PPK: Password as Private Key

By: Mehdi Sotoodeh
mehdisotoodeh@gmail.com

PPK was developed to provide an effective method of secure file transfer in an insecure environment. It uses state of the art crypto for high security while maintaining small footprint in terms of application as well as key sizes.

Background

Encryption refers to algorithmic schemes that encode plain text into non-readable form or cyphertext, providing privacy. The receiver of the encrypted text uses a "key" to decrypt the message, returning it to its original plain text form.
In symmetric crypto systems, both sender and receiver share the same secret key.
However, in an asymmetric system two keys are involved; public and private keys.
Each side can have their own private and public key pairs and only share the public parts.
This would be ideal as you can publish your public key like your phone number.
This is the major advantage of asymmetric systems as in the symmetric case, passing the shared secret has its own weaknesses and security threats.

Public and private keys are mathematically related. In another word, you can calculate public keys if you know the private part but the reverse operation is computationally very time consuming and practically not feasible using existing technologies.
You can safely share your public key with others as long as you keep the private key safe and difficult to guess.

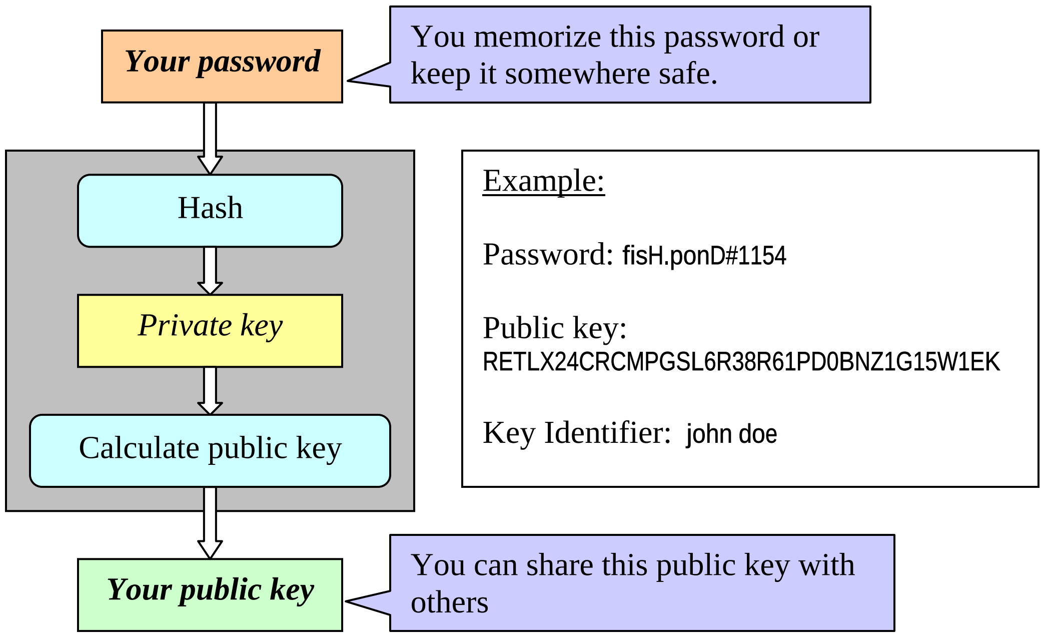
PPK Implementation

Asymmetric keys are not very efficient for bulk data encryption. For this reason
PPK uses a combination of symmetric and asymmetric cryptographic algorithms.
It uses public-private keys for a random key exchange and uses symmetric algorithms for the actual data encryption.

See “About PPK …” in the system menu of the main window for information about used crypto operations.

Key Generation

You can use PPK to create your own key pair. PPK uses a user defined password to derive the private and public keys from it. This eliminates the need for secure storage as your own memory plays that role.
For additional security, PPK does not provide any mechanism to store your passwords or private keys.



Your password
You memorize this password or keep it somewhere safe.
Hash
Private key
Calculate public key
Your public key
You can share this public key with others
Example:
Password: fisH.ponD#1154
Public key: RETLX24CRCMPGSL6R38R61PD0BNZ1G15W1EK
Key Identifier: john doe

Both of the above functions are one-way operations and cannot be done in reverse direction.
Generated public key is a series of alphanumeric characters. PPK provides a user defined string as the key ID which is a friendly name to identify the key owner.
You may use your name or email address as the key ID.
The generated key is presented as:

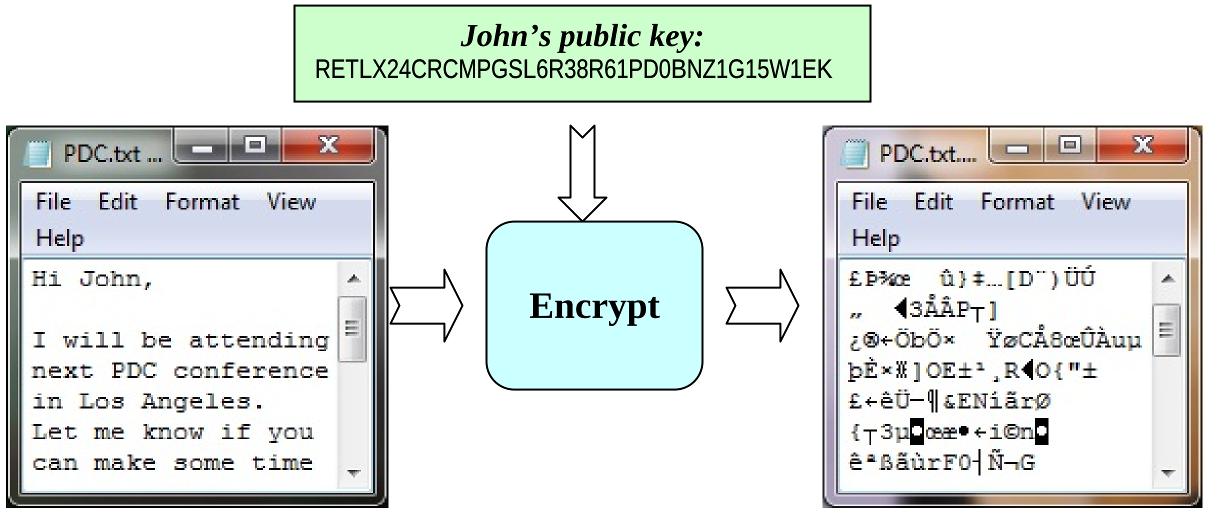
{RETLX24CRCMPGSL6R38R61PD0BNZ1G15W1EK/john doe}

You can email or publish your generated key for others to use.
Anyone can use your public key to encrypt files that can only be decrypted by you.

Note that passwords are case sensitive and exact match is needed to end up with the same private & public keys.

Encryption

PPK encryption encodes its input file into a non-readable format. The encrypted file can only be decrypted by the person who knows the associated password.



```
Hi John,

I will be attending
next PDC conference
in Los Angeles.
Let me know if you
can make some time
```

This process can be applied to pictures; videos; executables or any document or file type. PPK encryption also includes file integrity checks which enables decryption operation to detect file corruptions during file transfer.
You also can use PPK to encrypt your own files. For this purpose use your own public key for encryption.

To perform this operation:

1. From PPK main windows click Encrypt button
2. Enter or browse for input file.
3. Optionally you may select a file name and location for the output file. You may leave this field blank which in this case PPK creates the output file in the same folder that your input file is.
4. Select a key from the drop down list box.
5. Click Encrypt

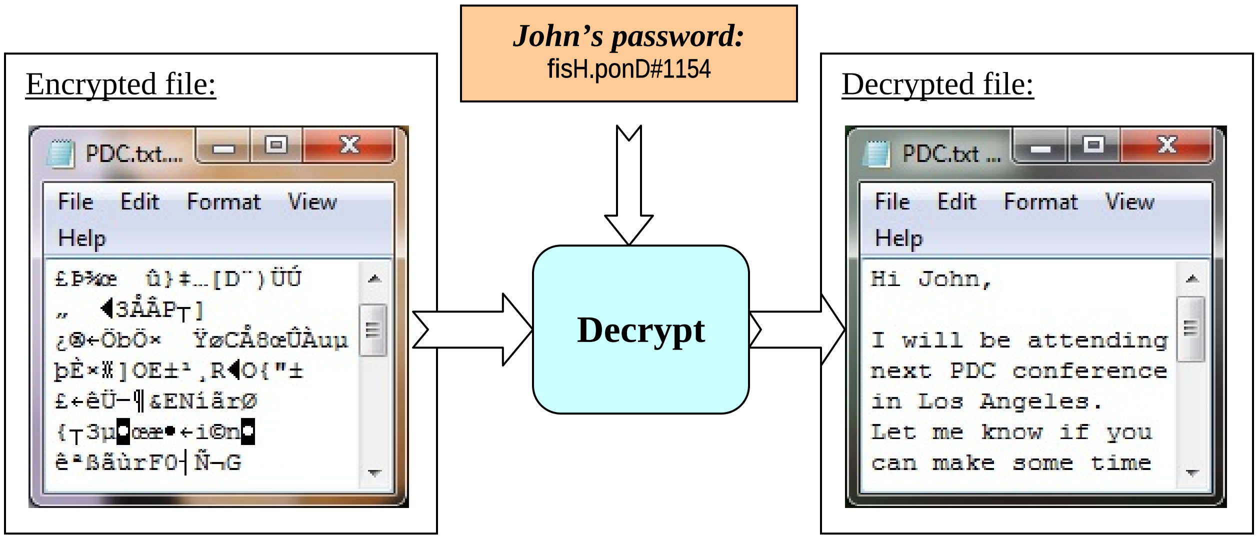
Another alternative is to drag and drop files into PPK icon or PPK window.

PPK embeds input file name in the encrypted image. This gives you the option to choose any name for the output file.

Note that if you need to compress files do it before encryption. Encryption kills the redundancies which affects your compression ratio.

Decryption

Decryption transforms the encrypted file to its original form. This operation needs the other half of the key, your password.



```
£Þ¾œ  û}‡…[D¨)ÜŰ
„  ◀3ÅÂP┬]
¿®←ÖbÖ×  ŸøCÅ8œÛÀuµ
þÈ×※]OE±¹¸R◀O{"±
£←êÜ–¶&ENiãrØ
{┬3µ◘œæ•←i©n◘
ê ª ßãùrFO┤Ñ¬G
```

```
Hi John,

I will be attending
next PDC conference
in Los Angeles.
Let me know if you
can make some time
```

For this operation, click Decrypt button on the PPK main window and enter input file and output path. This operation prompts for the password and if correct value is entered, file will be restored into its original form.

You may use drag and drop files into PPK icon or windows to trigger this operation.

Note that if you repeat this process for multiple files and they are encrypted by the same public key you will be prompted for the password only once.

Key Management

Public keys are maintained in *PPKDir.csv*. This file is created first time you run PPK. PPK allows you to add or remove keys to this file.
The format of this file is standard CSV (Comma Separated Value).
You can open this file using any text viewer, spreadsheet or database applications.

This file is located where application specific data is stored.
Example:

C:\Users\mehdi\AppData\Local\PPK\PPKDir.csv

Where, *C:\Users\mehdi\AppData\Local* is OS-specific and it could be different from system to system.
You have the option of moving this file to the folder that PPK is installed. This gives you the option of using PPK on a removable media like USB drives.

Key management dialog starts by clicking 'Manage Keys' button or pressing Alt-K on the main page. It gives you following options:

Add: Import someone's public key.
To minimize typing errors, copy and paste the public key that you like to add.
Format of key is expected to be like:

{93467BHM38L5GYUSYZARNUPJE0U3F92BP0TH/mehdi's key #1}

Remove*:* Remove a key from the list.
Select the entry in the loaded keys list box and press Remove button.

New Key: Create your own key pair.

Your PPKDir.csv file is updated when you click OK button.

Security Considerations

MITM (Man-In-The-Middle) attack is carried out when someone sits between you and the expected recipient.
It works by attacker passes his public key instead of the intended recipient.

For example Adam wants to send an encrypted file to Bob.
In a normal case:

- Adam encrypts a file using Bob's public key: {09UDR2P7DGW04FJE5H7NRLSSH56X9HU460ES/bob}
- Adam sends the encrypted file to Bob.
- Bob decrypts the file using his own password.

Now let’s assume an attacker controls the communication channel.

- Attacker creates his own key: {6H651J780D0FP482WYFDMUAS8YRE614FZ5T9/middle-man}
- Attacker intercepts Bob's email and replaces the key part of Bob's public key with his own.
- Adam receives the public key {6H651J780D0FP482WYFDMUAS8YRE614FZ5T9/bob} assuming this is truly Bob's key.
- Adam encrypts file using {6H651J780D0FP482WYFDMUAS8YRE614FZ5T9/bob}
- Attacker receives the encrypted file and decrypts it using his own password
- Attacker encrypts the file using the actual Bob’s public key: {09UDR2P7DGW04FJE5H7NRLSSH56X9HU460ES/bob}
- Attacker sends the encrypted file to Bob.
- Bob decrypts the file using his own password.

With this scenario, both Adam and Bob are not aware of the presence of attacker. For this reason, it is recommended to validate the actual public key through a secondary channel (like phone). In this case you are sure the public keys that you are using are trustworthy.