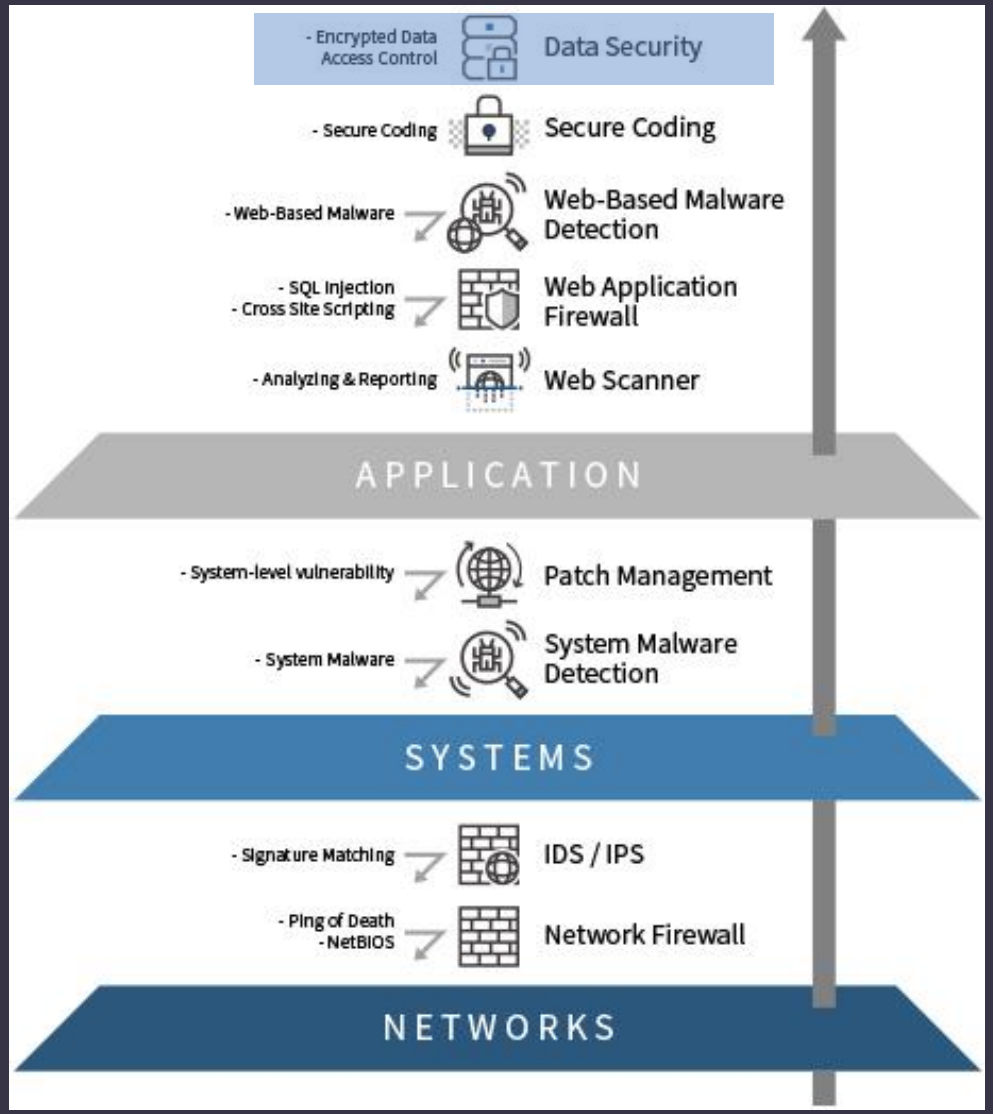
1시간(아마?) 보안

단방향 암호화(HASH)



1. SW 보안?

- 정의 : 안전한 소프트웨어 개발을 위해 소스 코드 등에 존재할 수 있는 잠재적인 보안 취약점을 제거하고, 보안을 고려하여 기능을 설계 및 구현하는 등 소프트웨어 개발 과정에서 지켜야 할 일련의 보안 활동을 말한다.
- 목적 : 기업기밀, 사용자 핵심정보 등 중요 자료를 지킨다.
- 종류 : 사용자인증, 전송구간, 파일, 접근제어 권한, 데이터베이스 등등...

2. 법에서 말하는 정보 암호화 대상

암호화해야 하는 개인정보	정보통신망법	개인정보보호법	적용 암호기술
비밀번호	O	O	해쉬함수
바이오 정보	O	O	
주민등록번호	O	O	
신용카드번호	O	-	
계좌번호	O	-	블록암호
여권번호	-	O	
운전면허번호	-	O	
외국인등록번호	-	O	

3. SW 암호화 기법의 종류

구분	알고리즘 명칭
대칭키 암호 알고리즘	SEED ARIA-128/192/256 AES-128/192/256 Blowfish Camelia-128/192/256 MISTY1 KASUMI 등
공개키 암호 알고리즘	RSA KCDSA(전자서명용) RSAES-OAEP RSAES-PKCS1 등
일방향 암호 알고리즘	SHA-224/256/384/512 Whirlpool 등

예전에는 비밀번호를...

아이디	비밀번호
JJW123	Aboww231
LIM10701	W31kenaaw
SI2_PSJ	skem3qq2131

보안취약점 요약

1. DATABASE 공격 취약
2. 서버공격취약
3. .....
4. 내부유출(관리자) 취약

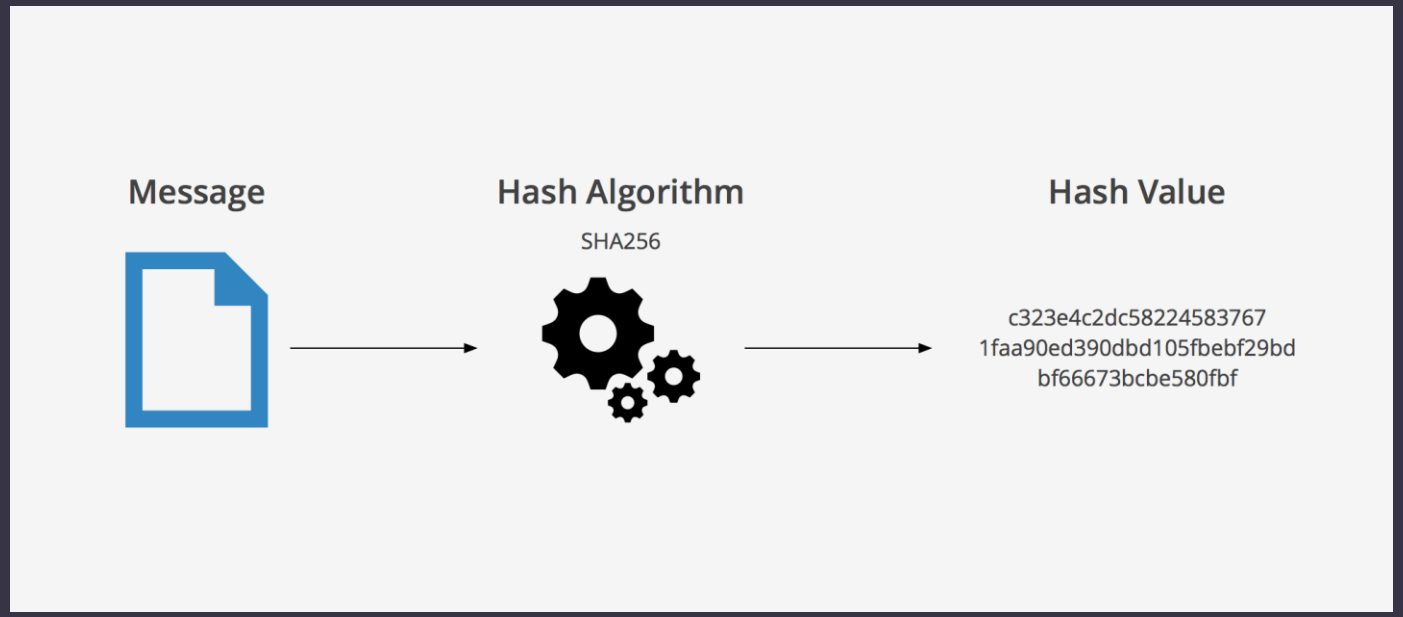
아이디	비밀번호(BLOCK암호)
JJW123	Aboww231->ASNDKFN231A
LIM10701	W31kenaaw->W32sdfnwe#
SI2_PSJ	Skem3qq2131->Weasdkem

보안취약점 요약

1. 키 유출 또는 해킹 시 취약
2. 내부유출(관리자) 취약

4. HASH?

- 목적
  - 데이터의 빠른 접근을 위해 탄생
- 특징
  - 평문을 받아 고정된 길이의 다이제스트 생성
  - 단방향(역상저항성)
  - 충돌 회피
  - 제2역상저항성
- 핫해진 배경...
  - 오래전부터 이론적으로만 존재.
  - 이유는 속도는 빠르나. 많은 메모리를 소모.. 하드웨어가 따라오지못함.
  - 갈수록 하드웨어가 좋아지고 성능이 중요시되며 핫핫



5. 암호화 해시 함수?

- 목적

- 비밀번호 및 평문의 역산되지 않아도 무관한, 자료의 암호화가 필요한경우 사용하면 좋을거같다?

- 전산 내부관리자 또는 개발자가 범죄에 가담하는경우 답이없음. 쌉노답 ㅇㅈ

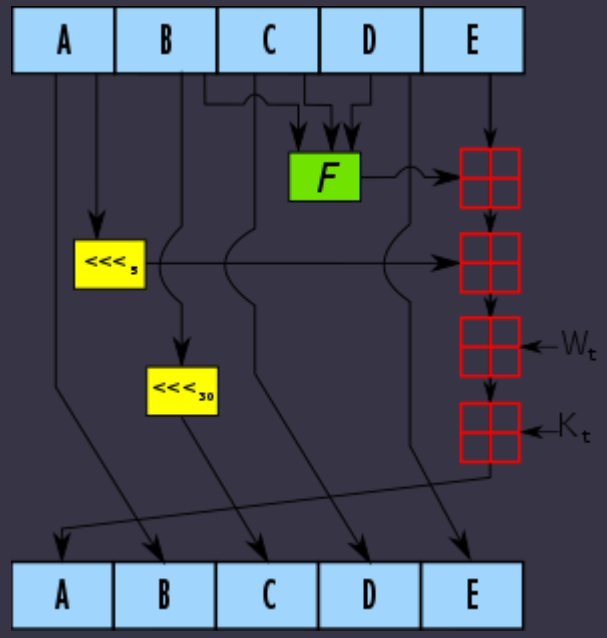
- 자료열람 권한이 있는 자라도, 비밀번호 등 해시된 평문자료는 알수가 없다.

- 이게 블록암호에서 빠지게된 핵심이유. (단방향 특징)

6. 그렇게 등장한 암호화 해시 함수들

알고리즘	출력 비트 수	내부 상태 크기[c 1]	블록 크기	Length size	Word size	라운드 수	공격 가능성 (복잡도:최대 라운드 수)[c 2]		
							충돌	2차 역상	역상
GOST	256	256	256	256	32	256	2^{105}	2^{192}	2^{192}
HAVAL	256/224/192/160/128	256	1,024	64	32	160/128/96	가능		
MD2	128	384	128	-	32	864	$2^{63.3}$		2^{73}
MD4	128	128	512	64	32	48	3	2^{64}	$2^{78.4}$
MD5	128	128	512	64	32	64	$2^{20.96}$		$2^{123.4}$
PANAMA	256	8,736	256	-	32	-	가능		
RadioGatún	Up to 608/1,216 (19 words)	58 words	3 words	-	1–64	-	2^{352} 또는 2^{704}		
RIPEMD	128	128	512	64	32	48	2^{18}		
RIPEMD-128/256	128/256	128/256	512	64	32	64			
RIPEMD-160	160	160	512	64	32	80	2^{51}:48		
RIPEMD-320	320	320	512	64	32	80			
SHA-0	160	160	512	64	32	80	$2^{33.6}$		
SHA-1	160	160	512	64	32	80	2^{51}		
SHA-256/224	256/224	256	512	64	32	64	$2^{28.5}$:24		$2^{248.4}$:42
SHA-512/384	512/384	512	1,024	128	64	80	$2^{32.5}$:24		$2^{494.6}$:42
Tiger(2)-192/160/128	192/160/128	192	512	64	64	24	2^{62}:19		$2^{184.3}$
WHIRLPOOL	512	512	512	256	8	10	2^{120}:4.5		

7. 완벽한 보안? 창방패



SHA-1

- 완벽?

- SHA-1는 NSA(미국 국가안보국)에서 만들었음. 하지만

- SHA-1의 알고리즘은 이미 보안취약점이 발견됨..

- 보안취약점이 하나라도 발견된경우 해당 암호화 해쉬는 더 이상 사용하기 힘듬.. (만들어낼수 있게되기때문에)

- 하여, 해커와시간과의 싸움 현재진행중.

- 현재 미국권고안은 SHA-2 (SHA-256/224, SHA-512/384)

- MD5, SHA-1 은 알고리즘 취약점이 발견되어 권장되지않음.

8. 해커의 입장에선..

- 비밀번호를 알아내고싶다.....

- 예상경로 : 패킷(전송계층), 디비 서버 침투, 내부관리자 유출 등등....
- 알고리즘 취약점 연구 : 오지게 천재아니면 X
- 해쉬값의 단방향 특징과 빠른 속도를 이용한 'RAINBOW TABLE' 공격

9. SW개발자의 입장은..

- 비밀번호 등 중요 정보가 뚫리는걸 막아야한다.....
- 어떻게하면 좀더 뚫리는걸 늦추거나 대량의 사용자 정보가 밝혀지는걸 막을수있나?
- 소수는 몬막더라도...

- 로그인 인증시간 일부러 느리게~
- **SALT**
- Key **Stretching**

10. SALT?

- 시나리오 : 사용자가 회원가입 시 비밀번호를 "18351123" 이라고 입력 후 진행.
  - 특수문자, 알파벳 등은 회원가입시 체크하지않는 사이트라고 가정

- HASH("18351123") → absei12341nansefn323 => 매우 취약

- 소금 치기 : (#$*!!A3#@!$#$#@**#!L
- HASH("18351123"+ (#$*!!A3#@!$#$#@**#!L) → sdjei231234j23mmfms8 => salt1
- OR
- HASH((#$*!!A3#@!$#$#@**#!L +
  HASH("18351123"+ (#$*!!A3#@!$#$#@**#!L)) --> hkjlkrmw23fm4n45ijma => salt2
- 해쉬 결과값을 이용하여 대조하여 해킹하는데에 시간, 비용이 기하급수적으로 늘어난다.

10. SALT?

User	Password
Stephen	auhsoJ
Lisa	hsifdrowS
James	1010NO1Z
Harry	sinocarD tupaC
Sarah	auhsoJ

User	Password Hash
Stephen	39e717cd3f5c4be78d97090c69f4e655
Lisa	f567c40623df407ba980bfad6dff5982
James	711f1f88006a48859616c3a5cbcc0377
Harry	fb74376102a049b9a7c5529784763c53
Sarah	39e717cd3f5c4be78d97090c69f4e655

User	Random Salt	Password Hash
Stephen	06917d7ed65c466fa180a6fb62313ab9	b65578786e544b6da70c3a9856cdb750
Lisa	51f2e43105164729bb46e7f20091adf8	2964e639aa7d457c8ec0358756cbffd9
James	fea659115b7541479c1f956a59f7ad2f	dd9e4cd20f134dda87f6ac771c48616f
Harry	30ebf72072134f1bb40faa8949db6e85	204767673a8d4fa9a7542ebc3eceb3a2
Sarah	711f51082ea84d949f6e3efecf29f270	e3afb27d59a34782b6b4baa0c37e2958

Figure 1. Password and Hash Tables

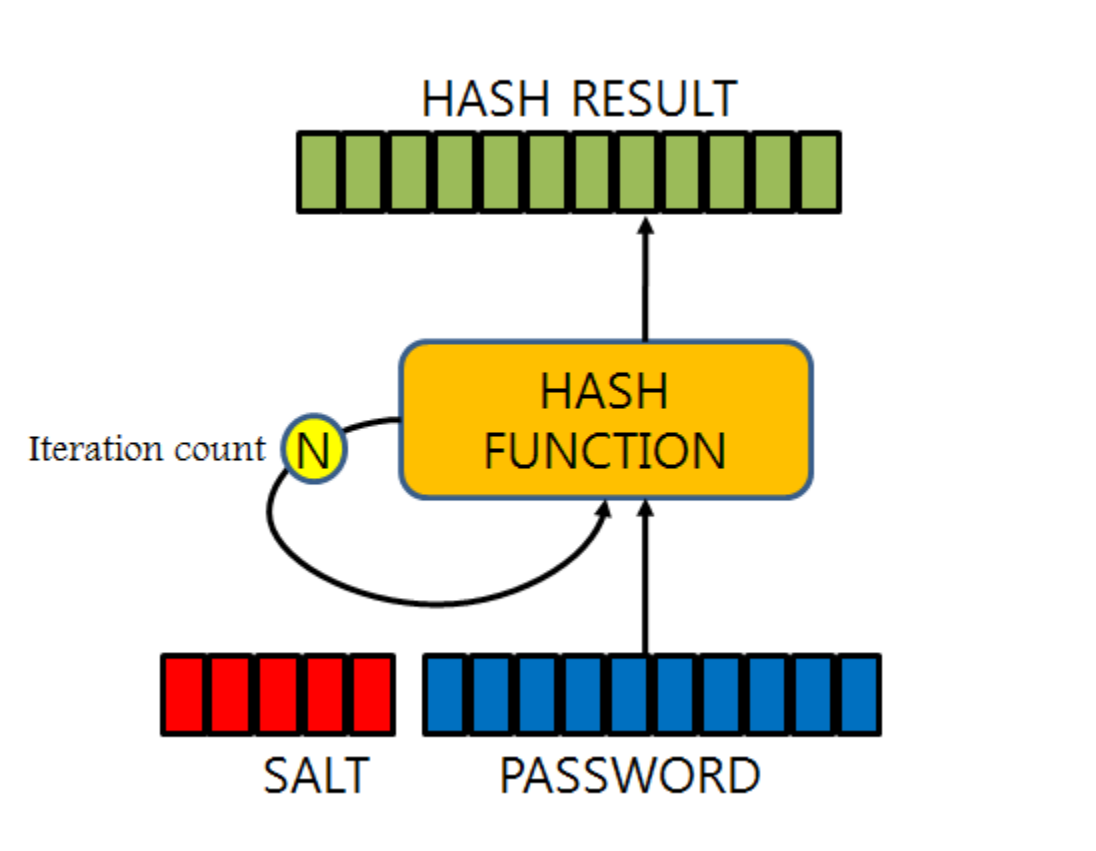
11. Stretching?

- 시나리오 : 사용자가 회원가입 시 비밀번호를 "18351123" 이라고 입력 후 진행.
  - 특수문자, 알파벳 등은 회원가입시 체크하지않는 사이트라고 가정

- HASH("18351123") → absei12341nansefn323 => 매우 취약

- Stretching count : 100
- HASH("18351123") → sdjei231234j23mmfms8 1번
- HASH("sdjei231234j23mmfms8") → hkjlkrmw23fm4n45ijma 2번
- HASH("hkjlkrmw23fm4n45ijma") → elrk23l56kl5mkgmerkk2 3번
- Loop 100… → cdimc34okfmor34o (저장될 비밀번호 HASH VALUE)

11. **Stretching?**



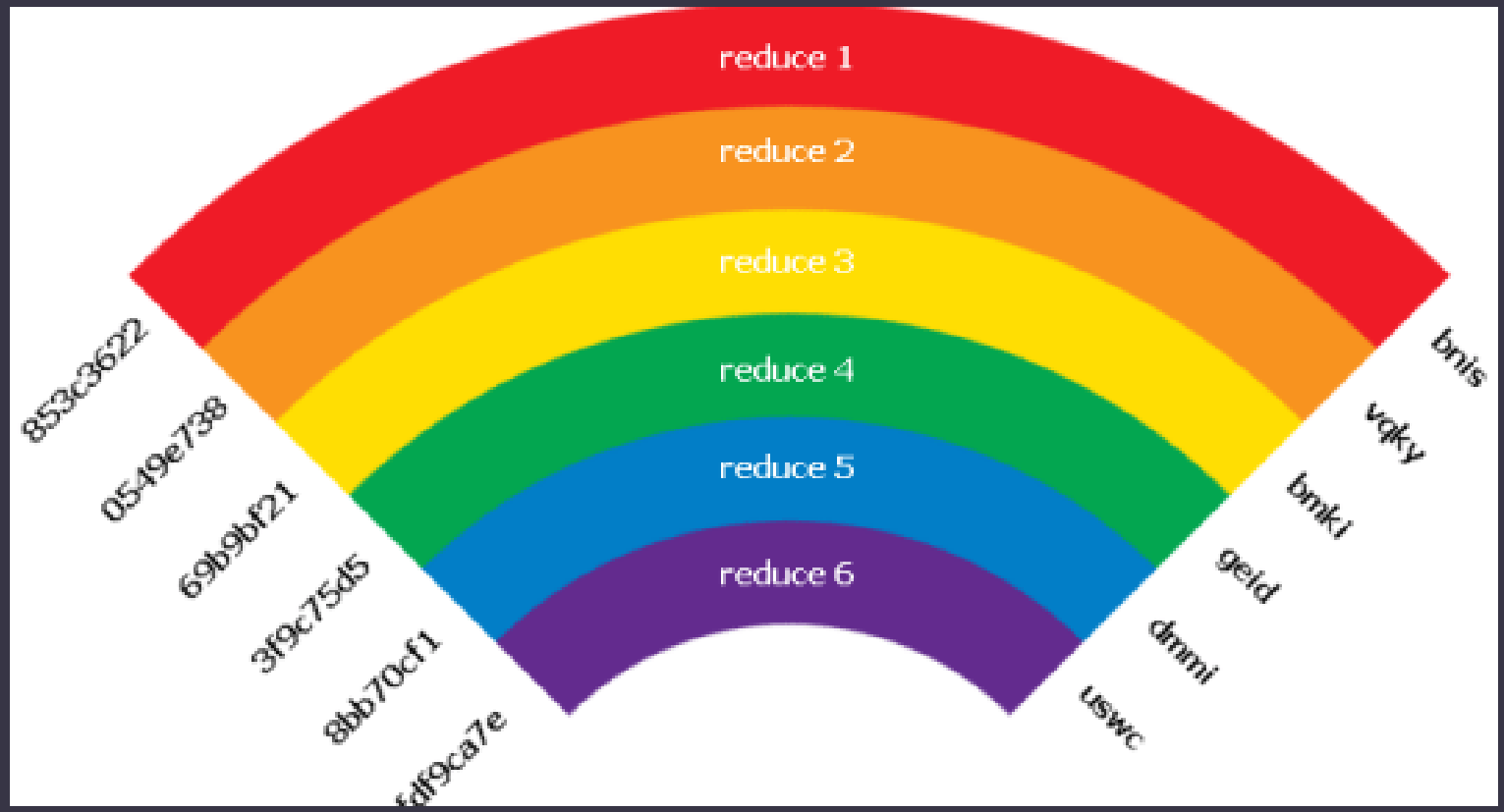
12. RAINBOW TABLE ?

- 정의 : 해시 함수를 사용하여 변환 가능한 모든 해시 값을 저장시켜 놓은 표
- 보통 해시 함수를 이용하여 저장된 비밀번호로 부터 원래의 비밀번호를 추출해 내는데 사용
- 향상된 하드웨어 (CPU, GPU) 연산성능으로 인해, MD5 단순 대입 비교는 1초당 56억개 다이제스트 대입 가능
- 100GB, 1TB, 1000TB 등에 모든 경우의 수의 헤시값들을 저장해놓은 후 단순 대입 비교 하여 해킹 가능.
- 보안취약점이 많은 회사의 경우 효과적(매우효과적)으로 공략가능하고 사용하기 쉬움

- 단 SALT, 또는 스트레칭의 유무에 따라 효과가 기하급수적으로 낮아지는 특성

  (SALT 유무에의해서 1초당 56억번 → 1초당 10번 등)

12. RAINBOW TABLE ?



13. 해킹 실습

- 대상 프로젝트 : 인사급여시스템
- 비밀번호 암호화 : MD5 (NO SALT, NO STRETCHING)
- DATABASE : DB2

- RAINBOW TABLE : https://freerainbowtables.com/
- 준비한 RAINBOW TABLE : MD5용 소문자,숫자,공백 커버되는 1-8 length
- HACKING SW : rcracki_mt

14. 그래서 어떻게하면되지?

- Adaptive Key Derivation Functions

- PBKDF2 (NIST(National Institute of Standards and Technology, 미국표준기술연구소)에 의해서 승인된 알고리즘)

  * DIGEST = PBKDF2(PRF, Password, Salt, c, DLen)

  * PRF: 난수(예: HMAC-SHA1)

  * Password: 패스워드

  * Salt: 암호학 솔트

  * c: 원하는 iteration 반복 수
  * DLen: 원하는 다이제스트 길이

- bcrypt

- scrypt (차이점중 예를들면 다이제스트가 생성될때 일부러 메모리 오버헤드를 만들어 억지 기법 공격 (brute-force attack) 을 시도할때 병렬화 처리가 어려워 pbkdf2보다 강점)

- 등등…

끝입니다. 질문요?