Recap

- We have a perfectly secret encryption scheme (one-time pad)...
- . . . but it requires long keys
- This is inevitable if we insist on perfect secrecy (recall that, in a perfectly secret scheme, $|\mathcal{K}| \geq |\mathcal{M}|$)

Recap

- We have a perfectly secret encryption scheme (one-time pad)...
- . . . but it requires long keys
- This is inevitable if we insist on perfect secrecy (recall that, in a perfectly secret scheme, $|\mathcal{K}| \geq |\mathcal{M}|$)
- We have a security definition that allows for short keys and works against adversaries with polynomially bounded running times

Recap

- We have a perfectly secret encryption scheme (one-time pad)...
- . . . but it requires long keys
- This is inevitable if we insist on perfect secrecy (recall that, in a perfectly secret scheme, $|\mathcal{K}| \geq |\mathcal{M}|$)
- We have a security definition that allows for short keys and works against adversaries with polynomially bounded running times

Is there a secure private-key encryption scheme (with short keys) according to this new definition?

Recap: Pseudorandom Number Generators (formal)

Let G be a deterministic polynomial-time algorithm such that for any n and any input $s \in \{0,1\}^n$, the output $G(s)$ is a string of length $\ell(n)$

Expansion factor of G

G is a **pseudorandom generator (PRG)** if the following conditions hold:

- **Expansion:** For every $n \geq 1$, $\ell(n) > n$
- **Pseudorandomness:** For any probabilistic polynomial-time algorithm D, there is a negligible function η such that

$$\left| \Pr[D(\,G(s)\,) = 1] - \Pr[D(r) = 1] \right| \leq \eta(n)$$

where s is a uniform random variable in $\{0,1\}^n$ and r is a uniform random variable in $\{0,1\}^{\ell(n)}$

Recap: Pseudorandom Number Generators (formal)

Let G be a deterministic polynomial-time algorithm such that for any n and any input $s \in \{0,1\}^n$, the output $G(s)$ is a string of length $\ell(n)$

Expansion factor of G

G is a **pseudorandom generator (PRG)** if the following conditions hold:

- **Expansion:** For every $n \geq 1$, $\ell(n) > n$
- **Pseudorandomness:** For any probabilistic polynomial-time algorithm D, there is a negligible function η such that

Probability over the randomness of D and the choice of s

$$\left| \Pr[D(\,G(s)\,) = 1] - \Pr[D(r) = 1] \right| \leq \eta(n)$$

where s is a uniform random variable in $\{0,1\}^n$ and r is a uniform random variable in $\{0,1\}^{\ell(n)}$

Recap: Pseudorandom Number Generators (formal)

Let G be a deterministic polynomial-time algorithm such that for any n and any input $s \in \{0,1\}^n$, the output $G(s)$ is a string of length $\ell(n)$ ← Expansion factor of G

G is a **pseudorandom generator (PRG)** if the following conditions hold:

- **Expansion:** For every $n \geq 1$, $\ell(n) > n$
- **Pseudorandomness:** For any probabilistic polynomial-time algorithm D, there is a negligible function η such that

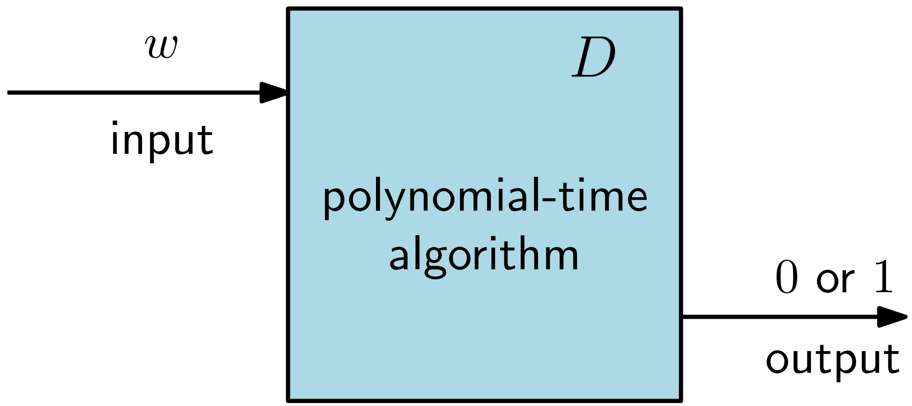
Probability over the randomness of D and the choice of s →

$$\left| \Pr[D(\,G(s)\,) = 1] - \Pr[D(r) = 1] \right| \leq \eta(n)$$

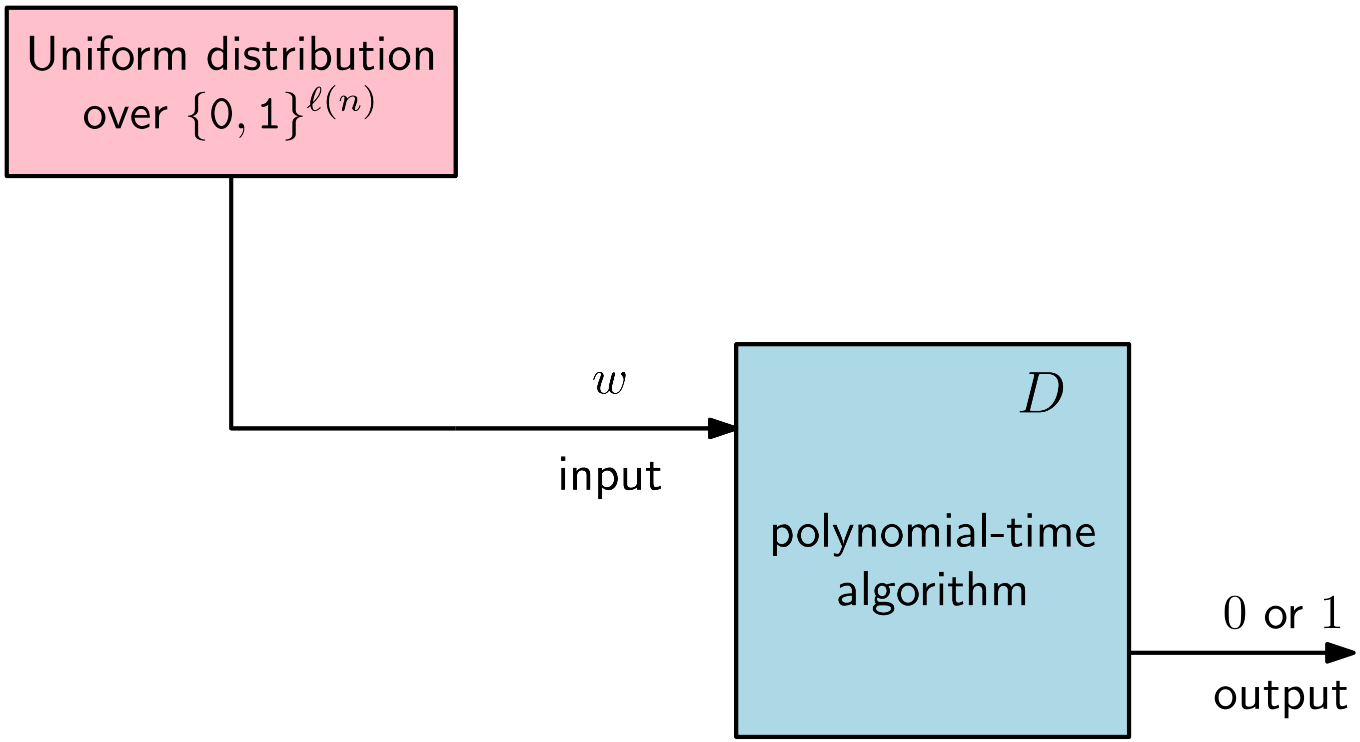
← Probability over the randomness of D and the choice of r

where s is a uniform random variable in $\{0,1\}^n$ and r is a uniform random variable in $\{0,1\}^{\ell(n)}$

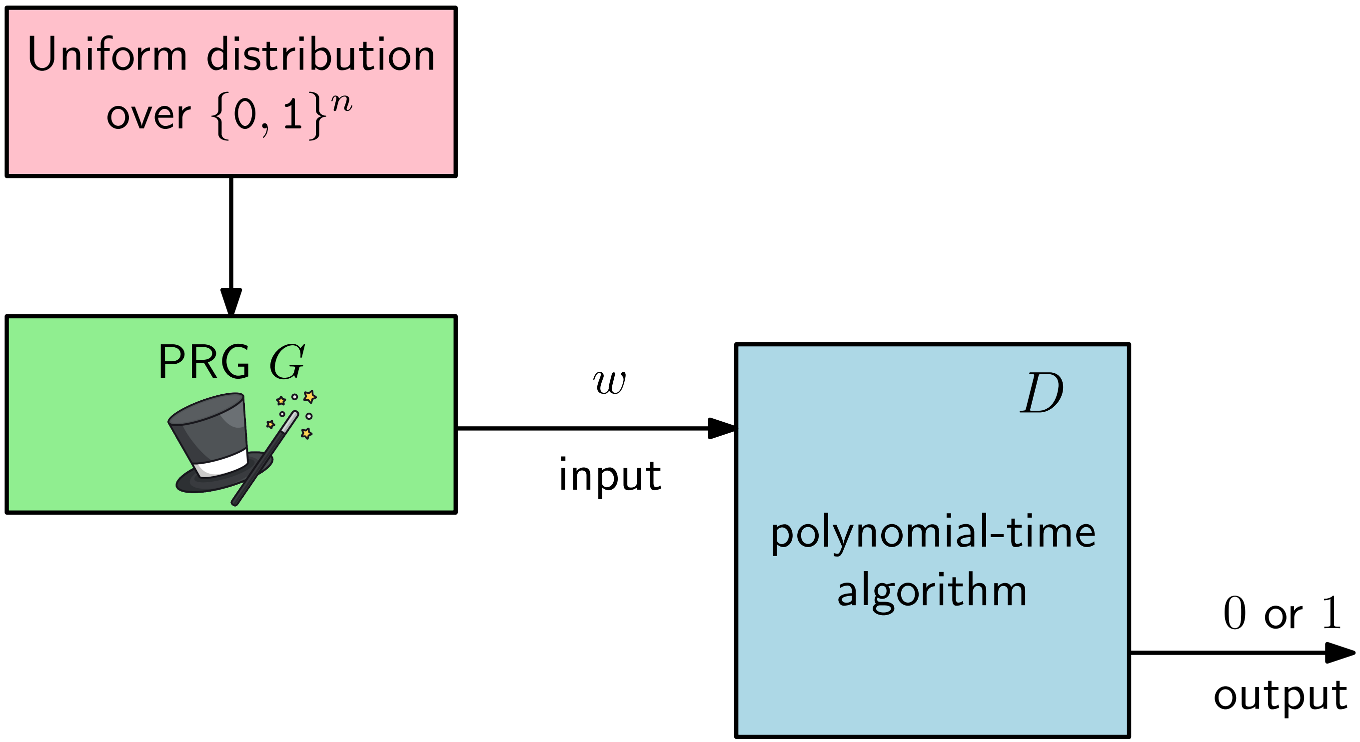
Distinguishers



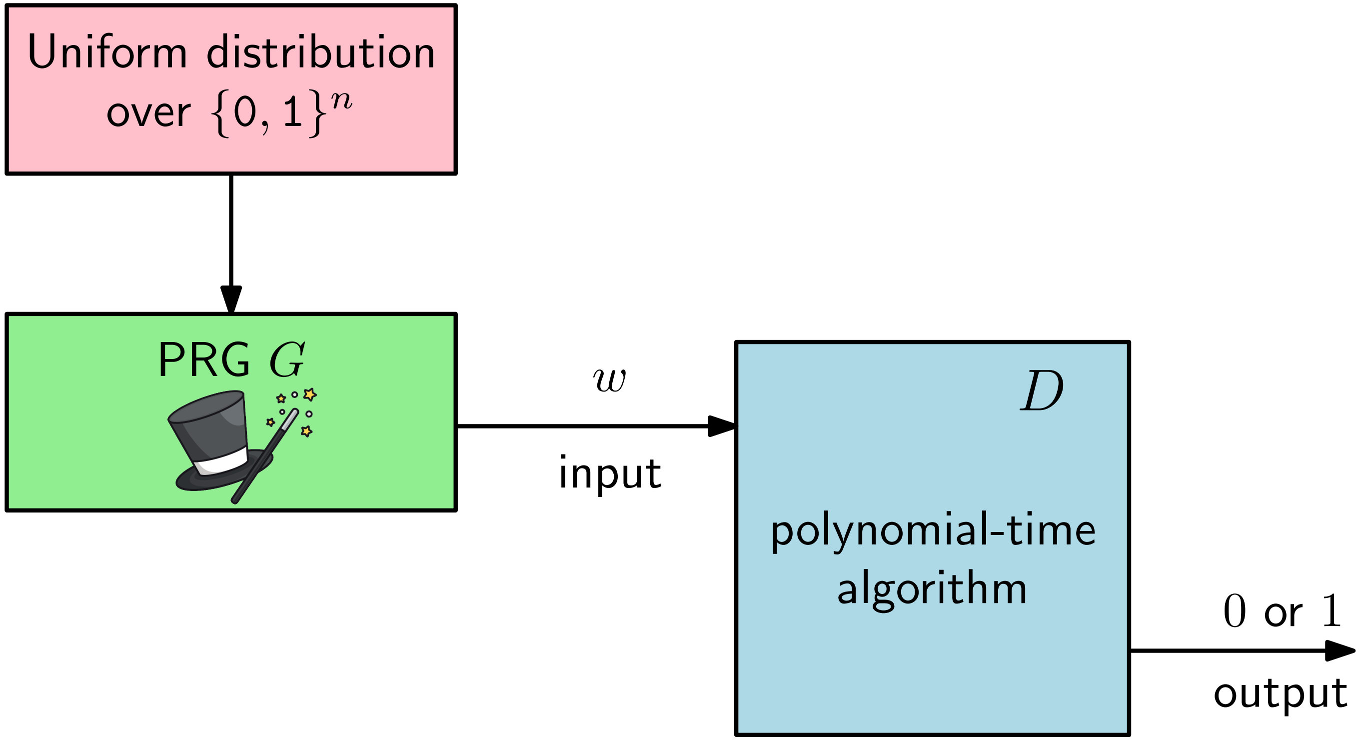
Distinguishers



Distinguishers



Distinguishers



Regardless of how the input x is generated, the probability that D outputs 1 should be almost the same (the two probabilities differ by at most a negligible function)

Examples

Consider a polynomial-time algorithm G that, with input $s = s_1 s_2 \ldots s_n$ outputs $G(s) = s \,\|\, \bigvee_{i=1}^{n} s_i$

$$s = 000000 \longrightarrow G(s) = 0000000$$

$$s = 001011 \longrightarrow G(s) = 0010111$$

Is it a PRG?

Examples

Consider a polynomial-time algorithm G that, with input $s = s_1 s_2 \dots s_n$ outputs $G(s) = s \,\|\, \bigvee_{i=1}^{n} s_i$

$s = 000000 \longrightarrow G(s) = 0000000$

$s = 001011 \longrightarrow G(s) = 0010111$

Is it a PRG?

Distinguisher $\mathcal{D}(w)$: $w = w_1 w_2 \dots w_n w_{n+1}$

- If $w_{n+1} = 1$:
  - Output 1 (guess that w "is pseudorandom")
- Otherwise output 0 (guess that w "is truly random")

Examples

Consider a polynomial-time algorithm G that, with input $s = s_1 s_2 \ldots s_n$ outputs $G(s) = s \,\|\, \bigvee_{i=1}^{n} s_i$

$s = 000000 \longrightarrow G(s) = 0000000$

$s = 001011 \longrightarrow G(s) = 0010111$

Is it a PRG?

Distinguisher $\mathcal{D}(w)$: $w = w_1 w_2 \ldots w_n w_{n+1}$

- If $w_{n+1} = 1$:
  - Output 1 (guess that w "is pseudorandom")
- Otherwise output 0 (guess that w "is truly random")

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

$= \Pr[s$ contains at least a 1$]$

$= 1 - \frac{1}{2^n}$

Examples

Consider a polynomial-time algorithm G that, with input $s = s_1 s_2 \dots s_n$ outputs $G(s) = s \,\|\, \bigvee_{i=1}^{n} s_i$

$s = 000000 \longrightarrow G(s) = 0000000$

$s = 001011 \longrightarrow G(s) = 0010111$

Is it a PRG?

Distinguisher $\mathcal{D}(w)$: $w = w_1 w_2 \dots w_n w_{n+1}$

- If $w_{n+1} = 1$:
  - Output 1 (guess that w "is pseudorandom")
- Otherwise output 0 (guess that w "is truly random")

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

  $= \Pr[s$ contains at least a 1$]$

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

Examples

Consider a polynomial-time algorithm G that, with input $s = s_1 s_2 \dots s_n$ outputs $G(s) = s \,\|\, \bigvee_{i=1}^{n} s_i$

$s = 000000$ ⟶ $G(s) = 0000000$

$s = 001011$ ⟶ $G(s) = 0010111$

Is it a PRG?

Distinguisher $\mathcal{D}(w)$: $w = w_1 w_2 \dots w_n w_{n+1}$

- If $w_{n+1} = 1$:
  - Output 1 (guess that w "is pseudorandom")
- Otherwise output 0 (guess that w "is truly random")

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

  $= \Pr[s$ contains at least a 1$]$

  $= 1 - \frac{1}{2^n}$

- $\Pr[D(r) = 1] = \Pr[w_{n+1} = 1] = \frac{1}{2}$

$$\left|1 - \frac{1}{2^n} - \frac{1}{2}\right| = \frac{1}{2} - \frac{1}{2^n} \text{ is not negligible}$$

Example

Consider a (polynomial-time) algorithm G that takes a binary string $s = s_1 \dots s_n \in \{0,\ 1\}^n$ and outputs a string in $f(s) \in \{0,\ 1\}^{n+1}$ such that:

$$G(s) = s \,\Big\|\, \bigoplus_{i=1}^{n} s_i$$

Example

Consider a (polynomial-time) algorithm G that takes a binary string $s = s_1 \ldots s_n \in \{\mathtt{0},\ \mathtt{1}\}^n$ and outputs a string in $f(s) \in \{\mathtt{0},\ \mathtt{1}\}^{n+1}$ such that:

$$G(s) = s \,\Big\|\, \bigoplus_{i=1}^{n} s_i$$

Is G a pseudorandom generator?

Example

Consider a (polynomial-time) algorithm G that takes a binary string $s = s_1 \dots s_n \in \{\texttt{0},\ \texttt{1}\}^n$ and outputs a string in $f(s) \in \{\texttt{0},\ \texttt{1}\}^{n+1}$ such that:

$$G(s) = s \,\Big\|\, \bigoplus_{i=1}^{n} s_i$$

Is G a pseudorandom generator? No Can we prove that?

Example

Consider a (polynomial-time) algorithm G that takes a binary string $s = s_1 \ldots s_n \in \{\mathtt{0},\ \mathtt{1}\}^n$ and outputs a string in $f(s) \in \{\mathtt{0},\ \mathtt{1}\}^{n+1}$ such that:

$$G(s) = s \,\Big\|\, \bigoplus_{i=1}^{n} s_i$$

Is G a pseudorandom generator? No

Can we prove that?

We need to design a distinguisher D for G...

Example

Consider a (polynomial-time) algorithm G that takes a binary string $s = s_1 \ldots s_n \in \{0, 1\}^n$ and outputs a string in $f(s) \in \{0, 1\}^{n+1}$ such that:

$$G(s) = s \,\Big\|\, \bigoplus_{i=1}^{n} s_i$$

Is G a pseudorandom generator? No

Can we prove that?

We need to design a distinguisher D for G...

$D(w_1 w_2 \ldots, w_{n+1})$:

- If $w_{n+1} = \bigoplus_{i=1}^{n} w_i$: return 1
- Otherwise, return 0

Example

Consider a (polynomial-time) algorithm G that takes a binary string $s = s_1 \dots s_n \in \{\texttt{0},\ \texttt{1}\}^n$ and outputs a string in $f(s) \in \{\texttt{0},\ \texttt{1}\}^{n+1}$ such that:

$$G(s) = s \, \Big\| \, \bigoplus_{i=1}^{n} s_i$$

Is G a pseudorandom generator? No

Can we prove that?

We need to design a distinguisher D for G...

$D(w_1 w_2 \dots, w_{n+1})$:

- If $w_{n+1} = \bigoplus_{i=1}^{n} w_i$: return 1
- Otherwise, return 0

$$\Pr[D(G(s)) = 1] = \Pr[w_{i+1} = \bigoplus_{i=1}^{n} s_i] = 1$$

Example

Consider a (polynomial-time) algorithm G that takes a binary string $s = s_1 \dots s_n \in \{\texttt{0},\ \texttt{1}\}^n$ and outputs a string in $f(s) \in \{\texttt{0},\ \texttt{1}\}^{n+1}$ such that:

$$G(s) = s \,\Big\|\, \bigoplus_{i=1}^{n} s_i$$

Is G a pseudorandom generator? No

Can we prove that?

We need to design a distinguisher D for G...

$D(w_1 w_2 \dots, w_{n+1})$:

- If $w_{n+1} = \bigoplus_{i=1}^{n} w_i$: return 1
- Otherwise, return 0

$$\Pr[D(G(s)) = 1] = \Pr[w_{i+1} = \bigoplus_{i=1}^{n} s_i] = 1$$

$$\Pr[D(r) = 1] = \Pr[r_{i+1} = \bigoplus_{i=1}^{n} r_i] = \frac{1}{2}$$

Example

Consider a (polynomial-time) algorithm G that takes a binary string $s = s_1 \ldots s_n \in \{0,\ 1\}^n$ and outputs a string in $f(s) \in \{0,\ 1\}^{n+1}$ such that:

$$G(s) = s \,\Big\|\, \bigoplus_{i=1}^{n} s_i$$

Is G a pseudorandom generator? No

Can we prove that?

We need to design a distinguisher D for G...

$D(w_1 w_2 \ldots, w_{n+1})$:

- If $w_{n+1} = \bigoplus_{i=1}^{n} w_i$: return 1
- Otherwise, return 0

$$\Pr[D(G(s)) = 1] = \Pr[w_{i+1} = \bigoplus_{i=1}^{n} s_i] = 1$$

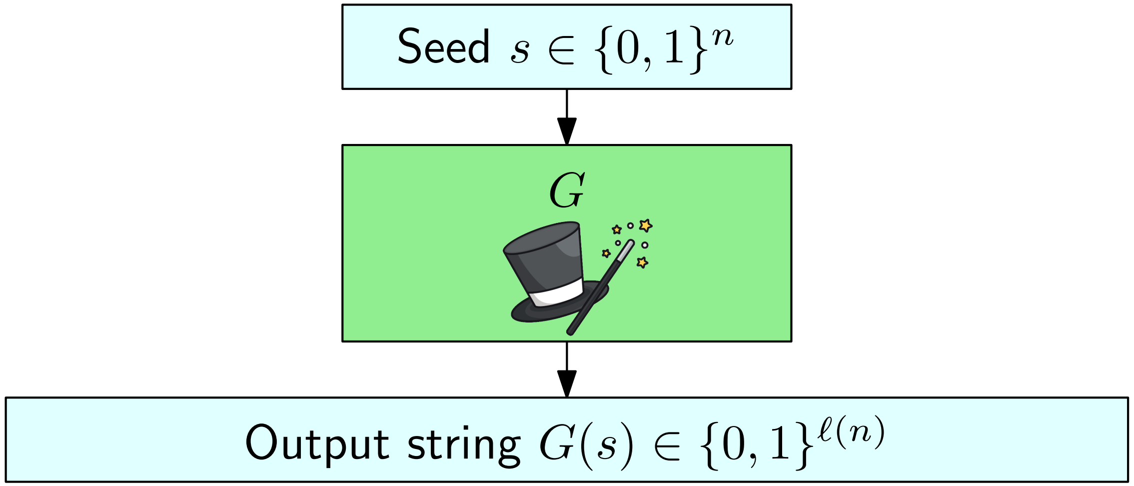
$$\Pr[D(r) = 1] = \Pr[r_{i+1} = \bigoplus_{i=1}^{n} r_i] = \frac{1}{2}$$

$$\Big| \Pr[D(\, G(s)\,) = 1] - \Pr[D(r) = 1] \Big| = \frac{1}{2}$$

Not negligible!

Why are PRGs useful?

As far as polynomial-time algorithms are concerned, the output of $G(s)$ with a random seed s is indistinguishable (up to some negligible probability) from a random string r

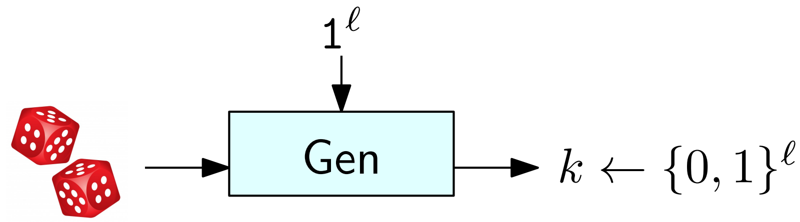


If we have a randomized polynomial-time algorithm that uses r random bits, and we replace those random bits with the output of $G(s)$, the resulting (randomized) algorithm "behaves the same" except for a negligible probability

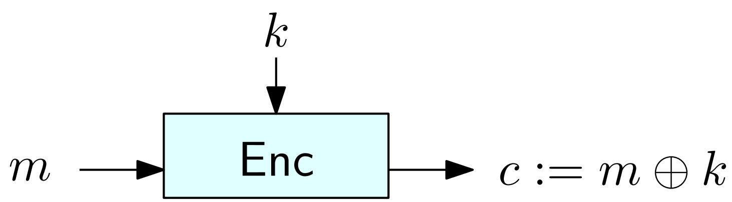
One-time pad (redefined with security parameter)

security parameter ℓ = length of the message   (for convenience we name the security parameter ℓ instead of n)

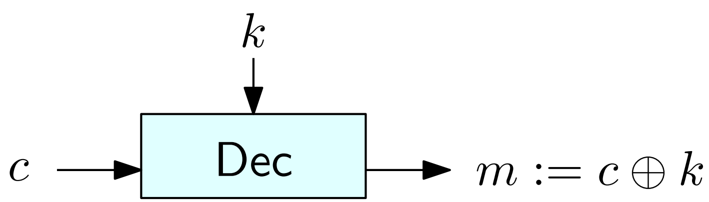
- Gen(1^ℓ):   return a key k chosen u.a.r. from $\{0,1\}^\ell$



- Enc$_k(m)$:   return $c := k \oplus m$



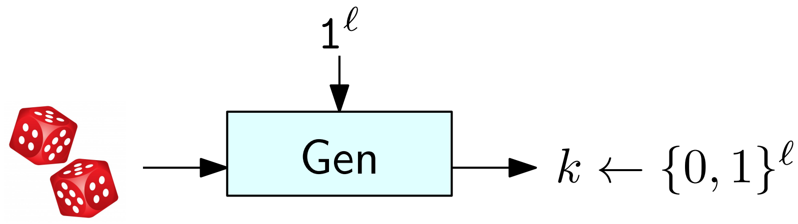
- Dec$_k(c)$:   return $m := k \oplus c$



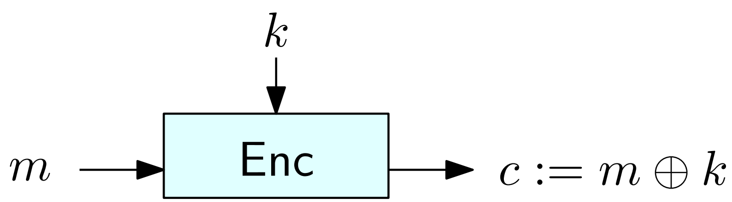
One-time pad (redefined with security parameter)

security parameter ℓ = length of the message (for convenience we name the security parameter ℓ instead of n)

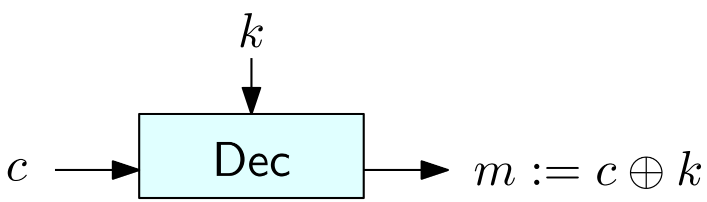
- Gen(1^ℓ): return a key k chosen u.a.r. from $\{0,1\}^\ell$



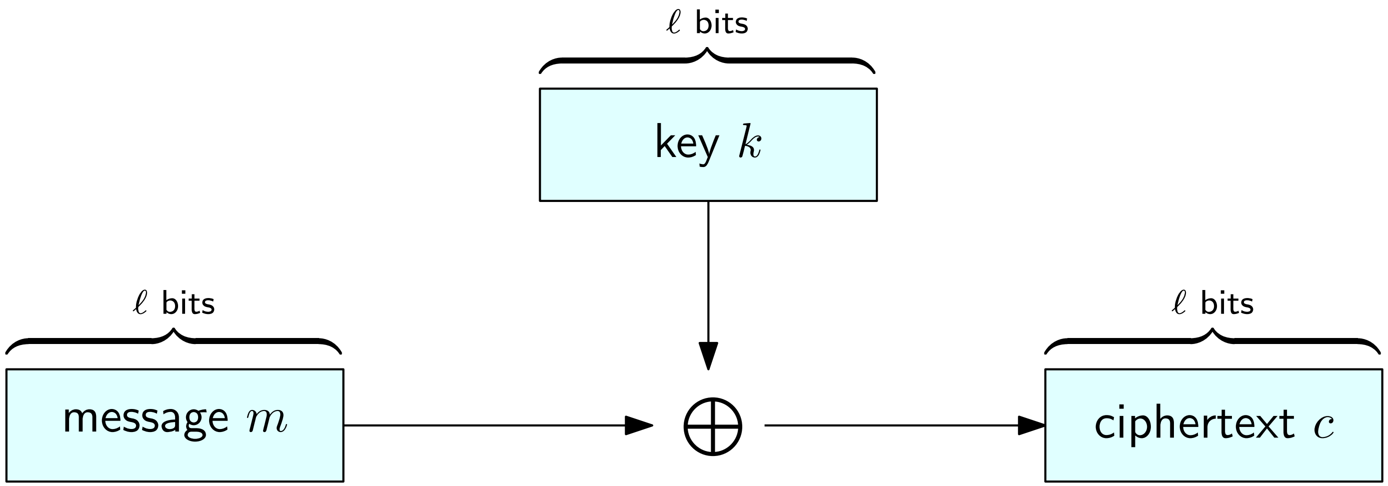
- Enc$_k(m)$: return $c := k \oplus m$



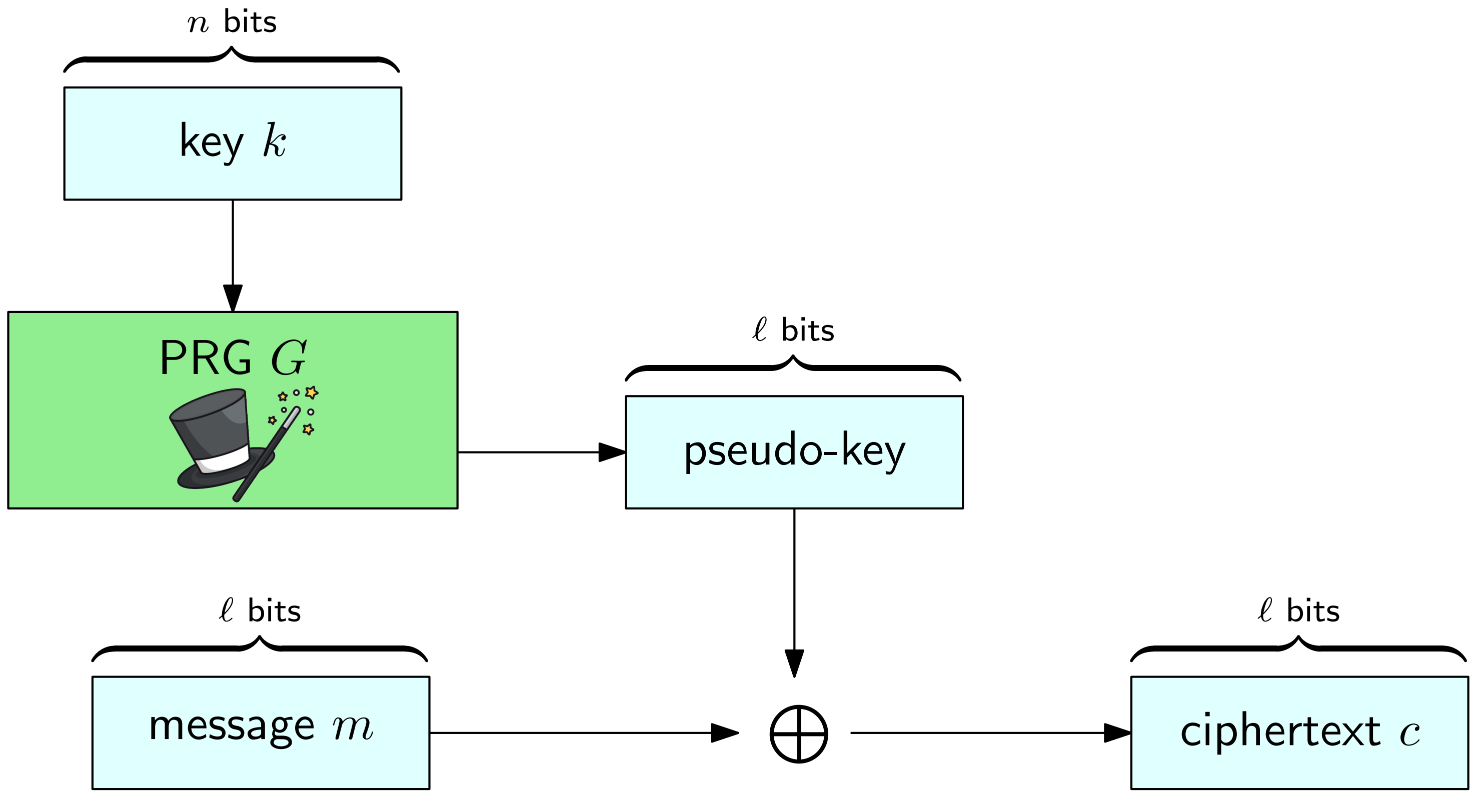
- Dec$_k(c)$: return $m := k \oplus c$



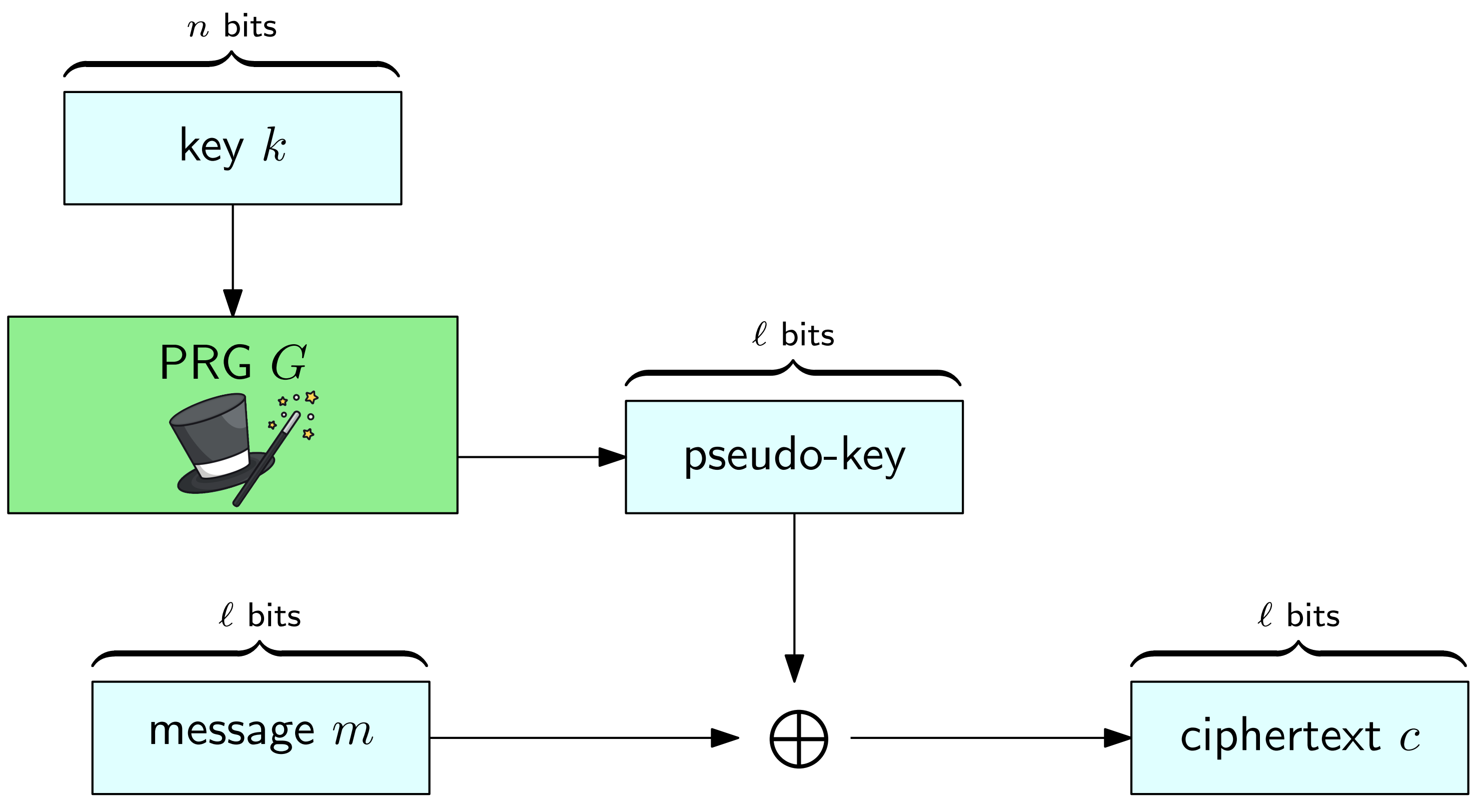
One-time pad, encryption



Pseudo one-time pad, encryption

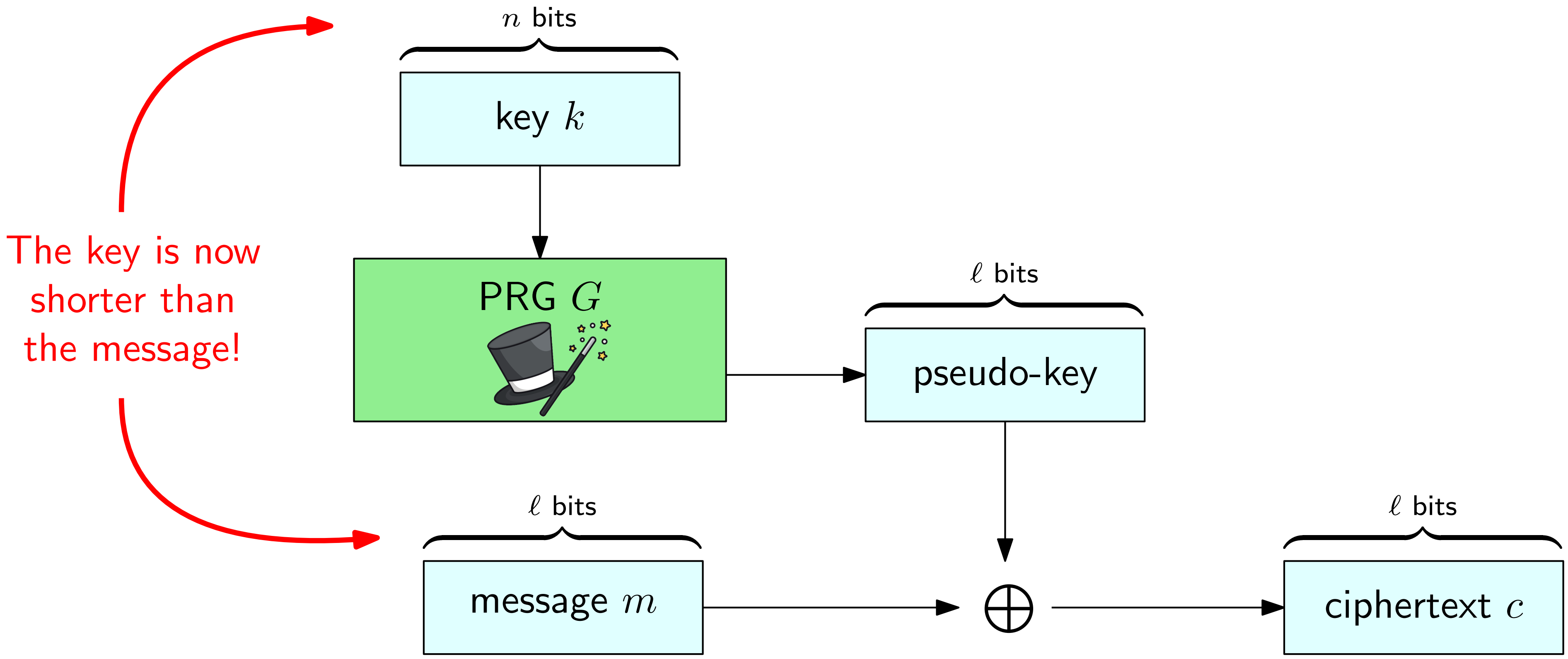


Pseudo one-time pad, encryption



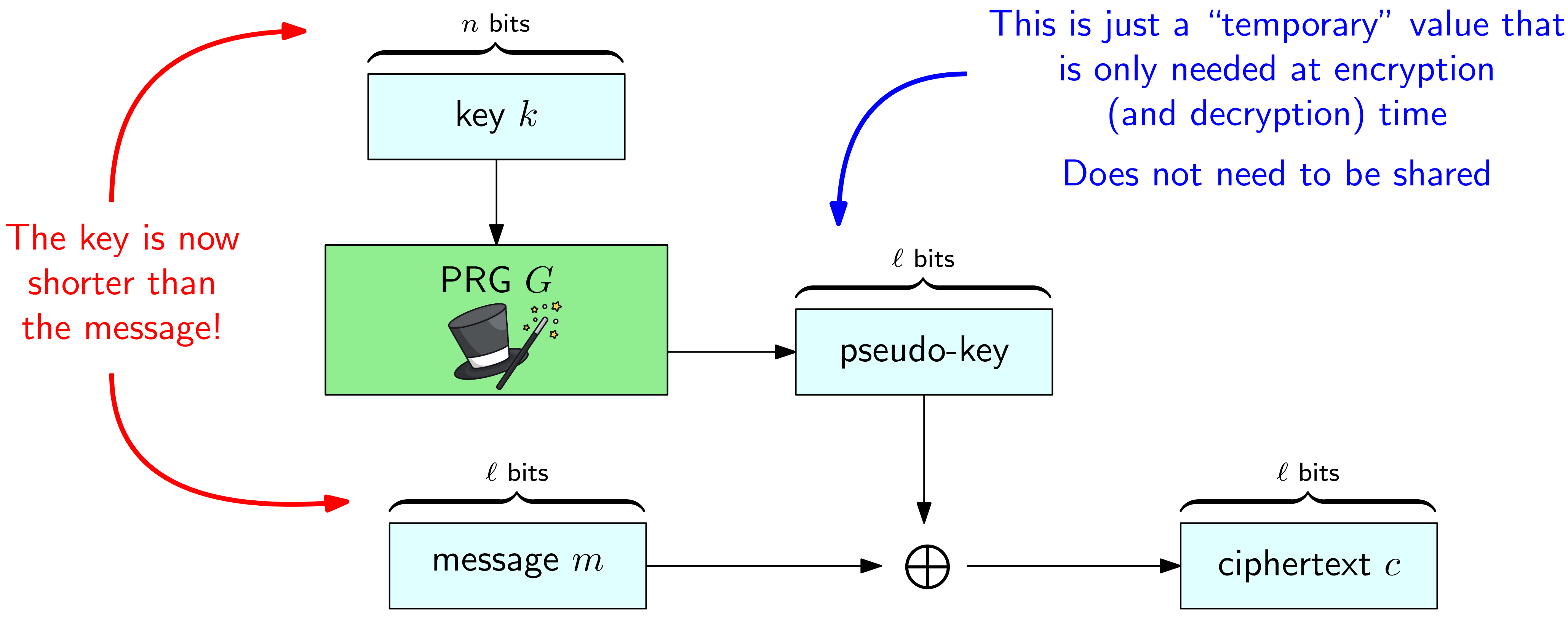
Intuitively, the output (distribution) should be indistinguishable to any polynomial-time adversary (except for a negligible probability)

Pseudo one-time pad, encryption



Intuitively, the output (distribution) should be indistinguishable to any polynomial-time adversary (except for a negligible probability)

Pseudo one-time pad, encryption

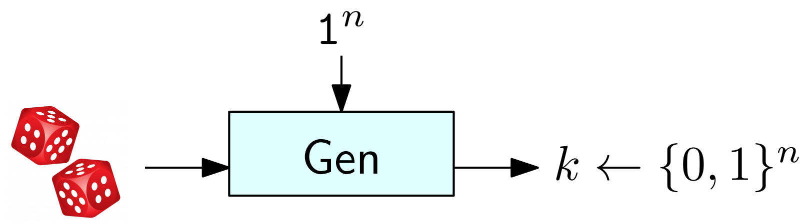


Intuitively, the output (distribution) should be indistinguishable to any polynomial-time adversary (except for a negligible probability)

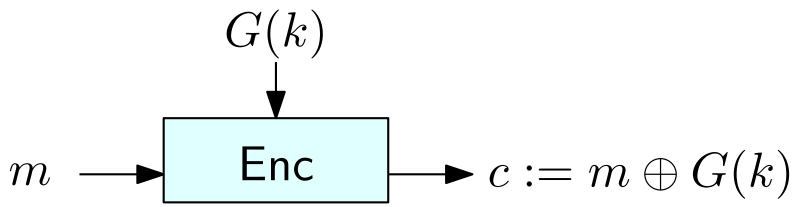
Pseudo one-time pad

Let G be a PRG with expansion factor $\ell(n)$

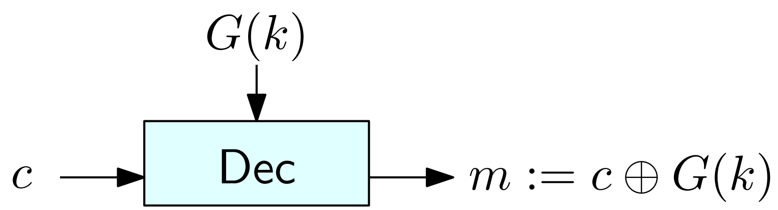
- Gen(1^n): return a key k chosen u.a.r. from $\{0,1\}^n$



- Enc$_k(m)$: return $c := G(k) \oplus m$



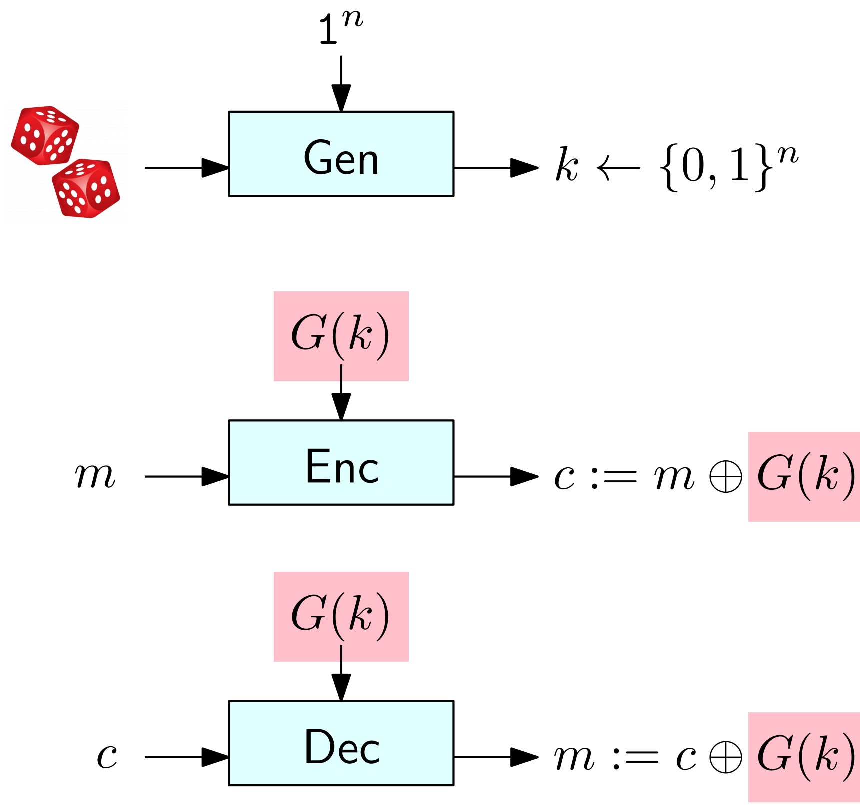
- Dec$_k(c)$: return $m := G(k) \oplus c$



Pseudo one-time pad

Let G be a PRG with expansion factor $\ell(n)$

- Gen(1^n): return a key k chosen u.a.r. from $\{0,1\}^n$
- Enc$_k(m)$: return $c := G(k) \oplus m$
- Dec$_k(c)$: return $m := G(k) \oplus c$



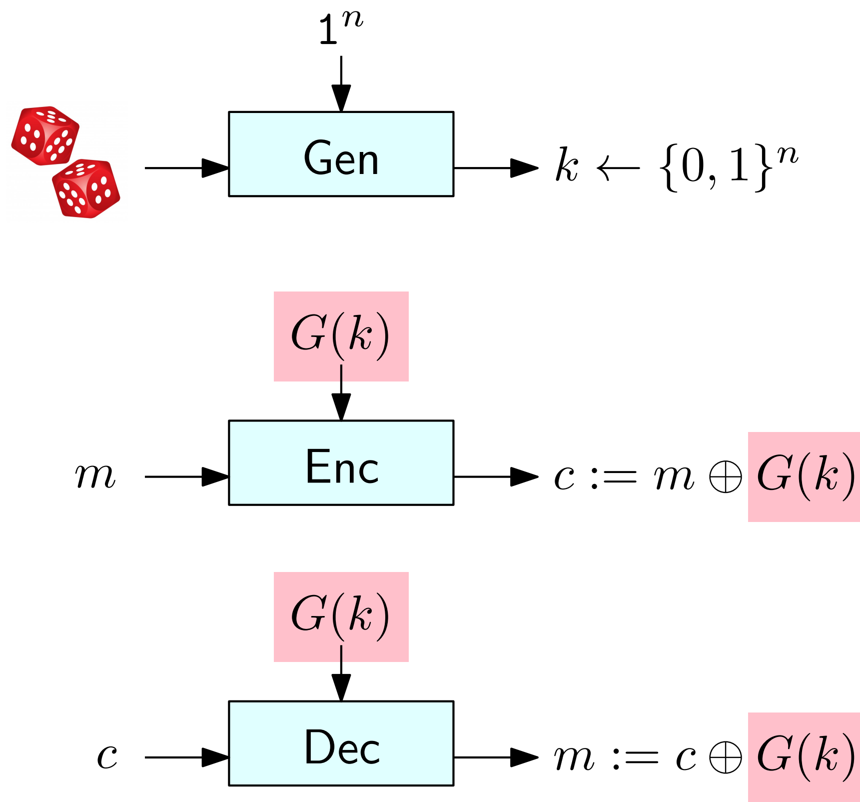
Pseudo one-time pad

Let G be a PRG with expansion factor $\ell(n)$

Key space: $\{0,1\}^n$

Message space: $\{0,1\}^{\ell(n)}$

- $\mathsf{Gen}(1^n)$: return a key k chosen u.a.r. from $\{0,1\}^n$
- $\mathsf{Enc}_k(m)$: return $c := G(k) \oplus m$
- $\mathsf{Dec}_k(c)$: return $m := G(k) \oplus c$



Cryptographic assumptions

Is pseudo OTP **EAV**-secure?

Cryptographic assumptions

Is pseudo OTP **EAV**-secure?

We cannot prove security **unconditionally**

Cryptographic assumptions

Is pseudo OTP **EAV**-secure?

We cannot prove security **unconditionally**

We can hope to prove security based on some **cryptographic assumption**

- The weaker the assumption, the better

Cryptographic assumptions

Is pseudo OTP **EAV**-secure?

We cannot prove security **unconditionally**

We can hope to prove security based on some **cryptographic assumption**

- The weaker the assumption, the better

In our case we prove security of pseudo-OTP, conditioned on the assumption that PRGs exist

- Stronger than $P \neq NP$

Cryptographic assumptions

Is pseudo OTP **EAV**-secure?

We cannot prove security **unconditionally**

We can hope to prove security based on some **cryptographic assumption**

- The weaker the assumption, the better

In our case we prove security of pseudo-OTP, conditioned on the assumption that PRGs exist

- Stronger than P $\neq$ NP

In general, even stronger cryptographic assumptions might be needed to prove that a scheme is secure

Reductions

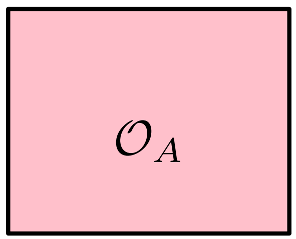
Think about (Cook) reductions in complexity theory:

- Let A and B be two decision problems, where B is NP-complete

Reductions

Think about (Cook) reductions in complexity theory:

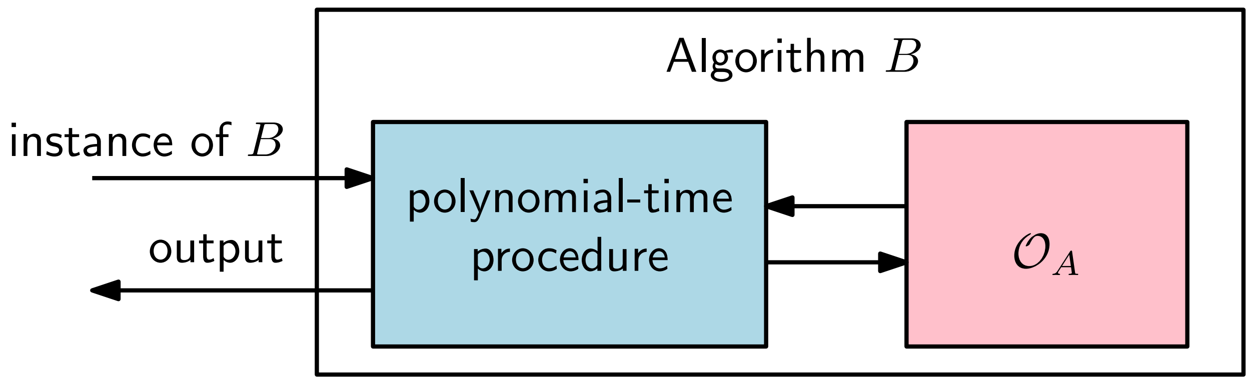
- Let A and B be two decision problems, where B is NP-complete
- Assume to have access an efficient (polynomial-time) “black-box” (an *oracle*) $\mathcal{O}_A$ that solves A



Reductions

Think about (Cook) reductions in complexity theory:

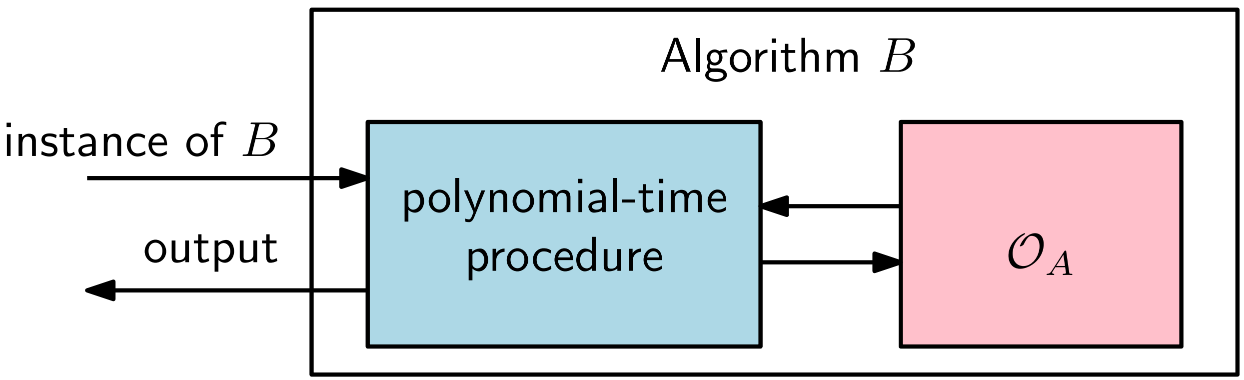
- Let A and B be two decision problems, where B is NP-complete
- Assume to have access an efficient (polynomial-time) “black-box” (an *oracle*) $\mathcal{O}_A$ that solves A
- Show that there is a polynomial-time algorithm that interacts with $\mathcal{O}_A$ and solves B



Reductions

Think about (Cook) reductions in complexity theory:

- Let A and B be two decision problems, where B is NP-complete
- Assume to have access an efficient (polynomial-time) "black-box" (an *oracle*) $\mathcal{O}_A$ that solves A
- Show that there is a polynomial-time algorithm that interacts with $\mathcal{O}_A$ and solves B
- If A is solvable in polynomial-time then B is solvable in polynomial-time

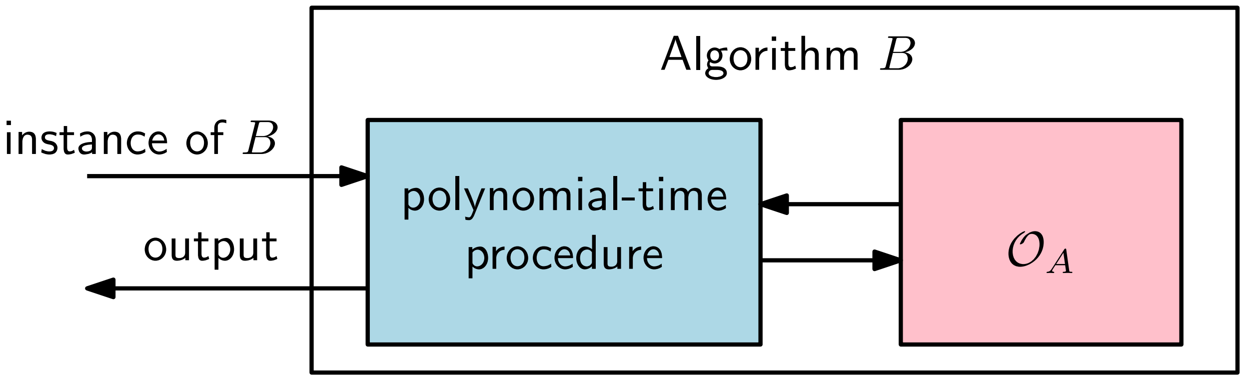


Reductions

Think about (Cook) reductions in complexity theory:

- Let A and B be two decision problems, where B is NP-complete
- Assume to have access an efficient (polynomial-time) “black-box” (an *oracle*) $\mathcal{O}_A$ that solves A
- Show that there is a polynomial-time algorithm that interacts with $\mathcal{O}_A$ and solves B
- If A is solvable in polynomial-time then B is solvable in polynomial-time

$\implies$ assuming P $\neq$ NP, A is not solvable in polynomial time



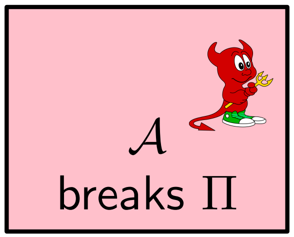
Security reductions

We want to show that Π is secure. We start from some problem X that is (conjectured to be) "hard to break" with a non-negligible advantage

Security reductions

We want to show that Π is secure. We start from some problem X that is (conjectured to be) "hard to break" with a non-negligible advantage

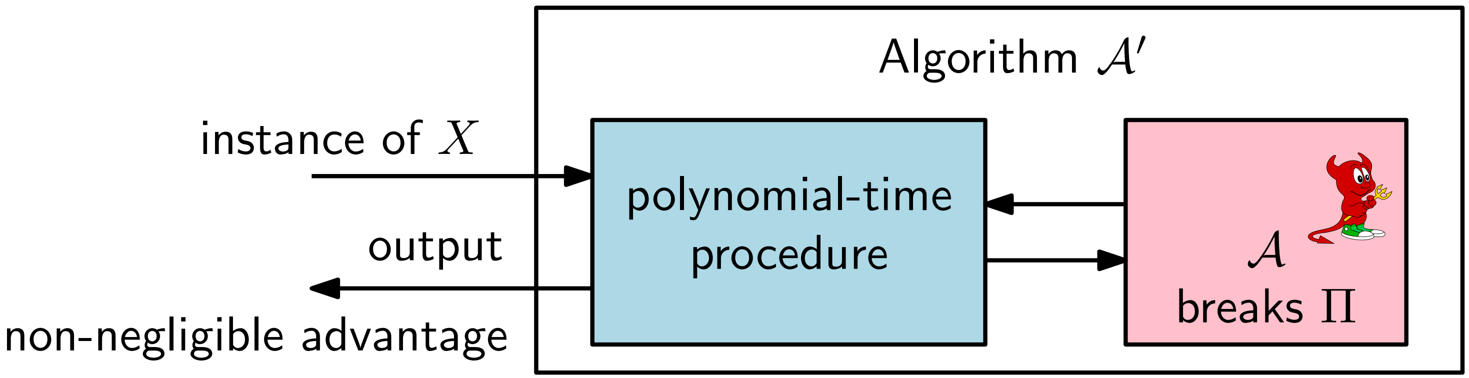
- Assume that there is some polynomial-time adversary $\mathcal{A}$ that breaks Π
  i.e., $\mathcal{A}$ "wins" the $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$ with non-negligible advantage $\varepsilon(n)$



Security reductions

We want to show that Π is secure. We start from some problem X that is (conjectured to be) "hard to break" with a non-negligible advantage

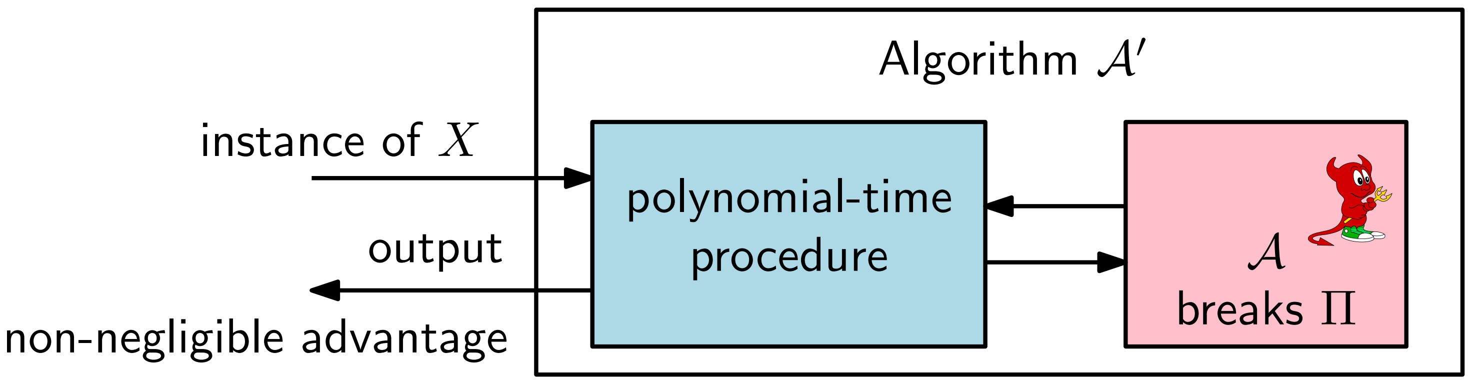
- Assume that there is some polynomial-time adversary $\mathcal{A}$ that breaks Π i.e., $\mathcal{A}$ "wins" the $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$ with non-negligible advantage $\varepsilon(n)$
- Use $\mathcal{A}$ as a "black box" in a polynomial-time algorithm $\mathcal{A}'$ that interacts with $\mathcal{A}$ and "breaks" X with non-negligible advantage (e.g., advantage at least $\frac{\varepsilon(n)}{p(n)}$, for some polynomial p)



Security reductions

We want to show that Π is secure. We start from some problem X that is (conjectured to be) "hard to break" with a non-negligible advantage

- Assume that there is some polynomial-time adversary $\mathcal{A}$ that breaks Π
  i.e., $\mathcal{A}$ "wins" the $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$ with non-negligible advantage $\varepsilon(n)$
- Use $\mathcal{A}$ as a "black box" in a polynomial-time algorithm $\mathcal{A}'$ that interacts with $\mathcal{A}$ and "breaks" X with non-negligible advantage (e.g., advantage at least $\frac{\varepsilon(n)}{p(n)}$, for some polynomial p)
- Since X cannot be broken with non-negligible advantage, no $\mathcal{A}$ exists

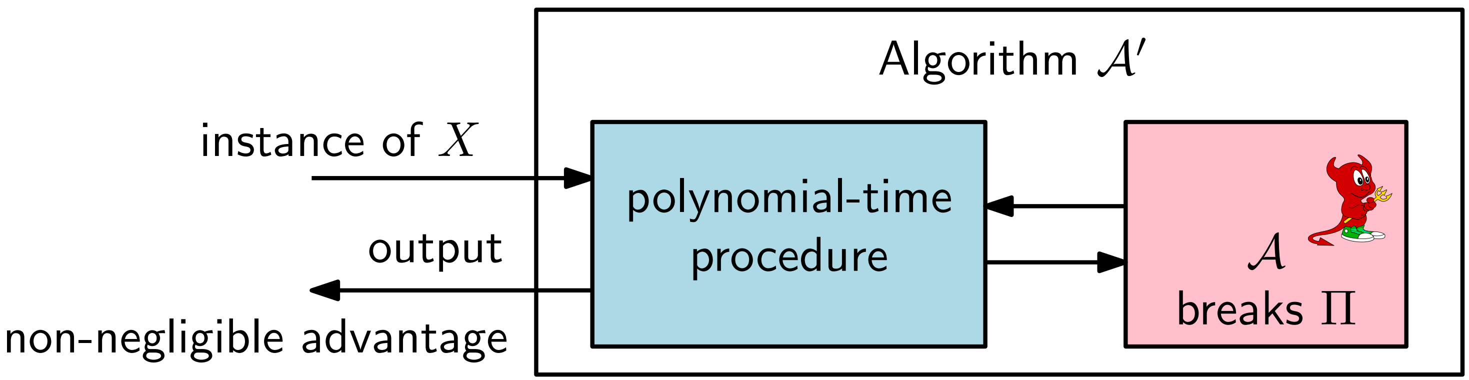


Security reductions

We want to show that Π is secure. We start from some problem X that is (conjectured to be) "hard to break" with a non-negligible advantage

- Assume that there is some polynomial-time adversary $\mathcal{A}$ that breaks Π
  i.e., $\mathcal{A}$ "wins" the $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$ with non-negligible advantage $\varepsilon(n)$
- Use $\mathcal{A}$ as a "black box" in a polynomial-time algorithm $\mathcal{A}'$ that interacts with $\mathcal{A}$ and "breaks" X with non-negligible advantage (e.g., advantage at least $\frac{\varepsilon(n)}{p(n)}$, for some polynomial p)
- Since X cannot be broken with non-negligible advantage, no $\mathcal{A}$ exists

  $\implies$ all poly-time adversaries for Π have negligible advantage (Π is secure)



Roadmap of our reduction

In our case, the problem X is that of telling apart the output of a PRG G from a random string

Roadmap of our reduction

In our case, the problem X is that of telling apart the output of a PRG G from a random string

- Assume that there is a polynomial-time adversary $\mathcal{A}$ that "breaks" pseudo OTP with non-negligible advantage

Roadmap of our reduction

In our case, the problem X is that of telling apart the output of a PRG G from a random string

- Assume that there is a polynomial-time adversary $\mathcal{A}$ that "breaks" pseudo OTP with non-negligible advantage
- Use $\mathcal{A}$ to build a polynomial-time distinguisher D for G

Roadmap of our reduction

In our case, the problem X is that of telling apart the output of a PRG G from a random string

- Assume that there is a polynomial-time adversary $\mathcal{A}$ that "breaks" pseudo OTP with non-negligible advantage
- Use $\mathcal{A}$ to build a polynomial-time distinguisher D for G
- Since G is a PRG, no such D can exist

Roadmap of our reduction

In our case, the problem X is that of telling apart the output of a PRG G from a random string

- Assume that there is a polynomial-time adversary $\mathcal{A}$ that "breaks" pseudo OTP with non-negligible advantage
- Use $\mathcal{A}$ to build a polynomial-time distinguisher D for G
- Since G is a PRG, no such D can exist

$\implies$ no such adversary $\mathcal{A}$ exists

Roadmap of our reduction

In our case, the problem X is that of telling apart the output of a PRG G from a random string

- Assume that there is a polynomial-time adversary $\mathcal{A}$ that "breaks" pseudo OTP with non-negligible advantage

- Use $\mathcal{A}$ to build a polynomial-time distinguisher D for G

- Since G is a PRG, no such D can exist

  $\implies$ no such adversary $\mathcal{A}$ exists

  $\implies$ pseudo OTP is secure

The actual reduction

Theorem: If G is a pseudorandom generator with expansion factor $\ell(n)$, then pseudo OTP is an EAV-secure, fixed-length private-key encryption scheme for messages of length $\ell(n)$.

Proof:

Let Π denote the pseudo-OTP scheme, and let $\widetilde{\Pi}$ be the "real" OTP scheme

The actual reduction

Theorem: If G is a pseudorandom generator with expansion factor $\ell(n)$, then pseudo OTP is an EAV-secure, fixed-length private-key encryption scheme for messages of length $\ell(n)$.

Proof:

Let Π denote the pseudo-OTP scheme, and let $\widetilde{\Pi}$ be the "real" OTP scheme

Assume that there is a polynomial-time adversary $\mathcal{A}$ such that $\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)] = \frac{1}{2} + \varepsilon(n)$ for a non-negligible $\varepsilon(n)$

The actual reduction

Theorem: If G is a pseudorandom generator with expansion factor $\ell(n)$, then pseudo OTP is an EAV-secure, fixed-length private-key encryption scheme for messages of length $\ell(n)$.

Proof:

Let Π denote the pseudo-OTP scheme, and let $\widetilde{\Pi}$ be the "real" OTP scheme

Assume that there is a polynomial-time adversary $\mathcal{A}$ such that $\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)] = \frac{1}{2} + \varepsilon(n)$ for a non-negligible $\varepsilon(n)$

Distinguisher $\mathcal{D}(w)$:

- Get the two messages m_0, m_1 from $\mathcal{A}$
- Pick b u.a.r. in $\{0, 1\}$ and let $c = m_b \oplus w$
- Send c to $\mathcal{A}$ and obtain a guess $b' \in \{0, 1\}$
- Output 1 if $b' = b$ and 0 otherwise

The actual reduction

Theorem: If G is a pseudorandom generator with expansion factor $\ell(n)$, then pseudo OTP is an EAV-secure, fixed-length private-key encryption scheme for messages of length $\ell(n)$.

Proof:

Let Π denote the pseudo-OTP scheme, and let $\widetilde{\Pi}$ be the "real" OTP scheme

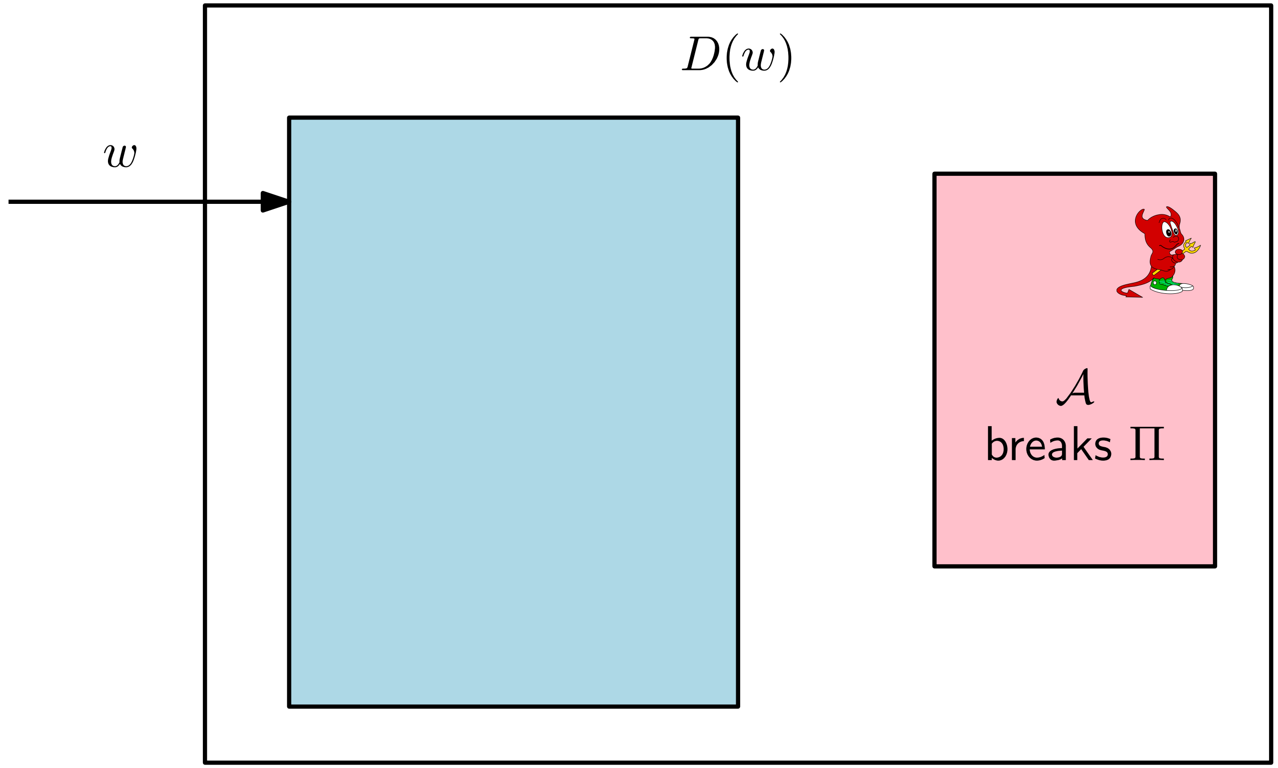
Assume that there is a polynomial-time adversary $\mathcal{A}$ such that $\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)] = \frac{1}{2} + \varepsilon(n)$ for a non-negligible $\varepsilon(n)$

Distinguisher $\mathcal{D}(w)$:

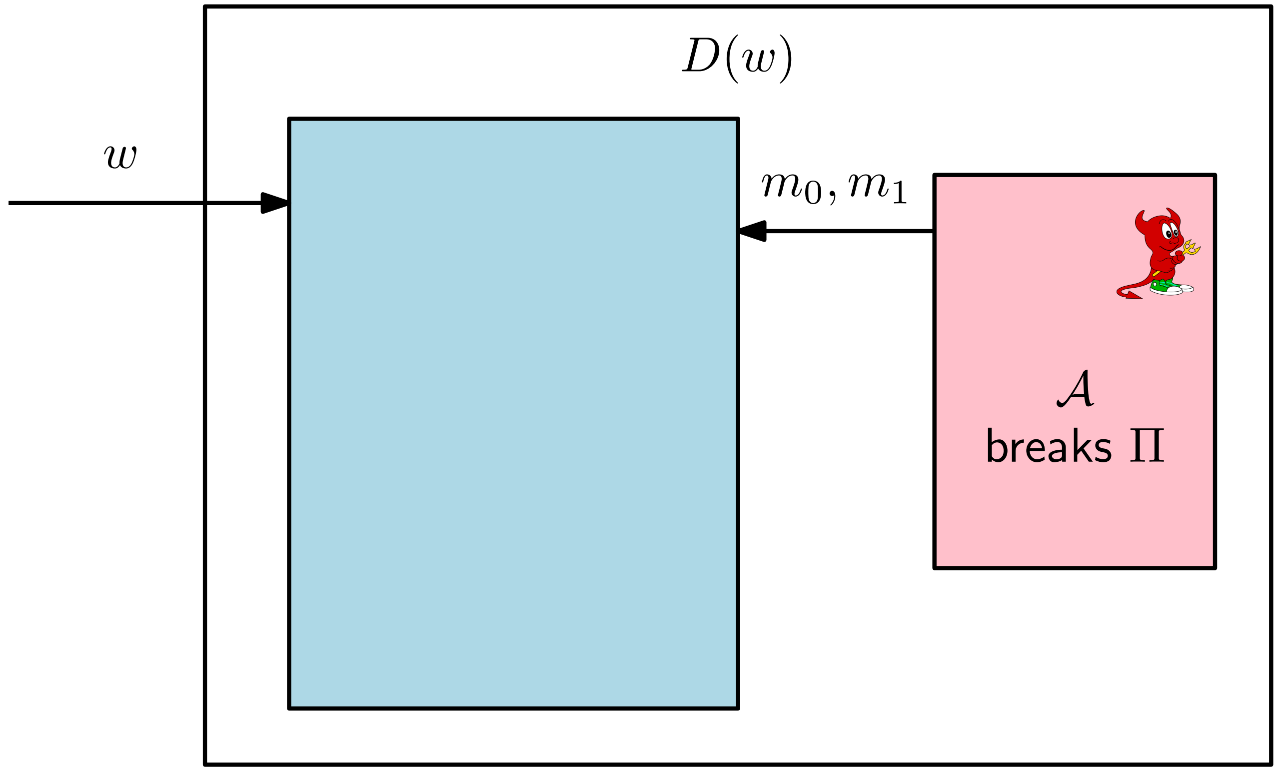
- Get the two messages m_0, m_1 from $\mathcal{A}$
- Pick b u.a.r. in $\{0, 1\}$ and let $c = m_b \oplus w$
- Send c to $\mathcal{A}$ and obtain a guess $b' \in \{0, 1\}$
- Output 1 if $b' = b$ and 0 otherwise

We need to bound $\left| \Pr[D(\, G(s)\,) = 1] - \Pr[D(r) = 1] \right|$

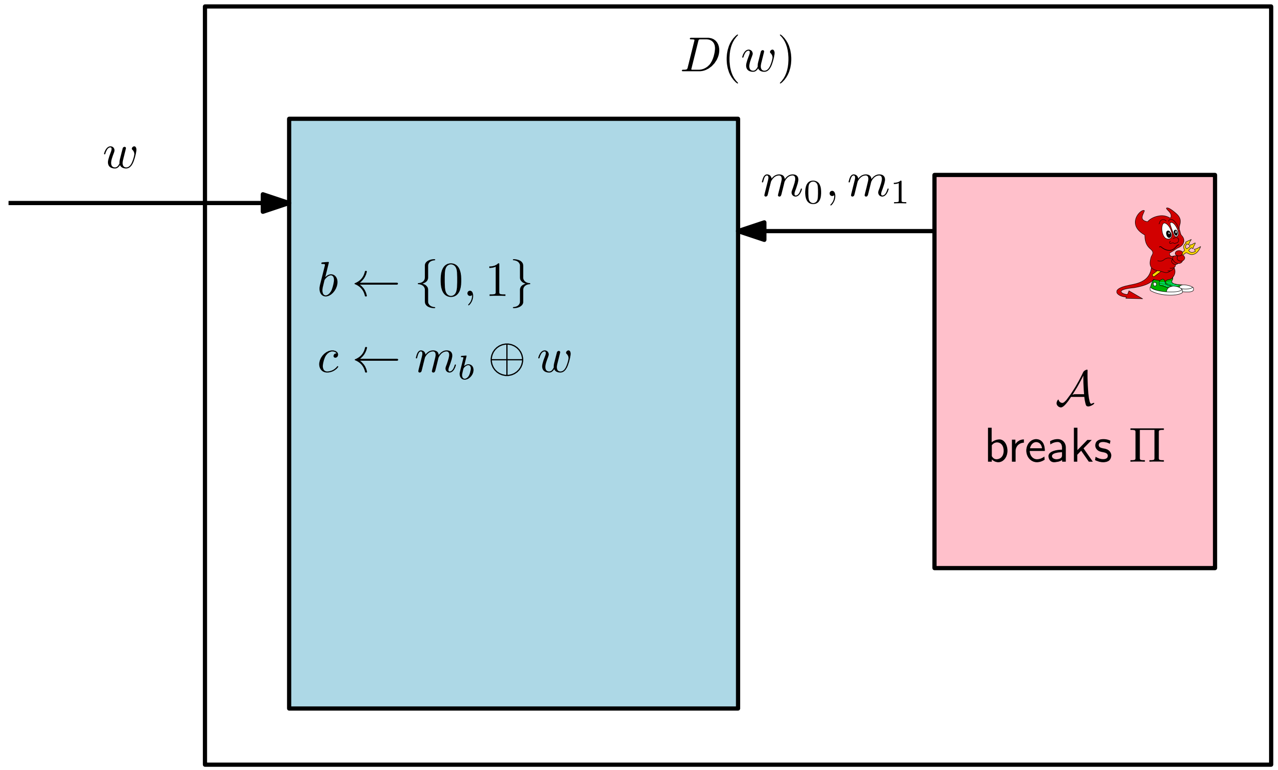
The actual reduction



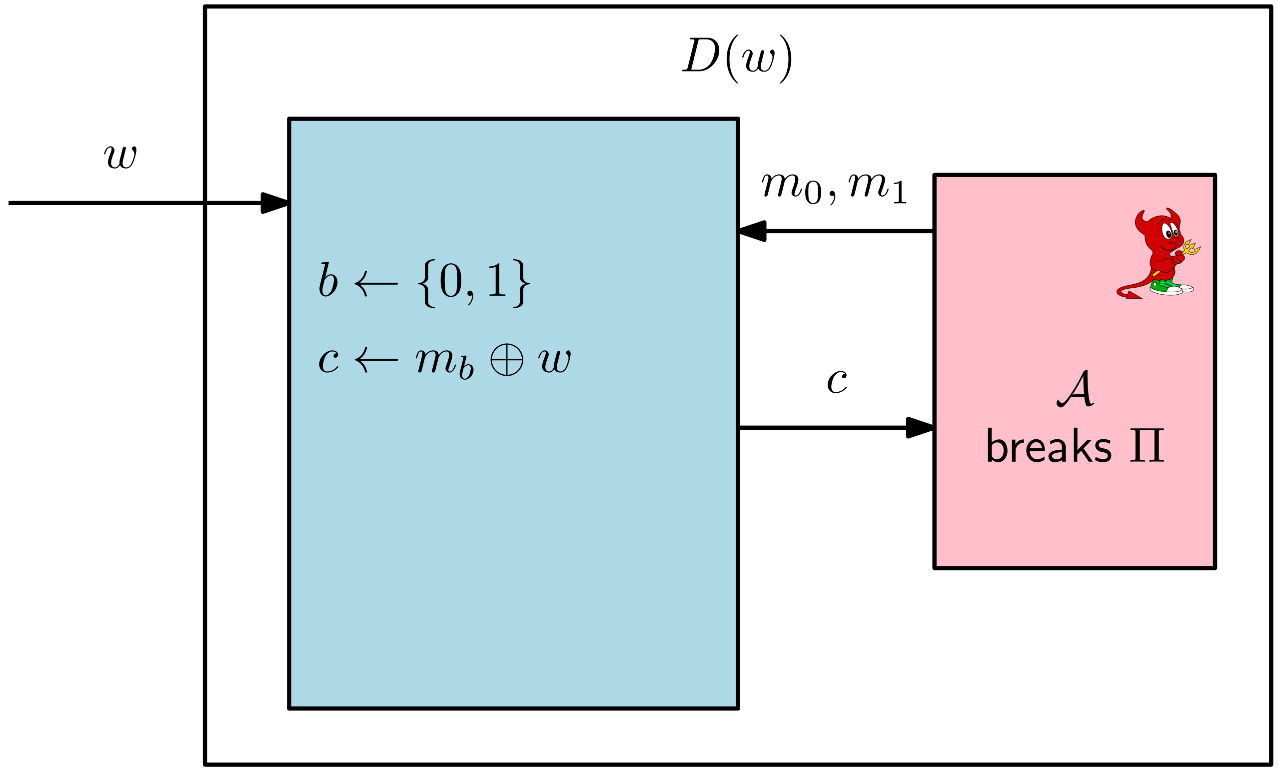
The actual reduction



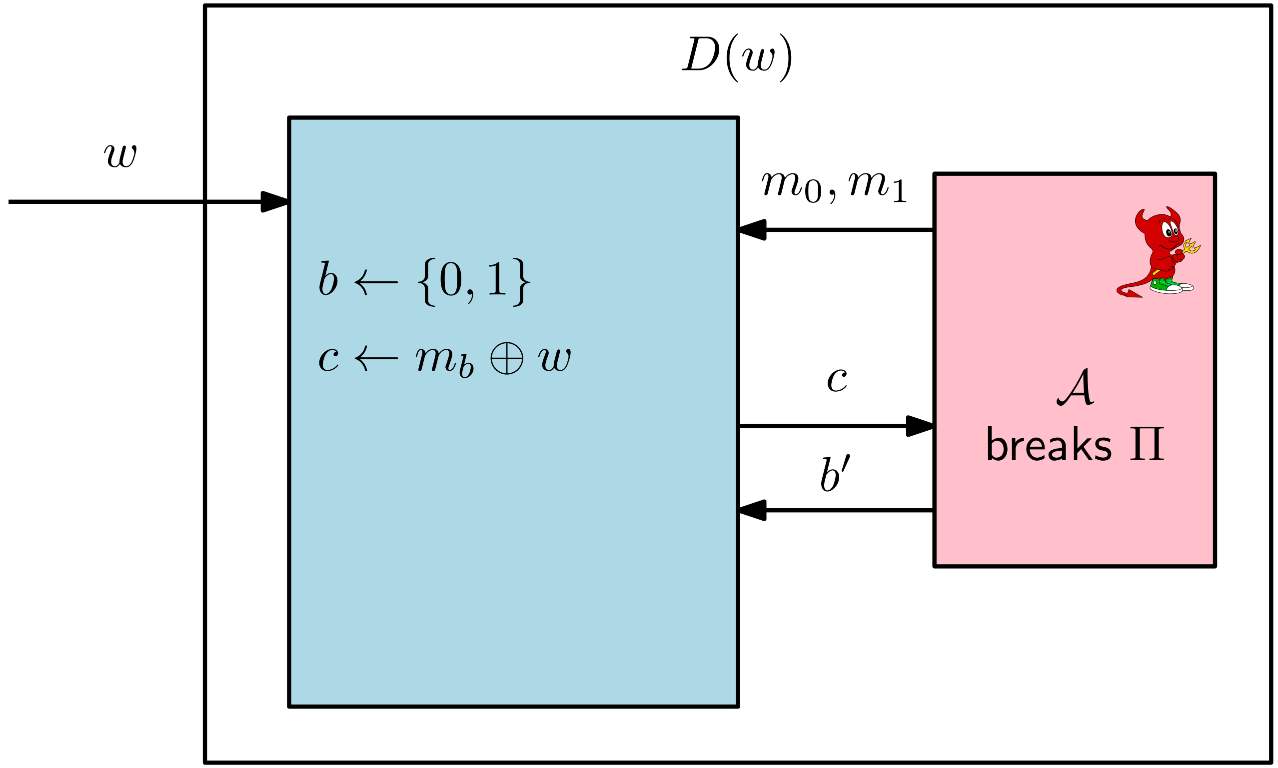
The actual reduction



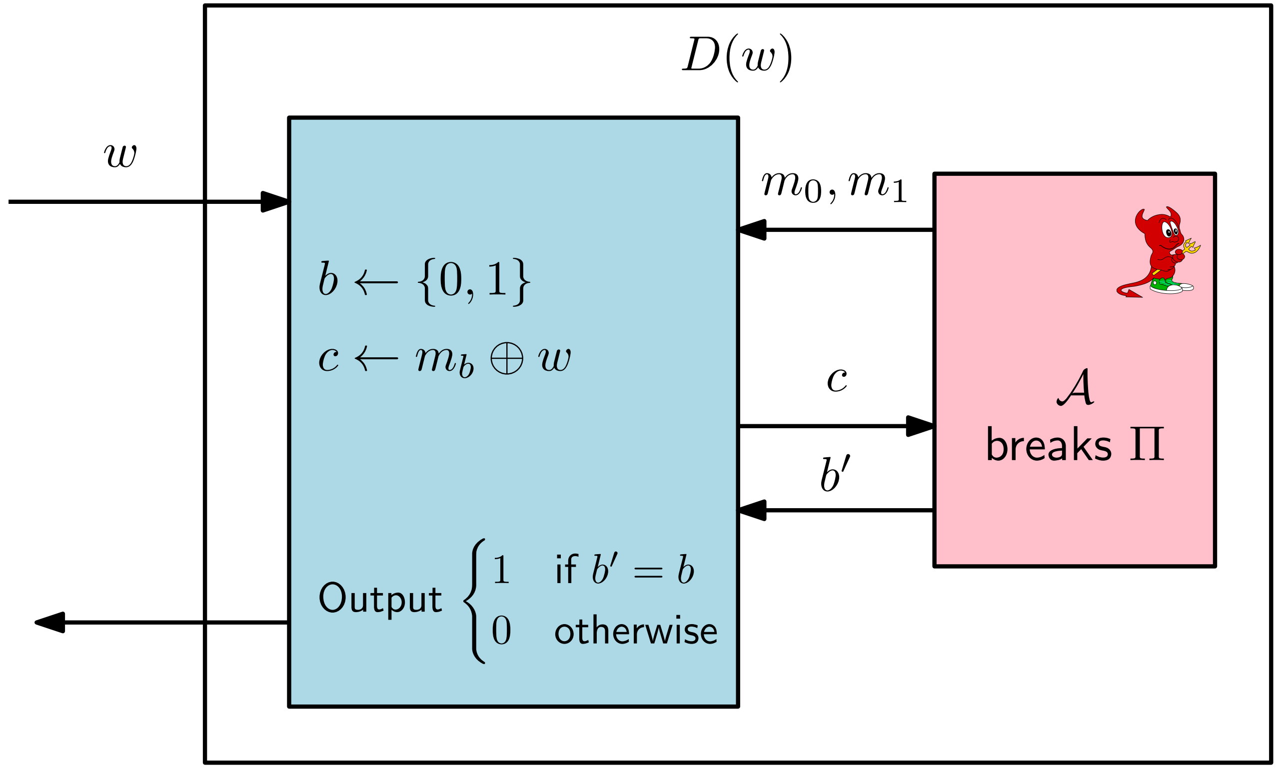
The actual reduction



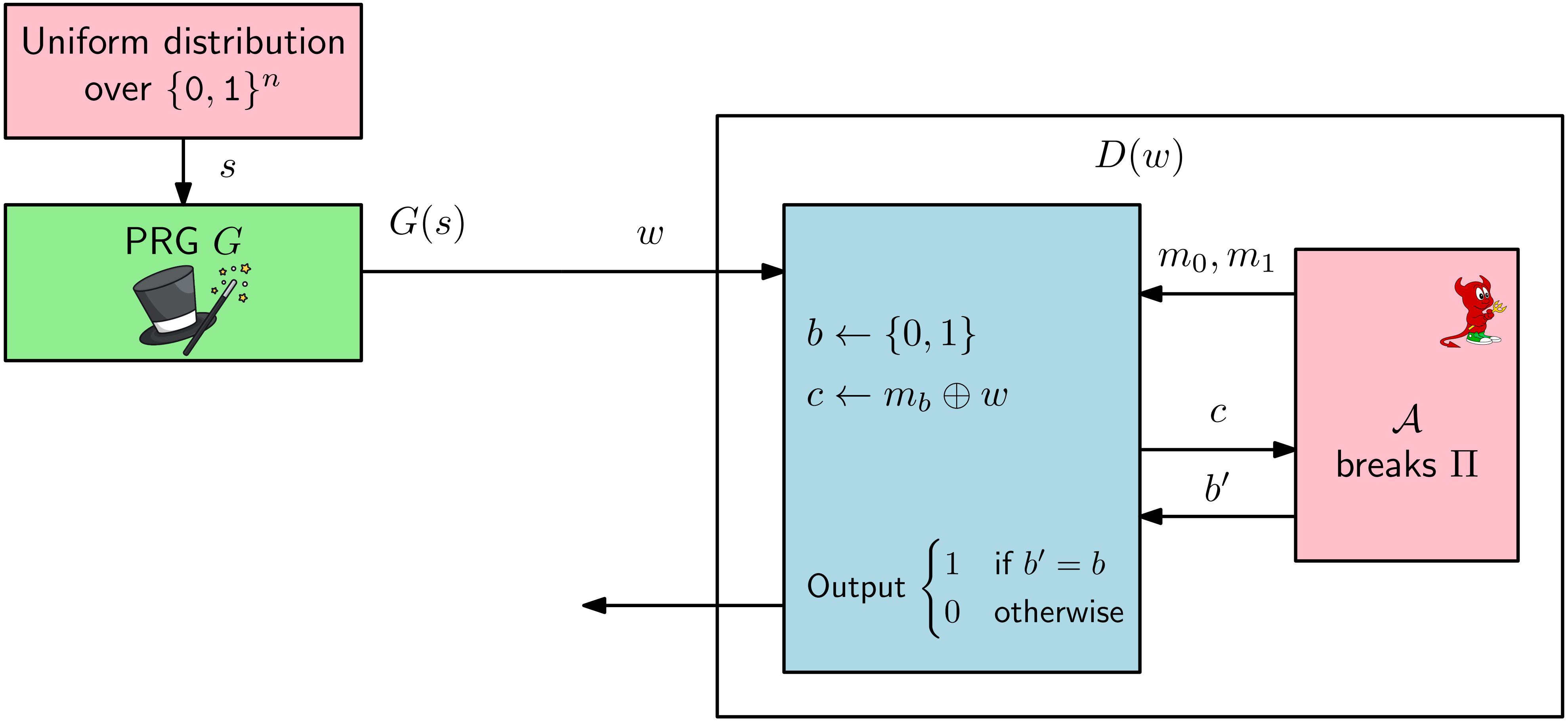
The actual reduction



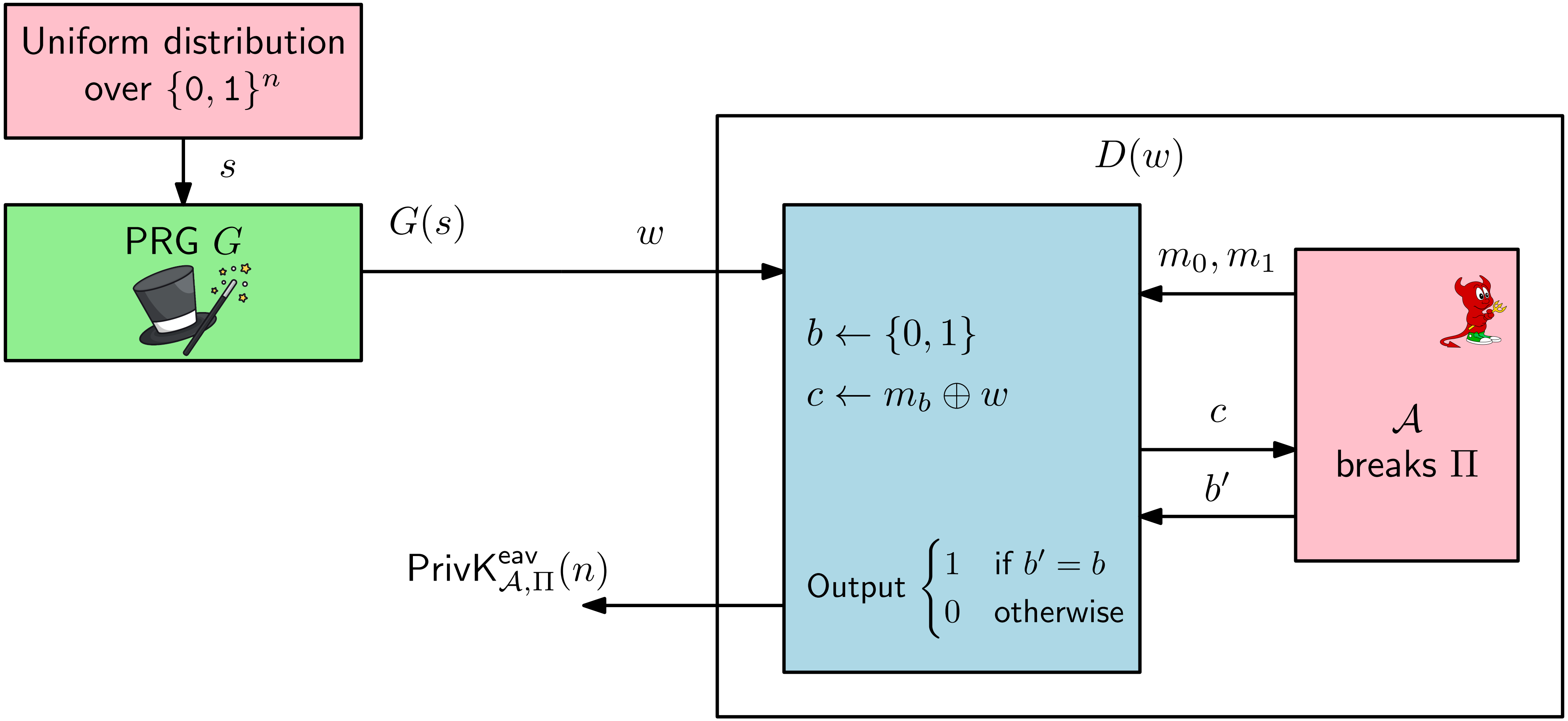
The actual reduction



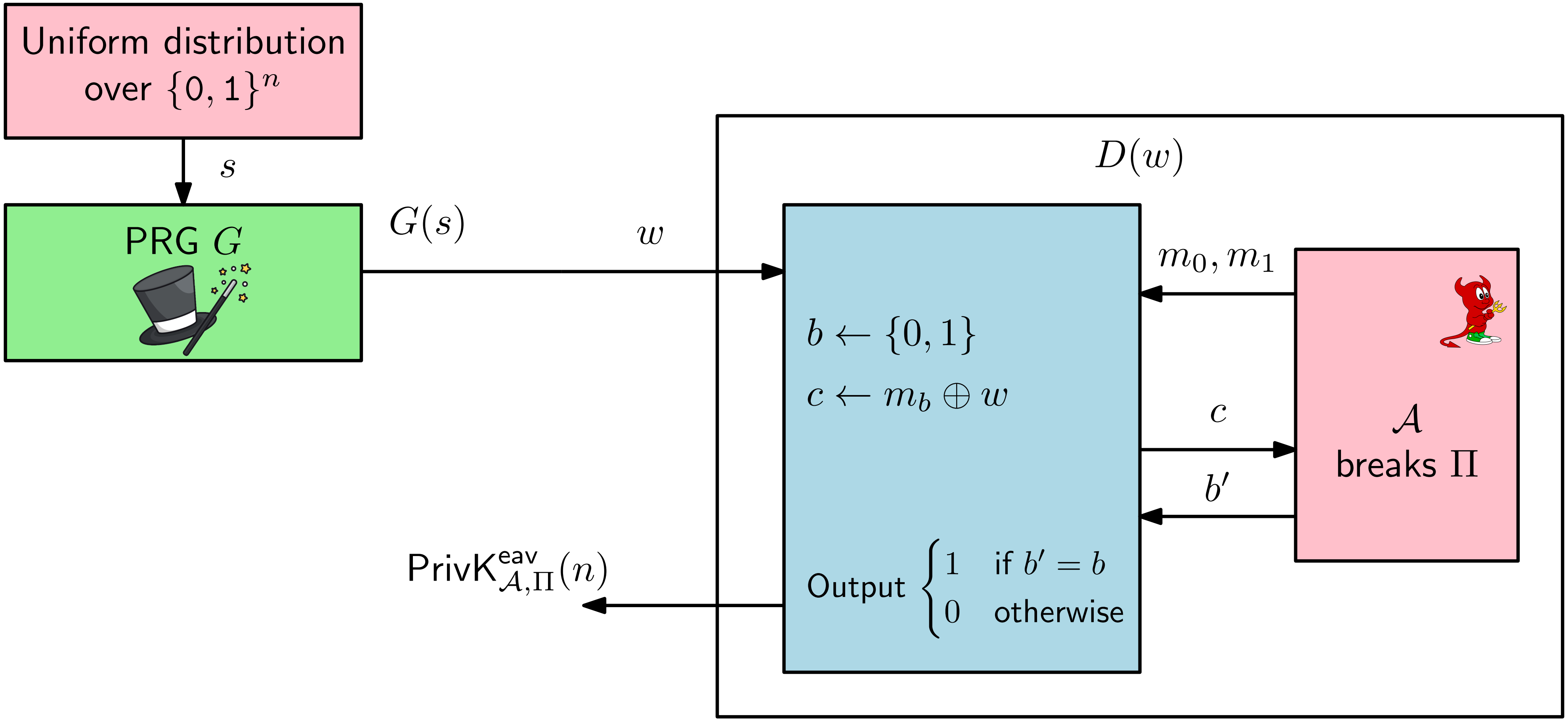
The actual reduction



The actual reduction

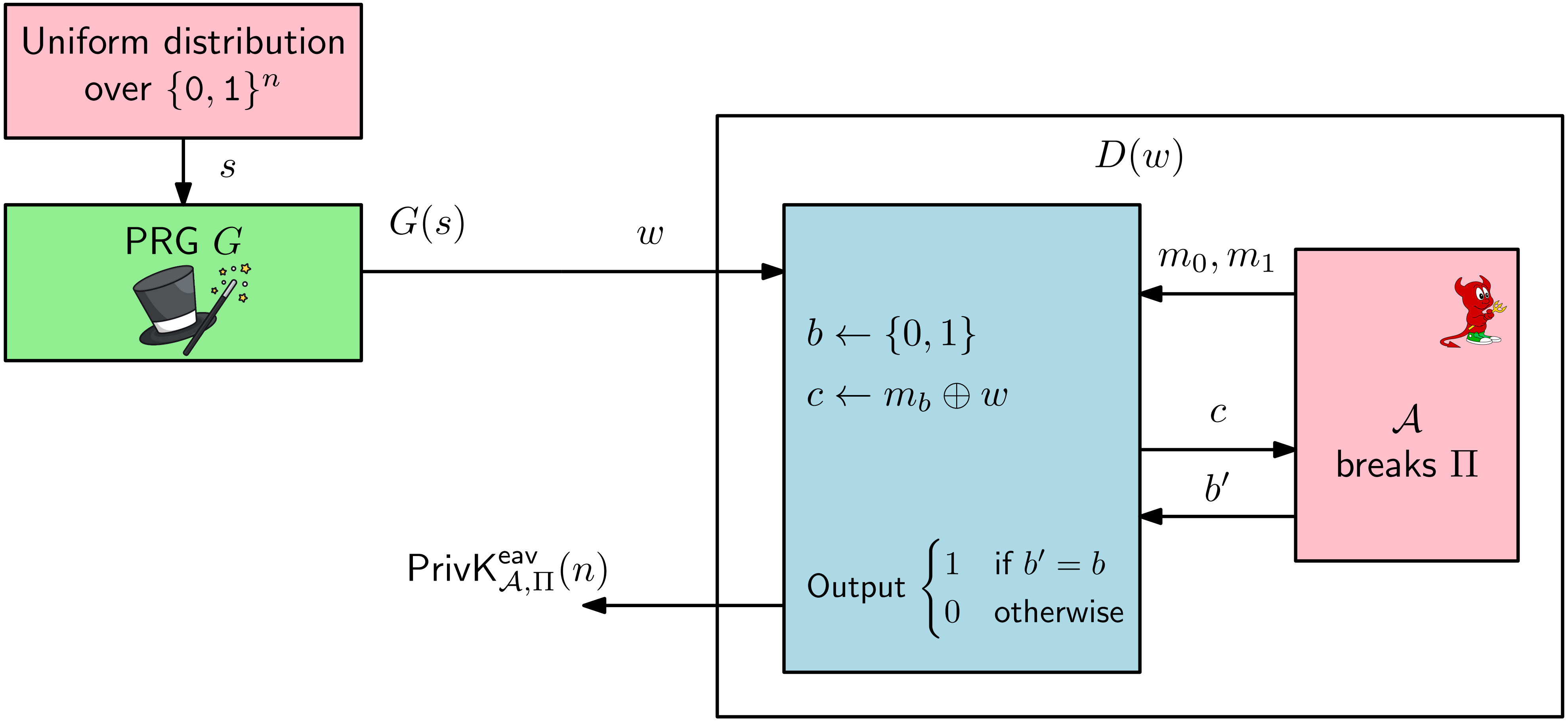


The actual reduction



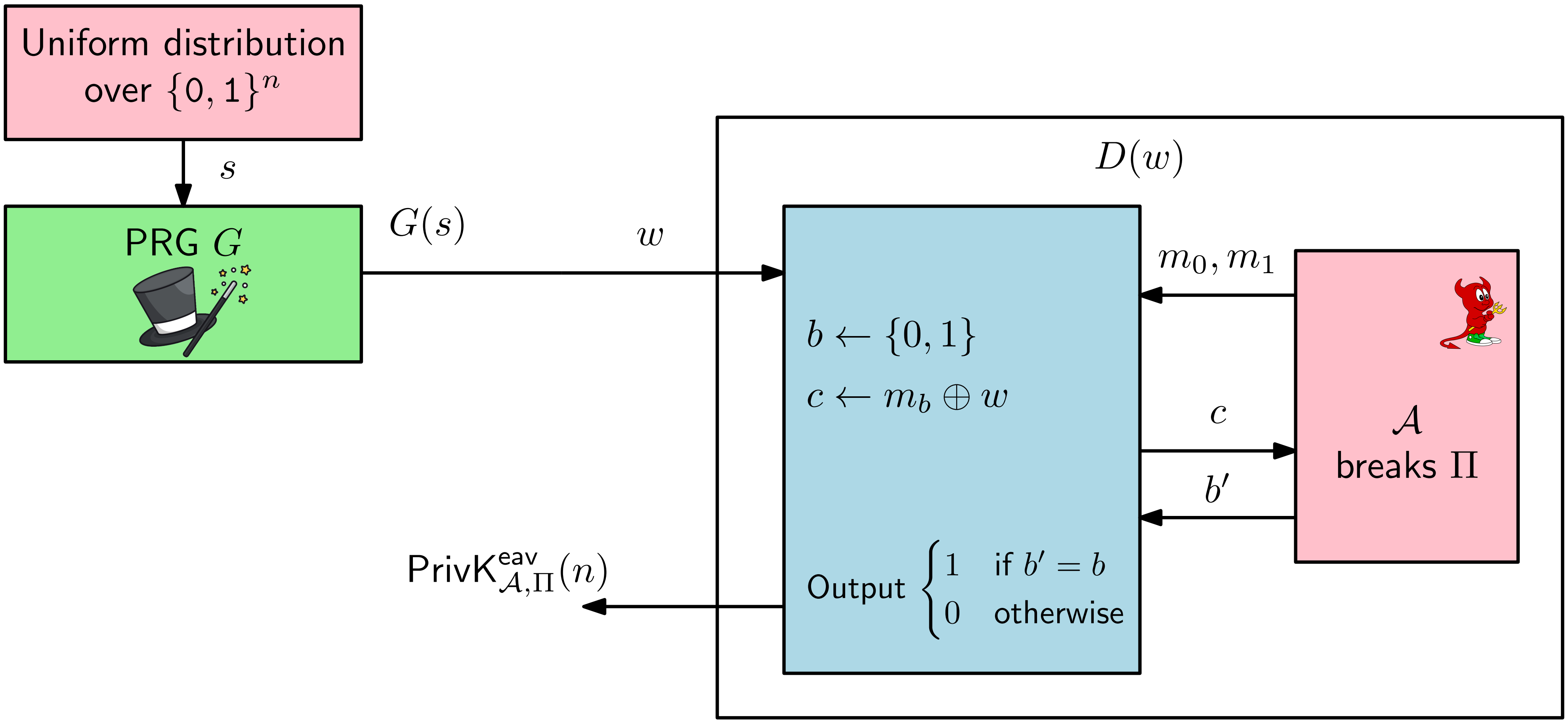
$$\Pr[D(G(s)) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1]$$

The actual reduction



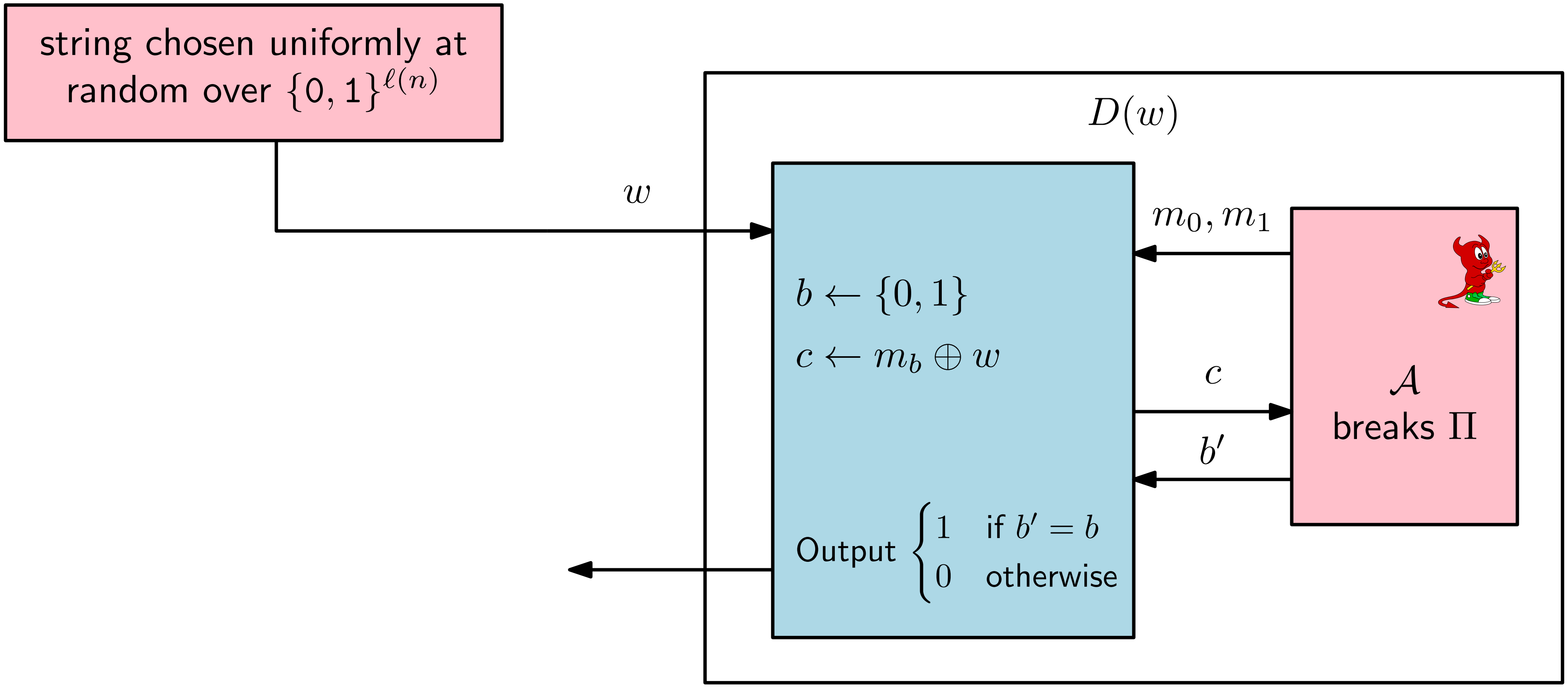
$$\Pr[D(G(s)) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] = \frac{1}{2} + \varepsilon(n)$$

The actual reduction



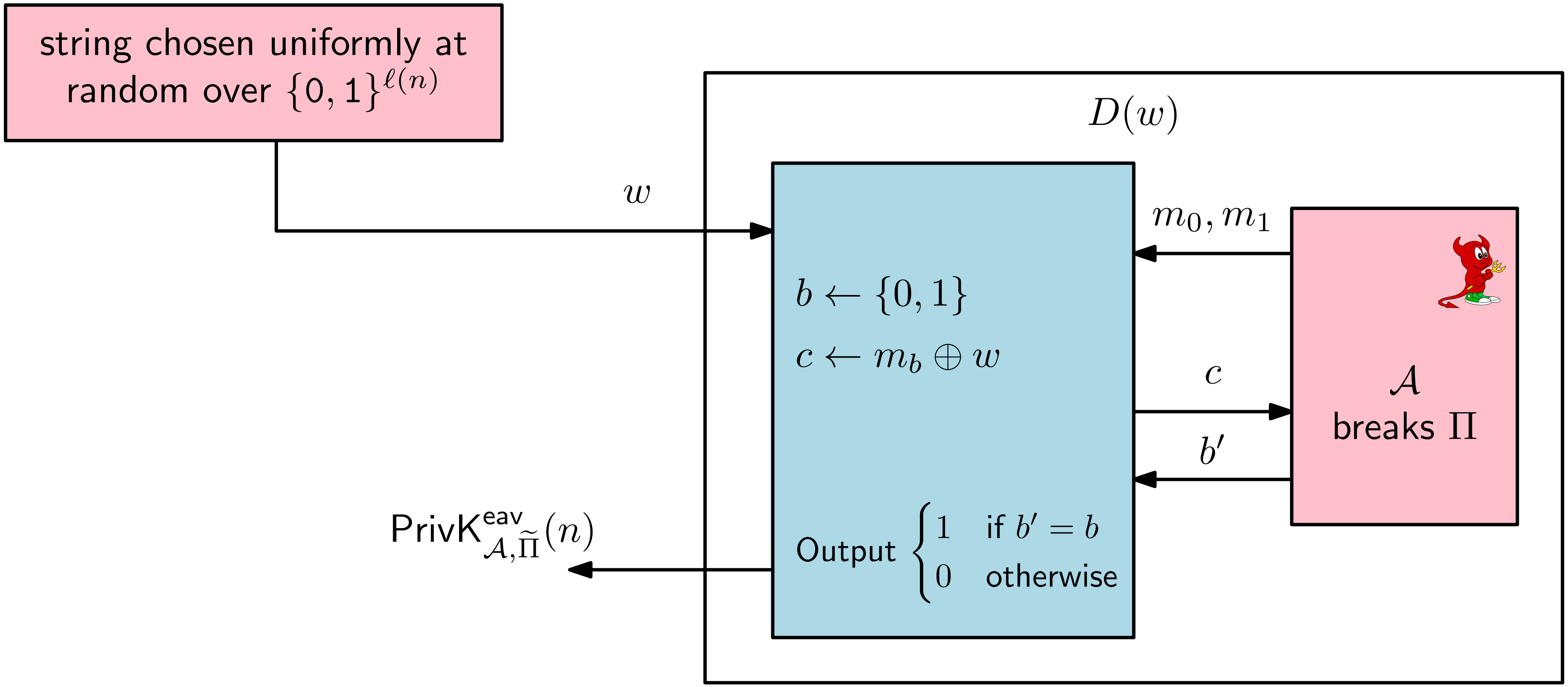
$\Pr[D(G(s)) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] \; = \frac{1}{2} + \varepsilon(n)$ ← non-negligible

The actual reduction



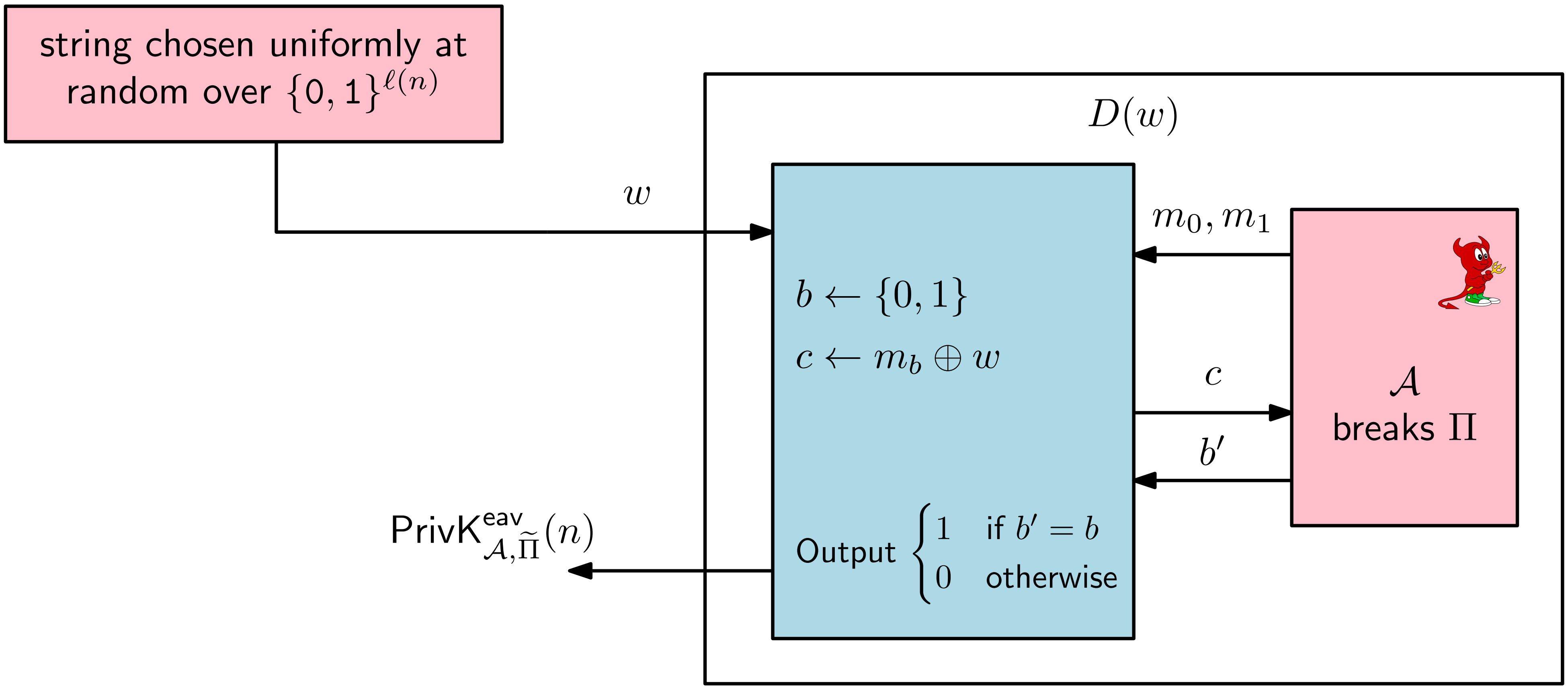
$\Pr[D(G(s)) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] = \frac{1}{2} + \varepsilon(n)$ ← non-negligible

The actual reduction



$\Pr[D(G(s)) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] = \frac{1}{2} + \varepsilon(n)$ ← non-negligible

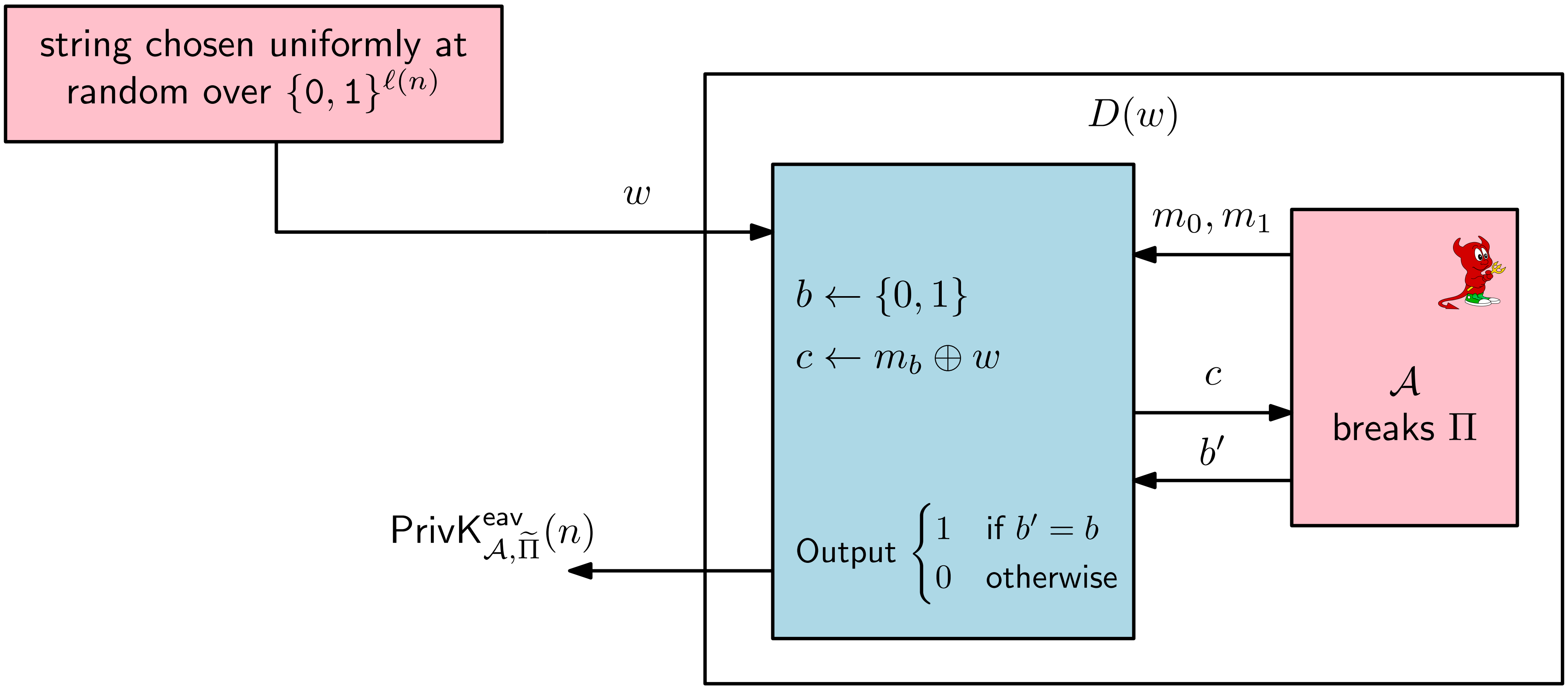
The actual reduction



$\Pr[D(G(s)) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] = \frac{1}{2} + \varepsilon(n)$ ← non-negligible

$\Pr[D(r) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1]$

The actual reduction



$\Pr[D(G(s)) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] \;=\; \frac{1}{2} + \varepsilon(n)$ ← non-negligible

$\Pr[D(r) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \;=\; \frac{1}{2}$ ($\widetilde{\Pi}$ is perfectly secret)

The actual reduction

We need to bound $\left| \Pr[D(\, G(s)\,) = 1] - \Pr[D(r) = 1] \right|$

$\Pr[D(G(s)) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] \; = \frac{1}{2} + \varepsilon(n)$ ← non-negligible

$\Pr[D(r) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \; = \frac{1}{2}$ ($\widetilde{\Pi}$ is perfectly secret)

The actual reduction

We need to bound $\left| \Pr[D(\,G(s)\,) = 1] - \Pr[D(r) = 1] \right|$

$\Pr[D(G(s)) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] \;= \frac{1}{2} + \varepsilon(n)$ ← non-negligible

$\Pr[D(r) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \;= \frac{1}{2}$ ($\widetilde{\Pi}$ is perfectly secret)

$$\left| \Pr[D(\,G(s)\,) = 1] - \Pr[D(r) = 1] \right| = \left| \frac{1}{2} + \varepsilon(n) - \frac{1}{2} \right| = \left| \varepsilon(n) \right|$$ non-negligible!

□

Once again

- We have a perfectly secret encryption scheme (one-time pad)...
- . . . but it requires long keys
- This is inevitable if we insist on perfect secrecy (recall that, in a perfectly secret scheme, $|\mathcal{K}| \geq |\mathcal{M}|$)
- We have a security definition that allows for short keys and works against adversaries with polynomially bounded running times

We came up with a (conditionally) secure private-key encryption scheme with keys shorter than the messages according to this new definition

Once again

- We have a perfectly secret encryption scheme (one-time pad)...
- . . . but it requires long keys
- This is inevitable if we insist on perfect secrecy (recall that, in a perfectly secret scheme, $|\mathcal{K}| \geq |\mathcal{M}|$)
- We have a security definition that allows for short keys and works against adversaries with polynomially bounded running times

We came up with a (conditionally) secure private-key encryption scheme with keys shorter than the messages according to this new definition

Are we done yet?

Several issues remain...

We can now use keys of length n to encrypt messages of length $\ell(n) > n$

- What about messages of length $\ell(n) + 1$?

Several issues remain...

We can now use keys of length n to encrypt messages of length $\ell(n) > n$

- What about messages of length $\ell(n) + 1$?
- What about very long messages?

Several issues remain...

We can now use keys of length n to encrypt messages of length $\ell(n) > n$

- What about messages of length $\ell(n) + 1$?
- What about very long messages?
- What about sending multiple messages?

Several issues remain...

We can now use keys of length n to encrypt messages of length $\ell(n) > n$

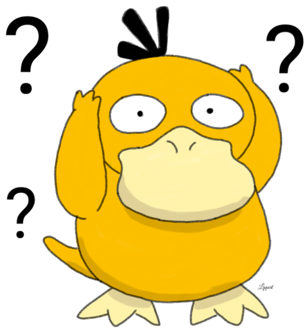
- What about messages of length $\ell(n) + 1$?
- What about very long messages?
- What about sending multiple messages?
- What about the malleability of OTP (and pseudo-OTP)?

Several issues remain...

We can now use keys of length n to encrypt messages of length $\ell(n) > n$

- What about messages of length $\ell(n) + 1$?
- What about very long messages?
- What about sending multiple messages?
- What about the malleability of OTP (and pseudo-OTP)?
- How do we build the PRG G in practice?

  we don't even know if PRGs exist...



Several issues remain...

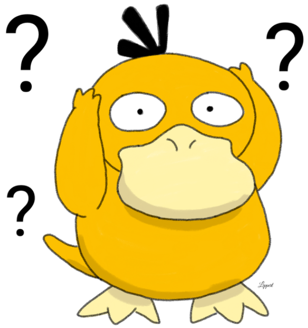
We can now use keys of length n to encrypt messages of length $\ell(n) > n$

- What about messages of length $\ell(n) + 1$?
- What about very long messages?
- What about sending multiple messages?

} CPA security, psedorandom functions, pseudorandom permutations, block ciphers

- What about the malleability of OTP (and pseudo-OTP)?
- How do we build the PRG G in practice?

  we don't even know if PRGs exist...



Several issues remain...

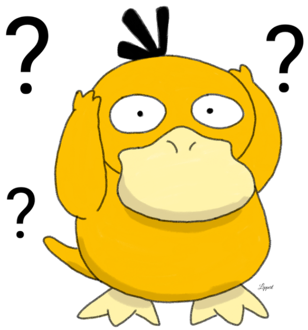
We can now use keys of length n to encrypt messages of length $\ell(n) > n$

- What about messages of length $\ell(n) + 1$?
- What about very long messages?
- What about sending multiple messages?
- What about the malleability of OTP (and pseudo-OTP)?
- How do we build the PRG G in practice?

  we don't even know if PRGs exist...

CPA security, psedorandom functions, pseudorandom permutations, block ciphers

message authentication codes, authenticated encryption



Several issues remain...

We can now use keys of length n to encrypt messages of length $\ell(n) > n$

- What about messages of length $\ell(n) + 1$?
- What about very long messages?
- What about sending multiple messages?

CPA security, psedorandom functions, pseudorandom permutations, block ciphers

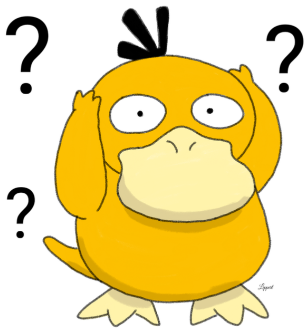
- What about the malleability of OTP (and pseudo-OTP)?

message authentication codes, authenticated encryption

- How do we build the PRG G in practice?

stream ciphers

we don't even know if PRGs exist...



Several issues remain...

We can now use keys of length n to encrypt messages of length $\ell(n) > n$

- What about messages of length $\ell(n) + 1$?
- What about very long messages?
- What about sending multiple messages?

} CPA security, psedorandom functions, pseudorandom permutations, block ciphers

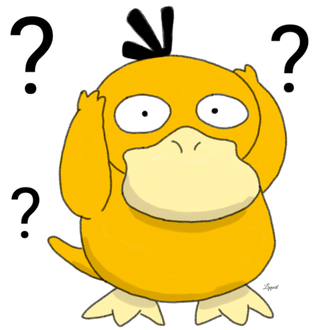
- What about the malleability of OTP (and pseudo-OTP)?

message authentication codes, authenticated encryption

- How do we build the PRG G in practice?

stream ciphers

we don't even know if PRGs exist...



Multiple messages

To handle the case in which multiple messages are encrypted, we need to update our security definition accordingly

- The adversary provides two **lists** $\vec{M}_0 = \langle m_{0,1}, m_{0,2}, \ldots, m_{0,t} \rangle, \vec{M}_1 = \langle m_{1,1}, m_{1,2}, \ldots, m_{1,t} \rangle$ of messages with $|m_{0,i}| = |m_{1,i}|$

Multiple messages

To handle the case in which multiple messages are encrypted, we need to update our security definition accordingly

- The adversary provides two **lists** $\vec{M}_0 = \langle m_{0,1}, m_{0,2}, \ldots, m_{0,t} \rangle, \vec{M}_1 = \langle m_{1,1}, m_{1,2}, \ldots, m_{1,t} \rangle$ of messages with $|m_{0,i}| = |m_{1,i}|$
- The two lists must have the same number t of messages (chosen by the adversary)

Multiple messages

To handle the case in which multiple messages are encrypted, we need to update our security definition accordingly

- The adversary provides two **lists** $\vec{M}_0 = \langle m_{0,1}, m_{0,2}, \ldots, m_{0,t} \rangle, \vec{M}_1 = \langle m_{1,1}, m_{1,2}, \ldots, m_{1,t} \rangle$ of messages with $|m_{0,i}| = |m_{1,i}|$
- The two lists must have the same number t of messages (chosen by the adversary)
- b is chosen u.a.r. from $\{0, 1\}$

Multiple messages

To handle the case in which multiple messages are encrypted, we need to update our security definition accordingly

- The adversary provides two **lists** $\vec{M}_0 = \langle m_{0,1}, m_{0,2}, \ldots, m_{0,t} \rangle, \vec{M}_1 = \langle m_{1,1}, m_{1,2}, \ldots, m_{1,t} \rangle$ of messages with $|m_{0,i}| = |m_{1,i}|$

- The two lists must have the same number t of messages (chosen by the adversary)

- b is chosen u.a.r. from $\{0, 1\}$

- All messages in $\vec{M}_b$ are encrypted (using the same key) to produce a list $\vec{C} = \langle c_1, c_2, \ldots, c_t \rangle$ of chipertexts

Multiple messages

To handle the case in which multiple messages are encrypted, we need to update our security definition accordingly

- The adversary provides two **lists** $\vec{M}_0 = \langle m_{0,1}, m_{0,2}, \ldots, m_{0,t} \rangle, \vec{M}_1 = \langle m_{1,1}, m_{1,2}, \ldots, m_{1,t} \rangle$ of messages with $|m_{0,i}| = |m_{1,i}|$

- The two lists must have the same number t of messages (chosen by the adversary)

- b is chosen u.a.r. from $\{0, 1\}$

- All messages in $\vec{M}_b$ are encrypted (using the same key) to produce a list $\vec{C} = \langle c_1, c_2, \ldots, c_t \rangle$ of chipertexts

- The list $\vec{C}$ is given to the adversary

Multiple messages

To handle the case in which multiple messages are encrypted, we need to update our security definition accordingly

- The adversary provides two **lists** $\vec{M}_0 = \langle m_{0,1}, m_{0,2}, \ldots, m_{0,t} \rangle, \vec{M}_1 = \langle m_{1,1}, m_{1,2}, \ldots, m_{1,t} \rangle$ of messages with $|m_{0,i}| = |m_{1,i}|$
- The two lists must have the same number t of messages (chosen by the adversary)
- b is chosen u.a.r. from $\{0, 1\}$
- All messages in $\vec{M}_b$ are encrypted (using the same key) to produce a list $\vec{C} = \langle c_1, c_2, \ldots, c_t \rangle$ of chipertexts
- The list $\vec{C}$ is given to the adversary
- The adversary needs to provide a guess b' for the value of b

Multiple messages

To handle the case in which multiple messages are encrypted, we need to update our security definition accordingly

- The adversary provides two **lists** $\vec{M}_0 = \langle m_{0,1}, m_{0,2}, \ldots, m_{0,t} \rangle, \vec{M}_1 = \langle m_{1,1}, m_{1,2}, \ldots, m_{1,t} \rangle$ of messages with $|m_{0,i}| = |m_{1,i}|$
- The two lists must have the same number t of messages (chosen by the adversary)
- b is chosen u.a.r. from $\{0, 1\}$
- All messages in $\vec{M}_b$ are encrypted (using the same key) to produce a list $\vec{C} = \langle c_1, c_2, \ldots, c_t \rangle$ of chipertexts
- The list $\vec{C}$ is given to the adversary
- The adversary needs to provide a guess b' for the value of b

$$\text{PrivK}^{\text{mult}}_{\mathcal{A},\Pi}(n) = \begin{cases} 1 & \text{if } b' = b \\ 0 & \text{otherwise} \end{cases}$$

Multiple messages: security definition

Definition: *A private key encryption scheme* $\Pi = (Gen, Enc, Dec)$ *has* **indistinguishable multiple encryptions** *in the presence of an eavesdropper if, for every probabilistic polynomial-time adversary* $\mathcal{A}$*, there is a negligible function* ε *such that:*

$$\Pr[\mathit{PrivK}^{mult}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \varepsilon(n)$$

Multiple messages: security definition

Definition: *A private key encryption scheme* $\Pi = (Gen, Enc, Dec)$ *has* **indistinguishable multiple encryptions** *in the presence of an eavesdropper if, for every probabilistic polynomial-time adversary* $\mathcal{A}$*, there is a negligible function* ε *such that:*

$$\Pr[\mathit{PrivK}^{mult}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \varepsilon(n)$$

Observation: This is a stronger requirement than having indistinguishable encryptions in the presence of an eavesdropper

Multiple messages: security definition

Definition: *A private key encryption scheme* $\Pi = (Gen, Enc, Dec)$ *has* **indistinguishable multiple encryptions** *in the presence of an eavesdropper if, for every probabilistic polynomial-time adversary* $\mathcal{A}$*, there is a negligible function* ε *such that:*

$$\Pr[\mathit{PrivK}^{\mathit{mult}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \varepsilon(n)$$

Observation: This is a stronger requirement than having indistinguishable encryptions in the presence of an eavesdropper

The adversary is more powerful!

(it can simulate an adversary for the $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}$ experiment)

Multiple messages: security definition

Definition: *A private key encryption scheme* $\Pi = (Gen, Enc, Dec)$ *has* **indistinguishable multiple encryptions** *in the presence of an eavesdropper if, for every probabilistic polynomial-time adversary* $\mathcal{A}$*, there is a negligible function* ε *such that:*

$$\Pr[\mathit{PrivK}^{\mathit{mult}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \varepsilon(n)$$

Observation: This is a stronger requirement than having indistinguishable encryptions in the presence of an eavesdropper

The adversary is more powerful!

(it can simulate an adversary for the $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}$ experiment)

$$\Downarrow$$

If a scheme has **indistinguishable multiple encryptions** in the presence of an eavesdropper then it is also **EAV-secure**

Multiple message (in)security of OTP

Does OTP have indistinguishable multiple encryptions in the presence of an eavesdropper?

Multiple message (in)security of OTP

Does OTP have indistinguishable multiple encryptions in the presence of an eavesdropper?

Distinguisher $\mathcal{A}$:

- Output $\vec{M}_0 = \langle 0^\ell, 0^\ell \rangle$ and $\vec{M}_1 = \langle 0^\ell, 1^\ell \rangle$
- Upon receiving $\vec{C} = \langle c_1, c_2 \rangle$:
  - Output $b' = 0$ if $c_1 = c_2$
  - Otherwise output $b' = 1$

Advantage?

Multiple message (in)security of OTP

Does OTP have indistinguishable multiple encryptions in the presence of an eavesdropper?

Distinguisher $\mathcal{A}$:

- Output $\vec{M}_0 = \langle 0^\ell, 0^\ell \rangle$ and $\vec{M}_1 = \langle 0^\ell, 1^\ell \rangle$
- Upon receiving $\vec{C} = \langle c_1, c_2 \rangle$:
  - Output $b' = 0$ if $c_1 = c_2$
  - Otherwise output $b' = 1$

Advantage?

- If $b = 0$, then $c_0 = c_1 = \mathsf{Enc}_k(0^\ell) \implies \mathcal{A}$ guesses correctly with probability 1

Multiple message (in)security of OTP

Does OTP have indistinguishable multiple encryptions in the presence of an eavesdropper?

Distinguisher $\mathcal{A}$:

- Output $\vec{M}_0 = \langle 0^\ell, 0^\ell \rangle$ and $\vec{M}_1 = \langle 0^\ell, 1^\ell \rangle$
- Upon receiving $\vec{C} = \langle c_1, c_2 \rangle$:
  - Output $b' = 0$ if $c_1 = c_2$
  - Otherwise output $b' = 1$

Advantage?

- If $b = 0$, then $c_0 = c_1 = \mathsf{Enc}_k(0^\ell) \implies \mathcal{A}$ guesses correctly with probability 1
- If $b = 1$, then $c_0 = \mathsf{Enc}_k(0^\ell)$ and $c_1 = \mathsf{Enc}_k(1^\ell)$

Multiple message (in)security of OTP

Does OTP have indistinguishable multiple encryptions in the presence of an eavesdropper?

Distinguisher $\mathcal{A}$:

- Output $\vec{M}_0 = \langle 0^\ell, 0^\ell \rangle$ and $\vec{M}_1 = \langle 0^\ell, 1^\ell \rangle$
- Upon receiving $\vec{C} = \langle c_1, c_2 \rangle$:
  - Output $b' = 0$ if $c_1 = c_2$
  - Otherwise output $b' = 1$

Advantage?

- If $b = 0$, then $c_0 = c_1 = \mathsf{Enc}_k(0^\ell) \implies \mathcal{A}$ guesses correctly with probability 1
- If $b = 1$, then $c_0 = \mathsf{Enc}_k(0^\ell)$ and $c_1 = \mathsf{Enc}_k(1^\ell)$

  $\implies c_0 \neq c_1$ since otherwise either $\mathsf{Dec}_k(c_0) \neq m_0$ or $\mathsf{Dec}_k(c_1) \neq m_1$

Multiple message (in)security of OTP

Does OTP have indistinguishable multiple encryptions in the presence of an eavesdropper?

Distinguisher $\mathcal{A}$:

- Output $\vec{M}_0 = \langle 0^\ell, 0^\ell \rangle$ and $\vec{M}_1 = \langle 0^\ell, 1^\ell \rangle$
- Upon receiving $\vec{C} = \langle c_1, c_2 \rangle$:
  - Output $b' = 0$ if $c_1 = c_2$
  - Otherwise output $b' = 1$

Advantage?

- If $b = 0$, then $c_0 = c_1 = \text{Enc}_k(0^\ell) \implies \mathcal{A}$ guesses correctly with probability 1
- If $b = 1$, then $c_0 = \text{Enc}_k(0^\ell)$ and $c_1 = \text{Enc}_k(1^\ell)$

  $\implies c_0 \neq c_1$ since otherwise either $\text{Dec}_k(c_0) \neq m_0$ or $\text{Dec}_k(c_1) \neq m_1$

  $\implies \mathcal{A}$ guesses correctly with probability 1

$$\Pr[\text{PrivK}^{\text{mult}}_{\mathcal{A},\Pi}(n) = 1] = 1$$

Multiple message (in)security of OTP

Does OTP have indistinguishable multiple encryptions in the presence of an eavesdropper?

Distinguisher $\mathcal{A}$:

- Output $\vec{M}_0 = \langle 0^\ell, 0^\ell \rangle$ and $\vec{M}_1 = \langle 0^\ell, 1^\ell \rangle$
- Upon receiving $\vec{C} = \langle c_1, c_2 \rangle$:
  - Output $b' = 0$ if $c_1 = c_2$
  - Otherwise output $b' = 1$

Advantage?

- If $b = 0$, then $c_0 = c_1 = \mathsf{Enc}_k(0^\ell) \implies \mathcal{A}$ guesses correctly with probability 1
- If $b = 1$, then $c_0 = \mathsf{Enc}_k(0^\ell)$ and $c_1 = \mathsf{Enc}_k(1^\ell)$

  $\implies c_0 \neq c_1$ since otherwise either $\mathsf{Dec}_k(c_0) \neq m_0$ or $\mathsf{Dec}_k(c_1) \neq m_1$

  $\implies \mathcal{A}$ guesses correctly with probability 1

$$\Pr[\mathsf{PrivK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n) = 1] = 1$$

We are exploiting the fact that, in OTP (and in pseudo OTP), the function Enc_k is **deterministic**!

Multiple message security and deterministic schemes

Observation: The previous adversary works against all schemes with a deterministic encryption function

Multiple message security and deterministic schemes

Observation: The previous adversary works against all schemes with a deterministic encryption function

Theorem If Π is a encryption scheme in which Enc is a deterministic function of the key and the message, then Π cannot have indistinguishable multiple encryptions in the presence of an eavesdropper.

Multiple message security and deterministic schemes

Observation: The previous adversary works against all schemes with a deterministic encryption function

> **Theorem** If Π is a encryption scheme in which Enc is a deterministic function of the key and the message, then Π cannot have indistinguishable multiple encryptions in the presence of an eavesdropper.

Not just a theoretical result: consider the case of yes/no messages

How do we circumvent this limitation?

Multiple message security and deterministic schemes

Observation: The previous adversary works against all schemes with a deterministic encryption function

Theorem If Π is a encryption scheme in which Enc is a deterministic function of the key and the message, then Π cannot have indistinguishable multiple encryptions in the presence of an eavesdropper.

Not just a theoretical result: consider the case of yes/no messages

How do we circumvent this limitation?

- Randomized encryption functions (multiple encryptions of the same message result in different ciphertexts)

Multiple message security and deterministic schemes

Observation: The previous adversary works against all schemes with a deterministic encryption function

Theorem If Π is a encryption scheme in which Enc is a deterministic function of the key and the message, then Π cannot have indistinguishable multiple encryptions in the presence of an eavesdropper.

Not just a theoretical result: consider the case of yes/no messages

How do we circumvent this limitation?

- Randomized encryption functions (multiple encryptions of the same message result in different ciphertexts)
- Stateful schemes (Enc stores some additional information that is preserved between calls and it is used to produce different ciphertexts even when the same message is encrypted twice)

An even stronger threat model

We will **not** focus on designing schemes with indistinguishable multiple encryptions

An even stronger threat model

We will **not** focus on designing schemes with indistinguishable multiple encryptions

We adopt an even stronger threat model instead!

security against chosen-plaintext attacks (CPA)

An even stronger threat model

We will **not** focus on designing schemes with indistinguishable multiple encryptions

We adopt an even stronger threat model instead!

security against chosen-plaintext attacks (CPA)

All modern encryption schemes should be **at least** CPA-secure

Chosen-plaintext attack

The adversary learns the ciphertexts corresponding to one or more plaintexts <u>of its choice</u>.

The adversary wants to deduce information about the underlying plaintext of some other ciphertext produced using the same key

Chosen-plaintext attack

The adversary learns the ciphertexts corresponding to one or more plaintexts of its choice.

The adversary wants to deduce information about the underlying plaintext of some other ciphertext produced using the same key

How can the adversary learn ciphertexts of the desired plaintexts?

Chosen-plaintext attack

The adversary learns the ciphertexts corresponding to one or more plaintexts <u>of its choice</u>.

The adversary wants to deduce information about the underlying plaintext of some other ciphertext produced using the same key

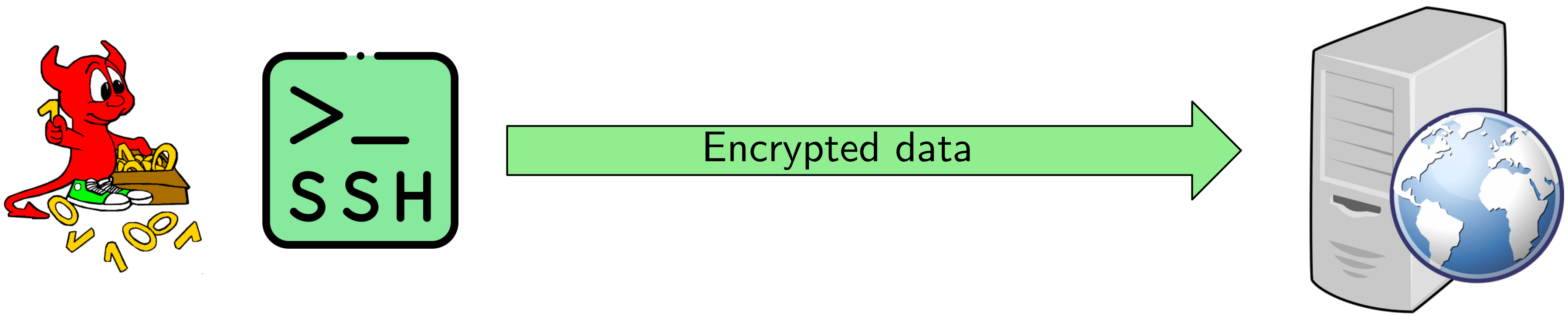
How can the adversary learn ciphertexts of the desired plaintexts?

Chosen-plaintext attack

The adversary learns the ciphertexts corresponding to one or more plaintexts <u>of its choice</u>.

The adversary wants to deduce information about the underlying plaintext of some other ciphertext produced using the same key

How can the adversary learn ciphertexts of the desired plaintexts?

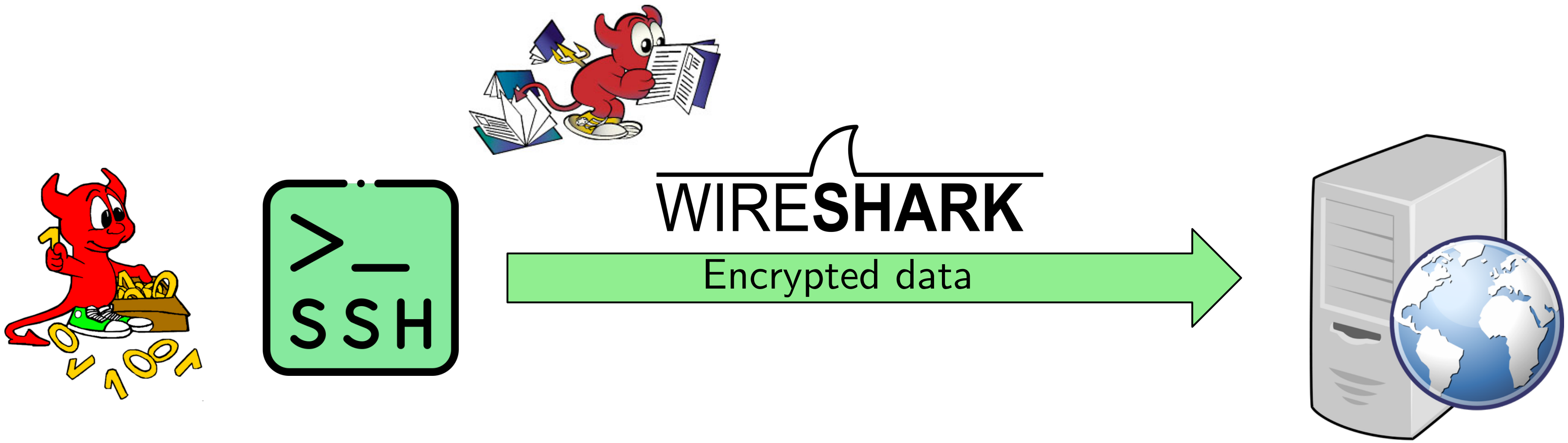


Chosen-plaintext attack

The adversary learns the ciphertexts corresponding to one or more plaintexts <u>of its choice</u>.

The adversary wants to deduce information about the underlying plaintext of some other ciphertext produced using the same key

How can the adversary learn ciphertexts of the desired plaintexts?

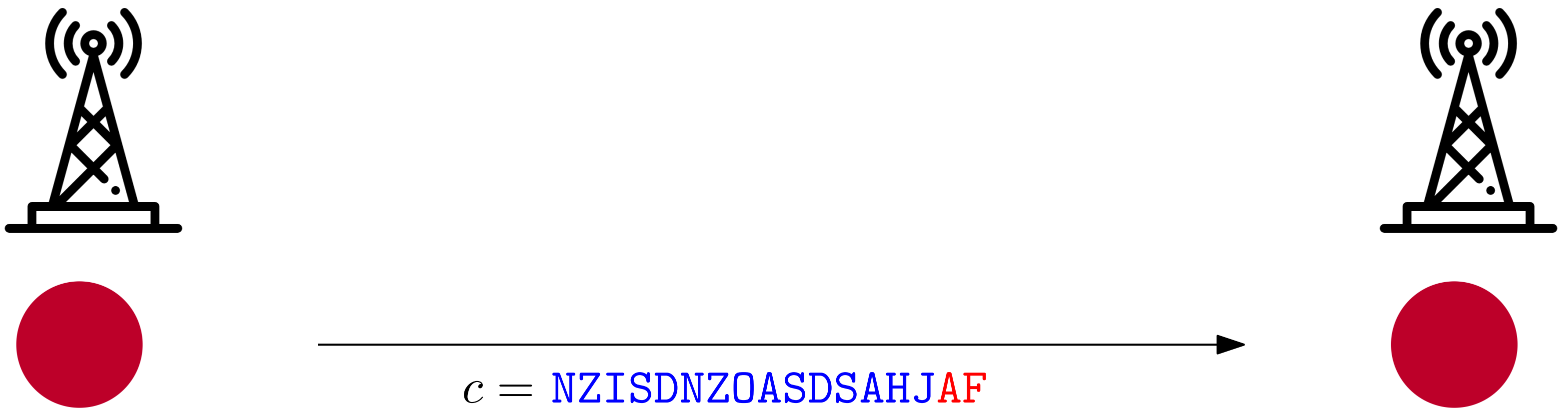


Chosen-plaintext attack

The adversary learns the ciphertexts corresponding to one or more plaintexts <u>of its choice</u>.

The adversary wants to deduce information about the underlying plaintext of some other ciphertext produced using the same key

How can the adversary learn ciphertexts of the desired plaintexts?

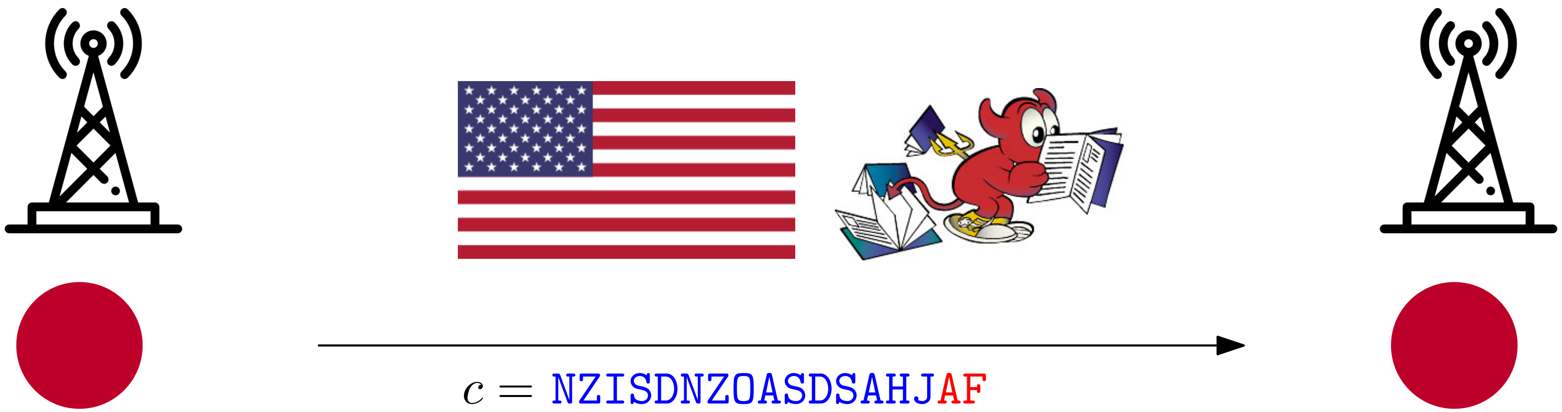


Chosen-plaintext attack

The adversary learns the ciphertexts corresponding to one or more plaintexts <u>of its choice</u>.

The adversary wants to deduce information about the underlying plaintext of some other ciphertext produced using the same key

How can the adversary learn ciphertexts of the desired plaintexts?



Chosen-plaintext attack

The adversary learns the ciphertexts corresponding to one or more plaintexts <u>of its choice</u>.

The adversary wants to deduce information about the underlying plaintext of some other ciphertext produced using the same key

How can the adversary learn ciphertexts of the desired plaintexts?

Chosen-plaintext attack

The adversary learns the ciphertexts corresponding to one or more plaintexts <u>of its choice</u>.

The adversary wants to deduce information about the underlying plaintext of some other ciphertext produced using the same key

How can the adversary learn ciphertexts of the desired plaintexts?

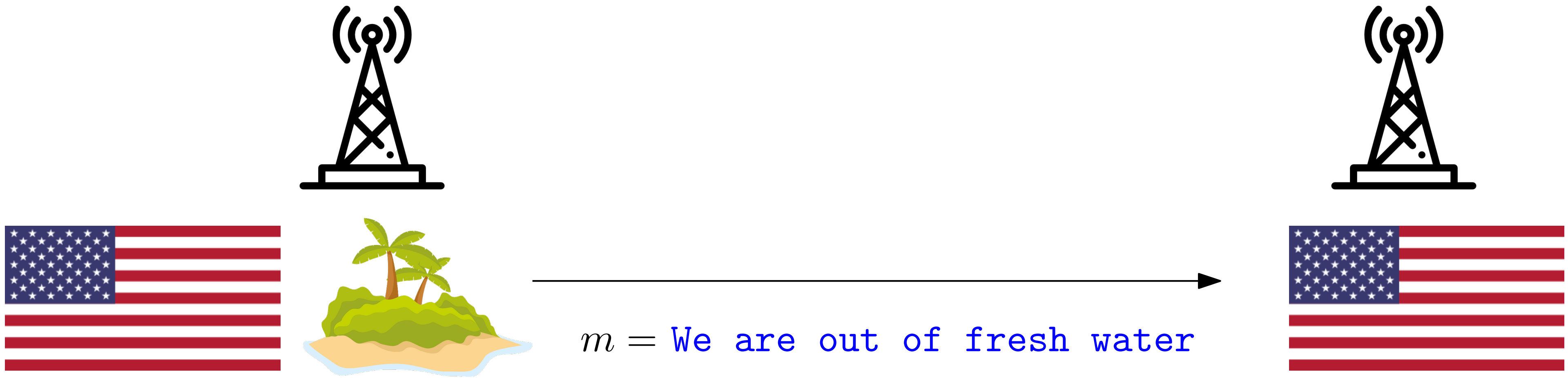
The U.S. cryptanalysts believed that AF meant Midway Island, but they were not 100% sure

Chosen-plaintext attack

The adversary learns the ciphertexts corresponding to one or more plaintexts of its choice.

The adversary wants to deduce information about the underlying plaintext of some other ciphertext produced using the same key

How can the adversary learn ciphertexts of the desired plaintexts?



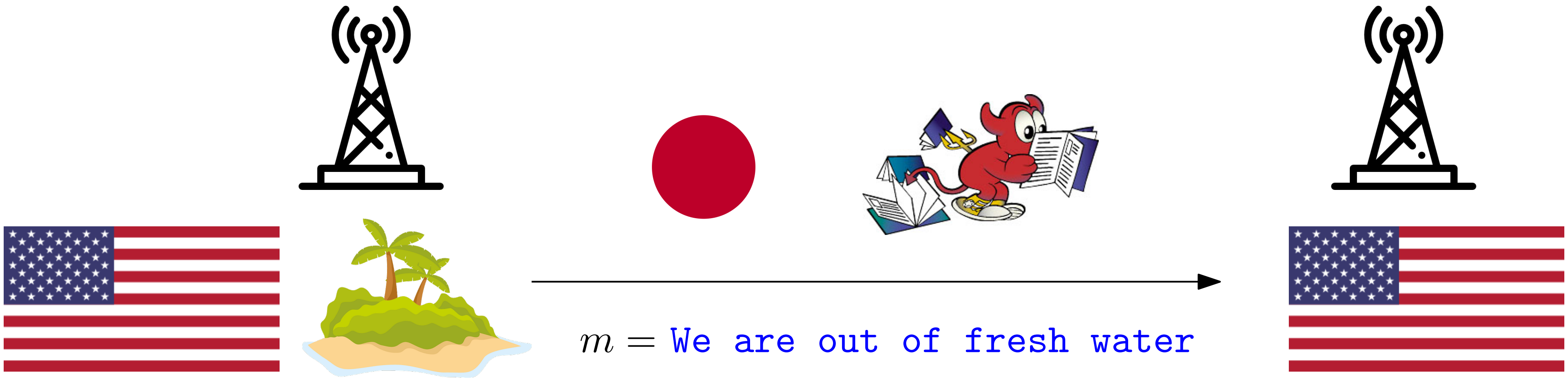
They sent a fake unencrypted message from Midway Island

Chosen-plaintext attack

The adversary learns the ciphertexts corresponding to one or more plaintexts <u>of its choice</u>.

The adversary wants to deduce information about the underlying plaintext of some other ciphertext produced using the same key

How can the adversary learn ciphertexts of the desired plaintexts?



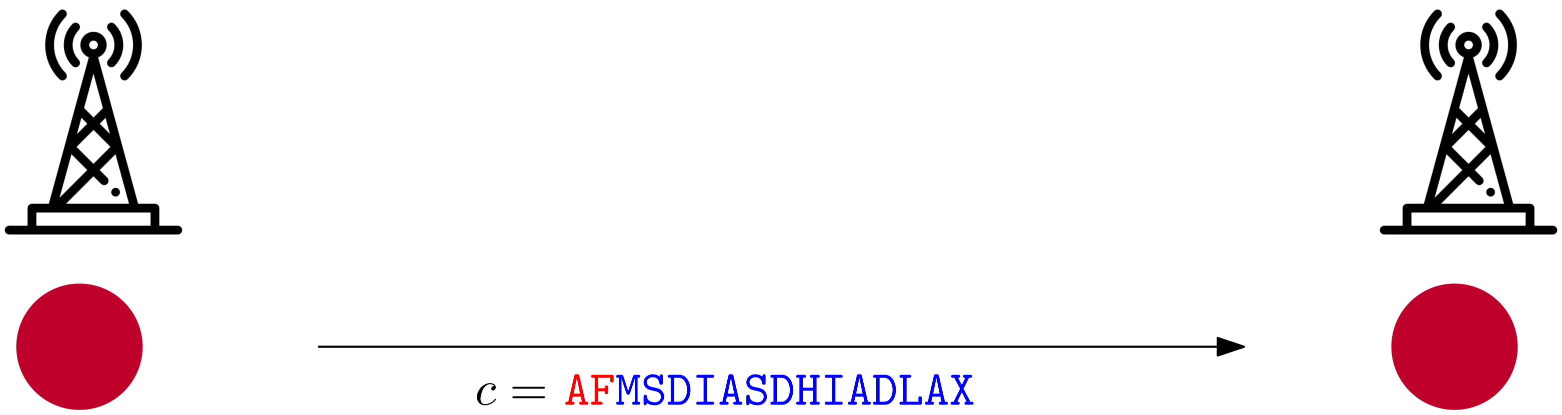
They sent a fake unencrypted message from Midway Island

Chosen-plaintext attack

The adversary learns the ciphertexts corresponding to one or more plaintexts <u>of its choice</u>.

The adversary wants to deduce information about the underlying plaintext of some other ciphertext produced using the same key

How can the adversary learn ciphertexts of the desired plaintexts?

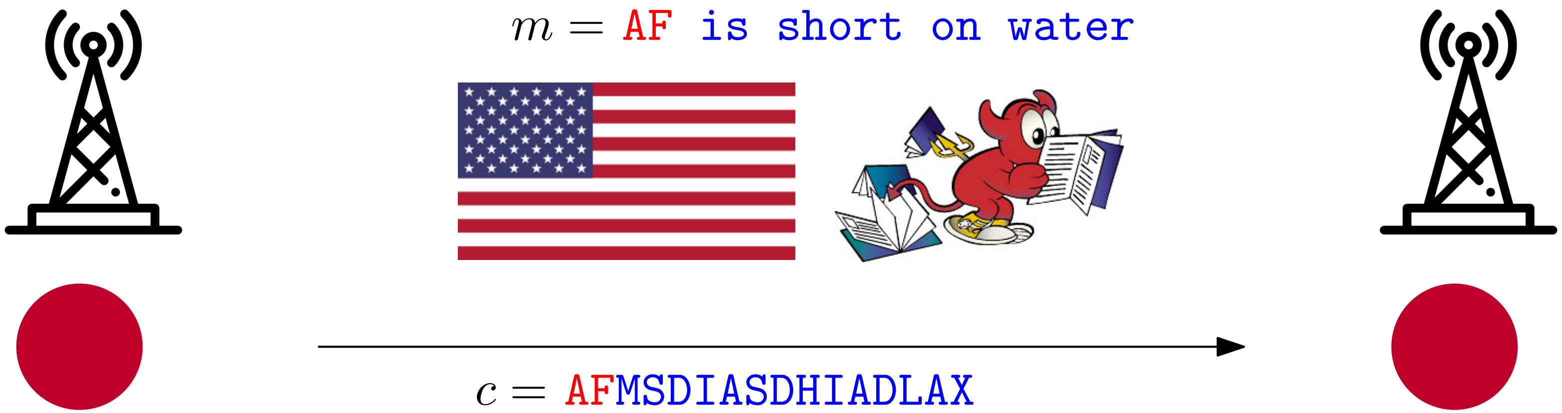


Chosen-plaintext attack

The adversary learns the ciphertexts corresponding to one or more plaintexts <u>of its choice</u>.

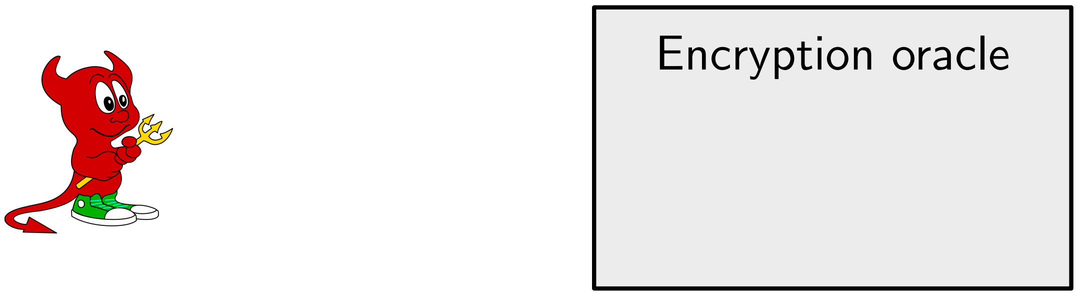
The adversary wants to deduce information about the underlying plaintext of some other ciphertext produced using the same key

How can the adversary learn ciphertexts of the desired plaintexts?



Modeling CPA security

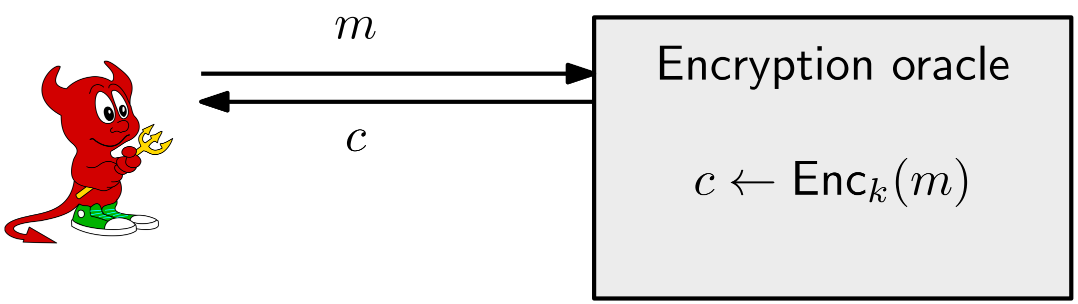
A key $k \leftarrow \mathsf{Gen}(1^n)$ is generated and the adversary $\mathcal{A}$ has access to an **encryption oracle**



Modeling CPA security

A key $k \leftarrow \mathsf{Gen}(1^n)$ is generated and the adversary $\mathcal{A}$ has access to an **encryption oracle**

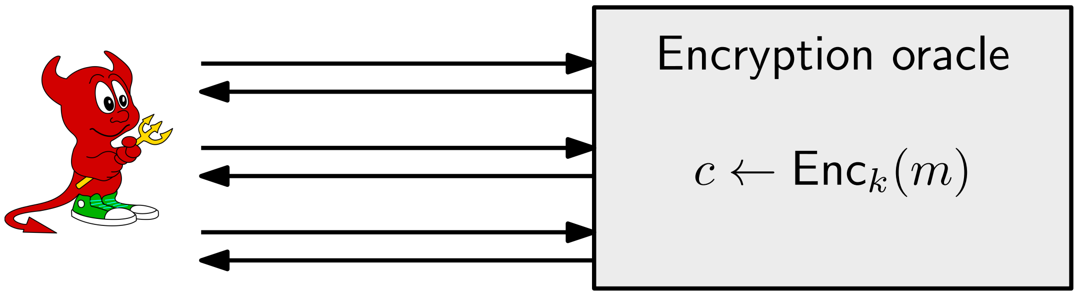
- The encryption oracle acts as a black-box that can be **queried** with a message m and returns an encryption c of m



Modeling CPA security

A key $k \leftarrow \mathsf{Gen}(1^n)$ is generated and the adversary $\mathcal{A}$ has access to an **encryption oracle**

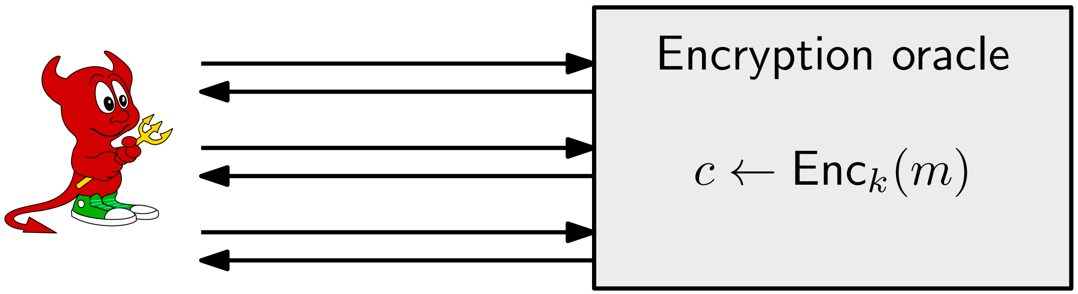
- The encryption oracle acts as a black-box that can be **queried** with a message m and returns an encryption c of m
- There is no limit on the number of queries the adversary can make
(other than the time limit of hte aversary, each query requries constant time)



Modeling CPA security

A key $k \leftarrow \mathsf{Gen}(1^n)$ is generated and the adversary $\mathcal{A}$ has access to an **encryption oracle**

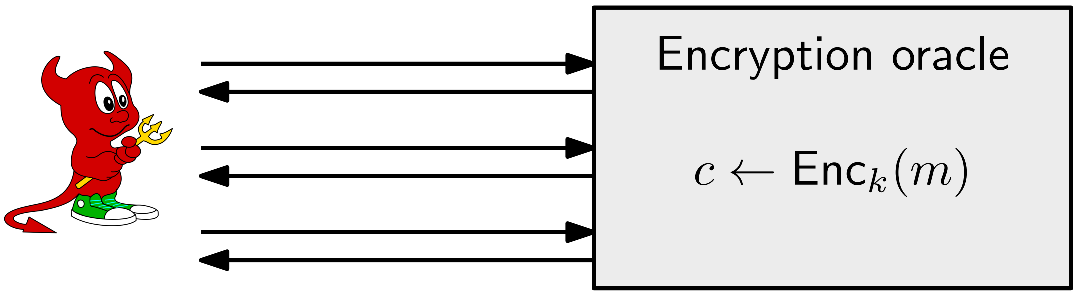
- The encryption oracle acts as a black-box that can be **queried** with a message m and returns an encryption c of m
- There is no limit on the number of queries the adversary can make
(other than the time limit of hte aversary, each query requries constant time)
- All messages are encrypted using the same key k, i.e., the oracle returns $c \leftarrow \mathsf{Enc}_k(m)$



Modeling CPA security

A key $k \leftarrow \mathsf{Gen}(1^n)$ is generated and the adversary $\mathcal{A}$ has access to an **encryption oracle**

- The encryption oracle acts as a black-box that can be **queried** with a message m and returns an encryption c of m
- There is no limit on the number of queries the adversary can make
  (other than the time limit of hte aversary, each query requries constant time)
- All messages are encrypted using the same key k, i.e., the oracle returns $c \leftarrow \mathsf{Enc}_k(m)$
- The key k is **unknown** to the adversary



Modeling CPA security

A key $k \leftarrow \mathsf{Gen}(1^n)$ is generated

Encryption oracle

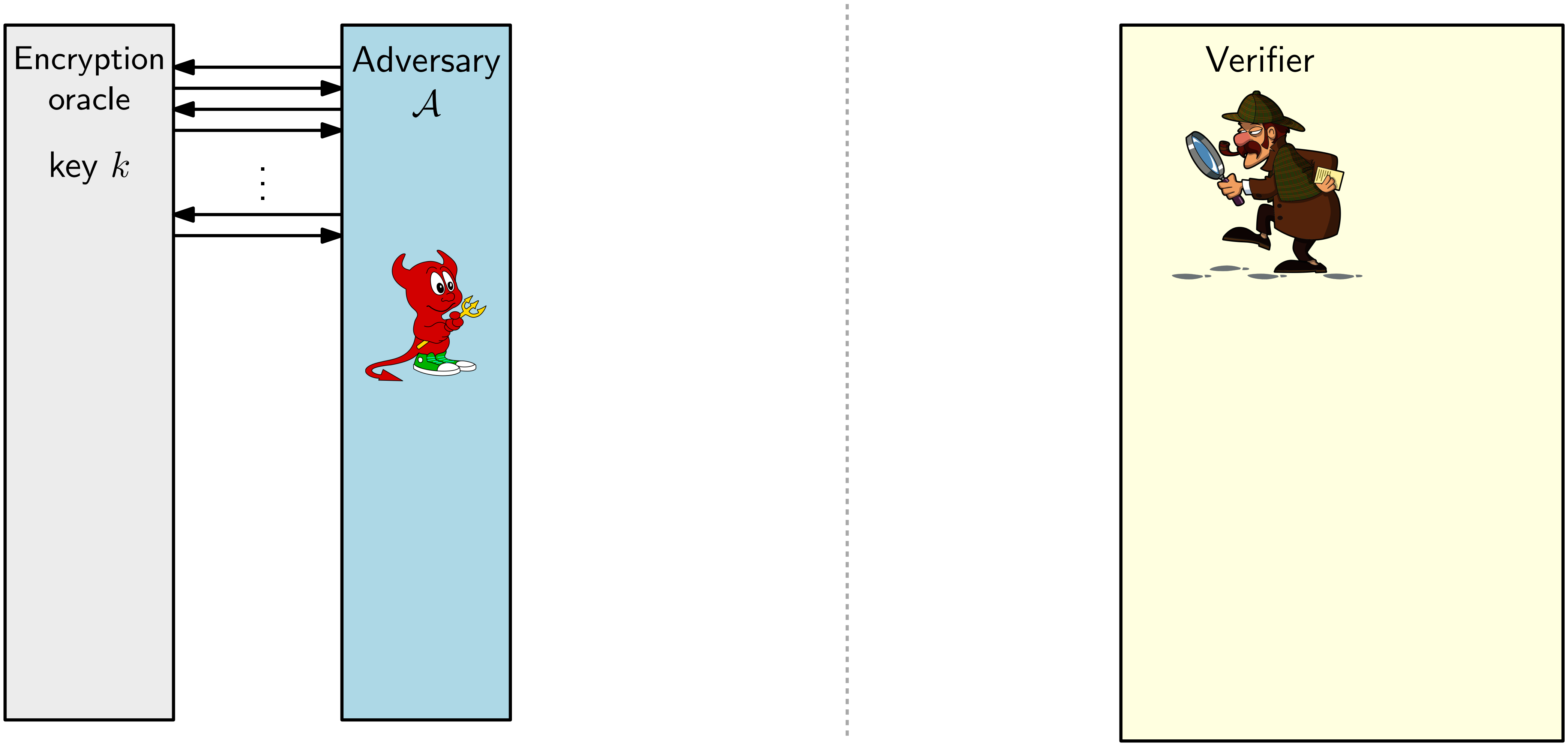
key k

Adversary $\mathcal{A}$

Verifier

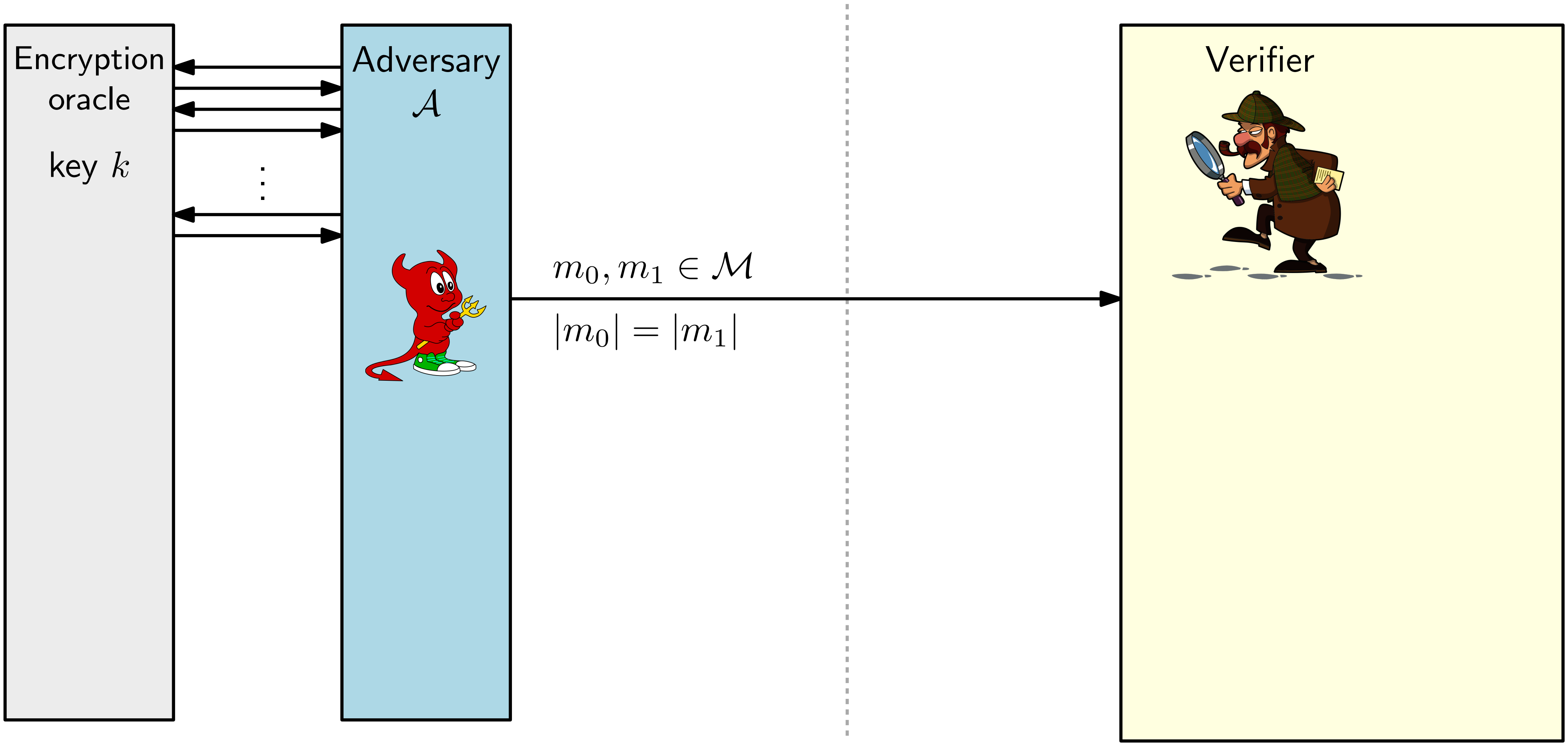
Modeling CPA security

A key $k \leftarrow \mathsf{Gen}(1^n)$ is generated



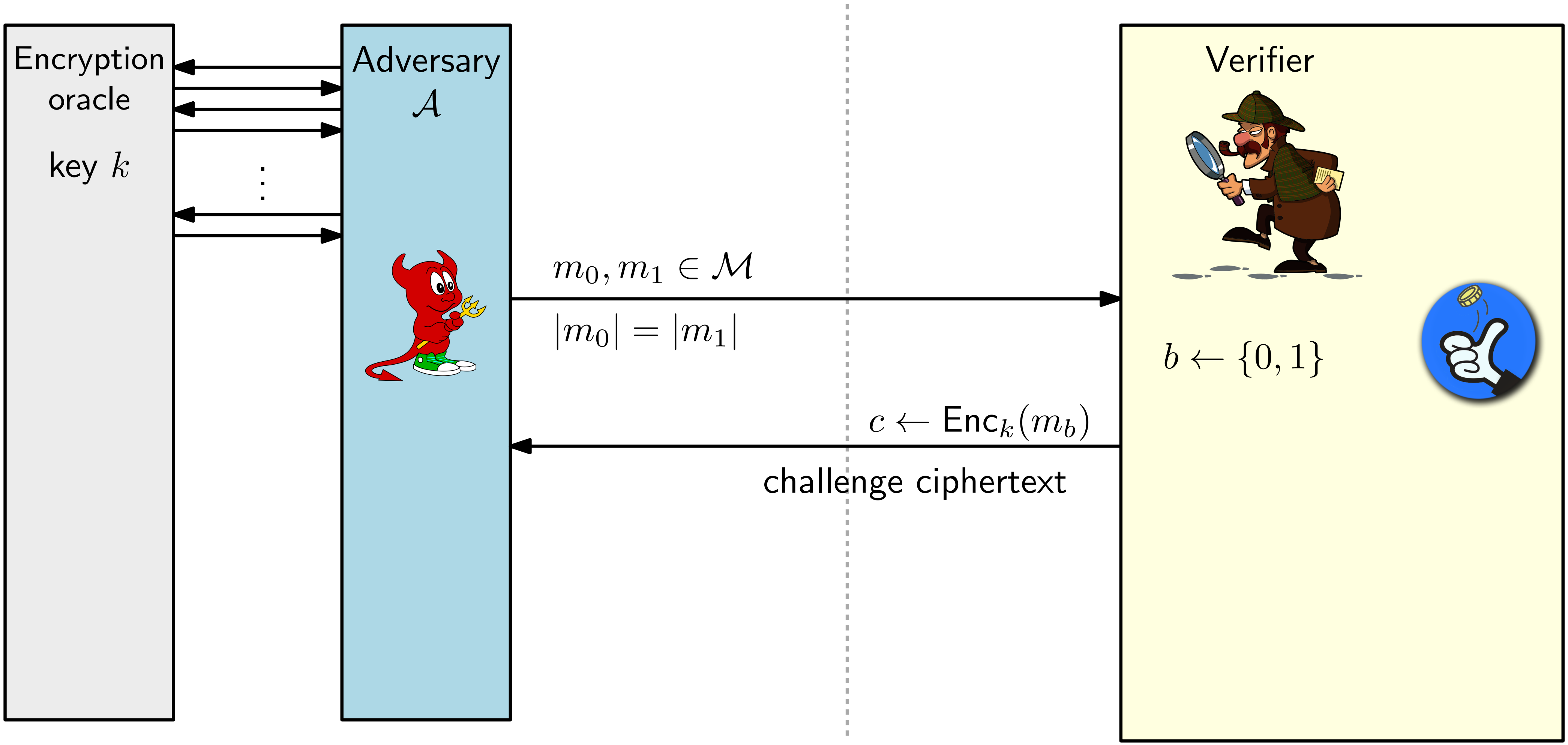
Modeling CPA security

A key $k \leftarrow \mathsf{Gen}(1^n)$ is generated



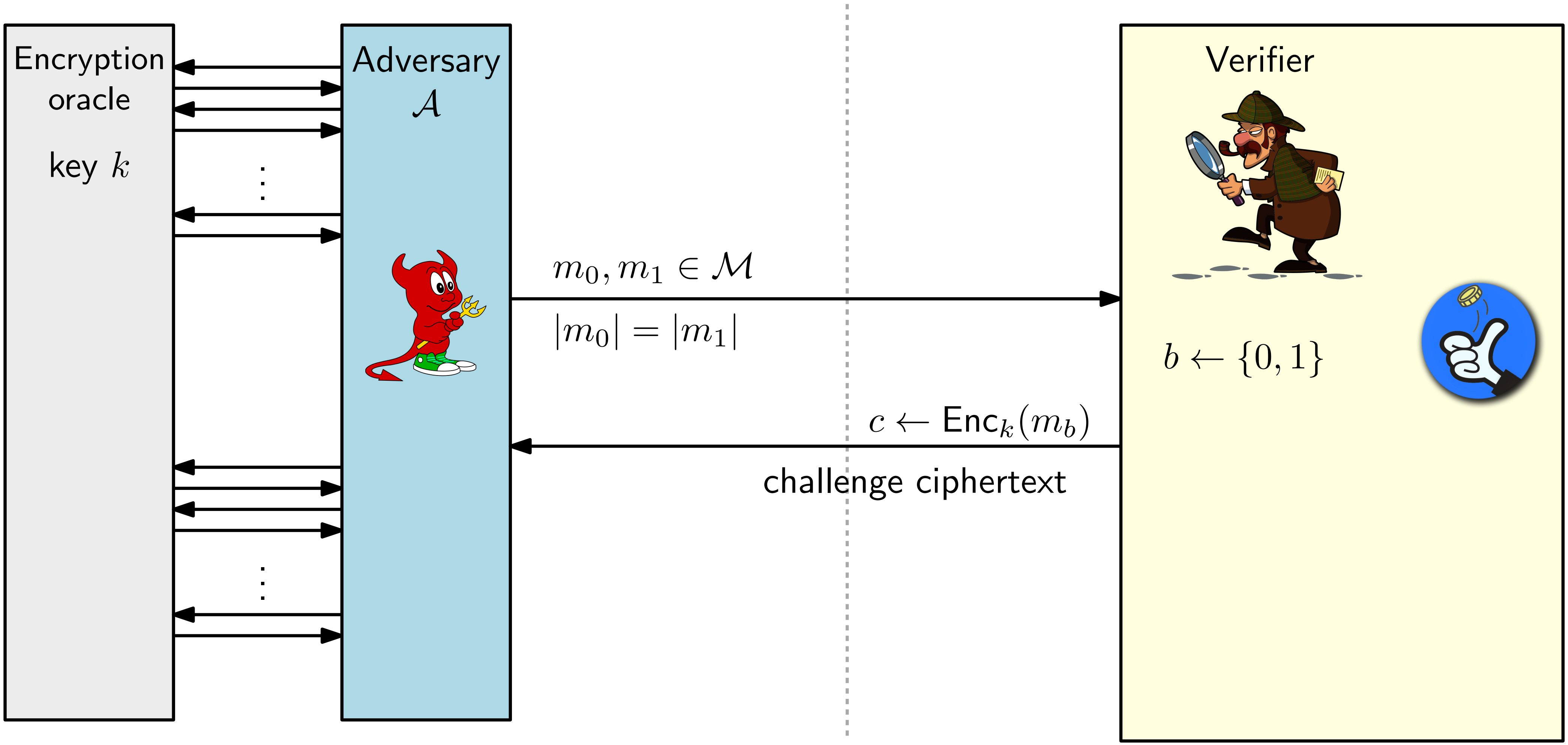
Modeling CPA security

A key $k \leftarrow \mathsf{Gen}(1^n)$ is generated



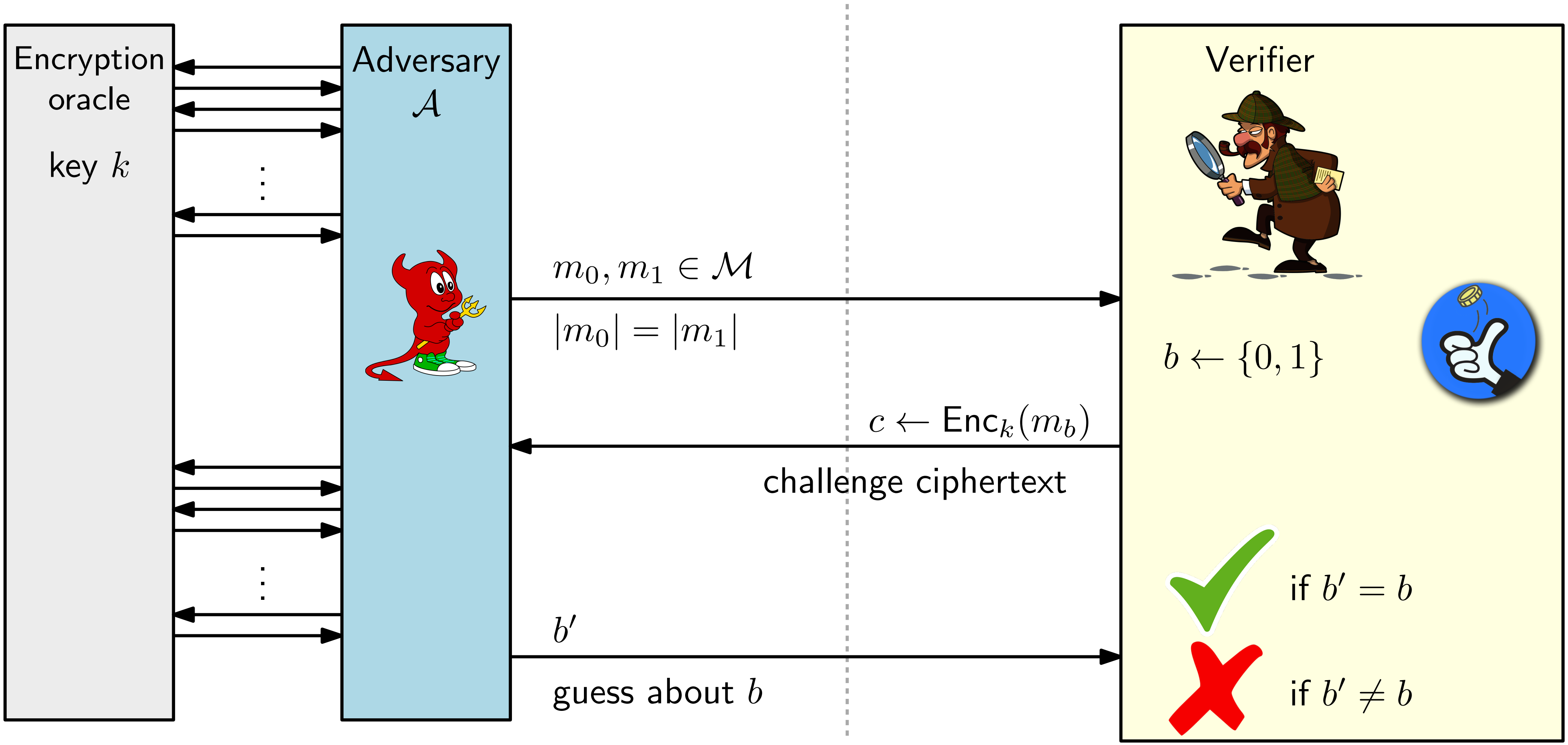
Modeling CPA security

A key $k \leftarrow \mathsf{Gen}(1^n)$ is generated



Modeling CPA security

A key $k \leftarrow \mathsf{Gen}(1^n)$ is generated



Modeling CPA security

Formally, if $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a private key encryption scheme with message space $\mathcal{M}$, we denote the following experiment by $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}$

Modeling CPA security

Formally, if $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a private key encryption scheme with message space $\mathcal{M}$, we denote the following experiment by $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}$

- A key $k \leftarrow \mathsf{Gen}(1^n)$ is generated
- $\mathcal{A}$ can interact with an encryption oracle that provides access to $\mathsf{Enc}_k(\cdot)$
- $\mathcal{A}$ chooses two distinct messages $m_0, m_1 \in \mathcal{M}$ with $|m_0| = |m_1|$
- A uniform random bit $b \in \{0, 1\}$ is generated
- The *challenge ciphertext* c is computed by $\mathsf{Enc}_k(m_b)$, and given to $\mathcal{A}$
- $\mathcal{A}$ can interact with an encryption oracle that provides access to $\mathsf{Enc}_k(\cdot)$
- $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ about b
- The *output of the experiment* is defined to be 1 if $b' = b$, and 0 otherwise

Modeling CPA security

Formally, if $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a private key encryption scheme with message space $\mathcal{M}$, we denote the following experiment by $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}$

- A key $k \leftarrow \mathsf{Gen}(1^n)$ is generated
- $\mathcal{A}$ can interact with an encryption oracle that provides access to $\mathsf{Enc}_k(\cdot)$
- $\mathcal{A}$ chooses two distinct messages $m_0, m_1 \in \mathcal{M}$ with $|m_0| = |m_1|$
- A uniform random bit $b \in \{0,1\}$ is generated
- The *challenge ciphertext* c is computed by $\mathsf{Enc}_k(m_b)$, and given to $\mathcal{A}$
- $\mathcal{A}$ can interact with an encryption oracle that provides access to $\mathsf{Enc}_k(\cdot)$
- $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$ about b
- The *output of the experiment* is defined to be 1 if $b' = b$, and 0 otherwise

k and b are unknown to $\mathcal{A}$

Definition of CPA-security

Definition: *A private-key encryption scheme* Π *has indistinguishable encryptions under a chosen-plaintext attack (is* **CPA-secure***) if, for every probabilistic polynomial-time adversary* $\mathcal{A}$*, there is a negligible function* ε *such that:*

$$\Pr[\mathit{PrivK}^{\mathit{cpa}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \varepsilon(n)$$

CPA-security

Any private-key encryption scheme that is CPA-secure is also CPA-secure for multiple encryptions

CPA-security

Any private-key encryption scheme that is CPA-secure is also CPA-secure for multiple encryptions

$\Downarrow$

If Π is CPA-secure then Π has indistinguishable multiple encryptions in the presence of an eavesdropper (and hence it is also EAV-secure)

CPA-security

Any private-key encryption scheme that is CPA-secure is also CPA-secure for multiple encryptions

$\Downarrow$

If Π is CPA-secure then Π has indistinguishable multiple encryptions in the presence of an eavesdropper (and hence it is also EAV-secure)

$+$

No stateless, deterministic encryption scheme has indistinguishable multiple encryptions in the presence of an eavesdropper

CPA-security

Any private-key encryption scheme that is CPA-secure is also CPA-secure for multiple encryptions

$\Downarrow$

If Π is CPA-secure then Π has indistinguishable multiple encryptions in the presence of an eavesdropper (and hence it is also EAV-secure)

$+$

No stateless, deterministic encryption scheme has indistinguishable multiple encryptions in the presence of an eavesdropper

$\Downarrow$

No stateless, deterministic encryption scheme can be CPA-secure