Theory and Practice of Cryptography

Using Cryptography in Practice

Recap of Week 1

Course materials here: http://saweis.net/crypto.shtml

Lecture 1 is on YouTube:
http://www.youtube.com/watch?v=IzVCrSrZIX8

Why Johnny Can't Encrypt: About 20 minutes to complete

Hash Collision: 77-bits

Two-time Pad Cryptanalyst: (The secret message)

Today's Lecture

This lecture will mostly focus on what not to do.

After this talk, I hope that you:

- Understand some of the common implementation mistakes
- Will want to talk to the security team before using crypto
- Will want to use the tools we've built

Crypto is easy...

Today, there are plenty of comprehensive crypto libraries:

- Java JCE
- OpenSSL
- Crypto++
- GnuCrypto
- Bouncy Castle
- Hundreds more

Any engineer can easily use crypto primitives. Should they?

Crypto is easy to get wrong.

And the consequences can be dire.

Don't roll your own. Ever.

"Nobody ever got fired for using AES."

Avoid obsolete or unscrutinized crypto

- DES: Broken. Don't use.
- 3DES: Still used for legacy apps, don't use for new apps.
- MD5: Don't use for signatures. Okay for HMACs (AFAIK).
- SHA-1: Will soon be questionable for sigs. Okay for HMACs.
- TEA: Some attacks. Not a lot of scrutiny.

Use reasonable key lengths

Good for foreseeable future:

- AES-256 (top secret)
- SHA-256
- DSA/RSA-4096
- ECC-512

Protection until ~2030:

- AES-128 (secret)
- HMAC-SHA1
- DSA/RSA-2048
- ECC-256

Protection until ~2010:

- 3DES
- DSA-SHA-1
- DSA/RSA-1024
- ECC-160

Source: keylength.com

Don't misuse primitives

Think about Microsoft's XBox TEA hash

Use a good random source

Acceptable sources:

- /dev/random
- /dev/urandom
- j.security.SecureRandom
- /util/random/
- c.g.security.random

Weak sources:

- Time of day
- random()
- java.util.Random
- LFSRs
- Something you made up

Caveats:

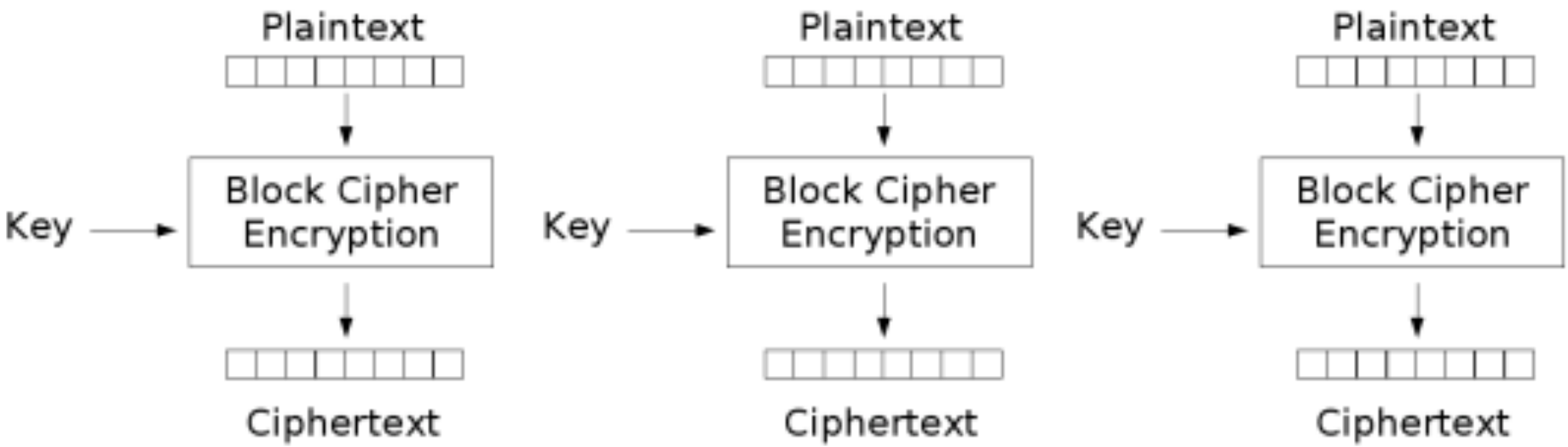
- Blocking issues
- Consider forward security
- Assess your risk

Don't neglect key rotation

"Stuff happens." Be prepared.

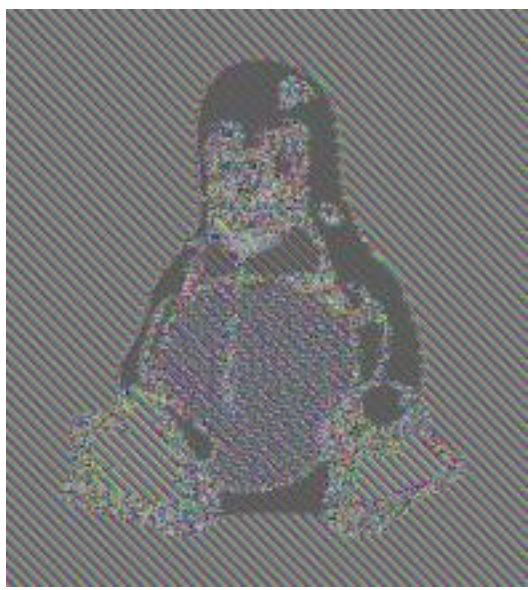
Don't keep keys in code

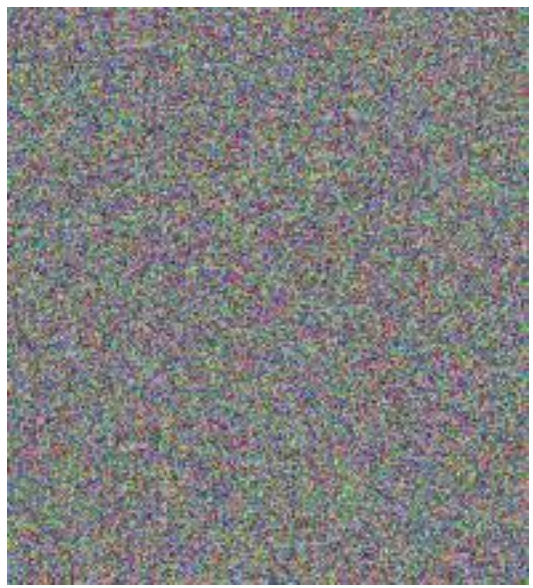
Use the right cipher mode



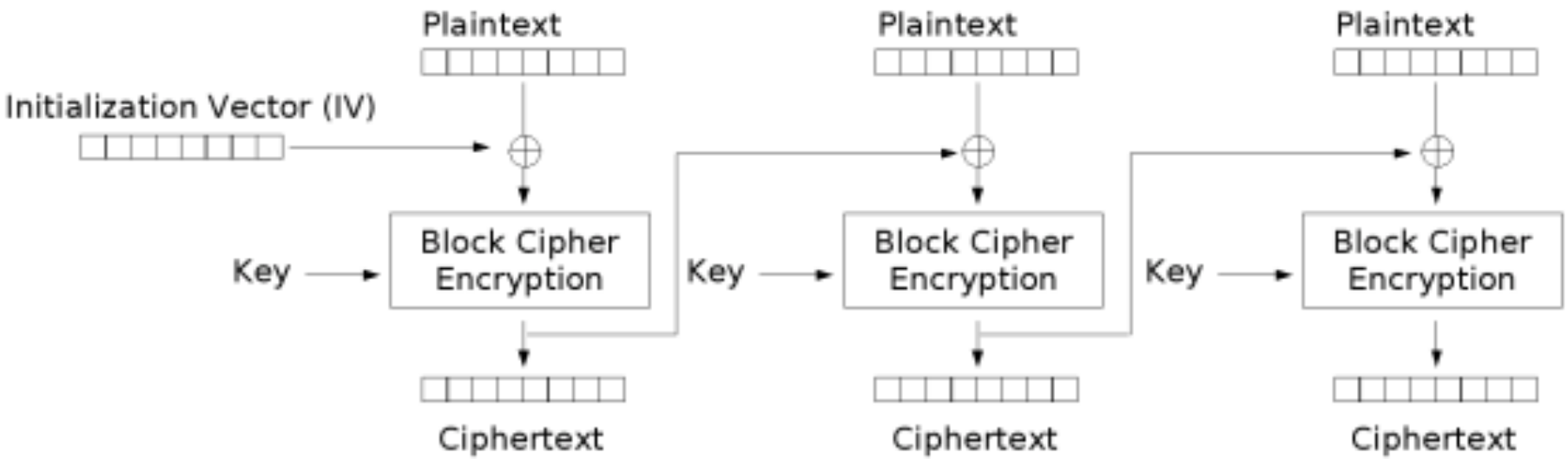
Electronic Codebook (ECB) mode encryption

ECB Misuse



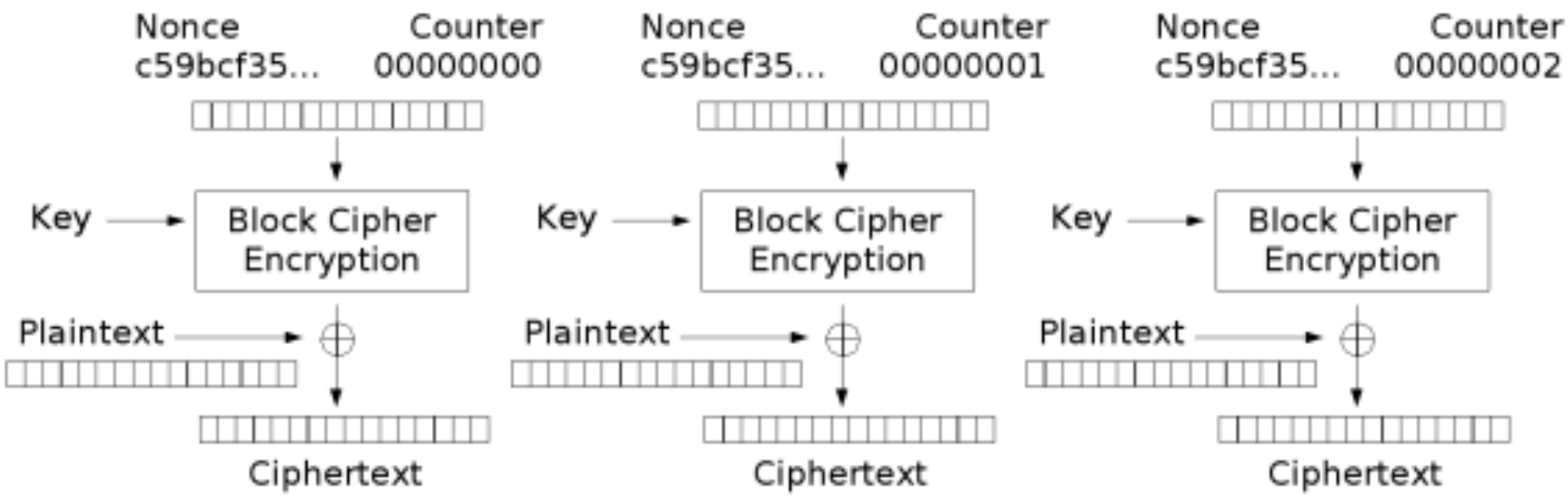


Cipher Modes: CBC



Cipher Block Chaining (CBC) mode encryption

Cipher Modes: CTR



Counter (CTR) mode encryption

Don't reuse IVs

To be safe, use a fresh random IV

Don't reuse stream cipher keys

Same issue as with reusing IVs

Sign all ciphertexts

Consider chosen ciphertext attacks.

Pad and encode properly

Think about WSJ cookies and RSA certificate forgeries

Use nonces properly

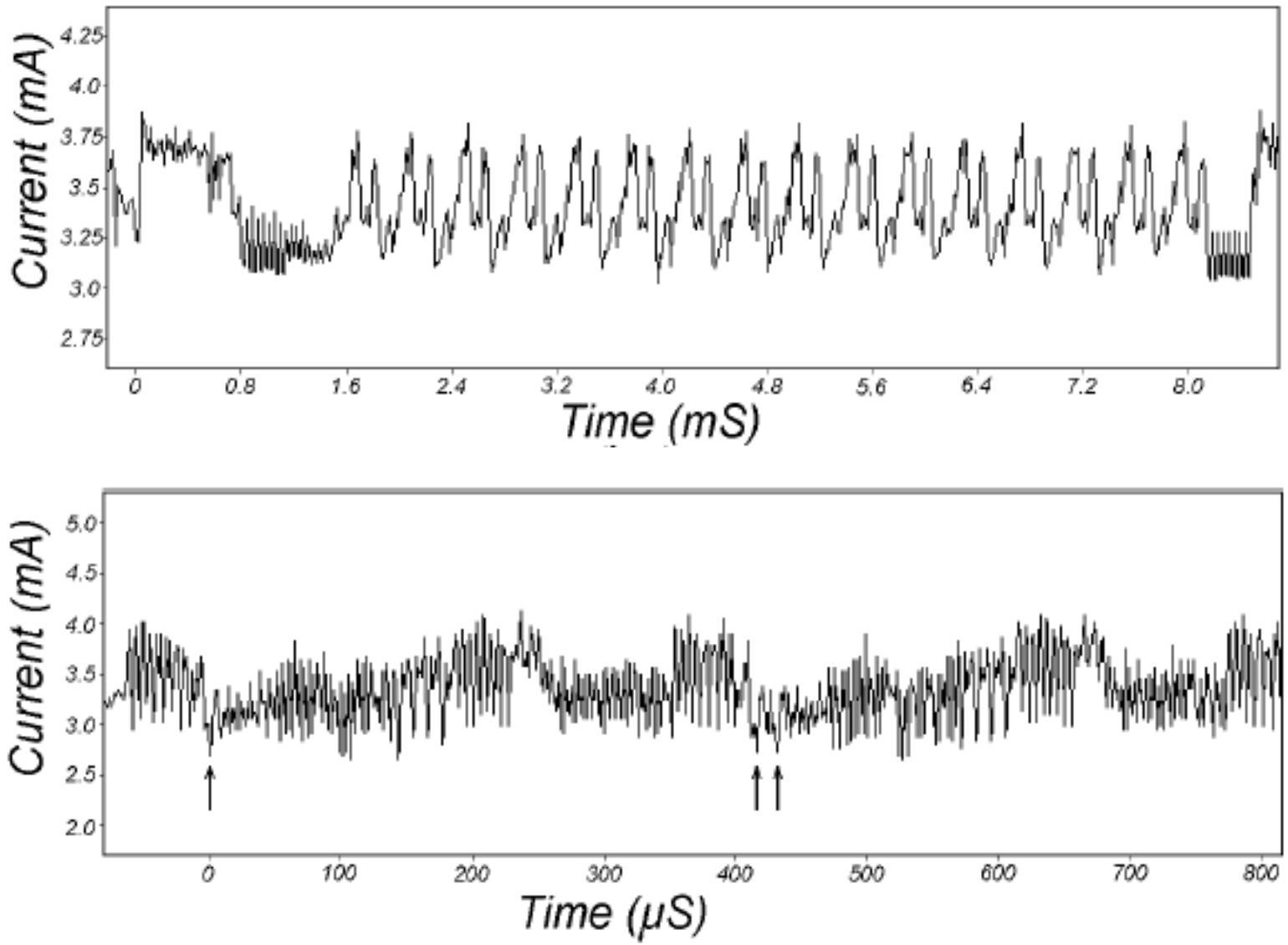
OAEP construction is non-trivial

Use keys properly

HMAC construction is non-trivial

Don't use encrypting keys for signing, or vice versa

Mind the side-channel



Don't fixate on the crypto

Don't put a bank vault door on a tent.

Beware the snake oil salesman

Warning Signs:

1. Pseudo-science: *"Show me the paper"*
2. Based on some new mathematical field*
3. Proprietary technology
4. Clueless claims
5. Huge key lengths
6. Unsubstantiated claims
7. Security proofs*
8. Cracking contests*
9. One-time pads

One Shibboleth: *"Is it semantically secure?"*

Just US$7.50! Going Fast!

"Most file encryptors use methods that rely on the theory of computational security, that is difficulty of key factorisation prevents decryption of the file. But this method may not work forever. It used to be considered that a 56 bit key was unbreakable to brute force attacks, but the government of the USA now requires all Top Secret data to use keys of at least 192 bits. This bar will keep raising as computing power increases. (It is argued by some though that this will never happen due to the laws of physics!) [Product X] is designed to use conventional XOR encryption on keys that are the same size as the file to be encrypted. Furthermore, if you use an unpredictable file that is the same size (or larger) than the original file and you use this file only once, this is known as a "one-time pad" and it is completely unbreakable, even to computers 1000 years from now. This is because there is no algorithm at all, just a big key, and thus there is no pattern to follow once you have decrypted any part of the file. Thus attempts to decrypt it could result in any file, and there would be no way to verify that it has decrypted correctly."