x) Read and summarize (with some bullet points)

Schneier 2015: Applied Cryptography: Chapter 2 - Protocol Building Blocks, sections

- 2.5 Communications Using Public-Key Cryptography
    - Public-Key Cryptography works well because it's computationally hard to deduce private key from public key and only person with private key can decrypt a message.
    - Public-key cryptography solves the key-management problem with symmetric cryptosystems - practically this mean that the sender and receiver of the encrypted message don't have to share a key in private, but the encryption happens through this public-key exchange.
    - Public key algorithms are used to encrypt keys.
- 2.6 Digital Signatures
    - (Digital) signatures can be forged and possess other problems
    - This chapter
    - Different protocols are demonstrated with examples such as Signing Documents with Symmetric Cryptosystems and an Arbitrator, Digital Signature Trees, Signing Documents with Public-Key Cryptography, Signing Documents and Timestamps
- 2.7 Digital Signatures With Encryption
    - By combining digital signatures and public-key encryption you can have the good sides of both concepts and mitigate some of the problems associated with them.
- 2.8 Random And Pseudo-Random-Sequence Generation
    - Pseudo-Random Sequences, Cryptographically Secure Pseudo-Random Sequences & Real Random Sequences are different ways to use encryption with random numbers, but are not suitable or stable option for encryption.

Rosenbaum 2019: Grokking Bitcoin, Chapter 2. Cryptographic hash functions and digital signatures.

- Hashes are extremely crucial technology in Bitcoin - “trying to learn Bitcoin without knowing what cryptographic hashes are is like trying to learn chemistry without knowing what an atom is.”
- Public-key derivation is a one-way function which means that you can’t create private key from public key.

Karvinen 2023: PGP - Send Encrypted and Signed Message - gpg

- This manual describes in detail how to use gpg in sending encrypted messages.
- The document also includes tips for troubleshooting.

a) Pubkey today

The target environments that I’ve worked with for the last five years as IT consultant have used public key cryptography in many contexts. There has been SSL termination in load balancers and encryption between services. I have also used pgp encryption for encrypting messages a few times.

I was once a IAM-developer for a large company in finance sector. The AM system that was used by the client was IBM’s IBM Security Verify Access also known as ISVA. ISVA consists of many components that offer AM capabilities to large enterprises such as SAML and OIDC -federations and STS chains.

The component that we focus on now is called WebSEAL which is defined by IBM as “a high performance, multi-threaded web server that applies fine-grained security policies to the Security Access Manager protected web object space. Use WebSEAL so that you can manage access to your private and internal resources.”

Imagine that we have an application running on a server and WebSEAL in front of it. WebSEAL allows us to encrypt the traffic between the application server. There are many ways to configure this connection, but on a high level the encryption is established with certifications and is based on public key encryption.

In situations where the certificate gets old or is removed from either WebSEAL or from the application servers the connection doesn't work anymore. This can cause a production take down.

Sources:

https://www.ibm.com/docs/it/sva/10.0.0?topic=configuration-federation-overview

https://www.setgetweb.com/p/isva/wrp_config/concept/con_key_mgmt_overview.htm

b) Messaging.

A keypair I created with pgp:

```
└─$ gpg --fingerprint
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: next trustdb check due at 2025-11-21
/home/kali/.gnupg/pubring.kbx
------------------------------
pub   rsa3072 2023-11-22 [SC] [expires: 2025-11-21]
      0FB0 30FF 09D6 5DF5 6A68  E5FC 6CCC 8EEB 9E85 0216
uid           [ultimate] Jarkko Kela DEMO KEY <jarkko@kela.com>
sub   rsa3072 2023-11-22 [E] [expires: 2025-11-21]
```

Let's export the public key:

```
┌──(kali㉿kali)-[~]
└─$ gpg --export --armor --output samuli.pub
```

Let's create the receiver 'alice':

```
┌──(kali㉿kali)-[~/alice]
└─$ gpg --homedir . --fingerprint
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: next trustdb check due at 2025-11-21
/home/kali/alice/pubring.kbx
------------------------
pub   rsa3072 2023-11-22 [SC] [expires: 2025-11-21]
      98F0 350B 2B41 DAF4 CDE0  9585 CF0C B34D 26B5 EDBC
uid           [ultimate] alice <alice@dontexist.com>
sub   rsa3072 2023-11-22 [E] [expires: 2025-11-21]
```

Alice imports my public key:

```
┌──(kali㉿kali)-[~]
└─$ cp -v samuli.pub alice/
'samuli.pub' → 'alice/samuli.pub'
```

Creaton of secret message ‘helloworld’ and encrypting it:

```
┌──(kali㉿kali)-[~/alice]
└─$ micro message.txt
┌──(kali㉿kali)-[~/alice]
└─$ gpg --homedir . --encrypt --recipient jarkko@kela.com --sign --output encrypted.pgp --armor message.txt
```

Then crash:

```
┌──(kali㉿kali)-[~/alice]
└─$ gpg --decrypt encrypted.pgp
gpg: decrypt_message failed: Unknown system error
```

I run out of time so no change to debug this. My wild guess is that I mixed up the keys someway, but it might be that we’ll never know…

Sources:

https://terokarvinen.com/2023/pgp-encrypt-sign-verify/

c) Other tool

Luckily there are more straightforward (and not as developed) ways to encrypt txt files. My choice for the other tool is OpenSSL which is actually familiar to me from work. I just used AES256:

```
┌──(kali㉿kali)-[~/alice]
└─$ openssl enc -aes-256-cbc -salt -in encryptable.txt -out encryptable.txt.enc
enter AES-256-CBC encryption password:
Verifying - enter AES-256-CBC encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
```

Then decryption:

```
┌──(kali㉿kali)-[~/alice]
└─$ openssl enc -d -aes-256-cbc -salt -in encryptable.txt.enc -out encryptable.txt
enter AES-256-CBC decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
```

I would say that this passphrase based AES encryption is not as efficient and easy to manage than the public key technique with possibility to manage the public and private keys fluently.

d) Eve and Mallory. In many crypto stories.

PGP uses a recipient's public key to encrypt data. When you encrypt a message with someone's public key, only the recipient can decrypt it with a private key. This means that even if Eve intercepts the encrypted message, she cannot decrypt it without access to the private key.

PGP allows the sender to sign a message using their private key. This signature can be verified by anyone who has access to the sender's public key, proving that the message hasn't been tampered with and confirming the identity of the sender.

Source: https://chat.openai.com/

f) Password management. Demonstrate use of a password manager. What kind of attacks take advantage of people not using password managers? (You can use any password manager, some examples include pass and KeePassXC.)

Password managers makes it possible to use 50~ character long passwords and use the passwords with a philosophy that they are just some long strings in the password manager and the target environments database that you just briefly copy paste from one place to another.

Password manager works in a way that you store username / password combinations to it’s file which is encrypted and behind a password. I’m using KeePassXC quite a lot actually and also 1password and keepass. In general I’m just trying not to use password if possible and use some other authentication mechanism like using my mobile phone (google services).

g) Refer to sources.

Done