May 2003

Inside PGP® Key Reconstruction

A PGP Corporation White Paper
Author: Will Price

Introduction

One of the most common problems with a large-scale PGP deployment is that users sometimes forget their passphrases or even lose their private keys. This leads to help desk calls, which can be costly for an organization. Over the years, PGP has taken several major steps to deal with this problem. Various pieces of the PGP solution when factored together present a solid front against this problem, including: use of split-key ADKs for data recovery, automatic binding of designated revokers to keys upon generation, and now Key Reconstruction.

First, a terminology clarification. PGP 6.0 introduced Split Keys, where a key is mathematically split into a given number of shares that must be reconstituted in order to use the key via the approval of the parties holding the shares. This is a common method used for creating ADKs or other high-security keys that should require an approval process to use. This "Key Reconstitution" feature is not related to the Key Reconstruction discussed here, although they sound similar and both use similar cryptographic technologies.

Key Reconstruction is a feature designed to solve the lost passphrase or key problem entirely on the user's end, obviating the need for a help desk call or, in the worst case, a reconstitution of the ADK to decrypt the user's important data, revoke the user's key, followed by generation of a new key, and finally re-encryption of the data.

Users know when they have forgotten their passphrases, so there should be a way for them to solve the problem themselves. Some companies have resorted to generating all keys centrally in order to bypass this problem—in such a scenario a central administrator is able to access and thus use all the keys in the organization. That kind of solution significantly decreases the security of the whole system and eliminates non-repudiation of signatures. It is imperative that any solution for passphrase or key reconstruction is entirely within the control of the user, and that no third party is ever allowed access to any of the private portions of the key material.

An Overview of Key Reconstruction

At key generation time (or at a later time, if the user chooses it from the menu), the user is presented with a window on which he or she is to provide five questions and five answers; this happens automatically if configured by the administrator, or the administrator can configure the feature as an option for those who choose to use it. Sample questions are shown on this dialog, but these are only samples. The questions should always be specific to the user, and the answers should be phrases.

A good example of a question and answer here would be "Where did I dance when I was young?" and the answer "on the table at the beach house." This is a user-specific question. No one would guess that answer, guess how the user would type it, or be able to force the answer out of someone familiar with the individual. At the same time, the answer—chosen from the user's long-term memory—is not something the user would ever need to write down and it is likely to come readily to them when prompted with the question.

The user does need to receive direction regarding how to formulate these questions and answers. A **very bad** question would be "What is your mother's maiden name?" This question is easily answered by any motivated attacker. A dictionary of last names would break this without any research at all. The window text and sample questions attempt to provide guidance here, and the online help for the window provides greater guidance.

An alternative, very-high-security method is to not use the questions at all. Some users are likely to have their own methods for coming up with passphrases and passwords. Often these are a mix of various words with varying punctuation and capitalization. Mixing these factors into five different passphrases the user is likely to be able to guess later is a good idea. The only cues in this method are in the head of the user. Some users will not be amenable to this method however, and the most important class of users to target with Key Reconstruction are the novice users most likely to forget their passphrases.

When the user has entered her five questions and answers (the questions are optional, the answers are not), the whole package is wrapped up into a block of data that will be described in the next section. The administrator configures what happens to this block. Currently, there are two choices: sending it to a PGP Keyserver, or sending it to a standard LDAP directory.

The two methods operate slightly differently, and each has its advantages and disadvantages. Sending it to a PGP Keyserver stores it on the server under the key ID of the key. Anyone with access to the server can download the questions for a given key from the server—all such accesses are logged. The only thing that someone can retrieve from PGP Keyserver without the correct answers to the questions is the questions themselves.

Sending it to a standard LDAP directory requires that the directory be configured for username/password access control. In a corporate environment where an LDAP directory already exists with username/password accounts, this could be very simple to use. In this case, no information can be obtained without authenticating as the user first. Once authenticated, the entire block is transmitted back to the user rather than only the questions.

The LDAP directory method is not as simple because it requires entering a username/password. If a user doesn't remember his PGP passphrase, who knows whether he'll remember his LDAP password? (In the general case, one would assume their LDAP password is the same as the NT Domain login account and thus it would be remembered). Also, if an attacker can hack this password, he will get the entire block—and, while an attacker obtaining the reconstruction block is not a serious threat, it is generally undesirable.

The PGP Keyserver method is easiest, but leaves open the possibility that an attacker could attempt to guess the answers to the questions because they can be retrieved for any key. However, no more information is available to the attacker unless he or she successfully answers at least three of the questions.

If the user later forgets his passphrase, he simply chooses "Reconstruct Key" from the menu in PGPkeys, his questions are retrieved from the server, the user gives the answers, and, provided three of the answers are correct, his entire key is reconstructed with a new passphrase the user provides at that time. Note that the entire key can be lost and still be reconstructed. The system is not limited to passphrase loss.

Indeed, a possible use for this feature is a user that goes on a trip with a company laptop borrowed for the occasion, and has forgotten to take his private key. He might simply download his public key from the server, select Reconstruct Key, answer the questions, and moments later his private key will be reconstructed for use on the new machine.

Key Reconstruction Internals

Key Reconstruction makes heavy use of the Blakely-Shamir key splitting technology used since PGP 6.0 to distribute shares of a key among separate parties. In this case, shares of a key that can be used to decrypt the private key are distributed among the five answers provided by the user, and the split threshold is set to three.

The following steps are performed to create the Key Reconstruction block sent to the server:

1. An exported version of the public and private keypair is converted in memory to a binary passphrase-less key. This is never written to disk. The block is padded to the length of the symmetric cipher block length (a multiple of eight bytes).

2. A random 128-bit symmetric key is generated (symKey A).

3. SymKey A is split into five shares (shares A-E) with a threshold of three.

4. A random number of hash repetitions is generated that is greater than zero and less than 16384 (hashReps).

5. The five answers are converted into symmetric keys (symKey B-F) by hashing them using the SHA-1 cipher, concatenating the hash with a fixed magic value of 0x750A0F2E (magic1), and iterating this hash a random number of times determined by HashReps.

6. Each Share A-E is then encrypted using the CAST5 cipher in CBC mode with a null initialization vector by one of symKey B-F, creating encrypted shares (encShares A-E).

7. The exported keypair, with a preceding length value to correct for the padding in Step 1, is encrypted using CAST 5 in CBC mode with a null initialization vector by SymKey A, thus creating encKey.

8. The final "InnerBlock" is created in memory in the format:

   [keyID] [hashReps] [encShares A-E] [encKey]

9. If the length of InnerBlock is not divisible by the symmetric cipher block length, it is padded to that length.

10. Another random 128-bit symmetric key is generated (symKey G).

11. SymKey G is split into five shares (shares F-J) with a threshold of three.

12. The five answers are converted into symmetric keys (symKey H-L) by hashing them using SHA-1, concatenating the hash with a different fixed magic value of 0x21B8E41F (magic2), and iterating this hash a random number of times determined by hashReps.

13. Each split share F-J is then encrypted using CAST5 in CBC mode with a null initialization vector by one of symKey H-L, creating encrypted shares (encShares F-J).

14. InnerBlock, plus its padding, is encrypted using symKey G to form encInnerBlock.

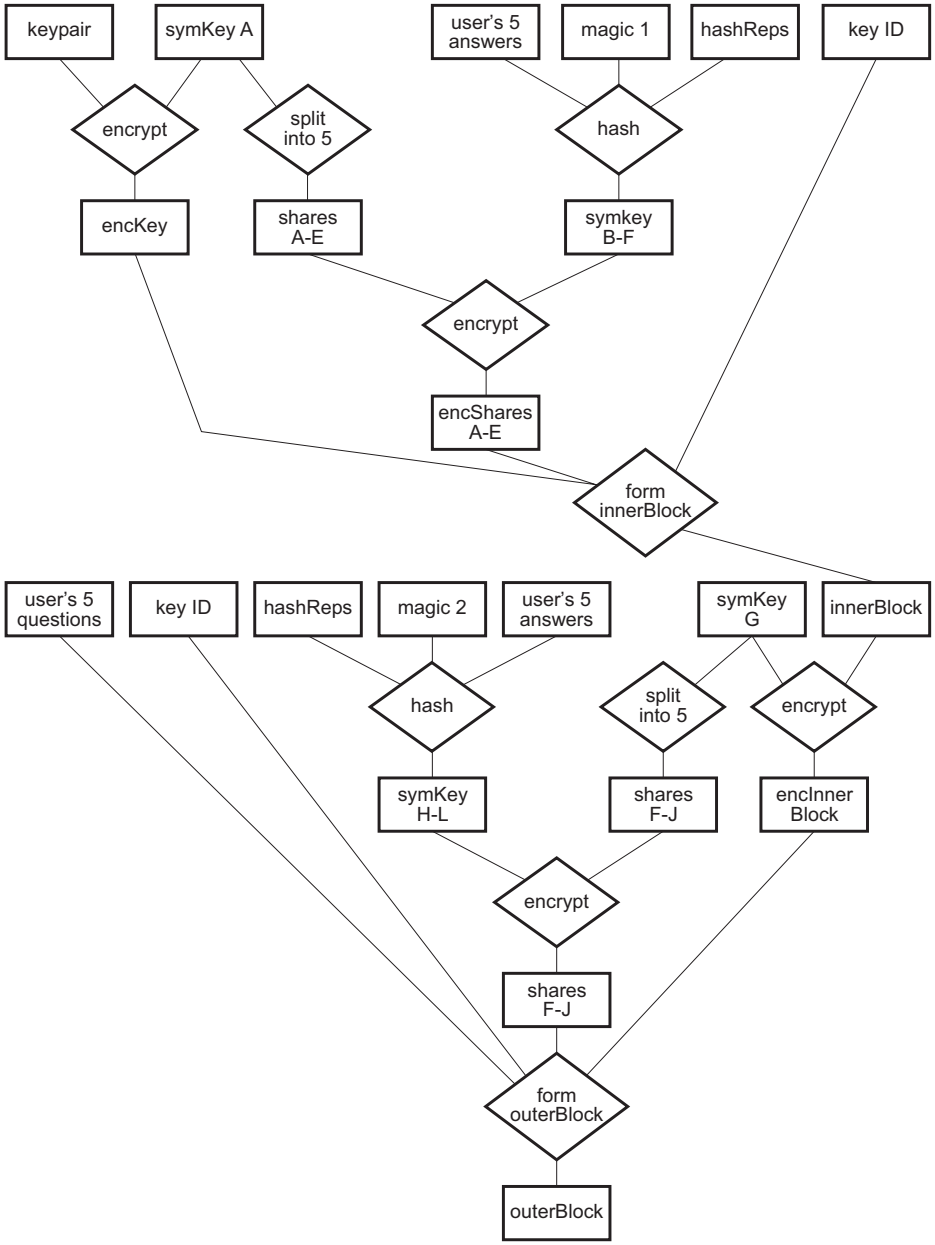
15. OuterBlock is created in memory in the format:

    [keyID] [questions] [hashReps] [PadLen] [encShares F-J] [encInnerBlock]

    Where [questions] is an array of the five questions unencrypted and [padLen] is the size of the padding used to form encInnerBlock.

16. The creation of the Key Reconstruction block is complete and ready to be sent to the server. OuterBlock is sent to the server; InnerBlock and all other key information is cleared from memory.

Diagram of Key Reconstruction block creation



The philosophy behind the construction of this block is that the outer layer allows a third party to play middleman in authenticating the user without actually giving that third party any valuable information whatsoever. PGP Keyserver, for instance, is queried for the questions associated with a given Key ID. It sends those back to the user, the user enters their answers, these answers are hashed with magic2 many times in order to (1) defeat dictionary attacks and (2) differentiate the resulting hashed answers (symKey H-L) from the actual hashed answers (symKey B-F), which would be used to decrypt the private keypair. Since the fundamental property of the hash algorithm is that it cannot be reversed, the use of a different magic value in the stages makes it impossible to determine symKey B-F from H-L. The user sends symKey H-L to the server. The server authenticates that the hashes do successfully unlock the encInnerBlock by verifying the Key ID contained in the header of innerBlock.

Sending these answers to the server only provides it with enough information to verify that the user is likely to be successful in reconstructing his key—a decision made based on the fact that the user has proven the ability to generate symKey H-L, which indicates that the user should also be able to generate B-F. It does not bring an evil server one inch closer to being able to decrypt encKey because the server has no knowledge of symKey B-F, which must be used to decrypt the shares in innerBlock.

Of course, to defeat replay attacks, which would allow sniffing parties to gain access to the innerBlock if a user authenticates successfully, all reconstruction URLs should use "ldaps://" in order to enable TLS for the connection. There is no user interface in PGP to set the reconstruction URL. It must be set by the administrator from PGPadmin. If no reconstruction URL has been set, all reconstruction functionality is disabled.

Once the user has received outerBlock from the server, he repeats the process the server just performed, unwrapping encInnerBlock to find innerBlock, and then recalculating the hashed answers with Magic1 to decrypt the shares using symKey B-F, reconstituting those shares, and finally decrypting encKey. The user is then prompted to choose a new passphrase for the reconstituted key.

Using a standard LDAP directory is slightly different. In that case, the user authenticates with a username/password and is immediately able to retrieve the entire outerBlock on which all the same operations will be performed. The difference is that PGP Keyserver actually authenticates that the user will be successful, whereas the standard LDAP directory authenticates that the user is who he says he is. The difference here is roughly nil, although extra features provided by PGP Keyserver are a built-in delay to prevent hacking a particular Reconstruction Block, and detailed logging of attempts.

An important note about this system is that without getting at least three answers correct, an attacker has no idea whether any other answer is correct. The shares will reconstitute happily into a bogus key if the wrong hashed answers are provided, and since the shares appear identical to random numbers, no determination can be made as to why decryption failed.

Another important note is that every answer is tried against every share. There is no retained order when attempting to reconstruct the key. Question 1, Answer 1, and encShare A do not need to be correlated. When generating the reconstruction block, the user could enter answers A-E, and then enter those same answers in reverse order when reconstructing as E-A.

This is useful in the case where the user has not used questions, but rather has entered a series of memorable passphrase-like combinations. While it may be likely that the user will remember such combinations, it is highly unlikely he will remember the order in which they were entered. Thus, every answer is tried against every share. In order to do this, the shares are reconstituted in every combination of shares/answers. This can cause some delay when reconstructing the key, but that is a good thing. It only makes the attacker's job harder.

Conclusion

Since developing the Key Reconstruction feature, we've received lots of good feedback from customers. Some of this feedback is likely to be integrated into future versions. For instance, a commonly requested feature is to be able to save Key Reconstruction blocks to the local disk rather than or in addition to sending them to the server. This is a good idea, which can help out in the case of lost passphrases; although it doesn't do much for the case of lost keys where one might assume the Key Reconstruction file has also been lost.

However, some have asked questions that caused us to write this White Paper so that they could better understand the feature and how it works. For instance, an administrator asked whether he would be able to configure the default questions for his users. There is no benefit to such a feature. We have provided sample questions, which guide the user towards the right path (which is to write their own questions!). No administrator, and no one here at PGP, can think up the "right" questions or the "best" questions for a given set of users. The only logic in allowing these questions to be configurable by administrators would be to dumb down the system using questions such as "What is your birthdate?" Thus, this will not be offered as a feature.

Another question that has been asked is whether the number of questions or threshold of the split will be configurable. Currently, the split is five with a threshold of three. This was arrived at after careful analysis, and seems to be the tradeoff point between security and the ability of the user to (1) create good questions and (2) remember the answers. In order to allow other configurations we would have to see convincing heuristic evidence that different values provide a similar or better tradeoff here. We don't want to allow administrators to "configure" weak-crypto into their system, so lowering these limits is probably out of the question. At the same time, the class of target users for this feature is not likely to be capable of a perfect five out of five. If the questions are written correctly, three questions is more than enough to provide security that should be considered greater than or equal to that of the user's normal passphrase.

Some of the technologies presented in this document are patent pending by PGP Corporation.

Legal Notices

Copyright Information

Patent Information

The information described in this document may be protected by one or more U.S. patents, foreign patents, or pending applications.

Trademark Information

PGP and the PGP Logo are the trademarks (or registered trademarks) of PGP Corporation. Product and brand names used in this document may be trademarks or registered trademarks of their respective owners. Any such trademarks or registered trademarks are the sole property of their respective owners.

Disclaimer

The information in this document is provided "as is" without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. This document could include technical inaccuracies or typographical errors. Changes to this document may be made at any time without notice.