Foundations

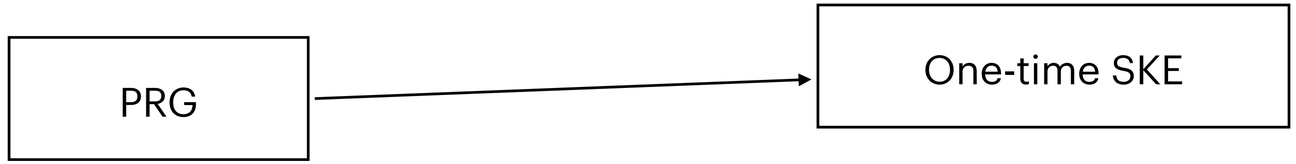
601.442/642 Modern Cryptography

3rd March 2026

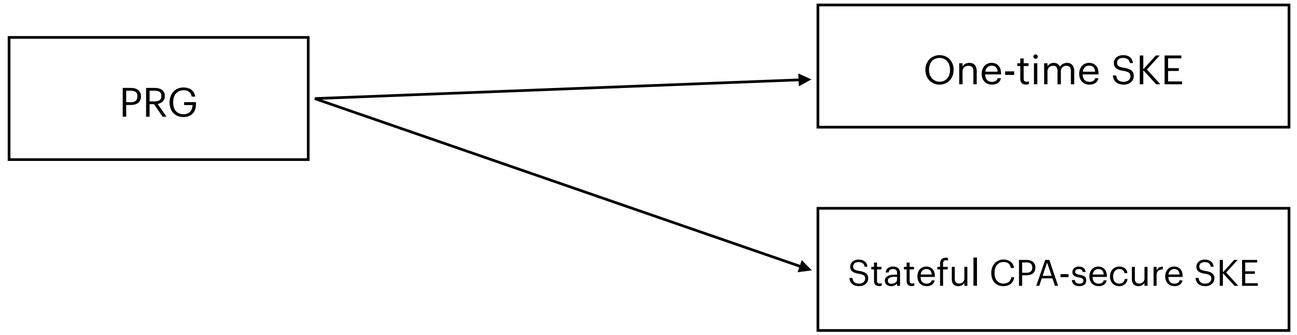
The Picture So Far

The Picture So Far

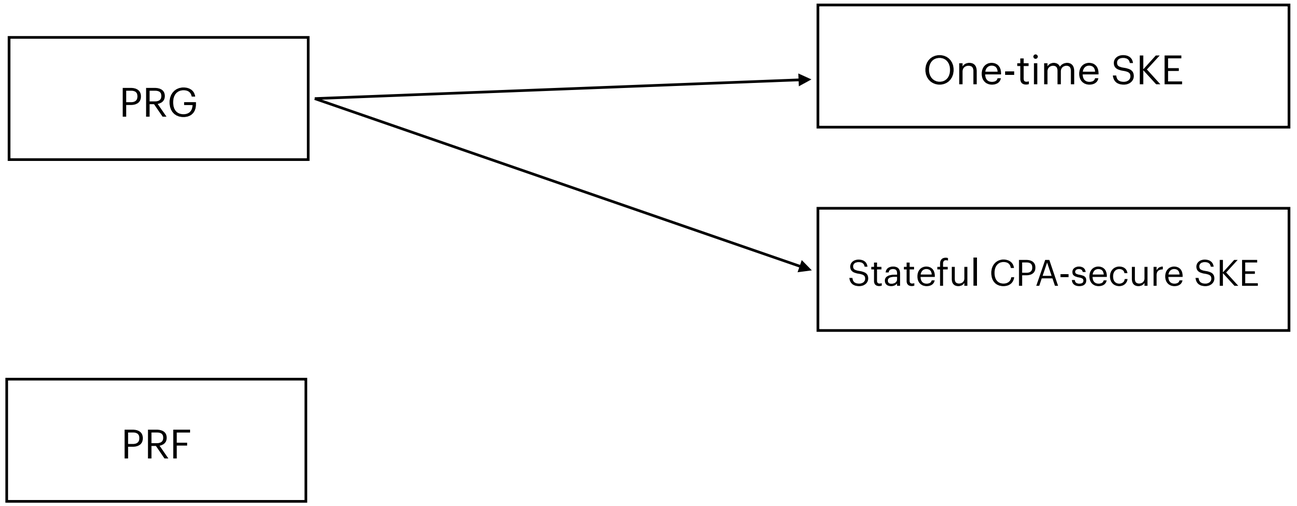
The Picture So Far



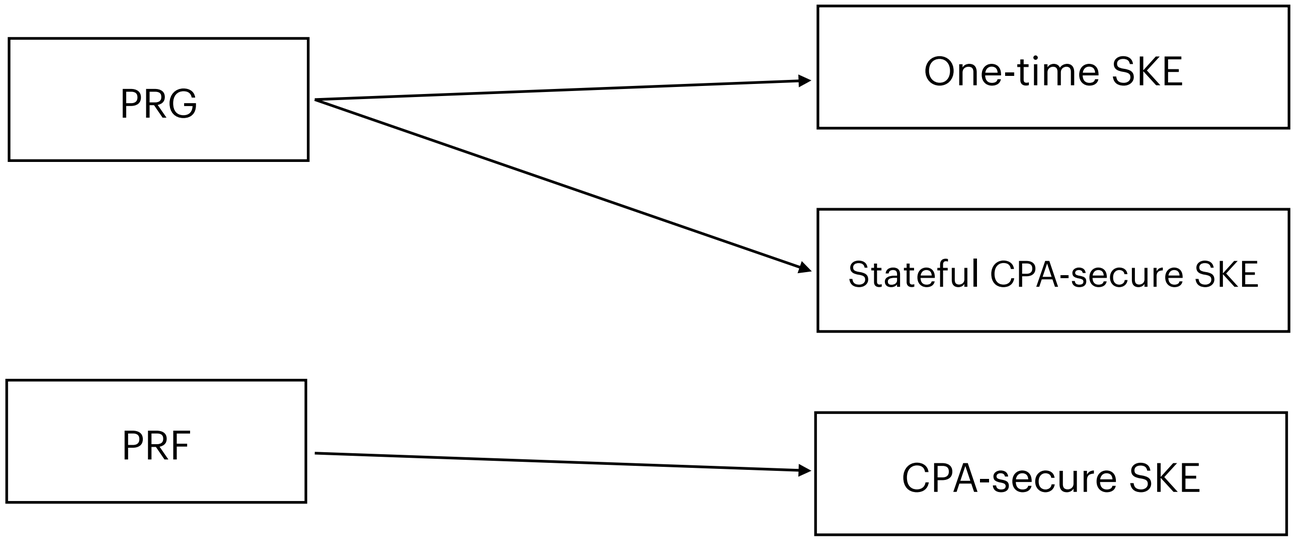
The Picture So Far



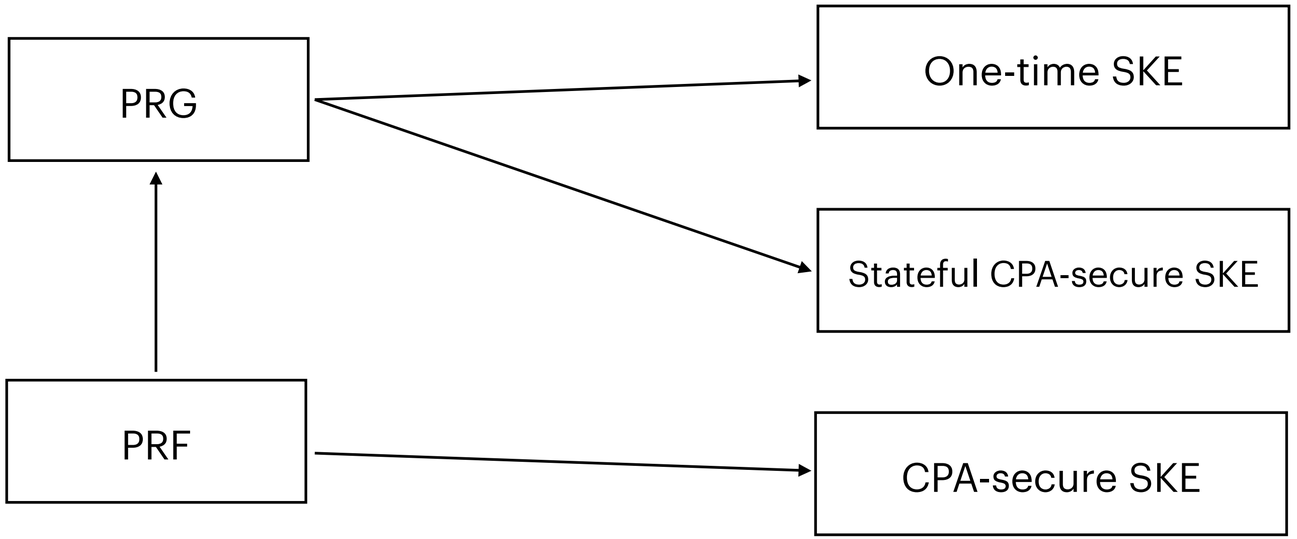
The Picture So Far



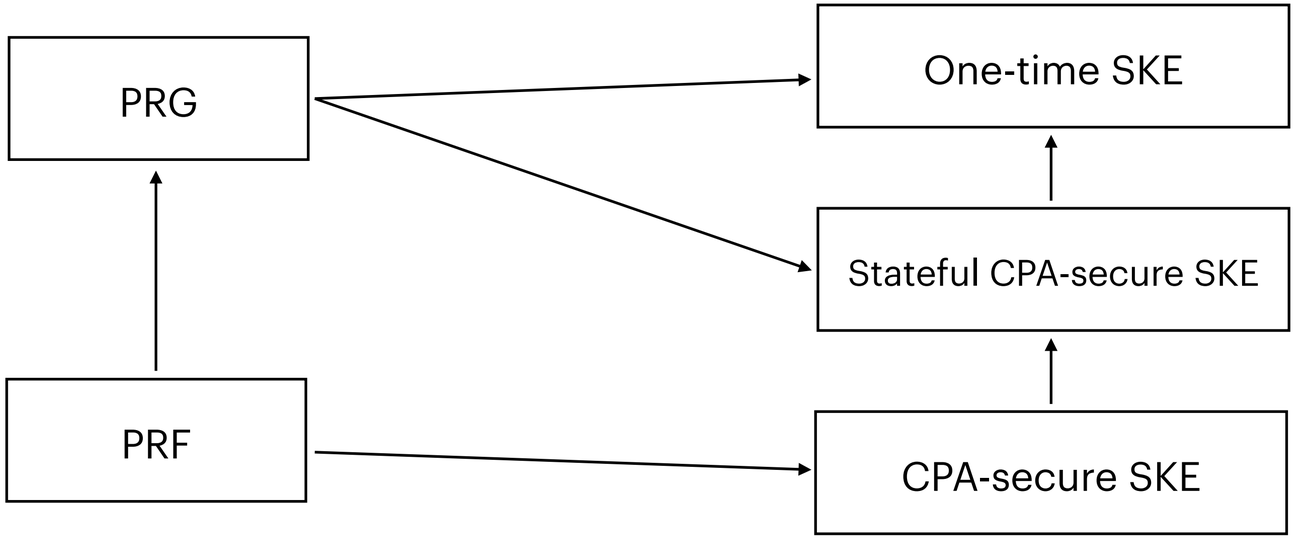
The Picture So Far



The Picture So Far

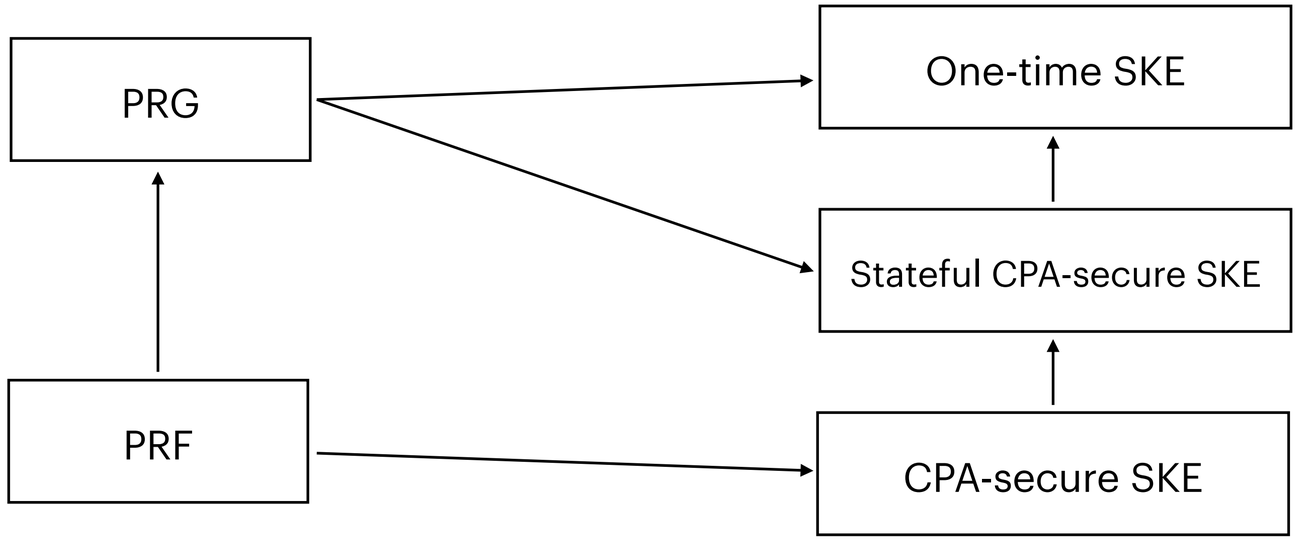


The Picture So Far

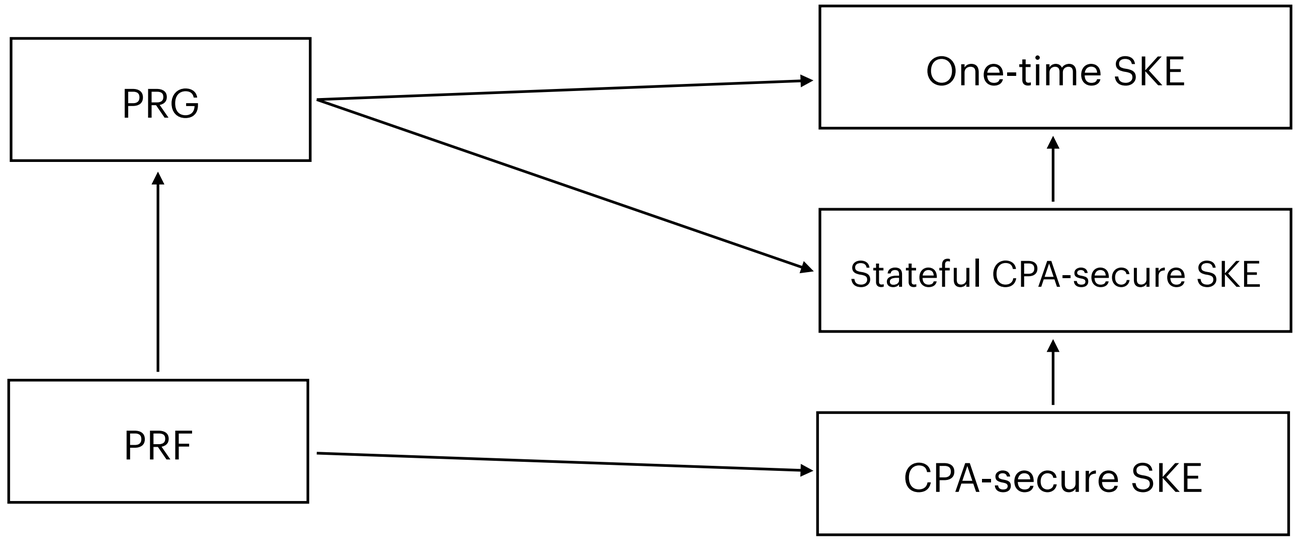


The Picture So Far

Why do we believe PRGs and PRFs exist?
How to construct them?



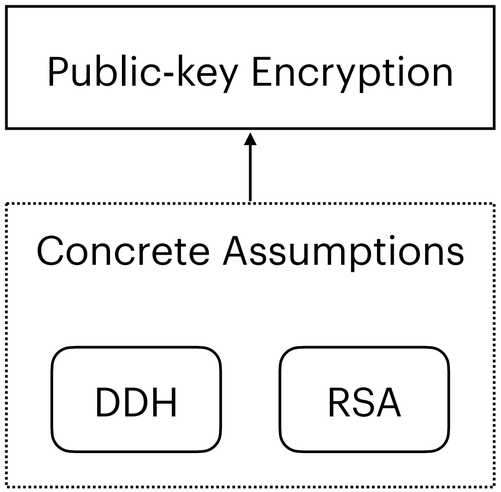
The Picture So Far

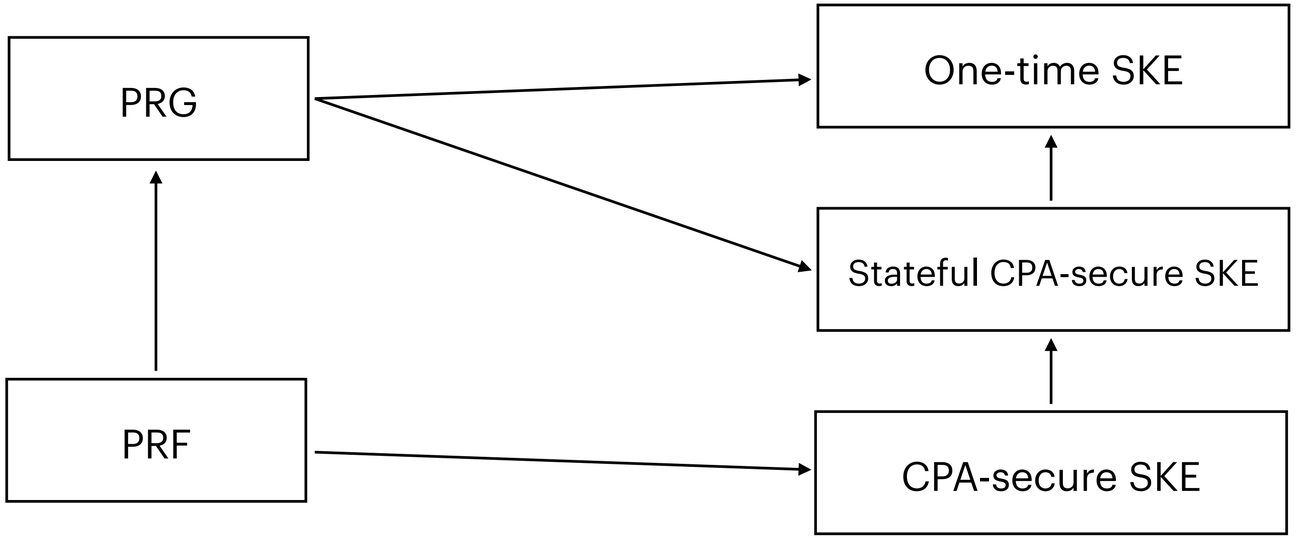


Why do we believe PRGs and PRFs exist?
How to construct them?

Can we construct PRFs from PRGs?

The Picture So Far

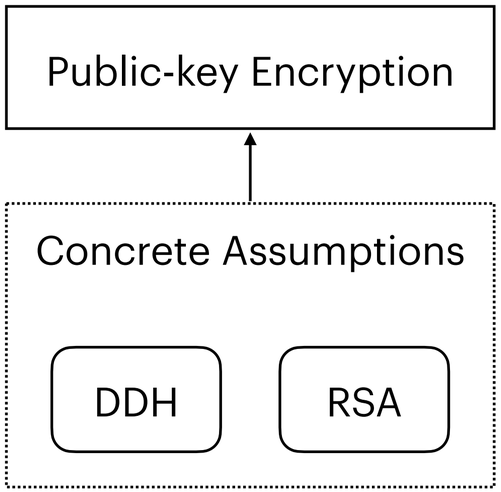


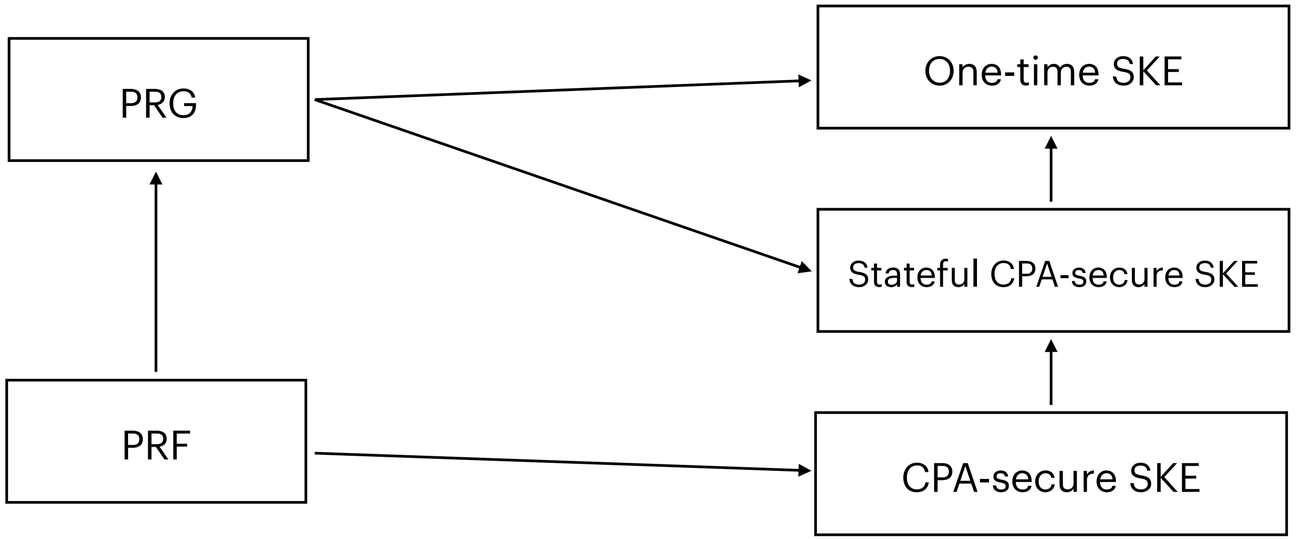


Why do we believe PRGs and PRFs exist?
How to construct them?

Can we construct PRFs from PRGs?

The Picture So Far



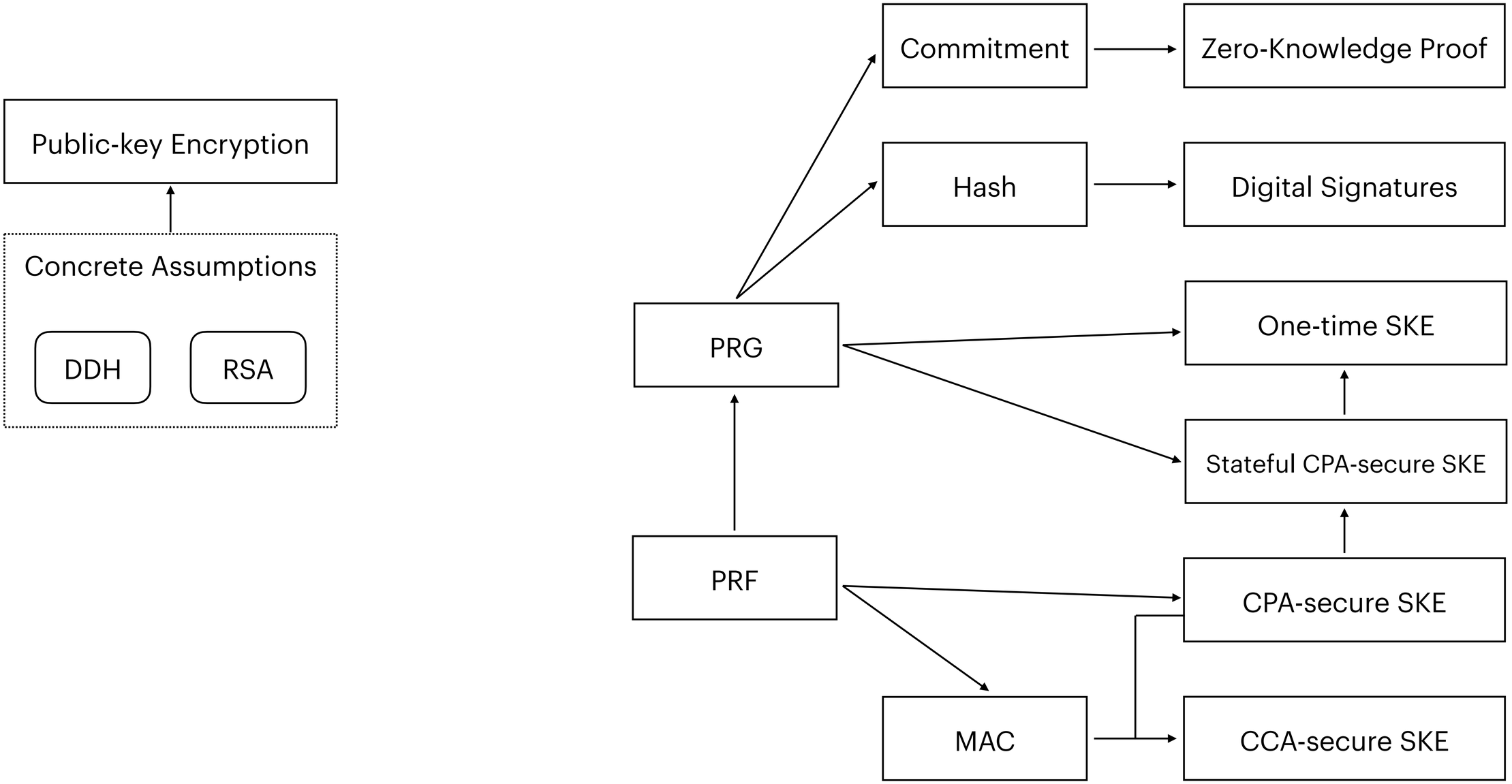


Why do we believe PRGs and PRFs exist?
How to construct them?

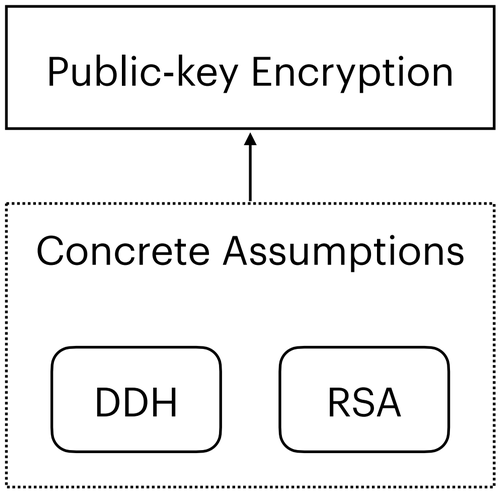
Can we construct PRFs from PRGs?

Can we construct PKE from PRFs?

The Cryptography Landscape

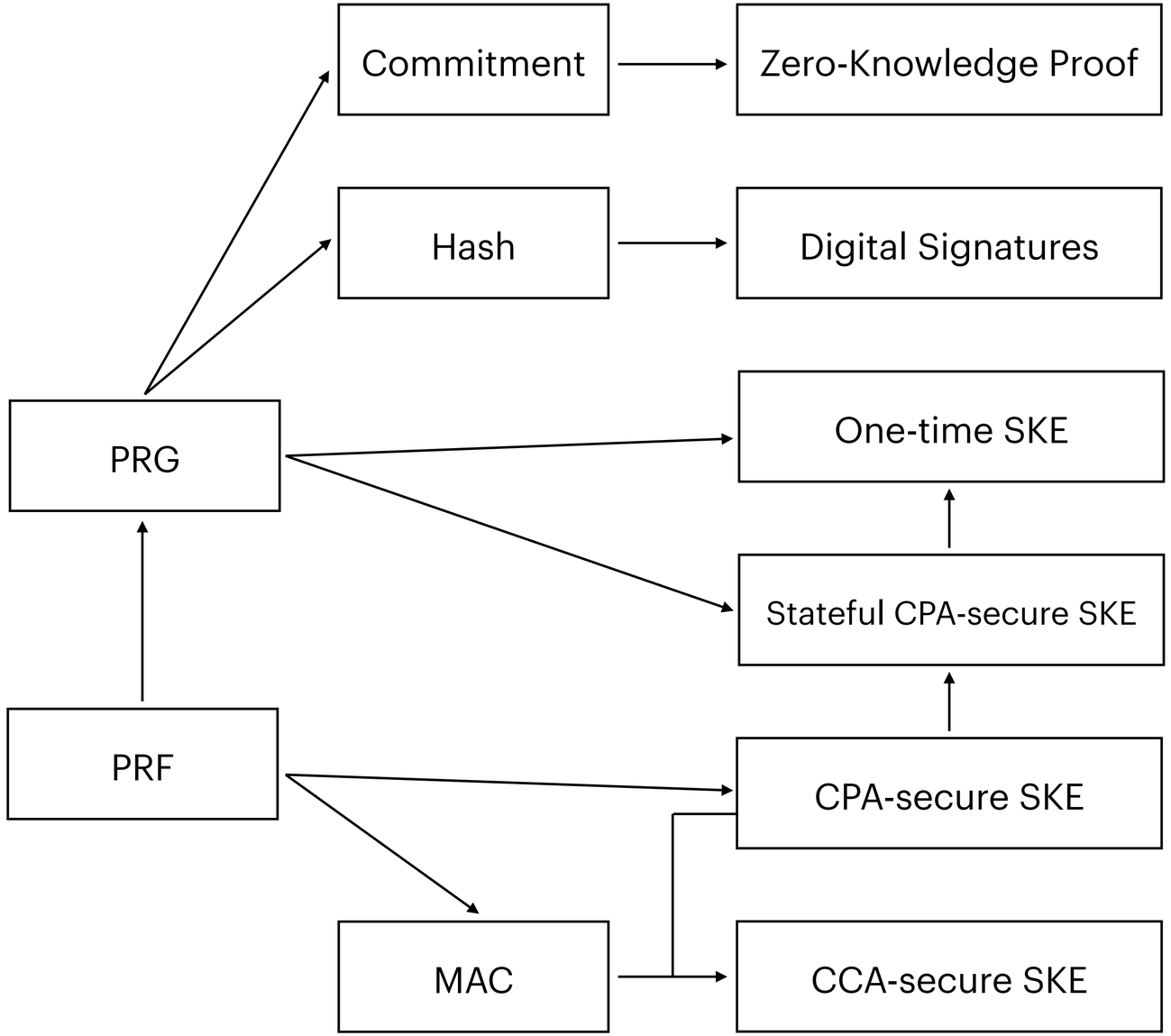


The Cryptography Landscape

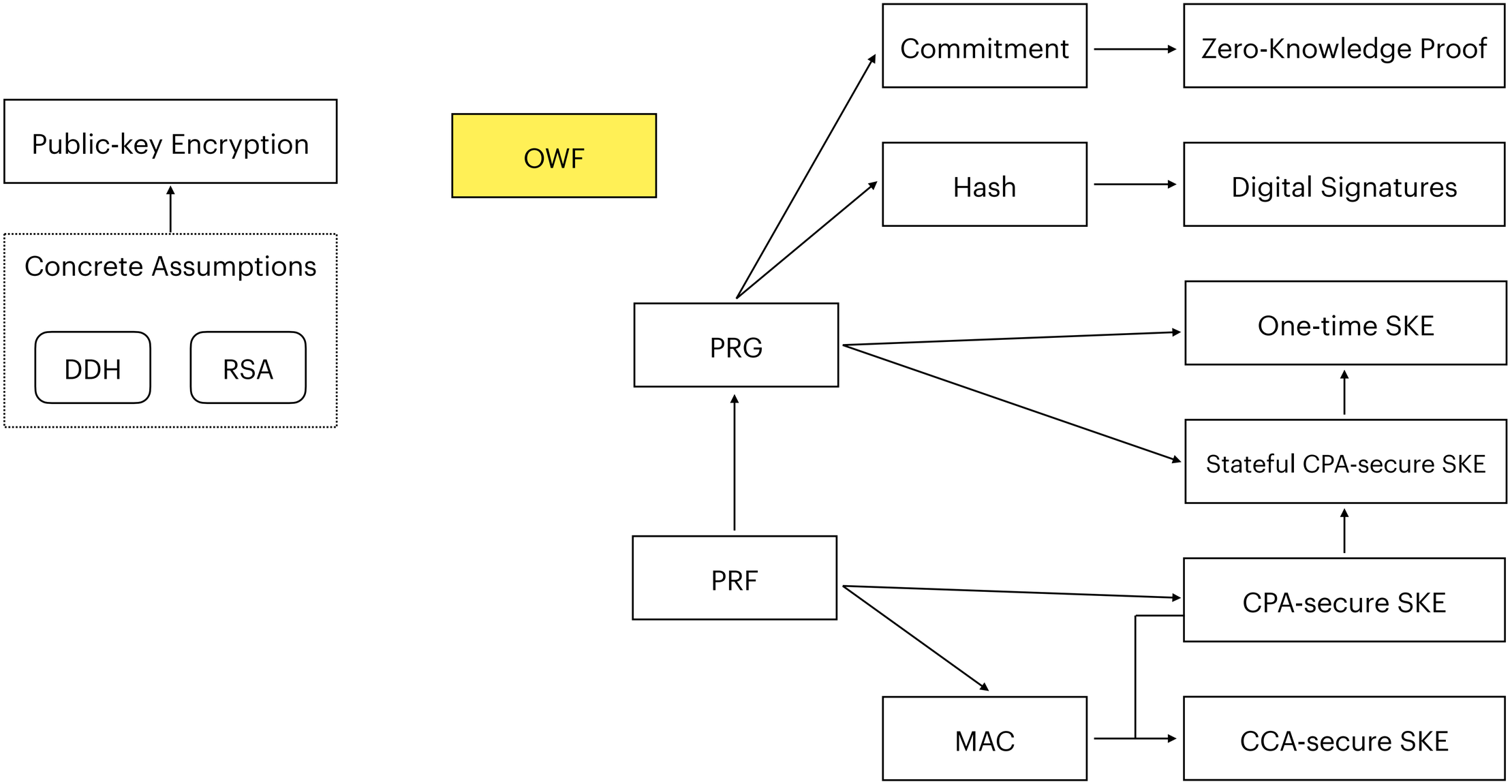


All results are of the form: $X \implies Y$.

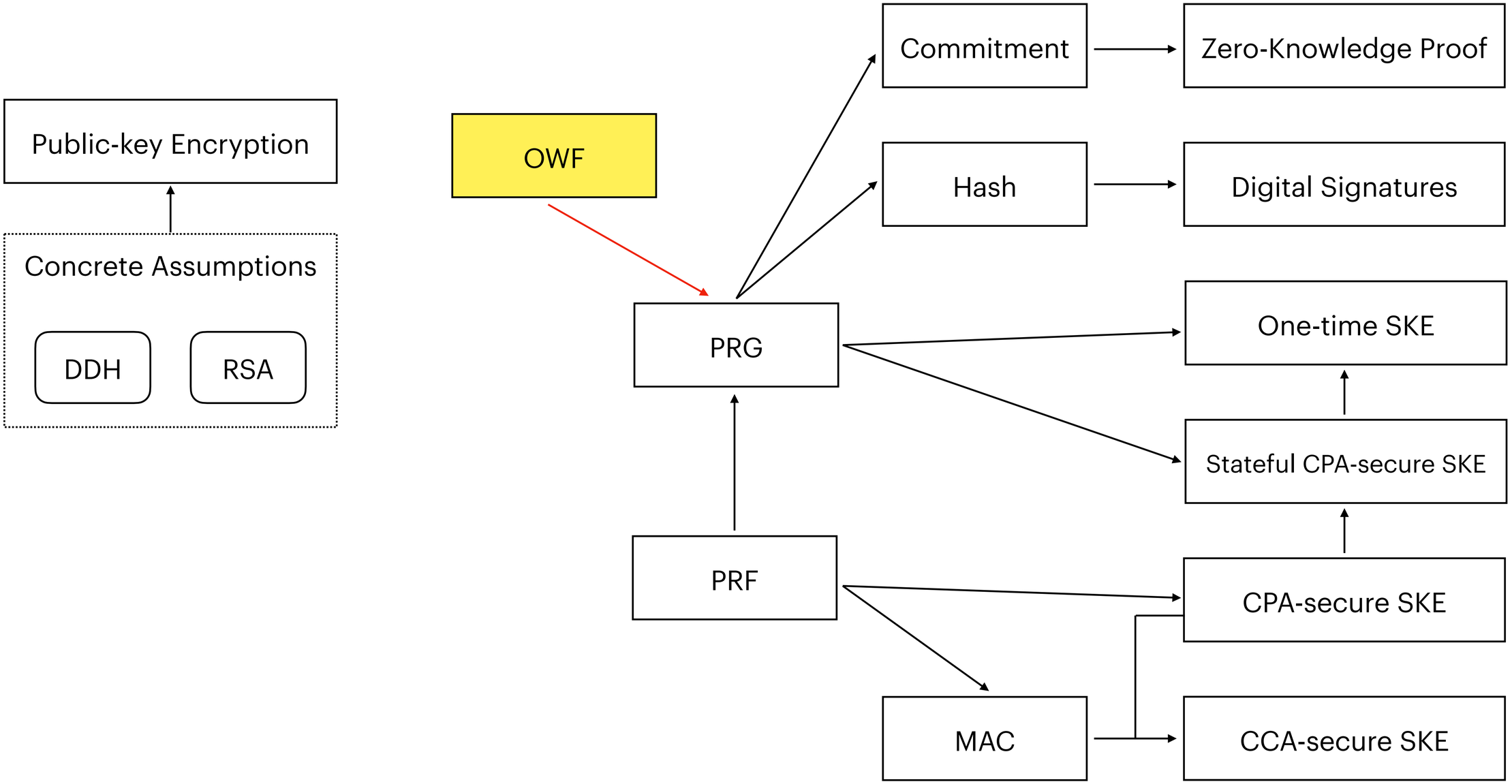
What is the minimum assumption X we need to make?



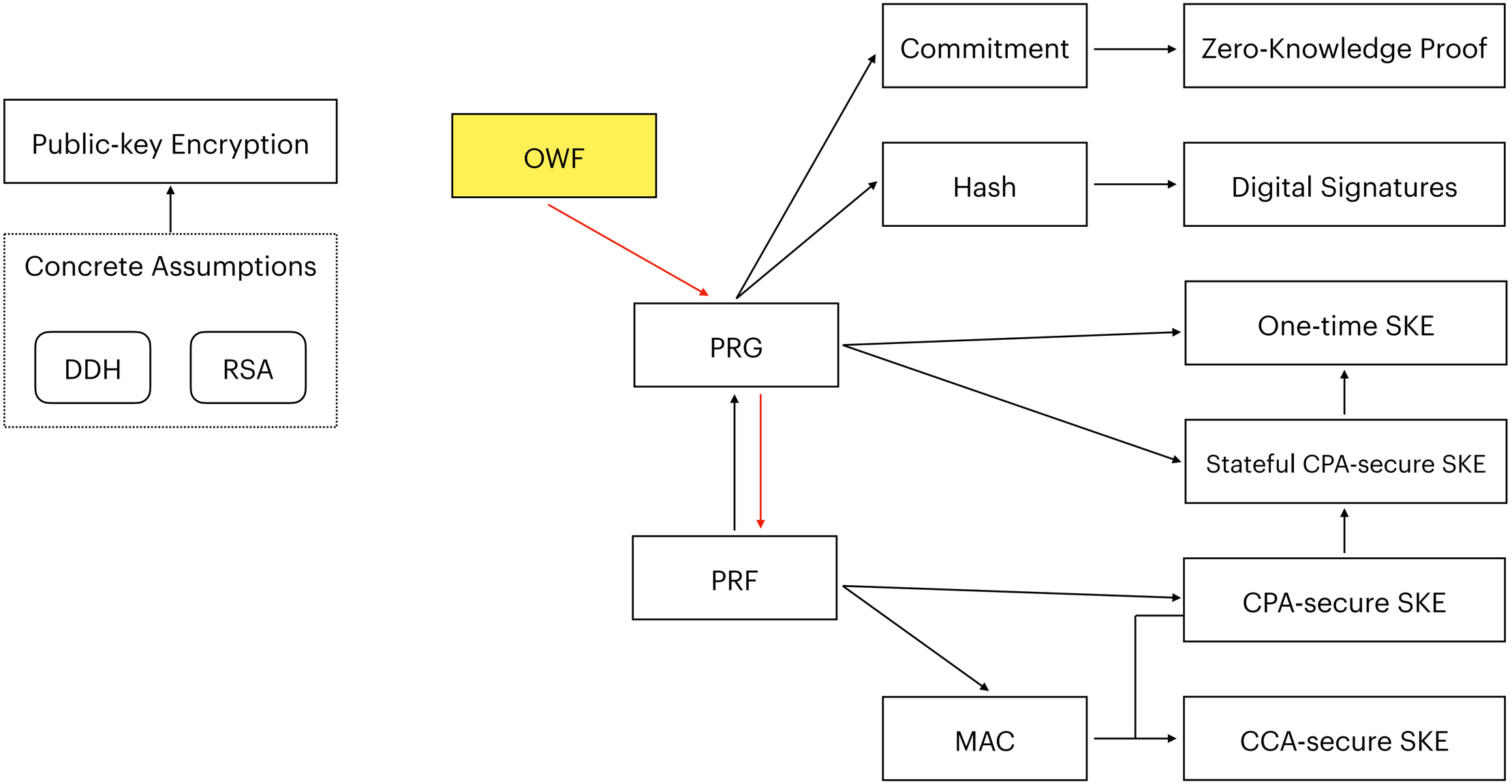
The Cryptography Landscape



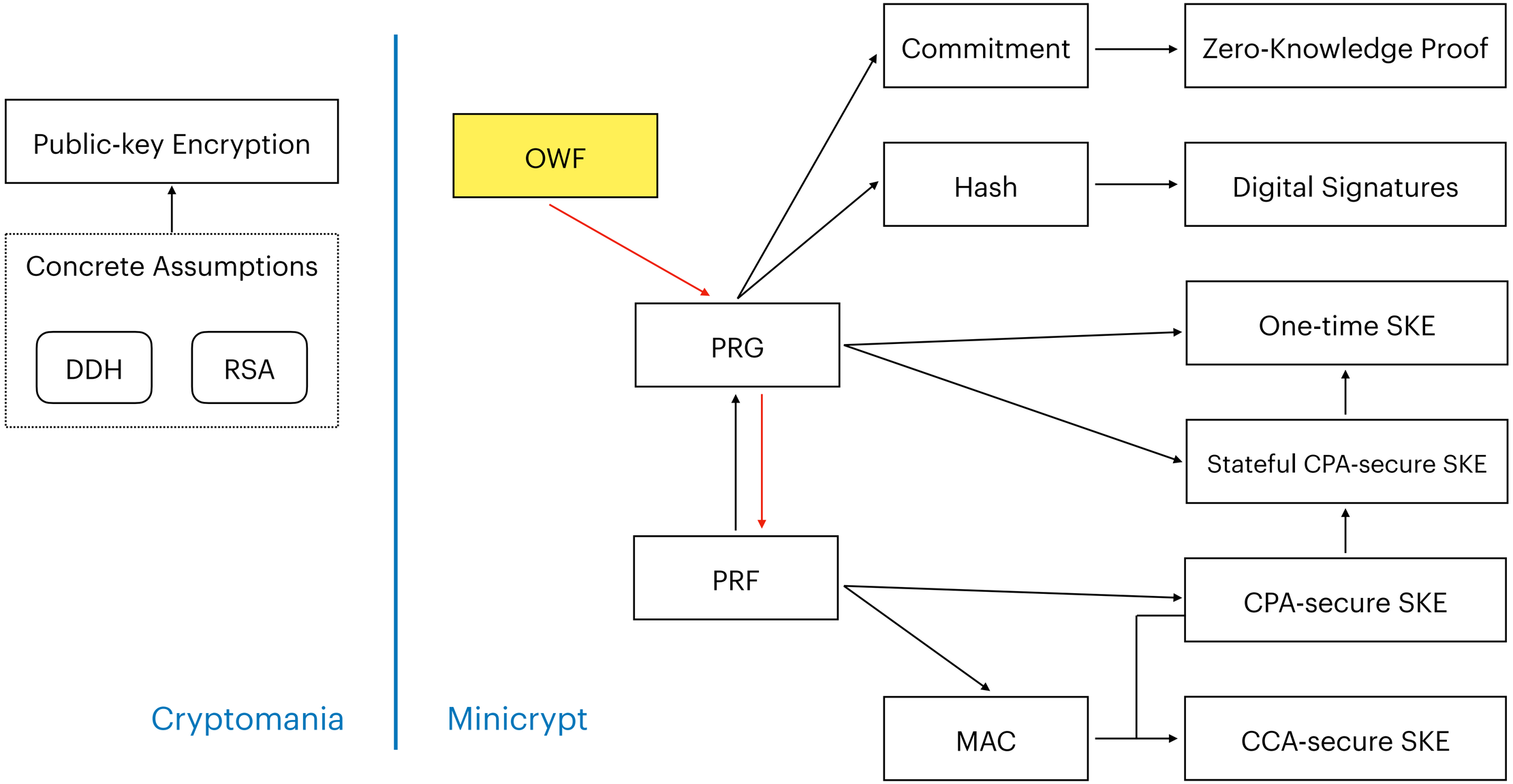
The Cryptography Landscape



The Cryptography Landscape



The Cryptography Landscape



What makes SKE Schemes Secure

- **Security:** No efficient adversary should be able to distinguish between encryptions of two different messages.
  - But it is hard to reason about all efficient adversaries.

What makes SKE Schemes Secure

- **Security:** No efficient adversary should be able to distinguish between encryptions of two different messages.
  - But it is hard to reason about all efficient adversaries.
- Consider Psueodrandom OTP: $\mathsf{Enc}(k, m) := G(k) \oplus m$

What makes SKE Schemes Secure

- **Security:** No efficient adversary should be able to distinguish between encryptions of two different messages.
  - But it is hard to reason about all efficient adversaries.
- Consider Psueodrandom OTP: $\mathsf{Enc}(k, m) := G(k) \oplus m$
  - How can an adversary attack the scheme?

What makes SKE Schemes Secure

- **Security:** No efficient adversary should be able to distinguish between encryptions of two different messages.
  - But it is hard to reason about all efficient adversaries.
- Consider Psueodrandom OTP: $\mathsf{Enc}(k, m) := G(k) \oplus m$
  - How can an adversary attack the scheme?
  - Our attacks boil down to searching for the key.

What makes SKE Schemes Secure

- **Security:** No efficient adversary should be able to distinguish between encryptions of two different messages.
  - But it is hard to reason about all efficient adversaries.
- Consider Psueodrandom OTP: $\mathsf{Enc}(k, m) := G(k) \oplus m$
  - How can an adversary attack the scheme?
  - Our attacks boil down to searching for the key.
- Often, this is the best attack we know against encryption schemes.

What makes SKE Schemes Secure

- **Security:** No efficient adversary should be able to distinguish between encryptions of two different messages.
  - But it is hard to reason about all efficient adversaries.
- Consider Psueodrandom OTP: $\mathsf{Enc}(k, m) := G(k) \oplus m$
  - How can an adversary attack the scheme?
  - Our attacks boil down to searching for the key.
- Often, this is the best attack we know against encryption schemes.

Security stems from searching for a solution being hard.

How Secure is an SKE Scheme

- Consider the following security definition for SKE:

  There exists a key k such that no efficient adversary can distinguish between $\mathsf{Enc}(k, m_0)$ and $\mathsf{Enc}(k, m_1)$.

How Secure is an SKE Scheme

- Consider the following security definition for SKE:

  There exists a key k such that no efficient adversary can distinguish between $\mathsf{Enc}(k, m_0)$ and $\mathsf{Enc}(k, m_1)$.

  - The encryption scheme might be insecure 99% of the time! Most of the keys don't ensure indistinguishability.

How Secure is an SKE Scheme

- Consider the following security definition for SKE:

  There exists a key k such that no efficient adversary can distinguish between $\mathsf{Enc}(k, m_0)$ and $\mathsf{Enc}(k, m_1)$.

  - The encryption scheme might be insecure 99% of the time! Most of the keys don't ensure indistinguishability.
  - **Worst-case hardness:** There exists an instance which is hard to solve.

How Secure is an SKE Scheme

- Consider the following security definition for SKE:

  There exists a key k such that no efficient adversary can distinguish between $\mathsf{Enc}(k, m_0)$ and $\mathsf{Enc}(k, m_1)$.

  - The encryption scheme might be insecure 99% of the time! Most of the keys don't ensure indistinguishability.
  - **Worst-case hardness:** There exists an instance which is hard to solve. But hard instances might be sparse.

How Secure is an SKE Scheme

- Consider the following security definition for SKE:

  There exists a key k such that no efficient adversary can distinguish between $\mathsf{Enc}(k, m_0)$ and $\mathsf{Enc}(k, m_1)$.

  - The encryption scheme might be insecure 99% of the time! Most of the keys don't ensure indistinguishability.
  - **Worst-case hardness:** There exists an instance which is hard to solve. But hard instances might be sparse.

- What we actually need

  If k is sampled uniformly at random, then no efficient adversary can distinguish between $\mathsf{Enc}(k, m_0)$ and $\mathsf{Enc}(k, m_1)$.

How Secure is an SKE Scheme

- Consider the following security definition for SKE:

  There exists a key k such that no efficient adversary can distinguish between $\mathsf{Enc}(k, m_0)$ and $\mathsf{Enc}(k, m_1)$.

  - The encryption scheme might be insecure 99% of the time! Most of the keys don't ensure indistinguishability.
  - **Worst-case hardness:** There exists an instance which is hard to solve. But hard instances might be sparse.

- What we actually need

  If k is sampled uniformly at random, then no efficient adversary can distinguish between $\mathsf{Enc}(k, m_0)$ and $\mathsf{Enc}(k, m_1)$.

  - **Average-case hardness:** Hard instances are dense.

How Secure is an SKE Scheme

- Consider the following security definition for SKE:

  There exists a key k such that no efficient adversary can distinguish between $\mathsf{Enc}(k, m_0)$ and $\mathsf{Enc}(k, m_1)$.

  - The encryption scheme might be insecure 99% of the time! Most of the keys don't ensure indistinguishability.
  - **Worst-case hardness:** There exists an instance which is hard to solve. But hard instances might be sparse.

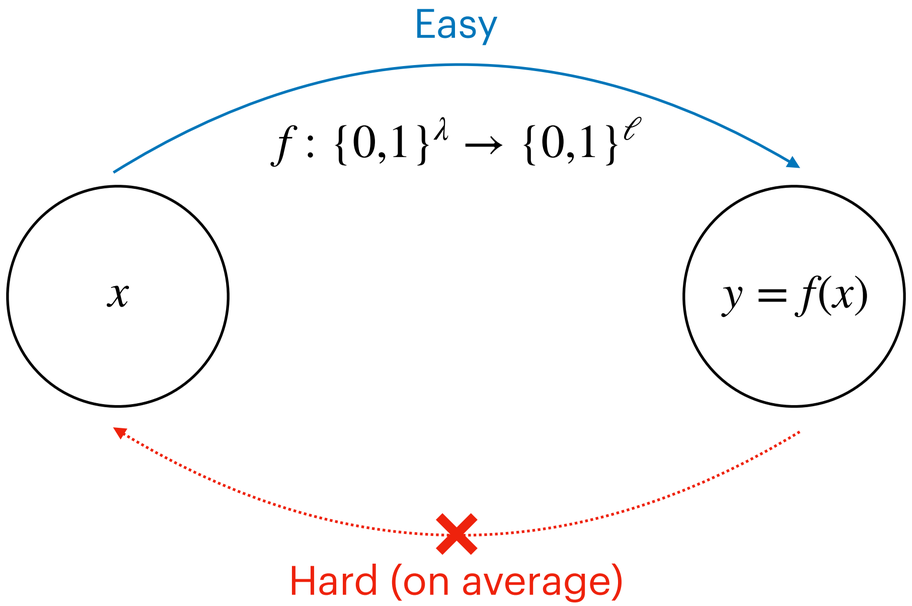
- What we actually need

  If k is sampled uniformly at random, then no efficient adversary can distinguish between $\mathsf{Enc}(k, m_0)$ and $\mathsf{Enc}(k, m_1)$.

  - **Average-case hardness:** Hard instances are dense.

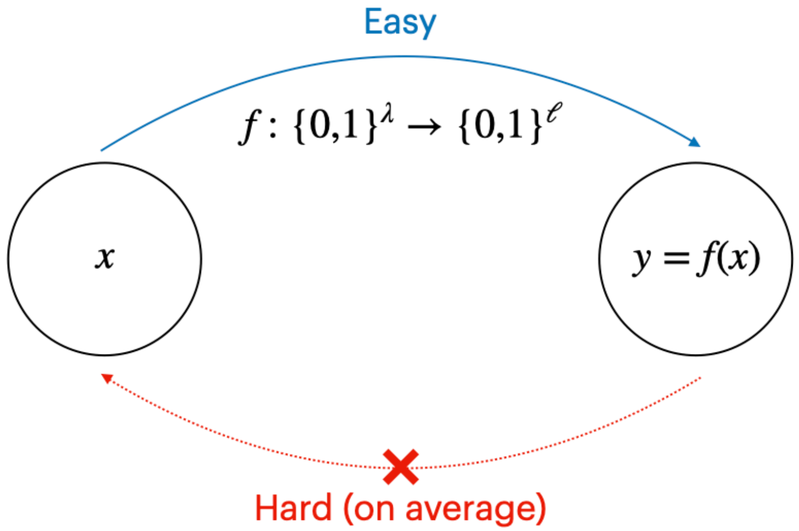
Security stems from searching for a solution being hard on average.

One-Way Functions



One-Way Functions

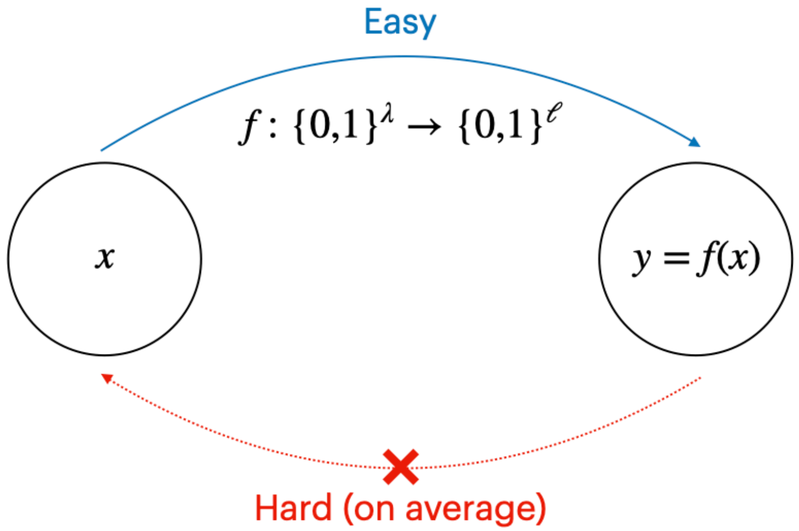
One-Way Function



One-Way Functions

One-Way Function

A family of functions $\{f_\lambda : \{0,1\}^\lambda \to \{0,1\}^{\ell}\}_\lambda$ is a **one-way function** if it satisfies the following properties.

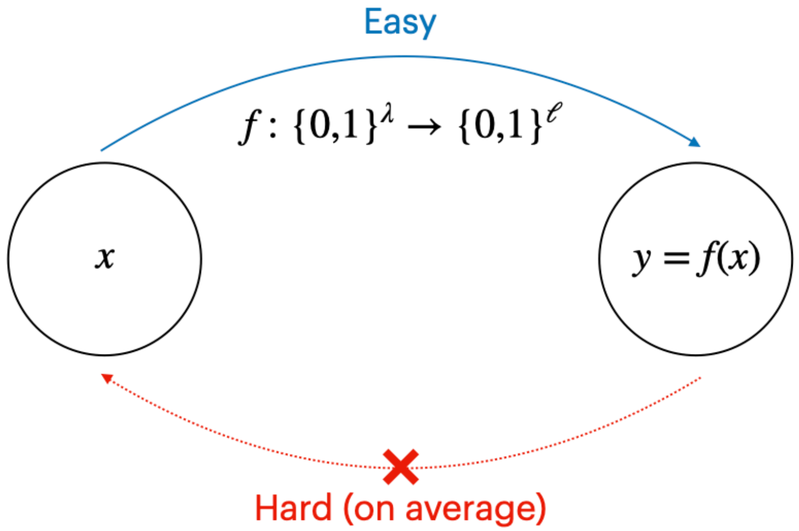


One-Way Functions

One-Way Function

A family of functions $\{f_\lambda : \{0,1\}^\lambda \to \{0,1\}^{\ell}\}_\lambda$ is a **one-way function** if it satisfies the following properties.

- **Easy to compute:** For all $\lambda \in \mathbb{N}$, f_λ can be computed in polynomial time.



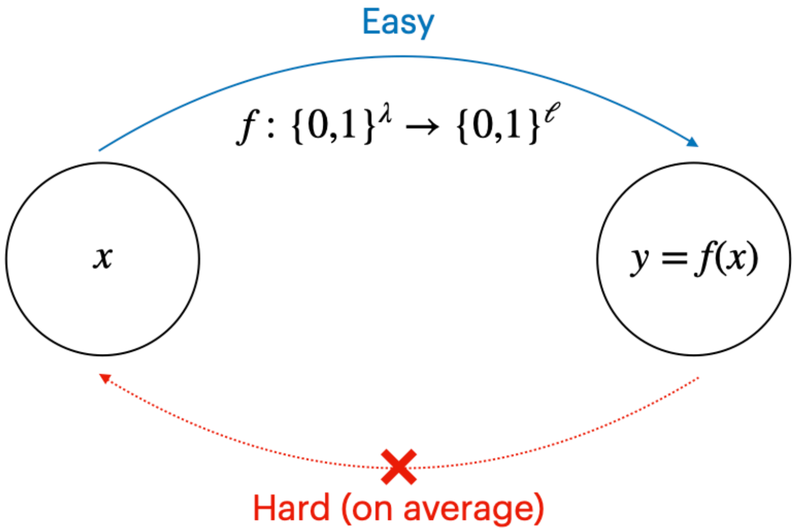
One-Way Functions

One-Way Function

A family of functions $\{f_\lambda : \{0,1\}^\lambda \to \{0,1\}^\ell\}_\lambda$ is a **one-way function** if it satisfies the following properties.

- **Easy to compute:** For all $\lambda \in \mathbb{N}$, f_λ can be computed in polynomial time.
- **Hard to invert:** For all NUPPT adversaries A, there exists a negligible function **negl**, such that for all $\lambda \in \mathbb{N}$

$$\Pr\left[A(1^\lambda, y) = x : \begin{array}{c} x \xleftarrow{\$} \{0,1\}^\lambda \\ y := f(x) \end{array}\right] \leq \text{negl}(\lambda).$$



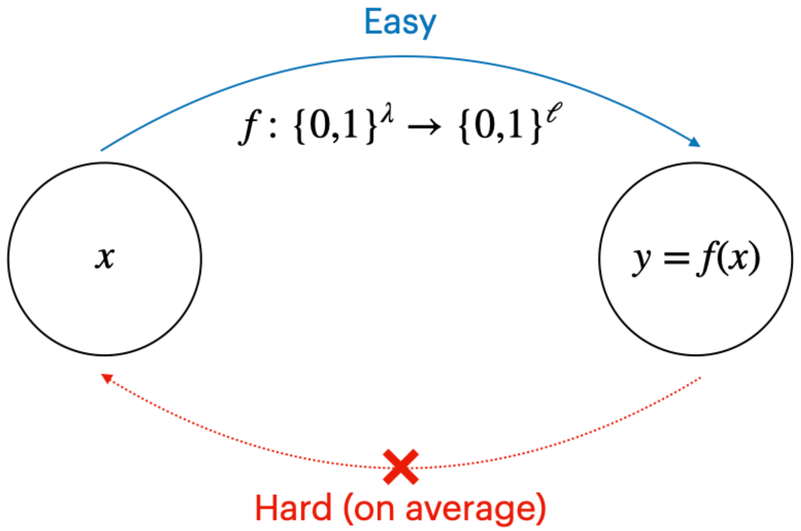
One-Way Functions

One-Way Function

A family of functions $\{f_\lambda : \{0,1\}^\lambda \to \{0,1\}^\ell\}_\lambda$ is a **one-way function** if it satisfies the following properties.

- **Easy to compute:** For all $\lambda \in \mathbb{N}$, f_λ can be computed in polynomial time.
- **Hard to invert:** For all NUPPT adversaries A, there exists a negligible function **negl**, such that for all $\lambda \in \mathbb{N}$

$$\Pr\left[A(1^\lambda, y) = x : \begin{array}{c} x \xleftarrow{\$} \{0,1\}^\lambda \\ y := f(x) \end{array}\right] \leq \mathsf{negl}(\lambda).$$



Does $f(x) = 0$ satisfy this definition?

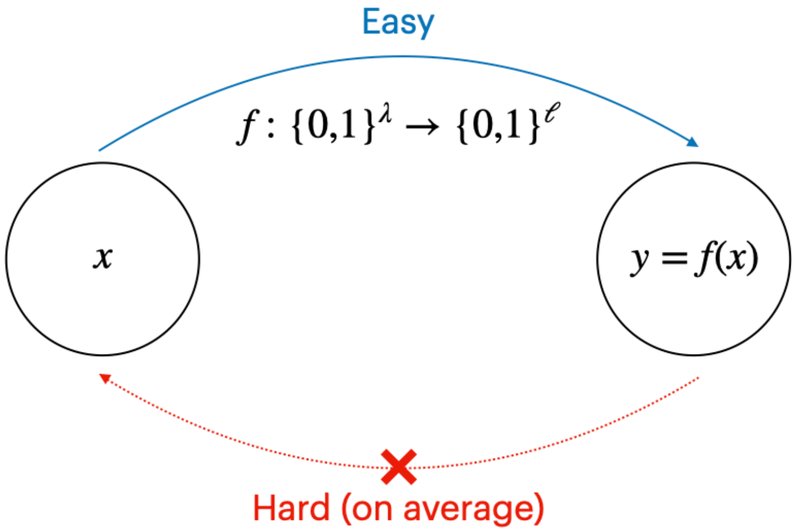
One-Way Functions

One-Way Function

A family of functions $\{f_\lambda : \{0,1\}^\lambda \to \{0,1\}^\ell\}_\lambda$ is a **one-way function** if it satisfies the following properties.

- **Easy to compute:** For all $\lambda \in \mathbb{N}$, f_λ can be computed in polynomial time.
- **Hard to invert:** For all NUPPT adversaries A, there exists a negligible function **negl**, such that for all $\lambda \in \mathbb{N}$

$$\Pr\left[A(1^\lambda, y) = x : \begin{array}{c} x \xleftarrow{\$} \{0,1\}^\lambda \\ y := f(x) \end{array}\right] \leq \text{negl}(\lambda).$$



Does $f(x) = 0$ satisfy this definition?

Yes! Even an unbounded adversary cannot find the inverse with high probability.

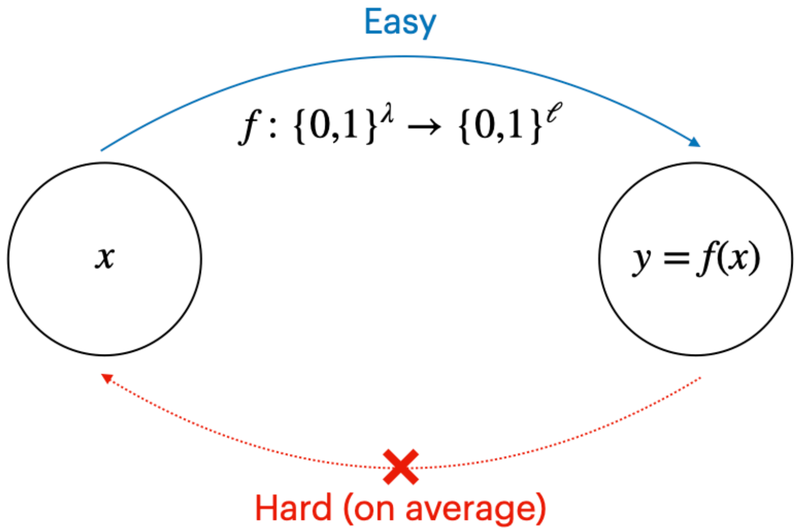
One-Way Functions

One-Way Function

A family of functions $\{f_\lambda : \{0,1\}^\lambda \to \{0,1\}^{\ell}\}_\lambda$ is a **one-way function** if it satisfies the following properties.

- **Easy to compute:** For all $\lambda \in \mathbb{N}$, f_λ can be computed in polynomial time.
- **Hard to invert:** For all NUPPT adversaries A, there exists a negligible function **negl**, such that for all $\lambda \in \mathbb{N}$

$$\Pr\left[A(1^\lambda, y) = x \leftarrow \{0,1\}^\lambda \;:\; y = f(x)\right] \leq \text{negl}(\lambda).$$



Does $f(x) = 0$ satisfy this definition?

Yes! Even an unbounded adversary cannot find the inverse with high probability.

This definition is not useful.

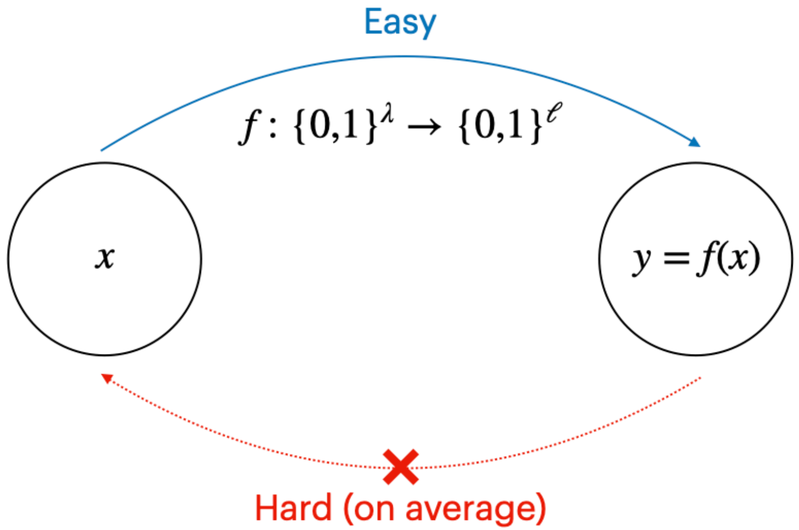
One-Way Functions

One-Way Function

A family of functions $\{f_\lambda : \{0,1\}^\lambda \to \{0,1\}^{\ell}\}_\lambda$ is a **one-way function** if it satisfies the following properties.

- **Easy to compute:** For all $\lambda \in \mathbb{N}$, f_λ can be computed in polynomial time.
- **Hard to invert:** For all NUPPT adversaries A, there exists a negligible function **negl**, such that for all $\lambda \in \mathbb{N}$

$$\Pr\left[A(1^\lambda, y) = x \leftarrow \{0,1\}^\lambda : y = f(x)\right] \leq \text{negl}(\lambda).$$



Does $f(x) = 0$ satisfy this definition?

Yes! Even an unbounded adversary cannot find the inverse with high probability.

This definition is not useful.

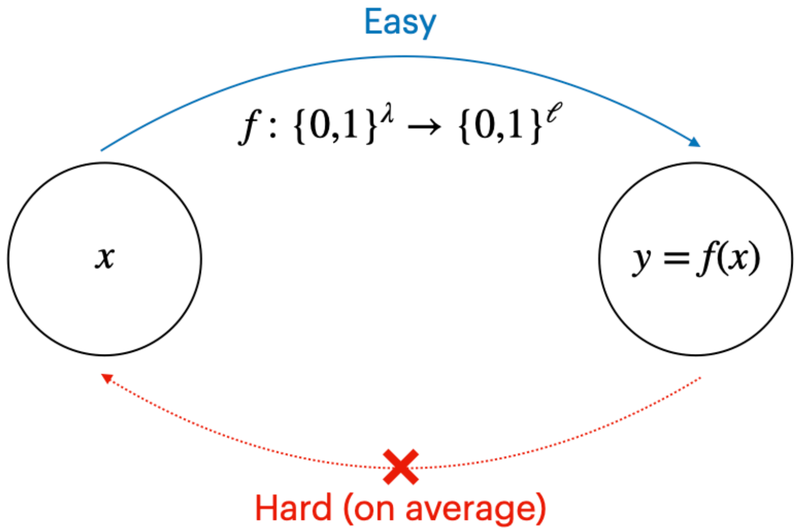
We want to capture that it is hard to find **an** inverse of y.

One-Way Functions

One-Way Function

A family of functions $\{f_\lambda : \{0,1\}^\lambda \to \{0,1\}^{\ell}\}_\lambda$ is a **one-way function** if it satisfies the following properties.

- **Easy to compute:** For all $\lambda \in \mathbb{N}$, f_λ can be computed in polynomial time.
- **Hard to invert:** For all NUPPT adversaries A, there exists a negligible function **negl**, such that for all $\lambda \in \mathbb{N}$



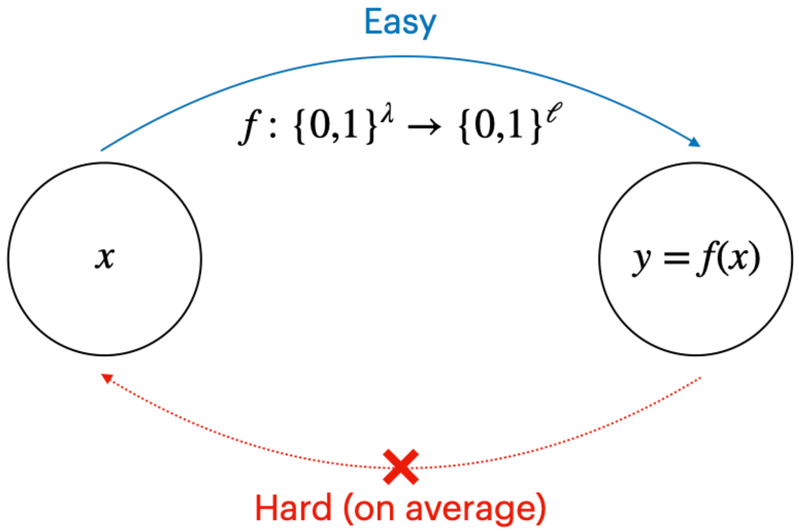
One-Way Functions

One-Way Function

A family of functions $\{f_\lambda : \{0,1\}^\lambda \to \{0,1\}^{\ell}\}_\lambda$ is a **one-way function** if it satisfies the following properties.

- **Easy to compute:** For all $\lambda \in \mathbb{N}$, f_λ can be computed in polynomial time.
- **Hard to invert:** For all NUPPT adversaries A, there exists a negligible function **negl**, such that for all $\lambda \in \mathbb{N}$

$$\Pr\left[f(x') = y : \begin{array}{c} x \xleftarrow{\$} \{0,1\}^\lambda \\ y := f(x) \\ x' \leftarrow A(1^\lambda, y) \end{array}\right] \leq \mathsf{negl}(\lambda).$$



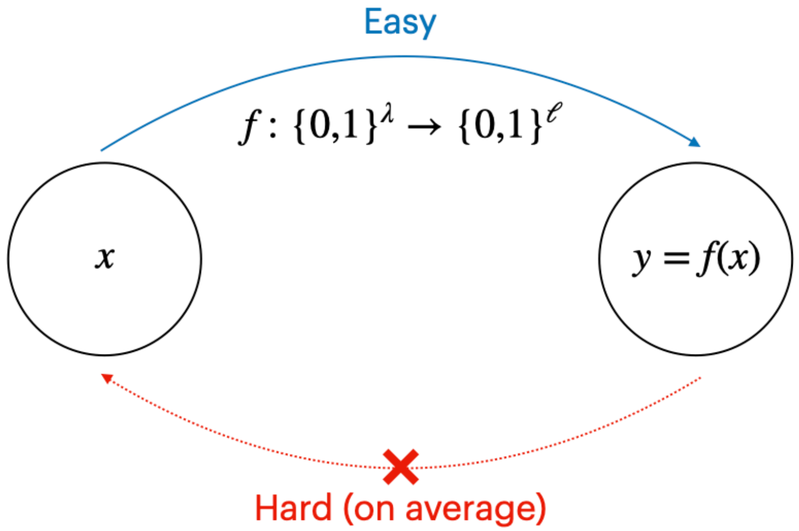
One-Way Functions

One-Way Function

A family of functions $\{f_\lambda : \{0,1\}^\lambda \to \{0,1\}^{\ell}\}_\lambda$ is a **one-way function** if it satisfies the following properties.

- **Easy to compute:** For all $\lambda \in \mathbb{N}$, f_λ can be computed in polynomial time.
- **Hard to invert:** For all NUPPT adversaries A, there exists a negligible function **negl**, such that for all $\lambda \in \mathbb{N}$

$$\Pr\left[f(x') = y : \begin{array}{c} x \xleftarrow{\$} \{0,1\}^\lambda \\ y := f(x) \\ x' \leftarrow A(1^\lambda, y) \end{array}\right] \leq \mathsf{negl}(\lambda).$$



One-way Permutation: One-one OWF with $\ell = \lambda$.

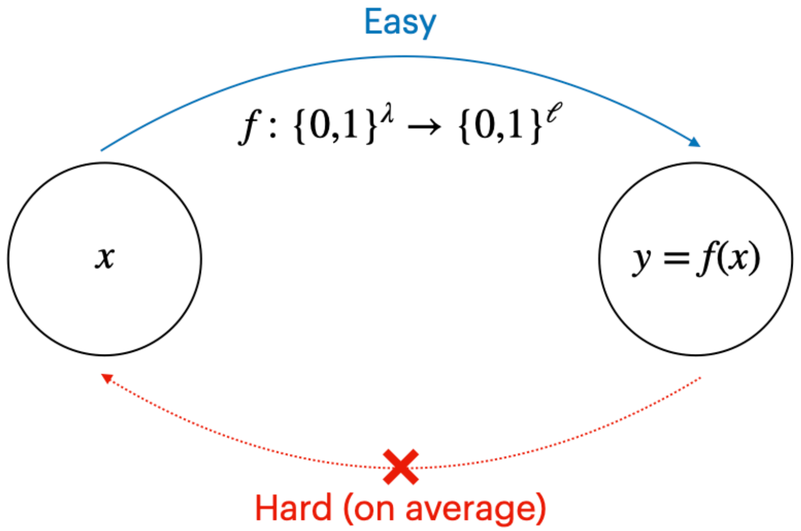
One-Way Functions

One-Way Function

A family of functions $\{f_\lambda : \{0,1\}^\lambda \to \{0,1\}^{\ell}\}_\lambda$ is a **one-way function** if it satisfies the following properties.

- **Easy to compute:** For all $\lambda \in \mathbb{N}$, f_λ can be computed in polynomial time.
- **Hard to invert:** For all NUPPT adversaries A, there exists a negligible function **negl**, such that for all $\lambda \in \mathbb{N}$

$$\Pr\left[f(x') = y : \begin{array}{c} x \xleftarrow{\$} \{0,1\}^\lambda \\ y := f(x) \\ x' \leftarrow A(1^\lambda, y) \end{array}\right] \leq \mathsf{negl}(\lambda).$$



One-way Permutation: One-one OWF with $\ell = \lambda$.

Each image y has a unique pre-image x

One-Way Functions: Properties

One-Way Function

A family of functions $\{f_\lambda : \{0,1\}^\lambda \to \{0,1\}^{\ell}\}_\lambda$ is a **one-way function** if it satisfies the following properties.

- **Easy to compute:** For all $\lambda \in \mathbb{N}$, f_λ can be computed in polynomial time.
- **Hard to invert:** For all NUPPT adversaries A, there exists a negligible function **negl**, such that for all $\lambda \in \mathbb{N}$

$$\Pr\left[f(x') = y : \begin{array}{c} x \xleftarrow{\$} \{0,1\}^\lambda \\ y := f(x) \\ x' \leftarrow A(1^\lambda, y) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

One-Way Functions: Properties

One-Way Function

A family of functions $\{f_\lambda : \{0,1\}^\lambda \to \{0,1\}^{\ell}\}_\lambda$ is a **one-way function** if it satisfies the following properties.

- **Easy to compute:** For all $\lambda \in \mathbb{N}$, f_λ can be computed in polynomial time.
- **Hard to invert:** For all NUPPT adversaries A, there exists a negligible function **negl**, such that for all $\lambda \in \mathbb{N}$

$$\Pr\left[f(x') = y : \begin{array}{c} x \xleftarrow{\$} \{0,1\}^\lambda \\ y := f(x) \\ x' \leftarrow A(1^\lambda, y) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

- **Average-case Hardness:** $y = f(x)$ for a uniformly random x is hard to invert.

One-Way Functions: Properties

One-Way Function

A family of functions $\{f_\lambda : \{0,1\}^\lambda \to \{0,1\}^{\ell}\}_\lambda$ is a **one-way function** if it satisfies the following properties.

- **Easy to compute:** For all $\lambda \in \mathbb{N}$, f_λ can be computed in polynomial time.
- **Hard to invert:** For all NUPPT adversaries A, there exists a negligible function **negl**, such that for all $\lambda \in \mathbb{N}$

$$\Pr\left[f(x') = y : \begin{array}{c} x \xleftarrow{\$} \{0,1\}^\lambda \\ y := f(x) \\ x' \leftarrow A(1^\lambda, y) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

- **Average-case Hardness:** $y = f(x)$ for a uniformly random x is hard to invert.
- **Efficient Generation:** Efficiently generate a hard instance y with a solution x.

One-Way Functions: Properties

One-Way Function

A family of functions $\{f_\lambda : \{0,1\}^\lambda \to \{0,1\}^{\ell}\}_\lambda$ is a **one-way function** if it satisfies the following properties.

- **Easy to compute:** For all $\lambda \in \mathbb{N}$, f_λ can be computed in polynomial time.
- **Hard to invert:** For all NUPPT adversaries A, there exists a negligible function **negl**, such that for all $\lambda \in \mathbb{N}$

$$\Pr\left[f(x') = y : \begin{array}{c} x \xleftarrow{\$} \{0,1\}^\lambda \\ y := f(x) \\ x' \leftarrow A(1^\lambda, y) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

- **Average-case Hardness:** $y = f(x)$ for a uniformly random x is hard to invert.
- **Efficient Generation:** Efficiently generate a hard instance y with a solution x.
- **Efficient Verification:** Given x' it is efficient to verify if $f(x') = y$.

Do One-Way Functions Exist

Do One-Way Functions Exist

- Existence of OWF implies $P \neq NP$ (Inverting OWF is a hard problem, solution can be verified efficiently).

Do One-Way Functions Exist

- Existence of OWF implies $P \neq NP$ (Inverting OWF is a hard problem, solution can be verified efficiently).
    - But $P \neq NP$ is not sufficient to argue existence of OWF ($P \neq NP$ captures worst-case hardness, OWF requires average-case hardness).

Do One-Way Functions Exist

- Existence of OWF implies P ≠ NP (Inverting OWF is a hard problem, solution can be verified efficiently).
  - But P ≠ NP is not sufficient to argue existence of OWF (P ≠ NP captures worst-case hardness, OWF requires average-case hardness).

Discrete Logarithm Assumption

Let $(G, \cdot)$ be a cyclic group of order p with generator g, then for every NUPPT adversary A, there exists a negligible function negl such that

$$\Pr\left[a = a' : \begin{array}{c} a \xleftarrow{\$} \mathbb{Z}_p \\ a' \leftarrow A(G, p, g, g^a) \end{array}\right] \leq \text{negl}(\lambda)\,.$$

Do One-Way Functions Exist?

- Existence of OWF implies P ≠ NP (Inverting OWF is a hard problem, solution can be verified efficiently).
  - But P ≠ NP is not sufficient to argue existence of OWF (P ≠ NP captures worst-case hardness, OWF requires average-case hardness).

RSA Assumption

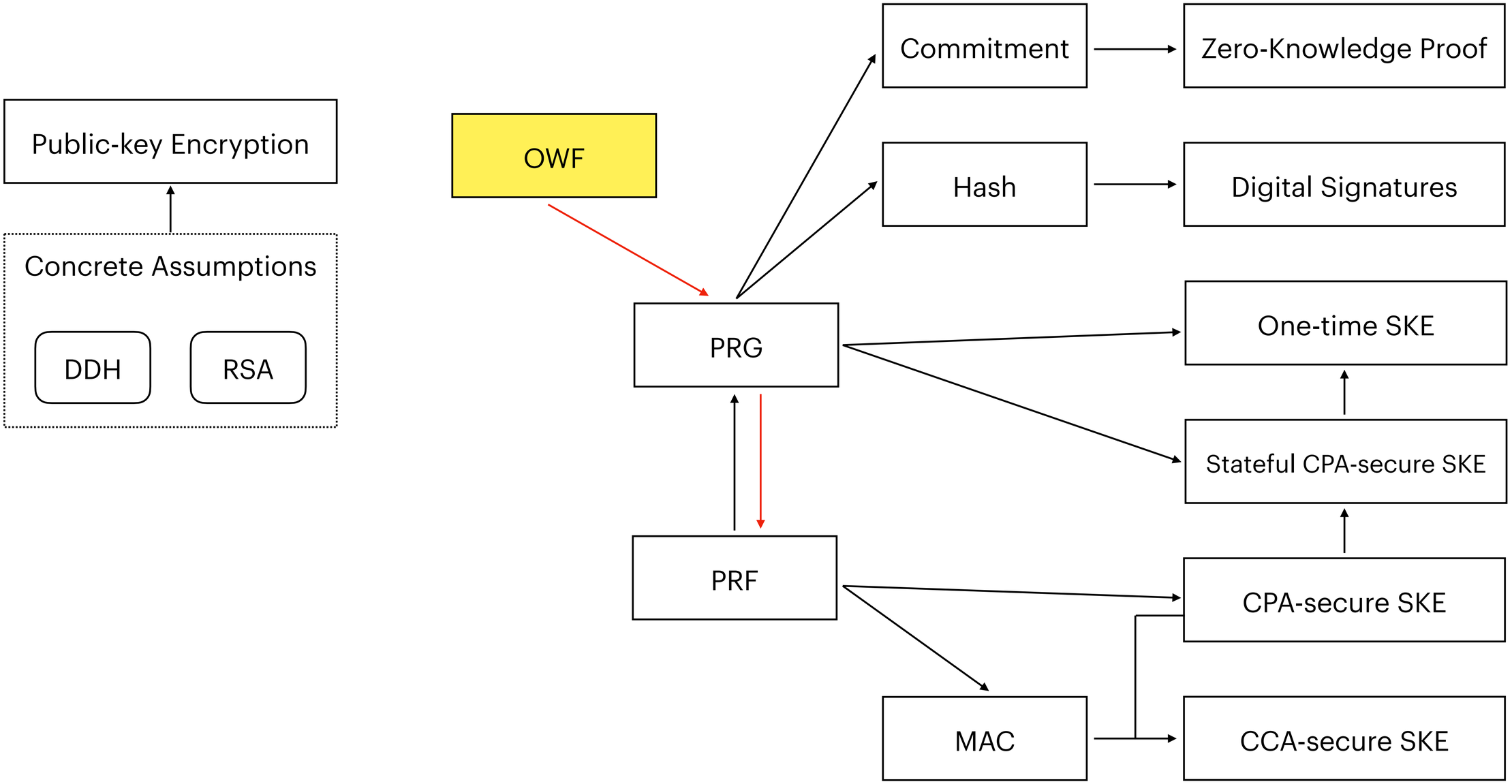
Let $\mathsf{GenRSA}(1^\lambda) \to (p, q)$ be a PPT algorithm that outputs two distinct λ-bit odd primes. Then for any non-uniform PPT adversary A

$$\Pr\left[A(N, e, x^e \bmod N) = x : \begin{array}{c}(p,q) \leftarrow \mathsf{GenRSA}(1^\lambda) \\ N = pq \\ e \xleftarrow{\$} \mathbb{Z}^\times_{\varphi(N)} \\ x \xleftarrow{\$} \mathbb{Z}^\times_N\end{array}\right] \leq \mathsf{negl}(\lambda)\,.$$

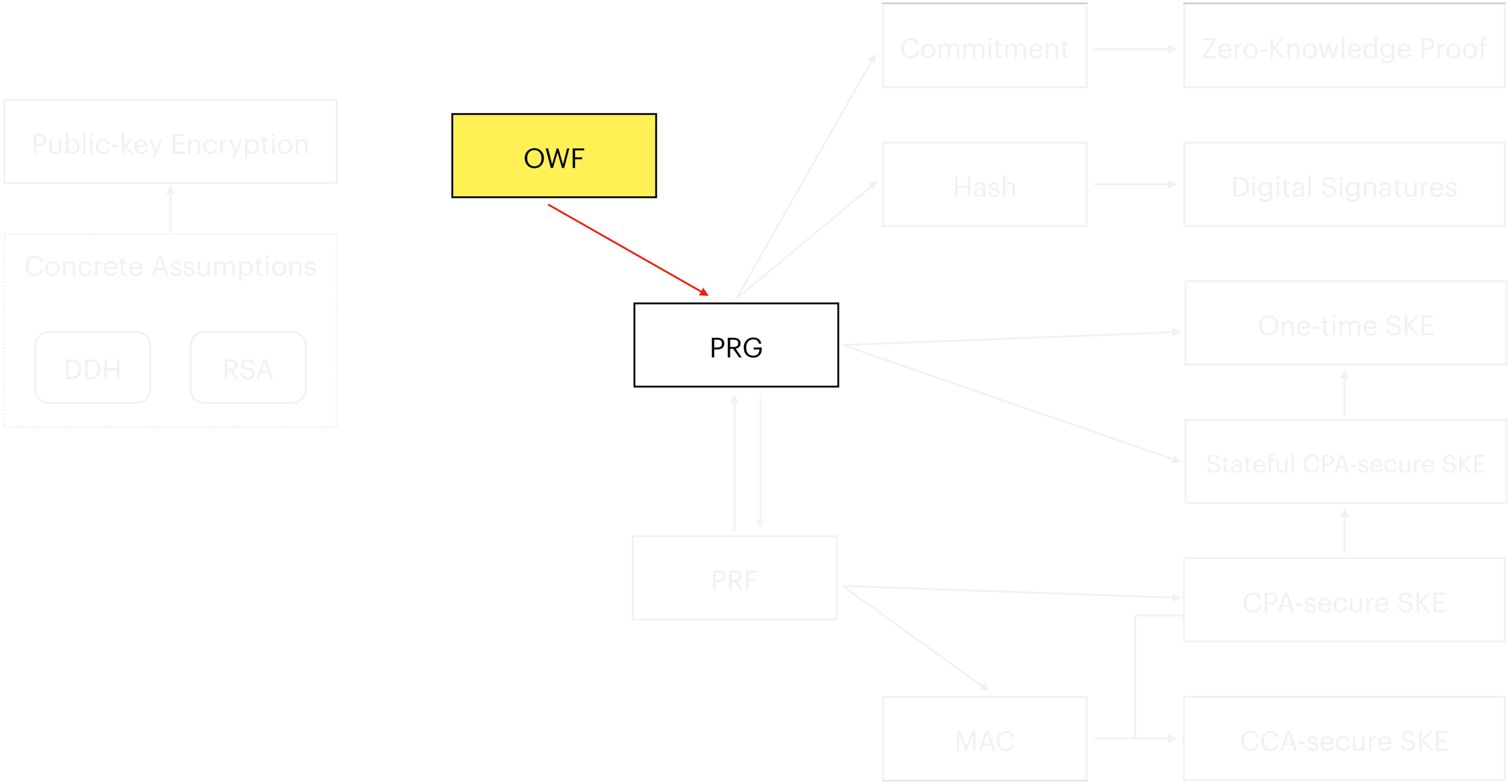
Do One-Way Functions Exist?

- Existence of OWF implies $P \neq NP$ (Inverting OWF is a hard problem, solution can be verified efficiently).
    - But $P \neq NP$ is not sufficient to argue existence of OWF ($P \neq NP$ captures worst-case hardness, OWF requires average-case hardness).
- In general, most cryptographic primitives imply OWF.

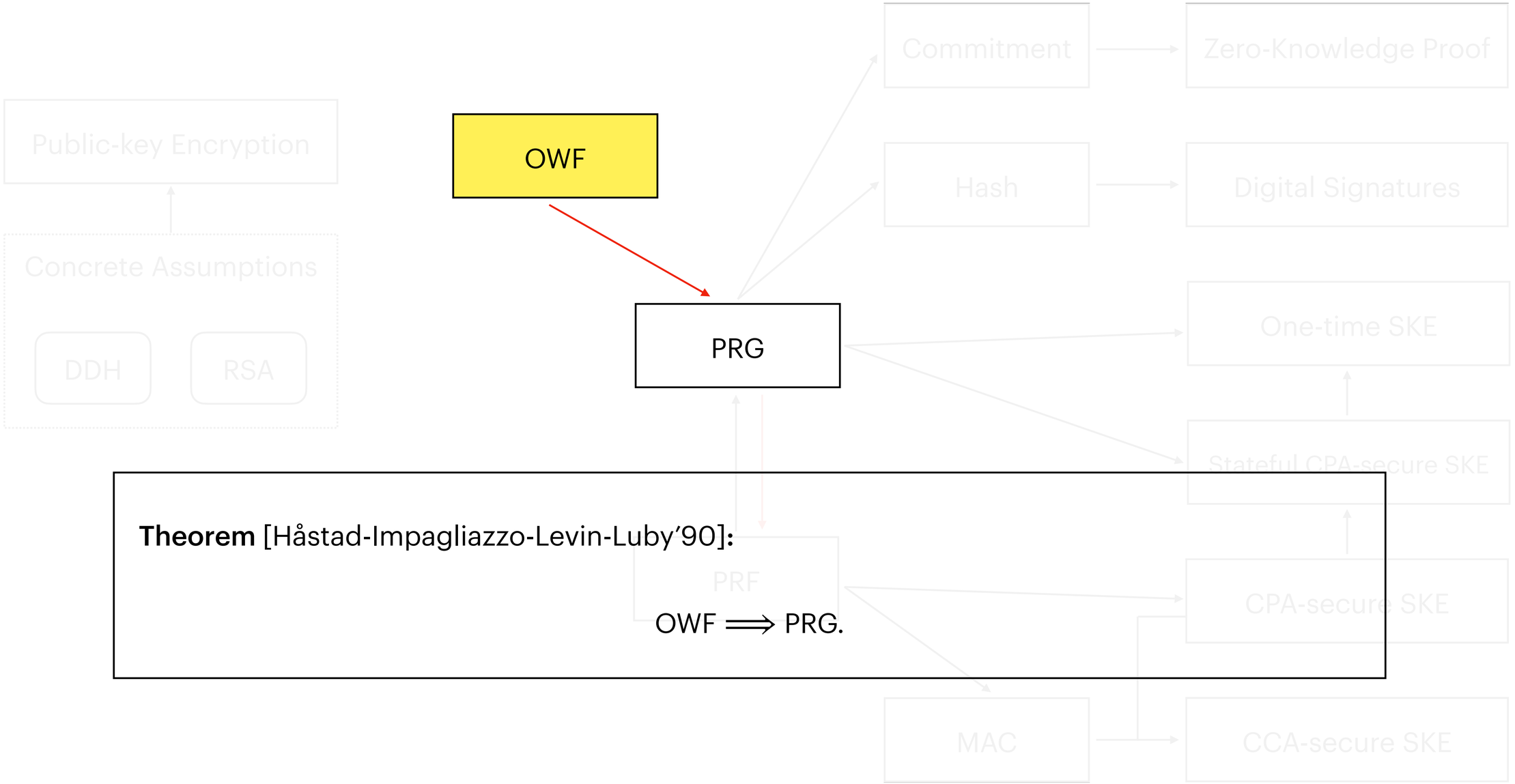
The Cryptography Landscape



The Cryptography Landscape



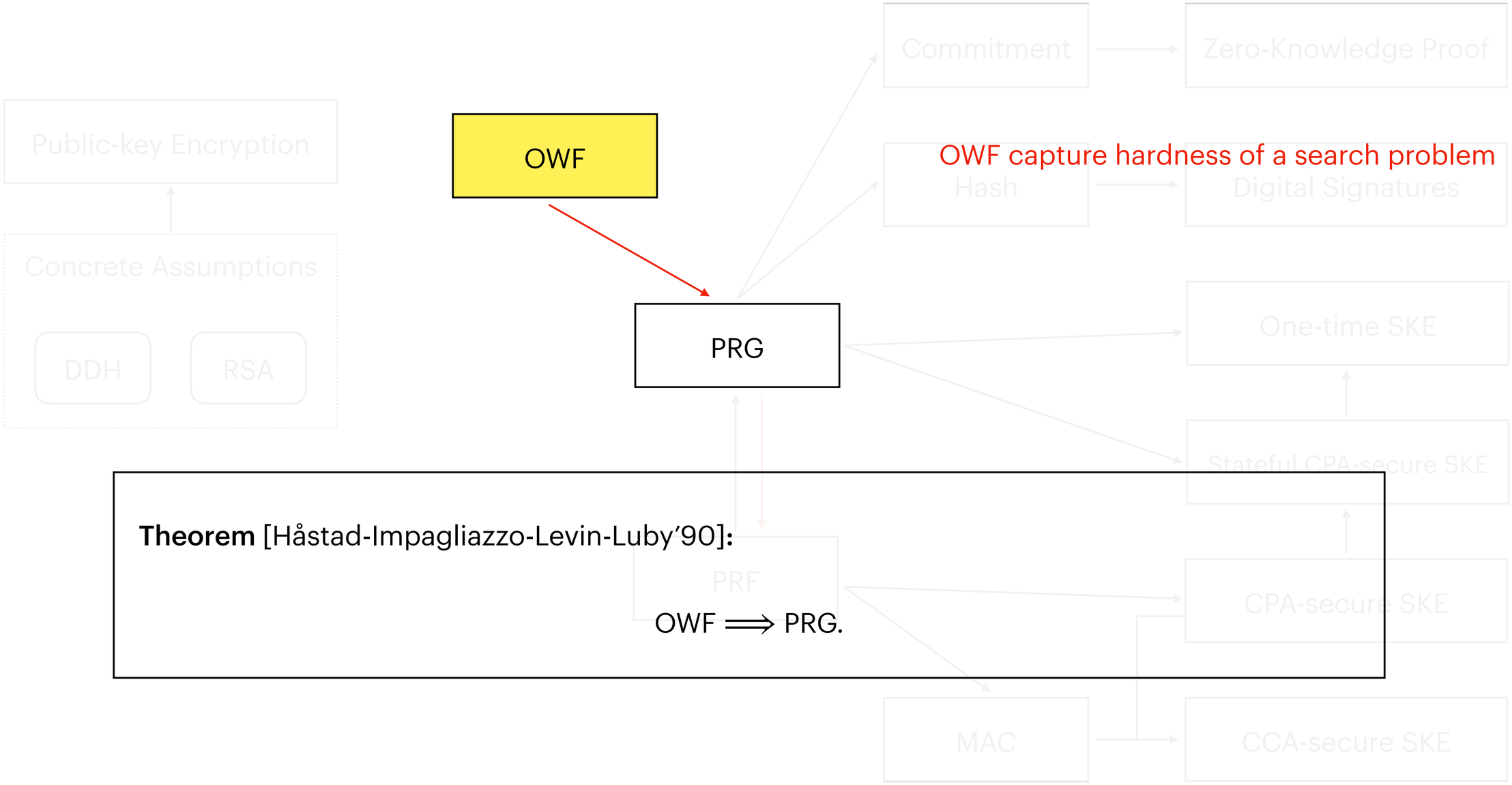
The Cryptography Landscape



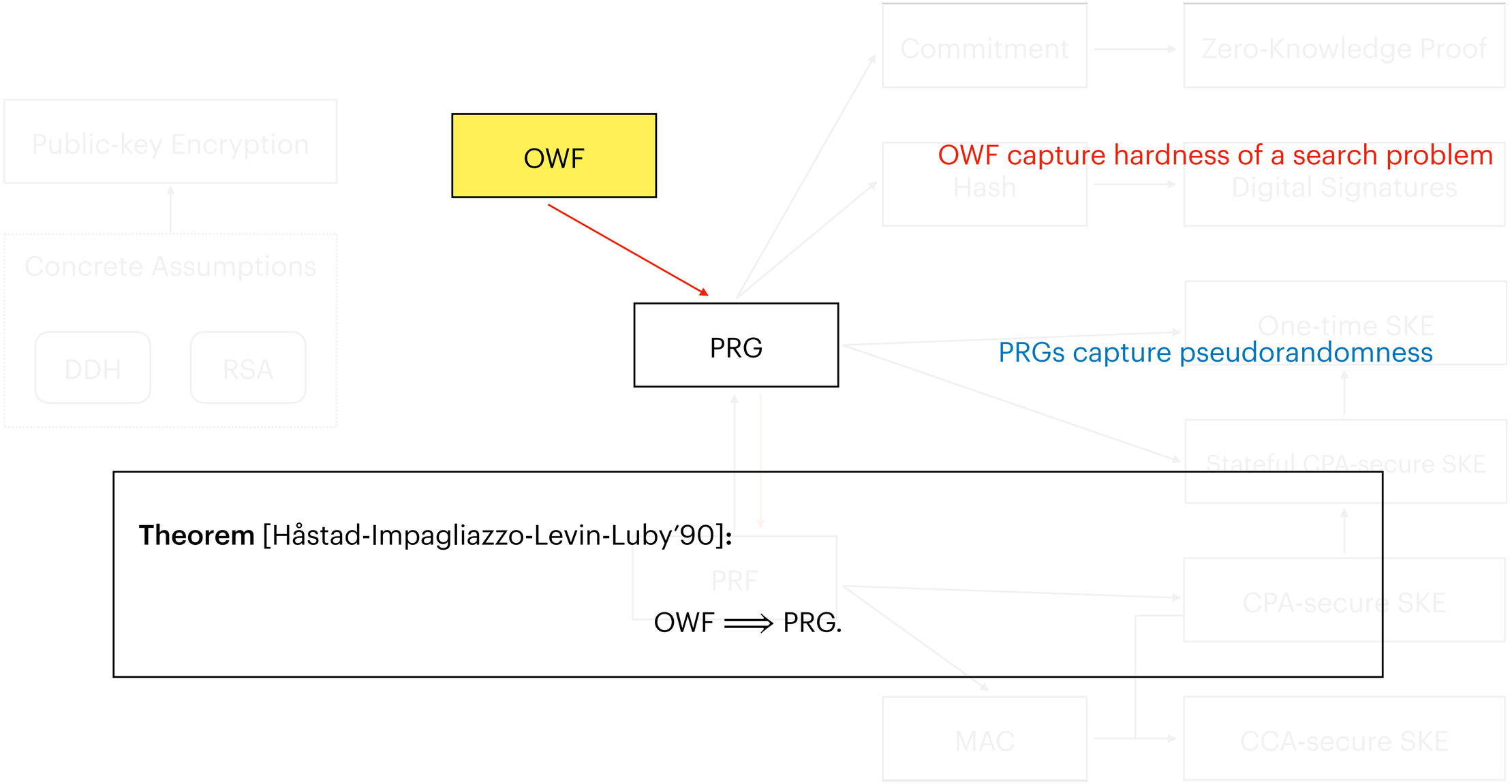
Theorem [Håstad-Impagliazzo-Levin-Luby'90]**:**

OWF $\Longrightarrow$ PRG.

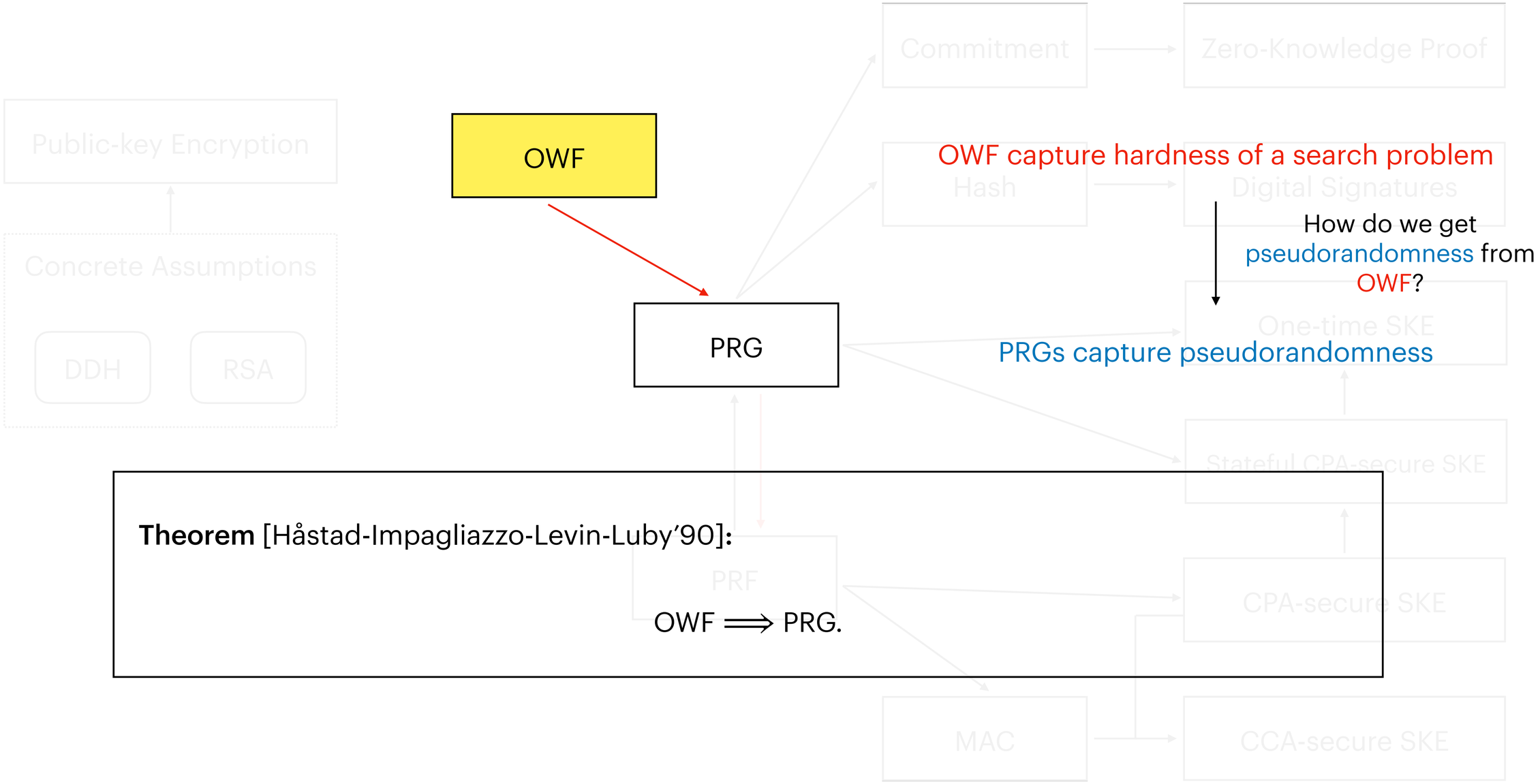
The Cryptography Landscape



The Cryptography Landscape



The Cryptography Landscape



What do OWF Hide

- We need to get pseudorandomness from OWF.

What do OWF Hide

- We need to get pseudorandomness from OWF.
- Given $y = f(x)$, can we leverage randomness of x?

What do OWF Hide

- We need to get pseudorandomness from OWF.
- Given $y = f(x)$, can we leverage randomness of x?
- OWFs guarantee that $f(x)$ hides x, but nothing more!

What do OWF Hide

- We need to get pseudorandomness from OWF.
- Given $y = f(x)$, can we leverage randomness of x?
- OWFs guarantee that $f(x)$ hides x, but nothing more!
  - If f is a OWF, then $g(x \| b) = f_{\lambda-1}(x) \| b$ is also a OWF.

What do OWF Hide

- We need to get pseudorandomness from OWF.
- Given $y = f(x)$, can we leverage randomness of x?
- OWFs guarantee that $f(x)$ hides x, but nothing more!
  - If f is a OWF, then $g(x \| b) = f_{\lambda-1}(x) \| b$ is also a OWF.
  - We cannot leverage randomness of b given $g(x)$.

What do OWF Hide

- We need to get pseudorandomness from OWF.
- Given $y = f(x)$, can we leverage randomness of x?
- OWFs guarantee that $f(x)$ hides x, but nothing more!
  - If f is a OWF, then $g(x \| b) = f_{\lambda-1}(x) \| b$ is also a OWF.
  - We cannot leverage randomness of b given $g(x)$.

Is there any non-trivial (non-identity) function of x, even 1 bit, that OWFs hide?

Hard-Core Predicate

- A **hard-core predicate** for a OWF $f : \{0,1\}^\lambda \to \{0,1\}^\ell$ is a function $\mathsf{hc} : \{0,1\}^\lambda \to \{0,1\}$ such that

Hard-Core Predicate

- A **hard-core predicate** for a OWF $f : \{0,1\}^\lambda \to \{0,1\}^\ell$ is a function $\mathsf{hc} : \{0,1\}^\lambda \to \{0,1\}$ such that
  - It takes the same input x as f,

Hard-Core Predicate

- A **hard-core predicate** for a OWF $f : \{0,1\}^\lambda \to \{0,1\}^\ell$ is a function $\mathsf{hc} : \{0,1\}^\lambda \to \{0,1\}$ such that
  - It takes the same input x as f,
  - It outputs a single bit called **hard-core bit**,

Hard-Core Predicate

- A **hard-core predicate** for a OWF $f : \{0,1\}^\lambda \to \{0,1\}^\ell$ is a function $\mathsf{hc} : \{0,1\}^\lambda \to \{0,1\}$ such that
  - It takes the same input x as f,
  - It outputs a single bit called **hard-core bit**,
  - It can be efficiently computed given x, and

Hard-Core Predicate

- A **hard-core predicate** for a OWF $f : \{0,1\}^\lambda \to \{0,1\}^\ell$ is a function $\mathsf{hc} : \{0,1\}^\lambda \to \{0,1\}$ such that
  - It takes the same input x as f,
  - It outputs a single bit called **hard-core bit**,
  - It can be efficiently computed given x, and
  - It is hard to compute given only $y = f(x)$.

Hard-Core Predicate

- A **hard-core predicate** for a OWF $f : \{0,1\}^\lambda \to \{0,1\}^\ell$ is a function $\mathsf{hc} : \{0,1\}^\lambda \to \{0,1\}$ such that
  - It takes the same input x as f,
  - It outputs a single bit called **hard-core bit**,
  - It can be efficiently computed given x, and
  - It is hard to compute given only $y = f(x)$.
- **Intuition:** $f(x)$ may leak many bits of x but it does not leak $\mathsf{hc}(x)$.

Hard-Core Predicate

- A **hard-core predicate** for a OWF $f : \{0,1\}^\lambda \to \{0,1\}^\ell$ is a function $\mathsf{hc} : \{0,1\}^\lambda \to \{0,1\}$ such that
  - It takes the same input x as f,
  - It outputs a single bit called **hard-core bit**,
  - It can be efficiently computed given x, and
  - It is hard to compute given only $y = f(x)$.
- **Intuition:** $f(x)$ may leak many bits of x but it does not leak $\mathsf{hc}(x)$.
  - Intuitively, learning $\mathsf{hc}(x)$, even given $f(x)$, is as hard as inverting f.

Hard-Core Predicate

Hard-Core Predicate

Given a one-way function f, a family of functions $\{\mathsf{hc}_\lambda : \{0,1\}^\lambda \to \{0,1\}\}_\lambda$ is a hard-core predicate for f if it satisfies the following properties.

Hard-Core Predicate

Hard-Core Predicate

Given a one-way function f, a family of functions $\{\mathsf{hc}_\lambda : \{0,1\}^\lambda \to \{0,1\}\}_\lambda$ is a hard-core predicate for f if it satisfies the following properties.

- **Easy to compute:** For all $\lambda \in \mathbb{N}$, hc_λ can be computed in polynomial time.

Hard-Core Predicate

Hard-Core Predicate

Given a one-way function f, a family of functions $\{\mathsf{hc}_\lambda : \{0,1\}^\lambda \to \{0,1\}\}_\lambda$ is a hard-core predicate for f if it satisfies the following properties.

- **Easy to compute:** For all $\lambda \in \mathbb{N}$, hc_λ can be computed in polynomial time.
- **Hard to predict:** For all NUPPT adversaries A, there exists a negligible function negl, such that for all $\lambda \in \mathbb{N}$

$$\Pr_{x \xleftarrow{\$} \{0,1\}^\lambda}\left[A(f(x)) = \mathsf{hc}(x)\right] \leq$$

Hard-Core Predicate

Hard-Core Predicate

Given a one-way function f, a family of functions $\{\mathsf{hc}_\lambda : \{0,1\}^\lambda \to \{0,1\}\}_\lambda$ is a hard-core predicate for f if it satisfies the following properties.

- **Easy to compute:** For all $\lambda \in \mathbb{N}$, hc_λ can be computed in polynomial time.
- **Hard to predict:** For all NUPPT adversaries A, there exists a negligible function negl, such that for all $\lambda \in \mathbb{N}$

$$\Pr_{x \xleftarrow{\$} \{0,1\}^\lambda}\left[A(f(x)) = \mathsf{hc}(x)\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Hard-Core Predicate

Hard-Core Predicate

Given a one-way function f, a family of functions $\{\mathsf{hc}_\lambda : \{0,1\}^\lambda \to \{0,1\}\}_\lambda$ is a hard-core predicate for f if it satisfies the following properties.

- **Easy to compute:** For all $\lambda \in \mathbb{N}$, hc_λ can be computed in polynomial time.
- **Hard to predict:** For all NUPPT adversaries A, there exists a negligible function negl, such that for all $\lambda \in \mathbb{N}$

$$\Pr_{x \xleftarrow{\$} \{0,1\}^\lambda}\left[A(f(x)) = \mathsf{hc}(x)\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Hard-core predicates help get pseudorandomness from OWF.

PRGs from OWF

Theorem [Håstad-Impagliazzo-Levin-Luby'90]**:**

$$\text{OWF} \Longrightarrow \text{PRG}.$$

PRGs from OWP

Theorem [Goldreich-Levin'89]**:**

$$\text{OWP} \Longrightarrow \text{PRG}.$$

PRGs from OWP

Theorem [Goldreich-Levin'89]**:**

Given a OWP f and a hard-core predicate hc for f, the following construction G is a PRG

PRGs from OWP

Theorem [Goldreich-Levin'89]**:**

Given a OWP f and a hard-core predicate hc for f, the following construction G is a PRG

$$G(x) = f(x) \,\|\, \mathsf{hc}(x)\,.$$

PRGs from OWP

Theorem [Goldreich-Levin'89]**:**

Given a OWP f and a hard-core predicate hc for f, the following construction G is a PRG

$$G(x) = f(x) \,\|\, \mathsf{hc}(x)\,.$$

PRG with single-bit stretch.

PRGs from OWP

Theorem [Goldreich-Levin'89]**:**

Given a OWP f and a hard-core predicate hc for f, the following construction G is a PRG

$$G(x) = f(x) \,\|\, \mathsf{hc}(x)\,.$$

PRG with single-bit stretch.

Intuition:

PRGs from OWP

Theorem [Goldreich-Levin'89]**:**

Given a OWP f and a hard-core predicate hc for f, the following construction G is a PRG

$$G(x) = f(x) \,\|\, \mathsf{hc}(x)\,.$$

PRG with single-bit stretch.

Intuition:

Since f is a one-one, when x is sampled uniformly at random, $f(x)$ is uniformly random over $\{0,1\}^{\lambda}$.

PRGs from OWP

Theorem [Goldreich-Levin'89]**:**

Given a OWP f and a hard-core predicate hc for f, the following construction G is a PRG

$$G(x) = f(x) \,\|\, \mathsf{hc}(x)\,.$$

PRG with single-bit stretch.

Intuition:

Since f is a one-one, when x is sampled uniformly at random, $f(x)$ is uniformly random over $\{0,1\}^{\lambda}$.

Thus, if $G(x)$ is distinguishable from a uniformly random string, then it must be because of appending $\mathsf{hc}(x)$. But $\mathsf{hc}(x)$ is hard to predict even given $f(x)$.