CSE 589 -- Lecture 9

One of the symptoms of an approaching nervous breakdown is the belief that one's work is terribly important.

Bertrand Russell

Plan for Today

- Review of digital signatures, one-way hash functions
- authenticated key-exchange
- MACs
- Example: SSL
- Secret Sharing and Splitting
- Example: Digital Cash
- Bit commitment and Mental Poker
- Zero-knowledge protocols.
- Pseudo-random number generators

Digital Signatures

Digital Signatures

- proof of authorship or agreement with contents of a document.
- Two components:
  - secret signing algorithm $S_k(M)$ = Signature
  - public verification algorithm $V_k(M, \text{Signature})$

    $V_k(M, \text{Signature})$ = true if $S_k(M)$ = Signature,
    false otherwise.

Signing Documents with RSA

- How Alice sends a **signed** message to Bob
  - Alice signs with her private key.

    $S(M) = D_A(M) = M^d \bmod n = S$
  - Alice sends Bob the message/signature pair (M,S)
  - Bob verifies with Alice's public key.

    $V(S,M) = \text{true} \iff E_A(S) = S^e \bmod n = M$
- Essential assumption: only Alice is able to compute S(M).

Forgery

- Anyone can forge Alice's signature on a "random" message, by picking y, computing

  $x = E_A(y) = y^e \bmod n$

  and sending (x,y) to Bob. [y is Alice's signature on x.]
- Solutions:
  - add redundancy to messages so that a forged signature of this type doesn't correspond to a meaningful message except with very low probability
  - use one-way hash functions.

Once again, using RSA...

- How Alice sends a **secret** message to Bob

  A B

  $C = E_B(M)$ C

  ----------> $M = D_B(C)$

- How Alice sends a **signed** message to Bob

  A B

  $C = D_A(M)$ M, C

  ----------> $M = E_A(C)$

Digital Signatures + Encryption
proof of authorship + privacy

- Alice signs with her private key then encrypts with Bob's public key

  A B

  $C = E_B(S_A(M))$ C

  ----------->

- Bob decrypts with his private key, then verifies with Alice's public key.

  $S_A(M) = D_B(C)$

  $M = V_A(C)$

One Way Hash Functions

Problems with signature schemes:

A solution: One-way hash functions.

message ----------> message digest, fingerprint

M ----------> h(M)

arbitrarily long at most m bits

such that:

- Given M, easy to compute h(M)
- Given M, hard to find another message M', s.t. h(M) = h(M')
- Even hard to find two random messages M and M' such that h(M) = h(M')
- Example one-way hash functions: MD4, MD5, SHS

Signatures and One-way Hash Functions

- How Alice sends a **signed** message to Bob

A		B
F = h(M)		
C = S_A(F)	C, M	
	---------->	if V_A(C) = h(M)
		signature verified.

Issues and Attacks

- If Mallory can find M' s.t. h(M')= h(M), could forge a message from Alice by sending C,M'
- If Mallory can find two message M' and M'' s.t. h(M')= h(M'') and convince Alice to sign h(M''), could forge C,M'.
- Birthday attack imposes a lower bound on the sizes of message digests.
  - Example: 40-bit message digest would be very insecure since a collision could be found with probability 1/2 with just over 2^{20} random hashes.
  - Secure Hash Standard: use 160 bit message digests.

Key Exchange

Classic Secret Key Exchange Diffie-Hellman

- A and B agree on large prime p and g < p primitive mod p.
- r1 (selected by A) and r2 (B) are random large numbers

A		B
$X = g^{r1} \bmod p$	------- X -------->	
	<------Y--------	$Y = g^{r2} \bmod p$
$K = Y^{r1} \bmod p$		$K = X^{r2} \bmod p$

- K is secret key that both A and B have computed independently.

Security Against Eve

- DH Problem: Given $g^{r1} \bmod p$, $g^{r2} \bmod p$, for r1 and r2 chosen at random in 1..p-1, compute
  $g^{r1\ r2} \bmod p$
- If discrete logarithm problem is easy, so is DH Problem.
- Believed to be computationally intractable

Security Against Mallory Man-in-the-Middle Attack

A	M	B
$X = g^{r1} \bmod p$ -- X-->	$X' = g^{m1} \bmod p$ -- X' --->	
<-Y'--	$Y' = g^{m2} \bmod p$ <--Y--	$Y = g^{r2} \bmod p$
$K1 = Y'^{r1} \bmod p$	$K1 = X^{m2} \bmod p$ $K2 = Y^{m1} \bmod p$	$K2 = X'^{r2} \bmod p$

- If they are communicating over a network in presence of active attacker, man-in-the-middle will always be a problem unless A and B share some information unavailable to Mallory.
- "Information advantage" -- some info predistributed over a trusted channel.

Approach typically taken

- Assume trusted authority (Trent) whose public key/verification algorithm is known to everyone.
- Trent hands out certificates.
  $C(A) = (ID(A), V_A, S_T (ID(A), V_A))$

Diffie-Hellman with Authentication

A and B agree on large prime p and g < p primitive mod p.

- A chooses random number r1, computes $X = g^{r1} \bmod p$ and sends it to B
- B chooses random number r2, computes $Y = g^{r2} \bmod p$ and sends Y, $Sig_B(X,Y)$, C(B) to A.
- A verifies C(B) using Ver_T, then verifies $Sig_B(X,Y)$ (using Ver_B) and computes the shared secret key $K = Y^{r1} \bmod p$
- A then sends $Sig_A(X,Y)$, C(A) to B.
- B verifies C(A) using Ver_T, then verifies $Sig_A(X,Y)$ (using Ver_A) and computes the shared secret key $K = X^{r2} \bmod p$

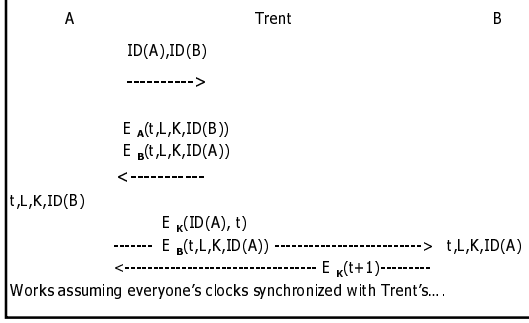
Authenticated Key Exchange Private-Key Setting

- Example: Kerberos
- Assumptions:
  - A and B each share secret keys with Trent (trusted authority/authentication server)
  - A wants to generate a session key for a conversation with Bob.
- Widely used because it's fast.

Kerberos

A key(T,A) **B** key(T,B)

t timestamp **K** session key **L** lifetime



Works assuming everyone's clocks synchronized with Trent's....

What's going on?

- Steps 1,2: A sends ID(A),ID(B), T returns $E_A(t,L,K,ID(B))$ $E_B(t,L,K,ID(A))$
- Step 3: A sends B $E_K(ID(A), t)$, $E_B(t,L,K,ID(A))$
- Step 4: B sends A $E_K(t+1)$

Replay Attacks

- Mallory can record old messages and then use them later (replay them) in an attempt to subvert the protocol.

Authenticated Key Exchange-- Public-Key Setting

- Alice and Bob get each other's public keys from Trent (KDC)
- Alice generates random session key K

A B

$C = E_{KB} S_{KA}(K)$ C

----------> $K = D_{KB} V_{KA}(C)$

- Problems?

Message Authentication Codes

So far

- Cryptosystems ----> privacy, secrecy
- Digital Signatures ---> authentication
- Key Exchange with Authentication ----> handing out public keys and exchanging private keys in a trustworthy way.
- Message Authentication Codes (MAC) -----> protect data integrity when communicating in private key setting

MAC

- One-way hash function with secret key.

  $h_K(M) = F$
- Same properties as one-way hash functions.
- Additional property: only someone with key can verify hash.
- Useful for providing authenticity/data integrity without secrecy.

Some Applications of MACs

- A and B communicating using symmetric crypto: Use MAC to prevent Mallory from garbling encrypted data => send M, $h_K(M)$.

  Upon receipt of message, verify the MAC against message.

  Mallory can't garble messages without being caught because he can't compute the MAC.
- can be used by a single user to determine if her files have been altered, say by virus. User computes MAC of her files and stores value in table. If she used one-way hash function instead, virus could compute the new hash value after infection and replace the table entry.

Constructing a MAC

- One easy way: encrypt hash value with a symmetric algorithm.

Application: SSL

Secure Socket Layer (SSL) used by Navigator/Explorer…

- For example, when I send my credit card over the Internet to amazon.com
- Disclaimer

SSL (Step 1)

- Client (my Netscape browser)
  - says hello
  - sends random string R
  - describes private-key crypto supported (e.g. 40 bit RC4)
  - describes key-exchange protocol supported (e.g., RSA)

SSL Step 2

- Server (amazon.com) responds with
  - its own random string R'
  - its public key certified by Trent (e.g., Verisign)
  - requests that client authenticate by providing certificate from Trent (optional)

SSL Step 3

- Client responds by
  - generating a random secret key, called pre-Master Secret.
  - encrypts Pre-Master Secret using servers public key and sends it
  - sends certificate for her own public key
  - signs fingerprint of everything exchanged so far (including random strings R and R') and sends it.

SSL Step 4

- Both sides derive Master Secret from Pre-Master Secret, R and R'.
- Server then
  - derives MAC from Master Secret
  - sends client MAC of everything that came before.

SSL Step 5

- Both sides take Master Secret and derive encryption key for each direction and a MAC key for each direction.
- Every subsequent message is encrypted and MACed with sequence number.
- Typical encryption scheme: stream cipher like RC4 -- XOR plaintext with pseudo-random string. Very fast.

Secret Splitting and Sharing

Secret Splitting

Imagine you've invented a new, extra fattening, extra delicious burger sauce that is even more tasteless than your competitors'. This is important -- you have to keep it secret. You can only tell your most trusted employees the exact mixture of ingredients, but what if one of them defects to the competition? There goes the secret, and before long every grease palace on the block will be making burgers with sauce as tasteless as yours.
[Schneier]

Secret splitting -- way to take a message and divide it up into pieces

- each piece by itself means nothing
- with all the pieces in place can reconstruct message.

Simplest secret splitting scheme
T splits a secret between A and B

- T takes the message M ---> two pieces of info and gives one to Alice and one to Bob so that each by itself can't reconstruct M, but together they can.

Secret Sharing

You're setting up a launch program for a nuclear missile. You want to make sure that no single raving lunatic can initiate a launch. You want to make sure that no two raving lunatics can initiate a launch. You want at least three out of five officers to be raving lunatics before you allow a launch. [Schneier]

Secret sharing, threshold scheme -- way to take a message and divide it up into n pieces (shares) s.t.

- any m of them can be used to reconstruct the message.
- With less than m the message can't be reconstructed.

- Called (m,n) threshold scheme.

Shamir -- Interpolating Poly Scheme

- M is the message. Want to construct (m,n) threshold scheme.
- Pick prime p > M. [prime is public.]
- Generate random polynomial of degree m-1.
- Example: (3,n) threshold scheme
  - Generate quadratic polynomial

    $(a x^2 + b x + M) \mod p$
  - Shares are obtained by evaluating the polynomial at n different points

Example: M= 11 (3,5) threshold scheme

- Generate a quadratic equation (7,8 chosen randomly)

  $F(x) = (7 x^2 + 8x + 11) \mod 13$
- The five shares are:
  - k1 = F(1) = (7 + 8 + 11) mod 13= 0
  - k2 = F(2) = (28 +16 + 11) mod 13= 3
  - k3 = F(3) = (63 +24 + 11) mod 13= 7
  - k4 = F(4) = (112 +32 + 11) mod 13= 12
  - k5 = F(5) = (175 +40 + 11) mod 13= 5
- To reconstruct M from three of the shares (say k2,k3,k5) solve the set of linear equations for a,b,M:
  - $(a\ 2^2 + b2 + M) = 3 \pmod{13}$ Solution: a = 7, b = 8, M = 11
  - $(a\ 3^2 + b3 + M) = 7 \pmod{13}$
  - $(a\ 5^2 + b5 + M) = 5 \pmod{13}$

The beauty of it

- (n,m) threshold scheme.
- If the coefficients are picked randomly, m-1 people with infinite computing power can't learn anything about the message => it's equally likely to be anything!

Application: Digital Cash

Digital Cash

- Goal: anonymity
- Desirable properties:
  - forgery hard
  - duplication should be either prevented or detected
  - preserve customers' anonymity
  - inexpensive to implement

First Try

- Withdrawal protocol:
  - allows Alice to obtain a digital coin from Bank
- Payment protocol:
  - Alice buys goods from Vendor in exchange for digital coin
- Deposit protocol:
  - Vendor gives back coin to Bank to be credited on his/her account.

Withdrawal Protocol

- Alice tells Bank she wants to withdraw $100
- Bank returns a signed $100 bill

  S_B(I am a $100 bill, #4527)

  and withdraws $100 from Alice's account.
- Alice verifies the signature and if it is valid accepts the bill.

Payment Protocol

- Alice pays Vendor with bill she got from Bank.
- Vendor checks the Bank's signature and if it's valid accepts the bill.

Deposit Protocol

- Vendor gives the bill to the Bank.
- Bank checks the signature and if it's valid, credits the Vendor's account.

Analysis

- +
  - If signature scheme is secure, can't forge the digital coins.
- -
  - coins can be intercepted
  - easy to duplicate and double-spend digital coins.
  - anonymity not preserved -- Bank can link Alice's name with serial number and figure out where she's shopping.

Solving anonymity problem

- Want blind signatures -- want bank to sign the bill without seeing the contents so can't determine the source.
- Of course, we'll need to worry about how to prevent the bank from signing phony bills...

RSA blind signatures

- RSA signature scheme: $S(M) = M^d \bmod n$
- Suppose M is $20 bill.
- Blind Withdrawal Protocol:
  - Alice chooses a random number $r < n$
  - Alice calculates $M' = M r^e \bmod n$
  - Alice gives the Bank M', requests $20 bill.
  - Bank returns S(M')
  - Bank debits Alice's account for $20
  - Alice computes $S(M) = M^d \bmod n$

Result

- Preserves Alice's anonymity.
- Still two problems:
  - bank can be fooled into signing something it shouldn't
  - coins can still be duplicated and double-spent.

How to avoid fooling bank

- Bank has several public keys, one for each denomination.
- "Cut and choose"
  - Alice makes up 100 $20 bills.
  - Blinds them all and gives all of them to Bank
  - Bank picks one at random to sign and requires that Alice unblind the rest (by revealing r's). Before signature is returned, Bank ensures that all bills unblinded were correct.

How to avoid double-spending.

- Require Bank to record all the bills it receives in database.
- During payment, Vendor transmits bill to Bank and asks if bill already received.
- If bill is being used for first time Vendor accepts, otherwise rejects.

- Not too efficient.

Off-line solution

- During payment, Alice writes Random Identity String (RIS) on the bill.
- RIS properties:
  - different RIS for every payment of coin
  - only Alice can create valid RIS
  - two different RIS on the same coin should allow Bank to retrieve Alice's name.
  - two identical bills with same RIS values -> Vendor has cheated.

- Use one-way hash function H.

Withdrawal Protocol

- Alice prepares 100 bills of $20 which look like

M_i = (I'm a $20 bill, #4527$i$, $h(x_{i,1})$ $h(x'_{i,1})$ ……$h(x_{i,K})$ $h(x_{i,K})$)

where $x_{i,j}$ $x'_{i,j}$ are randomly chosen under condition that

$x_{i,j}$ XOR $x'_{i,j}$ = Alice's name for every i,j

- Alice blinds all the M_i to random messages M'_i and sends them to Bank.
- Bank asks Alice to unblind 99 of them.
- When she unblinds, she also reveals associated $x_{i,j}$ $x'_{i,j}$
- Bank checks that bills are $20, verifies hashes and verifies that $x_{i,j}$ XOR $x'_{i,j}$ = Alice's name for unblinded bills.
- Bank returns signature on blind message M'_{17}
- Alice retrieves signature on bill M_{17}

Payment Protocol

- Alice gives M, $S_B(M)$ to Vendor
- Vendor checks Bank signature on bill and if valid, answers with random bit string $b_1,\ldots,b_K$
- If $b_j = 0$, Alice reveals x_j otherwise, she reveals x'_j
- Vendor computes $H(x_j)$ or $H(x'_j)$ and verifies against M. If everything is fine, he accepts the bill.

Deposit Protocol

- Vendor brings the coin M and S(M) back to bank.
- Bank verifies signature and checks if the coin has already been returned to Bank.
- If it has, Bank compares the RIS's of the two coins. If they're different, Alice has double-spent the coin, otherwise, Vendor (or someone who stole the coin from Vendor) is trying to deposit twice.

Bit Commitment and Mental Poker

Bit Commitment

Bob: "Pick five stocks for me. If they are all winners, I'll give you my business."

Alice: "If I pick five stocks for you, you could invest in them without paying me. Why don't I show you the stocks I picked last month?

Bob: "How do I know you didn't change last month's picks after you knew their outcome? If you tell me your picks now, I'll know that you can't change them. I won't invest in those stocks until after I've purchased your method. Trust me."

Alice: "I'd rather show you my picks from last month. I didn't change them. Trust me."

Alice wants to commit to a prediction (a bit or series of bits) but does not want to reveal them until later.

Bob wants to make sure that Alice cannot change her mind after she has committed to her prediction.

Bit Commitment Using Symmetric Crypto

- Bob generates random R, sends to Alice
- Alice picks a random key K, and sends Bob $E_K(R,b)$ [b is her bit]

When it comes time for Alice to reveal the bit

- Alice sends Bob K.
- Bob decrypts $E_K(R,b)$ and checks validity of R.

Bit Commitment Using One-Way Functions

- Alice generates two random strings R and R'
- Alice sends Bob f(R, R', b) and R.

When it comes time for Alice to reveal her bit

- Alice sends Bob the message M = (R, R', b)
- Bob computes f(M) and compares it and R with the values he previously received. If they match, the bit is valid.

Mental Poker

Step 1: coin-flipping over email.

Assumes commutativity of algorithm.

- A and B each generate a public-key/private-key pair.
- A generates two random strings H and T
- She encrypts them and sends them to Bob

  E_A (H), E_A(T)
- Bob picks one of these at random, call it M (either H or T) encrypts and sends back to A.

  E_B (E_A (M))

- Alice who can't read it, decrypts with her private key and sends it back to Bob

  D_A (E_B (E_A (M))) = D_A (E_A (E_B (M))) = E_B (M)
- Bob decrypts message to reveal result of coin flip

  D_B (E_B (M)) = M and sends it to Alice.
- Alice reads the result of the coin flip and verifies that random string is correct.
- Both Alice and Bob reveal their key pairs so both can verify the other didn't cheat.

Why does it work?

For A and B to play poker by email.

- Instead of making and encrypting two messages, H and T, she makes 52 messages, one for each card in the deck.
- Bob chooses 5 at random, encrypts them with his public key and sends them back to Alice.
- Alice decrypts the messages and sends them back to Bob who decrypts them to determine his hand.
- He then chooses 5 more at random sends to Alice. She decrypts them and they become her hand.
- During the game, additional cards can be dealt using the same procedure.
- At the end of the game, A and B reveal their cards and key pairs so each can be assured the other didn't cheat.

Zero Knowledge Proofs

Zero-knowledge Proofs

Alice: "I know the password to the Federal Reserve System computer, the ingredients in McDonald's secret sauce, and the contents of Volume 4 of Knuth."

Bob: "No, you don't"

Alice: "Yes, I do"

Bob: "Do not!"

Alice: "Do too!"

Bob: "Prove it!"

Alice: "All right. I'll tell you." She whispers in Bob's ear.

Bob: "That's interesting. Now I know it too. I'm going to tell *The New York Times*."

Alice: "Oops."

Zero-knowledge Proofs

- A prover (Peggy) wants to convince a verifier (Victor) that she has a certain piece of information without giving Victor any information about the information.
- Use interactive protocols

Graph Isomorphism

- Peggy knows an isomorphism between two graphs G1 and G2. She wants to convince Victor that she knows an isomorphism between them without showing the isomorphism.

Interactive Proof of Graph Isomorphism

- Peggy randomly permutes G1 to produce H.
- Peggy sends H to Victor
- Victor asks Peggy to either prove that H and G1 are isomorphic, or that H and G2 are isomorphic.
- She complies.
- Peggy and Victor repeat this process n times.

Application to User Identification

- ZK proofs provide revolutionary new way to realize passwords.
- User stores statement of theorem in his publicly readable directory, the proof of which only he knows.
- To login, the user engages in ZK proof of correctness of theorem.
- If proof is convincing, access permission is granted.
- Adversary who overhears ZK proof can't learn enough to gain unauthorized access.

Other

Electronic Elections

- M people each with private input xi, want to compute the result of f(x1,x2,..., x) without revealing values
- Desirable properties:
  - only authorized voters can vote
  - no one can vote more than once
  - secrecy of votes is maintained
  - no one can duplicate anyone else's vote
  - the tally is computed correctly
  - anybody should be able to check that the tally is computed correctly
  - protocol should be fault tolerant (works even with "bad" parties).
  - shouldn't be possible to coerce a voter into revealing how she voted.

Other Amazing and Fun Stuff

- Simultaneous contract signing
- Age problem
- Secure multi-party computations
- Electronic elections
- ….

Pseudo-random number generators

Pseudo-random number generators.

- Generation of random numbers critical to cryptographic systems.
  - Symmetric ciphers such as DES, RC2, and RC5 all require a randomly selected encryption key.
  - Public-key algorithms — like RSA, Diffie-Hellman, and DSA — begin with randomly generated values when generating prime numbers.
  - At a higher level, SSL and other cryptographic protocols use random challenges in the authentication process to foil replay attacks.

What is a random number?

- philosophical question
- "*Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin*" John von Neumann
- computers deterministic
- any random number generator on a computer is periodic.
- anything that is periodic is not random.
- anything that is predictable is not random.

Pseudo-random number generators.

- A (k,l) PRBG (pseudo-random bit generator) is a function f: k bit strings ------> l bit strings (l > k).
- The input k-bit string s_0 is called the seed and the output l-bit string $f(s_0)$ is the pr bit-string.
- Statistical randomness: passes all the statistical tests of randomness that we can find.
- Should have high degree of **entropy**

Linear Congruential Generator

- Let M >= 2, 0 <= s_0 <= M-1. (k = log M)
- Define

$$s_i = (a\, s_{i-1} + b) \bmod M$$

then $f(s_0) = (z_0 z_1, \ldots, z_i)$
where $z_i = s_i \bmod M$

- If a,b,M properly chosen, generator has period of M

Linear Congruential Generators

- Fast
- Useful for many applications, but not crypto…
- What does it mean for a PRBG to be secure?
  - It should be impossible in probabilistic polynomial time to distinguish l pseudo-random bits produced by the PRBG from l truly random bits.
  - <=> Not possible to predict the next bit of output from the PRBG with probability > $1/2 + \varepsilon$, except for very small ε.

- Linear congruential generator have good statistical behavior with respect to most reasonable empirical tests.
- However, doesn't pass the next-bit test.
- (nor does any polynomial congruential generator).

Linear Feedback Shift Registers

- A shift register and a feedback function.
- For LFSR, the feedback function is XOR
- Output sequence is the string of least significant bits.
- Period of an N-bit LFSR is at most

To be a maximal-period LFSR

- Polynomial formed from a tap sequence (the bits that are XORed) plus the constant 1 must be a primitive polynomial mod 2. (degree = length of shift register)
- Example: (32,7,5,3,2,1,0)
- Primitive polynomial of degree n is an irreducible polynomial that divides x^(2^(n-1)) + 1, but not x^d+1 for any d that divides 2^(n-1).

Properties of LFSRs

- Best when feedback polynomial is dense.
- Competent pr generator, but not cryptographically secure in the sense we want.
- Nonetheless, most practical stream-cipher designs center around LFSRs.

Insecurity of LFSR

- Show that knowledge of any 2k consecutive bits of output allows the seed to be determined.

Last but not least, the Blum,Blum Shub. First, background.

- Let p be an odd prime. $1 <= x <= p-1$.
- x is defined to be a quadratic residue modulo p if the congruence $y^2 = x \pmod p$ has a solution y in Z_p.
- x is a quadratic non-residue modulo p if x is not congruent to 0 (mod p) and x is not q quadratic residue mod p.

- Quadratic residues modulo 11 are 1,3,4,5,9.
- $1^2 = 1 \quad 5^2 = 3 \quad 2^2 = 4 \quad 4^2 = 5 \quad 3^2 = 9$
- $6^2 = 3 \quad 7^2 = 5 \quad 8^2 = 9$ ....

More background

- Euler's criterion: Let p be an odd prime. Then x is a quadratic residue modulo p iff

  $x^{(p-1)/2} = 1 \bmod p$
- => efficient algorithm for testing quadratic residuosity mod a prime.
- Legendre symbol L(a,p)

$$L(a,p) = \begin{cases} 0 & \text{if } a = 0 \bmod p \\ 1 & \text{if } a \text{ is a quadratic residue mod } p \\ -1 & \text{if } a \text{ is a quadratic non-residue mod } p \end{cases}$$

=> $L(a,p) = a^{(p-1)/2} \bmod p$

Last bit

n = pq. The Jacobi symbol J(x,n)

$$J(x,n) = L(x,p)\, L(x,q)$$

$QR(n) = \{ x^2 \bmod n \mid x \text{ in } Z_n^* \}$

x is in QR(n) if and only if $L(x,p) = L(x,q) = 1$.

$QR\tilde{}(n) = \{x \text{ in } Z_n^* \mid L(x,p) = L(x,q) = -1 \}$

Quadratic Residues Problem

- Given a positive integer n that is the product of two unknown primes p and q, and an integer x in Z_n^* such that J(x,n)= 1, determine if x is a quadratic residue modulo n?
- No more difficult than factoring. Appears to be intractable.

Blum-Blum-Shub Generator

- Let p,q be two k/2 bit primes such that p = q = 3 mod 4 and define n = pq. Let QR(n) denote the set of quadratic residues modulo n.
- A seed s_0 is any element of QR(n)

$$s_{i+1} = s_i^2 \bmod M$$

then $f(s_0) = (z_0 z_1, \ldots, z_l)$
where $z_i = s_i \bmod M$

Security of BBS generator

- If output can be ε-distinguished from true random bits, then there is a probabilistic algorithm for quadratic residue problem.
- Such an algorithm not believed to exist.
- Efficiency improvement -- extract more than one bit each time (log log n).

Summary: Security Risks of Internet Communication

- Eavesdropping
  - intermediaries listen in on private conversations
  - Solution: encryption (public or private-key)
- Manipulation
  - intermediaries change information in a private communication
  - Solution: methods for preserving data integrity (one-way hash fn's and MACs)
- Impersonation
  - a sender or receiver communicates under false identification
  - Solution: authentication (signatures, etc.)

Summary

- It's a brave new world.