Trapdoor functions from the Computational Diffie-Hellman Assumption

Sanjam Garg[1] **Mohammad Hajiabadi**[1,2]

[1]University of California, Berkeley

[2]University of Virginia

August 22, 2018

Classical Public-Key Crypto

TDF [DH76]

New Directions in Cryptography

Classical Public-Key Crypto

TDF [DH76]

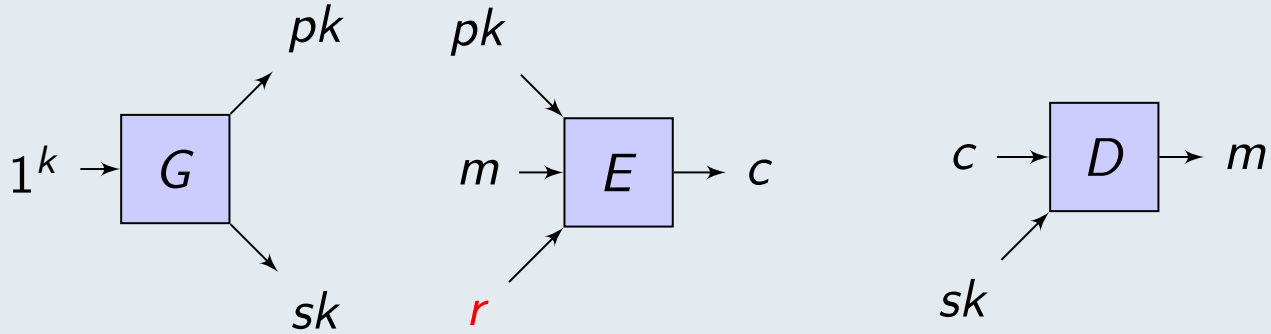
New Directions in Cryptography

PKE [GM82]

Probabilistic Encryption &
How To Play Mental Poker Keeping Secret
All Partial Information

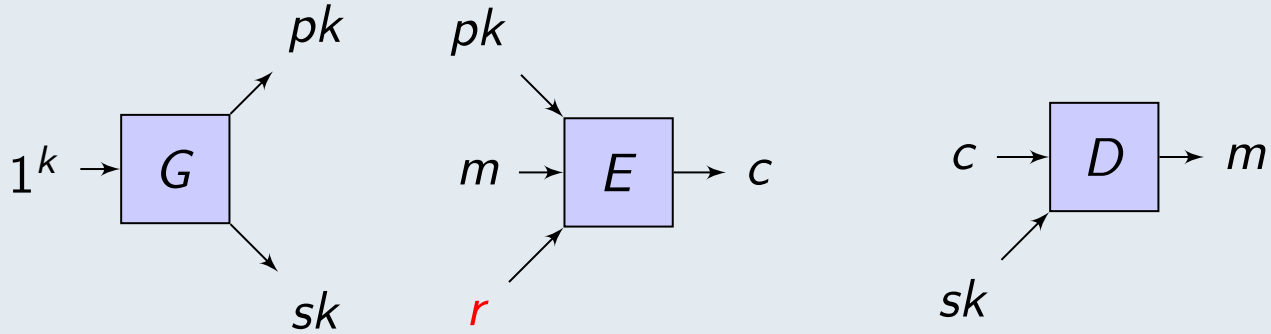
PKE and TDF

PKE



PKE and TDF

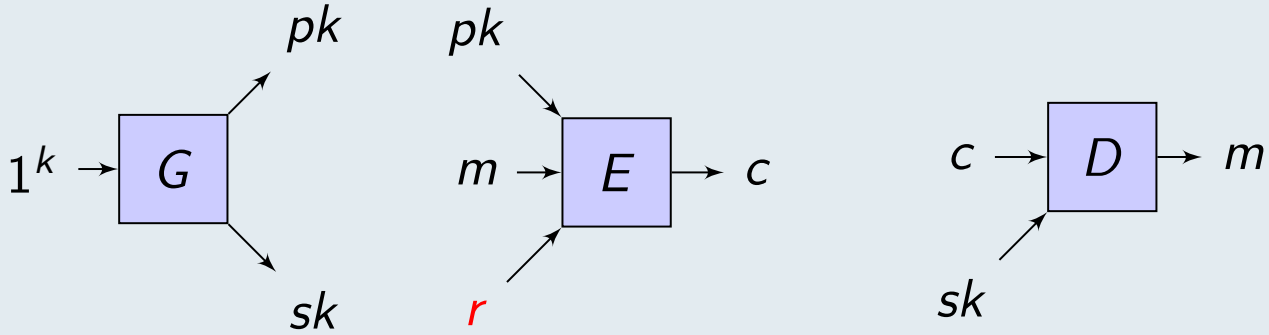
PKE



Security: $\forall m_0, m_1 : (pk, E(pk, m_0; r_0)) \stackrel{c}{\equiv} (pk, E(pk, m_1; r_1))$

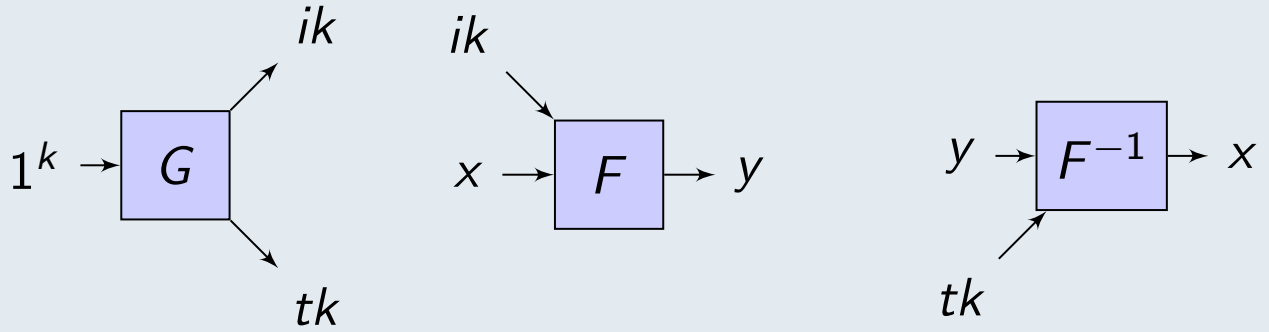
PKE and TDF

PKE



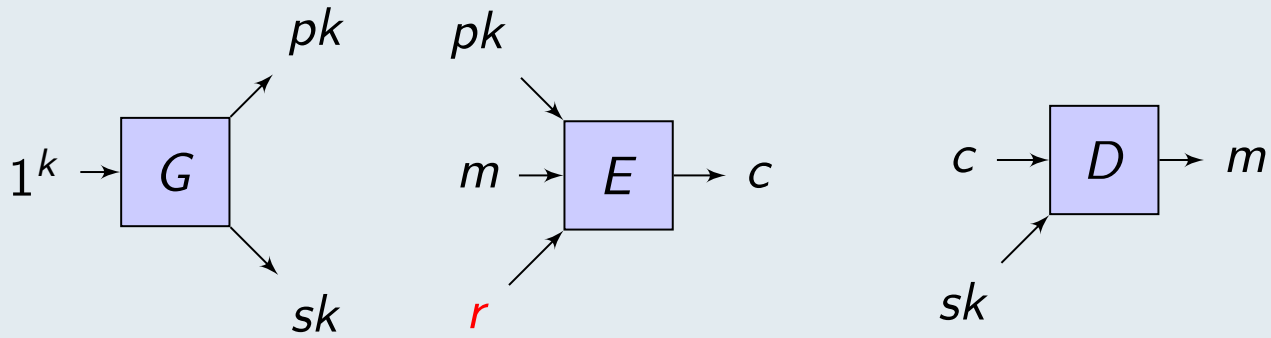
Security: $\forall m_0, m_1 : (pk, E(pk, m_0; r_0)) \stackrel{c}{\equiv} (pk, E(pk, m_1; r_1))$

TDF



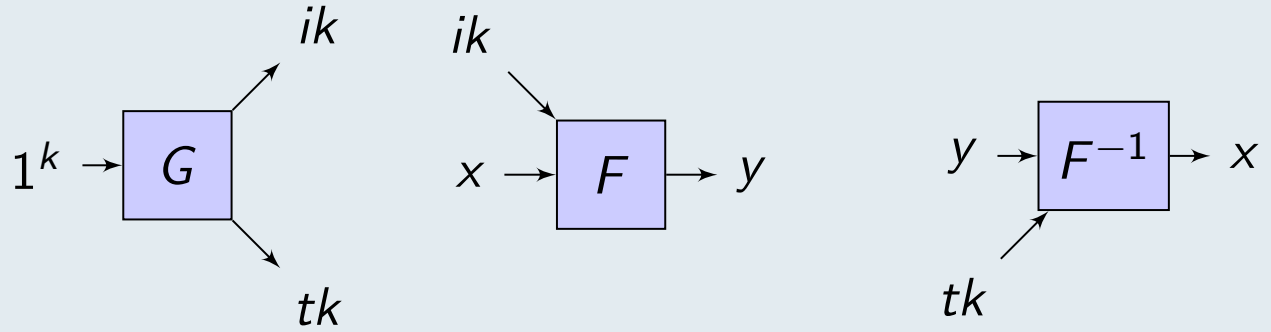
PKE and TDF

PKE



Security: $\forall m_0, m_1 : (pk, E(pk, m_0; r_0)) \stackrel{c}{\equiv} (pk, E(pk, m_1; r_1))$

TDF



One-wayness Security: $(ik, \mathrm{F}(ik, x)) \stackrel{?}{\rightarrow} x$ is hard for random ik, x.

TDF vs PKE

Main Difference

- No randomness used in the evaluation algorithm of TDF.

TDF vs PKE

Main Difference

- No randomness used in the evaluation algorithm of TDF.

Relations

- TDF implies the existence of PKE. [Yao'82, GM'82].

TDF vs PKE

Main Difference

- No randomness used in the evaluation algorithm of TDF.

Relations

- TDF implies the existence of PKE. [Yao'82, GM'82].
- TDF impossible from PKE w.r.t. black-box techniques [GMR'01].

TDF Usefulness

ik_1, ik_2

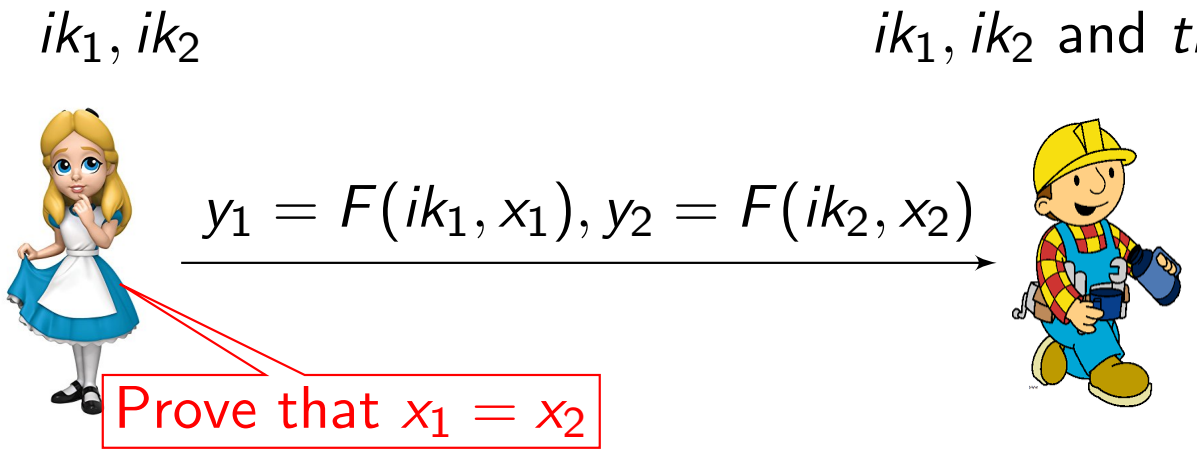
ik_1, ik_2 and tk_1

TDF Usefulness

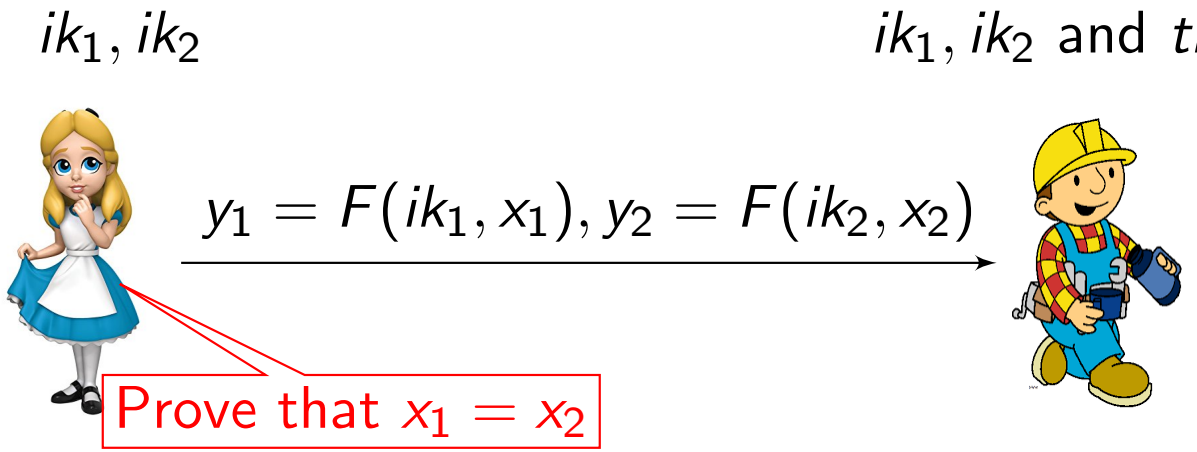
ik_1, ik_2 $\qquad$ ik_1, ik_2 and tk_1

$$y_1 = F(ik_1, x_1), y_2 = F(ik_2, x_2)$$

TDF Usefulness



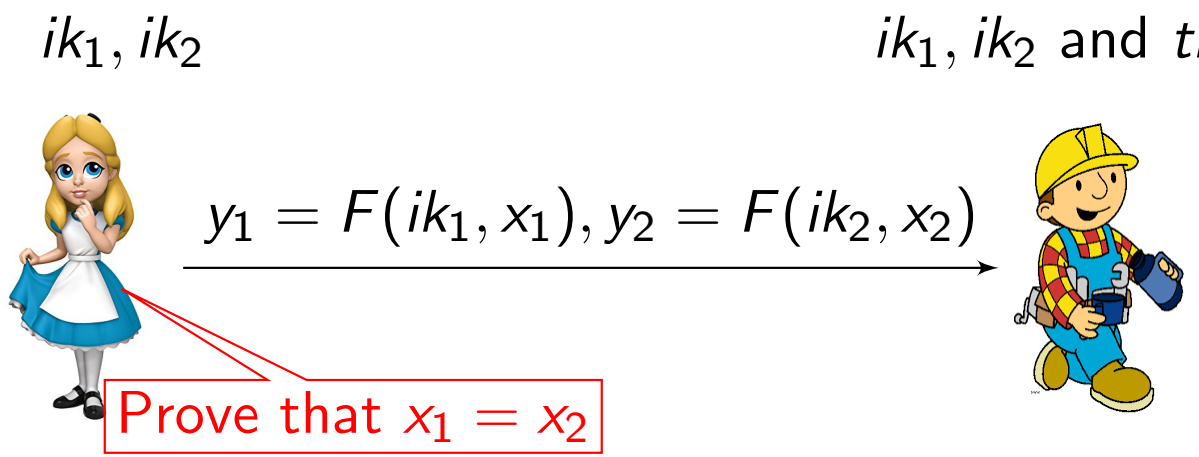
TDF Usefulness



Bob: Compute $x_1 = \mathsf{F}^{-1}(\mathsf{tk}_1, y_1)$ and check if $y_2 = \mathsf{F}(\mathsf{ik}_2, x_1)$.

- Application: black-box constructions of CCA-secure PKE ([PW'08,RS'09, etc]).

TDF Usefulness



Bob: Compute $x_1 = \mathrm{F}^{-1}(\mathrm{tk}_1, y_1)$ and check if $y_2 = \mathrm{F}(\mathrm{ik}_2, x_1)$.

- Application: black-box constructions of CCA-secure PKE ([PW'08,RS'09, etc]).

PKE instead of TDF

- Consistency check: require some kind of proof (e.g., NIZK). [BFY90,NY90]

What assumptions are sufficient for TDFs?

- Factoring
- DDH and LWE [PW08]

What assumptions are sufficient for TDFs?

- Factoring
- DDH and LWE [PW08]

Big gap from PKE!

What assumptions are sufficient for TDFs?

- Factoring
- DDH and LWE [PW08]

Big gap from PKE!

This talk: We can do it from CDH.

CDH and DDH

$\mathbb{G}$: group of order p and generator g.

CDH and DDH

$\mathbb{G}$: group of order p and generator g.

Computational Diffie-Hellman (CDH)

- Hard to compute g^{xy} from (g, g^x, g^y), where $x, y \leftarrow \mathbb{Z}_p$.

CDH and DDH

$\mathbb{G}$: group of order p and generator g.

Computational Diffie-Hellman (CDH)

- Hard to compute g^{xy} from (g, g^x, g^y), where $x, y \leftarrow \mathbb{Z}_p$.

Decisional Diffie-Hellman (DDH)

- $(g, g^x, g^y, g^{xy}) \stackrel{c}{\equiv} (g, g^x, g^y, g^z)$, where $x, y, z \leftarrow \mathbb{Z}_p$

Why is CDH Preferable?

Why is CDH Preferable?

- CDH is a weaker assumption.
  - There are groups in which CDH is conjectured to be hard but DDH is easy (e.g., $\mathbb{Z}_p^*$, groups with pairings).

Main Challenge in Building TDF from DH-Related Assumptions

Why is constructing TDF from Diffie-Hellman assumptions difficult?

Main Challenge in Building TDF from DH-Related Assumptions

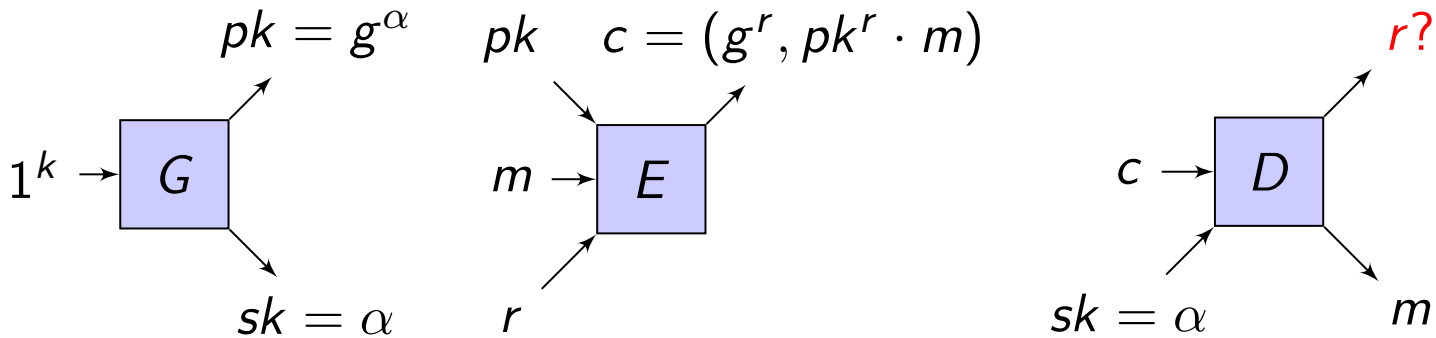
Why is constructing TDF from Diffie-Hellman assumptions difficult?

It doesn't naturally offer trapdoors!

TDF from DDH (Failed Idea Using ElGamal Encryption)

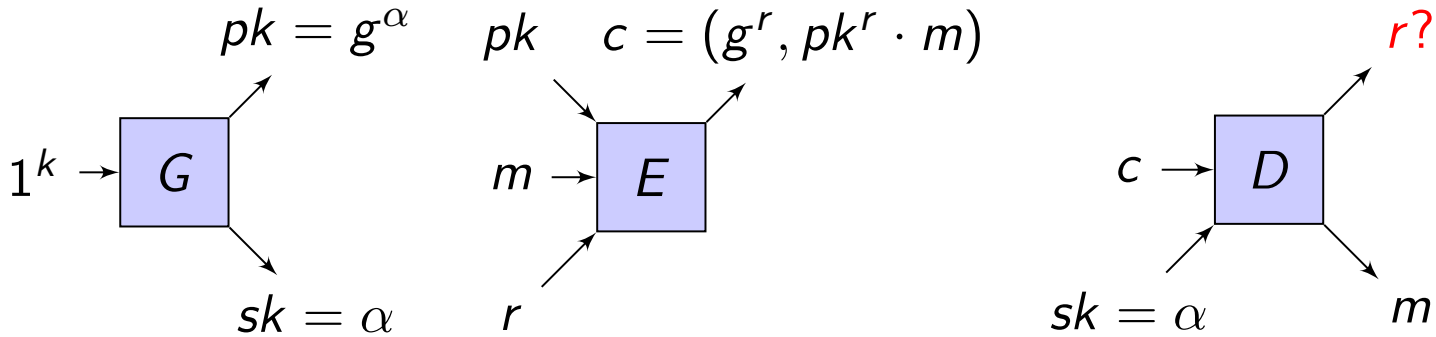
TDF from DDH (Failed Idea Using ElGamal Encryption)

$(\mathbb{G}, g)$, $|\mathbb{G}| = p$.



TDF from DDH (Failed Idea Using ElGamal Encryption)

$(\mathbb{G}, g)$, $|\mathbb{G}| = p$.



Main bottleneck in designing TDFs

- Recovering r: solving the Discrete Log!

DDH-Based TDF [Peikert-Waters'08]

$(\mathbb{G}, g)$, $|\mathbb{G}| = p$.

DDH-Based TDF [Peikert-Waters'08]

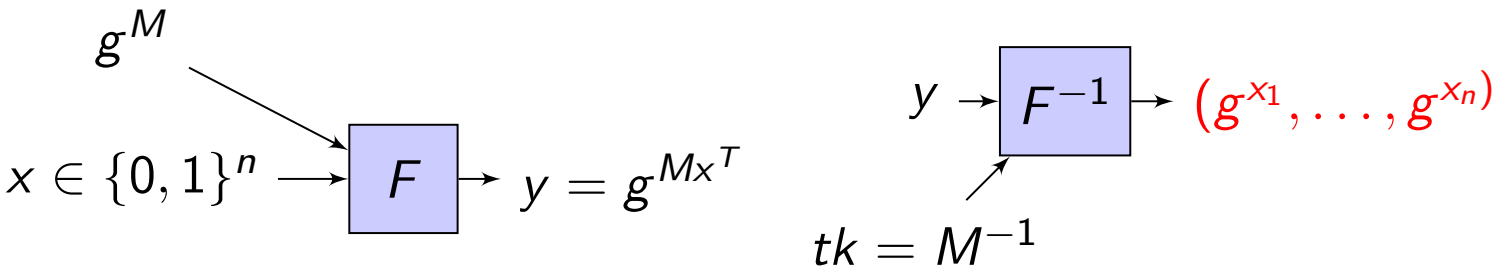
$(\mathbb{G}, g)$, $|\mathbb{G}| = p$.

- $ik = g^M$ where $M \in \mathbb{Z}_p^{n \times n}$ (and invertible) and $tk = M^{-1}$

DDH-Based TDF [Peikert-Waters'08]

$(\mathbb{G}, g)$, $|\mathbb{G}| = p$.

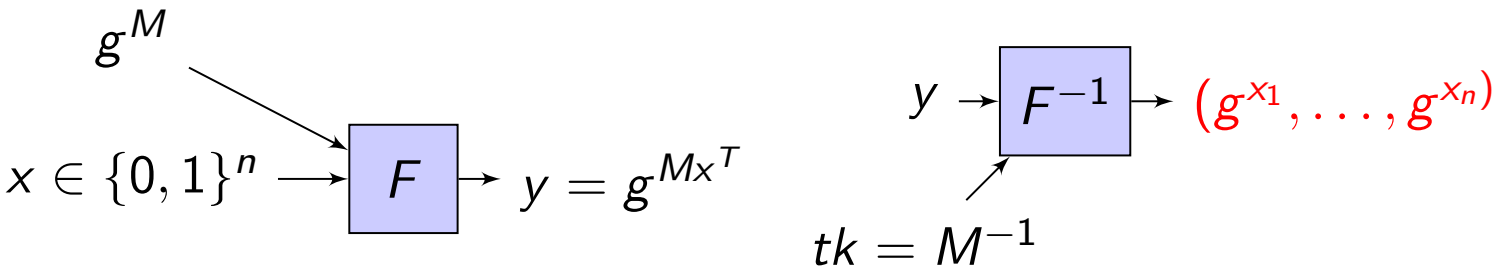
- $ik = g^M$ where $M \in \mathbb{Z}_p^{n \times n}$ (and invertible) and $tk = M^{-1}$



DDH-Based TDF [Peikert-Waters'08]

$(\mathbb{G}, g)$, $|\mathbb{G}| = p$.

- $ik = g^M$ where $M \in \mathbb{Z}_p^{n \times n}$ (and invertible) and $tk = M^{-1}$

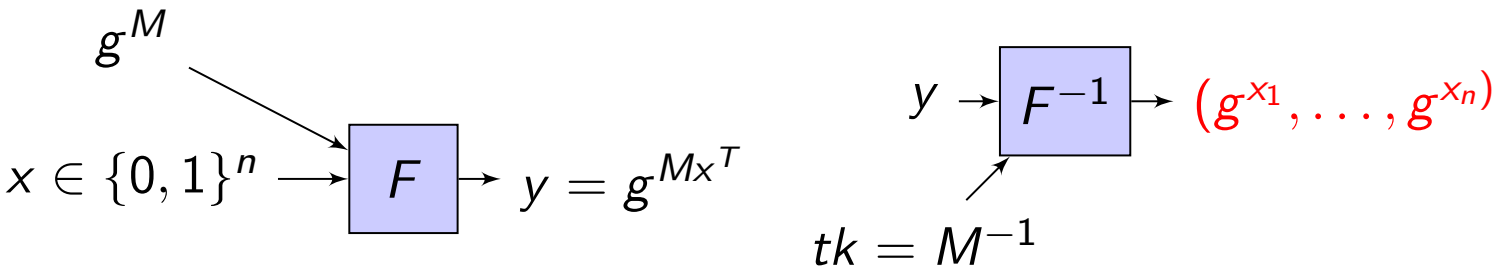


- Can solve discrete-log as $x_1 \ldots x_n \in \{0,1\}$!

DDH-Based TDF [Peikert-Waters'08]

$(\mathbb{G}, g)$, $|\mathbb{G}| = p$.

- $ik = g^M$ where $M \in \mathbb{Z}_p^{n \times n}$ (and invertible) and $tk = M^{-1}$



- Can solve discrete-log as $x_1 \ldots x_n \in \{0,1\}$!

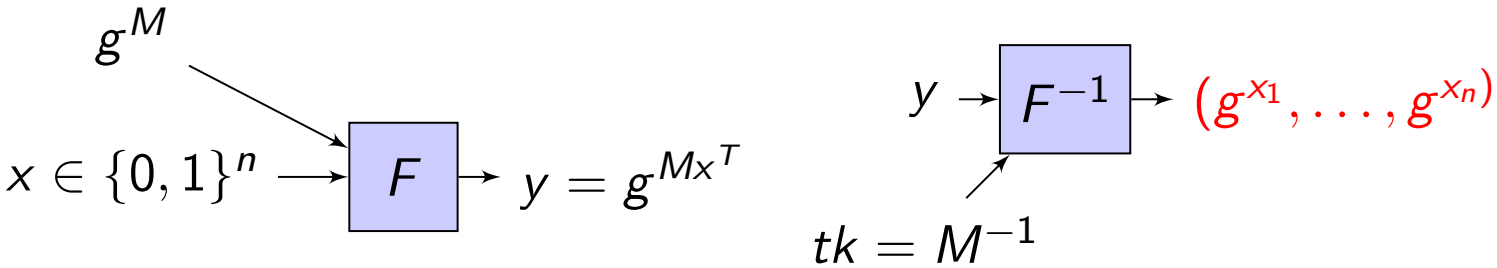
One-wayness

- Matrix pseudorandomness [NR97]: DDH implies $g^M \stackrel{c}{\equiv} g^{M'}$, where M is a random invertible matrix and M' is a random rank-one matrix.

DDH-Based TDF [Peikert-Waters'08]

$(\mathbb{G}, g)$, $|\mathbb{G}| = p$.

- $ik = g^M$ where $M \in \mathbb{Z}_p^{n \times n}$ (and invertible) and $tk = M^{-1}$



- Can solve discrete-log as $x_1 \ldots x_n \in \{0,1\}$!

One-wayness

- Matrix pseudorandomness [NR97]: DDH implies $g^M \stackrel{c}{\equiv} g^{M'}$, where M is a random invertible matrix and M' is a random rank-one matrix.
- CDH is not known to imply rank indistinguishability.

Our Methodology for building TDF from CDH

- Derandomizing a class of PKE

Our Methodology for building TDF from CDH

- Derandomizing a class of PKE
  - TDFs from *recyclable targeted key-encapsulation schemes* (Recyclable Targeted KEMs) [DG'17, BBS'03]

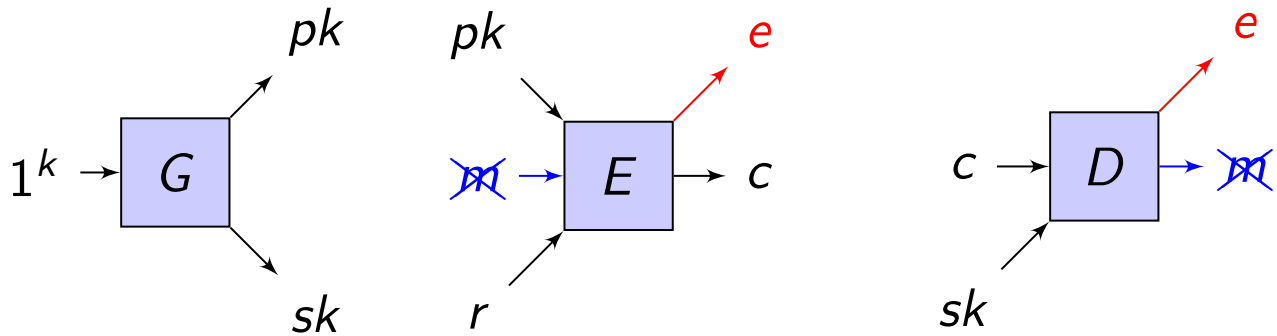
Our Methodology for building TDF from CDH

- Derandomizing a class of PKE
  - TDFs from *recyclable targeted key-encapsulation schemes* (Recyclable Targeted KEMs) [DG'17, BBS'03]

Plan for the Rest of the talk

- Define Recyclable Targeted KEM
- CDH $\Rightarrow$ Recyclable Targeted KEM (Not discussed. See [DG'17].)
- Recyclable Targeted KEM $\Rightarrow$ TDF

Key-Encapsulation Mechanism



e is always a single bit.

Recyclable Targetted KEM

Recyclable Targetted KEM

Targeting Property [DG'17]

- $\mathsf{E}(\mathsf{pk}, (i, b); r) = (\mathsf{ct}, e)$
- $\mathsf{D}(\mathsf{sk}, \mathsf{ct}) = e$ if $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{K}(1^\lambda)$ and $\mathsf{sk}_i = b$.

Recyclable Targetted KEM

Targeting Property [DG'17]

- $\mathsf{E}(\mathsf{pk}, (i, b); r) = (\mathsf{ct}, e)$
- $\mathsf{D}(\mathsf{sk}, \mathsf{ct}) = e$ if $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{K}(1^\lambda)$ and $\mathsf{sk}_i = b$.
- Security: $(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}, e) \stackrel{c}{\equiv} (\mathsf{pk}, \mathsf{sk}, \mathsf{ct}, e')$, where $(\mathsf{ct}, e) \stackrel{\$}{\leftarrow} \mathsf{E}(\mathsf{pk}, (i, 1 - \mathsf{sk}_i); r)$ and $e' \stackrel{\$}{\leftarrow} \{0, 1\}$.

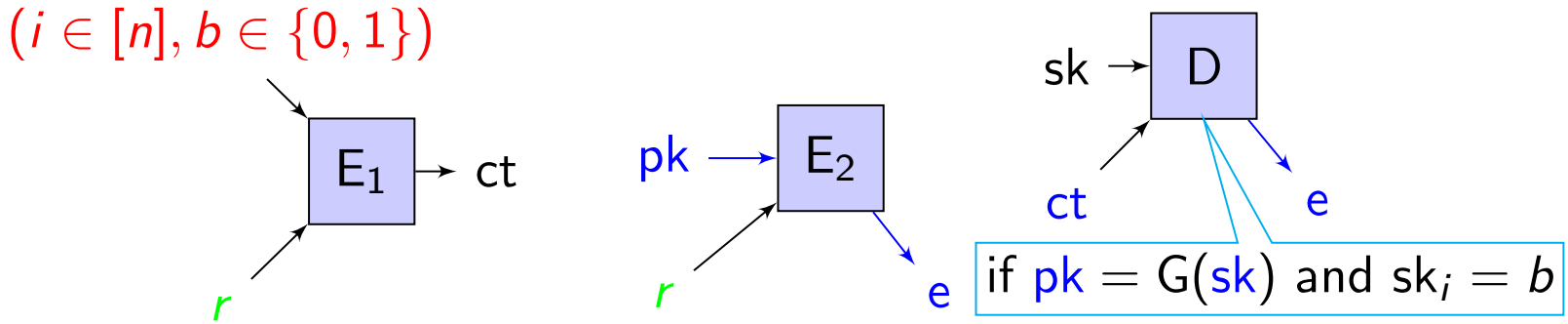
Recyclable Targetted KEM

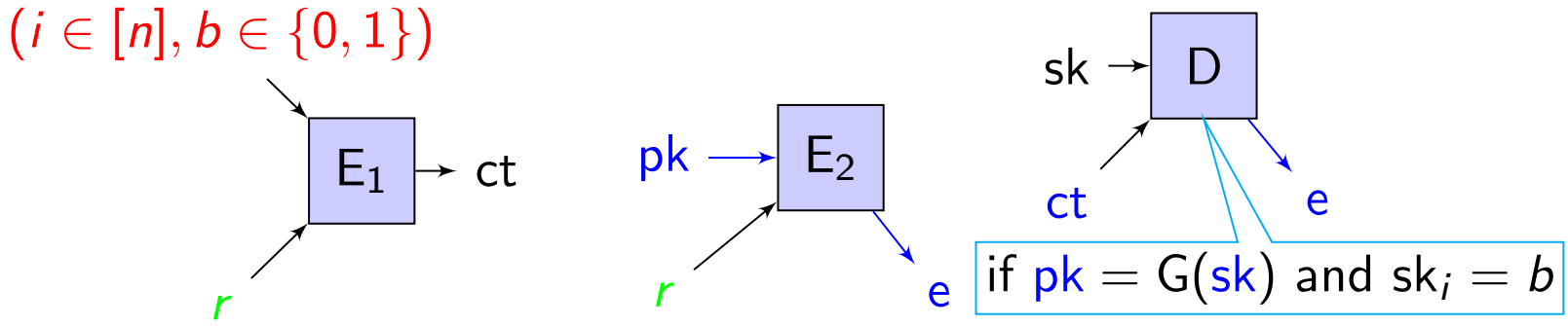
Targeting Property [DG'17]

- $\mathsf{E}(\mathsf{pk}, (i, b); r) = (\mathsf{ct}, e)$
- $\mathsf{D}(\mathsf{sk}, \mathsf{ct}) = e$ if $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{K}(1^\lambda)$ and $\mathsf{sk}_i = b$.
- Security: $(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}, e) \stackrel{c}{\equiv} (\mathsf{pk}, \mathsf{sk}, \mathsf{ct}, e')$, where $(\mathsf{ct}, e) \stackrel{\$}{\leftarrow} \mathsf{E}(\mathsf{pk}, (i, 1 - \mathsf{sk}_i); r)$ and $e' \stackrel{\$}{\leftarrow} \{0, 1\}$.

Recyclability

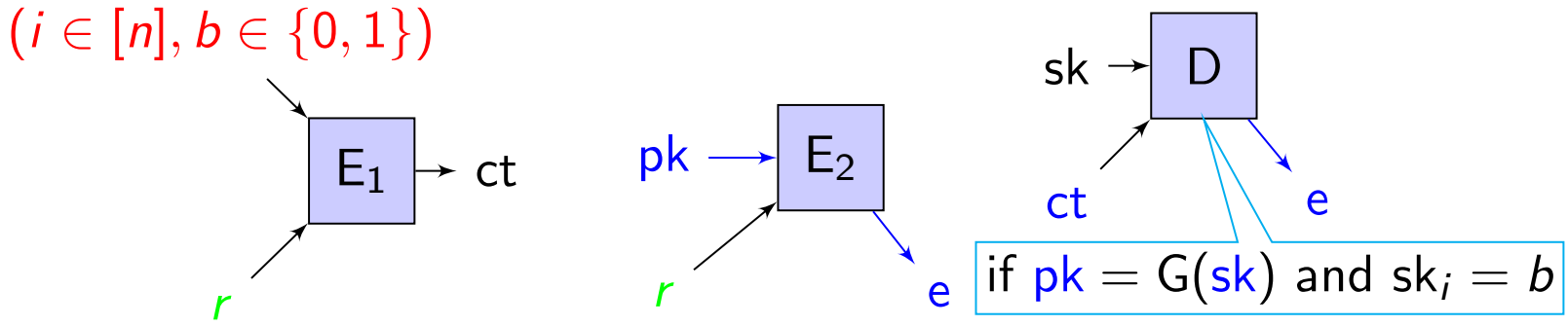
ct does not depend on pk. So
$\mathsf{E}(\mathsf{pk}, (i, b); r) = (\mathsf{E}_1((i, b); r), \mathsf{E}_2(\mathsf{pk}, (i, b); r)) = (\mathsf{ct}, e)$


$(i \in [n], b \in \{0,1\})$
E_1
ct
r
pk
E_2
r
e
sk
D
ct
e
if pk = G(sk) and $sk_i = b$



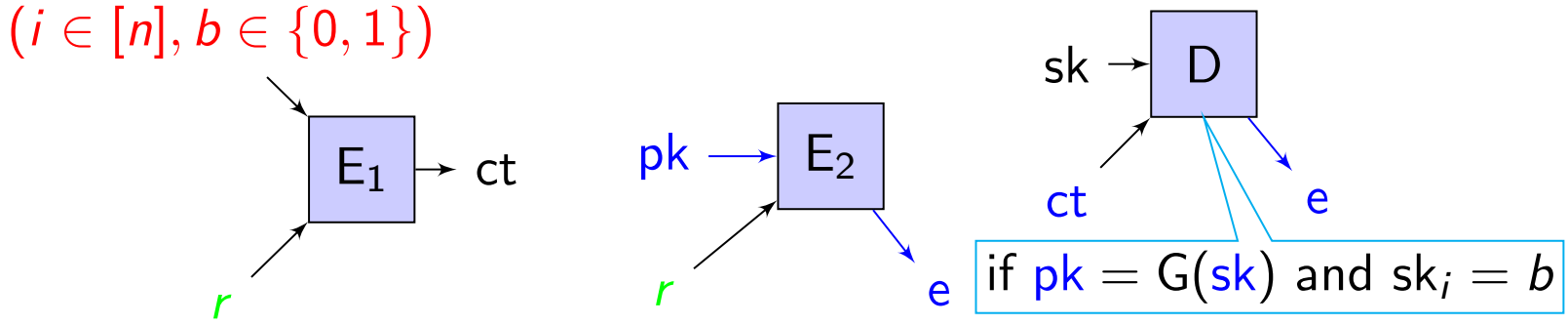
Simple construction for recovering the first bit of the input.

- $\mathsf{tk} = \begin{pmatrix} r_1 \\ r_1' \end{pmatrix}$ and $\mathsf{ik} = \begin{pmatrix} \mathsf{ct}_1 \\ \mathsf{ct}_1' \end{pmatrix} = \begin{pmatrix} \mathsf{E}_1((i{=}1, b{=}0); r_1) \\ \mathsf{E}_1((i{=}1, b{=}1); r_1') \end{pmatrix}$



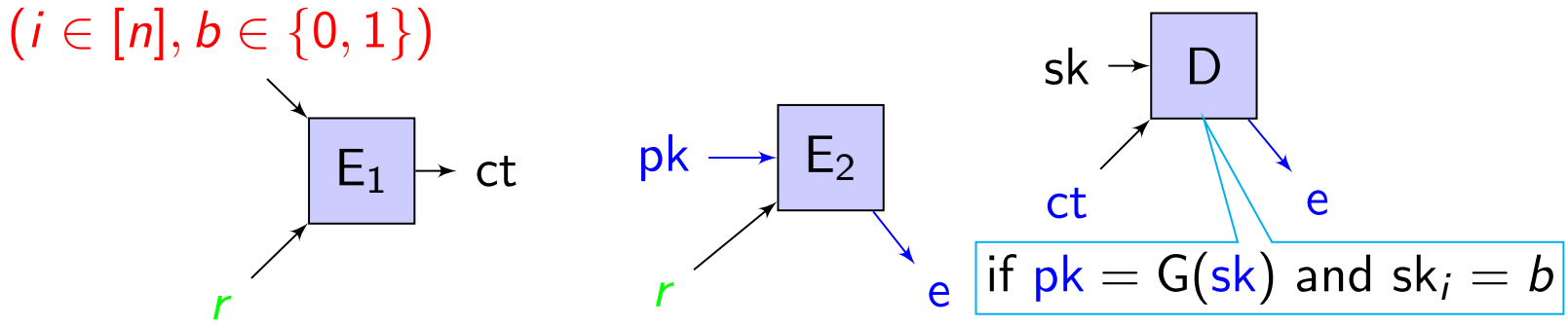
Simple construction for recovering the first bit of the input.

- $\mathsf{tk} = \begin{pmatrix} r_1 \\ r_1' \end{pmatrix}$ and $\mathsf{ik} = \begin{pmatrix} \mathsf{ct}_1 \\ \mathsf{ct}_1' \end{pmatrix} = \begin{pmatrix} \mathsf{E}_1((i{=}1, b{=}0); r_1) \\ \mathsf{E}_1((i{=}1, b{=}1); r_1') \end{pmatrix}$
- $\mathsf{F}(\mathsf{ik}, \mathsf{sk})$:



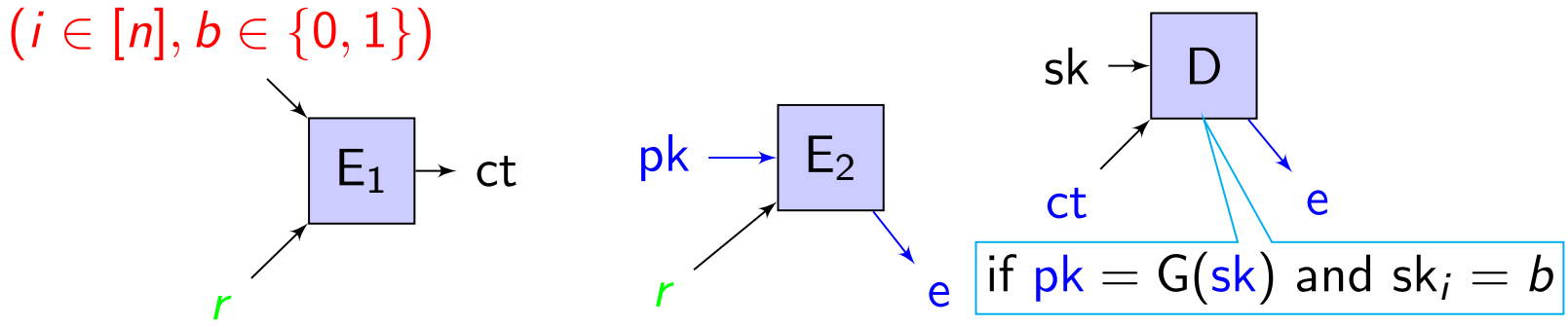
Simple construction for recovering the first bit of the input.

- $\mathsf{tk} = \begin{pmatrix} r_1 \\ r_1' \end{pmatrix}$ and $\mathsf{ik} = \begin{pmatrix} \mathsf{ct}_1 \\ \mathsf{ct}_1' \end{pmatrix} = \begin{pmatrix} \mathsf{E}_1((i{=}1, b{=}0); r_1) \\ \mathsf{E}_1((i{=}1, b{=}1); r_1') \end{pmatrix}$
- $\mathsf{F}(\mathsf{ik}, \mathsf{sk})$: let $\mathsf{pk} = \mathsf{G}(\mathsf{sk})$.



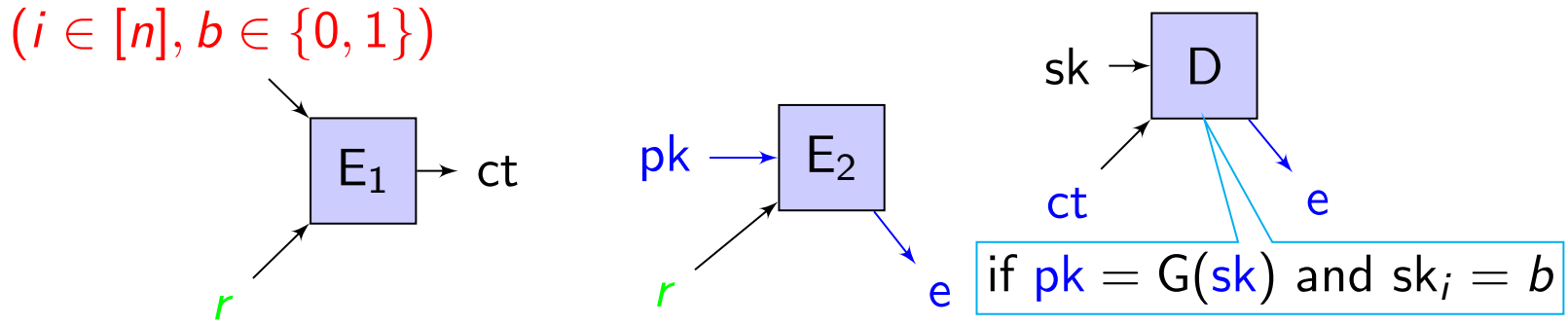
Simple construction for recovering the first bit of the input.

- $\mathsf{tk} = \begin{pmatrix} r_1 \\ r_1' \end{pmatrix}$ and $\mathsf{ik} = \begin{pmatrix} \mathsf{ct}_1 \\ \mathsf{ct}_1' \end{pmatrix} = \begin{pmatrix} \mathsf{E}_1((i{=}1, b{=}0); r_1) \\ \mathsf{E}_1((i{=}1, b{=}1); r_1') \end{pmatrix}$
- $\mathsf{F}(\mathsf{ik}, \mathsf{sk})$: let $\mathsf{pk} = \mathsf{G}(\mathsf{sk})$.
  - if $\mathsf{sk}_1 = 0$, then return $(\mathsf{pk}, \mathsf{D}(\mathsf{sk}, \mathsf{ct}_1))$



Simple construction for recovering the first bit of the input.

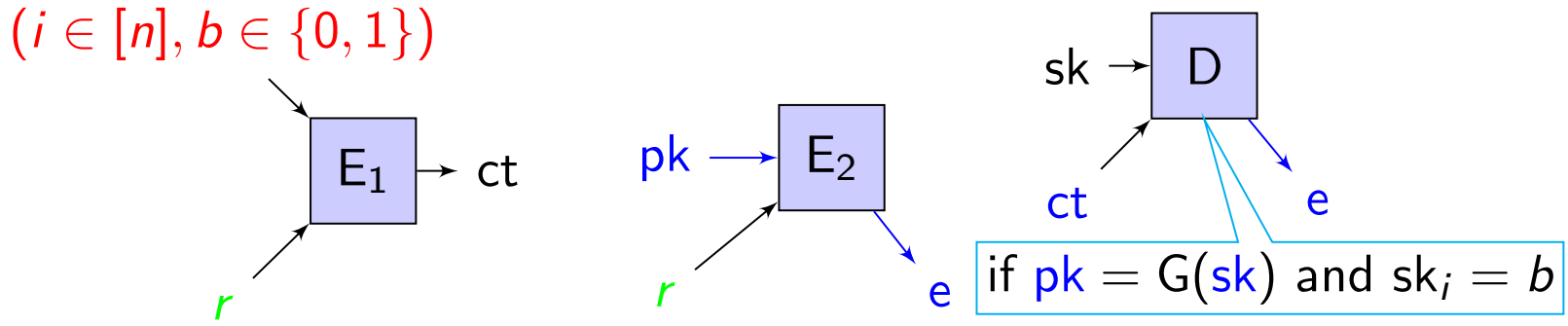
- $\mathsf{tk} = \begin{pmatrix} r_1 \\ r_1' \end{pmatrix}$ and $\mathsf{ik} = \begin{pmatrix} \mathsf{ct}_1 \\ \mathsf{ct}_1' \end{pmatrix} = \begin{pmatrix} \mathsf{E}_1((i{=}1, b{=}0); r_1) \\ \mathsf{E}_1((i{=}1, b{=}1); r_1') \end{pmatrix}$
- $\mathsf{F}(\mathsf{ik}, \mathsf{sk})$: let $\mathsf{pk} = \mathsf{G}(\mathsf{sk})$.
  - if $\mathsf{sk}_1 = 0$, then return $(\mathsf{pk}, \mathsf{D}(\mathsf{sk}, \mathsf{ct}_1)) = (\mathsf{pk}, \mathsf{E}_2(\mathsf{pk}; r_1))$.
  - if $\mathsf{sk}_1 = 1$, then return $(\mathsf{pk}, \mathsf{D}(\mathsf{sk}, \mathsf{ct}_1')) = (\mathsf{pk}, \mathsf{E}_2(\mathsf{pk}; r_1'))$.



Simple construction for recovering the first bit of the input.

- $\mathsf{tk} = \begin{pmatrix} r_1 \\ r_1' \end{pmatrix}$ and $\mathsf{ik} = \begin{pmatrix} \mathsf{ct}_1 \\ \mathsf{ct}_1' \end{pmatrix} = \begin{pmatrix} \mathsf{E}_1((i{=}1, b{=}0); r_1) \\ \mathsf{E}_1((i{=}1, b{=}1); r_1') \end{pmatrix}$
- $\mathsf{F}(\mathsf{ik}, \mathsf{sk})$: let $\mathsf{pk} = \mathsf{G}(\mathsf{sk})$.
  - if $\mathsf{sk}_1 = 0$, then return $(\mathsf{pk}, \mathsf{D}(\mathsf{sk}, \mathsf{ct}_1)) = (\mathsf{pk}, \mathsf{E}_2(\mathsf{pk}; r_1))$.
  - if $\mathsf{sk}_1 = 1$, then return $(\mathsf{pk}, \mathsf{D}(\mathsf{sk}, \mathsf{ct}_1')) = (\mathsf{pk}, \mathsf{E}_2(\mathsf{pk}; r_1'))$.
- F^{-1}: Check for a match:

$$\begin{pmatrix} \mathsf{E}_2(\mathsf{pk}; r_1) \\ \mathsf{E}_2(\mathsf{pk}; r_1') \end{pmatrix}$$

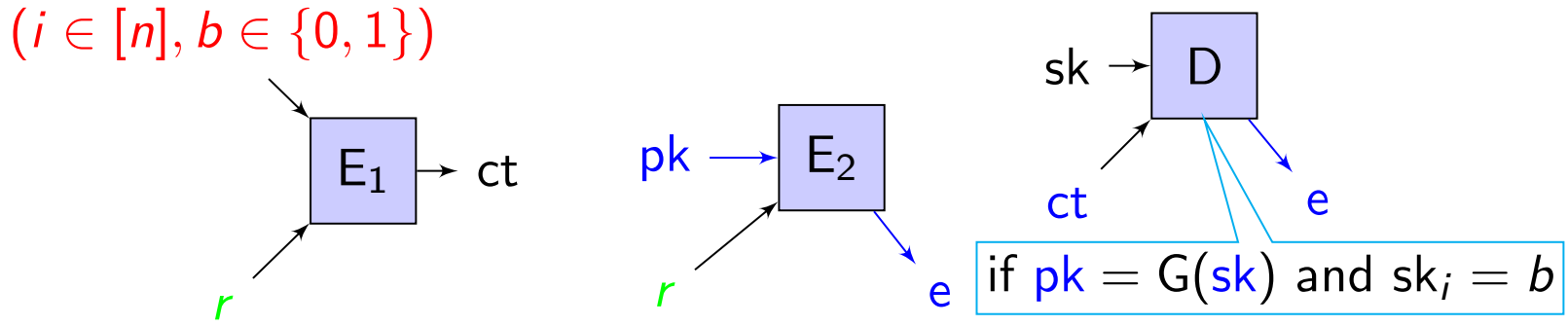


Simple construction for recovering the first bit of the input.

- $\mathsf{tk} = \begin{pmatrix} r_1 \\ r_1' \end{pmatrix}$ and $\mathsf{ik} = \begin{pmatrix} \mathsf{ct}_1 \\ \mathsf{ct}_1' \end{pmatrix} = \begin{pmatrix} \mathsf{E}_1((i{=}1, b{=}0); r_1) \\ \mathsf{E}_1((i{=}1, b{=}1); r_1') \end{pmatrix}$
- $\mathsf{F}(\mathsf{ik}, \mathsf{sk})$: let $\mathsf{pk} = \mathsf{G}(\mathsf{sk})$.
  - if $\mathsf{sk}_1 = 0$, then return $(\mathsf{pk}, \mathsf{D}(\mathsf{sk}, \mathsf{ct}_1)) = (\mathsf{pk}, \mathsf{E}_2(\mathsf{pk}; r_1))$.
  - if $\mathsf{sk}_1 = 1$, then return $(\mathsf{pk}, \mathsf{D}(\mathsf{sk}, \mathsf{ct}_1')) = (\mathsf{pk}, \mathsf{E}_2(\mathsf{pk}; r_1'))$.
- F^{-1}: Check for a match:

$$\begin{pmatrix} \mathsf{E}_2(\mathsf{pk}; r_1) \\ \mathsf{E}_2(\mathsf{pk}; r_1') \end{pmatrix}$$

- Can recover sk_1 with probability $1/2$. This can be boosted via repetition.

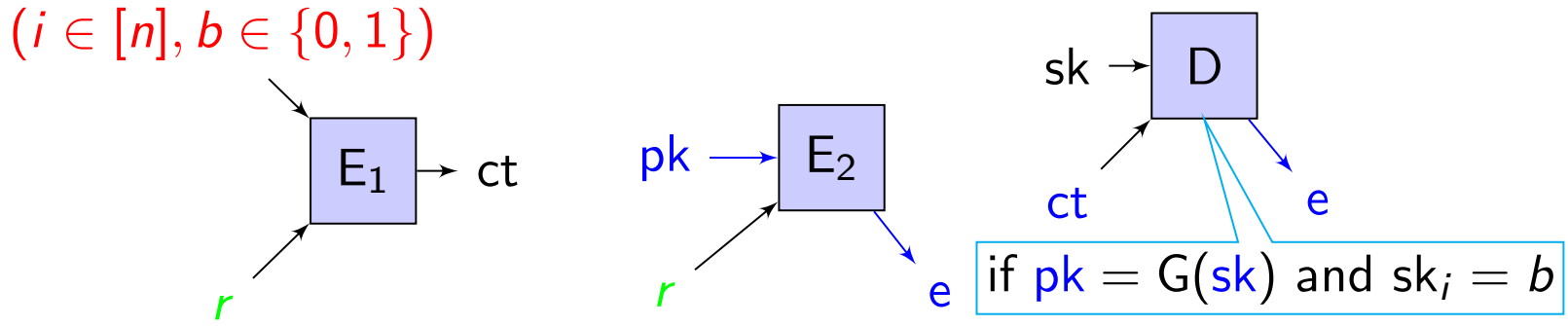


Simple construction for recovering the first bit of the input.

- $\mathsf{tk} = \begin{pmatrix} r_1 \\ r_1' \end{pmatrix}$ and $\mathsf{ik} = \begin{pmatrix} \mathsf{ct}_1 \\ \mathsf{ct}_1' \end{pmatrix} = \begin{pmatrix} \mathsf{E}_1((i{=}1, b{=}0); r_1) \\ \mathsf{E}_1((i{=}1, b{=}1); r_1') \end{pmatrix}$
- $\mathsf{F}(\mathsf{ik}, \mathsf{sk})$: let $\mathsf{pk} = \mathsf{G}(\mathsf{sk})$.
  - if $\mathsf{sk}_1 = 0$, then return $(\mathsf{pk}, \mathsf{D}(\mathsf{sk}, \mathsf{ct}_1)) = (\mathsf{pk}, \mathsf{E}_2(\mathsf{pk}; r_1))$.
  - if $\mathsf{sk}_1 = 1$, then return $(\mathsf{pk}, \mathsf{D}(\mathsf{sk}, \mathsf{ct}_1')) = (\mathsf{pk}, \mathsf{E}_2(\mathsf{pk}; r_1'))$.
- F^{-1}: Check for a match:

$$\begin{pmatrix} \mathsf{E}_2(\mathsf{pk}; r_1) \\ \mathsf{E}_2(\mathsf{pk}; r_1') \end{pmatrix}$$

- Can recover sk_1 with probability $1/2$. This can be boosted via repetition.
- Not clear how to prove security!



Simple construction for recovering the first bit of the input.

- $\mathsf{tk} = \begin{pmatrix} r_1 \\ r_1' \end{pmatrix}$ and $\mathsf{ik} = \begin{pmatrix} \mathsf{ct}_1 \\ \mathsf{ct}_1' \end{pmatrix} = \begin{pmatrix} \mathsf{E}_1((i{=}1, b{=}0); r_1) \\ \mathsf{E}_1((i{=}1, b{=}1); r_1') \end{pmatrix}$
- $\mathsf{F}(\mathsf{ik}, \mathsf{sk})$: let $\mathsf{pk} = \mathsf{G}(\mathsf{sk})$.
  - if $\mathsf{sk}_1 = 0$, then return $(\mathsf{pk}, \mathsf{D}(\mathsf{sk}, \mathsf{ct}_1)) = (\mathsf{pk}, \mathsf{E}_2(\mathsf{pk}; r_1))$.
  - if $\mathsf{sk}_1 = 1$, then return $(\mathsf{pk}, \mathsf{D}(\mathsf{sk}, \mathsf{ct}_1')) = (\mathsf{pk}, \mathsf{E}_2(\mathsf{pk}; r_1'))$.
- F^{-1}: Check for a match:

$$\begin{pmatrix} \mathsf{E}_2(\mathsf{pk}; r_1) \\ \mathsf{E}_2(\mathsf{pk}; r_1') \end{pmatrix}$$

- Can recover sk_1 with probability $1/2$. This can be boosted via repetition.
- Not clear how to prove security!
  - Fix: Put a random bit in the place you cannot apply D.

Recovering the First Bit

- $\mathsf{Gen}(1^\lambda)$: $\mathsf{tk} = \begin{pmatrix} r_1 \\ r_1' \end{pmatrix}$ and

$$\mathsf{ik} = \begin{pmatrix} \mathsf{ct}_1 \\ \mathsf{ct}_1' \end{pmatrix} = \begin{pmatrix} \mathsf{E}_1((i{=}1, b{=}0); r_1) \\ \mathsf{E}_1((i{=}1, b{=}1); r_1') \end{pmatrix}$$

- $\mathsf{F}(\mathsf{ik}, \mathsf{sk} || b_1)$: let $\mathsf{pk} = \mathsf{G}(\mathsf{sk})$. Then:
  - if $\mathsf{sk}_1 = 0$ then $\mathsf{M}_1 := \begin{pmatrix} \mathsf{D}(\mathsf{sk}, \mathsf{ct}_1) \\ \mathsf{b}_1 \end{pmatrix}$
  - if $\mathsf{sk}_1 = 1$ then $\mathsf{M}_1 := \begin{pmatrix} \mathsf{b}_1 \\ \mathsf{D}(\mathsf{sk}, \mathsf{ct}_1') \end{pmatrix}$

  Return $\mathsf{Y} = (\mathsf{pk}, \mathsf{M}_1)$.
- $\mathsf{F}^{-1}(\mathsf{tk}, \mathsf{Y})$:

$$\mathsf{M}_1' = \begin{pmatrix} \mathsf{E}_2(\mathsf{pk}, (1, 0); r_1) \\ \mathsf{E}_2(\mathsf{pk}, (1, 1); r_1') \end{pmatrix}$$

Summary and Future Work

Summary

- A Construction of TDFs from CDH.

Future Work

- Extended forms of TDFs from CDH (e.g., lossy trapdoor functions)
- Trapdoor Permutations from CDH/DDH?