LEAKAGE-RESILIENT PUBLIC-KEY ENCRYPTION FROM OBFUSCATION

Dana Dachman-Soled, S. Dov Gordon, Feng-Hao Lui, **Adam, O'Neill**, and Hong-Sheng Zhou

OUTLINE OF TALK

Leakage Models for PKE —

- Bounded, Continual, and Continual w/ Leakage on Key Update

Results in Continual Model: A Generic Compiler to Achieve Leakage on Key Update

Results in Bounded Model: A New Approach to Optimal Leakage Rate

Conclusion and Open Problems

OUTLINE OF TALK

Leakage Models for PKE

- Bounded, Continual, a
  Update

Uses indistinguishability obfuscation [BGIRSVY'01,GGHRSW'13] and techniques from "deniable encryption" [SW'14].

Results in Continual Model: A Generic Compiler to Achieve Leakage on Key Update

Results in Bounded Model: A New Approach to Optimal Leakage Rate

Conclusion and Open Problems

OUTLINE OF TALK

Leakage Models for PKE —

Bounded, Continual, and Continual w/ Leakage on Key Update

Results in Continual Model: A Generic Compiler to Achieve Leakage on Key Update

Results in Bounded Model: A New Approach to Optimal Leakage Rate

Conclusion and Open Problems

OUTLINE OF TALK

Leakage Models for PKE —

Bounded, Continual, an
Update

Results in Continual Mo
Leakage on Key Update

Uses indistinguishability obfuscation [BGIRSVY'01,GGHRSW'13] and "punctured programming" [SW'14].

Results in Bounded Model: A New Approach to Optimal Leakage Rate

Conclusion and Open Problems

OUTLINE OF TALK

Leakage Models for PKE —

Bounded, Continual, and Continual w/ Leakage on Key Update

Results in Continual Model: A Generic Compiler to Achieve Leakage on Key Update

Results in Bounded Model: A New Approach to Optimal Leakage Rate

Conclusion and Open Problems

OUTLINE OF TALK

Leakage Models for PKE —

- Bounded, Continual, and Continual w/ Leakage on Key Update

Results in Continual Model: A Generic Compiler to Achieve Leakage on Key Update

Results in Bounded Model: A New Approach to Optimal Leakage Rate

Conclusion and Open Problems

BOUNDED LEAKAGE FOR PKE [AGV'09]

Fix a public-key encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$.

$(pk, sk) \leftarrow_{\$} \mathcal{K}$

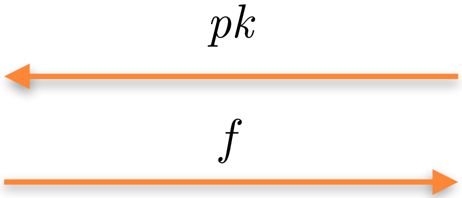
Adversary

Challenger

BOUNDED LEAKAGE FOR PKE [AGV'09]

Fix a public-key encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$.

$(pk, sk) \leftarrow\!\!\$\, \mathcal{K}$



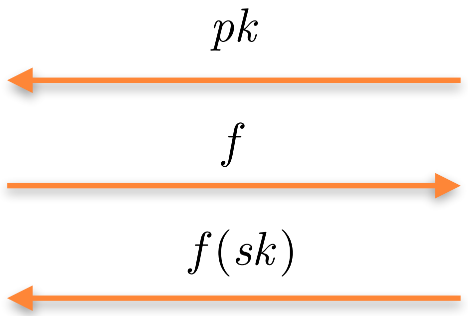
Adversary

Challenger

BOUNDED LEAKAGE FOR PKE [AGV'09]

Fix a public-key encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$.

$(pk, sk) \leftarrow_{\$} \mathcal{K}$

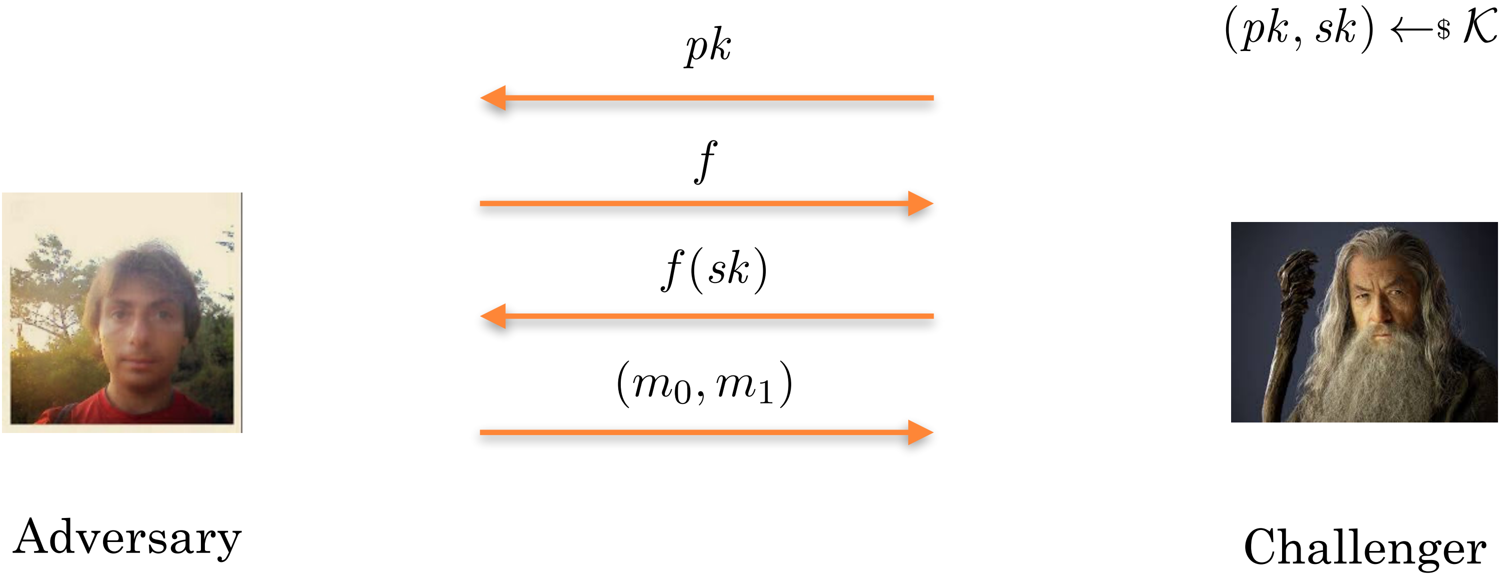


Adversary

Challenger

BOUNDED LEAKAGE FOR PKE [AGV'09]

Fix a public-key encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$.



$(pk, sk) \leftarrow_{\$} \mathcal{K}$

pk

f

$f(sk)$

(m_0, m_1)

Adversary

Challenger

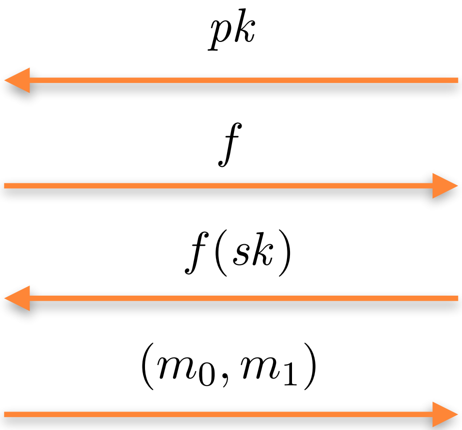
BOUNDED LEAKAGE FOR PKE [AGV'09]

Fix a public-key encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$.

$$(pk, sk) \leftarrow\$ \mathcal{K}$$

$$b \leftarrow\$ \{0,1\}$$

$$c \leftarrow\$ \mathcal{E}(pk, m_b)$$



Adversary

Challenger

BOUNDED LEAKAGE FOR PKE [AGV'09]

Fix a public-key encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$.

Adversary

$$\xleftarrow{\quad pk \quad}$$

$$\xrightarrow{\quad f \quad}$$

$$\xleftarrow{\quad f(sk) \quad}$$

$$\xrightarrow{\quad (m_0, m_1) \quad}$$

$$\xleftarrow{\quad c \quad}$$

$(pk, sk) \leftarrow\!\!\$\ \mathcal{K}$

$b \leftarrow\!\!\$\ \{0, 1\}$

$c \leftarrow\!\!\$\ \mathcal{E}(pk, m_b)$

Challenger

BOUNDED LEAKAGE FOR PKE [AGV'09]

Fix a public-key encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$.

Adversary

$$\xleftarrow{\quad pk \quad}$$

$$\xrightarrow{\quad f \quad}$$

$$\xleftarrow{\quad f(sk) \quad}$$

$$\xrightarrow{\quad (m_0, m_1) \quad}$$

$$\xleftarrow{\quad c \quad}$$

$$\xrightarrow{\quad b' \quad}$$

$(pk, sk) \leftarrow_{\$} \mathcal{K}$

$b \leftarrow_{\$} \{0, 1\}$

$c \leftarrow_{\$} \mathcal{E}(pk, m_b)$

Challenger

BOUNDED LEAKAGE FOR PKE [AGV'09]

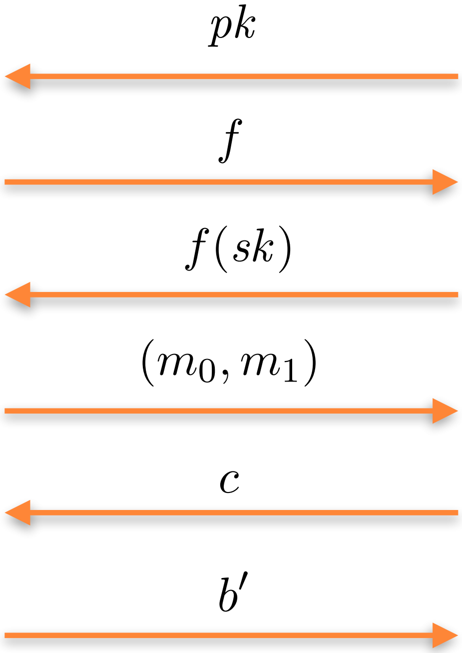
Fix a public-key encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$.

$(pk, sk) \leftarrow_{\$} \mathcal{K}$

$b \leftarrow_{\$} \{0,1\}$

$c \leftarrow_{\$} \mathcal{E}(pk, m_b)$

Adversary



$\xleftarrow{\quad pk \quad}$

$\xrightarrow{\quad f \quad}$

$\xleftarrow{\quad f(sk) \quad}$

$\xrightarrow{\quad (m_0, m_1) \quad}$

$\xleftarrow{\quad c \quad}$

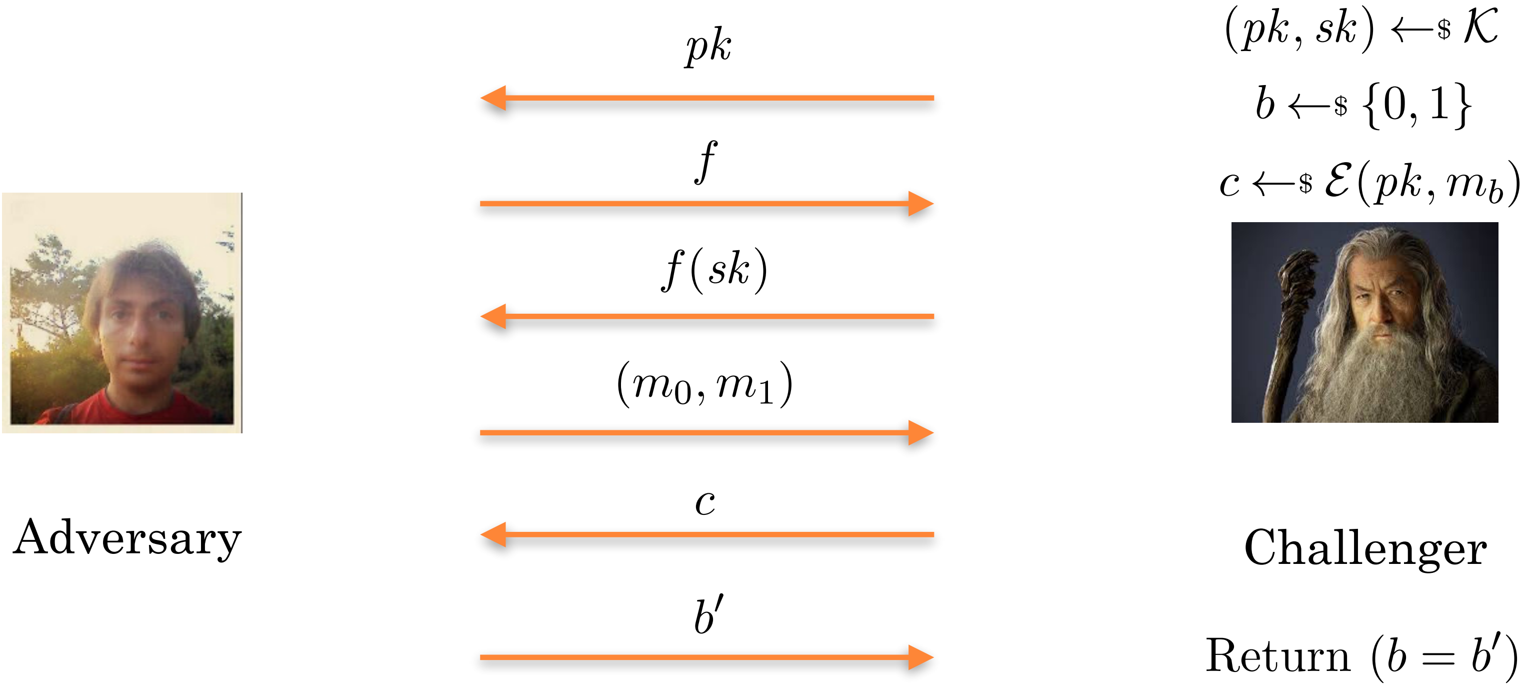
$\xrightarrow{\quad b' \quad}$

Challenger

Return $(b = b')$

BOUNDED LEAKAGE FOR PKE [AGV'09]

Fix a public-key encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$.



$(pk, sk) \leftarrow_{\$} \mathcal{K}$

$b \leftarrow_{\$} \{0,1\}$

$c \leftarrow_{\$} \mathcal{E}(pk, m_b)$

$\xleftarrow{\quad pk \quad}$

$\xrightarrow{\quad f \quad}$

$\xleftarrow{\quad f(sk) \quad}$

$\xrightarrow{\quad (m_0, m_1) \quad}$

$\xleftarrow{\quad c \quad}$

$\xrightarrow{\quad b' \quad}$

Adversary

Challenger

Return $(b = b')$

Require $2 \cdot \Pr\left[\, b = b' \,\right] - 1$ is negligible.

BOUNDED LEAKAGE FOR PKE [AGV'09]

Fix a public-key encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$.

Adversary

$\xleftarrow{\quad pk \quad}$

$\xrightarrow{\quad f \quad}$

Must be bounded length!

$\xleftarrow{\quad f(sk) \quad}$

$\xrightarrow{\quad (m_0, m_1) \quad}$

$\xleftarrow{\quad c \quad}$

$\xrightarrow{\quad b' \quad}$

Challenger

Return $(b = b')$

Require $2 \cdot \Pr\left[\, b = b' \,\right] - 1$ is negligible.

CONTINUAL LEAKAGE FOR PKE [BKKV'10,DHLW'10]

Fix a public-key encryption scheme "with key update" $(\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{U})$
i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow_{\$} \mathcal{U}(sk)$.

$(pk, sk_0) \leftarrow_{\$} \mathcal{K}$

Adversary

Challenger

CONTINUAL LEAKAGE FOR PKE [BKKV'10,DHLW'10]

Fix a public-key encryption scheme "with key update" $(\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{U})$ i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow_{\$} \mathcal{U}(sk)$.

$(pk, sk_0) \leftarrow_{\$} \mathcal{K}$

pk

f

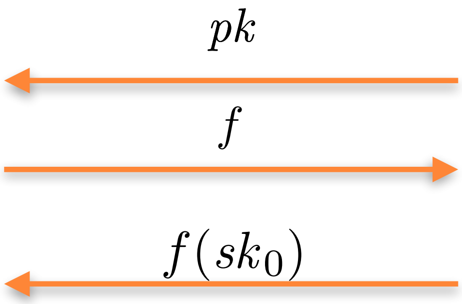
Adversary

Challenger

CONTINUAL LEAKAGE FOR PKE [BKKV'10,DHLW'10]

Fix a public-key encryption scheme "with key update" $(\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{U})$
i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow_{\$} \mathcal{U}(sk)$.

$(pk, sk_0) \leftarrow_{\$} \mathcal{K}$



Adversary

Challenger

CONTINUAL LEAKAGE FOR PKE [BKKV'10,DHLW'10]

Fix a public-key encryption scheme "with key update" $(\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{U})$
i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow_{\$} \mathcal{U}(sk)$.

$sk_i \leftarrow_{\$} \mathcal{U}(sk_{i-1})$

pk

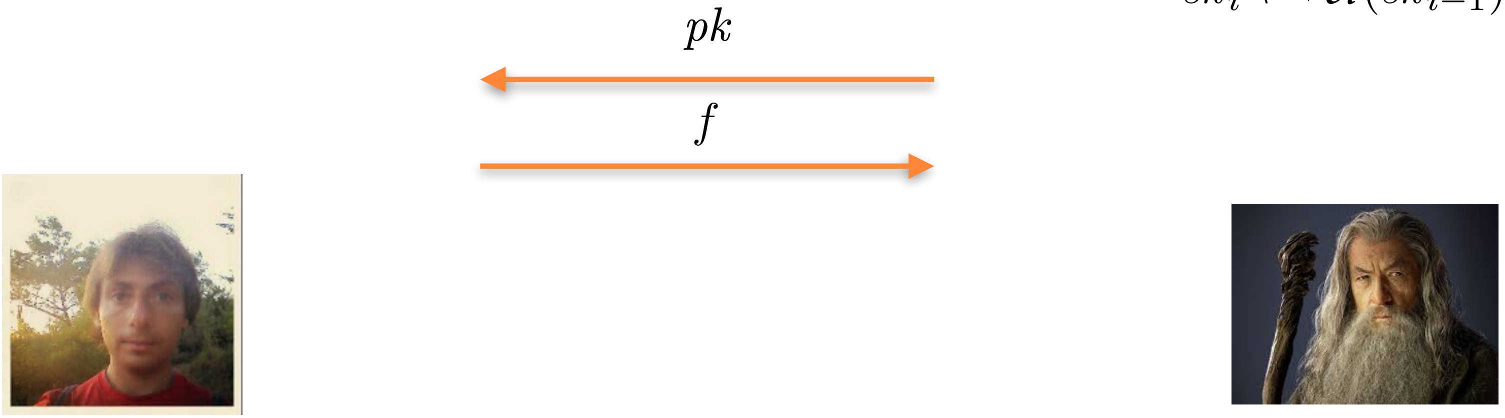
Adversary

Challenger

CONTINUAL LEAKAGE FOR PKE [BKKV'10,DHLW'10]

Fix a public-key encryption scheme "with key update" $(\mathcal{K},\mathcal{E},\mathcal{D},\mathcal{U})$ i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow_{\$} \mathcal{U}(sk)$.

$sk_i \leftarrow_{\$} \mathcal{U}(sk_{i-1})$



pk

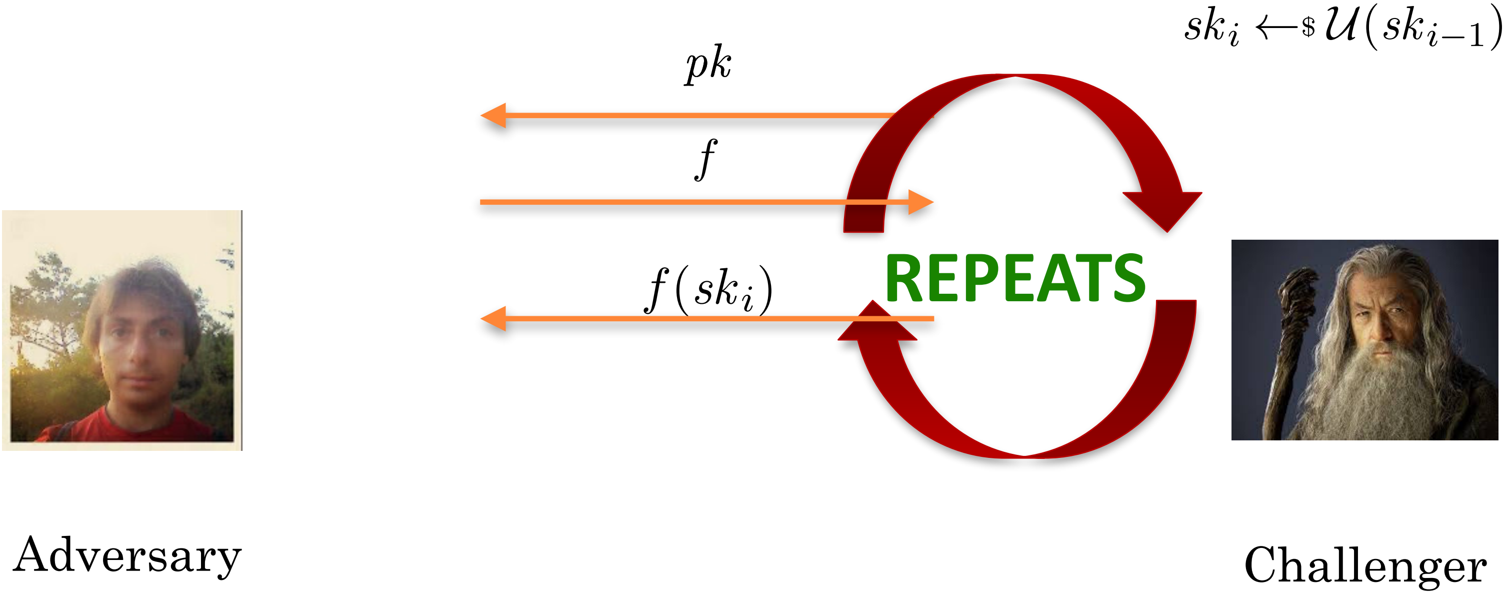
f

Adversary

Challenger

CONTINUAL LEAKAGE FOR PKE [BKKV'10,DHLW'10]

Fix a public-key encryption scheme "with key update" $(\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{U})$
i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow_{\$} \mathcal{U}(sk)$.



$sk_i \leftarrow_{\$} \mathcal{U}(sk_{i-1})$

pk

f

$f(sk_i)$

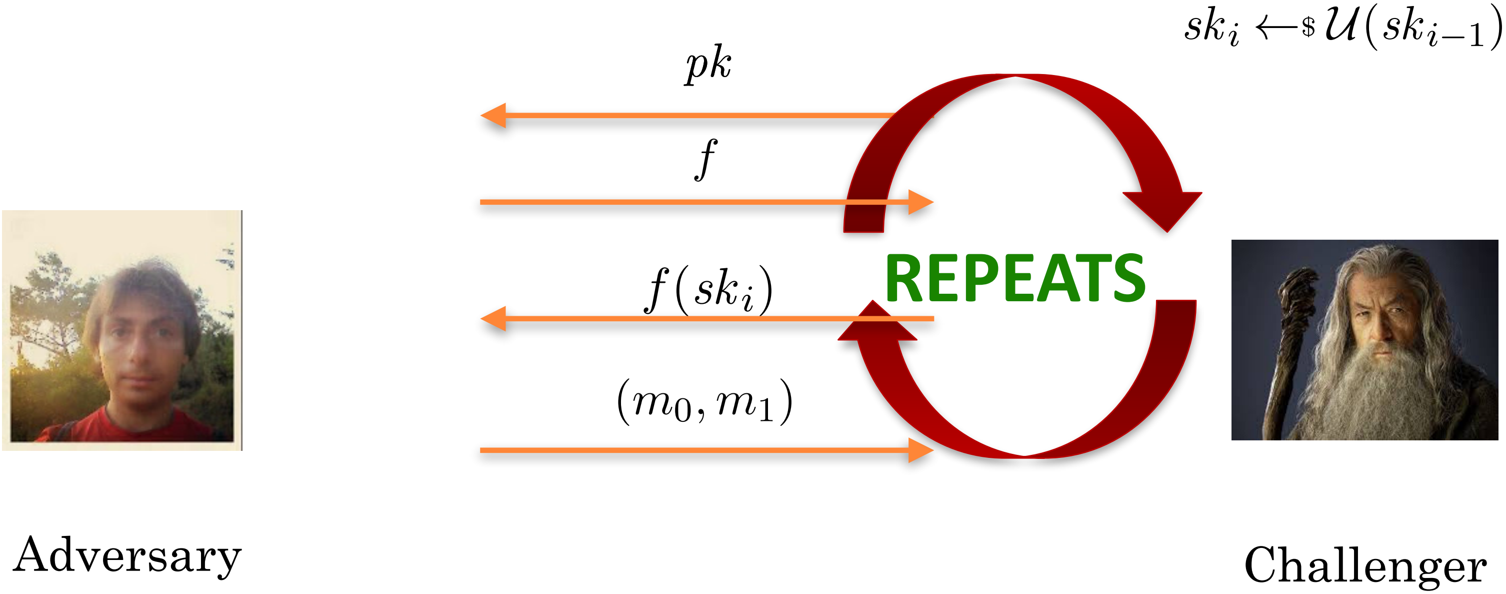
REPEATS

Adversary

Challenger

CONTINUAL LEAKAGE FOR PKE [BKKV'10,DHLW'10]

Fix a public-key encryption scheme "with key update" $(\mathcal{K},\mathcal{E},\mathcal{D},\mathcal{U})$
i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow_{\$} \mathcal{U}(sk)$.



$sk_i \leftarrow_{\$} \mathcal{U}(sk_{i-1})$

pk

f

$f(sk_i)$

(m_0, m_1)

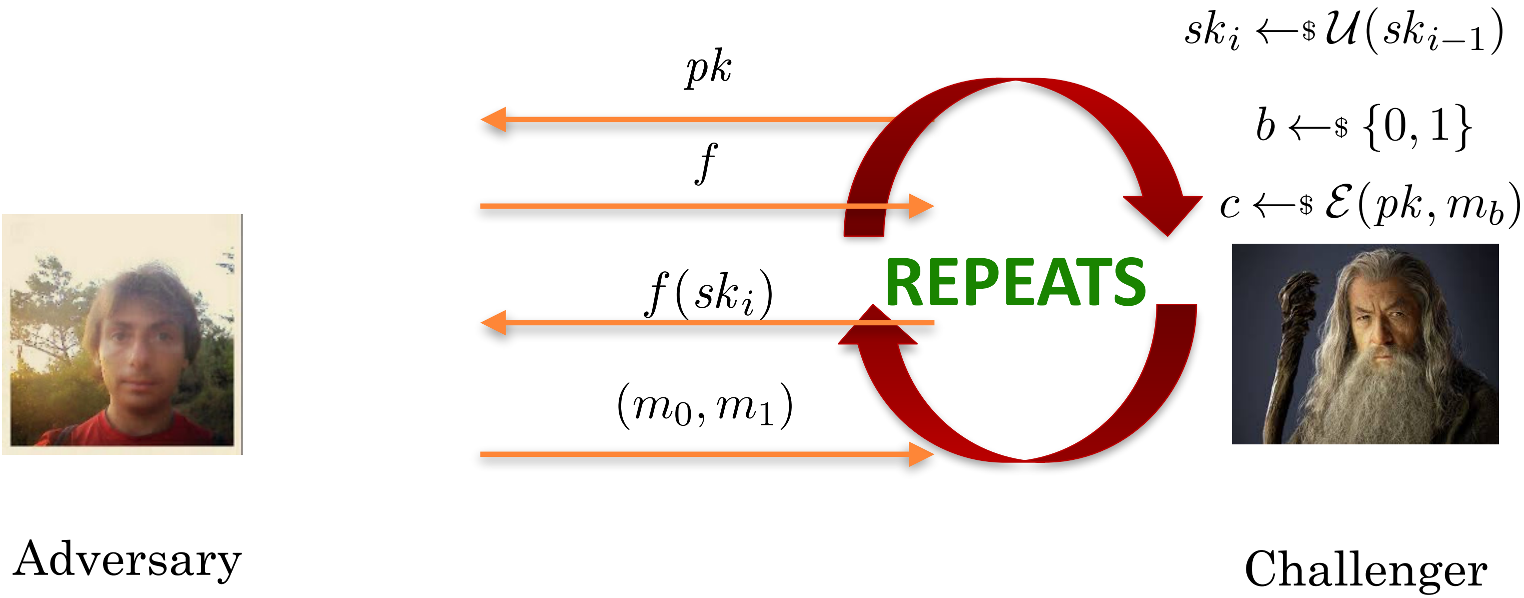
REPEATS

Adversary

Challenger

CONTINUAL LEAKAGE FOR PKE [BKKV'10,DHLW'10]

Fix a public-key encryption scheme "with key update" $(\mathcal{K},\mathcal{E},\mathcal{D},\mathcal{U})$
i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow_{\$} \mathcal{U}(sk)$.



$sk_i \leftarrow_{\$} \mathcal{U}(sk_{i-1})$

pk

$b \leftarrow_{\$} \{0,1\}$

f

$c \leftarrow_{\$} \mathcal{E}(pk, m_b)$

REPEATS

$f(sk_i)$

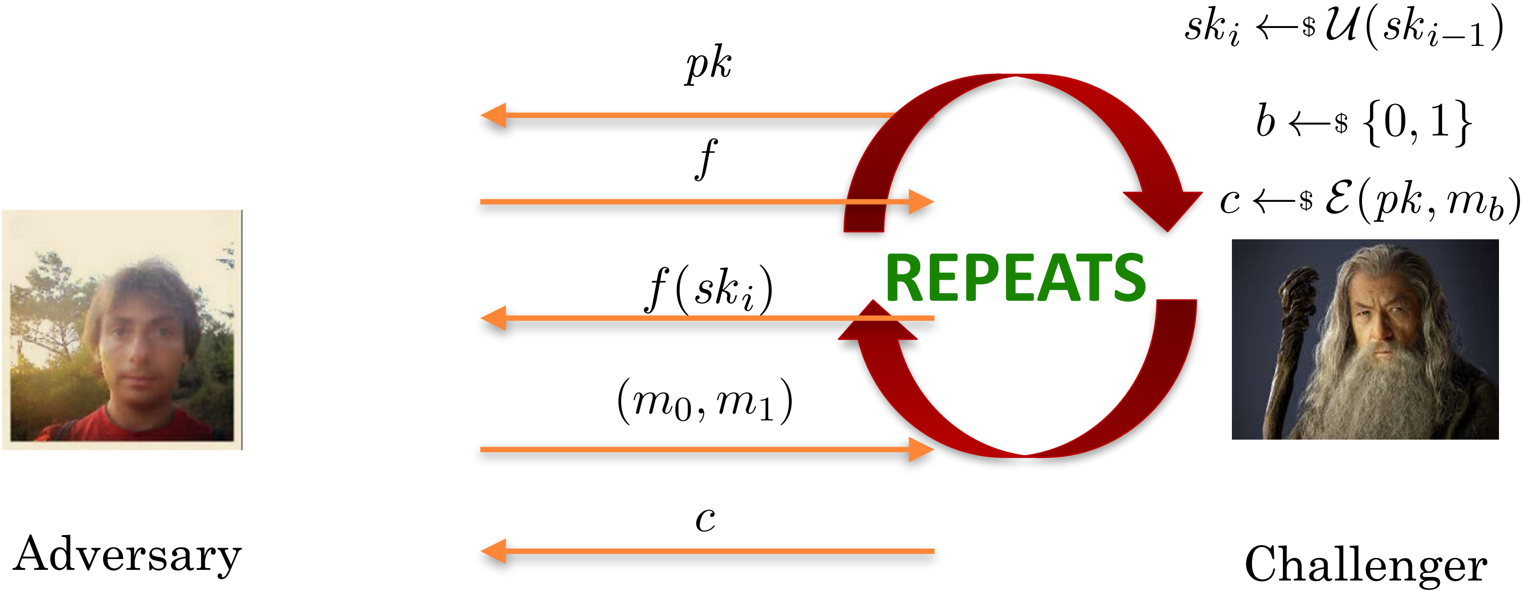
(m_0, m_1)

Adversary

Challenger

Continual leakage for PKE [BKKV'10,DHLW'10]

Fix a public-key encryption scheme "with key update" $(\mathcal{K},\mathcal{E},\mathcal{D},\mathcal{U})$
i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow_{\$} \mathcal{U}(sk)$.



$sk_i \leftarrow_{\$} \mathcal{U}(sk_{i-1})$

pk

$b \leftarrow_{\$} \{0,1\}$

f

$c \leftarrow_{\$} \mathcal{E}(pk, m_b)$

REPEATS

$f(sk_i)$

(m_0, m_1)

c

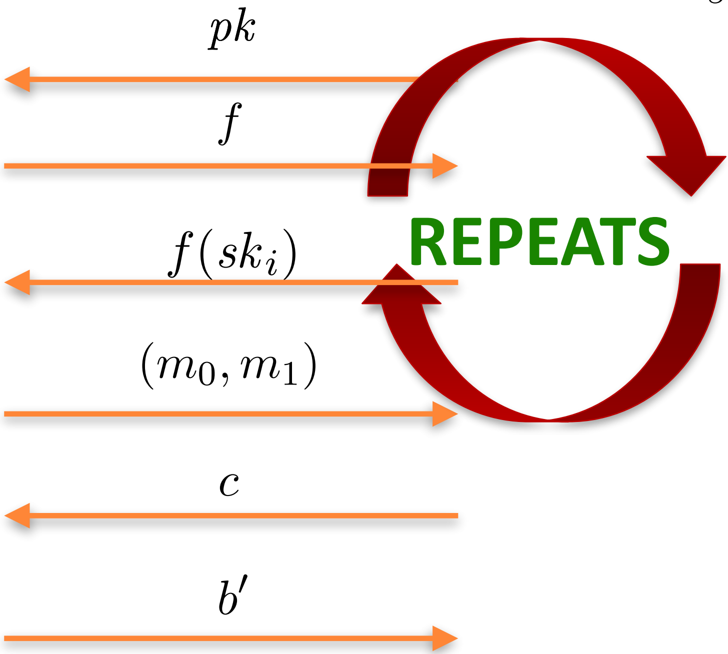
Adversary

Challenger

CONTINUAL LEAKAGE FOR PKE [BKKV'10,DHLW'10]

Fix a public-key encryption scheme "with key update" $(\mathcal{K},\mathcal{E},\mathcal{D},\mathcal{U})$
i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow_{\$} \mathcal{U}(sk)$.

Adversary



pk

f

$f(sk_i)$

(m_0, m_1)

c

b'

$sk_i \leftarrow_{\$} \mathcal{U}(sk_{i-1})$

$b \leftarrow_{\$} \{0,1\}$

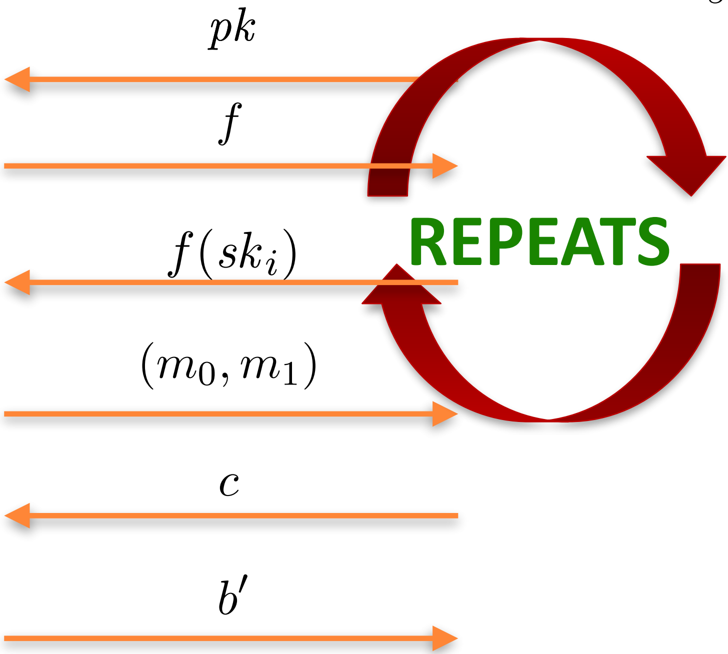
$c \leftarrow_{\$} \mathcal{E}(pk, m_b)$

Challenger

CONTINUAL LEAKAGE FOR PKE [BKKV'10,DHLW'10]

Fix a public-key encryption scheme "with key update" $(\mathcal{K},\mathcal{E},\mathcal{D},\mathcal{U})$
i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow\$\ \mathcal{U}(sk)$.

Adversary



$sk_i \leftarrow\$\ \mathcal{U}(sk_{i-1})$

$b \leftarrow\$\ \{0,1\}$

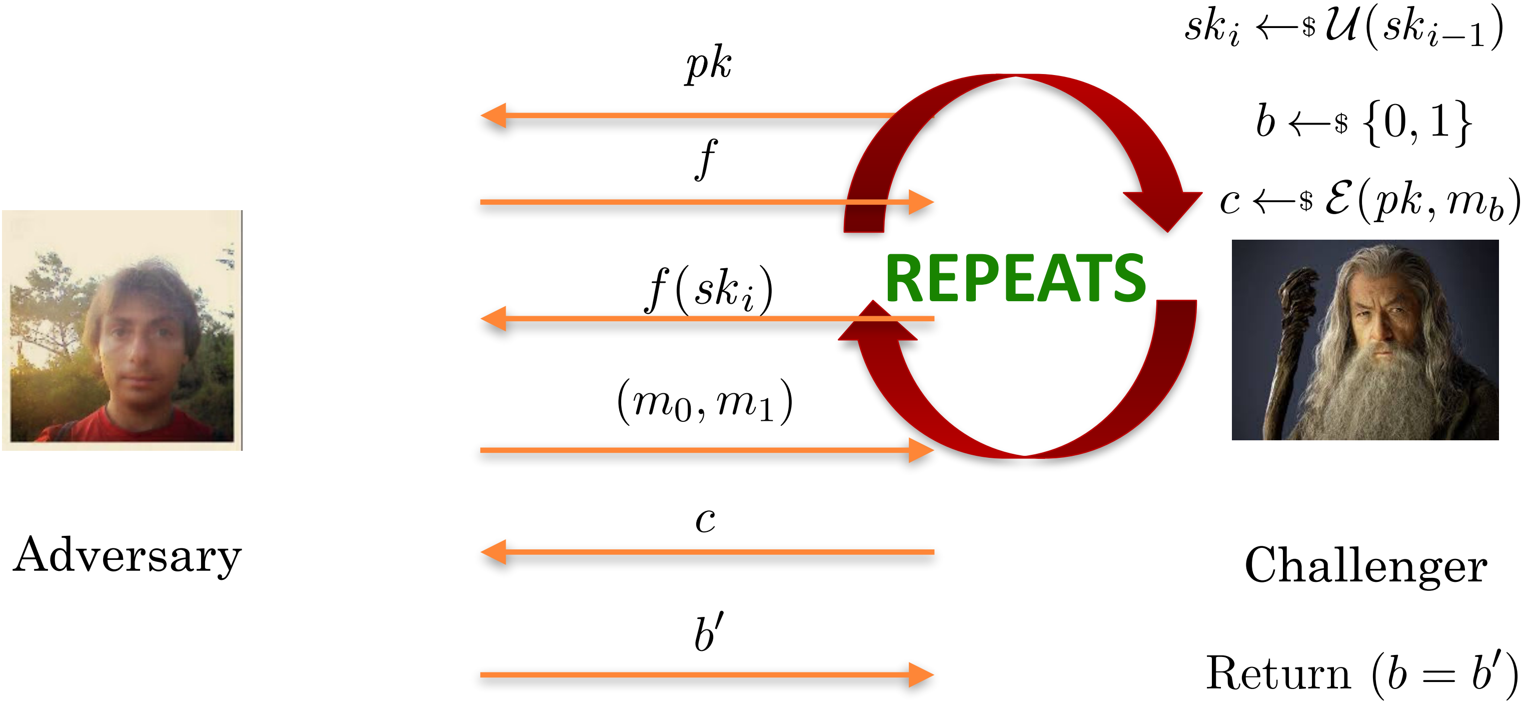
$c \leftarrow\$\ \mathcal{E}(pk, m_b)$

Challenger

Return $(b = b')$

CONTINUAL LEAKAGE FOR PKE [BKKV'10,DHLW'10]

Fix a public-key encryption scheme "with key update" $(\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{U})$
i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow_{\$} \mathcal{U}(sk)$.



$sk_i \leftarrow_{\$} \mathcal{U}(sk_{i-1})$

pk

$b \leftarrow_{\$} \{0,1\}$

f

$c \leftarrow_{\$} \mathcal{E}(pk, m_b)$

REPEATS

$f(sk_i)$

(m_0, m_1)

Adversary

c

Challenger

b'

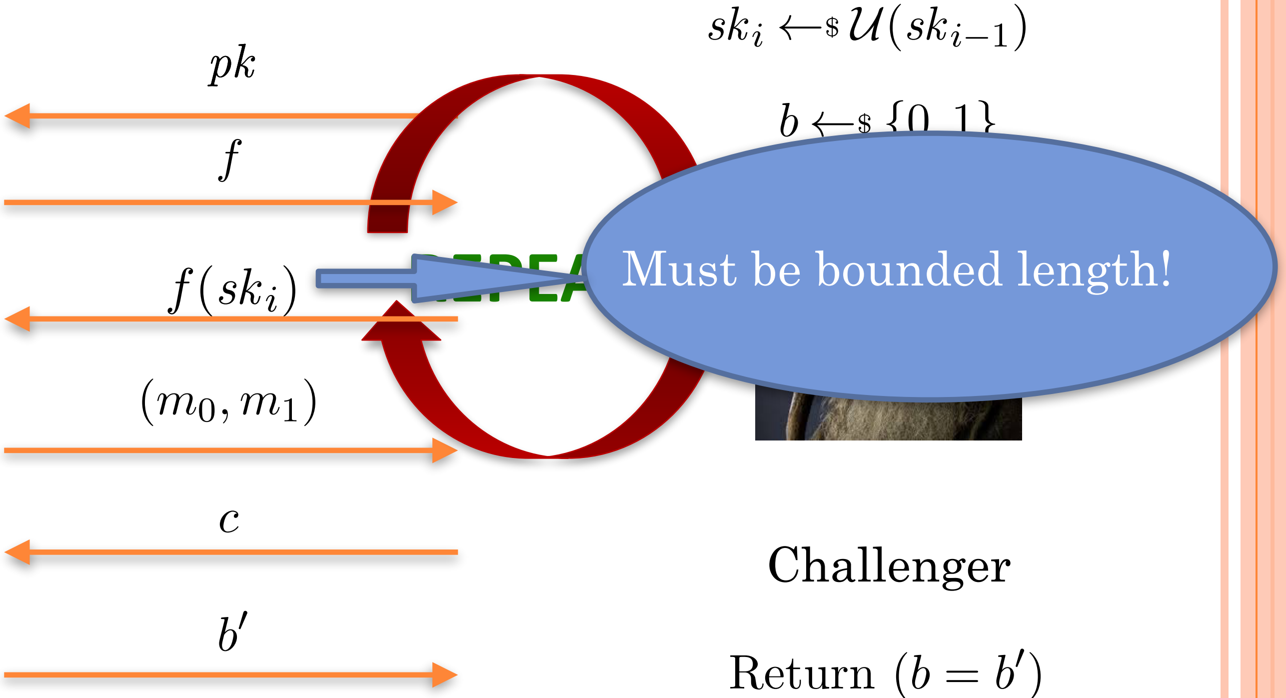
Return $(b = b')$

Require $2 \cdot \Pr\left[\, b = b' \,\right] - 1$ is negligible.

CONTINUAL LEAKAGE FOR PKE [BKKV'10,DHLW'10]

Fix a public-key encryption scheme "with key update" $(\mathcal{K},\mathcal{E},\mathcal{D},\mathcal{U})$
i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow_{\$} \mathcal{U}(sk)$.

Adversary



Must be bounded length!

Challenger

Return $(b = b')$

Require $2 \cdot \Pr\left[\, b = b' \,\right] - 1$ is negligible.

LEAKAGE ON KEY-UPDATE FOR PKE [BKKV'10,LLW'11]

Fix a public-key encryption scheme "with key update" $(\mathcal{K},\mathcal{E},\mathcal{D},\mathcal{U})$
i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow_\$ \mathcal{U}(sk)$.

$(pk, sk_0) \leftarrow_\$ \mathcal{K}$

pk

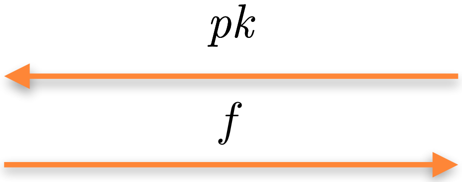
Adversary

Challenger

LEAKAGE ON KEY-UPDATE FOR PKE [BKKV'10,LLW'11]

Fix a public-key encryption scheme "with key update" $(\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{U})$
i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow\!\!{\scriptstyle\$}\, \mathcal{U}(sk)$.

$(pk, sk_0) \leftarrow\!\!{\scriptstyle\$}\, \mathcal{K}$



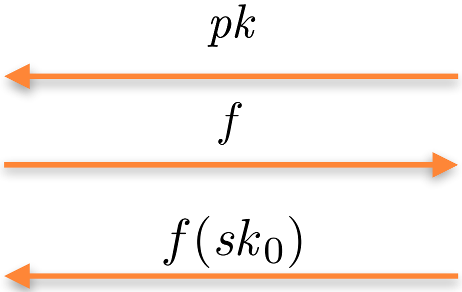
Adversary

Challenger

LEAKAGE ON KEY-UPDATE FOR PKE [BKKV'10,LLW'11]

Fix a public-key encryption scheme "with key update" $(\mathcal{K},\mathcal{E},\mathcal{D},\mathcal{U})$
i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow_{\$} \mathcal{U}(sk)$.

$(pk, sk_0) \leftarrow_{\$} \mathcal{K}$



pk

f

$f(sk_0)$

Adversary

Challenger

LEAKAGE ON KEY-UPDATE FOR PKE [BKKV'10,LLW'11]

Fix a public-key encryption scheme "with key update" $(\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{U})$
i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow\!\!{\scriptstyle\$}\, \mathcal{U}(sk)$.

$sk_i \leftarrow\!\!{\scriptstyle\$}\, \mathcal{U}(sk_{i-1}; r_i)$

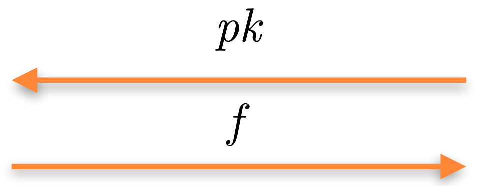
$$\xleftarrow{\quad pk \quad}$$

Adversary

Challenger

LEAKAGE ON KEY-UPDATE FOR PKE [BKKV'10,LLW'11]

Fix a public-key encryption scheme "with key update" $(\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{U})$
i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow\$\, \mathcal{U}(sk)$.

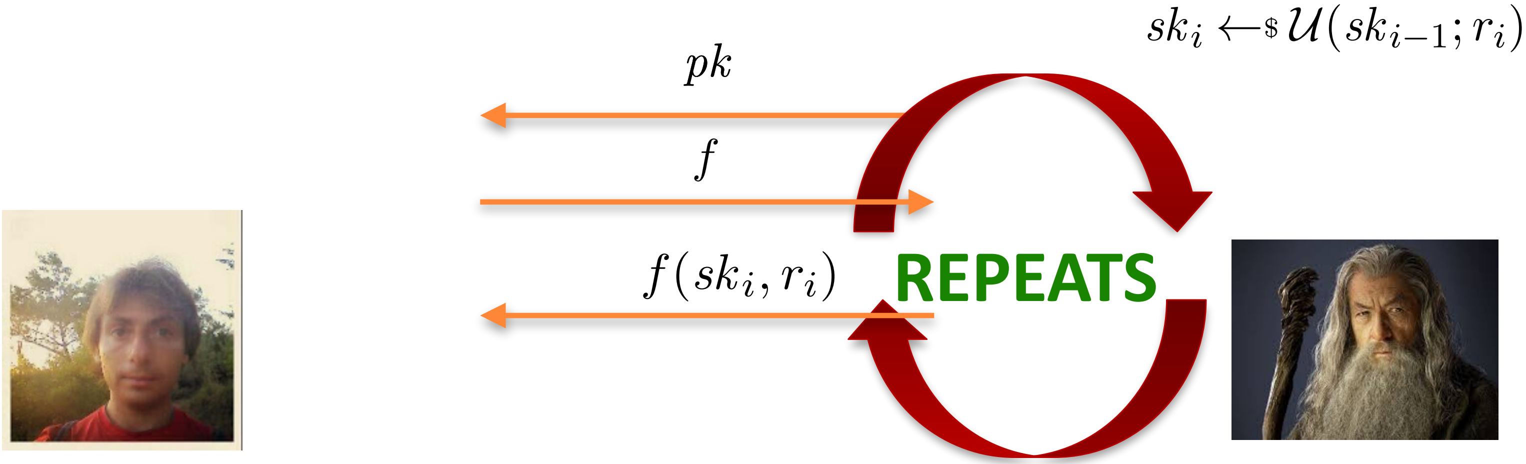


Adversary

Challenger

LEAKAGE ON KEY-UPDATE FOR PKE [BKKV'10,LLW'11]

Fix a public-key encryption scheme "with key update" $(\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{U})$
i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow_{\$} \mathcal{U}(sk)$.

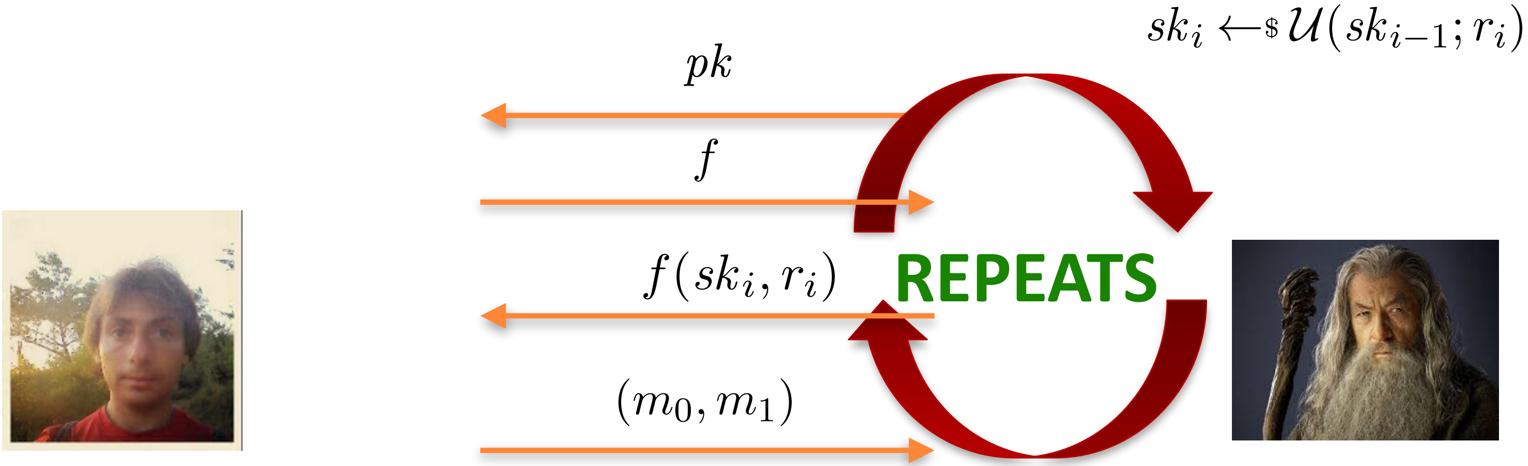


Adversary

Challenger

LEAKAGE ON KEY-UPDATE FOR PKE [BKKV'10,LLW'11]

Fix a public-key encryption scheme "with key update" $(\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{U})$
i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow_{\$} \mathcal{U}(sk)$.



$sk_i \leftarrow_{\$} \mathcal{U}(sk_{i-1}; r_i)$

pk

f

$f(sk_i, r_i)$

REPEATS

(m_0, m_1)

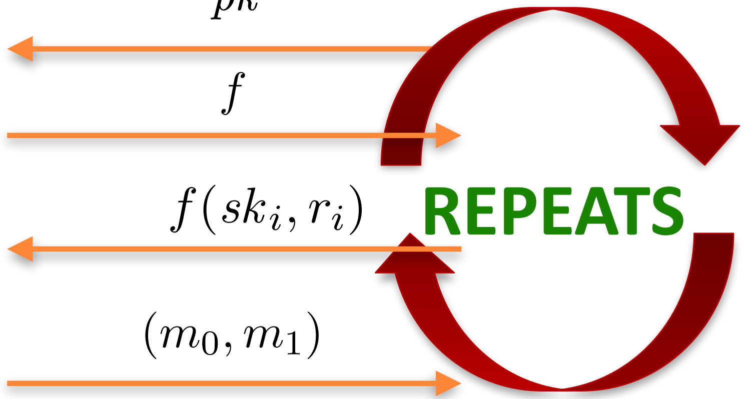
Adversary

Challenger

LEAKAGE ON KEY-UPDATE FOR PKE [BKKV'10,LLW'11]

Fix a public-key encryption scheme "with key update" $(\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{U})$
i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow_{\$} \mathcal{U}(sk)$.

$sk_i \leftarrow_{\$} \mathcal{U}(sk_{i-1}; r_i)$



pk

f

$f(sk_i, r_i)$

REPEATS

(m_0, m_1)

$b \leftarrow_{\$} \{0, 1\}$

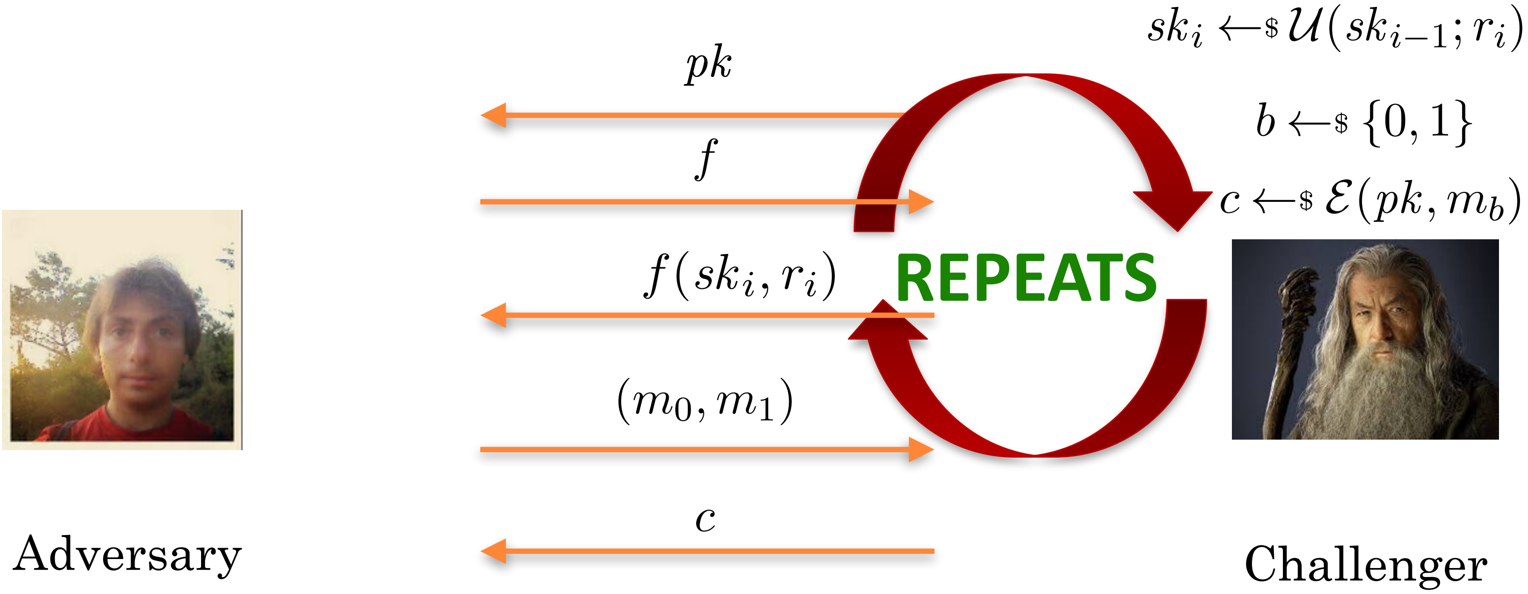
$c \leftarrow_{\$} \mathcal{E}(pk, m_b)$

Adversary

Challenger

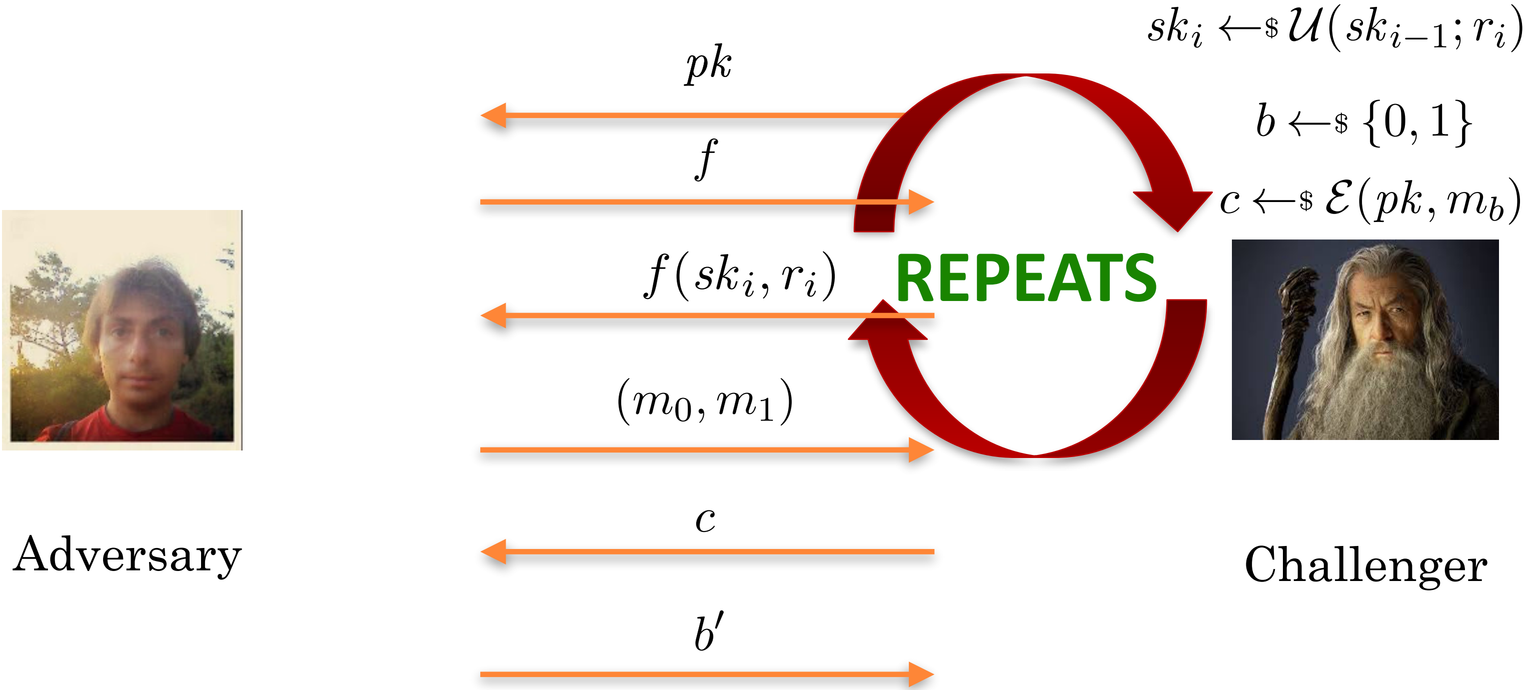
LEAKAGE ON KEY-UPDATE FOR PKE [BKKV'10,LLW'11]

Fix a public-key encryption scheme "with key update" $(\mathcal{K},\mathcal{E},\mathcal{D},\mathcal{U})$
i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow_{\$} \mathcal{U}(sk)$.



LEAKAGE ON KEY-UPDATE FOR PKE [BKKV'10,LLW'11]

Fix a public-key encryption scheme "with key update" $(\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{U})$
i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow_{\$} \mathcal{U}(sk)$.



$sk_i \leftarrow_{\$} \mathcal{U}(sk_{i-1}; r_i)$

pk

f

$f(sk_i, r_i)$

REPEATS

$b \leftarrow_{\$} \{0,1\}$

(m_0, m_1)

$c \leftarrow_{\$} \mathcal{E}(pk, m_b)$

c

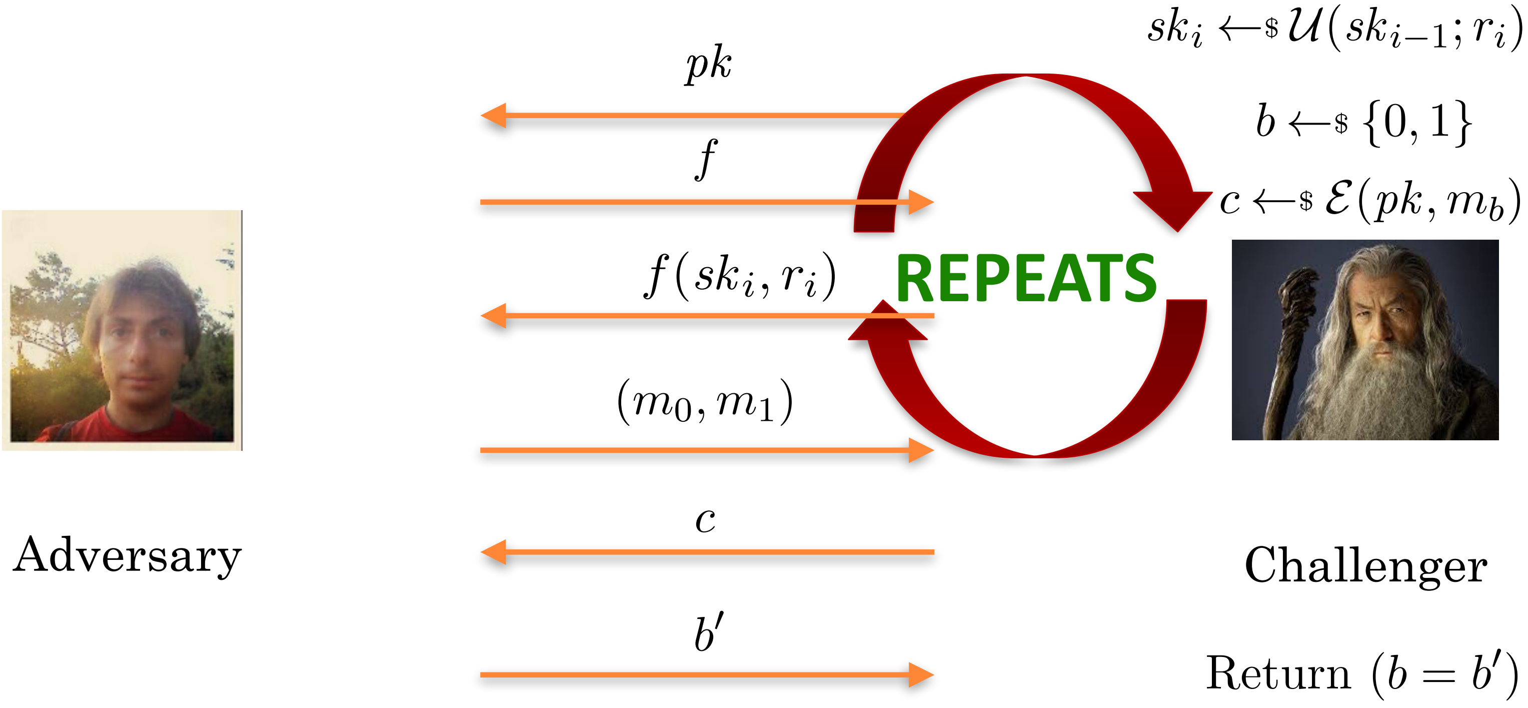
b'

Adversary

Challenger

LEAKAGE ON KEY-UPDATE FOR PKE [BKKV'10,LLW'11]

Fix a public-key encryption scheme "with key update" $(\mathcal{K},\mathcal{E},\mathcal{D},\mathcal{U})$
i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow\$ \mathcal{U}(sk)$.



$sk_i \leftarrow\$ \mathcal{U}(sk_{i-1}; r_i)$

pk

f

$f(sk_i, r_i)$

REPEATS

$b \leftarrow\$ \{0,1\}$

$c \leftarrow\$ \mathcal{E}(pk, m_b)$

(m_0, m_1)

c

b'

Adversary

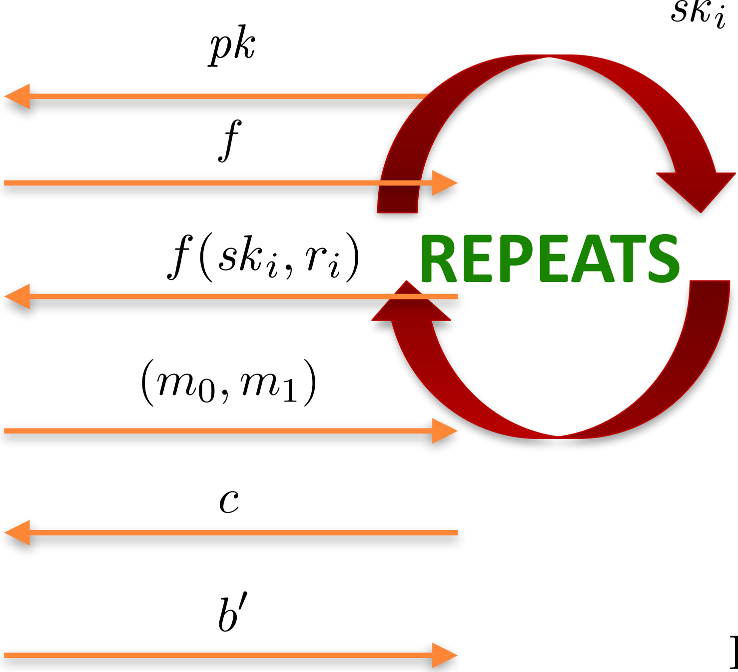
Challenger

Return $(b = b')$

LEAKAGE ON KEY-UPDATE FOR PKE [BKKV'10,LLW'11]

Fix a public-key encryption scheme "with key update" $(\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{U})$
i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow_{\$} \mathcal{U}(sk)$.

Adversary



pk

f

$f(sk_i, r_i)$

REPEATS

(m_0, m_1)

c

b'

$sk_i \leftarrow_{\$} \mathcal{U}(sk_{i-1}; r_i)$

$b \leftarrow_{\$} \{0, 1\}$

$c \leftarrow_{\$} \mathcal{E}(pk, m_b)$

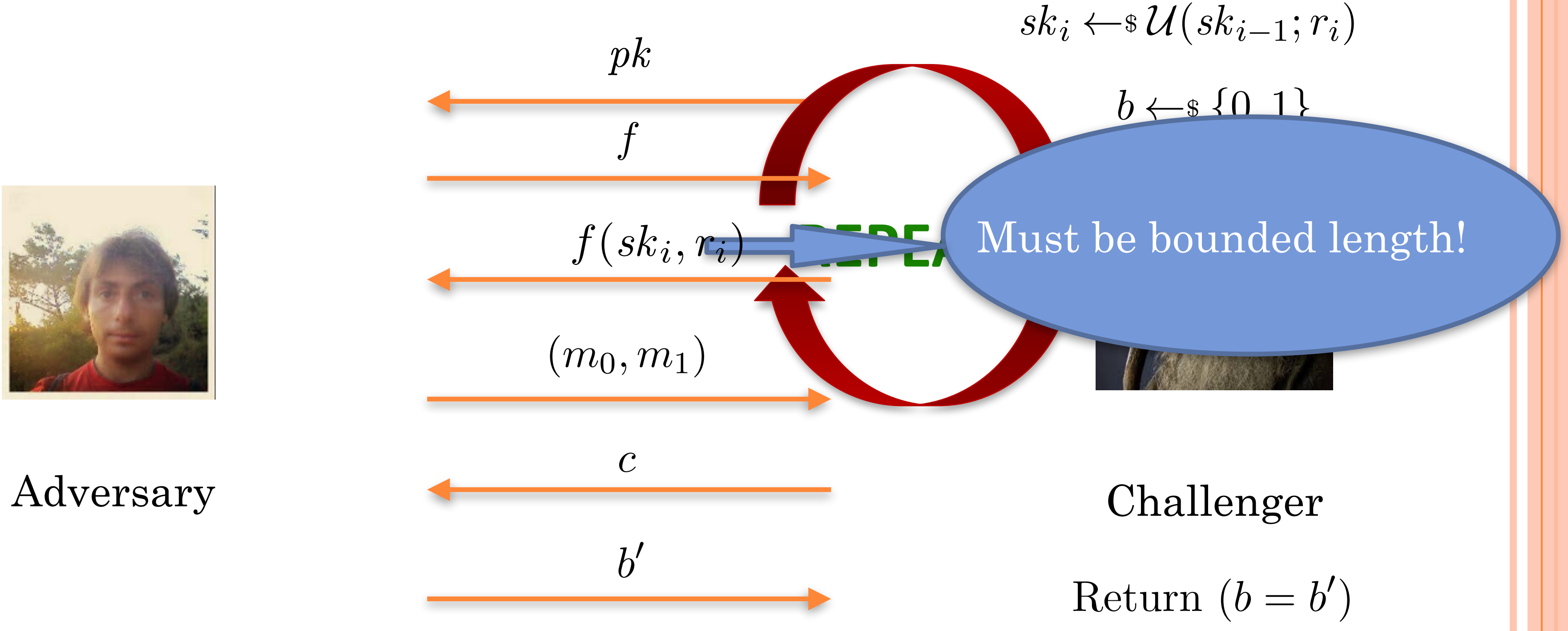
Challenger

Return $(b = b')$

Require $2 \cdot \Pr\left[\, b = b' \,\right] - 1$ is negligible.

LEAKAGE ON KEY-UPDATE FOR PKE [BKKV'10,LLW'11]

Fix a public-key encryption scheme "with key update" $(\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{U})$
i.e. where update algorithm $\mathcal{U}$ computes $sk' \leftarrow_{\$} \mathcal{U}(sk)$.



Require $2 \cdot \Pr\left[\, b = b' \,\right] - 1$ is negligible.

OUTLINE OF TALK

Leakage Models for PKE —

- Bounded, Continual, and Continual w/ Leakage on Key Update

Results in Continual Model: A Generic Compiler to Achieve Leakage on Key Update

Results in Bounded Model: A New Approach to Optimal Leakage Rate

Conclusion and Open Problems

COMPILER INTUITION

Suppose we start with a PKE scheme secure in the continual leakage model.

COMPILER INTUITION

Suppose we start with a PKE scheme secure in the continual leakage model.

COMPILER INTUITION

Suppose we start with a PKE scheme secure in the continual leakage model.

For leakage on key updates, simulator needs to be able to provide "honest-looking" output of function on the update randomness that it doesn't know.

COMPILER INTUITION

Suppose we start with a PKE scheme secure in the continual leakage model.

For leakage on key updates, simulator needs to be able to provide "honest-looking" output of function on the update randomness that it doesn't know.

COMPILER INTUITION

Suppose we start with a PKE scheme secure in the continual leakage model.

For leakage on key updates, simulator needs to be able to provide "honest-looking" output of function on the update randomness that it doesn't know.

Main idea: Make it possible to publicly compute some "honest-looking" update randomness.

COMPILER INTUITION

Suppose we start with a PKE scheme secure in the continual leakage model.

For leakage on key updates, simulator needs to be able to provide "honest-looking" output of function on the update randomness that it doesn't know.

Main idea: Make it possible to publicly compute some "honest-looking" update randomness.

COMPILER INTUITION

Suppose we start with a PKE scheme secure in the continual leakage model.

For leakage on key updates, simulator needs to be able to provide “honest-looking” output of function on the update randomness that it doesn’t know.

Main idea: Make it possible to publicly compute some “honest-looking” update randomness.

This is very similar to deniable encryption as recently achieved by Sahai and Waters [SW’14].

THE COMPILER

Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Update})$ be a PKE scheme with key update.

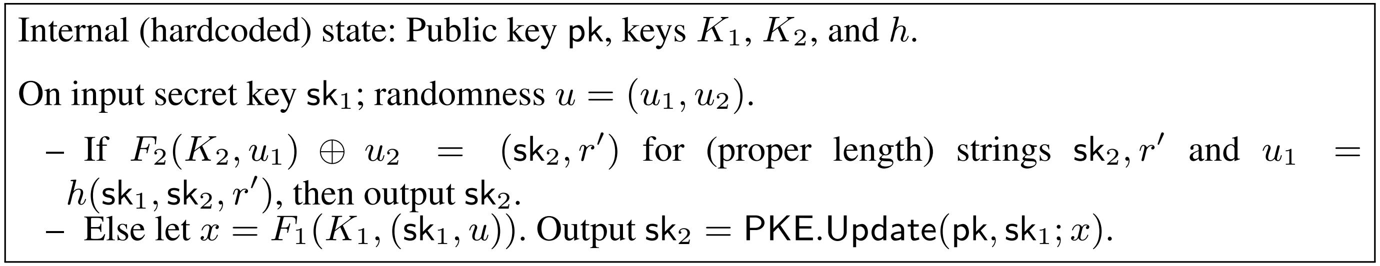
THE COMPILER

Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Update})$ be a PKE scheme with key update.

THE COMPILER

Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Update})$ be a PKE scheme with key update.

Define a new scheme whose public-key additionally contains obfuscations of two programs:

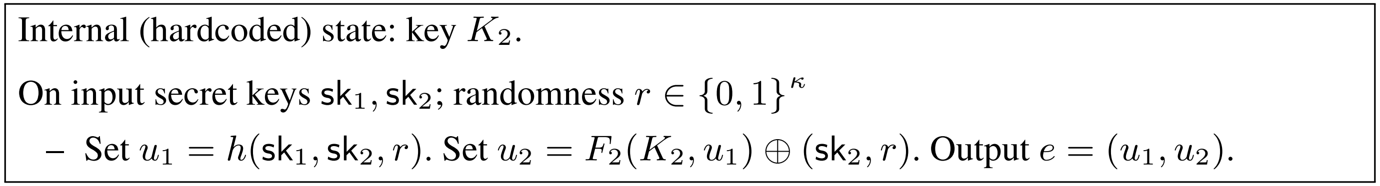


Internal (hardcoded) state: Public key pk, keys K_1, K_2, and h.

On input secret key sk_1; randomness $u = (u_1, u_2)$.

- If $F_2(K_2, u_1) \oplus u_2 = (\mathsf{sk}_2, r')$ for (proper length) strings sk_2, r' and $u_1 = h(\mathsf{sk}_1, \mathsf{sk}_2, r')$, then output sk_2.
- Else let $x = F_1(K_1, (\mathsf{sk}_1, u))$. Output $\mathsf{sk}_2 = \mathsf{PKE.Update}(\mathsf{pk}, \mathsf{sk}_1; x)$.

Fig. 1. Program Update



Internal (hardcoded) state: key K_2.

On input secret keys $\mathsf{sk}_1, \mathsf{sk}_2$; randomness $r \in \{0,1\}^\kappa$

- Set $u_1 = h(\mathsf{sk}_1, \mathsf{sk}_2, r)$. Set $u_2 = F_2(K_2, u_1) \oplus (\mathsf{sk}_2, r)$. Output $e = (u_1, u_2)$.

Fig. 2. Program Explain

ANALYSIS 1

Main Idea: Simulator uses obfuscated Explain to produce "honest-looking" randomness.

ANALYSIS 1

Main Idea: Simulator uses obfuscated Explain to produce “honest-looking” randomness.

ANALYSIS 1

Main Idea: Simulator uses obfuscated Explain to produce "honest-looking" randomness.

But this requires the simulator to access two consecutive keys simultaneously!

ANALYSIS 1

Main Idea: Simulator uses obfuscated Explain to produce "honest-looking" randomness.

But this requires the simulator to access two consecutive keys simultaneously!

ANALYSIS 1

Main Idea: Simulator uses obfuscated Explain to produce “honest-looking” randomness.

But this requires the simulator to access two consecutive keys simultaneously!

We thus need to define a new notion of consecutive continual leakage-resilience where the adversary can ask for leakage functions on consecutive keys.

ANALYSIS 1

Main Idea: Simulator uses obfuscated Explain to produce “honest-looking” randomness.

But this requires the simulator to access two consecutive keys simultaneously!

We thus need to define a new notion of consecutive continual leakage-resilience where the adversary can ask for leakage functions on consecutive keys.

ANALYSIS 2

Theorem (informal). The compiled scheme is secure with leakage on key-updates if the original scheme is consecutive continual leakage resilient and the obfuscator is a "public-coin" differing-inputs [IPS'15] obfuscator.

ANALYSIS 2

Theorem (informal). The compiled scheme is secure with leakage on key-updates if the original scheme is consecutive continual leakage resilient and the obfuscator is a "public-coin" differing-inputs [IPS'15] obfuscator.

ANALYSIS 2

Theorem (informal). The compiled scheme is secure with leakage on key-updates if the original scheme is consecutive continual leakage resilient and the obfuscator is a “public-coin” differing-inputs [IPS’15] obfuscator.

Note: Worse leakage rate achievable only using indistinguishability obfuscation.

ACHIEVING CONSECUTIVE CONTINUAL LEAKAGE-RESILIENCE

We show that existing continual leakage-resilient PKE schemes [BKKV’10,DHLW’10] can be upgraded to consecutive continual leakage without changing the underlying assumptions.

ACHIEVING CONSECUTIVE CONTINUAL LEAKAGE-RESILIENCE

We show that existing continual leakage-resilient PKE schemes [BKKV'10,DHLW'10] can be upgraded to consecutive continual leakage without changing the underlying assumptions.

ACHIEVING CONSECUTIVE CONTINUAL LEAKAGE-RESILIENCE

We show that existing continual leakage-resilient PKE schemes [BKKV'10,DHLW'10] can be upgraded to consecutive continual leakage without changing the underlying assumptions.

Via our compiler we get PKE with leakage on key-updates with optimal leakage rate under bilinear map assumptions + public-coin differing-inputs obfuscation [IPS'15].

COMPARISON TO PRIOR WORK

[LLW'11] achieves continual leakage resilience with leakage on key updates from bilinear map assumptions but worse leakage rate.

COMPARISON TO PRIOR WORK

[LLW'11] achieves continual leakage resilience with leakage on key updates from bilinear map assumptions but worse leakage rate.

OUTLINE OF TALK

Leakage Models for PKE —

  Bounded, Continual, and Continual w/ Leakage on Key Update

Results in Continual Model: A Generic Compiler to Achieve Leakage on Key Update

Results in Bounded Model: A New Approach to Optimal Leakage Rate

Conclusion and Open Problems

BACKGROUND: SW-PKE [SW'13]

Key-Generation: Choose a key K and output K as the secret key and the obfuscation of a program Encrypt that on inputs x,r outputs $F(K,r) + x$.

BACKGROUND: SW-PKE [SW'13]

Key-Generation: Choose a key K and output K as the secret key and the obfuscation of a program Encrypt that on inputs x,r outputs $F(K,r) + x$.

BACKGROUND: SW-PKE [SW'13]

Key-Generation: Choose a key K and output K as the secret key and the obfuscation of a program Encrypt that on inputs x,r outputs $F(K,r) + x$.

Encryption: To encrypt x choose random r and compute $y =$ Encrypt(x,r); output (r,y).

BACKGROUND: SW-PKE [SW'13]

Key-Generation: Choose a key K and output K as the secret key and the obfuscation of a program Encrypt that on inputs x,r outputs $F(K,r) + x$.

Encryption: To encrypt x choose random r and compute y = Encrypt(x,r); output (r,y).

BACKGROUND: SW-PKE [SW'13]

Key-Generation: Choose a key K and output K as the secret key and the obfuscation of a program Encrypt that on inputs x,r outputs $F(K,r) + x$.

Encryption: To encrypt x choose random r and compute $y =$ Encrypt(x,r); output (r,y).

SW'13 shows (a modification of) this scheme is IND-CPA using indistinguishability obfuscation.

MAKING IT LEAKAGE-RESILIENT

To make the scheme bounded leakage-resilient, we modify it in two ways:

MAKING IT LEAKAGE-RESILIENT

To make the scheme bounded leakage-resilient, we modify it in two ways:

1. Assume that F is not just a PRF but also a randomness extractor.

MAKING IT LEAKAGE-RESILIENT

To make the scheme bounded leakage-resilient, we modify it in two ways:

1. Assume that F is not just a PRF but also a randomness extractor.
2. Make the secret decryption key not K but obfuscation of program Decrypt that on input y,r outputs F(K,r)+y.

ANALYSIS

Theorem (informal). The modified scheme is bounded leakage-resilient using indistinguishability obfuscation.

ANALYSIS

Theorem (informal). The modified scheme is bounded leakage-resilient using indistinguishability obfuscation.

ANALYSIS

Theorem (informal). The modified scheme is bounded leakage-resilient using indistinguishability obfuscation.

Intuition: Following [SW'13] we use a puncturable PRF and switch F(K,r) used in the challenge ciphertext to a truly random, hardcoded value.

ANALYSIS

Theorem (informal). The modified scheme is bounded leakage-resilient using indistinguishability obfuscation.

Intuition: Following [SW'13] we use a puncturable PRF and switch F(K,r) used in the challenge ciphertext to a truly random, hardcoded value.

ANALYSIS

Theorem (informal). The modified scheme is bounded leakage-resilient using indistinguishability obfuscation.

Intuition: Following [SW'13] we use a puncturable PRF and switch F(K,r) used in the challenge ciphertext to a truly random, hardcoded value.

But note we can now leak on this hardcoded value since encryption uses a randomness extractor.

IMPROVING THE LEAKAGE RATE

This initial idea does not give optimal leakage rate because the secret key is large (contains the obfuscated decryption program).

IMPROVING THE LEAKAGE RATE

This initial idea does not give optimal leakage rate because the secret key is large (contains the obfuscated decryption program).

IMPROVING THE LEAKAGE RATE

This initial idea does not give optimal leakage rate because the secret key is large (contains the obfuscated decryption program).

Can we just make this obfuscated program public? Of course not! Then anyone could decrypt.

IMPROVING THE LEAKAGE RATE

This initial idea does not give optimal leakage rate because the secret key is large (contains the obfuscated decryption program).

Can we just make this obfuscated program public? Of course not! Then anyone could decrypt.

IMPROVING THE LEAKAGE RATE

This initial idea does not give optimal leakage rate because the secret key is large (contains the obfuscated decryption program).

Can we just make this obfuscated program public? Of course not! Then anyone could decrypt.

Solution: Make the program take an additional short signed input to run, this short signed input then becomes the new secret key.

COMPARISON TO PRIOR WORK

[HLWW'13] showed that any PKE scheme can be made bounded leakage resilient generically but with a suboptimal leakage rate.

COMPARISON TO PRIOR WORK

[HLWW'13] showed that any PKE scheme can be made bounded leakage resilient generically but with a suboptimal leakage rate.

COMPARISON TO PRIOR WORK

[HLWW'13] showed that any PKE scheme can be made bounded leakage resilient generically but with a suboptimal leakage rate.

Our result can be viewed as showed that obfuscation + OWF is sufficient for optimal leakage rate.

COMPARISON TO PRIOR WORK

[HLWW'13] showed that any PKE scheme can be made bounded leakage resilient generically but with a suboptimal leakage rate.

Our result can be viewed as showed that obfuscation + OWF is sufficient for optimal leakage rate.

COMPARISON TO PRIOR WORK

[HLWW'13] showed that any PKE scheme can be made bounded leakage resilient generically but with a suboptimal leakage rate.

Our result can be viewed as showed that obfuscation + OWF is sufficient for optimal leakage rate.

Optimal leakage rate is also known from other specific assumptions, e.g. DDH [NS'09].

OUTLINE OF TALK

Leakage Models for PKE —

- Bounded, Continual, and Continual w/ Leakage on Key Update

Results in Continual Model: A Generic Compiler to Achieve Leakage on Key Update

Results in Bounded Model: A New Approach to Optimal Leakage Rate

Conclusion and Open Problems

SUMMARY

We gave two main results:

SUMMARY

We gave two main results:

1. Compiler from (consecutive) continual leakage-resilience to leak on key-updates.

SUMMARY

We gave two main results:

1. Compiler from (consecutive) continual leakage-resilience to leak on key-updates.
2. Modification of [SW'13] to achieve bounded leakage with optimal leakage rate.

OPEN QUESTIONS

Can we achieve leakage on key-updates with optimal leakage rate?

OPEN QUESTIONS

Can we achieve leakage on key-updates with optimal leakage rate?

OPEN QUESTIONS

Can we achieve leakage on key-updates with optimal leakage rate?

Can we achieve optimal leakage rate in the bounded leakage model from indistinguishability (not differing-inputs) obfuscation?

OPEN QUESTIONS

Can we achieve leakage on key-updates with optimal leakage rate?

Can we achieve optimal leakage rate in the bounded leakage model from indistinguishability (not differing-inputs) obfuscation?

OPEN QUESTIONS

Can we achieve leakage on key-updates with optimal leakage rate?

Can we achieve optimal leakage rate in the bounded leakage model from indistinguishability (not differing-inputs) obfuscation?

Can we achieve continual leakage resilience from (differing-inputs) obfuscation?

THANK YOU!

adam@cs.georgetown.edu