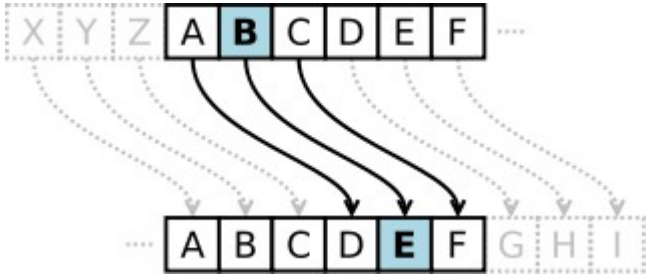


Cryptography

Criptografía para humanos

Luis González Fernández

@luisgf_2001

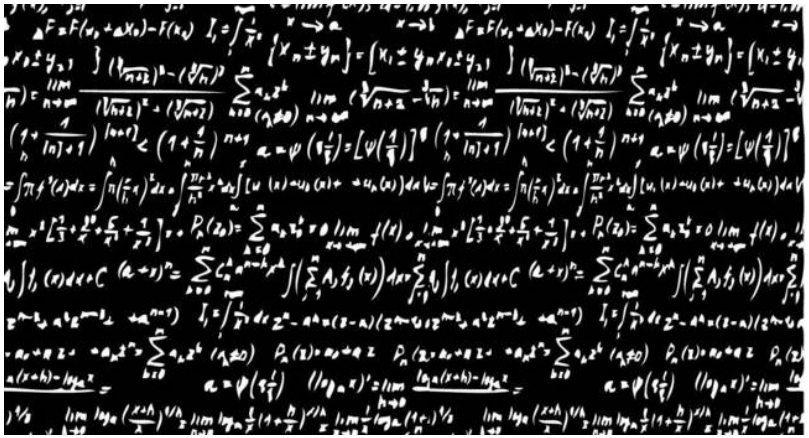
luisgf@luisgf.es

¿Qué NO es esta presentación?

No es una clase de criptografía.

Nada de Fermat, Euler,

Funciones Feistel, Coprimos…



Es una introducción al uso de esta biblioteca acompañada de algunas demos.

Implementar criptografía en aplicaciones de una forma sencilla.

Cryptography 101

- Es una biblioteca que exporta funciones criptográficas de una forma sencilla pero te permite profundizar en sus tripas si así lo deseas.

- Soporta Python => 2.6

- Está muy bien documentada: https://cryptography.io/en/latest/

¿Porqué una nueva biblioteca criptográfica?

- Existen diversos problemas con las bibliotecas actuales (M2Crypto, PyCrypt, PyOpenSSL, ECDSA)
- Falta de soporte Python 3.x
- Poco mantenimiento
- Mala implementación de los algoritmos (algunos de ellos contienen side-channel attacks)
- API compleja y difícil de usar
- Ausencia de algoritmos fuertes como AES-GCM

Porqué usar cryptography

- Porque tienen una política de estabilidad de su API estricta.
- La API pública no se elimina o renombra sin antes proveer de un alias que aporte compatibilidad.
- Se comprometen a no cambiar el comportamiento del API existente en las próximas 2 versiones.
- Se reservan el derecho a saltarse esta política en caso de fallo de diseño que implique un problema de seguridad.
- Las implementaciones criptográficas no son nativas. Utiliza "*bindings*" con los diferentes "*backends*" disponibles usando ***libffi***. Es decir, criptografía rápida.

Porqué usar cryptography

- En caso de un eliminación o cambio en la API, el protocolo de actuación es el siguiente.

cryptography **X.Y**	La funcionalidad existe
cryptography **X.Y+1**	El uso de la funcionalidad emite un warning del tipo **PendingDeprecationWarning**
cryptography **X.Y+2**	El uso de la funcionalidad emite un warning del tipo **DeprecatedWarning**
cryptography **X.Y+3**	La funcionalidad es **eliminada o cambiada**.

Instalación y puesta en marcha

- Simple, usando Pip

```
$ pip install cryptography
```

- En linux se compila y enlaza dinámicamente con el OpenSSL del sistema (aunque se puede especificar otro).

- Mac OS X utiliza CommonCrypto y en Windows se distribuye de forma binaria con la última versión de OpenSSL.

Composición de la biblioteca

- Distribuida en 2 grandes bloques. Llamados *recipes* y *primitives*

- **Recipes**. Esta parte contiene expone una serie de métodos cuya configuración por defecto es “segura”.Dá poco margen a los programadores para toquetear.

- **Primitives**: Esta expone primitivas criptográficas más complejas donde resulta muy fácil meter la pata. Usar con cautela.

Organización de Módulos

- **cryptography**
  - fernet
  - x509
  - exceptions
  - hazmat
    - primitives
    - bindings
    - backends

```
>>> import cryptography.fernet
>>> import cryptography.hazmat.primitives
```

Funcionamiento del Módulo Fernet

- Es una implementación de cifrado autenticado.
- Usa el algoritmo AES en modo CBC, con una clave de 128 bits.
- Firmado con HMAC y hash SHA256.
- El resultado es un token codificado en base64-urlsafe.
- Soporta múltiples claves (MultiFernet) (*)
- Expiración de los tokens (*)

(*) Es una argucia de la implementación, no del cifrado.

Formato de clave Fernet

- La clave debe de ser una cadena de 32 **bytes** codificada en base64-urlsafe.
- Esta cadena se empleará para generar 2 claves de 16 bytes (128 bits)
  - Una se empleará en el proceso de firma
  - Y la otra se empleará para el cifrado

Clave Fernet = b64encode(Clave de Firma || Clave de Cifrado)

Formato de token Fernet (1/2)

- Es una cadena de bytes codificada en base64-urlsafe
- El token contiene los siguientes elementos:
  - Número de versión (8 bits)
  - Timestamp (64 bits)
  - IV (128 bits)
  - Criptograma (Longitud variable, múltiplos de 128 bits)
  - HMAC (256 bits)

Formato de token Fernet (2/2)

Token = b64encode(Versión || Timestamp || IV || Criptograma || HMAC)

```
0000000  80 00 00 00 00 56 A0 91 C7 67 86 17 21 88 01 5A  .....V...g..!..Z
0000010  E8 ED 9C 88 8C 21 86 50 5F 96 EE 7C 6B 84 4B AC  .....!.P_..|k.K.
0000020  B8 3F 4E 28 96 B4 C1 1C C9 BB 66 A5 63 06 DA DC  .?N(......f.c...
0000030  03 34 E0 32 97 02 89 DC 9E AF AF 17 74 67 2D 76  .4.2........tg-v
0000040  CE 0A C2 C2 81 16 F9 18 AE 4D 10 C0 76 5D 4A 9A  .........M..v]J.
0000050  CA A0 B6 B3 1F 4E                                .....N
```

¿Que implementa cryptography?

- Funciones de resumen (hashes) (Familia SHA 1 y 2,RIPEMD160,Whirlpool y MD5)
- Funciones de resumen autenticadas (HMAC ó CMAC)
- Algoritmos simétricos:
  - AES
  - Camellia
  - 3DES
  - CAST5
  - SEED
  - Cifrados débiles (Blowfish, RC4, IDEA)

¿Que implementa cryptography?

- Distintos Modos de Cifrado
  - CBC
  - CTR
  - OFB
  - CFB
  - CFB8
  - GCM (Cifrado autenticado en 1 sola fase, sin necesidad de cifrar y firmar al final)

- Modos Inseguros
  - ECB

¿Que implementa cryptography?

- Funciones de relleno (Padding)
  - PKCS5
  - PKCS7
- Funciones de derivación de clave (Key Stretching)
  - HDKF
  - PBKDF2

¿Que implementa cryptography?

- Protección de Claves (Key Wrapping)

- Mecanismo para proteger una clave privada y enviarla por un canal inseguro.

- Consiste en cifrar la clave con una contraseña.

¿Que implementa cryptography?

- Funciones de tiempo constante
- Algoritmos Asimétricos
  - Se usan cuando se requiere autenticación y confidencialidad.
    - DSA
    - ECC (Elliptic Curve Cryptography). Todas las curvas NIST definidas en FIPS 186-3 y FIPS 186-4
    - RSA
    - Diffie-Hellman
    - Serialización de claves (PEM,DER, OpenSSH)

¿Que implementa cryptography?

- Autenticación de doble factor
- One-Time-Password
  - HOTP (HMAC-Base One-Time-Password)
    - Basada en un contador
  - TOTP (Time Base One-Time-Password)
    - Basada en tiempo

¿Que implementa cryptography?

- Gestión de Certificados x509
- Carga de certificados.
  - Permite examinar las propiedades de los certificados.
- Carga de listas de revocación (CRL)
- Generación de peticiones (CSR)
- Generación de revocaciones.

Código de las demos.

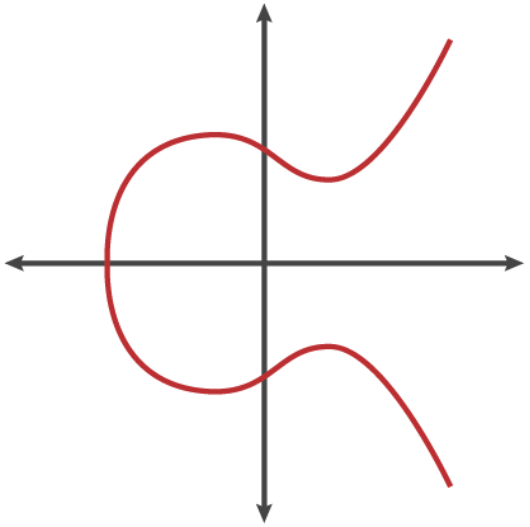
En GitHub:

https://github.com/luisgf/pyvigo-cryptography

GRACIAS!

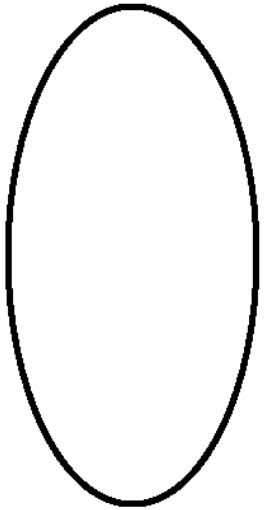
Elliptic curve cryptography

I DON'T UNDERSTAND WHY PEOPLE GET CONFUSED ... I DON'T LOOK ANYTHING LIKE YOU!



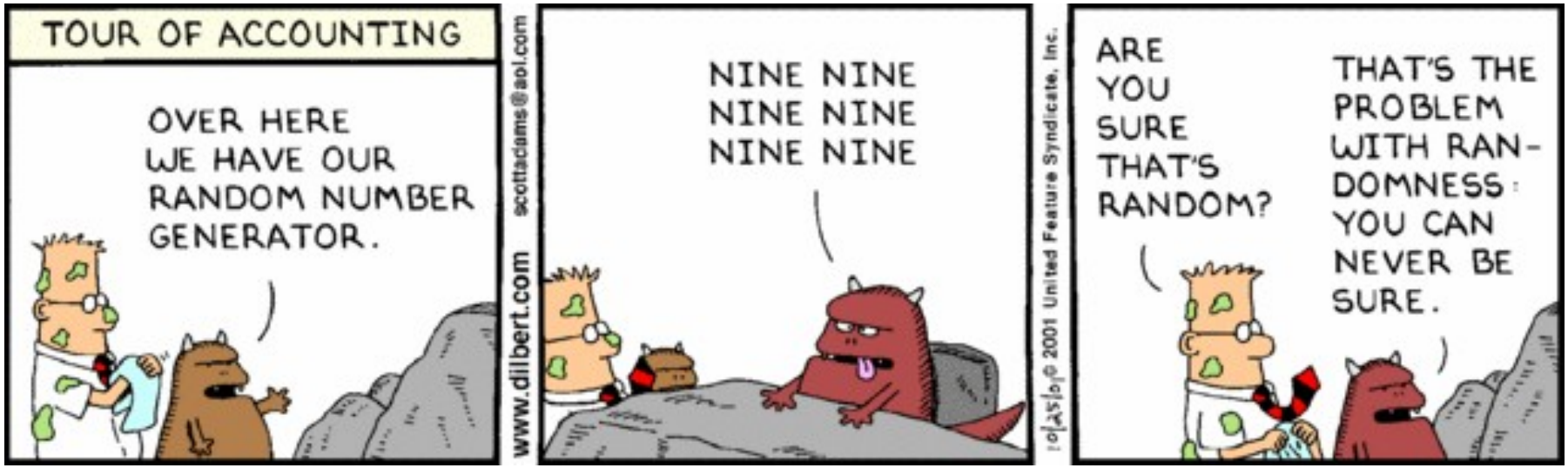
ELLIPTIC CURVE

I THINK IT'S THE NAME! LET'S ASK JAVA AND JAVASCRIPT TO SEE HOW THEY DEAL WITH IT



ELLIPSE

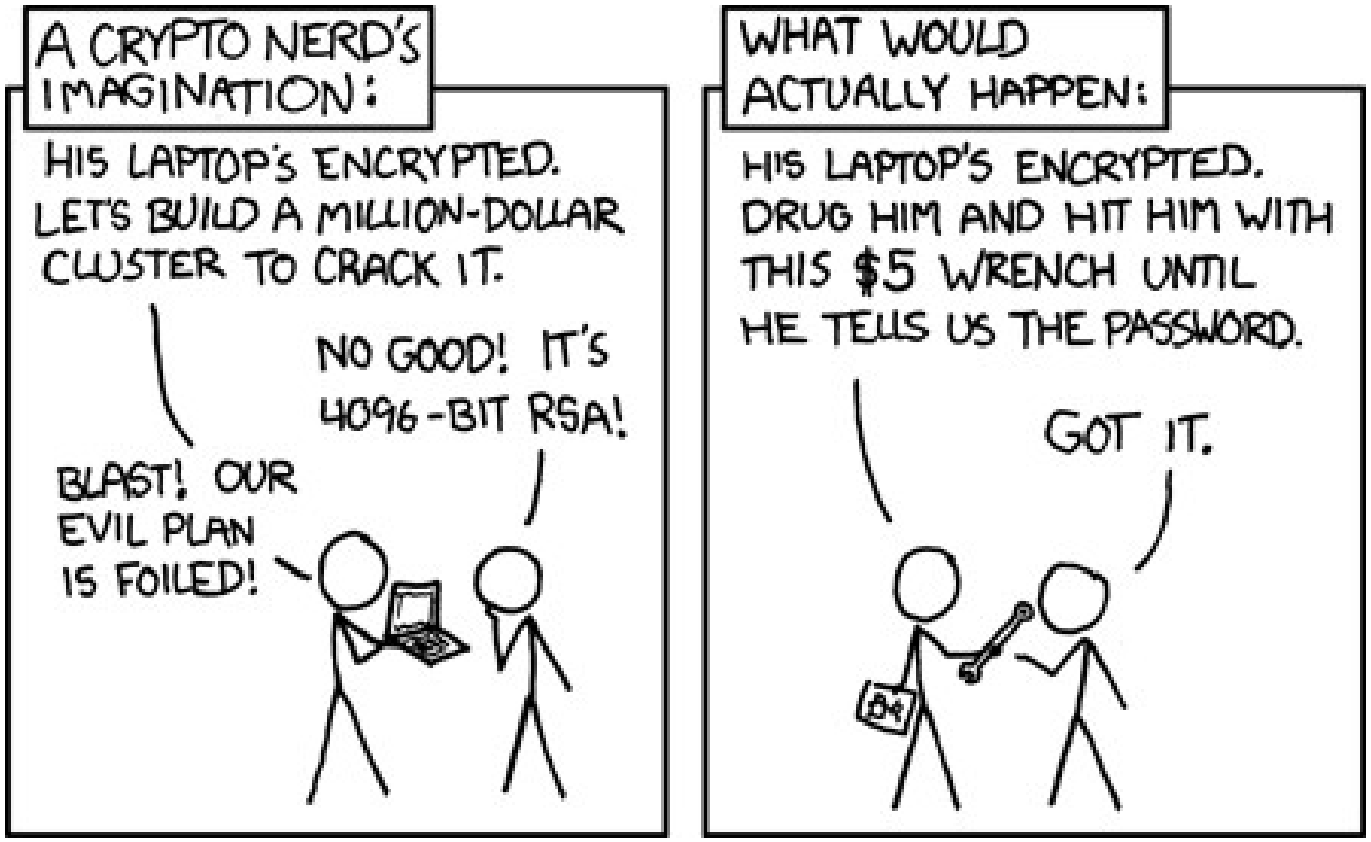
PRNG


TOUR OF ACCOUNTING
OVER HERE WE HAVE OUR RANDOM NUMBER GENERATOR.
www.dilbert.com scottadams@aol.com
NINE NINE NINE NINE NINE NINE
ARE YOU SURE THAT'S RANDOM?
THAT'S THE PROBLEM WITH RAN-DOMNESS: YOU CAN NEVER BE SURE.
© 2001 United Feature Syndicate, Inc.

Ley de Murphy… Por si el efecto “demo” ataca.

- QR HOTP

- QR TOTP


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.