FIDELITY NATIONAL INFORMATION SERVICES

GT.M Database Encryption

Protecting Data At Rest (DAR)

April 30, 2009

What it is

- Protects “data at rest” (DAR)
  - Data records in database & journal files are encrypted

What it is not

- Data not at rest not protected
  - During computation within the process address space
    - Processes need to manipulate actual data
  - In transit between systems and processes
    - *Database encryption is only a part of a complete security architecture*
- Doesn't include algorithms & key management
  - *You must choose encryption libraries*
    - No encryption scheme meets all needs
    - Plug-in architecture for you to use encryption library of your choice
    - Example reference plug-in included with GT.M
  - *You must implement key management*
    - Key management is determined by your encryption library and organizational security policy
    - Reference plug-in uses GnuPG (http://gnupg.org)

- *Encryption libraries (even those used by reference plug-in) and key management are <u>not</u> supported by FIS*

FIDELITY NATIONAL INFORMATION SERVICES

Limitations

- General
  - Long lived keys
  - Large volume of encrypted data
  - No key recovery (“back doors”) built into GT.M
    - Losing or forgetting your keys will make your data indistinguishable from random ones and zeros on disk
  - No protection against weak key management
    - Leaving unencrypted keys on disk, even in an obscure location or “secured” with a known password is like leaving your front door key under the doormat
- GT.M Specific
  - Only BG access method is supported; MM is not supported with encryption
  - Encryption is only supported for databases – use PIPE device to read/write encrypted flat files

- *Database encryption is only a part of a complete security architecture*

Plug-in API

- Functions
  - `gtmcrypt_init()`
  - `gtmcrypt_getkey_by_name()`
  - `gtmcrypt_getkey_by_hash()`
  - `gtmcrypt_hash_gen()`
  - `gtmcrypt_encode()`
  - `gtmcrypt_decode()`
  - `gtmcrypt_close()`
  - `gtmcrypt_strerror()`
- Data structures
  - `gtmcrypt_key_t` – handle to a key
  - `xc_fileid_ptr_t` – file identifier

- Rest of the slides are about sample reference implementation

FIDELITY NATIONAL INFORMATION SERVICES

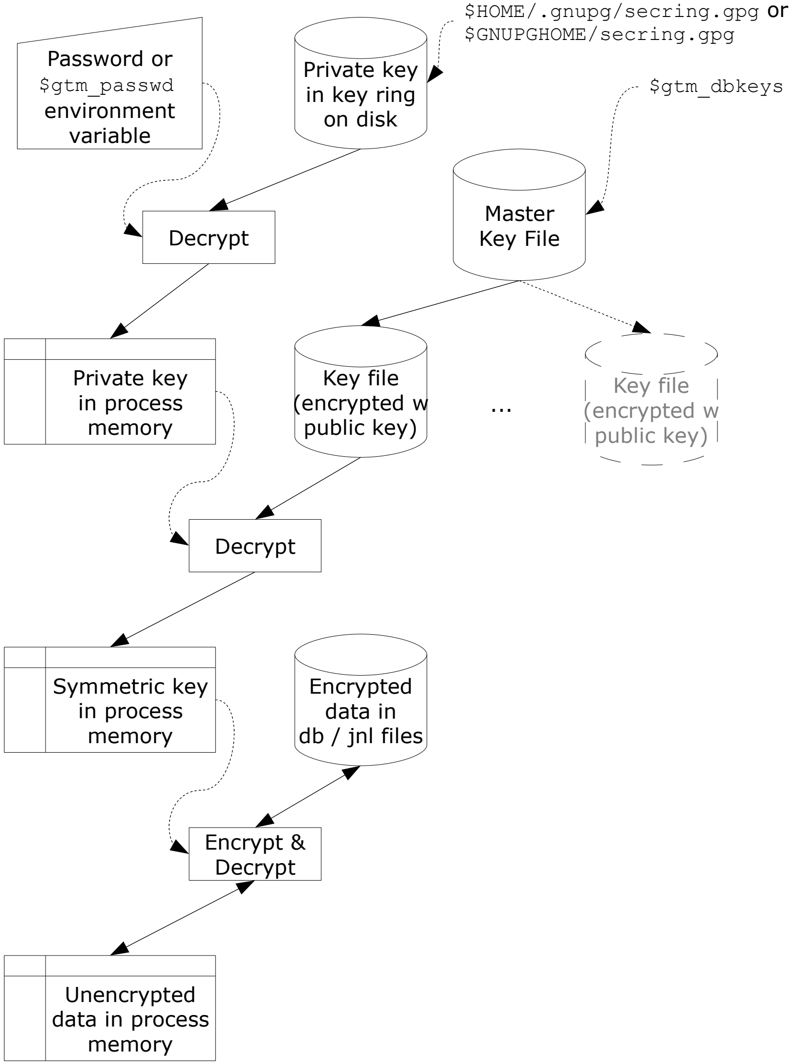
Sample Reference Implementation

- Plug-in architecture – your choice of encryption library / libraries – with sample reference implementation
- Works out of the box with GNU Privacy Guard (libcrypto from OpenSSL for some functionality on AIX)
- Complete source code included
- You can freely modify / adapt to your needs
- Reference implementation is supported by FIS as part of Profile/GT.M support; *encryption libraries are not supported by FIS*

Ciphers & Hashes

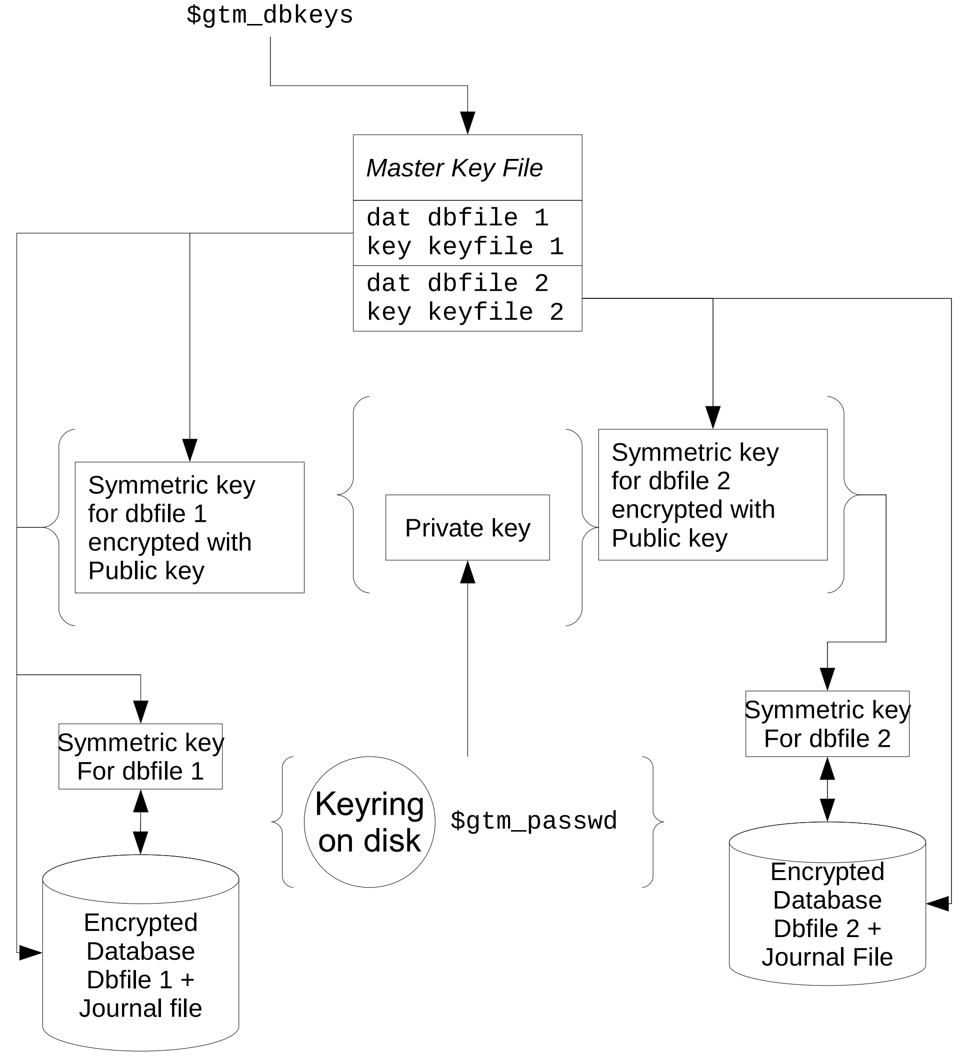
- Symmetric ciphers are computationally faster
  - Keys are hard to distribute securely
- Asymmetric ciphers are computationally slower
  - Public / private keys make key distribution & management easy
- Sample reference implementation uses both:
  - Data records in databases secured with symmetric ciphers
    - Blowfish CFB from OpenSSL libcrypto on AIX; AES 256 CFB from GPG on all others
  - Keys for symmetric ciphers secured with asymmetric ciphers: RSA from GPG
  - Key ring on disk secured with password: GPG
- Key + Cipher description hashed and stored in database file header; validated when file opened
  - SHA-512 hash
  - Hash can be changed separately from cipher – no need to extract / load data

Password flow within GT.M process



FIDELITY NATIONAL INFORMATION SERVICES

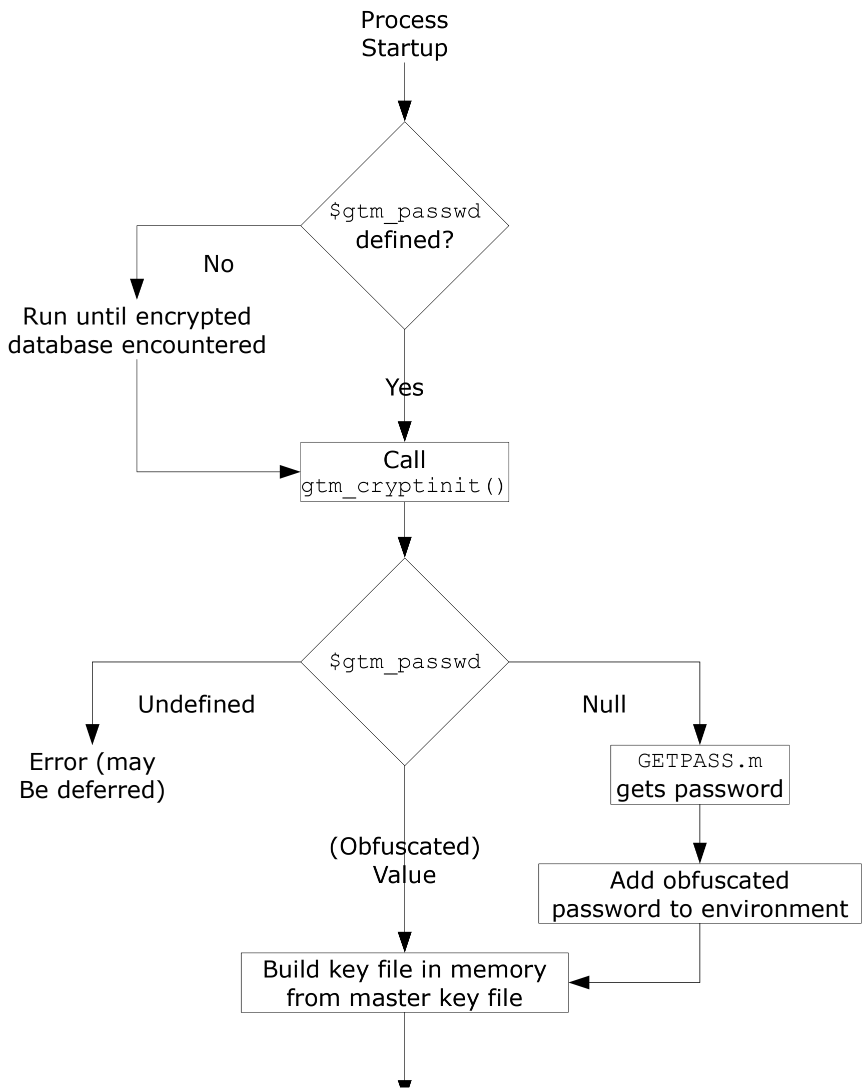
Password flow within GT.M process – multiple db regions



$gtm_passwd

- Functional requirements
  - Interactive entry
  - Inherit from parent process for Job & ZSYstem commands
- $gtm_passwd cases
  - Not set – mumps process assumes application code will set obfuscated password in environment when it is ready to open database / journal file
  - Null string – mumps process prompts user for password at process startup and sets $gtm_passwd to obfuscated password
  - String value – assumed to be obfuscated password (used by parent to pass password to child process)
- Obfuscation is not encryption
  - Obfuscation uses low level information accessible within the system to allow one process to pass the password for the key ring on disk to another process on the same system
  - Obfuscated passwords are not usable outside the system, so if a process environment is dumped and sent to FIS for a support issue, the obfuscated password does not provide access to the actual password

$gtm_passwd – mumps process logic flow



Providing passwords to utility programs

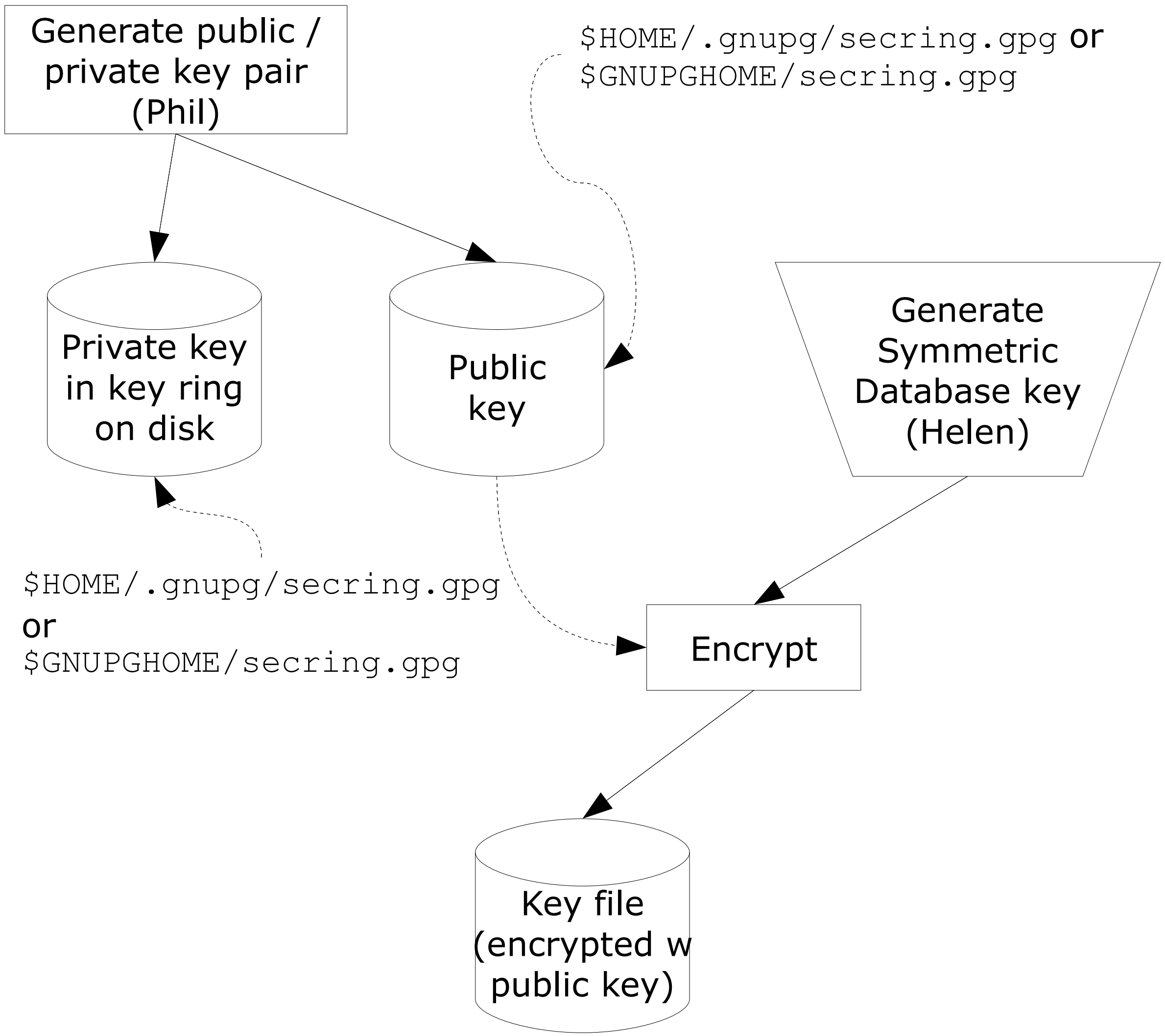
- maskpass program, e.g.
  - ```
    $gtm_dist/plugin/gtmcrypt/maskpass
    Enter Password: 3D303E34213F
    $ echo -n "Enter Password: ";export
    gtm_passwd=`$gtm_dist/plugin/gtmcrypt/maskpass|cut -f 3 -d " "`;echo
    $gtm_passwd
    Enter Password: 3D303E342438
    ```
- Invoke via mumps program
  - Create a one line GT.M program as follows:
    ```
    zcmd        ZSYstem $ZCMdline Quit
    ```
    and use it invoke the MUPIP or DSE command, e.g.
    ```
    $ gtm_passwd="" mumps -run zcmd mupip backup -region \"*\"
    ```

FIDELITY NATIONAL INFORMATION SERVICES
    ```

Key Management

- Every user and administrator id needs a public / private key pair
  - Consider putting keys in a Key server, e.g., http://pgp.mit.edu
- For every encrypted database that a user id needs access to, the symmetric encryption key needs to be encrypted with the public key of that user and put in a file that user id has access to
- Each user id will have a key file for each database file that user id has access to
- Each user id can have a single database keys file that lists all the database files that user id has access to and points to the key file for each database file

Key Management – Simplified schematic



FIDELITY NATIONAL INFORMATION SERVICES

Other

- Changing database encryption requires extracting and loading the data with MUPIP
  - Use a logical multi-site (LMS) configuration to provide application availability during the process
- Use different keys for each instance, so that if a key is compromised, only that instance requires changing keys
- Database Operation
  - Global buffer pool twice as large – each buffer now has two versions
  - Some performance impact is inevitable – benchmark before putting encrypted databases into production
- Supported platforms: AIX, HP-UX (Itanium), GNU/Linux (x86, x8664 & Itanium), Solaris (SPARC), z/OS

Discussion

- K.S. Bhaskar
  SVP, Fidelity Information Services, Inc.
  ks.bhaskar@fnis.com
  +1 610.578.4265

FIDELITY NATIONAL INFORMATION SERVICES