Practical Cryptography and Autonomic Web Computing

John Black

University of Colorado, Boulder

IFIP WG 10.4
January 29, 2005
Rincón, Puerto Rico

Issues Exciting to *Theoretical* Cryptographers

- Primes 2 P?   // yes [AKS02]
- (Extended) Riemann Hypothesis
- P = NP?
- Factoring $\cdot_p$ RSA?

Issues Exciting to *Practical* Cryptographers

- Key Distribution
- Factoring 2 P?
- Secure hashing
- Fast Crypto
- Crypto for Constrained Environments

Key Distribution

- Chicken and Egg
  - If we had a secure infrastructure, we could distribute keys securely
- Would solve a lot of major problems
  - ARP and DNS poisoning
  - SSH/SSL/IPSec
    - CA structure is far from ideal trust model
  - DDoS attacks
    - Though privacy types would protest if we traced every IP packet
    - Is the crypto fast enough for this (more later)

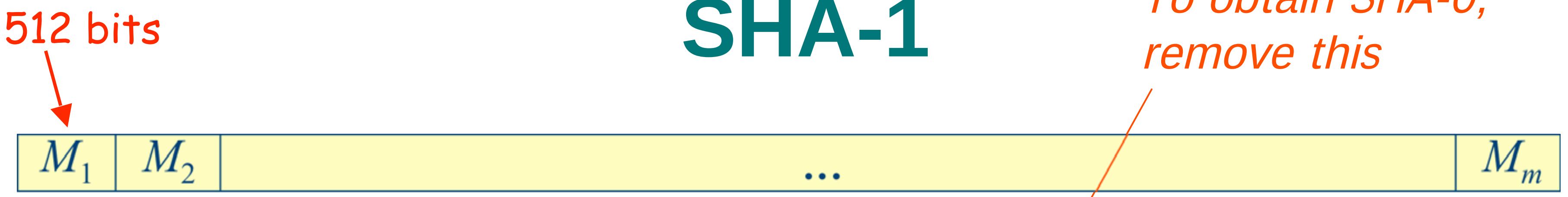
Factoring 2 P?

- Efficient factoring breaks RSA (and others)
- Twinkle
  - Spinning Mirrors
- Integer Factorization Circuits, TWIRL
  - 512-bit RSA modulus: 10 mins, \$10K
  - 1024-bit modulus: < 1 yr, \$10M
- Quantum Computers

Secure Hashing

- Important, useful objects
- Thin theoretical foundations
  - Blockcipher-based methods from 80’s
  - Few proofs
- Differential attacks [Wang et al, 2004]
  - SHA-0, MD5, and others “broken”
- SHA-1 appears safe still [Rijmen05]
  - Can break 53-round SHA-1 with < 2^{80} work

SHA-1



To obtain SHA-0, remove this

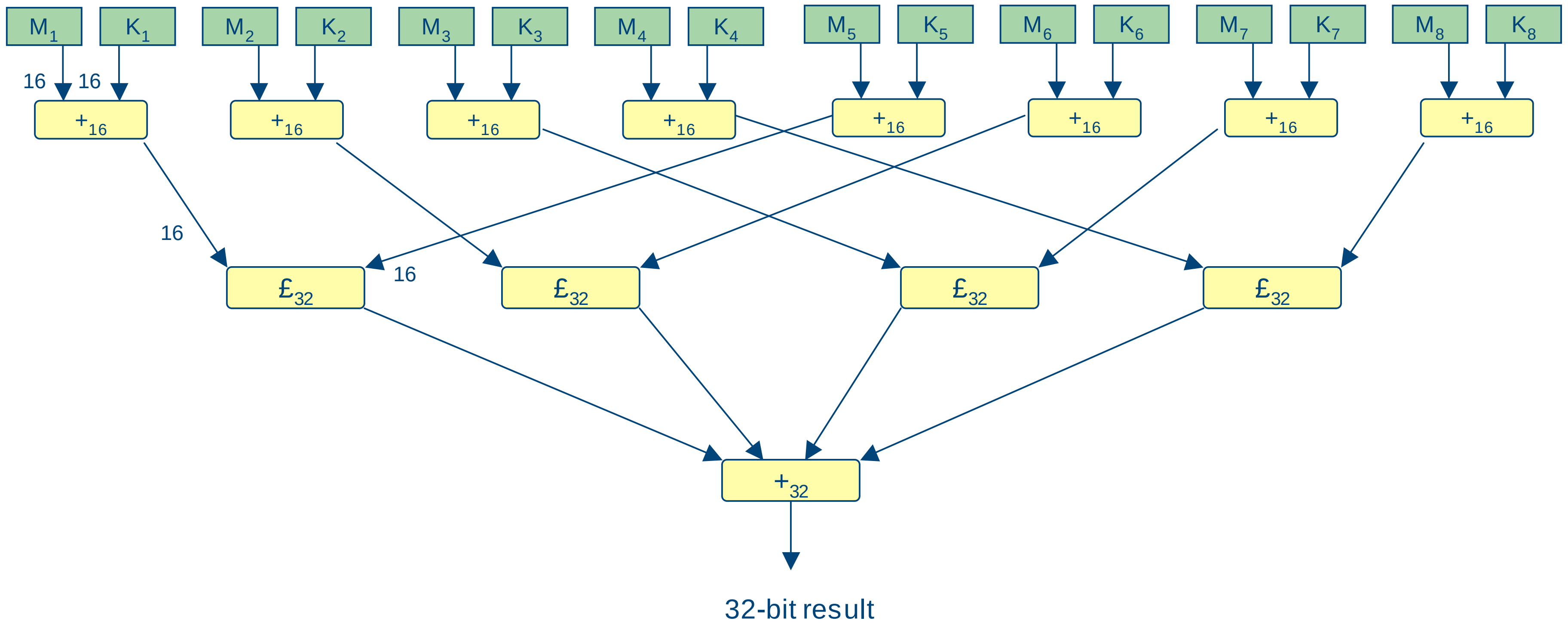
```
for $i$ = 1 to $m$ do
    $W_t =$
    $\left\{ \begin{array}{ll} t\text{-th word of } M_i & 0 \leq t \leq 15 \\ ( W_{t-3} \copyright W_{t-8} \copyright W_{t-14} \copyright W_{t-16} ) \boxed{<< 1} & 16 \leq t \leq 79 \end{array} \right.$
    $A \leftarrow H_0^{i-1}$;   $B \leftarrow H_1^{i-1}$;   $C \leftarrow H_2^{i-1}$;   $D \leftarrow H_3^{i-1}$;   $E \leftarrow H_4^{i-1}$

    for $t$ = 1 to 80 do
        $T \leftarrow A << 5 + g_t(B, C, D) + E + K_t + W_t$
        $E \leftarrow D$;   $D \leftarrow C$;   $C \leftarrow B >> 2$;   $B \leftarrow A$;  $A \leftarrow T$
    end

    $H_0^i \leftarrow A + H_0^{i-1}$;   $H_1^i \leftarrow B + H_1^{i-1}$;   $H_2^i \leftarrow C + H_2^{i-1}$;
    $H_3^i \leftarrow D + H_3^{i-1}$;   $H_4^i \leftarrow E + H_4^{i-1}$

end
return $H_0^m\ H_1^m\ H_2^m\ H_3^m\ H_4^m$
```

160 bits

Fast Parallelizable Crypto

- Slow and serial crypto is an impediment
  - High-end web servers
  - Routers
- Recent research has sought to find fast (and parallelizable, often) algorithms
  - AES is much faster than DES was
  - HMAC, UMAC, Poly1305
  - Many AE schemes (OCB, CWC, EAX, etc)
- Proof-of-correctness now a requirement
  - Some are skeptical about the value of this, but none suggests it's better than no proof at all

The Heart of the UMAC algorithm



M_1 K_1 M_2 K_2 M_3 K_3 M_4 K_4 M_5 K_5 M_6 K_6 M_7 K_7 M_8 K_8
16
16
$+_{16}$
16
16
$£_{32}$
$+_{32}$
32-bit result

The above represents **three** MMX instructions (2 **paddw**'s and a **pmaddwd**)

Crypto in Constrained Environments

- We can do standard crypto on a laptop
  - But a cell phone has a lot fewer cycles to spare
    - Indeed, they've blown it a few times already
  - Sensor nodes have ever fewer (and radio constraints)
  - RFIDs present an extreme challenge
- We need simple algorithms, even if they don't provide industrial-strength cryptography
  - TinySec [KSW04] is a start

And What Virtually No Cryptographers Find Exciting…

- Software Engineering and Education
  - In my opinion, where a lot of security problems start
- Software Engineering:
  - Security was not “built in” from the start
    - More examples than non-examples
  - Software not built according to “best practices”
    - Every vulnerability is a bug, so security is really a *quality* problem
    - Code is not agile, so when something breaks (eg, PKCS #1) it’s hard to plug in something new

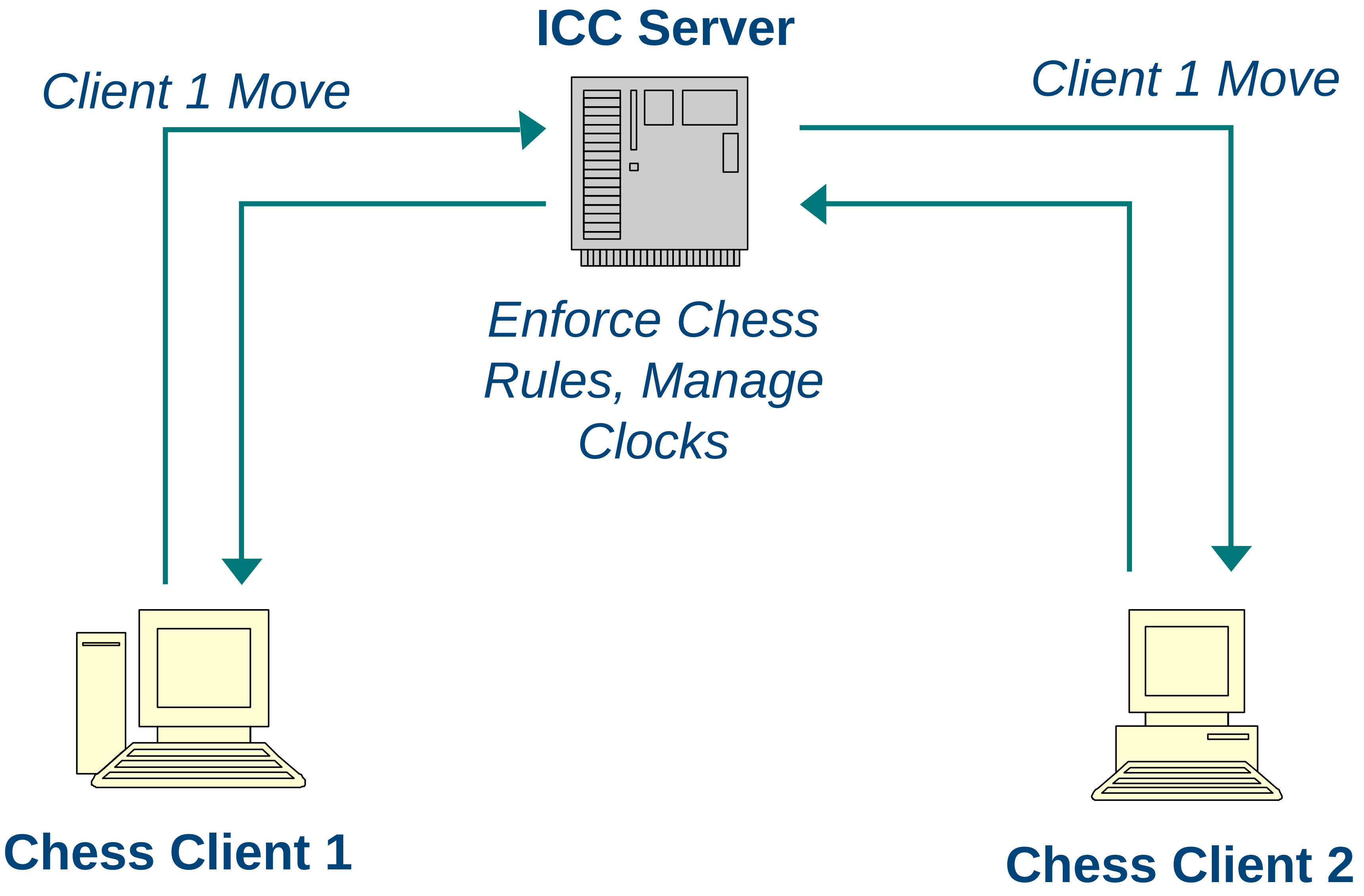
Education

- Students emerge with a degree in Computer Science with little to no training in security
  - Not a standard part of most curricula
  - Not enough knowledgeable people available to train students
- On the crypto side, two important themes
  - 1) Don’t create your own crypto; you’ll get it wrong
    - Example: Internet Chess Club
  - 2) Perfectly good crypto primitives get misused and are rendered worthless
    - Example: MS Office

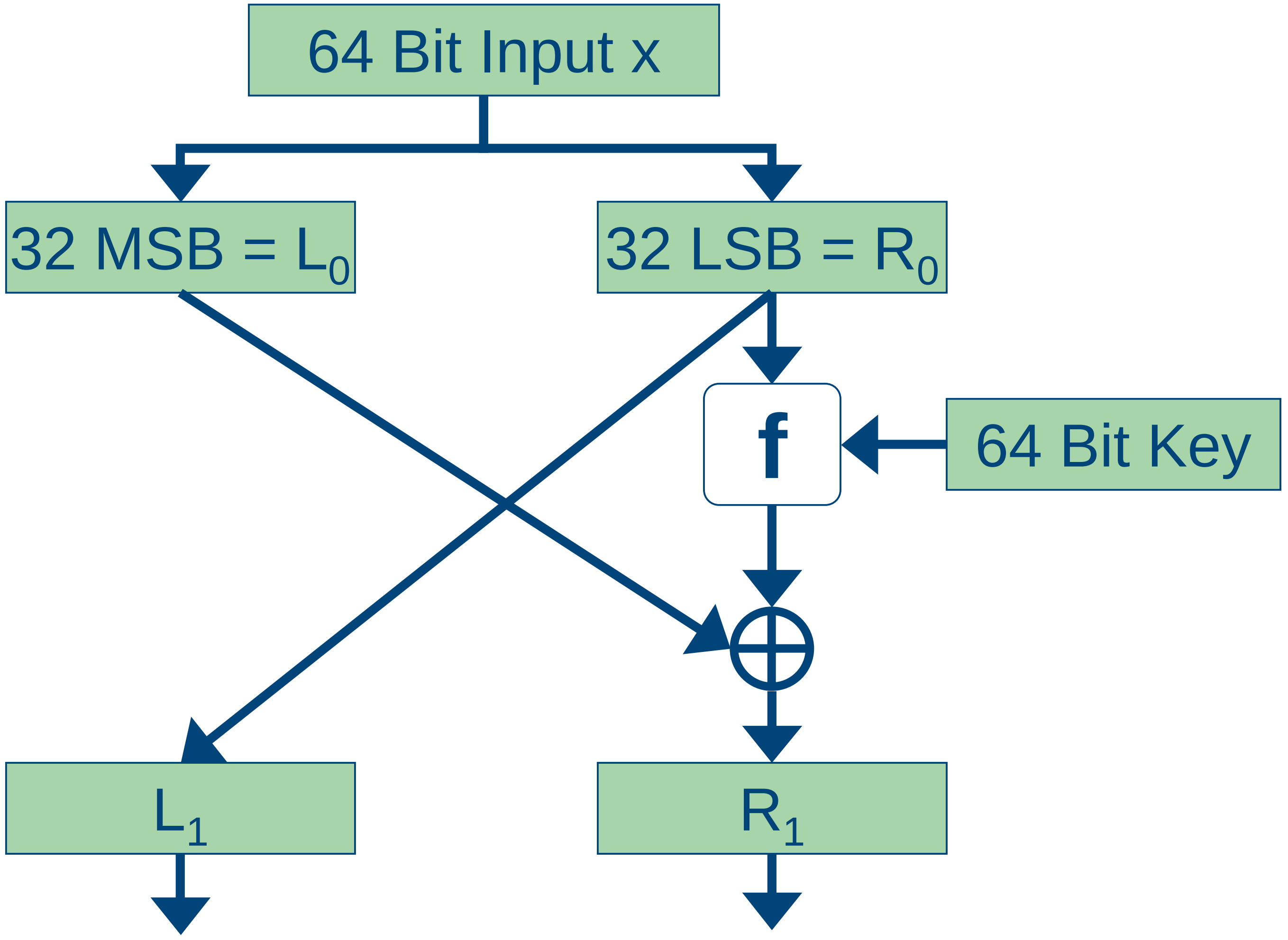
Internet Chess Club

- Over 30,000 members
- Pay Site ($60/year)
- Madonna, Nicholas Cage, Will Smith, Sting, even Kasparov
- Best choice for online chess
- Written by and run by a CMU CS Professor
  - Specializes in theory, interested in cryptography

Basic Idea


ICC Server
Client 1 Move
Client 1 Move
Enforce Chess Rules, Manage Clocks
Chess Client 1
Chess Client 2

Block Cipher



16 Rounds

$$f(V,K,r) = S[V_0 + V_1 + K_{r \bmod 8} \bmod 256] + V^2$$

Mode of Operation

- Pad formed by XOR of two LCGs

$$x_{n+1} = 3x_n + 1 \bmod 43060573$$

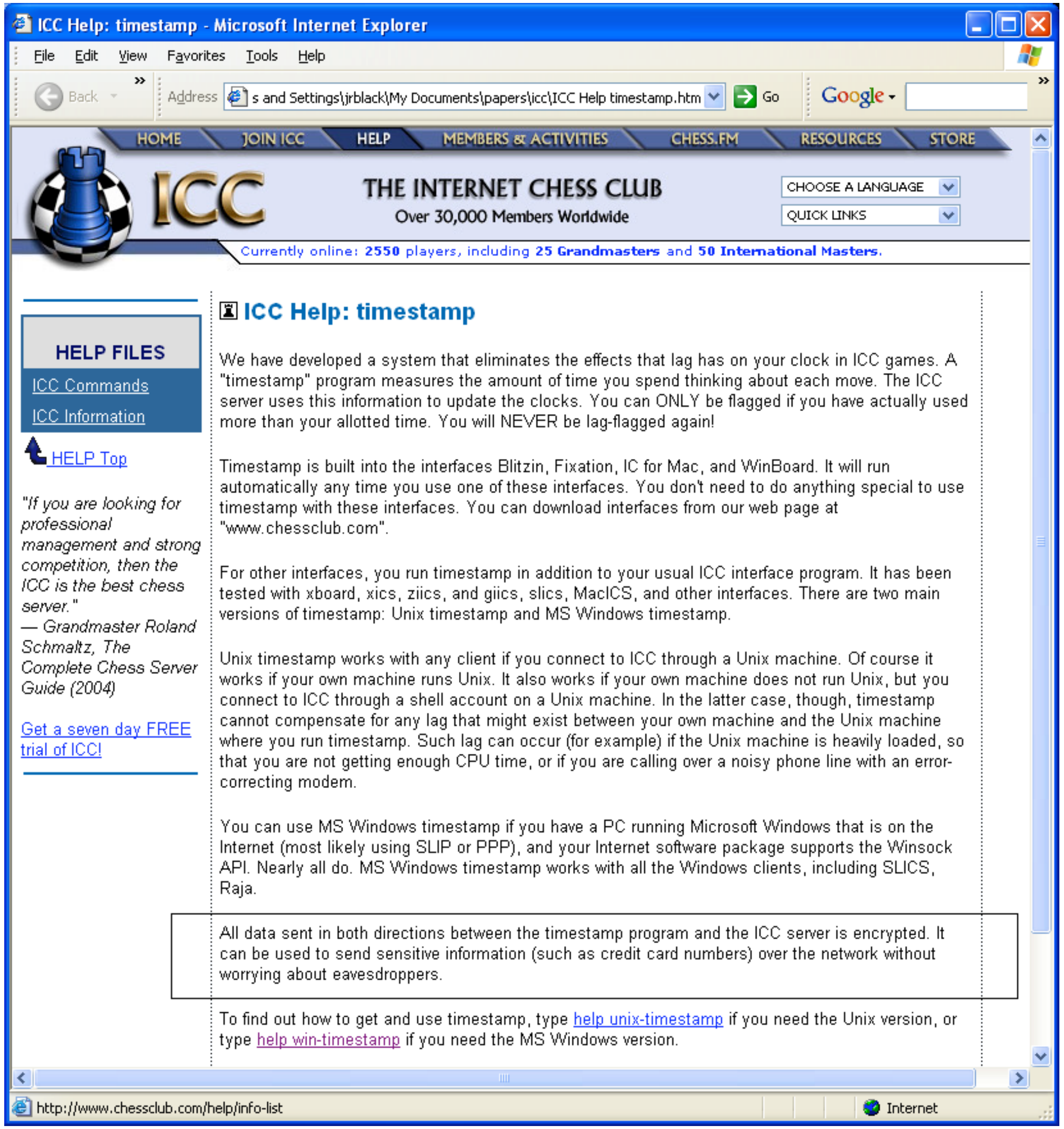
$$y_{n+1} = 17y_n \bmod 2413871$$

$$\text{pad} = x_n \oplus y_n$$ *(just low byte)*

- Given 10 pad bytes, we get the rest
- 1.1 secs on Martin's laptop

Key Exchange

- Seeds for symmetric keys exchanged ***in the clear!!!***
- We sniff the connection (pcap) and read all the traffic trivially
  - Get CC #s
  - Get usernames and passwords
- Active attacks would be even MORE damaging


ICC Help: timestamp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Address s and Settings\jrblack\My Documents\papers\icc\ICC Help timestamp.htm
Go
Google
HOME JOIN ICC HELP MEMBERS & ACTIVITIES CHESS.FM RESOURCES STORE
ICC
THE INTERNET CHESS CLUB
Over 30,000 Members Worldwide
CHOOSE A LANGUAGE
QUICK LINKS
Currently online: 2550 players, including 25 Grandmasters and 50 International Masters.
HELP FILES
ICC Commands
ICC Information
HELP Top
"If you are looking for professional management and strong competition, then the ICC is the best chess server."
— Grandmaster Roland Schmaltz, The Complete Chess Server Guide (2004)
Get a seven day FREE trial of ICC!
http://www.chessclub.com/help/info-list
Internet

ICC Help: timestamp

We have developed a system that eliminates the effects that lag has on your clock in ICC games. A "timestamp" program measures the amount of time you spend thinking about each move. The ICC server uses this information to update the clocks. You can ONLY be flagged if you have actually used more than your allotted time. You will NEVER be lag-flagged again!

Timestamp is built into the interfaces Blitzin, Fixation, IC for Mac, and WinBoard. It will run automatically any time you use one of these interfaces. You don't need to do anything special to use timestamp with these interfaces. You can download interfaces from our web page at "www.chessclub.com".

For other interfaces, you run timestamp in addition to your usual ICC interface program. It has been tested with xboard, xics, ziics, and giics, slics, MacICS, and other interfaces. There are two main versions of timestamp: Unix timestamp and MS Windows timestamp.

Unix timestamp works with any client if you connect to ICC through a Unix machine. Of course it works if your own machine runs Unix. It also works if your own machine does not run Unix, but you connect to ICC through a shell account on a Unix machine. In the latter case, though, timestamp cannot compensate for any lag that might exist between your own machine and the Unix machine where you run timestamp. Such lag can occur (for example) if the Unix machine is heavily loaded, so that you are not getting enough CPU time, or if you are calling over a noisy phone line with an error-correcting modem.

You can use MS Windows timestamp if you have a PC running Microsoft Windows that is on the Internet (most likely using SLIP or PPP), and your Internet software package supports the Winsock API. Nearly all do. MS Windows timestamp works with all the Windows clients, including SLICS, Raja.

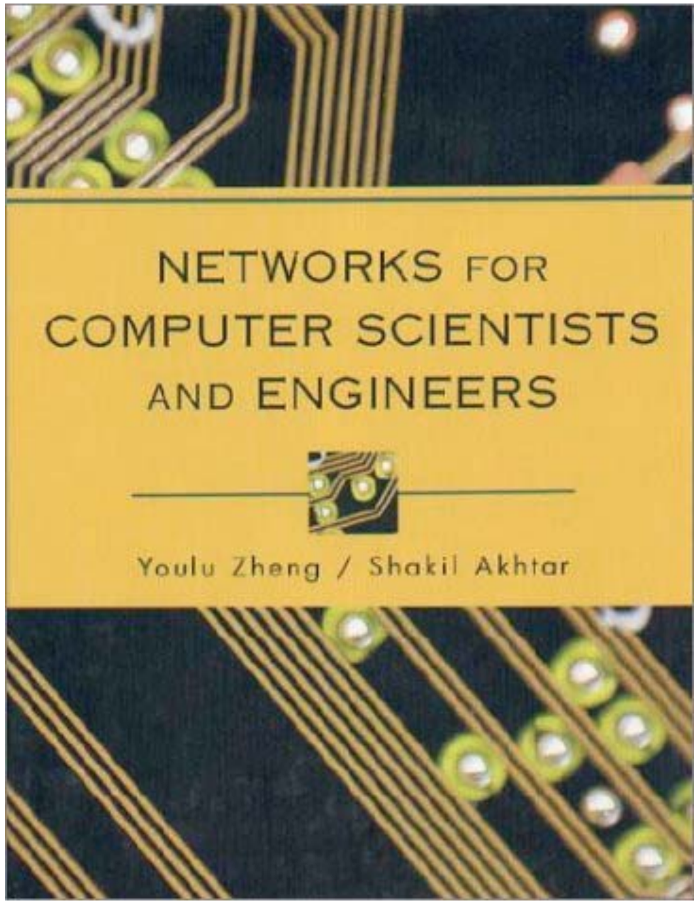
All data sent in both directions between the timestamp program and the ICC server is encrypted. It can be used to send sensitive information (such as credit card numbers) over the network without worrying about eavesdroppers.

To find out how to get and use timestamp, type help unix-timestamp if you need the Unix version, or type help win-timestamp if you need the MS Windows version.

MS Office

- Office 95
  - Just xor’ed password with plaintext over and over
- Later RC4 employed, but exportable versions forced to use 40-bit key
  - Easily broken by brute-force
- Office 2000, XP, 2003 use 128-bit RC4
  - But use the same IV (seed) each time

Closing Example: Pedagogy


NETWORKS FOR COMPUTER SCIENTISTS AND ENGINEERS

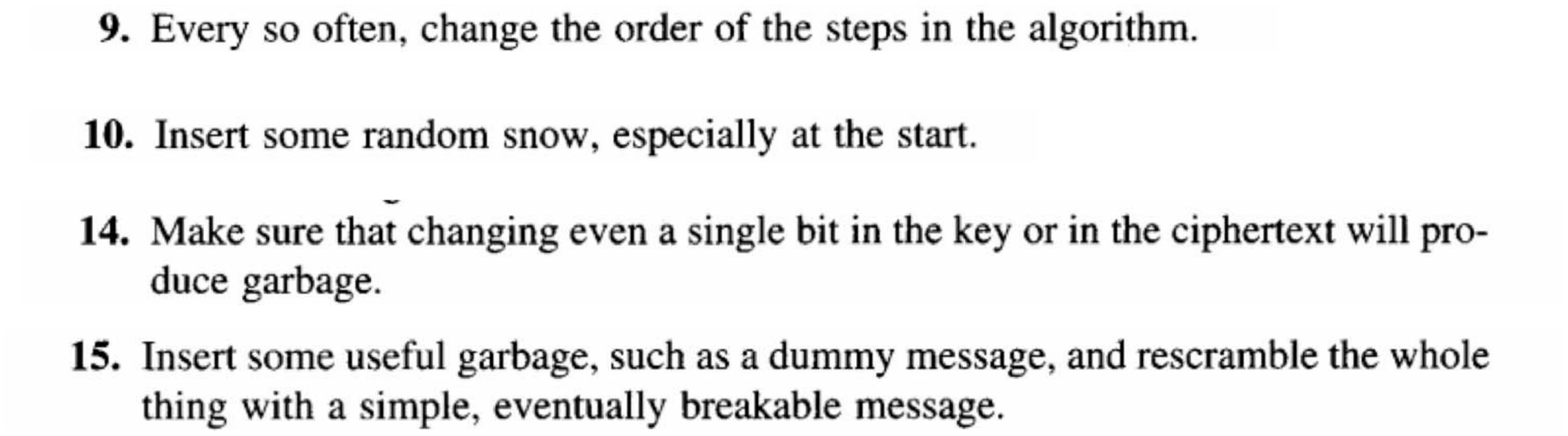
Youlu Zheng / Shakil Akhtar

10.1.8 Write Your Own Encryption Algorithm

People are often discouraged from writing a personal encryption algorithm because of a fear that a small bug in the code will render their decrypted messages meaningless. On the other hand, trusting the security of your transmissions to "experts" can also be a questionable practice.

If you follow the principles outlined here, writing your own encryption system should be easy. For practice, the laboratory manual (part of the Instructor's manual and CD accompanying the book) provides an encryption program written in X86 assembler code. The program incorporates several encryption steps to produce a multiple product cipher and chooses steps that are aimed at thwarting various attack methods. Here are the steps contained in the sample program and some suggestions for designing an encryption system:

Encryption Algorithm (Cont) *[ZA 2002]*



9. Every so often, change the order of the steps in the algorithm.

10. Insert some random snow, especially at the start.

14. Make sure that changing even a single bit in the key or in the ciphertext will produce garbage.

15. Insert some useful garbage, such as a dummy message, and rescramble the whole thing with a simple, eventually breakable message.

Moral

- Security Education is sorely inadequate
- Even if we did more, there would still be vulnerabilities, but it wouldn’t be nearly this bad