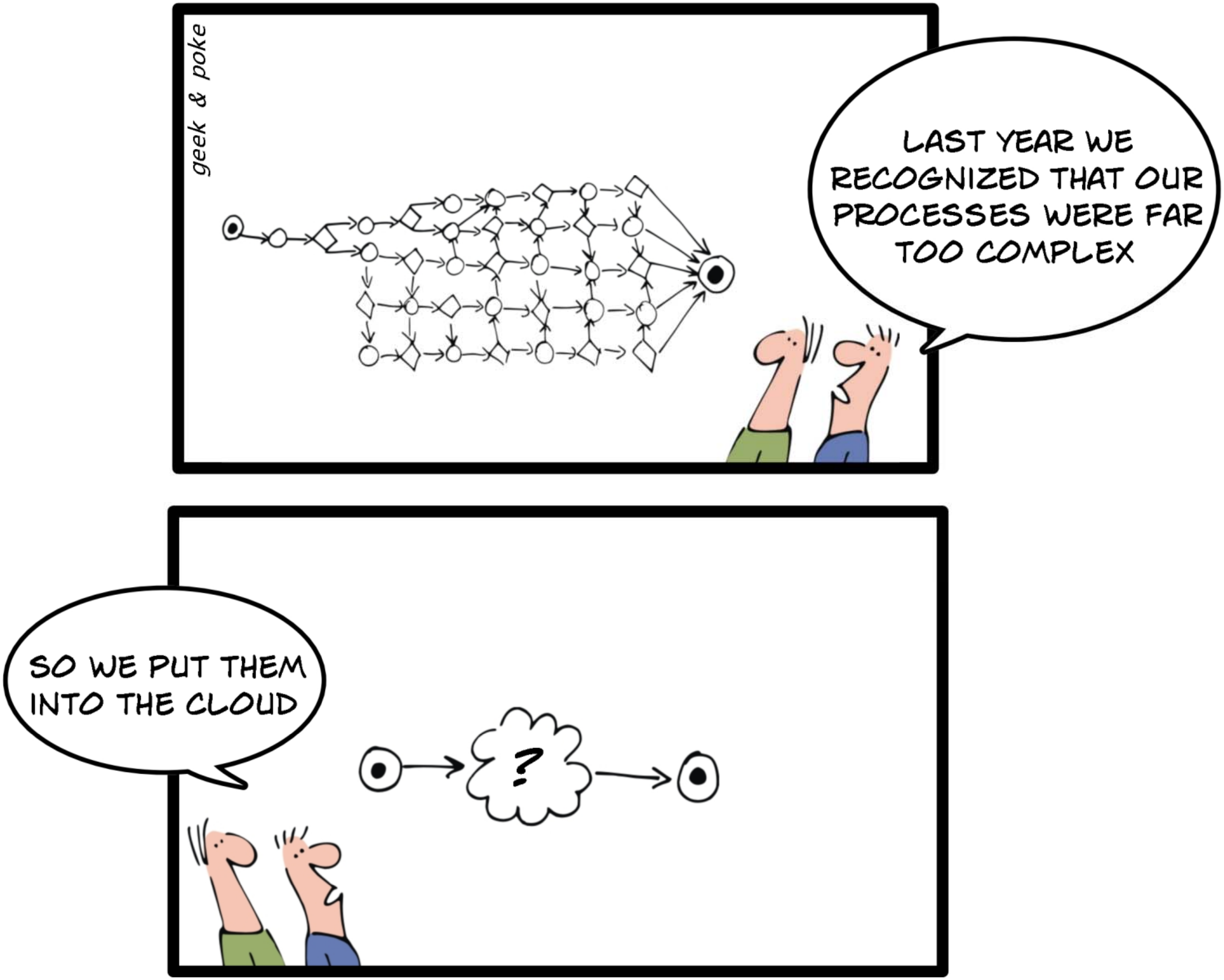


LET THE CLOUDS MAKE YOUR LIFE EASIER

Enterprise Encryption 101

SHARE 116

Session 8396

Phil Smith III

Voltage Security, Inc.

March 2011

Agenda

- Why we're here
- Encryption basics: terminology and types
- What is "enterprise encryption"?
- Why encryption is difficult and scary
- The five Ws of encryption
- Encryption key management: the "other" gotcha
- A realistic approach to enterprise encryption
- Example: Voltage SecureData

Enterprise Encryption In Sixty Minutes

Why We're Here

- Encryption is on many folks' minds these days
  - C*x*Os, CISOs are saying "Gotta encrypt stuff ***now!***"
- Breaches are in the news
  - Heartland, TJX, RBS WorldPay, et al.
- Many sites have implemented several point solutions
  - Different platforms, different problems…not interoperable!
- DLP (data leakage prevention) is not foolproof
  - If it's leaked but encrypted, you care a ***whole lot*** less!
- The h4xx0rs are out there…
  - …and they're getting smarter and more creative
- Internal breaches are increasing
  - Gartner et al. agree: 70%++ breaches are internal

Encryption Basics

Encryption Basics

- *Encryption* means
  - using an algorithm (*cipher*)
  - plus a secret value (*key*)
  - to transform data (*plaintext*)
  - into another format (*ciphertext*)
  - so it is no longer readable without *decryption*
- In other words:
  - **Make important data useless to anyone who isn't authorized to read it!**
- **Note**: Encryption tends to talk in terms of "messages"
  - Stored data may not go anywhere, but same principles apply

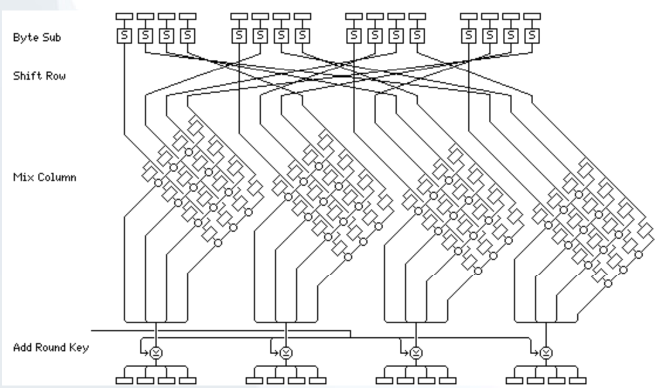
```
THE MISSILE LAUNCH CODE
IS XYZZY123plover
```

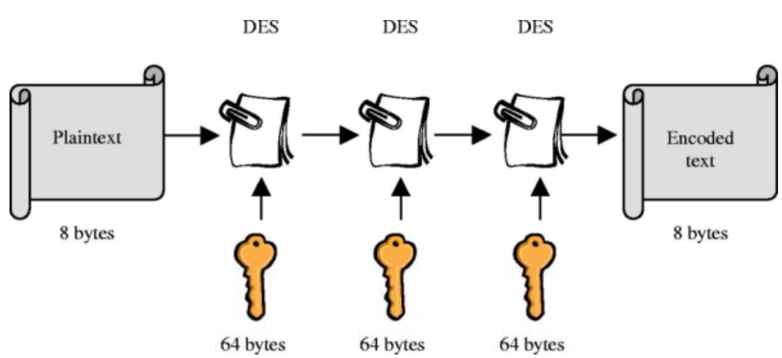
→

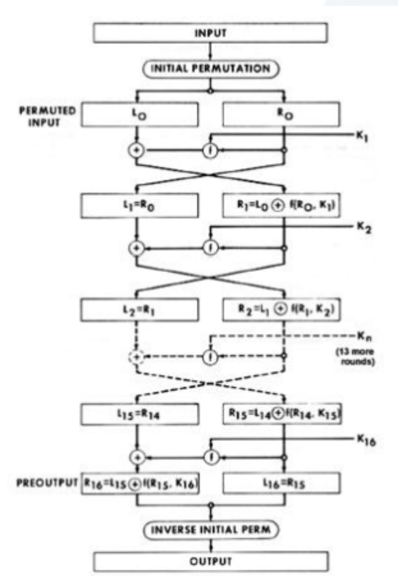
```
MV*U24AT2HaIKUewzqWPzvL
XaT9UGM!\zj(`iwPO…
```

Encryption Types: Symmetric

- Symmetric encryption means same key is used to encrypt and decrypt
  - Means both parties need access to the same keys
- Many varieties (algorithms):
  - DES, TDES, AES, Twofish, RC4, CAST5, IDEA, Blowfish…
- Can be strong and also fairly high-performance
  - “Strength” determined by key length in bits as well as algorithmic integrity

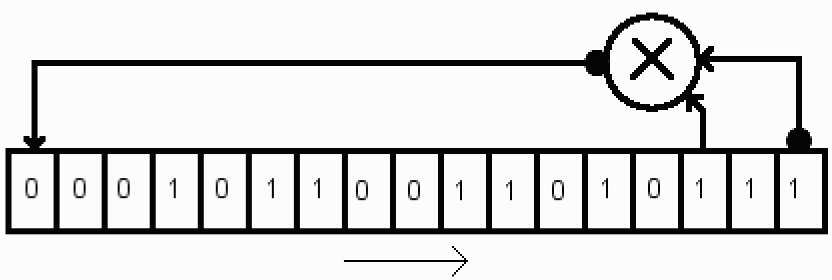




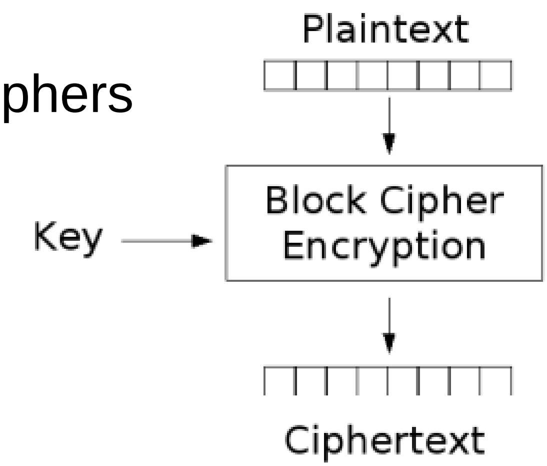


Symmetric Encryption: Stream and Block

- Symmetric encryption comes in two flavors:
  - *Stream* ciphers transform the key as they progress, processing one chunk (bit, byte, whatever) at a time
  - *Block* ciphers use fixed keys every block (blocksize=keysize)
- Difference matters little in practice
  - Stream generally faster, but requires more key complexity
  - Many block ciphers have modes that effectively operate like stream ciphers
  - Most data protection products use block ciphers



Stream Cipher



Asymmetric *aka* Public Key Encryption

- Asymmetric encryption means what it sounds like:
  - Different keys needed to encrypt and decrypt
  - Each entity has two keys: *public* and *private*
  - Invented in 1970s (Diffie-Hellman, RSA, UK government)
- Makes key distribution much easier:
  - I can publish my public key safely
  - You encrypt using public key, I decrypt using my private key
- Downside is performance
  - Symmetric algorithms are typically ***much*** faster—public key often too expensive for application data protection
  - Requires significant data layout/application changes

Asymmetric Encryption Uses

- Some use cases are ideal for public key encryption
  - Hassle-free (public) key exchange makes some things easy
  - A key is a key, so either (private/public) usable for encryption ***or*** decryption, provided “other” used for opposite function
- Better yet, encrypt twice: my private, your public
  - You and I can email each other our public keys
  - I encrypt with my private, your public
  - You decrypt with your private, my public
- You now know the data was encrypted ***by me***, I know ***only you*** could decrypt it
  - Provided neither of us has exposed our private keys!

Hybrids: Key "Wrapping"

- Because asymmetric encryption is expensive, hybrid solutions are attractive:
  - Sender generates random symmetric key
  - Encrypts actual data ("payload") using that symmetric key
  - Encrypts symmetric key using target's public key
  - Sends encrypted symmetric key with data
- To decrypt:
  - Key decrypted using (expensive) asymmetric (private key)
  - Payload decrypted using cheaper symmetric algorithm

Cryptographic Hashes and Digests

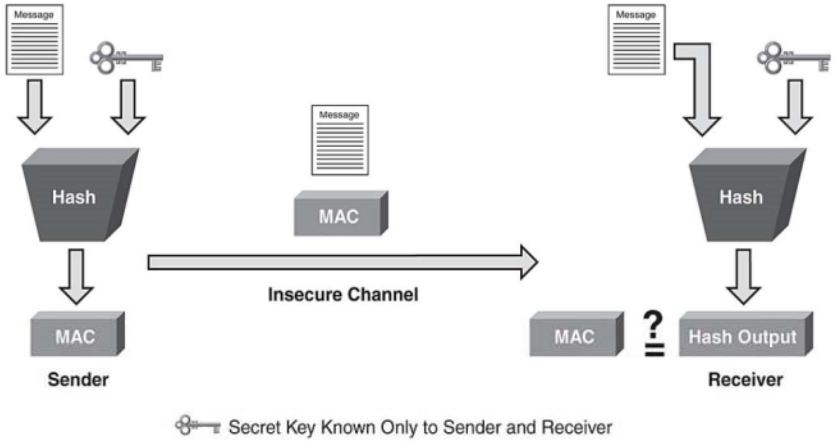
- Related to encryption: *cryptographic hashes* aka *digests*
  - Functions that convert variable-length input to fixed-length output
  - Any change to original data changes the hash
  - Used in digital signatures, as checksums, etc.
- Good hashes (SHA-1/2/3, MD4/5) have these properties:
  - Easy to compute for given data
  - Infeasible to reconstruct data from hash
  - Infeasible to modify data without changing hash
  - Collisions (same hash from different data) very rare
- A good way to represent data without leakage risk
  - Frequently used for things like verifying downloads

Digital Signatures

- *Digital signatures* are also related to cryptography
  - Generated from the data using public/private-like key pairs
  - Result is a hash-like blob
- Signatures prove data authenticity and integrity
  - **Authenticity**: Data is from who it says it's from
  - **Integrity**: Data has not been tampered with (since signing)
- Implements important concept: *non-repudiation*
  - Means sender cannot (reasonably) say "I didn't sign that"
- Frequently used for things like secure email
  - Avoids problems due to forged mail

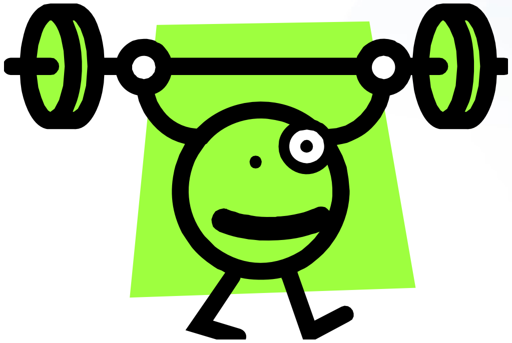
Message Authentication Codes (MACs)

- A **MAC** (**M**essage **A**uthentication **C**ode) is a keyed hash
  - Created using a hash function plus a secret key
  - Verify both data integrity and authenticity
- Different from digital signatures: same secret key used by creator/reader
  - Thus more like symmetric encryption, where digital signatures are more like public key encryption
- Generally faster to generate than digital signatures
  - MAC sent along with data
  - Receiver re-generates MAC against data, confirms match
  - Useful for verifying transactions



A Few Words About "Encryption Strength"

- **Encryption strength** refers to the likelihood that an attacker can "break" encrypted data
  - Typically tied to bit length of encryption key
  - Exponential: 128-bit key is 2^{64} times as strong as 64-bit
  - See "*Understanding Cryptographic Key Strength*" on youtube.com/user/VoltageOne for a good discussion/illustration
- The encryption community is collaborative
  - Research, algorithms are all published and peer-reviewed
  - Cryptographers look for weaknesses in their own and each others' work



More About “Encryption Strength”

- Cryptographers “cheat” in favor of ***attacker*** when analyzing
  - Make assumptions like “attacker has multiple known examples of encrypted data and matching plaintext”
  - Also assume they’ll know plaintext when they find it, and that the encryption algorithm is known
- “Weaknesses” reported are often largely theoretical—only NSA could really exploit
  - Huge amounts of time, brute-force computing power required

More About "Encryption Strength"

- This "cheating" ensures encryption strength is real*
  - This approach increases security for all
  - By the time an algorithm is accepted as a standard and implemented in products, confidence is high
  - Even if a weakness is later discovered, it's likely largely theoretical/impractical for most to exploit
- Makes it easy to spot the charlatans
  - Companies whose proprietary algorithms are ***not*** peer-reviewed
  - Also look for claims like "unbreakable encryption", or focus on key length rather than standards-based cryptography

*** Well, as real as the smartest minds in the business can make it!**

Encryption Algorithm Examples

- DES: Data Encryption Standard
  - Selected as standard by US government in 1976
  - Block cipher, uses 56-bit keys
  - Considered insecure: as of 1999, “breakable” in < 24 hours
- TDES: Triple DES
  - What it sounds like: DES applied three times
  - Uses two or three different keys
  - Thus at least 2^{112}-bit key strength (168-bit with three keys)
  - Considered secure, though relatively slow

More Encryption Algorithm Examples

- AES: Advanced Encryption Standard
  - Adopted as US standard in 2001
  - 128-, 192-, or 256-bit keys
  - Relatively fast
- Blowfish, Twofish, Serpent…
  - Similar to AES in strength
  - Mostly a bit slower (with exceptions)
  - Algorithms are public domain (as is AES)
- Dozens (hundreds!) more exist, of course
  - Given AES's ubiquity and proven strength, generally no reason to use anything else

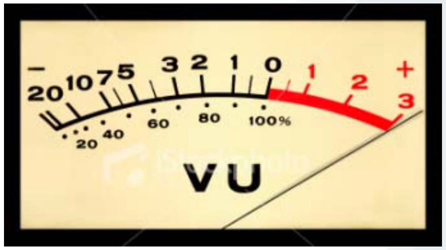
System z Encryption Facilities

Integrated Cryptographic Services Facility

- Encryption can be done in software routines, in software using specialized instructions, or in hardware
  - The U.S. considers encryption a “munition”, thus places restrictions on its export
  - Thus some hardware facilities not available in some countries
- Integrated Cryptographic Services Facility (ICSF)
  - z/OS Started Task providing crypto interfaces for applications
  - Invoked using well-documented API
  - Requires hardware facilities for some functions
- Active area for IBM development
  - New ICSF levels often appear between z/OS releases

Cryptography and Hardware

- Cryptographic algorithms tend to be CPU-intensive
  - Easy to peg CPU when encrypting via software
  - Optimized hardware is thus appealing



- Plaintext encryption keys in memory are worrisome
  - Auditors are paid to worry about this stuff
  - Even though ***we*** know z hardware protection is solid, Evil Sysprog could conceivably troll through storage
- These are different problems, with different solutions

Problem: CPU-Intensiveness

- Most crypto uses one of the common algorithms
  - DES, TDES, AES, RSA, SHA-1…
  - Means “90-10” rule applies to optimization
- System z offers CP Assist for Cryptographic Functions
  - CPACF is no-cost Feature Code (3863), enabled per CEC
  - Adds hardware instructions (KM/KMC, with subcodes)
  - Implements common crypto algorithms ***on the z chip***
  - Not quite “free”, but ***way*** faster than software implementations!
  - More capabilities on z10 than z9
  - zEnterprise adds even more

Problem: Plaintext Keys in Memory

- Plaintext key problem not unique to System z
  - Perhaps even more critical on less inherently secure systems
- Solution: Hardware Security Modules (HSMs)
  - Typically tamper-resistant, plug-in cards
  - Cryptographic operations sent off to HSM, results returned
  - Non-System z: nCipher (now Thales), Futurex, Atalla (HP) …
  - System z: Crypto Express2 & 3 (CEX2 & CEX3)
- CEX2/3 include two processors per card
  - Each supports up to 16 cryptographic domains
  - A single CEC can have up to eight CEX installed
  - CEX2-1P and CEX3-1P also exist: one processor per card (BC)

Problem: Plaintext Keys in Memory

- CEX stores Master Key (Key Encryption Key, or KEK)
  - Entered via ICSF or using Trusted Key Entry (TKE) Workstation feature
  - Operational keys are encrypted in CEX using KEK
  - ***Encrypted*** keys are stored on System z (in CKDS/PKDS)
- Operation:
  1. Application reads encrypted key, passes to ICSF
  2. ICSF passes request to CEX
  3. Key decrypted inside CEX, operation performed
  4. Crypto result returned to ICSF, thence to application
  5. ***Plaintext keys never reside in System z memory***
- This is called **Secure Key** operation

CPACF vs. Crypto Express

- ICSF exploits both CPACF and Crypto Express
  - Uses CPACF or CEX as appropriate (and if available)
  - **Note**: Linux for System z crypto drivers also exploit both
- CPACF and Crypto Express are often confused
  - "We have a CEX, so encryption should be fast"
  - Not necessarily: CEX is for ***security***, CPACF for ***performance***
- ***BUT…*** CEX can be used in performance-related ways:
  - To offload processing from expensive System z MIPS when throughput less critical (requires large data chunks to be a "win")
  - When configured as "accelerator" for SSL operations

Protected Key Operations

- Secure Key operations using CEX are “very” slow*
  - Throughput requirements often preclude use of Secure Key
- Latest ICSF and microcode add **Protected Key**
  - Hybrid solution, providing (most of) “Best of both worlds”
  - Exploits combination of CPACF and CEX (via ICSF)
- Stored keys in z/OS are still encrypted
  - CEX decrypts secure key, re-encrypts with “wrapping key”
  - Copies wrapping key to protected HSA memory
  - Wrapped key returned and used on CPACF calls
- “Most of the performance with most of the security”
  - But some auditors may not “buy” it, even though protected memory cannot be dumped, even with HSM diagnostics

*** FSVO “very” – certainly much slower than Clear Key operations via CPACF**

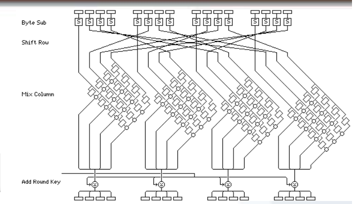
Implementing Encryption

What is "Enterprise Encryption"?

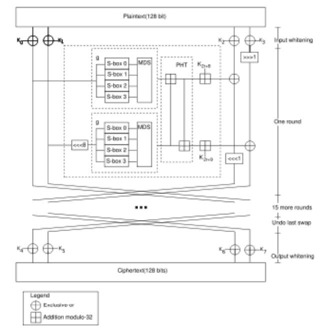
- A scalable, manageable data protection plan
  - Standards-based, provably secure
- Applies across multiple data sources (databases etc.)
  - Not just point solutions for specific data sources
- Cross-platform
  - Everyone has multiple platforms nowadays
- Includes key management

Encryption Is Difficult

- Lots of different technologies
  - Hardware-based, software-based, hardware-assisted
  - DES, TDES, AES, Blowfish, Twofish, CAST, PGP, GPG … !



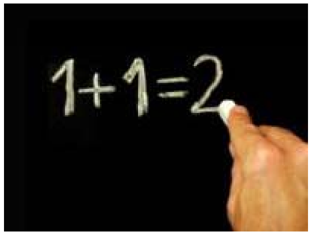
- Companies have ***lots*** of data in ***lots*** of places
  - Much of it probably of unknown value/use
  - The sheer volume is daunting



- Difficult to imagine how to get started
  - Easier to stick your head in the sand and hope it goes away
- For mainframe folks, it's even easier to (try to) ignore
  - System z OSes are traditionally more secure than distributed

Encryption Is Scary

- Most of us don't understand the technologies
  - Math classes were a looong time ago
- It changes constantly
  - We hear "DES has been broken, use AES"
  - What does that mean? Is DES useless? Is AES next to fall?



- Lots of snake-oil salesmen in encryption
  - www.meganet.com touts "unbreakable encryption"
- Easy to decide encryption is unapproachably complex
  - Like buying your first house, or doing your own taxes…
- Yes, if you get it wrong, you ***will*** lose data!
  - Another reason prompting avoidance behavior…

IRS
Department of the Treasury
Internal Revenue Service

The Five Ws of Encryption

- ***Why*** encrypt data?
- ***What*** should be encrypted?
- ***Where*** should it be encrypted?
- ***When*** should it be encrypted?
- ***Who*** should be able to encrypt/decrypt?
- ***How*** will you encrypt it?

Why Encrypt?

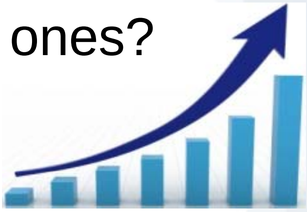
- Every company has data to protect
  - NPPI, PII, or just PI
  - Customer information
  - Internal account information
  - Intellectual property
  - Financial data
- Every company moves data around
  - Backup tapes
  - Networks
  - Laptops
  - Flash drives
  - Data for test systems

Why Encrypt?

- Different media have different issues
  - Very few backup tapes get lost…but it does happen
  - Networks get compromised fairly regularly
  - Laptops are lost or stolen ***every day***
  - Flash drives are disposable nowadays
- Different media types mean different levels of risk
  - Deliberate, targeted network breaches are obvious concern
  - Missing backups ***probably*** won't be read
  - Missing laptops ***probably*** won't be analyzed for PII
  - Found flash drives are probably given to the kids

Why Encrypt?

- Breaches happen!
  - 2009: 498; 2010: 662 (per Identity Theft Resource Center)
  - A healthy increase…and what about undetected/small ones?
  - Can you afford to bet your job/business?



- Data encryption is ***not*** a luxury
  - Claimed cost per compromised card is $154–$215!!! *****
  - Heartland breach: 130M cards; TJX: 94M cards
  - Do the math…

* Source: Ponemon Institute
$154 = negligent inside
$215 = malicious/criminal act

Why Encrypt?

- Data breach sources:
  - 73%: external
  - 18%: insiders
  - 39%: business partners
  - 30%: multiple parties

Source: Verizon Business, 2009 Data Breach Investigations Report

- But insider breaches far more expensive:
  - External attack costs averages $57,000
  - Insider attacks average $2,700,000!

Why Encrypt?

- Commonalities:
  - 66%: victim unaware data was on system
  - 75%: not discovered by victim
  - 83%: not “highly difficult”
  - 85%: opportunistic
  - 87%: avoidable through “reasonable” controls
- Causes:
  - 62%: attributed to a “significant error”
  - 59%: from hacking or intrusions
  - 31%: used malicious code
  - 22%: exploited vulnerability
  - 15%: physical attacks

The real card reader slot.

The capture device

The side cut out is not visible when on the ATM.

Why Encrypt?

- The law is catching up with the reality
  - PCI DSS (Payment Card Industry Data Security Standard)
  - Red Flag Identity Theft Rules (FACTA)
  - GLBA (Gramm-Leach-Bliley Act)
  - SB1386 (California)
  - Directive 95/46/EC (EU)
  - HIPAA
  - etc.

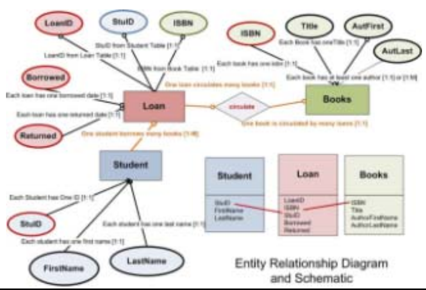
- PCI DSS not only requires data encryption, but also:
  - Restrict cardholder data access by business need-to-know
  - This is called **separation of duties**

What To Encrypt?

- Everything! (Well, maybe not…)
  - Performance, usability, cost are barriers
  - Partners likely use different encryption technology
  - Changing ***every*** application that uses the data is prohibitive
- No single answer
  - Laptops, flash drives: at least PII, probably all data
  - Backup tapes: all data
  - Whole-database encryption possible but not a good answer

What To Encrypt?

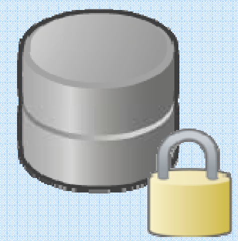
- Whole database encryption fails on several counts
  - Can impose unacceptable performance penalty
  - Prevents data compression, using more disk space etc.
  - Violates separation of duties requirements
  - Better to just encrypt the PII (whatever that is)!
- What about referential integrity and other data relationships?
  - Database 1 & database 2 both use SSN as key
  - If you encrypt them, encrypted SSNs better match!
  - Else must decrypt every access, and indexes useless



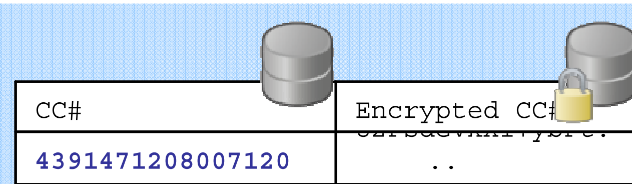
Voltage security

Application & Database Encryption Today:
Four Approaches

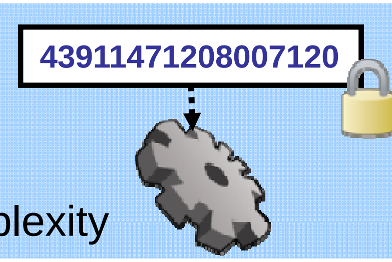
- Whole Database Encryption
  - Encrypt all data in DB—slows ***all*** applications
  - No granular access control, no separation of duties
  - No security of data ***within*** applications



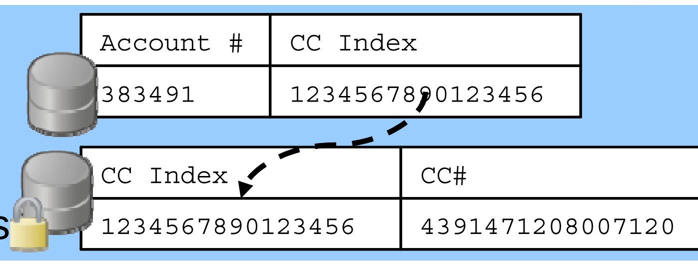
- Column Encryption Solutions
  - Encrypt data via DB API or stored procedure
  - Major DB type/version dependencies
  - No data masking support and poor separation of duties



- Traditional Application-level Encryption
  - Encrypt data itself via complex API
  - Requires DB schema/application format changes
  - High implementation cost plus key management complexity



- Lookaside Database (aka “Tokenization”)
  - CC# indexed, actual CC# in protected DB
  - Requires online lookup for ***every*** access
  - Requires major rearchitecting; scope issues



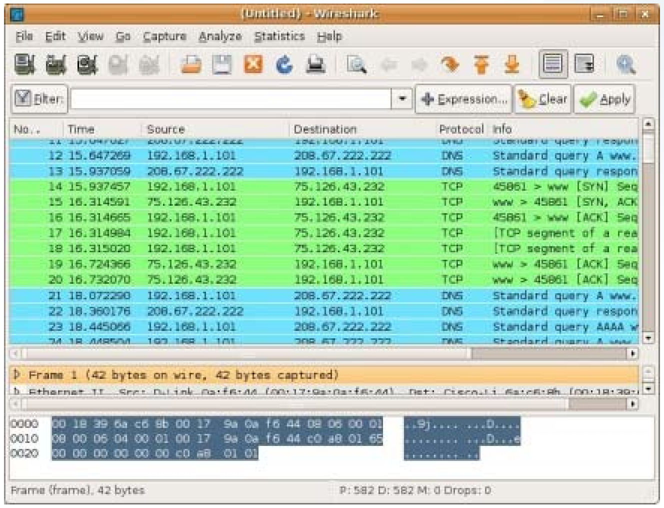
Voltage security

Where To Encrypt?

- Different question than "what":
  - Data **at rest** and **in motion**
- Data at rest
  - "Brown, round, and spinning" (DASD of all types)
  - On tape (backup or otherwise)
- Data in motion
  - Traversing the network

Where To Encrypt?

- Data in motion particularly troublesome
  - How do you know if it's been sniffed as it went by?
- Data at rest ***somewhat*** easier
  - Intrusion detection systems fairly effective (if installed and configured, and if someone actually checks the logs)
  - ESMs very effective on z/OS (if administered correctly)
- Different issues, thus different criteria!



When To Encrypt?

- Ideally, data is encrypted as it's captured
  - By the data entry application, or the card swipe machine

- In reality, it's often done far downstream
  - The handheld the flight attendant just used—is it encrypting?
  - Did last night's restaurant encrypt your credit card number?
  - If the data goes over a wireless network, is it WEP? WPA?
- "Doing it right" is harder: more touchpoints
  - Easier (if less effective) to say "Just encrypt at the database"
  - Avoids interoperability issues (ASCII/EBCDIC, partners)

Who Can Encrypt/Decrypt?

- Usual question is: who ***decrypts***?
  - Who should have the ability to decrypt PII?
- Should your staff have full access to all data?
  - Many unreported (or undetected) internal breaches occur
- What if someone leaves the company?
  - How do you ensure their access is ended?
- What if an encryption key is compromised?
  - Can you revoke it, so it's no longer useful?
- PCI DSS et al. ***require*** these kinds of controls
  - This is a big deal—***not*** trivial to implement

pci Security Standards Council™

How Will You Encrypt Data?

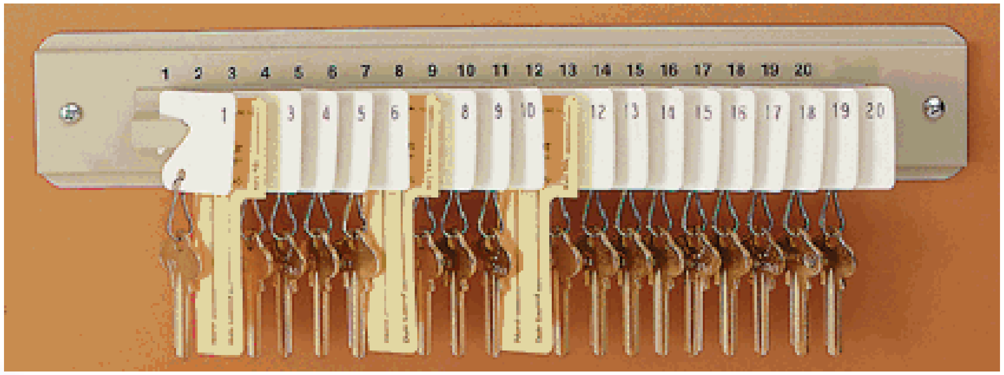
- Hardware? Software?
  - Many options exist for both
- Is a given solution cross-platform?
  - If not, you ***must*** decrypt/re-encrypt when data moves
- AES? TDES? Symmetric? Public/private key?
  - Many, ***many*** choices exist—too many!

How Will You Encrypt Data?

- Different issue: How do you get from here to there?
  - 100M++ data records—how to encrypt without outage?
  - “Customer database down next week while we encrypt”?!
- What about data format changes?
  - Encrypted data usually larger than original
  - Does not compress well (typically “not at all”)
  - Database schema, application fields expect current format
  - ***Can you change everything that touches the data?***
  - (Should you need to?)

Key Management

- "Encryption is easy, key management is hard"
  - Ultimately, encryption is just some function applied to data
  - To recover the original data, you need key management
- Three main key management functions:
  1. Give encryption keys to applications that must protect data
  2. Give decryption keys to users/applications that correctly authenticate according to some policy
  3. Allow administrators to specify that policy: who can get what keys, and how they authenticate



Voltage security

Key Management

- Key servers generate keys for each new request
  - Key server must back those up—an ongoing nightmare
  - What about keys generated between backups?
  - Maybe punch a card every time a key is generated…

- What about distributed applications?
  - How do you distribute keys among isolated networks?
- What about partners?
  - If you distribute encrypted data, how do they get the keys?
- “Allow open key server access” not a good answer
  - Suggest it, watch network security folks’ heads explode

Voltage security

Voltage security

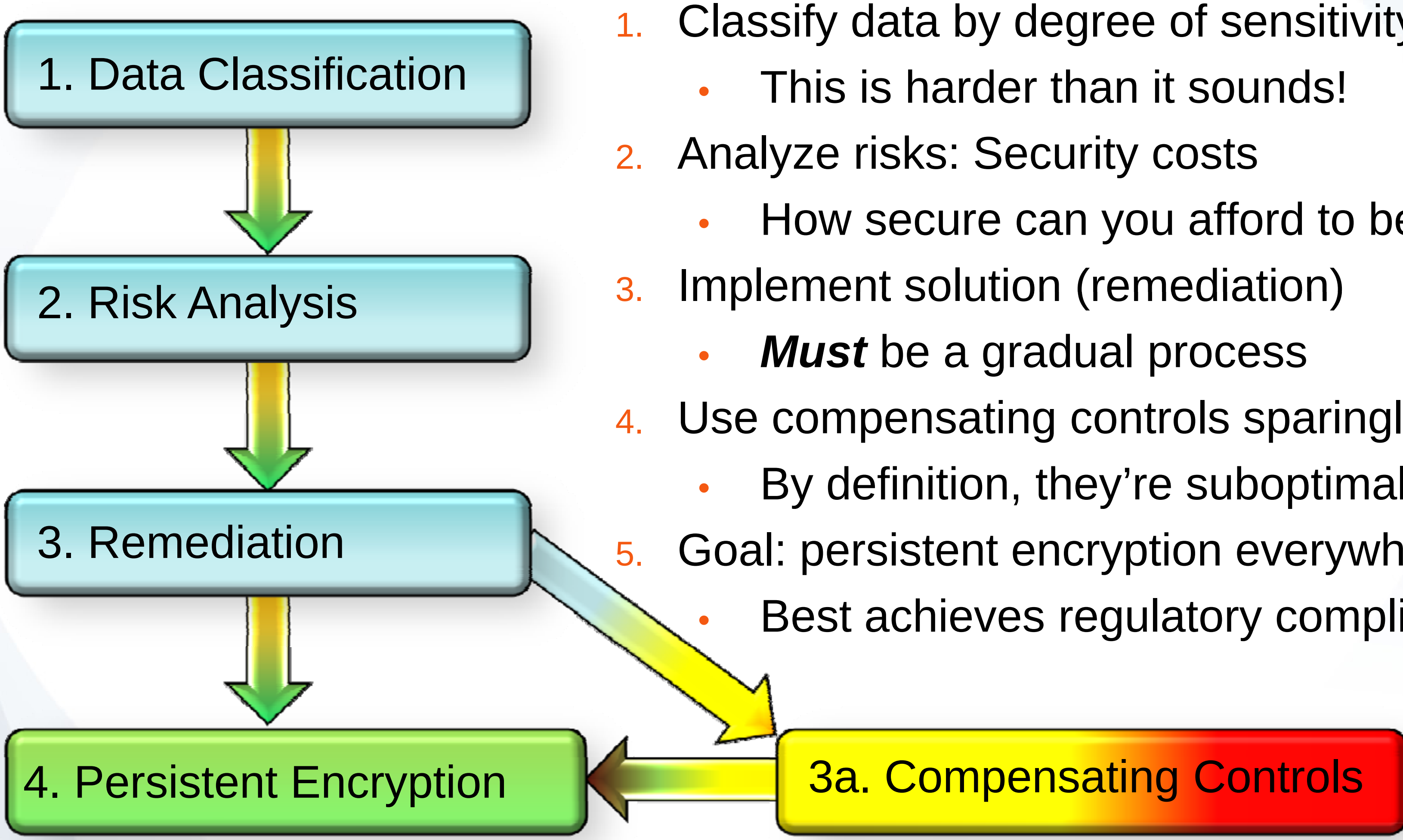
Getting There From Here: A Realistic Approach

A Realistic Approach: Take A Deep Breath

- Investigate encryption, now or soon
  - Better now than ***after*** breach
  - That light at the end of the tunnel ***is*** a train!
- Understand that choices have far-reaching effects
  - Data tends to live on for a very long time
- Expect to use multiple solutions
  - Backups, laptops, databases all have different requirements
  - “Right” answer differs
  - E.g., for backups, hardware-based solution; for customer database, column-based encryption

A Realistic Approach: High-Level Roadmap

1. Classify data by degree of sensitivity
   - This is harder than it sounds!
2. Analyze risks: Security costs
   - How secure can you afford to be?
3. Implement solution (remediation)
   - ***Must*** be a gradual process
4. Use compensating controls sparingly
   - By definition, they're suboptimal
5. Goal: persistent encryption everywhere
   - Best achieves regulatory compliance



Voltage security

A Realistic Approach: Key Steps

- **Key:** Involve stakeholders across the enterprise
  - "No database is an island": multiple groups use the data
  - Partners, widespread applications need access too…
- **Key:** Find a "starter" application
  - Generating test data from production is a good beachhead
  - If you "get it wrong", you haven't lost anything "real"
- **Key:** Designate data by sensitivity:

  Red: Regulated (legally required to be protected)

  Yellow: Intellectual property or other internal (unregulated)

  Green: Public

  - Each requires a different level of isolation/encryption

A Realistic Approach: Proof of Concept

- Encrypt a representative database
  - “Database” could be DB2, IMS, VSAM, flat file...
- Update application(s) that access it
  - You know what all your applications do, right? ☺
- Validate performance, usability, integrity
  - Encryption is ***not*** free: may see significant performance hit
- Demonstrate to other groups
  - Invite discussion, counter-suggestions
- Once (if!) project approved, request executive mandate
  - Otherwise, some groups may simply not participate

A Realistic Approach: Finishing the Job

- Doing ***all*** databases/applications takes time
  - Expect glitches
  - Perhaps most difficult: understanding data relationships
  - Table A and Table B seem unrelated, but aren't
- Lather, rinse, repeat…
  - Each database will have its own issues/surprises

Voltage security

Alternatives to Traditional Encryption

Tokenization

- Tokenization is another approach to data protection
  - Replaces values with randomly generated values
  - Index to real values stored in database
  - Detokenization thus requires database lookup
- Confusion abounds re tokenization vs. encryption
  - Some QSAs think tokenization is better because “there is no encryption key to be cracked”
  - Cryptographers see the database index itself as the key
  - Standards currently don’t help much here; hopefully will clarify

Format-Preserving Encryption

- **Format-Preserving Encryption** is another choice
  - Data encrypted with FPE has ***same format*** as input
  - Encrypted SSN still 9 digits; name has same number of characters; credit card number has same number of digits…

Name	SS#	Credit Card #	Street Address	Zip
James Potter	385-12-1199	5421 9852 8235 6981	1279 Farland Avenue	77901
Ryan Johnson	857-64-4190	5587 0806 2212 0139	111 Grant Street	75090
Carrie Young	761-58-6733	5348 9261 0695 2829	4513 Cambridge Court	72801
Brent Warner	604-41-6687	4929 4358 7398 4379	1984 Middleville Road	91706
Anna Berman	416-03-4226	4556 2525 1285 1830	2893 Hamilton Drive	21842

Name	SS#	Credit Card #	Street Address	Zip
James **Cqvzgk**	**161-82-1292**	5**184 2292 5001** 6981	**289 Ykzbpoi Clpppn**	77901
Ryan **Iounrfo**	**200-79-7127**	5**662 9566 7734** 0139	**406 Cmxto Osfalu**	75090
Carrie **Wntob**	**095-52-8683**	5**774 6343 6896** 2829	**1498 Zejojtbbx Pqkag**	72801
Brent **Gzhqlv**	**178-17-8353**	4**974 7815 8270** 4379	**8261 Saicbmeayqw Yotv**	91706
Anna **Tbluhm**	**525-25-2125**	4**288 0276 0003** 1830	**8412 Wbbhalhs Ueyzg**	21842

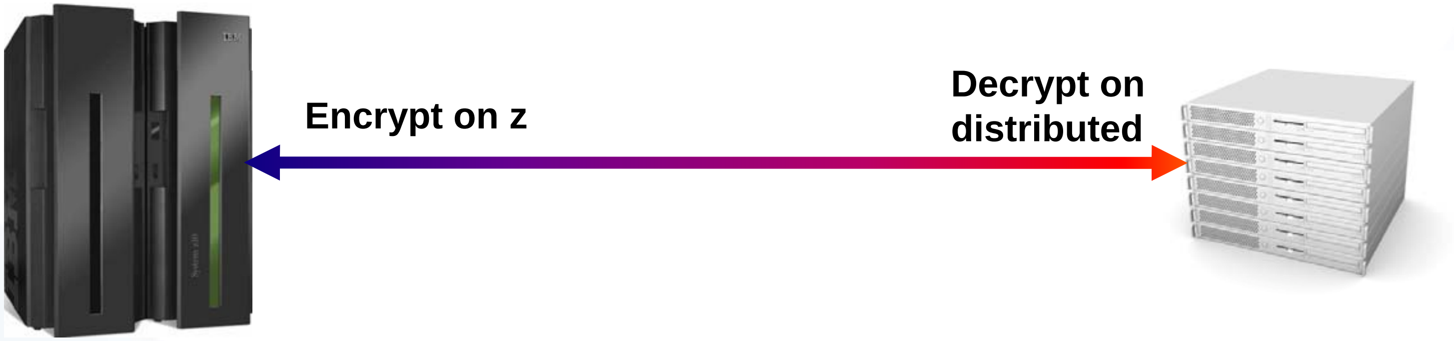
Format-Preserving Encryption

- Format-Preserving Encryption benefits:
  - Avoids database schema changes
  - Minimizes application changes
  - In fact, most applications can operate ***on the encrypted data:*** Fewer than 10% of applications need actual data
- FPE is a proposed mode of AES
  - Google “ffx mode” or look for “FFX” on http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html
  - Invented by Voltage Security, based on work at Stanford
  - Peer-reviewed, proven technology—not snake oil!

Voltage security

Cross-Platform Capable

- ASCII/EBCDIC issues go away
  - Data converted to UTF-8 before encryption/decryption
  - Stored in native format on host (ASCII or EBCDIC)
  - Possible because character sets are deterministic (FPE!)
  - Result: z/OS is a full partner in protected data management
- Encrypt/decrypt ***where the data is created/used***
  - Avoids plaintext data ever traversing the network



Data Masking

- Application testing needs realistic datasets
  - Fake sample datasets typically too small, not varied enough
- Best bet: Use production data…**but:**
  - Test systems may not be as secure
  - Testing staff should not have full access to PII!
- Answer: Use FPE to mask (anonymize) test data
  - With FPE, encrypted production data is perfectly usable for test
  - No extra steps required!

Voltage security

Voltage SecureData

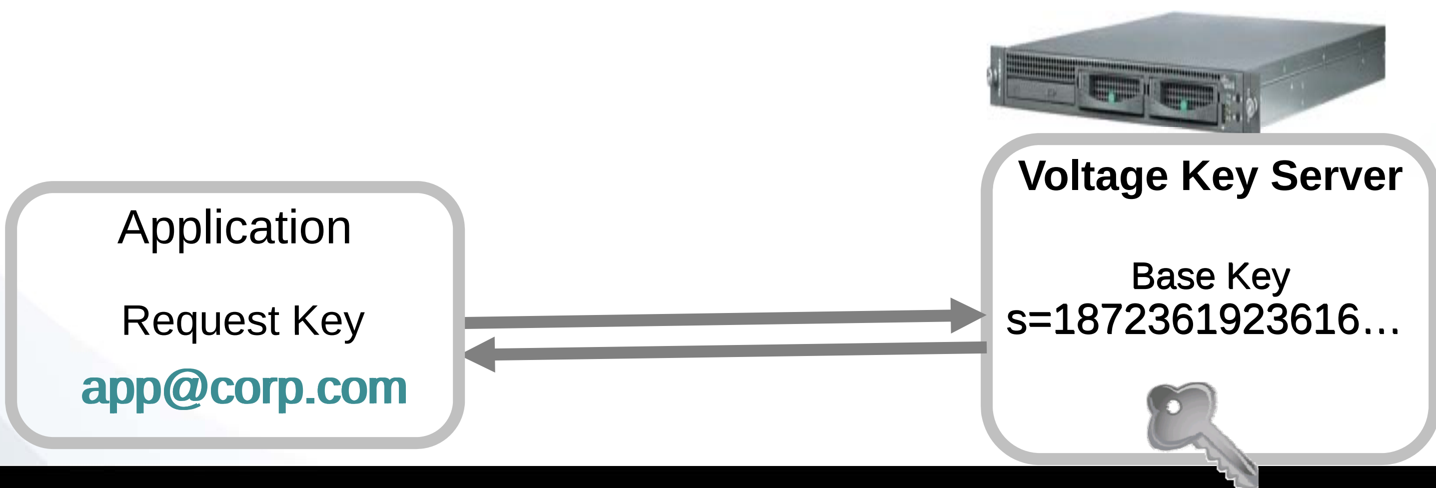
Voltage SecureData

- Voltage SecureData: Yet Another Encryption Product
  - With some key differences, of course!
- Available on z/OS, Windows, Linux, z/Linux, HP/UX, AIX
  - Built on platform-agnostic codebase (easy to port)
  - Can add platforms quickly as customers require them
- Complete suite of options:
  - Toolkit (APIs) for application integration
  - Bulk data encryption tools for scripting/data masking (z/FPE, CL)
  - SOA server for legacy/lightweight platforms
  - Tokenization supported via SOA for sites that require it

SecureData Toolkit | **z/FPE** | **SecureData CL** | **SecureData SOA**

Key Management

- Simplified key management eases most headaches
  - Keys are generated dynamically based on **identity**
  - Enables multiple key servers, serving ***same*** keys
  - Allows geographic/network isolation
  - Requires backup ***only*** when key server configuration changes
- Key request authentication allows separation of duties
  - Users/applications without access ***cannot*** get keys
  - Voltage SecureData makes full compliance much easier



Voltage SecureData Benefits

- FPE minimizes implementation difficulty
  - Most databases require no schema changes
  - Most applications require minimal or no code changes
- Persistent encryption prevents accidental leakage
  - Compensating controls only cover holes you know about
  - Integrate with existing monitoring and scanning tools
- True separation of duties
  - DBAs can still do their jobs, no access to “Red” data without authorization
- Role-based access model allows granular data policies
  - CSR only sees last 4 of credit card; fraud investigator sees all 16
  - Full re-use of identity/access management systems

Voltage security

Summary

Conclusion

- Encryption is not a luxury, not optional today
- A complex topic, but one that ***can*** be tamed
- Many solutions exist
- Different data/media require different solutions

- Voltage SecureData solves many of the problems for data at rest and data in motion
  - Not a solution for whole-disk, whole-tape encryption
  - The best solution for existing data, existing applications

Encryption Resources

- InfoSecNews.org: email/RSS feed of security issues
  - http://www.infosecnews.org/mailman/listinfo/isn
- Voltage security, cryptography, and usability blog
  - http://superconductor.voltage.com
- Bruce Schneier's CRYPTO-GRAM monthly newsletter
  - http://www.schneier.com/crypto-gram.html
- RISKS Digest: moderated forum on technology risks
  - http://catless.ncl.ac.uk/risks
- US Computer Emergency Response Team advisories
  - http://www.us-cert.gov/cas/signup.html
- Track breaches: www.privacyrights.org and datalossdb.org and www.idtheftcenter.org

Questions?

Phil Smith III

703.476.4511 (direct)

phil@voltage.com

www.voltage.com