Secure Internet Communication and Commerce

Amir Herzberg
E-Business and Security Technologies
IBM Haifa Research Lab - Tel Aviv Annex

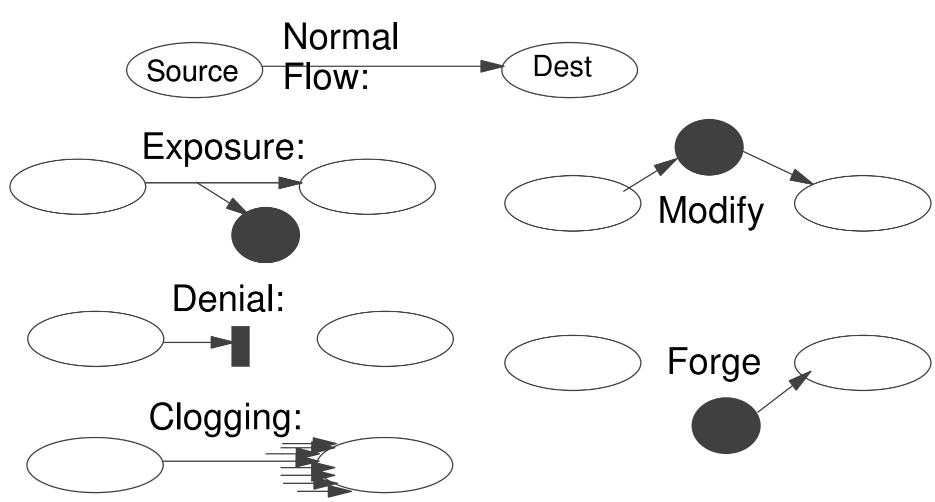
Agenda - Secure Internet Communication and Commerce

- Secure Communication
  - SSL
  - IP-SEC
  - Other
- Break
- Secure Payments
  - Credit Cards: SSL, SET, others
  - Micropayments: MiniPay

Agenda - Secure Communication

- ➢ Secure Communication
  - Threats and security services
  - Security functions - at which network layer?
  - Firewalls
  - IPSEC
  - SSL
  - Secure e-Mail: PGP, S/Mime, Moss
- Secure Payments

Network Security Threats

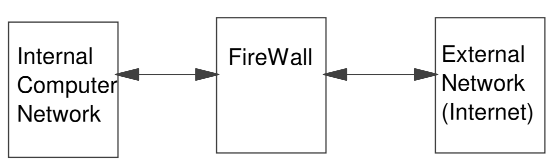


Network Security Services

- Entity authentication: identifying parties
- Message authentication/integrity: data was not modified - delivered as sent by party
- Confidentiality: protecting data against eavesdropping
  - Privacy: protecting location, actions, identity
- Non-Repudiation: proof of action to third party and/or at later time (e.g. signature)
- Availability: preventing clogging, denial of service
- Access control: limiting access to resource / data

Firewalls

- Firewall: a trusted machine which limits access from one domain to another.

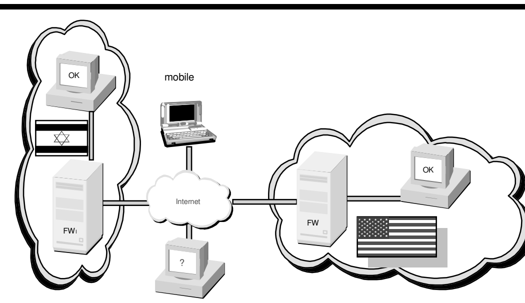


FireWall functions

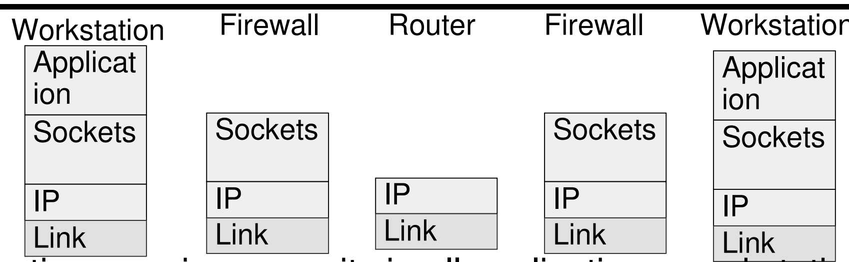
- FireWall is more secure than other machines:
  - Security-focused management
  - Minimal set of programs and users
- FireWall tries to filter out bad packets:
  - Prevent use of insecure protocols
  - Check for known attacks (e.g. scan for viruses)
  - Limit (some) protocols to `one-way` (invoked from within)
  - Filtering at packet, socket or application layer

Secure Tunnels across Internet

- Threat: inject/eavesdrop
- Solution: encrypt and authenticate end to end or firewall to firewall
- Which layer?
  - higher:
    - easy to implement
    - understand semantics
  - lower: more secure, efficient
    - can implement in firewall (easier to deploy)



Security vs. Network Layers



- Application: requires security in all applications, workstations, and lower layers... but knows semantics (sign, selective encrypt)
- Sockets: easy to implement and deploy.... SSL, PCT
- IP (Internet Protocol): more secure, but change to kernel
- Link: impossible (Internet routers not trusted)

IP Security Features

- Lowest layer - protect against impl. bugs
- Trustworthy, conservative design
- Modularity: replace broken/custom modules
- Limited protection against denial of service
- Anonymity
- Protections against key exposure
  - Forward secrecy: past traffic protected from key exposure in the future
  - Limited recovery: replace keys periodically (proactively i.e. without detecting break-in). Attacker may lose control.

IP Security: High-Level Design

- Anti-Clogging exchange
  - Receiver selects random cookie
  - Sender must append cookie to every msg
- Key Sharing (send key encrypted by RSA)
- Authenticated Key Exchange (DH)
  - Select new key every time
  - Forward secrecy; limited recovery from exposure
- Periodical Shared-Key Refresh

Perfect Forward Secrecy
[D90,DOW92]

Key	Key 1	Key 2	Key 3	Key 4
Period	Period 1	Period 2	Period 3	Period 4

PFS: exposure of Key3 does not reveal Keys 1,2.

How? Periodical Key Agreements [DH76]:

Select x
calculate g^x $\xrightarrow{g^x}$ Select y
calculate g^y

$\xleftarrow{g^y}$

calculate $K=(g^y)^x=g^{xy}$ calculate $K=(g^x)^y=g^{xy}$

assumes discrete log is hard

Proactive IP Security

Key	Key 1	Key 2	Key 3	Key 4
Period	Period 1	Period 2	Period 3	Period 4

Goal: exposure of Key3 does not reveal Keys 1,2, or 4

Assume: not all machines broken at once

How? [CHH96]:

- ➢ At each period, select new public/private keys
- ➢ Broadcast new public key
- ➢ Confirm receipt by a proactive group signature [HJJKY95,...]:
  - Each machine has fraction of secret group key
  - Secret fractions re-distributed every period (old=worthless)
  - Public group key is fixed, in ROM of machines (like program)

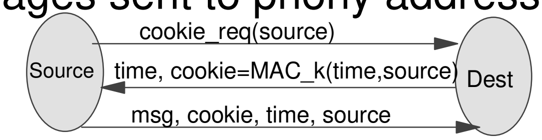
IP-SEC Key Distribution Protocol

SHARE	EXCH	AUTH
A -> B: PKEB (KA)	A -> B: gx mod p	A -> B: FK0 (gy,gx, idA, idB)
B -> A: PKEA (KB)	B -> A: gy mod p	B -> A: FK0 (gx,gy, idB, idA)
K0 = H (KA, KB)		

Or combined unto three messages:
A -> B: PKEB (idA , KA), gx
B -> A: PKEA (KB), gy , FK0 (gx, gy, idB, idA)
A -> B: FK0 (gy, gx, idA, idB)
SK := H (gxy mod p)

Cookies - Protection against Clogging

- Threat: hacker `clogs` server with many requests - loss of time (pub-key ops) or space
- Goal: identify address of hacker and ignore it
- Hacker will use phony source address...
- Assumption: hacker cannot intercept messages sent to phony address

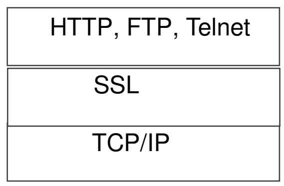


IP Security - Status

- Proposed standard for packet formats
- Key management converging
- Good performance
- Available in IBM Firewall (Internet Secure Connection)
- S/WAN: interoperability test run by RSA inc
  - IBM has best interoperability (all vendors!)
- Key contributor - Hugo Krawczyk

SSL - Secure Socket Layer

- Proposed by NetScape, now at ver 3 & proposed standard
- Motivation: IP security requires change of OS -- difficult!
- Solution: Socket-layer interface
  - Common to almost all TCP/IP stacks
  - Application layer code - highly portable, easy install
  - Operates between applications and transport (TCP)



SSL Functions

- Key exchange: set up shared key from signed digital certificates (public keys)
- Key refresh (repeat exchange)
- Entity authentication using certificates:
  - Server authenticated to client
  - Client authenticated to server (optional)
- Message authentication (integrity)
- Encryption

SSL Certificates

- Site certificates
  - Defined/viewed in browser
    - Options > Security > Site certificates
  - Site Ceritificates and Certificate Authorities
    - Several predefined certificate authorities: VeriSign, MCI, AT&T, Thawte,...
    - Site Certificate: sign_CA(cert_url, cert_PK)
  - Risk: some CAs do not validate identity; attacker gets sign_CA(bank_url, attacker_PK), and then can decrypt password sent to bank.

SSL Certificates

- Personal certificates
  - Signature over client's public key
  - Viewed in browser
    - Options > Security > Personal certificates
  - Received from Certificate Authorities
    - VeriSign (thru NetScape): class 1 (free) or class 2 (12$/year, limited verification of ID)
    - Others (AT&T, MCI,...) - thru their site, using undocumented HTML tags
  - Use: server authenticates client securely.

SSL Protocol (simplified)

- Client sends `hello`
- Server responds with certificate
- Clients sends Pre-Master key (48B) - encrypted with server's key.
- Pre-Master key used to derive encryption and authentication keys
- Encryption key is 128bits in US, 40 bits worldwide; use RC4 encryption (from RSA)

Secure E-Mail

- Functions:
  - Encryption
  - Signatures
- Main challenges:
  - Certificates
  - Availability at recipient
  - Interoperability (or lack of) - many competing proposals: PGP, S/MIME, MOSS (PEM), RIPEM, Lotus Notes, CC: mail...

Some Secure E-Mail Systems

- S/MIME from RSA:
  - Wide support in industry (e.g. Netscape)
  - Supports MIME (mail attachments)
  - Interoperability among S/MIME implementations
  - Security concerns.
- PGP: popular freeware from Phil Zimmerman
  - Originally w/o MIME, now two proposals w/ MIME
- MIME Object Security Services (MOSS)
  - Currently defined in RFCs (Requests for Comment) 1847 and 1848.
- Lotus Notes, cc:mail: popular in industry.

Agenda - Secure Payments

- Secure Communication
- ➤ Secure Payments
  - Payments on the Net: Why?
  - First steps: First Virtual, SSL, ...
  - Credit cards: SET (& iKP, STT, ...)
  - Micro-Payments: MiniPay
  - Other payment schemes
- Java and JECF

Secure Payments on the Net

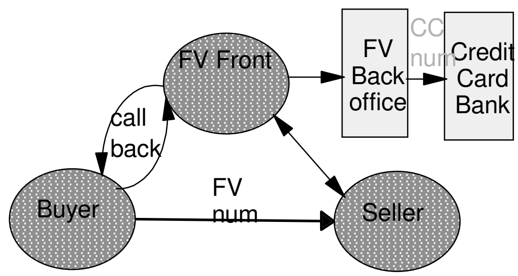
- Internet as global marketplace
- Security is a major concern
- Secure payments - a critical enabler
- Goals:
  - Open market, inter-operability, standards
  - Migration from established solutions
  - 20/80 rule, or: pick low-hanging fruits first
  - Keep It Simple
- Credit Cards first...

Goals for Net Payments

- Security
  - Main threat: hackers collecting credit card #s
  - Solutions: avoid cc#s (FV) or encrypt (SSL, SET)
- Overhead: cost and delay
  - Low cost and delay required for Net applications:
    - Info per click, games, services, applets
  - Solutions: MiniPay, CyberCoin, MilliCent,...
- Anonymity: protect buyer's privacy
  - Solutions: digital cash (DigiCash,...), or limited

First Net Payment Solutions

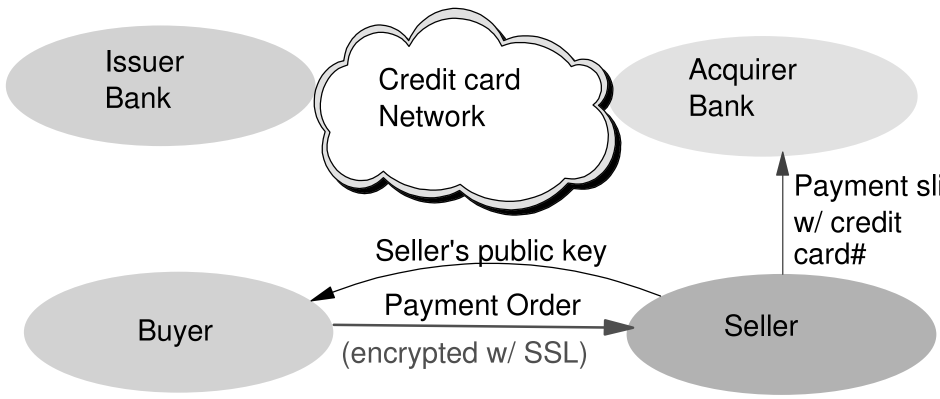
- First Virtual (Oct 94)
  - Problem solved: sending credit card on Net
  - 1. Use a different account number on Net
  - 2. E-mail confirmation call-back
    - Assumption: hard to intercept e-mail
  - No cryptography



Secure Payments with SSL

- SSL: Secure Socket Layer (NetScape, 94)
- Encryption and authentication between Web browser and server (=seller)
- Common use: send encrypted credit card#
- Remaining problems:
  - Use of server as oracle to test guesses of card #s
  - Seller server certification, fraud by seller
  - Seller server and seller-to-bank security
  - Non-repudiation (signature on slip)

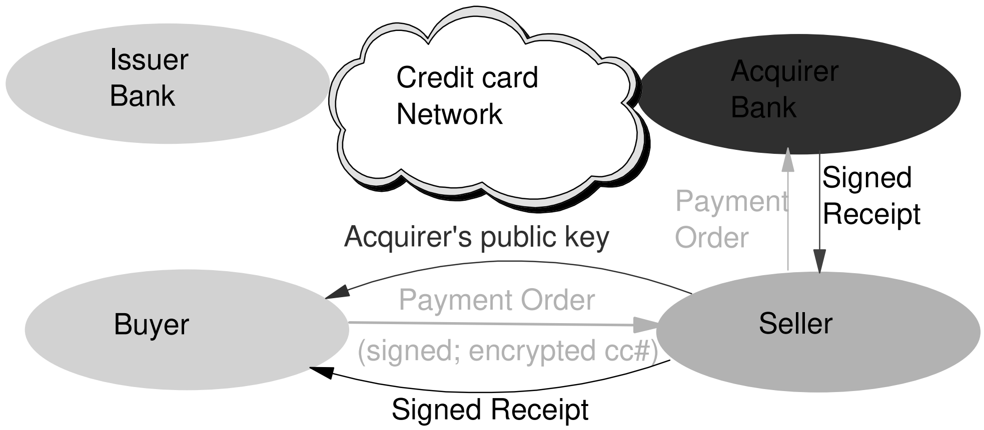
Credit Card Payments w/ SSL



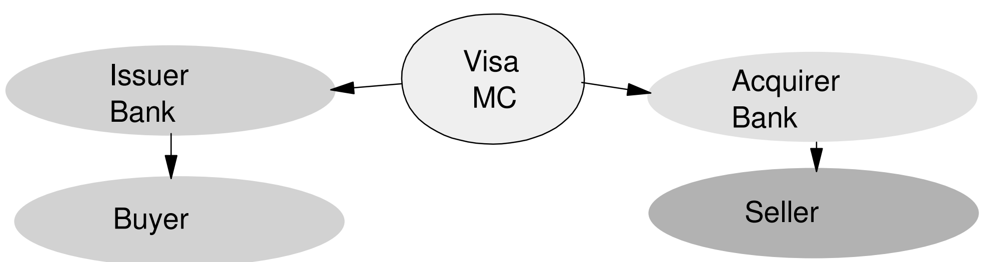
Evolution of SET

- Nov 94: Visa & Microsoft announce STT
- March 95: IBM publish iKP; EuroPay joins
- June 95: CyberCash publishes their spec
- July 95: IETF BOF on payments
- Sept 95: Visa & MicroSoft publish STT
- Oct 95: MasterCard, IBM, NetScape, GTE, CyberCash publish SEPP
- Feb 96, June 96: SET by MasterCard+Visa
- Products: 1H 1997 (IBM, Verifone, others)

Credit Card Payments with SET, iKP, STT, CyberCash



Distributing Public Key Certificates



Payment Order=cert_Buyer, E_Acq(cc#, salt),
Sign_buyer($,h(goods),time,Seller)

Credit Cards are great, but...

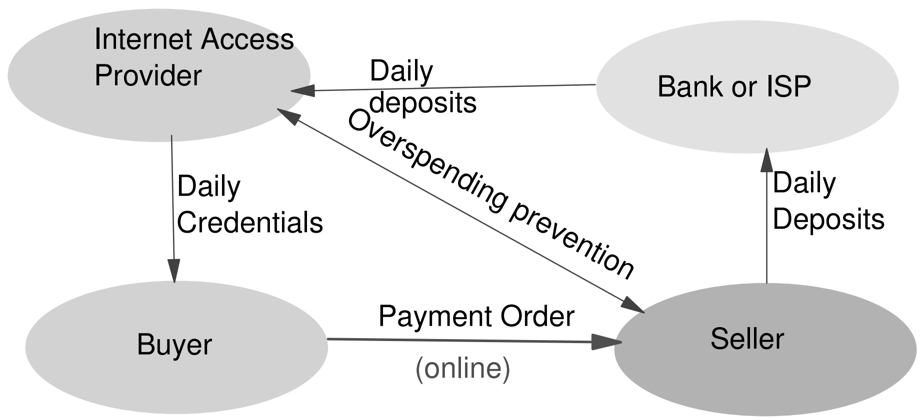
- High fees:
  - Minimal: about 20 cents
  - As percent: 2%-6%
- Difficult to become merchant
- Not available to weak populations (China)
- Substantial delay
- Confirmation snag to `click-and-pay`

How to pay a cent per click?

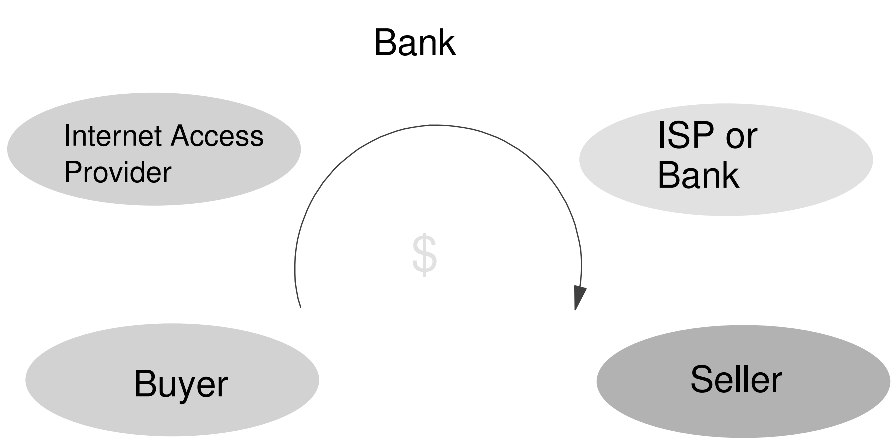
Click and Pay with Mini-Pay

- **Low Cost**
- **Universality & Scalability**
  - Multiple interoperable systems, open design
- Ease of use: buying, selling, managing
- Negligible delay
- No special hardware required
- No extra messages
- Security: contain forgery, overspending; prevent denial of service; non-repudiation

Mini-Pay Architecture

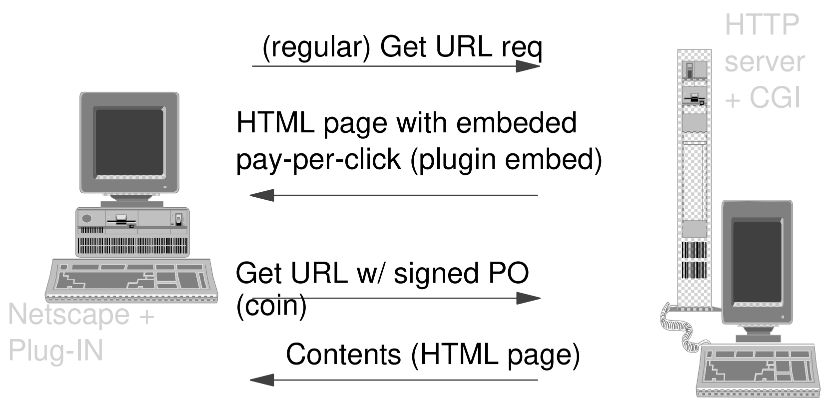


Financial flow and parties



Payment Process Flows

Payment info added to existing browsing messages:



Short-Lived Certificates

- Goal: most payments done offline
- Method: buyer gets a daily certificate: "good for up to 1$ today".
- Over 1$ - merchant asks for extension.
- Transparent to user.
- Saves: communication, computations, bottleneck on payment system.
- Problem: overspending... (see next foil)

Over-Spending Prevention

- Daily per seller: online verify over x$
  - x$=Fraction of total limit (account balance)
  - Exceeded - ask for extension (online)
- 1000-sellers attack: spend limit (1$) *1000
  - Limited risk to seller
  - Best solution: total limit by smartcard / proxy
  - Limited protection: by customer software
  - Inherent risk to off-line payments

Limited Consumer Protection

- All sales are final - no refunds, no returns
  - Seller may offer refunds/returns
- Pay once, guaranteed delivery:
  - Reload does not charge again
  - Same for clicking again on same page
  - Policy: allow repeated clicks (for a day)
    - Exceptions, e.g.: pay per service / game
- Results in legal limit on transaction (25$)

Risk Management

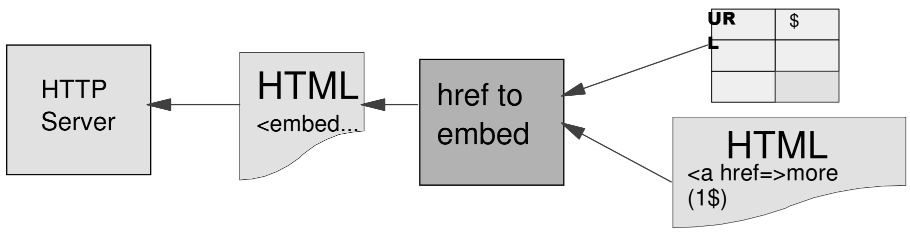
- Credit risk: buyer does not pay IAP
  - IAP still needs to pay ISP, Seller
  - IAP may protect itself - only prepaid
- Over-spending risk (1000-seller attack)
  - Seller loses up to offline limit
  - Also: local software bound, proxy server
- Merchant fraud: N/A (buyer risk)
- `Stolen account`: buyer risk (up to 50$)

Interface: Click and Pay

- Extend existing link interface:
  - Existing (free) link
  -
  - For-fee link
- Cues for cost - sound, color, pop-up
  - Implemented: changing cursor, amount in message area of browser

Migration tool from existing HTML

- Existing hyper-links done w/ <A HREF=...> tag (anchor)
- Pay-per-click hyper-links implemented by <EMBED> tag (invoke plug in, with amount and URL as parms)
- Auto-generated from existing pages and cost table
- Run as pre-processor over (static) HTML pages



MiniPay Competitors

- CyberCoin (from CyberCash)
  - Very centralized - account with CyberCash
  - Less scalable
  - Pop-up screen to approve (not click'N'pay)
- MilliCent (from Digital)
  - Avoids public key operations - but at cost of added complexity and communication
  - Centralized
  - Pop-up screen to approve (above threshold)

Other Works on Payments

- E-Check (FSTC): a full check mechanism
- Eliminate signature: MilliCent, MicroMint, NetBill,... --- cost: more communication!!
- Reuse signatures: 1$=(sign(1$,h(x)); 2$=x
- Faster signature operations
- MiniPay: one signature, two verifications
  - Can reduce... but not needed!!
- Anonymity: DigiCash/E-Cash variants

Micro-Billing vs. E-Cash

- Anonymity: strong (DigiCash), weak, none.
  - A feature but also potential legal problem.
- Anonymity is not for free: time, complexity
- DigiCash already deployed, but few sellers
- DigiCash: `no fee for payment', but...
  - Attacks on anonymity, availability
  - Reality: each coin is used once
  - Substantial fees to move in/out and monthly
  - Substantial delay and complex user interface

Status and Forecast

- 1998 will see secure payments as reality
- SET (secure credit card payments):
  - Published standard (by Visa, MasterCard)
  - Implementations
    - IBM, Terisa, HP/Veriphone, CyberCash
- Micro Payments
  - Market is ready

For more information...

- Our internal page: w3.research.telaviv.ibm.com
- Our external page: www.ibm.net.il/ibm_il/int-lab
- Great source and archive of security info: http://www.cs.purdue.edu/homes/spaf/hotlists/csec-top.html
- NetScape Developer's Edge Archive: http://developer.netscape.com/library/archivedconf/index.html
- Inside Internet E-Mail Encryption: http://www.data.com/Tutorials/Emailencrypt.html
- The IETF IP Security Working Group News: http://www.cs.arizona.edu/xkernel/www/ipsec/ipsec.html
- For SET: www.mastercard.com