Another Perspective on Provable Security. Another perspective on provable security is that a cryptographer turns lower-level primitives into higher-level primitives. Here “lower-level” means the primitive doesn’t solve any problem of practical interest but rather serves as a source of computational hardness. We will see lower-level primitives that come in two flavors, the first being “heuristic” like hash functions and blockciphers and the other being “mathematical” like the factoring problem. A higher-level primitive does solve a problem of practical interest. Cryptographers build higher-level primitives out of lower-level ones and prove that a higher-level primitive is “good” (i.e., secure) if the lower-level primitives it uses are.

Blockciphers. Blockciphers are a lower-level primitive which you can think of as a rudimentary form of symmetric-key encryption. Formally, let \( E : \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^\ell \) be a function. Here \( k \) is the key-length and \( \ell \) is the block-length. For every key \( K \in \{0,1\}^k \) we define the function \( E_K : \{0,1\}^\ell \to \{0,1\}^\ell \) by \( E_K(x) = E(K,x) \) for all inputs \( x \in \{0,1\}^\ell \). We say that \( E \) is a blockcipher if (1) \( E_K \) is a permutation for every \( K \in \{0,1\}^k \), and (2) \( E_K, E_K^{-1} \) are efficiently computable for every \( K \in \{0,1\}^k \) (here \( E_K^{-1} \) denotes the inverse function of \( E_K \), which exists by condition (1)). The first blockcipher was DES, designed by IBM in the 1970s. The design principles used for it go back to Shannon’s notions of “confusion,” that every bit of the key should influence an output bit, and “diffusion,” that each input bit should influence every output bit. DES has \( k = 56, \ell = 64 \).

Key Recovery Attacks and “Game-Playing.” What security do we want from a blockcipher? A first thought that comes to mind is security against key recovery attacks. That is, consider a game in which an adversary is given input-output examples \( E_K(M_1), \ldots, E_K(M_n) \) under an unknown random key \( K \). It should be hard for such an adversary to output \( K \). We will even give the adversary the power to choose \( M_1, \ldots, M_n \) here.

To formalize this, we will use a general framework for specifying security games. Formally, a game will consist of an Initialize procedure, a Finalize procedure, and some number of additional procedures \( P_1, \ldots, P_n \). To execute a game with an adversary, we first run the Initialize procedure to obtain some output, then run the adversary on this output. The adversary can make calls to \( P_1, \ldots, P_n \) as oracles while it runs. At some point the adversary halts with some output and we run the Finalize procedure on this output. The output of the game is whatever the Finalize procedure outputs. Procedures of the game share state (i.e., the same variables can be accessed by the procedures) and variables are implicitly initialized to a default value (false for boolean variables, empty for arrays).

So, here is the key recovery game \( KR_E \) associated to blockcipher \( E \):

\(^1\)By “primitive” we mean “notion” or a type of scheme/algorithm.
proc \textbf{Initialize} \\
\quad K \leftarrow K \\
proc \textbf{Fn}(M) \\
\quad \text{Return } E_K(M) \\
proc \textbf{Finalize}(K') \\
\quad \text{Return } (K = K')

As you can see above, there is just one procedure in addition to the Initialize and Finalize procedures. We also omit a return statement when nothing is returned, as is the case for the Initialize procedure. Finally we define the “key recovery advantage” of adversary $A$ against $E$, which is the probability the above game returns 1 when executed with $A$, that is

$$\text{Adv}_{E}^{Kr}(A) = \Pr[\text{KR}_{E}(A) \text{ outputs 1}].$$

Above $\text{KR}_{E}(A)$ denotes the execution of game $\text{KR}_E$ with $A$. As usual the probability is over any coins used in the game as well as those of the adversary.

**Exhaustive Key Search.** Let $M_1, M_2, \ldots$ be an enumeration of the inputs and $K_1, K_2, \ldots$ be an enumeration of the keys. Consider a $q$-query exhaustive key search adversary $A_q$ defined as follows:

adversary $A_q$

\begin{align*}
\text{For } i = 1 \text{ to } q \text{ do: } & C_i \leftarrow \text{Fn}(M_i) \\
\text{For } j = 1 \text{ to } 2^k \text{ do: } & \\
& \quad \text{If } E_{K_j}(M_i) = C_i \text{ for all } 1 \leq i \leq q \\
& \quad \text{Then return } K_j
\end{align*}

The running-time of $A_q$ is $q + 2^k$ computations of $E$. (Our convention is to include in the running-time of an adversary the size of its code as well as the running-time of its overlying game.) But these can be done in parallel, and hence with $k = 56$ DES is effectively “broken” by exhaustive key search. Question: Is it the case that for every $E$ there is a $q$ such that $\text{Adv}_{E}^{Kr}(A_q) = 1$? This is a tricky question because there could be multiple “consistent” keys. But typically for “large enough” $q$ we do recover the “target” key.

**Increasing Key Length.** Since a lot of effort went into the design of DES, a natural question is whether there is there a way of using DES to get a blockcipher of longer key-length which resists key recovery attacks. One attempt is 2DES, defined as

$$2\text{DES}_{K_1, K_2}(X) = \text{DES}_{K_2}(\text{DES}_{K_1}(X))$$

so the key-length of 2DES is 128 but the block-length remains the same. We would like that any key-recovery attack therefore has time-complexity about $2^{57}$, but unfortunately there is a meet in the middle attack on 2DES which recovers the key with time complexity $2^{57}$ computations of DES. (In other words the “effective” key-length of 2DES is only 57 bits, so effectively nothing is gained over DES itself.) It is a good exercise to find this attack.

One can generalize in the natural way, defining 3DES via

$$3\text{DES}_{K_1, K_2, K_3}(X) = \text{DES}_{K_3}(\text{DES}_{K_2}(\text{DES}_{K_1}(X)))$$

In general, this technique is known as “cascading” the block cipher or cascade encryption. 3DES is in fact widely standardized and used. A meet in the middle attack reduces its effective key-length to
112 bits. There is also a proof that it’s at least 78 bits, see https://eprint.iacr.org/2004/331.pdf and http://www.iacr.org/archive/asiacrypt2009/59120035/59120035.pdf, although this turns out to be quite hard to prove.

However, block-length remains the same. We will see “birthday” attacks that break a blockcipher in time $2^{\ell/2}$ so we would like a larger block-length as well. AES was chosen to replace DES in 2001, and it has $k, \ell = 128$. Dedicated hardware support (AES-NI) makes it very fast, about 1 cycle per byte.

**Other Security Properties?** What else is important besides key recovery security?