Trusted Computing

Foundations for secure e-commerce
(bmevihim219)

Dr. Levente Buttyán
associate professor
BME Hálózati Rendszerek és Szolgáltatások Tanszék
Lab of Cryptography and System Security (CrySyS)
buttyan@hit.bme.hu, buttyan@crysys.hu
Insecure PCs

- today's desktop and laptop computers are essentially open platforms, giving the user total choice about what software he runs on them, and the power to read, modify or delete files
- this has led to some problems
  - open platforms are prone to infection by viruses and worms, and to inadvertent installation of other malicious software (e.g., Trojans, backdoors, rootkits, etc.)
  - compromised PCs endanger other computers in the network (virus epidemics, botnets, denial-of-service attacks)
  - open platforms allow programs, music files, images to be copied without limit and without loss of quality (this causes problems to software makers and content providers)
- essentially PCs cannot be trusted by their owners, by other parties in the network, an by content providers
Trusted systems

- trusted systems are those whose correct (or predictable) operation we can rely on
- but how to establish this trust?
- correctness proofs?
  - a system on the scale of a modern desktop computer is too complex to be amenable to formal specification, and even less for formal proofs
- extensive testing? long experience of use? third party evaluation and certification?
  - can be very expensive
  - the larger the system in question is, the less effective these methods are
  - comprehensive testing is infeasible, usage experience may be incomplete, third party evaluation may be implausible, and certification is then useless
Trusted Computing Base (TCB)

- it makes sense to limit testing, evaluation, and certification to crucial parts of the system

- Trusted Computing Base (TCB)
  - a small amount of software and hardware that security depends on and that we distinguish from a much larger amount that can misbehave without affecting security (Lampson et al., 1992)
  - the totality of protection mechanisms including hardware, firmware, and software, the combination of which is responsible for enforcing a computer security policy (Orange Book, 1985)

- the question is then if we can expand trust established in a small core to the entire system?
The Trusted Computing Group

- an industry consortium including about 160 organisations
  - Microsoft, HP, Dell, Sony, Lenovo, Toshiba, Vodafone, Seagate, …

- develops, defines and promotes open, vendor-neutral, global industry standards, supportive of a hardware-based root of trust, for interoperable trusted computing platforms

- its main output is the *Trusted Platform Module* specification
  - a hardware chip currently included in 100M laptops
    - HP, Dell, Sony, Lenovo, Toshiba …
  - soldered onto the motherboard, on the LPC bus
  - manufactured by many companies
    - Atmel, Broadcom, Infineon, Sinosun, STMicroelectronics, …
TPM

- provides three roots of trust:
  - Root of trust for measurement (RTM)
    a trusted implementation of a hash algorithm, responsible for the first measurement on the platform — whether at boot time, or in order to put the platform into a special, trusted state;
  - Root of trust for storage (RTS)
    a trusted implementation of a shielded location for one or more secret keys—probably just one, the storage root key (SRK);
  - Root of trust for reporting (RTR)
    a trusted implementation of shielded location to hold a secret key representing a unique platform identity, the endorsement key (EK)
TPM

- enables an entire platform:
  - to strongly identify itself — using public key cryptography, involving a secret key strongly tied to the platform itself
  - to strongly identify its current configuration and running software—using cryptographic hashes of code and other mechanisms
Applications of TC

- DRM
  - music and video files are encrypted, and can only be played by recognized application software on a TC enabled platform
  - the software would prevent users from making copies, and can restrict them in arbitrary other ways
    - e.g. by playing files only a certain number of times, or for a limited period
  - Bill Gates: “We came at this thinking about music, but then we realized that e-mail and documents were far more interesting domains”
  - imagine e-mails that cannot be printed or forwarded, and self-destruct after a specified period
  - similarly, document authors could restrict the ways copies are made or extracts taken by cut-and-paste, etc.
Applications of TC

- third party computing
  - TC can guarantee that the results produced by a remote computer are correct and have been kept confidential

- data security
  - TC can help keep data safe by encryption supported by the hardware key
  - passwords and other private keys can also be kept safe
  - theft of data from stolen laptops can be circumvented

- preventing cheating in multiplayer games
  - cheating by various means is currently prevalent in a number of multiplayer games
    - modification of the game executable or of video drivers
    - modification of game network traffic in transit between the client and server
  - with TC, players can use remote attestation to confirm that neither the game executable nor the video driver has been modified on the other players computer
Criticism of TC

- “With a plan they call trusted computing, large media corporations, together with computer companies such as Microsoft and Intel, are planning to make your computer obey them instead of you.” (Richard Stallman)

- “In 2010 President Clinton may have two red buttons on her desk - one that sends the missiles to China, and another that turns off all the PCs in China - and guess which the Chinese will fear the most?” (Ross Anderson)

- “My fear is that Palladium will lead us down a road where our computers are no longer our computers, but are instead owned by a variety of factions and companies all looking for a piece of our wallet. To the extent that Palladium facilitates that reality, it's bad for society. I don't mind companies selling, renting, or licensing things to me, but the loss of the power, reach, and flexibility of the computer is too great a price to pay.” (Bruce Schneier)
Criticism of TC

- TC removes control of the PC from its owner/user, and gives the control to the software and content providers; this can easily be abused
  - e.g., TC can enforce censorship: if someone writes a paper that a court decides is defamatory, the software company that wrote the word processor could be ordered to delete this paper when encountered
- TC will also allow software companies to increase their monopolies by locking-in users
  - e.g., companies will receive TC-Office documents, and will need TC-Office to read them
  - moreover, they will need to keep paying the rent for TC-Office in perpetuity, if they want to continue to have access to their archives
  - in the long run, the non-TC world may continue to exist, but soon it will be perceived as GNU/Linux is today: great because it gives you more freedom, but a pain because it gives you less choice
TCG doc roadmap
TPM architecture

- Processor
- Hash engine
- RSA key generation
- RSA sign & encrypt
- PRNG
- Non-volatile memory (Endorsement Key, Storage Root Key)
- Volatile memory (Platform Configuration Registers, loaded keys)
TPM functionality

- Secure storage
  user processes can store content that is encrypted by keys only available to the TPM

- Platform measurement and attestation
  a platform can create reports of its integrity and configuration state that can be provided to a remote verifier

- Platform authentication
  a platform can obtain keys by which it can authenticate itself reliably
Secure storage – overview

user process

TPM_CreateWrapKey(keyinfo)

keyblob

TPM_LoadKey2(keyblob)

handle

TPM_Seal(handle, data)

sealed data
TPM authdata

- each TPM object or resource (e.g., a key) is associated with an authdata value
  - a 160-bit shared secret between a user process and the TPM
  - think of it as a password that has to be cited to use the object or resource

- authdata may be a weak (guessable) secret
  - may be based on a user-chosen password (e.g. in Microsoft Bitlocker)
  - the TPM resists online guessing attacks of weak authdata by locking out a user that repeatedly tries wrong guesses
    - details are left to the TPM manufacturer

- user processes issuing a command that uses a TPM object or resource must provide proof of knowledge of the associated authdata
  - HMAC of command arguments, keyed with a value based on the authdata
TPM keys

- TPM keys are arranged in a tree

- Storage Root Key (SRK)
  - created on TPM initialization by calling `TPM_TakeOwnership(…)`
  - stored inside the TPM

- child keys
  - a user process can create a new child key of an existing key by calling `TPM_CreateWrapKey(…)` with the following input
    - handle of the parent key of the key to be created
    - new authdata of the key to be created encrypted with the parent key
    - other key information, e.g., key type (sealing, binding, signature etc.)
    - authorization HMAC based on the authdata of the parent key
  - this returns a key blob consisting of
    - the public part of the new key
    - a package encrypted with the parent key containing
      - the private part of the key
      - authdata associated with the new key
Secure storage – details

- **TPM_LoadKey2(…)**
  - takes as input
    - the key blob of the key to be loaded
    - handle of the parent key (in order to decrypt the key blob)
    - authorization HMAC based on the parent key’s authdata
  - returns a handle (a pointer) to the key stored in the TPM memory temporarily

- SRK is permanently loaded and has a well-known handle value, and therefore never requires to be loaded

- **TPM_Seal(…)**
  - takes as input
    - handle of the encryption key (loaded before)
    - data to be encrypted
    - (information about PCRs to which the seal should be bound)
    - new authdata for the sealed data encrypted with the encryption key
    - authorization HMAC based on the authdata for the encryption key
  - returns the sealed (encrypted) data
Authorization sessions

- allow for issuing multiple commands (with the same setup)

- Object Independent Authorisation Protocol (OIAP)
  - creates a session that can manipulate any object, but works only for certain commands

- Object Specific Authorisation Protocol (OSAP)
  creates a session that manipulates a specific object specified when the session is set up
TPM_OIAP

- creates a session that can manipulate multiple objects at the same time, but works only for certain commands
  - e.g., cannot be used to issue commands that introduce new authdata
- starts with the establishment of an authorization handle
  - each command in the session then repeats this authorization handle
- exchange of new nonces in each step of the session
  - called “odd” nonces (user process) and “even” nonces (TPM)
- all authorization HMACs sent in the session
  - include the most recent odd nonce and even nonce
  - are keyed on the authdata for the resource (e.g., key) being accessed by the command
also used for several commands, but the commands must manipulate a single object specified when the session is set up

it can be used for commands that introduce new authdata to the TPM

the establishment of the authorization handle and the use of odd and even nonces are similar to the case of OIAP

however, in this case, at the beginning of the session, the user process and the TPM compute a shared secret $K$

- $K = \text{HMAC of the odd OSAP nonce and the even OSAP nonce} \ \text{keyed on the object’s authdata}$

OSAP secret $K$ is then used

- to encrypt new authdata introduced to the TPM
- to key the authorization HMACs of commands within the session
  - this permit the user process to cache $K$ for a possibly extended session duration, without compromising the security of the authdata on which it is based
Example – creating a child key

```
user process

TPM

TPM_OSAP(parentKeyHandle, No_OSAP)

authHandle, Ne, Ne_OSAP

K = HMAC(parentKeyAuthData; Ne_OSAP, No_OSAP)

TPM_CreateWrapKey(
    parentKeyHandle, encAuthData, 
    keyInfo, authHandle, No)

HMAC(K; encAuthData, keyInfo, Ne, No)

Ne’, keyBlob

HMAC(K; keyBlob, Ne’, No)
```
Example – loading a key

user process

TPM

TPM_OIAP()

authHandle, Ne

TPM_LoadKey2(
    parentKeyHandle, wrappedKey,
    authHandle, No)

HMAC(parentKeyAuthData;
    wrappedKey, Ne, No)

Ne’, keyHandle

HMAC(parentKeyAuthData;
    keyHandle, Ne’, No)
Example – using a key

user process

TPM

TPM_OSAP(keyHandle, No_OSAP)

authHandle, Ne, Ne_OSAP

K = HMAC(keyAuthData; Ne_OSAP, No_OSAP)

TPM_SeaI(
  keyHandle, encAuthData, pcrInfo,
  data, No, authHandle)

HMAC(K; encAuthData, pcrInfo, data, Ne, No)

Ne’, sealedBlob

HMAC(K; sealedBlob, Ne’, No)
Platform measurement

- measuring a component = computing its hash
- a component can “measure” another component and insert the measurement result into a platform configuration register (PCR) in the TPM
- PCRs can be extended with any number of measurements by calling TPM_Extend(…)

\[
\text{TPM}_\text{Extend}(p, x) \rightarrow p := \text{SHA}-1(p|x)
\]

- this is used to establish a secure chain of trust during the booting process…
Secure booting

- every component A that loads another component B and passes control to it first measures B and extends a PCR with the measurement
- the PCR then represent an accumulated measurement of the history of the executed code from power-up to the present
Attestation and access control

- attestation:
  - the TPM can prove the state of the platform to a third party by signing the value of the PCR that represents the accumulated state of the booting process (TPM_Quote)

- access control:
  - PCR values can be used to ensure that certain data is accessible only to authorised software
    - the TPM_Sea1(…) command can take (PCR, value) pairs as argument
    - then, TPM_Unseal(…) is performed by the TPM only if the specified PCRs have the values that were stipulated at seal time
  - similarly, a TPM key can be locked to work only if certain PCRs have certain values (specified at the time when the key was created)
Static vs. dynamic root if trust

- problems with the approach of ‘static’ root of trust
  - the number of elements in the boot chain is actually very large → maintaining expected values for measurements becomes cumbersome
  - elements are subject to frequent patching and updating → so determining ‘correct’ values for them is non-trivial
  - there are good reasons why some of the elements in the chain may be executed in different orders → leads to different measurement values
  - performance limitations (each step in the chain requires storing away the measurement value in a PCR)
Static vs. dynamic root if trust

- dynamic root of trust approach
  - can be used at any point in time, not just at booting
    - typically before launching a virtual machine monitor or some special, short-lived program which requires a high degree of trust
  - a fixed piece of code is loaded from a trustworthy source which is able to measure and launch a nominated (white-listed) piece of software
  - in this way, the platform can jump directly into a trustworthy state — a ‘measured launched environment’
Limitations of attestation

- attestation tells a remote party exactly what executable code was launched on a platform
  - it does not provide guarantees that the launched programs are bug-free and reliable
  - in particular, programs could have security related bugs, and could be compromised by a buffer overflow attack, infected by a virus, etc.
  - the platform is only as trusted as the tamper resistance of the hardware and the level of assurance of the trusted OS
Platform authentication

- **Endorsement Key (EK)**
  - each TPM has a unique public/private key pair set at manufacturing time and certified by the manufacturer
  - EK is intended to be a long-term key and can be thought of to be the identity of the TPM
  - if EK was used to sign messages from the TPM, then privacy would be breached...

- **Application Identity Keys (AIKs) are used to sign application data and PCR values (TPM Quote)**
  - AIKs need to be certified as belonging to the TPM
  - for reasons of user privacy, the certificates do not specify which TPM they belong to; they will just specify that they belong to a TPM
  - such anonymous certificates are issued by Privacy CAs
Issuing anonymous certs

- drawbacks:
  - the Privacy CA must be trusted for not colluding with relying parties
  - it may be a performance bottleneck
- potential solution: Direct Anonymous Attestation (DAA)