Private-key encryption is perhaps the most basic cryptographic task. In the simplest model of encryption there are two honest participants, Alice and Bob, who interact over a communication channel. The channel interaction is observed by a third party Eve who may be malicious. Alice's goal is to send a single message $M$ to Bob so that Bob can recover the message but Eve cannot obtain information about what was sent.

In the private-key setting, Alice and Bob are assumed to have agreed upon a common key $K$ that is not known to Eve. Let us model messages and keys as binary strings: The message $M$ can be any string from the message space $\{0,1\}^m$, and the key $K$ is a uniformly random string from $\{0,1\}^n$.

If $n$ is at least as large as $m$, the following simple solution called the one-time pad achieves perfect secrecy: Alice encrypts the message $M$ under key $K$ into the ciphertext $M \oplus K$ obtained by taking the pairwise XOR of the bits of $M$ and $K$. Upon receiving $C$, Bob decrypts to $C \oplus K$. Clearly the decryption is correct. Intuitively, it is also secret because no matter what $M$ is, $M \oplus K$ is a uniformly random string in $\{0,1\}^m$, so the distribution that Eve observes is completely independent of the message being sent.

However the assumption that $n$ is at least as large as $m$ is often unrealistic. In usual applications Alice and Bob want to agree on a fairly short key (at most several thousand bits) and use it to encrypt much longer messages (megabytes or gigabytes long). But even if $m = n+1$ it is impossible to make the encryption of a message statistically independent of the message.

The solution is to extend the $n$-bit key $K$ into an $m$-long bit string $G(K)$ which "looks" uniformly random, even though it is statistically far from being random. Alice now encrypts by sending $M \oplus G(K)$, and Bob decrypts by computing $C \oplus G(K)$. From Eve's perspective, $G(K)$ looks like a uniformly random string in $\{0,1\}^n$, and so does $M \oplus G(K)$.

# 1 Pseudorandom generators and one-way permutations

What does it mean for a string $y$ coming from some distribution over $\{0,1\}^m$ to "look" uniformly random? Let's ask the opposite question – what does it mean for $y$ to *not* look random? It means that we should have some way of distinguishing $y$ from a uniformly random string $u$ of the same length. In computational complexity and cryptography, we model the distinguisher as an efficient algorithm that takes $y$ or $u$ as an input, tends to accept when its input is $y$, and tends to reject when its input is $u$.

This suggests the following definition: A distribution $\mathcal{Y}$ over $\{0,1\}^m$ is $(s,\varepsilon)$-pseudorandom if for every algorithm $D$ of complexity at most $s$,

$$\Pr_{y\sim\mathcal{Y}}[D(y) \text{ accepts}] - \Pr_{u\sim\{0,1\}^m}[D(u) \text{ accepts}] \leq \varepsilon.$$

We won't define complexity formally, but you can think of it as the size of the program for $D$ plus the worst-case running time of this program on inputs of length $m$. A *pseudorandom generator* is an algorithm that takes $n$ uniformly random bits and expands them deterministically into $m$ pseudorandom bits.

**Definition 1.** A function $G\colon \{0,1\}^n \to \{0,1\}^m$, where $m > n$, is an $(s,\varepsilon)$ pseudorandom generator if for every algorithm $D$ of complexity at most $s$,

$$\Pr_{x\sim\{0,1\}^n}[D(G(x)) \text{ accepts}] - \Pr_{u\sim\{0,1\}^m}[D(u) \text{ accepts}] \leq \varepsilon.$$

In a typical application like private-key encryption, we may think of the input length $n$ as being 1000 or 2000 bits long, while $s$ as much larger and $\varepsilon$ as tiny, e.g. $s = 2^{100}$ and $\varepsilon = 2^{-100}$. What about the output length $m$? Once we have a pseudorandom generator that produces $n+1$ bits of output, we can bootstrap it to obtain as many output bits as we want, so we will focus on the case $m = n+1$.

It is somewhat tricky to construct pseudorandom generators because the definition requires us to argue about all possible distinguishers $D$ and we may not know how such a distinguisher works. It may be easier to build pseudorandom generators out of potentially more primitive objects.

One such object are one-way permutations. A one-way function is a function that is easy to compute, but hard to invert, even for random inputs. A one-way permutation is a pseudorandom function that is also a permutation, i.e. every output comes from exactly one input.

**Definition 2.** A permutation $\pi\colon \{0,1\}^n \to \{0,1\}^n$ is $(s,\varepsilon)$-one-way if for every algorithm $Inv$ of complexity at most $s$, $\Pr_{x\sim\{0,1\}^n}[Inv(\pi(x)) = x] \leq \varepsilon$.

In 1982 Yao showed how to obtain a pseudorandom generator from any one-way permutation. His construction was simplified considerably by Goldreich and Levin who proved the following theorem:

**Theorem 3** (Goldreich and Levin)**.** *If $f\colon \{0,1\}^n \to \{0,1\}^n$ is a $(\mathrm{poly}(n/\varepsilon)(s+s_\pi), \varepsilon/2)$ one-way permutation of complexity $s_\pi$, then the function $G\colon \{0,1\}^{2n} \to \{0,1\}^{2n+1}$ given by*

$$G(x,r) = (\pi(x), r, \langle x, r\rangle)$$

*is an $(s,\varepsilon)$-pseudorandom generator.*

## 2 Fourier analysis of the Hadamard code

The proof of the Goldreich-Levin theorem is closely related to algorithmic aspects of decoding the $[2^n, n, 2^n/2]$ Hadamard code. (We now change convention and use $n$ to denote message length and not block length as before.) Suppose we are given a corrupted codeword $f$ of the Hadamard code. We can decode $f$ by brute force: Look at all $2^n$ possible codewords $Had_a$, compute their distances to $f$ and output the one that is closest to $f$. Since the block length is $2^n$, the running time of this decoding algorithm is about $2^{2n}$.

Can we decode any faster? The corrupted codeword $f$ is $2^n$ bits long, so merely inspecting the whole codeword will take $2^n$ time. This suggests we may not be able to substantially improve upon the brute-force algorithm. However, this intuition is incorrect: We will show how to perform the decoding by only inspecting a small number of random entries inside the codeword.

We will in fact solve a more general problem called *list-decoding*. Recall that in a code of distance $d$, decoding is only possible (in the worst case) if the number of errors $t$ is at most $(d-1)/2$. If $t$ is larger, there may be ambiguity in the decoding as there can be more than one answer within

distance $t$ of the corrupted codeword. In this setting, a sensible possibility would be to ask for a description of all codewords within distance $t$. The maximum number of such codewords is called the *list size* of the code at radius $t$.

Recall that the Hadamard encoding of a message $a$ in $\{0,1\}^n$ consists of the evaluations $\langle a, x\rangle \bmod 2$ over all $x$ in $\{0,1\}^n$. Let's represent the codeword entries by $\{1,-1\}$ instead of $\{0,1\}$. Then the encoding of $a$ consists of the evaluations of the character function $\chi_a(x) = (-1)^{\langle a,x\rangle}$. We will identify the codewords of the Hadamard code with the character functions.

Under this convention, a corrupted codeword can be viewed as some function $f\colon \{0,1\}^n \to \{1,-1\}$. The list decoding problem asks us to find all codewords $\chi_a$ that has large agreement with the function $f$; specifically, given an agreement parameter $\varepsilon > 0$, we want all $a$ such that $\Pr_{x\sim\{0,1\}^n}[f(x) = \chi_a(x)] \geq (1+\varepsilon)/2$, or equivalently all $a$ such that

$$\hat{f}_a = \mathrm{E}[f(x)\chi_a(x)] \geq \varepsilon.$$

From this Fourier-analytic point of view, the list size of the Hadamard code can be bounded immediately via Parseval's identity: Every codeword $\chi_a$ in the list must contribute $\hat{f}_a^2 \geq \varepsilon^2$ to the square sum of the Fourier coefficients, so the list size of the Hadamard code can be at most $1/\varepsilon^2$.

## 3 The Kushilevitz-Mansour algorithm

We will generalize our objective a little bit and seek to find all $a$ such that $\hat{f}_a^2 \geq \varepsilon^2$, and maybe even allow for a few $a$s that don't quite satisfy this condition. The idea is to try to locate these relevant $a$s by a divide-and-conquer strategy. One nice way to visualize this strategy is as a search process along the following full binary tree of depth $n$. The root of this binary tree is labeled by the value $\sum_{a\in\{0,1\}^n} \hat{f}_a^2$. Its left and right children are labeled by the partial sums

$$\sum_{a\colon a_1=0} \hat{f}_a^2 \quad \text{and} \quad \sum_{a\colon a_1=1} \hat{f}_a^2.$$

In general, a node at level $i$ is indexed by a string $v \in \{0,1\}^i$ and is labeled by the value

$$\sum_{a\colon a_1=v_1,\ldots,a_i=v_i} \hat{f}_a^2$$

so that the leaf indexed by $a$ is labeled by $\hat{f}_a^2$.

Let's say a node $v$ is relevant if its label is at least $\varepsilon^2$. Although there are exponentially many nodes in the tree, there can be at most $n/\varepsilon^2$ relevant ones because the labels in each level sum to 1. If we could calculate the labels, it would be easy to identify all the relevant nodes via depth-first search starting at the root and pruning the search path at irrelevant nodes.

How do we calculate the values of the labels? Using the Fourier coefficient formula

$$\hat{f}_a = \mathrm{E}[f(x)\chi_a(x)] \tag{1}$$

we can obtain these values in time exponential in $n$. But if we are willing to settle for a probabilistic approximation, we can do much better. Let's start at the leaves. From the formula (1) we get

$$\hat{f}_a^2 = \mathrm{E}[f(x)\chi_a(x)]\,\mathrm{E}[f(y)\chi_a(y)] = \mathrm{E}[f(x)f(y)\chi_a(x+y)].$$

This suggests that to estimate $\hat{f}_a^2$, we ought to sample some number of random pairs $(x, y)$ and output the average of the values $f(x)f(y)\chi_a(x+y)$.

Now let $v \in \{0,1\}^i$ be an arbitrary node in the tree at level $i$ and $FIX(v)$ be the set of those $a \in \{0,1\}^n$ with $a_1 = v_1, \ldots, a_i = v_i$. We want to estimate the value

$$\sum_{a \in FIX(v)} \hat{f}_a^2 = \mathrm{E}\left[ f(x)f(y) \sum_{a \in FIX(v)} \chi_a(x+y) \right].$$

The set $FIX(v)$ could be exponentially large so we have to be a bit careful here. Recall that $\chi_a(z) = (-1)^{\langle a,z \rangle}$ so:

$$\sum_{a \in FIX(v)} \chi_a(z) = \sum_{a \in FIX(v)} (-1)^{\langle a,z \rangle}$$

If $z$ is nonzero along any of the coordinates $i+1$ up to $n$, this sum vanishes; otherwise, it equals $2^{n-i}\chi_v(z)$. So the only $(x, y)$ pairs that contribute to the sum are those in which $x$ and $y$ agree on the last $n-i$ coordinates, and we can rewrite the identity as

$$\sum_{a \in FIX(v)} \hat{f}_a^2 = \mathrm{E}_{x',y' \sim \{0,1\}^i, u \sim \{0,1\}^{n-i}}[f(x'u)f(y'u)\chi_v(x'+y')].$$

Here, the first $i$ bits $x'$ and $y'$ of $x$ and $y$ are chosen independently at random, while the last $n-i$ bits are random but identical in $x$ and $y$. (When $i = 0$ the right side equals $\mathrm{E}[f(u)^2] = 1$, which is a good sign.)

We now have all the ingredients for the Kushilevitz-Mansour algorithm. First, we have a probabilistic procedure $\hat{\mathbf{S}}\mathbf{amp}(f, v)$ which estimates the label of node $v$ as follows: Sample $O(n/\varepsilon^6)$ random triples $(x', y', u)$ and output the average of the values $f(x'u)f(y'u)\chi_v(x'+y')$.

**Lemma 4.** *With probability at least $1 - \varepsilon^2/20n$, $\hat{\mathbf{S}}\mathbf{amp}(f, v)$ outputs a value between $\ell(v) - \varepsilon^2/3$ and $\ell(v) + \varepsilon^2/3$, where*

$$\ell(v) = \sum_{a\colon a_1 = v_1, \ldots, a_i = v_i} \hat{f}_a^2.$$

Now here is the Kushilevitz-Mansour algorithm:

> **Algorithm KM:** On input a function $f\colon \{0,1\}^n \to \{1,-1\}$ and $\varepsilon > 0$,
> Apply the following recursive procedure $\mathbf{P}(v)$ starting with $v$ equal to the empty string:
> If $\hat{\mathbf{S}}\mathbf{amp}(f, v) \geq 2\varepsilon^2/3$:
> If $v$ has length $n$, output $v$.
> Otherwise, call $\mathbf{P}(v0)$ and $\mathbf{P}(v1)$.

**Theorem 5.** *With probability at least $1/2$, the outputs of $\mathbf{KM}(f, \varepsilon)$ include all $a$ such that $\hat{f}_a^2 \geq \varepsilon^2$, but it produces no more than $O(n/\varepsilon^2)$ outputs in total.*

*Proof.* Let $v$ be any node such that $\ell(v) \geq \varepsilon^2$. By Lemma 4,

$$\Pr[\hat{\mathbf{S}}\mathbf{amp}(f) < 2\varepsilon^2/3] \leq \varepsilon^2/20n$$

Since there are at most $n/\varepsilon^2$ such nodes $v$, by a union bound we have

$$\Pr[\hat{\mathbf{S}}\mathbf{amp}(f) < 2\varepsilon^2/3 \text{ for some } v \text{ s.t. } \ell(v) \geq \varepsilon^2] \leq \frac{n}{\varepsilon^2} \cdot \frac{\varepsilon^2}{20n} \leq \frac{1}{20}.$$

Therefore, all $a \in \{0,1\}^n$ such that $\ell(a) = \hat{f}_a^2 \geq \varepsilon^2$ will be included in the output of $\mathbf{KM}(f, \varepsilon)$ with probability at least $1 - 1/20 = 19/20$.

Let $B$ be the set of nodes whose label exceeds $\varepsilon^2/3$ and $B'$ be the set of nodes outside $B$ whose parent node is in $B$. Since the nodes in $B$ form a tree, we must have $|B'| \leq |B| + 1$. There must be fewer than $3n/\varepsilon^2$ nodes in $B$, so $B'$ can have at most $3n/\varepsilon^2 + 1$ nodes. By a very similar calculation as above,

$$\Pr[\hat{\mathbf{Samp}}(f, v) \geq 2\varepsilon^2/3 \text{ for some } v \text{ in } B'] \leq \left(\frac{3n}{\varepsilon^2} + 1\right) \cdot \frac{\varepsilon^2}{20n} \leq \frac{1}{5}.$$

Therefore, with probability at least 4/5, $\hat{\mathbf{Samp}}(f, v)$ will output a value smaller than than $2\varepsilon^2/3$ on all nodes $v$ in $B'$, so $\mathbf{KM}(f, \varepsilon)$ will not make any recursive calls to $\mathbf{P}$ on a node outside $B \cup B'$. Since there are at most $O(n/\varepsilon^2)$ nodes inside $B \cup B'$, $\mathbf{KM}(f, \varepsilon)$ can produce at most this many outputs.

With probability at least $1 - 1/20 - 1/5 \geq 1/2$, both of these conditions are met. □

It remains to prove Lemma 4. We make use of Chebyshev's inequality:

**Theorem 6** (Chebyshev's inequality)**.** *For any random variable $X$ and $t > 0$,*

$$\Pr\left[|X - \mathrm{E}[X]| > t\sqrt{\mathrm{Var}[X]}\right] < 1/t^2.$$

*Proof of Lemma 4.* Let $X_i = f(x_i'u_i)f(y_i'u_i)\chi_v(x_i'+y_i')$, where $(x_i', y_i', u_i)$ is the $i$-th sample. $\hat{\mathbf{Samp}}(f, v)$ outputs the value $X = \frac{1}{m}(X_1 + \cdots + X_m)$, where $m$ is the number of samples used. By linearity of expectation,

$$\mathrm{E}[X] = \frac{1}{m}(\mathrm{E}[X_1] + \cdots + \mathrm{E}[X_m]) = \mathrm{E}[f(x'u)f(y'u)\chi_v(x' + y')] = \ell(v)$$

and by independence of $X_i$ and $X_j$ for every pair $i \neq j$,

$$\mathrm{Var}[X] = \frac{1}{m^2}(\mathrm{Var}[X_1] + \cdots + \mathrm{Var}[X_m]) \leq \frac{1}{m}$$

since the variables $X_1, \ldots, X_m$ are $\{-1, 1\}$ valued and can have variance at most 1. From Chebyshev's inequality we get that

$$\Pr\left[|X - \ell(v)| > t/\sqrt{m}\right] < 1/t^2.$$

To get the desired conclusion, we choose $m$ and $t$ so that $t/\sqrt{m} = \varepsilon^2/3$ and $1/t^2 = \varepsilon^2/20n$. □

## 4 Proof of the Goldreich-Levin theorem

We prove the contrapositive statement: Suppose that $G$ is not an $(s, \varepsilon)$-pseudorandom generator, namely there is a distinguisher $D$ of complexity $s$ such that

$$\Pr_{x,r \sim \{0,1\}^n}[D(G(x, r)) \text{ accepts}] - \Pr_{u \sim \{0,1\}^{2n+1}}[D(u) \text{ accepts}] > \varepsilon.$$

We will argue that there is then an algorithm $Inv$ of complexity $\mathrm{poly}(n/\varepsilon)(s + s_\pi)$ such that

$$\Pr_{x \sim \{0,1\}^n}[Inv(\pi(x)) = x] > \varepsilon/2$$

and so $\pi$ is not $(\mathrm{poly}(n/\varepsilon)(s+s_\pi), \varepsilon/2)$-one-way.

Without loss of generality, let us assume that $D$ outputs 1 when it accepts and $-1$ when it rejects. Because $\mathrm{E}[D(\cdot)] = 2\Pr[D(\cdot)=1]-1$, we can rewrite our assumption on $D$ as

$$\mathrm{E}_{x,r\sim\{0,1\}^n}[D(G(x,r))] - \mathrm{E}_{u\sim\{0,1\}^{2n+1}}[D(u)] > 2\varepsilon.$$

Unwinding the definition of $G$, we get

$$\mathrm{E}_{x,r\sim\{0,1\}^n}[D(\pi(x),r,\langle x,r\rangle)] - \mathrm{E}_{u\sim\{0,1\}^{2n+1}}[D(u)] > 2\varepsilon.$$

We can write $u$ in the form $(\pi(x),r,b)$, where $x,r\sim\{0,1\}^n$ and $b\sim\{0,1\}$ are independent. (Since $\pi$ is a permutation, $(\pi(x),r,b)$ is uniformly distributed in $\{0,1\}^{2n+1}$.)

$$\mathrm{E}_{x,r\sim\{0,1\}^n}[D(\pi(x),r,\langle x,r\rangle)] - \mathrm{E}_{x,r\sim\{0,1\}^n,b\sim\{0,1\}}[D(\pi(x),r,b)] > 2\varepsilon.$$

We now make use of the following technical lemma. This lemma tells us that if $F(X)$ is distinguishable from $F(\tilde{X})$, then $\tilde{X}F(\tilde{X})$ can predict $X$ to some advantage.

**Lemma 7.** *Let $F(-1), F(1) \sim \mathbb{R}$ and $X\sim\{-1,1\}$ be (possibly dependent) random variables, and $\tilde{X}\sim\{-1,1\}$ be uniformly random and independent of $F$ and $X$. Then*

$$\mathrm{E}[\tilde{X}F(\tilde{X})\cdot X] = \mathrm{E}[F(X)] - \mathrm{E}[F(\tilde{X})].$$

Applying the lemma to $F(\cdot) = D(\pi(x),r,\cdot)$, $X=(-1)^{\langle x,r\rangle}$, and $\tilde{X}=(-1)^b$ we get that

$$\mathrm{E}_{x,b,r}[(-1)^b D(\pi(x),r,b)\cdot(-1)^{\langle x,r\rangle}] > 2\varepsilon$$

from where

$$\mathrm{E}_{x,b}[\mathrm{E}_r[(-1)^b D(\pi(x),r,b)\cdot(-1)^{\langle x,r\rangle}]] > 2\varepsilon$$

It follows that with probability at least $\varepsilon$ over the choice of $x$ and $b$, we must have

$$\mathrm{E}_r[(-1)^b D(\pi(x),r,b)\cdot(-1)^{\langle x,r\rangle}] > \varepsilon. \tag{2}$$

Now consider the following algorithm $Inv$: On input $\pi(x)$, choose a random $b$ and run $\mathbf{KM}(f,\varepsilon)$, where $f(r) = (-1)^b D(\pi(x),r,b)$. If the output of $\mathbf{KM}(f,\varepsilon)$ contains an $a$ such that $\pi(a)=\pi(x)$, output this $a$.

If $x$ and $b$ satisfy (2), then by Theorem 5 with probability at least $1/2$, the output of $\mathbf{KM}(f,\varepsilon)$ will contain $x$, and $Inv(\pi(x))$ outputs $x$ with probability at least $\varepsilon/2$.

We now analyze the running time of $Inv$. From Theorem 5 (more precisely, from its proof) it follows that algorithm $\mathbf{KM}$ makes no more than $O(n/\varepsilon^2)$ calls to $\hat{\mathbf{S}}\mathbf{amp}$, and each of these calls results in $O(n/\varepsilon^6)$ evaluations of $D$. Since each evaluation of $G$ has complexity $s$, the complexity of this part of the algorithm is $O(n^2/\varepsilon^8)\cdot s$. In addition, $Inv$ evaluates $\pi$ on the $O(n/\varepsilon^2)$ outputs of $\mathbf{KM}$. This part has complexity $O(n/\varepsilon^2)\cdot s_\pi$. Thus $Inv$ has complexity $O(n/\varepsilon^2)s_\pi + O(n^2/\varepsilon^8)s = \mathrm{poly}(n/\varepsilon)(s+s_\pi)$.

*Proof of Lemma 7.* Let $P = F(\tilde{X})(1+X\tilde{X})$. Since $\tilde{X}$ is random and independent of $F, X$ we have

$$\mathrm{E}[P] = \tfrac{1}{2}\mathrm{E}[P \mid X=\tilde{X}] + \tfrac{1}{2}\mathrm{E}[P\mid X\neq\tilde{X}] = \tfrac{1}{2}\mathrm{E}[2F(X)] + \tfrac{1}{2}\cdot 0 = \mathrm{E}[F(X)].$$

Therefore $\mathrm{E}[F(\tilde{X})(1+X\tilde{X})] = \mathrm{E}[F(X)]$. The lemma follows by linearity of expectation. □