Last Revised: 10.3.96.

ECS 227 — Modern Cryptography — Fall 96
Lecture 2

The Data Encryption Standard

1 October 1996

Reading for Lecture 2:

[**Co92**] Don Coppersmith, "The Data Encryption Standard (DES) and its Strength Against Attacks." IBM RC 18613. *A nice article; we covered only the section on why DES generated a large group.*

[**BiSh93**] Eli Biham and Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard."* Springer-Verlag, 1993. *I gave out the appendix, so you could see the complete description of DES. The book itself is recommended to anyone interested in cryptanalysis.*

2 Confusion/Diffusion Primitives

Last time we talked about provable security. Provable security is a *reductionist* methodology: *assuming* some underlying cryptographic primitive does what it's supposed to do, we *prove* (by making correct use of that primitive) that some (higher-level) cryptographic goal can be accomplished.

We could go through our work this quarter and never describe —concretely— what sort of low-level primitives we might employ. But this would make everything we do seem a bit too abstract. You'd have little idea how to carry through our ideas to practice. Thus we spend today's lecture taking a look at the sort of primitives which can serve as our reduction targets.

We'll look at two different primitives: the *Data Encryption Standard* (DES), and the *Secure Hash Algorithm* (SHA). The first is a type of object called a *block cipher*; the second is a type of object called a *collision-intractable hash function.* Later in the term we'll consider a third primitive, the *Rivest-Shamir-Adleman*

Function (RSA). It's a type of object called a *trapdoor permutation.* For now, we won't be describing what these primitives are for, or how to use them; we'll just define them.

2.1 Two Breeds of Primitives

Concrete primitives of contemporary cryptography come in two basic flavors:

1. **Confusion/Diffusion Primitives.** DES and SHA are examples. I've heard these primitives called *bit-twiddling constructions.* That may sound a bit flippant, but, in fact, it aptly describes the flavor of their definitions. Typically one uses straight-line code of simple Boolean operations, trying to create computational hardness by doing enough "strange stuff."

2. **Algebraic/Combinatorial Primitives.** RSA and knapsack-based ciphers are two examples. Here one starts with a "known" computational problem and uses it as the basis for cryptography. Mathematical domains which have been used to provide these computational problems include number theory, computational group theory, automata theory, and the theory of error correcting codes. Number theory seems to get most of the attention.

Some theoretical computer scientists have little regard for the confusion/diffusion approach. They view it as somehow "inelegant" when compared to primitives like RSA. Personally, I think the bias against confusion/diffusion primitives is outdated and unfair. For one thing, the "mathematical structure" which accounts for the elegance of many algebraic/combinatorial primitives provides a potential route for attack. Second, confusion/diffusion primitives tend to be more efficient. This, together with their apparent arbitrariness, gives confusion/diffusion primitives a certain elegance of their own.

Some cryptographic aims aren't known to be solvable starting from a confusion/diffusion primitive. In particular, we don't know how to achieve asymmetric encryption under a confusion/diffusion primitive, nor the the possibly simpler problem of *secret key exchange* under a confusion/diffusion primitive.[1] In other domains, including digital signatures, it is known how to achieve a solution using a confusion/diffusion primitive, but the solutions are complex and less efficient than solutions which make use of algebraic primitives in addition to a confusion/diffusion primitive.

The distinction we have drawn between confusion/diffusion primitives and algebraic/combinatorial ones is not a distinction we know how to formalize. Indeed an algebraic/combinatorial primitive can always be realized by a (large) Boolean circuit which is doing the same sort of bit-twiddling operations of a typical confusion/diffusion primitive. Despite this, when you are presented with

[1] In this problem A and B communicate over a channel subject to eavesdropping (only) by adversary E. Parties A and B want to come into possession of a shared, random, secret key.

a new primitive you'll immediately recognize if its of the confusion/diffusion flavor, or of the algebraic/combinatorial sort.

2.2 The Data Encryption Standard (DES)

The algorithm of the U.S. Data Encryption Standard was developed at IBM around 1974. It became a U.S. NBS Standard in 1977. The algorithm of this standard is sometimes called the Data Encryption Algorithm (DEA), and sometimes it is called again by DES. The algorithm maps a 64-bit *plaintext* to a 64-bit *ciphertext* under control of a 56-bit *key*. We'll denote this map by

$$\mathrm{DES} : \{0,1\}^{56} \times \{0,1\}^{64} \to \{0,1\}^{64}.$$

Usually we write the first argument (the key) as a subscript, preferring $\mathrm{DES}_a(x)$ to $\mathrm{DES}(a,x)$.

2.2.1 Definition of DES

The complete specification of DESwas given in FIPS Pub 46, and in a handout I distributed. In class we talked through the definition, which won't be repeated here. We described, in particular, how any construction of the same basic structure —a *Feistel* cipher— yields a *permutation* on $\{0,1\}^{blocklength}$ for each key. (That is, the particulars of the function F is irrelevant for achieving this.) We also described why DES^{-1} could be computed by simply reordering the subkeys $K_{16}, \ldots, K_1$.

We certainly won't repeat DES's definition in these notes. Do look at the handout to get a feel for this, and Coppersmith's article to get some sense of how the S-boxes, especially, were selected.

2.2.2 Attacks on DES

There has been much worked aimed at breaking DES. To date, the most effective "mathematical" attacks are from differential cryptanalysis (see the Biham and Shamir book) and linear cryptanalysis (see Mitsuro Matsui, Eurocrypt '93 and CRYPTO '94). Neither of these techniques is practical, requiring an enormous number of (plaintext, ciphertext) pairs. (Differential cryptanlysis of DES requires $\approx 2^{47}$ pairs, while linear cryptanlysis uses about $\approx 2^{43}$ pairs.) Currently, the most practical known attack is exhaustive search: the key space is just 2^{56}, which can be exhausted in a large, but not overwhelming, amount of computational work.

The ideas underlying differential and linear cryptanalysis are not terribly hard, but they would take us a bit too far afield of our main efforts. Instead of describing those attacks, I describe a "refutation" for a "potential" attack put forward by Kaliski and Rivest.

2.2.3 DES generates a huge group

Consider the group $S_{2^{64}}$ of all permutation on $\{0,1\}^{64}$. This is an enormous group, having $2^{64}!$ elements. Now consider the subgroup G of $S_{2^{64}}$ which is the group generated by all permutations of the form $\mathrm{DES}_k(\cdot)$, for some key $k \in \{0,1\}^{56}$. What if G were a tiny, tiny subgroup of G'? This sounds like it wouldn't be good. For example, what if each permutation $\mathrm{DES}_k(\cdot)$ could be realized as some $\mathrm{DES}_a(\cdot) \circ \mathrm{DES}_b(\cdot)$ for one of a small number of keys a, b. Then we could hope to break encipherment under DES_k not by finding k, but by finding a representation for an "equivalent" key, a, b.

Coppersmith's article describes a lovely (computer-assisted) proof for why this is not true: he explains how to show that $|G|$ is huge. The argument was described in lecture, but it is very clearly written up in the paper, so I will not repeat it in these notes. The idea is to collect up a bunch of elements of $|G|$ which have easy-to-compute order, and then note, using Lagrange's theorem, that the least common multiple of those orders has to be a divisor or $|G|$, and therefore functions as a lower bound on the size of G.

Mark Dilger pointed out in class a potential concern. The group used in the proof is actually a subgroup of G, the one generated by just the permutations $\mathrm{DES}_{\mathbf{0}}$ and $\mathrm{DES}_{\mathbf{1}}$. Maybe *most* keys generate a subgroup of G, and it just so happens that $\mathrm{DES}_{\mathbf{0}}$ and $\mathrm{DES}_{\mathbf{1}}$ generate a large one. True enough.