Algebraic Pseudorandom Functions with Improved Efficiency from Augmented Cascade

Dan Boneh
Hart Montgomery
Ananth Raghunathan

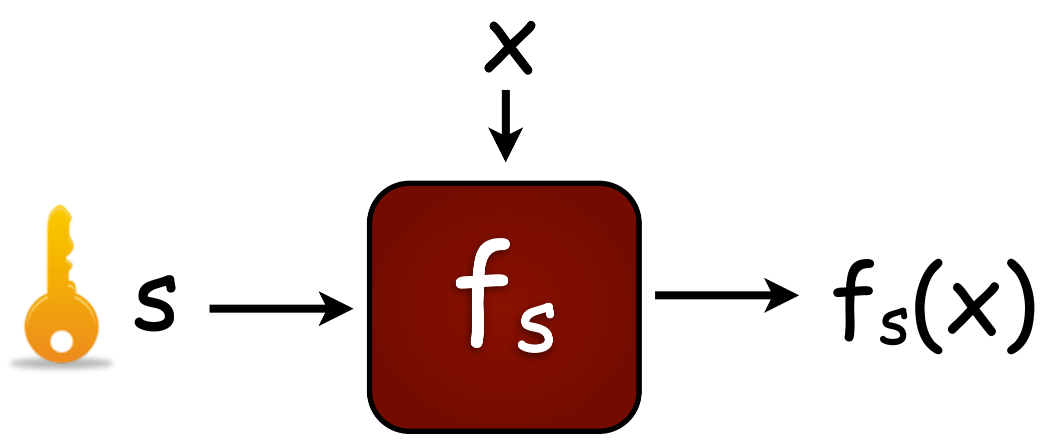
Stanford University

Pseudorandom Functions

- Definition

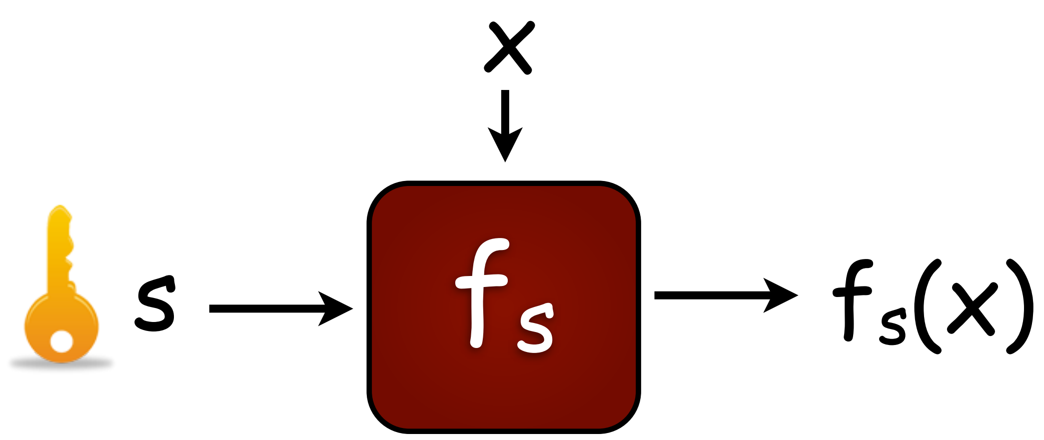
Pseudorandom Functions

- Definition



Pseudorandom Functions

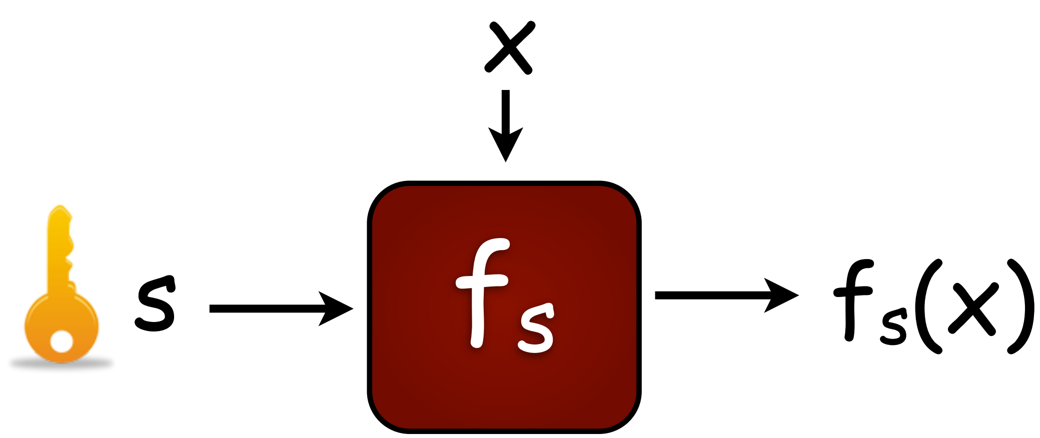
- Definition



The function f looks like a random function

Pseudorandom Functions

- Definition



The function f looks like a random function

- Security Game

Pseudorandom Functions

- Definition

$$s \longrightarrow \boxed{f_s} \longrightarrow f_s(x), \quad x \downarrow$$

The function f looks like a random function

- Security Game

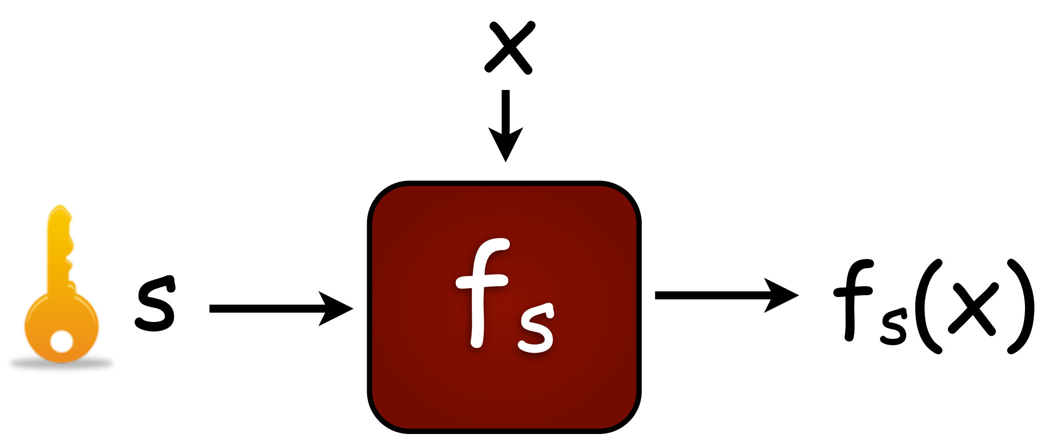
Exp-PRF: choose a random key k and set $f(x)=f_k(x)$

Exp-rand: choose a random function $f(x)$

$$\xleftarrow{\;x\;}$$
$$\xrightarrow[f(x)]{}$$

Pseudorandom Functions

- Definition

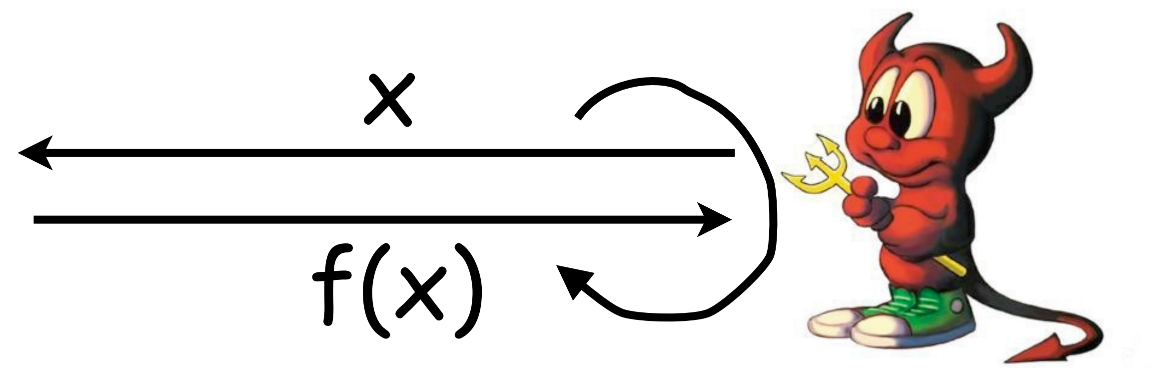


The function f looks like a random function

- Security Game

Exp-PRF: choose a random key k and set $f(x)=f_k(x)$

Exp-rand: choose a random function $f(x)$



Pseudorandom Functions

- Definition

$$s \longrightarrow \boxed{f_s} \longrightarrow f_s(x), \quad x \downarrow f_s$$

The function f looks like a random function

- Security Game

Exp-PRF: choose a random key k and set $f(x)=f_k(x)$

Exp-rand: choose a random function $f(x)$

x

$f(x)$

PRF? or rand?

Pseudorandom Functions

- Definition

$$s \rightarrow \boxed{f_s} \rightarrow f_s(x), \quad x \downarrow$$

The function f looks like a random function

- Security Game

Exp-PRF: choose a random key k and set $f(x)=f_k(x)$

Exp-rand: choose a random function $f(x)$

$\xleftarrow{\;x\;}$

$\xrightarrow[\;f(x)\;]{}$

PRF? or rand?

Secure if  cannot guess "PRF" or "rand" with probability better than 1/2

Applications of PRFs

- Workhorse of cryptography. Lots of applications!
  - Private Key Crypto; parties share PRF secret
  - Message Integrity and User Authentication
  - Key Derivation schemes
  - Stateless Signature Schemes
  - Defend against denial-of-service attacks [Ber ’96, CW03]
  - Prove lower bounds in learning theory; impossibility results in complexity theory

Heuristic PRFs

Heuristic PRFs

DES IDEA **AES**
Triple-DES Serpent

Heuristic PRFs

DES
IDEA
Triple-DES
Serpent
AES

Very fast but unfortunately rely on interactive assumptions

Heuristic PRFs

DES IDEA Triple-DES Serpent **AES**

Very fast but unfortunately rely on interactive assumptions

AES assumption: AES is a secure PRF!

Heuristic PRFs

DES Triple-DES IDEA Serpent **AES**

Very fast but unfortunately rely on interactive assumptions

AES assumption: AES is a secure PRF!

Requires interactions between Challenger and Adversary

Heuristic PRFs

DES
Triple-DES
IDEA
Serpent
AES

Very fast but unfortunately rely on interactive assumptions

AES assumption: AES is a secure PRF!

Requires interactions between Challenger and Adversary

Algebraic Constructions

Heuristic PRFs

DES Triple-DES IDEA Serpent **AES**

Very fast but unfortunately rely on interactive assumptions

AES assumption: AES is a secure PRF!

Requires interactions between Challenger and Adversary

Algebraic Constructions

Non-interactive assumptions: Challenges posted. Require no interaction.

Heuristic PRFs

DES Triple-DES IDEA Serpent **AES**

Very fast but unfortunately rely on interactive assumptions

AES assumption: AES is a secure PRF!

Requires interactions between Challenger and Adversary

Algebraic Constructions

Non-interactive assumptions: Challenges posted. Require no interaction.

Eg., DDH, Discrete Log, etc.

Cascade Construction

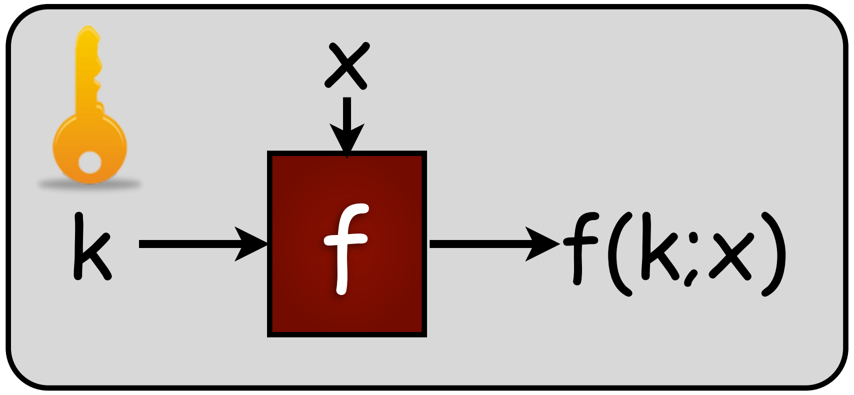
Cascade Construction

- Introduced by Bellare-Canetti-Krawczyk [Crypto ’96]

Cascade Construction

- Introduced by Bellare-Canetti-Krawczyk [Crypto '96]

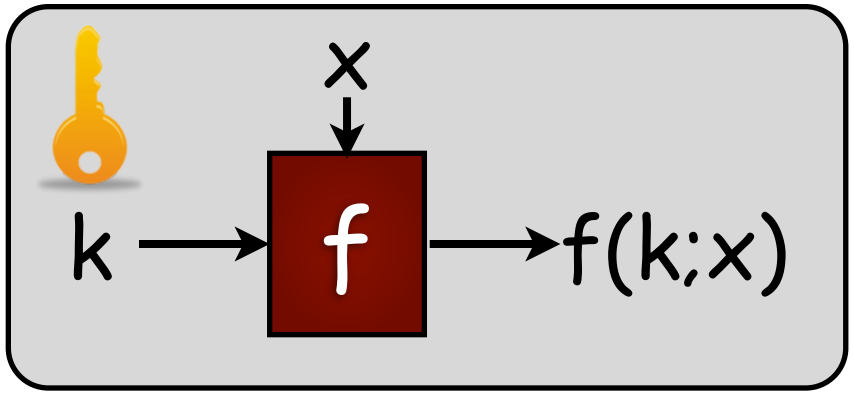
$$f : K \times X \rightarrow K$$



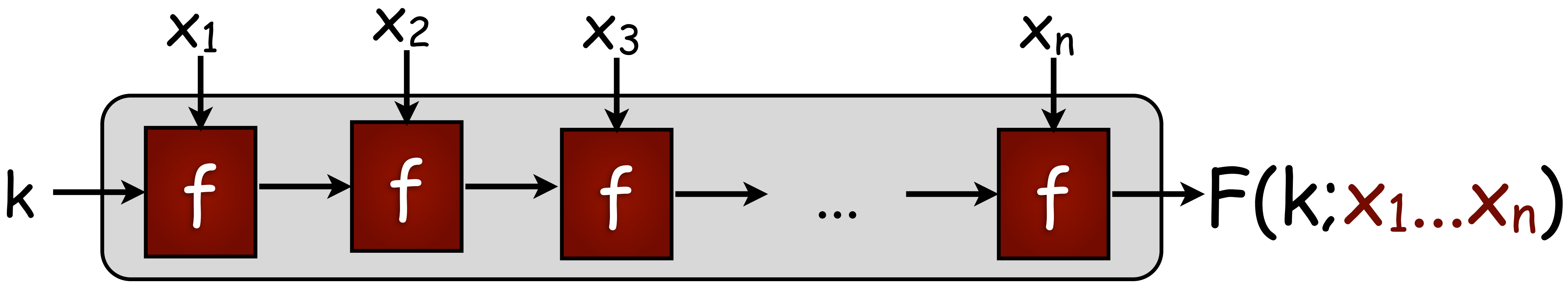
Cascade Construction

- Introduced by Bellare-Canetti-Krawczyk [Crypto '96]

$$f : K \times X \to K$$



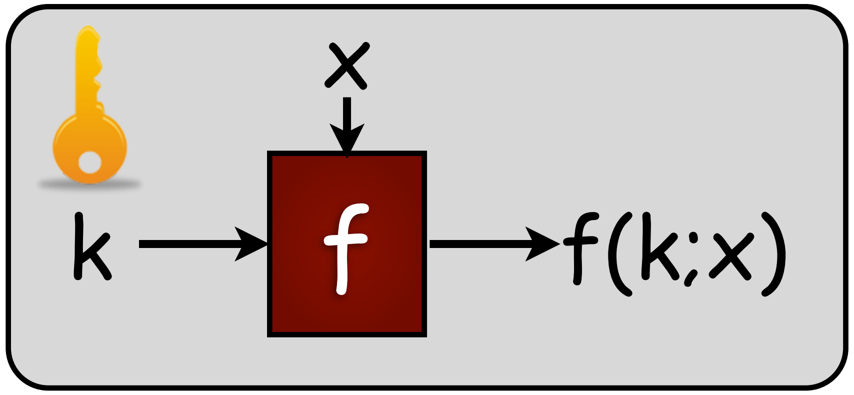
Cascade:



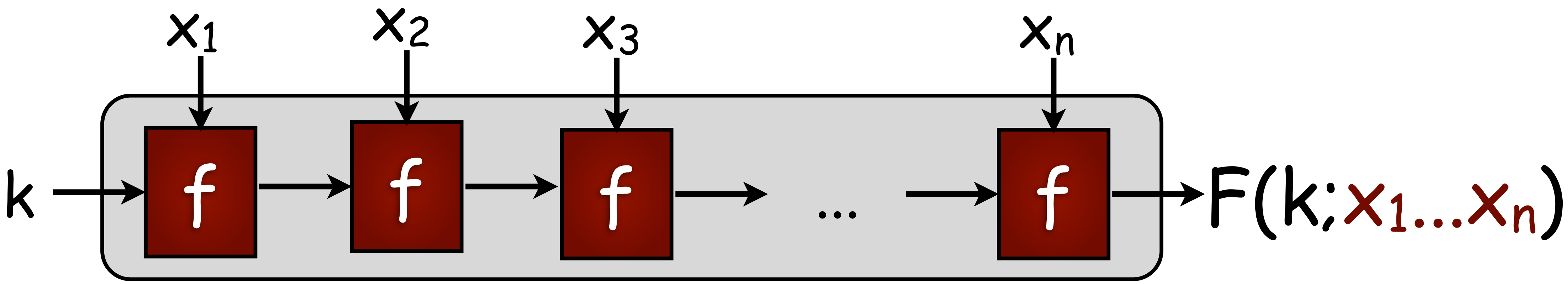
Cascade Construction

- Introduced by Bellare-Canetti-Krawczyk [Crypto '96]

$$f : K \times X \to K$$



Cascade:

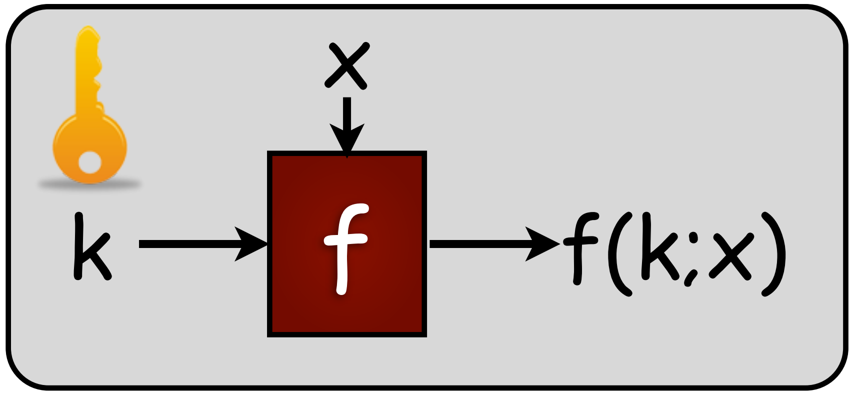


Key from **K**, inputs from $\mathbf{X}^n$

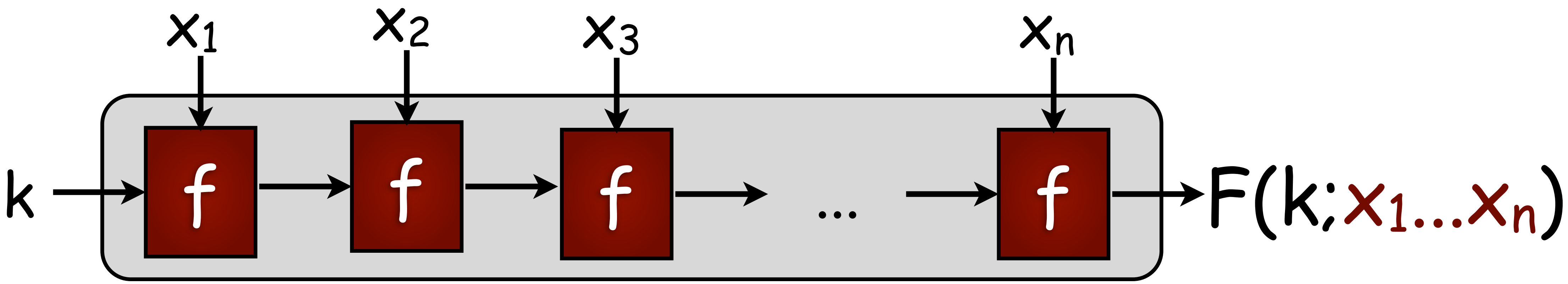
Cascade Construction

- Introduced by Bellare-Canetti-Krawczyk [Crypto '96]

$$f : K \times X \to K$$



Cascade:



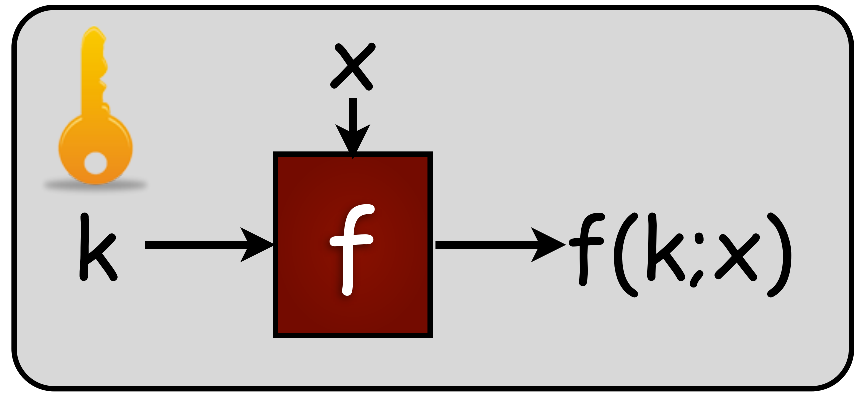
Key from **K**, inputs from $\mathbf{X}^n$

Theorem f is a secure PRF $\Longrightarrow$ F is a secure PRF

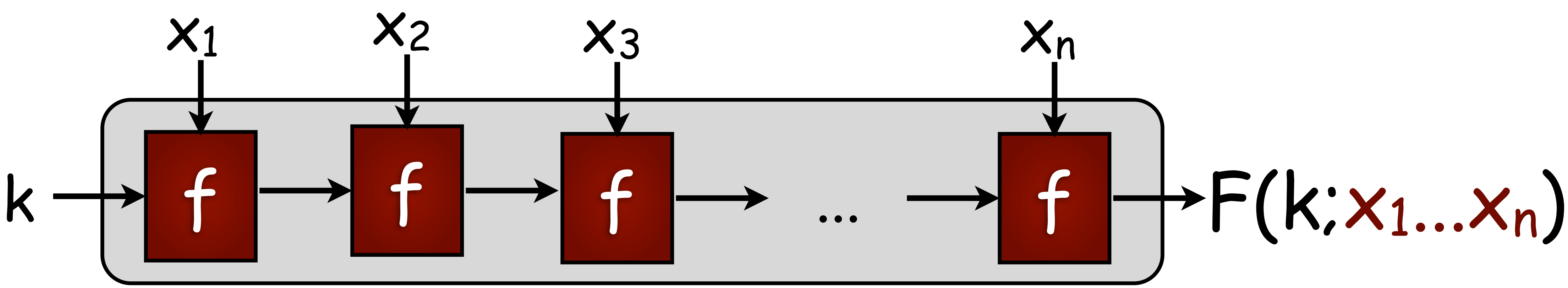
Cascade Construction

- Introduced by Bellare-Canetti-Krawczyk [Crypto '96]

$$f : K \times X \to K$$



Cascade:



Key from **K**, inputs from $\mathbf{X}^n$

Theorem f is a secure PRF $\Longrightarrow$ F is a secure PRF

Generalizes first algebraic construction of PRF by Goldreich-Goldwasser-Micali [GGM '86] from Pseudorandom Generators

Augmented Cascade

Augmented Cascade

Range of the function smaller than key domain.

Augmented Cascade

Range of the function smaller than key domain.

Can extend output using pseudorandom generators; but breaks the *algebraic structure*

Augmented Cascade

Range of the function smaller than key domain.

Can extend output using pseudorandom generators; but breaks the *algebraic structure*

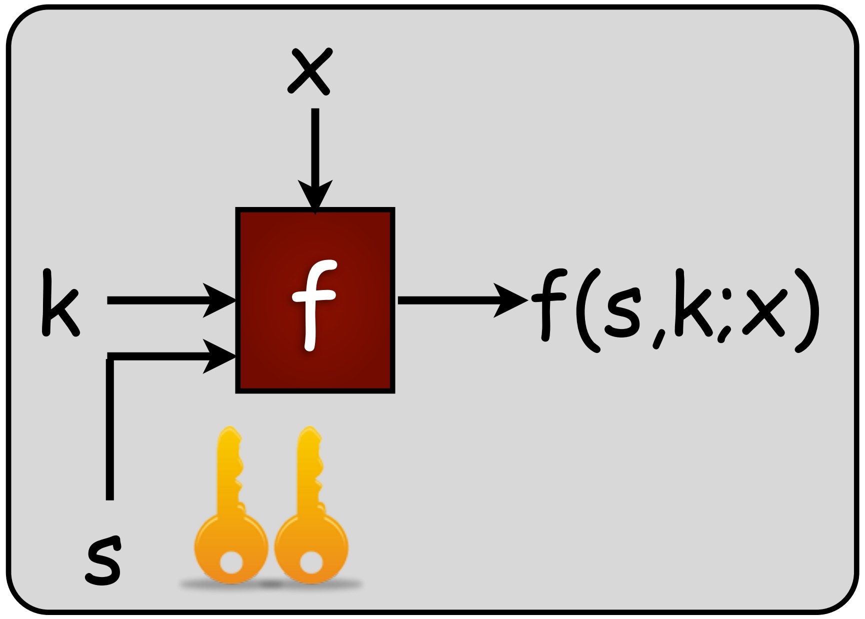
$$f : \underbrace{(S \times K)}_{\text{key}} \times X \to K$$

Augmented Cascade

Range of the function smaller than key domain.

Can extend output using pseudorandom generators; but breaks the *algebraic structure*

$$f : \underbrace{(S \times K)}_{\text{key}} \times X \to K$$

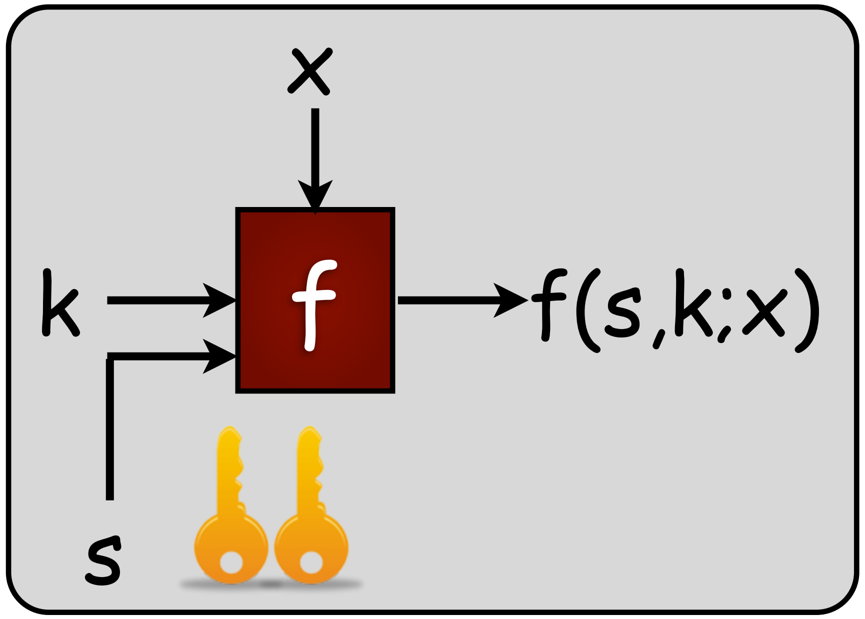


Augmented Cascade

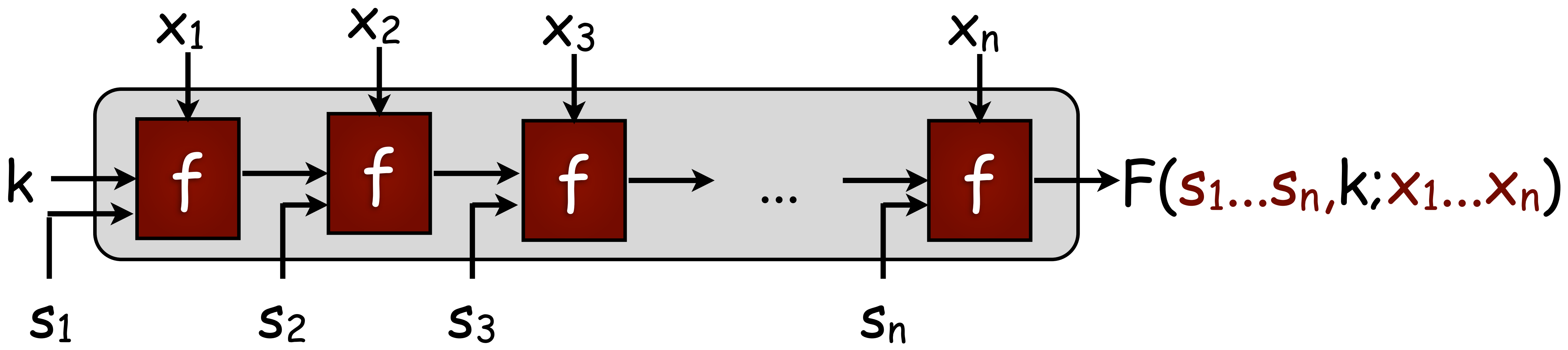
Range of the function smaller than key domain.

Can extend output using pseudorandom generators; but breaks the *algebraic structure*

$$f : \underbrace{(S \times K)}_{\text{key}} \times X \to K$$



Augmented Cascade:

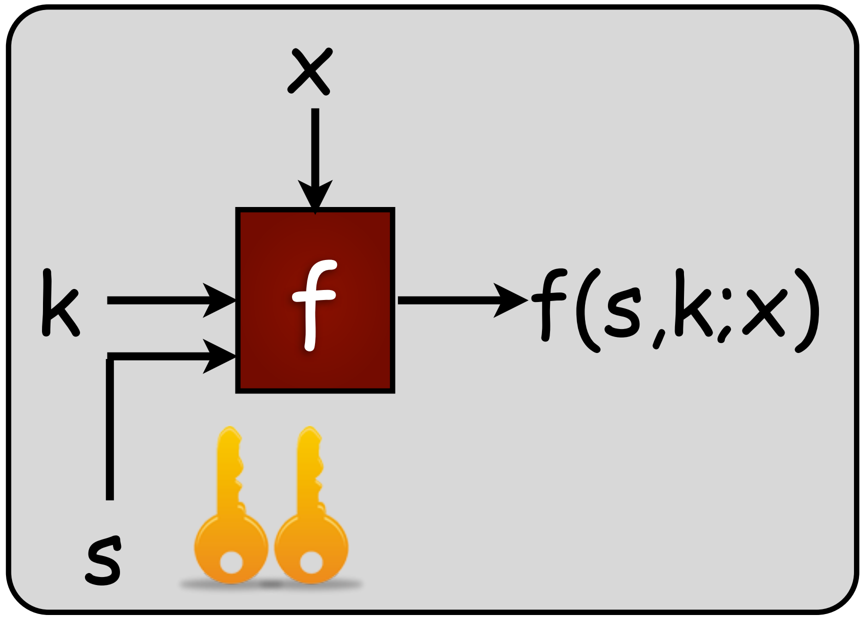


Augmented Cascade

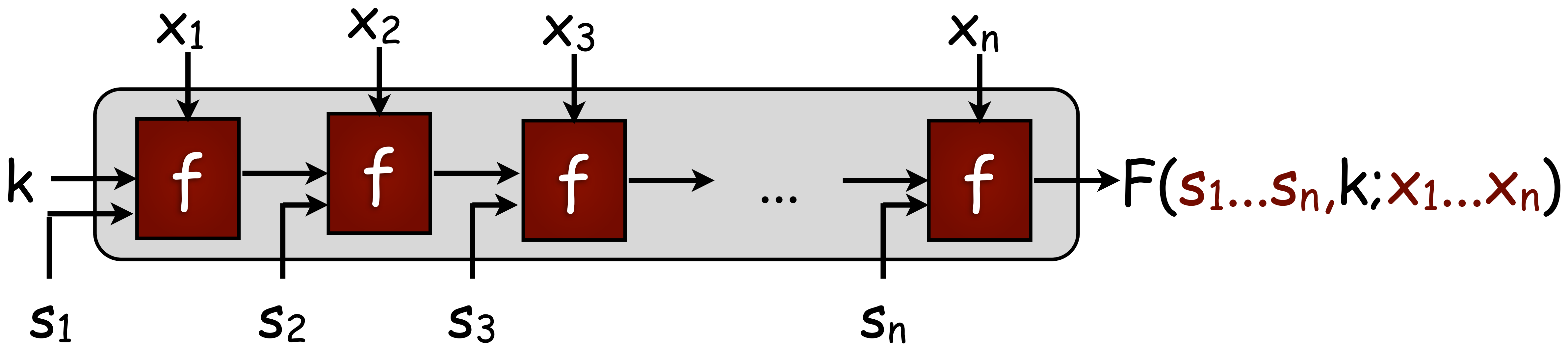
Range of the function smaller than key domain.

Can extend output using pseudorandom generators; but breaks the *algebraic structure*

$$f : \underbrace{(S \times K)}_{\text{key}} \times X \to K$$



Augmented Cascade:



Key from $S^n \times K$, inputs from X^n

Is Augmented Cascade secure?

Is Augmented Cascade secure?

f is a secure PRF

Is Augmented Cascade secure?

f is a secure PRF $\Longrightarrow$ F is a secure PRF ?

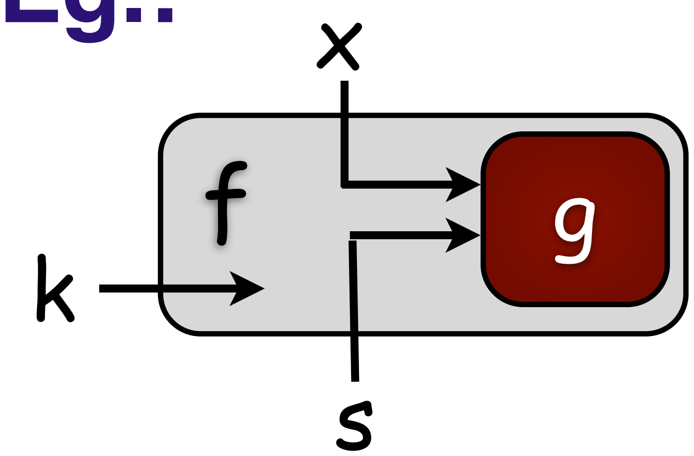
Is Augmented Cascade secure?

f is a secure PRF $\not\Longrightarrow$ F is a secure PRF ?

Is Augmented Cascade secure?

f is a secure PRF $\not\Rightarrow$ F is a secure PRF ?

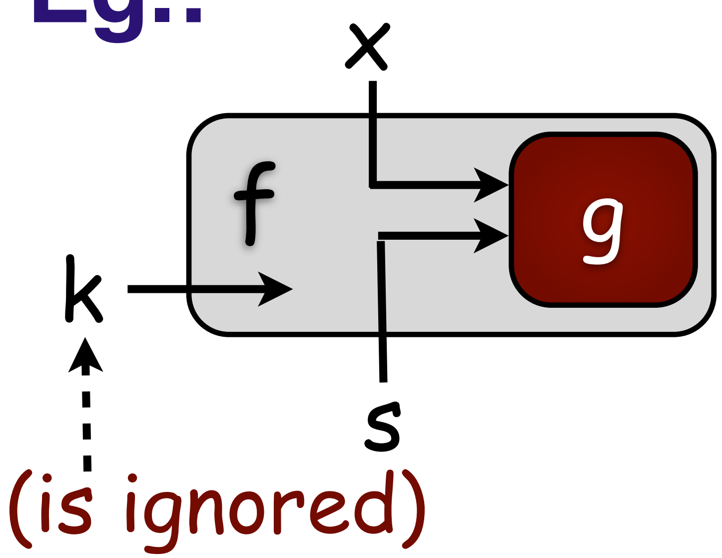
Eg.:



Is Augmented Cascade secure?

f is a secure PRF $\not\Rightarrow$ F is a secure PRF ?

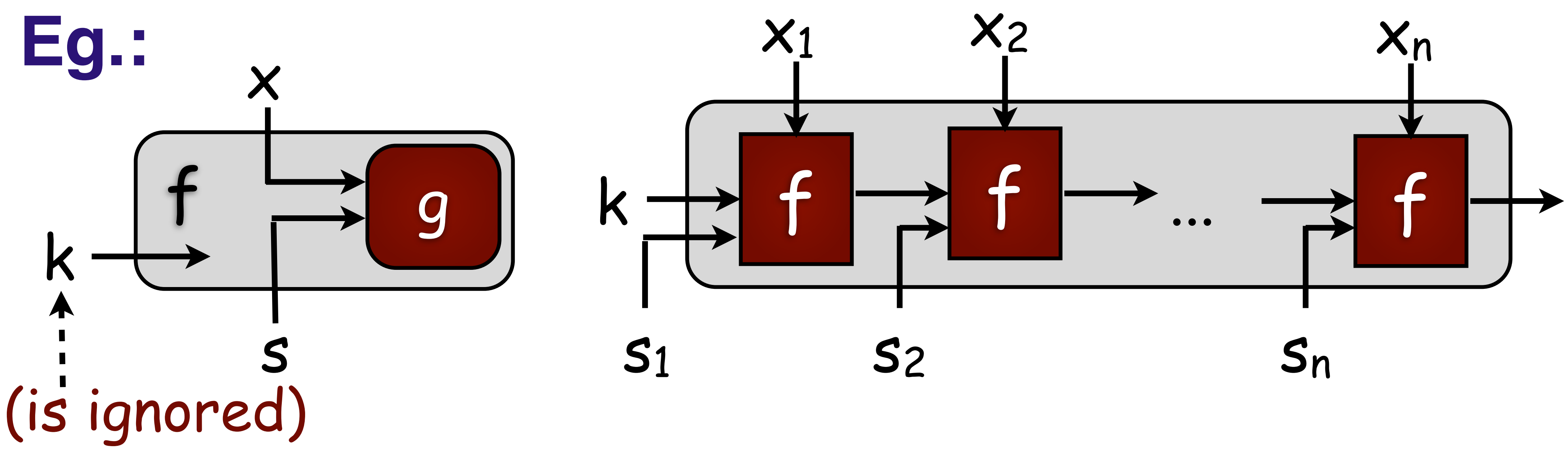
Eg.:



Is Augmented Cascade secure?

f is a secure PRF $\not\Rightarrow$ F is a secure PRF ?

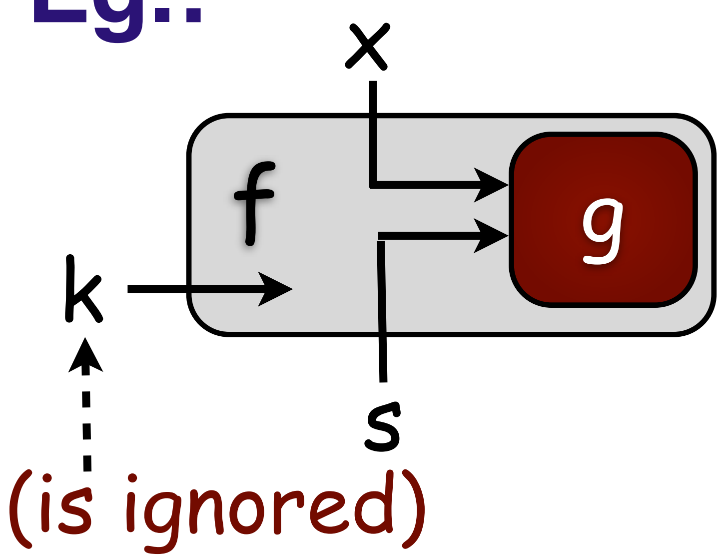
Eg.:

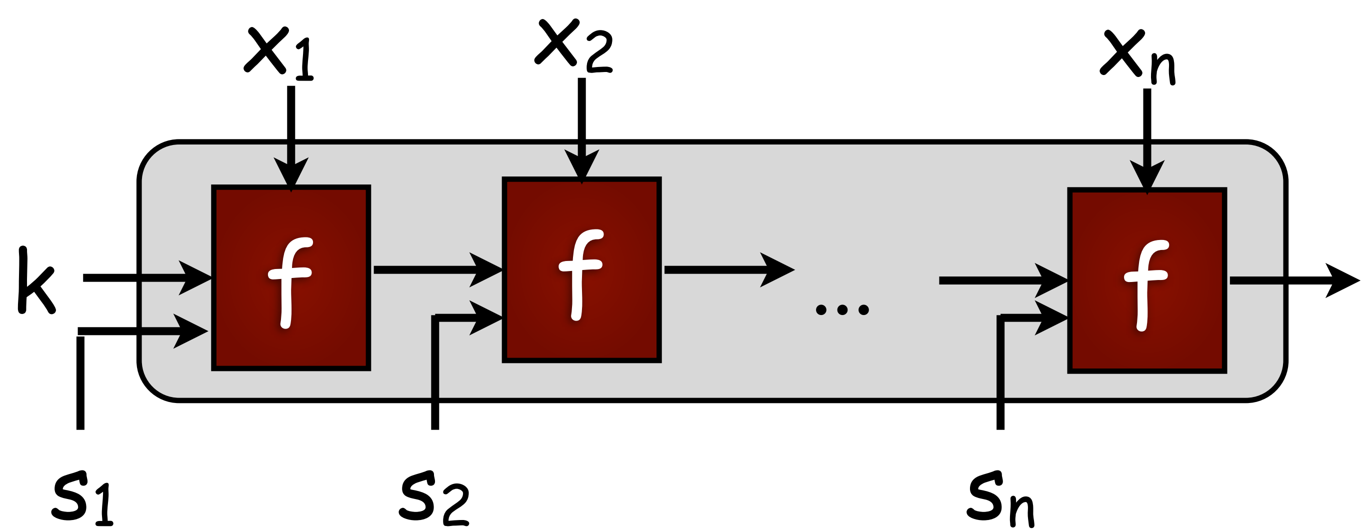


Is Augmented Cascade secure?

f is a secure PRF $\not\Rightarrow$ F is a secure PRF ?

Eg.:



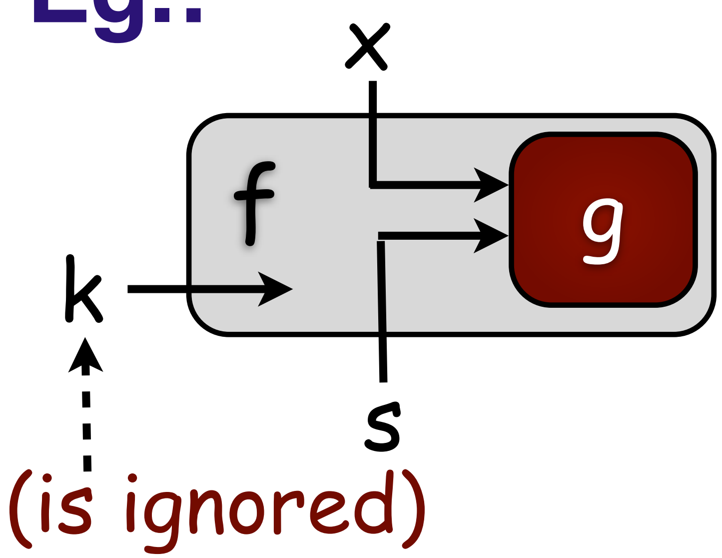


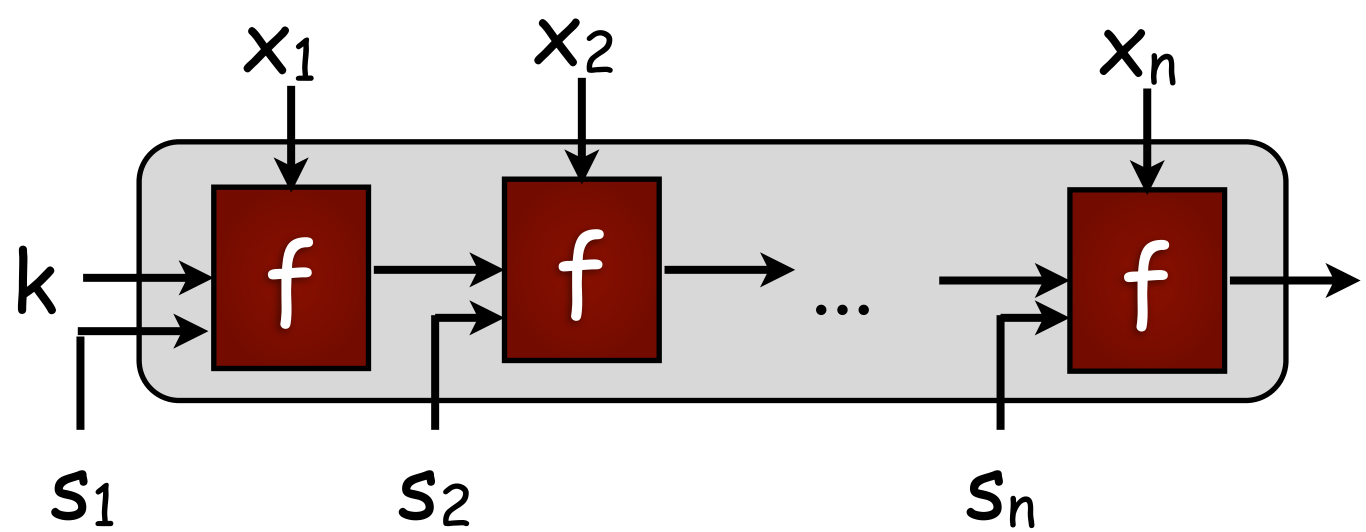
Easy to see: $F(s_1 \ldots s_n, k; x_1 \ldots x_n) = g(s_n, x_n)$

Is Augmented Cascade secure?

f is a secure PRF $\not\Rightarrow$ F is a secure PRF ?

Eg.:





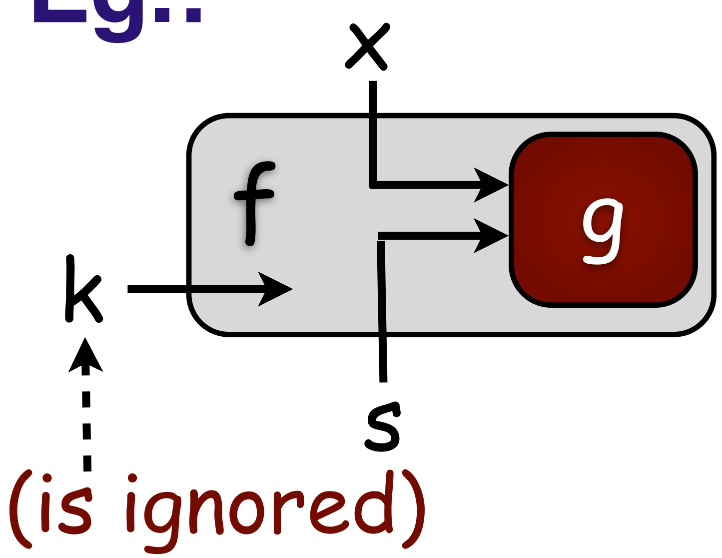
Easy to see: $F(s_1 \ldots s_n, k; x_1 \ldots x_n) = g(s_n, x_n)$

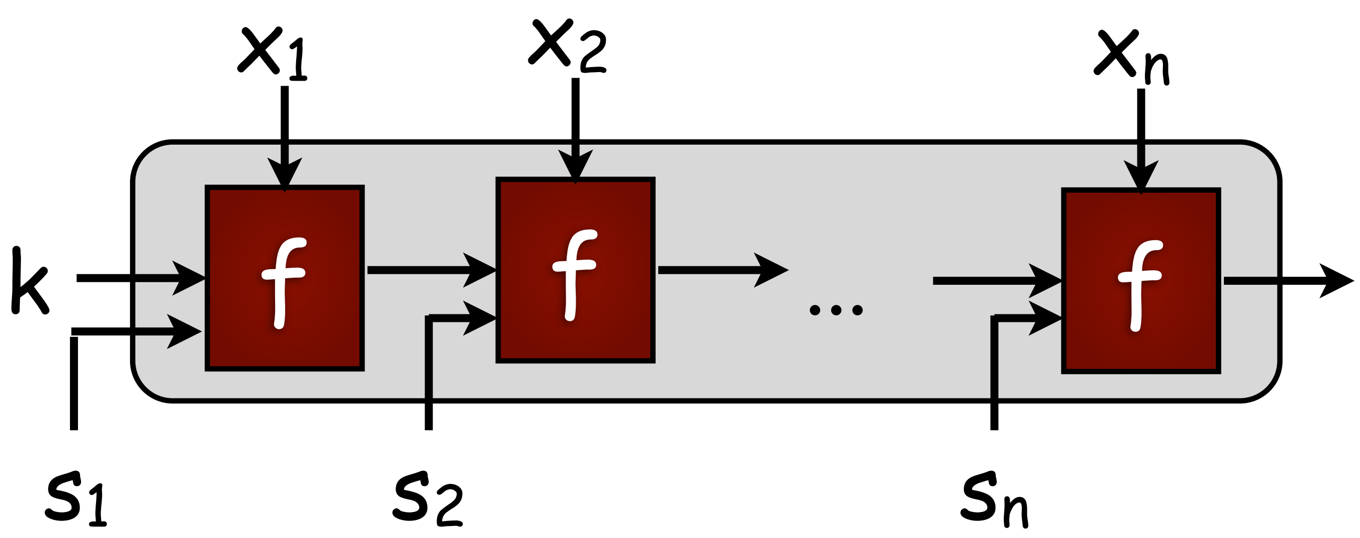
Challenger

Is Augmented Cascade secure?

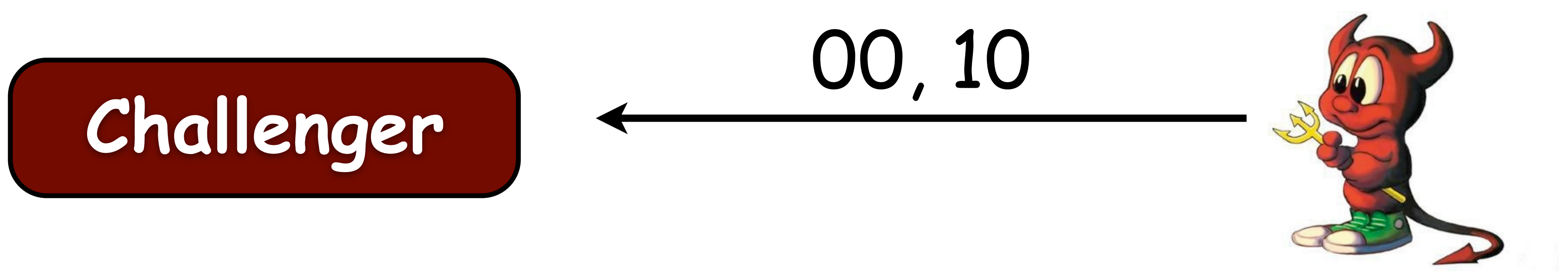
f is a secure PRF $\not\Rightarrow$ F is a secure PRF ?

Eg.:





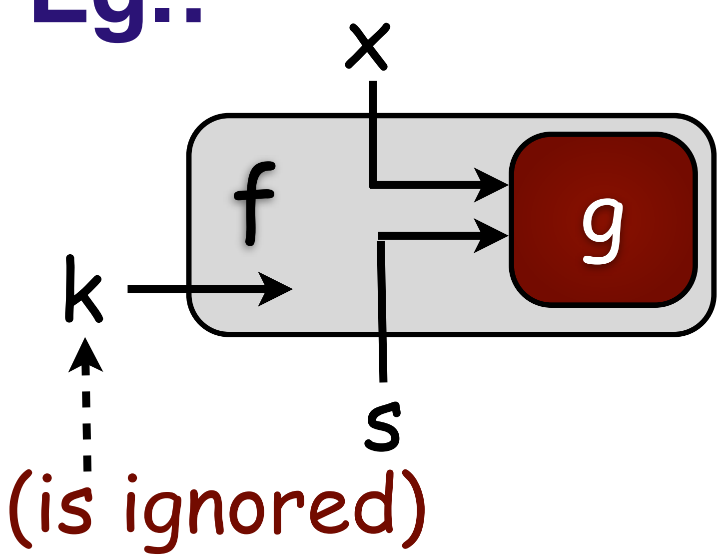
Easy to see: $F(s_1 \ldots s_n, k; x_1 \ldots x_n) = g(s_n, x_n)$

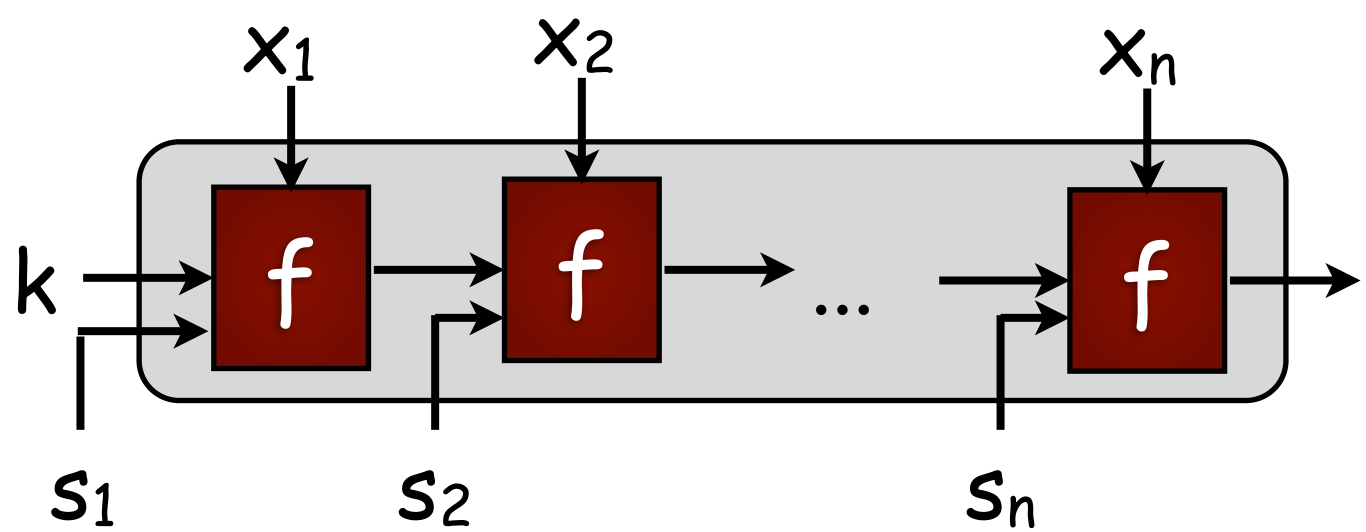


Is Augmented Cascade secure?

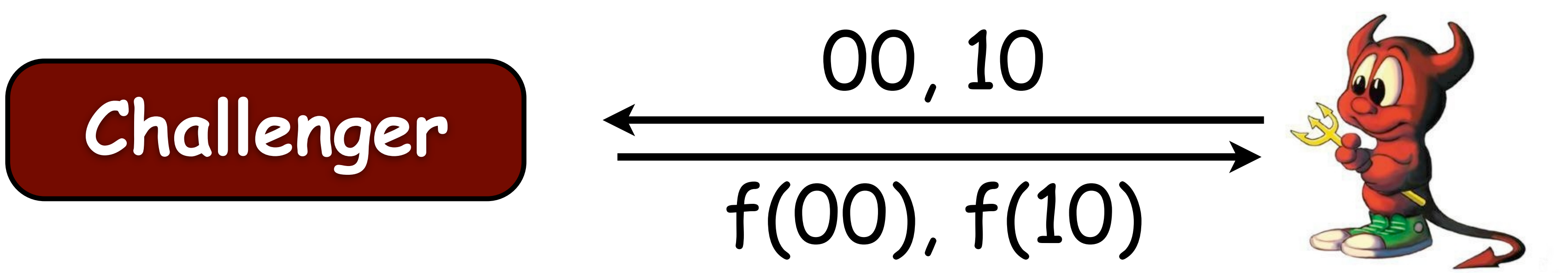
f is a secure PRF $\not\Rightarrow$ F is a secure PRF ?

Eg.:





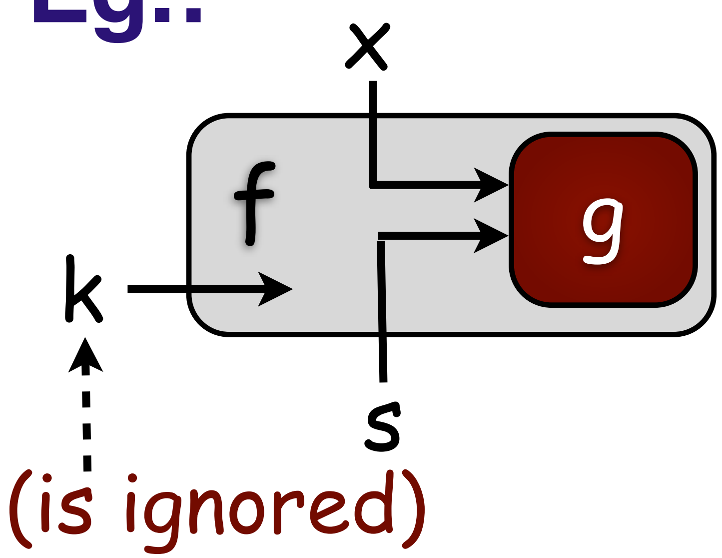
Easy to see: $F(s_1 \ldots s_n, k; x_1 \ldots x_n) = g(s_n, x_n)$

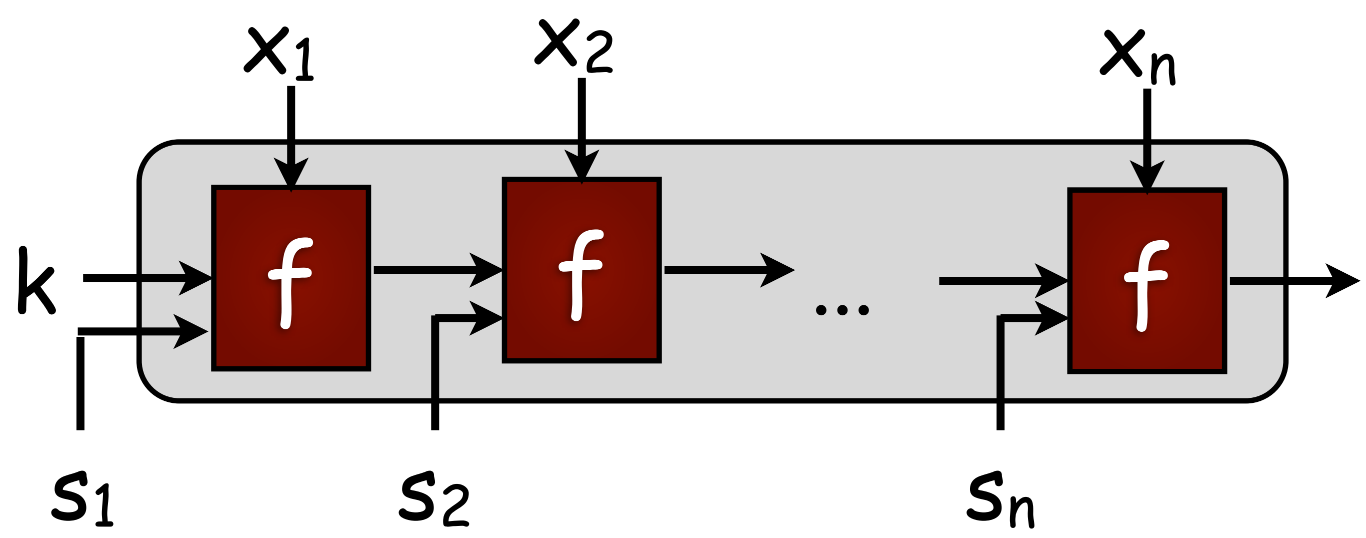


Is Augmented Cascade secure?

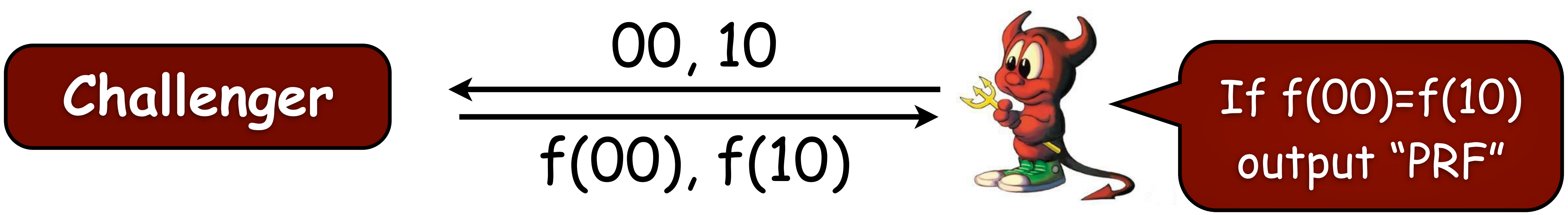
f is a secure PRF $\not\Rightarrow$ F is a secure PRF ?

Eg.:





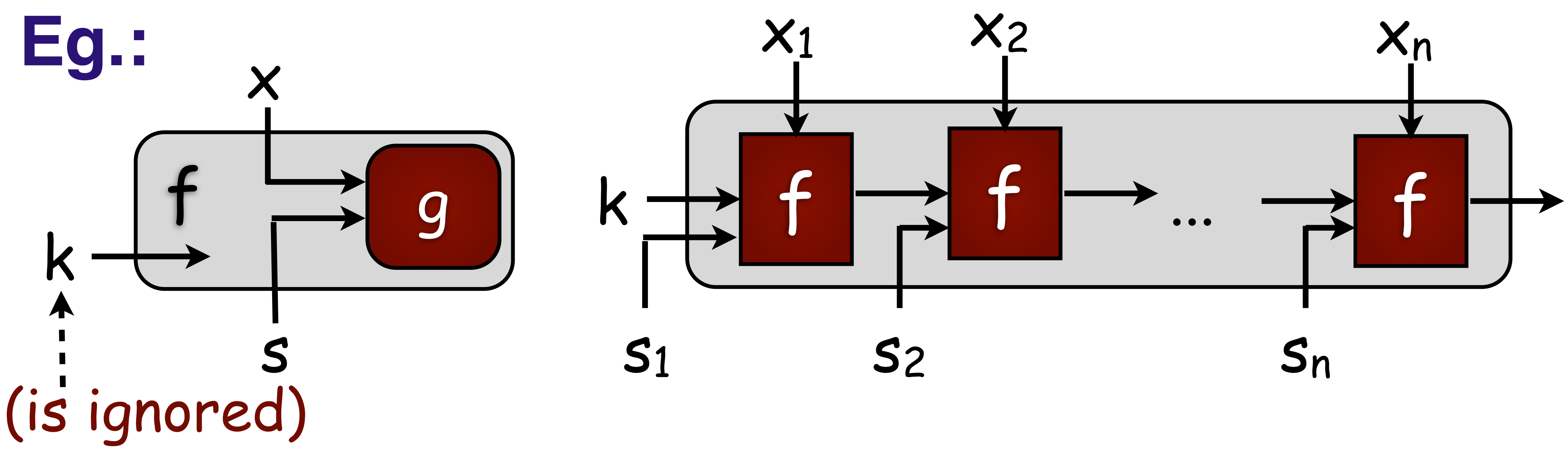
Easy to see: $F(s_1 \ldots s_n, k; x_1 \ldots x_n) = g(s_n, x_n)$



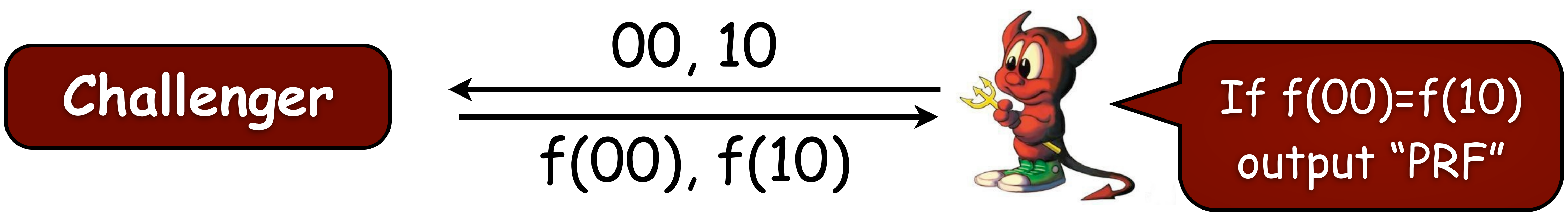
Is Augmented Cascade secure?

f is a secure PRF $\not\Rightarrow$ F is a secure PRF ?

Eg.:



Easy to see: $F(s_1 \ldots s_n, k; x_1 \ldots x_n) = g(s_n, x_n)$



f(00)=f(10) for random f with very low probability

Therefore, F is not a secure PRF!

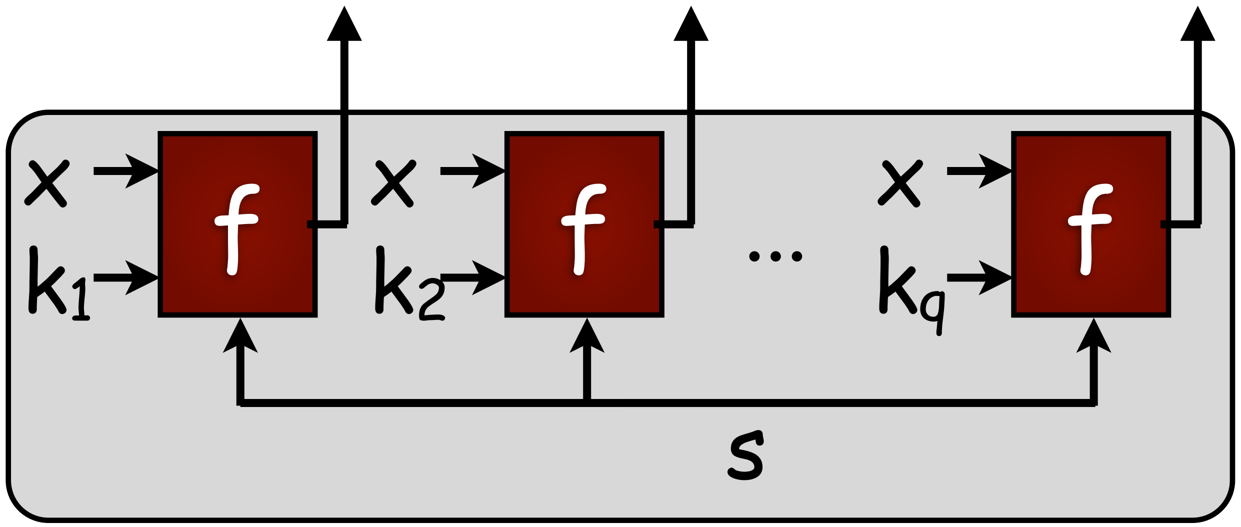
Parallel Composition Security

Parallel Composition Security

Consider q related keys (s,k_1), ..., (s,k_q)

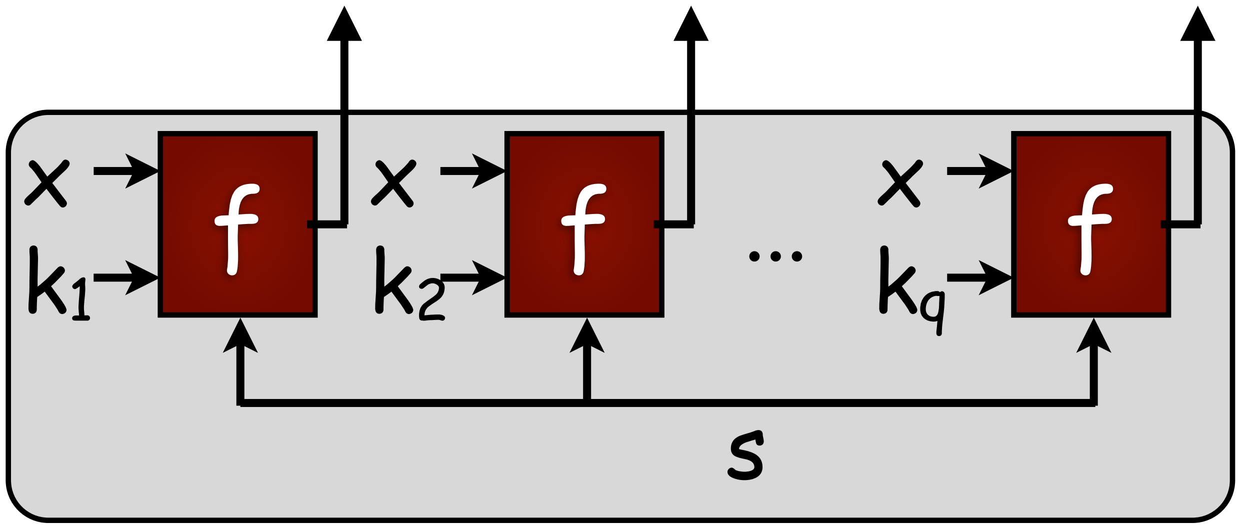
Parallel Composition Security

Consider q related keys $(s,k_1), \ldots, (s,k_q)$

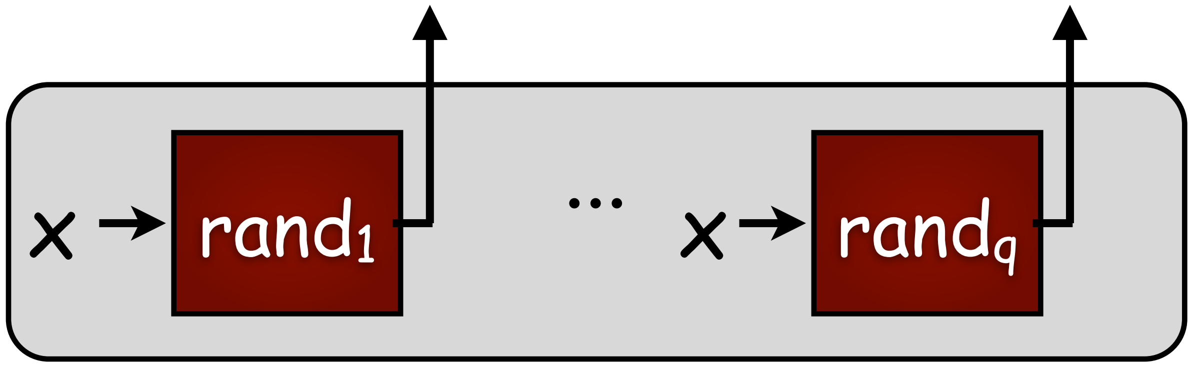


Parallel Composition Security

Consider q related keys $(s,k_1), \ldots, (s,k_q)$

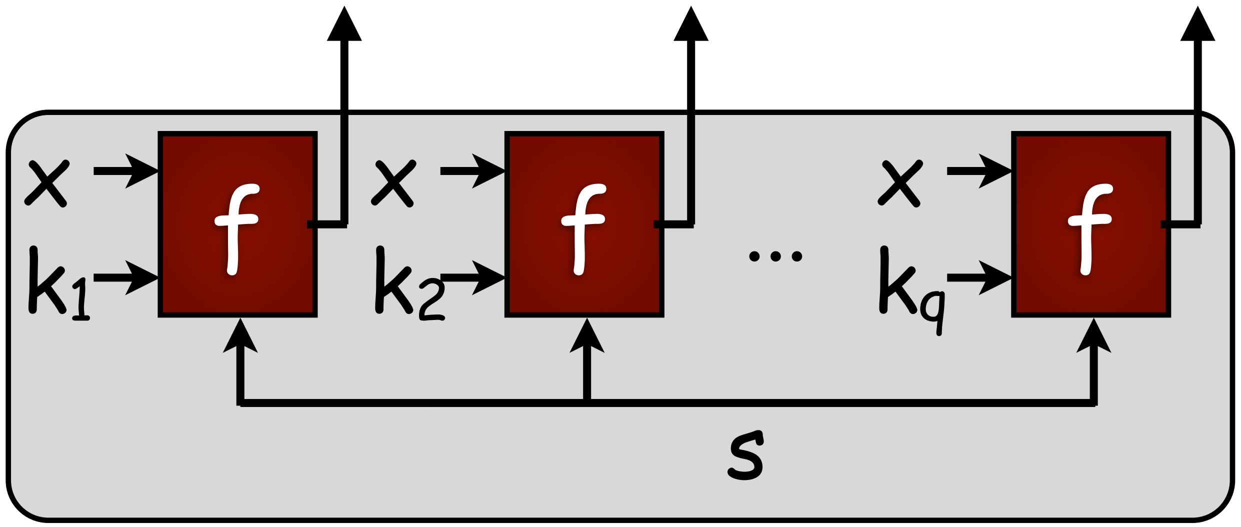


These functions look like q random functions

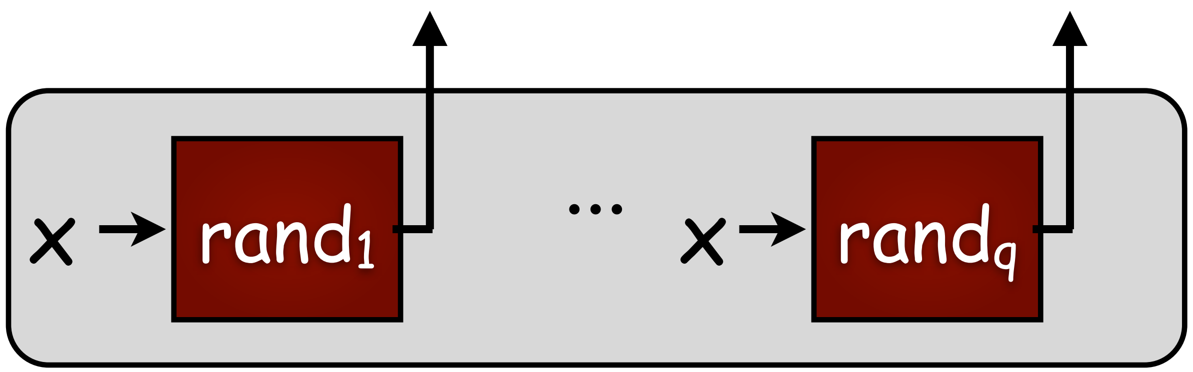


Parallel Composition Security

Consider q related keys $(s,k_1), \ldots, (s,k_q)$



These functions look like q random functions

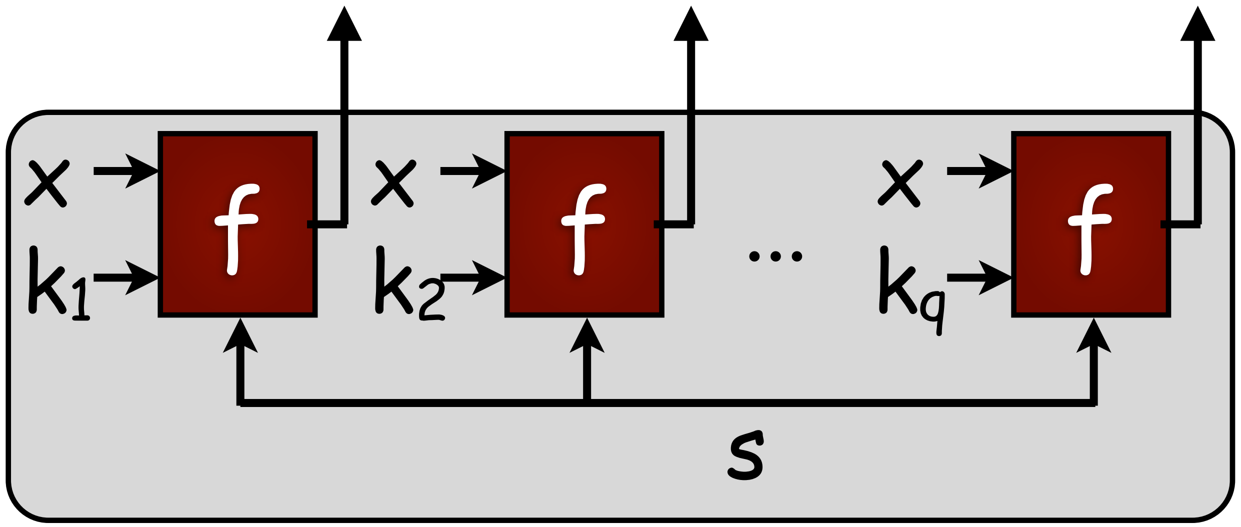


In other words:

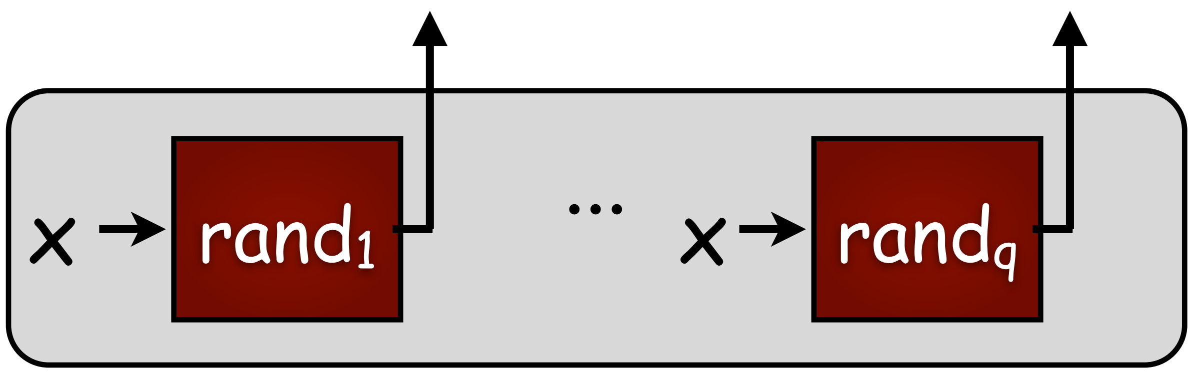
"Simultaneously Secure"

Parallel Composition Security

Consider q related keys $(s,k_1), \ldots, (s,k_q)$



These functions look like q random functions



In other words:

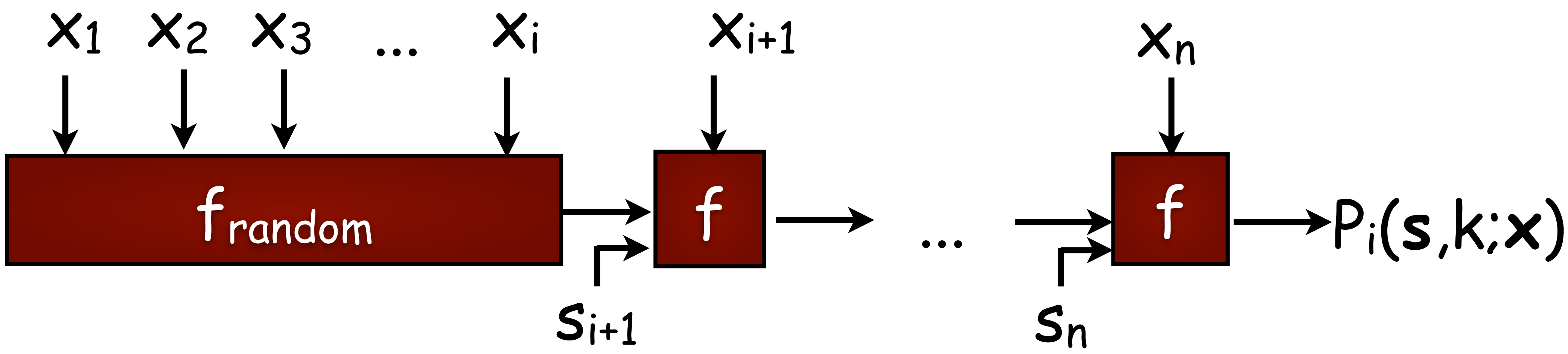
"Simultaneously Secure"

Theorem

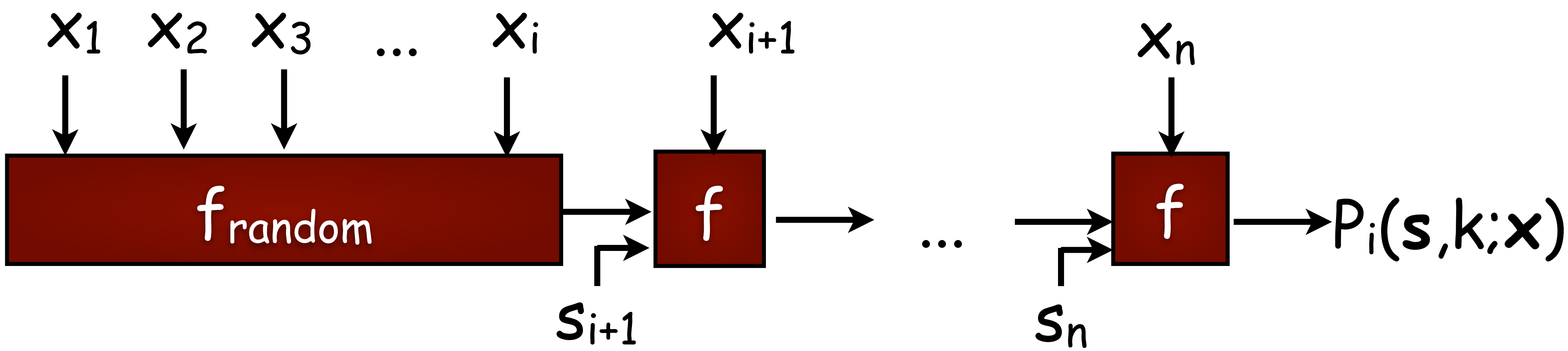
f is a q-parallel secure PRF $\Longrightarrow$ F is a secure PRF

Proof Outline

Proof Outline

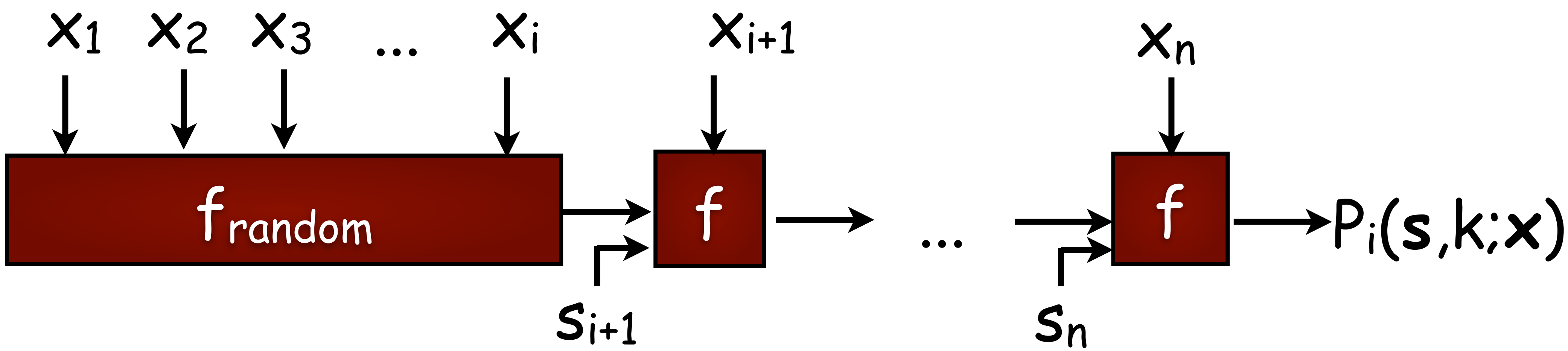


Proof Outline

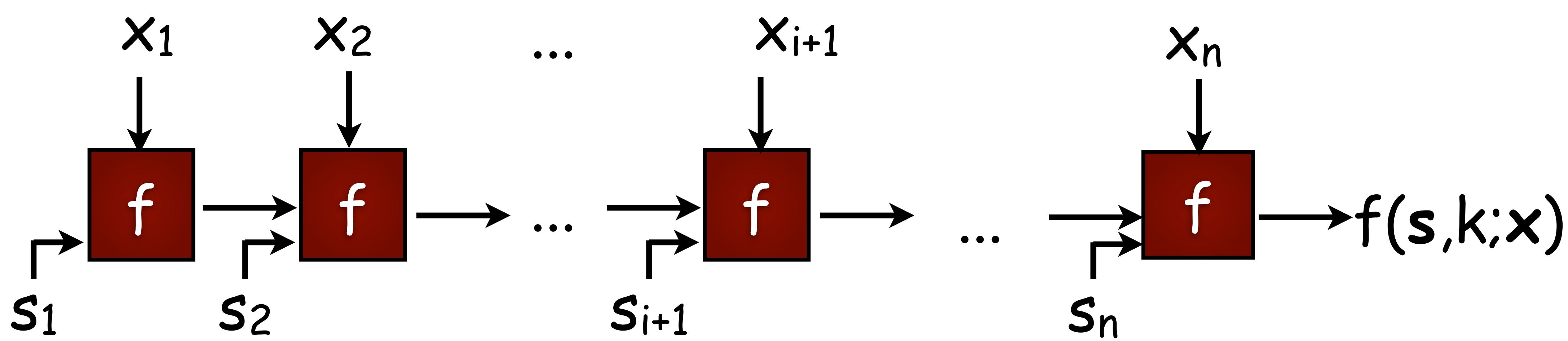


Initially:

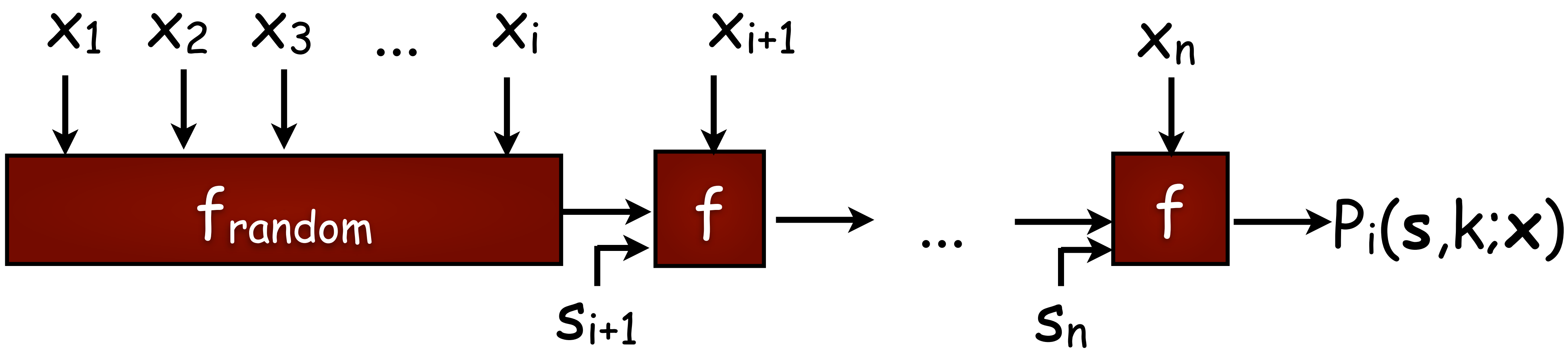
Proof Outline



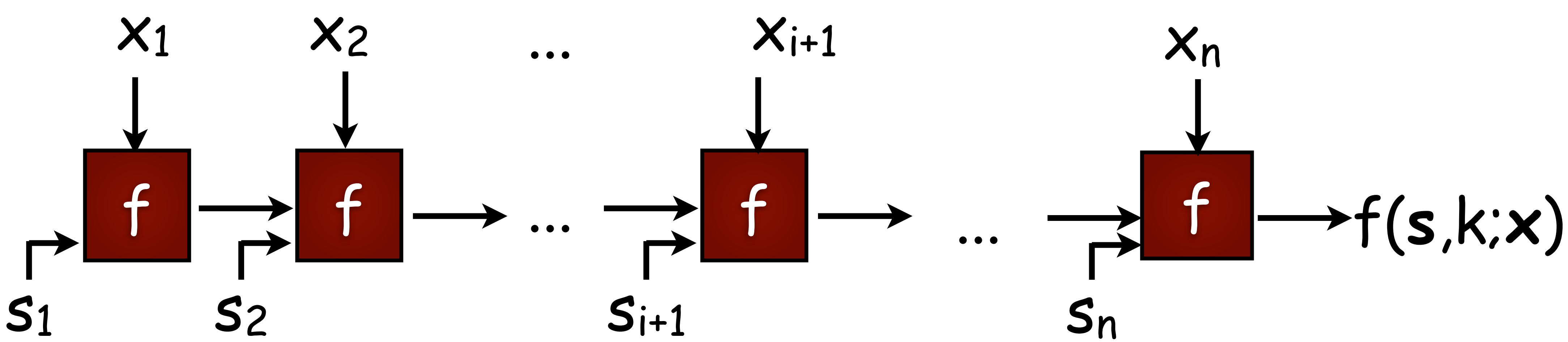
Initially:



Proof Outline

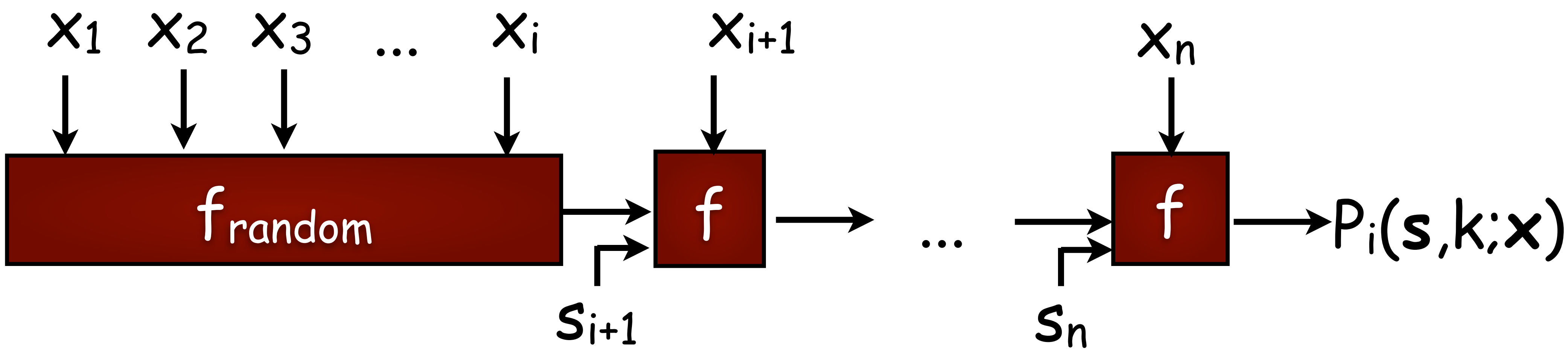


Initially:

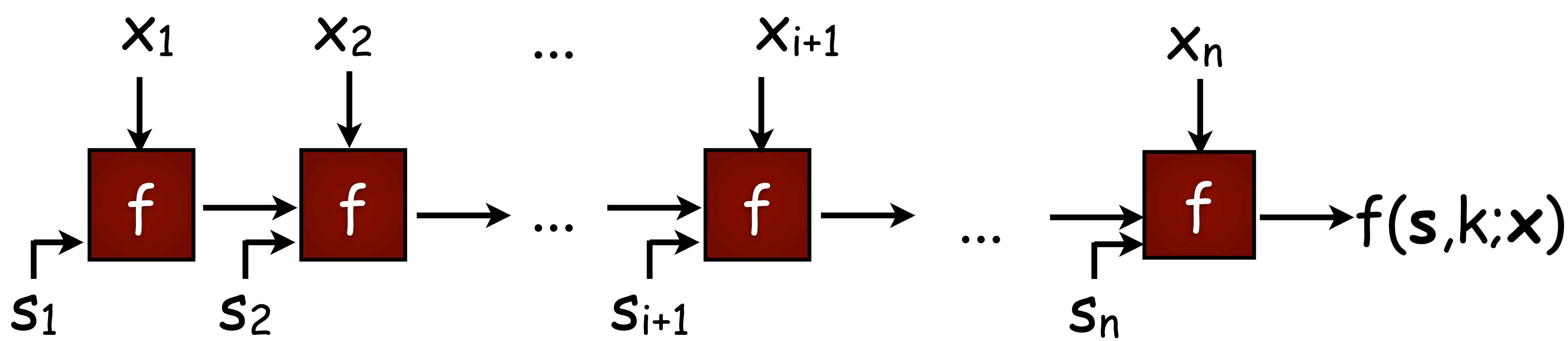


Finally:

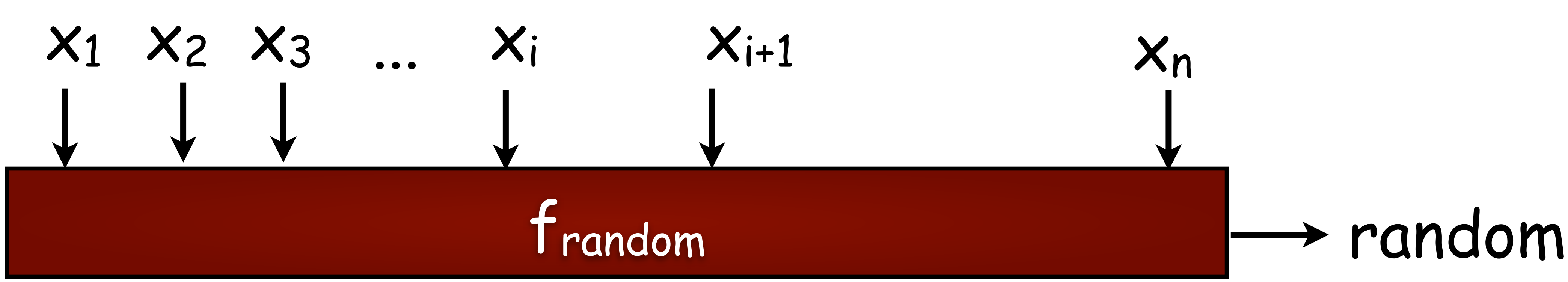
Proof Outline



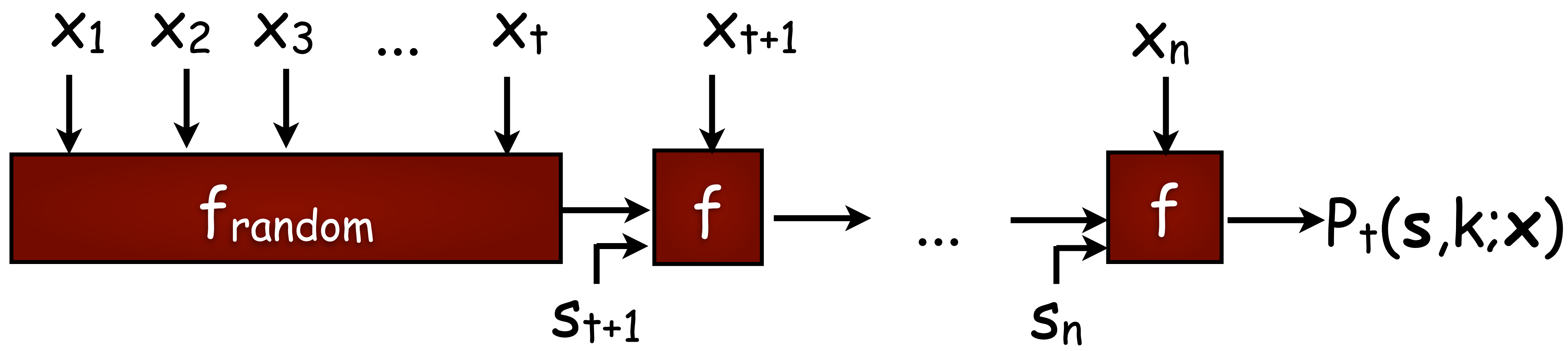
Initially:



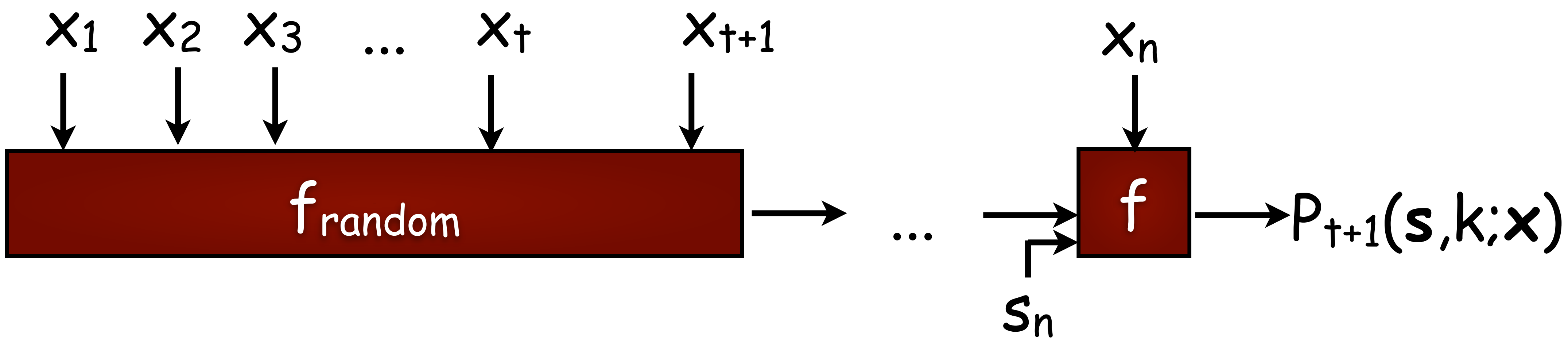
Finally:



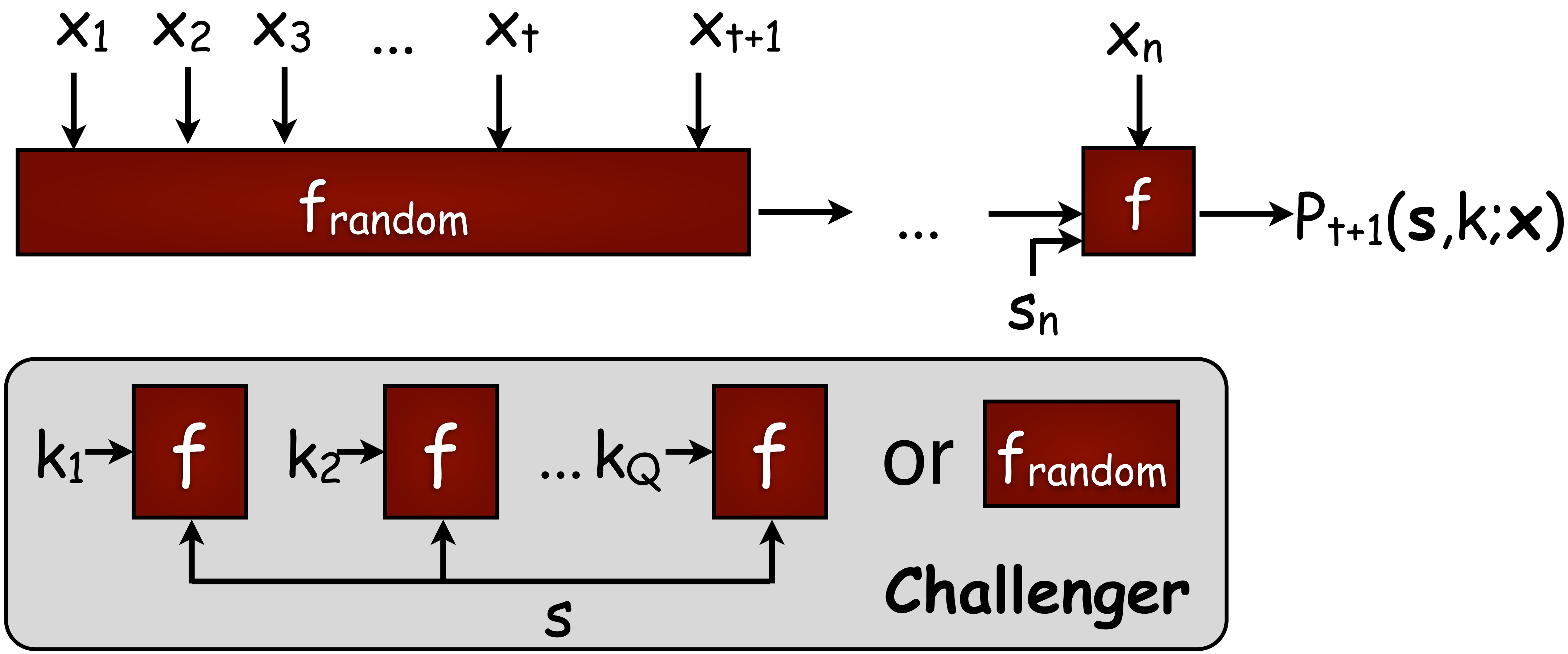
Proof Outline



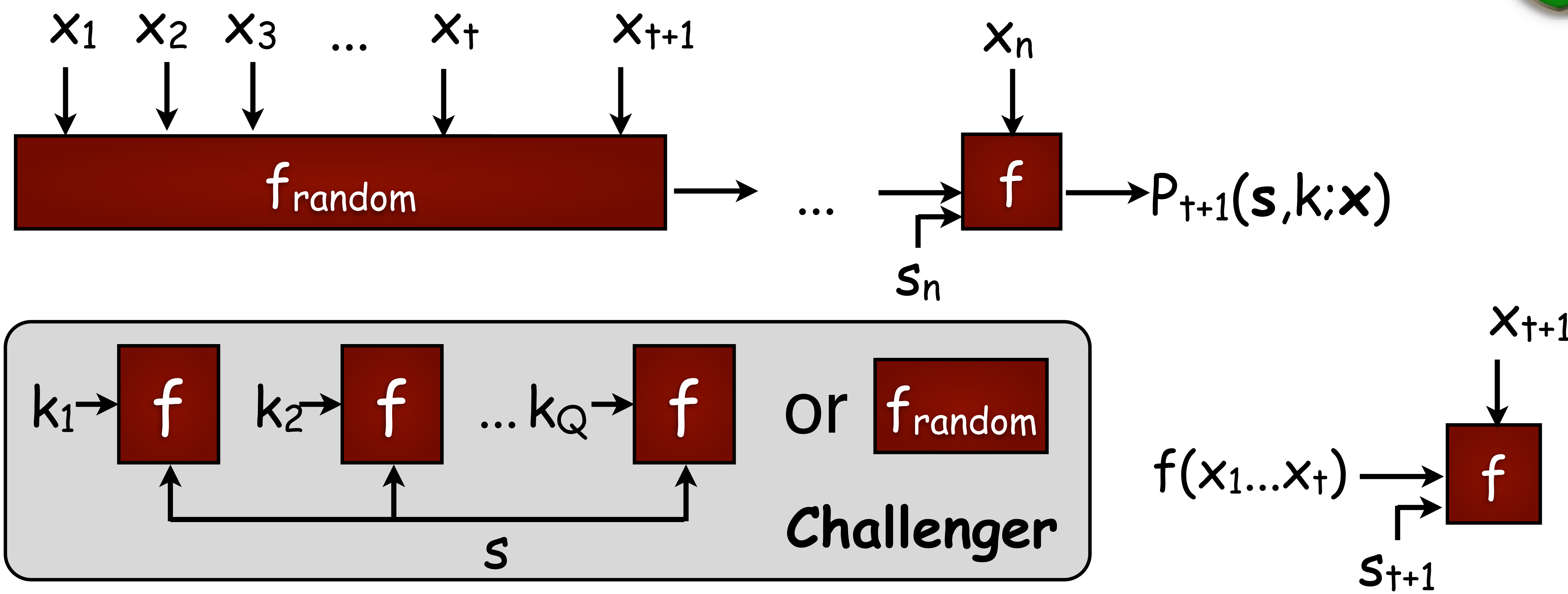
Proof Outline



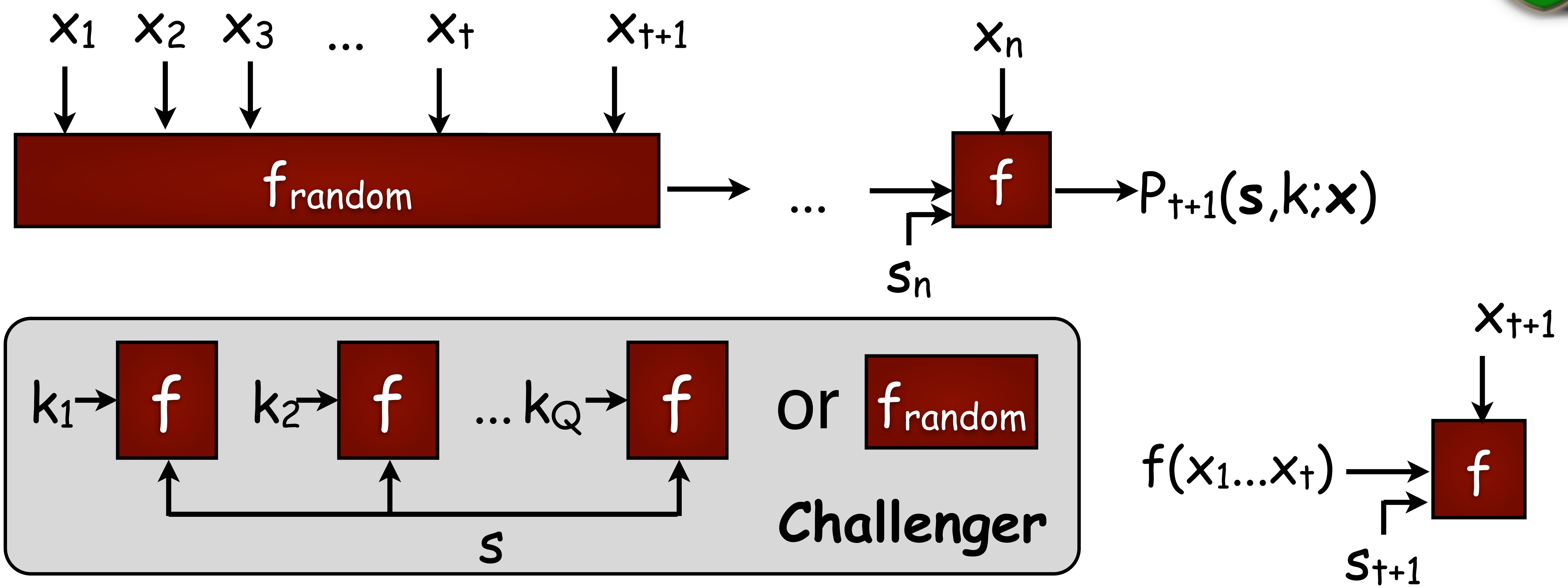
Proof Outline



Proof Outline



Proof Outline



s_{t+1} is set to s

Depending on $x_1 \ldots x_t$ we choose the value of $f(x_1 \ldots x_t)$ to be some key k_j – maintained in an associative table

Easy to see: We simulate either P_t or P_{t+1} depending upon Challenger

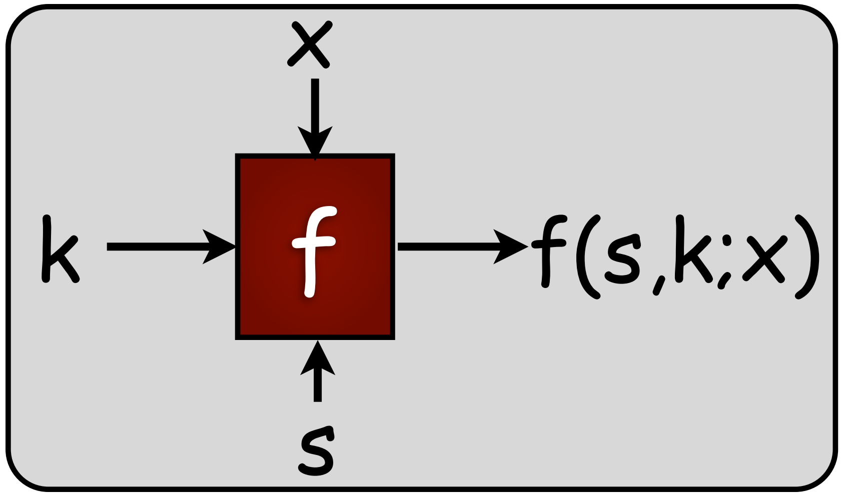
Why Augmented Cascade?

Why Augmented Cascade?

- Generic tool to build efficient algebraic PRFs

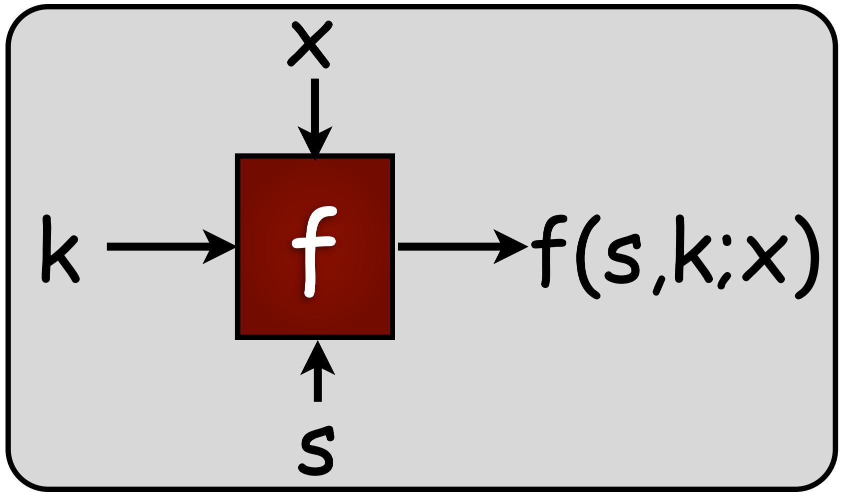
Why Augmented Cascade?

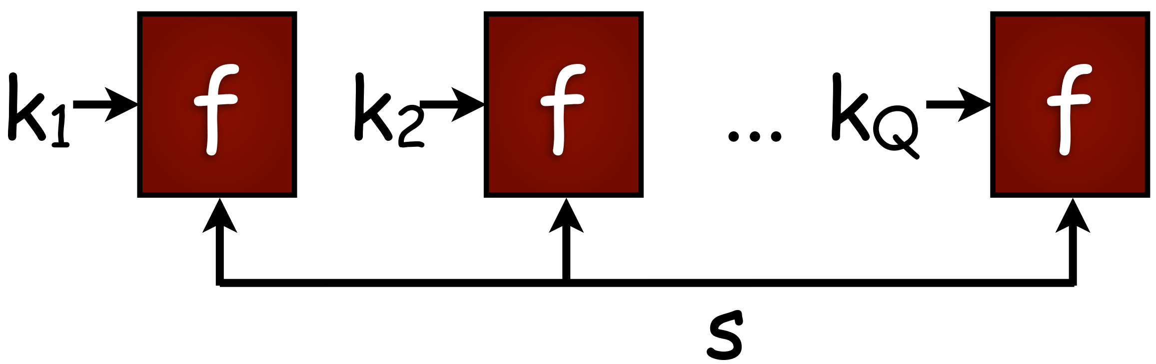
- Generic tool to build efficient algebraic PRFs



Why Augmented Cascade?

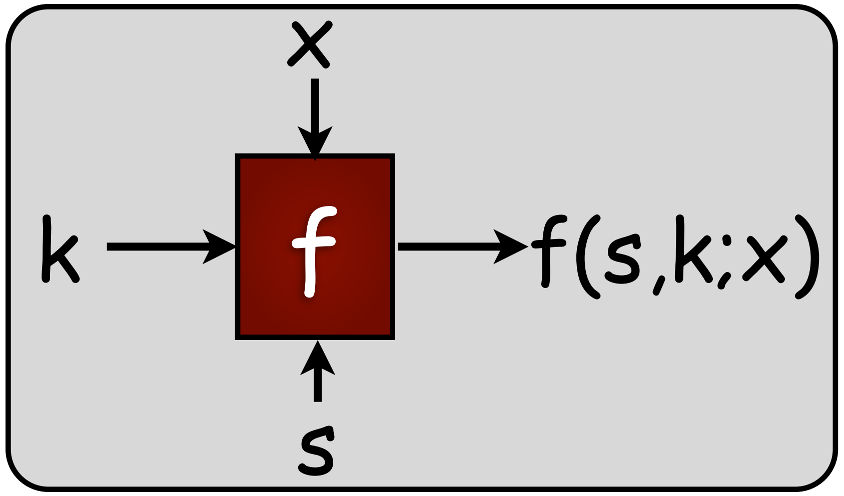
- Generic tool to build efficient algebraic PRFs

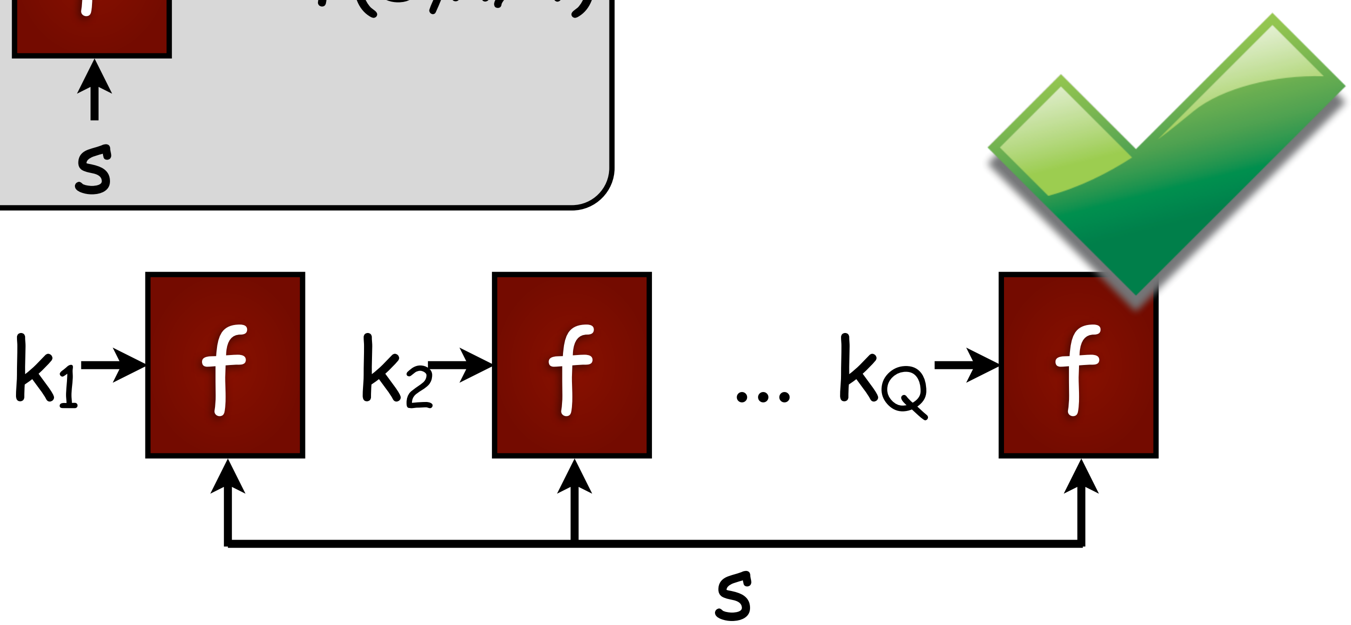




Why Augmented Cascade?

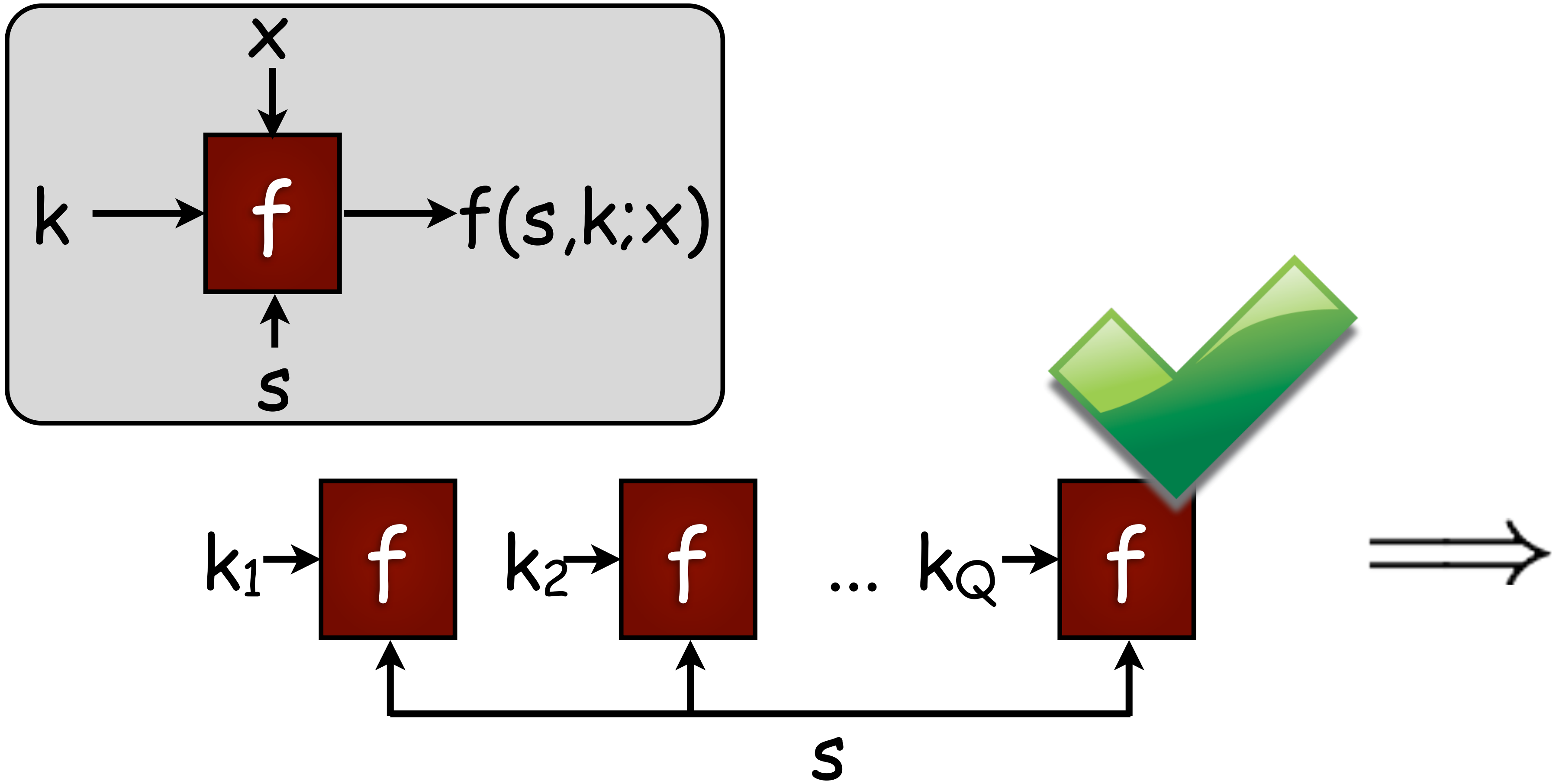
- Generic tool to build efficient algebraic PRFs





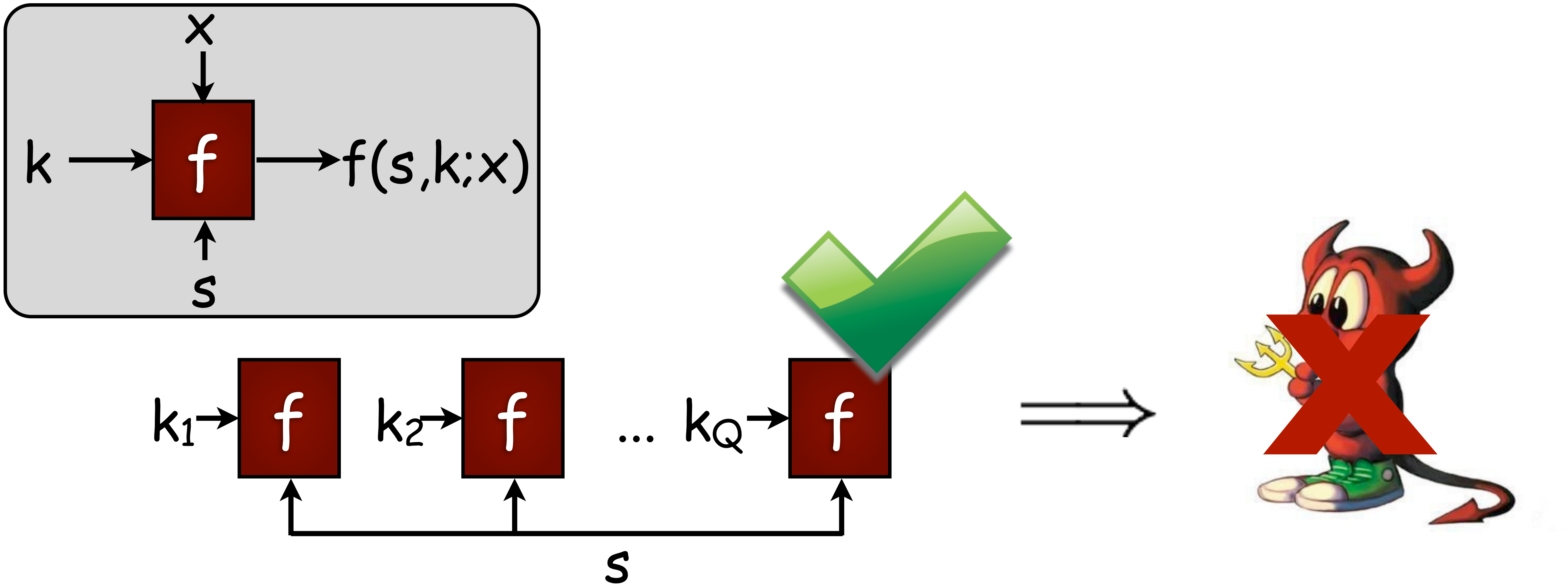
Why Augmented Cascade?

- Generic tool to build efficient algebraic PRFs



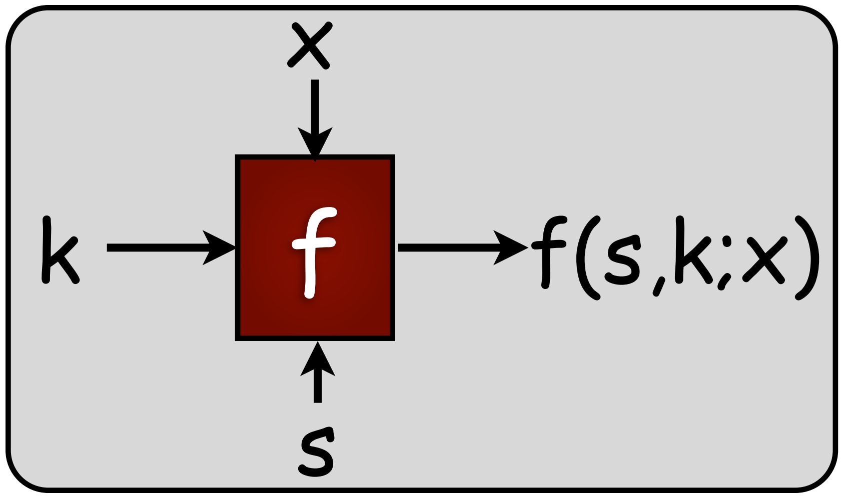
Why Augmented Cascade?

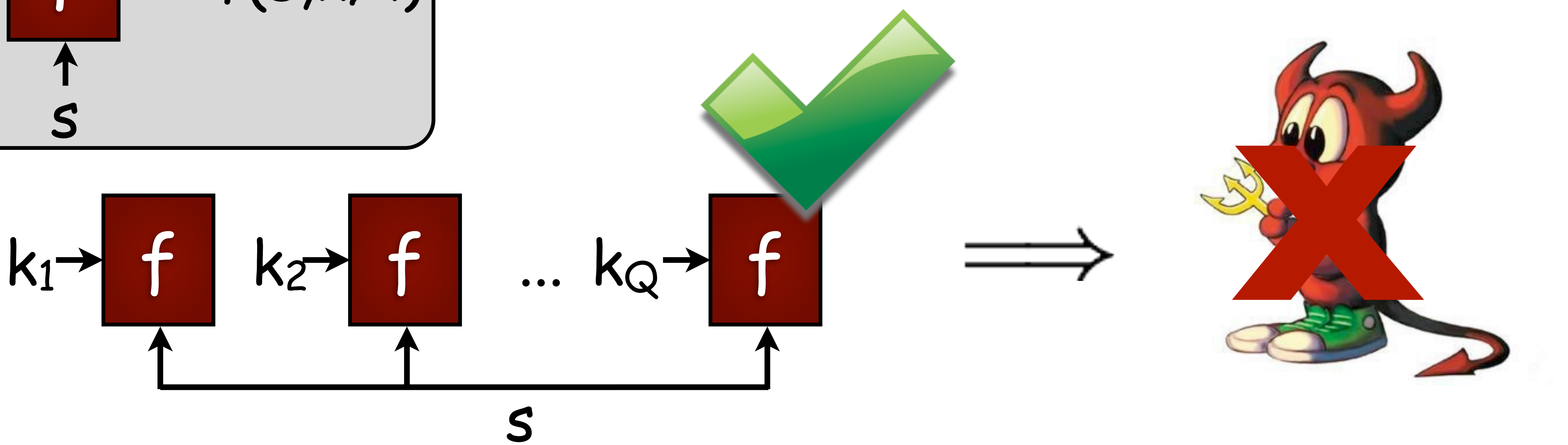
- Generic tool to build efficient algebraic PRFs



Why Augmented Cascade?

- Generic tool to build efficient algebraic PRFs

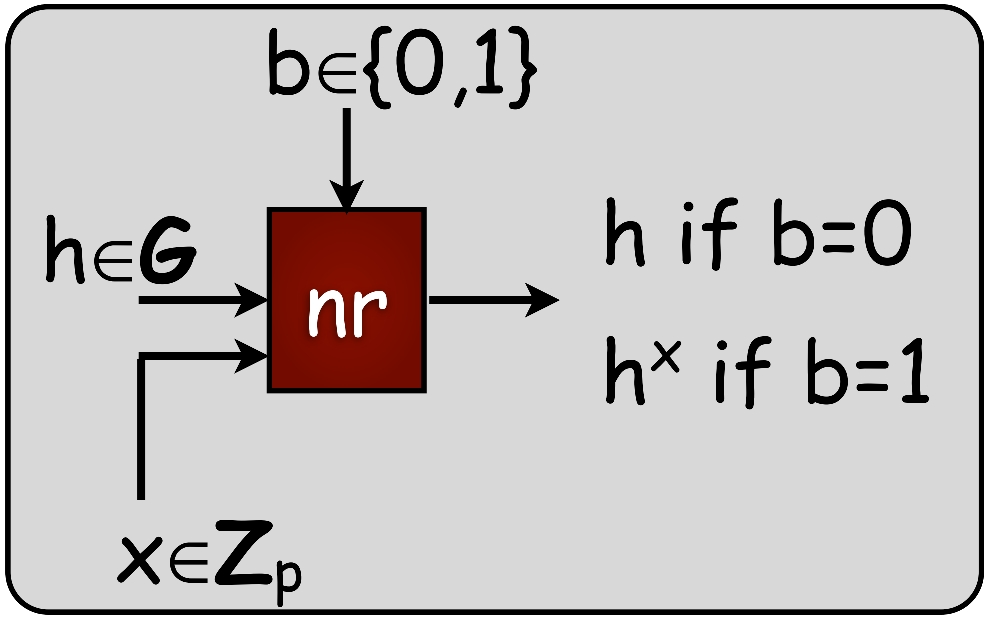




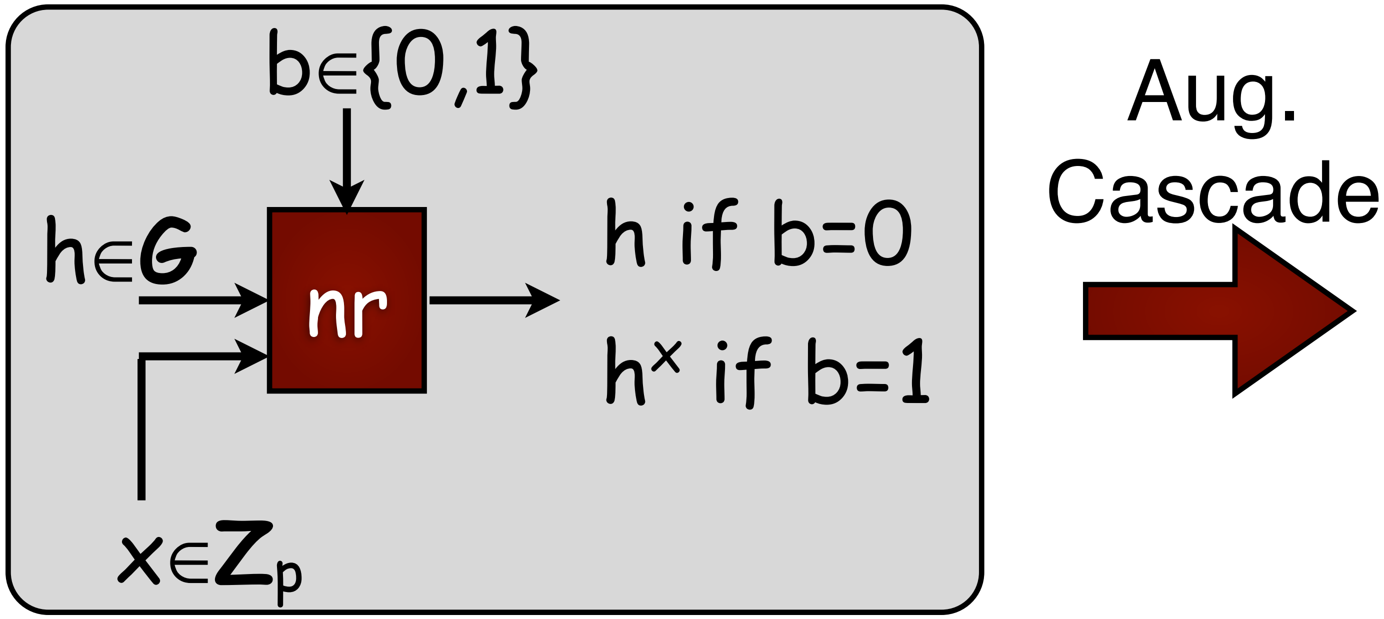
- Simple proofs for existing algebraic PRFs. We show q-parallel security of:
  Naor-Reingold [FOCS '97] & Lewko-Waters [CCS '09]

Naor-Reingold [FOCS '97]

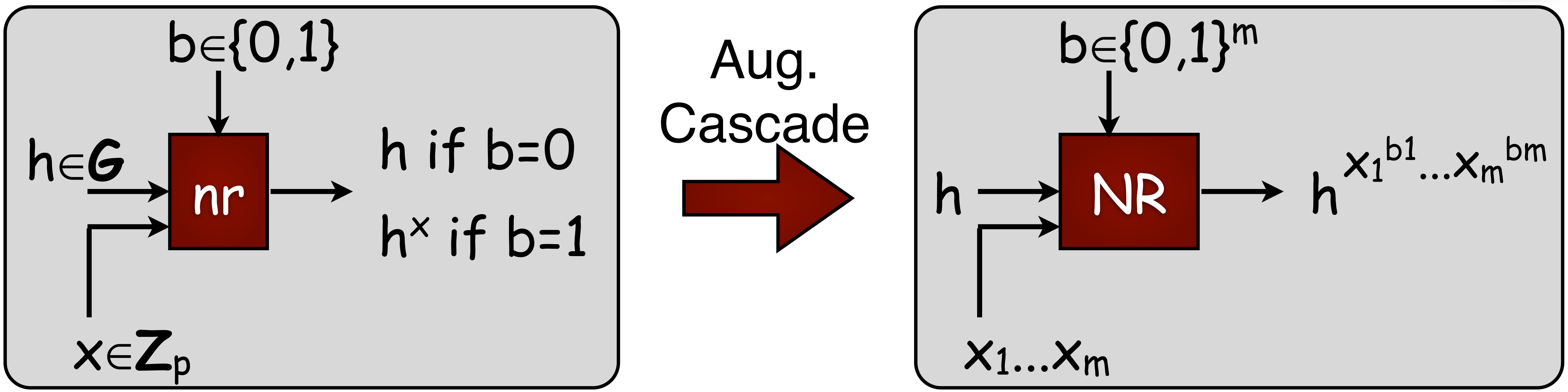
Naor-Reingold [FOCS '97]



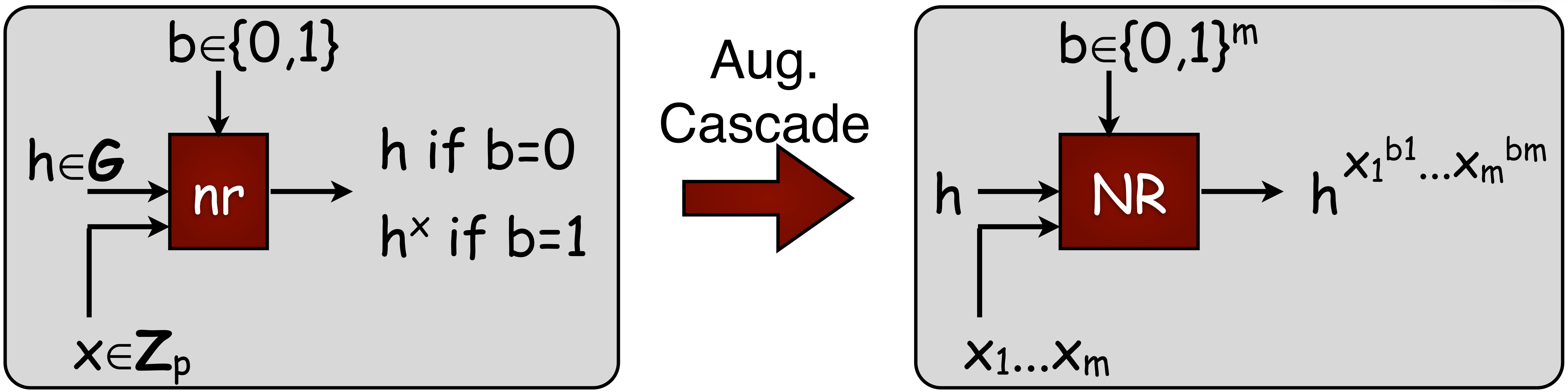
Naor-Reingold [FOCS '97]



Naor-Reingold [FOCS '97]

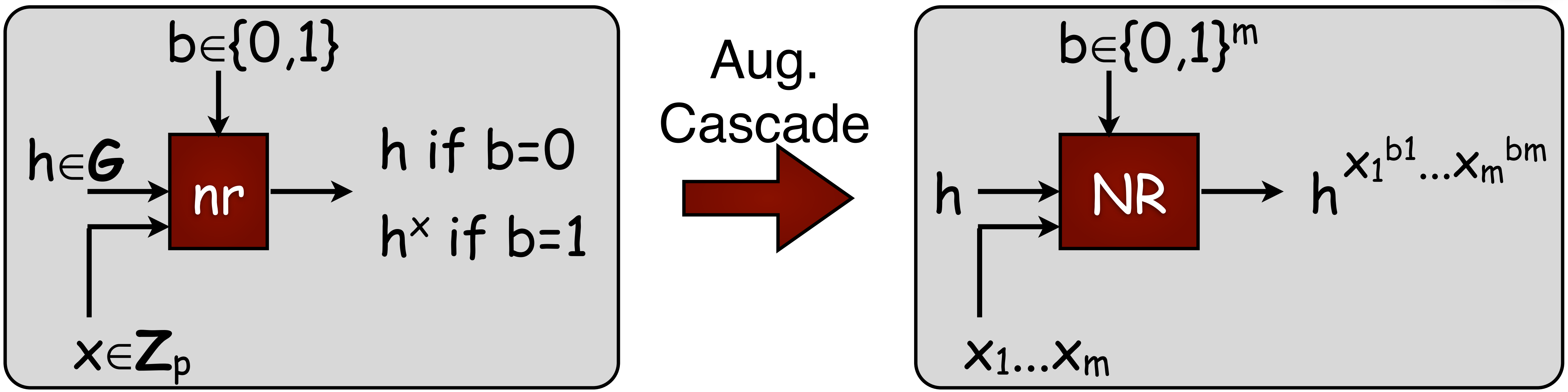


Naor-Reingold [FOCS ’97]



Theorem: $\boxed{nr}$ is q-parallel secure under the DDH assumption

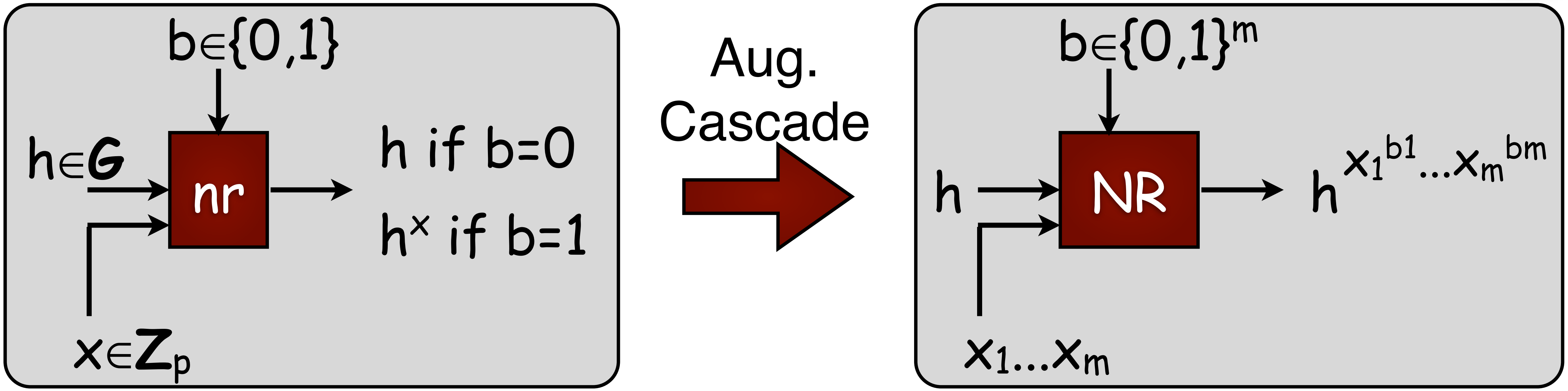
Naor-Reingold [FOCS '97]



Theorem: nr is q-parallel secure under the DDH assumption

Group **G** of prime order p with generator *g*.
g^{ab} looks random in G**,**given g^a,g^b for random *a*,*b*,

Naor-Reingold [FOCS '97]



Theorem: $\boxed{nr}$ is q-parallel secure under the DDH assumption

Group **G** of prime order p with generator *g*.
g^{ab} looks random in G, given g^a, g^b for random a,b,

Corollary: $\boxed{NR}$ is secure PRF under the DDH assumption

Lewko-Waters [CCS '09]

- Details in the paper

Lewko-Waters [CCS '09]

- Details in the paper

Theorem: $\boxed{\mathsf{lw}}$ is q-parallel secure under the k-linear assumption

Lewko-Waters [CCS ’09]

- Details in the paper

Theorem: $\boxed{\mathrm{lw}}$ is q-parallel secure under the k-linear assumption

Follows from the randomized self-reducibility (due to Lewko-Waters) of the k-linear assumption

Lewko-Waters [CCS '09]

- Details in the paper

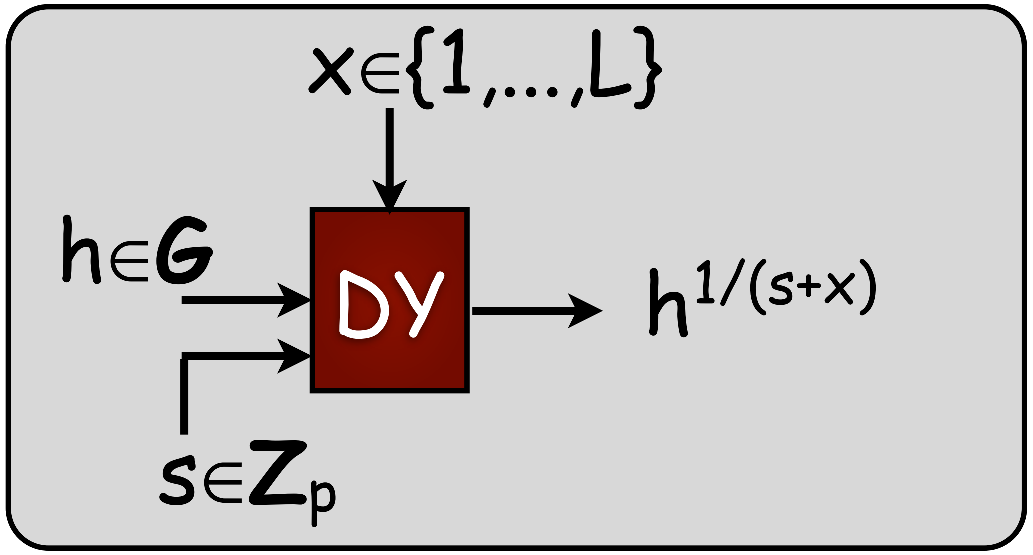
Theorem: lw is q-parallel secure under the k-linear assumption

Follows from the randomized self-reducibility (due to Lewko-Waters) of the k-linear assumption

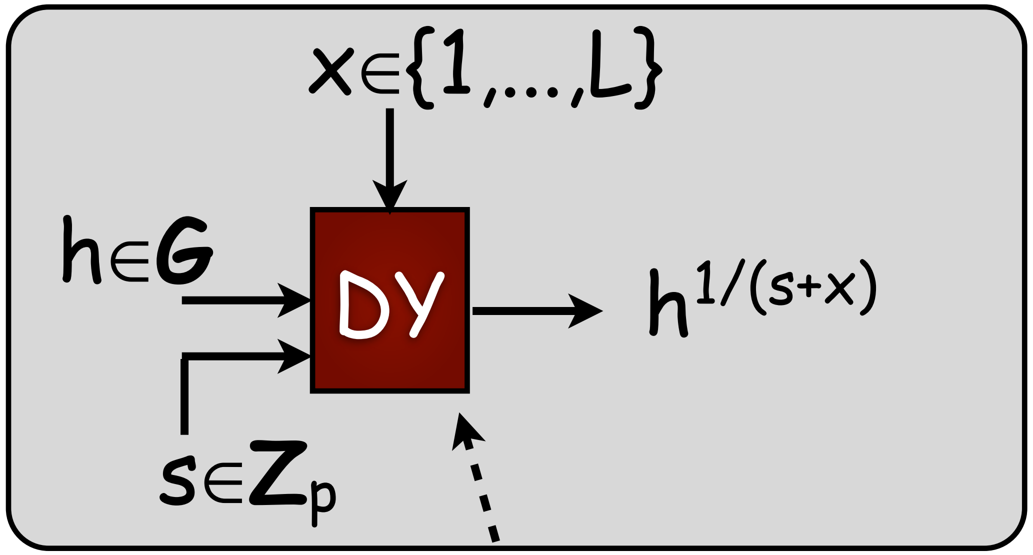
Corollary: LW is secure PRF under the k-linear assumption

New Algebraic PRF

New Algebraic PRF

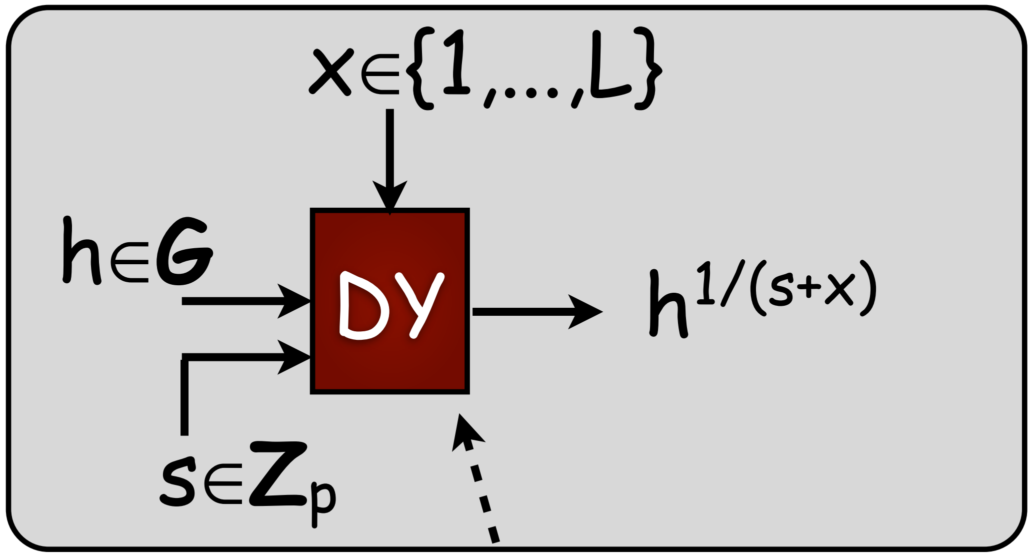


New Algebraic PRF



Dodis-Yampolskiy [PKC '05]

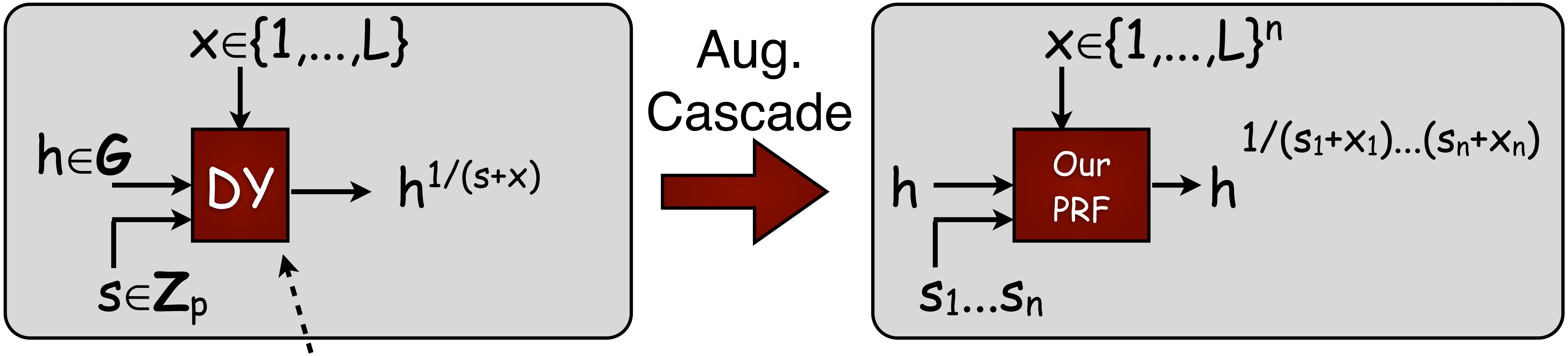
New Algebraic PRF



Aug.
Cascade

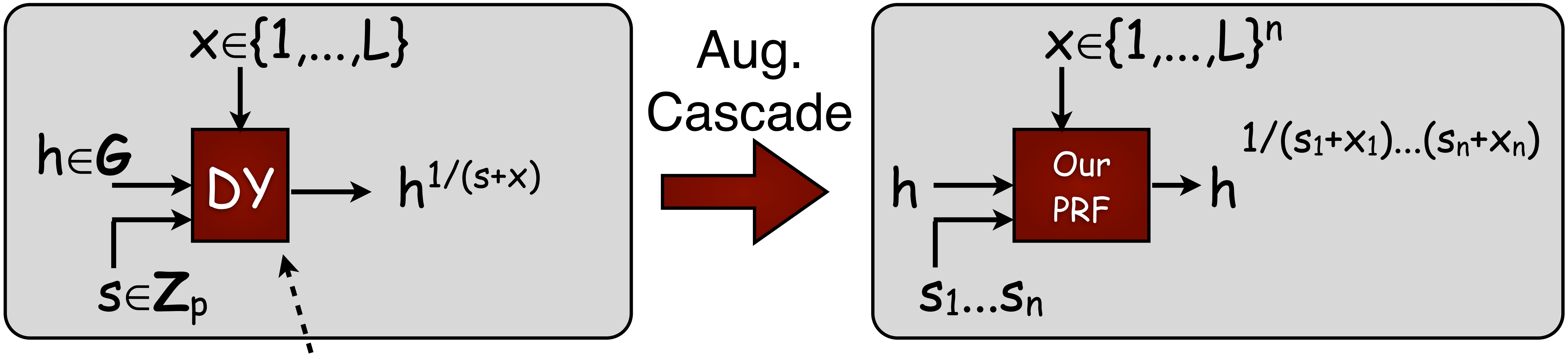
Dodis-Yampolskiy [PKC '05]

New Algebraic PRF



Dodis-Yampolskiy [PKC '05]

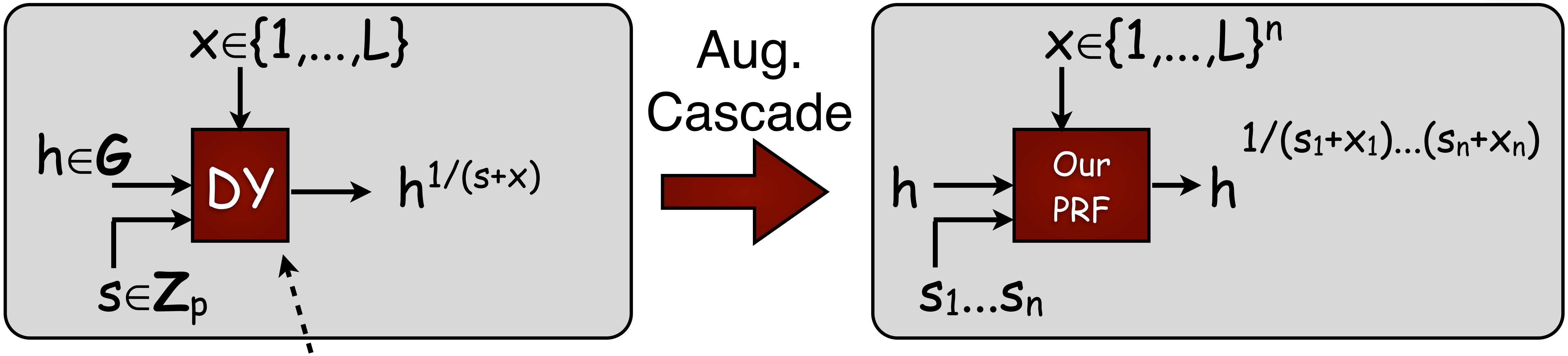
New Algebraic PRF



Dodis-Yampolskiy [PKC ’05]

Theorem: DY is q-parallel secure under the L-DDH (inversion) assumption

New Algebraic PRF

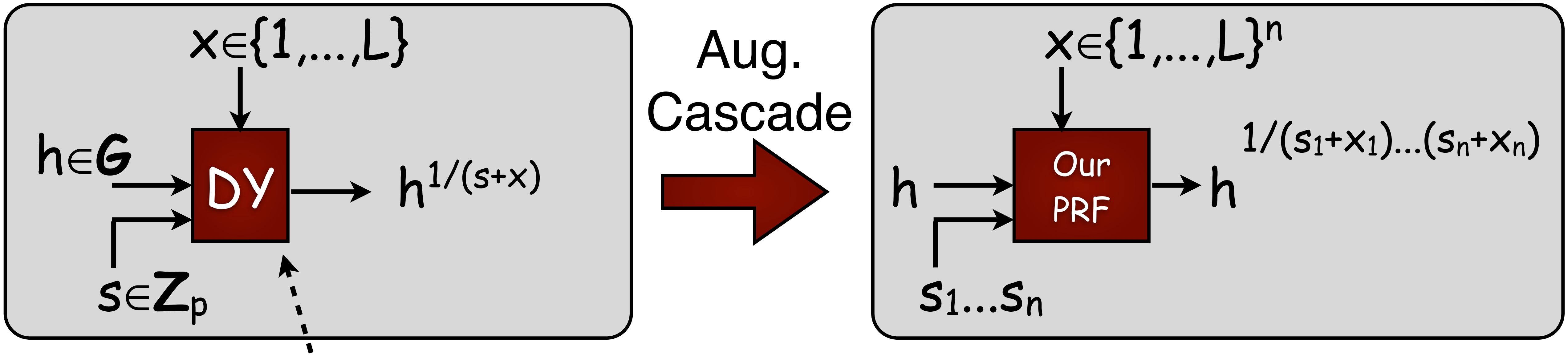


Dodis-Yampolskiy [PKC '05]

Theorem: DY is q-parallel secure under the L-DDH (inversion) assumption

$g^{1/x}$ looks random in **G** given $(g, g^x, g^{x^2}, \ldots, g^{x^L})$

New Algebraic PRF



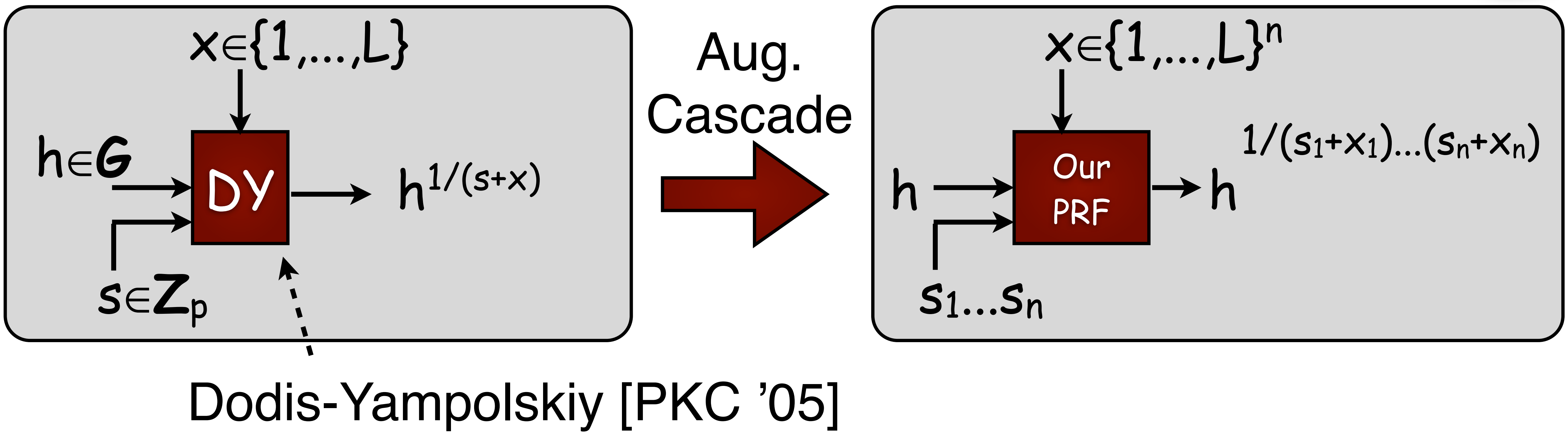
Dodis-Yampolskiy [PKC '05]

Theorem: $\boxed{\text{DY}}$ is q-parallel secure under the L-DDH (inversion) assumption

$g^{1/x}$ looks random in **G** given $(g, g^x, g^{x^2}, \ldots, g^{x^L})$

(Details in the paper)

New Algebraic PRF



Theorem: DY is q-parallel secure under the L-DDH (inversion) assumption

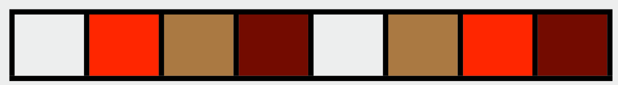
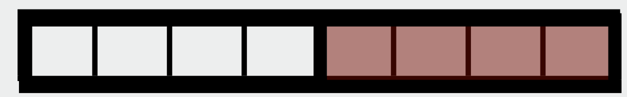
(Details in the paper)

$g^{1/x}$ looks random in **G** given $(g, g^x, g^{x^2}, \ldots, g^{x^L})$

Corollary: Our PRF is secure PRF under the L-DDH assumption

Comparison to Naor-Reingold

Comparison to Naor-Reingold

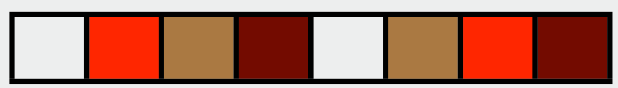
	Naor-Reingold PRF	Our PRF
Domain	$\{0,1\}^m$	$[L]^n$ ($L=2^l$, $n=m/l$)
Keys	h in $\mathbf{G}$, $x_1,\ldots,x_m$ in $\mathbf{Z}_p$	h in $\mathbf{G}$, $s_1,\ldots,s_n$ in $\mathbf{Z}_p$
PRF	 $\beta = x_1^{\square} . x_2^{\blacksquare} . x_3^{\blacksquare} . x_4^{\blacksquare} . x_5^{\square} . x_6^{\blacksquare} . x_7^{\blacksquare} . x_8^{\blacksquare}$ $F(x)=h^{\beta}$	 $\beta=(s_1 + \boxed{\square\square\square\square})(s_2+\boxed{\blacksquare\blacksquare\blacksquare\blacksquare})$ $F(x)=h^{1/\beta}$

Comparison to Naor-Reingold

	Naor-Reingold PRF	Our PRF
Domain	$\{0,1\}^m$	$[L]^n$ ($L=2^l$, $n=m/l$)
Keys	h in $\mathbf{G}$, $x_1,\ldots,x_m$ in $\mathbf{Z}_p$	h in $\mathbf{G}$, $s_1,\ldots,s_n$ in $\mathbf{Z}_p$
PRF	$\beta = x_1 . x_2 . x_3 . x_4 . x_5 . x_6 . x_7 . x_8$ $F(x)=h^{\beta}$	$\beta=(s_1 + \boxed{})(s_2+\boxed{})$ $F(x)=h^{1/\beta}$

Secret key shorter by a factor of l

Comparison to Naor-Reingold

	Naor-Reingold PRF	Our PRF
Domain	$\{0,1\}^m$	$[L]^n$ ($L=2^l$, $n=m/l$)
Keys	h in $\mathbf{G}$, $x_1,\ldots,x_m$ in $\mathbf{Z}_p$	h in $\mathbf{G}$, $s_1,\ldots,s_n$ in $\mathbf{Z}_p$
PRF	 $\beta = x_1.x_2.x_3.x_4.x_5.x_6.x_7.x_8$ $F(x)=h^{\beta}$	 $\beta=(s_1+\boxed{\square\square\square\square})(s_2+\boxed{\blacksquare\blacksquare\blacksquare\blacksquare})$ $F(x)=h^{1/\beta}$

Secret key shorter by a factor of l

Factor of l fewer multiplications to evaluate ß

Comparison to Naor-Reingold

	Naor-Reingold PRF	Our PRF
Domain	$\{0,1\}^m$	$[L]^n$ ($L=2^l$, $n=m/l$)
Keys	h in $\mathbf{G}$, $x_1,\ldots,x_m$ in $\mathbf{Z}_p$	h in $\mathbf{G}$, $s_1,\ldots,s_n$ in $\mathbf{Z}_p$
PRF	$\beta = x_1 . x_2 . x_3 . x_4 . x_5 . x_6 . x_7 . x_8$ $F(x)=h^{\beta}$	$\beta=(s_1+\boxed{\square\square\square\square})(s_2+\boxed{\blacksquare\blacksquare\blacksquare\blacksquare})$ $F(x)=h^{1/\beta}$

Secret key shorter by a factor of l

Factor of l fewer multiplications to evaluate ß

However, security becomes weaker as l increases

Comparison to Naor-Reingold

	Naor-Reingold PRF	Our PRF
Domain	$\{0,1\}^m$	$[L]^n$ ($L=2^l$, $n=m/l$)
Keys	h in $\mathbf{G}$, $x_1,\ldots,x_m$ in $\mathbf{Z}_p$	h in $\mathbf{G}$, $s_1,\ldots,s_n$ in $\mathbf{Z}_p$
PRF	$\beta = x_1 . x_2 . x_3 . x_4 . x_5 . x_6 . x_7 . x_8$ $F(x)=h^\beta$	$\beta = (s_1 + \boxed{})(s_2 + \boxed{})$ $F(x)=h^{1/\beta}$

Secret key shorter by a factor of l

Factor of l fewer multiplications to evaluate ß

However, security becomes weaker as l increases

$l = 4$ or 8 is reasonable

Verifiable Random Functions

Verifiable Random Functions

Verifiable Random Functions

Verifiable Random Functions

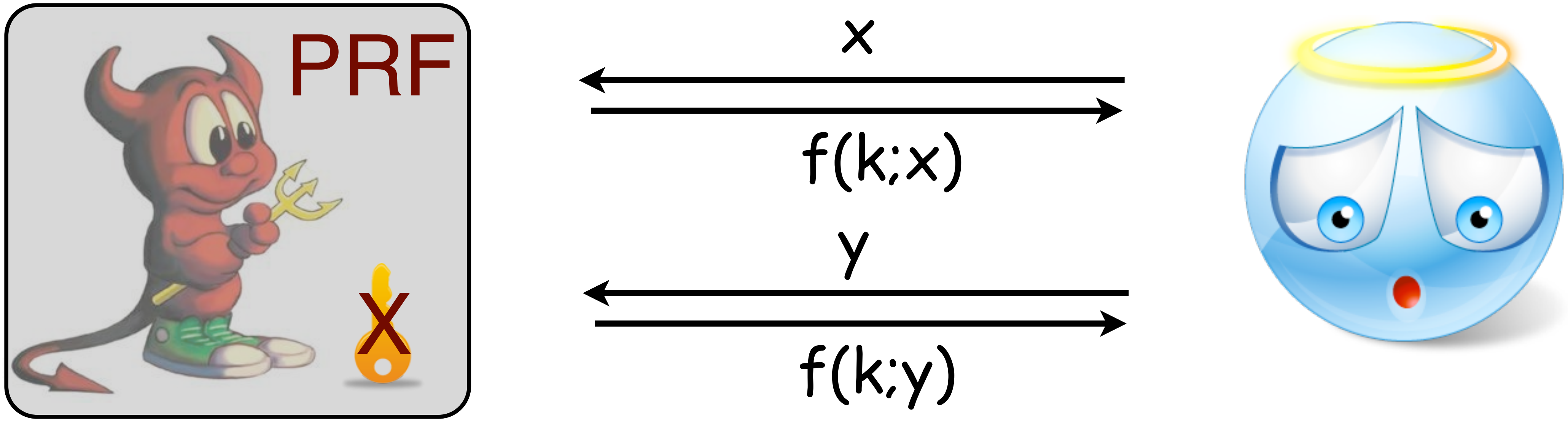
$\xleftarrow{x}$

$\xrightarrow[f(k;x)]{}$

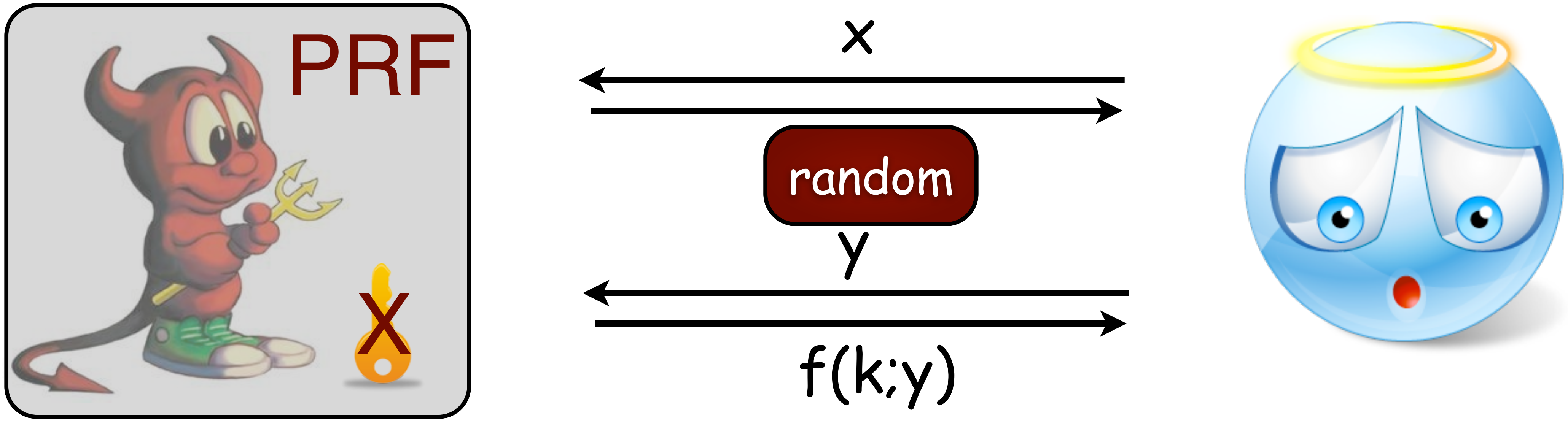
$\xleftarrow{y}$

$\xrightarrow[f(k;y)]{}$

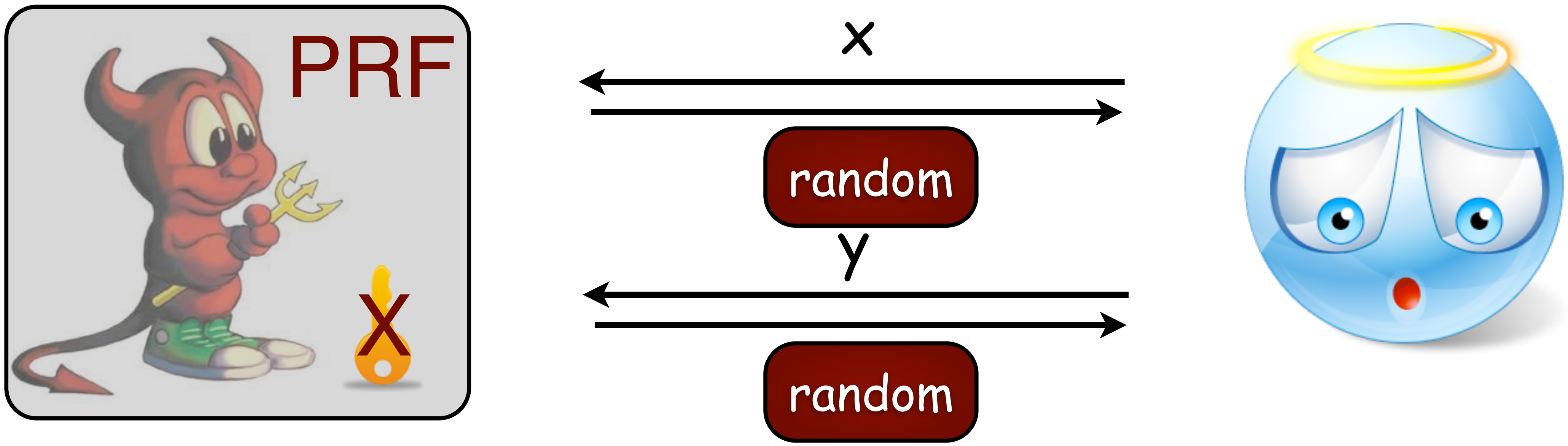
Verifiable Random Functions



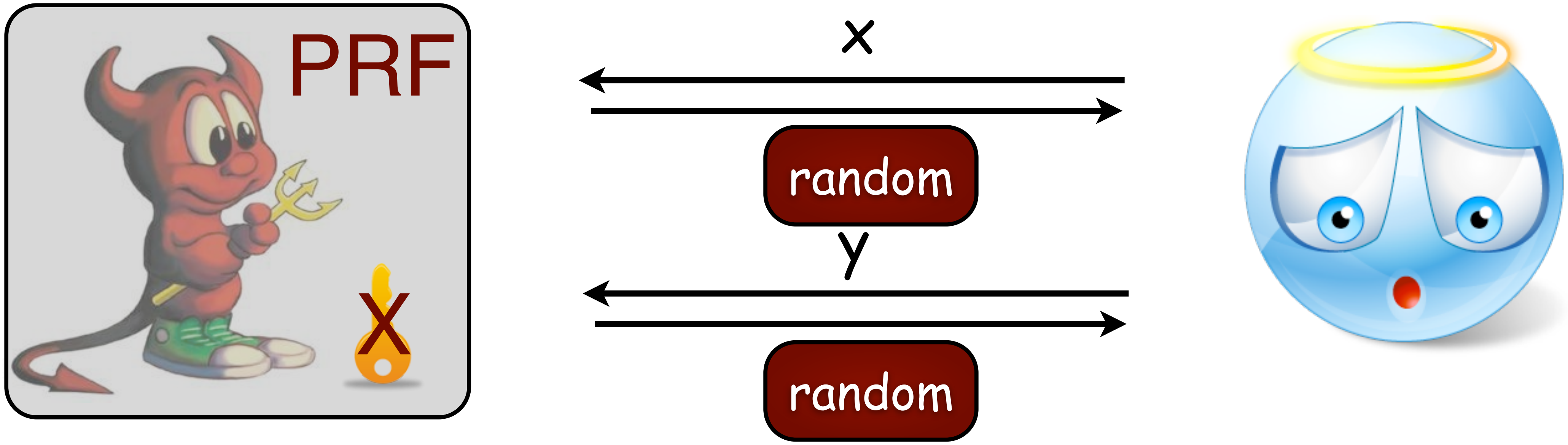
Verifiable Random Functions



Verifiable Random Functions

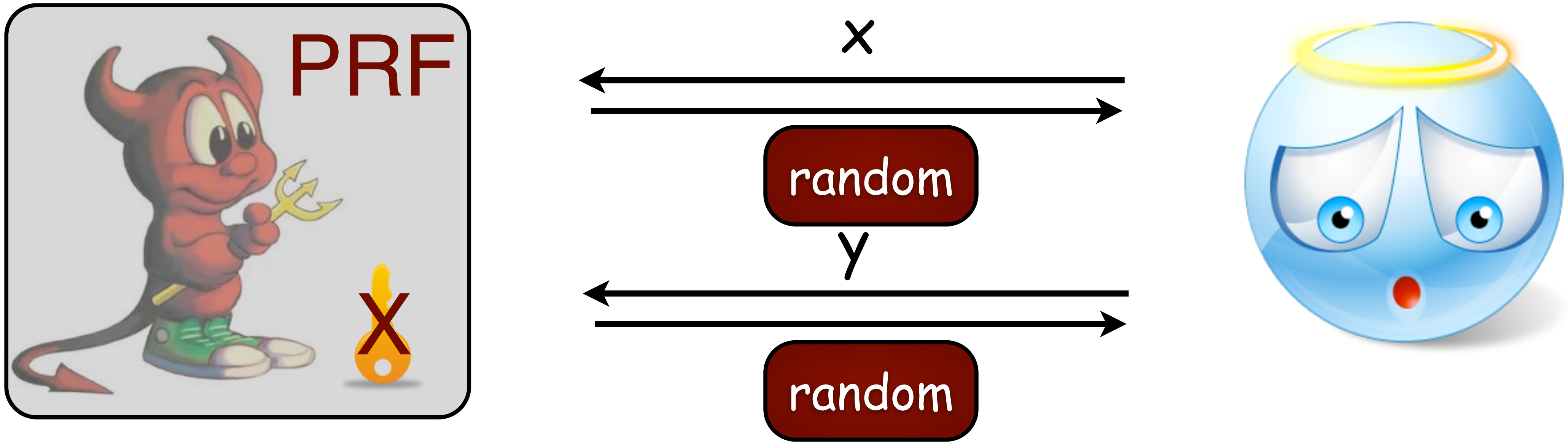


Verifiable Random Functions



How to guarantee honest evaluation?

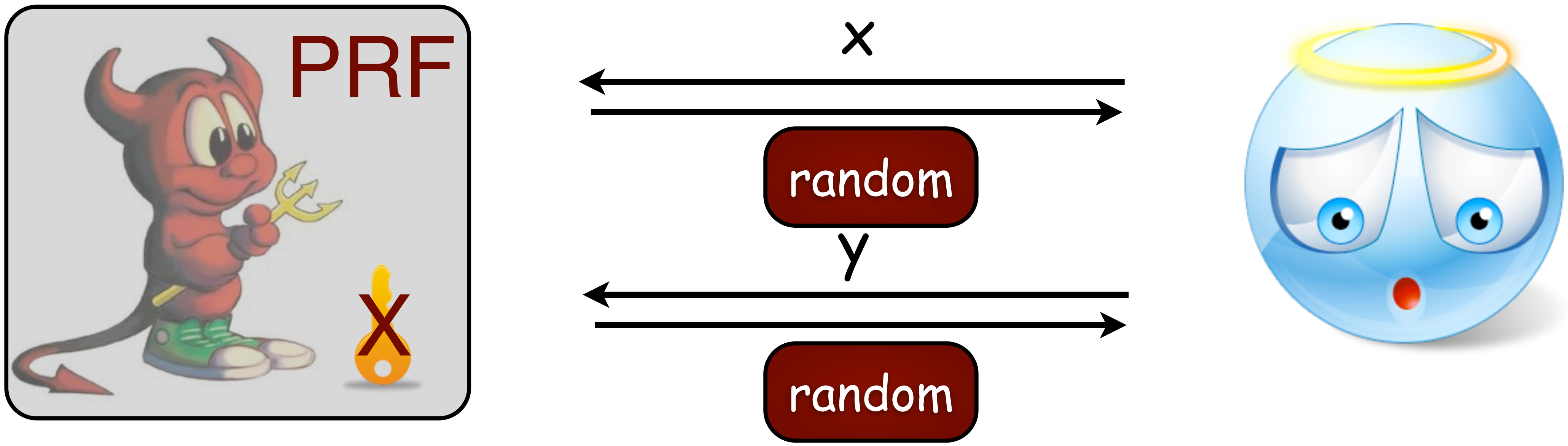
Verifiable Random Functions



How to guarantee honest evaluation?

- Introduced by Micali-Rabin-Vadhan [FOCS ’99]

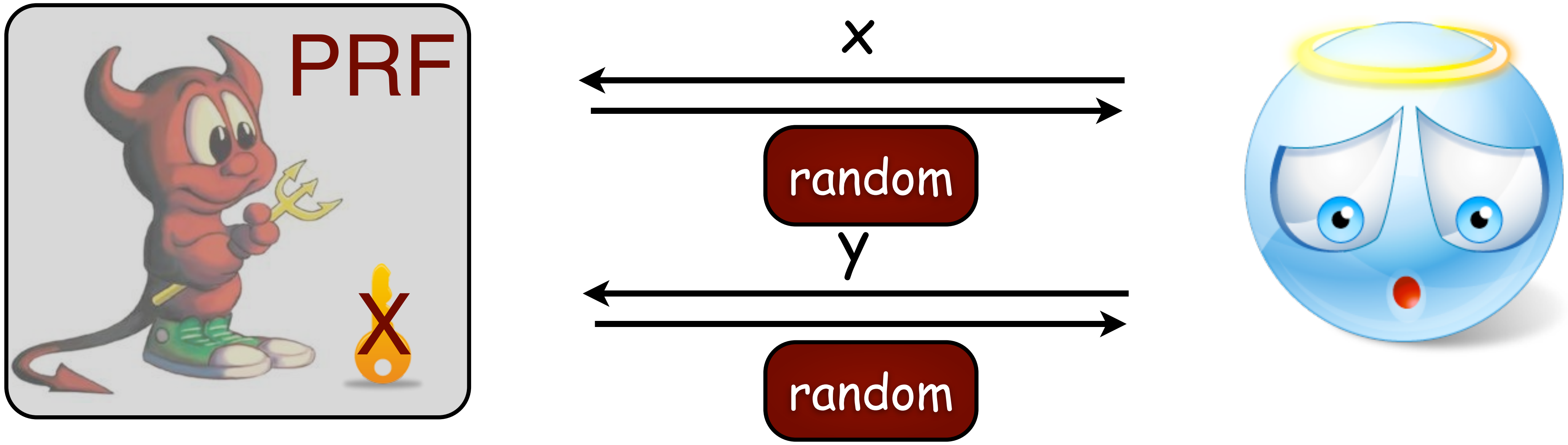
Verifiable Random Functions



How to guarantee honest evaluation?

- Introduced by Micali-Rabin-Vadhan [FOCS ’99]
- Produce non-interactive proofs (that can be verified by anyone) that the PRF is evaluated correctly

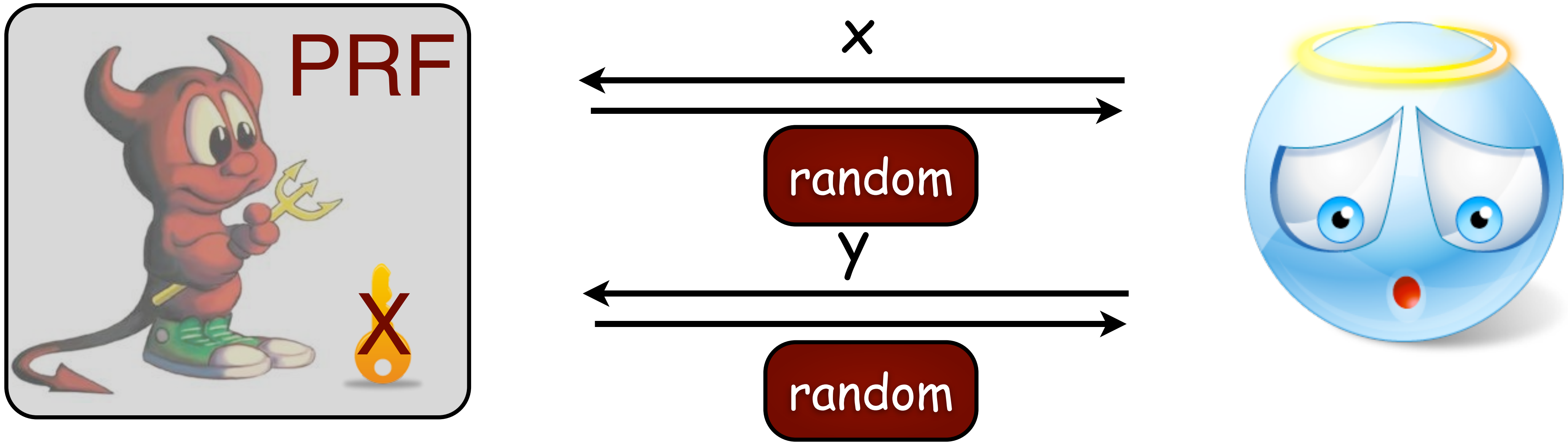
Verifiable Random Functions



How to guarantee honest evaluation?

- Introduced by Micali-Rabin-Vadhan [FOCS '99]
- Produce non-interactive proofs (that can be verified by anyone) that the PRF is evaluated correctly
- Not at the cost of pseudorandomness!

Verifiable Random Functions



How to guarantee honest evaluation?

- Introduced by Micali-Rabin-Vadhan [FOCS '99]
- Produce non-interactive proofs (that can be verified by anyone) that the PRF is evaluated correctly
- Not at the cost of pseudorandomness!
- Applications: Unique-signature schemes, e-cash schemes [BCKL '09, ASM '07], updatable zero-knowledge databases [Liskov '05] etc.

Verifiable Random Functions

Verifiable Random Functions

- We construct VRFs using the augmented cascade

Verifiable Random Functions

- We construct VRFs using the augmented cascade
- Security from Bilinear Diffie-Hellman Inversion assumption

Verifiable Random Functions

- We construct VRFs using the augmented cascade
- Security from Bilinear Diffie-Hellman Inversion assumption

VRF	Domain Size	Assumption Size
[DY '05]	L^n (poly in sec. param.)	$\boldsymbol{L^n}$
our small-domain VRF	L^n (poly in sec. param.)	$\boldsymbol{nL}$

Verifiable Random Functions

- We construct VRFs using the augmented cascade
- Security from Bilinear Diffie-Hellman Inversion assumption

VRF	Domain Size	Assumption Size
[DY '05]	L^n (poly in sec. param.)	$\boldsymbol{L^n}$
our small-domain VRF	L^n (poly in sec. param.)	$\boldsymbol{nL}$

- Hohenberger-Waters [Eurocrypt '10] constructed an efficient large-domain VRF

Verifiable Random Functions

- We construct VRFs using the augmented cascade
- Security from Bilinear Diffie-Hellman Inversion assumption

VRF	Domain Size	Assumption Size
[DY '05]	L^n (poly in sec. param.)	$\boldsymbol{L^n}$
our small-domain VRF	L^n (poly in sec. param.)	$\boldsymbol{nL}$

- Hohenberger-Waters [Eurocrypt '10] constructed an efficient large-domain VRF
- Require weaker assumption than [HW '10] but less efficient

Verifiable Random Functions

- We construct VRFs using the augmented cascade
- Security from Bilinear Diffie-Hellman Inversion assumption

VRF	Domain Size	Assumption Size
[DY '05]	L^n (poly in sec. param.)	L^n
our small-domain VRF	L^n (poly in sec. param.)	nL

- Hohenberger-Waters [Eurocrypt '10] constructed an efficient large-domain VRF
- Require weaker assumption than [HW '10] but less efficient

VRF	Domain Size	Assumption Size
[HW '10]	2^m (arbitrary)	$O(mQ)$
our large-domain VRF	2^m (arbitrary)	$O(m)$

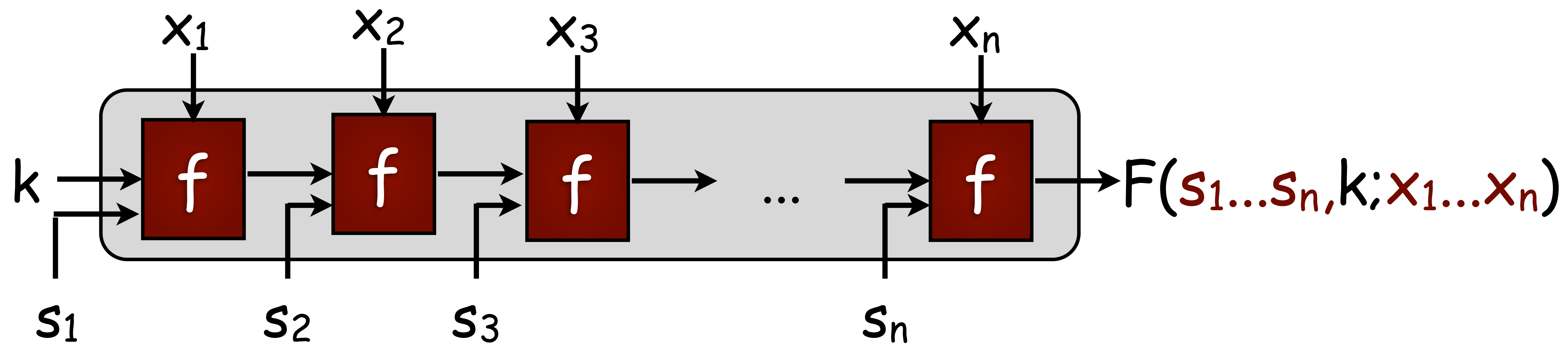
Conclusions

Conclusions

1) New generic tool to construct PRFs – **augmented cascade**

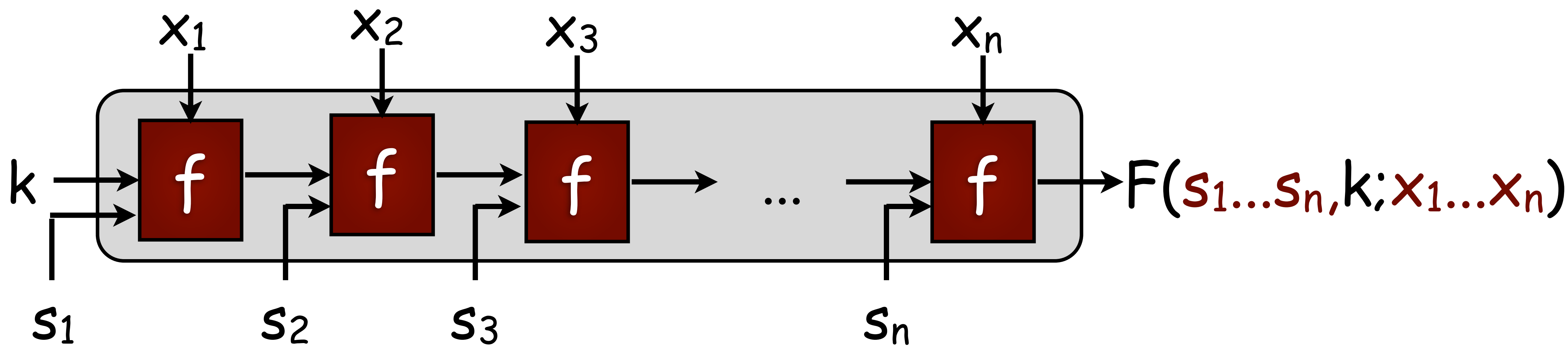
Conclusions

1) New generic tool to construct PRFs – **augmented cascade**



Conclusions

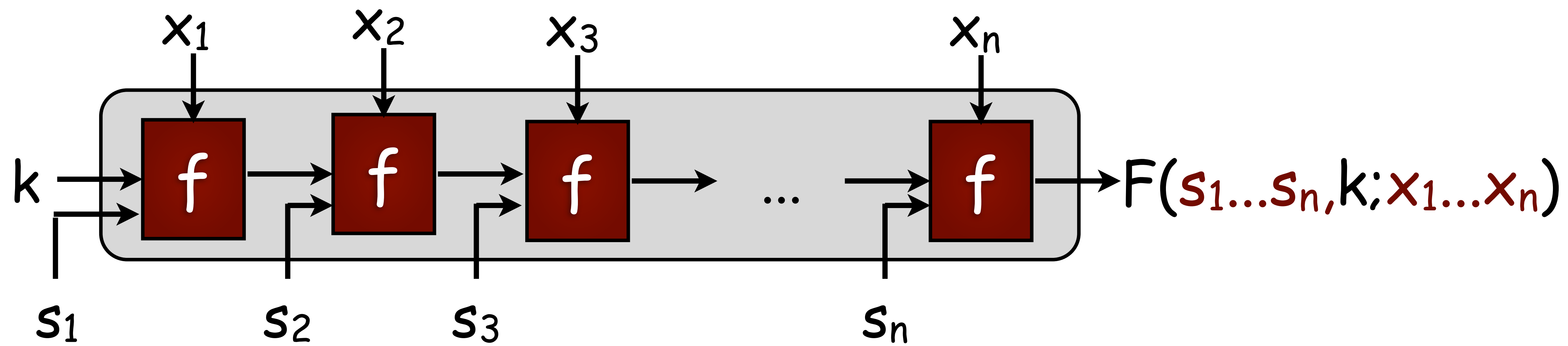
1) New generic tool to construct PRFs – **augmented cascade**



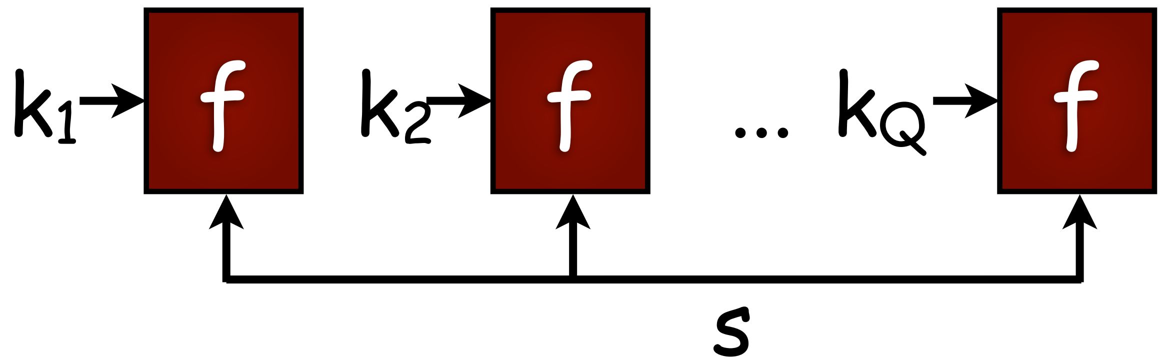
2) Simple proofs of security for existing algebraic PRFs

Conclusions

1) New generic tool to construct PRFs – **augmented cascade**

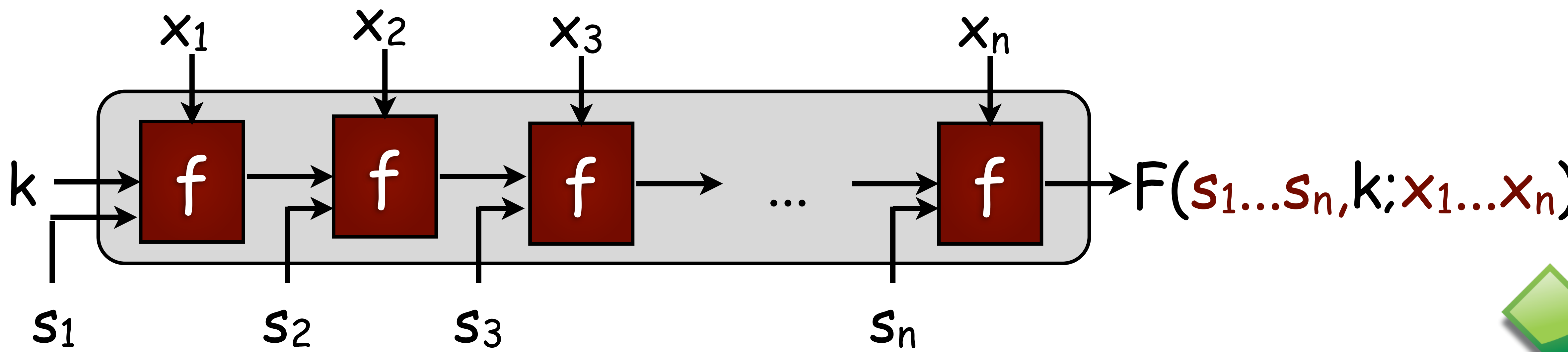


2) Simple proofs of security for existing algebraic PRFs

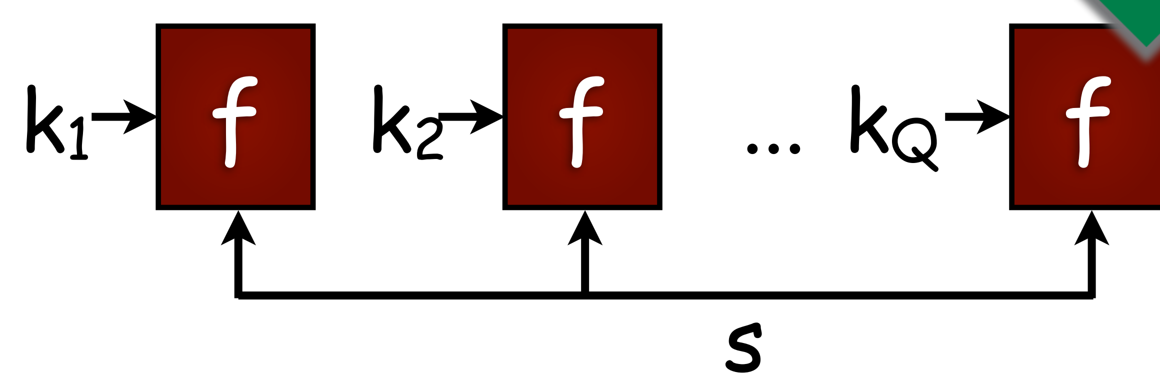


Conclusions

1) New generic tool to construct PRFs – **augmented cascade**

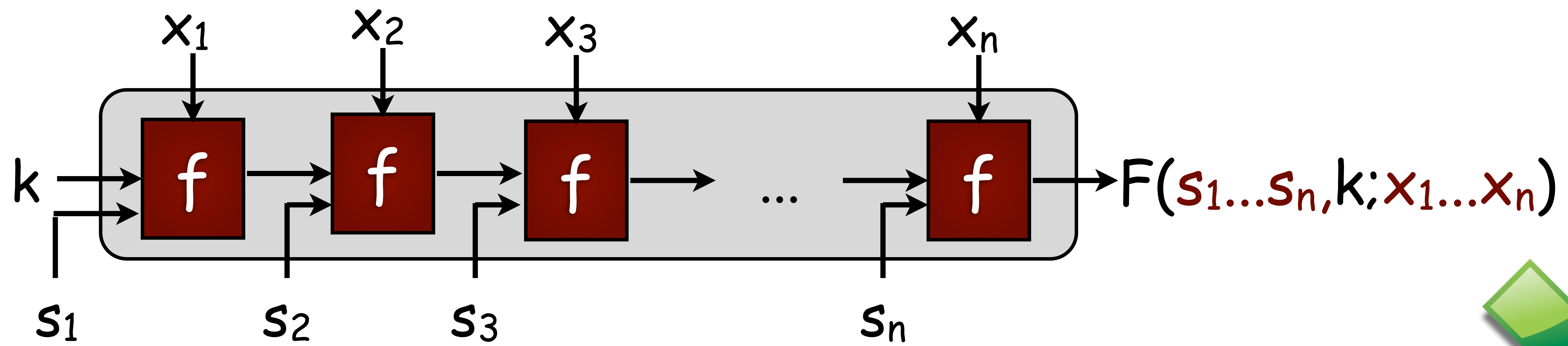


2) Simple proofs of security for existing algebraic PRFs



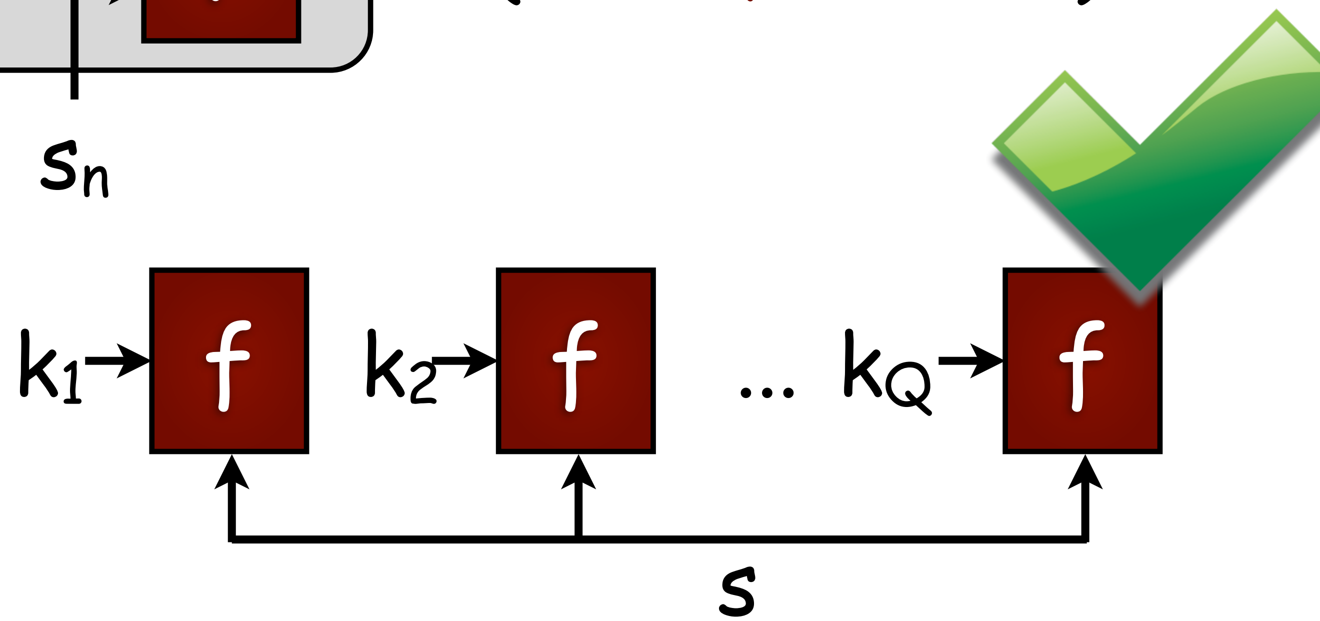
Conclusions

1) New generic tool to construct PRFs – **augmented cascade**



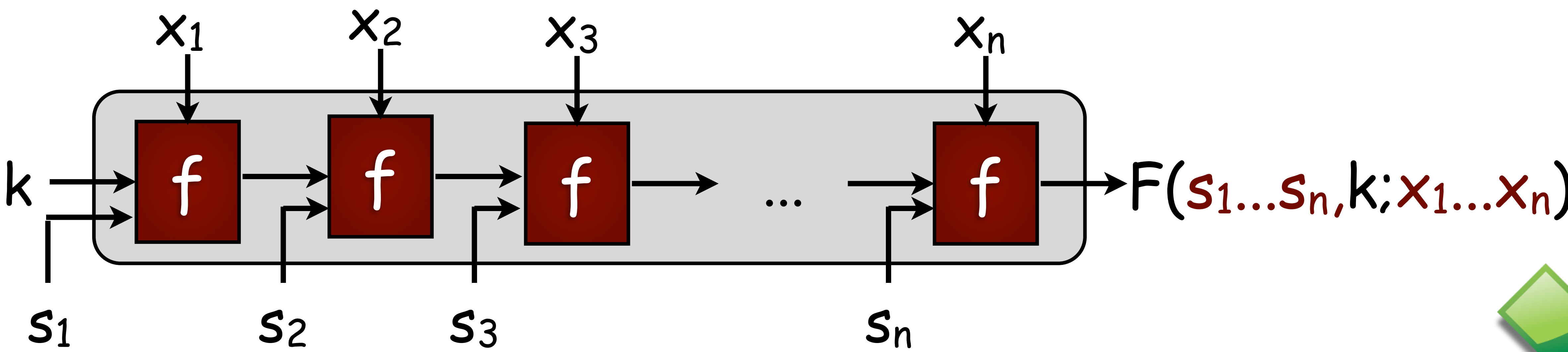
2) Simple proofs of security for existing algebraic PRFs

3) New efficient large-domain PRF

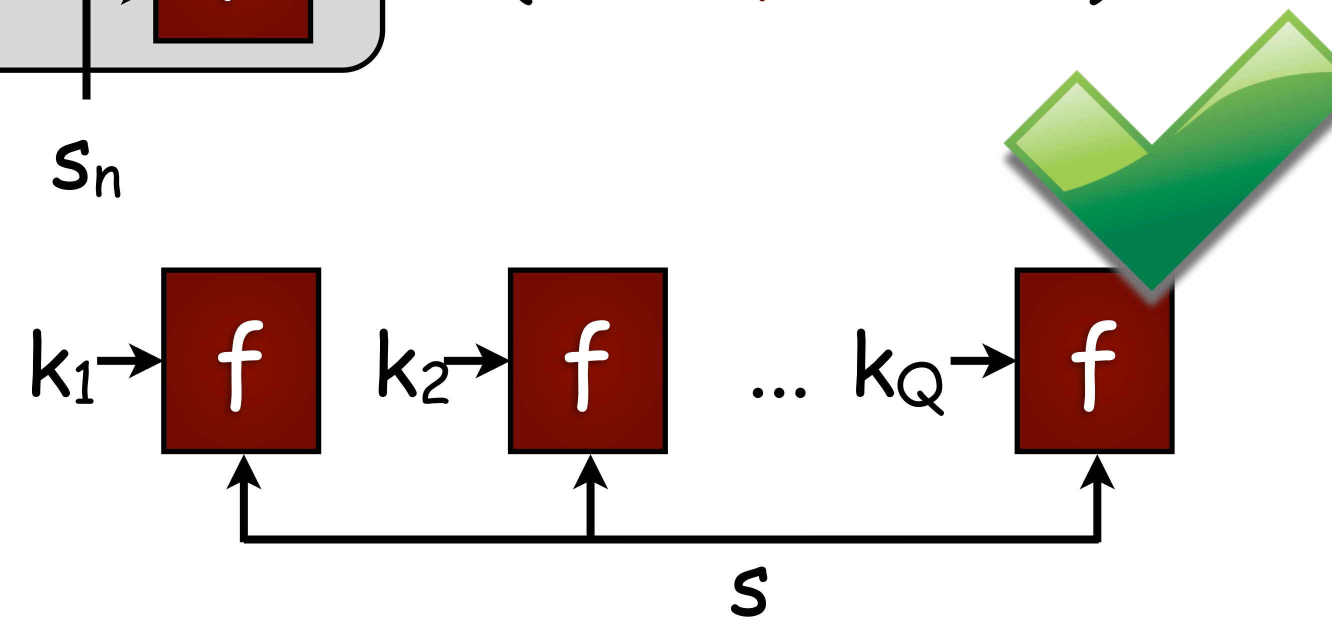


Conclusions

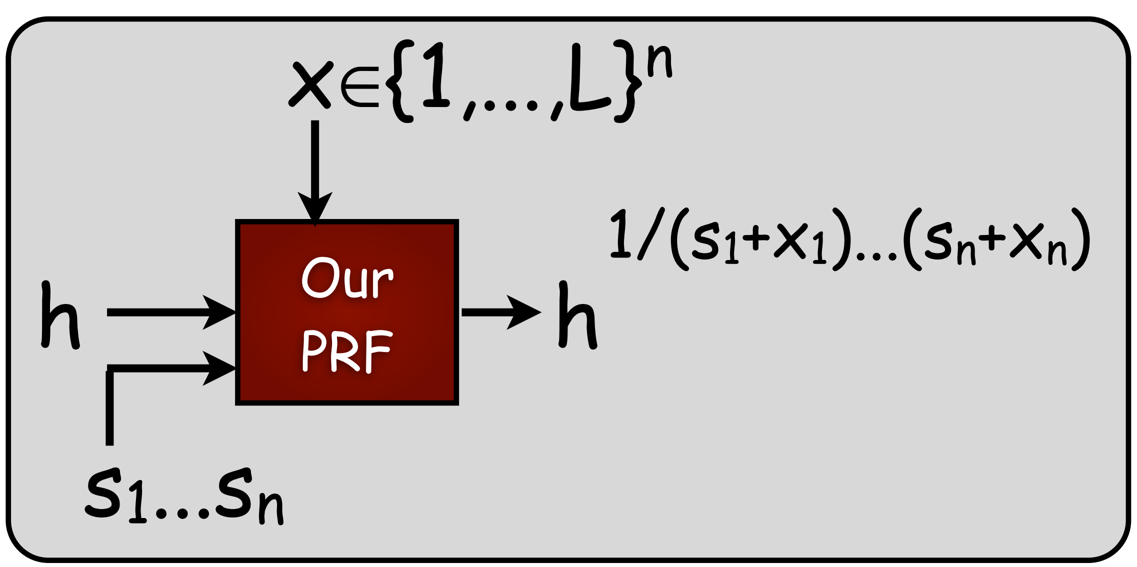
1) New generic tool to construct PRFs – **augmented cascade**



2) Simple proofs of security for existing algebraic PRFs

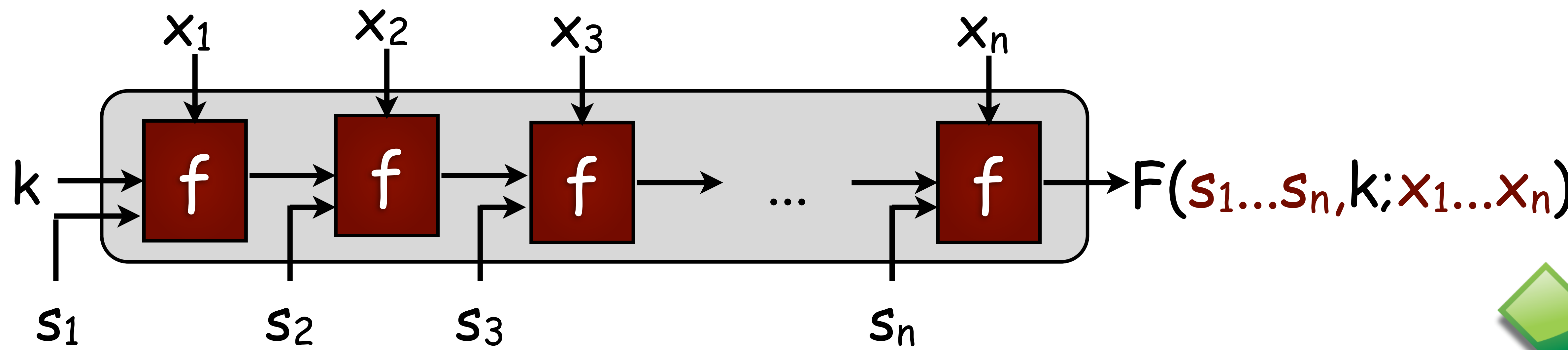


3) New efficient large-domain PRF

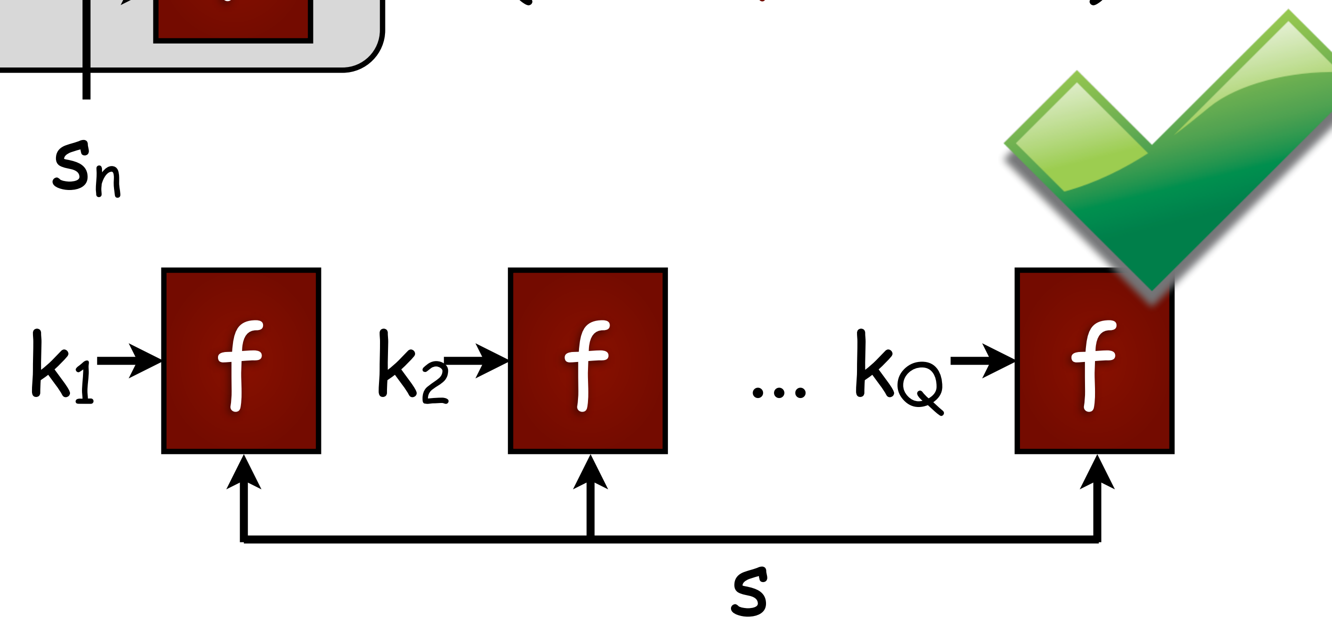


Conclusions

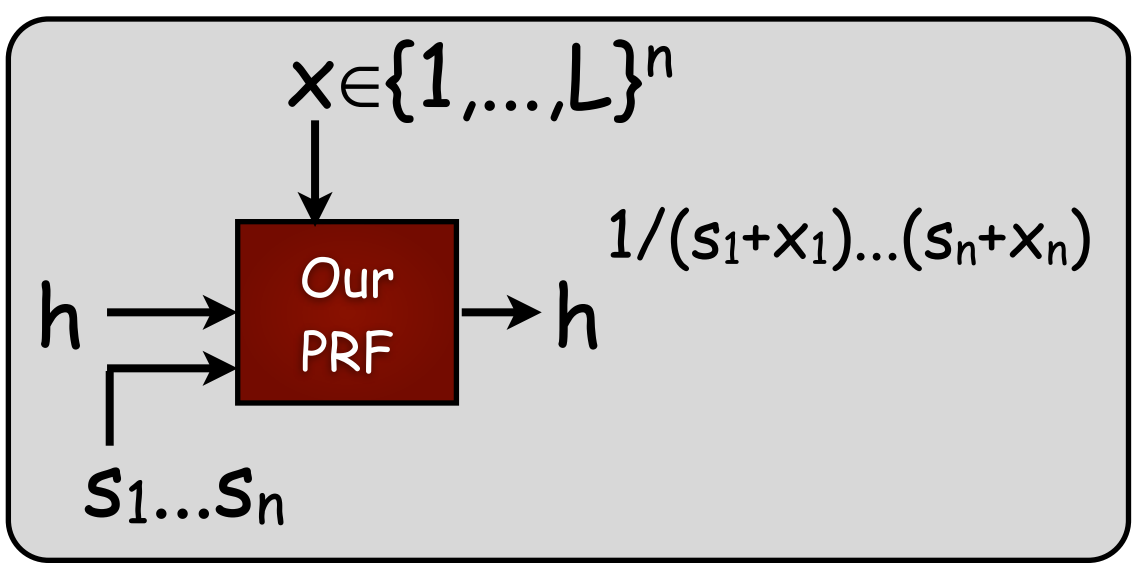
1) New generic tool to construct PRFs – **augmented cascade**



2) Simple proofs of security for existing algebraic PRFs



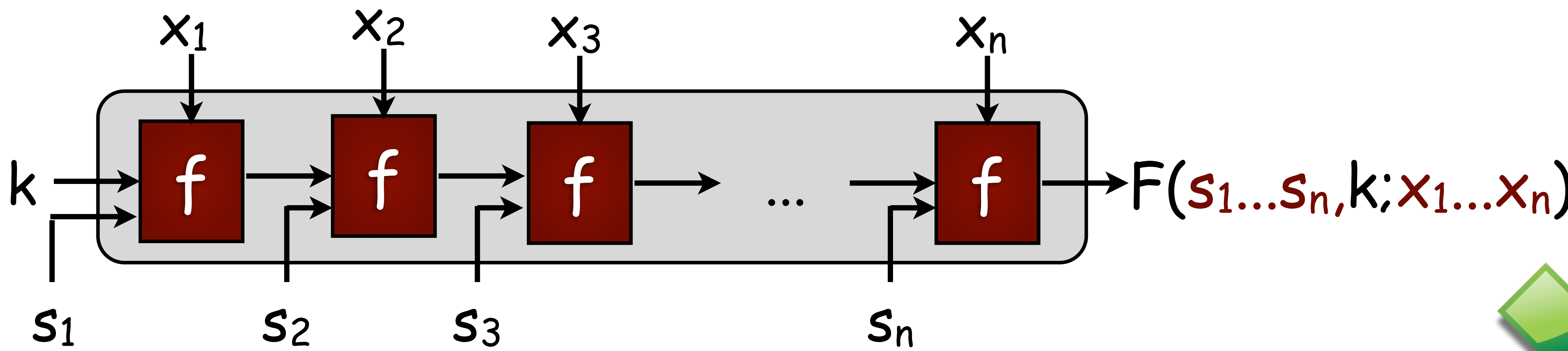
3) New efficient large-domain PRF



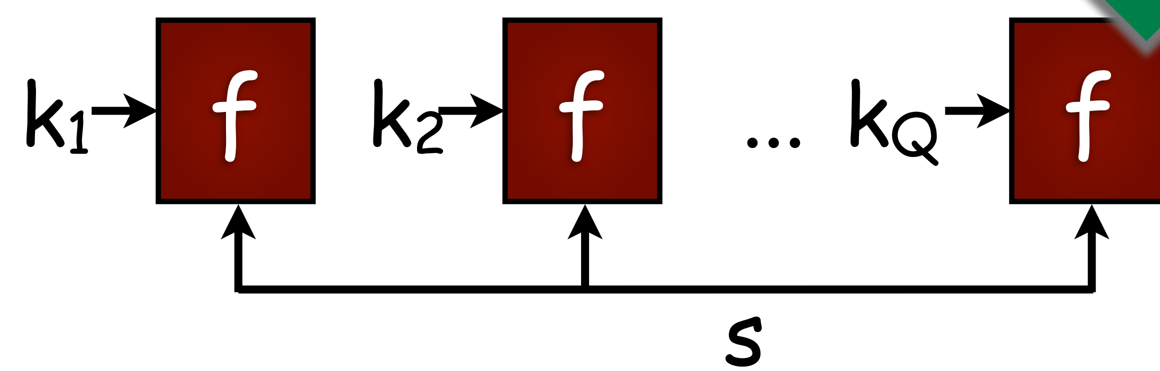
4) VRFs based on better security assumptions

Conclusions

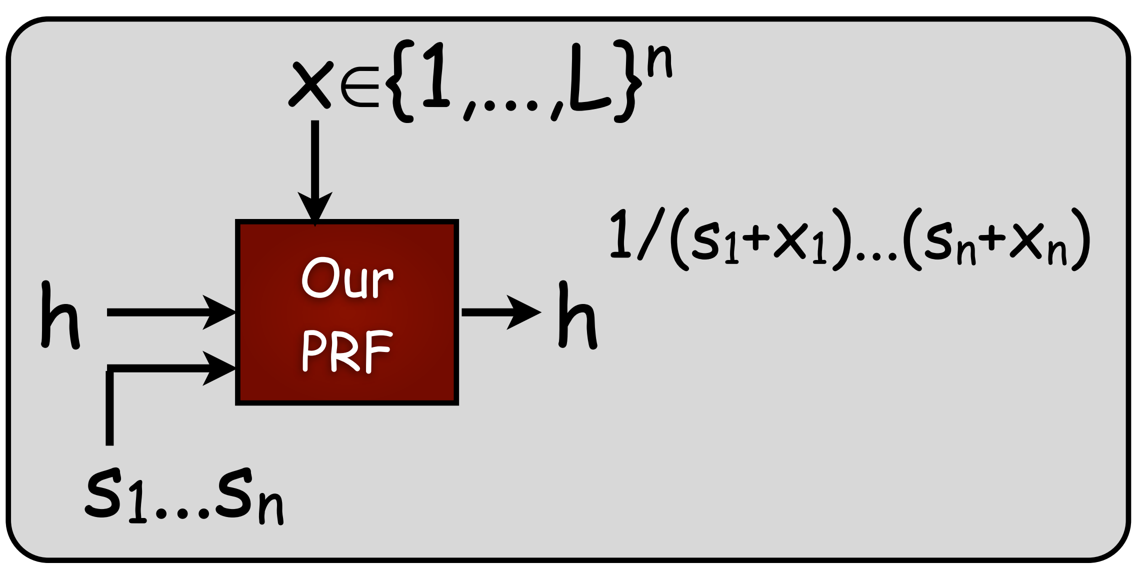
1) New generic tool to construct PRFs – **augmented cascade**



2) Simple proofs of security for existing algebraic PRFs



3) New efficient large-domain PRF



4) VRFs based on better security assumptions

5) (in paper) Simulatable VRFs

Thank You!

Any questions?