Chapter 1
Introduction and Background

February 15, 2010

1

The notes forming this collection are essentially those given out to third and fourth year undergraduate students of a course on Cryptology (Cryptography plus Crypt-analysis; of constructing cryptograms/cyphers plus attacking/breaking them). The course is given in the School of Mathematics, Trinity College, University of Dublin, and the students are essentially local maths students; although some physics, statistics and indeed Erasmus students from Germany have attended.

The course is optional and is given every second year. Originally there was no intention of examining the students at the end of the course, but they themselves requested an examination. They said the material interested them, and they knew their own characters sufficiently well to be sure that without an examination they would never study it properly.

There is much material in the course and although an ability in maths - notably number theory - is desirable, the course is certainly not confined to mathematics. Much of it is hard to categorise: Scientific common sense? Logical reasoning? Clever patterns? Perhaps the one thread that runs right through any course on cryptology is suspicion. Is there a flaw in the argument? Can the procedure be subverted? Can interference be detected? After all, from the very start we know that cyphers, cryptograms, encrypted data - call them what you will - are not what they appear to be. They conceal something.

The course could be considered to be tough. Nevertheless, year after year, some 30 per cent of students perform excellently in the examination and failures amongst the remainder are very few. This has surprised examiners, but this also convinced them that although the course may appear intimidating it

obviously is not unfair, let alone impossible. More, if the students did not respond with their enthusiasm and interest the course would have ceased being given years ago. These notes are dedicated to them and to follow enthusiasts whoever and wherever they maybe.

Finally, the notes which follow were distributed in lectures, and have only been modified in a minor way - correcting errors and obscurities and removing less relevant material. However, in a lecture course there is much talk and this appears here as 'continuity' or 'introduction' to and between the various modules. This 'continuity' is much more brief written than spoken, and may appear abrupt. The reader should therefore ponder for themselves: What is the purpose of this? Is this relevant? Why is this technique chosen rather than some other? Suspicion and scepticism are the makings of a cryptologist.

1.1 Background

The three main strands of cryptography are very ancient. They are:

1. Creating, holding and exchanging secrets or secret messages.
2. Assuring the authenticity of documents (such as written orders).
3. Controlling access to sources of materials or information.

Within these general areas of security *services* there are many subdivisions. For example: confidentiality may apply to data, to the identity of persons (anonymity), or to the existence of communication. In turn such services are provided by *mechanisms* such as encryption, aliasing, spread spectrum techniques or steganography. Each such mechanism may have many different methods of implementation. For example there are innumerable encryption algorithms.

Similarly authenticity *services* can give proof of the identity of the sender (of a message), or proof of receipt, or proof of transmission, or proof of integrity of data (no corruption), and so forth. The *mechanism* providing these services are for example: MACs (Message Authentication Checks) or Digital Signatures. These mechanisms use algorithms based on modular arithmetic or elliptic curves, to name but two.

Again, access control services may take many forms, in particular one way (a PIN, for example), or reciprocal (some challenge-response procedure), or indeed multi-person (threshold - based systems - any m keys of n allow access). And here, besides mathematics, the mechanisms may include physical items, even biometrics. "What you know, what you have and what you are".

Examples from antiquity of such security services are many. Secret messages may be written on the shaved head of a slave. His hair is let grow,

and shaved again or reception (if his head is not cut off to stop him talking , see Herodotus.) Caesar's cypher (see below) is a simple old example of encryption.

Documents have been authenticated by seals in medieval Europe. (Forging of seals, guarded by a lord privy seal, would be punishable by death - and of course one or more signatures, per page.)

And passwords have applied at town gates (not to mention keys and padlocks) since time began. The ordinary man is used to such things - you may take the ass if you give the password 'The master has need of him'.

The following notes concentrate on confidentiality (encryption) techniques and on authenticity (MACs and digital signatures) although other topics are occasionally raised, and we begin with early primitive cypher.

The Latin often inscribed on sundials, has the wise meaning "I do not record the hour unless they are fine".

1.2 Caesar's Cypher

This is a substitution cypher in which each letter (we assume we are handling letters, e.g. the 26 letters of the English alphabet) is replaced by another. We regard the letters as numbered 0 to 25 with A=1, B=2, C=3, . . ., Y=25, Z=0. The substitution rule for Caesar's cypher is to add the one-letter key, k, to the plain-text so $c_i = k + p_i \ mod\ 26$ where c_i and p_i are the ith cypher-text and plain-text letters respectively.

Example The key k, is the letter $C = 3$ so we have

P=	O	R	A	S	N	O	N	C	O	N	S	T	O	N	I
C=	R	U	D	V	Q	R	Q	F	R	Q	V	W	R	Q	L

P=	S	I	S	E	R	E	N	A	S
C=	V	L	V	H	U	H	Q	D	V

(Decryption is subtracting the key from c)

Problems with this simple cypher are:

1. Open to brute force attack (try all the keys, A to Z)
2. A known plain-text - cypher-text pair reveals the key by subtraction
3. More subtly, the relative frequency of letters is transferred but not changed.

In English (not Latin) the most frequent letters are $E(0.131)$, $T(0.090)$, $O(0.082)$, $A(0.078)$, $N(0.073)$, $I(0.068)$, $R(0.066)$, $S(0.065)$ etcetera. Adding key = C simply makes H it the most frequency then W, R, D etcetera. As attacker can count the frequency of occurrence of a letter in the cypher-text and deduce the key from that - assuming Caesar is applied to English.

1.3 Poly-alphabetic Cyphers

These are also substitution cyphers, but the key is several letters long, example: $key = k_i$ for $i = 1$ *to* m and then repeated. We may write $y = f_i(x)$ where y is the cypher-text and x is the plain-text and $f_i()$ is the substitution function using k_i.

Example

Vigenere cypher

$$y = x + k_i \ mod\ 26$$

Beaufort cypher

$$y = k_i - x \ mod\ 26$$

Variant Beaufort cypher

$$y = x - k_i \ mod\ 26$$

With the key = "Latin" = 12, 1, 20, 9, 14 we have:

Plain-text x=	O	R	A	S	N	O	N	C	O	N	S	T	O	N	I
Key 1	L	A	T	I	N	L	A	T	I	N	L	A	T	I	N
Vigenere y=	A	S	U	B	B	A	O	W	X	B	E	U	I	W	W
Key 2	V	B	V	B	V	B	V	B	V	B	V	B	V	B	V
Variant Beaufort y=	E	Q	Y	Z	.	.	.	.	.	.	.	.	.	.	.

Plain-text x=	S	I	S	E	R	E	N	A	S
Key 1	L	A	T	I	N	L	A	T	I
Vigenere y=	E	J	M	N	F	Q	O	U	B
Key 2	B	V	B	V	B	V	B	V	B
Variant Beaufort y=	.	.	.	.	.	.	.	.	.

Here we have a "product cypher". The plain-text is first encrypted with Vigenere and $key =''$ *Latin*", then re-encrypted with Variant Beaufort and $key =''$ VB". The net result is a key of length $5 * 2 = 10$ characters.

An attacker can still "try all the keys" $(26^5 * 26^2)$. He can also search very long samples of cypher-text for relative character frequencies. More profitably he can match two samples of cypher-text against each other and

count the frequency of characters coinciding. When the two samples are "in phase" the probability of a coincidence is p_i^2 where p_i is the probability of character i. When not in phase the probability is $p_i p_j$. In English, for letter E, $p_i^2 = 0.017$ and overall the average $p_i^2 = 0.0673$, whereas out of phase the probability of coincidence is $\frac{1}{26} = 0.0385$.

Shifting the two samples with respect to each other to maximise the non-uniformity of coincidences will reveal the "phasing" of the keys, and hence the actual key value is easily found.

Shannon's theory of secrecy systems (see Appendix M11-X) discusses the importance of key length from the Information Theory view point:the ability to decrypt a cypher-text by trying possible keys and looking for "meaningful" resultant plain-text. The assumption is that not all text is possible (and meaningful) and if only one meaningful plain-text is found then that must be the right one. This assumption does not hold if the plain-text is apparently random. Hence many systems reversibly randomise data before encryption - example: by pre-whitening

1.4 Permutation Cyphers

These simply permute the letters of the plain-text, for example: THEREWASAYOUNGLADYOFCHICH(ESTER etc) is permuted to GOARCHNYSEIOAWTCDAHHLFYEH using a $5 * 5$ square array with the key "GREAT POEMS". The student may work it out for himself.

Note that if there are t letters in the alphabet and the key is of length m, there are $m!$ possible permutations and $(t!)^m$ possible substitutions. Substitution is more complex and so more secure, *but*, if $t = 2$, binary, permutations begin to be important.

1.5 Mechanical Encryptor/Decryptor

Mechanical means of encryption/decryption can save time when the procedure or algorithm is complex (and to be secure it must be complex). They can also ensure accuracy, avoiding human errors. Many have been invented some examples are:

1.5.1 The Jefferson Wheels

A number of wheels m have mounted on a common horizontal axle. Each wheel can turn independently. Each has the 26 letters of the alphabet, in some permutation p_i (for the i_{th} wheel) unique to that wheel, written on the "tread" of the "tyre". To encrypt an m-letter word the wheels are turned

until one can read that word across the wheels (in a line parallel with the axle). One then reads off as cypher-text another word two, three or whatever number of displacements round the wheels from the start word. The result clearly depends on the permumations written on the wheels and the order in which they are mounted on the axle.

To decrypt one must know this key - the permutations on , and the order of the wheels. If one has those correctly mounted one sets up the cypher-text as a horizontal reading and looks for 'meaningful' plain-text at one of 26 displacements from it. The relevance of Shannon's Theory is obvious. What are the chances of finding wrong meaningful plain-text? (The probability of finding random meaningful plain-text of length m *is* $26 * 2^{-2.4m}$ which is $< 1\%$ if $m \geq 5$). Thus if meaningful text is found by valid user or attacker it is almost certainly correct.

1.5.2 The Wheatstone Disc

Two discs, an inner and an outer, are mounted on a common axle like the face of a clock. On the outer are 26 letters plus a space, on the inner are 26 letters in some permuted order. Two hands, geared to move together like clock hands, rotate on the axle. The outer hand is set to point to the next plain-text character. As characters are encoded the hands move around clockwise, but the inner hand advances an extra portion each full rotation so the substitution changes according to the plain-text encoded. What does one do with a repeated plain-text character? Move round 360 degrees and get the same result as if one had moved one position less than 360 degrees? It is clear there is a possibility of decryption ambiguity. Solution? Ban repeated characters. A more serious problem is the "feed forwards" nature of encryption: the plain-text changes the "rules". This means that decryption is "feed back". As a result a decryption error (for example caused by a single character of cypher-text in error) can cause a permanent plain-text error because the "rules" (the relative position of the two hands) have been changed.

1.5.3 The Enigma Machine

The famous German Enigma encryptor of World War II went through various developments (the basic design was captured by Polish Resistance and delivered to the British who cracked the cyphers by it using one of the earliest computers, Colossus, at Bletchley Park). Very roughly, the machine consisted of three discs mounted on an axle, with respectively 25, 26 and 26 positions on each. Characters were encoded one by one by pressing a contact

at the first disc, which lit a lamp at the third. The contact corresponded to the next plain-text character, the lamp to the ensuing cypher-text character. On each operation the first disc advanced a position and before recommencing after 25 it ratcheted on the second disc one position; and similarly the second moved on the third after a revolution. Thus the substitution sequence only repeated after $25 * 26 * 26 = 16900$ characters.

Each disc bore a permutation of the alphabet round the rim in the form of cross wirings, so it one pressed A the result on the first disc might be G. G in turn had electrical contact with a letter point on the second disc cross wired also to another character with electrical contact to the third disc, which would be cross wired to the final character and light bulb.

Decryption is obviously to press a button corresponding to the cypher-text character at the third disc and observe the light at the first (assuming the set-up is symmetrical). Clearly the ratcheting mechanism for advancing the discs must be suitably arranged to work from disc three to one, rather one to three, when decrypting.

The "key" for Enigma is the cross-wiring permutations on the discs (involutions) and the choice and order of mounting of discs for a specific occasion - there was a library of discs.

It would require a very long sequence of cypher-text to find out the plain-text by counting character frequencies. Even brute force attacks are laborious (how do we recognise success? see Shannon Appendix M11-X). But one approach is to plant known plain-text or partial plain-text and search for it when trying keys. This is not too difficult: Organise some extraordinary event which the enemy is bound to comment on in his plain-text, and look for key words.

1.5.4 Some Diagrams

In general one may represent encryption/ decryption operations by figure 1.F1.

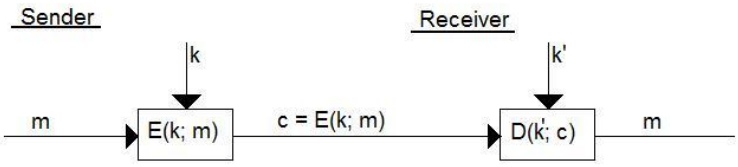

Figure 1.F1

Here m=message (plain-text), k=encryption key, $E()$=encryption operation which produces c, c=cypher-text, $D() = E^{-1}$=decryption, and $k' = \textit{decryption key}$. In symmetric systems (see Chapter 2) $k' = k$, and sender and receiver share the same key. In asymmetric, public key systems $k' \neq k$ and k' cannot be deduced from k.

A central problem is: Who generates the keys? How are they distributed to participants? More generally: How are keys managed, including withdrawals, changes, safeguarding, etcetera?

Figure 1.F2 represents a general scheme for authentication.

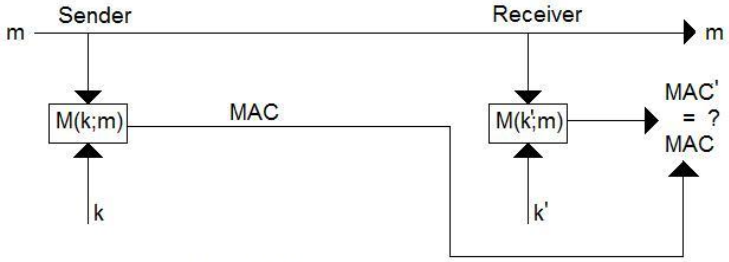

Figure 1.F2

Message m is put through algorithm M using key k to produce a Message Authentication Check or MAC. The received message (hopefully uncorrupted m) is put through algorithm M' using key k' to produce an output we call MAC$'$. MAC$'$ is tested for compatibility with MAC, and if successful the conclusion is:'The message m was uncorrupted, the sender was using the correct key k corresponding to M, M' and k' and so must be who he claims to be. In many cases $M() = M'()$ and $k = k'$ and the test is $MAC' = MAC$? In other cases, digital signatures, tests and keys are different (see Chapter 4) and the compatibility test ($MAC' = MAC$) is more complex. As with encryption, key management is critical.

Access control systems can be represented by many diagrams, for example one-way password-submission. Figure 1.F3 shows a common form of three-way access control.

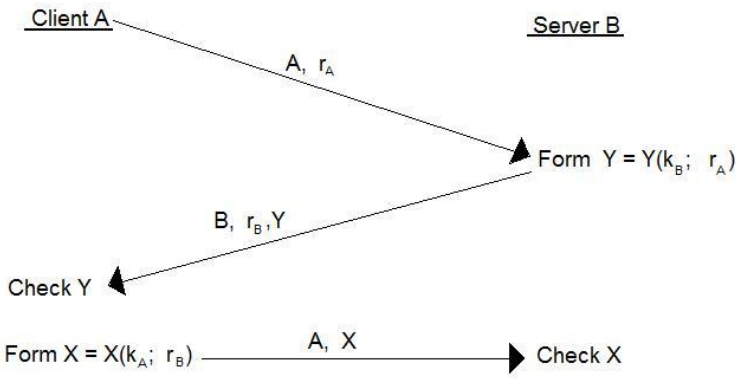

Figure 1.F3

Client A wishes to access Server B. A sends his claimed identity and a "challenge" (example: a random number r_A). B sends back his identity and response Y to $A's$ challenge generated using his key k_B. B sends a challenge r_B to A. A checks Y (which will involve some key compatible with k_B) to see if $A's$ response is valid. If it is, A knows that he is communicating with B. A then produces X his response to $B's$ challenge r_B using k_A and sends X back to B. B checks X (which will involve some key compatible with k_A) to see if $B's$ response is valid. If all this succeeds then A and B conclude that they are talking to the real A and B and grant further access or communication rights.

The purpose of challenge/response is so that access control does not rely on a fixed item, visible to an eavesdropper on the channel. Responses X, Y cannot be foreseen, or generated except by those holding k_A, k_B.

The figure 1.F3 is often elaborated to include, the establishment for example of a secret key to encrypt all subsequent conversation, or to deliver authorisation parameters to A, and so forth.

Note the A and B are really mechanised procedures; In a smart-card? In a computer? Note also that $A's$ claim might be supported by further information such as a biometric reading, or even a simple password to ensure that B only processes plausible clients, rejecting all those who cannot produce a basic password before spending time on more complex checks.

Appendix X
Unicity Key Length and Unicity Distance

If 1111 there are P plain-texts, C cypher-text and K keys, then on average each cypher-text arises from $\frac{KP}{C}$ plain-texts.

In a brute force (try-all-the-keys) attack if $\frac{KP}{C} >> 1$ then the attacker will find very many possible plain-texts and be unable to choose between them. Thus the cypher can be made secure by choosing: Large K, $C \approx P$. ($P \leq C$ for reversibility, but if P is non-redundant then all plain-texts (nearly) are valid and $P \approx C$).

Conversely, given K, and given $P < C$ an attacker can look for very large cypher-texts (very long) and get $\frac{KP}{C} < 1$. Now if he finds a meaningful plain-text when trying to decrypt he is almost sure that it is *the* plain-text.

The number of meaningful plain-texts is determined by the entropy $H(P)$. We define $H(P)$ by

$$H(P) = \sum_{i=1,t} p_i \; log \; \frac{1}{p_i}$$

where p_i is the probability of occurrence of the i_{th} character in an alphabet of t characters. (This is a crude definition, taking no account of double and triple letter combinations, etcetera; but it will serve).

We take two as the base of the logarithms. We could take something else (example: t). With two as base the entropy tells us how many binary information units (bits) are needed to encode one character. If all the p_i are equal, then $H(P) = log_2 t$, since $p_i = \frac{1}{t}$. In most languages $H(P) << log_2 t$. Suppose plain-text, cypher-text and keys are all formed from this alphabet of t characters. Suppose plain-text and cypher-text are of the same length n, and the key is of length k. Then

$$\begin{aligned} \mathrm{C} &= \mathrm{t}^n = 2^{\alpha n} \quad \text{with } \alpha = log_2 t \\ \mathrm{K} &= 2^{\alpha k} \\ \mathrm{P} &= 2^{H(P)n} \end{aligned}$$

Note that P is now the number of meaningful plain-texts, and is much less than $2^{\alpha n}$.

Given P and C the unicity key length, k', is that key length which gives K' keys with $K' = \frac{C}{P}$ that is $2^{\alpha k'} = \frac{2^{\alpha n}}{2^{H(P)n}}$.

We want (for unbreakable cypher - many valid plain-texts when an attacker tries all the keys) $k >> k'$ that is:

$$k >> \ Unicity\ key\ length = k' = n(1 - \frac{H(P)}{\alpha})$$

Note: $\frac{H(P)}{\alpha}$ is the entropy of p with t as the base of logs.

If $H(P) \approx \alpha$, as occurs with non-redundant text, then, to make our cypher secure, we can work with smaller k'.

Conversely, given K, the unicity distance is that length of cypher-text which enables an attacker (if he finds many examples) to break the cypher because $C >> KP$ that is $n >> n' =$ *Unicity distance* where n' is given by $C = KP$ i.e. $2^{\alpha n'} = 2^{\alpha k} * 2^{H(P)n'}$ so $n' = \frac{k}{1 - \frac{H(P)}{\alpha}}$. The quantity $(1 - \frac{H(P)}{\alpha})$ is called the Redundancy, $R < 1$.

$$\begin{aligned} Therefore\ Unicity\ Key\ Length &= \mathrm{k}' = nR \\ Unicity\ Distance &\quad \mathrm{n}' = \frac{k}{R} \end{aligned}$$

For security we want $k >> k'$. To break, we want $n >> n'$. If $R = 0$ the cypher is theoretically unbreakable.

In English $\alpha = log_2 26 \sim 4.6$, $H(P) \sim 2.3$ say so $R = \frac{1}{2}$, so $k' = \frac{n}{2}$ and $n' = 2k$.

1.1.X Perfect Secrecy

Shannon also puts forward the notion of perfect secrecy where the probability of plain-text p given knowledge of the cypher-text c, $Prob(p \mid c)$ is equal to the probability of $Prob(p)$. By Bayes' Theorem $Prob(p|c).Prob(c) = Prob(c|p).Prob(p)$.

So perfect secrecy implies $Prob(c|p) = Prob(c)$ evaluated over all keys k. So the distribution of c is independent of p. Now
$Prob(c|p) = (\sum Probabilities\ of\ all\ keys\ which\ perform\ the\ mapping\ p \rightarrow c)$
and independence means that p is mapped into all cypher-texts. Therefore K where $K = Number\ of\ keys$ and $C = Numberofcypher - texts$.

But $C \geq P$ the number of plain-texts, for reversibility.

Therefore $K \geq P$ is necessary for perfect secrecy. The number of keys must be greater or equal to the number of plain-texts.