Hashing Functions and Digital Signatures

Izmir Institute of Technology
Dept. of Computer Engineering
Asst. Prof. Dr. Serap ŞAHİN

Ref: *W.Stallings"Cryptography and Network Security, Chapter 11, Chapter 13, Fifth Edition", Lecture slides by Lawrie Brown*

Cryptographic Hash Functions

Each of the messages, like each one he had ever read of Stern's commands, began with a number and ended with a number or row of numbers. No efforts on the part of Mungo or any of his experts had been able to break Stern's code, nor was there any clue as to what the preliminary number and those ultimate numbers signified.

***—Talking to Strange Men,* Ruth Rendell**

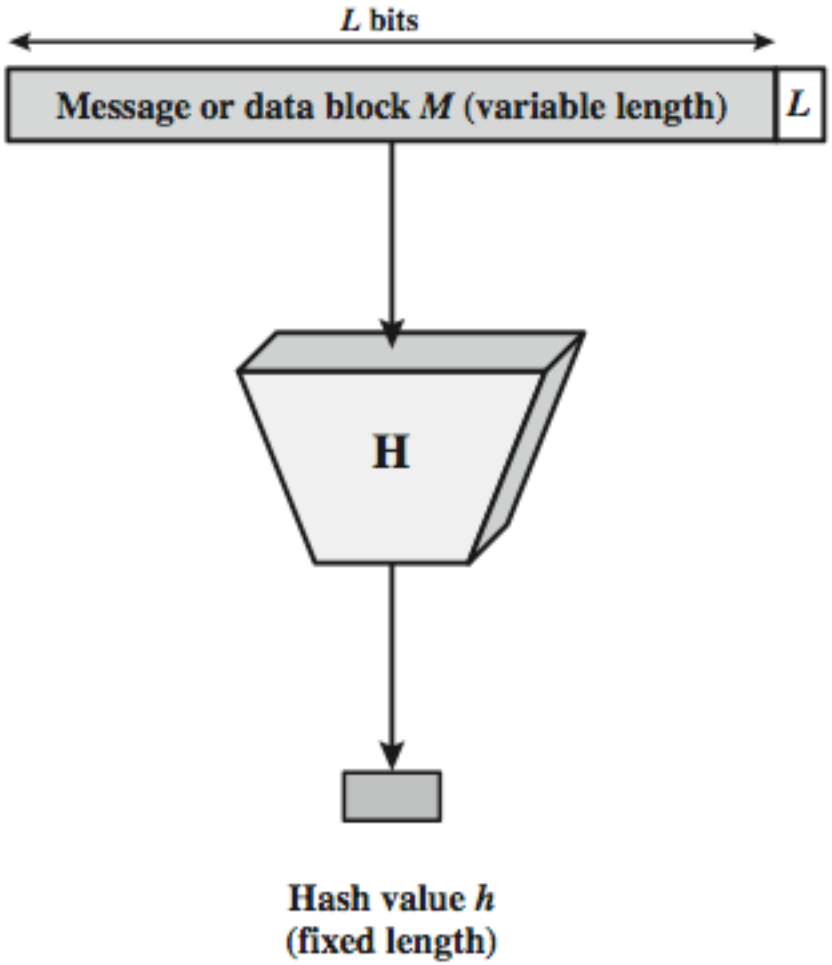
Hash Functions

- condenses arbitrary message to fixed size

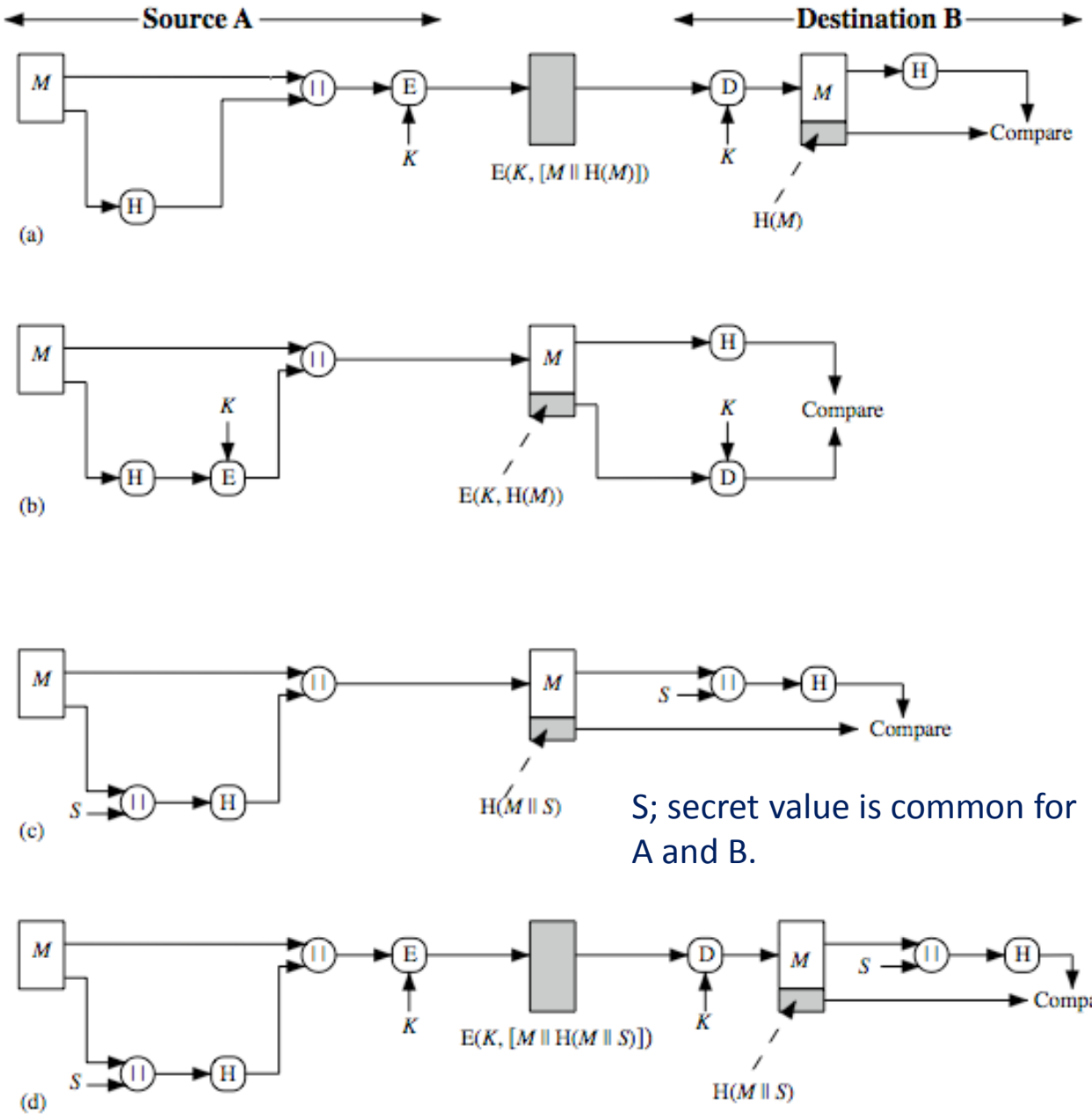
  `h = H(M)`

- usually assume hash function is public
- hash used to detect changes to message
- want a cryptographic hash function
  - computationally infeasible to find data mapping to specific hash (one-way property)
  - computationally infeasible to find two data to same hash (collision-free property)

Cryptographic Hash Function

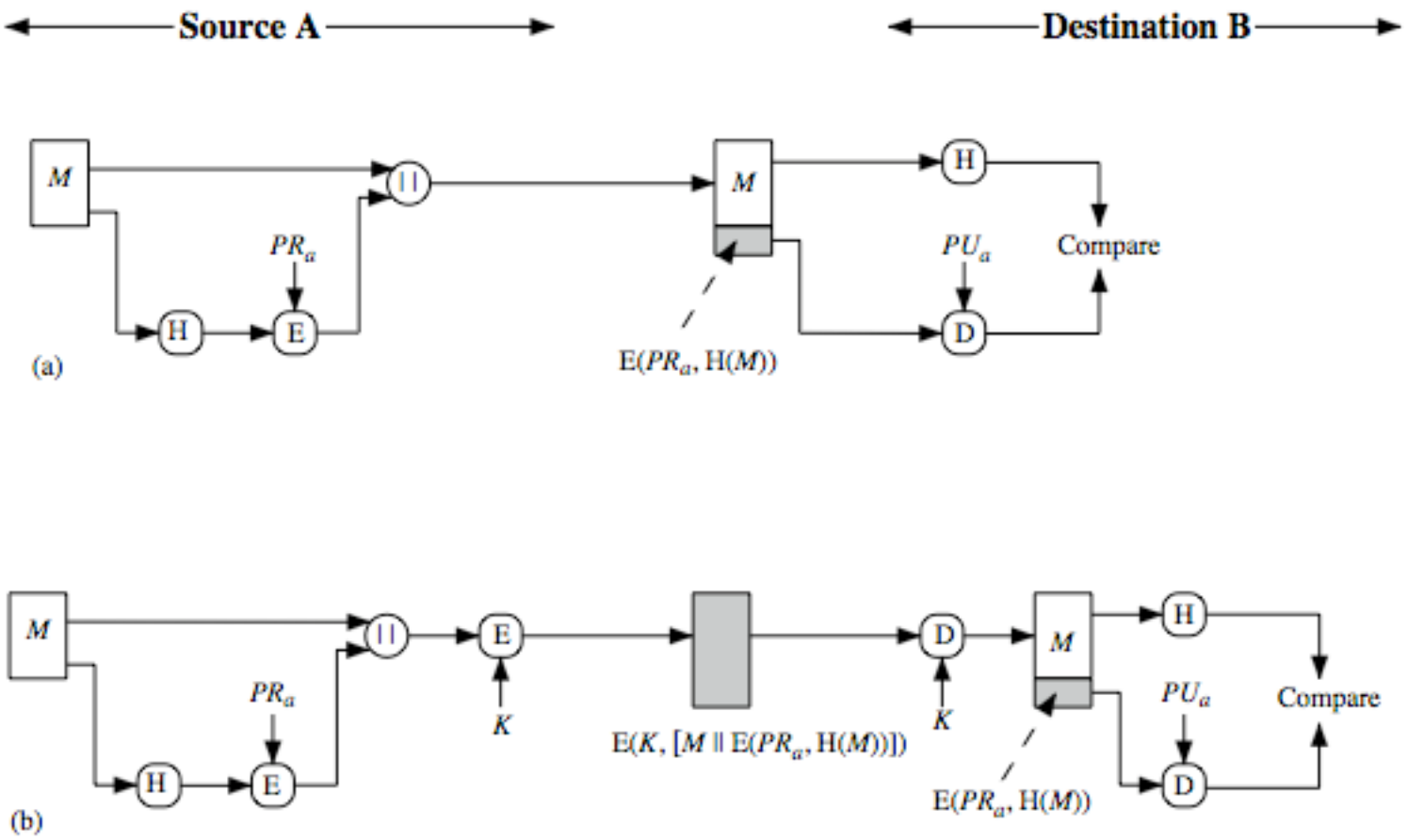


Hash Functions & Message Authent-ication



S; secret value is common for A and B.

Hash Functions & Digital Signatures



Other Hash Function Uses

- to create a one-way password file
  - store hash of password not actual password
- for intrusion detection and virus detection
  - keep & check hash of files on system
- pseudorandom function (PRF) or pseudorandom number generator (PRNG)
  - a common application for a hash-based PRF is for the generation of symmetric keys.

Two Simple Insecure Hash Functions

- consider two simple insecure hash functions
- bit-by-bit exclusive-OR (XOR) of every block
  - $C_i = b_{i1}$ *xor* b_{i2} *xor . . . xor* b_{im}
  - This operation produces a simple parity for each bit position and is known as a longitudinal redundancy check.
  - reasonably effective as data integrity check
- one-bit circular shift on hash value
  - for each successive *n-bit* block
    - rotate current hash value to left by 1bit and XOR block
  - good for data integrity but useless for security

Hash Function Requirements

Requirement	Description
Variable input size	H can be applied to a block of data of any size.
Fixed output size	H produces a fixed-length output.
Efficiency	H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.
Preimage resistant (one-way property)	For any given hash value h, it is computationally infeasible to find y such that H(y) = h.
Second preimage resistant (weak collision resistant)	For any given block x, it is computationally infeasible to find y ! x with H(y) = H(x).
Collision resistant (strong collision resistant)	It is computationally infeasible to find any pair (x, y) such that H(x) = H(y).
Pseudorandomness	Output of H meets standard tests for pseudorandomness

Attacks on Hash Functions

- have brute-force attacks and cryptanalysis
  - The brute force method is to pick values of y at random and try each value until a collision occurs. For an *m-bit* hash value, the level of effort is proportional to 2^m.
- a preimage or second preimage attack
  - find y s.t. $H(y)$ equals a given hash value
- collision resistance
  - find two messages x & y with same hash so $H(x) = H(y)$

Attacks on Hash Functions

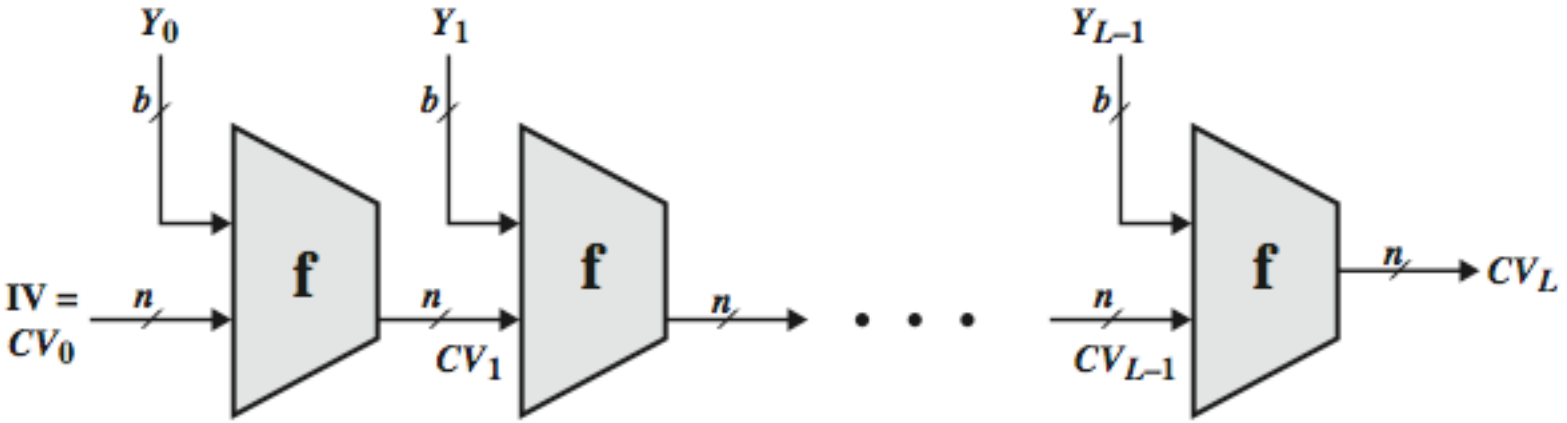
- hence value $2^{m/2}$ determines strength of hash code against brute-force attacks “128-bits inadequate, 160-bits suspect”
  - Van Oorschot and Wiener presented a design for a $10 million collision search machine for MD5, which has a 128-bit hash length, that could find a collision in 24 days. Thus a 128-bit code may be viewed as inadequate. (1994)
  - The next step up, with a hash length of 160 bits, the same search machine would require over four thousand years to find a collision.
  - With today's technology, the time would be much shorter, so that 160 bits now appears suspect.

Birthday Attacks

- might think a 64-bit hash is secure
- but by **Birthday Paradox** is not
  - the birthday paradox – the chance that in a group of people two will share the same birthday – only 23 people are needed for a Pr>0.5 of this.
  - Can generalize the problem to one wanting a matching pair from any two sets, and show need $2^{m/2}$ in each to get a matching m-bit hash.
- **birthday attack** works thus:
  - given user prepared to sign a valid message x
  - opponent generates $2^{m/2}$ variations x' of x, all with essentially the same meaning, and saves them
  - opponent generates $2^{m/2}$ variations y' of a desired fraudulent message y
  - two sets of messages are compared to find pair with same hash (probability > 0.5 by birthday paradox)
  - have user sign the valid message, then substitute the forgery which will have a valid signature
- conclusion is that need to use larger MAC/hash

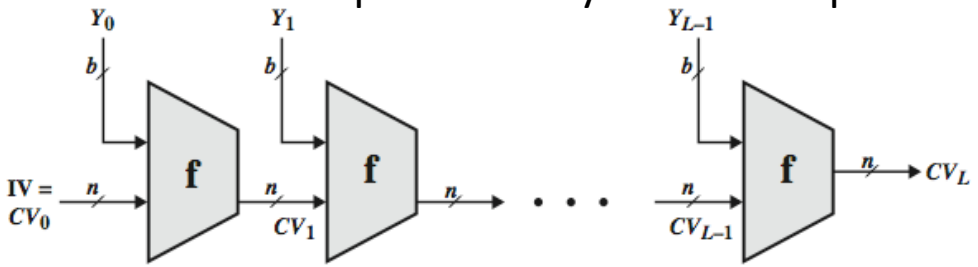
Hash Function Cryptanalysis

- cryptanalytic attacks exploit some property of algorithms so faster than exhaustive search
- hash functions use iterative structure
  - process message in blocks (include length)
- attacks focus on collisions in function f



Hash Function Cryptanalysis

- This was proposed by Merkle and is the structure of most hash functions in use today.
- The hash function takes an input message and partitions it into *L* fixed-sized blocks of *b* bits each.
- The final block includes the value of the total length of the input to the hash function. The inclusion of the length makes the job of the opponent more difficult. The hash algorithm involves repeated use of a compression function, *f*, that takes two inputs and produces an *n*-bit output.
- Often, $b > n$; hence the term compression. The motivation for this iterative structure stems from the observation by Merkle and Damgard that if the compression function is collision resistant, then so is the resultant iterated hash function.
- Cryptanalysis of hash functions focuses on the internal structure of f and is based on attempts to find efficient techniques for producing collisions for a single execution of f. Once that is done, the attack must take into account the fixed value of IV. The attack on f depends on exploiting its internal structure. The attacks that have been mounted on hash functions are rather complex and beyond our scope here.



Block Ciphers as Hash Functions

- A number of proposals have been made for hash functions based on using a cipher block chaining technique, but without the secret key
  - One of the first such proposals was that of Rabin, which divided a message M into fixed-size blocks, and used a symmetric encryption system such as DES to compute the hash code. This is similar to the CBC technique, but in this case there is no secret key.
- can use block ciphers as hash functions
  - using H_0=0 and zero-pad of final block
  - compute: $H_i = E_{M_i} [H_{i-1}]$
  - and use final block as the hash value
  - similar to CBC but without a key
- resulting hash is too small (64-bit)
  - both due to direct birthday attack
  - and to "meet-in-the-middle" attack
- other variants also susceptible to attack

Secure Hash Algorithm

- SHA originally designed by NIST & NSA in 1993
- was revised in 1995 as SHA-1
- US standard for use with DSA signature scheme
  - standard is FIPS 180-1 1995, also Internet RFC3174
  - nb. the algorithm is SHA, the standard is SHS
- based on design of MD4 with key differences
- produces 160-bit hash values
- recent 2005 results on security of SHA-1 have raised concerns on its use in future applications

Revised Secure Hash Standard

- NIST issued revision FIPS 180-2 in 2002
- adds 3 additional versions of SHA-2
  - SHA-256, SHA-384, SHA-512
- designed for compatibility with increased security provided by the AES cipher
- structure & detail is similar to SHA-1
- hence analysis should be similar
- but security levels are rather higher

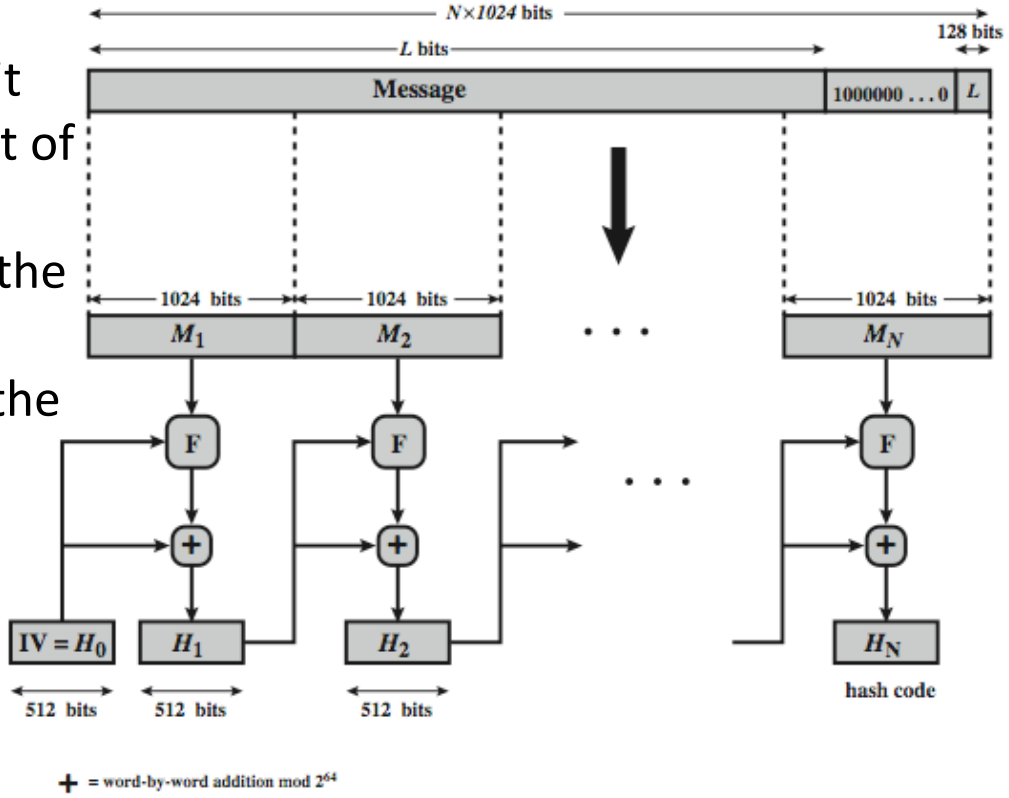
SHA Versions

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message digest size	160	224	256	384	512
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size	512	512	512	1024	1024
Word size	32	32	32	64	64
Number of steps	80	64	64	80	80

SHA-512 Overview

Step 1: Append padding bits, consists of a single 1-bit followed by the necessary number of 0-bits, so that its length is congruent to 896 modulo 1024

- Step 2: Append length as an (big-endian) unsigned 128-bit integer
- Step 3: Initialize hash buffer to a set of 64-bit integer constants
- Step 4: Process the message in 1024-bit (128-word) blocks, which forms the heart of the algorithm. Each round takes as input the 512-bit buffer value H_i, and updates the contents of that buffer.
- Step 5: Output the final state value as the resulting hash



SHA-3

- SHA-1 not yet "broken"
  - but similar to broken MD5 & SHA-0
  - so considered insecure
- SHA-2 (esp. SHA-512) seems secure
  - shares same structure and mathematical operations as predecessors so have concern
- NIST announced in 2007 a competition for the SHA-3 next gen NIST hash function
  - goal to have in place by 2012 but not fixed

Cryptographic Hash Algorithm Competition

- NIST opened a public competition on Nov. 2, 2007 to develop a new cryptographic hash algorithm, which converts a variable length message into a short "message digest" that can be used for digital signatures, message authentication and many other security applications in the information infrastructure.
- The new hash algorithm will be referred to as "SHA-3" and will augment the hash algorithms currently specified in the Federal Information Processing Standard (FIPS) 180-3, Secure Hash Standard.
- **December 10, 2010; NIST Announces SHA-3 Finalists :**BLAKE, Grøstl, JH, Keccak, and Skein.
- The competition ended in October 2012 and Keccak was selected as winner.
- SHA-3 NIST Standard: http://csrc.nist.gov/groups/ST/hash/sha-3/sha-3_standardization.html

SHA-3 Requirements

- replace SHA-2 with SHA-3 in any use
  - so use same hash sizes; 224, 256, 384, and 512 bits
- preserve the online nature of SHA-2
  - so must process small blocks (512 / 1024 bits)
- evaluation criteria
  - security close to theoretical max for hash sizes
  - cost in time & memory efficient over a range of hardware platforms
  - Algorithm and implementation characteristics: such as flexibility & simplicity

Digital Signatures

To guard against the baneful influence exerted by strangers is therefore an elementary dictate of savage prudence. Hence before strangers are allowed to enter a district, or at least before they are permitted to mingle freely with the inhabitants, certain ceremonies are often performed by the natives of the country for the purpose of disarming the strangers of their magical powers, or of disinfecting, so to speak, the tainted atmosphere by which they are supposed to be surrounded.

—*The Golden Bough*, Sir James George Frazer

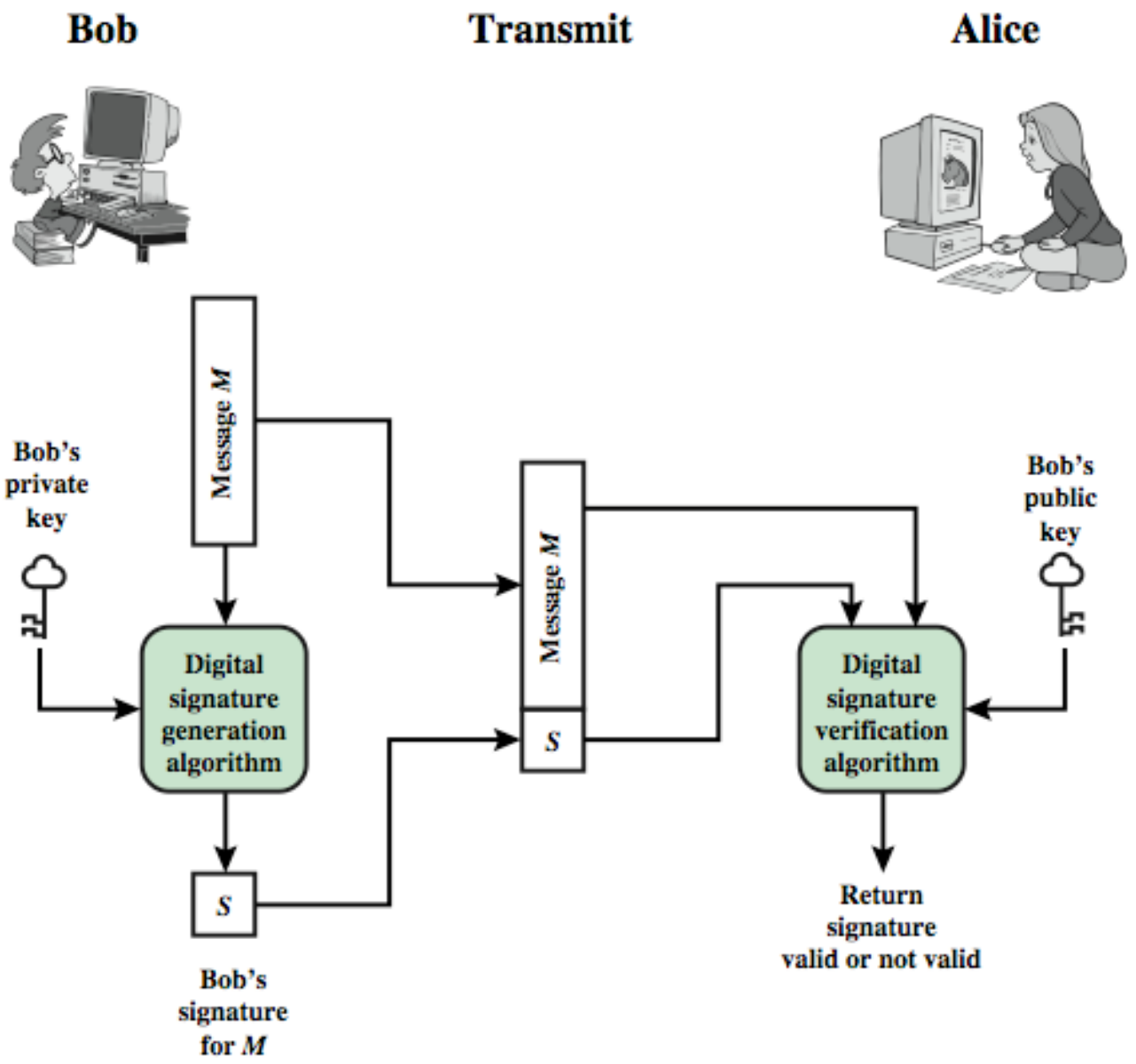
Digital Signatures

- Message authentication protects two parties who exchange messages from any third party.
- However, it does not protect the two parties against each other either fraudulently creating, or denying creation, of a message.
- A digital signature is analogous to the handwritten signature, and provides a set of security capabilities that would be difficult to implement in any other way. It must have the following properties:
  - It must verify the author and the date and time of the signature
  - It must to authenticate the contents at the time of the signature
  - It must be verifiable by third parties, to resolve disputes
  - thus, the digital signature function includes the authentication function.

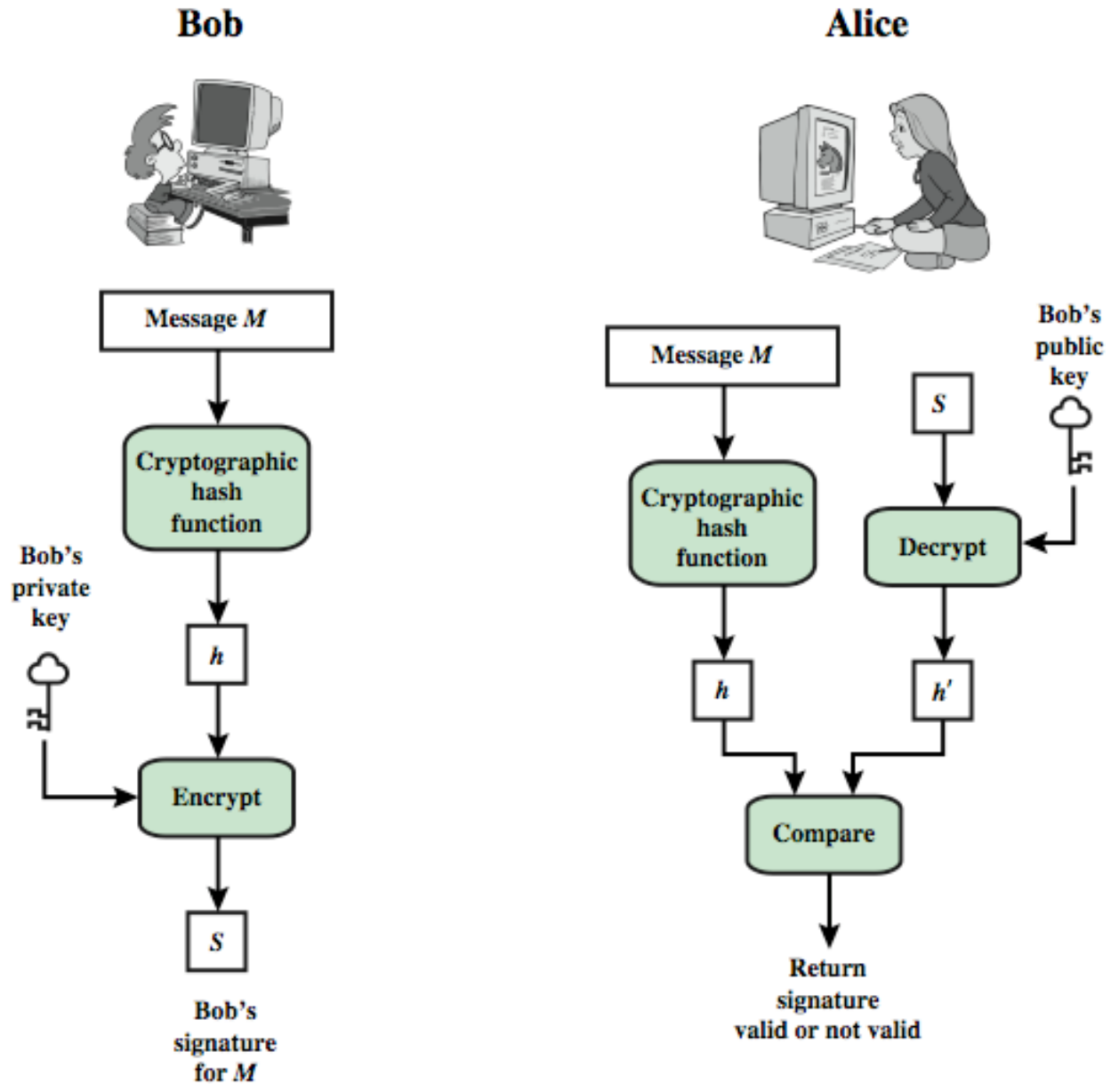
Digital Signatures

- have looked at message authentication
  - but does not address issues of lack of trust
- digital signatures provide the ability to:
  - verify author, date & time of signature
  - authenticate message contents
  - be verified by third parties to resolve disputes
- hence include authentication function with additional capabilities

Digital Signature Model



Digital Signature Model



Attacks and Forgeries

- Attacks (Here A denotes the user whose signature is being attacked and C denotes the attacker.)
  - **key-only attack**; C only knows A's public key.
  - **known message attack**; C is given access to a set of messages and signatures.
  - **generic chosen message attack**; C chooses a list of messages before attempting to breaks A's signature scheme, independent of A's public key. C then obtains from A valid signatures for the chosen messages. The attack is generic because it does not depend on A's public key; the same attack is used against everyone.

Attacks and Forgeries

  - **directed chosen message attack**; Similar to the generic attack, except that the list of messages is chosen after C knows A's public key but before signatures are seen.
  - **adaptive chosen message attack**; C is allowed to use A as an "oracle." This means the A may request signatures of messages that depend on previously obtained message-signature pairs.
- break success levels; C determines A's private key.
- **Universal forgery**: C finds an efficient signing algorithm that provides an equivalent way of constructing signatures on arbitrary messages.
- **Selective forgery**: C forges a signature for a particular message chosen by C.
- **Existential forgery**: C forges a signature for at least one message. C has no control over the message. Consequently this forgery may only be a minor nuisance to A.

Digital Signature Requirements

- must depend on the message signed
- must use information unique to sender
  - to prevent both forgery and denial
- must be relatively easy to produce
- must be relatively easy to recognize & verify
- be computationally infeasible to forge
  - with new message for existing digital signature
  - with fraudulent digital signature for given message
- be practical save digital signature in storage

Direct Digital Signatures

- involve only sender & receiver
- assumed receiver has sender's public-key
- digital signature made by sender signing entire message or hash with private-key
- can encrypt using receivers public-key
- important that sign first then encrypt message & signature
- security depends on sender's private-key

ElGamal Digital Signatures

- signature variant of ElGamal, related to D-H
  - so uses exponentiation in a finite (Galois)
  - with security based difficulty of computing discrete logarithms, as in D-H
- uses private key for encryption (signing)
- uses public key for decryption (verification)
- each user (eg. A) generates their key
  - chooses a secret key (number): $1 < x_A < q-1$
  - compute their **public key**: $y_A = a^{x_A} \bmod q$

ElGamal Digital Signature

- Alice signs a message M to Bob by computing
  - the hash $m = H(M)$, $0 <= m <= (q-1)$
  - chose random integer K with $1 <= K <= (q-1)$ and $\gcd(K, q-1) = 1$
  - compute temporary key: $S_1 = a^k \bmod q$
  - compute K^{-1} the inverse of $K \bmod (q-1)$
  - compute the value: $S_2 = K^{-1}(m - x_A S_1) \bmod (q-1)$
  - signature is: (S_1, S_2)
- any user B can verify the signature by computing
  - $V_1 = a^m \bmod q$
  - $V_2 = y_A^{S_1} S_1^{S_2} \bmod q$
  - signature is valid if $V_1 = V_2$

ElGamal Signature Example

- use field GF(19) $q=19$ and $a=10$
- Alice computes her key:
  - A chooses $x_A=16$ & computes $y_A=10^{16} \bmod 19 = 4$
- Alice signs message with hash $m=14$ as $(3,4)$:
  - choosing random $K=5$ which has $\gcd(18,5)=1$
  - computing $S_1 = 10^5 \bmod 19 = 3$
  - finding $K^{-1} \bmod (q-1) = 5^{-1} \bmod 18 = 11$
  - computing $S_2 = 11(14-16.3) \bmod 18 = 4$
- any user B can verify the signature by computing
  - $V_1 = 10^{14} \bmod 19 = 16$
  - $V_2 = 4^3.3^4 = 5184 = 16 \bmod 19$
  - since $16 = 16$ signature is valid

Schnorr Digital Signatures

- also uses exponentiation in a finite (Galois)
  - security based on discrete logarithms, as in D-H
- The main work for signature generation does not depend on the message and can be done during the idle time of the processor. Minimizes message dependent computation,
  - The message dependent part of the signature generation requires multiplying a 2*n-bit* integer with an *n-bit* integer
- main work can be done in idle time
- have using a prime modulus *p*
  - *p–1* has a prime factor *q* of appropriate size
  - typically *p* 1024-bit and *q* 160-bit numbers

Schnorr Key Setup

- choose suitable primes *p , q;* , such that *q* is a prime factor of p – 1
- choose *a* such that $a^q = 1 \bmod p$
- (a,p,q) are global parameters for all
- each user (eg. A) generates a key
  - chooses a secret key (number): $0 < s_A < q$
  - compute their **public key**: $v_A = a^{-s_A} \bmod q$

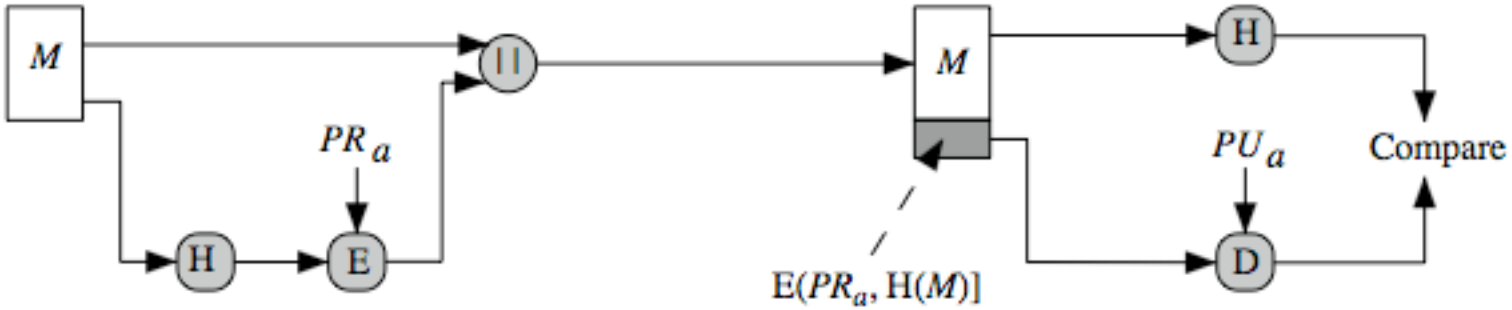
Schnorr Signature

- A user with public key v and private key s generates a signature as follows:
- user signs message by
  - choosing random r with $0<r<q$ and computing $x = a^r \bmod p$
  - concatenate message with x and hash result to computing: $e = H(M \,||\, x)$
  - computing: $y = (r + se) \bmod q$
  - signature is pair (e, y)
- any other user can verify the signature as follows:
  - computing: $x' = a^y v^e \bmod p$
  - verifying that: $e = H(M \,||\, x')$

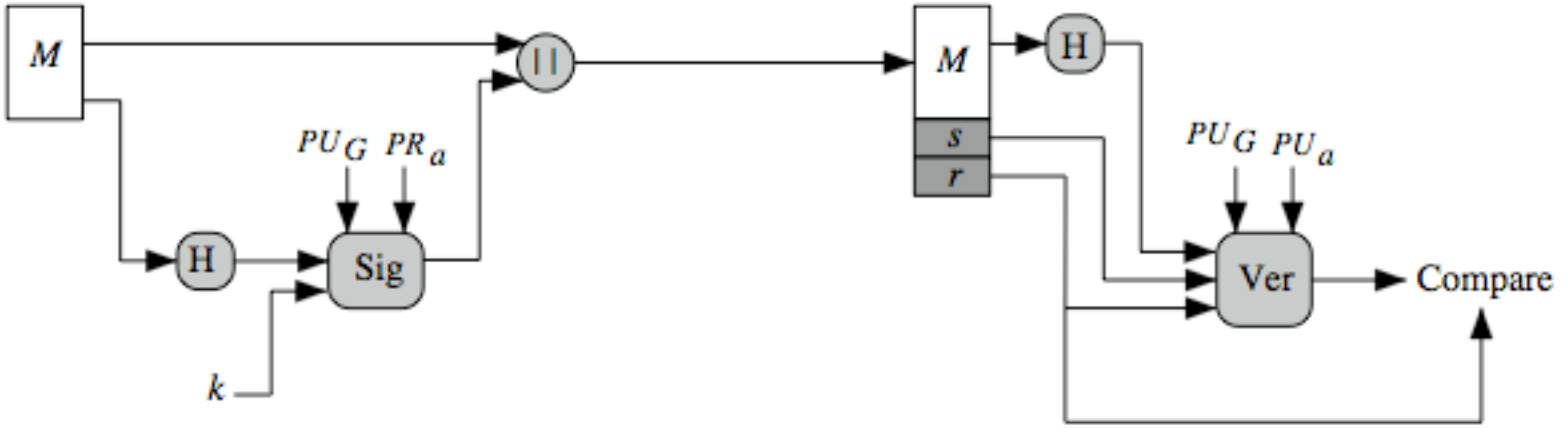
Digital Signature Standard (DSS)

- US Govt approved signature scheme
- designed by NIST & NSA in early 90's
- published as FIPS-186 in 1991
- revised in 1993, 1996 & then 2000
- uses the SHA hash algorithm
- DSS is the standard, DSA is the algorithm
- FIPS 186-2 (2000) includes alternative RSA & elliptic curve signature variants
- DSA is digital signature only unlike RSA
- is a public-key technique

DSS vs RSA Signatures



(a) RSA Approach



(b) DSS Approach

PU_G; a set of parameters known to a group of communicating principals. Global public key.

Digital Signature Algorithm (DSA)

The DSA is based on the difficulty of computing discrete logarithms and is based on schemes originally presented by ElGamal and Schnorr.

- creates a 320 bit signature
- with 512-1024 bit security
- smaller and faster than RSA
- a digital signature scheme only
- security depends on difficulty of computing discrete logarithms
- variant of ElGamal & Schnorr schemes

The DSA signature scheme has advantages, being both smaller (320 vs 1024bit) and faster (much of the computation is done modulo a 160 bit number), over RSA. Unlike RSA, it cannot be used for encryption or key exchange. Nevertheless, it is a public-key technique.

DSA Key Generation

- have shared global public key values (p,q,g):
  - choose 160-bit prime number q
  - choose a large prime p with $2^{L-1} < p < 2^{L}$
    - where L= 512 to 1024 bits and is a multiple of 64
    - such that q is a 160 bit prime divisor of $(p-1)$
  - choose $g = h^{(p-1)/q}$
    - where $1<h<p-1$ and $h^{(p-1)/q} \bmod p > 1$
- users choose private & compute public key:
  - choose random private key: $x<q$
  - compute public key: $y = g^{x} \bmod p$

DSA Signature Creation

- to **sign** a message M the sender:
  - generates a random signature key k, k<q
  - k must be random, be destroyed after use, and never be reused
  - the public key components (p,q,g), the user's private key (x)
- then computes signature pair:

  $r = (g^k \bmod p) \bmod q$

  $s = [k^{-1}(H(M) + xr)] \bmod q$
- sends signature (r,s) with message M

Note that computing r only involves calculation mod p and does not depend on message, hence can be done in advance. Similarly with randomly choosing k's and computing their inverses.

DSA Signature Verification

- having received M & signature (r,s)
- to **verify** a signature, recipient computes:

$$w = s^{-1} \bmod q$$

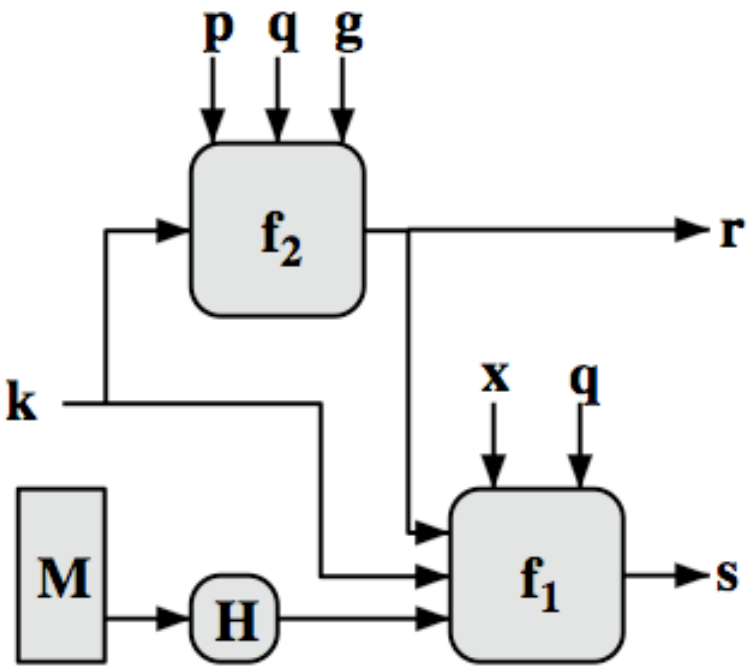
$$u1 = [H(M)w] \bmod q$$

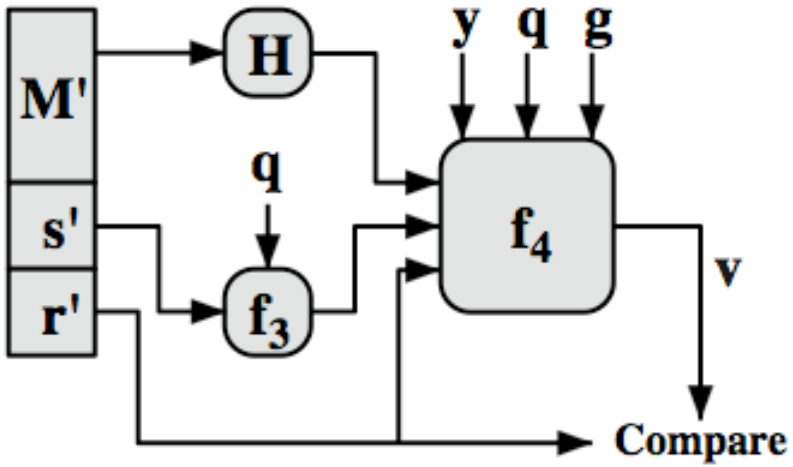
$$u2 = (rw) \bmod q$$

$$v = [(g^{u1}\ y^{u2}) \bmod p] \bmod q$$

- if $v=r$ then signature is verified

DSS Overview





$s = f_1(H(M), k, x, r, q) = (k^{-1} (H(M) + xr)) \bmod q$

$r = f_2(k, p, q, g) = (g^k \bmod p) \bmod q$

(a) Signing

$w = f_3(s', q) = (s')^{-1} \bmod q$

$v = f_4(y, q, g, H(M'), w, r')$

$= ((g^{(H(M')w) \bmod q} y^{r'w \bmod q}) \bmod p) \bmod q$

(b) Verifying

Summary

- have discussed:
  - digital signatures
  - ElGamal & Schnorr signature schemes
  - digital signature algorithm and standard