The Road to Secure Cryptography:

Understanding and Preventing Common Misuses

Grazer Linuxtage 2023

Markus Schiffermüller

whoami

- Markus Schiffermüller
- Master Computer Science Student @TU Graz
- Interested in Cryptography
- Like to play CTFs
- Team Captain of LosFuzzys

What are misuses?

What are misuses?

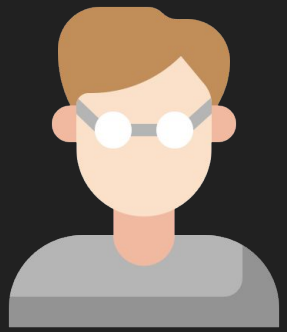
- Incorrect usage of cryptographic algorithms
- Confidentiality & Integrity & Authenticity

```
from Crypto.Cipher import AES

key = b'Sixteen byte key'
cipher = AES.new(key, AES.MODE_EAX)

nonce = cipher.nonce
ciphertext, tag = cipher.encrypt_and_digest(data)
```

What are misuses?



- Cryptography is the bedrock of a secure application
- Cryptographic misuse can happen easily
- Sometimes hard to instantly spot

Your RSA Security Is On Its Last Legs. What's Next?

Vincent Berk Forbes Councils Member

Forbes Technology Council COUNCIL POST | Membership (Fee-based)

AES-256 Encryption Keys Cracked Wirelessly Using Inexpensive Kit

Dutch researchers have demonstrated that secret AES encryption keys can be extracted using recording equipment that costs under 200 euros

Matthew Broersma, June 26, 2017, 10:33 am | Updated on 5 November 2019, 18:40

Invent my own crypto

Invent my own crypto

- Why not invent our own crypto systems?
- *Security by obscurity*

```
# rsa decryption
plaintext = power_mod(ciphertext, key.d, key.n)
# rsa encryption
ciphertext = power_mod(plaintext, key.e, key.n)
```

Never invent your own crypto

> “Anyone, from the most clueless amateur to the best cryptographer, can create an algorithm that he himself can’t break. It's not even hard. What is hard is creating an algorithm that no one else can break, even after years of analysis.”
>
> - Bruce Schneier

Never invent your own crypto

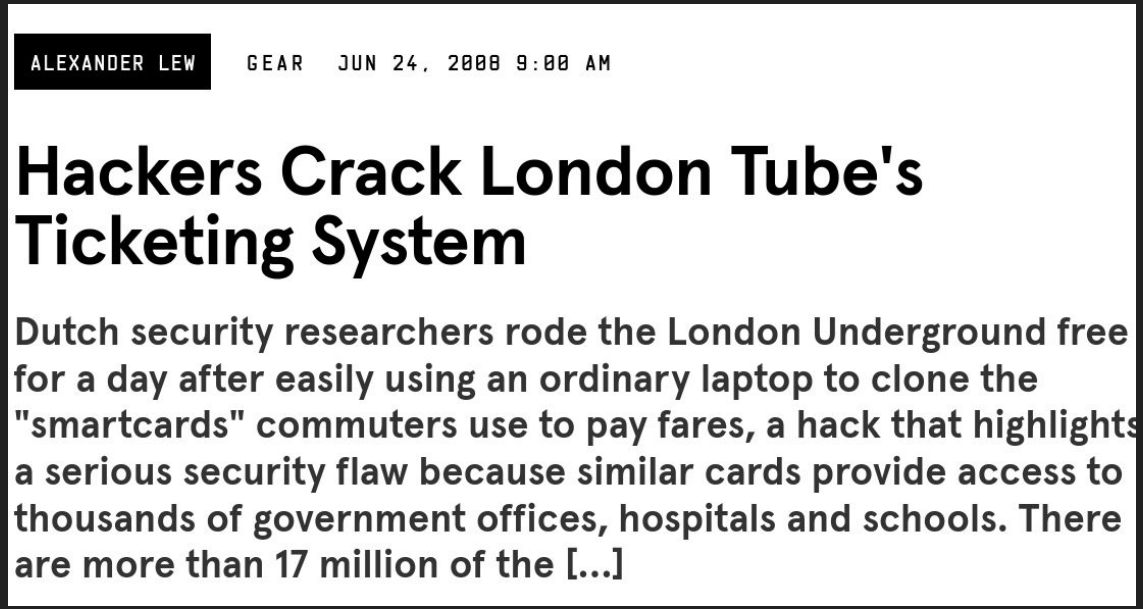
This happened

- 1 billion units sold
- CRYPTO-1
- Own version of a stream cipher
- BROKEN

[1] Garcia, Flavio D., et al. "Dismantling MIFARE Classic." ESORICS. Vol. 5283. 2008.

Never invent your own crypto

- London
- Boston
- South Korea
- Beijing
- Australia
- …



ALEXANDER LEW GEAR JUN 24, 2008 9:00 AM

Hackers Crack London Tube's Ticketing System

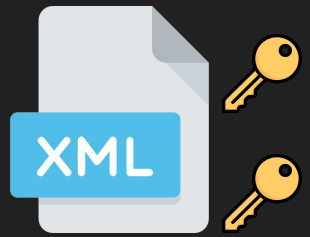
Dutch security researchers rode the London Underground free for a day after easily using an ordinary laptop to clone the "smartcards" commuters use to pay fares, a hack that highlights a serious security flaw because similar cards provide access to thousands of government offices, hospitals and schools. There are more than 17 million of the [...]

[2] https://www.engadget.com/2008-03-14-oyster-cards-vulnerable-to-rfid-hack-lots-of-other-systems-too.html

Hard coded keys

Hard coded keys and credentials

- Just save keys and credentials in strings.xml
- Our application is closed source anyway what should happen?



Don't use hard coded keys

- Retrieving hard coded keys might not be difficult
- Attack can gain access to confidential data
- “It's common to find them [keys] hardcoded into the application as a string or byte array in the code or in an asset file such as strings.xml” - developer.android.com

Don't use hard coded keys

Android application:

- 100K+ Downloads

```
public class Security {
    private static byte[] INITIAL_KEY_VECTOR = {16, 8, 34, ...};
    private static final String KEY = "D:<Q…";
    ...
}
```

Don't use hard coded keys

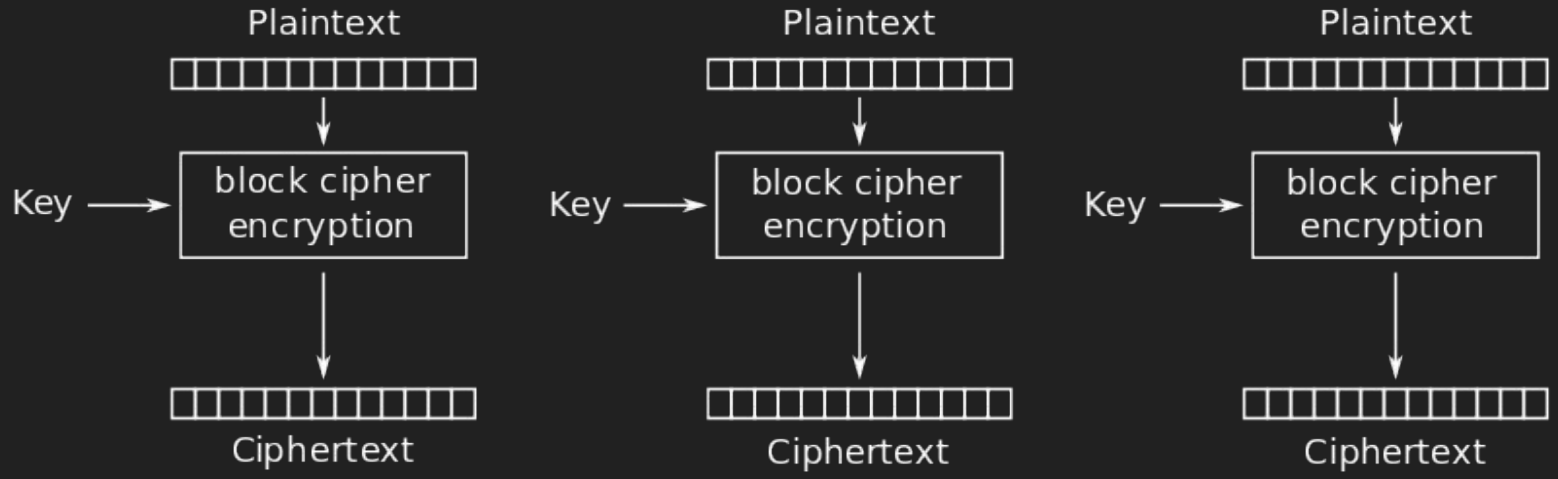
Android application:

- 100K+ Downloads

```
public static String PUBLIC_MODULUS = "bf6d...";
public static String PUBLIC_EXPONENT = "1d";
public static String PRIVATE_EXPONENT = "5c69...";
```

Do I really need nonces?

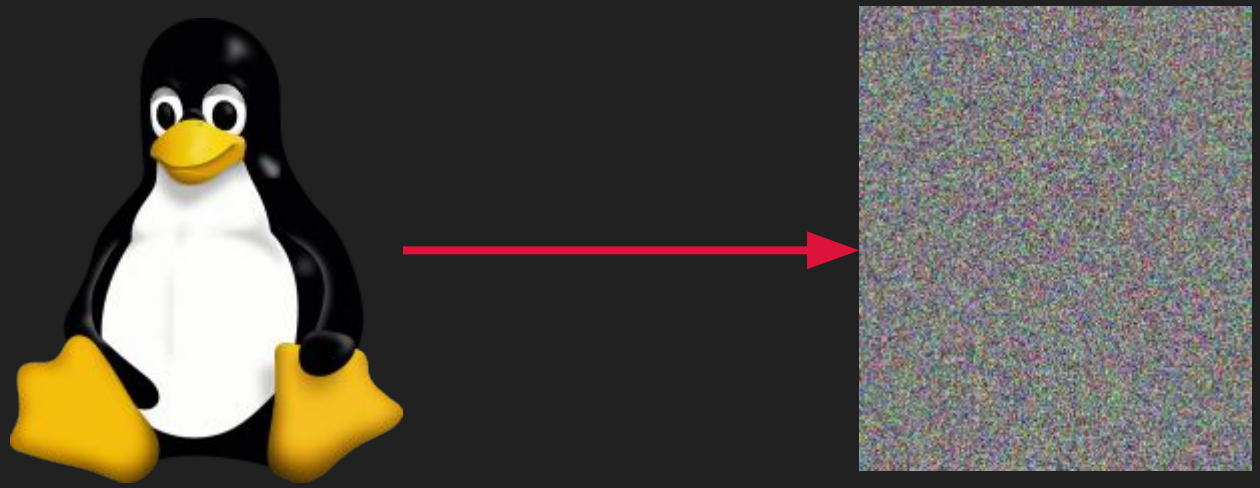
Using no nonces



Electronic Codebook (ECB) mode encryption

Don't use ECB mode

- Encryption using Nonce



Don't use ECB mode

Don't use ECB mode

Reusing nonces

Reusing Nonces

- Generating secure random number is expensive
- Why not just reuse the ones we have?

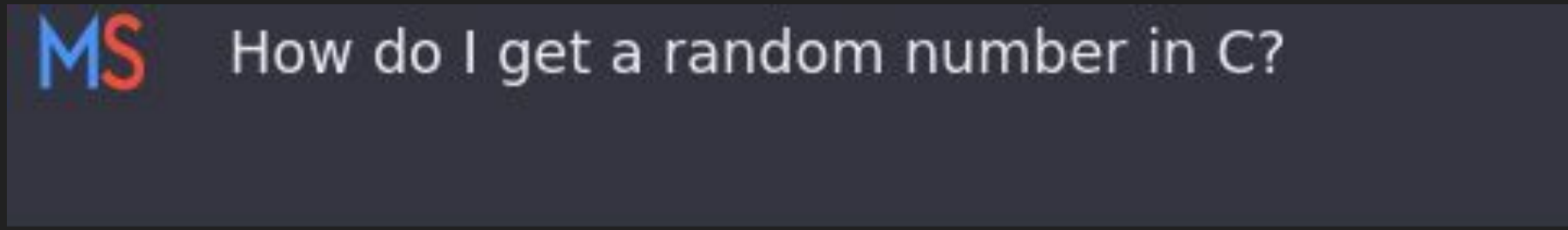
```
//Declaring a global const nonce
const int nonce = 5231903233190;
```

Never reuse Nonces

- Never ever reuse nonces
- Best case: severely reduced security
- Worst case: Entirely broken scheme

Randomness is easy right?

Randomness is easy right?



Randomness is easy right?

MS How do I get a random number in C?

In C, you can get a random number by using the `rand()` function provided by the standard library. Here's an example:

c Copy code

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    // Initialize the random number generator with the current time
    srand(time(NULL));

    // Generate a random number between 0 and 99
    int random_number = rand() % 100;

    printf("Random number: %d\n", random_number);
    return 0;
}
```

In this example, we first include the necessary headers (`stdio.h`, `stdlib.h`, and `time.h`). We then use the `srand()` function to seed the random number generator with the current time. This ensures that we get a different sequence of random numbers each time we run the program.

We then use the `rand()` function to generate a random number between 0 and 99. To get a random number within a specific range, you can use the modulus operator (`%`) to get the remainder of the random number divided by the range size, and then add the minimum value of the range.

```c
#include <time.h>
#include <stdlib.h>
srand(time(NULL));
int r = rand();
```

Randomness is easy right?

Github Copilot:

```
// Seed randoms and generate nonce
srand(time(NULL));
int nonce = rand();
```

Be careful with randomness

- Bad randomness = broken encryption
- Predictable session tokens
- Look for: *Suitable for cryptographic use*
- /dev/urandom

Be careful with randomness

This happened

- CVE-2016-10180
- D-Link DWR-932B router
- Generating the WPS PIN using time seeding

Use “well established” Cryptographic Algorithms

Use “well established” Cryptographic Algorithms

- Bigger Keys = worse performance
- Data encryption standard (DES) for encryption
- MD5 for hashing
- 128-bit should be fine for RSA right?

Never use weak or outdated cryptographic algorithms

- Don't use depreciated algorithms (SHA1, MD5, DES)
- Elliptic Curves greatly improve speed
- Select a secure key size (check NIST recommendation)

[3] https://csrc.nist.gov/Projects/Hash-Functions/NIST-Policy-on-Hash-Functions
[4] https://www.ibm.com/docs/en/zos/2.4.0?topic=2-algorithms-key-sizes

Never use weak or outdated cryptographic algorithms

Evaluation of Cryptography Usage in Android Applications

- At least one cryptographic misuse: 87.8%
- Most common issue: Weak cryptography

[5] Chatzikonstantinou, Alexia, et al. "Evaluation of cryptography usage in android applications." EAI Endorsed Transactions on Security and Safety 3.9 (2016): 83-90.

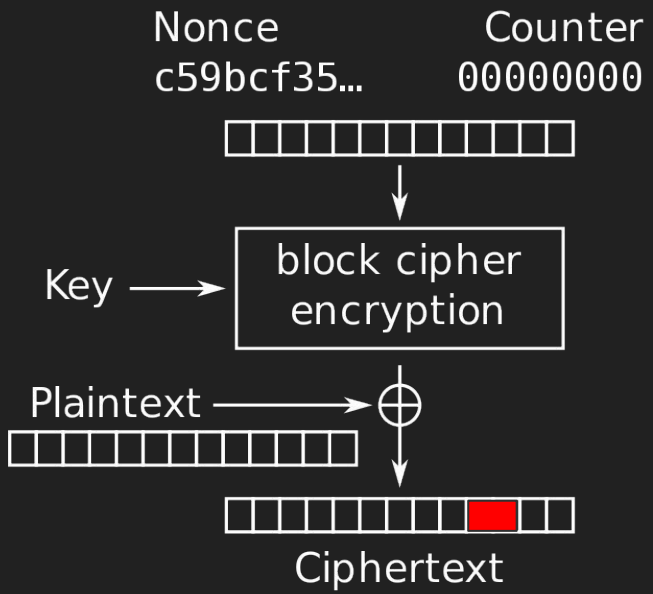
Encryption is sufficient

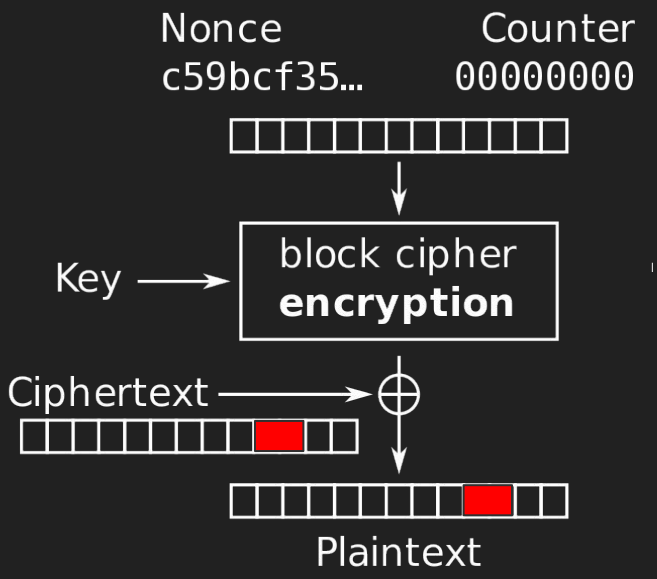
Encryption is sufficient

- Once the messages are securely encrypted we are done right?
- *Confidentiality* is ensured, what more to do?

Authentication encryption is important

AES-CTR





Authenticated encryption is important

- Confidentiality and Integrity are both important
- Ensure integrity of messages
- Use Message Authentication Codes (MAC)
- Use AES-GCM for example

[5]https://www.ibm.com/docs/sk/zos/2.4.0?topic=SSLTBW_2.4.0/com.ibm.tcp.ipsec.ipsec.help.doc/ipsec/IpsecDataOffer_Encrypt.CB_DataOfferAuthentication.html

Recap

Recap

- Never invent your own crypto
- Don't use hard coded keys
- Avoid ECB mode
- Never reuse nonces/IVs
- Be careful when choosing PRNGs
- Make sure the algorithms in place are secure
- Use encryption with authentication

How to prevent

- Use existing libraries
- Be mindful when using crypto libraries

Thank you