\ShellMates:# echo ' Welcome to @workshop '

APPLIED CRYPTOGRAPHY

PRESENTED BY: MEHDI NACER KERKAR

HACK-INI

$_whoami :

MEHDI NACER KERKAR

- Information Security Student at USDB
- Security Researcher and development at CDTA
- CTF-Player
- Board Member at Shellmates Club

Qu'est-ce que la «Cryptographie» !?


1
- Introduction:

Cryptographie

- La cryptographie est la science qui utilise les mathématiques pour chiffrer et déchiffrer des données.

Phil Zimmermann

- La cryptographie est l'art et la science de sécuriser les messages.

Bruce Schneier

Cryptographie

Cryptographie

Vient des mots *kruptos* « caché » et *graphein* « écrire »

- Dans un sens étroit
  - Transformation d'un message en clair en un message incompréhensible
- Dans un sens plus large
  - Techniques mathématiques liées à la sécurité de l'information
  - À propos de la communication sécurisée en présence d'adversaires

Cryptanalyse

- L'étude de méthodes permettant d'obtenir la signification d'informations chiffrées sans accéder aux informations secrètes

Cryptologie

- Cryptographie + analyse cryptographique

Attaques de sécurité

- Attaques passives
  - –Obtenir le contenu du message
  - –Surveillance des flux de trafic

- Attaques actives
  - –Mascarade d'une entité comme d'une autre
  - –Rejouer les messages précédents
  - –Modifier les messages en transmission
  - –Ajouter, supprimer des messages
  - –Déni de service

Objectifs de la Cryptographie

Confidentialité (secret)

- Seuls l'expéditeur et le destinataire doivent être en mesure de comprendre le contenu du message transmis.

Authentification

- L'expéditeur et le destinataire doivent tous deux confirmer l'identité de l'autre partie impliquée dans la communication.

Intégrité des données

- Le contenu de leur communication n'est pas altéré, que ce soit par malveillance ou par accident, lors de sa transmission.

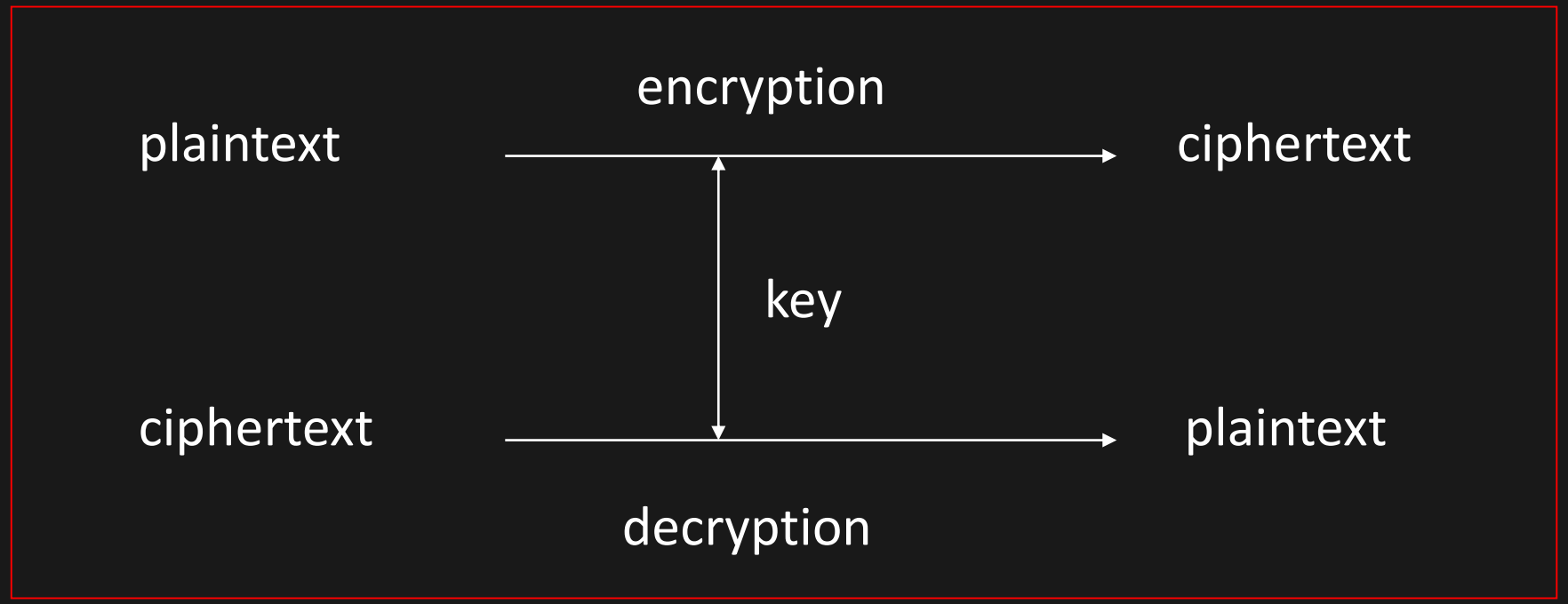
Disponibilité

- Accessibilité rapide des données aux entités autorisées.

Types de Fonctions Cryptographiques

- Fonctions de clé secrète
- Fonctions de clé publique
- Fonctions de hachage

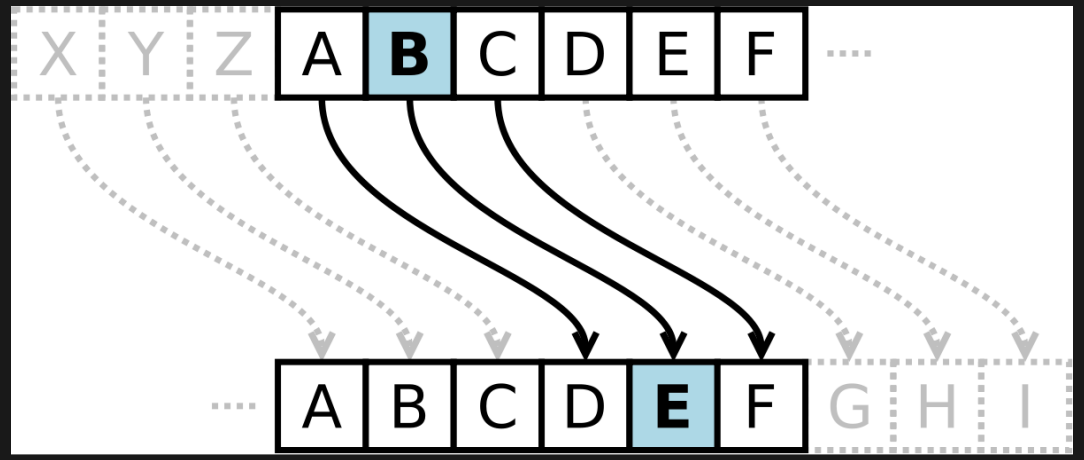
Fonctions de clé secrète



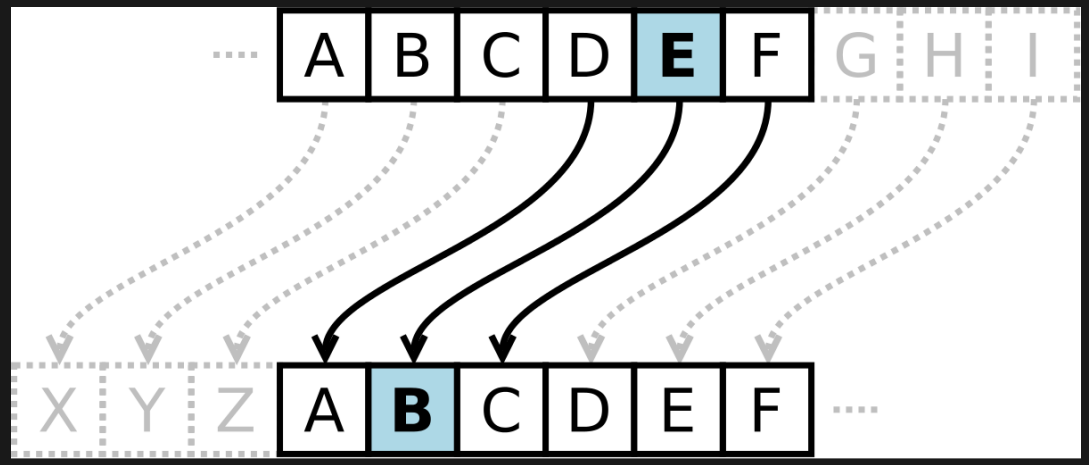
- Utiliser une clé unique pour le cryptage / décryptage.
- Le texte en clair et le texte chiffré ayant la même taille.
- Aussi appelé cryptographie à clé symétrique

SKC: Exemple César

Cryptage



Décryptage



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

SKC: Exemple

Vigenère

Lettre claire	L	E	S	C	O	D	E	S	N	U	C
Clef	T	P	E	T	P	E	T	P	E	T	P
Décalage	20	16	5	20	16	5	20	16	5	20	16
Lettre chiffrée	E	T	W	D	H	X	H	R	N	R	P

	1	2	3	4	5	6	7	8
1	A	B	C	D	E	F	G	H
2	I	J	K	L	M	N	O	P
3	Q	R	S	T	U	V	W	X
4	Y	Z	0	1	2	3	4	5
5	6	7	8	9		!	"	#
6	$	%	&	‘	(	)	*	+
7	,	-	.	/	:	;	<	=
8	>	?	@	[	\	]	^	_

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Carré de Polybe

Message	B	O	N	J	O	U	R
Message Chiffré	12	34	33	24	34	45	42

SKC: Utilisations de la sécurité

Transmission sur un canal non sécurisé

- Le message transmis est crypté par l'expéditeur et peut être déchiffré par le destinataire, avec la même clé
- Empêcher les attaquants d'espionner

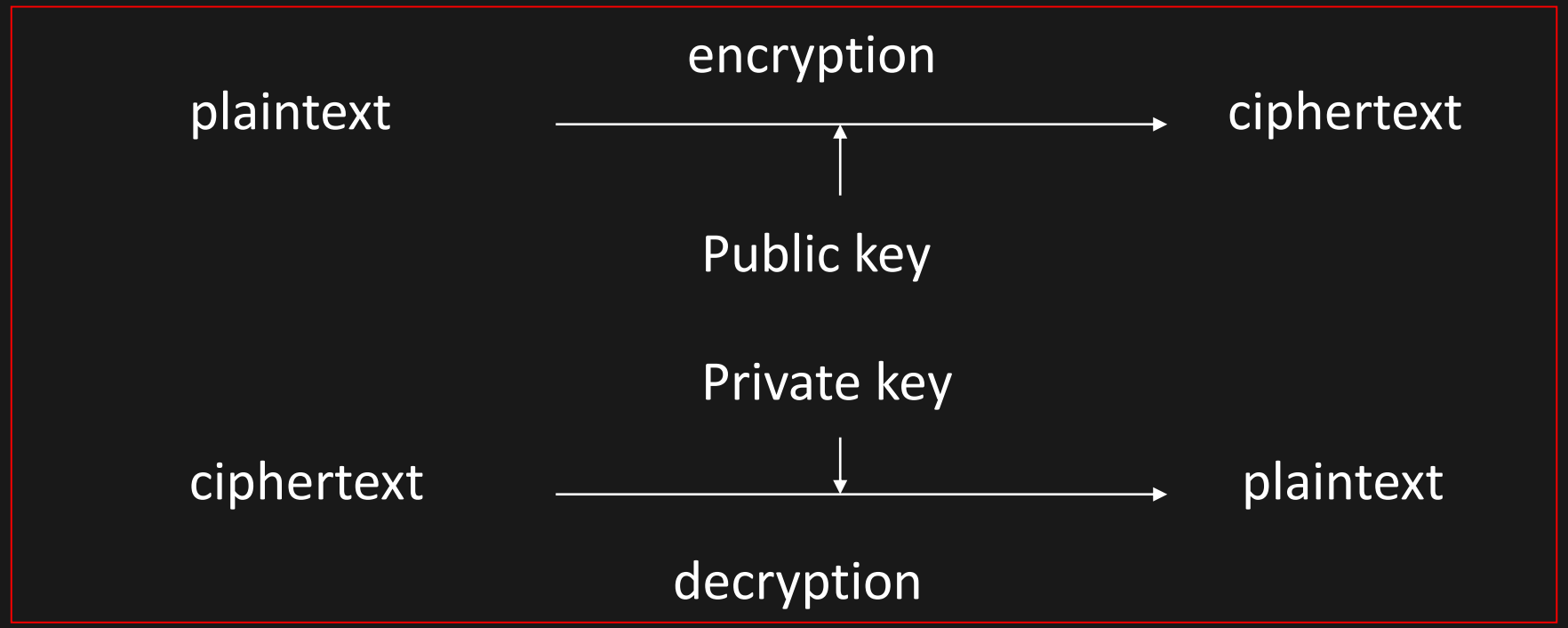
Stockage sécurisé sur des supports non sécurisés

- Les données sont cryptées avant d'être stockées quelque part
- Seules les entités connaissant la clé peuvent la déchiffrer

Algorithmes cryptographiques clé secrète

- DES (Data Encryption Standard)
- 3DES (Triple DES)
- IDEA (International Data Encryption Algorithm)
- AES (Advanced Encryption Standard)

Fonctions de clé publique



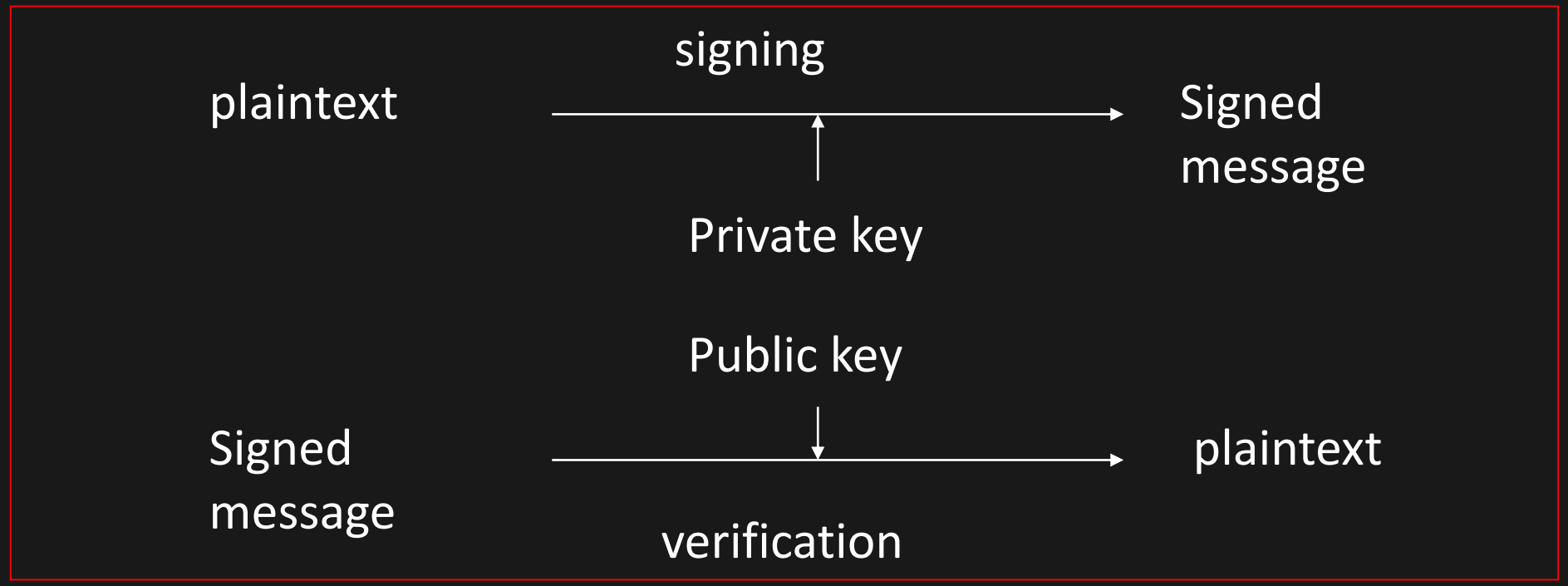
Chaque personne a deux clés

- une clé privée (d): ne doit être révélée à personne
- une clé publique (e): connue de préférence du monde entier

La crypto à clé publique est aussi appelée crypto asymétrique.

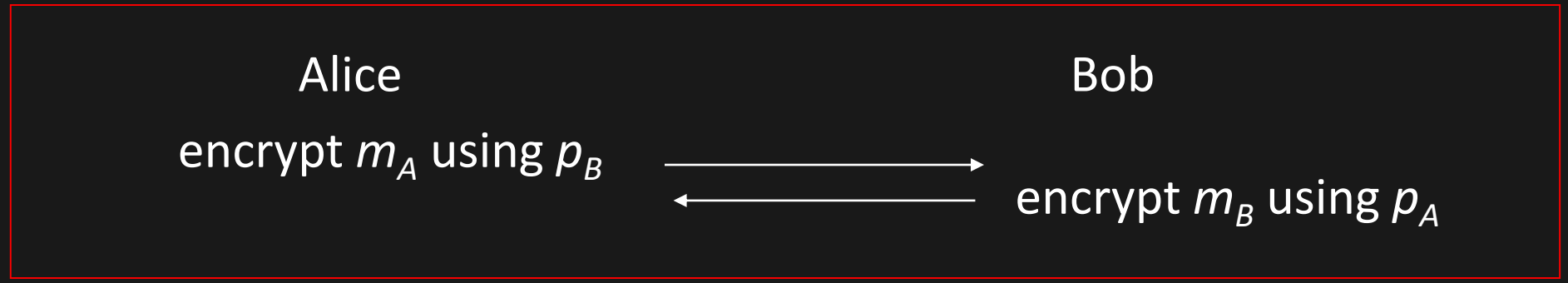
PKC: Utilisations de la sécurité

- Digital Signatures
  - Proving that a message is generated by a particular individual
  - Non-repudiation: the signing individual can not be denied, because only him/her knows the private key.



PKC: Utilisations de la sécurité

- Transmission sur un canal non sécurisé



- Stockage sécurisé sur des supports non sécurisés
  - Les données sont cryptées avec la clé publique de la source, avant d'être stockées quelque part
  - Personne d'autre ne peut le déchiffrer (ne connaissant pas la clé privée de la source de données)

Fonctions de hachage

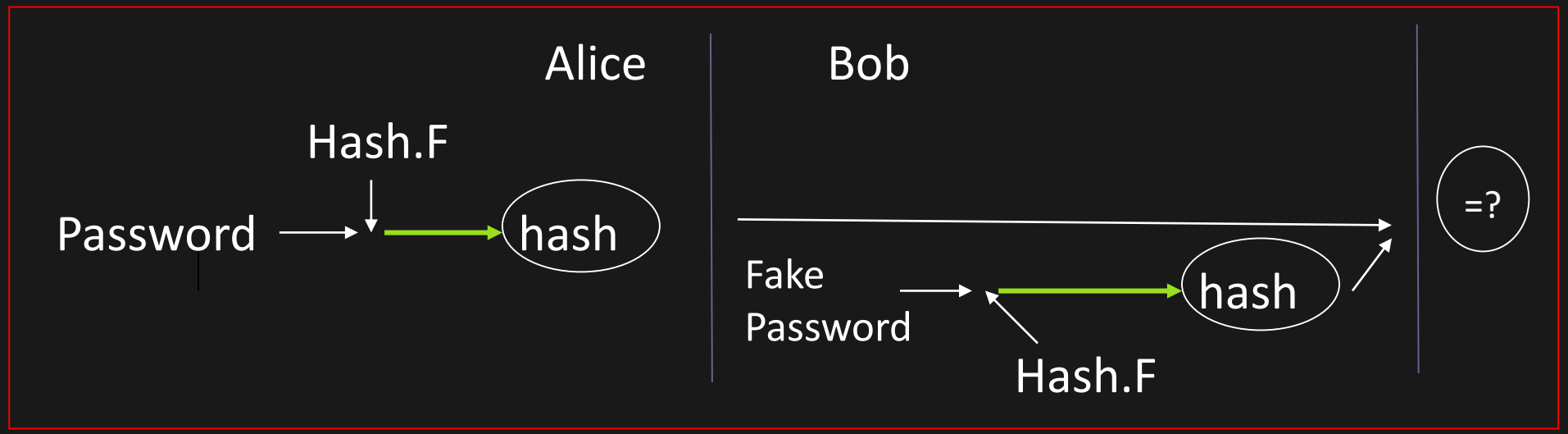
- Fonction de hachage cryptographique
  - Une transformation mathématique qui prend un message de longueur arbitraire et lui calcule un nombre de longueur fixe (courte).

- Propriétés
- (Soit le hash d'un message m qui est h (m))

  - Pour tout m, il est relativement facile de calculer h (m)
  - Étant donné h (m), il n'existe aucun moyen de trouver un m qui se hache en h (m).
  - Il est impossible de calculer informatiquement de trouver deux valeurs qui ont la même chose.

Hachage: utilisations de la sécurité

- Mot de passe haché
  - Le système stocke un hachage du mot de passe



Hachage: utilisations de la sécurité

- Empreinte digitale du message
  - Enregistrez le résumé de message des données sur un magasin de sauvegarde inviolable
  - Recalculez périodiquement le résumé des données pour vous assurer qu'il ne soit pas modifié.

- Sécurité de chargement en aval
  - Utilisation d'une fonction de hachage pour s'assurer qu'un programme de téléchargement n'est pas modifié

- Améliorer l'efficacité de la signature
  - Calculez un résumé de message (en utilisant une fonction de hachage) et signez-le.

Hachage: utilisations de la sécurité

HxD20, English	**installable (can create portable)**	2.1	Maël Hörz	September 27, 2018	2.88 MiB Download per HTTPS	c72ba64ede219eb80bcf44040ef6514b79a19d81
HxD20, French	**installable (can create portable)**	2.1	Le Ch@land	September 27, 2018	2.88 MiB Download per HTTPS	c72ba64ede219eb80bcf44040ef6514b79a19d81

Attaques de sécurité

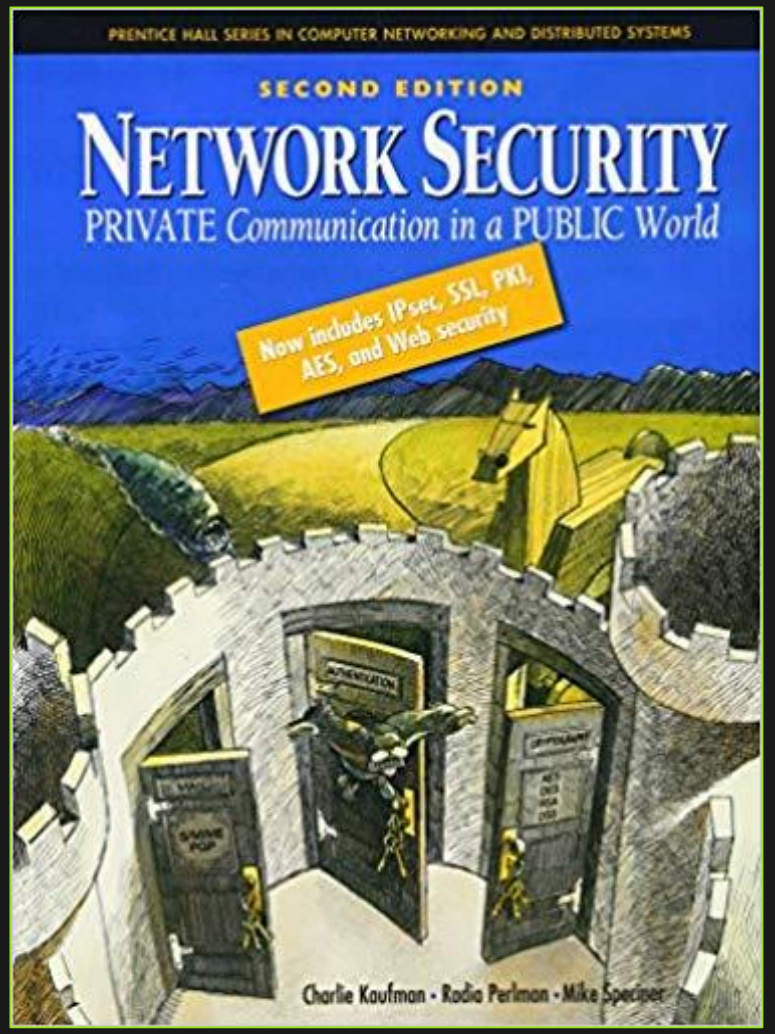
Recherche par force brute

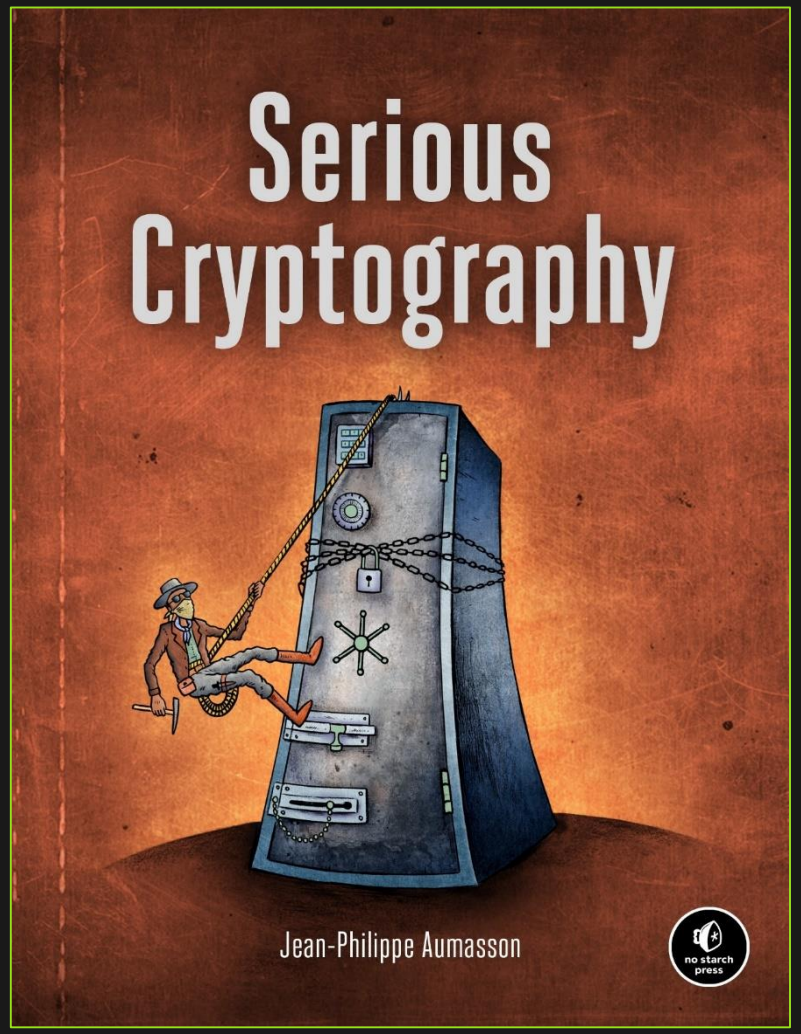
- Supposons qu'on peut reconnaître le texte en clair
- Essayez simplement chaque clé

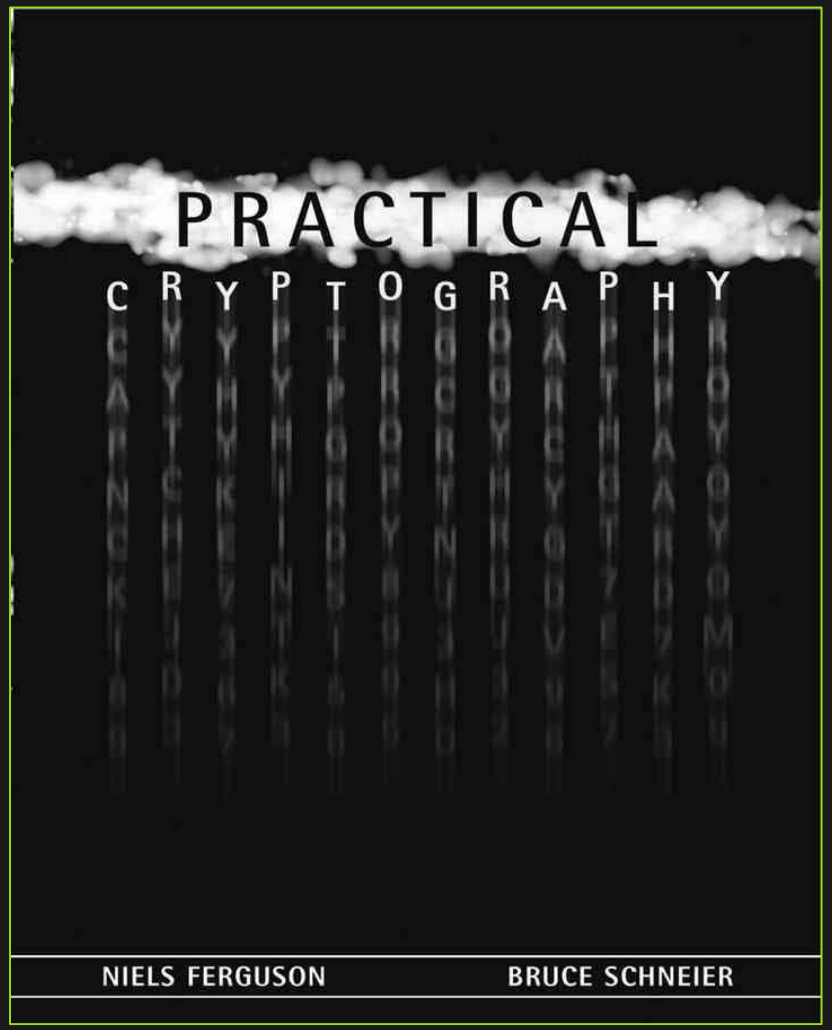
Cryptoanalyse

- Ciphertext seulement
  - Avec plusieurs textes chiffré
- Texte en clair connu
  - Les paires < Ciphertext, Plaintext> sont connues

Cryptography Resources







Plateformes pour pratiquer

- https://www.mysterytwisterc3.org
- https://www.hackerrank.com/
- https://microcorruption.com
- http://overthewire.org
- http://cryptopals.com/

WORKSHOP

APPLIED CRYPTOGRAPHY

PRESENTED BY: MEHDI NACER KERKAR

HACK-INI

OpenSSL

OpenSSL est une bibliothèque C qui implémente les principales opérations cryptographiques telles que le chiffrement symétrique, le chiffrement à clé publique, la signature numérique, les fonctions de hachage, etc.

OpenSSL est disponible pour une grande variété de plates-formes. Le code source peut être téléchargé sur www.openssl.org. Ce didacticiel présente certaines fonctionnalités de base de l'outil de ligne de commande OpenSSL.

OpenSSL

- Une fois l'installation terminée, vous devriez pouvoir vérifier la version.

```
# openssl version
OpenSSL 0.9.7e 25 Oct 2004
```

- OpenSSL a beaucoup de commandes. Voici le moyen de les lister:

```
# openssl list-standard-commands
asn1parse
ca
ciphers
cms
crl
crl2pkcs7
dgst...
```

OpenSSL

Voyons une brève description de chaque commande:

enc Pour chiffrer / déchiffrer à l'aide d'algorithmes à clé secrète. Il est possible de générer à l'aide d'un mot de passe ou directement une clé secrète stockée dans un fichier.
dgst Pour calculer les fonctions de hachage.

password Génération de «mots de passe hachés».
rand Génération de chaînes de bits pseudo-aléatoires.

genrsa Cette commande permet de générer une paire de clé publique / privée pour l'algorithme RSA.
rsa RSA gestion de données.
rsautl Pour chiffrer / déchiffrer ou signer / vérifier la signature avec RSA.

Cryptographie clé secrète

La liste contient l'algorithme base64.

```
# touch password.txt
# echo “this is my passsword" > password.txt
# openssl enc -base64 -in password.txt
MTIzNDU2Nzg5Cg==
```

Chiffrement du texte "this my password" avec l'algorithme AES utilisant le mode CBC et une clé de 256 bits.

```
# touch plain.txt > echo “this is my password” > plain.txt
# openssl enc -aes-256-cbc -in plain.txt -out encrypted.bin
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
```

Comment ça marche!

La clé secrète de 256 bits est calculée à partir du mot de passe. Notez bien sûr que le choix du mot de passe "mehdi" est vraiment INSECURE! Veuillez prendre le temps de choisir un meilleur mot de passe pour protéger votre vie privée! Le fichier de sortie encpass.bin est binaire. Si je veux déchiffrer ce fichier, j'écris:

```
# openssl enc -aes-256-cbc -d -in encpass.bin -pass pass: “mehdi”
This my password
```

Cryptographie clé publique

Pour illustrer la manière dont OpenSSL gère les algorithmes à clé publique, nous allons utiliser le célèbre algorithme RSA. D'autres algorithmes existent bien sûr, mais le principe reste le même.

Génération de clé

Nous devons d'abord générer une paire de clés publique / privée. Dans cet exemple, nous créons une paire de clés RSA de 1024 bits.

```
# openssl genrsa -out key.pem 1024
Generating RSA private key, 1024 bit long modulus
.....++++++
.....++++++
e is 65537 (0x10001)
```

Exemple clé privée

Le fichier généré a une clé publique et une clé privée. De toute évidence, la clé privée doit être conservée dans un endroit sûr ou, mieux, cryptée. Mais avant, regardons le fichier key.pem. La clé privée est codée à l'aide du standard PEM (Privacy Enhanced Email).

```
# cat key.pem
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQDd7ugn0wGokU66w0zIFS1tl8YBuuBRakxTtUgvmqjCyEChR2CV
……
UPFagY0/q9/Y6EIcKCECQQCVlr3jKQ/cehMrd6X2Xv3FszHDh9K0QIAa74LVkB4T
JB58YZPqAdHQQegCKokTA+cMKoKKFFFRjXxBrDw34e+k
-----END RSA PRIVATE KEY-----

# openssl rsa -in key.pem -text -noout
détails de la paire de clés RSA (module, exposant public et privé entre eux)
```

Exemple Clé Publique

Alors maintenant, il est temps de chiffrer la clé privée:

```
# openssl rsa -in key.pem -des3 -out enc-key.pem
writing RSA key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

Alors, extrayons le public du fichier key.pem

```
# openssl rsa -in key.pem -pubout -out pub-key.pem
```

Cryptage

- Nous sommes prêts à effectuer le cryptage ou à produire une signature numérique.

```
# openssl rsautl -encrypt -in plain.txt -inkey pub-key.pem –pubin -out
plain.rsa

# cat plain.rsa
z;*f ü {<D {Ř 1Y  0vz =
:4Io5z* j jJ 7QS z j 6I $5`E u O ◌ؚX
R} -g

# openssl rsautl -decrypt -in plain.rsa -inkey  key.pem -out rsa-dec-plain.txt

# rsa-dec-plain.txt
"this my password"
```

Signatures numériques

La prochaine étape consiste à créer une signature numérique et à la vérifier. Il n’est pas très efficace de signer un gros fichier en utilisant directement un algorithme à clé publique. C'est pourquoi nous calculons d'abord le condensé de l'information à signer.

```
# openssl dgst -sha512 -out plain.sha512 plain.txt
```

Filehash 0.1.dev3

Module Python pour faciliter le calcul de la somme de contrôle ou du hachage d'un fichier. Testé contre Python 2.7, Python 3.6, PyPy 2.7 et PyPy 3.5. Prend actuellement en charge Adler-32, CRC32, MD5, SHA-1, SHA-256 et SHA-512. >> pip install filehash

https://pypi.org/project/filehash/

```
# import os
# from filehash import FileHash
# md5hasher = FileHash('md5')
# md5hasher.hash_file("./testdata/lorem_ipsum.txt")
'72f5d9e3a5fa2f2e591487ae02489388'
# sha1hasher = FileHash('sha1')
# sha1hasher.hash_dir("./testdata", "*.zip") [FileHashResult(filename='lorem_ipsum.zip',
hash='03da86258449317e8834a54cf8c4d5b41e7c7128')]
# sha512hasher = FileHash('sha512')
# os.chdir("./testdata")
# sha512hasher.verify_checksums("./hashes.sha512")
[VerifyHashResult(filename='lorem_ipsum.txt', hashes_match=True),
VerifyHashResult(filename='lorem_ipsum.zip', hashes_match=True)]
```

Chkfilehash Command Line Tool

Un outil de ligne de commande appelé chkfilehash est également inclus dans le package filehash. Voici un exemple d'utilisation de l'outil:

```
# chkfilehash -a sha512 plain.txt
11a89a3bb3561e2e8e874de6d5a7ee1659e3211a7e3c172e4246e18e69734a7b34c9460
2b82329941d11f3a2f74cd1e3682566b4d56f33db2b7bd627cf39ea58 *plain.txt
#chkfilehash -a sha512 plain.txt > plain.sha512

# cat plain.sha512
11a89a3bb3561e2e8e874de6d5a7ee1659e3211a7e3c172e4246e18e69734a7b34c9460
2b82329941d11f3a2f74cd1e3682566b4d56f33db2b7bd627cf39ea58 *plain.txt

#chkfilehash -a sha512 -c plain.sha512
plain.txt: OK
```

Références

- CSS 432: Introduction to Cryptography UWB
- CPSC499 Information Security Management UTC
- https://www.itu.int/en/ITU-D/Cybersecurity/Documents/01-Introduction to Cryptography.pdf
- https://users.dcc.uchile.cl/~pcamacho/tutorial/crypto/openssl/openssl_intro.html

THANKS FOR LISTENING!
HAVE A GOOD **CRACKING** DAY ☺

Website: www.mehdikerkar.keraterra.com

E-mail: mehdi.kerkar@tuta.io

LinkedIn: /in/mehdi-nacer-kerkar/

Twitter: @mehdi_kerkar

SHELLMATES HACK-INI