Bill of Rights on the Electronic Frontier
Peeping Tom installing your window blinds

I think we are getting Bush-whacked. Would it not be *better* if WE were getting *credit* for creating the intelligent development of a Bill of Rights for Cyberspace? Which might well include some very serious agreements on the need surveillance.

Who will vote for Clipper which must fast be becoming a lose-lose proposition in an election year. Could we not do better getting votes for an AFFIRMATIVE position?

Mike, thanks for the 25 pages of talking points and backgrounders.

Regards,
Jock

Jonathan P. Gill
Special Projects
Office of Media Affairs
The White House (202) 456-7150

---------- Forwarded message ----------
Date: Tue, 22 Feb 1994 20:46:59 -0800 (PST)
From: Wired Online address <online@wired.com>
To: WIRED Magazine Subscribers <subscribers@wired.com>
Subject: Electronic Privacy -- A Call to Action

(Sorry for those of you who have seen this before. For those that haven't, this is a one-time mailing - further mass mailings will only be done over the HotWired mailing list. Thanks!)

=-
=-=-=-=-=-=-=-=-=PLEASE REDISTRIBUTE THIS MESSAGE WIDELY!!=-=-=-=-=-=-=-=-=-=-=-=
-=-=-=-For copyright information, please see the end of this file.-=-=-=-=-=-
=-=

To: WIRED Online Information Services <hotwired@wired.com>
Subject: Electronic Privacy -- A Call to Action

This is a pivotal moment in history.

The national security state, with the backing of the Clinton-Gore administration, is attempting a stealth strike on our rights. If they succeed, we could shortly find ourselves under a government with the automated ability to log the time, origin, and recipient of every call and e-mail message, to monitor our most private communications, to track our physical whereabouts continuously, and to keep better account of our financial transactions than we do -- all without a warrant.

Fact: On Friday, February 4, 1994, the Clinton administration announced support for the Clipper Chip and SKIPJACK encryption scheme as national

standards.

Fact: Federal security agencies have been meeting with telecommunications companies to design back doors into the entire National Information Infrastructure (NII), including every telephone and data network, even including fax machines. In other words, any system connected to the NII would be required to include a "back door" in order to facilitate monitoring by government agencies.

We at WIRED Online believe that the adoption of these administration initiatives could result in a profound infringement of individual freedom and privacy, ours as well as yours. We urge you to read the rest of this letter, to examine the available materials, to consider these important issues for yourself, and to act to preserve the Bill of Rights in cyberspace.

The proposed encryption scheme, which uses the SKIPJACK encryption algorithm and the Clipper Chip, relies on a "key escrow" system with a built-in "back door" so that security agents can decrypt and monitor even supposedly "secure" communications. While the administration claims that there will be "safeguards," the technology was developed by the virtually insular National Security Agency, and its algorithms remain classified.

The scope of Clipper is significantly broader than any previous surveillance strategy. The Clipper Chip will be installed directly into telecommunications devices such as telephones, computers, and digital set-top boxes for interactive TV. Since the system can be used to encrypt any communications that pass across telecommunications lines (including text, sound and images), ANY AND ALL communication that passes through your system has the possibility of being intercepted.

In addition, the administration's Information Infrastructure Task Force Working Group on Privacy is attempting to "front load" the NII with trapdoor technologies that would allow security agencies easy access to digitial conversations, including capturing electronic communications midstream. No communication system would be exempt from this effort, from the national telephone network to your local office computer network.

Of course, the administration claims that these trapdoors will be used only to catch criminals and that your privacy will be protected. But, as John Perry Barlow has put it, "trusting the government with your privacy is like trusting a Peeping Tom to install your window blinds."

These government inititatives, taken together, constitute one of the most grievous threats to our constitutional liberties in modern times. The security agencies and the administration are involved in a stealth strike at our freedoms that could effectively abrogate the Bill of Rights in cyberspace, where we and our descendants will be spending increasingly larger parts of lives.

The Clipper initiative and the plans to require "back doors" throughout the NII demands immediate critical assessment. WIRED encourages you to seriously consider how these proposals might affect you. To help inform your decision, WIRED Online has set up a Clipper information archive through our Infobot mail server, Internet Gopher, World Wide Web, and other online sites.

The WIRED Online Clipper Archive features crucial essays written for WIRED by John Perry Barlow and Brock N. Meeks. If you do nothing else, read these stories. You can have them sent to you immediately by electronic mail by

copying the following three lines into the body of an electronic mail message addressed to infobot@wired.com:

```
send clipper/privacy.meeks
send clipper/privacy.barlow
end
```

The WIRED Online Clipper Archive also includes re-posted comments from Jerry Berman (of the Electronic Frontier Foundation (EFF)) and Dorothy Denning (encryption expert and Clipper proponent), a copy of the EFF's _EFFector Online_ newsletter documenting the Clipper controversy, and an electronic anti-Clipper petition circulated by the Computer Professionals for Social Responsibility (CPSR). We have also set up links to other valuable sources of information on Clipper, including those maintained by the EFF and CPSR.

You can access our archive via the following WIRED Online services:

o WIRED Infobot e-mail server	send e-mail to infobot@wired.com, containing the words "send clipper/index" on a single line inside the message body
o WIRED Gopher	gopher to gopher.wired.com select "Clipper Archive"
o WIRED on World Wide Web	http://www.wired.com select "Clipper Archive"
o WIRED on America Online	keyword: WIRED
o WIRED on the WELL	type "go wired" from any "OK" prompt type "clipper" to access the menu

WIRED Online encourages you to take the time to familiarize yourself with these issues, beginning with the tools and access we've provided. Then take the next step -- ACT!!!

Support the Cantwell bill. Write cantwell@eff.org and put "I support HR 3627" in the Subject header. This bill is designed to give rise to a mass-market in cryptographic software, which is a necessary step to beating Clipper. Feel free to include in your letter to Rep. Maria Cantwell your reasons for supporting the growth of the encryption industry and reasons for opposing Clipper.

To call for Senate hearings on Clipper, write Sen. Patrick Leahy to leahy@eff.org and express your concern that the Clipper process has been closed to the public.

Express your sentiments to Rep. Lee Hamilton, D-Indiana, the House Committee on Foreign Affairs chair, by e-mailing hamilton@eff.org.

Sign the CPSR petition against Clipper.

Call or write your Congressional representatives and let them know how you feel about the Clipper and NII "backdoor" initiatives, BEFORE a decision is made for you that will have a profound effect on the future of your freedom and privacy.

Please do not reply to this message directly. To discuss these issues with WIRED readers and staff members, please use discussion areas on the WELL, America Online, and USENET (alt.wired). If you have questions or comments about Clipper that are not answered in the online archives or these discussion spaces, please address them to online@wired.com and be sure to include the word "clipper" in the subject line.

If you would like to receive future WIRED-related bulletins, you can subscribe to our new Hotwired mailing list. To do so, just send an e-mail message to infobot@wired.com containing the line

    subscribe hotwired

This low-volume moderated list is a great way to keep abreast of important issues on the Digital Frontier and to find out about new services offered here at WIRED Online.

Thanks for your attention.

-- The staff of WIRED Online

I thought you might be interested in the following message I got
over the net this weekend.
Paul

-------- Original Message --------

Return-path: <SFRaves-owner@techno.Stanford.EDU>
Received: from mhis.bix.com by bix.com (CoSy3.31.1.45) id
 <9402182341.memo.3449@BIX.com>; Fri, 18 Feb 1994 23:41:28 -0500 (EST)
Received: by mhis.BIX.com (smail2.5/gw1.1)
        id AA26270; 18 Feb 94 23:41:27 EST (Fri)
Received: by delphi.com with UUCP/PMDF (DECUS UUCP);
          Fri, 18 Feb 1994 22:31:16 EST
Received: from techno.Stanford.EDU by delphi.com (PMDF V4.2-11 #6311) id
 <01H91PK0XMU88WWMS6@delphi.com>; Fri, 18 Feb 1994 22:31:09 EST
Received: by techno.Stanford.EDU (4.1/1.34) id AA03029; Fri,
 18 Feb 94 19:24:33 PST
Received: by techno.Stanford.EDU (4.1/1.34) id AA02995; Fri,
 18 Feb 94 19:22:46 PST
Received: from mail.netcom.com (netcom7.netcom.com) by techno.Stanford.EDU
 (4.1/1.34) id AA02988; Fri, 18 Feb 94 19:22:39 PST
Received: from localhost by mail.netcom.com (8.6.4/SMI-4.1/Netcom) id TAA13145;
 Fri, 18 Feb 1994 19:23:27 -0800
From: "Niels P. Mayer" <mayer@netcom.com>
Date: Fri, 18 Feb 1994 19:23:23 -0800
To: San FranDisco Ravers List <sfraves@techno.Stanford.EDU>
Message-id: <199402190323.TAA13145@mail.netcom.com>
Subject: Fwd on Clipper Chip -- an interesting take on all this
Sender: sfraves-owner@techno.Stanford.EDU
Content-transfer-encoding: 7BIT
X-Envelope-to: bix.com!prichard

Jacking in from a Non-Government Approved Encryption Port:

Washington, DC -- The Clinton Administration today gang raped your
privacy.

The White House Friday announced its endorsement of a sweeping new
security and privacy initiative. Privacy, as we know it, will
never be the same. All the rules have changed. Forever. The catch
is that the government gets to write all the rules; you get no
vote. None. Worse, you can't even read the fucking rule book
because it's classified.

The initiative involves the creation of "new products to accelerate
the development and use of advanced and secure telecommunications
networks and wireless telecommunications links," the White House
said. In English: Law enforcement and intelligence agencies now
have an easy way to fuck with any and all forms of spoken or
electronically transmitted communications.

The policy is voluntary, of course. You don't have to sign on to it. You don't have to use government approved encryption devices. But if you plan to do any business with the government, you'll have to use them. And if the government gets its way, well, you'll end using them whether you want to or not. You'll have no choice (are you sensing a trend here?). All telephones, computers, fax machines, modems, etc. will come "wiretap ready." It will be the de facto standard.

If you don't use the government standard, you'll be branding yourself a CryptoRebel. Big fucking deal? Maybe, maybe not. But think for a second. Perhaps some agency will be able to check your "crypto-approval rating." Perhaps those favorable bank loans, mortgage rates or low insurance premiums will only go to those with high crypto-approval ratings.

But the White House is adamant about making sure you understand this whole damn thing is voluntary. And don't let anything sway you from believing that, not even the White House backgrounder materials that say no U.S. citizen "as a matter of right, is entitled to an unbreakable commercial encryption product."

Just use the "balanced" approach of the government system, where in this case the "breakability" of the encryption belongs only with them. Everything will work out fine. Just listen: "Encryption is a law and order issue since it can be use by criminals to thwart wiretaps and avoid detection and prosecution," said Vice President Gore. "Our policy is designed to provide better encryption to individuals and businesses while ensuring that the needs of law enforcement and national security are met."

The Administration won't tell you exactly why they expect you simply hand over all your privacy safeguards to them. "Listen, if you knew what we knew about criminal activity, this issue wouldn't even be debated," said Mike Nelson with the Office of Science and Technology Policy and co-chair of the Working Group on Data Security, a newly created interagency task force.

Chicken or the Egg?

The new policy was hatched in the super-secret recesses of the National Security Agency (NSA). And while Clinton was still trying to find the instruction manual for his White House telephone system, the NSA, FBI and other assorted agencies shoved their ideas onto the National Security Council table. Before the Administration could blink, it found itself in the unenviable position of having backed a severely flawed security policy that has compromised the privacy of every U.S. citizen and drawn the ire of every civil liberties in the country.

But the White House quickly put the breaks on, calling for a full scale, government wide "review" of its security and privacy policies. It gave privacy advocacy groups some breathing room. Surely the Administration, once it had a chance to actually study this damn thing, would see it through it.

But the White House punted. The review was a smoke screen. Instead, it provided momentum inside the Administration. It was

from this review, ordered last April, that this new initiative springs.

And when all was said and done, the White House screwed the pooch.

Clipper Sails On

The one trick pony here is the Clipper Chip, a device that can be installed in virtually any communications device. The chip scrambles all conversations. No one can crack the code, expect the government, of course. The Feds hold all the keys. Rather, they hold the only keys that count.

Each Clipper chip is made with 3 unique keys. All three are needed to descramble the encrypted messages streaming through them. But only the government's keys matter. The key you get with your Clipper Chip is essentially the chip's social security number. You'll never actually see this key, have any idea what its number is or get your hands on it. If you try to sneak a peak at it, the damn thing self destructs. Honest.

The other two keys will be held in electronic vaults; fraternal twins, separated by mandate. Each of these keys will be held by government agencies, called "escrow agents." One will be held by the National Institute for Standards and Technology, the other by the Automated Systems Division of the Dept. of Treasury.

When a law enforcement agency, which could be your local sheriff's department, wants to wiretap a conversation that's been encrypted by Clipper they apply to each of the escrow agents. The agents send their respective key, electronically, to a "black box" operated by the law enforcement agency. As encrypted conversations stream into the box, they come out the back side in nice, neat sounding vowels and consonants, or in the case of electronic mail, in plain ol' ASCII.

Yes, all law enforcement agencies need a court approved wiretap before they can pull this whole scheme off. This, the Administration says, is where you're privacy is protected. "We're not going to use Clipper to listen in on the American public," said Raymond Kammer of NIST deputy director. It will only be used to catch criminals. Honest.

We Don't Need No Stinkin' Warrant

Maybe now would be a good time to mention the National Security Agency. You know these guys. Super-secret, spook agency. Their mission? To monitor and intercept foreign communications. Did you catch that word FOREIGN? I hope so, it's crucial.

The NSA is only allowed to intercept foreign communications spying on U.S. citizens is a crime. They can't even pry into a U.S. citizen's business a court ordered wiretap. A judge would never allow it. Yet it was the NSA that cooked up this whole Clipper Chip scheme. Why you ask? Good question. But the Administration refuses to discuss the issue.

Here's another they can't answer. Suppose the NSA intercepts a

message from Iraq and finds it's Clipper encrypted (that damn little black box is specially made to sniff out the Clipper's algorithm and descramble it's social security number). What does the NSA do with this encrypted Iraqi message? How does it decrypt the message? There's a classic Catch-22 running here.

Agencies need the Clipper keys from the escrow agents to read the message or listen in on the conversation. But to get the keys you need proof that you have a warrant. The NSA is *never* issued a warrant. You see, the NSA doesn't need a warrant to spy on FOREIGN communications.

So, this begs the $64,000 question is: How does the NSA get the escrow agents to give them the keys to decrypt the message if they can't show a warrant?

Answer: They don't have to show a warrant; they don't have to cause; they don't have to show spit.

What's wrong with this picture?

"We have appropriate procedures and safeguards built into the system for the NSA," Nelson said. "I can't tell you what those are, of course, that would divulging too much about the NSA's operation."

Fox Guarding the Chickens
==========

There will be absolutely no abuse of the system. This is what the Administration would like you to believe. They also would like you to believe that President's don't approve Watergate break-ins, that arms are never traded for hostages, that the FBI never secretly records civil rights leaders in the heat of infidelity and that FBI directors have never shown a proclivity for red sequined dresses and shiny high-heeled cruel shoes.

Representatives from four government organizations stood before the press and outlined all the careful thinking and rigorous safeguards that have gone into this system. There are at least 9 different steps that must be followed to get these Clipper keys transferred from the escrow agents to the agency authorized to do the wiretap.

Fair enough, isn't it? Well, it would be except for the fact that the Justice Department intentionally wrote a giant fucking loop-hole into the law.

Buried in the Justice Department briefing papers, outlining the authorization procedures for release of the escrow keys, is this gem: "These procedures do not create, and are not intended to create, any substantive rights for individuals intercepted through electronic surveillance, and noncompliance with these procedures shall not provide the basis for any motion to suppress or other objection to the introduction of electronic surveillance evidence lawfully acquired."

So, if somebody screws up, like for instance, asks for the keys to be sent before they actually have a wiretap in hand, or has no wiretap authority at all! there is no recourse provision.

Criminals As Dumb Shits

But what about that wily criminal element? Once they get wind of this, won't they seek out another type of encryption? The FBI doesn't think so. In fact, the FBI thinks criminals are such dumb shits that they'll forget all about the fact that Clipper even exists.

"I predict that few criminals will remember years from now what they've read in the Wall Street Journal" about how these devices were installed in telephones, said FBI's James Kallestrom. (Of course, if criminals are so stupid, why are they perusing the Wall St. Journal... maybe he really meant the New York Times...)

So let's get this right. The FBI is sure that criminals will "just forget" that Clipper is installed in their phones and use them anyway. These are the criminals that also would be forgetting that their multi-million dollar drug deals, not to mention their own sweet ass, could be in jeopardy every time they make a call. Yes, the government really thinks so.

It's more likely that some bright, enterprising criminal mind will create a worldwide black market that deals in "non-Clipper Installed" encryption devices. Damn, talk about an industry with some growth potential.

Getting To the Data Stream

The whole damn program goes into the crapper, however, if the government can't get access to source, to the digital data stream, as it comes out of the telephone switch. In order to do this you have to tap the digital conversation. That's right, you guessed it: Digital Wiretap Access.

The FBI failed on its own last year to generate any support in Congress for this digital wiretap proposal. Hell, the FBI couldn't even get a single member of Congress to introduce the thing. So the FBI broke the chain of command: They got the President and Vice President to sign off on the idea.

The Administration will soon announce its decision on how it will give the FBI the right to easily wiretap even your unencrypted conversations. "Within a few months at most we should have something decided," said Barry Smith of the FBI's Congressional Affairs office. The FBI's Kallestrom said it was "all but a done deal."

This isn't a question of whether or not the Administration will line up behind the FBI on this. It already has. It's only a matter of paperwork, and the nagging little issue of how to pay for making the telephone companies comply with the new rules. But these are small details, compared to the heat the Administration already knows it'll take when they finally unwrap this puppy.

Private No More

OSTP's Nelson quipped that these security and privacy issues are

the Cyberspace version of the Administration's muddied Bosnia policy.

Like Bosnia, the White House expects the American public to "trust us" on this issue. After all, the Administration says, they know a hell of a lot more than we do about what kind criminal activity is really going down.

Trusting these law enforcement and intelligence agencies is one thing; tempting them by putting all-powerful tools right into their hip pockets is something that should generate a hue and cry loud enough for all of Washington to hear.

So, if you're really pissed off, just pick up the phone can call your neighbor. Somebody in Washington is bound to hear it.

Meeks out....

=-=
= Niels Mayer -- mayer@eit.com =
= MultiMedia Engineering Collaborative Environment =
= Enterprise Integration Technologies, 459 Hamilton Ave, Palo Alto CA 94301 =
=-=

Embargoed until 3:00 p.m. EST
Feb. 4, 1994

QUESTIONS AND ANSWERS ABOUT THE CLINTON ADMINISTRATION'S ENCRYPTION POLICY

Q. What were the findings of the encryption technology review?

A. The review confirmed that sound encryption technology is needed to help ensure that digital information in both computer and telecommunications systems is protected against unauthorized disclosure or tampering. It also verified the importance of preserving the ability of law enforcement to understand encrypted communications when conducting authorized wiretaps. Key escrow technology meets these objectives.

Specific decisions were made to enable federal agencies and the private sector to use the key escrow technology on a voluntary basis and to allow the export of key escrow encryption products.

In addition, the Department of State will streamline export licensing procedures for products that can be exported under current regulations in order to help U.S. companies to sell their products abroad.

To meet the critical need for ways to verify the author and sender of an electronic message -- something that is crucial to business applications for the National Information Infrastructure -- the federal government is committed to ensuring the availability of a royalty-free, public-domain Digital Signature Standard.

Finally, an interagency working group has been established to continue to address these issues and to maintain a dialogue with industry and public interest groups.

Q: Who has been consulted during this review? The Congress? Industry? What mechanism is there for continuing consultation?

A: Following the President's directive announced on April 16, 1993, extensive discussions have been held with Congress, industry, and privacy rights groups on encryption issues. Formal public comment was solicited on the Escrowed Encryption Standard and on a wide variety of issues related to the review through the Computer System Security and Privacy Advisory Board.

The White House Office of Science and Technology Policy and the National Security Council will chair the interagency working group. The group will seek input from the private sector both informally and through several existing advisory committees. It also will work closely with the Information Policy Committee of the Information Infrastructure Task Force, which is responsible for coordinating Administration telecommunications and information policy.

A: They would have to obtain legal authorization, normally a court order, to do the wiretap in the first place. They would then present documentation, including a certification of this authorization, to the two entities responsible for safeguarding the keys. (The key is split into component parts, which are stored separately in order to ensure the security of the key escrow system.) They then obtain the components for the keys for the device being used by the drug smugglers. The components are then combined and the message can be read.

Q: Who will hold the escrowed keys?

A: The Attorney General has selected two U.S. agencies to hold the escrowed key components: the Treasury Department's Automated Systems Division and the Commerce Department's National Institute of Standards and Technology.

Q: How strong is the security in the device? How can I be sure how strong the security is?

A: This system is more secure than many other voice encryption system readily available today. While the algorithm upon which the Escrowed Encryption Standard is based will remain classified to protect the security of the system, an independent panel of cryptography experts found that the algorithm provides significant protection. In fact, the panel concluded that it will be 36 years until the cost of breaking the algorithm will be equal to the cost of breaking the current Data Encryption Standard now being used.

Q. Is there a "trap door" that would allow unauthorized access to the keys?

A. No. There is no trapdoor.

Q: Whose decision was it to propose this product?

A: The National Security Council, the Justice Department, the Commerce Department, and other key agencies were involved in this decision. The approach has been endorsed by the President, the Vice President, and appropriate Cabinet officials.

Clipper/Key Escrow Encryption

2/16/94

The following are focused on Clipper/EES. They exclude other issues such as encryption export controls and telephony.

EES = Escrowed Encryption Standard

1. Privacy

The standard is subject to misuse and compromise by providing government agencies with copies of the keys that protect electronic communications. (CPSR) The Government cannot be trusted.

- Clipper/EES is being implemented with strict procedural and technical safeguards.
- Unlawful electronic surveillance is a violation of Federal law.
- Clipper/EES expands the paper trail required during wiretaps, making it more difficult to gain information from electronic communications.

This is only the first step toward outlawing encryption.

- Very early in the Administration's review of cryptographic policy, we stated that we would not seek legislation (or other means) to control domestic use of any type of cryptography.
- Use of Clipper is voluntary -- both within and outside of the government.

The system isn't really voluntary.

- We have repeatedly stated that the use of Clipper is voluntary.
- The EES standard is voluntary for users inside the federal government. It may be used voluntarily by those outside of the government seeking excellent data security.

2. Technical

There is a hidden trapdoor.

- There is no trapdoor.
- We invited a team of outside cryptographic experts in to examine the algorithm. They confirmed that there is no trapdoor.

The integrity of the technology is doubtful. Clipper was developed in secret and its specifications are classified. (CPSR)

- The algorithm must remain classified or else interoperable products utilizing this highly secure algorithm could be built without the key escrow mechanism.
- The algorithm was initially developed to protect classified information.
- Disclosure of the algorithm may reveal design techniques which would be beneficial to those with interests adverse to the U.S.

3. NSA

The underlying technology was developed by NSA, an intelligence agency. Congressional investigations in the 1970s disclosed widespread NSA abuses, including the illegal interception of millions of cables sent by Americans. (CPSR) They can't be trusted.

- While it is certainly true that NSA is responsible to collect foreign intelligence, the fact is NSA also has a mission to develop technology to protect government systems that process classified information.

- NSA's expertise in making secure encryption is second to none. It was because of their encryption expertise that they were requested to develop, in conjunction with NIST and the Justice Department, a technology to provide users truly excellent encryption while preserving an ability for government agencies to access the encryption when lawfully authorized.

- Since the 1970s, a variety of statutes and directives have been written to assure NSA's intelligence activities, particularly those that may affect the privacy of Americans, are strictly controlled. NSA's compliance with the requirements of these laws and regulations is actively reviewed within the Executive Branch by, for example, the Attorney general, and within the Legislative Branch by the House and Senate Intelligence Committees. There has been nothing in these reviews to suggest that NSA is abusing its authorities.

NSA overstepped its legal authority in developing the standard. The Computer Security Act of 1987 limits the intelligence agency's power to set standards for the communications network. (CPSR)

- NSA's role regarding the key escrow technology satisfies both the letter and the spirit of the Computer Security Act of 1987. In recognition of NSA's expertise in developing secure communications systems, NSA

was asked to work with NIST and other agencies to develop s system that would provide excellent security without jeopardizing the ability of law enforcement to access communications when legally authorized.

- Under the Computer Security Act, NSA is to provide technical assistance and support to NIST.

- NIST, not NSA, promulgates standards for systems other than those designed for defense or intelligence missions.

Clipper is just another step toward allowing NSA to read U.S. citizen data/listen to phone calls.

- Key escrow encryption makes it more difficult for anyone, including government agencies to eavesdrop on communications without proper legal authorization.

- This initiative does not, in any way, expand the authority of any government agency to listen to phone calls.

- Clipper provides excellent security to the user so no one else can decrypt their communications without obtaining the escrowed keys. The systems for escrowing keys has been designed to prevent any access to the keys without proper legal authorizations.

American business will only want to use one product for their international communications. The keys won't be safeguarded against NSA and their foreign counterparts.

- American businesses using key escrow encryption for their communications will be protected against any illegal, foreign intelligence, or industrial eavesdropping.

- Key escrow encryption is 16 million times stronger than the current federal unclassified encryption standard. Its strength was confirmed by a group of independent expert cryptographers.

4. Law Enforcement

There is no need for Clipper. The Government cannot cite a single instance in which encryption hindered law enforcement's wiretapping.

- The problem is not what may have happened to date, but rather, what is clearly on the horizon. For the first time, the public has access to

commercially available telephone encryption devices that are high quality, user-friendly and relatively inexpensive. That will radically increase the number of persons using encryption for telephone and faxes -- and that means wiretaps will encounter encryption more frequently.

- We see a real problem in the very near future and are trying to deal with it before a tragic situation occurs. Were we to wait until lives had been lost, we would be criticized for failing to foresee the problem and prevent it.

Criminals are not stupid. They won't use Clipper.

- Most criminals will not be able to arrange for creation of their own cryptographic equipment; like the rest of us, they will use what is commercially available and attractive.

- To the extent that devices using key-escrow encryption proliferate and gain commercial acceptance, those devices will be among the principal available choices.

- As key-escrow encryption gains commercial acceptance among legitimate businesses, criminal enterprises will need to use key-escrow encryption to be able to communicate with, for example, financial institutions.

- As a practical matter, criminal enterprises still transact a great deal of business over unencrypted telephone lines and are unlikely to worry very much about whether the encryption device they use is one that can be decrypted by government.

The cost of the system is too high for unknown/dubious law enforcement benefits.

- The system does not add significantly, if at all, to the cost of encryption devices sold to the consumer.

- We do not anticipate the cost of maintaining the system to be high. Certainly, it will not be out of keeping with the expected savings in lives and property, as well as the fines, forfeitures and penalties expected to be recovered as a result of the investigative efforts involving wiretaps and decrypted key-escrow communications.

5. National Security

NSA intercept targets (other governments, terrorists, etc.) will not use Clipper.

- This misconstrues the purpose of the initiative.
- It would be unwise to make such strong cryptography available (e.g, to terrorists) without the key escrow feature. Terrorists and others would almost certainly take advantage of it, with significant harm to our interests.
- Key escrow encryption makes the benefits available without damaging our interests.

The national security arguments have not been made public.

- Revealing these to the American public would also reveal them to foreign parties.
- Revealing how we protect our interests can nullify their effectiveness and place the U.S. at risk.

Clipper is a front for the government suppressing advanced cryptography.

- There has been no attempt and there is no intention by the government to suppress in any way the use of advanced cryptography in the U.S. or by U.S. persons/businesses abroad.
- The purpose of the key escrow initiative has been fully presented to the public by the Administration.

6. Key Escrow Agents/Procedures

The escrow agencies are part of the government. They are subject to political pressure for unauthorized release of the keys.

- The fundamental protections for communications of all American citizens, whether encrypted or not, are the Constitution and the statutes governing wire or electronic surveillance. Unlawful electronic surveillance is a violation of Federal law. Because anyone requesting key components will leave an enormous paper trail, anyone seeking unauthorized release will do so only at great risk of being caught.

- The key release procedures call for careful monitoring for compliance at all stages of the process. They also call for advising Congress, in conjunction with normal wiretap reports, of all wiretaps for which key components have been made available.

The Justice Department key release procedures state that if they are not followed there is no legal recourse. That cannot reassure users.

- The fundamental issue is that of the privacy of communications, whether encrypted or not, and a person's communications are entitled to no greater expectation of privacy merely because they are encrypted. A person whose communications have been subjected to an unauthorized wiretap does have legal recourse.

- The protections afforded communications under the Constitution and the statutes pertaining to wiretaps apply with equal force whether key-escrow encryption (or any other kind of encryption) is used.

There is nothing to prevent unauthorized use of the black box or decryption keys after expiration of the warrant.

- Continuing to conduct a wiretap after expiration of authorization to do so is a serious offense. Assuming, for the sake of argument, that a person were not to terminate the decrypt device's capability to decrypt communications using a particular chip, the criminal penalties would apply to wiretaps after the end of the authorized period -- regardless of whether the calls were encrypted.

- We will require law enforcement agencies to terminate the decrypt capability and to so advise the key escrow agents. The Department of Justice intends to monitor compliance with the key release procedures at all stages of the process.

- Once we are past the prototype versions of decrypt devices, the decrypt capability will be able to be automatically terminated within the device.

7. Economic

Setting Clipper as a standard will harm American computer businesses. No foreigners will buy the technology, greatly hampering our ability to compete.

- Clipper is not being forced upon American business.
- In consonance with applicable export controls, American business can offer a wide variety of encryption products on the international market.

There is no international support for Clipper.

- Clipper was designed to meet the needs of federal agencies to protect their sensitive information. It also provides a high degree of security

for other users who voluntarily choose to avail themselves of the technology.

- Clipper's primary use was not intended for international users.

Clipper, in hardware, is too expensive.

- We agree that Clipper in hardware is not ideal.
- We are seeking a way to do key escrow encryption in software.
- We have announced a Cooperative Research and Development Agreement opportunity with industry to seek software solutions.

Clipper's hardware is inflexible.

- We have announced a second Cooperative Research and Development Agreement opportunity with industry to seek other technical approaches.
- We welcome and solicit industry participation in this effort.

Clipper is doomed to fail. It's just a waste of money.

- Clipper will provide excellent protection for federal agencies.
- It's 16 million times stronger than the current voluntary federal standard (DES).

Clipper will be import controlled and not allowed into some countries. This won't meet the needs of American business to protect their information.

- We believe that Clipper will be importable to many destinations around the world for use by American firms protecting their data.
- We realize there will be a few exceptions.

8. Public Comments on Clipper/EES ignored

The Department of Commerce moved ahead with Clipper ignoring the vast majority of public comments it solicited.

- The comments received by the public were carefully reviewed.
- Many comments stated that Clipper use would be mandatory; it is voluntary as we have restated.
- We weighed the balance of interests in moving ahead.

THE WHITE HOUSE
OFFICE OF THE VICE PRESIDENT

EMBARGOED UNTIL 3:00 PM EST
February 4, 1994

CONTACT: 202/456-7035

STATEMENT OF THE VICE PRESIDENT

Today's announcements on encryption represent important steps in the implementation of the Administration's policy on this critical issue. Our policy is designed to provide better encryption to individuals and businesses while ensuring that the needs of law enforcement and national security are met.

Encryption is a law and order issue since it can be used by criminals to thwart wiretaps and avoid detection and prosecution. It also has huge strategic value. Encryption technology and cryptoanalysis turned the tide in the Pacific and elsewhere during World War II.

##

THE WHITE HOUSE
OFFICE OF THE PRESS SECRETARY

EMBARGOED UNTIL 3 PM (EST)
FRIDAY, February 4, 1994

CONTACT: 202-456-7035

STATEMENT OF THE PRESS SECRETARY

Last April, the Administration announced a comprehensive interagency review of encryption technology, to be overseen by the National Security Council. Today, the Administration is taking a number of steps to implement the recommendations resulting from that review.

Advanced encryption technology offers individuals and businesses an inexpensive and easy way to encode data and telephone conversations. Unfortunately, the same encryption technology that can help Americans protect business secrets and personal privacy can also be used by terrorists, drug dealers, and other criminals.

In the past, Federal policies on encryption have reflected primarily the needs of law enforcement and national security. The Clinton Administration has sought to balance these needs with the needs of businesses and individuals for security and privacy. That is why, today the National Institute of Standards and Technology (NIST) is committing to ensure a royalty-free, public-domain Digital Signature Standard. Over many years, NIST has been developing digital signature technology that would provide a way to verify the author and sender of an electronic message. Such technology will be critical for a wide range of business applications for the National Information Infrastructure. A digital signature standard will enable individuals to transact business electronically rather than having to exchange signed paper contracts. The Administration has determined that such technology should not be subject to private royalty payments, and it will be taking steps to ensure that royalties are not required for use of a digital signature. Had digital signatures been in widespread use, the recent security problems with the Internet would have been avoided.

Last April, the Administration released the Key Escrow chip (also known as the "Clipper Chip") that would provide Americans with secure telecommunications without compromising the ability of law enforcement agencies to carry out legally authorized wiretaps. Today, the Department of Commerce and the Department of Justice are taking steps to enable the use of such technology both in the U.S. and overseas. At the same time, the Administration is announcing its intent to work with industry to develop other key escrow products that might better meet the needs of individuals and industry, particularly the American computer and telecommunications industry. Specific steps being announced today include:

- Approval by the Commerce Secretary of the Escrowed Encryption Standard (EES) as a voluntary Federal Information Processing Standard, which will enable government agencies to purchase the Key Escrow chip for use with telephones and modems. The department's National Institute of Standards and Technology (NIST) will publish the standard.

- Publication by the Department of Justice of procedures for the release of escrowed keys and the announcement of NIST and the Automated Services Division of the Treasury Department as the escrow agents that will store the keys needed for decryption of communications using the Key Escrow chip. Nothing in these procedures will diminish the existing legal and procedural requirements that protect Americans from unauthorized wiretaps.

- New procedures to allow export of products containing the Key Escrow chip to most countries.

In addition, the Department of State will streamline export licensing procedures for encryption products that can be exported under current export regulations in order to help American companies sell their products overseas. In the past, it could take weeks for a company to obtain an export license for encryption products, and each shipment might require a separate license. The new procedures announced today will substantially reduce administrative delays and paperwork for encryption exports.

To implement the Administration's encryption policy, an interagency Working Group on Encryption and Telecommunications has been established. It will be chaired by the White House Office of Science and Technology Policy and the National Security Council and will include representatives of the Departments of Commerce, Justice, State, and Treasury as well at the FBI, the National Security Agency, the Office of Management and Budget, and the National Economic Council. This group will work with industry and public-interest groups to develop new encryption technologies and to review and refine Administration policies regarding encryption, as needed.

The Administration is expanding its efforts to work with industry to improve on the Key Escrow chip, to develop key-escrow software, and to examine alternatives to the Key Escrow chip. NIST will lead these efforts and will request additional staff and resources for this purpose.

We understand that many in industry would like to see all encryption products exportable. However, if encryption technology is made freely available worldwide, it would no doubt be used extensively by terrorists, drug dealers, and other criminals to harm Americans both in the U.S. and abroad. For this reason, the Administration will continue to restrict export of the most sophisticated encryption devices, both to preserve our own foreign intelligence gathering capability and because of the concerns of our allies who fear that strong encryption technology would inhibit their law enforcement capabilities.

At the same time, the Administration understands the benefits that encryption and related technologies can provide to users of computers and telecommunications networks. Indeed, many of the applications of the evolving National Information Infrastructure will require some form of encryption. That is why the Administration plans to work more closely with the private sector to develop new forms of encryption that can protect privacy and corporate secrets without undermining the ability of law-enforcement agencies to conduct legally authorized wiretaps. That is also why the Administration is committed to make available free of charge a Digital Signature Standard.

The Administration believes that the steps being announced today will help provide Americans with the telecommunications security they need without compromising the capability of law enforcement agencies and national intelligence agencies. Today, any American can purchase and use any type of encryption product. The Administration does not intend to change that policy. Nor do we have any intention of restricting domestic encryption or mandating the use of a particular technology.

##

—— Forwarded Message

Subject: Barlow Wired article on Clipper - "Jackboots on the Infobahn" !

[note: this article and other Clipper material are archived at:
ftp://ftp.eff.org/pub/EFF/Policy/Clipper/
Similar material can be found at soda.berkeley.edu.]

WIRED 2.04
Electrosphere
****

Jackboots on the Infobahn
^^^^^^^^^^^^^^^^^^^^^^^^^

Clipper is a last ditch attempt by the United States, the last great power from the old Industrial Era, to establish imperial control over cyberspace.

By John Perry Barlow

[Note: The following article will appear in the April 1994 issue of WIRED. We, the editors of WIRED, are net-casting it now in its pre-published form as a public service. Because of the vital and urgent nature of its message, we believe readers on the Net should hear and take action now. You are free to pass this article on electronically; in fact we urge you to replicate it throughout the net with our blessings. If you do, please keep the copyright statements and this note intact. For a complete listing of Clipper-related resources available through WIRED Online, send email to <infobot@wired.com> with the following message: "send clipper.index". - The Editors of WIRED]

On January 11, I managed to schmooze myself aboard Air Force 2. It was flying out of LA, where its principal passenger had just outlined his vision of the information superhighway to a suited mob of television, show-biz, and cable types who fervently hoped to own it one day - if they could ever figure out what the hell it was.

From the standpoint of the Electronic Frontier Foundation the speech had been wildly encouraging. The administration's program, as announced by Vice President Al Gore, incorporated many of the concepts of open competition, universal access, and deregulated common carriage that we'd been pushing for the previous year.

But he had said nothing about the future of privacy, except to cite among the bounties of the NII its ability to "help law enforcement agencies thwart criminals and terrorists who might use advanced telecommunications to commit crimes."

On the plane I asked Gore what this implied about administration policy on cryptography. He became as noncommittal as a cigar-store Indian. "We'll be making some announcements.... I can't tell you anything more." He hurried to the front of the plane, leaving me to troubled speculation.

Despite its fundamental role in assuring privacy, transaction security, and reliable identity within the NII, the Clinton administration has not demonstrated an enlightenment about cryptography up to par with the rest of its digital vision.

The Clipper Chip - which threatens to be either the goofiest waste of federal dollars since President Gerald Ford's great Swine Flu program or, if actually deployed, a surveillance technology of profound malignancy - seemed at first an ugly legacy of the Reagan-Bush modus operandi. "This is going to be our Bay of Pigs," one Clinton White House official told me at the time Clipper was introduced, referring to the disastrous plan to invade Cuba that Kennedy inherited from Eisenhower.

(Clipper, in case you're just tuning in, is an encryption chip that the National Security Agency and FBI hope will someday be in every phone and computer in America. It scrambles your communications, making them unintelligible to all but their intended recipients. All, that is, but the government, which would hold the "key" to your chip. The key would separated into two pieces, held in escrow, and joined with the appropriate "legal authority.")

Of course, trusting the government with your privacy is like having a Peeping Tom install your window blinds. And, since the folks I've met in this White House seem like extremely smart, conscious freedom-lovers - hell, a lot of them are Deadheads - I was sure that after they were fully moved in, they'd face down the National Security Agency and the FBI, let Clipper die a natural death, and lower the export embargo on reliable encryption products.

Furthermore, the National Institutes of Standards and Technology and the

National Security Council have been studying both Clipper and export embargoes since April. Given that the volumes of expert testimony they had collected overwhelmingly opposed both, I expected the final report would give the administration all the support it needed to do the right thing.

I was wrong. Instead, there would be no report. Apparently, they couldn't draft one that supported, on the evidence, what they had decided to do instead.

THE OTHER SHOE DROPS

On Friday, February 4, the other jackboot dropped. A series of announcements from the administration made it clear that cryptography would become their very own "Bosnia of telecommunications" (as one staffer put it). It wasn't just that the old Serbs in the National Security Agency and the FBI were still making the calls. The alarming new reality was that the invertebrates in the White House were only too happy to abide by them. Anything to avoid appearing soft on drugs or terrorism.

So, rather than ditching Clipper, they declared it a Federal Data Processing Standard, backing that up with an immediate government order for 50,000 Clipper devices. They appointed the National Institutes of Standards and Technology and the Department of Treasury as the "trusted" third parties that would hold the Clipper key pairs. (Treasury, by the way, is also home to such trustworthy agencies as the Secret Service and the Bureau of Alcohol, Tobacco, and Firearms.)

They reaffirmed the export embargo on robust encryption products, admitting for the first time that its purpose was to stifle competition to Clipper. And they outlined a very porous set of requirements under which the cops might get the keys to your chip. (They would not go into the procedure by which the National Security Agency could get them, though they assured us it was sufficient.)

They even signaled the impending return of the dread Digital Telephony, an FBI legislative initiative requiring fundamental reengineering of the information infrastructure; providing wiretapping ability to the FBI would then become the paramount design priority.

INVASION OF THE BODY SNATCHERS

Actually, by the time the announcements thudded down, I wasn't surprised by them. I had spent several days the previous week in and around the White House.

I felt like I was in another remake of The Invasion of the Body Snatchers. My friends in the administration had been transformed. They'd been subsumed by the vast mindfield on the other side of the security clearance membrane, where dwell the monstrous bureaucratic organisms that feed on fear. They'd been infected by the institutionally paranoid National Security Agency's Weltanschauung.

They used all the telltale phrases. Mike Nelson, the White House point man on the NII, told me, "If only I could tell you what I know, you'd feel the same way I do." I told him I'd been inoculated against that argument during Vietnam. (And it does seem to me that if you're going to initiate a process that might end freedom in America, you probably need an argument that isn't classified.)

Besides, how does he know what he knows? Where does he get his information? Why, the National Security Agency, of course. Which, given its strong interest in the outcome, seems hardly an unimpeachable source.

However they reached it, Clinton and Gore have an astonishingly simple bottom line, to which even the future of American liberty and prosperity is secondary: They believe that it is their responsibility to eliminate, by whatever means, the possibility that some terrorist might get a nuke and use it on, say, the World Trade Center. They have been convinced that such plots are more likely to ripen to hideous fruition behind a shield of encryption.

The staffers I talked to were unmoved by the argument that anyone smart enough to steal a nuclear device is probably smart enough to use PGP or some other uncompromised crypto standard. And never mind that the last people who popped a hooter in the World Trade Center were able to get it there without using any cryptography and while under FBI surveillance.

We are dealing with religion here. Though only ten American lives have been lost to terrorism in the last two years, the primacy of this threat has become as much an article of faith with these guys as the Catholic conviction that human life begins at conception or the Mormon belief that the Lost Tribe of Israel crossed the Atlantic in submarines.

In the spirit of openness and compromise, they invited the Electronic Frontier Foundation to submit other solutions to the "problem" of the nuclear-enabled terrorist than key escrow devices, but they would not admit into discussion the argument that such a threat might, in fact, be some kind of phantasm created by the spooks to ensure their lavish budgets into the post-Cold War era.

As to the possibility that good old-fashioned investigative techniques

might be more valuable in preventing their show-case catastrophe (as it was after the fact in finding the alleged perpetrators of the last attack on the World Trade Center), they just hunkered down and said that when wiretaps were necessary, they were damned well necessary.

When I asked about the business that American companies lose because of their inability to export good encryption products, one staffer essentially dismissed the market, saying that total world trade in crypto goods was still less than a billion dollars. (Well, right. Thanks more to the diligent efforts of the National Security Agency than to dim sales potential.)

I suggested that a more immediate and costly real-world effect of their policies would be to reduce national security by isolating American commerce, owing to a lack of international confidence in the security of our data lines. I said that Bruce Sterling's fictional data-enclaves in places like the Turks and Caicos Islands were starting to look real-world inevitable.

They had a couple of answers to this, one unsatisfying and the other scary. The unsatisfying answer was that the international banking community could just go on using DES, which still seemed robust enough to them. (DES is the old federal Data Encryption Standard, thought by most cryptologists to be nearing the end of its credibility.)

More frightening was their willingness to counter the data-enclave future with one in which no data channels anywhere would be secure from examination by one government or another. Pointing to unnamed other countries that were developing their own mandatory standards and restrictions regarding cryptography, they said words to the effect of, "Hey, it's not like you can't outlaw the stuff. Look at France."

Of course, they have also said repeatedly - and for now I believe them - that they have absolutely no plans to outlaw non-Clipper crypto in the US. But that doesn't mean that such plans wouldn't develop in the presence of some pending "emergency." Then there is that White House briefing document, issued at the time Clipper was first announced, which asserts that no US citizen "as a matter of right, is entitled to an unbreakable commercial encryption product."

Now why, if it's an ability they have no intention of contesting, do they feel compelled to declare that it's not a right? Could it be that they are preparing us for the laws they'll pass after some bearded fanatic has gotten himself a surplus nuke and used something besides Clipper to conceal his plans for it?

If they are thinking about such an eventuality, we should be doing so as well. How will we respond? I believe there is a strong, though currently untested, argument that outlawing unregulated crypto would violate the First Amendment, which surely protects the manner of our speech as clearly as it protects the content.

But of course the First Amendment is, like the rest of the Constitution, only as good as the government's willingness to uphold it. And they are, as I say, in the mood to protect our safety over our liberty.

This is not a mind-frame against which any argument is going to be very effective. And it appeared that they had already heard and rejected every argument I could possibly offer.

In fact, when I drew what I thought was an original comparison between their stand against naturally proliferating crypto and the folly of King Canute (who placed his throne on the beach and commanded the tide to leave him dry), my government opposition looked pained and said he had heard that one almost as often as jokes about roadkill on the information superhighway.

I hate to go to war with them. War is always nastier among friends. Furthermore, unless they've decided to let the National Security Agency design the rest of the National Information Infrastructure as well, we need to go on working closely with them on the whole range of issues like access, competition, workplace privacy, common carriage, intellectual property, and such. Besides, the proliferation of strong crypto will probably happen eventually no matter what they do.

But then again, it might not. In which case we could shortly find ourselves under a government that would have the automated ability to log the time, origin and recipient of every call we made, could track our physical whereabouts continuously, could keep better account of our financial transactions than we do, and all without a warrant. Talk about crime prevention!

Worse, under some vaguely defined and surely mutable "legal authority," they also would be able to listen to our calls and read our e-mail without having to do any backyard rewiring. They wouldn't need any permission at all to monitor overseas calls.

If there's going to be a fight, I'd rather it be with this government than the one we'd likely face on that hard day.

Hey, I've never been a paranoid before. It's always seemed to me that most governments are too incompetent to keep a good plot strung together all the

way from coffee break to quitting time. But I am now very nervous about the government of the United States of America.

Because Bill 'n' Al, whatever their other new-paradigm virtues, have allowed the very old-paradigm trogs of the Guardian Class to define as their highest duty the defense of America against an enemy that exists primarily in the imagination - and is therefore capable of anything.

To assure absolute safety against such an enemy, there is no limit to the liberties we will eventually be asked to sacrifice. And, with a Clipper Chip in every phone, there will certainly be no technical limit on their ability to enforce those sacrifices.

WHAT YOU CAN DO

GET CONGRESS TO LIFT THE CRYPTO EMBARGO

The administration is trying to impose Clipper on us by manipulating market forces. By purchasing massive numbers of Clipper devices, they intend to induce an economy of scale which will make them cheap while the export embargo renders all competition either expensive or nonexistent.

We have to use the market to fight back. While it's unlikely that they'll back down on Clipper deployment, the Electronic Frontier Foundation believes that with sufficient public involvement, we can get Congress to eliminate the export embargo.

Rep. Maria Cantwell, D-Washington, has a bill (H.R. 3627) before the Economic Policy, Trade, and Environment Subcommittee of the House Committee on Foreign Affairs that would do exactly that. She will need a lot of help from the public. They may not care much about your privacy in DC, but they still care about your vote.

Please signal your support of H.R. 3627, either by writing her directly or e-mailing her at cantwell@eff.org. Messages sent to that address will be printed out and delivered to her office. In the subject header of your message, please include the words "support HR 3627." In the body of your message, express your reasons for supporting the bill. You may also express your sentiments to Rep. Lee Hamilton, D-Indiana, the House Committee on Foreign Affairs chair, by e-mailing hamilton@eff.org.

Furthermore, since there is nothing quite as powerful as a letter from a constituent, you should check the following list of subcommittee and committee members to see if your congressional representative is among them. If so, please copy them your letter to Rep. Cantwell.

> Economic Policy, Trade, and Environment Subcommittee:

Democrats: Sam Gejdenson (Chair), D-Connecticut; James Oberstar, D-Minnesota; Cynthia McKinney, D-Georgia; Maria Cantwell, D-Washington; Eric Fingerhut, D-Ohio; Albert R. Wynn, D-Maryland; Harry Johnston, D-Florida; Eliot Engel, D-New York; Charles Schumer, D-New York.

Republicans: Toby Roth (ranking), R-Wisconsin; Donald Manzullo, R-Illinois; Doug Bereuter, R-Nebraska; Jan Meyers, R-Kansas; Cass Ballenger, R-North Carolina; Dana Rohrabacher, R-California.

> House Committee on Foreign Affairs:

Democrats: Lee Hamilton (Chair), D-Indiana; Tom Lantos, D-California; Robert Torricelli, D-New Jersey; Howard Berman, D-California; Gary Ackerman, D-New York; Eni Faleomavaega, D-Somoa; Matthew Martinez, D-California; Robert Borski, D-Pennsylvania; Donal Payne, D-New Jersey; Robert Andrews, D-New Jersey; Robert Menendez, D-New Jersey; Sherrod Brown, D-Ohio; Alcee Hastings, D-Florida; Peter Deutsch, D-Florida; Don Edwards, D-California; Frank McCloskey, D-Indiana; Thomas Sawyer, D-Ohio; Luis Gutierrez, D-Illinois.

Republicans: Benjamin Gilman (ranking), R-New York; William Goodling, R-Pennsylvania; Jim Leach, R-Iowa; Olympia Snowe, R-Maine; Henry Hyde, R-Illinois; Christopher Smith, R-New Jersey; Dan Burton, R-Indiana; Elton Gallegly, R-California; Ileana Ros-Lehtinen, R-Florida; David Levy, R-New York; Lincoln Diaz-Balart, R-Florida; Ed Royce, R-California.

BOYCOTT CLIPPER DEVICES AND THE COMPANIES WHICH MAKE THEM.

Don't buy anything with a Clipper Chip in it. Don't buy any product from a company that manufactures devices with Big Brother inside. It is likely that the government will ask you to use Clipper for communications with the IRS or when doing business with federal agencies. They cannot, as yet, require you to do so. Just say no.

LEARN ABOUT ENCRYPTION AND EXPLAIN THE ISSUES TO YOUR UNWIRED FRIENDS

The administration is banking on the likelihood that this stuff is too technically obscure to agitate anyone but nerds like us. Prove them wrong by patiently explaining what's going on to all the people you know who have never touched a computer and glaze over at the mention of words like "cryptography."

Maybe you glaze over yourself. Don't. It's not that hard. For some hands-on experience, download a copy of PGP - Pretty Good Privacy - a shareware encryption engine which uses the robust RSA encryption algorithm. And learn to use it.

GET YOUR COMPANY TO THINK ABOUT EMBEDDING REAL CRYPTOGRAPHY IN ITS PRODUCTS

If you work for a company that makes software, computer hardware, or any kind of communications device, work from within to get them to incorporate RSA or some other strong encryption scheme into their products. If they say that they are afraid to violate the export embargo, ask them to consider manufacturing such products overseas and importing them back into the United States. There appears to be no law against that. Yet.

You might also lobby your company to join the Digital Privacy and Security Working Group, a coalition of companies and public interest groups - including IBM, Apple, Sun, Microsoft, and, interestingly, Clipper phone manufacturer AT&T - that is working to get the embargo lifted.

ENLIST!

Self-serving as it sounds coming from me, you can do a lot to help by becoming a member of one of these organizations. In addition to giving you access to the latest information on this subject, every additional member strengthens our credibility with Congress.

> Join the Electronic Frontier Foundation by writing membership@eff.org.

> Join Computer Professionals for Social Responsibility by e-mailing cpsr.info@cpsr

.org. CPSR is also organizing a protest, to which you can lend your support by sending e-mail to clipper.petition@cpsr.org with "I oppose Clipper" in the message body. Ftp/gopher/WAIS to cpsr.org /cpsr/privacy/

crypto/clipper for more info.

In his LA speech, Gore called the development of the NII "a revolution." And it is a revolutionary war we are engaged in here. Clipper is a last ditch attempt by the United States, the last great power from the old Industrial Era, to establish imperial control over cyberspace. If they win, the most liberating development in the history of humankind could

become, instead, the surveillance system which will monitor our grandchildren's morality. We can be better ancestors than that.

San Francisco, California

Wednesday, February 9, 1994

* * *

John Perry Barlow (barlow@eff.org) is co-founder and Vice-Chairman of the Electronic Frontier Foundation, a group which defends liberty, both in Cyberspace and the Physical World. He has three daughters.

----- End of Forwarded Message

From: Dave Banisar <Banisar@washofc.cpsr.org>
Date: 15-FEB-1994 13:45:57
Description: Over 10,000 Have Signed Petition to Oppose Clipper

==========

Washington,DC
February 15,1994

Computer Professionals for Social Responsibility (CPSR)

OVER 10,000 SIGN PETITION TO OPPOSE CLIPPER

In only two weeks, over 10,000 users of the nation's computer networks have signed the CPSR petition calling for President Clinton to withdraw the Clipper proposal.

Opposition has been widespread, from CEOs of large firms to college students in small towns, from librarians and civil libertarians to computer programmers and product marketers.

To sign the petition, email <clipper.petition@cpsr.org> with the message "I Oppose Clipper"

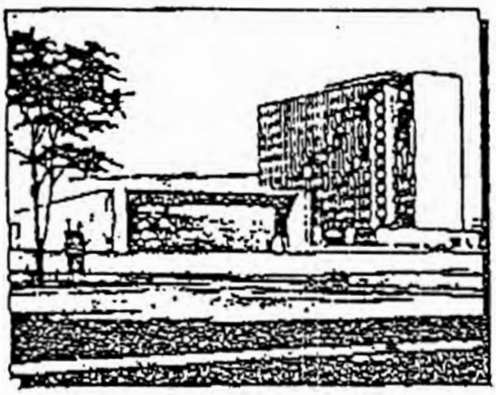
Encourage friends to sign.

In 1990, over 30,000 people sent email message to Lotus asking that a product containing detailed personal information called "Marketplace" be withdrawn. Eventually Lotus withdrew the product.

CPSR is a non-profit, membership organization based in Palo Alto, CA. CPSR's mission is to provide analysis of the effects of new technological developments on society. For more information, please email cpsr@cpsr.org or call 415-322-3778.

==========

NIST | UNITED STATES DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
Gaithersburg, Maryland 20899



CANON FAX L-700
COMMERCIAL FAX
301 948-1784
VOICE
301 975-3587

COMPUTER SYSTEMS LABORATORY

To: Mike Nelson
Tele:
Fax: 202|456-6023

From: Ed Roback
Tele: 975-3696
Fax: 948-1784

No. of pages transmitted 12
(including cover sheet)

Brief Message:
FYI, 2 items of interest.

If transmittal is unreadable or message incomplete, please call Sherry Hankey x2820.

Answers To

Frequently Asked Questions About Today's Cryptography

Paul Fahn
RSA Laboratories
100 Marine Parkway
Redwood City, CA 94065

Version 2.0, draft 2f.
Last update: September 20, 1993.

Part #002-903002-200-02f-000

Contents

1 General

1.1 *What is encryption?*

Encryption is the transformation of data into a form unreadable by anyone without a secret decryption key. Its purpose is to ensure privacy by keeping the information hidden from anyone for whom it is not intended, even those who can see the encrypted data. For example, one may wish to encrypt files on a hard disk to prevent an intruder from reading them.

In a multi-user setting, encryption allows secure communication over an insecure channel. The general scenario is as follows: Alice wishes to send a message to Bob so that no one else besides Bob can read it. Alice encrypts the message, which is called the plaintext, with an encryption key; the encrypted message, called the ciphertext, is sent to Bob. Bob decrypts the ciphertext with the decryption key and reads the message. An attacker, Charlie, may either try to obtain the secret key or to recover the plaintext without using the secret key. In a secure cryptosystem, the plaintext cannot be recovered from the ciphertext except by using the decryption key. In a symmetric cryptosystem, a single key serves as both the encryption and decryption keys.

Cryptography has been around for millennia; see Kahn [37] for a good history of cryptography; see Rivest [69] and Brassard [10] for an introduction to modern cryptography.

1.2 *What is authentication? What is a digital signature?*

Authentication in a digital setting is a process whereby the receiver of a digital message can be confident of the identity of the sender and/or the integrity of the message. Authentication protocols can be based on either conventional secret-key cryptosystems like DES or on public-key systems like RSA; authentication in public-key systems uses digital signatures.

In this document, authentication will generally refer to the use of digital signatures, which play a function for digital documents similar to that played by handwritten signatures for printed documents: the signature is an unforgeable piece of data asserting that a named person wrote or otherwise agreed to the document to which the signature is attached. The recipient, as well as a third party, can verify both that the document did indeed originate from the person whose signature is attached and that the document has not been altered since it was signed. A secure digital signature system thus consists of two parts: a method of signing a document such that forgery is infeasible, and a method of verifying that a signature was actually generated by whomever it represents. Furthermore, secure digital signatures cannot be repudiated; i.e., the signer of a document cannot later disown it by claiming it was forged.

Unlike encryption, digital signatures are a recent development, the need for which has arisen with the proliferation of digital communications.

1.3 *What is public-key cryptography?*

Traditional cryptography is based on the sender and receiver of a message knowing and using the same secret key: the sender uses the secret key to encrypt the message, and the receiver uses the same secret key to decrypt the message. This method is known as secret-key cryptography. The main problem is getting the sender and receiver to agree on the secret key without anyone else finding out. If they are in separate physical locations, they must trust a courier, or a phone system, or some other transmission system to not disclose the secret key being communicated. Anyone who overhears or intercepts the key in transit can later read all messages encrypted using that key. The generation, transmission and storage of keys is called key management; all cryptosystems must deal with key management issues. Secret-key cryptography often has difficulty providing secure key management.

Public-key cryptography was invented in 1976 by Whitfield Diffie and Martin Hellman [29] in order to solve the key management problem. In the new system, each person gets a pair of keys, called the public key and the private key. Each person's public key is published while the private key is kept secret. The need for sender and receiver to share secret information is eliminated: all communications involve only public keys, and no private key is ever transmitted or shared. No longer is it necessary to trust some communications channel to be secure against eavesdropping or betrayal. Anyone can send a confidential message just using public information, but it can only be decrypted with a private key that is in the sole possession of the intended recipient. Furthermore, public-key cryptography can be used for authentication (digital signatures) as well as for privacy (encryption).

Here's how it works for encryption: when Alice wishes to send a message to Bob, she looks up Bob's public key in a directory, uses it to encrypt the message and sends it off. Bob then uses his private key to decrypt the message and read it. No one listening in can decrypt the message. Anyone can send an encrypted message to Bob but only Bob can read it. Clearly, one requirement is that no one can figure out the private key from the corresponding public key.

Here's how it works for authentication: Alice, to sign a message, does a computation involving both her private key and the message itself; the output is called the digital signature and is attached to the message, which is then sent. Bob, to verify the signature, does some computation involving the message, the purported signature, and Alice's public key. If the results properly hold in a simple mathematical relation, the signature is verified as genuine; otherwise, the signature may be fraudulent or the message altered, and they are discarded.

A good history of public-key cryptography, by one of its inventors, is given by Diffie [27].

1.4 *What are the advantages and disadvantages of public-key cryptography over secret-key cryptography?*

The primary advantage of public-key cryptography is increased security: the private keys do not ever need to be transmitted or revealed to anyone. In a secret-key system, by contrast, there is always a chance that an enemy could discover the secret key while it is being transmitted.

Another major advantage of public-key systems is that they can provide a method for digital signatures. Authentication via secret-key systems requires the sharing of some secret and sometimes requires trust of a third party as well. A sender can then repudiate a previously signed message by claiming that the shared secret was somehow compromised by one of the parties sharing the secret. For example, the Kerberos secret-key authentication system [79] involves a central database that keeps copies of the secret keys of all users; a Kerberos-authenticated message would most likely not be held legally binding, since an attack on the database would allow widespread forgery. Public-key authentication, on the other hand, prevents this type of repudiation; each user has sole responsibility for protecting his or her private key. This property of public-key authentication is often called non-repudiation.

Furthermore, digitally signed messages can be proved authentic to a third party, such as a judge, thus allowing such messages to be legally binding. Secret-key authentication systems such as Kerberos were designed to authenticate access to network resources, rather than to authenticate documents, a task which is better achieved via digital signatures.

A disadvantage of using public-key cryptography for encryption is speed: there are popular secret-key encryption methods which are significantly faster than any currently available public-key encryption method. But public-key cryptography can share the burden with secret-key cryptography to get the best of both worlds.

For encryption, the best solution is to combine public- and secret-key systems in order to get both the security advantages of public-key systems and the speed advantages of secret-key systems. The public-key system can be used to encrypt a secret key which is then used to encrypt the bulk of a file or message. This is explained in more detail in Question 2.12 in the case of RSA. *Public-key cryptography is not meant to replace secret-key cryptography, but rather to supplement it, to make it more secure.* The first use of public-key techniques was for secure key exchange in an otherwise secret-key system [29]; this is still one of its primary functions.

Secret-key cryptography remains extremely important and is the subject of much ongoing study and research. Some secret-key encryption systems are discussed in Questions 5.1 and 5.5.

1.5 *Is cryptography patentable in the U.S.?*

Cryptographic systems are patentable. Many secret-key cryptosystems have been patented, including DES (see Question 5.1). The basic ideas of public-key cryptography are contained in U.S. Patent 4,200,770, by M. Hellman, W. Diffie, and R. Merkle, issued 4/29/80 and in U.S. Patent 4,218,582, by M. Hellman and R. Merkle, issued 8/19/80; similar patents have been issued throughout the world. The exclusive licensing rights to both patents are held by Public Key Partners (PKP), of Sunnyvale, California, which also holds the rights to the RSA patent (see Question 2.19). Usually all of these public-key patents are licensed together.

All legal challenges to public-key patents have been settled before judgment. In a recent case, for example, PKP brought suit against the TRW Corporation which was using public-key cryptography (the ElGamal system) without a license; TRW claimed it did not need to license. In June 1992 a settlement was reached in which TRW agreed to license to the patents.

Some patent applications for cryptosystems have been blocked by intervention by the NSA (see Question 7.3) or other intelligence or defense agencies, under the authority of the Invention Secrecy Act of 1940 and the National Security Act of 1947; see Landau [46] for some recent cases related to cryptography.

1.6 *Is cryptography exportable from the U.S.?*

All cryptographic products need export licenses from the State Department, acting under authority of the International Traffic in Arms Regulation (ITAR), which defines cryptographic devices, including software, as munitions. The U.S. government has historically been reluctant to grant export licenses for encryption products stronger than some basic level (not publicly stated).

Under current regulations, a vendor seeking to export a product using cryptography first submits an request to the State Department's Defense Trade Control office. Export jurisdiction may then be passed to the Department of Commerce, whose export procedures are generally simple and efficient. If jurisdiction remains with the State Department, further review, perhaps lengthy, is required before export is either approved or denied; the National Security Agency (NSA, see Question 7.3) may become directly involved at this point. The details of the export approval process change frequently.

The NSA has de facto control over export of cryptographic products. The State Department will not grant a license without NSA approval and routinely grants licenses whenever NSA does approve. Therefore, the policy decisions over exporting cryptography ultimately rest with the NSA.

It is the stated policy of the NSA not to restrict export of cryptography for authentication; it is only concerned with the use of cryptography for privacy. A vendor seeking to export a product for authentication only will be granted an export license as long as it can demonstrate that the product cannot be easily

modified for encryption; this is true even for very strong systems, such as RSA with large key sizes. Furthermore, the bureaucratic procedures are simpler for authentication products than for privacy products. An authentication product needs NSA and State Dept. approval only once, whereas an encryption product may need approval for every sale or every product revision.

Export policy is currently a matter of great controversy, as many software and hardware vendors consider current export regulations overly restrictive and burdensome. The Software Publishers Association (SPA), a software industry group, has recently been negotiating with the government in order to get export license restrictions eased; one agreement was reached that allows simplified procedures for export of two bulk encryption ciphers, RC2 and RC4 (see Question 8.6), when the key size is limited. Also, export policy is less restrictive for foreign subsidiaries and overseas offices of U.S. companies.

In March 1992, the Computer Security and Privacy Advisory Board voted unanimously to recommend a national review of cryptography policy, including export policy. The Board is an official advisory board to NIST (see Question 7.1) whose members are drawn from both the government and the private sector. The Board stated that a public debate is the only way to reach a consensus policy to best satisfy competing interests: national security and law enforcement agencies like restrictions on cryptography, especially for export, whereas other government agencies and private industry want greater freedom for using and exporting cryptography. Export policy has traditionally been decided solely by agencies concerned with national security, without much input from those who wish to encourage commerce in cryptography. U.S. export policy may undergo significant change in the next few years.

2 RSA

2.1 *What is RSA?*

RSA is a public-key cryptosystem for both encryption and authentication; it was invented in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman [74]. It works as follows: take two large primes, p and q, and find their product $n = pq$; n is called the modulus. Choose a number, e, less than n and relatively prime to $(p-1)(q-1)$, and find its inverse, d, mod $(p-1)(q-1)$, which means that $ed \equiv 1 \bmod (p-1)(q-1)$; e and d are called the public and private exponents, respectively. The public key is the pair (n, e); the private key is d. The factors p and q must be kept secret, or destroyed.

It is difficult (presumably) to obtain the private key d from the public key (n, e). If one could factor n into p and q, however, then one could obtain the private key d. Thus the entire security of RSA is predicated on the assumption that factoring is difficult; an easy factoring method would "break" RSA (see Questions 2.5 and 4.4).

Here is how RSA can be used for privacy and authentication (in practice, actual use is slightly different; see Questions 2.12 and 2.13):

RSA privacy (encryption): suppose Alice wants to send a private message, m, to Bob. Alice creates the ciphertext c by exponentiating: $c = m^e \bmod n$, where e and n are Bob's public key. To decrypt, Bob also exponentiates: $m = c^d \bmod n$, and recovers the original message m; the relationship between e and d ensures that Bob correctly recovers m. Since only Bob knows d, only Bob can decrypt.

RSA authentication: suppose Alice wants to send a signed document m to Bob. Alice creates a digital signature s by exponentiating: $s = m^d \bmod n$, where d and n belong to Alice's key pair. She sends s and m to Bob. To verify the signature, Bob exponentiates and checks that the message m is recovered: $m = s^e \bmod n$, where e and n belong to Alice's public key.

Thus encryption and authentication take place without any sharing of private keys: each person uses only other people's public keys and his or her own private key. Anyone can send an encrypted message or verify a signed message, using only public keys, but only someone in possession of the correct private key can decrypt or sign a message.

2.2 *Why use RSA rather than DES?*

RSA is not an alternative or replacement for DES; rather it supplements DES (or any other fast bulk encryption cipher) and is used together with DES in a secure communications environment. (Note: for an explanation of DES, see Question 5.1.)

RSA allows two important functions not provided by DES: secure key exchange without prior exchange of secrets, and digital signatures. For encrypting messages, RSA and DES are usually combined as follows: first the message is encrypted with a random DES key, and then, before being sent over an insecure communications channel, the DES key is encrypted with RSA. Together, the DES-encrypted message and the RSA-encrypted DES key are sent. This protocol is known as an RSA digital envelope.

One may wonder, why not just use RSA to encrypt the whole message and not use DES at all? Although this may be fine for small messages, DES (or another cipher) is preferable for larger messages because it is much faster than RSA (see Question 2.3).

In some situations, RSA is not necessary and DES alone is sufficient. This includes multi-user environments where secure DES-key agreement can take place, for example by the two parties meeting in private. Also, RSA is usually not necessary in a single-user environment; for example, if you want to keep your personal files encrypted, just do so with DES using, say, your personal password as the DES key. RSA, and public-key cryptography in general, is best suited for a multi-user environment. Also, any system in which digital signatures are desired needs RSA or some other public-key system.

2.3 *How fast is RSA?*

An "RSA operation," whether for encrypting or decrypting, signing or verifying, is essentially a modular exponentiation, which can be performed by a series of modular multiplications.

In practical applications, it is common to choose a small public exponent for the public key; in fact, entire groups of users can use the same public exponent. This makes encryption faster than decryption and verification faster than signing. Algorithmically, public-key operations take $O(k^2)$ steps, private key operations take $O(k^3)$ steps, and key generation takes $O(k^4)$ steps, where k is the number of bits in the modulus; O-notation refers to the an upper bound on the asymptotic running time of an algorithm [22].

There are many commercially available hardware implementations of RSA, and there are frequent announcements of newer and faster chips. The fastest current RSA chip [76] has a throughput greater than 600 Kbits per second with a 512-bit modulus, implying that it performs over 1000 RSA private-key operations per second. It is expected that RSA speeds will reach 1 Mbit/second within a year or so.

By comparison, DES is much faster than RSA. In software, DES is generally at least 100 times as fast as RSA. In hardware, DES is between 1,000 and 10,000 times as fast, depending on the implementations. RSA will probably narrow the gap a bit in coming years, as it finds growing commercial markets, but will never match the performance of DES.

2.4 *How much extra message length is caused by using RSA?*

Only a very small amount of data expansion is involved when using RSA. For encryption, a message may be padded to a length that is a multiple of the block length, usually 64 bits, since RSA is usually combined with a secret-key block cipher such as DES (see Question 2.12). Encrypting the DES key takes as many additional bits as the size of the RSA modulus.

For authentication, an RSA digital signature is appended to a document. An RSA signature, including information such as the name of the signer, is typically a few hundred bytes long. One or more certificates (see Question 3.5) may be included as well; certificates can be used in conjunction with any digital signature method. A typical RSA certificate is a few hundred bytes long.

2.5 *What would it take to break RSA?*

There are a few possible interpretations of "breaking RSA". The most damaging would be for an attacker to discover the private key corresponding to a given public key; this would enable the attacker both to read all messages encrypted with the public key and to forge signatures. The obvious way to do this attack is to factor the public modulus, n, into its two prime factors, p and q. From

p, q, and e, the public exponent, the attacker can easily get d, the private key. The hard part is factoring n; the security of RSA depends of factoring being difficult. In fact, the task of recovering the private key is equivalent to the task of factoring the modulus: you can use d to factor n, as well as use the factorization of n to find d. See Questions 4.5 and 4.6 regarding the state of the art in factoring. It should be noted that hardware improvements alone will not weaken RSA, as long as appropriate key lengths are used; in fact, hardware improvements should increase the security of RSA (see Question 4.5).

Another way to break RSA is to find a technique to compute e-th roots mod n. Since $c = m^e$, the e-th root of c is the message m. This attack would allow someone to recover encrypted messages and forge signatures even without knowing the private key. This attack is not known to be equivalent to factoring. No methods are currently known that attempt to break RSA in this way.

The attacks just mentioned are the only ways to break RSA in such a way as to be able to recover all messages encrypted under a given key. There are other methods, however, which aim to recover single messages; success would not enable the attacker to recover other messages encrypted with the same key.

The simplest single-message attack is the guessed plaintext attack. An attacker sees a ciphertext, guesses that the message might be "Attack at dawn", and encrypts this guess with the public key of the recipient; by comparison with the actual ciphertext, the attacker knows whether or not the guess was correct. This attack can be thwarted by appending some random bits to the message. Another single-message attack can occur if someone sends the same message m to three others, who each have public exponent $e = 3$. An attacker who knows this and sees the three messages will be able to recover the message m; this attack and ways to prevent it are discussed by Hastad [35]. There are also some "chosen ciphertext" attacks, in which the attacker creates some ciphertext and gets to see the corresponding plaintext, perhaps by tricking a legitimate user into decrypting a fake message; Davida [23] gives some examples.

Of course, there are also attacks that aim not at RSA itself but at a given insecure implementation of RSA; these do not count as "breaking RSA" because it is not any weakness in the RSA algorithm that is exploited, but rather a weakness in a specific implementation. For example, if someone stores his private key insecurely, an attacker may discover it. One cannot emphasize strongly enough that to be truly secure RSA requires a secure implementation; mathematical security measures, such as choosing a long key size, are not enough. In practice, most successful attacks will likely be aimed at insecure implementations and at the key management stages of an RSA system. See Section 3 for discussion of secure key management in an RSA system.

2.6 *Are strong primes necessary in RSA?*

In the literature pertaining to RSA, it has often been suggested that in choosing a key pair, one should use "strong" primes p and q to generate the modulus n.

Strong primes are those with certain properties that make the product n hard to factor by specific factoring methods; such properties have included, for example, the existence of a large prime factor of $p-1$ and a large prime factor of $p+1$. The reason for these concerns is that some factoring methods are especially suited to primes p such that $p-1$ or $p+1$ has only small factors; strong primes are resistant to these attacks.

However, recent advances in factoring (see Question 4.6) appear to have obviated the advantage of strong primes; the elliptic curve factoring algorithm is one such advance. The new factoring methods have as good a chance of success on strong primes as on "weak" primes; therefore, choosing strong primes does not significantly increase resistance to attacks. So for now the answer is negative: strong primes are not necessary when using RSA, although there is no danger in using them, except that it takes longer to generate a key pair. However, new factoring algorithms may be developed in the future which once again target primes with certain properties; if so, choosing strong primes may again help to increase security.

2.7 *How large a modulus (key) should be used in RSA?*

The best size for an RSA modulus depends on one's security needs. The larger the modulus, the greater the security but also the slower the RSA operations. One should choose a modulus length upon consideration, first, of one's security needs, such as the value of the protected data and how long it needs to be protected, and, second, of how powerful one's potential enemy is. It is also possible that a larger key size will allow a digitally signed document to be valid for a longer time; see Question 3.17.

A good analysis of the security obtained by a given modulus length is given by Rivest [72], in the context of discrete logarithms modulo a prime, but it applies to RSA as well. Rivest's estimates imply that a 512-bit modulus can be factored with an $8.2 million effort, less in the future. It may therefore be advisable to use a longer modulus, perhaps 768 bits in length. Those with extremely valuable data (or large potential damage from digital forgery) may want to use a still longer modulus. A certifying authority (see Question 3.5) might use a modulus of length 1000 bits or more, because the validity of so many other key pairs depends on the security of the one central key.

The key of an individual user will expire after a certain time, say, two years (see Question 3.12). Upon expiration, the user will generate a new key which should be at least a few digits longer than the old key to reflect the speed increases of computers over the two years. Recommended key length schedules will probably be published by some authority or public body.

Users should keep in mind that the estimated times to break RSA are averages only. A large factoring effort, attacking many thousands of RSA moduli, may succeed in factoring at least one in a reasonable time. Although the security of any individual key is still strong, with some factoring methods there is

always a small chance that the attacker may get lucky and factor it quickly.

As for the slowdown caused by increasing the key size (see Question 2.3), doubling the modulus length would, on average, increase the time required for public-key operations (encryption and signature verification) by a factor of 4, and increase the time taken by private key operations (decryption and signing) by a factor of 8. The reason that public-key operations are affected less than private-key operations is that the public exponent can remain fixed when the modulus is increased, whereas the private exponent increases proportionally. Key generation time would increase by a factor of 16 upon doubling the modulus, but this is a relatively infrequent operation for most users.

2.8 *How large should the primes be?*

The two primes, p and q, which compose the modulus, should be of roughly equal length; this will make the modulus harder to factor than if one of the primes was very small. Thus if one chooses to use a 512-bit modulus, the primes should each have length approximately 256 bits.

2.9 *How does one find random numbers for keys?*

One needs a source of random numbers in order to find two random primes to compose the modulus. If one used a predictable method of generating the primes, an adversary could mount an attack by trying to recreate the key generation process.

Random numbers obtained from a physical process are in principle the best. One could use a hardware device, such as a diode; some are sold commercially on computer add-in boards for this purpose. Another idea is to use physical movements of the computer user, such as keystroke timings measured in microseconds. By whichever method, the random numbers may still contain some correlations preventing sufficient statistical randomness. Therefore, it is best to run them through a good hash function (see Question 8.2) before actually using them.

Another approach is to use a pseudorandom number generator fed by a random seed. Since these are deterministic algorithms, it is important to find one that is very unpredictable and also to use a truly random seed. There is a wide literature on the subject of pseudorandom number generators. See Knuth [41] for an introduction.

Note that one does not need random numbers to determine the public and private exponents in RSA, after choosing the modulus. One can simply choose an arbitrary value for the public exponent, which then determines the private exponent, or vice versa.

2.10 *What if users of RSA run out of distinct primes?*

There are enough prime numbers that RSA users will never run out of them. For example, the number of primes of length 512 bits or less exceeds 10^{150}, according to the prime number theorem; this is more than the number of atoms in the known universe.

2.11 *How do you know if a number is prime?*

It is generally recommended to use probabilistic primality testing, which is much quicker than actually proving a number prime. One can use a probabilistic test that decides if a number is prime with probability of error less than 2^{-100}. For further discussion of some primality testing algorithms, see the papers in the bibliography of [5]. For some empirical results on the reliability of simple primality tests see Rivest [70]; one can perform very fast primality tests and be extremely confident in the results. A simple algorithm for choosing probable primes was recently analyzed by Brandt and Damgård [9].

2.12 *How is RSA used for encryption in practice?*

RSA is combined with a secret-key cryptosystem, such as DES, to encrypt a message by means of an RSA digital envelope.

Suppose Alice wishes to send an encrypted message to Bob. She first encrypts the message with DES, using a randomly chosen DES key. Then she looks up Bob's public key and uses it to encrypt the DES key. The DES-encrypted message and the RSA-encrypted DES key together form the RSA digital envelope and are sent to Bob. Upon receiving the digital envelope, Bob decrypts the DES key with his private key, then uses the DES key to decrypt to message itself.

2.13 *How is RSA used for authentication in practice?*

Suppose Alice wishes to send a signed message to Bob. She uses a hash function on the message (see Question 8.2) to create a message digest, which serves as a "digital fingerprint" of the message. She then encrypts the message digest with her RSA private key; this is the digital signature, which she sends to Bob along with the message itself. Bob, upon receiving the message and signature, decrypts the signature with Alice's public key to recover the message digest. He then hashes the message with the same hash function Alice used and compares the result to the message digest decrypted from the signature. If they are exactly equal, the signature has been successfully verified and he can be confident that the message did indeed come from Alice. If, however, they are not equal, then the message either originated elsewhere or was altered after it was signed, and he rejects the message. Note that for authentication, the roles of the public and

private keys are converse to their roles in encryption, where the public key is used to encrypt and the private key to decrypt.

In practice, the public exponent is usually much smaller than the private exponent; this means that the verification of a signature is faster than the signing. This is desirable because a message or document will only be signed by an individual once, but the signature may be verified many times.

It must be infeasible for anyone to either find a message that hashes to a given value or to find two messages that hash to the same value. If either were feasible, an intruder could attach a false message onto Alice's signature. Hash functions such as MD4 and MD5 (see Question 8.3) have been designed specifically to have the property that finding a match is infeasible, and are therefore considered suitable for use in cryptography.

One or more certificates (see Question 3.5) may accompany a digital signature. A certificate is a signed document attesting to the identity and public key of the person signing the message. Its purpose is to prevent someone from impersonating someone else, using a phony key pair. If a certificate is present, the recipient (or a third party) can check the authenticity of the public key, assuming the certifier's public key is itself trusted.

2.14 *Does RSA help detect altered documents and transmission errors?*

An RSA digital signature is superior to a handwritten signature in that it attests to the contents of a message as well as to the identity of the signer. As long as a secure hash function (see Question 8.2) is used, there is no way to take someone's signature from one document and attach it to another, or to alter the signed message in any way. The slightest change in a signed document will cause the digital signature verification process to fail. Thus, RSA authentication allows people to check the integrity of signed documents. Of course, if a signature verification fails, it may be unclear whether there was an attempted forgery or simply a transmission error.

2.15 *What are alternatives to RSA?*

Many other public-key cryptosystems have been proposed, as a look through the proceedings of the annual Crypto and Eurocrypt conferences quickly reveals. A mathematical problem called the knapsack problem was the basis for several systems [52], but these have lost favor because several versions were broken. Another system, designed by ElGamal [30], is based on the discrete logarithm problem. The ElGamal system was, in part, the basis for several later signature methods, including one by Schnorr [75], which in turn was the basis for DSS, the digital signature standard proposed by NIST (see Question 6.8). Because of the NIST proposal, the relative merits of these signature systems versus RSA signatures has received a lot of attention; see [57] for a discussion. The ElGamal

system has been used successfully in applications; it is slower for encryption and verification than RSA and its signatures are larger than RSA signatures.

In 1976, before RSA, Diffie and Hellman [29] proposed a system for key exchange only; it permits secure exchange of keys in an otherwise conventional secret-key system. This system is in use today.

Cryptosystems based on mathematical operations on elliptic curves have also been proposed [43, 56], as have cryptosystems based on discrete exponentiation in the finite field $GF(2^n)$. The latter are very fast in hardware; however, doubts have been raised about their security because the underlying problem may be easier to solve than factoring [64, 34]. There are also some probabilistic encryption methods [8, 32], which have the attraction of being resistant to a guessed ciphertext attack (see Question 2.5), but at a cost of data expansion. In probabilistic encryption, the same plaintext encrypted twice under the same key will give, with high probability, two different ciphertexts.

For digital signatures, Rabin [68] proposed a system which is provably equivalent to factoring; this is an advantage over RSA, where one may still have a lingering worry about an attack unrelated to factoring. Rabin's method is susceptible to a chosen message attack, however, in which the attacker tricks the user into signing messages of a special form. Another signature scheme, by Fiat and Shamir [31], is based on interactive zero-knowledge protocols, but can be adapted for signatures. It is faster than RSA and is provably equivalent to factoring, but the signatures are much larger than RSA signatures. Other variations, however, lessen the necessary signature length; see [17] for references. A system is "equivalent to factoring" if recovering the private key is provably as hard as factoring; forgery may be easier than factoring in some of the systems.

Advantages of RSA over other public-key cryptosystems include the fact that it can be used for both encryption and authentication, and that it has been around for many years and has successfully withstood much scrutiny. RSA has received far more attention, study, and actual use than any other public-key cryptosystem, and thus RSA has more empirical evidence of its security than more recent and less scrutinized systems. In fact, a large number of public-key cryptosystems which at first appeared secure were later broken; see [13] for some case histories.

2.16 *Is RSA currently in use today?*

The use of RSA is undergoing a period of rapid expansion and may become ubiquitous within a few years. It is currently used in a wide variety of products, platforms and industries around the world. It is found in many commercial software products and planned for many more. RSA is built into current or planned operating systems by Microsoft, Apple, Sun, and Novell. In hardware, RSA can be found in secure telephones, on Ethernet network cards, and on smart cards. RSA is also used internally in many institutions, including branches of the U.S. government, major corporations, national laboratories, and universities.

Adoption of RSA seems to be proceeding more quickly for authentication (digital signatures) than for privacy (encryption), perhaps in part because products for authentication are easier to export than those for privacy (see Question 1.6).

2.17 *Is RSA an official standard today?*

RSA is part of many official standards worldwide. The ISO (International Standards Organization) 9796 standard lists RSA as a compatible cryptographic algorithm, as does the Consultative Committee in International Telegraphy and Telephony (CCITT) X.509 security standard. RSA is part of the Society for Worldwide Interbank Financial Telecommunications (SWIFT) standard, the French financial industry's ETEBAC 5 standard, and the ANSI X9.31 draft standard for the U.S. banking industry. The Australian key management standard, AS2805.6.5.3, also specifies RSA.

RSA is found in Internet's proposed PEM (Privacy Enhanced Mail) standard (see Question 8.7) and the PKCS standard for the software industry (see Question 8.9). The OSI Implementors' Workshop (OIW) has issued implementers' agreements referring to PKCS and PEM, which each include RSA.

A number of other standards are currently being developed and will be announced over the next couple of years; many are expected to include RSA as either an endorsed or a recommended system for privacy and/or authentication. See [38] for a more comprehensive survey of cryptography standards.

2.18 *Is RSA a de facto standard? Why is a de facto standard important?*

RSA is the most widely used public-key cryptosystem today and has often been called a de facto standard. Regardless of the official standards, the existence of a de facto standard is extremely important for the development of a digital economy. If one public-key system is used everywhere for authentication, then signed digital documents can be exchanged between users in different nations using different software on different platforms; this interoperability is necessary for a true digital economy to develop.

The lack of secure authentication has been a major obstacle in achieving the promise that computers would replace paper; paper is still necessary almost everywhere for contracts, checks, official letters, legal documents, and identification. With this core of necessary paper transaction, it has not been feasible to evolve completely into a society based on electronic transactions. Digital signatures are the exact tool necessary to convert the most essential paper-based documents to digital electronic media. Digital signatures makes it possible, for example, to have leases, wills, passports, college transcripts, checks, and voter registration forms that exist only in electronic form; any paper version would

just be a "copy" of the electronic original. All of this is enabled by an accepted standard for digital signatures.

2.19 *Is RSA patented?*

RSA is patented under U.S. Patent 4,405,829, issued 9/20/83 and held by Public Key Partners (PKP), of Sunnyvale, California; the patent expires 17 years after issue, in 2000. RSA is usually licensed together with other public-key cryptography patents (see Question 1.5). PKP has a standard, royalty-based licensing policy which can be modified for special circumstances. If a software vendor, having licensed the public-key patents, incorporates RSA into a commercial product, then anyone who purchases the end product has the legal right to use RSA within the context of that software. The U.S. government can use RSA without a license because it was invented at MIT with partial government funding. RSA is not patented outside North America.

In North America, a license is needed to "make, use or sell" RSA. However, PKP usually allows free non-commercial use of RSA, with written permission, for personal, academic or intellectual reasons. Furthermore, RSA Laboratories has made available (in the U.S. and Canada) at no charge a collection of cryptographic routines in source code, including the RSA algorithm; it can be used, improved and redistributed non-commercially (see Question 8.10).

2.20 *Can RSA be exported from the U.S.?*

Export of RSA falls under the same U.S. laws as all other cryptographic products. See Question 1.6 for details.

RSA used for authentication is more easily exported than when used for privacy. In the former case, export is allowed regardless of key (modulus) size, although the exporter must demonstrate that the product cannot be easily converted to use for encryption. In the case of RSA used for privacy (encryption), the U.S. government generally does not allow export if the key size exceeds 512 bits. Export policy is currently a subject of debate, and the export status of RSA may well change in the next year or two.

Regardless of U.S. export policy, RSA is available abroad in non-U.S. products.

3 Key Management

3.1 *What key management issues are involved in public-key cryptography?*

Secure methods of key management are extremely important. In practice, most attacks on public-key systems will probably be aimed at the key management

levels, rather than at the cryptographic algorithm itself. The key management issues mentioned here are discussed in detail in later questions.

Users must be able to obtain securely a key pair suited to their efficiency and security needs. There must be a way to look up other people's public keys and to publicize one's own key. Users must have confidence in the legitimacy of others' public keys; otherwise an intruder can either change public keys listed in a directory, or impersonate another user. Certificates are used for this purpose. Certificates must be unforgeable, obtainable in a secure manner, and processed in such a way that an intruder cannot misuse them. The issuance of certificates must proceed in a secure way, impervious to attack. If someone's private key is lost or compromised, others must be made aware of this, so that they will no longer encrypt messages under the invalid public key nor accept messages signed with the invalid private key. Users must be able to store their private keys securely, so that no intruder can find it, yet the keys must be readily accessible for legitimate use. Keys need to be valid only until a specified expiration date. The expiration date must be chosen properly and publicized securely. Some documents need to have verifiable signatures beyond the time when the key used to sign them has expired.

Although most of these key management issues arise in any public-key cryptosystem, for convenience they are discussed here in the context of RSA.

3.2 *Who needs a key?*

Anyone who wishes to sign messages or to receive encrypted messages must have a key pair. People may have more than one key. For example, someone might have a key affiliated with his or her work and a separate key for personal use. Other entities will also have keys, including electronic entities such as modems, workstations, and printers, as well as organizational entities such as a corporate department, a hotel registration desk, or a university registrar's office.

3.3 *How does one get a key pair?*

Each user should generate his or her own key pair. It may be tempting within an organization to have a single site that generates keys for all members who request one, but this is a security risk because it involves the transmission of private keys over a network as well as catastrophic consequences if an attacker infiltrates the key-generation site. Each node on a network should be capable of local key generation, so that private keys are never transmitted and no external key source need be trusted. Of course, the local key generation software must itself be trustworthy. Secret-key authentication systems, such as Kerberos, often do not allow local key generation but instead use a central server to generate keys.

Once generated, a user must register his or her public key with some central administration, called a certifying authority. The certifying authority returns

to the user a certificate attesting to the veracity of the user's public key along with other information (see Questions 3.5 and following). Most users should not obtain more than one certificate for the same key, in order to simplify various bookkeeping tasks associated with the key.

3.4 *Should a public key or private key be shared among users?*

In RSA, each person should have a unique modulus and private exponent, i.e., a unique private key. The public exponent, on the other hand, can be common to a group of users without security being compromised. Some public exponents in common use today are 3 and $2^{16} + 1$; because these numbers are small, the public-key operations (encryption and signature verification) are fast relative to the private key operations (decryption and signing). If one public exponent becomes a standard, software and hardware can be optimized for that value.

In public-key systems based on discrete logarithms, such as ElGamal, Diffie-Hellman, or DSS, it has often been suggested that a group of people should share a modulus. This would make breaking a key more attractive to an attacker, however, because one could break every key with only slightly more effort than it would take to break a single key. To an attacker, therefore, the average cost to break a key is much lower with a common modulus than if every key has a distinct modulus. Thus one should be very cautious about using a common modulus; if a common modulus is chosen, it should be very large.

3.5 *What are certificates?*

Certificates are digital documents attesting to the binding of a public key to an individual or other entity. They allow verification of the claim that a given public key does in fact belong to a given individual. Certificates help prevent someone from using a phony key to impersonate someone else.

In their simplest form, certificates contain a public key and a name. As commonly used, they also contain the expiration date of the key, the name of the certifying authority that issued the certificate, the serial number of the certificate, and perhaps other information. Most importantly, it contains the digital signature of the certificate issuer. The most widely accepted format for certificates is defined by the CCITT X.509 international standard [19]; thus certificates can be read or written by any application complying with X.509. Further refinements are found in the PKCS set of standards (see Question 8.9), and the PEM standard (see Question 8.7). A detailed discussion of certificate format can also be found in Kent [40].

A certificate is issued by a certifying authority (see Question 3.7) and signed with the certifying authority's private key.

3.6 *How are certificates used?*

A certificate is displayed in order to generate confidence in the legitimacy of a public key. Someone verifying a signature can also verify the signer's certificate, to insure that no forgery or false representation has occurred. These steps can be performed with greater or lesser rigor depending on the context.

The most secure use of authentication involves enclosing one or more certificates with every signed message. The receiver of the message would verify the certificate using the certifying authority's public key and, now confident of the public key of the sender, verify the message's signature. There may be two or more certificates enclosed with the message, forming a hierarchical chain, wherein one certificate testifies to the authenticity of the previous certificate. At the end of a certificate hierarchy is a top-level certifying authority, which is trusted without a certificate from any other certifying authority. The public key of the top-level certifying authority must be independently known, for example by being widely published.

The more familiar the sender is to the receiver of the message, the less need there is to enclose, and to verify, certificates. If Alice sends messages to Bob every day, Alice can enclose a certificate chain on the first day, which Bob verifies. Bob thereafter stores Alice's public key and no more certificates or certificate verifications are necessary. A sender whose company is known to the receiver may need to enclose only one certificate (issued by the company), whereas a sender whose company is unknown to the receiver may need to enclose two certificates. A good rule of thumb is to enclose just enough of a certificate chain so that the issuer of the highest level certificate in the chain is well-known to the receiver.

According to the PKCS standards for public-key cryptography (see Question 8.9), every signature points to a certificate that validates the public key of the signer. Specifically, each signature contains the name of the issuer of the certificate and the serial number of the certificate. Thus even if no certificates are enclosed with a message, a verifier can still use the certificate chain to check the status of the public key.

3.7 *Who issues certificates and how?*

Certificates are issued by a certifying authority (CA), which can be any trusted central administration willing to vouch for the identities of those to whom it issues certificates. A company may issue certificates to its employees, a university to its students, a town to its citizens. In order to prevent forged certificates, the CA's public key must be trustworthy: a CA must either publicize its public key or provide a certificate from a higher-level CA attesting to the validity of its public key. The latter solution gives rise to hierarchies of CAs.

Certificate issuance proceeds as follows. Alice generates her own key pair and sends the public key to an appropriate CA with some proof of her identification.

The CA checks the identification and takes any other steps necessary to assure itself that the request really did come from Alice, and then sends her a certificate attesting to the binding between Alice and her public key, along with a hierarchy of certificates verifying the CA's public key. Alice can present this certificate chain whenever desired in order to demonstrate the legitimacy of her public key.

Since the CA must check for proper identification, organizations will find it convenient to act as a CA for its own members and employees. There will also be CAs that issue certificates to unaffiliated individuals.

Different CAs may issue certificates with varying levels of identification requirements. One CA may insist on seeing a driver's license, another may want the certificate request form to be notarized, yet another may want fingerprints of anyone requesting a certificate. Each CA should publish its own identification requirements and standards, so that verifiers can attach the appropriate level of confidence in the certified name-key bindings.

An example of a certificate-issuing protocol is Apple Computer's Open Collaborative Environment (OCE). Apple OCE users can generate a key pair and then request and receive a certificate for the public key; the certificate request must be notarized.

3.8 *What is a CSU, or, How do certifying authorities store their private keys?*

It is extremely important that private keys of certifying authorities are stored securely, because compromise would enable undetectable forgeries. One way to achieve the desired security is to store the key in a tamperproof box; such a box is called a Certificate Signing Unit, or CSU. The CSU would, preferably, destroy its contents if ever opened, and be shielded against attacks using electromagnetic radiation. Not even employees of the certifying authority should have access to the private key itself, but only the ability to use the private key in the process of issuing certificates.

There are many possible designs for CSUs; here is a description of one design found in some current implementations. The CSU is activated by a set of data keys, which are physical keys capable of storing digital information. The data keys use secret-sharing technology such that several people must all use their data keys to activate the CSU. This prevents one disgruntled CA employee from producing phony certificates.

Note that if the CSU is destroyed, say in a fire, no security is compromised. Certificates signed by the CSU are still valid, as long as the verifier uses the correct public key. Some CSUs will be manufactured so that a lost private key can be restored into a new CSU. See Question 3.10 for discussion of lost CA private keys.

Bolt, Beranek, and Newman (BBN) currently sells a CSU, and RSA Data Security sells a full-fledged certificate issuing system built around the BBN CSU.

3.9 *Are certifying authorities susceptible to attack?*

One can think of many attacks aimed at the certifying authority, which must be prepared against them.

Consider the following attack. Suppose Bob wishes to impersonate Alice. If Bob can convincingly sign messages as Alice, he can send a message to Alice's bank saying "I wish to withdraw $10,000 from my account. Please send me the money." To carry out this attack, Bob generates a key pair and sends the public key to a certifying authority saying "I'm Alice. Here is my public key. Please send me a certificate." If the CA is fooled and sends him such a certificate, he can then fool the bank, and his attack will succeed. In order to prevent such an attack the CA must verify that a certificate request did indeed come from its purported author, i.e., it must require sufficient evidence that it is actually Alice who is requesting the certificate. The CA may, for example, require Alice to appear in person and show a birth certificate. Some CAs may require very little identification, but the bank should not honor messages authenticated with such low-assurance certificates. Every CA must publicly state its identification requirements and policies; others can then attach an appropriate level of confidence to the certificates.

An attacker who discovers the private key of a certifying authority could then forge certificates. For this reason, a certifying authority must take extreme precautions to prevent illegitimate access to its private key. The private key should be kept in a high-security box, known as a Certificate Signing Unit, or CSU (see Question 3.8).

The certifying authority's public key might be the target of an extensive factoring attack. For this reason, CAs should use very long keys, preferably 1000 bits or longer, and should also change keys regularly. Top-level certifying authorities are exceptions: it may not be practical for them to change keys frequently because the key may be written into software used by a large number of verifiers.

In another attack, Alice bribes Bob, who works for the certifying authority, to issue to her a certificate in the name of Fred. Now Alice can send messages signed in Fred's name and anyone receiving such a message will believe it authentic because a full and verifiable certificate chain will accompany the message. This attack can be hindered by requiring the cooperation of two (or more) employees to generate a certificate; the attacker now has to bribe two employees rather than one. For example, in some of today's CSUs, three employees must each insert a data key containing secret information in order to authorize the CSU to generate certificates. Unfortunately, there may be other ways to generate a forged certificate by bribing only one employee. If each certificate request is checked by only one employee, that one employee can be bribed and slip a false request into a stack of real certificate requests. Note that a corrupt employee cannot reveal the certifying authority's private key, as long as it is properly stored.

Another attack involves forging old documents. Alice tries to factor the modulus of the certifying authority. It takes her 15 years, but she finally succeeds, and she now has the old private key of the certifying authority. The key has long since expired, but she can forge a certificate dated 15 years ago attesting to a phony public key of some other person, say Bob; she can now forge a document with a signature of Bob dated 15 year ago, perhaps a will leaving everything to Alice. The underlying issue raised by this attack is how to authenticate a signed document dated many years ago; this issue is discussed in Question 3.17.

Note that these attacks on certifying authorities do not threaten the privacy of messages between users, as might result from an attack on a secret-key distribution center.

3.10 *What if the certifying authority's key is lost or compromised?*

If the certifying authority's key is lost or destroyed but not compromised, certificates signed with the old key are still valid, as long as the verifier knows to use the old public key to verify the certificate.

In some CSU designs, encrypted backup copies of the CA's private key are kept. A CA which loses its key can then restore it by loading the encrypted backup into the CSU, which can decrypt it using some unique information stored inside the CSU; the encrypted backup can only be decrypted using the CSU. If the CSU itself is destroyed, the manufacturer may be able to supply another with the same internal information, thus allowing recovery of the key.

A compromised CA key is a much more dangerous situation. An attacker who discovers a certifying authority's private key can issue phony certificates in the name of the certifying authority, which would enable undetectable forgeries; for this reason, all precautions must be taken to prevent compromise, including those outlined in Questions 3.8 and 3.9. If a compromise does occur, the CA must immediately cease issuing certificates under its old key and change to a new key. If it is suspected that some phony certificates were issued, all certificates should be recalled, and then reissued with a new CA key. These measures could be relaxed somewhat if certificates were registered with a digital time-stamping service (see Question 3.18). Note that compromise of a CA key does not invalidate users' keys, but only the certificates that authenticate them. Compromise of a top-level CA's key should be considered catastrophic, since the key may be built into applications that verify certificates.

3.11 *What are Certificate Revocation Lists (CRLs)?*

A Certificate Revocation List (CRL) is a list of public keys that have been revoked before their scheduled expiration date. There are several reasons why a key might need to be revoked and placed on a CRL. A key might have been compromised. A key might be used professionally by an individual for a company; for example, the official name associated with a key might be "Alice Avery, Vice

President, Argo Corp." If Alice were fired, her company would not want her to be able to sign messages with that key and therefore the company would place the key on the CRL.

When verifying a signature, one can check the relevant CRL to make sure the signer's key has not been revoked. Whether it is worth the time to perform this check depends on the importance of the signed document.

CRLs are maintained by certifying authorities (CAs) and provide information about revoked keys originally certified by the CA. CRLs only list current keys, since expired keys should not be accepted in any case; when a revoked key is past its original expiration date it is removed from the CRL. Although CRLs are maintained in a distributed manner, there may be central repositories for CRLs, that is, sites on networks containing the latest CRLs from many organizations. An institution like a bank might want an in-house CRL repository to make CRL searches feasible on every transaction.

3.12 *What happens when a key expires?*

In order to guard against a long-term factoring attack, every key must have an expiration date after which it is no longer valid. The time to expiration must therefore be much shorter than the expected factoring time, or equivalently, the key length must be long enough to make the chances of factoring before expiration extremely small. The validity period for a key pair may also depend on the circumstances in which the key will be used, although there will also be a standard period. The validity period, together with the value of the key and the estimated strength of an expected attacker, then determines the appropriate key size.

The expiration date of a key accompanies the public key in a certificate or a directory listing. The signature verification program should check for expiration and should not accept a message signed with an expired key. This means that when one's own key expires, everything signed with it will no longer be considered valid. Of course, there will be cases where it is important that a signed document be considered valid for a much longer period of time; Question 3.17 discusses ways to achieve this.

After expiration, the user chooses a new key, which should be longer than the old key, perhaps by several digits, to reflect both the performance increase of computer hardware and any recent improvements in factoring algorithms. Recommended key length schedules will likely be published. A user may recertify a key that has expired, if it is sufficiently long and has not been compromised. The certifying authority would then issue a new certificate for the same key, and all new signatures would point to the new certificate instead of the old. However, the fact that computer hardware continues to improve argues for replacing expired keys with new, longer keys every few years. Key replacement enables one to take advantage of the hardware improvements to increase the security of the cryptosystem. Faster hardware has the effect of increasing security, perhaps

vastly, but only if key lengths are increased regularly (see Question 4.5).

3.13 *What happens if I lose my private key?*

If your private key is lost or destroyed, but not compromised, you can no longer sign or decrypt messages, but anything previously signed with the lost key is still valid. This can happen, for example, if you forget the password used to access your key, or if the disk on which the key is stored is damaged. You need to choose a new key right away, to minimize the number of messages people send you encrypted under your old key, messages which you can no longer read.

3.14 *What happens if my private key is compromised?*

If your private key is compromised, that is, if you suspect an attacker may have obtained your private key, then you must assume that some enemy can read encrypted messages sent to you and forge your name on documents. The seriousness of these consequences underscores the importance of protecting your private key with extremely strong mechanisms (see Question 3.15).

You must immediately notify your certifying authority and have your old key placed on a Certificate Revocation List (see Question 3.11); this will inform people that the key has been revoked. Then choose a new key and obtain the proper certificates for it. You may wish to use the new key to re-sign documents that you had signed with the compromised key; documents that had been time-stamped as well as signed might still be valid. You should also change the way you store your private key, to prevent compromise of the new key.

3.15 *How should I store my private key?*

Private keys must be stored securely, since forgery and loss of privacy could result from compromise. The private key should never be stored anywhere in plaintext form. The simplest storage mechanism is to encrypt the private key under a password and store the result on a disk. Of course, the password itself must be maintained with high security, not written down and not easily guessed. Storing the encrypted key on a disk that is not accessible through a computer network, such as a floppy disk or a local hard disk, will make some attacks more difficult. Ultimately, private keys may be stored on portable hardware, such as a smart card. Furthermore, a challenge-response protocol will be more secure than simple password access. Users with extremely high security needs, such as certifying authorities, should use special hardware devices to protect their keys (see Question 3.8).

3.16 *How do I find someone else's public key?*

Suppose you want to find Bob's public key. There are several possible ways. You could call him up and ask him to send you his public key via e-mail; you could request it via e-mail as well. Certifying authorities may provide directory services; if Bob works for company Z, look in the directory kept by Z's certifying authority. Directories must be secure against unauthorized tampering, so that users can be confident that a public key listed in the directory actually belongs to the person listed. Otherwise, you might send private encrypted information to the wrong person.

Eventually, full-fledged directories will arise, serving as online white or yellow pages. If they are compliant with CCITT X.509 standards [19], the directories will contain certificates as well as public keys; the presence of certificates will lower the directories' security needs.

3.17 *How can signatures remain valid beyond the expiration dates of their keys, or, How do you verify a 20-year-old signature?*

Normally, a key expires after, say, two years and a document signed with an expired key should not be accepted. However, there are many cases where it is necessary for signed documents to be regarded as legally valid for much longer than two years; long-term leases and contracts are examples. How should these cases be handled? Many solutions have been suggested but it is unclear which will prove the best. Here are some possibilities.

One can have special long-term keys as well as the normal two-year keys. Long-term keys should have much longer modulus lengths and be stored more securely than two-year keys. If a long-term key expires in 50 years, any document signed with it would remain valid within that time. A problem with this method is that any compromised key must remain on the relevant CRL until expiration (see Question 3.11); if 50-year keys are routinely placed on CRLs, the CRLs could grow in size to unmanageable proportions. This idea can be modified as follows. Register the long-term key by the normal procedure, i.e., for two years. At expiration time, if it has not been compromised, the key can be recertified, that is, issued a new certificate by the certifying authority, so that the key will be valid for another two years. Now a compromised key only needs to be kept on a CRL for at most two years, not fifty.

One problem with the previous method is that someone might try to invalidate a long-term contract by refusing to renew his key. This problem can be circumvented by registering the contract with a digital time-stamping service (see Question 3.18) at the time it is originally signed. If all parties to the contract keep a copy of the time-stamp, then each can prove that the contract was signed with valid keys. In fact, the time-stamp can prove the validity of a contract even if one signer's key gets compromised at some point after the contract was signed. This time-stamping solution can work with all signed digital

documents, not just multi-party contracts.

3.18 *What is a digital time-stamping service?*

A digital time-stamping service (DTS) issues time-stamps which associate a date and time with a digital document in a cryptographically strong way. The digital time-stamp can be used at a later date to prove that an electronic document existed at the time stated on its time-stamp. For example, a physicist who has a brilliant idea can write about it with a word processor and have the document time-stamped. The time-stamp and document together can later prove that the scientist deserves the Nobel Prize, even though an arch rival may have been the first to publish.

Here's one way such a system could work. Suppose Alice signs a document and wants it time-stamped. She computes a message digest of the document using a secure hash function (see Question 8.2) and then sends the message digest (but not the document itself) to the DTS, which sends her in return a digital time-stamp consisting of the message digest, the date and time it was received at the DTS, and the signature of the DTS. Since the message digest does not reveal any information about the content of the document, the DTS cannot eavesdrop on the documents it time-stamps. Later, Alice can present the document and time-stamp together to prove when the document was written. A verifier computes the message digest of the document, makes sure it matches the digest in the time-stamp, and then verifies the signature of the DTS on the time-stamp.

To be reliable, the time-stamps must not be forgeable. Consider the requirements for a DTS of the type just described. First, the DTS itself must have a long key if we want the time-stamps to be reliable for, say, several decades. Second, the private key of the DTS must be stored with utmost security, as in a tamperproof box. Third, the date and time must come from a clock, also inside the tamperproof box, which cannot be reset and which will keep accurate time for years or perhaps for decades. Fourth, it must be infeasible to create time-stamps without using the apparatus in the tamperproof box.

A cryptographically strong DTS using only software [4] has been implemented by Bellcore; it avoids many of the requirements just described, such as tamperproof hardware. The Bellcore DTS essentially combines hash values of documents into data structures called binary trees, whose "root" values are periodically published in the newspaper. A time-stamp consists of a set of hash values which allow a verifier to recompute the root of the tree. Since the hash functions are one-way (see Question 8.2), the set of validating hash values cannot be forged. The time associated with the document by the time-stamp is the date of publication.

The use of a DTS would appear to be extremely important, if not essential, for maintaining the validity of documents over many years (see Question 3.17). Suppose a landlord and tenant sign a twenty-year lease. The public keys used to

sign the lease will expire after, say, two years; solutions such as recertifying the keys or resigning every two years with new keys require the cooperation of both parties several years after the original signing. If one party becomes dissatisfied with the lease, he or she may refuse to cooperate. The solution is to register the lease with the DTS at the time of the original signing; both parties would then receive a copy of the time-stamp, which can be used years later to enforce the integrity of the original lease.

In the future, it is likely that a DTS will be used for everything from long-term corporate contracts to personal diaries and letters. Today, if an historian discovers some lost letters of Mark Twain, their authenticity is checked by physical means. But a similar find 100 years from now may consist of an author's computer files; digital time-stamps may be the only way to authenticate the find.

4 Factoring and Discrete Log

4.1 *What is a one-way function?*

A one-way function is a mathematical function that is significantly easier to perform in one direction (the forward direction) than in the opposite direction (the inverse direction). One might, for example, compute the function in minutes but only be able to compute the inverse in months or years. A trap-door one-way function is a one-way function where the inverse direction is easy if you know a certain piece of information (the trap door), but difficult otherwise.

4.2 *What is the significance of one-way functions for cryptography?*

Public-key cryptosystems are based on (presumed) trap-door one-way functions. The public key gives information about the particular instance of the function; the private key gives information about the trap door. Whoever knows the trap door can perform the function easily in both directions, but anyone lacking the trap door can perform the function only in the forward direction. The forward direction is used for encryption and signature verification; the inverse direction is used for decryption and signature generation.

In almost all public-key systems, the size of the key corresponds to the size of the inputs to the one-way function; the larger the key, the greater the difference between the efforts necessary to compute the function in the forward and inverse directions (for someone lacking the trap door). For a digital signature to be secure for years, for example, it is necessary to use a trap-door one-way function with inputs large enough that someone without the trap door would need many years to compute the inverse function.

All practical public-key cryptosystems are based on functions that are believed to be one-way, but have not been proven to be so. This means that it is theoretically possible that an algorithm will be discovered that can compute the inverse function easily without a trap door; this development would render any cryptosystem based on that one-way function insecure and useless.

4.3 *What is the factoring problem?*

Factoring is the act of splitting an integer into a set of smaller integers (factors) which, when multiplied together, form the original integer. For example, the factors of 15 are 3 and 5; the factoring problem is to find 3 and 5 when given 15. Prime factorization requires splitting an integer into factors that are prime numbers; every integer has a unique prime factorization. Multiplying two prime integers together is easy, but as far as we know, factoring the product is much more difficult.

4.4 *What is the significance of factoring in cryptography?*

Factoring is the underlying, presumably hard problem upon which several public-key cryptosystems are based, including RSA. Factoring an RSA modulus (see Question 2.1) would allow an attacker to figure out the private key; thus, anyone who can factor the modulus can decrypt messages and forge signatures. The security of RSA therefore depends on the factoring problem being difficult. Unfortunately, it has not been proven that factoring must be difficult, and there remains a possibility that a quick and easy factoring method might be discovered (see Question 4.7), although factoring researchers consider this possibility remote.

Factoring large numbers takes more time than factoring smaller numbers. This is why the size of the modulus in RSA determines how secure an actual use of RSA is; the larger the modulus, the longer it would take an attacker to factor, and thus the more resistant to attack the RSA implementation is.

4.5 *Has factoring been getting easier?*

Factoring has become easier over the last fifteen years for two reasons: computer hardware has become more powerful, and better factoring algorithms have been developed.

Hardware improvement will continue inexorably, but it is important to realize that *hardware improvements make RSA more secure, not less.* This is because a hardware improvement that allows an attacker to factor a number two digits longer than before will at the same time allow a legitimate RSA user to use a key dozens of digits longer than before; a user can choose a new key a dozen digits longer than the old one without any performance slowdown, yet a factoring attack will become much more difficult. Thus although the hardware

improvement does help the attacker, it helps the legitimate user much more. This general rule may fail in the sense that factoring may take place using fast machines of the future, attacking RSA keys of the past; in this scenario, only the attacker gets the advantage of the hardware improvement. This consideration argues for using a larger key size today than one might otherwise consider warranted. It also argues for replacing one's RSA key with a longer key every few years, in order to take advantage of the extra security offered by hardware improvements. This point holds for other public-key systems as well.

Better factoring algorithms have been more help to the RSA attacker than have hardware improvements. As the RSA system, and cryptography in general, have attracted much attention, so has the factoring problem, and many researchers have found new factoring methods or improved upon others. This has made factoring easier, for numbers of any size and irrespective of the speed of the hardware. However, factoring is still a very difficult problem.

Overall, any recent decrease in security due to algorithm improvement can be offset by increasing the key size. In fact, between general computer hardware improvements and special-purpose RSA hardware improvements, increases in key size (maintaining a constant speed of RSA operations) have kept pace or exceeded increases in algorithm efficiency, resulting in no net loss of security. As long as hardware continues to improve at a faster rate than that at which the complexity of factoring algorithms decreases, the security of RSA will increase, assuming RSA users regularly increase their key size by appropriate amounts. The open question is how much faster factoring algorithms can get; there must be some intrinsic limit to factoring speed, but this limit remains unknown.

4.6 *What are the best factoring methods in use today?*

Factoring is a very active field of research among mathematicians and computer scientists; the best factoring algorithms are mentioned below with some references and their big-O asymptotic efficiency. O notation measures how fast an algorithm is; it gives an upper bound on the number of operations (to order of magnitude) in terms of n, the number to be factored, and p, a prime factor of n. For textbook treatment of factoring algorithms, see [41], [42], [47], and [11]; for a detailed explanation of big-O notation, see [22].

Factoring algorithms come in two flavors, special purpose and general purpose; the efficiency of the former depends on the unknown factors, whereas the efficiency of the latter depends on the number to be factored. Special purpose algorithms are best for factoring numbers with small factors, but the numbers used for the modulus in the RSA system do not have any small factors. Therefore, general purpose factoring algorithms are the more important ones in the context of cryptographic systems and their security.

Special purpose factoring algorithms include the Pollard rho method [66], with expected running time $O(\sqrt{p})$, and the Pollard $p-1$ method [67], with running time $O(p')$, where p' is the largest prime factor of $p-1$. Both of these

take an amount of time that is exponential in the size of p, the prime factor that they find; thus these algorithms are too slow for most factoring jobs. The elliptic curve method (ECM) [50] is superior to these; its asymptotic running time is $O(\exp(\sqrt{2 \ln p \ln \ln p}))$. The ECM is often used in practice to find factors of randomly generated numbers; it is not strong enough to factor a large RSA modulus.

The best general purpose factoring algorithm today is the number field sieve [16], which runs in time approximately $O(\exp(1.9(\ln n)^{1/3}(\ln \ln n)^{2/3}))$. It has only recently been implemented [15], and is not yet practical enough to perform the most desired factorizations. Instead, the most widely used general purpose algorithm is the multiple polynomial quadratic sieve (mpqs) [77], which has running time $O(\exp(\sqrt{\ln n \ln \ln n}))$. The mpqs (and some of its variations) is the only general purpose algorithm that has successfully factored numbers greater than 110 digits; a variation known as ppmpqs [49] has been particularly popular.

It is expected that within a few years the number field sieve will overtake the mpqs as the most widely used factoring algorithm, as the size of the numbers being factored increases from about 120 digits, which is the current threshold of general numbers which can be factored, to 130 or 140 digits. A "general number" is one with no special form that might make it easier to factor; an RSA modulus is a general number. Note that a 512-bit number has about 155 digits.

Numbers that have a special form can already be factored up to 155 digits or more [48]. The Cunningham Project [14] keeps track of the factorizations of numbers with these special forms and maintains a "10 Most Wanted" list of desired factorizations. Also, a good way to survey current factoring capability is to look at recent results of the RSA Factoring Challenge (see Question 4.8).

4.7 *What are the prospects for theoretical factoring breakthroughs?*

Although factoring is strongly believed to be a difficult mathematical problem, it has not been proved so. Therefore there remains a possibility that an easy factoring algorithm will be discovered. This development, which could seriously weaken RSA, would be highly surprising and the possibility is considered extremely remote by the researchers most actively engaged in factoring research.

Another possibility is that someone will prove that factoring is difficult. This negative breakthrough is probably more likely than the positive breakthrough discussed above, but would also be unexpected at the current state of theoretical factoring research. This development would guarantee the security of RSA beyond a certain key size.

4.8 *What is the RSA Factoring Challenge?*

RSA Data Security Inc. (RSADSI) administers a factoring contest with quarterly cash prizes. Those who factor numbers listed by RSADSI earn points

toward the prizes; factoring smaller numbers earns more points than factoring larger numbers. Results of the contest may be useful to those who wish to know the state of the art in factoring; the results show the size of numbers factored, which algorithms are used, and how much time was required to factor each number. Send e-mail to `challenge-info@rsa.com` for information.

4.9 *What is the discrete log problem?*

The discrete log problem, in its most common formulation, is to find the exponent x in the formula $y = g^x \bmod p$; in other words, it seeks to answer the question, To what power must g be raised in order to obtain y, modulo the prime number p? There are other, more general, formulations as well.

Like the factoring problem, the discrete log problem is believed to be difficult and also to be the hard direction of a one-way function. For this reason, it has been the basis of several public-key cryptosystems, including the ElGamal system and DSS (see Questions 2.15 and 6.8). The discrete log problem bears the same relation to these systems as factoring does to RSA: the security of these systems rests on the assumption that discrete logs are difficult to compute.

The discrete log problem has received much attention in recent years; descriptions of some of the most efficient algorithms can be found in [47], [21], and [33]. The best discrete log problems have expected running times similar to that of the best factoring algorithms. Rivest [72] has analyzed the expected time to solve discrete log both in terms of computing power and money.

4.10 *Which is easier, factoring or discrete log?*

The asymptotic running time of the best discrete log algorithm is approximately the same as for the best general purpose factoring algorithm. Therefore, it requires about as much effort to solve the discrete log problem modulo a 512-bit prime as to factor a 512-bit RSA modulus. One paper [45] cites experimental evidence that the discrete log problem is slightly harder than factoring: the authors suggest that the effort necessary to factor a 110-digit integer is the same as the effort to solve discrete logarithms modulo a 100-digit prime. This difference is so slight that it should not be a significant consideration when choosing a cryptosystem.

Historically, it has been the case that an algorithmic advance in either problem, factoring or discrete logs, was then applied to the other. This suggests that the degrees of difficulty of both problems are closely linked, and that any breakthrough, either positive or negative, will affect both problems equally.

5 DES

5.1 *What is DES?*

DES is the Data Encryption Standard, an encryption block cipher defined and endorsed by the U.S. government in 1977 as an official standard; the details can be found in the official FIPS publication [59]. It was originally developed at IBM. DES has been extensively studied over the last 15 years and is the most well-known and widely used cryptosystem in the world.

DES is a secret-key, symmetric cryptosystem: when used for communication, both sender and receiver must know the same secret key, which is used both to encrypt and decrypt the message. DES can also be used for single-user encryption, such as to store files on a hard disk in encrypted form. In a multi-user environment, secure key distribution may be difficult; public-key cryptography was invented to solve this problem (see Question 1.3). DES operates on 64-bit blocks with a 56-bit key. It was designed to be implemented in hardware, and its operation is relatively fast. It works well for bulk encryption, that is, for encrypting a large set of data.

NIST (see Question 7.1) has recertified DES as an official U.S. government encryption standard every five years; DES was last recertified in 1993, by default. NIST has indicated, however, that it may not recertify DES again.

5.2 *Has DES been broken?*

DES has never been "broken", despite the efforts of many researchers over many years. The obvious method of attack is brute-force exhaustive search of the key space; this takes 2^{55} steps on average. Early on it was suggested [28] that a rich and powerful enemy could build a special-purpose computer capable of breaking DES by exhaustive search in a reasonable amount of time. Later, Hellman [36] showed a time-memory trade-off that allows improvement over exhaustive search if memory space is plentiful, after an exhaustive precomputation. These ideas fostered doubts about the security of DES. There were also accusations that the NSA had intentionally weakened DES. Despite these suspicions, no feasible way to break DES faster than exhaustive search was discovered. The cost of a specialized computer to perform exhaustive search has been estimated by Wiener at one million dollars [80].

Just recently, however, the first attack on DES that is better than exhaustive search was announced by Eli Biham and Adi Shamir [6, 7], using a new technique known as differential cryptanalysis. This attack requires encryption of 2^{47} chosen plaintexts, i.e., plaintexts chosen by the attacker. Although a theoretical breakthrough, this attack is not practical under normal circumstances because it requires the attacker to have easy access to the DES device in order to encrypt the chosen plaintexts. Another attack, known as linear cryptanalysis [51], does not require chosen plaintexts.

The consensus is that DES, when used properly, is secure against all but the most powerful enemies. In fact, triple encryption DES (see Question 5.3) may be secure against anyone at all. Biham and Shamir have stated that they consider DES secure. It is used extensively in a wide variety of cryptographic systems, and in fact, most implementations of public-key cryptography include DES at some level.

5.3 *How does one use DES securely?*

When using DES, there are several practical considerations that can affect the security of the encrypted data. One should change DES keys frequently, in order to prevent attacks that require sustained data analysis. In a communications context, one must also find a secure way of communicating the DES key to both sender and receiver. Use of RSA or some other public-key technique for key management solves both these issues: a different DES key is generated for each session, and secure key management is provided by encrypting the DES key with the receiver's RSA public key. RSA, in this circumstance, can be regarded as a tool for improving the security of DES (or any other secret key cipher).

If one wishes to use DES to encrypt files stored on a hard disk, it is not feasible to frequently change the DES keys, as this would entail decrypting and then re-encrypting all files upon each key change. Instead, one should have a master DES key with which one encrypts the list of DES keys used to encrypt the files; one can then change the master key frequently without much effort.

A powerful technique for improving the security of DES is triple encryption, that is, encrypting each message block under three different DES keys in succession. Triple encryption is thought to be equivalent to doubling the key size of DES, to 112 bits, and should prevent decryption by an enemy capable of single-key exhaustive search [53]. Of course, using triple-encryption takes three times as long as single-encryption DES.

Aside from the issues mentioned above, DES can be used for encryption in several officially defined modes. Some are more secure than others. ECB (electronic codebook) mode simply encrypts each 64-bit block of plaintext one after another under the same 56-bit DES key. In CBC (cipher block chaining) mode, each 64-bit plaintext block is XORed with the previous ciphertext block before being encrypted with the DES key. Thus the encryption of each block depends on previous blocks and the same 64-bit plaintext block can encrypt to different ciphertext depending on its context in the overall message. CBC mode helps protect against certain attacks, although not against exhaustive search or differential cryptanalysis. CFB (cipher feedback) mode allows one to use DES with block lengths less than 64 bits. Detailed descriptions of the various DES modes can be found in [60].

In practice, CBC is the most widely used mode of DES, and is specified in several standards. For additional security, one could use triple encryption with CBC, but since single DES in CBC mode is usually considered secure enough,

triple encryption is not often used.

5.4 *Can DES be exported from the U.S.?*

Export of DES, either in hardware or software, is strictly regulated by the U.S. State Department and the NSA (see Question 1.6). The government rarely approves export of DES, despite the fact that DES is widely available overseas; financial institutions and foreign subsidiaries of U.S. companies are exceptions.

5.5 *What are the alternatives to DES?*

Over the years, various bulk encryption algorithms have been designed as alternatives to DES. One is FEAL (Fast Encryption ALgorithm), a cipher for which attacks have been discovered [6], although new versions have been proposed. Another recently proposed cipher designed by Lai and Massey [44] and known as IDEA seems promising, although it has not yet received sufficient scrutiny to instill full confidence in its security. The U.S. government recently announced a new algorithm called Skipjack (see Question 6.5) as part of its Capstone project. Skipjack operates on 64-bit blocks of data, as does DES, but uses 80-bit keys, as opposed to 56-bit keys in DES. However, the details of Skipjack are classified, so Skipjack is only available in hardware from government-authorized manufacturers.

Rivest has developed the ciphers RC2 and RC4 (see Question 8.6), which can be made as secure as necessary because they use variable key sizes. Faster than DES, at least in software, they have the further advantage of special U.S. government status whereby the export approval is simplified and expedited if the key size is limited to 40 bits.

5.6 *Is DES a group?*

It has been frequently asked whether DES encryption is closed under composition; i.e., is encrypting a plaintext under one DES key and then encrypting the result under another key always equivalent to a single encryption under a single key? Algebraically, is DES a group? If so, then DES might be weaker than would otherwise be the case; see [39] for a more complete discussion. However, the answer is no, DES is not a group [18]; this issue was settled only recently, after many years of speculation and circumstantial evidence. This result seems to imply that techniques such as triple encryption do in fact increase the security of DES.

6 Capstone, Clipper, and DSS

6.1 *What is Capstone?*

Capstone is the U.S. government's long-term project to develop a set of standards for publicly-available cryptography, as authorized by the Computer Security Act of 1987. The primary agencies responsible for Capstone are NIST and the NSA (see Section 7). The plan calls for the elements of Capstone to become official U.S. government standards, in which case both the government itself and all private companies doing business with the government would be required to use Capstone.

There are four major components of Capstone: a bulk data encryption algorithm, a digital signature algorithm, a key exchange protocol, and a hash function. The data encryption algorithm is called Skipjack (see Question 6.5), but is often referred to as Clipper, which is the encryption chip that includes Skipjack (see Question 6.2). The digital signature algorithm is DSS (see Question 6.8) and the hash function is SHS (see Question 8.4 about SHS and Question 8.2 about hash functions). The key exchange protocol has not yet been announced.

All the parts of Capstone have 80-bit security: all the keys involved are 80 bits long and other aspects are also designed to withstand anything less than an "80-bit" attack, that is, an effort of 2^{80} operations. Eventually the government plans to place the entire Capstone cryptographic system on a single chip.

6.2 *What is Clipper?*

Clipper is an encryption chip developed and sponsored by the U.S. government as part of the Capstone project (see Question 6.1). Announced by the White House in April, 1993 [65], Clipper was designed to balance the competing concerns of federal law-enforcement agencies with those of private citizens and industry. The law-enforcement agencies wish to have access to the communications of suspected criminals, for example by wire-tapping; these needs are threatened by secure cryptography. Industry and individual citizens, however, want secure communications, and look to cryptography to provide it.

Clipper technology attempts to balance these needs by using escrowed keys. The idea is that communications would be encrypted with a secure algorithm, but the keys would be kept by one or more third parties (the "escrow agencies"), and made available to law-enforcement agencies when authorized by a court-issued warrant. Thus, for example, personal communications would be impervious to recreational eavesdroppers, and commercial communications would be impervious to industrial espionage, and yet the FBI could listen in on suspected terrorists or gangsters.

Clipper has been proposed as a U.S. government standard [62]; it would then be used by anyone doing business with the federal government as well as for communications within the government. For anyone else, use of Clipper

is strictly voluntary. AT&T has announced a secure telephone that uses the Clipper chip.

6.3 *How does the Clipper chip work?*

The Clipper chip contains an encryption algorithm called Skipjack (see Question 6.5), whose details have not been made public. Each chip also contains a unique 80-bit unit key U, which is escrowed in two parts at two escrow agencies; both parts must be known in order to recover the key. Also present is a serial number and an 80-bit "family key" F; the latter is common to all Clipper chips. The chip is manufactured so that it cannot be reverse engineered; this means that the Skipjack algorithm and the keys cannot be read off the chip.

When two devices wish to communicate, they first agree on an 80-bit "session key" K. The method by which they choose this key is left up to the implementer's discretion; a public-key method such as RSA or Diffie-Hellman seems a likely choice. The message is encrypted with the key K and sent; note that the key K is not escrowed. In addition to the encrypted message, another piece of data, called the law-enforcement access field (LEAF), is created and sent. It includes the session key K encrypted with the unit key U, then concatenated with the serial number of the sender and an authentication string, and then, finally, all encrypted with the family key. The exact details of the law-enforcement field are classified.

The receiver decrypts the law-enforcement field, checks the authentication string, and decrypts the message with the key K.

Now suppose a law-enforcement agency wishes to tap the line. It uses the family key to decrypt the law-enforcement field; the agency now knows the serial number and has an encrypted version of the session key. It presents an authorization warrant to the two escrow agencies along with the serial number. The escrow agencies give the two parts of the unit key to the law-enforcement agency, which then decrypts to obtain the session key K. Now the agency can use K to decrypt the actual message.

Further details on the Clipper chip operation, such as the generation of the unit key, are sketched by Denning [26].

6.4 *Who are the escrow agencies?*

It has not yet been decided which organizations will serve as the escrow agencies, that is, keep the Clipper chip keys. No law-enforcement agency will be an escrow agency, and it is possible that at least one of the escrow agencies will be an organization outside the government.

It is essential that the escrow agencies keep the key databases extremely secure, since unauthorized access to both escrow databases could allow unauthorized eavesdropping on private communications. In fact, the escrow agencies

are likely to be one of the major targets for anyone trying to compromise the Clipper system; the Clipper chip factory is another likely target.

6.5 *What is Skipjack?*

Skipjack is the encryption algorithm contained in the Clipper chip; it was designed by the NSA. It uses an 80-bit key to encrypt 64-bit blocks of data; the same key is used for the decryption. Skipjack can be used in the same modes as DES (see Question 5.3), and may be more secure than DES, since it uses 80-bit keys and scrambles the data for 32 steps, or "rounds"; by contrast, DES uses 56-bit keys and scrambles the data for only 16 rounds.

The details of Skipjack are classified. The decision not to make the details of the algorithm publicly available has been widely criticized. Many people are suspicious that Skipjack is not secure, either due to oversight by its designers, or by the deliberate introduction of a secret trapdoor. By contrast, there have been many attempts to find weaknesses in DES over the years, since its details are public. These numerous attempts (and the fact that they have failed) have made people confident in the security of DES. Since Skipjack is not public, the same scrutiny cannot be applied towards it, and thus a corresponding level of confidence may not arise.

Aware of such criticism, the government invited a small group of independent cryptographers to examine the Skipjack algorithm. They issued a report [12] which stated that, although their study was too limited to reach a definitive conclusion, they nevertheless believe that Skipjack is secure.

Another consequence of Skipjack's classified status is that it cannot be implemented in software, but only in hardware by government-authorized chip manufacturers.

6.6 *Why is Clipper controversial?*

The Clipper chip proposal has aroused much controversy and has been the subject of much criticism. Unfortunately two distinct issues have become confused in the large volume of public comment and discussion.

First there is controversy about the whole idea of escrowed keys. Those in favor of escrowed keys see it as a way to provide secure communications for the public at large while allowing law-enforcement agencies to monitor the communications of suspected criminals. Those opposed to escrowed keys see it as an unnecessary and ineffective intrusion of the government into the private lives of citizens. They argue that escrowed keys infringe their rights of privacy and free speech. It will take a lot of time and much public discussion for society to reach a consensus on what role, if any, escrowed keys should have.

The second area of controversy concerns various objections to the specific Clipper proposal, that is, objections to this particular implementation of escrowed keys, as opposed to the idea of escrowed keys in general. Common

objections include: the Skipjack algorithm is not public (see Questions 6.5) and may not be secure; the key escrow agencies will be vulnerable to attack; there are not enough key escrow agencies; the keys on the Clipper chips are not generated in a sufficiently secure fashion; there will not be sufficient competition among implementers, resulting in expensive and slow chips; software implementations are not possible; and the key size is fixed and cannot be increased if necessary.

Micali [55] has recently proposed an alternative system that also attempts to balance the privacy concerns of law-abiding citizens with the investigative concerns of law-enforcement agencies. Called fair public-key cryptography, it is similar in function and purpose to the Clipper chip proposal but users can choose their own keys, which they register with the escrow agencies. Also, the system does not require secure hardware, and can be implemented completely in software.

6.7 *What is the current status of Clipper?*

Clipper is under review. Both the executive branch and Congress are considering it, and an advisory panel recently recommended a full year-long public discussion of cryptography policy. NIST has invited the public to send comments, as part of its own review.

6.8 *What is DSS?*

DSS is the proposed Digital Signature Standard, which specifies a Digital Signature Algorithm (DSA), and is a part of the U.S. government's Capstone project (see Question 6.1). It was selected by NIST, in cooperation with the NSA (see Section 7), to be the digital authentication standard of the U.S. government; whether the government should in fact adopt it as the official standard is still under debate.

DSS is based on the discrete log problem (see Question 4.9) and derives from cryptosystems proposed by Schnorr [75] and ElGamal [30]. It is for authentication only. For a detailed description of DSS, see [63] or [57].

DSS has, for the most part, been looked upon unfavorably by the computer industry, much of which had hoped the government would choose the RSA algorithm as the official standard; RSA is the most widely used authentication algorithm. Several articles in the press, such as [54], discuss the industry dissatisfaction with DSS. Criticism of DSS has focused on a few main issues: it lacks key exchange capability; the underlying cryptosystem is too recent and has been subject to too little scrutiny for users to be confident of its strength; verification of signatures with DSS is too slow; the existence of a second authentication standard will cause hardship to computer hardware and software vendors, who have already standardized on RSA; and that the process by which NIST chose DSS was too secretive and arbitrary, with too much influence wielded by NSA. Other criticisms were addressed by NIST by modifying the original proposal.

A more detailed discussion of the various criticisms can be found in [57], and a detailed response by NIST can be found in [78].

In the DSS system, signature generation is faster than signature verification, whereas in the RSA system, signature verification is faster than signature generation (if the public and private exponents are chosen for this property, which is the usual case). NIST claims that it is an advantage of DSS that signing is faster, but many people in cryptography think that it is better for verification to be the faster operation.

6.9 *Is DSS secure?*

The most serious criticisms of DSS involve its security. DSS was originally proposed with a fixed 512-bit key size. After much criticism that this is not secure enough, NIST revised DSS to allow key sizes up to 1024 bits. More critical, however, is the fact that DSS has not been around long enough to withstand repeated attempts to break it; although the discrete log problem is old, the particular form of the problem used in DSS was first proposed for cryptographic use in 1989 by Schnorr [75] and has not received much public study. In general, any new cryptosystem could have serious flaws that are only discovered after years of scrutiny by cryptographers. Indeed this has happened many times in the past; see [13] for some detailed examples. RSA has withstood over 15 years of vigorous examination for weaknesses. In the absence of mathematical proofs of security, nothing builds confidence in a cryptosystem like sustained attempts to crack it. Although DSS may well turn out to be a strong cryptosystem, its relatively short history will leave doubts for years to come.

Some researchers warned about the existence of "trapdoor" primes in DSS, which could enable a key to be easily broken. These trapdoor primes are relatively rare however, and are easily avoided if proper key generation procedures are followed [78].

6.10 *Is use of DSS covered by any patents?*

NIST has filed a patent application for DSS and there have been claims that DSS is covered by other public-key patents. NIST recently announced its intention to grant exclusive sublicensing rights for the DSS patent to Public Key Partners (PKP), which also holds the sublicensing rights to other patents that may cover DSS (see Question 1.5). In the agreement between NIST and PKP, PKP publicly stated uniform guidelines by which it will grant licenses to practice DSS. PKP stated that DSS can be used on a royalty-free basis in the case of personal, noncommercial, or U.S. government use. See [61] for details on the agreement and the licensing policy.

6.11 *What is the current status of DSS?*

After NIST issued the DSS proposal in August 1991, there was a period in which comments from the public were solicited; NIST then revised its proposal in light of the comments. DSS may be issued as a FIPS and become the official U.S. government standard, but it is not clear when this might happen. DSS is currently in the process of becoming a standard, along with RSA, for the financial services industry; a recent draft standard [1] contains the revised version of DSS.

7 NIST and NSA

7.1 *What is NIST?*

NIST is an acronym for the National Institute of Standards and Technology, a division of the U.S. Department of Commerce; it was formerly known as the National Bureau of Standards (NBS). Through its Computer Systems Laboratory it aims to promote open systems and interoperability that will spur development of computer-based economic activity. NIST issues standards and guidelines that it hopes will be adopted by all computer systems in the U.S., and also sponsors workshops and seminars. Official standards are published as FIPS (Federal Information Processing Standards) publications.

In 1987 Congress passed the Computer Security Act, which authorized NIST to develop standards for ensuring the security of sensitive but unclassified information in government computer systems. It encouraged NIST to work with other government agencies and private industry in evaluating proposed computer security standards.

7.2 *What role does NIST play in cryptography?*

NIST issues standards for cryptographic routines; U.S. government agencies are required to use them, and the private sector often adopts them as well. In January 1977, NIST declared DES (see Question 5.1) the official U.S. encryption standard and published it as FIPS Publication 46; DES soon became a de facto standard throughout the U.S.

A few years ago, NIST was asked to choose a set of cryptographic standards for the U.S.; this has become known as the Capstone project (see Section 6). After a few years of rather secretive deliberations, and in cooperation with the NSA, NIST issued proposals for various standards in cryptography, including digital signatures (DSS) and data encryption (the Clipper chip); these are pieces of the overall Capstone project.

NIST has been criticized for allowing the NSA too much power in setting cryptographic standards, since the interests of the NSA conflict with that of the Commerce Department and NIST. Yet, the NSA has much more experience

with cryptography, and many more qualified cryptographers and cryptanalysts, than does NIST; it would be unrealistic to expect NIST to forego such available assistance.

7.3 *What is the NSA?*

The NSA is the National Security Agency, a highly secretive agency of the U.S. government that was created by Harry Truman in 1952; its very existence was kept secret for many years. For a history of the NSA, see Bamford [2]. The NSA has a mandate to listen to and decode all foreign communications of interest to the security of the United States. It has also used its power in various ways (see Question 7.4) to slow the spread of publicly available cryptography, in order to prevent national enemies from employing encryption methods too strong for the NSA to break.

As the premier cryptographic government agency, the NSA has huge financial and computer resources and employs a host of cryptographers. Developments in cryptography achieved at the NSA are not made public; this secrecy has led to many rumors about the NSA's ability to break popular cryptosystems like DES and also to rumors that the NSA has secretly placed weaknesses, called trap doors, in government-endorsed cryptosystems, such as DES. These rumors have never been proved or disproved, and the criteria used by the NSA in selecting cryptography standards have never been made public.

Recent advances in the computer and telecommunications industries have placed NSA actions under unprecedented scrutiny, and the agency has become the target of heavy criticism for hindering U.S. industries that wish to use or sell strong cryptographic tools. The two main reasons for this increased criticism are the collapse of the Soviet Union and the development and spread of commercially available public-key cryptographic tools. Under pressure, the NSA may be forced to change its policies.

7.4 *What role does the NSA play in commercial cryptography?*

The NSA's charter limits its activities to foreign intelligence. However, the NSA is concerned with the development of commercial cryptography because the availability of strong encryption tools through commercial channels could impede the NSA's mission of decoding international communications; in other words, the NSA is worried lest strong commercial cryptography fall into the wrong hands.

The NSA has stated that it has no objection to the use of secure cryptography by U.S. industry. It also has no objection to cryptographic tools used for authentication, as opposed to privacy. However, the NSA is widely viewed as following policies that have the practical effect of limiting and/or weakening the cryptographic tools used by law-abiding U.S. citizens and corporations; see Barlow [3] for a discussion of NSA's effect on commercial cryptography.

The NSA exerts influence over commercial cryptography in several ways. First, it controls the export of cryptography from the U.S. (see Question 1.6); the NSA generally does not approve export of products used for encryption unless the key size is strictly limited. It does, however, approve for export any products used for authentication only, no matter how large the key size, so long as the product cannot be converted to be used for encryption. The NSA has also blocked encryption methods from being published or patented, citing a national security threat; see Landau [46] for a discussion of this practice. Additionally, the NSA serves an "advisory" role to NIST in the evaluation and selection of official U.S. government computer security standards; in this capacity, it has played a prominent, and controversial, role in the selection of DES and in the development of the group of standards known as the Capstone project (see Section 6), which includes DSS and the Clipper chip. The NSA can also exert market pressure on U.S. companies to produce (or refrain from producing) cryptographic goods, since the NSA itself is often a large customer of these companies.

Cryptography is in the public eye as never before and has become the subject of national public debate. The status of cryptography, and the NSA's role in it, will probably change over the next few years.

8 Miscellaneous

8.1 *What is the legal status of documents signed with digital signatures?*

If digital signatures are to replace handwritten signatures they must have the same legal status as handwritten signatures, i.e., documents signed with digital signatures must be legally binding. NIST has stated that its proposed Digital Signature Standard (see Question 6.8) should be capable of "proving to a third party that data was actually signed by the generator of the signature." Furthermore, U.S. federal government purchase orders will be signed by any such standard; this implies that the government will support the legal authority of digital signatures in the courts. Some preliminary legal research has also resulted in the opinion that digital signatures would meet the requirements of legally binding signatures for most purposes, including commercial use as defined in the Uniform Commercial Code (UCC). A GAO (Government Accounting Office) decision requested by NIST also opines that digital signatures will meet the legal standards of handwritten signatures [20].

However, since the validity of documents with digital signatures has never been challenged in court, their legal status is not yet well-defined. Through such challenges, the courts will issue rulings that collectively define which digital signature methods, key sizes, and security precautions are acceptable for a digital signature to be legally binding.

Digital signatures have the potential to possess greater legal authority than handwritten signatures. If a ten-page contract is signed by hand on the tenth page, one cannot be sure that the first nine pages have not been altered. If the contract was signed by digital signatures, however, a third party can verify that not one byte of the contract has been altered.

Currently, if two people wish to digitally sign a series of contracts, they may wish to first sign a paper contract in which they agree to be bound in the future by any contracts digitally signed by them with a given signature method and minimum key size.

8.2 *What is a hash function? What is a message digest?*

A hash function is a computation that takes a variable-size input and returns a fixed-size string, which is called the hash value. If the hash function is one-way, i.e., hard to invert, it is also called a message-digest function, and the result is called a message digest. The idea is that a digest represents concisely the longer message or document from which it was computed; one can think of a message digest as a "digital fingerprint" of the larger document. Examples of well-known hash functions are MD4, MD5, and SHS (see Questions 8.3 and 8.4).

Although hash functions in general have many uses in computer programs, in cryptography they are used to generate a small string (the message digest) that can represent securely a much larger string, such as a file or message. Since the hash functions are faster than the signing functions, it is much more efficient to compute a digital signature using a document's message digest, which is small, than using the arbitrarily large document itself. Additionally, a digest can be made public without revealing the contents of the document from which it derives. This is important in digital time-stamping, where, using hash functions, one can get a document time-stamped without revealing its contents to the time-stamping service (see Question 3.18).

A hash function used for digital authentication must have certain properties that make it secure enough for cryptographic use. Specifically, it must be infeasible to find a message that hashes to a given value and it must be infeasible to find two distinct messages that hash to the same value. The ability to find a message hashing to a given value would enable an attacker to substitute a fake message for a real message that was signed. It would also enable someone to falsely disown a message by claiming that he or she actually signed a different message hashing to the same value, thus violating the non-repudiation property of digital signatures. The ability to find two distinct messages hashing to the same value could enable an attack whereby someone is tricked into signing a message which hashes to the same value as another message with a quite different meaning. The digest must therefore be long enough to prevent an attacker from doing an exhaustive search for a collision. For example, if a hash function produces 100-bit strings, exhaustive search would take 2^{100} attempts on average to match a given value, and approximately 2^{50} attempts on average to find two

inputs producing the same digest.

A digital signature system can be broken by attacking either the difficult mathematical problem on which the signature method is based or the hash function used to create the message digests. When choosing an authentication system, it is generally a good idea to choose a signature method and a hash function that require comparable efforts to break; any extra security in one of the two components is wasted, since attacks will be directed at the weaker component. Actually, attacking the hash function is harder in practice, since it requires a large amount of memory and the ability to trick the victim into signing a special message. With 2^{64} operations, an attacker can find two messages that hash to the same digest under any of the MD hash functions; this effort is comparable to that necessary to break 512-bit RSA; thus MD5 is a good choice when using RSA with a 512-bit modulus. However, those with greater security needs, such as certifying authorities, should use a longer modulus and a hash function that produces a longer message digest; either SHS (160-bit digest) or a modified version of MD4 that produces a 256-bit digest [71] would suffice.

8.3 *What are MD2, MD4 and MD5?*

MD2, MD4 and MD5 (MD stands for Message Digest) are widely used hash functions designed by Ron Rivest specifically for cryptographic use. They produce 128-bit digests and there is no known attack faster than exhaustive search.

MD2 is the slowest of the three; MD4 [71] is the fastest. MD5 [73] has been dubbed "MD4 with safety belts" by Rivest, since it has a more conservative design than MD4; the design gives it increased security against attack, but at a cost of being approximately 33% slower than MD4. MD5 is the most commonly used of the three algorithms. MD4 and MD5 are publicly available for unrestricted use; MD2 is available for use with PEM (see Question 8.7). Details of MD2, MD4, and MD5 with sample C code are available in Internet RFCs (Requests For Comments) 1319, 1320, and 1321, respectively.

No feasible attacks on any of the MD algorithms have been discovered, although some recent theoretical work has found some interesting structural properties [24, 25].

8.4 *What is SHS?*

The Secure Hash Standard (SHS) [58] is a hash function proposed by NIST (see Question 7.1) and adopted as a U.S. government standard. It is designed for use with the proposed Digital Signature Standard (see Question 6.8) and is part of the government's Capstone project (see Question 6.1). SHS produces a 160-bit hash value from a variable-size input. SHS is structurally similar to MD4 and MD5. It is roughly 25% slower than MD5 but may be more secure, because it produces message digests that are 25% longer than those produced by the MD

functions. SHS is currently the only part of Capstone that has been officially adopted as a government standard.

8.5 *What is Kerberos?*

Kerberos is a secret-key network authentication system developed at MIT [79]; it uses DES for encryption and authentication. Unlike a public-key authentication system, it does not produce digital signatures: Kerberos was designed to authenticate requests for network resources rather than to authenticate authorship of documents. Kerberos provides real-time authentication in a distributed environment, but does not provide for future third-party verification of documents.

In a Kerberos system, there is a designated site on the network, called the Kerberos server, which performs centralized key management and administrative functions. The server maintains a database containing the secret keys of all users, generates session keys whenever two users wish to communicate securely, and authenticates the identity of a user who requests certain network services.

Kerberos, like other secret-key systems, requires trust in a third party, in this case the Kerberos server. If the server were compromised, the integrity of the whole system would fall. Public-key cryptography was designed precisely to avoid the necessity to trust third parties or communication lines (see Question 1.4). Kerberos may be adequate for those who do not need the more robust functions and properties of public-key systems.

8.6 *What are RC2 and RC4?*

RC2 and RC4 are variable-key-size cipher functions designed by Ron Rivest for fast bulk encryption. They are alternatives to DES (see Question 5.1) and are as fast or faster than DES. They can be more secure than DES because of their ability to use long key sizes; they can also be less secure than DES if short key sizes are used.

RC2 is a variable-key-size symmetric block cipher and can serve as a drop-in replacement for DES, for example in export versions of products otherwise using DES. RC2 can be used in the same modes as DES (see Question 5.3), including triple encryption. RC2 is approximately twice as fast as DES, at least in software. RC4 is a variable-key-size symmetric stream cipher and is 10 or more times as fast as DES in software. Both RC2 and RC4 are very compact in terms of code size.

An agreement between the Software Publishers Association (SPA) and the U.S. government gives RC2 and RC4 special status by means of which the export approval process is simpler and quicker than the usual cryptographic export process. However, to qualify for quick export approval a product must limit the RC2 and RC4 key sizes to 40 bits; 56 bits is allowed for foreign subsidiaries and overseas offices of U.S. companies. An additional 40-bit string, called a *salt*,

can be used to thwart attackers who try to precompute a large look-up table of possible encryptions. The salt is appended to the encryption key, and this lengthened key is used to encrypt the message; the salt is then sent, unencrypted, with the message. RC2 and RC4 have been widely used by developers who want to export their products; DES is almost never approved for export. RC2 and RC4 are proprietary algorithms of RSA Data Security, Inc.; details have not been published.

8.7 *What is PEM?*

PEM is the Internet Privacy-Enhanced Mail standard, designed, proposed, but not yet officially adopted, by the Internet Activities Board in order to provide secure electronic mail over the Internet. Designed to work with current Internet e-mail formats, PEM includes encryption, authentication, and key management, and allows use of both public-key and secret-key cryptosystems. Multiple cryptographic tools are supported: for each mail message, the specific encryption algorithm, digital signature algorithm, hash function, and so on are specified in the header. PEM explicitly supports only a few cryptographic algorithms; others may be added later. DES in CBC mode is currently the only message encryption algorithm supported, and both RSA and DES are supported for the key management. PEM also supports the use of certificates, endorsing the CCITT X.509 standard for certificate structure.

The details of PEM can be found in Internet RFCs (Requests For Comments) 1421 through 1424. PEM is likely to be officially adopted by the Internet Activities Board within one year. Trusted Information Systems has developed a free non-commercial implementation of PEM, and other implementations should soon be available as well.

8.8 *What is RIPEM?*

RIPEM is a program developed by Mark Riordan that enables secure Internet e-mail; it provides both encryption and digital signatures, using RSA and DES routines from RSAREF (see Question 8.10). RIPEM is not fully PEM-compatible; for example, it does not currently support certificates. However, future versions will include certificates and will be fully compliant with the PEM standard. RIPEM is available free for non-commercial use in the U.S. and Canada. To get RIPEM, obtain an ftp account at `ripem.msu.edu`.

8.9 *What is PKCS?*

PKCS (Public-Key Cryptography Standards) is a set of standards for implementation of public-key cryptography. It has been issued by RSA Data Security, Inc. in cooperation with a computer industry consortium, including Apple, Microsoft, DEC, Lotus, Sun and MIT. PKCS has been cited by the OIW (OSI

Implementors' Workshop) as a method for implementation of OSI standards. PKCS is compatible with PEM (see Question 8.7) but extends beyond PEM. For example, where PEM can only handle ASCII data, PKCS is designed for binary data as well. PKCS is also compatible with the CCITT X.509 standard.

PKCS includes both algorithm-specific and algorithm-independent implementation standards. Specific algorithms supported include RSA, DES, and Diffie-Hellman key exchange. It also defines algorithm-independent syntax for digital signatures, digital envelopes (for encryption), and certificates; this enables someone implementing any cryptographic algorithm whatsoever to conform to a standard syntax and thus preserve interoperability. Documents detailing the PKCS standards can be obtained by sending e-mail to **pkcs@rsa.com** or by anonymous ftp to **rsa.com**.

8.10 *What is RSAREF?*

RSAREF is a collection of cryptographic routines in portable C source code, available at no charge from RSA Laboratories, a division of RSA Data Security, Inc. It includes RSA, MD2, MD5, and DES; Diffie-Hellman key exchange will be included in a forthcoming version. It includes both low-level subroutines, such as modular exponentiation, and high-level cryptographic functions, such as verification of digital signatures. The arithmetic routines can handle multiple-precision integers, and the RSA algorithm routines can handle variable key sizes. RSAREF is fully compatible with the PEM and PKCS standards.

RSAREF is available to citizens of the U.S. or Canada and to permanent residents of the U.S. It can be used in personal, non-commercial applications but cannot be used commercially or sent outside the U.S. and Canada. The RSAREF license contains more details on the usage allowed and disallowed. RSAREF is available on the Internet by sending e-mail to **rsaref@rsa.com** or by ftp to **rsa.com**.

9 Acknowledgements

I would like to thank the following people, who have provided information and helpful suggestions: Burt Kaliski, Jim Bidzos, Matt Robshaw, Steve Dusse, Kurt Stammberger, George Parsons, John Gilmore, Stuart Haber, Dorothy Denning, and Dennis Branstad.

References

[1] American National Standards Institute. *Working Draft: American National Standard X9.30-199X: Public Key Cryptography Using Irreversible*

Algorithms for the Financial Services Industry: Part 1: The Digital Signature Algorithm (DSA). American Bankers Association, Washington, D.C., March 4, 1993.

[2] J. Bamford. *The Puzzle Palace.* Houghton Mifflin, Boston, 1982.

[3] J.P. Barlow. Decrypting the puzzle palace. *Communications of the ACM*, 35(7):25–31, July 1992.

[4] D. Bayer, S. Haber, and W.S. Stornetta. Improving the efficiency and reliablility of digital time-stamping. In R.M. Capocelli, editor, *Sequences '91: Methods in Communication, Security, and Computer Science*, Springer-Verlag, Berlin, 1992.

[5] P. Beauchemin, G. Brassard, C. Crepeau, C. Goutier, and C. Pomerance. The generation of random numbers that are probably prime. *J. of Cryptology*, 1:53–64, 1988.

[6] E. Biham and A. Shamir. *Differential Cryptanalysis of the Data Encryption Standard.* Springer-Verlag, New York, 1993.

[7] E. Biham and A. Shamir. Differential cryptanalysis of the full 16-round DES. In *Advances in Cryptology — Crypto '92*, Springer-Verlag, New York, 1993.

[8] M. Blum and S. Goldwasser. An efficient probabilistic public-key encryption scheme which hides all partial information. In *Advances in Cryptology — Crypto '84*, pages 289–299, Springer-Verlag, New York, 1985.

[9] J. Brandt and I. Damgård. On generation of probable primes by incremental search. In *Advances in Cryptology — Crypto '92*, Springer-Verlag, New York, 1993.

[10] G. Brassard. *Modern Cryptology.* Volume 325 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1988.

[11] D.M. Bressoud. *Factorization and Primality Testing. Undergraduate Texts in Mathematics*, Springer-Verlag, New York, 1989.

[12] E.F. Brickell, D.E. Denning, S.T. Kent, D.P. Maher, and W. Tuchman. *Skipjack Review, Interim Report: The Skipjack Algorithm.* July 28, 1993.

[13] E.F. Brickell and A.M. Odlyzko. Cryptanalysis: A survey of recent results. *Proceedings of the IEEE*, 76:578–593, 1988.

[14] J. Brillhart, D.H. Lehmer, J.L. Selfridge, B. Tuckerman, and S.S. Wagstaff Jr. *Factorizations of $b^n \pm 1$, $b = 2, 3, 5, 6, 7, 10, 11, 12$ up to High Powers.* Volume 22 of *Contemporary Mathematics*, American Mathematical Society, Providence, Rhode Island, 2nd edition, 1988.

[15] J. Buchmann, J. Loho, and J. Zayer. An implementation of the general number field sieve. In *Advances, in Cryptology — Crypto '93*, Springer-Verlag, New York, 1994. To appear.

[16] J.P. Buhler, H.W. Lenstra, and C. Pomerance. Factoring integers with the number field sieve. 1992. To appear.

[17] M.V.D. Burmester, Y.G. Desmedt, and T. Beth. Efficient zero-knowledge identification schemes for smart cards. *Computer Journal*, 35:21–29, 1992.

[18] K.W. Campbell and M.J. Wiener. Proof that DES is not a group. In *Advances in Cryptology — Crypto '92*, Springer-Verlag, New York, 1993.

[19] CCITT (Consultative Committee on International Telegraphy and Telephony). *Recommendation X.509: The Directory—Authentication Framework*. 1988.

[20] Comptroller General of the United States. *Matter of National Institute of Standards and Technology — Use of Electronic Data Interchange Technology to Create Valid Obligations*. December 13, 1991. File B-245714.

[21] D. Coppersmith, A.M. Odlyzko, and R. Schroeppel. Discrete logarithms in GF(p). *Algorithmica*, 1:1–15, 1986.

[22] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, Massachusetts, 1990.

[23] G. Davida. *Chosen signature cryptanalysis of the RSA public key cryptosystem*. Technical Report TR-CS-82-2, Dept of EECS, University of Wisconsin, Milwaukee, 1982.

[24] B. den Boer and A. Bosselaers. An attack on the last two rounds of MD4. In *Advances in Cryptology — Crypto '91*, pages 194–203, Springer-Verlag, New York, 1992.

[25] B. den Boer and A. Bosselaers. Collisions for the compression function of MD5. In *Advances in Cryptology — Eurocrypt '93*, 1993. Preprint.

[26] Dorothy E. Denning. The Clipper encryption system. *American Scientist*, 81(4):319–323, July–August 1993.

[27] W. Diffie. The first ten years of public-key cryptography. *Proceedings of the IEEE*, 76:560–577, 1988.

[28] W. Diffie and M.E. Hellman. Exhaustive cryptanalysis of the NBS Data Encryption Standard. *Computer*, 10:74–84, 1977.

[29] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22:644–654, 1976.

[30] T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31:469–472, 1985.

[31] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology — Crypto '86*, pages 186–194, Springer-Verlag, New York, 1987.

[32] S. Goldwasser and S. Micali. Probabilistic encryption. *J. of Computer and System Sciences*, 28:270–299, 1984.

[33] D.M. Gordon. Discrete logarithms using the number field sieve. March 28, 1991. To appear.

[34] D.M. Gordon and K.S. McCurley. Massively parallel computation of discrete logarithms. In *Advances in Cryptology — Crypto '92*, Springer-Verlag, New York, 1993.

[35] J. Hastad. Solving simultaneous modular equations of low degree. *SIAM J. Computing*, 17:336–241, 1988.

[36] M.E. Hellman. A cryptanalytic time-memory trade off. *IEEE Transactions on Information Theory*, IT-26:401–406, 1980.

[37] D. Kahn. *The Codebreakers*. Macmillan Co., New York, 1967.

[38] B.S. Kaliski. *A survey of encryption standards*. RSA Data Security, Inc., September 2, 1993.

[39] B.S. Kaliski Jr., R.L. Rivest, and A.T. Sherman. Is the data encryption standard a group? *J. of Cryptology*, 1:3–36, 1988.

[40] S. Kent. *RFC 1422: Privacy Enhancement for Internet Electronic Mail, Part II: Certificate-Based Key Management*. Internet Activities Board, February 1993.

[41] D.E. Knuth. *The Art of Computer Programming*. Volume 2, Addison-Wesley, Reading, Mass., 2nd edition, 1981.

[42] N. Koblitz. *A Course in Number Theory and Cryptography*. Springer-Verlag, New York, 1987.

[43] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[44] X. Lai and J.L. Massey. A proposal for a new block encryption standard. In *Advances in Cryptology — Eurocrypt '90*, pages 389–404, Springer-Verlag, Berlin, 1991.

[45] B.A. LaMacchia and A.M. Odlyzko. Computation of discrete logarithms in prime fields. *Designs, Codes and Cryptography*, 1:47–62, 1991.

[46] S. Landau. Zero knowledge and the Department of Defense. *Notices of the American Mathematical Society*, 35:5–12, 1988.

[47] A.K. Lenstra and H.W. Lenstra Jr. Algorithms in number theory. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, MIT Press/Elsevier, Amsterdam, 1990.

[48] A.K. Lenstra, H.W. Lenstra Jr., M.S. Manasse, and J.M. Pollard. The factorization of the ninth Fermat number. 1991. To appear.

[49] A.K. Lenstra and M.S. Manasse. Factoring with two large primes. In *Advances in Cryptology — Eurocrypt '90*, pages 72–82, Springer-Verlag, Berlin, 1991.

[50] H.W. Lenstra Jr. Factoring integers with elliptic curves. *Ann. of Math.*, 126:649–673, 1987.

[51] M. Matsui. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology — Eurocrypt '93*, Springer-Verlag, Berlin, 1993. To appear.

[52] R.C. Merkle and M.E. Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory*, IT-24:525–530, 1978.

[53] R.C. Merkle and M.E. Hellman. On the security of multiple encryption. *Communications of the ACM*, 24:465–467, July 1981.

[54] E. Messmer. NIST stumbles on proposal for public-key encryption. *Network World*, 9(30), July 27, 1992.

[55] S. Micali. Fair public-key cryptosystems. In *Advances in Cryptology — Crypto '92*, Springer-Verlag, New York, 1993.

[56] V.S. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology — Crypto '85*, pages 417–426, Springer-Verlag, New York, 1986.

[57] National Institute of Standards and Technology (NIST). The Digital Signature Standard, proposal and discussion. *Communications of the ACM*, 35(7):36–54, July 1992.

[58] National Institute of Standards and Technology (NIST). *FIPS Publication 180: Secure Hash Standard (SHS).* May 11, 1993.

[59] National Institute of Standards and Technology (NIST). *FIPS Publication 46-1: Data Encryption Standard.* January 22, 1988. Originally issued by National Bureau of Standards.

[60] National Institute of Standards and Technology (NIST). *FIPS Publication 81: DES Modes of Operation.* December 2, 1980. Originally issued by National Bureau of Standards.

[61] National Institute of Standards and Technology (NIST). Notice of proposal for grant of exclusive patent license. *Federal Register*, 58(108), June 8, 1993.

[62] National Institute of Standards and Technology (NIST). A proposed Federal Information Processing Standard for an Escrowed Encryption Standard (EES). *Federal Register*, 58(145), July 30, 1993.

[63] National Institute of Standards and Technology (NIST). *Publication XX: Announcement and Specifications for a Digital Signature Standard (DSS).* August 19, 1992.

[64] A.M. Odlyzko. Discrete logarithms in finite fields and their cryptographic significance. In *Advances in Cryptology — Eurocrypt '84*, pages 224–314, Springer-Verlag, Berlin, 1984.

[65] Office of the Press Secretary. *Statement.* The White House, April 16, 1993.

[66] J. Pollard. Monte Carlo method for factorization. *BIT*, 15:331–334, 1975.

[67] J. Pollard. Theorems of factorization and primality testing. *Proc. Cambridge Philos. Soc.*, 76:521–528, 1974.

[68] M.O. Rabin. *Digitalized signatures as intractable as factorization.* Technical Report MIT/LCS/TR-212, MIT, 1979.

[69] R.L. Rivest. Cryptography. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, MIT Press/Elsevier, Amsterdam, 1990.

[70] R.L. Rivest. Finding four million random primes. In *Advances in Cryptology — Crypto '90*, pages 625–626, Springer-Verlag, New York, 1991.

[71] R.L Rivest. The MD4 message digest algorithm. In *Advances in Cryptology — Crypto '90*, pages 303–311, Springer-Verlag, New York, 1991.

[72] R.L. Rivest. Response to NIST's proposal. *Communications of the ACM*, 35:41–47, July 1992.

[73] R.L. Rivest. *RFC 1321: The MD5 Message-Digest Algorithm.* Internet Activities Board, April 1992.

[74] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[75] C.P. Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology — Crypto '89*, pages 239–251, Springer-Verlag, New York, 1990.

[76] M. Shand and J. Vuillemin. Fast implementations of RSA cryptography. In *Proceedings of the 11th IEEE Symposium on Computer Arithmetic*, pages 252–259, IEEE Computer Society Press, Los Alamitos, CA, 1993.

[77] R.D. Silverman. The multiple polynomial quadratic sieve. *Math. Comp.*, 48:329–339, 1987.

[78] M.E. Smid and D.K. Branstad. Response to comments on the NIST proposed Digital Signature Standard. In *Advances in Cryptology — Crypto '92*, Springer-Verlag, New York, 1993.

[79] J.G. Steiner, B.C. Neuman, and J.I. Schiller. Kerberos: an authentication service for open network systems. In *Usenix Conference Proceedings*, pages 191–202, Dallas, Texas, February 1988.

[80] M.J. Wiener. Efficient DES key search. August 20, 1993. Presented at Crypto '93 rump session.

Efficient DES Key Search

Michael J. Wiener

Bell-Northern Research, P.O. Box 3511 Station C, Ottawa, Ontario, K1Y 4H7, Canada

1993 August 20

Abstract. Despite recent improvements in analytic techniques for attacking the Data Encryption Standard (DES), exhaustive key search remains the most practical and efficient attack. Key search is becoming alarmingly practical. We show how to build an exhaustive DES key search machine for $1 million that can find a key in 3.5 hours on average. The design for such a machine is described in detail for the purpose of assessing the resistance of DES to an exhaustive attack. This design is based on mature technology to avoid making guesses about future capabilities. With this approach, DES keys can be found one to two orders of magnitude faster than other recently proposed designs. The basic machine design can be adapted to attack the standard DES modes of operation for a small penalty in running time. The issues of development cost and machine reliability are examined as well. In light of this work, it would be prudent in many applications to use DES in a triple-encryption mode.

1. Introduction

From the time that details of the Data Encryption Standard (DES) [4] were made available, there has been a great deal of controversy surrounding its short 56-bit keys. One of the first criticisms came from Diffie and Hellman who argued that it was feasible to build a machine that could attack DES by exhaustively searching through all 2^{56} DES keys, and that the cost of building this machine would decrease rapidly with time [6]. Since then, several attempts have been made to assess the cost and time required for exhaustive key search [7, 8, 10, 15].

In recent years, analytic techniques have been found to attack DES. Biham and Shamir's differential cryptanalysis can be used to find a DES key given 2^{47} chosen plaintexts [1, 2]. Matsui's "linear cryptanalysis" can find DES keys given 2^{47} known plaintexts [12]. However, unless these attacks are improved greatly to lower the number of plaintexts required, they will be of only theoretical interest. For now, the most practical and economical way to attack DES is by exhaustive key search.

The purpose of this paper is to evaluate the resistance of DES to a known-plaintext key search attack. This is done by designing a machine which takes a plaintext-ciphertext pair (P, C) and finds the DES key or keys which encrypt P to produce C. There have been numerous unpublished and unverifiable claims about how fast the DES key space can be searched. To avoid adding to

this list of questionable claims, a great deal of detail in the design of a key search machine is included in the appendices. This detailed work was done to obtain an accurate assessment of the cost of the machine and the time required to find a DES key. There are no plans to actually build such a machine.

2. Key Search Chip

The most important component in the key search machine is the key search chip. This chip takes a plaintext-ciphertext pair (P, C), a starting key K, and searches through keys until one is found that encrypts P to produce C. The detailed design of this chip is given in Appendix A.

In order to maximize the rate at which keys are tested, the implementation of DES within the chip is fully pipelined. This means that there is a separate implementation of each round of DES and the rounds are separated by registers. This is similar in concept to an assembly line where at any given time, items are at varying stages of processing. On each clock tick, the plaintext P and a new key enter the pipeline, and all previous encryptions advance through another round of DES. In this way there are 16 encryptions taking place simultaneously. Although it takes 16 clock ticks to complete an encryption, one encryption completes at the end of the pipeline on each clock tick. As each encryption completes, the result is compared to the ciphertext C that was originally loaded into the chip. If there is a match, then the chip stops and saves the key that lead to the match.

An important feature of this chip is that there is very little need for input and output. This makes it possible to clock the chip at high speed without the requirement of getting data in and out of it at high speed. The chip described in Appendix A can be comfortably clocked at 50 MHz, and would cost $10.50 per chip to manufacture. This precise assessment of speed and cost is made possible by the fact that the chip has been designed down to the gate level in a mature CMOS process. Some cost estimates for key search chips in the literature suffer from incompleteness of design and guesses about future technology.

In summary, this chip can test keys at the impressive rate of 50 million per second. This is made possible by the use of pipelining and the elimination of high-speed input and output.

3. Key Search Machine

The basic approach taken to building a key search machine out of key search chips is to build a hierarchy of controllers. At the lowest level, a group of ten key search chips is controlled by a microcontroller which assigns tasks and polls the chips for any found keys. The key search board design described in Appendix B shows that for the selected board size, twelve groups of ten key search chips fit on a board. The twelve microcontrollers are themselves controlled by a master microcontroller. The master microcontroller assigns a plaintext-ciphertext pair and a range of

keys to each slave microcontroller. The slave microcontrollers subdivide the range of keys into ten subranges for the ten key search chips.

A description of the shelf and frame design is given in Appendix C. Each shelf contains twelve key search boards and a controller board. Four shelves are stacked together to form a frame which is controlled by a PC. Overall, a frame costs $100 000 and contains 5760 key search chips.

A key search machine would consist of a number of frames which would find one or more candidate keys. Each candidate key would have to be tested to see if it is correct (perhaps by using another plaintext-ciphertext pair).

4. Performance

Using the design described here, DES keys can be found alarmingly quickly. On average, one would expect to search through 2^{55} keys before the correct key is found. Using the fact that a key search chip tests 50 million keys per second and a frame costs $100 000 and contains 5760 key search chips, we get the following table of machine cost versus key search time.

Key Search Machine Cost	Expected Search Time
$100 000	35 hours
$1 000 000	3.5 hours
$10 000 000	21 minutes

5. Development Cost

A key search machine could be built in about ten months by three people (not including support groups for manufacture and chip and board layout). One person would be needed to capture the chip design, simulate it, and oversee its layout and fabrication. A second person would design, simulate, and oversee the fabrication of the key search board and the shelf controller board. This person would also handle the shelf and frame design. A third person would write software for the PC, shelf controller board, and the microcontrollers. Taking into account loaded labour rates, the cost of employing these people is about $100 000 each. Allowing an additional $200 000 for support in the chip layout, chip fabrication, and board layout brings the total development cost to about $500 000.

6. Comparison to Other Recent Key Search Designs

The approach to DES key search proposed here is considerably faster than other recently proposed designs. Comparisons to other designs are given below.

Garon and Outerbridge [8] used an approach to DES key search that is similar to the one used here; design a specialized key search chip and pack as many of them as possible into a machine. Based on Garon and Outerbridge estimates, a $1 million machine could search half of the DES key space in 9 days in 1995 and 43 hours in the year 2000. However, using the design in this paper, this task would take only 3.5 hours using today's technology. This is 60 times faster than the 1995 estimate and 12 times faster than the year 2000 estimate.

At Crypto '92, Eberle [7] described a 1 Gbit/s gallium-arsenide (GaAs) DES chip. This chip costs $300. Using $1 million worth of these chips, it would take 8 days to search through half of the key space. This is 55 times slower than the design in this paper. This difference is mainly due to the difference in chip cost and the use of pipelining. Using Eberle's chip has the advantage of not having to do a chip design, but the gain from designing a new chip is considerable.

During the rump session of Crypto '92, Wayner [15] proposed the idea of searching the DES key space using a content-addressable search engine. This approach has the advantage that it works on a general-purpose machine. If the content-addressable machine has already been acquired for general use by some organization, then one only needs to write a small amount of software and acquire time on the machine. Based on Wayner's estimate, one could search through half of the key space in 30 days on a $1 million content-addressable machine. The drawback of this approach is that it is 200 times slower than the design in this paper.

An often proposed method of attacking DES is to take advantage of the spare cycles on a large installed base of PCs and workstations. However, DES is poorly suited to software implementation. On a Sun SparcStation-2, a key can be tested in 50 µs. This means that it would take 2500 workstations to keep up with a single key search chip of the type described in Appendix A. One key search frame has the equivalent key searching power of 14 million workstations.

7. Reliability and Testing

When DES key search machines were first proposed, reliability was a difficult issue to deal with. But today, large systems are routinely built that are much more complex and have higher reliability requirements than a key search machine.

The analysis in Appendix D shows that the mean time between failures of a machine consisting of n key search frames is about $9500/n$ hours. From Section 4, the expected time of a single key search for n key search frames is $35/n$ hours. Dividing these two expressions, we find that a failure is expected every 270 keys found. Because the failure rate is low, a reasonable approach to testing is to perform a self-test between key searches and replace any faulty cards. There is no need to do testing during key searches.

8. Modes of DES

Thus far we have concentrated on attacking DES when it is used in the electronic codebook (ECB) mode. However, there are other standard modes of DES [5]. All of the 64-bit modes can be attacked using the same machine design (this is demonstrated below). To support modes which produce less than 64 bits of ciphertext at a time requires modifications to the chip design. The modes of DES are shown in the following figure. In all cases a 56-bit key and a 64-bit initialization vector (IV) are required. On each step, the key, IV, and plaintext block P are used to produce a ciphertext block C.

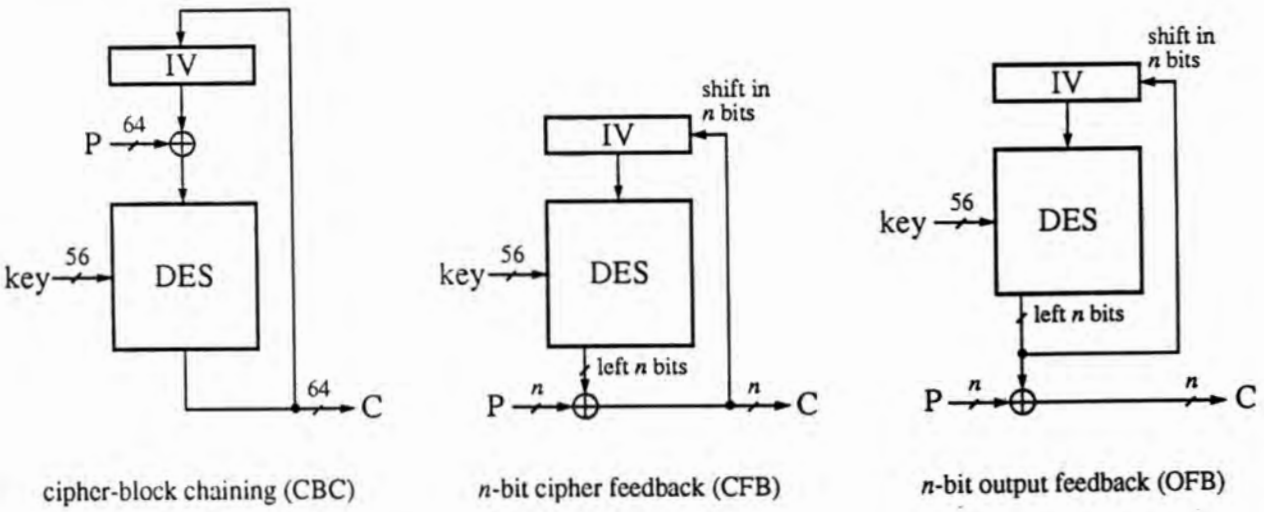


Figure 1: Encryption using the various modes of DES

For the 64-bit modes, it is possible to construct a pair (P', C') which can be used as plaintext and ciphertext inputs to the existing key search design to find the key. For CBC mode, if a plaintext block P_i is known, then use $P' = P_i \oplus C_{i-1}$ and $C' = C_i$. For 64-bit CFB mode, if a plaintext block P_i is known, then use $P' = C_{i-1}$ and $C' = P_i \oplus C_i$. For 64-bit OFB mode, if two consecutive plaintext blocks P_i and P_{i+1} are known, then use $P' = P_i \oplus C_i$ and $C' = P_{i+1} \oplus C_{i+1}$.

Among the modes of DES that produce less than 64 bits of ciphertext at a time, two popular ones are 1-bit and 8-bit CFB. The modification to the key search design required to support these modes as well as ECB mode are considered in Appendix E. We assume that there are 64 bits of known plaintext available and that multiple encryptions with the same key are required to obtain a full match between plaintext and ciphertext. The conclusion is that the new key search chip would cost $12. This increases the cost of a key search frame to the point where only nine frames could be built for $1 million. This slows down the attack on ECB mode somewhat. The CFB modes are slowed down further by the need to perform additional encryptions when there are partial matches with the ciphertext. For 1-bit CFB, there is a 50% chance that the output bit from

an encryption matches the stored ciphertext bit and that another encryption will have to be performed with the same key. As soon as a ciphertext bit does not match, the key is rejected. This makes the expected time to find a key for 1-bit CFB mode twice as long as for ECB mode. Using similar reasoning, 8-bit CFB mode takes approximately 256/255 times as long as ECB mode. This information is summarized in the following table.

Mode of DES	Expected Key Search Time for a $1 million Machine
ECB	4 hours
CBC	4 hours
64-bit OFB	4 hours
64-bit CFB	4 hours
8-bit CFB	4 hours
1-bit CFB	8 hours

9. Triple-DES

One method of improving the security of DES greatly is to use triple-DES. With triple-DES encryption, each block of plaintext P is processed three times using either two or three independent keys to produce the ciphertext C as shown in Figure 2.

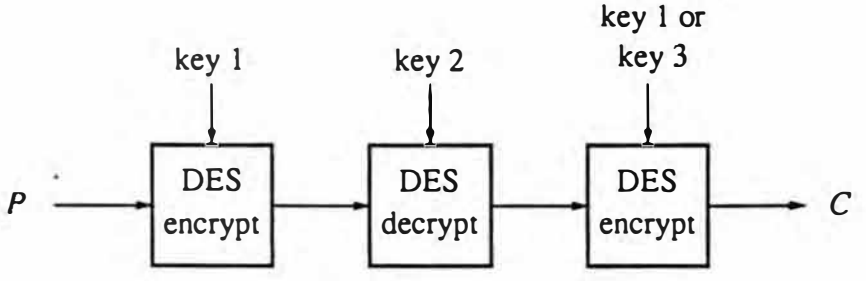


Figure 2: Triple-DES Encryption

The reason for making the middle operation a decryption is for compatibility with single-DES. If all keys are made equal, then triple-DES collapses to single-DES.

The two-key version of triple-DES was estimated to be 10^{13} times stronger than single-DES by van Oorschot and Wiener [14]. The three-key version is even stronger. In cases where security must be improved for a large installed base of equipment using DES, it is likely to be much less costly to switch to triple-DES than it is to switch to a different cryptosystem.

10. Conclusion

It is possible to build a $1 million machine that can attack DES and recover a key in an average time of 3.5 hours. The machine uses a known plaintext to exhaustively search through the DES key space and could be developed for $500 000 in about 10 months. Because a great deal of detail has gone into the design of the key search machine, we can have high confidence in the assessment of its cost and speed. The key search design presented here is one to two orders of magnitude faster than other recently proposed designs.

Even cryptosystems with 64-bit keys may be vulnerable. If DES were modified to use 64-bit keys, there would be 2^8 times as many keys to search through, and a $10 million machine would take an average of 3.7 days to find a key.

It is possible to build a key search machine that can support a range of modes of DES with little penalty in run-time. A $1 million machine would take 8 hours on average to find a key used in 1-bit CFB mode and 4 hours on average for any of ECB, CBC, 64-bit OFB, 64-bit CFB, or 8-bit CFB mode.

This work shows that exhaustive DES key search is alarmingly economical. If it ever was true that attacking DES was only within the reach of large governments, it is clearly no longer true. A fairly painless way to improve security dramatically is to switch to triple-DES.

Appendix A: Key Search Chip Detailed Design

This appendix contains the detailed design of a DES key search chip. The chip takes a plaintext P, a ciphertext C, and a starting key and searches through keys until one is found that encrypts P to produce C. An important feature of this chip is that the implementation of DES is pipelined. This means that there is separate silicon for each of the 16 rounds, and the rounds are separated by registers. At any given time, there are 16 keys in the pipeline. On each clock tick a new key enters the pipeline, other keys advance by one round, and the oldest key's encryption completes. As each encryption completes at the end of the pipeline, the result is compared to the ciphertext value C. Using this pipelined approach, keys can be tested at the rate of one key every clock tick.

The chip is designed using the standard cells in a 0.8 μm CMOS process. The chip is fully synchronous; all flip-flops use the same clock. The design is presented in a top-down fashion. The first subsection in this appendix contains the top-level design consisting of a number of blocks. In the following subsections, each block is subdivided into smaller blocks. At the end of this hierarchical decomposition, the standard cells (basic components such as and-gates, or-gates, and flip-flops) used in the design are described.

To help in distinguishing high-level blocks from standard cells, high-level blocks have arrowheads on their inputs and bold letters in their names. For standard cells, names are not bolded and there are no arrows on input lines.

Throughout the design, standard bit numbering is used with 0 as the least significant (rightmost) bit. This differs from the DES description [4] where the numbering of a 64-bit quantity begins with 1 for the most significant bit up to 64 for the least significant bit.

Sufficient buffering was included in the chip design to limit the fan-out of all gates to at most 8. For lines interconnecting the topmost blocks in the top level design, the fan-out was limited to 4.

Most modern large chip designs include built-in self test (BIST) circuitry. Because the key search chip is small and can be fully tested from its external pins, no BIST circuitry has been included in its design. If BIST were added, the cost of the key search chip would increase by about 30%.

The worst case delays from the output of one flip-flop to the input of another flip-flop occur in the rounds of encryption. Taking into account this delay, the delay through the flip-flop, and clock skew across the chip, this chip can be clocked comfortably at 50 MHz. This means that once the pipeline is full, a key is tested every 20 ns.

The most important factor in determining the cost of this chip is its area. In the CMOS process used here, units of area are called "sites". A site is 3.6 μm × 54.4 μm. Each of the standard cells requires an integral number of sites (see section A.44). For this chip, the total area of all gates not including clock regeneration or pads for external connections is 73637 sites. Clock regeneration requires 210 sites for each 100 flip-flops. Because there are 2164 flip-flops, 4620 sites are needed for clock regeneration. This makes a total of 78257 sites not including pads. A standard way to quote the size of a chip is to specify the number of equivalent gates using a 2-input nand gate as the standard. For this CMOS process, a 2-input nand gate requires 3 sites. Therefore, this chip design has 26086 equivalent gates. Taking into account the area required for interconnections and pads, this chip's area would be 260 mils × 260 mils.[1] Although, this chip only needs 27 pins, it is too large to fit in a 28-pin PLCC (plastic leaded chip carrier) package. The next standard size is a 44-pin PLCC package. Using a 44-pin PLCC package, the cost would be $10.50 per chip to fabricate and test in volumes of at least 10 000.

The power consumption of this chip would be about 450 mW. This is somewhat less than one might expect for a 50 MHz device. However, the chip is small and the I/O to it is quite slow. The I/O consists of loading values into the chip initially and polling it periodically to see if it has found the desired key. This means that very little power is dissipated in the pads.

---

[1] 1 mil = 0.001 inch = 25.4 μm.

The following table gives the area and page number for each block in the hierarchical design.

Block name	Area (sites)	Page number	Block name	Area (sites)	Page number
top level	73637	9	S_output	277	29
stop	21	12	minterm_4	15	30
compare	383	12	register_32	320	31
compare_16	94	12	register_8	80	31
compare_4	22	13	xor_48	192	32
plaintext and ciphertext	1148	13	xor_32	128	33
write_control	36	14	xor_8	32	33
mux2_register8	136	15	E_perm	0	34
key_counter	1437	16	P_perm	0	34
write_control7	33	17	key_round_1	560	34
poly_register	207	18	key_round_2	560	35
count_register	195	19	register_28	280	35
output_key	1395	21	register_7	70	36
mux2_register7	122	22	PC2_perm	0	36
mux7_1	51	23	rotate_left	0	36
buffer8	24	23	rotate_left_2	0	37
buffer3	9	24	initial_perm	0	37
pipelined_DES	67968	24	inverse_initial	0	37
encrypt_round	3688	26	PC1_perm	0	37
S_box	2728	27	inverse_PC1	0	38
dual_dec38	64	27	external_interface †	137	38
S1_wiring to S8_wiring	0	28			

† The area listed for the external interface does not include the pads.

A.1 Top-Level Design

The key search chip has a microprocessor interface which allows a plaintext, a ciphertext, and a starting key value to be written in, and the found key to be read out. In addition, the microprocessor can read and write the status ("go" or "stop") of the chip. The chip contains four registers which are visible to the microprocessor (see Figure A.1). The write-only registers include the 64-bit plaintext register, the 64-bit ciphertext register, and the 56-bit key counter register. There is also the read-only 56-bit output_key register. Note that in this design, key parity bits as defined in DES [4] are not included. The microprocessor can also read and write the stop bit which indicates the state of the chip.

A total of 27 pins are used. Five pins are allocated to each of power and ground. One pin is for the 50 MHz clock input, and there is a tri-state pin for testing purposes. The remaining 15 pins are the microprocessor interface. There is a chip enable line, a read/write line, 5 address lines, and 8 data lines for byte-wide access. The following table shows the address map of the chip.

Address	Read	Write
00-07	not used (zeros are read)	plaintext (bytes 0-7)
08-0F	not used (zeros are read)	ciphertext (bytes 0-7)
10-16	output_key (bytes 0-6)	key_counter (bytes 0-6)
17	not used (zeros are read)	not used
18-1F	stop bit (in bit 0 with 7 leading zeros)	stop bit (from bit 0 - other bits ignored)

While the chip is in the go state, the key_counter steps through the keys and puts one key per clock tick into the DES pipeline. This key is used with the contents of the plaintext register to perform the first round of encryption. All other stages of the pipeline perform one round of a previous encryption. From the last stage of the pipeline comes the result of an encryption and the key used in that encryption. The key is loaded into the output_key register (regardless of whether it is the desired key or not). The result of encryption is compared to the contents of the ciphertext register. If the ciphertext matches, then the chip goes into the stop state.

In the stop state, the key_counter does not count, and it is possible for the microprocessor to write a new value into the key_counter. The DES pipeline continues to run in the stop state, but the output_key register ceases to take in the keys coming out of the pipeline. In this way, the output_key register will be left holding the key which caused a match with the ciphertext register. In the case where the microprocessor stops the chip, the output_key register will simply hold the last key which came out of the pipeline before the chip was stopped.

The intended way for a microprocessor to use this chip is for it to first stop the chip, fill up the plaintext, ciphertext, and key_counter registers, and then put the chip in the go state. The microprocessor then polls the chip periodically to see if it has found a key and has stopped. This continues until a key is found or the microprocessor stops the chip to start a new task.

To save circuitry, the key_counter does not count linearly through the key space. The carry look-ahead logic for a 56-bit counter would be large, and it would be difficult to get it to run at the required speed. A linear-feedback shift register based on a primitive generator polynomial is used to step through all the non-zero keys. The all-zero key must be tested separately by the processor which controls the collection of key search chips. The generator polynomial used is $g = x^{56} + x^7 + x^4 + x^2 + 1$. If the first key tried is k, then the subsequent keys tried are $kx \bmod g$, $kx^2 \bmod g$, $kx^3 \bmod g$, and so on. The processor which controls the collection of key search chips will have to take this counting method into account when calculating the starting key values to load into each chip's key_counter.

In Figure A.1, clock, power, and ground are not shown leaving the external interface, but they go to all other blocks.

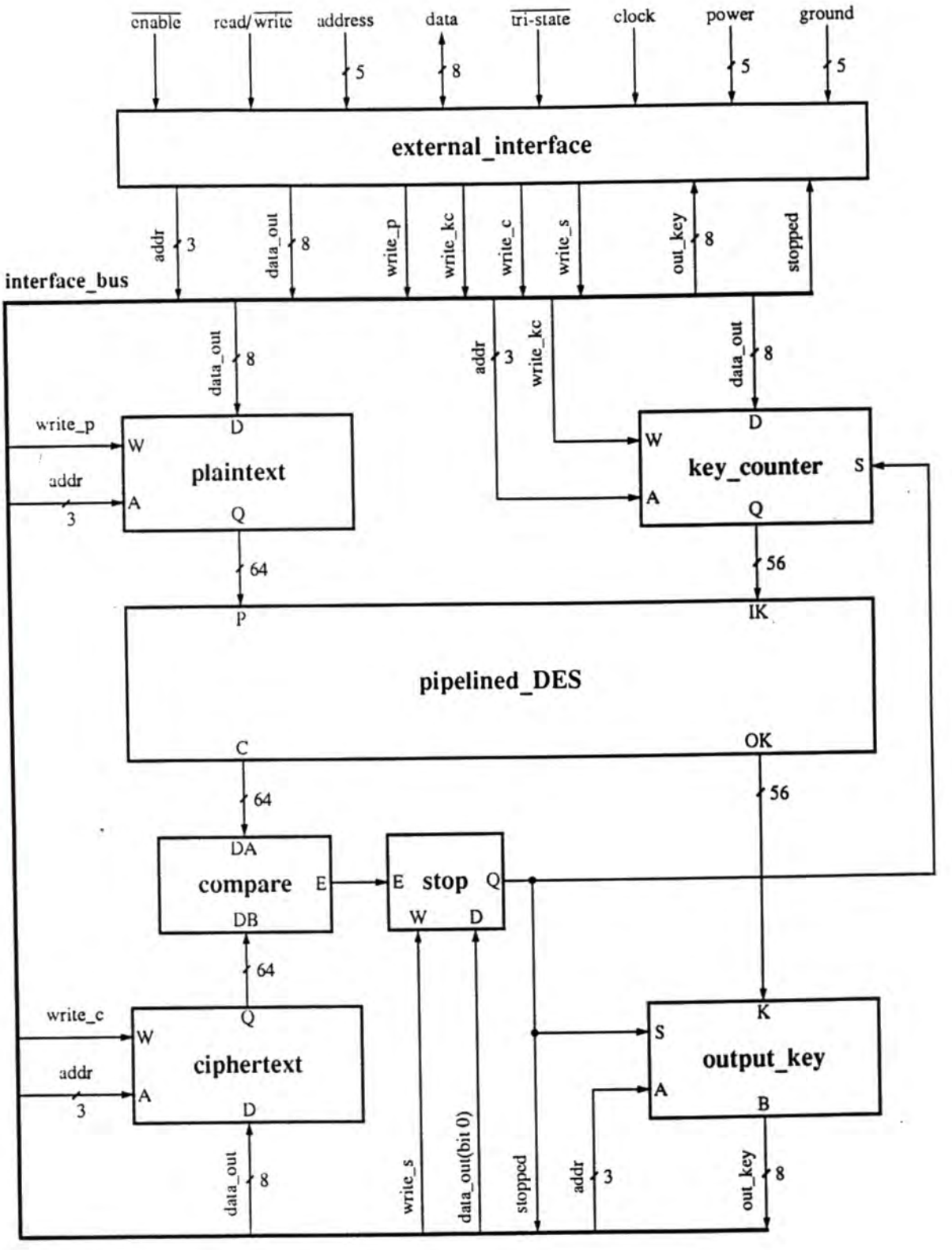


Figure A.1: Top-Level Key Search Chip Design

A.2 Stop Block

The stop block contains the flip-flop which indicates whether the chip is in the stop state (Q = 1) or the go state (Q = 0). The inputs to this block are a write line W, a data line D, and an equal line E which is a 1 if there has been a match with the ciphertext register. If W = 1, the stop bit takes on the value of D. If W = 0 and E = 1, the stop bit becomes a 1. If W = 0 and E = 0, the stop bit retains its old value.

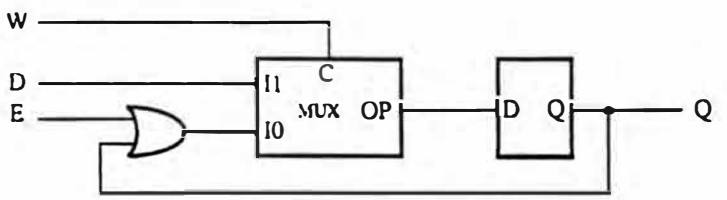


Figure A.2: Stop Block

A.3 Compare Block

The compare block outputs a 1 if all 64 bits of DA are equal to the corresponding bits of DB. The output is 0 otherwise.

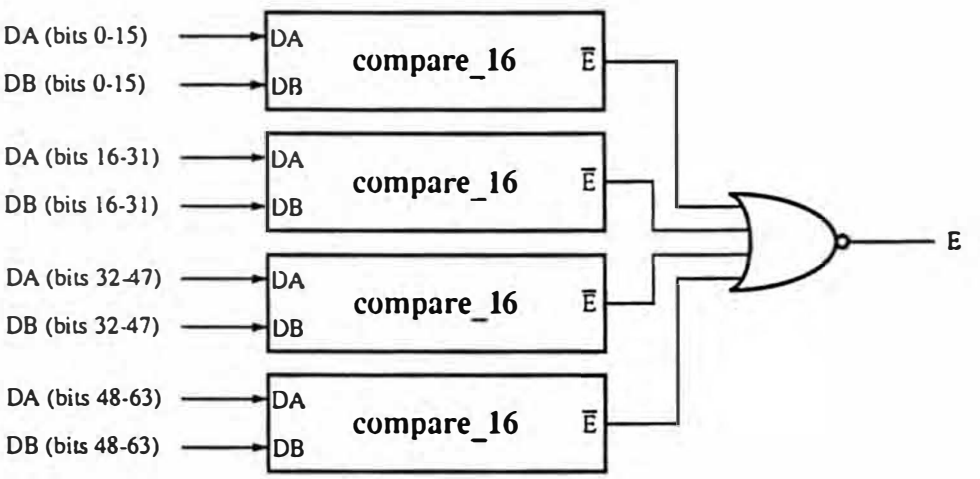


Figure A.3: Compare Block

A.4 Compare_16 Block

The compare_16 block outputs a 0 if all 16 bits of DA are equal to the corresponding bits of DB. The output is 1 otherwise.

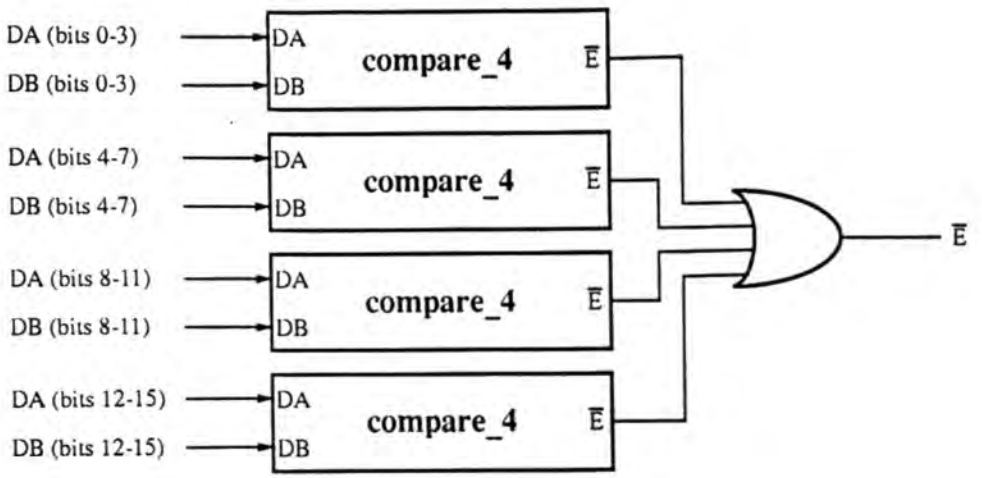


Figure A.4: Compare_16 Block

A.5 Compare_4 Block

The compare block outputs a 0 if all 4 bits of DA are equal to the corresponding bits of DB. The output is 1 otherwise.

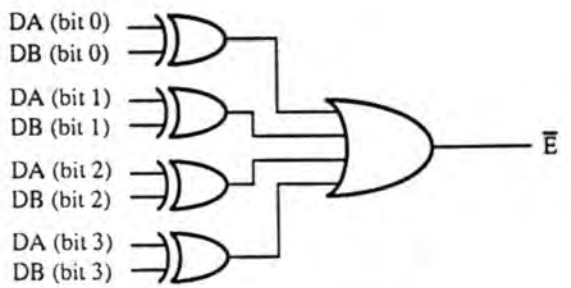


Figure A.5: Compare_4 Block

A.6 Plaintext and Ciphertext Blocks

The plaintext and ciphertext blocks are identical and each contains a 64-bit register. The inputs are a write line W, 3 address lines A, and 8 data lines D. If W = 0, the 64-bit register retains its old value. If W = 1, the data lines are written into the register byte specified by the address lines and the remaining register bytes retain their old values.

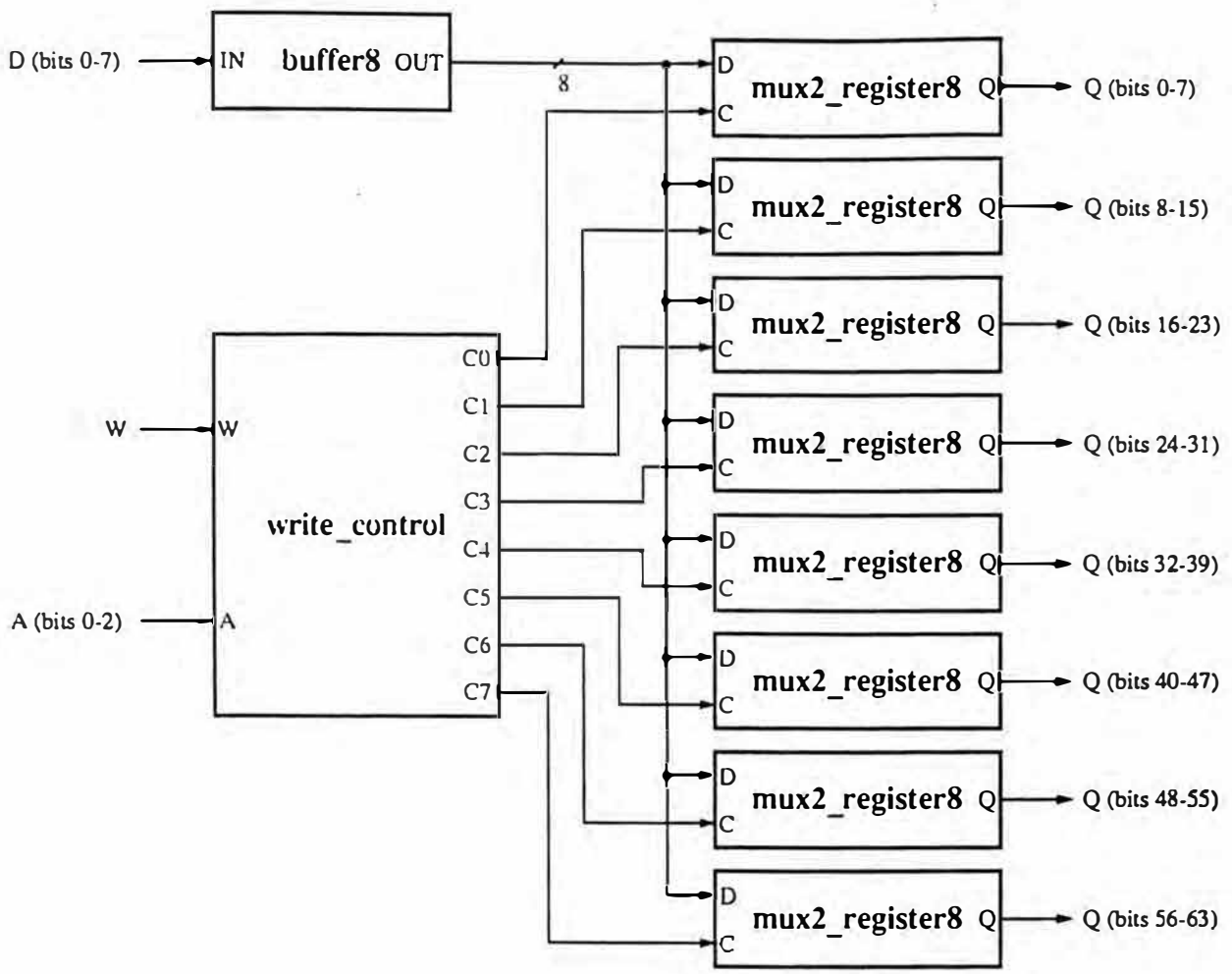


Figure A.6: Plaintext and Ciphertext Blocks

A.7 Write_control Block

The write_control block takes a write line W, 3 address lines A, and produces 8 output control lines. If W = 0, the outputs are all 0. If W = 1, the output corresponding to the address lines is a 1 and all other outputs are 0.

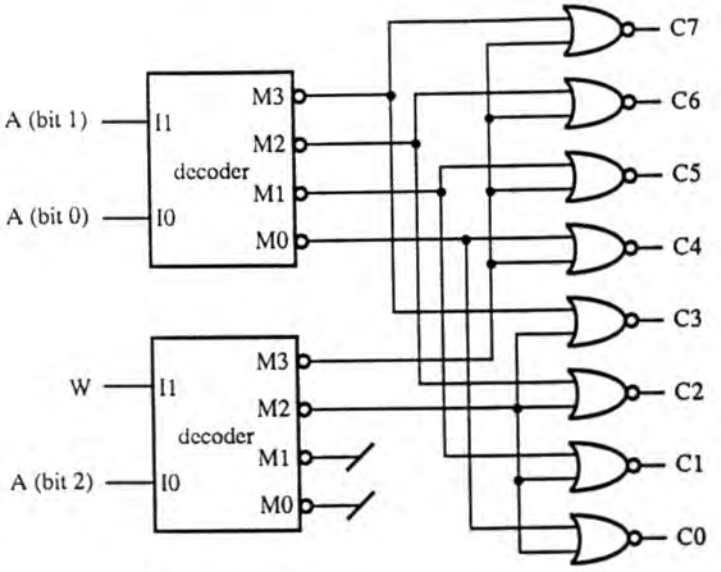


Figure A.7: Write_control Block

A.8 Mux2_register8 Block

The mux2_register8 block contains an 8-bit register. The inputs are a control line C and 8 data lines D. If C = 1, the register takes on the values on the data lines. If C = 0, the register retains its old value.

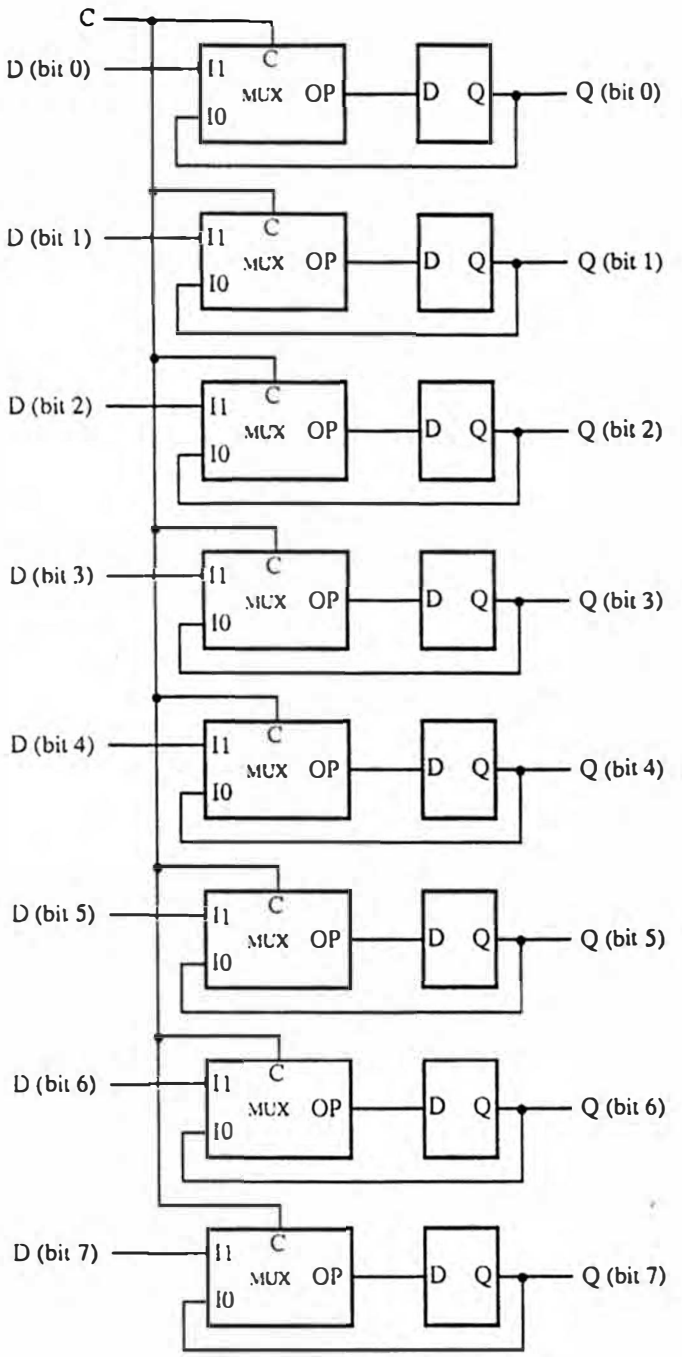


Figure A.8: Mux2_register8 Block

A.9 Key_counter Block

The key_counter block contains a 56-bit register. The inputs are the stop bit S indicating the state of the chip, a write line W, 3 address lines A, and 8 data lines D. If S = 1 and W = 0, the 56-bit register retains its old value. If S = 1 and W = 1, the byte of the register corresponding to the

value of the address lines takes the value of the data lines, and all other register bytes retain their old values (if the address is 7, the entire register retains its old value). If S = 0, the register counts to the next key value. In terms of polynomials, the next key value is kx mod g, where k is the current key value and $g = x^{56} + x^7 + x^4 + x^2 + 1$ is the generator polynomial. This is the same as shifting the register left and if a 1 shifts from the top end, exclusive-or-ing the generator polynomial onto the register.

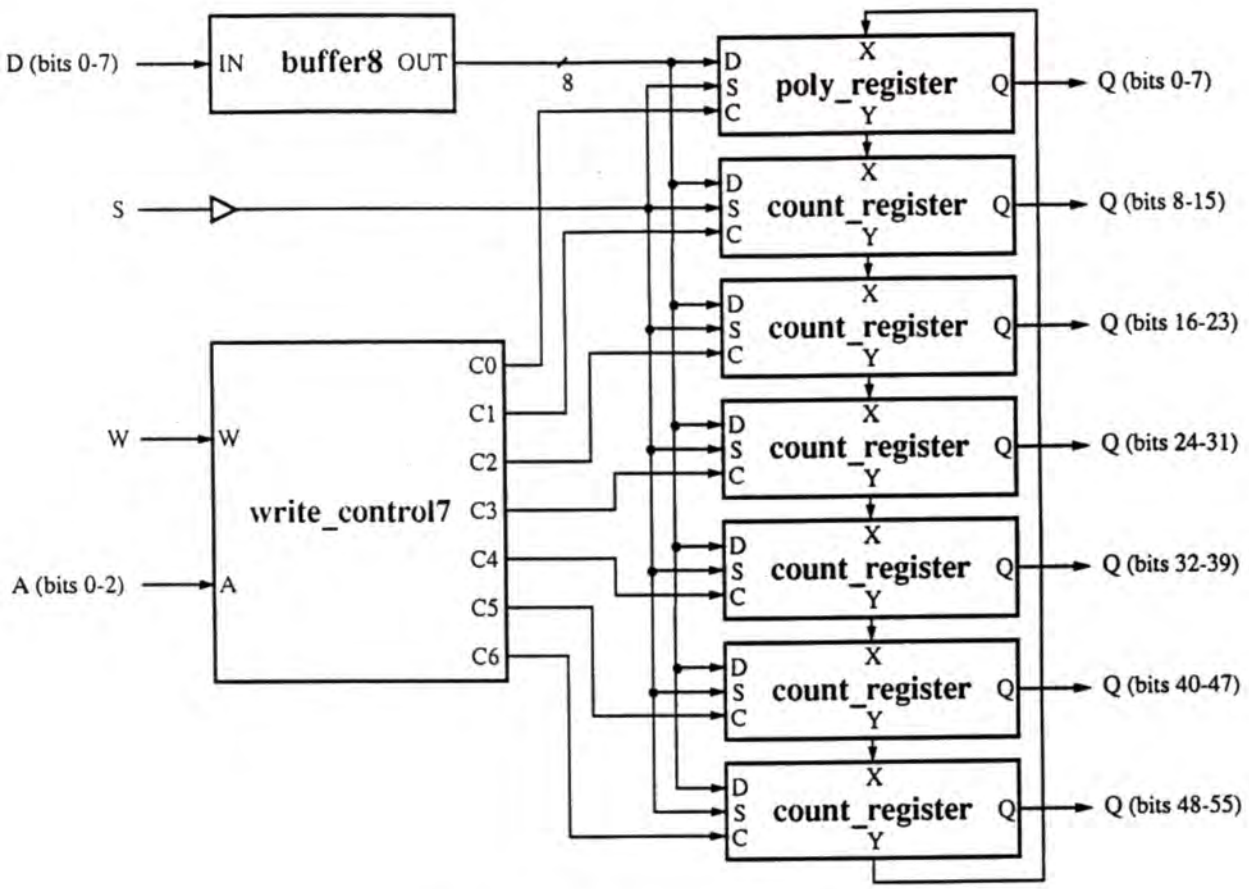


Figure A.9: Key_counter Block

A.10 Write_control7 Block

The write_control7 block takes a write line W, 3 address lines A, and produces 7 output control lines. If W = 0, all outputs are 0. If W = 1, the output corresponding to the address lines is a 1 and all other outputs are 0 (if the address is 7, all outputs are 0).

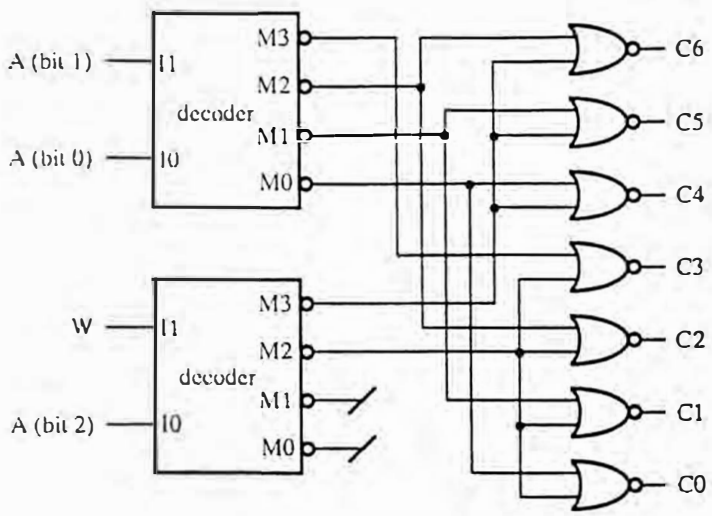


Figure A.10: Write_control7 Block

A.11 Poly_register Block

The poly_register block contains the least significant register byte of the key_counter. This is the only byte of the key_counter that is affected when we exclusive-or the generator polynomial onto the key_counter. The inputs are the stop bit S, the previous contents of the top bit of the key_counter X, a line to control writing C, and 8 data lines D. The outputs are the flip-flop outputs Q and the shifted out bit Y (which is just bit 7 of Q). If S = 1 and C = 0, the register retains its old value. If S = 1 and C = 1, the register takes the value of the data lines. If S = 0, the register is shifted left with X indicating whether we exclusive-or the bottom 8 bits of the generator polynomial onto the register.

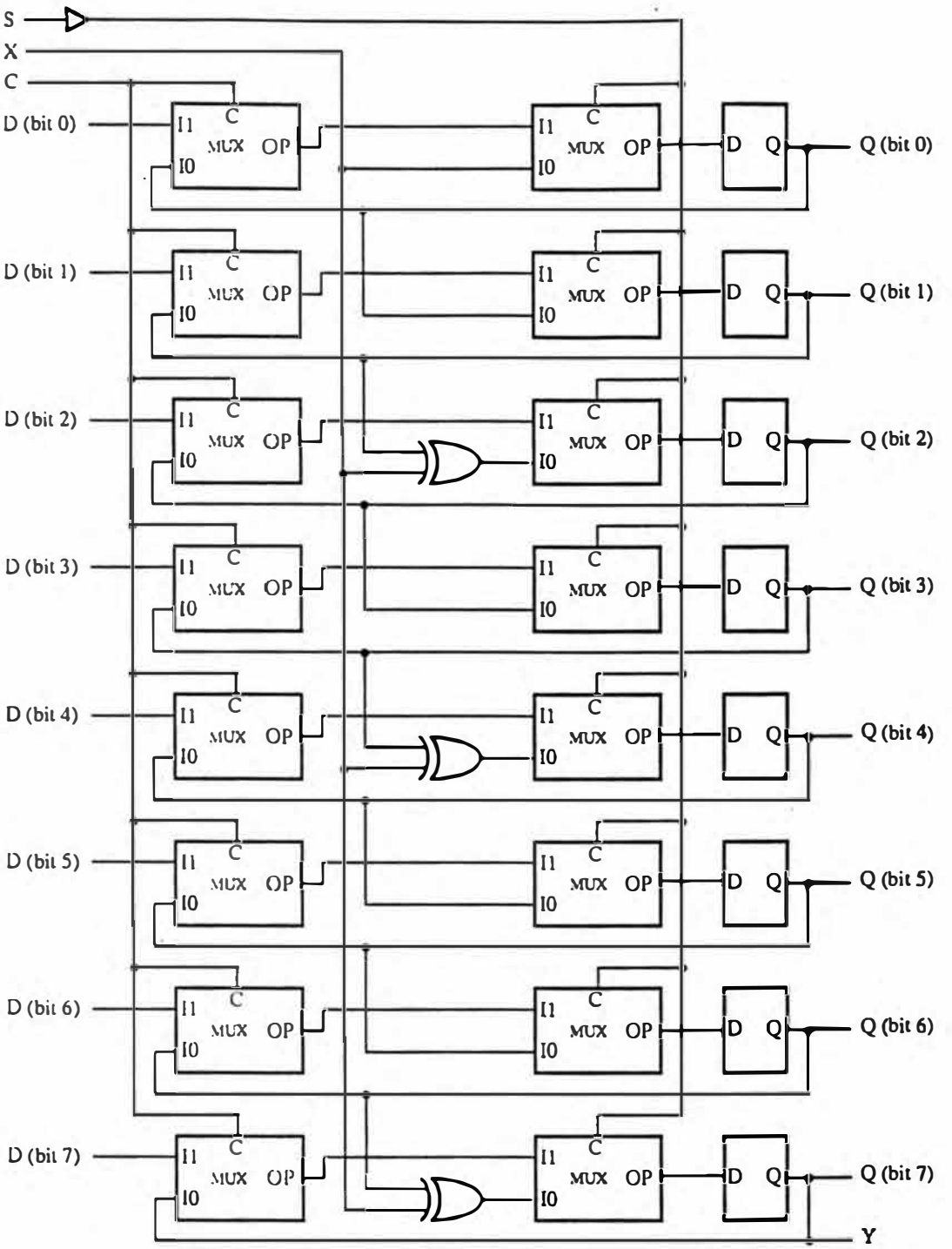


Figure A.11: Poly_register Block

A.12 Count_register Block

The count_register block contains a register byte of the key_counter. The inputs are the stop bit S, the bit to shift into the least significant flip-flop X, a line to control writing C, and 8 data lines D.

The outputs are the flip-flop outputs Q and the shifted out bit Y (which is just bit 7 of Q). If S = 1 and C = 0, the register retains its old value. If S = 1 and C = 1, the register takes the value of the data lines. If S = 0, the register is shifted left with X shifting into the least significant bit.

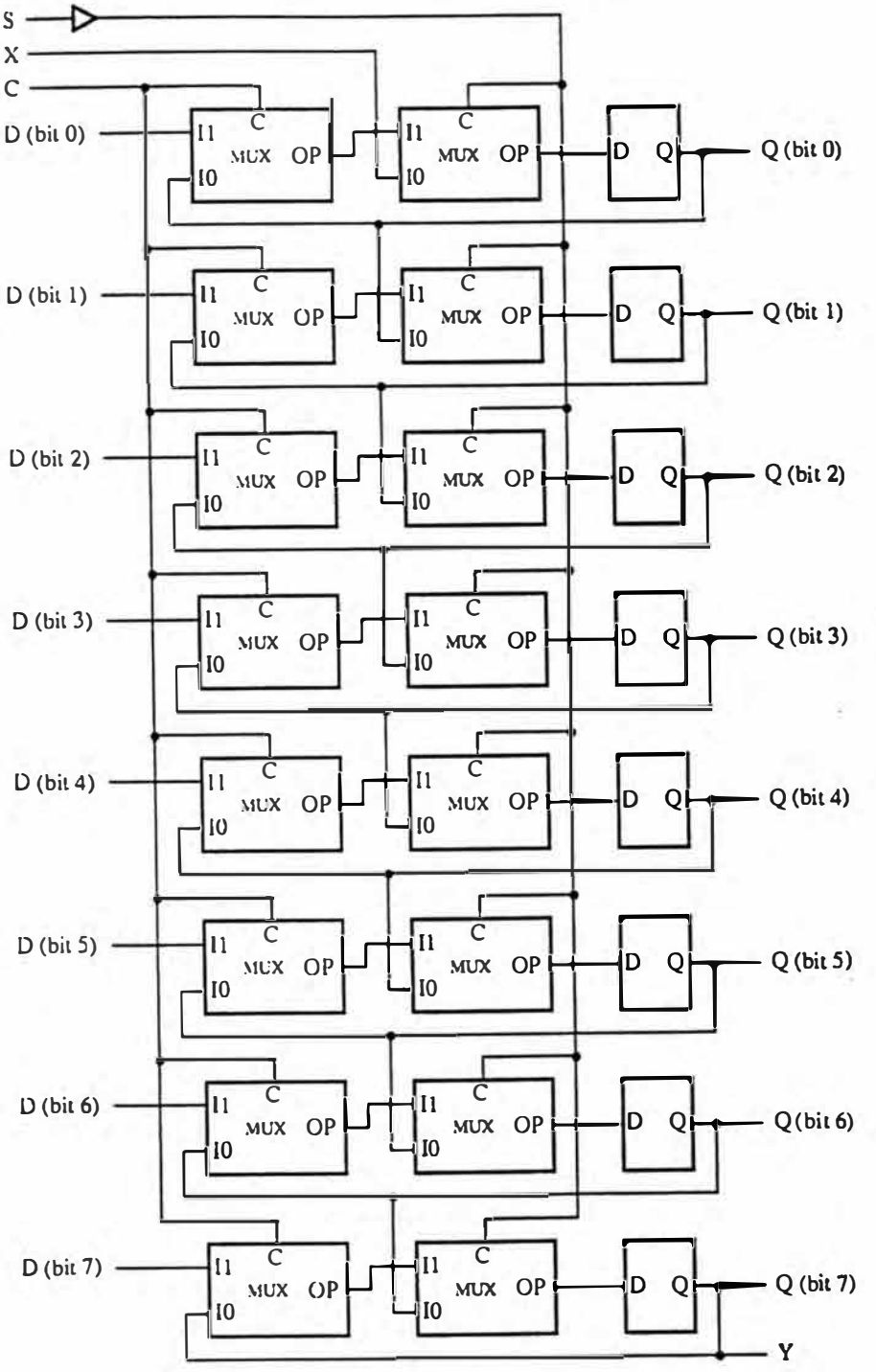


Figure A.12: Count_register Block

A.13 Output_key Block

The output_key block contains a 56-bit register. The inputs are the stop bit S, 3 address lines A, and 56 key lines K. If S = 0, the register takes on the value of the input key K. If S = 1, the register retains its old value. The output from this block is the byte B of the register specified by the address lines. If the specified address is 7, then all bits of B are 0.

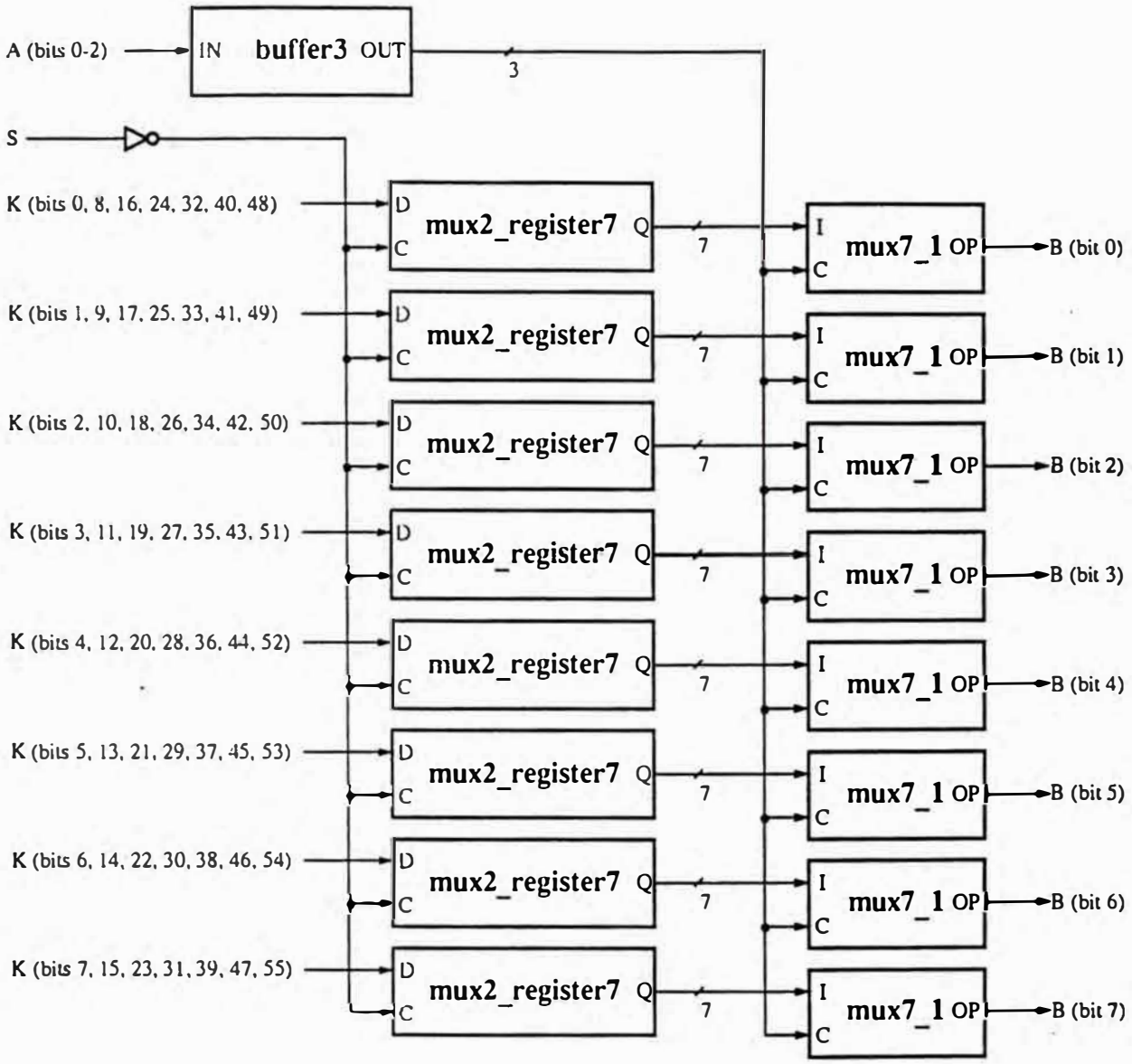


Figure A.13: Output_key Block

A.14 Mux2_register7 Block

The mux2_register7 block contains a 7-bit register. The inputs are a control line C and 7 data lines D. If C = 1, the register takes on the value of the data lines. If C = 0, the register retains its old value.

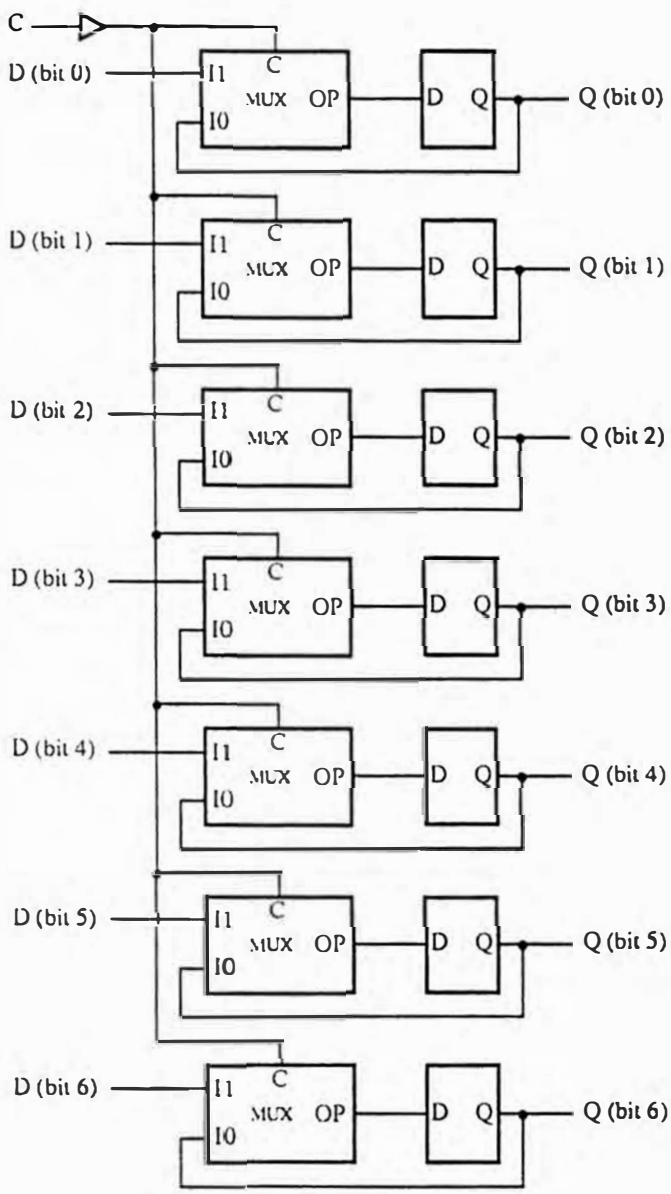


Figure A.14: Mux2_register7 Block

A.15 Mux7_1 Block

The inputs to the mux7_1 block are 3 control lines C and 7 input lines I. The output OP is equal to the input line that is specified by the control lines. If the control lines specify an address of 7, then OP = 0.

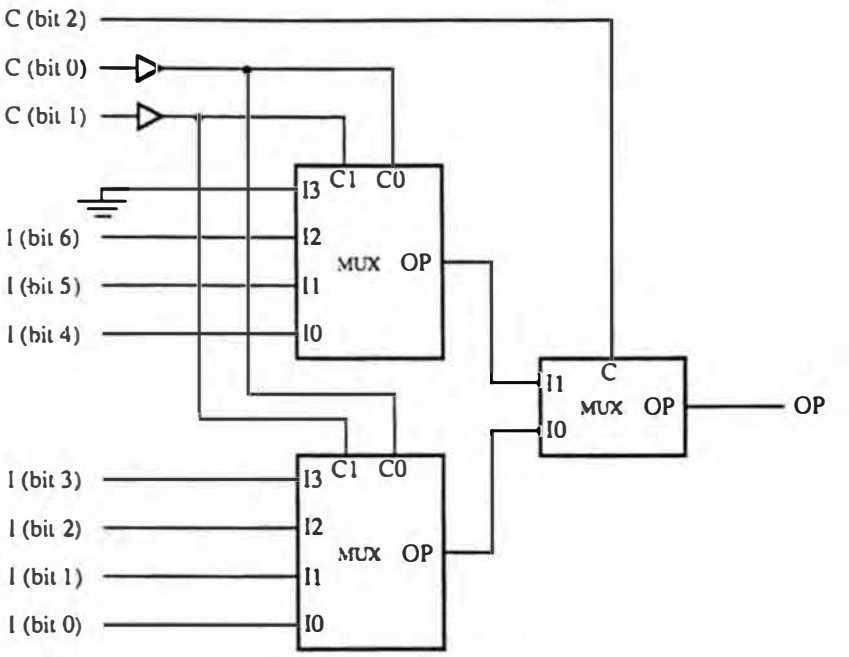


Figure A.15: Mux7_1 Block

A.16 Buffer8 Block

The buffer8 block simply buffers each of 8 input lines.

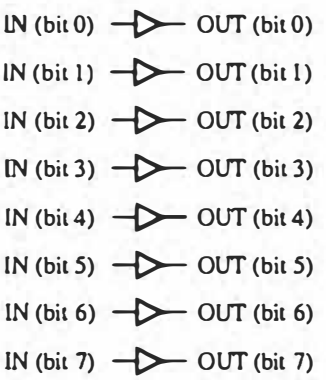


Figure A.16: Buffer8 Block

A.17 Buffer3 Block

The buffer3 block simply buffers each of 3 input lines.

Figure A.17: Buffer3 Block

A.18 Pipelined_DES Block

The pipelined_DES block performs the DES calculations at the core of the key search chip. The inputs are a 64-bit plaintext P and a 56-bit input key IK. The outputs are a 64-bit ciphertext C and a 56-bit output key OK. At the top of the block, the DES initial permutation is applied to the plaintext, and the PC1 permutation is applied to the input key. This is followed by a 16-stage pipeline. Each stage of the pipeline consists of key scheduling applied to the key to produce a sub-key, a round of encryption using the current data and current sub-key, and a 120-bit register to pipeline the 64 bits of data and 56 bits of key. After the pipeline, the data is permuted to form the ciphertext, and the key is permuted to restore the original order of the key bits. It takes 16 clock ticks to produce a ciphertext, but there are 16 encryptions in progress at any given time. Intermediate results from 16 different encryptions are held in the 120-bit registers within the pipeline stages. On each clock tick, an encryption completes producing a ciphertext and the associated key.

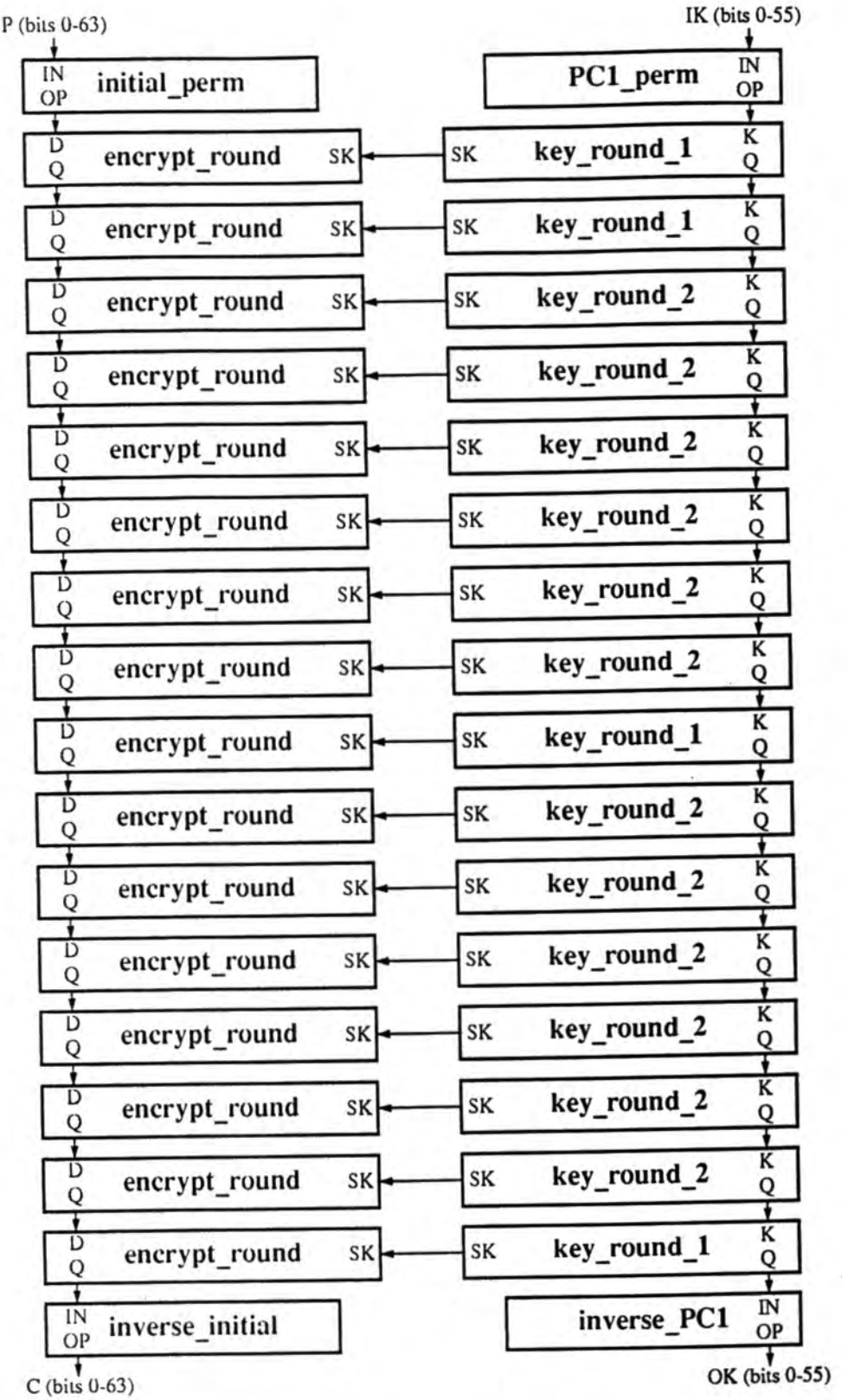


Figure A.18: Pipelined_DES Block

A.19 Encrypt_round Block

The encrypt_round block performs one round of DES not including key scheduling. The inputs are 64 bits of data D and 48 bits of sub-key SK. A round of DES begins with applying the E permutation to the right half of D expanding it to 48 bits, and exclusive-or-ing this result with the sub-key. The resulting 48 bits go through the S-boxes producing 32 bits which then go through the P permutation. The result of the P permutation is exclusive-or-ed with the left half of D to produce the right half of the output data for this round of DES. The left half of the output data for this round of DES is simply the same as the right half of the input data. The output data goes into a 64-bit pipelining register.

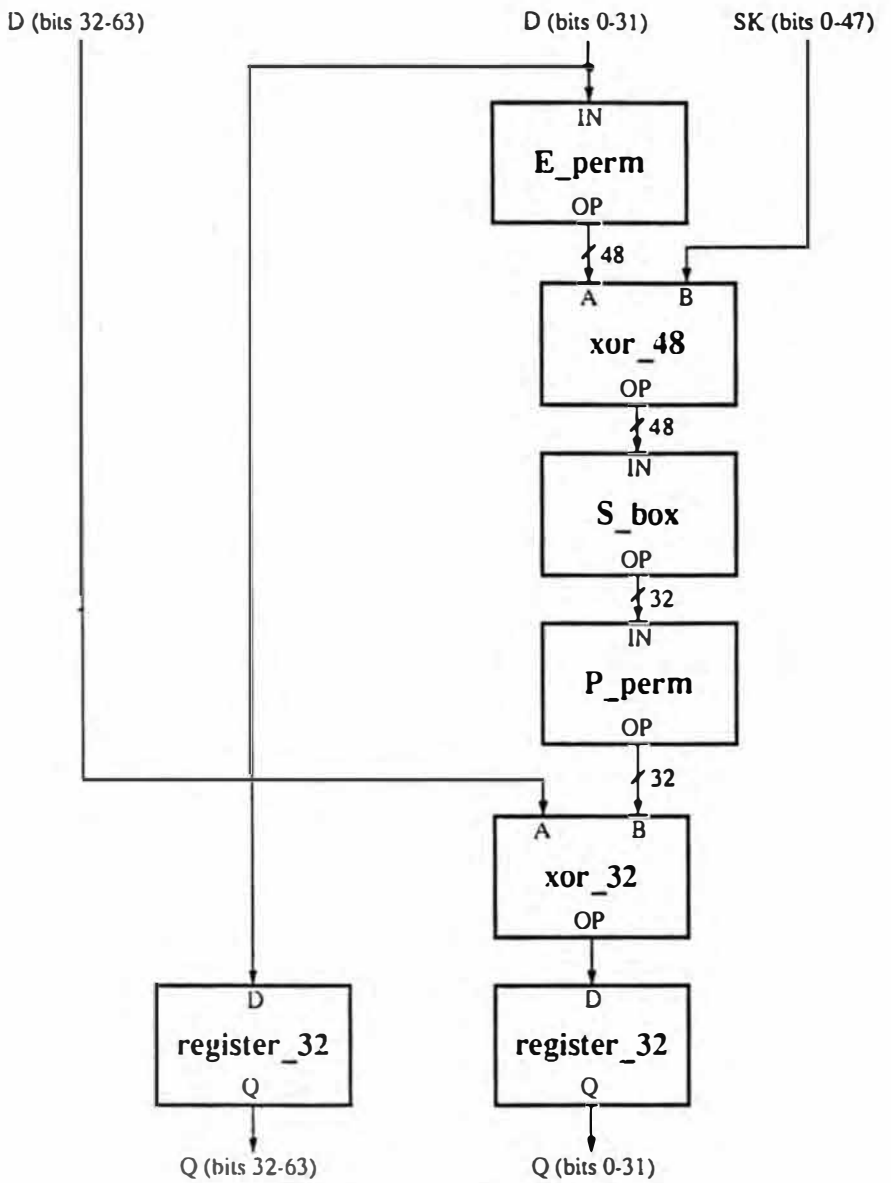


Figure A.19: Encrypt_round Block

A.20 S_box Block

There are 8 DES S-boxes. Each is a look-up table with 6 input bits and 4 output bits. The structure of each table is such that each of the outputs 0 to 15 occurs exactly 4 times. The approach to implementing the S-boxes taken here is to fully decode the 6 input bits, and then combine the 4 minterms for each of the non-zero table outputs. The 15 signals corresponding to the non-zero table outputs are then combined to form the 4 S-box output bits.

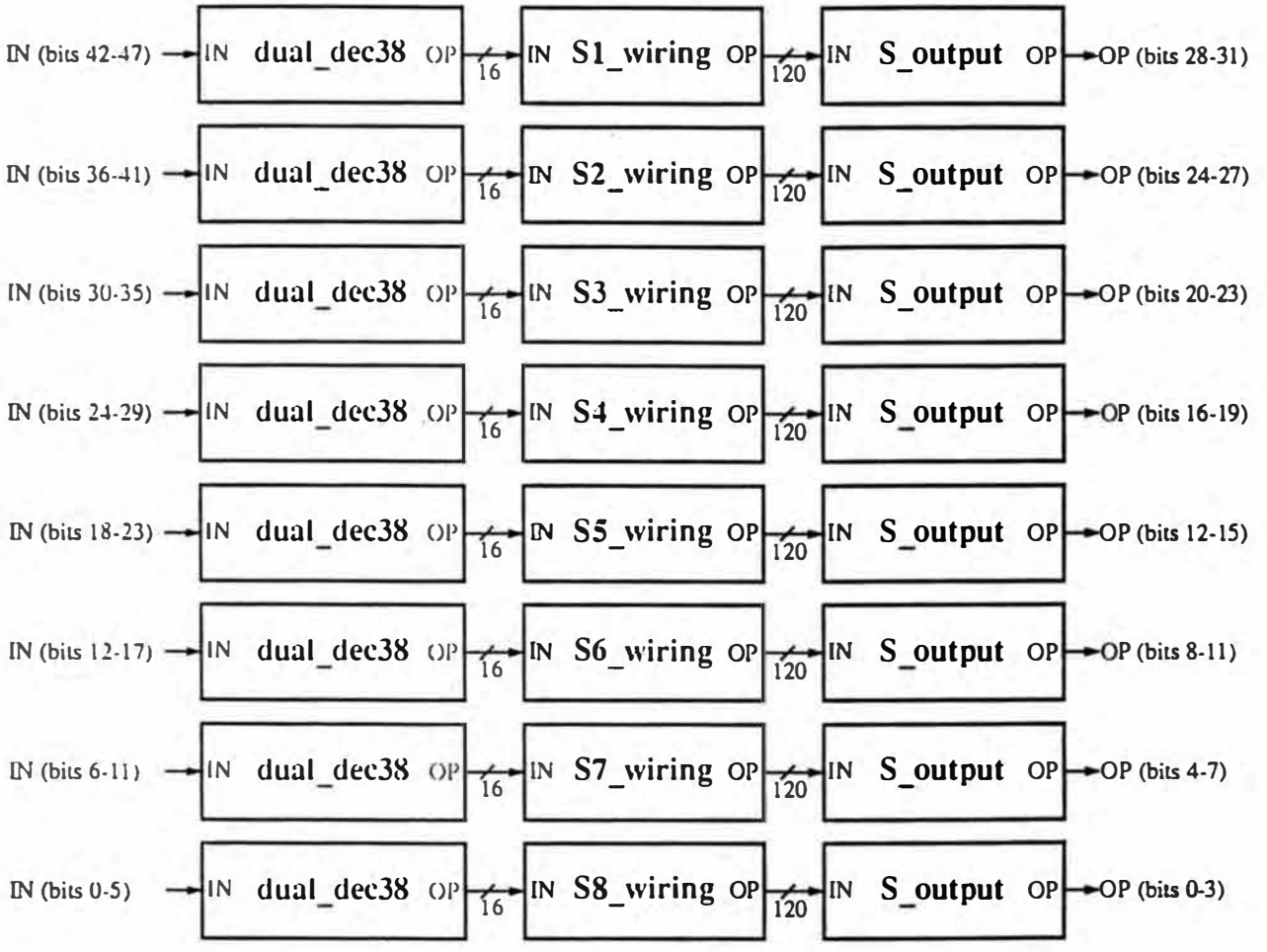


Figure A.20: S_box Block

A.21 Dual_dec38 Block

The dual_dec38 block takes 6 inputs which are split into two groups of 3 bits. Each group of 3 bits is fully decoded to 8 inverted outputs. The outputs from this block can be used to produce a minterm of the 6 inputs by combining one signal from each group of 8 outputs.

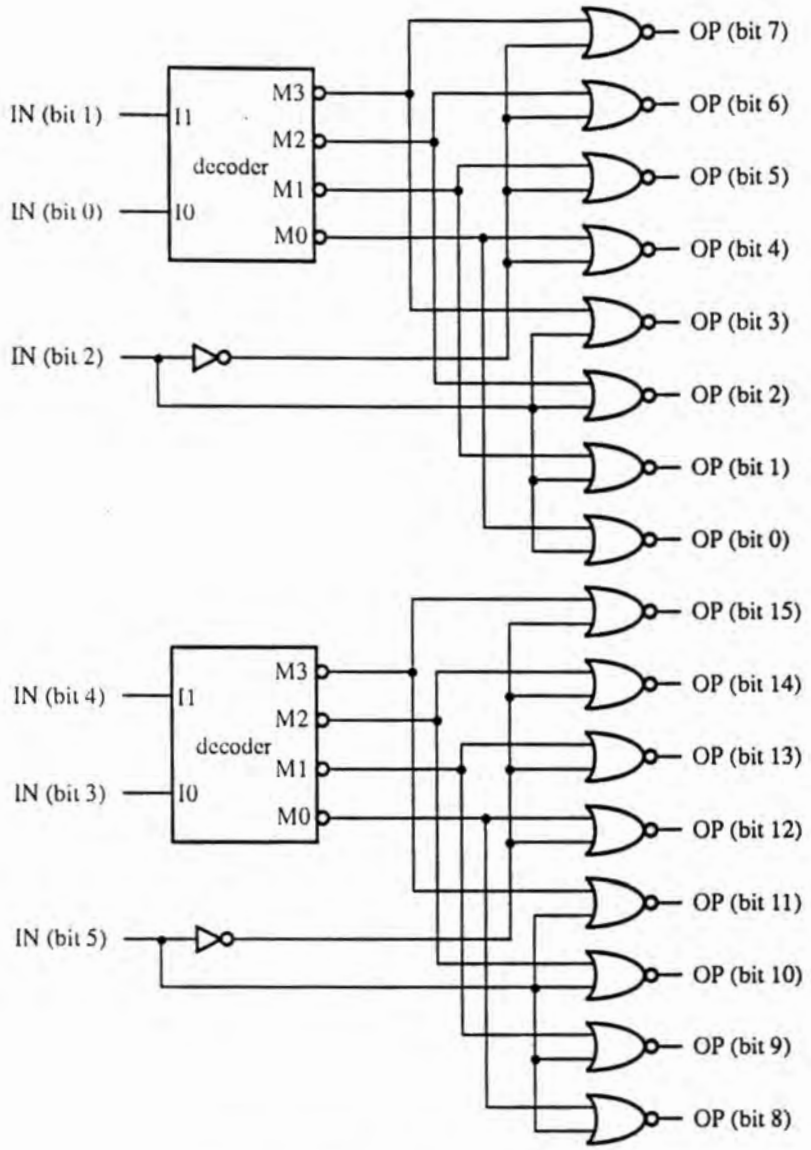


Figure A.21: Dual_dec38 Block

A.22 S1_wiring to S8_wiring Blocks

These wiring blocks do not contain any gates. Each output bit is connected to one of the input bits. Each pair of output bits specifies a minterm of the 6 S-box inputs. Each group of 8 output bits specifies the 4 minterms for one of the non-zero S-box table entry values. The following figure shows the input bit numbers for each output bit starting from the most significant output bit.

S1_wiring

```
2  9  3  8  0 14  1 12  0  8  1  9  4 12  7 14  4  8  5  9  0 13  7 15  6 10  5 10  2 14
3 12  4  9  7 10  6 13  3 14  2 10  1 10  2 15  1 15  2 11  1 11  4 14  3 13  6  9  7 11
6 12  5 12  6 11  5  8  6 14  7 13  4 10  3 10  2 13  5 15  0 11  3 11  4 15  1 14  2  8
7  8  0 12  1 13  0 10  5 11  0 15  5 14  0  9  3  9  4 13  7 12  6  8  7  9  2 12  5 13
```

S2_wiring

```
0  8  1  9  6 15  3 13  6  8  7  9  2 12  5 15  6 10  3  8  4 13  1 12  0 11  1 10  4 14
7 14  2  9  5 11  6 12  1 14  6 11  7 10  0 13  5 12  0 10  3 11  0 15  7 15  4  8  5  9
2 14  3 12  2 10  7  8  4 12  5 14  0  9  1 11  6 14  3 14  4 11  7 11  0 14  3 15  6  9
5  8  2 13  5 13  4  9  1  8  2 15  1 13  4 10  3  9  4 15  7 13  2  8  5 10  6 13  7 12
```

S3_wiring

```
4  9  5 11  2 13  3 14  6  8  7 10  4 15  5 14  2 10  1  8  0 12  5 12  4 10  1 11  6 14
7 15  0 11  3 11  0 14  1 15  0  8  7  9  2 15  3 12  4  8  7  8  6 12  3 13  6 11  3 10
0 13  5 13  6 10  3  8  6 15  7 13  0  9  5  9  2 12  1 13  6  9  5 10  0 15  3 15  2 11
3  9  4 12  1 14  2  9  1  9  4 13  7 14  4 11  1 10  4 14  5 15  0 10  7 11  2 14  1 12
```

S4_wiring

```
6 11  3  9  0 14  3 12  4  8  5 11  6 14  7 15  2  8  1  8  6 13  5 13  2 11  7 10  0 13
1 15  0 11  5  8  2 13  7 14  6  9  3 11  0 12  1 13  4  9  7 11  4 12  1 14  4 10  3  8
4 15  7 13  0  8  3 10  4 13  3 15  2  9  1  9  2 12  7 12  6 10  7  8  0 15  5 14  4 11
1 10  6 15  3 14  6  8  7  9  4 14  1 12  2 10  5 10  2 15  5 15  0 10  1 11  2 14  3 13
```

S5_wiring

```
6 10  5 10  0 14  3 14  4 11  1  8  6 15  3 13  0 11  5  9  2 13  7 13  2  8  7  8  4 14
5 12  4  9  3  8  6 12  1 12  2  9  7 10  0 13  1 15  6 11  3 11  2 14  7 14  0 10  5 11
6 13  3 12  0  9  3  9  4 13  7 12  6  9  7 11  0 15  1 14  2 10  1 10  6 14  5 15  4  8
1  9  0 12  3 15  4 10  1 11  2 15  7 15  0  8  5  8  2 12  5 13  6  8  7  9  4 12  1 13
```

S6_wiring

```
6  8  3  8  4 12  5 13  0 11  7 10  2 12  3 14  2 10  5 10  2 15  7 15  0  8  3  9  4 13
7 12  6 11  3 11  4 15  1 14  4  8  1  8  6 14  7 13  0  9  5  9  0 12  1 13  6  9  7 11
2 13  5 15  2 11  1  9  0 14  7 14  4  9  1 10  6 15  1 15  4 11  7  9  6 12  3 13  6 10
5  8  4 14  1 12  4 10  5 11  6 13  3 12  2  9  7  8  0 13  5 12  2  8  3 10  0 15  5 14
```

S7_wiring

```
0  9  3 11  2 14  7 14  6  8  1 10  6 13  1 15  6  9  1  8  6 12  5 12  2 10  7 10  0 13
7 15  2  8  5  8  4 12  3 12  2 11  7  9  0 14  5 13  4 10  3  9  4 15  1 14  4  9  5 11
6 14  7 12  6 10  7  8  4 13  7 13  4 11  7 11  4 14  1 12  0 11  5 10  2 15  3 14  0  8
1  9  2 12  3 13  0 10  3 10  2 13  5 15  4  8  1 11  6 15  3 15  6 11  5  9  0 12  1 13
```

S8_wiring

```
2  9  3  8  0 15  1 14  6 10  3 11  4 13  5 12  0  8  5  8  6 14  7 13  4 11  1 10  2 13
3 14  4  9  7 10  2 12  7 15  0 10  1  9  4 14  3 13  2 10  5 11  0 13  5 14  4  8  7  8
6 15  5 13  6 11  5  9  0 12  7 12  0  9  5 10  2 14  5 15  0 11  3 10  4 15  3 15  6  8
7  9  4 12  1 13  4 10  3  9  2 15  1 15  2  8  7 11  6 13  1 12  6  9  1  8  6 12  3 12
```

Figure A.22: S1_wiring to S8_wiring Blocks

A.23 S_output Block

The S_output block takes the permuted version of the decoded S-box inputs and produces the S-box outputs. The first 8 input bits are used to produce the signal associated with an S-box table entry of 1. The next 8 bits are used for a table entry of 2, and so on to a table entry of 15. The signals for S-box table entries are then combined to form the S-box outputs. Output bit 0 is

created by or-ing the odd table entry signals. Output bit 1 is created by or-ing table entry signals 2, 3, 6, 7, 10, 11, 14, and 15. Output bit 2 is created by or-ing table entry signals 4, 5, 6, 7, 12, 13, 14, and 15. Output bit 3 is created by or-ing table entry signals 8 to 15.

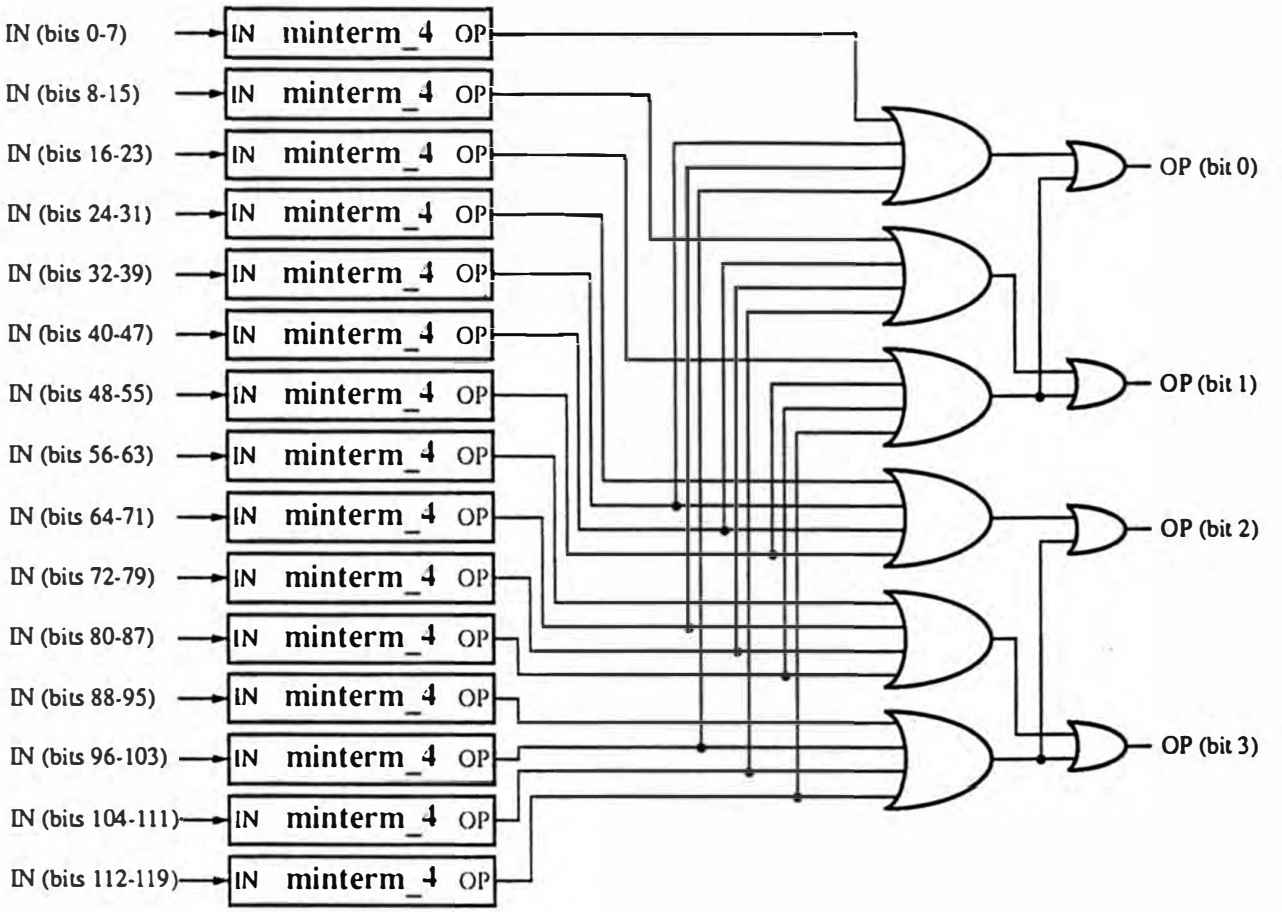


Figure A.23: S_output Block

A.24 Minterm_4 Block

The input to the minterm_4 block is 4 pairs of signals. The signals in each pair are and-ed together, and the resulting 4 signals are or-ed together.

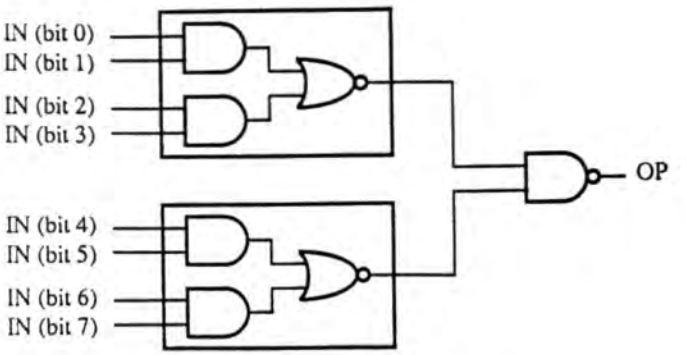


Figure A.24: Minterm_4 Block

A.25 Register_32 Block

The register_32 block consists of 32 flip-flops.

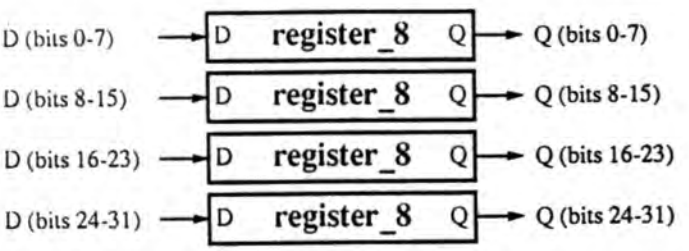


Figure A.25: Register_32 Block

A.26 Register_8 Block

The register_8 block consists of 8 flip-flops.

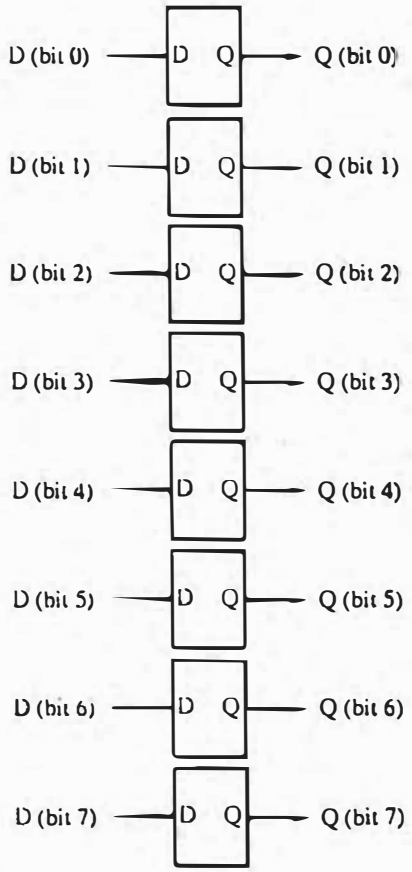


Figure A.26: Register_8 Block

A.27 Xor_48 Block

The xor_48 block exclusive-ors 48 pairs of signals together.

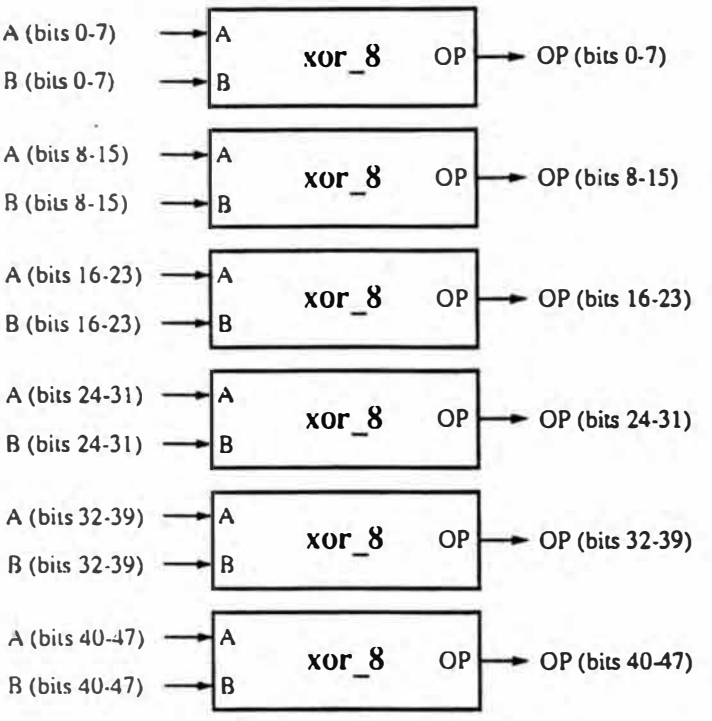


Figure A.27: Xor_48 Block

A.28 Xor_32 Block

The xor_32 block exclusive-ors 32 pairs of signals together.

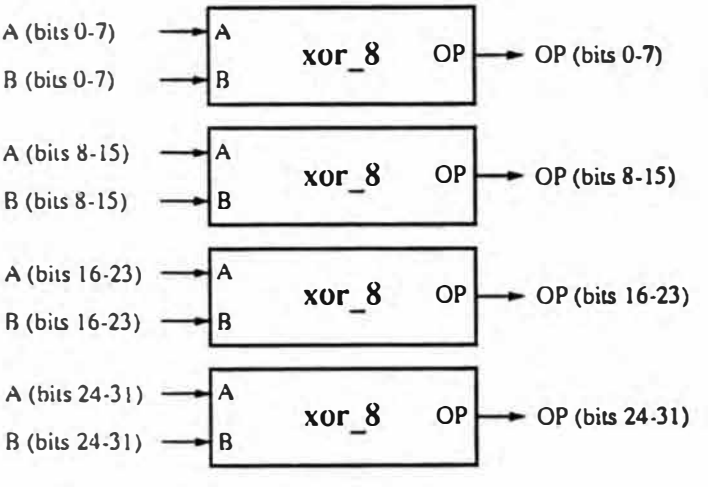


Figure A.28: Xor_32 Block

A.29 Xor_8 Block

The xor_8 block exclusive-ors 8 pairs of signals together.

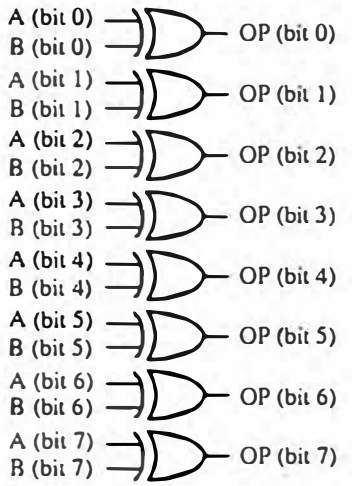


Figure A.29: Xor_8 Block

A.30 E_perm Block

This block does not contain any gates. Each output bit is connected to one of the input bits. The E permutation is one of the elements of DES. The following figure shows the input bit numbers for each output bit starting from the most significant output bit.

0	31	30	29	28	27	28	27	26	25	24	23	24	23	22	21	20	19	20	19	18	17	16	15
16	15	14	13	12	11	12	11	10	9	8	7	8	7	6	5	4	3	4	3	2	1	0	31

Figure A.30: E_perm Block

A.31 P_perm Block

This block does not contain any gates. Each output bit is connected to one of the input bits. The P permutation is one of the elements of DES. The following figure shows the input bit numbers for each output bit starting from the most significant output bit.

16	25	12	11	3	20	4	15	31	17	9	6	27	14	1	22	30	24	8	18	0	5	29	23	13	19	2	26	10	21	28	7

Figure A.31: P_perm Block

A.32 Key_round_1 Block

This block performs one round of the key scheduling part of DES. There are actually two types of key rounds. They only differ in the number of rotates performed at a certain point in the block. In this block, the 56-bit input key K is split into two 28-bit halves, each is rotated left one position, and the halves are rejoined to form the value which goes into the 56-bit pipeline register. The 48-bit sub-key output SK is formed by applying the PC2 permutation to the rotated key.

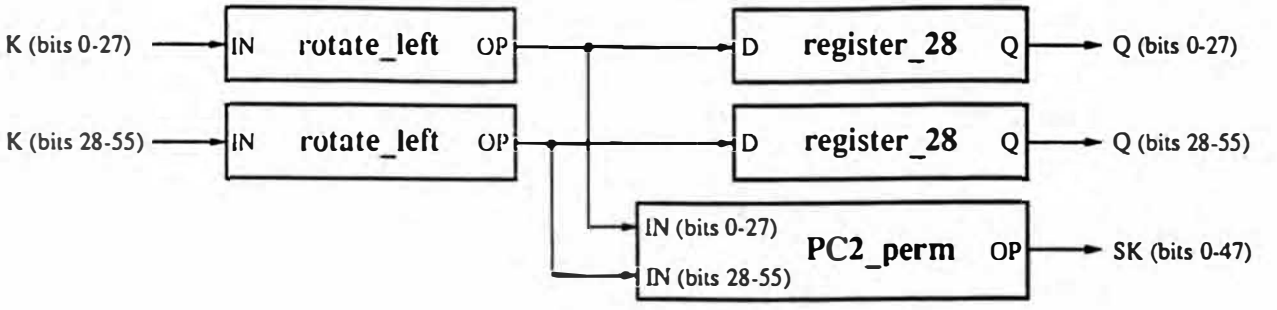


Figure A.32: Key_round_1 Block

A.33 Key_round_2 Block

This block performs one round of the key scheduling part of DES. There are actually two types of key rounds. They only differ in the number of rotates performed at a certain point in the block. In this block, the 56-bit input key K is split into two 28-bit halves, each is rotated left two positions, and the halves are rejoined to form the value which goes into the 56-bit pipeline register. The 48-bit sub-key output SK is formed by applying the PC2 permutation to the rotated key.

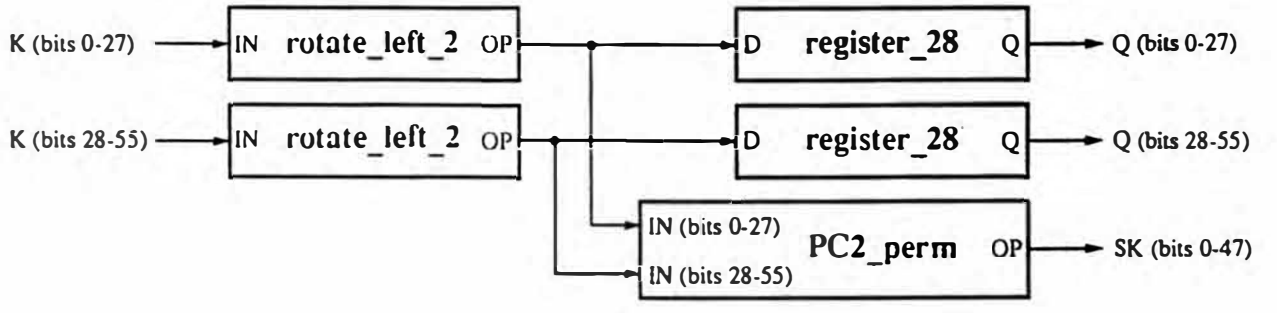


Figure A.33: Key_round_2 Block

A.34 Register_28 Block

The register_28 block consists of 28 flip-flops.

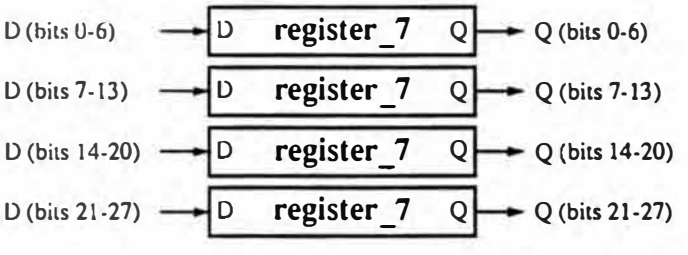


Figure A.34: Register_28 Block

A.35 Register_7 Block

The register_7 block consists of 7 flip-flops.

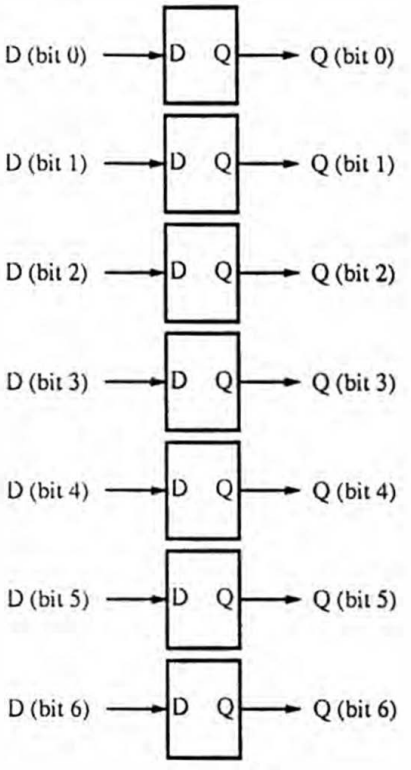


Figure A.35: Register_7 Block

A.36 PC2_perm Block

This block does not contain any gates. Each output bit is connected to one of the input bits. The PC2 permutation is one of the elements of DES. The following figure shows the input bit numbers for each output bit starting from the most significant output bit.

42	39	45	32	55	51	53	28	41	50	35	46	33	37	44	52	30	48	40	49	29	36	43	54
15	4	25	19	9	1	26	16	5	11	23	8	12	7	17	0	22	3	10	14	6	20	27	24

Figure A.36: PC2_perm Block

A.37 Rotate_left Block

This block does not contain any gates. Each output bit is connected to one of the input bits. The 28 input lines are rotated left one position. The following figure shows the input bit numbers for each output bit starting from the most significant output bit.

26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 27

Figure A.37: Rotate_left Block

A.38 Rotate_left_2 Block

This block does not contain any gates. Each output bit is connected to one of the input bits. The 28 input bits are rotated left 2 positions. The following figure shows the input bit numbers for each output bit starting from the most significant output bit.

25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 27 26

Figure A.38: Rotate_left_2 Block

A.39 Initial_perm Block

This block does not contain any gates. Each output bit is connected to one of the input bits. The initial permutation (IP) is one of the elements of DES. The following figure shows the input bit numbers for each output bit starting from the most significant output bit.

6 14 22 30 38 46 54 62 4 12 20 28 36 44 52 60 2 10 18 26 34 42 50 58 0 8 16 24 32 40 48 56
7 15 23 31 39 47 55 63 5 13 21 29 37 45 53 61 3 11 19 27 35 43 51 59 1 9 17 25 33 41 49 57

Figure A.39: Initial_perm Block

A.40 Inverse_initial Block

This block does not contain any gates. Each output bit is connected to one of the input bits. This permutation is not exactly the inverse of initial_perm. It also swaps the left and right halves. The following figure shows the input bit numbers for each output bit starting from the most significant output bit.

56 24 48 16 40 8 32 0 57 25 49 17 41 9 33 1 58 26 50 18 42 10 34 2 59 27 51 19 43 11 35 3
60 28 52 20 44 12 36 4 61 29 53 21 45 13 37 5 62 30 54 22 46 14 38 6 63 31 55 23 47 15 39 7

Figure A.40: Inverse_initial Block

A.41 PC1_perm Block

This block does not contain any gates. Each output bit is connected to one of the input bits. The PC1 permutation is one of the elements of DES. The following figure shows the input bit numbers for each output bit starting from the most significant output bit.

```
 6  13  20  27  34  41  48  55   5  12  19  26  33  40  47  54   4  11  18  25  32  39  46  53   3  10  17  24
 0   7  14  21  28  35  42  49   1   8  15  22  29  36  43  50   2   9  16  23  30  37  44  51  31  38  45  52
```

Figure A.41: PC1_perm Block

A.42 Inverse_PC1 Block

This block does not contain any gates. Each output bit is connected to one of the input bits. This block performs the inverse of the PC1_perm block. The following figure shows the input bit numbers for each output bit starting from the most significant output bit.

```
48  40  32   0   4  12  20  49  41  33   1   5  13  21  50  42  34   2   6  14  22  51  43  35   3   7  15  23
52  44  36  28   8  16  24  53  45  37  29   9  17  25  54  46  38  30  10  18  26  55  47  39  31  11  19  27
```

Figure A.42: Inverse_PC1 Block

A.43 External_interface Block

The external interface block is the interface between an external processor and the core of the key search chip. Included in the external_interface are 27 pads (not all of which are shown in Figure A.43) which serve as a buffer between the external pins and the inside of the chip. There are input pads for the chip enable, read/write line, tri-state line, and the 5 address lines. There are 8 input/output pads for the data lines. Not shown in Figure A.43 are the pad for the clock signal, 5 power pads, and 5 ground pads.

The flip-flops at the top of the block produce delayed versions of the chip enable. The chip enable is delayed to synchronize it to the chip's clock and to give the other signals coming into the chip time to become valid. When the delayed enable line makes a transition from 1 to 0 and the read/write line is a 0, then data is being written into the chip and address lines 3 and 4 select one of the write lines (write_s, write_kc, write_c, and write_p) to become a 1. Address lines 0, 1, and 2 are passed through as outputs from the external_interface. If the undelayed enable line is 0, the read/write line is a 1, and the tri-state line is a 1, then data is being read from the chip and the input/output pads on the data lines are set as outputs. The purpose of the tri-state line is to force the chip into an input mode to allow testing of a board. When data is being read from the chip and address lines 3 and 4 are both 1, then the stop bit is put out on data pin 0 and the rest of the data pins are 0. When data is being read from the chip, address line 3 is a 0, and address line 4 is a 1, then the output from the output_key block goes out on the data pins. A read from any other address causes zeros to be put out on the data pins. When data is written into the chip, the signals on the data pins are passed through to the data_out signals.

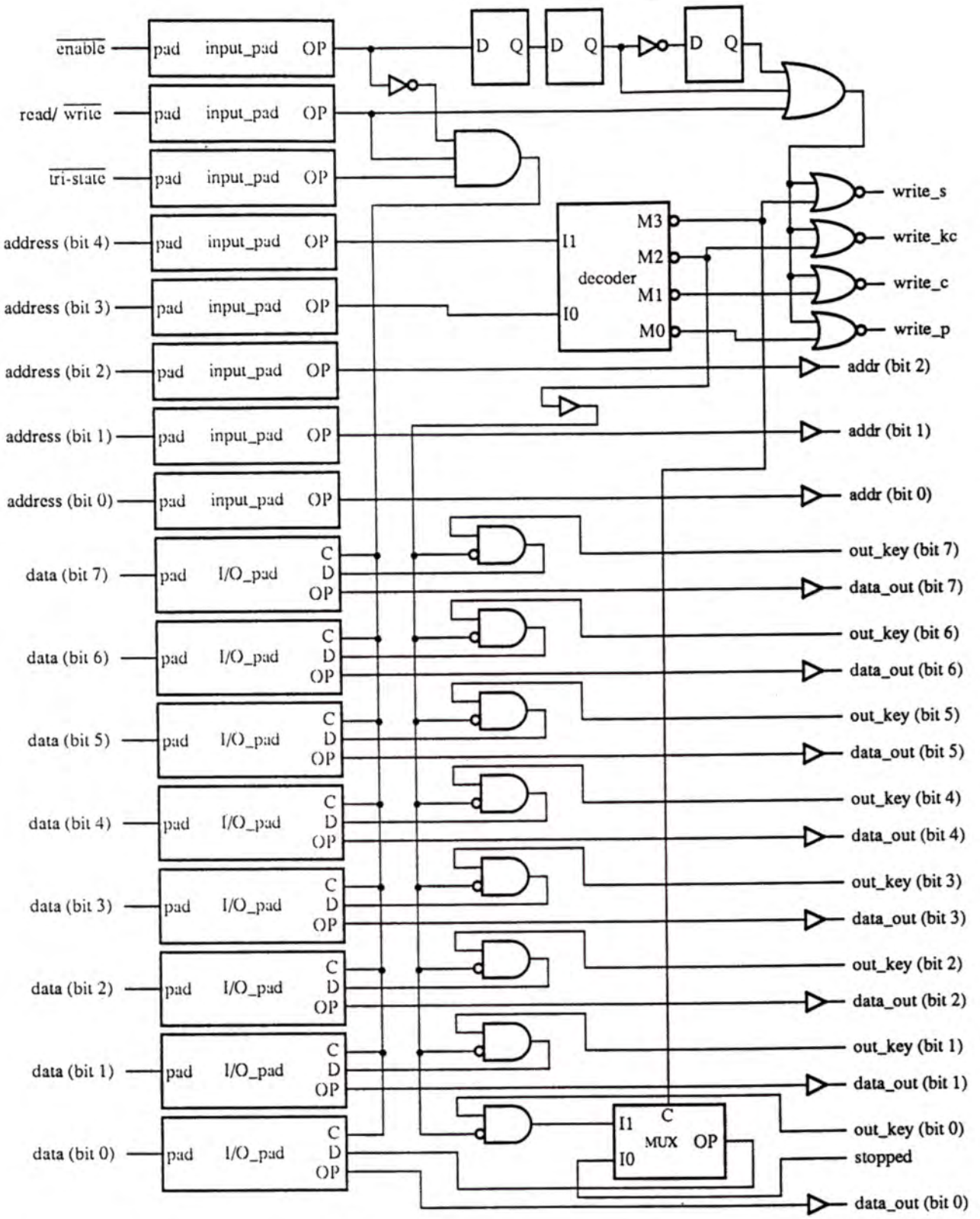
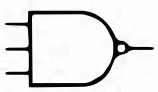
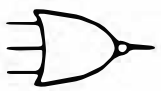
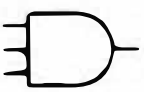
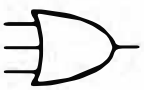
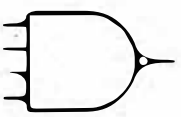
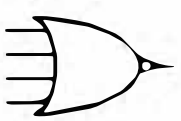


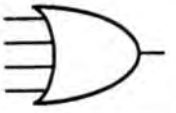
Figure A.43: External_interface Block

A.44 Standard Cells

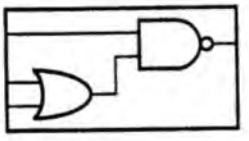
This section contains descriptions of each of the standard cells, their symbols, and their areas (in units of sites, where a site is 3.6 μm × 54.4 μm).

	2 sites	inverter
	3 sites	buffer
	3 sites	2-input nand gate
	3 sites	2-input nor gate
	4 sites	2-input and gate
	4 sites	2-input or gate
	4 sites	2-input exclusive-or gate
	5 sites	2-input exclusive-nor gate
	4 sites	2-input and gate with one inverted input
	4 sites	2-input or gate with one inverted input
	4 sites	3-input nand gate
	4 sites	3-input nor gate
	5 sites	3-input and gate
	5 sites	3-input or gate
	5 sites	4-input nand gate
	7 sites	4-input nor gate

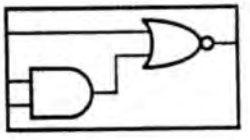
6 sites 4-input and gate



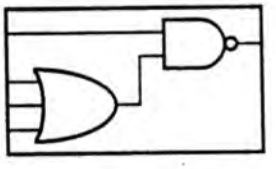
6 sites 4-input or gate



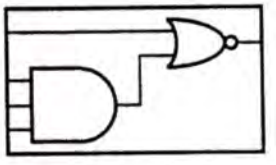
4 sites composite gate: 1,2 or-and-invert



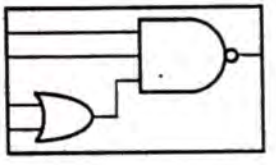
4 sites composite gate: 1,2 and-or-invert



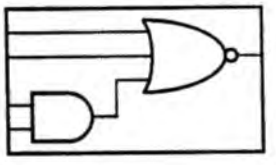
5 sites composite gate: 1,3 or-and-invert



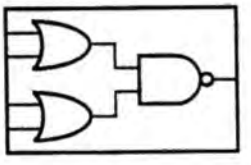
5 sites composite gate: 1,3 and-or-invert



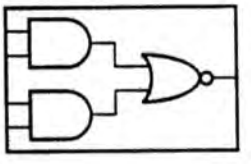
5 sites composite gate: 1,1,2 or-and-invert



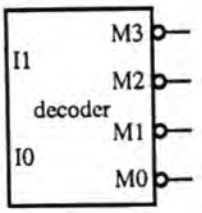
5 sites composite gate: 1,1,2 and-or-invert



6 sites composite gate: 2,2 or-and-invert



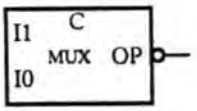
6 sites composite gate: 2,2 and-or-invert



6 sites

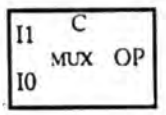
2-input inverting decoder:

I1	I0	M3	M2	M1	M0
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	1



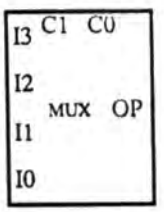
5 sites

2-input inverting multiplexor:
if C = 1, then OP = $\overline{I1}$
if C = 0, then OP = $\overline{I0}$



7 sites

2-input multiplexor:
if C = 1, then OP = I1
if C = 0, then OP = I0

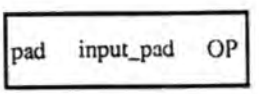


19 sites

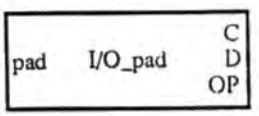
4-input multiplexor:
if C1 = 1 and C0 = 1, then OP = I3
if C1 = 1 and C0 = 0, then OP = I2
if C1 = 0 and C0 = 1, then OP = I1
if C1 = 0 and C0 = 0, then OP = I0

10 sites

flip-flop:
Q holds its value until the rising edge of the system-wide clock when Q takes the value of the input D.



An input pad is a buffer between an external input pin and the inside of the chip. The output is simply equal to the pad input. If the pad input is floating, the output is 1.



An I/O pad is a buffer between an external I/O pin and the inside of the chip. If C = 1, the pad is an output and the logic level at D is output. If C = 0, the pad is an input and OP is equal to the pad input. If C = 0 and the pad input is floating, then OP = 1.

Appendix B: Key Search Board Design

The main goal in designing a key search board is to pack as many key search chips onto it as possible with a minimum of overhead and additional cost. The majority of the overhead is microcontrollers to control the key search chips. The microcontroller selected for this design is the Motorola MC68HC705C8 [13]. The main features of this member of the 6805 family are 8192 bytes of one-time programmable ROM, 176 bytes of RAM, a serial peripheral interface (SPI), a serial communications interface (SCI), 24 general-purpose I/O pins, and it is available in

a 44-pin PLCC (surface-mount) package. The masked ROM version of this chip is less expensive, but the one-time programmable version was selected to avoid masking charges.

Each microcontroller can control 10 key search chips using its 24 I/O pins. Eight data lines, five address lines, and a read/write line are connected to all 10 key search chips. Each of the remaining 10 I/O pins goes to the enable pin of one of the key search chips.

The board needs a master microcontroller (also an MC68HC705C8) to control all of the other microcontrollers. The master is connected to its slaves using the serial peripheral interface. The SPI is designed to connect a master to several slaves directly without any other logic. The master microcontroller's serial communications interface is used as the board's external interface.

Other components that will be required are an octal buffer/line driver (74HC244) for the board's external interface lines, a 50 MHz oscillator, clock drivers to drive the clock to the key search chips, a counter (74HC191) to divide the clock by 16 to produce a suitable clock for the microcontrollers, a clock driver for the microcontrollers, decoupling capacitors on the underside of the board, a fuse for the power entry to the board, and the unpopulated board itself.

For the shelf and frame technology that this system is designed for, the usable area of each board is 9 inches × 13 inches. Using surface-mount components for everything except the fuse, there is enough room on the board for 120 key search chips.[1] To handle this number of chips, 5 clock drivers and 13 microcontrollers are needed. A possible layout of this board is shown in Figure B.1. The unlabeled chips are the key search chips. The layout shows connections between certain chips indicating the control of the slave microcontrollers by the master and the control of key search chips by each slave.

The board itself is a 6-layer board. There is one layer for each of power and ground. The remaining 4 layers are for routing clocks and the connections between chips. Because most of the connections are very localized, it may be possible to use a 4-layer board, but we will assume that a 6-layer board is necessary. The cost of fabricating, populating, and testing the board is about $300.

[1] A 44-pin PLCC package covers 0.72 inch × 0.72 inch, and 0.2 inch spacing between chips is required.

The following table shows the costs to build this board.

Component	Quantity	Unit Cost ($)	Extended ($)
key search chip	120	10.50	1260
MC68HC705C8	13	8	104
50 MHz oscillator	1	4	4
clock driver	5	8	40
74HC191	1	1	1
74HC244	1	1	1
decoupling capacitors and fuse			15
board and cost to populate and test it			300
Total Board Cost			$ 1725

The total power consumption of all of the key search chips is 120×450 mW = 54 W. Adding the power consumption of the other components brings the total board power to 60 W. This is fairly high power consumption. The issues of providing this power and cooling the boards will need to be addressed at the shelf and frame level.

The program in the slave microcontroller is not very complicated. Through its SPI it receives tasks which consist of a plaintext P, a ciphertext C, a starting key K, and chip step value s. The chip step value specifies the spacing between starting keys for each of the 10 key search chips. If each chip is responsible for m keys, then the chip step value supplied to the slave microcontroller is $s = x^m \bmod g$, where g is the generator polynomial. The 10 starting keys are K, $Ks \bmod g$, $Ks^2 \bmod g$, ..., $Ks^9 \bmod g$. To get from one chip's starting key to the next, the slave microcontroller performs a polynomial multiply of the previous starting key and the chip step value and reduces the result modulo the key_counter's generator polynomial g. This is a fairly simple calculation. After starting the key search chips, the slave polls them until one of them has the desired key or a new task comes in. Any found keys are reported back to the master. The slave should also be capable of performing a simple known-answer test on each of its key search chips and reporting any failures to the master.

The master must be able to take tasks through its SCI which consist of P, C, starting key K, slave step value t, and chip step value s. The values P, C, and s are passed directly to each slave. The starting keys sent to the 12 slaves are K, $Kt \bmod g$, $Kt^2 \bmod g$, ..., $Kt^{11} \bmod g$. The master should also be capable of performing a known-answer test on each of its slaves. The master reports any found keys or hardware failures over its SCI.

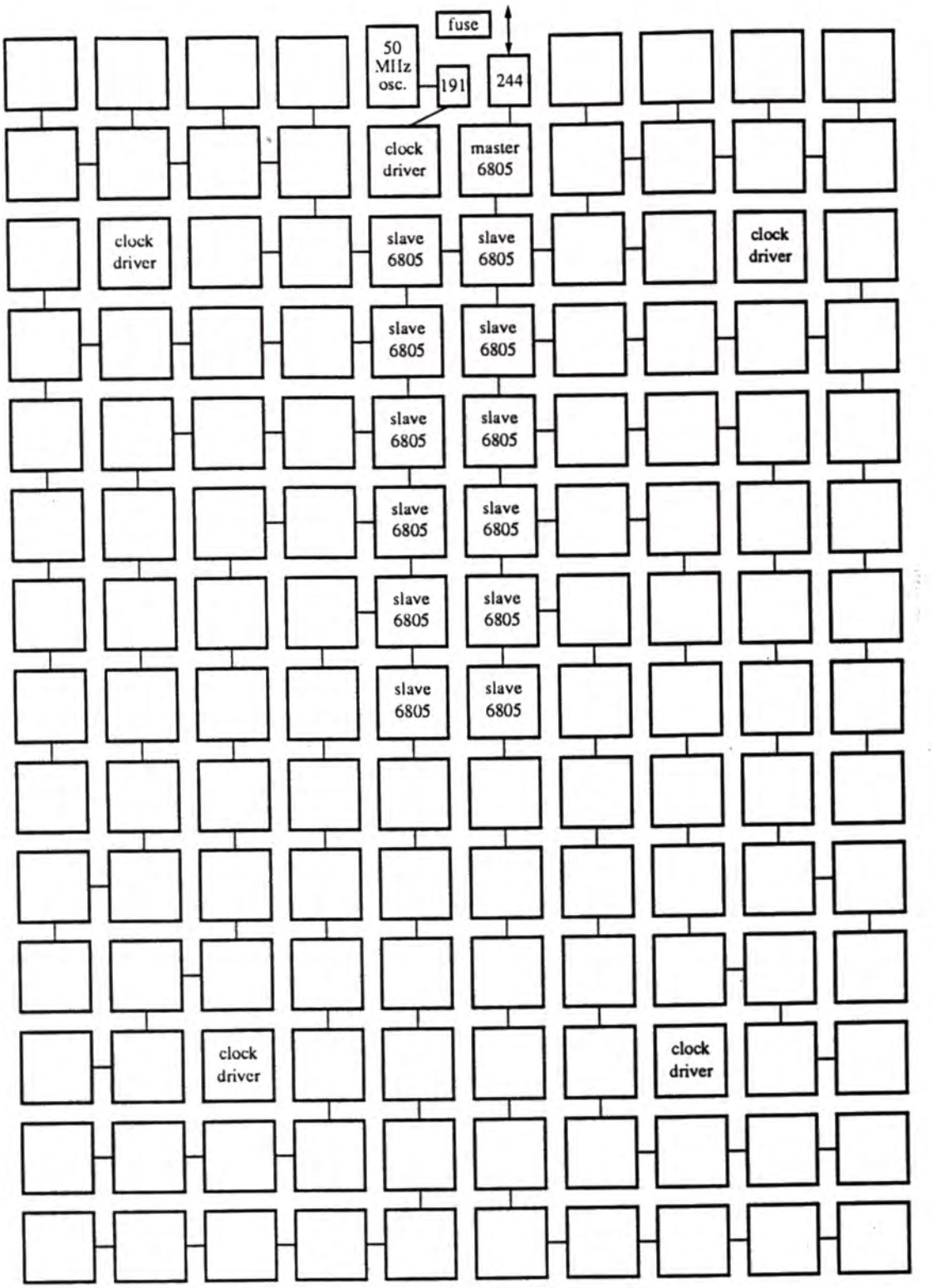


Figure B.1: Key Search Board Layout

Appendix C: Key Search Frame Design

Given that we have key search boards as described in Appendix B, we need a structure to hold, interconnect, and control all of these boards. An existing technology for this purpose was selected as the basis for this design. Boards are housed in frames. Each frame consists of 4 stacked shelves, and each shelf has 26 board slots and 4 slots for power supplies.

Each shelf requires a board to control the key search boards. This controller board needs only a processor and sufficient logic to interface serially to the key search boards. The controller boards could be made for about $500 each. Although there are 25 slots left in the shelf, the total power requirements for 25 key search boards at 60 W each is too much for the 4 shelf power supplies. To stay comfortably within the power supply limits and leave more space for power dissipation, only 12 key search boards are on each shelf. The following table shows the cost of a shelf.

Component	Quantity	Unit Cost ($)	Extended ($)
key search board	12	1725	20700
controller board	1	500	500
power supply	4	300	1200
mechanical structure, board interconnections, testing		1000	1000
Total Shelf Cost			$ 23400

Each frame consists of 4 shelves and fans for cooling. The frame also requires a PC to control the 4 shelf controller boards and to check the all-zero key. The following table shows the frame cost.

Component	Quantity	Unit Cost ($)	Extended ($)
key search shelf	4	23400	93600
mechanical structure, fans, testing		4000	4000
PC	1	2000	2000
Total Frame Cost			$ 99600

A frame costs about $100 000 and contains 5760 key search chips each of which can test 50 million DES keys per second. Overall, a frame can test 2.88×10^{11} keys per second.

Appendix D: Reliability of the Key Search Machine

A common criticism of past key search designs is that they are not sufficiently reliable to complete a single key search without breaking down. Today, this is no longer a concern. To show that the key search machine described in this paper is sufficiently reliable, we will calculate its mean time between failures (MTBF). All information on component reliability was obtained through internal BNR component testing.

The typical failure rate of chips designed in the CMOS process used for the key search chip is 10 FITs.[1] The actual failure rate for the key search chip may be somewhat less because it is small and contains no analog circuitry, but we will use a failure rate of 10 FITs. The total failure rate for a key search board is shown in the following table.

Component	Quantity	Unit Reliability (FITs)	Total Reliability (FITs)
key search chip	120	10	1200
MC68HC705C8	13	20	260
50 MHz oscillator	1	3	3
clock driver	5	10	50
74HC191	1	4	4
74HC244	1	4	4
fuse	1	3	3
decoupling capacitors	400	0.1	40
board	1	100	100
Total Board Failure Rate			1664

A key search shelf consists of 12 key search boards, a controller board and four power supplies. The controller board and power supplies each contain fewer components than a key search board, and so we will estimate their failure rates at 1000 FITs. The total failure rate for a shelf is shown in the following table.

Component	Quantity	Unit Reliability (FITs)	Total Reliability (FITs)
key search board	12	1664	19968
controller board	1	1000	1000
power supply	4	1000	4000
board interconnections	1	100	100
Total Shelf Failure Rate			25068

A key search frame consists of four shelves and a PC. We will estimate a PC's failure rate at 5000 FITs. The total failure rate for a frame is shown in the following table.

Component	Quantity	Unit Reliability (FITs)	Total Reliability (FITs)
key search shelf	4	25068	100272
PC	1	5000	5000
Total Frame Failure Rate			105272

[1] FIT is short for Failure unIT. One FIT is equal to one failure in 10^9 hours.

The failure rate for a frame is one failure every 9500 hours. For a machine consisting of n frames, the MTBF is $9500/n$ hours. In particular, a $1 million machine consisting of 10 frames would have an MTBF of 950 hours or about 40 days. The expected time to find a DES key using n frames is $35/n$ hours. Therefore, we expect one failure every 270 keys found regardless of the machine's size. This failure rate is much lower than is necessary to get useful key search results.

Appendix E: Support for 1-bit and 8-bit Cipher Feedback

The most popular modes of DES which are not handled by the key search chip design in Appendix A are the 1-bit and 8-bit cipher feedback (CFB) modes. A number of changes to the basic chip design would have to be made in order to support these modes. For the following discussion it will be useful to refer to Figure A.1.

From the external microcontroller's point of view, the main changes would be that it would have to specify which mode to use (ECB, 1-bit CFB, or 8-bit CFB), and for the CFB modes the microcontroller would have to specify the initialization vector (IV) value which immediately preceded the encryption of the known plaintext. We will assume that there are still 64 consecutive bits of known plaintext and ciphertext that are supplied to the key search chip. Within the chip, additional registers are needed to hold the mode and IV, and additional write logic is required in the external_interface.

The strategy for the CFB modes is to begin by taking each key and performing the encryption for the first bit or byte of plaintext. If there is no match with the corresponding bit or byte of ciphertext, then go on to the next key. If there is a match, then the key_counter must be stopped for one clock tick, the key from the bottom of the DES pipeline must be inserted into the top of the pipeline, and the next IV for CFB must be computed and fed into the plaintext input at the top of the DES pipeline. To keep track of the number of matches there have been for each key, we need an extra 6 flip-flops in each stage of the pipeline. Not until there has been a full 64 bits of matching with the ciphertext register will a key be declared found and stored in the output_key register.

We will require multiplexors at the inputs to the DES pipeline to choose between a new key and continuing with an old key. This adds too much delay to the first pipeline stage making it necessary to add an additional pipeline register at the beginning of the DES pipeline.

All of this additional circuitry enlarges the key search chip by 2000 gates. This increases the chip's cost from $10.50 to $12, which increases the cost of a key search frame to $108 240.

References

[1] E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", *Journal of Cryptology*, vol. 4, no. 1, 1991, pp. 3-72.

[2] E. Biham and A. Shamir, "Differential Cryptanalysis of the full 16-round DES", *Lecture Notes in Computer Science: Advances in Cryptology - Crypto '92 Proceedings*, Springer-Verlag, to appear.

[3] K.W. Campbell and M.J. Wiener, "DES is not a Group", *Lecture Notes in Computer Science: Advances in Cryptology - Crypto '92 Proceedings*, Springer-Verlag, to appear.

[4] "Data Encryption Standard", National Bureau of Standards (U.S.), Federal Information Processing Standards Publication (FIPS PUB) 46, National Technical Information Service, Springfield VA, 1977.

[5] "DES Modes of Operation", National Bureau of Standards (U.S.), Federal Information Processing Standards Publication (FIPS PUB) 81, National Technical Information Service, Springfield VA, 1981.

[6] W. Diffie and M. Hellman, "Exhaustive Cryptanalysis of the NBS Data Encryption Standard", *Computer*, vol. 10, no. 6, June 1977, pp. 74-84.

[7] H. Eberle, "A High-Speed DES Implementation for Network Applications", *Lecture Notes in Computer Science: Advances in Cryptology - Crypto '92 Proceedings*, Springer-Verlag, to appear.

[8] G. Garon and R. Outerbridge, "DES Watch: An examination of the Sufficiency of the Data Encryption Standard for Financial Institution Information Security in the 1990's", *Cryptologia*, vol. XV, no. 3, July 1991, pp. 177-193.

[9] M. Hellman, "DES will be totally insecure within 10 years", *IEEE Spectrum*, vol. 16, July 1979, pp. 32-39.

[10] F. Hoornaert, J. Goubert, and Y. Desmedt, "Efficient hardware implementation of the DES", *Lecture Notes in Computer Science: Advances in Cryptology - Crypto '84 Proceedings*, Springer-Verlag, pp. 147-173.

[11] B. Kaliski, R. Rivest, and A. Sherman, "Is the Data Encryption Standard a Group? (Results of Cycling Experiments on DES)", *Journal of Cryptology*, vol. 1, no. 1, 1988, pp. 3-36.

[12] M. Matsui, "Linear Cryptanalysis Method for DES Cipher", *Lecture Notes in Computer Science: Advances in Cryptology - Eurocrypt '93 Proceedings*, Springer-Verlag, to appear.

[13] "MC68HC705C8 Technical Data", Motorola Inc., 1989.

[14] P.C. van Oorschot and M.J. Wiener, "A Known-Plaintext Attack on Two-Key Triple Encryption", *Lecture Notes in Computer Science: Advances in Cryptology - Eurocrypt '90 Proceedings*, Springer-Verlag, pp. 318-325.

[15] P. Wayner, "Content-Addressable Search Engines and DES-like Systems", *Lecture Notes in Computer Science: Advances in Cryptology - Crypto '92 Proceedings*, Springer-Verlag, to appear.