Composition of Ciphers

Klaus Pommerening
Fachbereich Physik, Mathematik, Informatik
der Johannes-Gutenberg-Universität
Saarstraße 21
D-55099 Mainz

April 7, 1997—English version August 14, 2014
last change January 20, 2021

A first approach to constructing strong ciphers is the composition of several simple transformation steps. We saw some examples of this approach already in classical Cryptography, and we saw how this often resulted in much stronger ciphers. But this effect is not guaranteed, and does not crop up in all cases. In this section we consider some basic aspects of this approach that is an essential ingredient of the construction of strong bitblock ciphers.

Terminologically we distinguish between

Multiple ciphers: combinations of instances of the same encryption function but with different keys.

Cascades: combinations of different encryption functions (also called product ciphers).

1 Multiple Ciphers and Group Structures

Multiple Ciphers

Let $F = (f_k)_{k \in K}$ be a cipher over the alphabet Σ, where $f_k \colon \Sigma^* \longrightarrow \Sigma^*$ is the encryption function corresponding to the key $k \in K$. The set of all of encryption functions is denoted by

$$\tilde{F} = \{f_k \mid k \in K\} \subseteq \mathrm{Map}(\Sigma^*, \Sigma^*).$$

By forming the **double cipher**

$$F^{(2)} = (f_h \circ f_k)_{h,k \in K}$$

the key space is significantly enlarged from K to $K \times K$. In the same way we can construct the triple cipher $F^{(3)}$, ..., the n-fold cipher $F^{(n)}$. All this makes sense only when

(A) $\tilde{F}$ is not a semigroup.

If $\tilde{F}$ *is* a semigroup, then for each pair of keys $h, k \in K$ there exists a key $x \in K$ such that $f_h \circ f_k = f_x$, and we don't get any new encryption functions by this kind of composition—a typical case of an "illusory complication", the effective keysize didn't increase at all!

We observe an even better effect when

(B) $\tilde{F}$ generates a subsemigroup of $\mathrm{Map}(\Sigma^*, \Sigma^*)$ of large size.

And the best we can hope for is:

(C) The map $K \times K \longrightarrow \widetilde{F^{(2)}} \subseteq \mathrm{Map}(\Sigma^*, \Sigma^*)$ is injective.

For a finite key space K we can express this also in the form:

(C') $\#\widetilde{F^{(2)}} = \#\{f_h \circ f_k \mid h, k \in K\} = (\#K)^2$.

The Group Property of a Block Cipher

A block cipher is uniquely characterized by its effect on Σ^r for a given exponent r, the blocksize. (For the moment we don't care about continuing it to strings of arbitrary lengths or about "padding" shorter strings to full blocklength.)

A block cipher **preserves lengths** if it transforms Σ^r to itself. Then in a canonical way $\tilde{F}$ is a subset of the symmetric group $\mathcal{S}(\Sigma^r)$, hence finite, and without restriction we may assume that also the keyspace K is finite. For such block ciphers the semigroup property (the converse of (A) above) is equivalent with the group property. This follows from the well-known simple lemma:

Lemma 1 *Let G be a finite group, $H \leq G$ a subsemigroup, that is $H \neq \emptyset$ and $HH \subseteq H$. Then H is a group, in particular $\mathbf{1} \in H$.*

Proof. Each $g \in G$ has finite order, $g^m = \mathbf{1}$ for some m. If $g \in H$, then $\mathbf{1} = g^m \in H$, and $g^{-1} = g^{m-1} \in H$. $\diamond$

This proves:

Proposition 1 *Let F be a length preserving block cipher over a finite alphabet. Then the following statements are equivalent:*

(i) *For any two keys $h, k \in K$ there exists an $x \in K$ such that $f_h \circ f_k = f_x$.*

(ii) *The set $\tilde{F}$ of encryption functions is a group.*

Remark

The probability that two random elements of the symmetric group $\mathcal{S}_n$ generate the whole group $\mathcal{S}_n$ or at least the alternating group $\mathcal{A}_n$ is

$$> 1 - \frac{2}{(\ln \ln n)^2} \quad \text{for large } n.$$

Source: John Dixon, *The probability of generating the symmetric group.* Mathematische Zeitschrift 110 (1969), 199–205.

For $n = 2^{64}$, a typical size for a block cipher, this lower bound is ≈ 0.86. With high probability it should generate the full or at least the "half" permutation group on the blocks. The concrete proof however might be difficult. One would try to determine the order of some concrete encryption functions by their effect on certain concrete messages, and then take the lowest common multiple as a lower bound for the group order.

In any case it seems that in general a multiple cipher is stronger than the underlying simple cipher. We'll discuss this again in Sections 3 and 4.

2 Examples of Multiple Ciphers

Examples of Groups

Each of the following length preserving ciphers forms a group:

- The shift ciphers over Σ with respect to a group structure on Σ
- The monoalphabetic substitutions over Σ
- The Bellaso ciphers with a fixed period
- The block transpositions of a fixed length

DES

DES is a block cipher on $\mathbb{F}_2^{64}$ with keyspace $\mathbb{F}_2^{56}$. Campbell and Wiener in (Crypto 92) proved that DES generates the alternating group of order 2^{64}. Shortly before Coppersmith had shown that the group order is at least 10^{277}. Only much later someone noted that Moore and Simmons in Crypto 86 had published the lengths of several cycles that would have sufficed to show that DES is not a group—a fact that for several years was viewed as an open conjecture.

Historical Examples

The composition of a polyalphabetic cipher of period l and another one of period q has period $\operatorname{lcm}(l, q)$. **Application:** Key generating machines as mentioned in Part I, see the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/4_Cylinder/LongPeriods.html`.

Another historical example: the double columnar transposition that is considerably stronger than the simple columnar transposition.

Composition of Bellaso Cipher

The composition of two Bellaso ciphers of periods l and q has period $\operatorname{lcm}(l, q)$, essentially the product lq. However its security amounts at most to the sum $l + q$ in view of an attack with known plaintext:

Assume known plaintext of length $l + q$ (over the alphabet $\mathbb{Z}/n\mathbb{Z}$). This yields $l + q$ linear equations for $l + q$ unknows—the characters of the two keys. Assume that $l < q$. Then the situation is

Plaintext	a_0	a_1	...	a_{l-1}	a_l	...	a_{q-1}	...
Key 1	h_0	h_1	...	h_{l-1}	h_0	...	...	...
Key 2	k_0	k_1	...	k_{l-1}	k_l	...	k_{q-1}	...
Ciphertext	c_0	c_1	...	c_{l-1}	c_l	...	c_{q-1}	...

Taken together this is a Bellaso cipher with key

$$(h_0 + k_0, h_1 + k_1, \ldots)$$

and period $\operatorname{lcm}(l, q)$.

Let the known plaintext be $(a_0, \ldots, a_{l+q-1})$. Then the system of linear equations for the $l + q$ unknowns $h_0, \ldots, h_{l-1}, k_0, \ldots, k_{q-1} \in \mathbb{Z}/n\mathbb{Z}$ is:

$$\begin{aligned}
h_0 + k_0 &= c_0 - a_0, \\
h_1 + k_1 &= c_1 - a_1, \\
&\vdots \\
h_{l-1} + k_{l-1} &= c_{l-1} - a_{l-1}, \\
h_0 + k_l &= c_l - a_l, \\
&\vdots \\
h_{l+q-1 \bmod l} + k_{l+q-1 \bmod q} &= c_{l+q-1} - a_{l+q-1}.
\end{aligned}$$

This cannot have a unique solution: If we add a fixed value x to all h_i, and subtract x from all k_j, then we get another solution. Therefore for simplicity we may assume $h_0 = 0$. If the keys are not randomly chosen but built from keywords, then a simple "Caesar exhaustion" will reveal the "true" keys later. For decryption the shifted keys are equivalent. And since we eliminated one unknown quantity, in general even $l + q - 1$ known plaintext letters are enough for uniquely solving the remaining $l + q - 1$ equations. We won't go into the details but give an exercise for interested readers.

Exercise

Consider the ciphertext

```
CIFRX KSYCI IDJZP TINUV GGKBD CWWBF CGWBC UXSNJ LJFMC
LQAZV TRLFK CPGYK MRUHO UZCIM NEOPP LK
```

For an attack with known plaintext assume that

- the plaintext (is in German and) starts with "Sehr geehrter ..." (a common beginning of a letter)
- some keylengths are already ruled out by trial & error; the actual lengths to test for a double Belaso cipher are $42 = 6 \times 7$.

(A coincidence analysis, even if it doesn't give enough confidence in a definite period, should suffice to exclude all but a few combinations of possible keylengths.)

3 Cryptanalysis of Double Ciphers

Meet in the Middle

The name of this attack against double encryption goes back to Merkle and Hellman in 1981. (Don't confuse it with the "Man in the Middle" attack against cryptographic protocols.) They formalized an attack that worked in "classical times" against rotor machines, see the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/5_Rotor/AnalRot.html`.

Consider the composition of two encryption functions with different keys:

$$\begin{array}{ccccc} \Sigma^* & \xrightarrow{f_k} & \Sigma^* & \xrightarrow{f_h} & \Sigma^* \\ a & \mapsto & b & \mapsto & c. \end{array}$$

Assume a pair (a, c) of corresponding plaintext and ciphertext is known, and assume that the exhaustion of the simple cipher is feasible. Then the attacker builds two tables:

- all $f_k(a)$, $k \in K$,
- all $f_h^{-1}(c)$, $h \in K$,

and compares them. Each coincidence yields a possible pair $(h, k) \in K^2$ of keys that can be further inspected, say with further known (or guessed) plaintext.

Expenses

This attack needs

- $2 \cdot \#K$ encryptions (*not* $(\#K)^2$),
- $2 \cdot \#K$ memory cells.

Noting that we need only store one of the two tables we even halve the number of memory cells.

With the usual prefixes for memory sizes

2^{10}	2^{20}	2^{30}	2^{40}	2^{50}	2^{60}
Kilo	Mega	Giga	Tera	Peta	Exa

and using 1 byte = 8 bits we see that 60 bit keys need memory that exceeds the (actually) available capacities. However for cryptanalysis the time requirements are more critical than memory requirements. Therefore as a general finding we may state:

> *The security of a double cipher is not significantly better than the security of the underlying simple cipher.* In particular the bitlength of a key exhaustion is not doubled but only increased by 1 bit.

False Alarms

One question yet awaits an answer: How many of the coincidences in comparing the two tables lead to a wrong pair of suspected keys? That is, how likely are false alarms?

Here is a heuristic consideration: Assume we encrypt n-bit blocks with l-bit keys. Then the tables have 2^l entries, resulting in 2^{2l} comparisions. Since the number of possible values is 2^n we expect about $N_1 = 2^{2l-n}$ coincidences. (Implicitly assuming that the values behave like random. By the Birthday Paradox we expect the first coincidence after $2^{n/2}$ trials, but this is irrelevant in the present context.)

If we test the pitched key pairs with a second known plaintext block, then we are left with $N_2 = N_1/2^n = 2^{2l-2n}$ candidates. After testing t known plaintext blocks we expect to keep $N_t = 2^{2l-tn}$ candidates—but of course at least one, the right one.

Thus in general we find a unique solution as soon as

$$t \geq \frac{2l}{n}.$$

Examples

1. DES, $n = 64$, $l = 56$: $N_1 = 2^{48}$, $N_2 = 2^{-16}$. *We need about* 2 *blocks of known plaintext.*

2. IDEA, $n = 64$, $l = 128$: $N_1 = 2^{192}$, $N_2 = 2^{128}$, $N_3 = 2^{64}$, $N_4 = 1$. *We need about* 4 *blocks.*

3. AES, $n = 128$, $l = 128$: $N_1 = 2^{128}$, $N_2 = 1$. *We need about* 2 *blocks.* But the number $\#K = 2^{128}$ will by far exceed our time and memory resources (as in Example 2).

Time-Memory-Tradeoff

A more general consideration yields a "Time Memory Tradeoff": Undertaking a Meet in the Middle attack we may spare memory, allowing more execution time, by generating only partial tables:

If during a pass we fix s bits of both h and k, then we need 2^{l-s} memory cells for both of the tables of $f_k(a)$'s and $f_h^{-1}(c)$'s. As a compensation we have to go through 2^{2s} passes. The expenses are:

$2 \cdot 2^{l-s}$	encryptions for building one pair of tables,
2^{2s}	comparisions of one pair of tables, in total
$2 \cdot 2^{l+s}$	encryptions,
$2 \cdot 2^{l-s}$	memory cells.

Multiplying the number of encryptions and the number of needed memory cells we get $4 \cdot 2^{2l}$, independently from s. *This gives the attacker some freedom in using her resources in a flexible way.*

Example DES: If the attacker owns 128 terabytes of memory, she can generate 2 tables of 2^{40} blocks each, hence choose $s = 56 - 40 = 16$. Then she needs $2 \cdot 2^{72}$ encryptions. This is feasible, at least for the world's largest secret service.

Summary

Double ciphers don't improve the security of encryption in a worthwile way.

4 Triple Ciphers

The last section unveiled a principal weakness of double encryption. Therefore, to get a real improvement, we move on to triple encryption. An often used scheme is "EDE" (Encryption, Decryption, Encryption)

$$f_g \circ f_h^{-1} \circ f_k \quad \text{for } g, h, k \in K.$$

Why is f_h inverted? The advantage of this scheme is its compatibility with simple encryption by choosing keys $g = h = k$.

The Meet in the Middle attack also applies to this scheme. Thus the effective key length (for exhaustion) is not tripled but only doubled, but that's OK for 56 or 64-bit keys.

Often the scheme is somewhat simplified as "two-key triple encryption":

$$f = f_k \circ f_h^{-1} \circ f_k \quad \text{for } h, k \in K.$$

This scheme has a weakness under an attack with *chosen* plaintext that however worries only paranoiacs. Consider the scenario

$$\begin{array}{ccccccc} \Sigma^* & \xrightarrow{f_k} & \Sigma^* & \xrightarrow{f_h^{-1}} & \Sigma^* & \xrightarrow{f_k} & \Sigma^*, \\ a & \mapsto & b & \mapsto & b' & \mapsto & c. \end{array}$$

Step 1: Using $\#K$ encryptions and $\#K$ memory cells precalculate the table

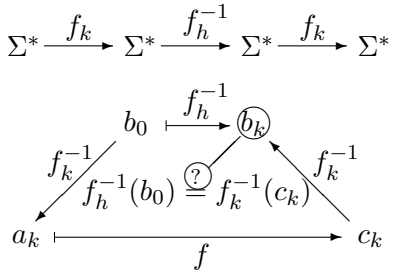
$$\{f_h^{-1}(b_0) \mid h \in K\}$$

for a fixed intermediate value of b_0.

Step 2: Then calculate for all keys $k \in K$:

$$\begin{array}{rcll} a_k & := & f_k^{-1}(b_0) & \text{(the chosen plaintext)}, \\ c_k & := & f(a_k), & \\ b_k & := & f_k^{-1}(c_k). & \end{array}$$

The second assignment is possible in an attack with chosen plaintext, which implies that we can evaluate f with any plaintexts. The expenses are $5 \cdot \#K$ simple encryptions. If $b_k = f_h^{-1}(b_0)$, then we keep the pair (h, k) of keys for further examination.



The most efficient known attack is described in:

- Van Oorschot/Wiener: A known plaintext attack on two-key triple encryption. Eurocrypt 90.

5 Cascades of Different Ciphers

Examples

1. Monoalphabetic substitutions and transpositions commute. Combining more than one of each doesn't make sense since each of these two types forms a group. Composing one monoalphabetic substitution and one (simple) transposition makes a weak cipher. Solving it by a ciphertext only attack starts with a frequency count that reveals the most common letters.

2. The same remark applies to periodic polyalphabetic ciphers and transpositions. But if we take different period lengths for each step we get a fairly complex cipher, however it is too complex for manual operation.

3. The Enigma composed a monoalphabetic cipher with several polyalphabetic substitutions of different periods, followed by one more monoalphabetic substitution. The result was a single polyalphabetic substitution with a very large period.

4. The ADFGVX cipher used by the German army in WW I consisted of a substitution followed by a columnar transposition. For the substitution the 26 letters and 10 digits were distributed into a 6-by-6 square in an order defined by the key. Then each character was replaced by its coordinates in this square that were denoted by A, D, F, G, V, X. The French (Painvin und Givierge) had many successes in breaking this cipher.

5. Composing a monoalphabetic cipher with an autokey cipher is one of the "modes" that make block ciphers a little bit harder, see Chapter 3.

6. Finally recall that Porta's disk cipher had a representation as composition of a monoalphabetic substitution with a Belaso (aka Vigenère) cipher.

As a résumé we may state that cascades of different ciphers in general increase the security, but not always. In any case the situation requires a careful analysis before we trust a newly constructed product cipher.