Cryptography and Network Security Chapter XXX

Sixth Edition

by William Stallings

Lecture slides by RHB

Chapter XXX – Miscellaneous 6th ED topics

That's enough quotations.

— RHB

Outline

- will discuss:
  - Key wrapping
  - RSA-PSS digital signature scheme
  - Elliptic curve digital signature scheme
  - SHA-3 and Keccak

Key Wrapping

- for standard mass data modes:
  - first cipher block affected by only first data block
  - last data block affects only last cipher block
- in a hierarchy of keys, want something better for transmitting session keys (using master keys)
- for session keys, want something robust, providing strong encryption and also authentication (for arbitrary, unstructured bitpatterns)

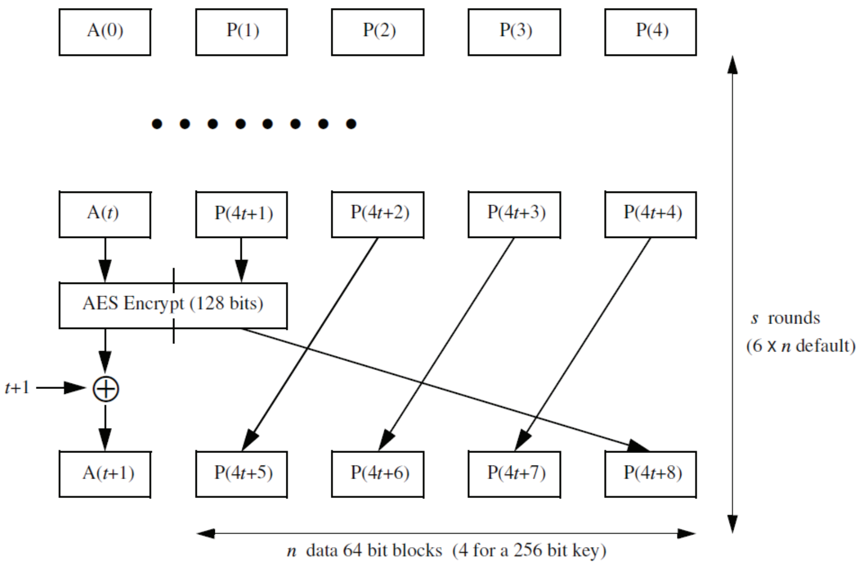
- Keywrapping gives more confusion and diffusion in encryption of a session key … plus authentication block

- 64 bit authentication block, A, initialised to A6A6A6A6A6A6A6A6
- 64 bit data blocks, P(1) … P(n)
- s = 6 x n rounds

- easy to encrypt / decrypt

A(0) := A6A6A6A6A6A6A6A6
P(1) ... P(n) := plaintext blocks
for t = 0 ... s-1 do
  W := AES[MKey, A(t) || P(nxt+1)]
  A(t+1) := (t+1) XOR MSB_{64}(W)
  P(nx(t+1)+n) := LSB_{64}(W)
  P(nx(t+1)+1) ... (nx(t+1)+(n-1)) :=
    P(nxt+2) ... (nxt+n)

Output: A(s) , P(nxs+1) ... P(nx(s+1))

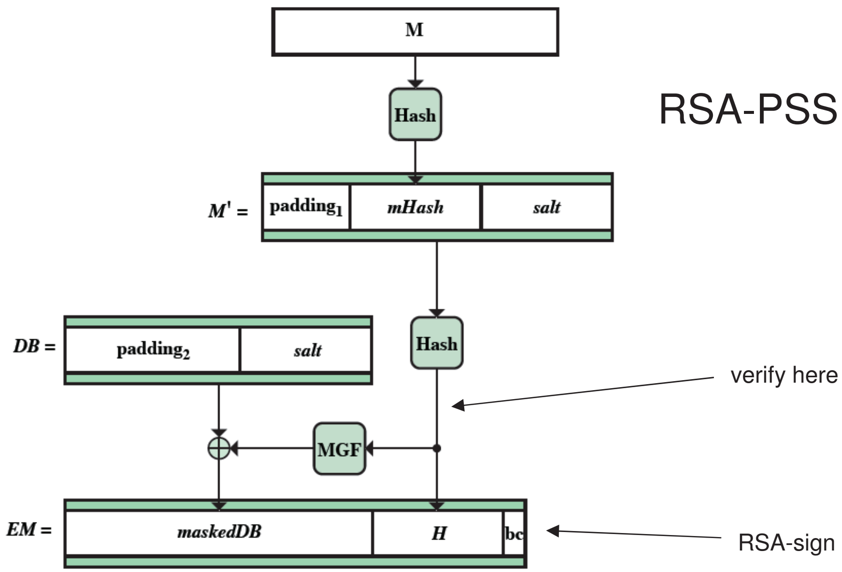


RSA-PSS Digital Signatures

- raw RSA is ‘malleable’ … vulnerable to chosen ciphertext attack … because of
  $C(M_1) \times C(M_2) = C(M_1 \times M_2)$
- RSA-PSS invented to give greater security to RSA based signatures
- used to create signatures

- includes padding / salt (c.f. OAEP)
- easy to create / verify

RSA-PSS



Elliptic Curve Digital Signatures

- a digital signature scheme, simpler than DSA, but with similar security properties
- uses elliptic curves with large prime order

- global parameters:
  - a large prime q
  - elliptic curve $E_q(a,b)$ specified by a,b
  - base point $G = (x_g, y_g)$ on curve $E_q(a,b)$
  - prime order of base point, n , i.e. n is smallest multiple of G such that $nG = O$

- key generation … signer side (Bob)
  - select d: $1 <= d <= n-1$ … private key
  - compute $Q = dG$ in $E_q(a,b)$ … public key

- signing
  - ⌘ select k: $1 <= k <= n-1$, k coprime to n
  - compute $P = kG = (x,y)$ in $E_q(a,b)$
  - compute $r = x \bmod n$ … if $r = 0$ go to ⌘
  - compute $k^{-1} \bmod n$
  - compute $s = k^{-1}(H(M)+dr) \bmod n$
    - if $s = 0$ go to ⌘ … else s coprime to n
  - signature is (r,s)

- verifying
  - check (both) $1 <= r,s <= n-1$
  - compute $w = s^{-1} \bmod n$
  - compute $u_1 = H(M)w$ and $u_2 = rw$
  - compute $X = (x_1, y_1) = u_1G + u_2Q$ in $E_q(a,b)$
  - if $X = O$ reject …
  - else compute $v = x_1 \bmod n$
  - accept signature if $v = r$

ECDSA ... verification.

Public info: q, a, b, $E_q(a, b)$

$G = (x_g, y_g)$ ... (base point on $E_q(a, b)$)

n ... order of G ... $nG = O$, $Q = dG$

$r = x \bmod n \neq 0$, where $P = kG = (x, y)$, $s = k^{-1}(H(M) + dr) \bmod n$

Secret info: d ... (with $0 < d < n$), k ... (with $0 < k < n$), $k^{-1} \bmod n$
(N.B. k^{-1} exists since k, n coprime)

Verify the signature by testing $v \stackrel{?}{=} r$ where

$v = x_1 \bmod n$ $\qquad (x_1, y_1) = X = u_1G + u_2Q \neq O$

$w = s^{-1} \bmod n$ $\qquad u_1 = H(M)w \qquad u_2 = rw$

$$
\begin{aligned}
X &= (x_1, y_1) \\
&= u_1G + u_2Q \\
&= u_1G + u_2dG \\
&= H(M)wG + rwdG \\
&= (H(M) + rd)wG \\
&= (H(M) + rd)[(k^{-1}(H(M) + dr) \bmod n)^{-1} \bmod n]G \\
&= (H(M) + rd)[k(H(M) + dr)^{-1} \bmod n]G \\
&= [(H(M) + rd) \bmod n][k(H(M) + dr)^{-1} \bmod n]G \quad \text{(because } nG = O\text{)} \\
&= kG \quad \text{(because } k < n\text{)} \\
&= P \\
&= (x, y)
\end{aligned}
$$

So $x_1 = x$. So $x_1 \bmod n = v = r = x \bmod n$.

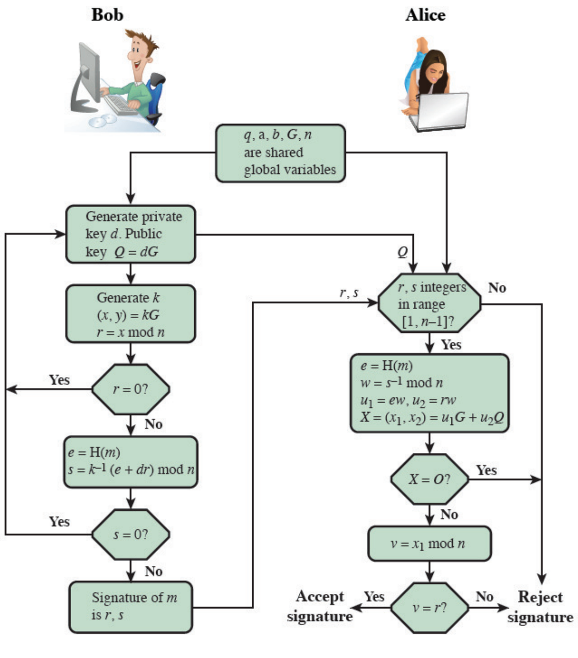


Figure 13.6 ECDSA Signing and Verifying

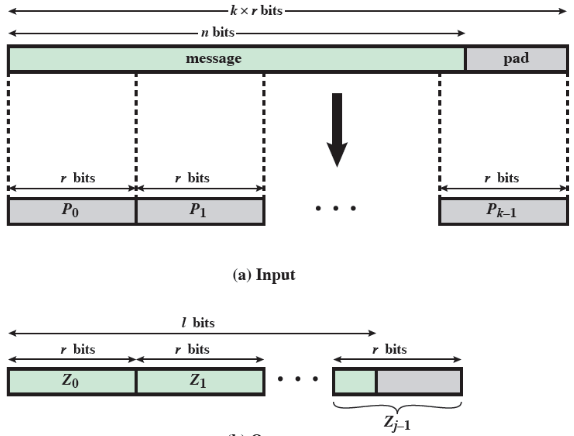
Elliptic Curve Digital Signature

SHA-3

- the new replacement for SHA-1 and SHA-2, plug-compatible with SHA-2
- ready and available in case it's ever needed due to any weakness in SHA-2 being discovered (none suspected so far)
- announced in 2012
- from the same stable that produced AES

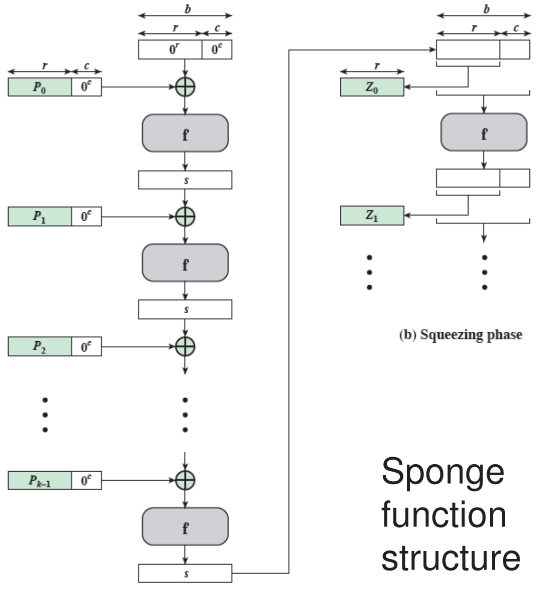
SHA-3 Hash Operation

- SHA-3 uses the idea of a sponge function



(a) Input

(b) Output



(a) Absorbing phase

(b) Squeezing phase

Sponge function structure

A sponge function uses an internal state with size b bits. This is bigger than the input (and) output block size of r bits, and allows for more complicated transformations of the input to output.

The capacity c = b - r, is a measure of the additional ‘scrambling power’ that can be applied to the input.

The squeezing phase iteratively rehashes the output-so-far until the required length is produced.

SHA-3 and Keccak

- the sponge function used in SHA-3 is called Keccak (f in the sponge iteration)
- internal blocksize = 1600 bits

SHA-3 Parameters

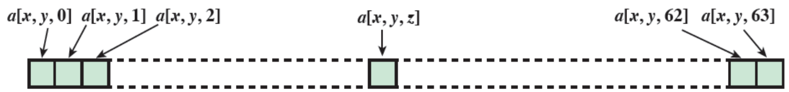
Total is 1600

Message Digest Size	224	256	384	512
Message Size	no maximum	no maximum	no maximum	no maximum
Block Size (bitrate *r*)	1152	1088	832	576
Word Size	64	64	64	64
Number of Rounds	24	24	24	24
Capacity *c*	448	512	768	1024
Collision resistance	2^{112}	2^{128}	2^{192}	2^{256}
Second preimage resistance	2^{224}	2^{256}	2^{384}	2^{512}

Keccak internal block structure

	$x = 0$	$x = 1$	$x = 2$	$x = 3$	$x = 4$
$y = 4$	$L[0, 4]$	$L[1, 4]$	$L[2, 4]$	$L[3, 4]$	$L[4, 4]$
$y = 3$	$L[0, 3]$	$L[1, 3]$	$L[2, 3]$	$L[3, 3]$	$L[4, 3]$
$y = 2$	$L[0, 2]$	$L[1, 2]$	$L[2, 2]$	$L[3, 2]$	$L[4, 2]$
$y = 1$	$L[0, 1]$	$L[1, 1]$	$L[2, 1]$	$L[4, 1]$	$L[4, 1]$
$y = 0$	$L[0, 0]$	$L[1, 0]$	$L[2, 0]$	$L[3, 0]$	$L[4, 0]$

(a) State variable as 5 ×5 matrix A of 64-bit words



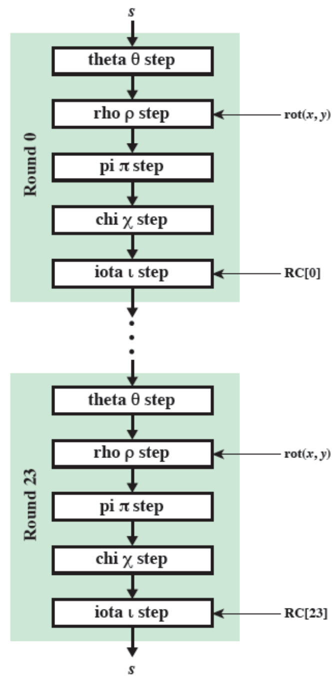
(b) Bit labeling of 64-bit words

Each $L[x,y]$ is a ‘lane’ $a[x,y,z]$ … $z = 0$ … 63
Lanes aggregate to columns $C[x]$ … $y = 0$ … 4

Keccak

- Keccak (f in the sponge iteration) works on the internal state
- internal block (1600 bits) is organised in a 3D grid (2D grid of lanes)
- Keccak (f in the sponge iteration) is a composition of five individual transformations … applied in 24 rounds
- theta (θ) ; rho (ρ) ; pi (π) ; chi (χ) ; iota (ι)

sequential composition



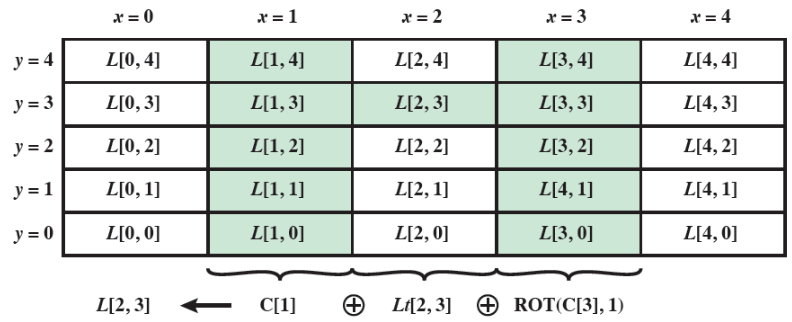
Keccak function structure

24 rounds

Step Functions in SHA-3

Function	Type	Description
θ	Substitution	New value of each bit in each word depends its current value and on one bit in each word of preceding column and one bit of each word in succeeding column.
ρ	Permutation	The bits of each word are permuted using a circular bit shift. $W[0, 0]$ is not affected.
π	Permutation	Words are permuted in the 5×5 matrix. $W[0, 0]$ is not affected.
χ	Substitution	New value of each bit in each word depends on its current value and on one bit in next word in the same row and one bit in the second next word in the same row..
ι	Substitution	$W[0, 0]$ is updated by XOR with a round constant.

Theta



(a) θ step function

Rho

rho ($a[x,y,z]$) = $a[x,y,(z-(t+1)(t+2)/2 \bmod 64)]$

where $0 <= t <= 24$ and

$$\begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^t \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \text{ in } GF(5)^{2\times 2}$$

z dimension manipulated / $a[0,0]$ unchanged

Rotation values by word position in matrix

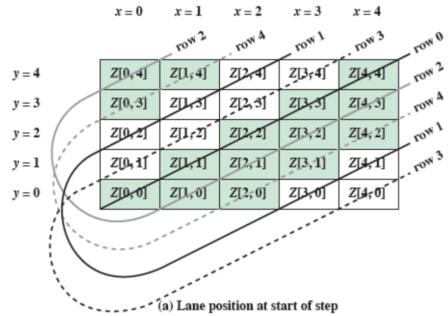
	$x = 0$	$x = 1$	$x = 2$	$x = 3$	$x = 4$
$y = 4$	18	2	61	56	14
$y = 3$	41	45	15	21	8
$y = 2$	3	10	43	25	39
$y = 1$	36	44	6	55	20
$y = 0$	0	1	62	28	27

Pi

pi ($L[x,y]$) = $L[x',y']$

where

$$\begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix}$$



(a) Lane position at start of step

	$x = 0$	$x = 1$	$x = 2$	$x = 3$	$x = 4$
$y = 4$	$Z[2, 0]$	$Z[3, 1]$	$Z[4, 2]$	$Z[0, 3]$	$Z[1, 4]$
$y = 3$	$Z[4, 0]$	$Z[0, 1]$	$Z[1, 2]$	$Z[2, 3]$	$Z[3, 4]$
$y = 2$	$Z[1, 0]$	$Z[2, 1]$	$Z[3, 2]$	$Z[4, 3]$	$Z[0, 4]$
$y = 1$	$Z[3, 0]$	$Z[4, 1]$	$Z[0, 2]$	$Z[1, 3]$	$Z[2, 4]$
$y = 0$	$Z[0, 0]$	$Z[1, 1]$	$Z[2, 2]$	$Z[3, 3]$	$Z[4, 4]$

(b) Lane position after permutation

Chi

chi ($C[x]$) =
$C[x]$ xor ((not $C[x+1]$) and $C[x+2]$))

	$x = 0$	$x = 1$	$x = 2$	$x = 3$	$x = 4$
$y = 4$	$L[0, 4]$	$L[1, 4]$	$L[2, 4]$	$L[3, 4]$	$L[4, 4]$
$y = 3$	$L[0, 3]$	$L[1, 3]$	$L[2, 3]$	$L[3, 3]$	$L[4, 3]$
$y = 2$	$L[0, 2]$	$L[1, 2]$	$L[2, 2]$	$L[3, 2]$	$L[4, 2]$
$y = 1$	$L[0, 1]$	$L[1, 1]$	$L[2, 1]$	$L[4, 1]$	$L[4, 1]$
$y = 0$	$L[0, 0]$	$L[1, 0]$	$L[2, 0]$	$L[3, 0]$	$L[4, 0]$

$$L[2,3] \leftarrow L[2,3] \oplus \overline{L[3,3]} \text{ AND } L[4,3]$$

(b) χ step function

Iota

iota ($L[0,0]$) = $L[0,0]$ xor RC(i_{round_number})

Table 11.8 Round Constants in SHA-3

Round	Constant (hexadecimal)	Number of 1 bits	Round	Constant (hexadecimal)	Number of 1 bits
0	0000000000000001	1	12	000000008000808B	6
1	0000000000008082	3	13	800000000000008B	5
2	800000000000808A	5	14	8000000000008089	5
3	8000000080008000	3	15	8000000000008003	4
4	000000000000808B	5	16	8000000000008002	3
5	0000000080000001	2	17	8000000000000080	2
6	8000000080008081	5	18	000000000000800A	3
7	8000000000008009	4	19	800000008000000A	4
8	000000000000008A	3	20	8000000080008081	5
9	0000000000000088	2	21	8000000000008080	3
10	0000000080008009	4	22	0000000080000001	2
11	000000008000000A	3	23	8000000080008008	4

SHA-3 Summary

- 3D structure unlike SHA-2 and SHA-1
- each step simple, focusing on one or two dimensions … simpler to analyse for assurance of security
- combination gives a strong hash function