 0	pcrExtendLocal for Locality 4, 3 ,2, 1, 0
0 – 15	Static RTM	0	0,0,0,0,0	1,1,1,1,1
16	Debug	1	1,1,1,1,1	1,1,1,1,1
17	Locality 4	1	1,0,0,0,0	1,1,1,0,0[2]
18	Locality 3	1	1,0,0,0,0	1,1,1,0,0
19	Locality 2	1	1,0,0,0,0	0,1,1,0,0
20	Locality 1	1	1,0,1,0,0	0,1,1,1,0
21	T/OS Controlled	1	0,0,1,0,0	0,0,1,0,0
22	T/OS Controlled	1	0,0,1,0,0	0,0,1,0,0
23	Application Specific	1	1,1,1,1,1	1,1,1,1,1

PCR Usage Rules of Thumb

- Static RTM boot process: 0-7
  - Only subset contains real content today
- DRTM launch and code: 17-20
- Need a non-resettable PCR? Use 8-15
  - Set aside for OS, but rarely used
  - Linux trusted boot loader uses 8,9, 12-14
- Need a resettable PCR? Use 16 or 23

The Fragility of PCR Values

- PCR contents are all hash chains
  - Most values extended into PCRs are also hashes
- *Any* change in value will change the hash unpredictably!
  - Did it update the date, or add a rootkit? We can't tell!
- Extremely difficult to predict
  - Holy grail of measurement: golden values reflecting good/bad state
  - Real-world systems generally more chaotic
- Improving predictability, reliability area of active work
  - Research: property-based attestation
  - Industry: standardized measurement techniques and targets
- Still useful!
  - "Are you the same as you were yesterday?"
  - "Are you running our 'gold disk'?"

What We'll Cover

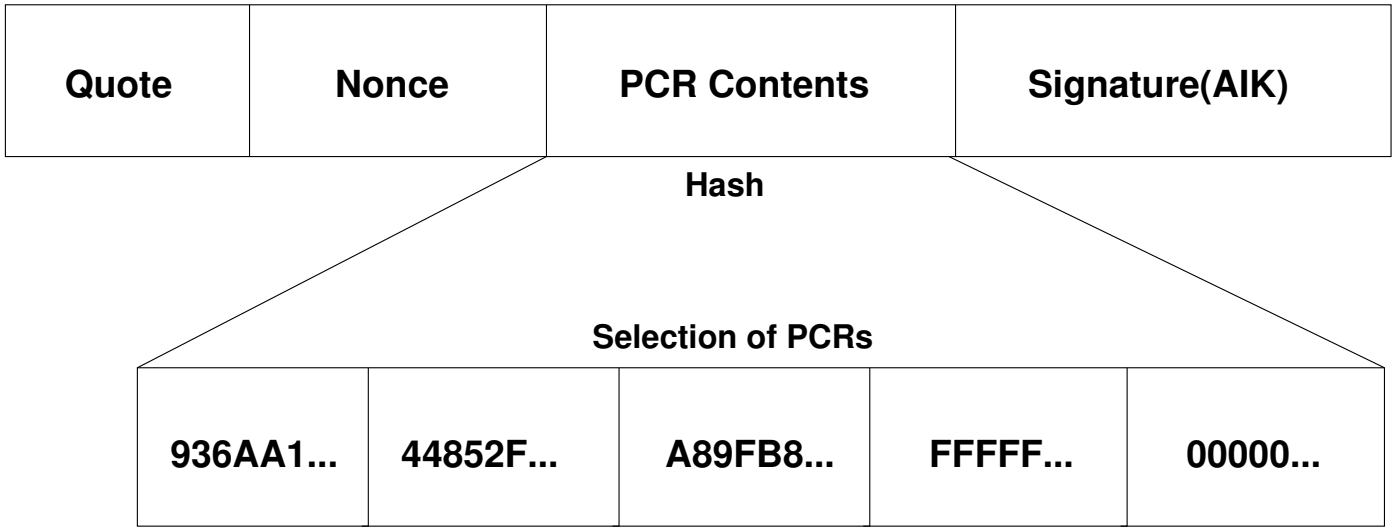
- Review and deep dive: PCRs and Locality
- Attestation
- Machine authentication

What Is Attestation, Again?

Attestation: *the presentation of verifiable evidence about a machine to a remote party*

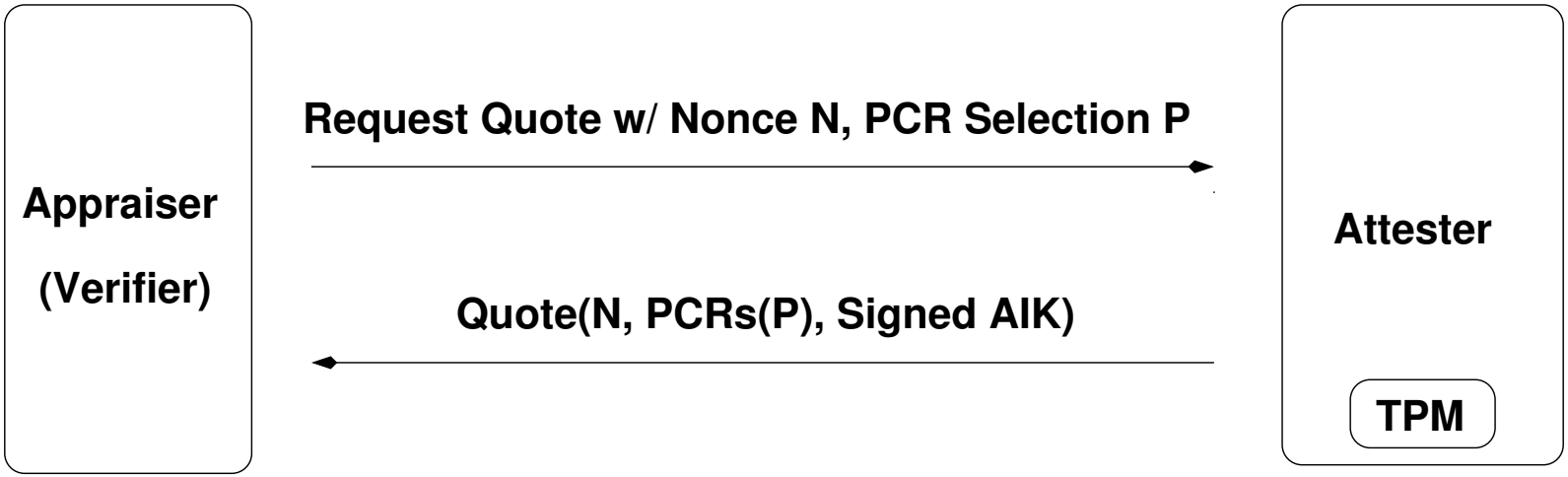
- In TPM context, evidence generally means PCRs
  - Can be augmented; we'll talk more about that
- Verifier (also called *appraiser*) can inspect PCRs, verify chain of trust
  - Trust in high-level components based on good low-level measurements
- Primary tool is *Quote*: signed report of current PCR values
- Any cryptographically verifiable evidence of PCR state counts

Quotes



- Data structured to distinguish from other TPM data
  - Slightly abstracted here for simplicity
- Nonce for freshness, provided by requestor
- Hash of current PCR values
  - Selection of any subset as desired
  - Full record of PCR contents should be provided for verification
- Should be signed using an AIK
  - **Insecure to sign with signing or legacy keys**

Using a Quote

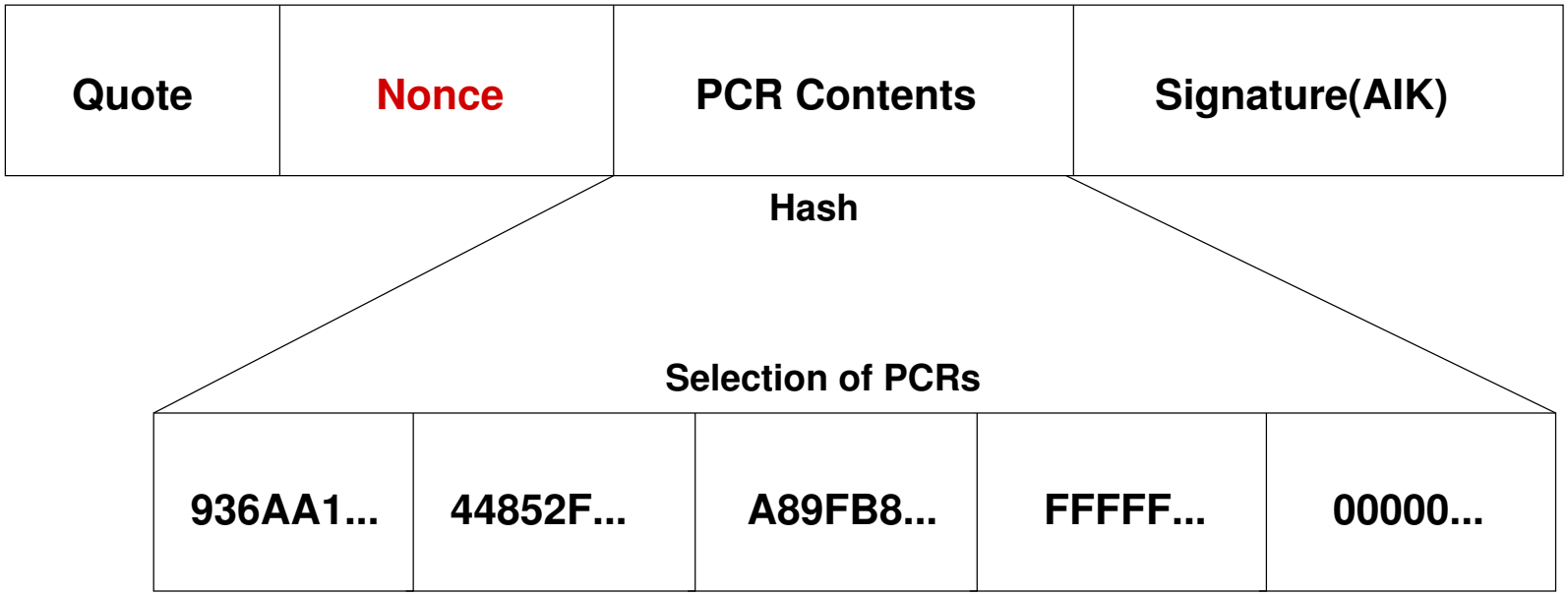


- Many variations, but all follow fundamental structure
- Attester decides:
  - Willing to provide this appraiser with that state info?
- Appraiser decides:
  - Is quote valid, and from legitimate TPM?
  - Is nonce the same one I provided? If fresh, proves quote current.
  - Are PCRs in state I approve of?

Including User Data In Attestation

- Sometimes, we want more than just boot-time system data
  - Add runtime measurements from applications (e.g. config checker)
  - Associate application data with good state (e.g. financial programs)
  - Tie external data to machine, now (e.g. user smartcard)
- Several approaches; simplest is to incorporate into quote
  - We'll cover the others next

How NOT to Include User Data

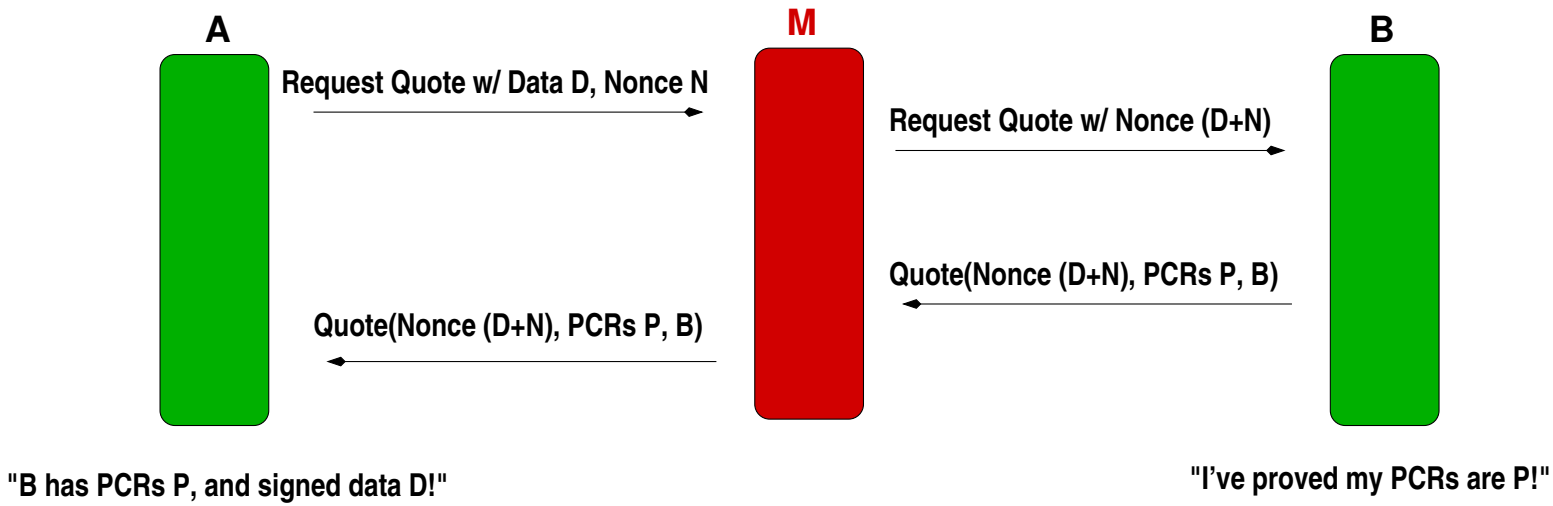


Common approach: replace nonce with user data, or hash(nonce, data)

Do not do this!

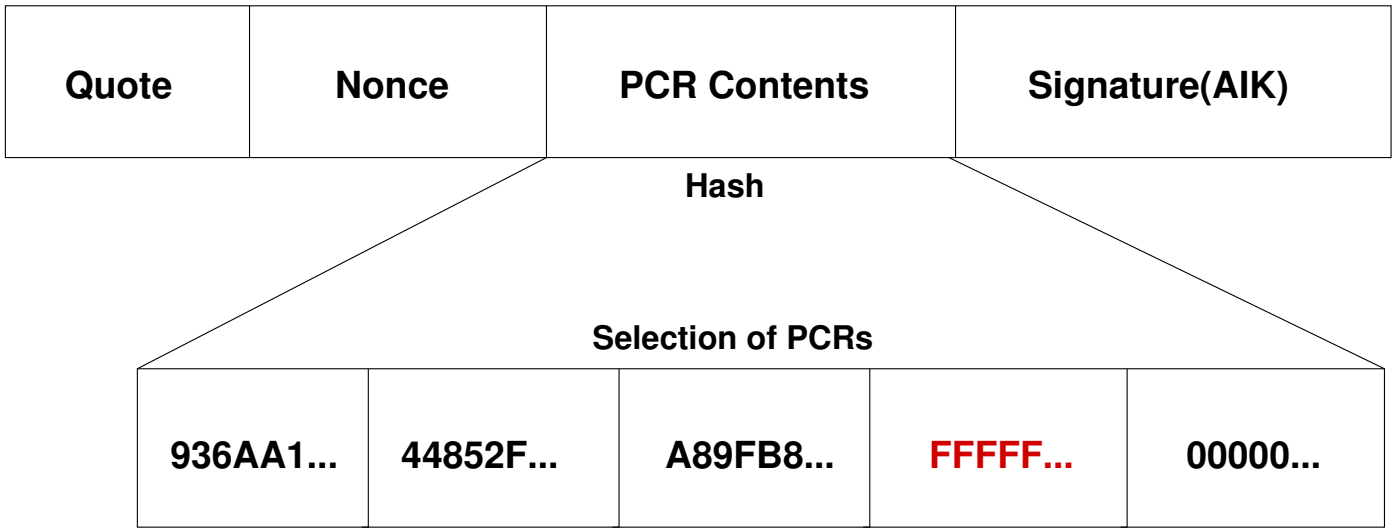
- Nonces are intended for freshness only
- Adding meaning enables a man-in-the-middle attack!

The Nonce-Data Attack



- B can't distinguish random nonce from one with user data
- A assumes B has signed something that B has never seen
- **This is the flaw in TNC's PTS-to-IF-M protocol.**
- Only occurs if multiple quote protocols on same network, but:
  - Other quote applications are powerful, and should not be eliminated
  - Other people may break your protocol accidentally!
  - Do you really have that much control over your network?

How to Safely Include User Data



- Extend data into an otherwise unused PCR
  - Unlike quote, extend not generally network-accessible!
  - Adversary would need platform access to forge
- Remotely verifiable, without changing overall meaning
- Note: Often wise to include freshness within data as well
  - Is this anti-virus report from today? Or yesterday?
- These can be used as long-term signatures!
  - "At the time this was signed, these were the PCR values"

Selecting the Right PCRs for Data

Broadly, two kinds of PCRs:

- Resettable
  - Can be reset to known values
  - Generally, reset before each use
  - **Resettable PCRs are good default for data**
- Non-resettable
  - Reset only on system boot
  - Note: Time since last boot not easy to determine!

Resettable	Non-Resettable
No history	History since boot
Low verifier overhead	Usable for offline updates
Good for one-offs	Good for audit logs

Beyond Quotes: Other Forms of Attestation

- Two primary categories:
  - PCR constraints on keys
  - PCR constraints on data
- Both of these share a common requirement:
  - "Good" PCR values must be known in advance
- Side note: whenever PCRs constrained, locality can be also
  - Very useful for certain DRTM applications, such as Flicker

Attestation with PCR-Constrained Keys

Previously discussed:

- Any TPM key can have PCR constraints associated with it
- Keys can only be used when PCRs match constraints
- CertifyKey allows remote party to verify constraints

Put together:

- Create a key with PCR constraints X.
- Certify the key with your choice of AIK.
- Now, whenever key used, remote party knows PCRs contain X.

PCR-Constrained Key Use Cases

- "When this data was signed, X was true"
  - Good for any case where X implies properties of data; e.g:
  - Certifying correct handling of sensitive data
  - Performing hard-to-verify or security-sensitive operations
- "When this data was encrypted, X was true"
  - Good for tracking possession over multiple boots
  - Good for evaluating data sources
  - Note: not for encrypting to remote platforms! (Seal, not bind)
- "When this data is decrypted, X will be true"
  - Good for ensuring state on receipt
  - Covered in more detail on next slide

Attestation with PCR-Constrained Data

- Sealing (local) allows PCR constraints on encrypted data
  - Can also Bind (remote), but constraints on key, not data
- Useful for data protection!
  - Know both target machine and target state at time of receipt
  - No simultaneous communication required– can create at any time
  - "You can only receive this data if you're in an acceptable state"
- If PCRs predictable, can encrypt to future state
  - Remote: "If you update, you'll gain access"
  - Local: Normal operation can encrypt to secure-mode code
  - Local: Secure-mode app A can encrypt to secure-mode app B
- Can be used to make data only accessible during windows of time
  - e.g., disk decryption key sealed to non-resettable PCRs containing default values; filled in as machine boots

Why no NVRAM attestation?

NVRAM can be constrained to PCR values; why can't it be used to attest?

- Evidence must be *remotely verifiable*.
- Owner sets NVRAM constraints; no way to directly verify!
  - Keys can be certified with CertifyKey
  - Sealed or Bound data can be directly inspected
  - Owner would need to issue certificate; not a standard process
- Proof of access: read data from NVRAM
  - Unless asymmetric key pair, verification requires knowledge of secret
  - If key pair, just use key constraints!
  - If not, certifying association between secret and state leaks info
- Generally, can only attest meaningfully to owner
  - Small enough use case to not worry about

PCR constrained NVRAM very useful! But not for remote attestation.

Attestation Summary

Attestation: *the presentation of verifiable evidence about a machine to a remote party*

- Any verifiable report of PCR contents can be used for attestation
- Quote is primary attestation tool
  - Signed report of current PCR state
  - Can be used to also sign user data via extended PCRs
  - Generally, use resettable PCR for signing data
  - **Never** use quote nonce to sign data!
- PCR constrained keys or data can be used for specialized applications

What We'll Cover

- Review and deep dive: PCRs and Locality
- Attestation
- Machine authentication

What is Machine Authentication?

- Remotely verifiable proof of machine identity
- Useful in many enterprise contexts
- Strict subset of attestation!
  - Attestation: "What is the state of machine X?"
  - Authentication: "Is this machine X?"
- Most TPM keys can be used for machine authentication.
  - TPM soldered to motherboard
  - Keys crytographically bound to TPM
  - All key activity must happen on that machine!

Types of Machine Authentication

Two primary types:

- Signing-based
- Decryption-based

Signing-Based Machine Authentication

- The most intuitive form of machine authentication
- "Machine X signed Y"
- Easy to use to derive trust in Y
- We'll focus on non-attestation signatures
  - Remember, attestation will get you auth for free!
  - But involves overhead you often don't want.

Choosing the right key for authentication is not as easy as it sounds!

- Signing keys
- Identity keys

Signing Key Choices

- Signing keys come in several varieties!
- Choice of key properties can make or break security.
  - Key length: 512-2048 bits; should be 2048
  - Migratability: **must be non-migratable for authentication**
  - Signature scheme: SHA1, DER, INFO
    - Note: all RSA keys; scheme determines what kind of data can be signed

Signing Key Choices

- Signing keys come in several varieties!
- Choice of key properties can make or break security.
  - Key length: 512-2048 bits; should be 2048
  - Migratability: **must be non-migratable for authentication**
  - Signature scheme: SHA1, DER, INFO
    - Note: all RSA keys; scheme determines what kind of data can be signed
    - **Security critical!**

Signature Schemes

SHA1 Signs only 20-byte chunks of data. (SHA-1 hash length!) Will sign user data or TPM data, but **should never be trusted to sign TPM data, as forgery is trivial.**

DER Signs DER-formatted data. Will only sign user data. Should ensure that nothing these sign appear to be TPM data.

INFO Signs data in custom TPM wrapper format. Designed to be a more flexible alternative to identity keys. **Vulnerable to SHA-1 collision attack. Do not use.**

Using Signing Keys for Machine Authentication

- `TPM_Sign` command
- Sign arbitrary user-provided data in key-appropriate format
- SHA1 or DER signatures can often be dropped into existing protocols
  - Use TPM key instead of software key; get machine auth for free
  - Note: slow! Do not use for high-frequency operations
- Associate any data with machine

Using Identity Keys for Machine Authentication

- Identity keys intended for TPM authentication
- **Identity keys are the only secure choice for TPM data**
  - Quotes, CertifyKey certs, audit logs
  - All imply machine auth
- As with signing keys, associate all signed data with machine
- To sign only user data:
  - Extend user data into PCRs; perform quote of only those PCRs
  - Note: high overhead for a simple signature!
- Best used for attestation or TPM-specific functions, however:
- It is not possible to misconfigure an identity key, making them easier to verify!

Verification is Critical

Trust requires a target: trust for what?

- When evaluating TPM signatures, it is critical to ask:
  - Is this data making any claims about TPM state or keys?
  - If so, is this key one I should trust for that purpose?
  - **Never trust a signing key without considering both key and data.**
- This is also the core question when evaluating attestation target!

Types of Machine Authentication

Two primary types:

- Signing-based
- Decryption-based

Decryption-Based Machine Authentication

- We prove identity by demonstrating secret possession
- Decrypting data proves possession of private key
- Remote party can prove identity by creating encrypted challenge
  - If challenge is decrypted and used, proves possession
- We use binding keys for this
  - Storage keys for local data only!

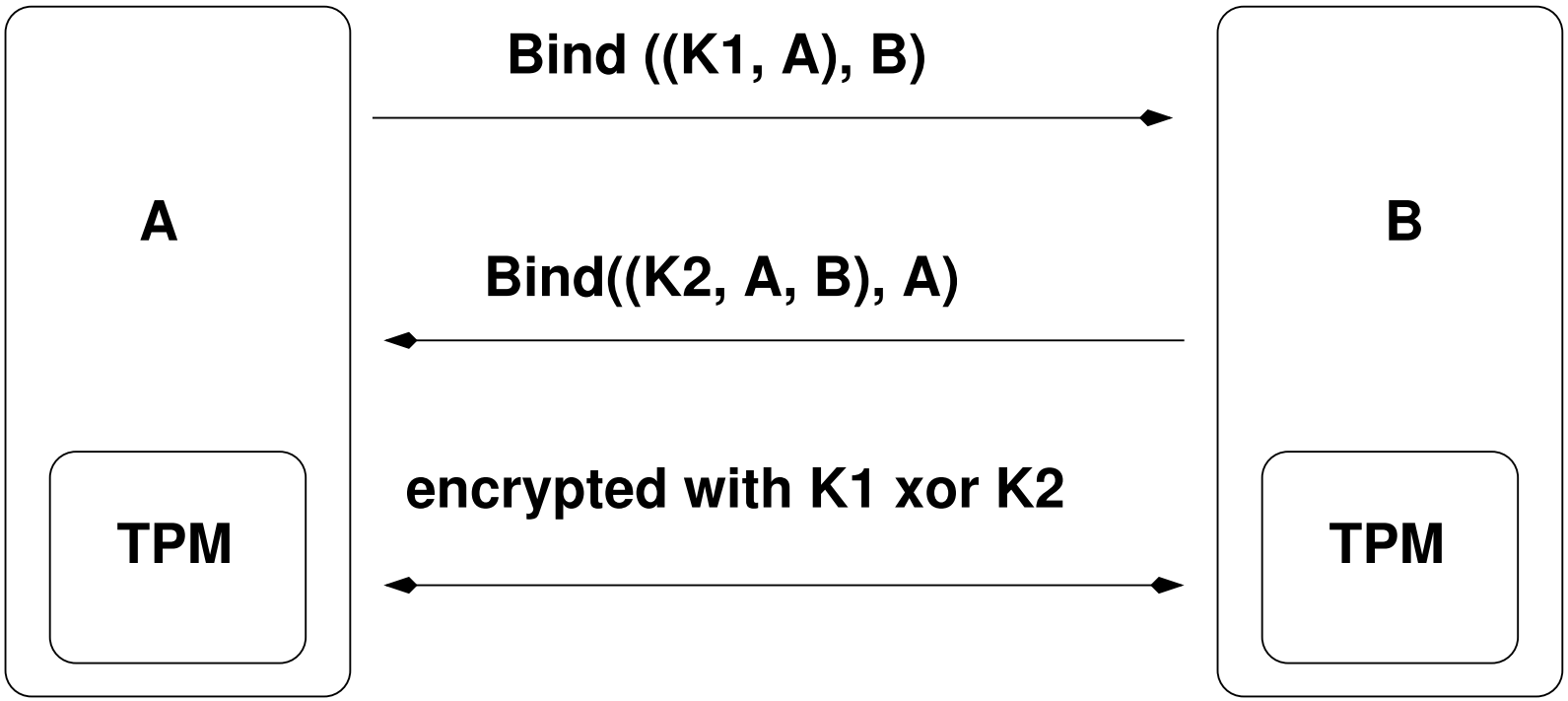
Authenticated Data Delivery

- Simplest form of decryption-based authentication
- Take data intended for target; *Bind* with target binding key
  - Note: `TPM_Unbind` is a TPM operation; Bind is not
  - For practical purposes, large data usually encrypted with symmetric key; symmetric key is bound
- Encrypted data can now be made public; only target can make use of
- Note: Can also perform attestation by binding to PCR-constrained key

Decryption-Based Authentication Protocols

- Many protocols encrypt a secret to prove identity!
  - Nonces, keys, session ID
- Powerful approach: Use TPM key to establish shared session key
  - Public key operations expensive
  - Use to set up shared secret symmetric key
- Simplest possible protocol: bind session key, send to authentication target
- Usually, however, want *mutual authentication*: both parties identified

A Sample Decryption-Based Attestation Protocol



- A knows only B will know K1
- B knows only A will know K2
- Secure channel can be built with combined key

Machine Authentication Summary

- Machine authentication provides proof of machine identity
- Signing or decryption based
- Use SHA1 or DER signing keys for most user data
- Use identity keys for TPM data or high-security applications
- Use binding keys to guarantee data recipient identity

What We've Covered

- Review and deep dive: PCRs and Locality
- Attestation
- Machine authentication

Summary

If you want...	use key type...	with...
Attestation	Identity	Quote
Signing-based Authentication	Signing	Sign
Decryption-based Authentication	Binding	Bind

Questions?