MATH456 - 0301

Ash Dorsey
ash@ash.lgbt

Last updated 2026-06-18.

Contents

1. Symmetric key encryption

1.1. Formally defining a symmetric key encryption scheme

The sender and receiver share a random key, that will be kept secret. This key-sharing occurred as a pre-processing step, where you could consider they met up and generated a random key together.

$$
\begin{aligned}
S &\text{ sender} \\
R &\text{ receiver} \\
k &\text{ key} \\
m &\text{ message} \\
c &\text{ ciphertext}
\end{aligned}
$$

We run a key generation algorithm:

$$k \leftarrow \text{Gen}()$$

This uses an arrow because Enc could be something that is random.

$$
\begin{aligned}
c &\leftarrow \text{Enc}(k, m) \\
&\leftarrow \text{Enc}_k(m)
\end{aligned}
$$

$$
\begin{aligned}
m &= \text{Dec}(k, c) \\
&= \text{Dec}_k(c)
\end{aligned}
$$

1.2. Correctness

Correctness requires that $\text{Dec}_k(\text{Enc}_k(m)) = m$

1.3. Cartesian Product

e.g. $\{0,1\} \times \{0,1\} = \{00, 01, 10, 11\}$

Then, $\{0,1\}^5 = \{00000, 00001, 00010, 00011, ...\}$.

So

$$\mathbb{S}^n = \begin{cases} \mathbb{S} & n = 1 \\ \mathbb{S} \times \mathbb{S}^{n-1} & \text{otherwise} \end{cases}$$

1.4. Cardinality

$|\mathbb{S}|$ is the number of elements in the set $\mathbb{S}$. In this class, $|\mathbb{S}| < \infty$.

1.5. Syntax

An encryption scheme is defined via three algorithms, $\text{Gen}, \text{Enc}, \text{Dec}$.

The message space M has $|\mathbb{M}| > 1$.

1.5.1. Key generation algorithm

Gen is probabilistic. It will output a key k according to some distribution. The keyspace $\mathbb{K}$ is the set of all possible keys.

1.5.2. Encryption algorithm

Enc takes an input key $k \in \mathbb{K}$ and message $m \in \mathbb{M}$. It may be probabilistic, and outputs a ciphertext $c \leftarrow \text{Enc}_k(m)$. The ciphertext space C is the set of all possible ciphertexts.

1.5.3. Decryption algorithm

Dec takes an input key $k \in \mathbb{K}$ and ciphertext $c \in \mathbb{C}$. Decryption must be deterministic. It outputs message $m := \mathrm{Dec}_{k(c)}$.

1.6. Distributions

For $k \in \mathbb{K}$, $\Pr[K = k]$ denotes the probability that the key output by Gen is equal to k. The distribution over $\mathbb{K}$ is defined by running Gen and taking the output.

For $m \in \mathbb{M}$, $\Pr[M = m]$ denotes the probability that the message is equal to m. This models the a priori knowledge of the adversary about the message, such as the message is english text.

Distributions over $\mathbb{K}$ and $\mathbb{M}$ are independent.

For $c \in \mathbb{C}$, $\Pr[C = c]$ denotes the probability that the ciphertext is c. Given Enc, the distribution over $\mathbb{C}$ is fully determined by the distributions over $\mathbb{K}$ and $\mathbb{M}$.

1.7. Adversary

An adversary will see the ciphertext c, but does not know the key k.

1.8. Definition: Perfect Secrecy

An encryption scheme $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ over a message space $\mathbb{M}$ is perfectly secret if for every probability distribution over $\mathbb{M}$, every message $m \in \mathbb{M}$, and every ciphertext $c \in \mathbb{C}$ for which $\Pr[C = c] > 0$:

$$\Pr[M = m | C = c] = \Pr[M = m]$$

In other words, knowing the ciphertext does not give you any additional information about the message.

1.9. Lemma

An encryption scheme $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ over a message space $\mathbb{M}$ is perfectly secret iff for every probability distribution over $\mathbb{M}$, every message $m \in \mathbb{M}$, and every ciphertext $c \in \mathbb{C}$:

$$\Pr[C = c | M = m] = \Pr[C = c]$$

In other words, knowing the message doesn't give you additional information about the ciphertext.

If you know the message but not the key, you don't get additional information about the ciphertext.

1.9.1. Proof

Let M be a probability distribution, let $m \in \mathbb{M}$, and let $c \in \mathbb{C}$.

$$\begin{aligned}
&\Pr[M = m | C = c] && = \Pr[M = m] \\
\leftrightarrow\ &\frac{\Pr[M = m \wedge C = c]}{\Pr[C = c]} && = \Pr[M = m] \\
\leftrightarrow\ &\Pr[M = m \wedge C = c] && = \Pr[C = c] \Pr[M = m] \\
\leftrightarrow\ &\frac{\Pr[C = c \wedge M = m]}{\Pr[M = m]} && = \Pr[C = c] \\
\leftrightarrow\ &\Pr[C = c | M = m] && = \Pr[C = c]
\end{aligned}$$

Therefore, it is perfectly secret by definition.

1.9.1.1. Alternative Proof

Via the definition of perfect secrecy:

$$\Pr[C = c | M = m] = \Pr[C = c]$$

$$\begin{aligned}
&\xleftrightarrow{\text{Bayes' theorem}} \frac{\Pr[M=m|C=c]\Pr[C=c]}{\Pr[M=m]} && = \Pr[C=c] \\
&\leftrightarrow \Pr[M=m|C=c]\Pr[C=c] && = \Pr[M=m]\Pr[C=c] \\
&\leftrightarrow \Pr[M=m|C=c] && = \Pr[M=m]
\end{aligned}$$

1.10. Perfect Indistinguishability

An encryption scheme $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ over a message space $\mathbb{M}$ is perfectly secret iff for every $m_0, m_1 \in M$ and every ciphertext $c \in C$:

$$\Pr[C=c|M=m_0] = \Pr[C=c|M=m_1]$$

The distribution over ciphertexts is the same when you start from m_0 or m_1 for any pair (even adversarially chosen) of $m_0, m_1 \in \mathbb{M}$.

1.11. One-Time-Pad

Let $n \in \mathbb{N}$.

Let the message space be:

$$\mathbb{M} = \{0,1\}^n$$

(Bit strings of length n).

Key generation outputs a uniformly random $k \in \{0,1\}^n$.

Then encryption is defined as $\mathrm{Enc}_k(m) \to m \oplus k$. Note $\mathbb{C} = \{0,1\}^n$.

Encryption is then defined as $\mathrm{Dec}_{k(c)} = c \oplus k$

This works because

$$\begin{aligned}
& c \oplus k \\
&= (m \oplus k) \oplus k \\
&= m \oplus (k \oplus k) \\
&= m \oplus (0^n) \\
&= m
\end{aligned}$$

1.11.1. Example.

Take $\{0,1\}^3$ as the message space, and therefore $\{0,1\}^3$ as the key space

Encrypting

$$\begin{aligned}
&101 \\
&011 \\
&110
\end{aligned}$$

Decrypting

$$\begin{aligned}
&110 \\
&011 \\
&101
\end{aligned}$$

1.11.2. Lemma

Fix $m \in \mathbb{M}$, $c \in \mathbb{C}$

$$\begin{aligned}
& \Pr[C = c \mid M = m] \\
&= \Pr[(M \oplus K) = c \mid M = m]
\end{aligned}$$

$$
\begin{aligned}
&= \Pr[K = c \oplus M \mid M = m] \\
&\overset{\text{definition of conditional probability}}{=} \frac{\Pr[K = c \oplus M \wedge M = m]}{\Pr[M = m]} \\
&= \frac{\Pr[K = c \oplus m \wedge M = m]}{\Pr[M = m]} \\
&\overset{\text{independent events}}{=} \frac{\Pr[K = c \oplus m]\cancel{\Pr[M = m]}}{\cancel{\Pr[M = m]}} \\
&= \Pr[K = c \oplus m] \\
&\overset{\text{uniform distribution}}{=} \frac{1}{|K|} = \frac{1}{2^n}
\end{aligned}
$$

1.11.3. Proof

We aim to show that one-time-pad is perfectly secure. Via 1.10. Perfect Indistinguishability, we change to perfect indistinguishability.

Fix $m_0, m_1 \in \mathbb{M}$, and fix $c \in \mathbb{C}$.

Via the lemma, $\Pr[C = c \mid M = m_0] = 1/2^n$, and $\Pr[C = c \mid M = m_1] = 1/2^n$, and $1/2^n = 1/2^n$, and therefore:

$$\Pr[C = c \mid M = m_0] = \Pr[C = c \mid M = m_1]$$

1.11.4. Drawbacks

The key length is the same as the message length, and therefore for every bit communicated over a public channel, a bit must be shared privately. Via the (future) Shannon's theorem, this is an inherent problem in perfectly secret encryption schemes.

Furthermore, the key can only be used once, which is also an inherent problem.

1.12. Small Keys imply not perfectly secret

Let $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ be a perfectly secret encryption scheme over a message space $\mathbb{M}$ and let $\mathbb{K}$ be the key space as determined by Gen, then $|\mathbb{K}| \geq |\mathbb{M}|$.

1.12.1. Proof

Assume we have an encryption scheme $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ such that $|\mathbb{K}| < |\mathbb{M}|$.

On the contrary, assume $|\mathbb{K}| < |\mathbb{M}|$.

Fix the uniform distribution over the message space. Fix an arbitrary $c \in \mathbb{C}$.

Let $\mathbb{M}(c) = \{m' \mid \exists k \in \mathbb{K}, \mathrm{Dec}_k(c) = m'\}$

Since there exists a function $\mathbb{K} \to \mathbb{M}$, $|\mathbb{M}(c)| \leq |\mathbb{K}|$. By assumption, $|\mathbb{K}| < |\mathbb{M}|$.

These imply that $|\mathbb{M}(c)| < |\mathbb{M}|$ by transitivity and therefore we have additional information by knowing the ciphertext. Now, we need to show the actual definition.

Let $m^* \in \mathbb{M}$, $m \notin \mathbb{M}(c)$.

By the uniform distribution, $\Pr[M = m^*] = 1/|M|$. But, since $m^* \notin \mathbb{M}(c)$, and the scheme is correct, $\Pr[M = m^* \mid C = c] = 0$.

1.13. Bad attempt at a definition of perfect security

An encryption scheme over the message space $\mathbb{M}$ is perfectly secret if for every probability distribution over $\mathbb{M}$, every ciphertext $C \in \mathbb{C}$ for which $\Pr[C = c] > 0$ and every $k \in \mathbb{K}$,

$$\Pr[K = k \mid C = c] = \Pr[K = k]$$

In english, knowing the ciphertext does not give you additional information about the key.

As an example, this does not work, since if you scheme has $|K| = 1$, then $\Pr[K = k \mid C = c] = 1 = \Pr[K = k]$

1.14. Another bad definition

An encryption scheme with message space $\mathbb{M}$ is perfectly secret if and only if for every probability distribution over $\mathbb{M}$, every $m, m' \in \mathbb{M}$ and every $c \in \mathbb{C}$ we have

$$\Pr[M = m \mid C = c] = \Pr[M = m' \mid C = c]$$

False:

$$\begin{aligned}
&\Pr[M = m \mid C = c] \quad = \Pr[M = m' \mid C = c] \\
\rightarrow &\frac{\Pr[M = m \wedge C = c]}{\Pr[C = c]} = \frac{\Pr[M = m' \wedge C = c]}{\Pr[C = c]} \\
\rightarrow &\frac{\Pr[M = m \wedge C = c]}{\Pr[C = c \mid M = m]} = \frac{\Pr[M = m' \wedge C = c]}{\Pr[C = c \mid M = m]} \\
\rightarrow &\frac{\Pr[M = m \wedge C = c]}{\Pr[C = c \mid M = m]} = \frac{\Pr[M = m' \wedge C = c]}{\Pr[C = c \mid M = m']}
\end{aligned}$$

1.15. Shannon's Theorem

Let $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ be an encryption scheme with message space $\mathbb{M}$, for which $|M| = |K| = |C|$. This scheme is perfectly secret iff:

1. Every key $k \in \mathbb{K}$ is chosen with equal probability by algorithm Gen
2. For every $m \in \mathbb{M}$ and every $c \in \mathbb{C}$, there exists a unique key $k \in \mathbb{K}$ such that $\mathrm{Enc}_k(m)$ can output c.

2. Computational Approach

Two main relaxations:

Security is only guaranteed against efficient adversaries that run for some feasible amount of time.

Adversaries can potentially succeed with some small probability.

2.1. Efficient

"Efficient" adversaries are ones that run in polynomial time as a factor of the security parameter

2.2. Security parameter

Integer valued security parameter denoted by n that parametrizes the scheme and all involved parties.

When honest parties start a scheme, they choose some value of n for their security parameter.

The security parameter usually basically the length of the key, which is true for symmetric key cryptography.

The security parameter is known to the adversary.

We will view the run time and success probability as functions of the security parameter.

2.3. Negligible probability

A function f is negligible if for every polynomial p and all sufficiently large values of n it holds that $f(n) < \frac{1}{p(n)}$.

In other words, if $\forall c, \frac{1}{f(n)} \notin O(n^c)$.

2.4. Practical Implications

For key size n, any adversary running in time $2^{n/2}$ breaks the scheme with probability $1/2^{n/2}$.

Meanwhile, $\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec}$ each take time n^2.

If $n = 128$, then $\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec}$ takes time 16384, and the adversarial runtime is 2^{64}.

If $n = 256$, then $\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec}$ takes time 65536, and then adversary run time is multiplied by 2^{64}, and becomes $2^{128} \approx 10^{38}$

2.5. Defining Computationally Secure Encryption

A private-key encryption scheme is a tuple of probabilistic polynomial-time algorithms $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ such that:

1. The key generation algorithm Gen takes as an input the security parameter 1^n and outputs a key k denoted $k \leftarrow \mathrm{Gen}(1^n)$. We assume WLOG that $|k| \geq n$.
2. The encryption algorithm Enc takes as input a key k and message $m \in \{0,1\}^*$ and outputs a ciphertext c denoted $c \leftarrow \mathrm{Enc}_k(m)$.
3. The decryption algorithm Dec takes as input a key k and ciphertext c and outputs a message m denoted by $m := \mathrm{Dec}_k(c)$.

2.6. Correctness

For every n, every $k \leftarrow \mathrm{Gen}(1^n)$ and every $m \in \{0,1\}^*$, it holds that $\mathrm{Dec}_k(\mathrm{Enc}_k(m)) = m$.

2.7. Indistinguishability in the presence of an eavesdropper

Consider a private-key encryption scheme $\Pi = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$, any adversary A, and any value n for the security parameter.

The adversary sends two messages m_0 and m_1. The challenger forms $k \leftarrow \mathrm{Gen}(1^n)$, $b \leftarrow \{0,1\}$, $c \leftarrow \mathrm{Enc}_k(m_b)$, and sends c to the adversary.

Then, $\mathrm{PrivK}^{\mathrm{eav}}_{A,\Pi}(n) = 1$ if $b' = b$ and $\mathrm{PrivK}^{\mathrm{eav}}_{A,\Pi}(n) = 0$ if $b' \neq b$.[1].

[1] eav is for eavesdropper.

A private key encryption scheme $\Pi = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ has indistinguishable encryptions in the presence of an eavesdropper if for all probabilistic polynomial-time adversaries A, there exists a negligible function negl such that:

$$\Pr\left[\mathrm{PrivK}^{\mathrm{eav}}_{A,\Pi}(n) = 1\right] \leq \frac{1}{2} + \mathrm{negl}(n)$$

where the probability is taken over the random coins used by A, as well as the random coins used in the experiment.

2.8. Probabilistic polynomial time (PPT)

An algorithm is PPT if it takes polynomial time to run, it can depend on randomness.

2.9. Pseudorandom Generator

A pseudorandom generator is a polynomial-time deterministic algorithm G that takes in as input s, a random seed, and outputs a string $G(s)$. This is only interesting when the output is longer than the input.

No PPT algorithm can distinguish $G(s)$ from a truly random string r.

This can be used to build a computational secure encryption scheme with $|K| < |M|$.

2.10. Pseudorandom Generator Game

2.10.1. Ideal world

Uniformly choose from $\{0,1\}^{\ell(n)}$, where $\ell(n)$ tells you the length of the output string.

2.10.2. Real world

Uniformly choose from $\{0,1\}^n$, call this s. Then output $G(s)$.

Any PPT algorithm D cannot tell which world it is in:

$$|\Pr[D(r) = 1] - \Pr[D(G(s) = 1)]| \leq \mathrm{negl}(n)$$

where negl is some negligible function.

2.11. Pseudorandom generators cannot be non-computationally secure

Consider a candidate $G : \{0,1\}^n \to \{0,1\}^{\ell(n)}$, where $l : S \to \mathbb{N}$ and $\forall n, l(n) > n$, and $S \subset \mathbb{N}$.

Construct the set

$$S = G(\{0,1\}^n) = \{G(s) \mid s \in \{0,1\}^n\}$$

Then, we construct the distinguisher

$$D(w) = \begin{cases} 1 & w \in S \\ 0 & w \notin S \end{cases}$$

Then:

$$\Pr[D(G(s)) = 1] = 1$$

$$\Pr[D(r) = 1] = \frac{|S|}{2^{\ell(n)}} \leq \frac{2^n}{2^{\ell(n)}} = \frac{1}{2^{\ell(n)-n}}$$

Since $\ell(n), n \in \mathbb{N}$ and $\ell(n) > n$ then $\ell(n) - n > 0$, and therefore $\ell(n) - n \geq 1$.

Therefore,

$$\Pr[D(r) = 1] \leq \frac{1}{2^{\ell(n)-n}} \leq \frac{1}{2^1} = \frac{1}{2}$$

And therefore, since $|\Pr[D(r)=1] - \Pr[D(G(s))=1]| \geq \frac{1}{2}$, and $\frac{1}{2}$ is not negligible, this distinguisher could work. However, this distinguisher is not efficient, and therefore is not valid as a proof against something being computationally secure.

2.12. Class Exercise

Let G be a pseudorandom generator where $|G(s)| = |s| + 1$.

1. **Let $G'(s) = G(s \parallel \overline{s})$, where $\overline{s}$ is the bitwise negation of s. Is G' necessarily a pseudorandom generator?**

   No, since the input is *not* uniformly random.

   Let us construct G.

   $$G(s_1 \parallel s_2) = \begin{cases} 0 & s_1 = \overline{s}_2 \\ \tilde{G}(s_1 \parallel s_2) & \text{otherwise} \end{cases}$$

   Where $\tilde{G}$ is some other pseudorandom generator.

   Intuition: $s_1 = \overline{s}_2$ only occurs with negligible probability ($1/2^{|s|}$), so this is probably still secure.

   Let's construct D' for G'. Since $\forall s, G'(s) = 0$, then:

   $$D'(w) = \begin{cases} 1 & w = 0 \\ 0 & \text{otherwise} \end{cases}$$

   Then:

   $$\Pr[D'(G'(s)) = 1] = 1$$
   $$\Pr[D'(r) = 1] = \frac{1}{2^{\ell(n)}}$$

   Therefore, the difference is non-negligible.

2. **Let $G'(s) = G(s) \parallel G(\overline{s})$, where $\overline{s}$ is the bitwise negation of s. Is G' necessarily a pseudorandom generator?**

   b denotes a bit.

   Let us construct G.

   $$G(b \parallel \tilde{s}) = \begin{cases} \tilde{G}(\tilde{s}) & b = 0 \\ \tilde{G}(\overline{\tilde{s}}) & b = 1 \end{cases}$$

   Where $\tilde{G}$ is some other pseudorandom generator.

   Then

   $$\begin{aligned} & G'(b \parallel \tilde{s}) \\ &= G(b \parallel \tilde{s}) \parallel G\left(\overline{b} \parallel \overline{\tilde{s}}\right) \\ &= \begin{cases} \tilde{G}(\tilde{s}) \parallel \tilde{G}(\tilde{s}) & b = 0 \\ \tilde{G}(\overline{\tilde{s}}) \parallel \tilde{G}(\overline{\tilde{s}}) & b = 1 \end{cases} \end{aligned}$$

   And then the distinguisher sees if the first and second half are the same.

   $$D'(w_1 \parallel w_2) = \begin{cases} 1 & w_1 = w_2 \\ 0 & w_1 = w_2 \end{cases}$$

Therefore

$$\Pr[D'(G'(s)) = 1] = 1$$

$$\Pr[D'(r) = 1] = \frac{1}{2^{\ell(n)}}$$

3. **Define $G'(s) = G(s)_1 \parallel G\big(G(s)_2, ..., G(s)_{|s|+1}\big)$, where $G(s)_i$ denotes the ith output bit of $G(s)$. Is G' necessarily a pseudorandom generator?**

   The answer is yes.

   The output is then of the size:

   $$|s| + 2$$

   We got one whole extra bit!

2.13. A secure fixed-length encryption scheme

$\mathbb{M} = \{0,1\}^{\ell(n)}$ and $\mathbb{K} = \{0,1\}^n$. Therefore, $|\mathbb{K}| < |\mathbb{M}|$.

Let G be some efficient pseudorandom generator.

$\mathrm{Gen}(1^n)$ samples a key k uniformly at random from $\{0,1\}^n$.

$\mathrm{Enc} := k, m \mapsto G(k) \oplus m$

$\mathrm{Dec} := k, c \mapsto G(k) \oplus c$.

This is non-ideal since you can only use any particular key once. $c_1 \oplus c_2 = G(k) \oplus m_1 \oplus G(k) \oplus m_2 = m_1 \oplus m_2$, leaking information about the messages. This implies that the security definition we have is rather weak, and we need to strengthen it.

2.13.1. Proof

Assume that this encryption scheme is not indistinguishable in the presence of an adversary. We plan to break the PRG G with this.

Therefore we have a probabilistic polynomial-time adversary A and a non-negligible function ρ such that

$$\Pr\left[\mathrm{PrivK}^{\mathrm{eav}}_{A,\Pi}(n) = 1\right] \geq \frac{1}{2} + \rho(n)$$

Let us construct the adversary D for the PRG G. D receives w, where $w = G(k)$ or $w = r$.

A chooses two messages m_0 and m_1.

Choose $b \leftarrow \{0,1\}$. Send A, $c = w \oplus m_b$. If A outputs b, then D outputs 0, and if A outputs $\neg b$, then D outputs 1.

Intuitively, if A is able to break the encryption, then it is more likely that w is from the PRG, because if $w = r$ and A had broken the encryption, A would be a program that breaks OTP, which is impossible.

See that

$$\Pr[D(G(s)) = 1] = \Pr\left[\mathrm{PrivK}^{\mathrm{eav}}_{A,\Pi}(n) = 1\right]$$

Therefore:

$$\Pr[D(G(s)) = 1] \geq \frac{1}{2} + \rho(n)$$

And

$$\Pr[D(r) = 1] = \frac{1}{2}$$

because beating OTP is impossible.

Therefore

$$\Pr[D(G(s)) = 1] - \Pr[D(r) = 1] \geq \frac{1}{2} + \rho(n) - \frac{1}{2} = \rho(n)$$

Therefore using A we can build a distinguisher D for the PRG D, which is impossible, and therefore this encryption scheme is indistinguishable in the presence of an adversary.

2.14. Weak Example

Let $\mathrm{Enc}_k(m)$ be defined by $r \leftarrow \{0,1\}^n$ and then $c = (r, G(k \parallel r) \oplus m)$.

Further, let $\mathrm{Dec}_k(c = (r, c'))$ be defined by $m = G(k \parallel r) \oplus c'$

This is insecure: Assume $G(k \parallel r) = G^*(r)$.

Let the attacker send $m_0 = 0^\ell, m_1 = 1^\ell$.

Send this to the challenger.

Then, you will get $(r, G(k \parallel r) \oplus m_i) = (r, G^*(r) \oplus m_i)$.

Then, $G^*(r) \oplus (G^*(r) \oplus m_i) = 1^n \oplus m_i = m_i$, so we can recover the exact m_i. If $m_i = 0^\ell$ respond with 0 and if $m_i = 1^\ell$ respond with 1.

2.15. Stream Cipher

A stream cipher is stateful. Previously, the only state allowed was the secret key.

Let $\mathbb{K}$ be the key space, $\mathbb{S}$ be the state space, $\mathbb{M}$ be the message space, $\mathbb{C}$ be the ciphertext space, where $\mathbb{K} \subset \mathbb{S}$. And then you have s_{i+1} be computed from s_i ($\mathrm{Next} : \mathbb{S} \to \mathbb{S}$), and then also $\mathrm{Gen} : \mathbb{S} \times \mathbb{M} \to \mathbb{C}$, and $\mathrm{Dec} : \mathbb{S} \times \mathbb{C} \to \mathbb{M}$.

But this is very difficult to make work and is now basically unused.

2.16. Chosen Plaintext Attack (CPA) Security

Consider a private-key encryption scheme $\Pi = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$, any PPT adversary A, and any value n for the security parameter.

The adversary A has oracle access to $\mathrm{Enc}_k(\cdot)$.[2]. Then, A will choose $m_0, m_1 \in \mathbb{M}$ and the challenger will then choose a random $b \in \{0,1\}$ and respond with m_b. Afterwards, the challenger can continue to make as many queries as it wants. Finally, the attacker will respond with b', the guess to if it was $b = 0$ or $b = 1$.

A private-key encryption scheme $\Pi = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ has indistinguishable encryptions under a chosen-plaintext attack if for all ppt adversaries A there exists a negligible function negl such that

$$\Pr\left[\mathrm{PrivK}^{\mathrm{cpa}}_{A,\Pi}(n) = 1\right] \leq \frac{1}{2} + \mathrm{negl}(n)$$

where the probability is taken over the random coins used by A, used in the experiment, and used for encryption.

2.17. CPA-security under multiple encryptions

Any private-key encryption scheme that has indistinguishable encryptions under a chosen plaintext attack also has indistinguishable multiple encryptions under a chosen-plaintext attack.

[2] IOW, the adversary can ask what $\mathrm{Enc}_k(m)$ is for any $m \in \mathbb{M}$ it wants as many times it wants, but A must still be polynomial-time, so it can't just keep on asking. If A was unbounded, this would always be broken, so we must constrain A to polynomial-time

2.18. CPA-security is non-deterministic

If $\Pi = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ is an encryption scheme in which Enc is a deterministic function of the key and the message, then Π is not CPA-secure.

2.18.1. Proof

Construct an adversary A.

Let $m_0 = 0^\ell$, $m_1 = 1^\ell$. Have A ask $c_0 = O(m_0), c_1 = O(m_1)$ before submitting m_0, m_1 as the messages.

The challenger responds with c. If $c = c_0$ output 0, and if $c = c_1$ output 1. If it's something else, something went wrong (it's non-deterministic).

Therefore:

$$\Pr\left[\mathrm{PrivK}^{\mathrm{cpa}}_{A,\Pi}(n) = 1\right] = 1$$

And then, since:

$$1 \geq \frac{1}{2} + \mathrm{negl}(n)$$

We have shown that any CPA-secure scheme is non-deterministic.

2.19. Keyed Function

$$F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$$

is a two-input function, where the first input is called the key and denoted k.

In practice, a key $k \leftarrow \{0,1\}^n$ once and then we view $F(k, \cdot) : \{0,1\}^n \to \{0,1\}^n$. We denote this as $F_k(\cdot)$.

2.20. Pseudorandom Function

f is a function chosen at random from all the functions $f : \{0,1\}^n \to \{0,1\}^n$. k is chosen at random from $\{0,1\}^n$. F_k is the pseudorandom function indexed by k.

An attacker A queries f or F_k as many times as it wants and then guesses what world it's in.

A PRF is a function such that for all PPT A:

$$|\Pr[A^f() = 1] - \Pr[A^{F_k}() = 1]| \leq \mathrm{negl}(n)$$

You can alternatively think of f as a cached true random number generator.

The set of f's is of size $(2^n)^{(2^n)}$.

But, using a construction by Goldreich, Goldwasser, and Micali, you can construct a PRF from any PRG.

2.21. Exercise

Let F be a length-preserving pseudorandom function. For the following constructions of a keyed function $F' : \{0,1\}^n \times \{0,1\}^{n-1} \to \{0,1\}^{2n}$, state whether F' is a pseudorandom function. If yes, prove it; if not, show an attack.

1. $F'_k(x) := F_k(0 \parallel x) \parallel F_k(x \parallel 1)$.

   Consider inputting $x_1 = 0^{n-1}$ and then $x_2 = 0^{n-2}1$

   Then,

   $$F'_k(x_1) = F_k(0^n) \parallel F_k(0^{n-1} \parallel 1)$$
   $$F'_k(x_2) = F_{k(0^{n-1} \parallel 1)} \parallel F_k(0^{n-2} \parallel 1^2)$$

Then, you compare the second half of the first query and the first half of the second query, and if they are equal you return 1 otherwise return 0.

If the attacker D is in the F'_k world, these will always be equal, so this happens with probability 1.

If the attacker D is in the f world, these are equal with probability $1/2^n$, since the input's aren't equal.

$$\Pr\left[D^{F'_k(\cdot)}(1^n) = 1\right] = 1$$

$$\Pr\left[D^{f(\cdot)}(1^n) = 1\right] = \frac{1}{2^n}$$

Then, for any $n > 0$:

$$\left|1 - \frac{1}{2^n}\right| \geq \frac{1}{2}$$

As $\frac{1}{2}$ is non-negligible, this attack works.

2. $F'_k(x) := F_k(0 \parallel x) \parallel F_k(1 \parallel x)$

   Intuitively, this works because F_k simulates a cached truly random function, and so, changing the first bit of F_k completely changes the output.

   To prove this for real, you would have to use an attack on F'_k to attack F_k.

   Basically, in A^{F_k}, for each query that $A^{F'_k}$ wants to do, you can just do the two queries and return them concatenated. If it is in the f world, adding the 1 or 0 completely partitions the space, so it is completely random and $A^{F'_k}$ has no chance.

2.22. Construction of CPA-secure encryption from a PRF

In key generation, we choose a uniformly random $k \leftarrow \{0,1\}^m$. Then, we just call a pseudorandom function $F_k : \{0,1\}^n \to \{0,1\}^n$ to create a computationally random string, using $r \leftarrow \{0,1\}^n$ uniformly, and then $c = (r, F_k(r) \oplus m)$. Everything becomes OTP![3] The receiver can just receive and then run the pseudorandom function themselves (as r is public).

2.22.1. Proof

Assume that there is an attacker A that can successfully break this encryption. Using A we will construct D to distinguish F_k from a truly random function.

Basically, to construct D we just emulate the encryption process but use the oracle instead of F_k. If the oracle is f, this is impossible to break as as long as r is unique every time, this is equivalent to OTP, and they are equal only a negligible number of times.

If A successfully breaks encryption we return 1 and if it does not respond 0.

More formally, the distinguisher D^O generates $r \leftarrow \{0,1\}^n$ for any m' that the attacker A responds with, and gives the attacker $c' = (r, O(r) \oplus m')$.

Then the distinguisher upon receiving two messages m_0, m_1, generates $b \leftarrow \{0,1\}$, $r^* \leftarrow \{0,1\}^n$, and responds with $c = (r^*, O(r^*) \oplus m_b)$. If $b' = b$, output 1, and otherwise output 0.

$$\Pr\left[D^{F_k(\cdot)}(1^n) = 1\right] = \Pr\left[\mathrm{PrivK}^{\mathrm{cpa}}_{A,\Pi}(n) = 1\right] \geq \frac{1}{2} + \rho(n)$$

Then:

[3] The cryptography version of carcinization.

$$\Pr\left[D^{f(\cdot)}(1^n) = 1\right] = \Pr[r^* \text{ exists}] + (1 - \Pr[r^* \text{ exists}]) \cdot \frac{1}{2}$$

But:

$$\Pr[r^* \text{ exists}] = 1 - \left(1 - \frac{1}{2^n}\right)^{q(n)}$$

where $q(n)$ is the number of queries that the attacker A does. But $\Pr[r^* \text{ exists}] \leq \frac{q(n)}{2^n}$, and this is negligible.

Therefore:

$$\Pr\left[D^{f(\cdot)}(1^n) = 1\right] \leq \frac{1}{2} + \text{negl}(n)$$

And in sum (we can assume $\rho(n) > \text{negl}(n)$):

$$\begin{aligned}
&\left|\Pr\left[D^{F_k(\cdot)}(1^n) = 1\right] - \Pr\left[D^{f(\cdot)}(1^n) = 1\right]\right| \\
&\geq \left|\frac{1}{2} + \rho(n) - \left(\frac{1}{2} + \text{negl}(n)\right)\right| \\
&\geq \rho'(n)
\end{aligned}$$

So it is non-negligible.

2.23. Pseudorandom Permutations

A pseudorandom permutation is a random function except for every key k, F_k must be invertible (and therefore a permutation, since S^n is the group of permutations) and it must be indistinguishable from a random permutation.

Both F_k and F_k^{-1} must be able to be efficiently computed. The game is that the distinguisher is given f, f^{-1} or F_k, F_k^{-1} and must distinguish which world it's in.

$$\left|\Pr\left[A^f() = 1\right] - \Pr\left[A^{F_k}() = 1\right]\right| \leq \frac{1}{2^n}$$

2.24. Cipher Block Chaining

$$c_0 = \text{IV}$$

where IV is the initialization vector, randomly selected from $\{0,1\}^n$.

Then, for encryption:

$$c_{i+1} = F_k(c_i \oplus m_{i+1})$$

To decrypt:

$$m_{i+1} = F_k^{-1}(c_{i+1}) \oplus c_i$$

2.25. Output Feedback (OFB) Mode

$$r_1 = \text{IV}$$

where IV is the initialization vector, randomly selected from $\{0,1\}^n$.

Let

$$r_{i+1} = F_k(r_i)$$

Then, $c_i = m_i \oplus F_k(r_i)$

2.26. Counter (CTR) Mode

Let ctr be randomly selected from $\{0,1\}^n$.

Then:

$$c_i = m_i \oplus F_k(\mathrm{mod}(\mathrm{ctr} + i, 2^n))$$

3. Message Integrity/Authenticity

A receiver wants to make sure that the message they got came from a particular sender.

3.1. Message Authentication Code (MAC)

A sender and a receiver meet up and exchange a key k.

Then, Eve sits between the sender and receiver, and the goal is that if Eve modifies the message, the receiver will be able to tell.

When sending, the sender generates $\text{tag} \leftarrow \text{Mac}_k(m)$, and sends (m, tag). The receiver then checks via $\text{Vrfy}_k(m, \text{tag})$. If 0, the receiver should reject, and if 1, if it should accept. Formally:

A Message Authentication Code (MAC) consists of three probabilistic polynomial-time algorithms $(\text{Gen}, \text{Mac}, \text{Vrfy})$ such that:

1. The key generation algorithm Gen takes as input the security parameter 1^n amd outputs a key k with $|k| \geq n$.
2. The tag-generation algorithm Mac takes as input a key k and a message $m \in \{0,1\}^*$, and outputs a tag t, $t \leftarrow \text{Mac}_k(m)$
3. The deterministic verification algorithm Vrfy takes as input a key k, a message m, and a tag t. It outputs a bit b with $b = 1$ meaning valid and $b = 0$ meaning invalid. $b := \text{Vrfy}_k(m, t)$.

It is required that for every n, every key k output by $\text{Gen}(1^n)$ and every $m \in \{0,1\}^*$, it holds that $\text{Vrfy}_k(m, \text{Mac}_k(m))) = 1$.

3.2. Unforgeability

A message encryption code $\Pi = (\text{Gen}, \text{Mac}, \text{Vrfy})$ is existentially unforgeable under an adaptive chosen message attack if for all probabilistic polynomial-time adversaries A, there is a negligible function negl such that:

$$\Pr\left[\text{MACsForge}_{A,\Pi}(n) = 1\right] \leq \text{negl}(n)$$

Let the adversary query messages m', and receive tags t' (has access to an oracle of $\text{Mac}_k(\cdot)$). Then, the adversary should send (m, t). Call the set of queried messages Q, and then $\text{MacForge}_{A,\Pi}(n) = 1$ iff $m \notin Q$ and $\text{Vrfy}_k(m, t) = 1$.

3.3. Strong Unforgeability for MACs

A message encryption code $\Pi = (\text{Gen}, \text{Mac}, \text{Vrfy})$ is a strong MAC if for all probabilistic polynomial-time adversaries A, there is a negligible function negl such that:

$$\Pr\left[\text{MACsForge}_{A,\Pi}(n) = 1\right] \leq \text{negl}(n)$$

Let the adversary query messages m', and receive tags t' (has access to an oracle of $\text{Mac}_k(\cdot)$). Then, the adversary should send (m, t). Call the set of queries and received tags $Q = \{(m', t'), ...\}$, and then $\text{MacForge}_{A,\Pi}(n) = 1$ iff $(m, t) \notin Q$ and $\text{Vrfy}_k(m, t) = 1$.

3.4. Secure MAC

Let $k \in \{0,1\}^n$. Then, $\text{Mac}_k := m \mapsto F_k(m)$. To verify (m, t), check $F_k(m) = t$. If yes, output, 1, otherwise, output 0.

3.4.1. Proof

On the contrary, assume the scheme is not secure. Then, there exists an adversary A such that there is some non-negligible ρ such that $\Pr\left[\text{MACsForge}_{A,\Pi}(n) = 1\right] \geq \rho(n)$. We seek to construct a distinguisher D that breaks the PRF using A.

A will make queries to an oracle. Given a query m' by A, D will return $t' = O(m')$. Given the forgery (m, t), if $m \in Q$, output 0. Otherwise output 1 if $O(m) = t$ else output 0.

If $O = F_k$:

$$\Pr\left[D^{F_k(\cdot)}(1^n)=1\right]=\Pr\left[\text{MACsForge}_{A,\Pi}(n)=1\right]\geq\rho(n)$$

$$\Pr\left[D^{f(\cdot)}(1^n)=1\right]=\Pr[A \text{ outputs } m\in Q]\cdot 0+\Pr[A \text{ outputs } m\notin Q](\Pr[f(m)=t])\leq\frac{1}{2^n}$$

Notice that $\Pr[f(m)=t]$ is $\frac{1}{2^n}$ since f has never been queried at m and f can be thought of as a cached random source. Then:

$$\left|\rho(n)-\frac{1}{2^n}\right| \rightsquigarrow \rho(n)$$

and therefore it is non-negligible.

3.5. Bad Domain Extension for MACs

Let $t=\text{Mac}_k(m_1)\parallel\text{Mac}_k(m_2)\parallel\cdots\parallel\text{Mac}_k(m_\ell)$.

This is does not work.

As an attacker query, $m=m_1\parallel m_2=0^n\parallel 1^n$, get back $t=t_1\parallel t_2$. Forge $m'=m_2\parallel m_1$, $t'=t_2\parallel t_1$.

3.6. Class Exercise

1. To authenticate a message $m=m_1\parallel m_2$ where $m_1,m_2\in\{0,1\}^n$, compute $t:=F_k(m_1)\parallel F_k(m_2\oplus F_k(m_1))$.

   Assume we want to forge some message m_1,m_2.

   First query m_1, ..., receive $t_1\parallel t_2$.

   Then, query $\tilde{m}_1=m_2\oplus t_1$, ..., receive $\tilde{t}_1\parallel\tilde{t}_2$.

   Now, we know that $t_1=F_k(m_1)$, and therefore $\tilde{t}_1=F_k(m_2\oplus F_k(m_1))$. Therefore we can forge message m_1,m_2 with tag $t_1\parallel\tilde{t}_1$

   Note that you *do not have to* be able to forge any message. The original version of this forged $1^n,0^n$ (which is much easier to see)
2. To authenticate a message $m=m_1\parallel\ldots\parallel m_\ell$ where $m_i\in\{0,1\}^n$, choose $r\in\{0,1\}^n$ at random and compute $t:=r\parallel F_k(m_1\oplus r)\parallel\cdots\parallel F_k(m_\ell\oplus r)$.

   This does not work. Query $m=m_1\parallel m_2\parallel\cdots\parallel m_\ell$. Get back:

   $$t=r\parallel F_k(m_1\oplus r)\parallel F_k(m_2\oplus r)\parallel\cdots\parallel F_k(m_\ell\oplus r)$$

   Forge:

   $$m'=m_1\oplus r\parallel m_2\oplus r\parallel\cdots\parallel m_\ell\oplus r$$
   $$t'=0\parallel t_1\parallel t_2\parallel\cdots\parallel t_\ell$$

   (this works for $r\neq 0$. Though it can generalize past that, it does not need to since $r=0$ a negligible amount of the time)

3.7. CBC-MAC

This only works on a fixed number of blocks.

Let

$$t_1=F_k(m_1)$$

$$t_i=F_k(m_i\oplus t_{i-1})$$

Output $t=t_m$, where m is the number of blocks.

To allow this to work for any number of blocks, prepend the length ℓ as the content of the first block (prefix free encoding).

Padding Attack: The first version doesn't work because if you had requested $m_1 \parallel m_2 \parallel m_3$, where $|m_1| = |m_2| = n$, and $|m_3| = m < n$ first, and then zero-extended to the proper length, then you could forge $m_1 \parallel m_2 \parallel m_3 \parallel 0^{n-m}$ (these messages are indistinguishable).

Length Extension Attack: Assume all queried messages and forgeries will be multiplies of n (but not necessarily all the same length).

1. Query m_1, receive $t_1 = F_k(m_1)$.
2. Query $m_2 \oplus t_1$, receive $t_2 = F_k(m_2 \oplus t_1)$.
3. Forge $m_1 \parallel m_2$ with $t = t_2$.

This works because $t_2 = F_k(m_2 \oplus F_k(m_1))$, which is exactly the required tag.

3.8. CCA Security

Consider a private-key encryption scheme $\Pi = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$, any adversary A, and any value n for the security parameter.

First, run $k \leftarrow \mathrm{Gen}(1^n)$.

Let the adversary given 1^n use $\mathrm{Enc}_k(\cdot)$, and $\mathrm{Dec}_k(\cdot)$ as oracles. Then, it must choose m_0, m_1, and then after c is returned (after being chosen randomly), it must decide which one of m_0, m_1 was encrypted. It can continue to use the oracles but it cannot query c into $\mathrm{Dec}_k(\cdot)$ (otherwise it fails the experiment and $\mathrm{PrivK}^{\mathrm{cca}}_{A,\Pi}(n) = 0$).

Note that sometimes $\perp$ is returned from $\mathrm{Dec}_k(\cdot)$, when there is nothing returned.

$$\mathrm{PrivK}^{\mathrm{cca}}_{A,\Pi}(n) = 1 \text{ if } b' = b \text{ and } \mathrm{PrivK}^{\mathrm{cca}}_{A,\Pi}(n) = 0 \text{ otherwise, if } b' \neq b$$

Then, a private-key encryption scheme has indistinguishable encryptions under a chosen-ciphertext attack if for all PPT adversaries A there exists a negligible function negl such that

$$\Pr\left[\mathrm{PrivK}^{\mathrm{cca}}_{A,\Pi} = 1\right] \leq \frac{1}{2} + \mathrm{negl}(n)$$

where the probability is taken over the random coins used by A and the random coins used in the experiment.

3.9. Unforgeable Encryption

The experiment $\mathrm{EncForge}_{A,\Pi}(n)$ is defined by:

1. Run $k \leftarrow \mathrm{Gen}(1^n)$.
2. The adversaries is given input 1^n and access to an encryption oracle $\mathrm{Enc}_k(\cdot)$. The adversary outputs a ciphertext c.
3. Let $m := \mathrm{Dec}_k(c)$, and let Q denote the set of all queries that A asked the encryption oracle. The output of the experiment is 1 iff $m \neq \perp$ and $m \notin Q$.

A private-key encryption scheme Π is unforgeable iff for all PPT adversaries A there is a negligible function negl such that:

$$\Pr\left[\mathrm{EncForge}_{A,\Pi}(n) = 1\right] \leq \mathrm{negl}(n)$$

3.10. Authenticated Encryption

A private-key encryption scheme is an authenticated encryption scheme if it is CCA-secure and unforgeable.

3.11. Encrypt-and-authenticate

Define $c \leftarrow \mathrm{Enc}_{k_E}(m)$ and $t \leftarrow \mathrm{Mac}_{k_M}(m)$, and output $\langle c, t \rangle$. This is not secure, because Mac could choose to also just output m.

Also, if you query m_0, m_1 to the encryption oracle, then you can compare the output tag to the tag computed via oracle and see which message was selected.

3.12. Authenticate-then-encrypt

Define $t \leftarrow \mathrm{Mac}_{k_M}(m)$ and $c \leftarrow \mathrm{Enc}_{k_E}(m \parallel t)$. This does not achieve CCA-security if the encryption scheme is not already CCA-secure.

3.13. Encrypt-then-authenticate

Define $c \leftarrow \mathrm{Enc}_{k_E}(m)$, $t \leftarrow \mathrm{Mac}_{k_E}(c)$, output $\langle c, t \rangle$. This is CCA-secure, assuming Enc is CPA-secure and Mac is strongly secure.

3.14. Merkle-Damgard

Consider $z_0 = \mathrm{IV}$. Then, $z_1 = h^s(z_0 \parallel k)$, and for $i > 1$, $z_i = h^s(z_{i-1} \parallel m_{i-1})$. When you finish the message, the last block, $t = h^s(z_n \parallel \langle n \rangle)$.

This builds a new collision-resistant hash function.

3.14.1. Attack

Query m_1, receive t_1.

Construct the forgery $m = m_1 \parallel \langle 2n \rangle$, with tag $h^s(t_1 \parallel \langle 3n \rangle)$.

3.15. Sponge

Consider a message padded by some padding function $p = P(m)$ so that $r \mid |p|$. Let $q = |p|/r$, $q \in \mathbb{Z}$.

Therefore, we have $p_1 \parallel p_2 \parallel \cdots \parallel p_q = p$.

Let $v_0 = (0)^{r+n}$. Then,

$$v_n = f(v_{n-1}[:r] \oplus p_n \parallel v_{n-1}[r:])$$

This is the “absorption step”.

Then,

$u_0 = v_q$, and $u_n = f(u_{n-1})$, and you output $z_n = u_n$.

3.16. Birthday bound

With m objects into n bins you have the probability

$$p = 1 - \frac{n!}{n^m (n-m)!}$$

and $p \in \Omega\left(\frac{m^2}{n}\right)$.

3.17.

??

4. Modular Arithmetic

$a \equiv b \bmod p$ is defined by $p \mid (a - b)$.

4.1. Examples

$$2 \equiv 13 \bmod 11$$
$$2 \equiv -9 \bmod 11$$

4.2. Unique Representative

By division, for any a, there is always a unique number k such that $0 \leq k < p$, and $a \equiv k \bmod p$

4.3. Addition

Take two numbers a, b, add them, and take them mod p.

$$8 + 10 \bmod 13 \equiv 18 \bmod 13 \equiv 5 \bmod 13$$

It may be convenient to apply modulus first:

$$8 + 10 \bmod 13 \equiv 8 - 3 \bmod 13 \equiv 5 \bmod 13$$

4.4. Properties of Addition

Define $\mathbb{Z}_p = \{0, 1, ..., p - 1\}$.

1. There is the additive identity 0, so $a + 0 \equiv a \bmod p$ for any a, p.
2. There is the additive inverse $-a$, so $a - a \equiv 0 \bmod p$ for any a, p.
3. Closure: For any $a, b \in \mathbb{Z}_p$, there is an element $k \in \mathbb{Z}_p$ such that $a + b \equiv k \bmod p$.
4. Associative: for all $a, b, c \in \mathbb{Z}_p$, $(a + b) + c \equiv a + (b + c) \bmod p$.

4.5. Group

A group is a set G along with a binary operation $\circ$ for which:

1. Closure: $\forall g, h \in G, g \circ h \in G$.
2. Identity: $\exists e \in G, \forall g \in G, e \circ g = g = g \circ e$
3. Inverse: $\forall a \in G, \exists a^{-1}, a^{-1} \circ a = e$ (where e is constrained as above).
4. Associative: $a, b, c \in G$, $(a \circ b) \circ c = a \circ (b \circ c)$.

4.6. Abelian

For an Abelian group G, commutativity holds: $\forall g, h \in G, g \circ h = h \circ g$.

4.7. Multiplication over the modular integers

Is a $\mathbb{Z}_p$, where $p \in \mathbb{P}$ (the set of primes), a group?

1. Closure: still applies by division.
2. Identity: 1 is the identity.
3. Associative: Yes (nontrivial?)
4. Inverse: No! There is no inverse for 0, as anything times 0 is just zero.

Is a $\mathbb{Z}_p^* = \{1, ..., p - 1\}$ a group?

1. Closure: non-trivial
2. Identity: 1 is the identity
3. Associative: Yes

4.8. Brute-force find multiplicative inverses $\mod p$

Just try multiplying a by every value $\{1, ..., p-1\}$, and see which one results in 1! But this is exponential, as it is $O(p)$ additions (multiplication is repeated addition), but to describe the value of p, you only need $\log p$ bits, and therefore it is $O\left(2^{\log p}\right)$ additions. As your input size is $\ell = 2 \times \log(p)$ for the number of bits in p and in a, not the value of the input, this is clearly exponential, $O\left(2^{\ell/2}\right)$.

4.9. Bezout's Identity (Euclidean Algorithm)

Let a, b be positive integers. Then, there exist $X, Y \in \mathbb{Z}$, such that $Xa + Yb = \gcd(a, b)$.

4.10. Why inverses exist

Let $a \in \mathbb{Z}_p$, $p \in P$.

See that $\gcd(a, p) = 1$ as p is prime, and $0 < a \leq p - 1$.

Therefore by Bezout, there exists $X, Y \in \mathbb{Z}$ such that

$$\begin{aligned} aX + pY &= 1 \\ pY &= 1 - aX \end{aligned}$$

Therefore, $p \mid (1 - aX)$, and therefore $aX \equiv 1 \mod p$. Since there exists a value X, there is a multiplicative inverse.

4.11. Extended Euclidean Algorithm

Consider the fact that $\gcd(a, b) = \gcd(a, b \bmod a)$ where $a \leq b$.

If you recurse, until the smaller number is zero, $\gcd(a, 0) = a$.

The size of the problem reduces in half every 2 iterations.

4.11.1. Example

Consider $1 = aX + pY$, where $a = 9$ and $p = 23$.

Value	X	Y
9	1	0
23	0	1
$5 = 23 - 2 \cdot 9$	-2	1
$4 = 9 - 1 \cdot 5$	3	-1
$1 = 5 - 1 \cdot 4$	-5	2

Therefore, $1 = 2 \cdot 23 - 5 \cdot 9$, and further -5 is the multiplicative inverse (otherwise known as 18)

4.11.2. Time complexity

Consider the worst case at every step.

Consider you have two numbers a, b at some step i.

WLOG, assume $a \leq b$. Then, $b' = a - kb$ for some integer $k \geq 1$, and further $b' \leq a$.

Run the algorithm again: $a' = b' - ja$, for some integer $k \geq 1$, and further, $a' \leq b'$.

Therefore,

$$a' \leq b' \leq a \leq b$$

So at the very least the problem is shrinking. But if you consider the worst case, $k = 1, j = 1$, $b' = a - b$, $a' = b' - a = a - b - a = -b$, we have confusion (see slides ig)

4.12. Fermat's Little Theorem

For prime p, integer a:

$$a^p \equiv a \bmod p$$

Furthermore, as a corollary, for p, a such that $\gcd(p, a) = 1$:

$$a^{p-1} \equiv 1 \bmod p$$

The corollary is easy to prove by multiplying by the inverse.

The generalization to all groups is, for a group G with order $m = |G|$, for any element $g \in G$:

$$g^m = 1$$

4.13. Composite Multiplicative Groups

Which numbers $\{1, ..., N-1\}$ have inverses mod N?

Exactly the numbers a such that $\gcd(a, N) = 1$.

Note that $\mathbb{Z}_N^*$ is closed since if you have two elements that have inverses, $x, y \in \mathbb{Z}_N^*$, xy has an inverse: $(xy)y^{-1}x^{-1} = 1$.

What is the order of such a group? $\varphi(N)$!

This can be computed via Euler's product formula:

$$\varphi(N) = N \prod_{p|n} \left(1 - \frac{1}{p}\right)$$

Note that this required the prime factorization of N. No algorithm does not require the factorization of N.[4]

Take the special case of $N = pq$, where p, q are distinct prime numbers. Then, consider the set $\{0, 1, ..., N-1\}$. The numbers that cannot be inverted are the numbers a such that $\gcd(a, N) \neq 1$, so they must have a factor of p or q. Clearly, 0 is not invertible, and then $1 \cdot p, 2 \cdot p, ..., (q-1) \cdot p$ are non-invertible, and $1 \cdot q, 2 \cdot q, ..., (p-1) \cdot q$ are also not invertible. This is the complete set of non-invertible elements, and they do not overlap. So, to find the order of the subset of invertible elements, it is $(N-1) - (p-1) - (q-1) = N - p - q + 1 = pq - p - q + 1 = (p-1)(q-1)$.

4.14. Power reduction for any group

Let G be a finite group with $m = |G| > 1$, then for any $g \in G$ and any integer x we have $g^x = g^{x \bmod m}$. This is easy to prove. Write $x = am + b$, where $a \in \mathbb{Z}$ and $b = x \bmod m$ by division.

Then, $g^x = g^{am+b} = g^{am} g^b = (g^m)^a g^b = 1^a g^b = g^b = g^{x \bmod m}$.

This can be used to compute $3^{25} \bmod 35$, because $\varphi(35) = (5-1)(7-1) = 24$, and therefore $3^{25} \equiv 3^1 \bmod 35$.

4.15. Cyclic Groups

For a finite group G of order m and $g \in G$ consider:

$$\langle g \rangle = \{g^0, g^1, ..., g^{m-1}\}$$

$\langle g \rangle$ always forms a cyclic subgroup of G, but it is possible that there are repeats in the above list, so perhaps it is a strictly smaller subgroup. If $\langle g \rangle = G$, then we say that G is a cyclic group and that g is a generator of G.

[4] You can φ as an oracle to compute the prime factorization, so if φ was polynomial time, prime factorization is polynomial time (certainly is for the two prime case, since $p + q = N - \varphi(N) + 1$, and then you can just solve $x^2 - Sx + n = 0$ via the quadratic formula).

4.16. Finite group element order divides

Note that, for any finite group G and any $g \in G$, $|\langle g \rangle|$ divides $|G|$.

4.17. Order of an element

Let G be a finite group, and let $g \in G$. The order of g, denoted $|g|$, is defined as the smallest positive integer such that $g^i = 1$.

This implies $|\langle g \rangle| = |g|$.

4.18. Power reduction for elements of a group

Let G be a finite group, and let $g \in G$. Then, $g^a \equiv g^{a \bmod |g|}$.

Furthermore, it is clear that

$$g^i = g^j \text{ iff } i \equiv j \bmod |g|$$

4.19.

If G is a group of prime order p, then G is cyclic, and all elements of G except the identity are generators of G. This follows from 4.16. Finite group element order divides, since the only numbers that divide p are 1 and p, and the only number that has order 1 is the identity (since $a^1 = a$ by definition, and if $a = 1$, a is the identity), and therefore all others have order p.

If p is prime, Z_p^* is a cyclic group of order $p - 1$.

4.20. Strong Primes

A strong prime is a prime p such that $p = 2q + 1$, where q is prime.

4.21. Strong Prime group

Consider z_p^* where p is a strong prime. Recall that Z_p^* has order $p - 1$, and therefore the subgroup of quadratic residues in Z_p^* is a subgroup of prime order q, since $\langle g^2 \rangle = \{g^0, g^2, ..., g^{2(q-1)}\}$, as $g^{2q} = g^0$ since it has order $2q$.

4.21.1. Example

Consider Z_{11}^*. 11 is a strong prime.

Then, $g = 2$ is a generator of Z_{11}^*, $\langle 2 \rangle = \{1, 2, 4, 8, 5, 10, 9, 7, 3, 6\}$, and then the quadratic residues are $\langle 2^2 \rangle = \{1, 4, 5, 9, 3\}$.

This subgroup does satisfy closure, and cyclic (all even powers of g can be generated by g^2).

4.22. Legendre Symbol

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{is a quadratic residue} \bmod p \\ -1 & \text{is not a quadratic residue} \bmod p \\ 0 & a \equiv 0 \bmod p \end{cases}$$

This can be directly computed via $a^{(p-1)/2} \bmod p$.

4.23. Discrete Log

The discrete-log experiment $\mathrm{DLog}_{A,G}(n)$ is:

1. Run $\boldsymbol{G}(1^n)$ to obtain (G, q, g) where G is a cyclic group of order q with $||q|| = n$ and g as a generator of G.
2. Choose a uniform $h \in G$
3. A is given G, q, g, h and outputs $x \in \mathbb{Z}_q$.
4. The output if the experiment is 1 if $g^x = h$ and 0 otherwise.

We say discrete log is hard relative to G if for all ppt algorithms A there exists a negligible function negl such that:

$$\Pr\left[\mathrm{DLog}_{A,G}(n) = 1\right] \leq \mathrm{negl}(n)$$

This is considered a logarithm since:

$$\mathrm{Dlog}_g(h) = x$$

Also:

$$\mathrm{Dlog}_g(ab) \equiv \mathrm{DLog}_g(a) + \mathrm{DLog}_g(b) \bmod g$$

4.24. Computational Diffie-Hellman

Given (G, q, g) and uniform $h_1 = g^{x_1}$ and $h_2 = g^{x_2}$, compute $g^{x_1 x_2}$.

It is clear that if you break discrete log, you can break CDH. It is an open question if CDH implies you break Discrete Log.

4.25. Distinguishing Diffie-Hellman

We say the DDH problem is hard relative to G if for all ppt algorithms A, there exists a negligible function negl such that:

$$|\Pr[A(G, q, g, g^x, g^y, g^z) = 1] - \Pr[A(G, q, g, g^x, g^y, g^{xy}) = 1]| \leq \mathrm{negl}(n)$$

It is believed that in $\mathbb{Z}_p^*$, CDH is hard. But DDH is easy in $\mathbb{Z}_p^*$.

Also, breaking CDH implies you can break DDH (just compute and compare).

4.25.1. Attack in $\mathbb{Z}_p^*$

Using the Legendre symbol, you can compute whether or not something is a quadratic residue: just compute $h^{(p-1)/2} \bmod p$, which can be done quickly by repeated squaring.

Applying this, we can see if either g^x or g^y are a quadratic residue, g^{xy} is.

WLOG, assume g^x is a quadratic residue, and therefore $2 \mid x$. Let $x = 2k$. Then, $g^{xy} = g^{2ky}$, and therefore, as $2 \mid (2ky)$, g^{xy} is not a quadratic residue.[5]

If neither are quadratic residues, then g^{xy} should also not be quadratic residue.

Then, in terms of the Legendre symbols of g^x, g^y and g^z, if we have $\left(\left(\frac{g^x}{p}\right), \left(\frac{g^y}{p}\right), \left(\frac{g^z}{p}\right)\right) \in \{(1,1,1), (1,-1,1), (-1,1,1), (-1,-1,-1)\}$, output 1, as this is consistent with $g^z = g^{xy}$, otherwise output 0.

Therefore:

$$\Pr[A^{\text{ideal}}(G, q, g, g^x, g^y, g^{xy}) = 1] = 1$$
$$\Pr[A^{\text{real}}(G, q, g, g^x, g^y, g^z) = 1] = \frac{1}{2}$$

And therefore the difference is $\frac{1}{2}$, and $\frac{1}{2}$ is non-negligible, and therefore DDH is easy in $\mathbb{Z}_p^*$.

[5] The reverse implication works because if you consider some $(g^x)^2 \equiv g^{2x}$, it is equivalent to something $\bmod p - 1$, which is even, so taking it $\bmod p - 1$ preserves the evenness.

5. Elliptic Curves

5.1. Finite Fields

A finite field is a set of elements that can be viewed as a group with respect to two operations, denoted by addition and multiplication where the identity element for addition does not require a multiplicative inverse.[6]

An example of this is $\mathbb{Z}_p = \{0, 1, ..., p-1\}$.

We can consider polynomials over the field $\mathbb{Z}_p$.

5.2. Elliptic Curve

Let $p \geq 5$ be prime. Consider the polynomial:

$$y^2 \equiv x^3 + Ax + B \bmod p$$

Where $4A^3 + 27B^3 \not\equiv 0$ to ensure non-repeated roots.

Let $E(\mathbb{Z}_p)$ denote the set of pairs $(x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p$ satisfying the above equation as well as a special value of O representing the point at infinity.

$$E(\mathbb{Z}_p) := \left\{(x, y) \;\middle|\; x, y \in \mathbb{Z}_p \wedge y^2 \equiv x^3 + Ax + B \bmod p\right\} \cup \{O\}$$

5.2.1. Example

Consider the quadratic residues over $\mathbb{Z}_7$, so $0^2 = 0, 1^2 = 1, 2^2 = 4, 3^2 = 9 = 2, 4^2 = 16 = 2, 5^2 = 25 = 4, 6^2 = 36 = 1$. Therefore the set of quadratic residues is $\{0, 1, 2, 4\}$.

Consider $f(x) = x^3 + 3x + 3$.

Now evaluate at all x values:

Output	Solutions
$f(0) \equiv 3 \bmod 7$	Non-residue.
$f(1) \equiv 0 \bmod 7$	$(1, 0)$
$f(2) \equiv 3 \bmod 7$	Non-residue.
$f(3) \equiv 4 \bmod 7$	$(3, 2), (3, 5)$
$f(4) \equiv 2 \bmod 7$	$(4, 3), (4, 4)$
$f(5) \equiv 3 \bmod 7$	Non-residue.
$f(6) \equiv 6 \bmod 7$	Non-residue.

5.3. Theorem about Existence of third point

Consider the point of infinity O to sit at the top of the y-axis and lies on every vertical line.

Every line intersecting $E(\mathbb{Z}_p)$ in 2 points, intersects it in exactly 3 points.

Note that a point P is counted 2 times if the line is tangent to the curve at P, and the point at infinity is also counted when the line is vertical.

5.4. Group Operation on $E(\mathbb{Z}_p)$

Consider two points P_1, P_2. Draw a line through P_1 and P_2, and then call the third point intersected (x, y). Then, $P_3 = (x, -y)$ and return that as the result of the operand. This third point must exist by the previous theorem.

For $P_1 = P_2$, consider the tangent line.

Define negation as $-(x, y) = (x, -y)$, and $-O = O$.

[6] I believe this is missing the distributive property.

For $P_2 = O$, we then return $-P_1$, so this also implies that we consider O the identity element.

This has closure by the above theorem, and it has identity via O, and inverse also exists by negation. Associativity is because there are exactly three elements.

This group is also abelian.

We can formally define this operation mathematically via formula. (You can see the slides if you care!)

5.5. Why useful

Practically, elliptic curves can use a smaller key size than other algorithms, so that is why they are used.

Consider the DDH over Elliptic curves where we seek to distinguish (aP, bP, abP) from (aP, bP, cP), where $a, b, c \in \mathbb{Z}$. In elliptic curves, multiplication is equivalent to exponentiation in a non-additive group.

Discrete log is equivalent to given aP return a, and for CDH, given aP, bP, return abP.

The best algorithms for these problems are slow in the size of the group. The fastest algorithms are $O(\sqrt{q})$, where $q = \left|E(\mathbb{Z}_p)\right| \approx p + 1$. But for $\mathbb{Z}_p$ it is $2^{\sqrt{\log(p)}\log\log(p)}$.

5.6. Hasse Bound

Let p be prime, and let E be an elliptic curve of $\mathbb{Z}_p$. Then:

$$p + 1 - 2\sqrt{p} \leq \left|E(\mathbb{Z}_p)\right| \leq p + 1 + 2\sqrt{p}$$

Heuristically, $y^2 = f(x)$ has two solutions whenever $f(x)$ is a quadratic residue, and 1 solution when $f(x) = 0$. Since half the elements of $\mathbb{Z}_p^*$ are quadratic residues, and if you take $f(x)$ as a random variable, you expect $2\left(\frac{p-1}{2}\right) + 1 = p$ points on the curve. Add O, and you get $\left|E(\mathbb{Z}_p)\right| \approx p + 1$ points.

5.7. Modular Exponentiation

Via repeated squaring, you can can compute the exponential:

```python
def mod_exp(a: int, m: int, N: int) -> int:
  """
  Computes a^m mod N.
  """
  if m == 0:
    return 1
  s = a
  temp = 1
  for i in range(0, m.bit_length()):
    bit = m & (1 << i) != 0
    if bit:
      temp = (temp * s) % N
    s = (s * s) % N

  return temp
```

This runs in time $O(\lg(m))$, so linear time with respect to the input size.

This process can apply to any repeated group operation.

6. Public Key Cryptography

The idea is that with a sender and receiver with an eavesdropper can exchange keys to then do private key cryptography. This is impossible information-theoretically, but is computationally feasible.

6.1. Key Agreement

The key-exchange experiment $\mathrm{KE}^{\mathrm{eav}}_{A,\Pi}(n)$:

1. Two parties execute the protocol Π and therefore generate a transcript, trans, of messages, and a key k output by each parties. They are given the input 1^n.
2. A uniform bit $b \in \{0,1\}$ is chosen, and if $b = 0$ set $\hat{k} := k$, and if $b = 1$, choose $\hat{k} \in \{0,1\}^n$ uniformly at random.
3. A is given trans and outputs a b', to guess which world it is in.
4. The output of the experiment is defined to be 1 if $b = b'$ and 0 otherwise.

A key exchange protocol Π is secure in the presence of adversaries if for all ppt adversaries A there is a negligible function negl such that:

$$\Pr\left[\mathrm{KE}^{\mathrm{eav}}_{A,\Pi}(n) = 1\right] \leq \frac{1}{2} + \mathrm{negl}(n)$$

This definition is written this way to be compatible with the prior requirements on the key k. Also, it is strictly stronger than a definition which required the adversary to guess the key.

6.2. Diffie-Hellman Key Exchange (G, g, q)

The party A chooses a random $x \in \mathbb{Z}_q$, computes $h_1 = g^x$, and sends h_1 to B. The party B chooses a random $y \in \mathbb{Z}_q$, computes $h_2 = g^y$ and sends h_2 to A.

Party A then has $k_A = h_2^x = (g^y)^x = g^{yx} = g^{xy}$, and party B then has $k_B = h_1^y = (g^x)^y = g^{xy}$, and therefore $k = k_A = k_B$.

This key exchange can be done in any group, including the elliptic curve group, where you would then "multiply" instead of exponentiate.

This is algorithm is secure in the presence of adversaries because this is exactly the distinguishing Diffie-Hellman problem (the transcript is (g^x, g^y), and the attacker seeks to see if the key is g^{xy}), and it is believed that the distinguishing Diffie-Hellman problem is hard over some groups, including the elliptic curve group, or the group of quadratic residues in $\mathbb{Z}_p^*$ where p is a strong prime.

6.3. Digital Signatures

A digital signature scheme consists of three ppt algorithms $(\mathrm{Gen}, \mathrm{Sign}, \mathrm{Vrfy})$ such that:

1. The key-generation algorithm Gen takes as input a security parameter 1^n and outputs a pair of keys $(\mathrm{pk}, \mathrm{sk})$[7]. We assume that each have length at least n, and that n can be determined from pk or sk.
2. The signing algorithm Sign takes as input a private key sk and a message m from some message space (that may depend on pk). It outputs a signature σ and we write this as $\sigma \leftarrow \mathrm{Sign}_{\mathrm{sk}}(m)$.

6.4. Schnorr Identification scheme

A prover has a value x. Then, at random, $k \leftarrow \mathbb{Z}_q$. Then, $I := g^k$, and it sends I to the verifier. A verifier will query some $r \leftarrow \mathbb{Z}_q$ and the prover will respond with $s := [rx + k \bmod q]$. To verify, the verifier will check whether $g^s y^{-r}$ equals I. If yes, accept, otherwise, reject.

This works because $g^{rx+k}y^{-r} = g^{rx+k}(g^x)^{-r} = g^{rx+k}g^{-rx} = g^{rx+k-rk} = g^k = I$.

There is a theorem that proves that if the Dlog problem is hard relative to G, then the Schnorr Identification scheme is secure.

[7] p is for private, s is for secret.

The prover has access to a value x, and the verifier has access to a value $y \coloneqq g^x$. The idea is that the prover can show that it knows what x is without leaking out the value of x.

This protocol is functional as a “honest-verifier” zero-knowledge proof, unclear if you can prove it for non-honest verifiers. This will not matter, as the Fiat-Shamir Transform only requires zero-knowledge.

6.5. Forking Lemma

6.6.

However, a simulator can generate a transcript:

Choose $(r, s) \leftarrow \mathbb{Z}_q \times \mathbb{Z}_q$ uniformly at random. Then, $I \coloneqq g^s y^{-r}$, and output (I, r, s).

Using this simulation, you can show that this is equivalent to DDH, because if anyone can generate values that satisfy the identification scheme, then you can’t tell if its just random data or a Schnorr identification scheme (I am confused how this works out, presumably explained in the video.)

6.7. Fiat-Shamir Transform

You can convert a identification scheme to a non-interactive proof by using a hash function (that behaves like a random oracle) to generate the value r. This allows the problem about transcripts generated by simulators be sidestepped, because, intuitively, the problem was that the I could be chosen after creating an r, but with the hash function, the I value must come first (assuming that the hash function cannot be inverted, nor can it be used in a way that is convenient for creating values that align in this way; assuming a random oracle is sufficient).

For the specific Schnorr Identification scheme, it goes as follows, forming the Schnorr Signature Scheme:

So, $r \coloneqq H(I \parallel m)$. Then, the signature is (r, s).

For verification, compute $I = g^s g^{-r}$. Check that $H(I \parallel m) = r$. If this holds, accept, otherwise, reject.

7. Post Quantum Security

7.1. Are current algorithms secure?

Our traditional assumptions are that factoring and discrete log are hard problems. However, algorithmic progress is continuing! The algorithms for factoring and for discrete log are only getting better, and using quantum computers, both can be done in $O(n^2)$ time with Shor's algorithm.

7.2. Post-quantum approach

There is a new set of assumptions based on finding short vectors in lattices, and this set of problems is believed to be hard for quantum computers.

7.3. Lattices

An n-dimensional lattice L is an additive discrete subgroup of R^n. A basis $B \in R^{n \times n}$ defines a lattice $L(B)$ in the following way:

$$L(B) = \{v \in R^n \mid v = Bz, z \in \mathbb{Z}\}$$

The i-th successive minima $\lambda_i(L(B))$ is the smallest radius r such that there are i linearly independent vectors $\{v_1, ..., v_i\}$ of length at most r.

Note that a basis is not unique! For example, the best basis could be:

$$\begin{pmatrix} 1 & -2 \\ 2 & 3 \end{pmatrix}$$

But it is also true that:

$$\begin{pmatrix} 1 & 0 \\ 2 & 7 \end{pmatrix}$$

is the same basis.

Given two bases, B and B', they define the same lattice iff $B' = BU$ where U is a unimodular matrix (determinant is ± 1)

7.4. Hard Lattice Problems

All hard lattice problems are parametrized by an approximation factor $\gamma > 1$. (usually, $\gamma = \sqrt{n}$ (?))

1. Shortest vector problem: Given a basis B, find a non-zero vector $v \in L(B)$ whose length is at most $\gamma \lambda_1(L(B))$.
2. Shortest independent vector problem: Given a basis B, find a linearly independent set $v_1, ..., v_n$, such that all vectors have length at most $\gamma \cdot \lambda_n(L(B))$.
3. Gap Shortest vector problem (GapSVP): Given a basis B and a radius $r > 0$:
   - Return YES if $\lambda_1(L(B)) \leq r$
   - Return NO if $\lambda_1(L(B)) > \gamma r$

7.5. Rejection Sampling

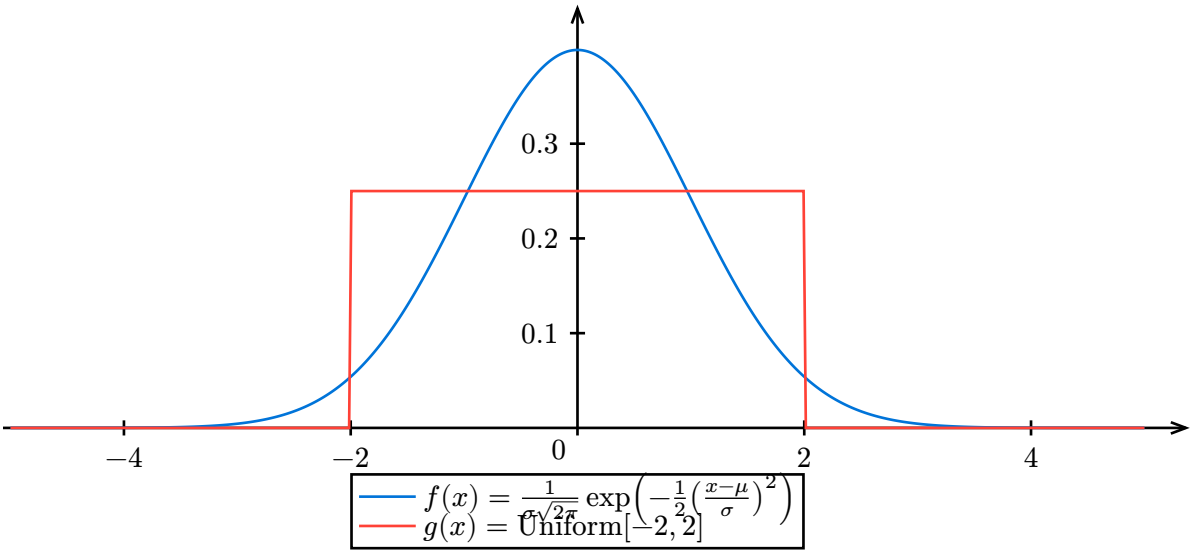
To sample from a distribution D_f with probability density function $f(x)$ given draws from a distribution D_g, assuming $\forall x, f(x) \leq M \cdot g(x)$.

1. Sample from $x \leftarrow D_g$
2. Accept x with probability $\frac{f(x)}{Mg(x)}$.

If the condition holds, then $\forall x, \frac{f(x)}{Mg(x)} \leq 1$.

The probability of output x is the probability of sampling x from D_g times the probability the sample is accepted, the pdf of this is then $g(x)\frac{f(x)}{Mg(x)} = \frac{f(x)}{M}$. Normalizing this, we ge the correct probability distribution.

7.5.1. Example



Since we need $\forall x, f(x) \leq M \cdot g(x)$, we need $M \geq \max\left(\frac{f(x)}{g(x)}\right)$. IOW, if $f(x) \geq Mg(x)$, you need to increase M to use this algorithm. For the specific example above, $f(0)$ maximizes (from –2 to 2), so $\frac{1}{\sqrt{2\pi}}$ is the max, and $g(x) = \frac{1}{4}$, so the minimal M is $M = \frac{4}{\sqrt{2\pi}} \approx 1.596$.

7.6. Lyubashevsky Lattice-Based Signatures

Take a matrix $A \in M_{n \times m}(\mathbb{Z}_p)$, and a second matrix $S \in M_{m \times k}$. Let A's entries be selected uniformly from $\mathbb{Z}_p$ and S's entries be selected uniform from $[-d, -d]$. Then, produce $T = AS$.

Consider A and T as public keys.

If we consider this as a identification scheme we can consider the prover to first sample $y \leftarrow D_\sigma^m$[8], and then produces $d = Ay$, to send to the challenger. (d is like I).

Then, the responder will query c (like r), uniformly selected.

Then the prover will calculate $z := Sc + y$, and sends a proof (like s).

To verify, the responder will verify that $Az - Tc = d$ and that z is short.

Then, apply the Fiat-Shamir Transform to create a signature scheme:

So, $c := H(d \parallel m)$. Then, the signature is (c, z).

For verification, compute $Az - Tc = d$ and check that z is short. Check that $H(d \parallel m) = c$. If this holds, accept, otherwise, reject.

Consider if we have two different transcripts with the same d:

$$d = Az_1 - Tc_1$$
$$d = Az_2 - Tc_2$$

Then:

$$A(z_1 - z_2) - T(c_1 - c_2) = 0$$
$$A(z_1 - z_2) - A(S(c_1 - c_2)) = 0$$

Finding z_1, z_2 is as hard as SIS.

[8] D_σ^m denotes the discrete Gaussian of dimension m with variance σ, and $D_{\sigma,\mu}^m$ denotes the discrete Gaussian of dimension m with variance σ and mean μ

Now, we must prove that the zero-knowledge proof is actually correct (specifically, the zero knowledge part).

The problem is that $Sc + y$ leaks information about S. In the Schnorr way, this d value has no relation to the secret key whatsoever. So this doesn't actually work! The problem is that the distribution of $Sc + y$, repeated enough times, has a mean that is non-zero, and that mean would leak Sc in it's entirety.

So, we want the d value to leak no information. So, we will use rejection sampling to ensure the mean (and distribution) is correct. specifically, we will output z with probability:

$$\frac{D_{\sigma}^{m}(z)}{MD_{\sigma,Sc}^{m}(z)}$$

This does not work in an interactive way, because failing leaks information. But it does work in a signature scheme! You can just select a new y again. So the new "honest prover" must also be very forgetful and never remember anything if it failed. Then, the same proof about recreating transcripts holds to show that this does not leak information.