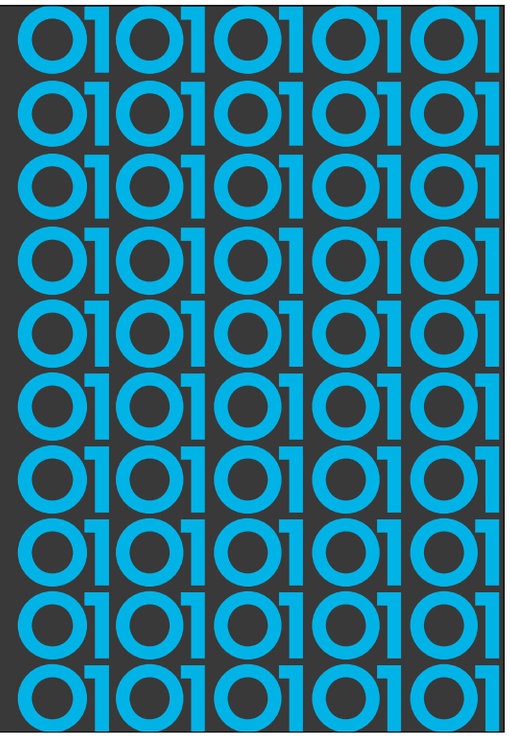
How to store sensitive information in 2020?



Mansi Sheth
CryptoVillage, DefCon
August - 2020

#whoami

VERACODE

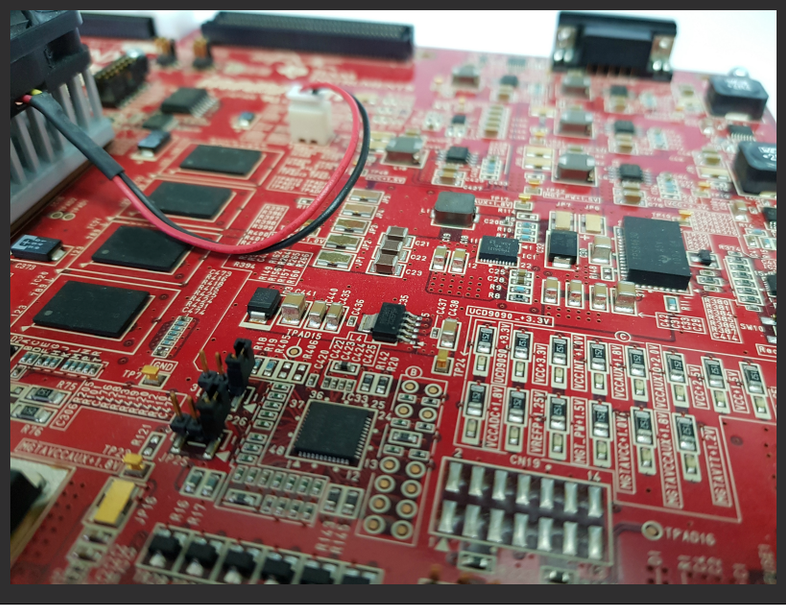
- Principal Security Researcher
- 10+ years experience in Security Research
- Crypto enthusiast

Some insight into data breaches

Thanks! https://haveibeenpwned.com

Total # of domains breached	453
Plain text	62 (13%)
Hashed	90 (20%)
Salted hash	138 (30%)
Key derivation function	67 (15%)
Undisclosed	104 (23%)

Modern Computer Architectures



Key Stretching

Computational Resources

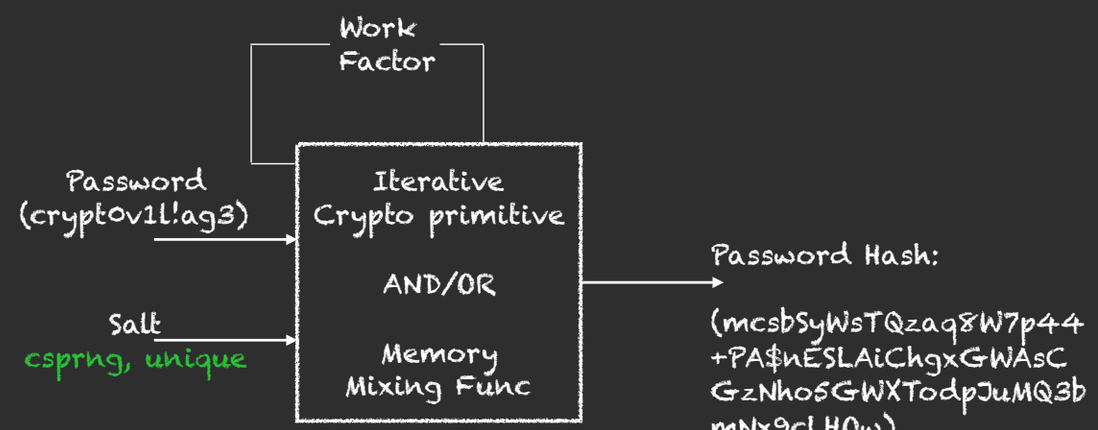
Speed of Password Calculations

Increase Offline Cracking Time

Resistance to Dictionary, rainbow table attacks

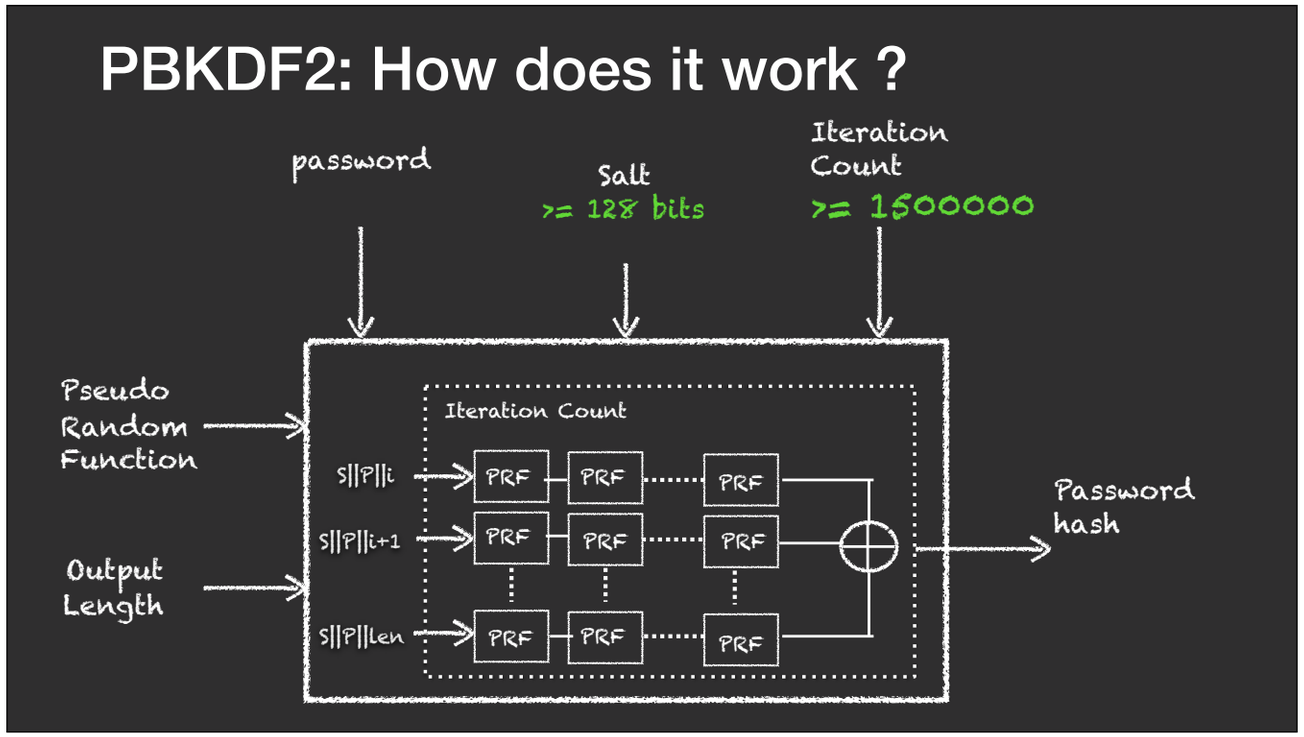
Resistant towards GPUs/ASICs/FPGAs

How to do that ? Using KDFs



PBKDF2: What Is it ?

- Only KDF which is government approved
- Most widely adopted
- Designed for generating keying material



Reference:

1. https://tools.ietf.org/html/rfc8018

PBKDF2: Design Considerations

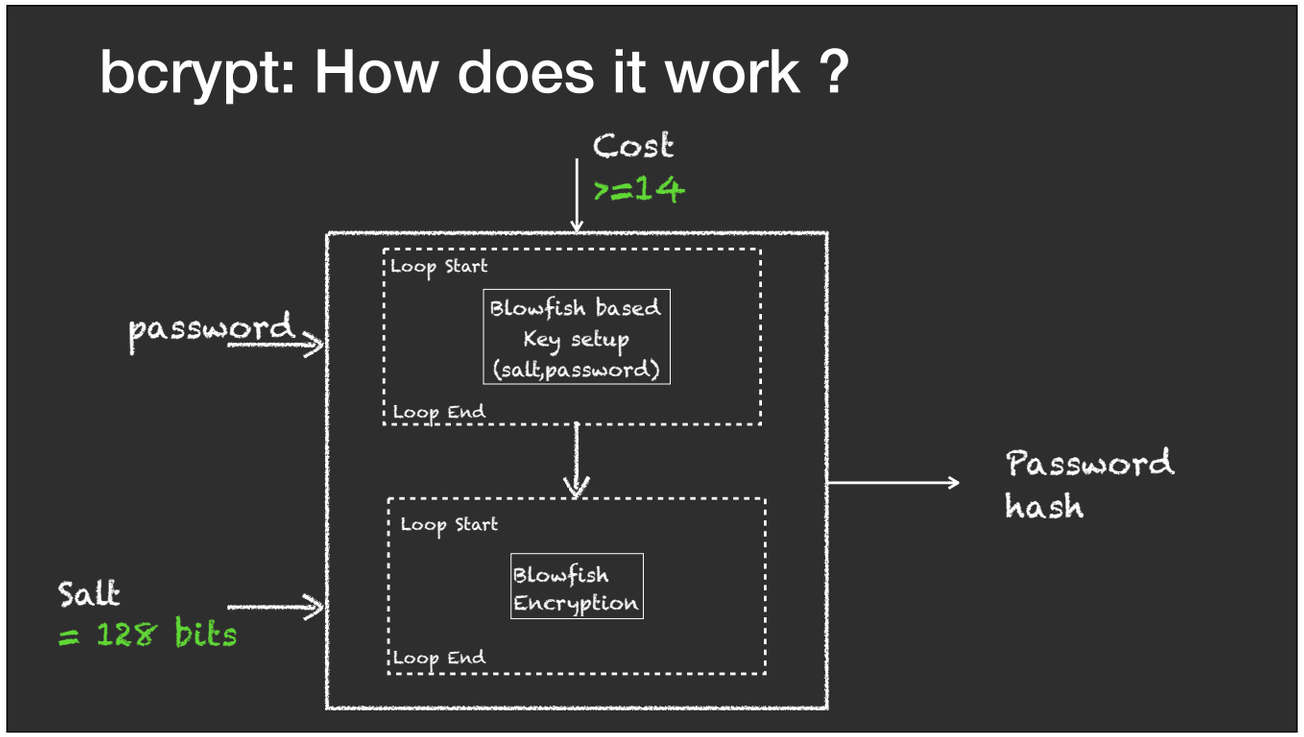
- Output Password Length <= block size of internal hash
- Configurable for high CPU time, but small memory usage
- Not resilient to brute-force attacks, aided by powerful hardware
- Need to stick to government standards ? Use PBKDF2

bcrypt: What Is it ?

- Based on blowfish block cipher
- Internal memory requirement; but not externally tunable
- Geared towards generating keying material

Reference:

https://www.usenix.org/legacy/events/usenix99/provos/provos_html/node5.html



Though setup step needs more memory than pbkdf2, its still computed with a single block of memory…

bcrypt: Design Considerations

- Provides little memory intensive advantage over pbkdf2
- Slightly stronger against brute-forcing, aided by powerful hardware
- Not government standardized, why not use memory hard functions ?

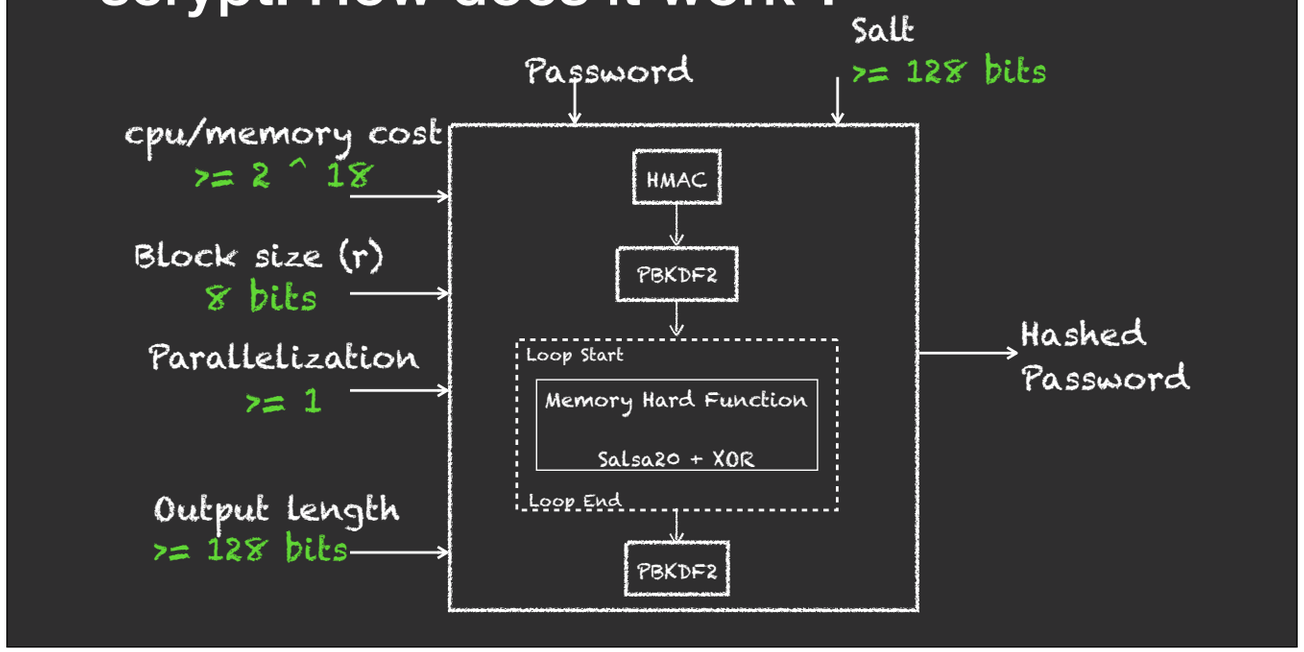
scrypt: What Is it ?

- Earlier generation memory hard function
- Increasing adoption in crypto-currencies
- Focused towards deriving keying material

Reference:

https://www.tarsnap.com/scrypt/scrypt.pdf

scrypt: How does it work ?



scrypt: Design Considerations

- Reduced TMTO attacks
- Only data-dependent mode; side channel attacks still possible
- CPU cost and memory cost can’t be tuned separately
- Lot more crypto; ↑ implementation and cryptanalysis complexity

Reference:

https://libpasta.github.io/technical-details/algorithm-choice/

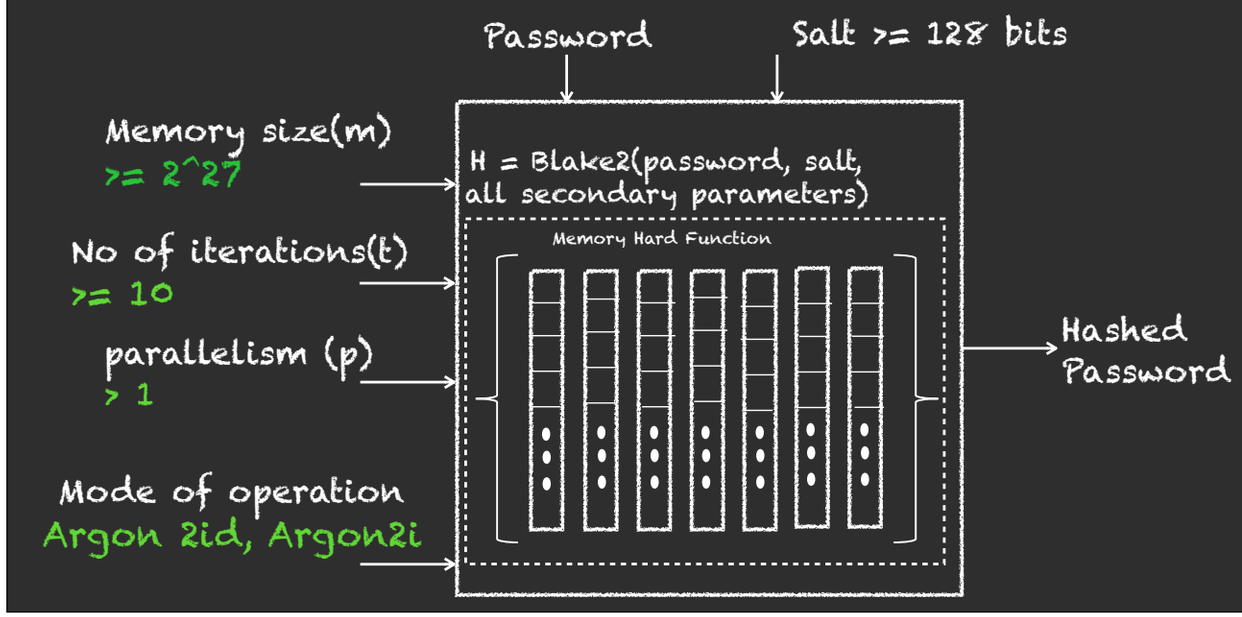
Argon2: What Is it ?

- Newest addition, Password hashing Competition winner - 2017
- GPU/ASICS-resistant; mitigating brute-forcing
- Fewer library implementations; but most modern languages equipped

Reference:

https://password-hashing.net/argon2-specs.pdf

Argon2: How does it work ?



Argon2: Design Considerations

- Attention to tweaking parameter choices
- Argon2i: Increased resistance towards side channel attacks.
- Use Argon2id, best of both worlds 🙂


return
Demo
control
alt

Cost per password guess

On the Economics of Offline Password Cracking

By Jeremiah Blocki, Ben Harsha, Samson Zhou

Efficiently Computing Data-Independent Memory-Hard Functions

By Jeremiah Blocki, Joel Alwen

Cost per password guess

Adaptive Functions :

# of iterations * Cost per calculations

== $\boxed{2 * 10^{-11}}$

Memory Hard Functions (data dependent) :

# of iterations * Cost per calculations

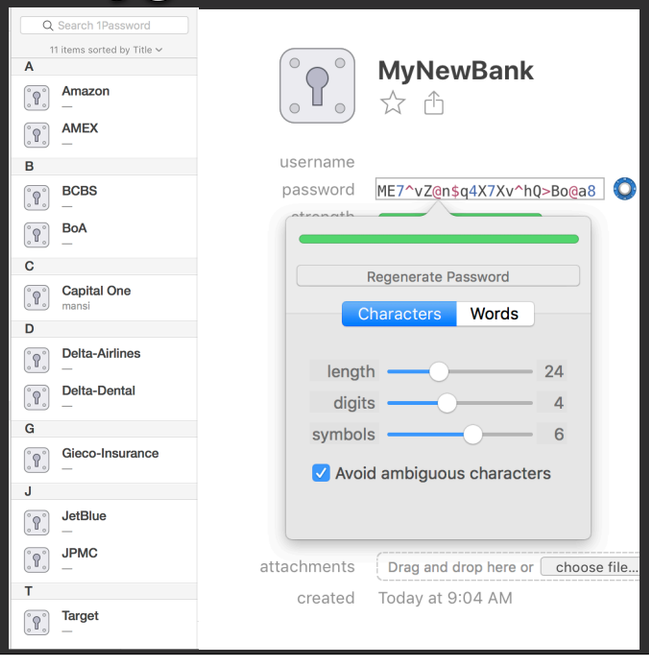
*

(# of iterations)^2 * Cost of memory

== $\boxed{5 * 10^{-4}}$

What data needs to be protected from offline attacks ?

Password Hygiene



In conclusion

- KDF Preference Order: Argon2id, scrypt, bcrypt, pbkdf2
- STAY AWAY: plain text, hashes with/without salt, DIY designs
- Upgrade work factors periodically
- Consider unique work factors, per secret
- Password Hygiene: longer, unique

How to store sensitive information in 2020?


msheth@veracode.com


1MansiS


http://www.veracode.com/blog/author/mansi-sheth


https://github.com/1MansiS

VERACODE