Developer Cryptography Mistakes

Top 10 Developer Crypto Mistakes

1. Hard-coded keys
2. Improperly choosing an IV
3. ECB mode of operation
4. Wrong use or misuse of a cryptographic primitive for password storage
5. Passwords are not cryptographic keys
6. MD5 just won't die. And SHA1 needs to go too!
7. Assuming encryption provides message integrity
8. Asymmetric key sizes too small
9. Insecure randomness
10. "Crypto soup"

https://littlemaninmyhead.wordpress.com/2017/04/22/top-10-developer-crypto-mistakes/

Hard Coded Keys

- **Don't hard-code keys into your programs**
- Problem 1: Whoever has the code knows the keys to decrypt the data
  - Should your developers have access to production data? *Probably not…*
- Problem 2: Key management challenge
  - If key is compromised, replacing it requires releasing a new program binary (time consuming)
- Best practice: Never seen by human eyes, never saved to disk

https://littlemaninmyhead.wordpress.com/2017/04/22/top-10-developer-crypto-mistakes/

Improperly Choosing an IV

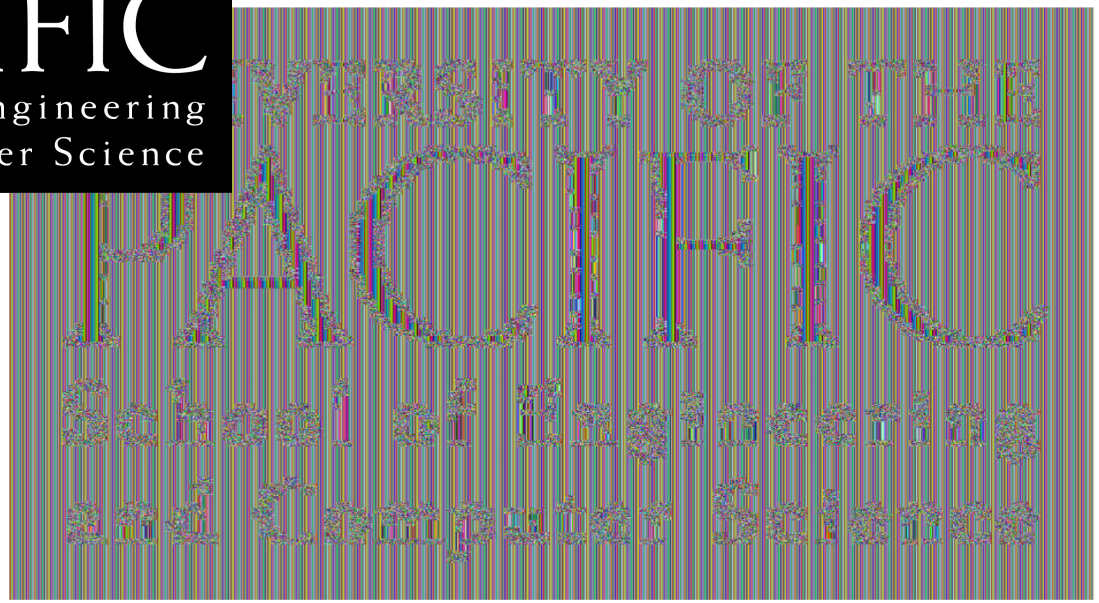
- **Don't hard-code your initialization vector**
  - Should not be all-zero either!
  - Should not be predictable!
- Problem: Constant IV negates cryptography
  - Example: BEAST SSL attack where developers used *ciphertext* from prior block as IV for next block – IV was now predictable!
  - https://blog.cryptographyengineering.com/2011/09/21/brief-diversion-beast-attack-on-tlsssl/
- Best practice: Cryptographically secure random number generator each time

https://littlemaninmyhead.wordpress.com/2017/04/22/top-10-developer-crypto-mistakes/

ECB Mode of Operation

- **Don't use ECB mode! (Electronic Code Book)**

UNIVERSITY OF THE PACIFIC
School of Engineering and Computer Science



https://littlemaninmyhead.wordpress.com/2017/04/22/top-10-developer-crypto-mistakes/

Don’t Hash Passwords!

- **Don’t use a hashing function! (MD5, SHA1, SHA256, …)**
  - Problem: Compute too quickly
- **Don’t use the same salt for each password!**
  - Problem: Identical passwords will map to identical hash values
- https://www.troyhunt.com/our-password-hashing-has-no-clothes/
- Best practice: KDF (bcrpt, scrypt, argon2, …) + random salt for each password

https://littlemaninmyhead.wordpress.com/2017/04/22/top-10-developer-crypto-mistakes/

Passwords Are Not Cryptographic Keys

- **Don't Use Passwords (directly) as a Cryptographic Key**
  - Password:
    - Remembered by humans
    - Arbitrary length
    - **Low entropy / brute force** *(for 90%+ of the passwords)*
  - Key:
    - Used by machines
    - Fixed length
    - Should be **full entropy**
- Best practice: KDF (bcrpt, scrypt, argon2, …) + random salt for each password

https://littlemaninmyhead.wordpress.com/2017/04/22/top-10-developer-crypto-mistakes/

MD5 Just Won't Die. And SHA1 Needs to Go Too!

- **Don't use MD5**
  - Broken due to collisions (2005)
- **Don't use SHA1**
  - Broken due to collisions
  - SHATTERED demonstration (2017)
    (Two PDFs w/identical SHA1 but different content)
- Best practice: SHA2, **SHA3**

https://littlemaninmyhead.wordpress.com/2017/04/22/top-10-developer-crypto-mistakes/

Assuming Encryption Provides Message Integrity

- **Encryption ≠ Authentication**
- Encryption provides *confidentiality*, but an attacker can modify ciphertext
- Modified ciphertext *typically* decodes as garbage, but attacker can try many attempts until garbage causes adverse behavior (bug) in program
- Best practices:
  - Authentication + Encryption: GCM, CCM
  - Authentication-***only***: GMAC, HMAC

https://littlemaninmyhead.wordpress.com/2017/04/22/top-10-developer-crypto-mistakes/

Asymmetric Key Sizes Too Small

- **Don't Use Too Short of Keys!**
- Problem: GPUs are too parallel / brute forcing is possible for short keys
- https://www.keylength.com
- Minimums (2017, IAD-NSA)
  - Symmetric ciphers: 256 bit minimum
  - Elliptic Curve Ciphers: 384 bit minimum
  - Hash: 384 bit minimum *(so no SHA-256)*

https://littlemaninmyhead.wordpress.com/2017/04/22/top-10-developer-crypto-mistakes/

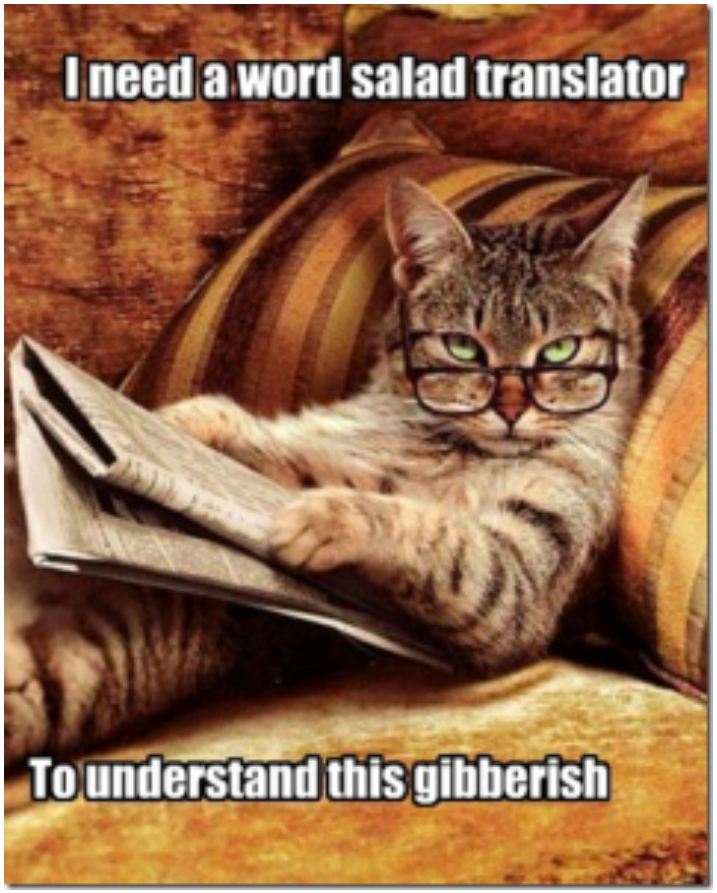
Insecure Randomness

- **Don't Use a Pseudo-Random Generator!**
  - **"Looks Random-ish" ≠ "Random"**

- Best practice: OS-provided mechanism
  - Accept no substitutes!
    (unless you have a fleet of lava lamps)
  - Cryptographically secure random number generator

https://littlemaninmyhead.wordpress.com/2017/04/22/top-10-developer-crypto-mistakes/

Crypto Soup



- **No “Crypto Soup”**
- **No “Buzzword Salad”**
- Don’t mix a bunch of crypto primitives together without a clear goal

Bonus Mistakes!

Insecure By Default

- **Don’t be Insecure by Default**
  - Security should not be optional
  - Security should not be configurable
  - Security should not be an advanced mode described in Chapter 14 of the manual
- **There should be one mode of operation, and it should be secure**
- **Bonus!** Safe from rollback attacks (where threat triggers a rollback to insecure crypto)

Traffic Analysis

- **Traffic analysis is still possible on encrypted data!**
  - Who sent it? Who received it?
  - When was it sent?
  - How much was sent?
  - *Metadata*
- Example: SSH protocol reveals timing between keystrokes when user enters password
  - https://www.usenix.org/legacy/events/sec01/full_papers/song/song.pdf
  - Timing leak – another form of side channel attack

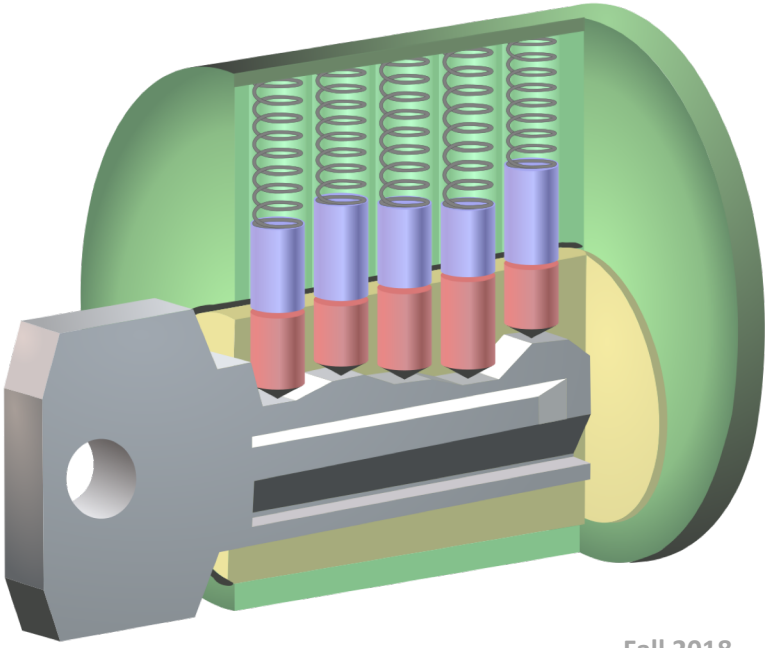
Not Using Best Algorithm Available

- **Use the best algorithm available**
- Many examples where this hasn't happened
  - Microsoft LANMAN password hashing algorithm
    - Crackable in seconds
    - Proprietary algorithm, instead of MD5 which was available at the same time (which at least took hours/days to crack)
  - DVD CSS
    - Proprietary algorithm w/40 bit keys (short!)
    - Easily crackable

http://www.sane.nl/events/sane2000/papers/burnett.pdf

Focusing *Only* On the Crypto

- **Don't focus *only* on the Cryptography!**
- House analogy
  - Front door lock with 4 pins, 10 positions
    - 10^4 combinations for burglar to try
  - Front door lock with 10 pins, 10 positions
    - 10^{10} combinations for burglar
  - ***So we're secure now, right?***



Focusing *Only* On the Crypto

Cleanup

- **Don’t Leave Private Data Around After Use!**
- Examples
  - Did you delete plaintext data after encryption?
  - Are there temporary files with plaintext data on disk? (What about swap memory?)
  - Does your GUI save the password text from the prompt dialog in memory somewhere?
    - *Are you sure the library cleaned up afterwards?*