Trusted Computing

David C Blight

Security Architect Voyager Systems dcblight@marzenka.com

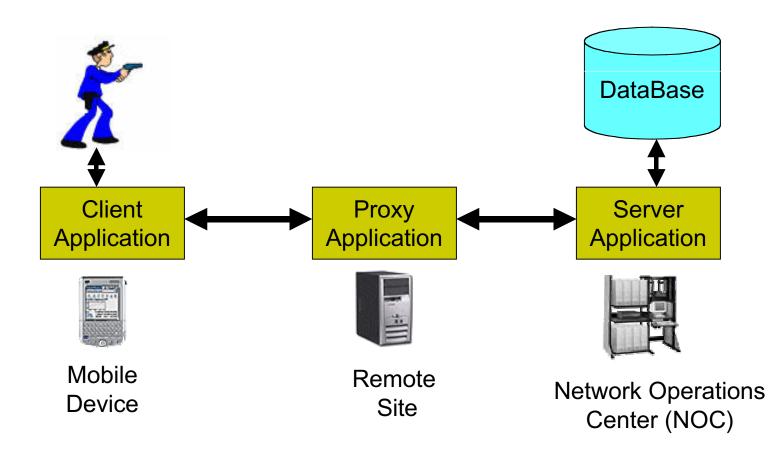
Can you trust your computer?

- By Richard Stallman
- http://www.newsforge.com/article.pl?sid=02/10/21/1449250
- Who should your computer take its orders from?
- Treacherous computing
 - the plan is designed to make sure your computer will systematically disobey you.
 - In fact, it is designed to stop your computer from functioning as a general-purpose computer. Every operation may require explicit permission.

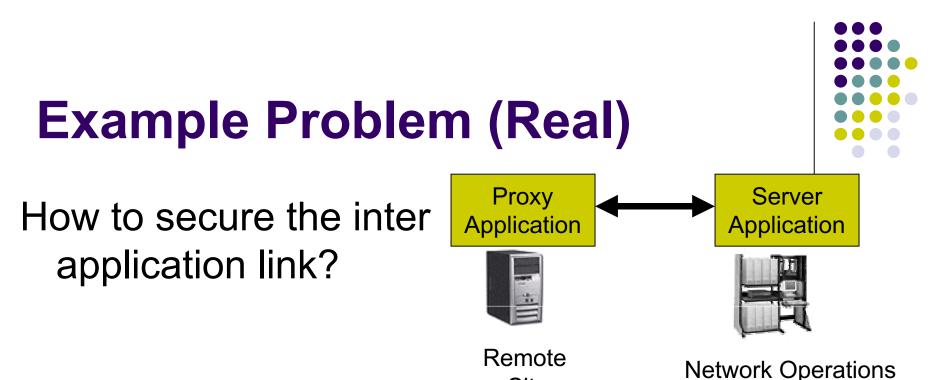
Can your computer trust you?

- Trusted Computing involves verifying that one computer is trustable to another
 - What you do on your computer may make it untrustworthy
- The real goal is to be able to bind data to applications, users, and/or computers.

Example Problem (Real)







Site

- Center (NOC) Encryption is possible without stored secrets
 - Diffie-Hellman
- Authentication requires stored secrets on both systems
 - Store secrets are a vulnerability
 - Applications

Why is this difficult



- Secrets must be stored in persistent storage
- Where is secret stored
 - In Application
 - Applications may be reversed engineered
 - In file system / database
 - Non secure
 - At best protected by encryption, but where is the key stored
 - Obfuscated
 - Non secure

Securing Information

- What is needed
 - Ability to store a secret on a computer such that
 - Application can get the secret
 - No other application can get the secret
 - Secret must be secure within the application
 - No other application can retrieve the secret from the application
- Can not be a software only solution
- Data (secret) needs to be bound to an application.

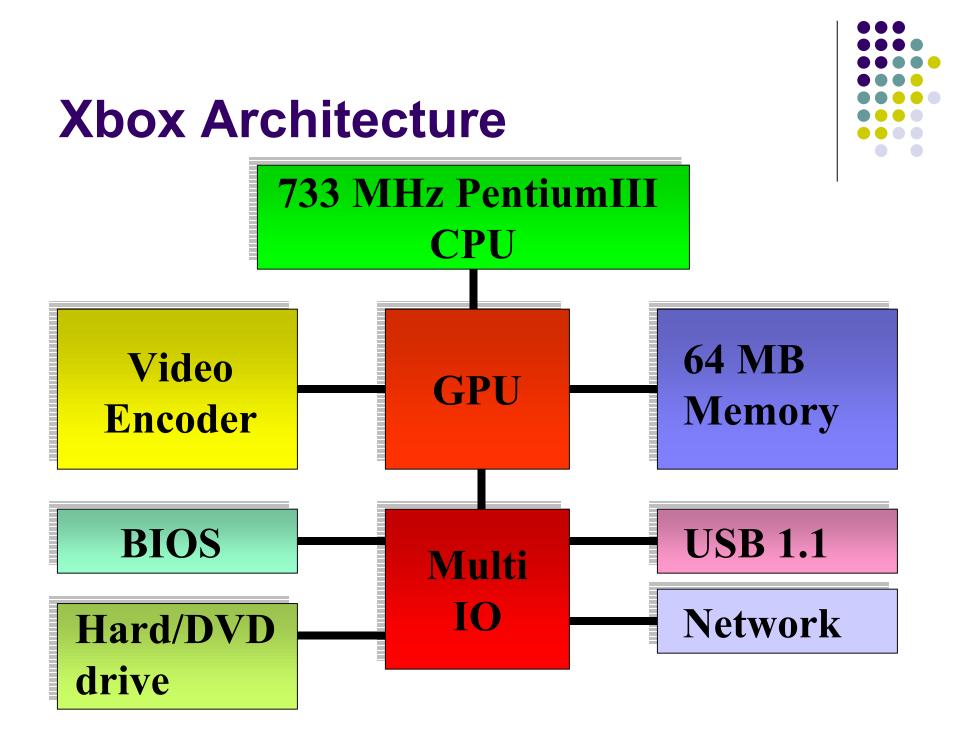


A lesson from Xbox

- Xbox: \$200 PC dedicated to video games
 - Microsoft looses money on each Xbox sold
 - Microsoft makes money on each game sold
- Gamers like to mod video game consoles
 - Increase functionality
 - Circumvent Copy protection







Xbox Security (1.0)

- Boot sequence
 - CPU starts execution at fixed location in ROM
 - This location has op codes to jump to appropriate place in BIOS to continue execution
 - Setup commands
 - GDT, IDT, Jam Table Interpreter
 - Decrypts ROM contents (key is in ROM)
 - Executes decrypted codes.
- BIOS should be replaceble and/or modifiable
 - All info in ROM





Xbox Security (1.0)

• Except

- The boot block in ROM is fake
- The Multi IO chip stored an alternative ROM used only during boot. This code replaces the setup commands in ROM.
- The only info really needed is the RC4 key
 - Alternative BIOS could be used
- Security Broken
 - Secret Data in Multi IO chip could be extracted
 - Sniffing internal buses
 - Security weaknesses in Multi IO chip



Xbox Security (1.1)

- The secret ROM modified
 - Checks hash of ROM section before decrypting
 - Flash Boot Loader (FBL)
 - TEA hash algorithm
- Potentially Stronger Security
 - Its not required to keep data in secret ROM confidential. Only integrity needs to be assured.
- How it was broken
 - Weak hash algorithm used
 - Modifying the FBL to jump to a new address, without changing the hash of the FBL.





Xbox Security

- What is needed for Xbox security
 - Need to ensure Xbox integrity
 - Correct BIOS
 - BIOS will only load intended OS
 - Correct OS
 - Will only load signed Applications (Games)
 - Correct Applications
 - Games must not open security holes





Xbox Security

- Xbox security was broken by people eating to run Linux on Xbox
- Security model is backwards
 - Each stage verifies the next
 - If the next stage is verified
 - It is executed
 - Each stage should verify all previous stages





Windows Media Player

- Windows Media Player and DRM
 - Displays files
 - Honors DRM restrictions encoded in formats
- Its just software application
 - It can be reverse engineered
 - And has been
 - Encryption keys, algorithms, and protocols have been extracted
 - New application can be constructed which does not honor DRM restrictions in content
- Server only



Windows Media Player

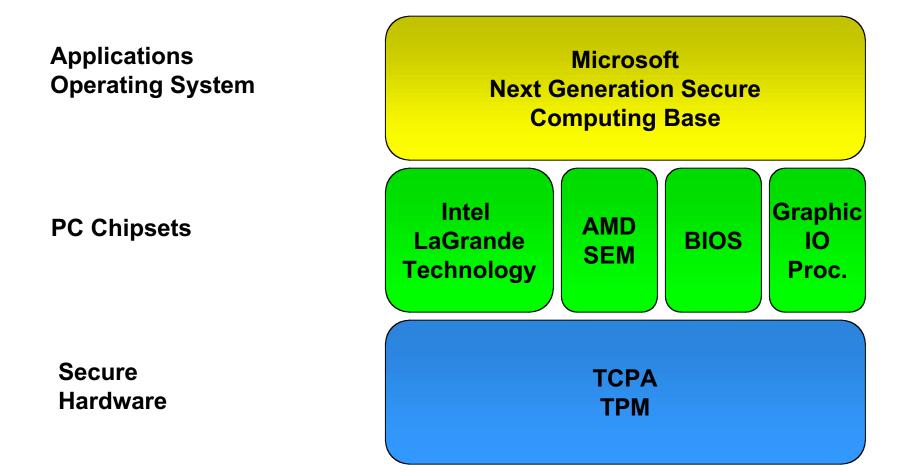
- What is required
 - Media Server needs to be sure that data is not going to imposter applications
 - Server needs to verify the application it is sending content to
 - Content needs to be bound to
 - Application
 - Application Environment
 - Software and hardware





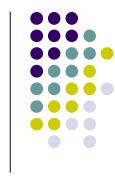
Security Initiatives

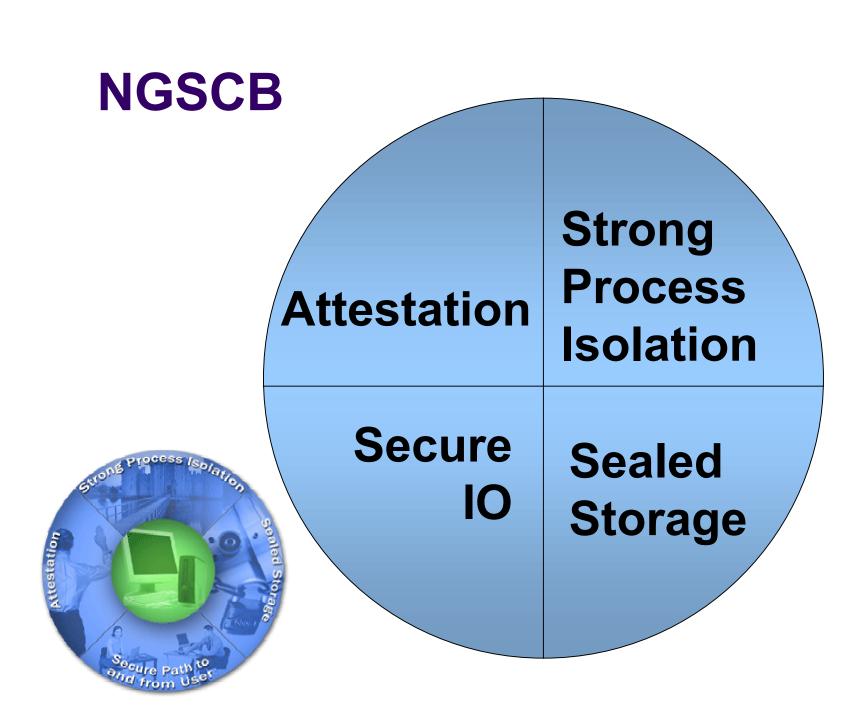




Next Generation Secure Computing Base (NGSCB)

- Formerly called Palladium
- Windows can not be made completely secure
 - Kernel is too big
 - Will always have bugs/security holes
 - Applications and services
 - Offer many potential holes to external attackers to get to kernel.
- Secure applications should run outside of Windows
 - Still have acsess to windows services





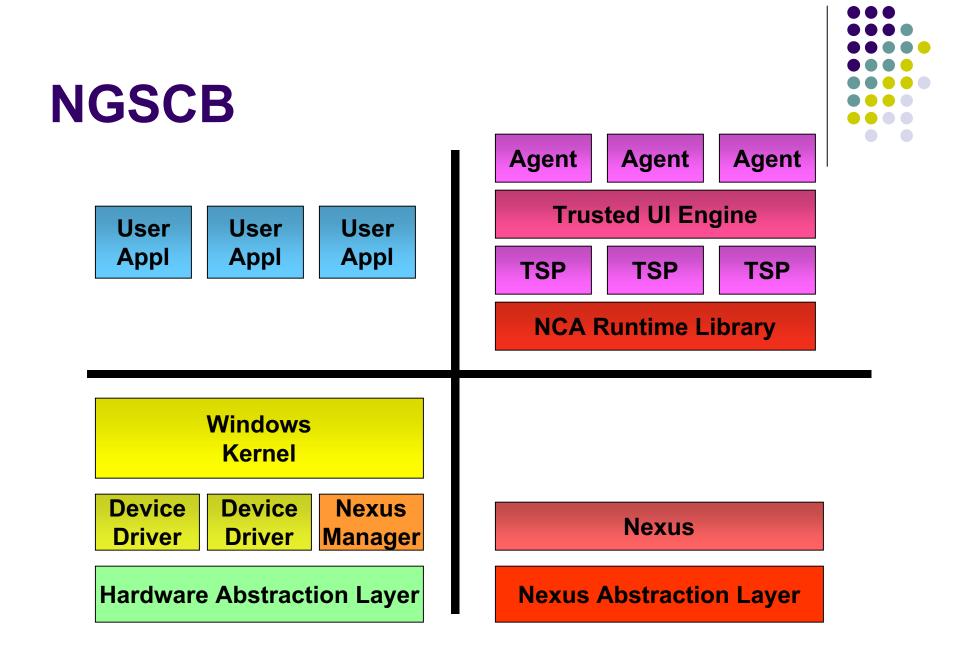


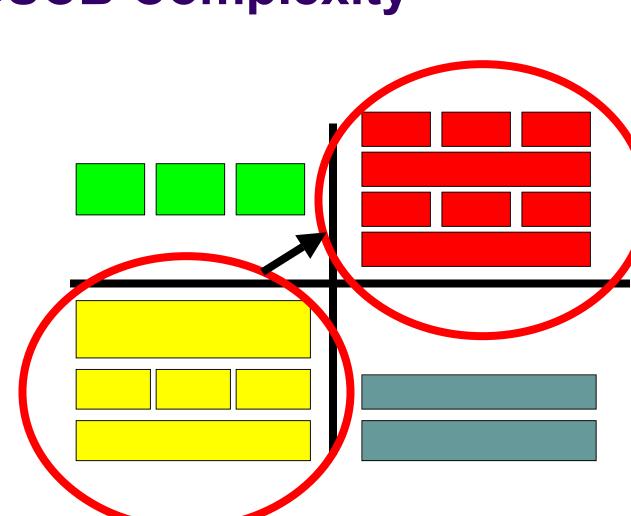
NGSCB

Attestation

- Ability to verify the operating environment
 - Remote verification
- Strong Process Isolation
 - Memory isolation (curtained memory)
- Sealed Storage
 - Data bound to operating environment
 - Application, OS, drivers, CPU, hardware, TPM,...
- Secure Path to IO
 - No keyboard sniffing
 - No framebuffer reading/writing







NGSCB Complexity



NGSCB

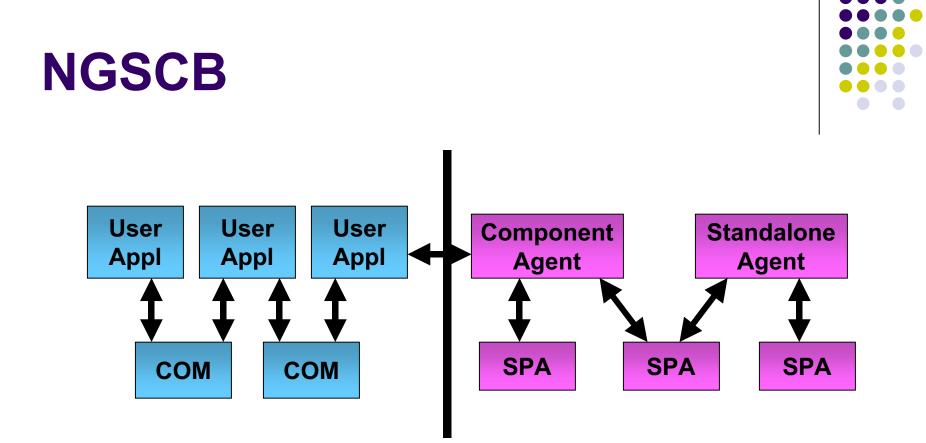


- Isolation of Nexus from Windows is done at hardware level
 - No windows bug will affect nexus applications
- Nexus
 - Only one nexus at a time
 - Not a complete Operating System
 - Implements
 - Process, thread, memory, and IO manager
 - Does not implement
 - File System, networking, device drivers, plugins, nor directX

Nexus Applications

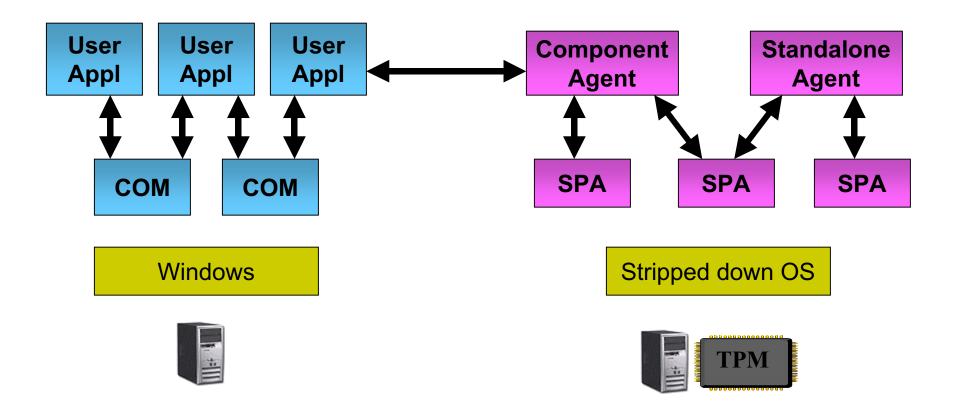
- Application Agents
 - Standalone program which runs in Nexus space
- Component Agents
 - Agents appear as external Com object or managed object
 - Windows proxy translate COM to IPC
- Service Provider Agents (SPA)
 - Agents provide services to other agents
 - IPC facility exists for agents to communicate







NGSCB (logical Equivalent)



Trusted UI Engine



- Nexus agents need to be able to securely put graphics on the display
 - Windows robust graphics systems are not available to nexus agents
 - Potential security hole
 - Nexus windows must not be hidden by windows applications
- Lightweight graphic system
 - XML based
 - Processed by graphics card

Attestation



- Attestation challenges must come from other computers
 - ????
 - Nexus and agents can not directly determine if they are running in secure mode
 - It is up to others to determine if they trust the nexus or the agents.

Manifests

- Each agent has a singed manifest
 - Extension of manifests to appear in Longhorn
- XML description of agent
 - Agent components and properties
 - Agent policy requests (non binding, controlled by owner)
 - System Requirements
 - Descriptive Properties
 - Secret migration
- For example
 - A flag indicates if the agent is debuggable



Debugging



- Debugging occurs in Windows
- Debugger communicates with agent
- A debuggable agent generates a different digest than a non debuggable agent
 - A remote entity can attest that the agent is not in debug mode when it interacts
- The nexus itself is debuggable
 - Special version of nexus



NGSCB Policies

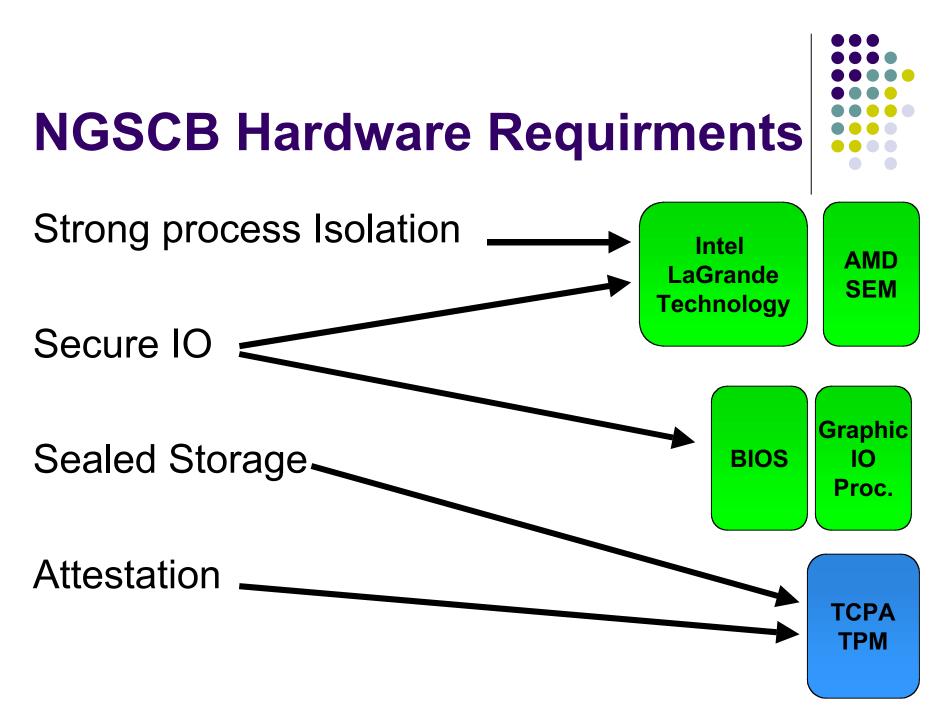


- Microsoft promises policies to control the operation and resources of nexus and agents
 - Running agents
 - Accessing secrets
 - Seal Storage
 - Networks and file systems
- Policies are a mixed blessing
 - Implies there is lots to manage

NGSCB Caveats

- Nexus does not mitigate bad/insecure software design
 - Onus is still on designer
 - Must carefully use windows services
- What protects nexus agents from each other
 - Nexus
 - Kept open(?) and simple



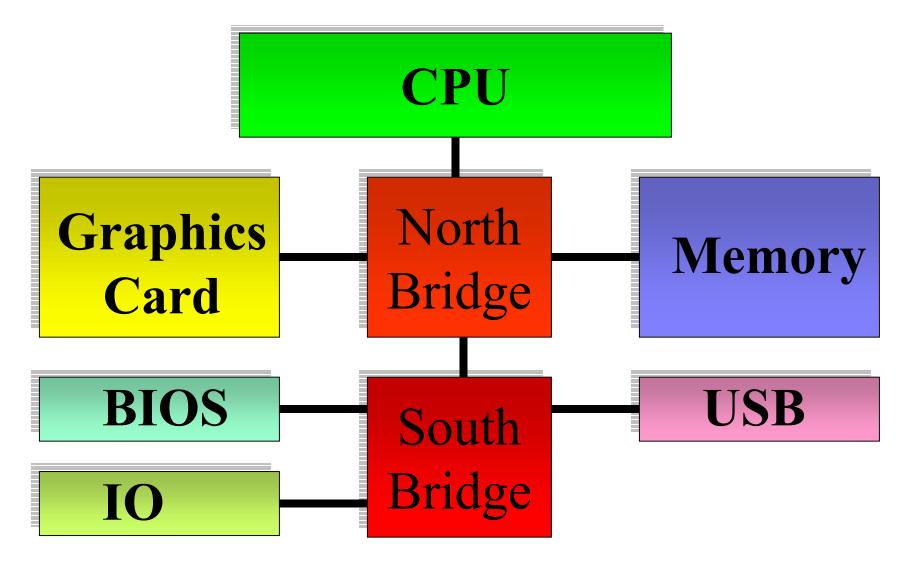


NGSCB Real Challenges

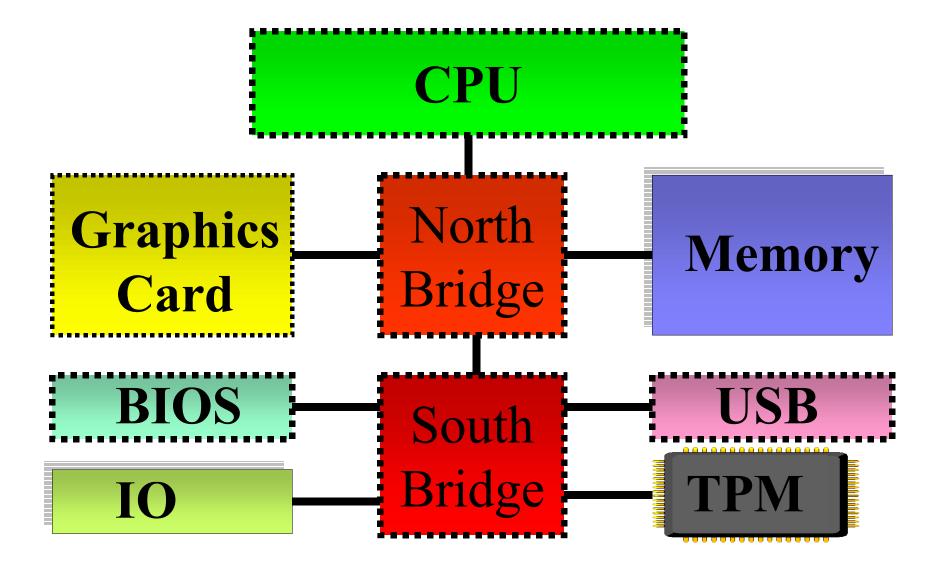
- Keep things from getting too complex
 - Putting IE in a nexus agent will not make it secure
- Manage Sealed Storage
 - Lots of potential to lose data with hardware/ software failures
 - How to backup data in sealed storage
 - Hardware management as part of data management



PC Architecture



Trusted Computing PC Architecture



Intel LaGrande Technologies

- Strong Processor Isolation
- Secure path to IO



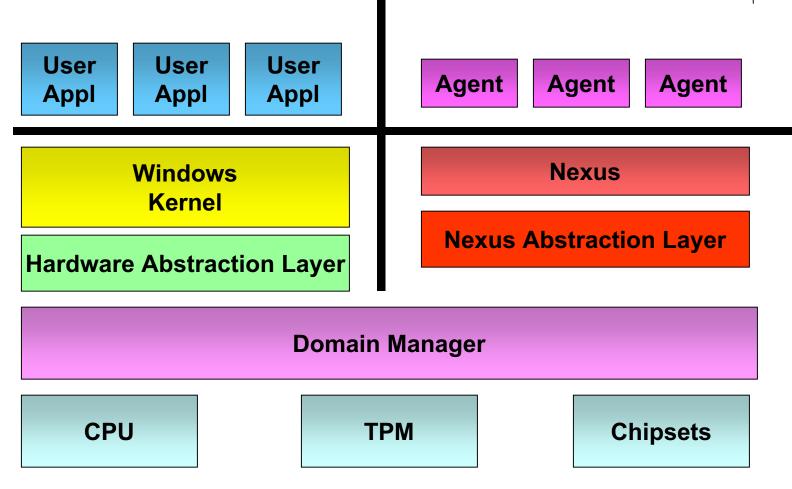
Secure Path

- Goal: to protect data within the PC
 - No keyboard sniffers
 - No reading/writing framebuffer
 - Input and output is secured to Agent
 - USB to nexus
 - Graphics card
 - Keyboard/pointer (for notebooks)





LaGrande Protection Model



Domain Manager

- Runs in processor
 - Software
 - Maintains process isolation
- Below Ring 0
 - Intel CPUs have rings
 - Kernel runs in Ring 0
 - Apps run in Ring 3
 - Ring "-1" ???



Memory Isolation



- Protecting memory is critical
 - Northbridge usually contains memory manager
 - Memory curtaining prohibits DMA from protected areas
- Devil in the details
 - Lots of things that need to be controlled
 - Memory during system resets
 - Memory during system sleeps
 - Initial trust ????

TCPA / TPM

- Trusted Computing Platform Alliance (TCPA)
 - http://www.trustedcomputing.org
- Trusted Computing Group
 - https://www.trustedcomputinggroup.org/home
 - Successor to TCPA
 - Same initiative
- Trusted Platform Module (TPM)
 - One component of TCPA





Anti-TCPA

- <u>http://www.againsttcpa.com/</u>
- The informational self-determination isn't existing anymore, it's not possible to save, copy, create, program, ..., the data like you want. This applies for privates as for companies

Against-

Don't let them take

- The free access to the IT/Software market is completely prevented for anyone except the big companies, the market as we know it today will get completely destroyed
- Restrictions in the usage of owned hardware would apply
- The liberty of opinion and the free speech on the internet would finally be eliminated
- The own rights while using IT-technologies are history.
- The national self-determination of the der particular countries would be fully in the hands of the USA
- Probably the world would break into two digital parts (Countries that express against TCPA)



SAY NO!

- Trusted Platform Module
- Current version 1.2
- Shipping Projects
 - IBM Thinkpad Notebooks
 - Chipsets
 - Infineon, Atmel, National SemiConductor, IBM

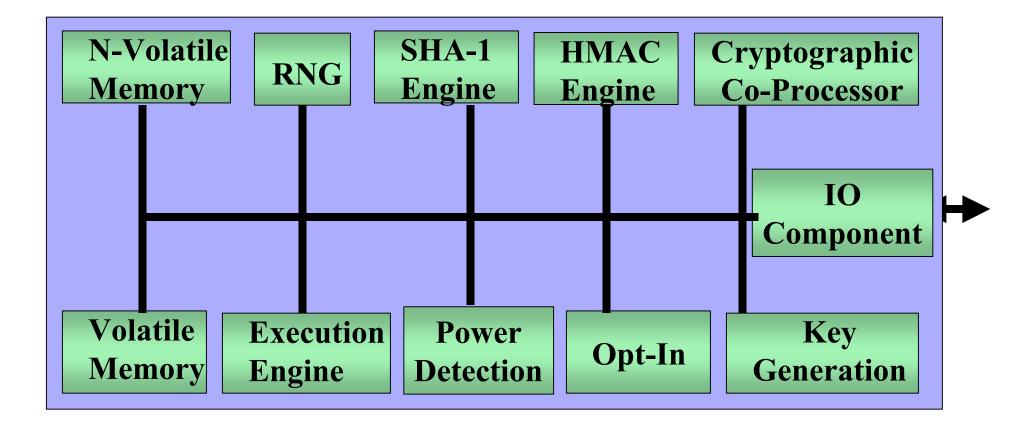
TPM Sealed Storage

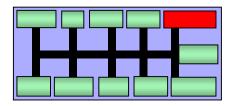




- Keys never leave TPM
- Data can only be unsealed
 - When system in is specified state
 - Authorized command

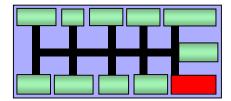
TPM Architecture





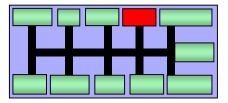
- Cryptographic Processor
 - RSA Engine (encryption and digital signatures)
 - PKCS #1
 - Key Sizes : 512, 768, 1024, 2048
 - Public exponent e: 2¹⁶+1
 - Symmetric Encryption Engine
 - Vernam one-time pad with XOR
 - The engine is for internal use, and not general message encryption.

Note : These are the required characteristics of the TPM, actual implementations may use a superset

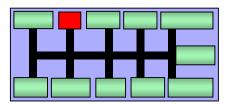


• Keys

- TPM can generate, store, and protect symmetric keys
- Key Generation
 - RSA Asymmetric
 - In accordance with IEEE P1363 standard

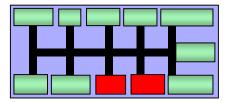


- HMAC Engine
 - Uses:
 - Proof of knowledge of authorized data
 - Command integrity
 - IETF RFC 2104 using SHA-1
 - 20 byte key, 64 byte blocks

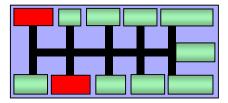


• RNG

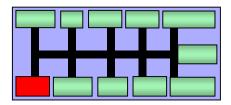
- Used for
 - Random values for nonces.
 - Key generation
 - Randomness in signatures
- May be RNG or PRNG
- SHA –1 Engine
 - As defined by FIPS 180-1, 20 byte output.



- Power Detection
 - TPM is required to be informed of all power state changes
- Opt-In
 - Allows the TPM module to be
 - Turned on/off
 - Enabled/disabled
 - Activated/deactivated



- Execution Engine
 - Processes TPM commands
- Non-Volatile Memory
 - Persistent identity
 - Data Integrity Registers (DIR)
 - Deprecated: Legacy from TPM 1.1
 - Still required

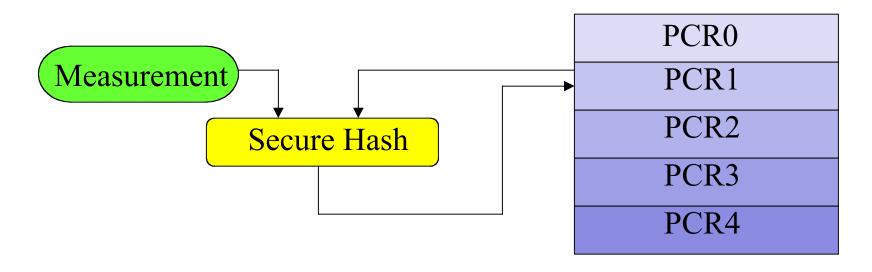


- Platform Configuration Registers (PCR)
 - Volatile storage
 - 16 or more (32 bit index, 230 and above are reserved)

160 bit Hash Sta	atus Locality
------------------	---------------

• Platform Configuration Registers (PCR)

• Can hold an unlimited number of measurements



 $PCR1_n = Hash(PCR_{n-1} || Measurement_n)$

Endorsement Key (EK)

- Each TPM contains a 2048 bit RSA key pair (PUKEK, PIVEK)
 - These keys are created before delivery to end user
 - When the EK is created, a credential is also created attesting to the validity of the EK
 - Any attempted to set/generate new keys must fail
 - PRIVEK never leaves the TPM

Endorsement Key (EK)

- Used ONLY for
 - TPM ownership insertion
 - AIK creation/verification
- EK is bound to Platform
- EK acts as Root of Trust for Reporting (RTR)

Attestation Identity Keys (AIK)

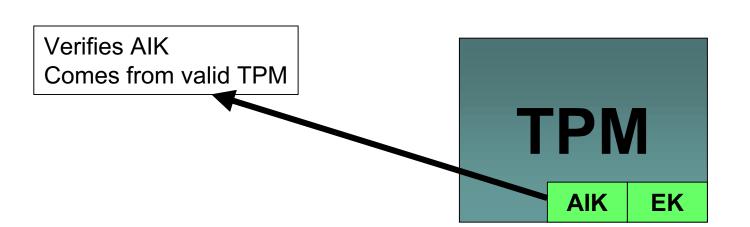
- Alias to the Endorsement Key (EK)
 - TPM Owner can create anytime
- A TPM can have multiple identities.
 - Increase privacy (different operation can be done with different identities)

AIK

AIK is used ONLY for

- Signing PCR data
 - If used for signing other data, it might be possible to create fake PCR signatures.
 - Must only sign data generated by TPM
- There is no migration of AIK from one computer to another
- AlK's may be stored externally to the TPM

Attestation



- EK is permanent
- AIK may be temporary
- Zero Knowledge Proof
 - Used to prove knowledge of EK without disclosing EK

TPM Ownership

- TPM must have no owner when shipped
- TPM ownership can always be reset via a physical presence
 - Old secrets are discarded
- TPM ownership can be asserted by physical presence
 - No secrets are exposed
- Taking ownership
 - A secret is encrypted with PUBEK
 - Ownership is proved by showing knowledge of shared secret

TPM Ownership

- TPM ownership is not equivalent "super-user"
 - Does not give access to all operations
 - Each authorization must be provided for entity or operation that has protection

Roots of Trust for Storage

- When ownership is established
 - New Storage Root Key (SRK)
 - New TPMProof value

Authorization

- Authorization data is 160 bit shared secret plus high entropy random number
 - Hashed together
- Dictionary attack
 - stateless
 - Response degradation/lockout ?

TPM Startup

- Options
 - Clear: TPM is to start with default values (specified by TPM Owner)
 - State: TPM is to recover a saved state and continue operation from this saved state
 - Deactivate: The TPM should not allow any further commands to be processed. Can only be reset by TPM_Init command.

TPM States

• Three operational state bits

• Enabled, Active, Owned

E	A	O		
\checkmark	\checkmark	\checkmark	S1 : Fully Operational State	
	<	\checkmark	S2 : ownership is and can be set	
\checkmark		\checkmark	S3	
		\checkmark	S4 : ownership can not be set	
\checkmark	\checkmark		S5 ; local or remote ownership possible	
	\checkmark		S6 : ownership can be set	
\checkmark			S7	
			S8 : All functions are off	

TPM States

- Enabled/Disabled
 - Disabled: TPM can not execute commands which use TPM resources
 - Any command needing a key is prohibited
 - SHA is still available (no keys)
 - Ownership can be disabled
 - Persistent Flag
 - Immediate

TPM States

- Deactive/Active
 - Similar to Enable except allows TPM_TakeOwnership command
 - Persistent
 - Deactivating does not take effect until reinitialization (reboot)

- Physical Presence
 - An indication to the TPM of a direct operation with a person/operator
 - Not maskable or setable via software.
 - Certain operations on TPM require physical presence
 - Clearing existing owner
 - Temorarily deactivating/disabling TPM

Authorization Protocols

- Object Independent Authorization Protocol (OIAP)
- Object Specific Authorization Protocol (OSAP)
- Delegate Specific Authorization Protocol (DSAP)

Authorization Protocols

- Based on shared secret
 - Gives access to operation
 - Does not give access to secrets
- Rolling nonce paradigm
 - Nonces from one command to the next

OIAP

TPM

TPM_OIAP

authHandle, authLastNonceEven

TPM_COMMAND, Arguments, nonceOdd, authHandle, HMAC(key,SHA-1(arguments), authHandle, authLastNonceEven,nonceOdd,.....),

TPM_COMMAND tag, Arguments, nonceEven, authHandle, HMAC(key,SHA-1(arguments), authHandle, nonceEven,nonceOdd,.....),

OSAP

TPM_OIAP, keyHandle, nonceOddOSAP

authHandle, authLastNonceEven, nonceOddOSAP

TPM

TPM_COMMAND, Arguments, nonceOdd, authHandle, HMAC(key,SHA-1(arguments), authHandle, authLastNonceEven,nonceOdd,.....),

TPM_COMMAND tag, Arguments, nonceEven, authHandle, HMAC(key,SHA-1(arguments), authHandle, nonceEven,nonceOdd,.....),

Maintenance

Things break

- There may be a need to migrate data from one TPM to another (eg replacing motherboard)
- Manufacturer or others must not be able to intercept data in migration.
- Only needs to work between boards of same model and manufacturer
- Requires owner and manufacturer authorization

Maintenance

- All maintenance features are optional
 - Specific mechanisms not defined
 - Security requirements defined

Counters

- TPM must support at least 4 counters
 Increment rate : Every 5 secs for 7 years
- Internal Base
 - Always moves forward, never reset

Transport Protection

- Sessions (set of commands)
 - Protection
 - Rolling nonces
 - MGF1 function
 - Shared secret, nonceOdd, nonceEven
 - Logging
 - Command, command parameters, and tick count

Audit Commands

- Ability for TPM owner to determine that certain operations have been executed
- Two parts
 - Internal Digest
 - External Log
- Which functions are audited is set by the TPM owner.

TPM Hardware

- Hardware Connection is not standardized (vendor specific)
 - Low Pin Count (LPC) Bus
 - Low bandwidth/volume
- Implementations
 - Infineon

(http://www.infineon.jp/event_topics/events/sched ule/wireless2003/img/tpm.pdf)

Locality



- New feature in version 1.2
- Used to distinguish different classes of processes
 - 1: reserved
 - 2: trusted OS
 - 3: trusted initialization software
 - 4: special initialization hardware

Locality

• Why locality

- Different requirements for a TPM from nexus agents than from hardware (eg BIOS)
- The digest model
 - Only the final is stored
 - If a new card is added
 - You can not replace old value
 - You must reboot, and recalculate all
 - Okay process for hardware
 - For software
 - Applications come and go
 - Some PCRs can be reset

TPM Summary

- TPM is a security resource for the PC architecture.
 - By itself it is harmless
 - It is opt-in. No need to worry about loosing control of your PC

• TPM is a piece of the NGSCB architecture

Trusted Computing

- What is the role of Linux (or other OSes)?
 - Linux can also use TPM and trusted computing hardware on PC
 - Its unlikely Linux can interoperate with MS NGSCB

Trusted Computing

Constraining Or Opportunity?

TPM Ownership

- Who owns the TPM ?
 - Somebody has to
 - Most useful operations require ownership
- Choices
 - BIOS
 - Its the first entity that requires ownership exists
 - Can not keep ownership secret secure
 - Operating System
 - Can not keep ownership secret secure



TPM Ownership

- Choices (cont)
 - Nexus
 - Seems logical
 - User
 - Most trustworthy
 - Least reliable
- It really doesn't matter
 - Ownership doesn't grant access to secrets
 - Ownership only controls services
 - If the wrong entity gains ownership, they can only do DOS



Back to Original Problems

- Secure communications between servers
 - All authentication secrets stored in sealed storage
 - Only a security hole in application can reveal secrets
 - System is not dependent upon OS security
- Xbox
 - NGSCB/TPM is only partial solution
 - OS and application self attest (TPM allows)
 - Still need to prohibit certain apps
 - Stronger version of what is currently done



Back to Original Problems

- Windows Media Player
 - Server attests client
 - Server sends content to client (securely)
 - Encrypted with unique key for application/device
 - Keys stored in sealed storage
 - Bound to application/device
 - Some additional info stored with keys
 - Number of time played (to prevent copy/play/restore)



What if Microsoft is Lying

- Nexus code will be available for inspection
 - You can run your own nexus*
 - You can trust nexus
 - There may be unknown security holes
- You can run your own Nexus
 - Build a nexus under linux.....
- NGSCB is Opt-In
 - Subject to DOS attacks from Windows
 - Disable TPM





Is this Safe Technology



Bonus Material - TPM

- TPM Info
- TPM Commands

TPM Information

- Trusted Computing Group

 (www.trustedcomputinggroup.org)
 - TPM Main Part 1 Design Principles (version 1.2)

- Admin
 - TPM_Init,TPM_Startup, TPM_SaveState, TPM_SelfTestFull, TPM_ContinueSelfTest, TPM_GetTestResult
- Opt-In
 - TPM_SetOwnerInstall, TPM_SetOwnerDisable, TPM_PhysicalEnable, TPM_PhysicalDisable, TPM_PhysicalSetDeactivated, TPM_SetTempDeactivated, TPM_SetOperatorAuth,

- Ownership
 - TPM_TakeOwnership, TPM_OwnerClear, TPM_ForceClear, TPM_DisaleOwnerClear, TPM_DisableForceClear, TSC_PhysicalPresence, TSC_ResetEstablishmentBit,
- Admin
 - TPM_GetCapability, TPM_FieldUpgrade, TM_SetRedirection,

- Auditing
 - TPM_GetAuditDigest, TPM_GetAuditDigestSigned, TPM_SetOrdinalAuditStatus,
- Storage
 - TPM_Seal, TPM_Unseal, TPM_UnBind, TPM_CreateWrapKey, TPM_LoadKey, TPM_GetPubKey,

- Migration
 - TPM_CreateMigrationBlob, TPM_ConvertMigrationBlob, TPM_AuthorizeMigrationKey, TPM_CMK_CreateKey, TPM_CMK_CreateTicket, TPM_CMK_CreateBlob, TPM_CMK_SetRestrictions,
- Maintenance Commands (Optional)
 - TPM_CreateMaintenanceArchive, TPM_LoadMaintenanceArchive, TPM_KillMaintenanceFeature, TPM_LoadManuMaintPub, TPM_ReadManuMaintPub,

- Cryptographics Functions
 - TPM_SHA1Start, TPM_SHA1Update, TPM_SHA1Complete, TPM_SHA1CompleteExtend, TPM_Sign, TPM_GetRandom, TPM_StirRandom, TPM_CertifyKey, TPM_CertifyKey2,
- Credential Handling
 - TPM_CrateEndorsementKeyPair, TPM_CreateRevocableEK, TPM_RevokeTrust, TPM_ReadPubek, TPM_DisablePubekRead, TPM_OwnerReadInternalPub,

- Identity Commands
 - TPM_MakeIdentity, TPM_ActivateIdentity,
- Integrity Commands
 - TPM_Extend, TPM_PCRRead, TPM_Quote, TPM_PCR_Reset,
- Authorization Commands
 - TPM_ChangeAuth, TPM_ChangeAuthOwner,
- Authorization Sessions
 - TPM_OIAP, TPM_OSAP, TPM_DSAP, TPM_SetOwnerPointer,
- Deleagtion
 - TPM_Delegate_Manage, TPM_CreatekeyDelegation, TPM_Delegate_CreateOwnerDelegation, TPM_Delegate_LoadOwnerDelegation, TPM_Delegate_ReadTable, TPM_Delegate_UpdateVerification, TPM_Delegate_VerifyDelegation,

- NV Storage
 - TPM_NV_DefineSpace, TPM_NV_WriteValue, TPM_NV_WriteValueAuth, TPM_NV_ReadValue, TPM_NV_ReadValueAuth,
- Session Management
 - TPM_KeyControlOwner, TPM_SaveContext, TPM_LoadContext, TPM_FlushSpecific,
- Timing Ticks
 - TPM_SetTickType, TPM_GetTicks, TPM_TickStampBlob,
- Session
 - TPM_EstablishTransport, TPM_ExecuteTransport, TPM_ReleaseTransportSigned,
- Counters
 - TPM_CreateCounter, TPM_IncrementCounter, TPM_ReadCounter, TPM_ReleaseCounter, TPM_ReleaseCounterOwner

- DAA Commands
 - TPM_DAA_Join, TPM_DAA_Sign,
- GPIO Commands
 - TPM_GPIO_AuthChannel, TPM_GPIO_ReadWrite,
- Deprecated commands
 - Not listed......