Formal verification of properties of block ciphers

Ruofan Yang

October 21, 2024

Abstract

Interactive Proving combines the strengths of manual and automated proofs. It is used to prove the correctness of intended algorithms in a more rigorous and formal manner, ensure no logical errors. Data Encryption Standard (DES) is a block cipher that player a significant role in data security, and is one of the most widely used cryptosystem. It contains many well known properties about complementation, weak and semi-weak keys that are concluded and common used. This paper use the HOL4 interactive theorem prover to define the properties based on the built-in DES implementation and prove the properties. The paper also includes the implementation of RC5 cryptosystem and initially implement the crypatanalytic attack Differential Crypatanalysis towards DES and prove some theorems of it. RC5 is a symmetric-key block cipher well known for its variable block and key size. The Differential Crypatanalysis is to break DES into fewer rounds. The work ensures the correctness and effectiveness of cryptosystem and attack method, enable error-free, reliable operation in real-world applications.

1 Introduction

Cryptography has been widely used in the world since centuries ago, it constructs protocols to protect the data and secure the communication. Block cipher is a deterministic algorithm used in Cryptography, it uses fixed-length blocks that are composed with bits to encrypt and decrypt data. Data Encryption Standard (DES) and RC5 are both famous algorithms of block cipher. They are commonly used, studied, researched many times daily by uncountable people, but rare people query the correctness of description about them. It may cause significant vulnerabilities and insufficient security level of encrypting data, thus the cryptosystem built upon them may fail to secure important data and data breaches may occur.

In this paper, I verify the properties of block cipher especially for DES and RC5 algorithm, ensure the correctness of the properties, thus provide direct evidence of the security and reliability of DES and RC5. This paper mainly include the proving of complementation property, weak key and semi-weak key property of DES, the implement of RC5 and the initial built of theorems and also related verification of Differential Crypatanalysis. Differential Crypatanalysis is a type of cryptanalytic attack used in DES-like cryptosystems, it can break variant DES of up to 15 rounds. The initial built definitions and verifications of Differential Crypatanalysis ensures the correctness of basics, thus support the future construction. This paper use the Interactive Theorem Prover HOL4 to do all the implementation and verifications, it provides solid and large bases of theorems, built-in decision procedures and tactics,so I can build the verification based on them straightly and expediently. As a result, the paper also enboard the theory base of HOL4, provide some fundamentals of block cipher construction which allow future research and verification efforts in HOL4.

2 Preliminary

2.1 HOL4

HOL4 theorem proving system is a proof assistant with built-in decision procedures, tactics and theorems to be convenient to prove harder theorems for users. It is composed with HOL types, terms, rules of inference and theories. The types in HOL4 contains the basic atomic types like ”int”, ”bool” or the ”word n” type which is commonly used in proving cryptography aspect. It also contains the compound types such as ”set” and ”list” , and also the function types which are types of function from one type to another.

In HOL4. each term should have a type and a term can be a variable, constant and function. The rules of inference are used to derivate new, more complex theorems from existing ones. They are implemented as ML functions and take terms or theorems as input and return theorems as outputs, each can be treated as a proving step during the proving. Theories are composed with the type structure, signature, set of axioms and set of theorems. Each theory is an extension of the parent theories that focusing on some more specific aspects from its parent theories and research them in more details. It builds upon the concepts from the parent theories, and they are overall organized hierarchically to form a tree-like framework.

2.2 basic use of HOL4

rw[] simp fs RW_TAC POP_ASSUME MP_TAC Q.ABBREV_TAC Abbr Suff Know Rewr’

2.3 Word theory in HOL4

In this paper, ”wordsTheory” in HOL4 are mainly used to prove the properties of DES, Differential Crypatanalysis and the implementation of RC5. It is built upon some basic theories and ”fcpTheory” which are also common used in this paper. The terms used in the verifications are mainly in type of wordn which means a word with length n. ”wordsTheory” contains many definitions and theorems about the words’ operations, analysis and properties.

There are some operations that are frequently applied. The ”word_concat_-def” defines the operator ”@@”, it takes two inputs v and w of type word n and word m, join them into a word of length n+m, the first m bits are the same as w, and the left n bits are the same as v.

$\vdash$ v `@@` w `= w2w (word_join` v w`)`

Then the ”word_extract_def” defines the ”><” operator, it uses two number inputs h and l where h>l, and is applied to a word. It extracts from the lth bits to the hth bits of the word, result a word of length (h-l+1)

$\vdash$ `(`h `><` l`) = w2w` $\circ$ `(`h `--` l`)`

The ”??” operator defined by ”word_xor_def” also takes two words of same length as inputs, each corresponding bit of the two words are applied with the xor (Exclusive-OR) operation and result a words of that length.

$\vdash$ $v \oplus w$ `= FCP` i. v `'` i $\not\Leftrightarrow$ w `'` i

At last, one common operator used in proving the property of complementation is ” ”, it takes a word as input, and converts each bit to the inverse (1 to 0 and 0 to 1 for binary form), produce the bitwise complement of word as result.

$\vdash$ $\neg w$ `= FCP` i. $\neg w$ `'` i

The ”wordsTheory” is built with the support of ”fcpTheory”, ”fcpTheory” build the foundation that enables the bit-level operations. Consequently, we can do the operations mentioned above as they need to analyze bits of a word.

2.4 Measure and Probability theory in HOL4

The ”measureTheory” and ”probabilityTheory” build the definitions of algebra (X,A), σ-algebra, thus the measure space (X,A,u) and probability space. The σ-algebra is algebra that the union of all subsets of A also belongs to A. Probability space is a measure space satisfy the property that u(X)=1, and are often represented as (Ω, F, P). The theories also includes the complementary definitions and theorems related to the properties of these definitions. The theories are built upon the ”extrealTheory”, ”extrealTheory” extends the range of standard real number, adds the positive infinite and negative infinity. It is because algebra is a set which include not only finite algebra but also infinite algebra.

$\vdash$ `measure_space` m $\iff$
`sigma_algebra (measurable_space` m`)` $\wedge$ `positive` m $\wedge$
`countably_additive` m

The ”measureTheory” and ”probabilityTheory” are used in the implementation of Differential Crypatanalysis as the need of working with sets. It requires to get the probabilities of subsets out of a universal set. In the cases of Differential Crypatanalysis attack against DES, the overall sets are usually the universal sets of words or word pairs of a specific length, the target sets are DES inputs of that length and satisfy particular properties.

3 Data Encryption Standard (DES)

Data Encryption Standard (DES) is a block cipher that inputs 64-bit blocks with a 56-bit key[3]. It is a Feistel cipher, consisting of 16 rounds of encryption that repeat the same encryption procedure for each round. The key entered by the users are 64-bit, but 8 parity bits are ignored. The round keys used in each round are implemented by the Permuted Choice 1 (PC1) first, it splits the 64-bit key to two 28-bit words with rearranging and ignores the 8 bits in positions $8 * n\ th$ of the key, load them to two registers C, D. Then to compute round keys in different rounds, C and D are rotated by one or two bit positions to the left in each round, therefore the two 28-bit words are different in different rounds. Finally, round keys are 48-bit keys that are extracted from Permuted Choice2 (PC2).

Next, to encrypt a plaintext, the input message first undergoes an initial permutation (IP) and then splits to two 32-bit halves. Then the halves u,v are passed to the round process, one of the halves v is applied to the round function with round keys, undergoes XOR operation with the other half u. The result 32-bit word v1 is swapped with v, so v1 will be the input of the round function in the next round and XOR with v, the process will repeat for 15 times. However, in the last round, two output halves will not be swapped, they are joined, undergoes the inverse permutation IP^{-1} to produce the final 64-bit output.

The round functions in each round is the same, they first expand the 32-bit half to 48 bits, then operate it with round key by XOR operation, the 48-bit intermediate value are separated into eight 6-bit words, put each into a different S-boxes, and the eight outputs of 4-bit are concatenated to form a 32-bit message. At last, the 32-bit message is permuted by A bit-level permutation (P) and produces the output of the round function.

The S-boxes are the only nonlinear components in DES [3]. Each S-box can be considered as a table of 4 rows and 16 columns, containing values from 0 to 15. the 6-bit input is split, two outer bits are formed to choose the row of table and the four inner bits are formed to get the column, the value in that row and column is the 4-bit output of that S-box.

It is worth noticing that DES, as a Feistel cipher, has a decryption process that follows the same procedures as encryption but use the round keys in reverse order.

4 DES properties

In this paper, some important DES properties are implemented and proved with the assistance of HOL4. They are the implementation properties, weak key and semi-weak key properties.

The DES complementation property [2] is for a plaintext input m and key k, and the bitwise complement of them m and k, the complement result after encrypting m using DES and k as key is the same as the result after the DES

$$\overline{\mathrm{DES}}_k(m) = \mathrm{DES}_{\overline{k}}(\overline{m}).$$

Figure 1: Formula for DES complementation property

$$\mathrm{DES}_w(\mathrm{DES}_w(m)) = m.$$

Figure 2: Formula for Weak key property

encryption step using w and k like shown in Figure **??**. By verifying this, we know a limitation of DES, in a chosen message attack this property can reduce the cost of exhaustive key search by half[2]. It is because to check if one guessing k is right, one known message and DES encryption can produce two ciphertext c and c by applying bitwise complementation to c. This process equals to two DES encryption for two known messages and keys, the exhaustive search of key of length 56 can reduce from 2^{55} to 2^{54}.

Weak keys are four special keys follow the property that DES encrypts a message twice with the same weak keys will return the same message shown in Figure **??**. It is because the weak keys cause round keys equal to the reverse of the round keys, thus the DES encryption and decryption are the same. Weak keys also have the property that each weak key has corresponding 2^{32} messages that encrypt these messages with that weak key will result message unchanged. These messages possess the property that their halves are the same after encryption process in the 8th rounds. By combining this with the weak keys' reverse order round keys property, it ensures the halves in the (8-n)th and (8+n)th round are the same, result the original message (0th round) the same as the ciphertext (16th round).

Similarly, semi-weak keys are six key pairs that encrypts a message with one of the key in the pair then encrypts the resulting ciphertext with the other key in the pair will keep the message unchanged. The formula is shown in Figure **??**. The reason is the same as weak keys, the round keys of the two semi-weak keys in a pair are arranged in the reverse order.

These properties cannot reduce the complexity for exhaustive key search especially for only 4 weak keys compare to the total of 2^{56} keys. However, their existence can be vital when DES is used within some construction[3].

$$\mathrm{DES}_{w_1}(\mathrm{DES}_{w_2}(m)) = m.$$

Figure 3: Formula for Semi-weak key property

4.1 DES in HOL4

DES algorithm has been built in the ”desScript” file in HOL4, and my paper build the verifications based on the DES implemented in this file.

In the file, it first defines some specific types used in the Data Encryption Standard implementation. It includes the halves of word/plaintext as a pair of word32, halves of keys in each round as pair of word28 and a function type of S-Box that take word6 as inputs and return word4. It also includes the data tables to help facilitate the IP, inverse IP permutations, E expansion, P, PC1 and PC2 permutations, round-dependent rotation values and the permutation values for the S-boxes. Then there are some functions and definitions that perform the expansion, permutations, reversion and the S-Boxes’ processing. The ”Key Scheduling” section in this file are definitions to build the round keys from the input key. As the last part of preparing, there are join and split functions to initially split the plaintext and form the result ciphertext. Theorems are also constructed to verify some basic properties and help ensure the correctness of implementations.

It then actually build the Feistel network, round function. Moreover, it applies the above operations, along with the round keys to form the actual process for each round and ultimately construct the complete DES process over a total of16 rounds. It also implements a help function that can return the halves in each round as a pair in the form of (M a, M b). It is beneficial for any future analysis of DES, including the verification of the proper DES implementation in ”desScript” and my additional DES properties proving. At last, it proved that as a Feistel network, the implemented encryption of DES using the round keys, followed by a second encryption using the reversed round keys will return the original plaintext.

Consequently, it not only verifies that the decryption of DES is the same process of encryption but with reversed round keys, but also confirms the correctness of DES implementation in HOL4 by proving that decrypting the encrypted message with the same key will return the original message.

5 DES property in HOL4

In my ”des_propScript” file in HOL4, there are some basic settings. It loads all the relevant theories and libraries which contains necessary definitions and theorems I need to use including the ”desScript”. I set the ”guessing_word_-lengths” to true so that I do not need to set the length for each word, they will be assigned to a length based on the conditions. Additionally, I also define a simpleset ”fcp_ss”, it is a simplification set that combines ”std_ss”, the basic simplification set, with ”FCP_ss”, the simpset fragment specifically simplifies finite Cartesian product expressions.

5.1 Complementation property

First of all, I proved the complementation property, it is implemented as the theorem "comple_property".

$$\vdash 0 < n \land n < 17 \land \texttt{DES}\ n\ k = (encrypt, decrypt) \land$$
$$\texttt{DES}\ n\ (\neg k) = (encrypt', decrypt') \Rightarrow$$
$$\neg encrypt\ m = encrypt'\ (\neg m)$$

It means for any key k, message m and number of rounds n, the complementation of encryption using m and k is the same as the encryption using the complementation of m and k.

I initially convert the "encrypt" function to more detailed components functions including inverse of initial permutation, join, swap, round, split and initial permutation functions. This allows the use of rewriting rules on separated functions. This includes some theorems proving the complementation property of each individual function mentioned above. The below theorem is an example for join function.

$$\vdash \texttt{Join}\ (\neg m, \neg n) = \neg\texttt{Join}\ (m, n)$$

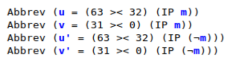
I then convert the "Round" function to the $(M\,a,\ M\,b)$ by the "half_message" help function, so that I can research on each 32-bit half of the block and use the relation between halves in the same and different rounds. Moreover, I simplify the proving goals by rewriting it with the theorems about the complementation property of each individual function, leave only solving the complementation property of a pair of M function.

$$(M\ (u', v')\ keys'\ n,\ M\ (u', v')\ keys'\ (SUC\,n)) = (\sim M\ (u, v)\ keys\ n,\ \sim M\ (u, v)\ keys\ (SUC\,n)) \quad (1)$$

I decide to use the Mathematical induction method using "Induct_on" in HOL4. I replace the variable n to x, define x to be <= n, and then perform induction on variable x. It is to avoid the failure of proving the induction step. As the prerequisite of using given case is $n > 0$, but in the step case, only $(SUCn) > 0$ (SUC means successor) is given. From $n+1 > 0$, it is not possible to derive the requirement of $n > 0$, thus given case cannot be used, verification cannot be pushed further. By converting to x, the prerequisites can be easily meet, and facilitating the proof process.

To prove the base case, we only need to convert back to the form of "Round" function and use the base case of the "Round" function. It is because only "Round" function is built using the relationships between its values based on inductively given inputs, and it has the base case for $x = 0$. To complete the base case verification, I also prove $(u,\ v) = (u',\ v')$ by extract the abbreviation of them back into their original forms, and apply the complementation rule of IP function. The abbreviation can be viewed below **??**.

It is important to note that $(M\,a,\ M\,b)$ can be converted from the "Round" function. It is defined as a pair of two halves taking nth round as one of the



```
Abbrev (u = (63 >< 32) (IP m))
Abbrev (v = (31 >< 0) (IP m))
Abbrev (u' = (63 >< 32) (IP (¬m)))
Abbrev (v' = (31 >< 0) (IP (¬m)))
```

Figure 4: The abbreviation

inputs, which are the two halves in the n-th round during the DES encryption process. According to the workflow of encryption, the right half of round n equals to the left half of round $n+1$ by the swap operation at the end of each round. Depend on this and the workflow, the right half of round $n+1$ can be expressed as the two halves in previous round with some operations. By using the relationship of halves between different rounds, each half of $(M\,a,\ M\,b)$ in round $n+1$ can be expressed by the halves in round n. Consequently, in the induction step, the step case which contains $(M\,a,\ M\,b)$ in round $SUC\,n$ can be expressed into formulas using $(M\,a,\ M\,b)$ in round n which are contained in the base case. As a result, I can use the assumptions provided in the base case to push forward my verification.

$$\begin{aligned} \mathit{Assumption}: \quad & (M\ (u',v')\ keys'\ x,\ M\ (u',v')\ keys'\ (SUC\,x)) \\ & = (\sim M\ (u,v)\ keys\ x,\ \sim M\ (u,v)\ keys\ (SUC\,x)) \end{aligned} \tag{2}$$

$$\begin{aligned} \mathit{Goal}: \quad & (M\ (u',v')\ keys'\ (SUC\,x),\ M\ (u',v')\ keys'\ (SUC\,(SUC\,x))) \\ & = (\sim M\ (u,v)\ keys\ (SUC\,x),\ \sim M\ (u,v)\ keys\ (SUC\,(SUC\,x))) \end{aligned} \tag{3}$$

By the above conversions and simplifications using base case, the proving goal is now transformed to verify only the complementation property about round function. It is to prove that round function with 32-bite message m and round key generated by key k equals to the round function with inputs of the complementation of both m and k. In HOL4, *RoundOp* function is built to represent round function of each round combining the expansion E, XOR operation with round key, S-boxes permutations S and the bit-level permutation P. By expanding the *RoundOp* function, the goal can now reduce to:

$$\begin{aligned} & P\ (S\ (E\ (\sim M\ (u,v)\ keys\ (x+1))\ \oplus\ (EL\ x\ keys'))) \\ & \qquad = P\ (S\ (E\ (M\ (u,v)\ keys\ (x+1))\ \oplus\ (EL\ x\ keys))) \end{aligned} \tag{4}$$

As the property of exclusive OR operation, $\sim\ a \oplus\ \sim\ b\ =\ a\ \oplus\ b$ and the support complementation property of the expansion function, $E\ (\sim\ \ a)\ \ = \sim E\ (a)$ once I prove the complementation of round key generated by key k is equal to the round key generated by the complementation of key k, the verification of complementation property in DES is done.

The implementation of the design about round key in HOL4 is generated by *RoundKey* and KS function. $RoundKey$ outputs a round key list that the elements in it are pairs of 28-bit words, each pair represents the two halves after the splitting of initial key, the rearrangement by PC1 and the rotation in different rounds. KS then using the MAP function to map each element in the output of $RoundKey$ to perform PC2 permutations. For convenience in the verification, I define three functions RK, RK_L and RK_R. $RoundKey$ can be converted to RK and thus split into $(RK\_L\ a\ , RK\_R\ a)$, and now I can work on each half of the pair separately, instead of only be able to work on the pair as a whole. The current form of goal can be expressed like:

$$\begin{aligned} &EL\ x\ (MAP\ f\ (list\ of\ input\ \sim k)) \\ &\qquad\qquad\qquad\qquad = \sim EL\ x\ (MAP\ f\ (list\ of\ input\ k)) \end{aligned} \tag{5}$$

By applying a series of rewriting rules in $listTheory$ that can deal with EL, MAP and some functions in f, the complementation theorems for RK_L and RK_R, the goal finally transforms to $MAP\ f\ l = MAP\ f'\ l$. At last, the complementation property of PC2 permutation is used to finish the proof of $f = f'$. The initial goal of DES complementation property is proved in HOL4 through the above steps.

As mentioned above, I proved many support theorems about the complementation property of each single operation, their proofs are done in a similar way. The operations are expanded by their definitions to transform the verification into each bit of $\sim m$ at the permuted indices equals to the complementation of each bit of m in the same indices. For example, the indices of IP permutation are transformed to $64 - EL\ (63 - i)\ IP_{data}$ where i is the original index of m before IP permutation. IP_{data} is the IP table to do the permutations, and it rearranges the bits in different positions of m to form a new message. Then I need to prove that the transformed indices are still less than the length of input message m to meet the prerequisite for using the theorem, FCP_BETA and then I can complete the proof.

$\vdash i <$ `dimindex (:`β`)` $\Rightarrow$ `(FCP)` g `'` i `=` $g\ i$

The proof that the transformed indices are all less than the length of input message m (64) given i is less than 64 uses the method of exhaustion. It repeatedly proves the transformed index is less than 64 for each value of i for 64 times, covering all cases from i=0 to i=63. The proof is completed as all 64 situations meet the condition. Overall, the support theorems are all proved with a similar way.

There are some special built-in tactics, conversions used here. The $MATCH\_MP\_TAC$ tactic takes a theorem as input, it can be applied to the goal if the current goal is in the form of the consequent of this theorem. The goal can then be converted to the antecedent of the theorem. rpt is a tactic that takes a tactic as input and repeatedly apply the tactic until it no longer succeeds. $CONV\_TAC$ is a special tactic that makes a tactic from a conversion. Then $BOUNDED\_FORALL\_CONV$ is a conversion that deal with universal quantification for bounded natural numbers, so it converts $\forall n.\ n\ <\ k$ to the conjunction of $n = 0$ to $n = k - 1$, like the exhaustion methods for i that I mentioned above.

There are also some theorems and definitions that are frequently used in the proof. $word_1 comp_d ef$ defines the complementation of a word m $\sim m$ equals to the concentration of the complementation of each bit of word m. It helps deal with the complementation in bit level.

$\vdash$ $\neg w$ `=` `FCP` i`.` $\neg w$ `'` i

When dealing with the lists and some operations applies to lists like the map function. MAP_TL shows that applying a function to the tail of a list is equal to the tail of a list after the mapping. Then the theorem MAP_MAP_o states applying two functions to a list using MAP twice is equivalent to applying the composition of the two function using MAP once.

$\vdash$ `MAP` f `(TL` l`) = TL (MAP` f l`)`

$\vdash$ `MAP` f `(MAP` g l`) = MAP (`$f \circ g$`)` l

5.2 Weak and semi-weak key property