Better proofs for rekeying

D. J. Bernstein

Security of AES-256 key k is far below 2^{256} in most protocols: $(\mathrm{AES}_k(0), \ldots, \mathrm{AES}_k(n-1))$ is distinguishable from uniform with probability $n(n-1)/2^{129}$, plus tiny key-guessing probability. Yes, distinguishers matter.

Attacker actually has T targets: independent keys $k_1, \ldots, k_T$. Success chance $\approx Tn(n-1)/2^{129}$.

"Rekeying" seems less dangerous.

Expand k into $F(k) = (\mathrm{AES}_k(0), \ldots, \mathrm{AES}_k(999999))$.

Split $F(k)$ into 500000 "subkeys".

Output $F(k')$ for each subkey k': i.e., $F(\mathrm{AES}_k(0), \mathrm{AES}_k(1))$; $F(\mathrm{AES}_k(2), \mathrm{AES}_k(3))$; ... $F(\mathrm{AES}_k(999998), \mathrm{AES}_k(999999))$.

Better proofs for rekeying

D. J. Bernstein

Security of AES-256 key k is far below 2^{256} in most protocols: $(\mathrm{AES}_k(0), \ldots, \mathrm{AES}_k(n-1))$ is distinguishable from uniform with probability $n(n-1)/2^{129}$, plus tiny key-guessing probability.
Yes, distinguishers matter.

Attacker actually has T targets: independent keys $k_1, \ldots, k_T$. Success chance $\approx Tn(n-1)/2^{129}$.

“Rekeying” seems less dangerous.

Expand k into $F(k) = (\mathrm{AES}_k(0), \ldots, \mathrm{AES}_k(999999))$.

Split $F(k)$ into 500000 “subkeys”.

Output $F(k')$ for each subkey k': i.e., $F(\mathrm{AES}_k(0), \mathrm{AES}_k(1))$; $F(\mathrm{AES}_k(2), \mathrm{AES}_k(3))$; ... $F(\mathrm{AES}_k(999998), \mathrm{AES}_k(999999))$.

Repeat for $k_1, \ldots, k_T$. **What is attacker’s success chance p_T?**

Better proofs for rekeying

D. J. Bernstein

Security of AES-256 key k is far below 2^{256} in most protocols: $(\text{AES}_k(0), \ldots, \text{AES}_k(n-1))$ is distinguishable from uniform with probability $n(n-1)/2^{129}$, plus tiny key-guessing probability. Yes, distinguishers matter.

Attacker actually has T targets: independent keys $k_1, \ldots, k_T$. Success chance $\approx Tn(n-1)/2^{129}$.

"Rekeying" seems less dangerous.

Expand k into $F(k) = (\text{AES}_k(0), \ldots, \text{AES}_k(999999))$.

Split $F(k)$ into 500000 "subkeys".

Output $F(k')$ for each subkey k': i.e., $F(\text{AES}_k(0), \text{AES}_k(1))$; $F(\text{AES}_k(2), \text{AES}_k(3))$; ... $F(\text{AES}_k(999998), \text{AES}_k(999999))$.

Repeat for $k_1, \ldots, k_T$. **What is attacker's success chance p_T?**

Intuitively clear that $p_T \le Tp_1$. So let's analyze p_1.

roofs for rekeying

ernstein

of AES-256 key k is
w 2^{256} in most protocols:
0)), ..., $\text{AES}_k(n-1)$)
guishable from uniform
bability $n(n-1)/2^{129}$,
y key-guessing probability.
tinguishers matter.

r actually has T targets:
dent keys $k_1, \ldots, k_T$.
chance $\approx Tn(n-1)/2^{129}$.

"Rekeying" seems less dangerous.

Expand k into $F(k) = (\text{AES}_k(0), \ldots, \text{AES}_k(999999))$.

Split $F(k)$ into 500000 "subkeys".

Output $F(k')$ for each subkey k': i.e., $F(\text{AES}_k(0), \text{AES}_k(1))$; $F(\text{AES}_k(2), \text{AES}_k(3))$; ... $F(\text{AES}_k(999998), \text{AES}_k(999999))$.

Repeat for $k_1, \ldots, k_T$. **What is attacker's success chance p_T?**

Intuitively clear that $p_T \leq Tp_1$.
So let's analyze p_1.

Attack s
master k
from a u

Years of
to distin
uniform
Distinct

ekeying

56 key k is
most protocols:
$S_k(n-1))$
from uniform
$(n-1)/2^{129}$,
sing probability.
matter.

has T targets:
$k_1, \ldots, k_T$.
$Tn(n-1)/2^{129}$.

"Rekeying" seems less dangerous.

Expand k into $F(k) =$
$(\text{AES}_k(0), \ldots, \text{AES}_k(999999))$.

Split $F(k)$ into 500000 "subkeys".

Output $F(k')$ for each subkey k':
i.e., $F(\text{AES}_k(0), \text{AES}_k(1))$;
$F(\text{AES}_k(2), \text{AES}_k(3))$; ...
$F(\text{AES}_k(999998), \text{AES}_k(999999))$.

Repeat for $k_1, \ldots, k_T$. **What is attacker's success chance p_T?**

Intuitively clear that $p_T \leq T p_1$.
So let's analyze p_1.

Attack strategy 1:
master key k. Dis
from a uniform ra

Years of cryptanal
to distinguish AES
uniform string of
Distinctness loses

s
ocols:
)
rm
29,
bility.

gets:
.
$)/2^{129}$.

"Rekeying" seems less dangerous.

Expand k into $F(k) =$
$(\text{AES}_k(0), \ldots, \text{AES}_k(999999))$.

Split $F(k)$ into 500000 "subkeys".

Output $F(k')$ for each subkey k':
i.e., $F(\text{AES}_k(0), \text{AES}_k(1))$;
$F(\text{AES}_k(2), \text{AES}_k(3))$; ...
$F(\text{AES}_k(999998), \text{AES}_k(999999))$.

Repeat for $k_1, \ldots, k_T$. **What is attacker's success chance p_T?**

Intuitively clear that $p_T \le T p_1$.
So let's analyze p_1.

Attack strategy 1: Attack th
master key k. Distinguish F
from a uniform random strin

Years of cryptanalysis say: h
to distinguish AES outputs f
uniform string of distinct blo
Distinctness loses $\approx 1/2^{89}$.

"Rekeying" seems less dangerous.

Expand k into $F(k) = (\text{AES}_k(0), \ldots, \text{AES}_k(999999))$.

Split $F(k)$ into 500000 "subkeys".

Output $F(k')$ for each subkey k': i.e., $F(\text{AES}_k(0), \text{AES}_k(1))$; $F(\text{AES}_k(2), \text{AES}_k(3))$; ... $F(\text{AES}_k(999998), \text{AES}_k(999999))$.

Repeat for $k_1, \ldots, k_T$. **What is attacker's success chance p_T?**

Intuitively clear that $p_T \le T p_1$. So let's analyze p_1.

Attack strategy 1: Attack the master key k. Distinguish $F(k)$ from a uniform random string.

Years of cryptanalysis say: hard to distinguish AES outputs from uniform string of distinct blocks. Distinctness loses $\approx 1/2^{89}$.

"Rekeying" seems less dangerous.

Expand k into $F(k) = (\text{AES}_k(0), \ldots, \text{AES}_k(999999))$.

Split $F(k)$ into 500000 "subkeys".

Output $F(k')$ for each subkey k': i.e., $F(\text{AES}_k(0), \text{AES}_k(1))$; $F(\text{AES}_k(2), \text{AES}_k(3))$; ... $F(\text{AES}_k(999998), \text{AES}_k(999999))$.

Repeat for $k_1, \ldots, k_T$. **What is attacker's success chance p_T?**

Intuitively clear that $p_T \le T p_1$. So let's analyze p_1.

Attack strategy 1: Attack the master key k. Distinguish $F(k)$ from a uniform random string.

Years of cryptanalysis say: hard to distinguish AES outputs from uniform string of distinct blocks. Distinctness loses $\approx 1/2^{89}$.

Attack strategy 2: Attack a subkey k'. Distinguish $F(k')$ from uniform, assuming k' is uniform.

"Rekeying" seems less dangerous.

Expand k into $F(k) = (\text{AES}_k(0), \ldots, \text{AES}_k(999999))$.

Split $F(k)$ into 500000 "subkeys".

Output $F(k')$ for each subkey k': i.e., $F(\text{AES}_k(0), \text{AES}_k(1))$; $F(\text{AES}_k(2), \text{AES}_k(3))$; ... $F(\text{AES}_k(999998), \text{AES}_k(999999))$.

Repeat for $k_1, \ldots, k_T$. **What is attacker's success chance p_T?**

Intuitively clear that $p_T \le T p_1$. So let's analyze p_1.

Attack strategy 1: Attack the master key k. Distinguish $F(k)$ from a uniform random string.

Years of cryptanalysis say: hard to distinguish AES outputs from uniform string of distinct blocks. Distinctness loses $\approx 1/2^{89}$.

Attack strategy 2: Attack a subkey k'. Distinguish $F(k')$ from uniform, assuming k' is uniform.

Intuition: No other attacks exist. **But where is this proven?**

ng" seems less dangerous.

k into $F(k) =$
$0), \ldots, \text{AES}_k(999999))$.

$k)$ into 500000 "subkeys".

$F(k')$ for each subkey k':
$\text{AES}_k(0), \text{AES}_k(1))$;
$(2), \text{AES}_k(3))$; ...
$(999998), \text{AES}_k(999999))$.

for $k_1, \ldots, k_T$. **What is**
r's success chance p_T?

ly clear that $p_T \le T p_1$.
analyze p_1.

Attack strategy 1: Attack the master key k. Distinguish $F(k)$ from a uniform random string.

Years of cryptanalysis say: hard to distinguish AES outputs from uniform string of distinct blocks. Distinctness loses $\approx 1/2^{89}$.

Attack strategy 2: Attack a subkey k'. Distinguish $F(k')$ from uniform, assuming k' is uniform.

Intuition: No other attacks exist.
But where is this proven?

FOCS 19
Krawczy
security

2-level c
(N_1, N_2)

less dangerous.

$k) =$

$S_k(999999))$.

0000 “subkeys”.

each subkey k':

$ES_k(1))$;

$(3))$; . . .

$AES_k(999999))$.

, k_T. **What is**

s chance p_T?

at $p_T \le T p_1$.

.

Attack strategy 1: Attack the master key k. Distinguish $F(k)$ from a uniform random string.

Years of cryptanalysis say: hard to distinguish AES outputs from uniform string of distinct blocks. Distinctness loses $\approx 1/2^{89}$.

Attack strategy 2: Attack a subkey k'. Distinguish $F(k')$ from uniform, assuming k' is uniform.

Intuition: No other attacks exist. **But where is this proven?**

FOCS 1996 Bellar

Krawczyk claims t

security of ℓ-level

2-level cascade: ke

(N_1, N_2); output S

erous.

9)).

keys".

ey k':

999)).

t is

p_T**?**

p_1.

Attack strategy 1: Attack the master key k. Distinguish $F(k)$ from a uniform random string.

Years of cryptanalysis say: hard to distinguish AES outputs from uniform string of distinct blocks. Distinctness loses $\approx 1/2^{89}$.

Attack strategy 2: Attack a subkey k'. Distinguish $F(k')$ from uniform, assuming k' is uniform.

Intuition: No other attacks exist.
But where is this proven?

FOCS 1996 Bellare–Canetti–
Krawczyk claims to prove
security of ℓ-level "cascade"

2-level cascade: key k; inpu
(N_1, N_2); output $S(S(k, N_1)$

Attack strategy 1: Attack the master key k. Distinguish $F(k)$ from a uniform random string.

Years of cryptanalysis say: hard to distinguish AES outputs from uniform string of distinct blocks. Distinctness loses $\approx 1/2^{89}$.

Attack strategy 2: Attack a subkey k'. Distinguish $F(k')$ from uniform, assuming k' is uniform.

Intuition: No other attacks exist.
But where is this proven?

FOCS 1996 Bellare–Canetti–Krawczyk claims to prove security of ℓ-level "cascade".

2-level cascade: key k; input (N_1, N_2); output $S(S(k, N_1), N_2)$.

Attack strategy 1: Attack the master key k. Distinguish $F(k)$ from a uniform random string.

Years of cryptanalysis say: hard to distinguish AES outputs from uniform string of distinct blocks. Distinctness loses $\approx 1/2^{89}$.

Attack strategy 2: Attack a subkey k'. Distinguish $F(k')$ from uniform, assuming k' is uniform.

Intuition: No other attacks exist.
But where is this proven?

FOCS 1996 Bellare–Canetti–Krawczyk claims to prove security of ℓ-level "cascade".

2-level cascade: key k; input (N_1, N_2); output $S(S(k, N_1), N_2)$.

Example: Define $S(k, N) = (\text{AES}_k(2N), \text{AES}_k(2N+1))$, with $N \in \{0, 1, \ldots, 499999\}$.
S expands AES-256 key k into $(\text{AES}_k(0), \ldots, \text{AES}_k(999999))$.

Attack strategy 1: Attack the master key k. Distinguish $F(k)$ from a uniform random string.

Years of cryptanalysis say: hard to distinguish AES outputs from uniform string of distinct blocks. Distinctness loses $\approx 1/2^{89}$.

Attack strategy 2: Attack a subkey k'. Distinguish $F(k')$ from uniform, assuming k' is uniform.

Intuition: No other attacks exist.
But where is this proven?

FOCS 1996 Bellare–Canetti–Krawczyk claims to prove security of ℓ-level "cascade".

2-level cascade: key k; input (N_1, N_2); output $S(S(k, N_1), N_2)$.

Example: Define $S(k, N) = (\text{AES}_k(2N), \text{AES}_k(2N+1))$, with $N \in \{0, 1, \ldots, 499999\}$.
S expands AES-256 key k into $(\text{AES}_k(0), \ldots, \text{AES}_k(999999))$.

Paper credits 1986 Goldwasser–Goldreich–Micali for 1-bit N_i:
S expands k into $S(k, 0), S(k, 1)$.

strategy 1: Attack the
key k. Distinguish $F(k)$
uniform random string.

cryptanalysis say: hard
guish AES outputs from
string of distinct blocks.
ness loses $\approx 1/2^{89}$.

strategy 2: Attack a
k'. Distinguish $F(k')$ from
assuming k' is uniform.

: No other attacks exist.
ere is this proven?

FOCS 1996 Bellare–Canetti–Krawczyk claims to prove security of ℓ-level "cascade".

2-level cascade: key k; input (N_1, N_2); output $S(S(k, N_1), N_2)$.

Example: Define $S(k, N) = (\text{AES}_k(2N), \text{AES}_k(2N+1))$, with $N \in \{0, 1, \ldots, 499999\}$. S expands AES-256 key k into $(\text{AES}_k(0), \ldots, \text{AES}_k(999999))$.

Paper credits 1986 Goldwasser–Goldreich–Micali for 1-bit N_i: S expands k into $S(k, 0), S(k, 1)$.

Theorem
omits fa

Here q i
The intu
why doe

Attack the
tinguish $F(k)$
ndom string.

ysis say: hard
S outputs from
distinct blocks.
$\approx 1/2^{89}$.

Attack a
guish $F(k')$ from
k' is uniform.

er attacks exist.
s proven?

FOCS 1996 Bellare–Canetti–Krawczyk claims to prove security of ℓ-level "cascade".

2-level cascade: key k; input (N_1, N_2); output $S(S(k, N_1), N_2)$.

Example: Define $S(k, N) =$ $(\text{AES}_k(2N), \text{AES}_k(2N+1))$, with $N \in \{0, 1, \ldots, 499999\}$.
S expands AES-256 key k into $(\text{AES}_k(0), \ldots, \text{AES}_k(999999))$.

Paper credits 1986 Goldwasser–Goldreich–Micali for 1-bit N_i:
S expands k into $S(k, 0), S(k, 1)$.

Theorem statemen
omits factor q. Fi

Here q is the num
The intuition didn
why does q matte

ne

(k)

ng.

ard

from

ocks.

) from

orm.

exist.

FOCS 1996 Bellare–Canetti–Krawczyk claims to prove security of ℓ-level "cascade".

2-level cascade: key k; input (N_1, N_2); output $S(S(k, N_1), N_2)$.

Example: Define $S(k, N) = (\text{AES}_k(2N), \text{AES}_k(2N + 1))$, with $N \in \{0, 1, \ldots, 499999\}$. S expands AES-256 key k into $(\text{AES}_k(0), \ldots, \text{AES}_k(999999))$.

Paper credits 1986 Goldwasser–Goldreich–Micali for 1-bit N_i: S expands k into $S(k, 0), S(k, 1)$.

Theorem statement is wrong

omits factor q. Fixed in 200

Here q is the number of que

The intuition didn't notice q

why does q matter for the p

FOCS 1996 Bellare–Canetti–Krawczyk claims to prove security of ℓ-level "cascade".

2-level cascade: key k; input (N_1, N_2); output $S(S(k, N_1), N_2)$.

Example: Define $S(k, N) = (\text{AES}_k(2N), \text{AES}_k(2N+1))$, with $N \in \{0, 1, \ldots, 499999\}$. S expands AES-256 key k into $(\text{AES}_k(0), \ldots, \text{AES}_k(999999))$.

Paper credits 1986 Goldwasser–Goldreich–Micali for 1-bit N_i: S expands k into $S(k, 0), S(k, 1)$.

Theorem statement is wrong: omits factor q. Fixed in 2005.

Here q is the number of queries. The intuition didn't notice q; why does q matter for the proof?

FOCS 1996 Bellare–Canetti–Krawczyk claims to prove security of ℓ-level "cascade".

2-level cascade: key k; input (N_1, N_2); output $S(S(k, N_1), N_2)$.

Example: Define $S(k, N) = (\text{AES}_k(2N), \text{AES}_k(2N+1))$, with $N \in \{0, 1, \ldots, 499999\}$. S expands AES-256 key k into $(\text{AES}_k(0), \ldots, \text{AES}_k(999999))$.

Paper credits 1986 Goldwasser–Goldreich–Micali for 1-bit N_i: S expands k into $S(k, 0), S(k, 1)$.

Theorem statement is wrong: omits factor q. Fixed in 2005.

Here q is the number of queries. The intuition didn't notice q; why does q matter for the proof?

Proof outline: Take any cascade attack A using at most q queries.

Proof has $q + 1$ steps.

FOCS 1996 Bellare–Canetti–Krawczyk claims to prove security of ℓ-level "cascade".

2-level cascade: key k; input (N_1, N_2); output $S(S(k, N_1), N_2)$.

Example: Define $S(k, N) = (\text{AES}_k(2N), \text{AES}_k(2N + 1))$, with $N \in \{0, 1, \ldots, 499999\}$.
S expands AES-256 key k into $(\text{AES}_k(0), \ldots, \text{AES}_k(999999))$.

Paper credits 1986 Goldwasser–Goldreich–Micali for 1-bit N_i:
S expands k into $S(k, 0), S(k, 1)$.

Theorem statement is wrong: omits factor q. Fixed in 2005.

Here q is the number of queries.
The intuition didn't notice q; why does q matter for the proof?

Proof outline: Take any cascade attack A using at most q queries.

Proof has $q + 1$ steps.

Step 0: Replace outputs from master key k with independent uniform random outputs.

Distinguisher for this step
$\Rightarrow$ attack against S.

996 Bellare–Canetti–
k claims to prove
of ℓ-level "cascade".

ascade: key k; input
); output $S(S(k, N_1), N_2)$.

e: Define $S(k, N) =$
$2N), \text{AES}_k(2N + 1))$,
$\in \{0, 1, \ldots, 499999\}$.
ds AES-256 key k into
$0), \ldots, \text{AES}_k(999999))$.

redits 1986 Goldwasser–
h–Micali for 1-bit N_i:
ds k into $S(k, 0), S(k, 1)$.

Theorem statement is wrong:
omits factor q. Fixed in 2005.

Here q is the number of queries.
The intuition didn't notice q;
why does q matter for the proof?

Proof outline: Take any cascade
attack A using at most q queries.

Proof has $q + 1$ steps.

Step 0: Replace outputs from
master key k with independent
uniform random outputs.

Distinguisher for this step
$\Rightarrow$ attack against S.

Step 1:
for *first*
uniform

Distingu
$\Rightarrow$ attac

e–Canetti–
o prove
“cascade”.

ey k; input
$S(S(k, N_1), N_2)$.

$S(k, N) =$
$(2N + 1))$,
$, 499999\}$.
56 key k into
$S_k(999999))$.

6 Goldwasser–
or 1-bit N_i:
$S(k, 0), S(k, 1)$.

Theorem statement is wrong:
omits factor q. Fixed in 2005.

Here q is the number of queries.
The intuition didn’t notice q;
why does q matter for the proof?

Proof outline: Take any cascade
attack A using at most q queries.

Proof has $q + 1$ steps.

Step 0: Replace outputs from
master key k with independent
uniform random outputs.

Distinguisher for this step
$\Rightarrow$ attack against S.

Step 1: Replace ca
for *first* subkey wi
uniform random o

Distinguisher for t
$\Rightarrow$ attack against

–

.

t

$), N_2)$.

,

.

nto

$))$.

ser–

$_i$:

$(k, 1)$.

Theorem statement is wrong: omits factor q. Fixed in 2005.

Here q is the number of queries. The intuition didn't notice q; why does q matter for the proof?

Proof outline: Take any cascade attack A using at most q queries.

Proof has $q + 1$ steps.

Step 0: Replace outputs from master key k with independent uniform random outputs.

Distinguisher for this step $\Rightarrow$ attack against S.

Step 1: Replace cascade out for *first* subkey with indepen uniform random outputs.

Distinguisher for this step $\Rightarrow$ attack against S.

Theorem statement is wrong: omits factor q. Fixed in 2005.

Here q is the number of queries. The intuition didn't notice q; why does q matter for the proof?

Proof outline: Take any cascade attack A using at most q queries.

Proof has $q + 1$ steps.

Step 0: Replace outputs from master key k with independent uniform random outputs.

Distinguisher for this step
$\Rightarrow$ attack against S.

Step 1: Replace cascade outputs for *first* subkey with independent uniform random outputs.

Distinguisher for this step
$\Rightarrow$ attack against S.

Theorem statement is wrong: omits factor q. Fixed in 2005.

Here q is the number of queries. The intuition didn't notice q; why does q matter for the proof?

Proof outline: Take any cascade attack A using at most q queries.

Proof has $q+1$ steps.

Step 0: Replace outputs from master key k with independent uniform random outputs.

Distinguisher for this step $\Rightarrow$ attack against S.

Step 1: Replace cascade outputs for *first* subkey with independent uniform random outputs.

Distinguisher for this step $\Rightarrow$ attack against S.

Step 2: Replace cascade outputs from next (distinct) subkey. . . .
Step q: Replace cascade outputs from qth (distinct) subkey.
Could skip steps if $q > \#\{N\}$.

Theorem statement is wrong: omits factor q. Fixed in 2005.

Here q is the number of queries. The intuition didn't notice q; why does q matter for the proof?

Proof outline: Take any cascade attack A using at most q queries.

Proof has $q + 1$ steps.

Step 0: Replace outputs from master key k with independent uniform random outputs.

Distinguisher for this step $\Rightarrow$ attack against S.

Step 1: Replace cascade outputs for *first* subkey with independent uniform random outputs.

Distinguisher for this step $\Rightarrow$ attack against S.

Step 2: Replace cascade outputs from next (distinct) subkey. . . .
Step q: Replace cascade outputs from qth (distinct) subkey.
Could skip steps if $q > \#\{N\}$.

Further complications in proof to monolithically handle ℓ levels.
2011 Bernstein: simpler to compose better 2-level theorem.

n statement is wrong:
ctor q. Fixed in 2005.

s the number of queries.
uition didn't notice q;
es q matter for the proof?

utline: Take any cascade
A using at most q queries.

as $q + 1$ steps.

Replace outputs from
key k with independent
random outputs.

isher for this step
k against S.

Step 1: Replace cascade outputs
for *first* subkey with independent
uniform random outputs.

Distinguisher for this step
$\Rightarrow$ attack against S.

Step 2: Replace cascade outputs
from next (distinct) subkey. . . .
Step q: Replace cascade outputs
from qth (distinct) subkey.
Could skip steps if $q > \#\{N\}$.

Further complications in proof
to monolithically handle ℓ levels.
2011 Bernstein: simpler to
compose better 2-level theorem.

Not hap

A differe
Crypto 1
Krawczy

nt is wrong:
xed in 2005.

ber of queries.
't notice q;
r for the proof?

ke any cascade
most q queries.

teps.

utputs from
independent
utputs.

his step
S.

Step 1: Replace cascade outputs for *first* subkey with independent uniform random outputs.

Distinguisher for this step $\Rightarrow$ attack against S.

Step 2: Replace cascade outputs from next (distinct) subkey. . . .
Step q: Replace cascade outputs from qth (distinct) subkey.
Could skip steps if $q > \#\{N\}$.

Further complications in proof to monolithically handle ℓ levels.
2011 Bernstein: simpler to compose better 2-level theorem.

Not happy with ca

A different proof a
Crypto 1996 Bella
Krawczyk NMAC/

g:
05.

eries.
q;
proof?

cade
ueries.

m
ent

Step 1: Replace cascade outputs for *first* subkey with independent uniform random outputs.

Distinguisher for this step $\Rightarrow$ attack against S.

Step 2: Replace cascade outputs from next (distinct) subkey. . . .
Step q: Replace cascade outputs from qth (distinct) subkey.
Could skip steps if $q > \#\{N\}$.

Further complications in proof to monolithically handle ℓ levels.
2011 Bernstein: simpler to compose better 2-level theorem.

Not happy with cascade pro

A different proof appears in
Crypto 1996 Bellare–Canetti
Krawczyk NMAC/HMAC pa

Step 1: Replace cascade outputs for *first* subkey with independent uniform random outputs.

Distinguisher for this step $\Rightarrow$ attack against S.

Step 2: Replace cascade outputs from next (distinct) subkey. . . .
Step q: Replace cascade outputs from qth (distinct) subkey.
Could skip steps if $q > \#\{N\}$.

Further complications in proof to monolithically handle ℓ levels.
2011 Bernstein: simpler to compose better 2-level theorem.

Not happy with cascade proofs?

A different proof appears in Crypto 1996 Bellare–Canetti–Krawczyk NMAC/HMAC paper.

Step 1: Replace cascade outputs for *first* subkey with independent uniform random outputs.

Distinguisher for this step $\Rightarrow$ attack against S.

Step 2: Replace cascade outputs from next (distinct) subkey. . . .
Step q: Replace cascade outputs from qth (distinct) subkey.
Could skip steps if $q > \#\{N\}$.

Further complications in proof to monolithically handle ℓ levels.
2011 Bernstein: simpler to compose better 2-level theorem.

Not happy with cascade proofs?

A different proof appears in Crypto 1996 Bellare–Canetti–Krawczyk NMAC/HMAC paper.

Given key k and input (N_1, N_2), NMAC computes $S(S(k, N_1), N_2)$, where S is a ~~stream cipher~~ "compression function".

(Tweaks: output is encrypted; no prefix-free requirement.)

Step 1: Replace cascade outputs for *first* subkey with independent uniform random outputs.

Distinguisher for this step $\Rightarrow$ attack against S.

Step 2: Replace cascade outputs from next (distinct) subkey. . . .
Step q: Replace cascade outputs from qth (distinct) subkey.
Could skip steps if $q > \#\{N\}$.

Further complications in proof to monolithically handle ℓ levels.
2011 Bernstein: simpler to compose better 2-level theorem.

Not happy with cascade proofs?

A different proof appears in Crypto 1996 Bellare–Canetti–Krawczyk NMAC/HMAC paper.

Given key k and input (N_1, N_2), NMAC computes $S(S(k, N_1), N_2)$, where S is a ~~stream cipher~~ "compression function".

(Tweaks: output is encrypted; no prefix-free requirement.)

Proof has weird assumptions.
Crypto 2006 Bellare proof: more reasonable-sounding assumptions.

Replace cascade outputs
subkey with independent
random outputs.

isher for this step
k against S.

Replace cascade outputs
xt (distinct) subkey. . . .
Replace cascade outputs
n (distinct) subkey.
kip steps if $q > \#\{N\}$.

complications in proof
lithically handle ℓ levels.
rnstein: simpler to
e better 2-level theorem.

Not happy with cascade proofs?

A different proof appears in Crypto 1996 Bellare–Canetti–Krawczyk NMAC/HMAC paper.

Given key k and input (N_1, N_2), NMAC computes $S(S(k, N_1), N_2)$, where S is a ~~stream cipher~~ “compression function”.

(Tweaks: output is encrypted; no prefix-free requirement.)

Proof has weird assumptions. Crypto 2006 Bellare proof: more reasonable-sounding assumptions.

Complic

2012 Ko
Bellare’s

ascade outputs
th independent
utputs.

his step
S.

ascade outputs
t) subkey. . . .
ascade outputs
) subkey.
f $q > \#\{N\}$.

ons in proof
handle ℓ levels.
mpler to
level theorem.

Not happy with cascade proofs?

A different proof appears in Crypto 1996 Bellare–Canetti–Krawczyk NMAC/HMAC paper.

Given key k and input (N_1, N_2), NMAC computes $S(S(k, N_1), N_2)$, where S is a ~~stream cipher~~ "compression function".

(Tweaks: output is encrypted; no prefix-free requirement.)

Proof has weird assumptions. Crypto 2006 Bellare proof: more reasonable-sounding assumptions.

Complicated; error

2012 Koblitz–Men
Bellare's assumpti

tputs
ndent

tputs
. . .
tputs

/}.

oof
vels.

rem.

Not happy with cascade proofs?

A different proof appears in Crypto 1996 Bellare–Canetti–Krawczyk NMAC/HMAC paper.

Given key k and input (N_1, N_2), NMAC computes $S(S(k, N_1), N_2)$, where S is a ~~stream cipher~~ "compression function".

(Tweaks: output is encrypted; no prefix-free requirement.)

Proof has weird assumptions. Crypto 2006 Bellare proof: more reasonable-sounding assumptions.

Complicated; error-prone.

2012 Koblitz–Menezes: Bellare's assumptions are wr

Not happy with cascade proofs?

A different proof appears in Crypto 1996 Bellare–Canetti–Krawczyk NMAC/HMAC paper.

Given key k and input (N_1, N_2), NMAC computes $S(S(k, N_1), N_2)$, where S is a ~~stream cipher~~ "compression function".

(Tweaks: output is encrypted; no prefix-free requirement.)

Proof has weird assumptions. Crypto 2006 Bellare proof: more reasonable-sounding assumptions.

Complicated; error-prone.

2012 Koblitz–Menezes: Bellare's assumptions are wrong.

Not happy with cascade proofs?

A different proof appears in Crypto 1996 Bellare–Canetti–Krawczyk NMAC/HMAC paper.

Given key k and input (N_1, N_2), NMAC computes $S(S(k, N_1), N_2)$, where S is a ~~stream cipher~~ "compression function".

(Tweaks: output is encrypted; no prefix-free requirement.)

Proof has weird assumptions. Crypto 2006 Bellare proof: more reasonable-sounding assumptions.

Complicated; error-prone.

2012 Koblitz–Menezes: Bellare's assumptions are wrong.

2012 Katz–Lindell: public denials.

Not happy with cascade proofs?

A different proof appears in Crypto 1996 Bellare–Canetti–Krawczyk NMAC/HMAC paper.

Given key k and input (N_1, N_2), NMAC computes $S(S(k, N_1), N_2)$, where S is a ~~stream cipher~~ "compression function".

(Tweaks: output is encrypted; no prefix-free requirement.)

Proof has weird assumptions. Crypto 2006 Bellare proof: more reasonable-sounding assumptions.

Complicated; error-prone.

2012 Koblitz–Menezes: Bellare's assumptions are wrong.

2012 Katz–Lindell: public denials.

2012 Bernstein–Lange: Bellare's assumptions are wrong.

Not happy with cascade proofs?

A different proof appears in Crypto 1996 Bellare–Canetti–Krawczyk NMAC/HMAC paper.

Given key k and input (N_1, N_2), NMAC computes $S(S(k, N_1), N_2)$, where S is a ~~stream cipher~~ "compression function".

(Tweaks: output is encrypted; no prefix-free requirement.)

Proof has weird assumptions. Crypto 2006 Bellare proof: more reasonable-sounding assumptions.

Complicated; error-prone.

2012 Koblitz–Menezes: Bellare's assumptions are wrong.

2012 Katz–Lindell: public denials.

2012 Bernstein–Lange: Bellare's assumptions are wrong.

2013 Pietrzak: fixed theorem from Koblitz–Menezes is wrong.

Not happy with cascade proofs?

A different proof appears in Crypto 1996 Bellare–Canetti–Krawczyk NMAC/HMAC paper.

Given key k and input (N_1, N_2), NMAC computes $S(S(k, N_1), N_2)$, where S is a ~~stream cipher~~ "compression function".

(Tweaks: output is encrypted; no prefix-free requirement.)

Proof has weird assumptions. Crypto 2006 Bellare proof: more reasonable-sounding assumptions.

Complicated; error-prone.

2012 Koblitz–Menezes: Bellare's assumptions are wrong.

2012 Katz–Lindell: public denials.

2012 Bernstein–Lange: Bellare's assumptions are wrong.

2013 Pietrzak: fixed theorem from Koblitz–Menezes is wrong.

2013 Pietrzak, 2013 Koblitz–Menezes, 2014 Gaži–Pietrzak–Rybár: another NMAC proof, as complicated as cascade proof.

py with cascade proofs?

ent proof appears in 1996 Bellare–Canetti– k NMAC/HMAC paper.

ey k and input (N_1, N_2), computes $S(S(k, N_1), N_2)$, is a ~~stream cipher~~ ession function".

s: output is encrypted; k-free requirement.)

as weird assumptions. 2006 Bellare proof: more ble-sounding assumptions.

Complicated; error-prone.

2012 Koblitz–Menezes: Bellare's assumptions are wrong.

2012 Katz–Lindell: public denials.

2012 Bernstein–Lange: Bellare's assumptions are wrong.

2013 Pietrzak: fixed theorem from Koblitz–Menezes is wrong.

2013 Pietrzak, 2013 Koblitz– Menezes, 2014 Gaži–Pietrzak– Rybár: another NMAC proof, as complicated as cascade proof.

Hmmm.

"A mod pseudo-r applicati RNG ou Another ascade proofs?

appears in
re–Canetti–
HMAC paper.

nput (N_1, N_2),
$S(S(k, N_1), N_2)$,
~~m cipher~~
tion".

s encrypted;
irement.)

ssumptions.
re proof: more
ng assumptions.

Complicated; error-prone.

2012 Koblitz–Menezes:
Bellare's assumptions are wrong.

2012 Katz–Lindell: public denials.

2012 Bernstein–Lange:
Bellare's assumptions are wrong.

2013 Pietrzak: fixed theorem
from Koblitz–Menezes is wrong.

2013 Pietrzak, 2013 Koblitz–Menezes, 2014 Gaži–Pietrzak–Rybár: another NMAC proof,
as complicated as cascade proof.

Hmmm. CCS 200
"A model and arc
pseudo-random ge
applications to /d
RNG outputs $F(k$
Another complicat

ofs?

i–

per.

N_2),

), N_2),

ed;

s.

more

tions.

Complicated; error-prone.

2012 Koblitz–Menezes:
Bellare's assumptions are wrong.

2012 Katz–Lindell: public denials.

2012 Bernstein–Lange:
Bellare's assumptions are wrong.

2013 Pietrzak: fixed theorem
from Koblitz–Menezes is wrong.

2013 Pietrzak, 2013 Koblitz–
Menezes, 2014 Gaži–Pietrzak–
Rybár: another NMAC proof,
as complicated as cascade proof.

Hmmm. CCS 2005 Barak–H
"A model and architecture f
pseudo-random generation w
applications to `/dev/rando`
RNG outputs $F(k)$, $F(G(k)$
Another complicated proof.

Complicated; error-prone.

2012 Koblitz–Menezes:
Bellare's assumptions are wrong.

2012 Katz–Lindell: public denials.

2012 Bernstein–Lange:
Bellare's assumptions are wrong.

2013 Pietrzak: fixed theorem
from Koblitz–Menezes is wrong.

2013 Pietrzak, 2013 Koblitz–Menezes, 2014 Gaži–Pietrzak–Rybár: another NMAC proof,
as complicated as cascade proof.

Hmmm. CCS 2005 Barak–Halevi
"A model and architecture for
pseudo-random generation with
applications to `/dev/random`"?
RNG outputs $F(k)$, $F(G(k))$, etc.
Another complicated proof.

Complicated; error-prone.

2012 Koblitz–Menezes:
Bellare's assumptions are wrong.

2012 Katz–Lindell: public denials.

2012 Bernstein–Lange:
Bellare's assumptions are wrong.

2013 Pietrzak: fixed theorem from Koblitz–Menezes is wrong.

2013 Pietrzak, 2013 Koblitz–Menezes, 2014 Gaži–Pietrzak–Rybár: another NMAC proof, as complicated as cascade proof.

Hmmm. CCS 2005 Barak–Halevi "A model and architecture for pseudo-random generation with applications to `/dev/random`"?
RNG outputs $F(k)$, $F(G(k))$, etc.
Another complicated proof.

How about 2006 Campagna "Security bounds for the NIST codebook-based deterministic random bit generator"? Doesn't prove anything about rekeying.

Complicated; error-prone.

2012 Koblitz–Menezes:
Bellare's assumptions are wrong.

2012 Katz–Lindell: public denials.

2012 Bernstein–Lange:
Bellare's assumptions are wrong.

2013 Pietrzak: fixed theorem from Koblitz–Menezes is wrong.

2013 Pietrzak, 2013 Koblitz–Menezes, 2014 Gaži–Pietrzak–Rybár: another NMAC proof, as complicated as cascade proof.

Hmmm. CCS 2005 Barak–Halevi "A model and architecture for pseudo-random generation with applications to `/dev/random`"?
RNG outputs $F(k)$, $F(G(k))$, etc.
Another complicated proof.

How about 2006 Campagna "Security bounds for the NIST codebook-based deterministic random bit generator"? Doesn't prove anything about rekeying.

2017 AES-GCM-SIV bounds?
Big errors found by Iwata–Seurin.

ated; error-prone.

blitz–Menezes:
assumptions are wrong.

tz–Lindell: public denials.

rnstein–Lange:
assumptions are wrong.

etrzak: fixed theorem
blitz–Menezes is wrong.

etrzak, 2013 Koblitz–
s, 2014 Gaži–Pietrzak–
another NMAC proof,
licated as cascade proof.

Hmmm. CCS 2005 Barak–Halevi "A model and architecture for pseudo-random generation with applications to /dev/random"? RNG outputs $F(k)$, $F(G(k))$, etc. Another complicated proof.

How about 2006 Campagna "Security bounds for the NIST codebook-based deterministic random bit generator"? Doesn't prove anything about rekeying.

2017 AES-GCM-SIV bounds? Big errors found by Iwata–Seurin.

A simple

Rememb

There ar
Cipher 1
Cipher 2

r-prone.

ezes:
ons are wrong.

: public denials.

ange:
ons are wrong.

ed theorem
ezes is wrong.

13 Koblitz–
ži–Pietrzak–
MAC proof,
cascade proof.

Hmmm. CCS 2005 Barak–Halevi "A model and architecture for pseudo-random generation with applications to `/dev/random`"? RNG outputs $F(k)$, $F(G(k))$, etc. Another complicated proof.

How about 2006 Campagna "Security bounds for the NIST codebook-based deterministic random bit generator"? Doesn't prove anything about rekeying.

2017 AES-GCM-SIV bounds? Big errors found by Iwata–Seurin.

<u>A simple tight new</u>

Remember the goa

There are T keys.
Cipher 1: key $\mapsto$ r
Cipher 2: subkey

rong.

enials.

rong.

m

ong.

–

k–

f,

roof.

Hmmm. CCS 2005 Barak–Halevi "A model and architecture for pseudo-random generation with applications to `/dev/random`"? RNG outputs $F(k)$, $F(G(k))$, etc. Another complicated proof.

How about 2006 Campagna "Security bounds for the NIST codebook-based deterministic random bit generator"? Doesn't prove anything about rekeying.

2017 AES-GCM-SIV bounds? Big errors found by Iwata–Seurin.

A simple tight new proof

Remember the goal: analyze

There are T keys.

Cipher 1: key $\mapsto$ many subk

Cipher 2: subkey $\mapsto$ outputs

Hmmm. CCS 2005 Barak–Halevi "A model and architecture for pseudo-random generation with applications to `/dev/random`"? RNG outputs $F(k)$, $F(G(k))$, etc. Another complicated proof.

How about 2006 Campagna "Security bounds for the NIST codebook-based deterministic random bit generator"? Doesn't prove anything about rekeying.

2017 AES-GCM-SIV bounds? Big errors found by Iwata–Seurin.

A simple tight new proof

Remember the goal: analyze p_T.

There are T keys.
Cipher 1: key $\mapsto$ many subkeys.
Cipher 2: subkey $\mapsto$ outputs.

Hmmm. CCS 2005 Barak–Halevi "A model and architecture for pseudo-random generation with applications to `/dev/random`"? RNG outputs $F(k)$, $F(G(k))$, etc. Another complicated proof.

How about 2006 Campagna "Security bounds for the NIST codebook-based deterministic random bit generator"? Doesn't prove anything about rekeying.

2017 AES-GCM-SIV bounds? Big errors found by Iwata–Seurin.

A simple tight new proof

Remember the goal: analyze p_T.

There are T keys.
Cipher 1: key $\mapsto$ many subkeys.
Cipher 2: subkey $\mapsto$ outputs.

New proof has just two steps.

Hmmm. CCS 2005 Barak–Halevi "A model and architecture for pseudo-random generation with applications to `/dev/random`"? RNG outputs $F(k)$, $F(G(k))$, etc. Another complicated proof.

How about 2006 Campagna "Security bounds for the NIST codebook-based deterministic random bit generator"? Doesn't prove anything about rekeying.

2017 AES-GCM-SIV bounds? Big errors found by Iwata–Seurin.

A simple tight new proof

Remember the goal: analyze p_T.

There are T keys.
Cipher 1: key $\mapsto$ many subkeys.
Cipher 2: subkey $\mapsto$ outputs.

New proof has just two steps.

Step 1. Replace all subkeys. Distinguisher $\Rightarrow$ T-target attack against cipher 1.

Hmmm. CCS 2005 Barak–Halevi "A model and architecture for pseudo-random generation with applications to `/dev/random`"? RNG outputs $F(k)$, $F(G(k))$, etc. Another complicated proof.

How about 2006 Campagna "Security bounds for the NIST codebook-based deterministic random bit generator"? Doesn't prove anything about rekeying.

2017 AES-GCM-SIV bounds? Big errors found by Iwata–Seurin.

A simple tight new proof

Remember the goal: analyze p_T.

There are T keys.
Cipher 1: key $\mapsto$ many subkeys.
Cipher 2: subkey $\mapsto$ outputs.

New proof has just two steps.

Step 1. Replace all subkeys. Distinguisher $\Rightarrow$ T-target attack against cipher 1.

Step 2. Replace all outputs. Distinguisher $\Rightarrow$ ($T \cdot$ many)-target attack against cipher 2.

CCS 2005 Barak–Halevi
el and architecture for
random generation with
ions to /dev/random"?
tputs $F(k)$, $F(G(k))$, etc.
complicated proof.

out 2006 Campagna
y bounds for the NIST
k-based deterministic
bit generator"? Doesn't
ything about rekeying.

ES-GCM-SIV bounds?
rs found by Iwata–Seurin.

A simple tight new proof

Remember the goal: analyze p_T.

There are T keys.

Cipher 1: key $\mapsto$ many subkeys.

Cipher 2: subkey $\mapsto$ outputs.

New proof has just two steps.

Step 1. Replace all subkeys.
Distinguisher $\Rightarrow$ T-target
attack against cipher 1.

Step 2. Replace all outputs.
Distinguisher $\Rightarrow$ ($T \cdot$ many)-target
attack against cipher 2.

multi-ta
two-le
securi

r

multi-ta
one-le
securi

i

multi-ta
many-l
securi

X: FOCS
Krawczy
not suita

5 Barak–Halevi
hitecture for
neration with
ev/random"?
), $F(G(k))$, etc.
ted proof.

Campagna
for the NIST
eterministic
tor"? Doesn't
out rekeying.

IV bounds?
y Iwata–Seurin.

A simple tight new proof

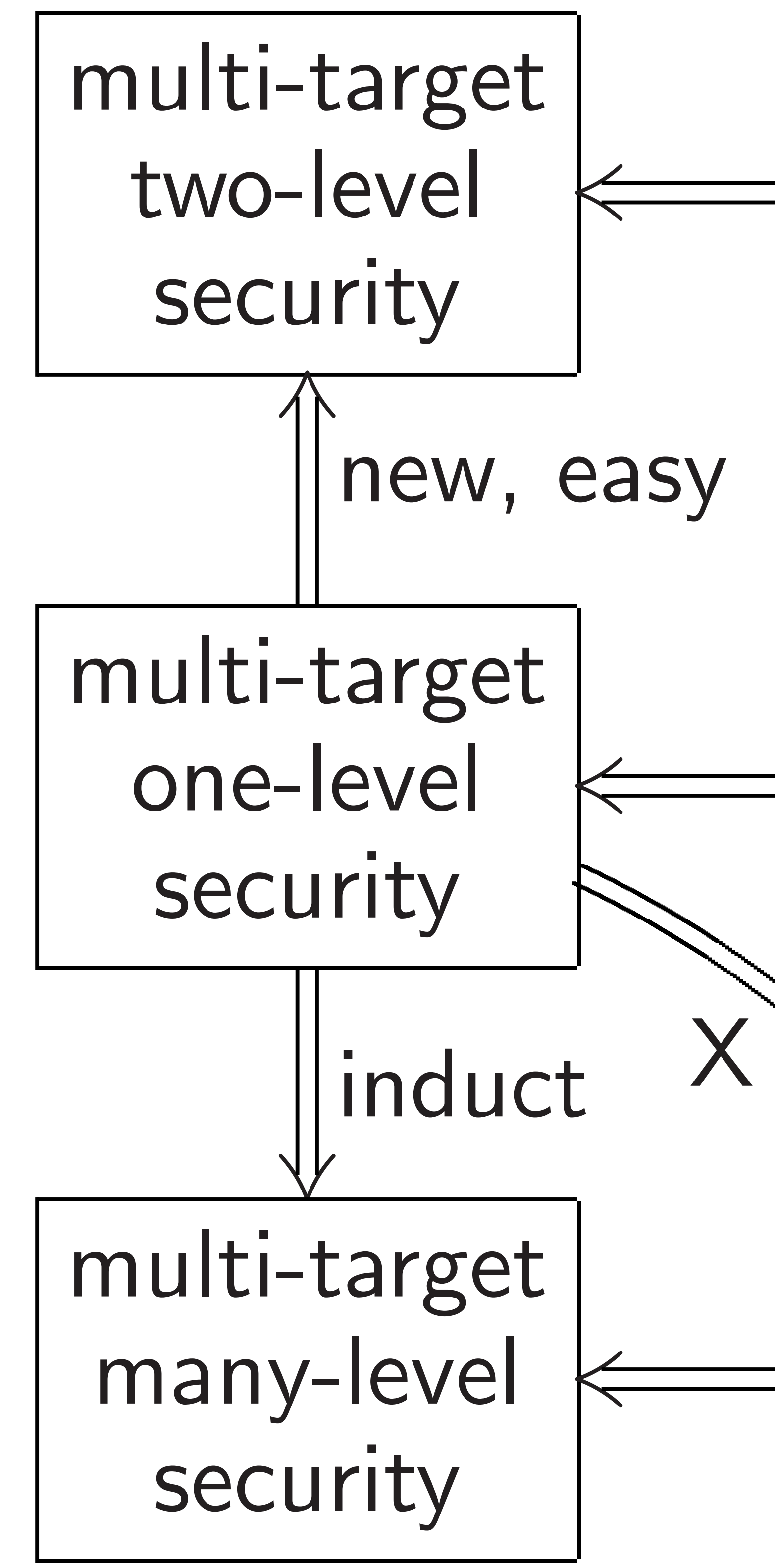
Remember the goal: analyze p_T.

There are T keys.
Cipher 1: key $\mapsto$ many subkeys.
Cipher 2: subkey $\mapsto$ outputs.

New proof has just two steps.

Step 1. Replace all subkeys.
Distinguisher $\Rightarrow$ T-target attack against cipher 1.

Step 2. Replace all outputs.
Distinguisher $\Rightarrow$ ($T \cdot$ many)-target attack against cipher 2.



X: FOCS 1996 Be
Krawczyk Lemma
not suitable for inc

Halevi
or
with
m"?
), etc.

ST
ic
esn't
ng.

s?
eurin.

A simple tight new proof

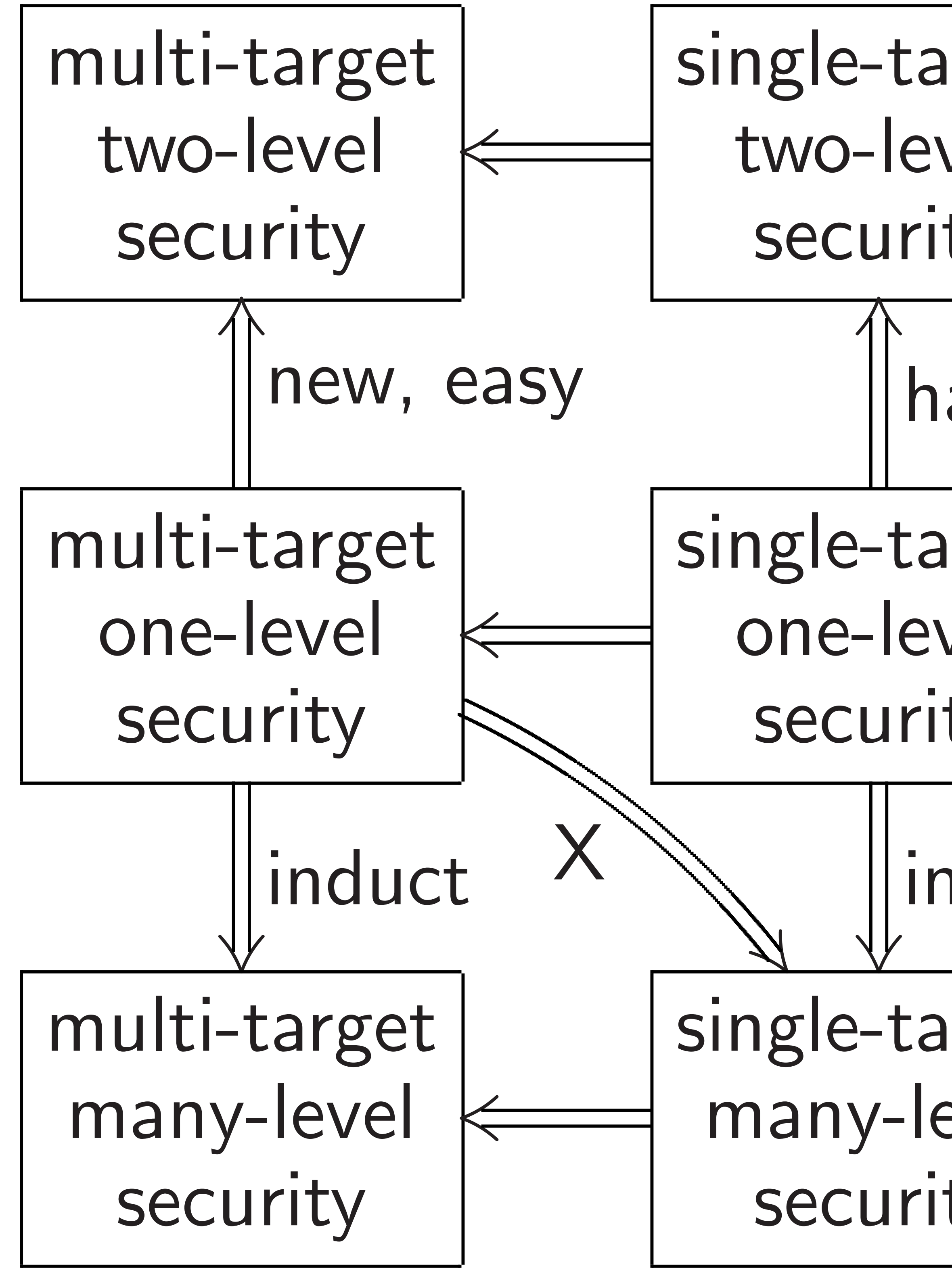
Remember the goal: analyze p_T.

There are T keys.
Cipher 1: key $\mapsto$ many subkeys.
Cipher 2: subkey $\mapsto$ outputs.

New proof has just two steps.

Step 1. Replace all subkeys.
Distinguisher $\Rightarrow$ T-target
attack against cipher 1.

Step 2. Replace all outputs.
Distinguisher $\Rightarrow$ $(T \cdot \text{many})$-target
attack against cipher 2.



X: FOCS 1996 Bellare–Cane
Krawczyk Lemma 3.2. Hard
not suitable for induction.

A simple tight new proof

Remember the goal: analyze p_T.

There are T keys.

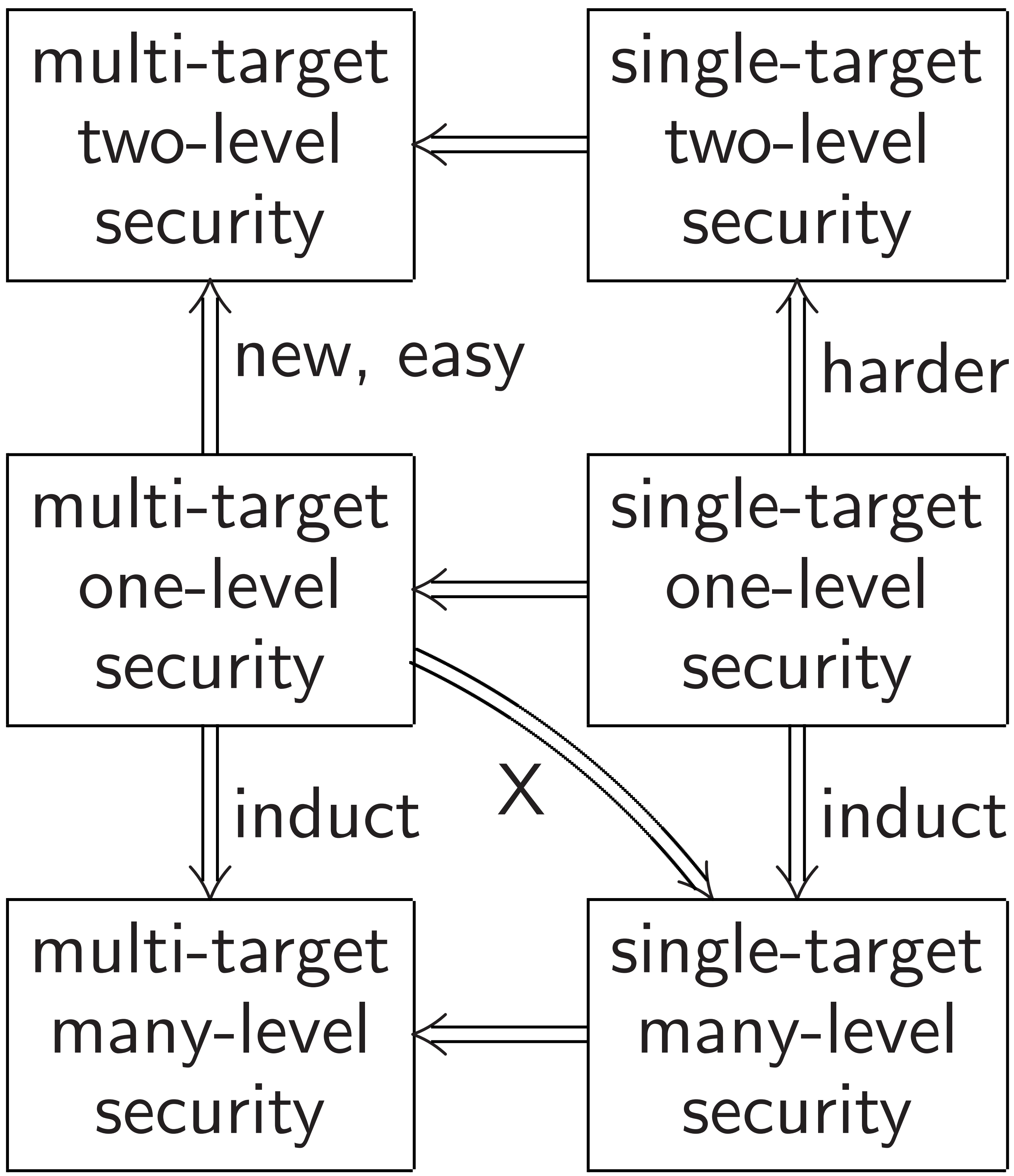
Cipher 1: key $\mapsto$ many subkeys.

Cipher 2: subkey $\mapsto$ outputs.

New proof has just two steps.

Step 1. Replace all subkeys.
Distinguisher $\Rightarrow$ T-target attack against cipher 1.

Step 2. Replace all outputs.
Distinguisher $\Rightarrow$ ($T \cdot$ many)-target attack against cipher 2.



X: FOCS 1996 Bellare–Canetti–Krawczyk Lemma 3.2. Harder; not suitable for induction.