Leakage Resilient ElGamal Encryption

Eike Kiltz and *Krzysztof Pietrzak*

RUHR UNIVERSITÄT BOCHUM RUB

CWI Centrum Wiskunde & Informatica

Asiacrypt 2010, December 9th, Singapore

Outline

1. Hybrid Encryption, the KEM/DEM framework
2. ElGamal KEM
3. Leakage Resilient Crypto
   - Why?
   - How?
   - Other models?
4. Leakage Resilient ElGamal

CCA1 secure KEM (Key Encapsulation Mechanism)

$KEM = \{\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}\} \approx PKE$ *for* **random** *messages.*
$KEM + DEM \Rightarrow PKE$

CCA1 secure KEM (Key Encapsulation Mechanism)

$KEM = \{\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}\} \approx PKE$ *for* **random** *messages.*
$KEM + DEM \Rightarrow PKE$

$$\Pr[\mathsf{Dec}(sk, C) = K \ : \ (pk, sk) \xleftarrow{\$} \mathsf{KG} \ ; \ (K, C) \xleftarrow{\$} \mathsf{Enc}(pk)] = 1$$

CCA1 secure KEM (Key Encapsulation Mechanism)

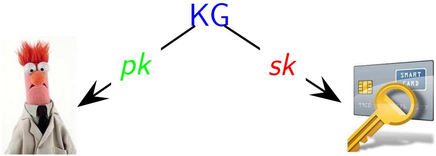
$KEM = \{\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}\} \approx PKE$ *for* **random** *messages.*
$KEM + DEM \Rightarrow PKE$

$$\Pr[\mathsf{Dec}(sk, C) = K \;:\; (pk, sk) \xleftarrow{\$} \mathsf{KG} \;;\; (K, C) \xleftarrow{\$} \mathsf{Enc}(pk)] = 1$$

CCA1 secure KEM (Key Encapsulation Mechanism)

$KEM = \{\textsf{KG}, \textsf{Enc}, \textsf{Dec}\} \approx PKE$ *for* **random** *messages.*
$KEM + DEM \Rightarrow PKE$

$$\Pr[\textsf{Dec}(sk, C) = K \;:\; (pk, sk) \stackrel{\$}{\leftarrow} \textsf{KG} \;;\; (K, C) \stackrel{\$}{\leftarrow} \textsf{Enc}(pk)] = 1$$

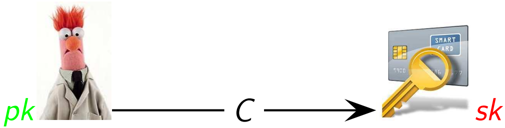


CCA1 secure KEM (Key Encapsulation Mechanism)

$KEM = \{\textsf{KG}, \textsf{Enc}, \textsf{Dec}\} \;\approx\; PKE$ *for* **random** *messages.*

$KEM + DEM \Rightarrow PKE$

$$\Pr[\textsf{Dec}(sk, C) = K \;:\; (pk, sk) \stackrel{\$}{\leftarrow} \textsf{KG}\;;\; (K, C) \stackrel{\$}{\leftarrow} \textsf{Enc}(pk)] = 1$$



CCA1 secure KEM (Key Encapsulation Mechanism)

KEM $= \{\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}\} \approx$ *PKE for* **random** *messages.*
KEM $+$ *DEM* $\Rightarrow$ *PKE*

$$\Pr[\mathsf{Dec}(sk, C) = K \ : \ (pk, sk) \xleftarrow{\$} \mathsf{KG} \ ; \ (K, C) \xleftarrow{\$} \mathsf{Enc}(pk)] = 1$$

pk  $\longleftarrow \mathsf{Dec}(sk, C)$ —— 

sk

CCA1 secure KEM (Key Encapsulation Mechanism)

$KEM = \{\textsf{KG}, \textsf{Enc}, \textsf{Dec}\} \approx PKE$ *for* **random** *messages.*
$KEM + DEM \Rightarrow PKE$

$$\Pr[\textsf{Dec}(sk, C) = K \;:\; (pk, sk) \xleftarrow{\$} \textsf{KG} \;;\; (K, C) \xleftarrow{\$} \textsf{Enc}(pk)] = 1$$

$K_b, C \longrightarrow pk$ 

sk

$$(K_0, C) \leftarrow \textsf{Enc}(pk)\ ,\ K_1 \xleftarrow{\$} \mathcal{K}\ ,\ b \leftarrow \{0, 1\}$$

CCA1 secure KEM (Key Encapsulation Mechanism)

$KEM = \{\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}\} \approx$ *PKE for* **random** *messages.*

$KEM + DEM \Rightarrow PKE$

$$\Pr[\mathsf{Dec}(sk, C) = K \ : \ (pk, sk) \xleftarrow{\$} \mathsf{KG} \ ; \ (K, C) \xleftarrow{\$} \mathsf{Enc}(pk)] = 1$$

$K_b, C \longrightarrow pk$ 

sk

$$(K_0, C) \leftarrow \mathsf{Enc}(pk) \ , \ K_1 \xleftarrow{\$} \mathcal{K} \ , \ b \leftarrow \{0, 1\}$$

CCA1 security: $\forall$  : $\Pr[$  guesses $b] - 1/2 = \text{negl}$

ElGamal KEM

public parameter: Cyclic group $\mathbb{G}$ of prime order p, $g = \langle \mathbb{G} \rangle$

ElGamal KEM

public parameter: Cyclic group $\mathbb{G}$ of prime order p, $g = \langle \mathbb{G} \rangle$

KG: $sk = x$, $pk = g^x$ where $x \xleftarrow{\$} \mathbb{Z}_p$

ElGamal KEM

public parameter: Cyclic group $\mathbb{G}$ of prime order p, $g = \langle \mathbb{G} \rangle$

KG: $sk = x$, $pk = g^x$ where $x \xleftarrow{\$} \mathbb{Z}_p$

Enc(pk): output $(C := g^r, K := g^{rx})$ where $r \xleftarrow{\$} \mathbb{Z}_p$

Dec(sk, C): output C^x

ElGamal KEM

public parameter: Cyclic group $\mathbb{G}$ of prime order p, $g = \langle \mathbb{G} \rangle$

KG: $sk = x$, $pk = g^x$ where $x \xleftarrow{\$} \mathbb{Z}_p$

Enc(pk): output $(C := g^r, K := g^{rx})$ where $r \xleftarrow{\$} \mathbb{Z}_p$

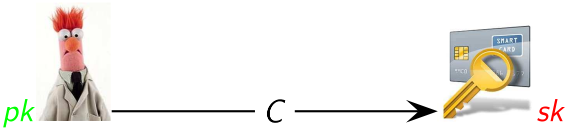
Dec(sk, C): output $C^x = g^{rx} = K$

Side-Channel Attacks

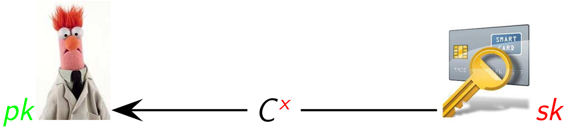
pk 

sk

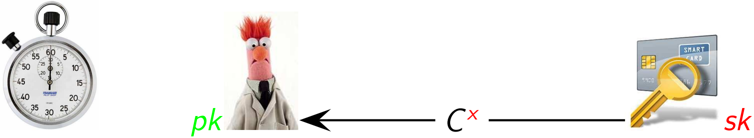
Side-Channel Attacks



Side-Channel Attacks



Side-Channel Attacks



- Can e.g. measure time it takes to compute C^x

Side-Channel Attacks

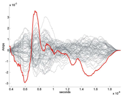
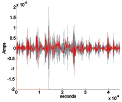
$pk \longleftarrow C^{x}$ ———  sk

- Can e.g. measure time it takes to compute C^{x}

> *Side-Channel Attack: Cryptanalytic attack exploring information leaked from a physical implementation of a cryptosystem.*

More side-channel attacks

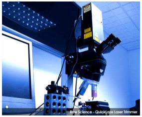
- power analysis

- radiation, sound, heat,. . .



- probing attacks

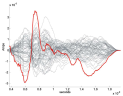


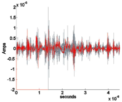
- cold-boot attacks

- cache attacks

More side-channel attacks

- power analysis
  [Eisenbarth et al. CRYPTO'08]
  break wireless car keys

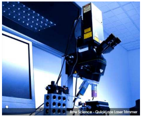




- radiation, sound, heat,. . .



- probing attacks



- cold-boot attacks
  [Halderman et al. USENIX'08]
  break disc-encryption schemes


- cache attacks
  [Ristenpart et al. CCS'09]
  break cloud computing

amazon webservices™

Side-Channel Countermeasures

Usually Ad-hoc

Implement countermeasures to prevent known attacks.

Side-Channel Countermeasures

Usually Ad-hoc

Implement countermeasures to prevent known attacks.

Timing Make computation time independent of inputs.

Side-Channel Countermeasures

Usually Ad-hoc

Implement countermeasures to prevent known attacks.

Timing Make computation time independent of inputs.

Radiation Shield the chip.

Side-Channel Countermeasures

Usually Ad-hoc

Implement countermeasures to prevent known attacks.

Timing Make computation time independent of inputs.

Radiation Shield the chip.

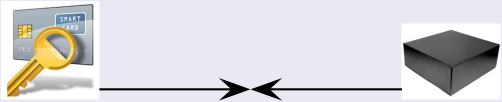
make physical device look more like a black-box

Leakage-Resilient Cryptography [DP'08]

extend black-box model to incorporte leakage

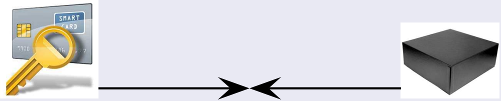
Leakage-Resilient Cryptography [DP'08]

extend black-box model to incorporte leakage



Leakage-Resilient Cryptography [DP'08]

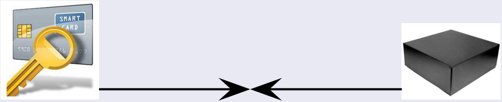
extend black-box model to incorporte leakage



- *Computation is split in steps.*
- *Adversary has black-box access + get bounded amount of arbitrary, adaptively chosen leakage of every step.*

Leakage-Resilient Cryptography [DP'08]

extend black-box model to incorporte leakage



- *Computation is split in steps.*
- *Adversary has black-box access + get bounded amount of arbitrary, adaptively chosen leakage of every step.*
  (only computation leaks "axiom" [MR04].)

Leakage Resilient Cryptography Cont.

- LR primitives must be stateful.

Leakage Resilient Cryptography Cont.

- LR primitives must be stateful.
- Key evolution:
  LR stream-cipher [DP'08,P09,YSPY'10]
  LR (tree-based) signatures [FKPR'10]
  Evolving PKE sk difficult: must decrypt for fixed pk.

Leakage Resilient Cryptography Cont.

- LR primitives must be stateful.
- Key evolution:
  LR stream-cipher [DP'08,P09,YSPY'10]
  LR (tree-based) signatures [FKPR'10]
  Evolving PKE sk difficult: must decrypt for fixed pk.
- We secret-share key (aka blinding.) Frequently re-share.

Leakage Resilient Cryptography Cont.

- LR primitives must be stateful.
- Key evolution:
  LR stream-cipher [DP'08,P09,YSPY'10]
  LR (tree-based) signatures [FKPR'10]
  Evolving PKE sk difficult: must decrypt for fixed pk.
- We secret-share key (aka blinding.) Frequently re-share.
- Scheme is very efficient ($\approx$ 2x basic ElGamal)
- Security proofs are very limited (generic group.)

Some Related Work

- General Compilers [Goldwasser-Rothblum,Juma-Vahlis Crypto'10]
  General but not practical (One Encryption get gate / Fully homomorphic encryption)
- Non-Continuous leakage (BRM/memory-attacks, auxiliary input), next talk.
- Continuous memory attacks [DHLW,BKKV FOCS'10], [LLW eprint 2010/562].

ElGamal KEM with shared key

g^{x} 

$x \in \mathbb{Z}_p$

ElGamal KEM with shared key

g^{x}  —— C $\longrightarrow$ 

$x \in \mathbb{Z}_p$

ElGamal KEM with shared key

g^{x}  $\leftarrow C^{x}$ —— 

$x \in \mathbb{Z}_p$

ElGamal KEM with shared key

g^x  —— $C, f(.)$ → 

$x \in \mathbb{Z}_p$

- Not Leakage-Resilient (learn x bit by bit.)

ElGamal KEM with shared key

g^{x}  $\longleftarrow C^{x}, f(x)$ —— 

$x \in \mathbb{Z}_p$

- Not Leakage-Resilient (learn x bit by bit.)

ElGamal KEM with shared key

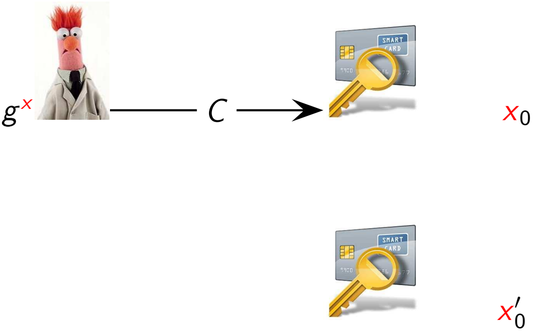
g^x 

x_0

x_0'

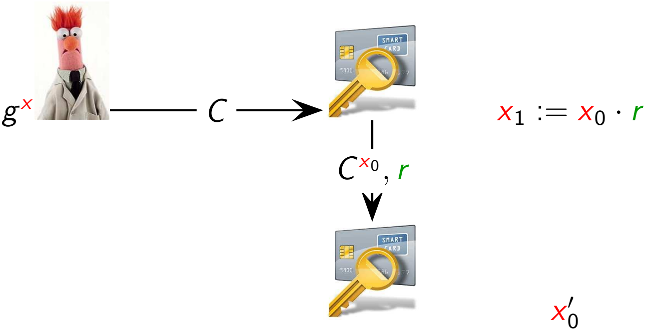
- Not Leakage-Resilient (learn x bit by bit.)
- Multiplicatively Secret-Share $x = x_0 \cdot x_0'$.

ElGamal KEM with shared key



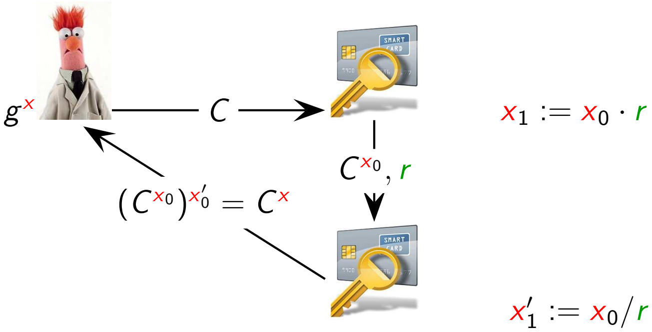
- Not Leakage-Resilient (learn x bit by bit.)
- Multiplicatively Secret-Share $x = x_0 \cdot x_0'$.

ElGamal KEM with shared key



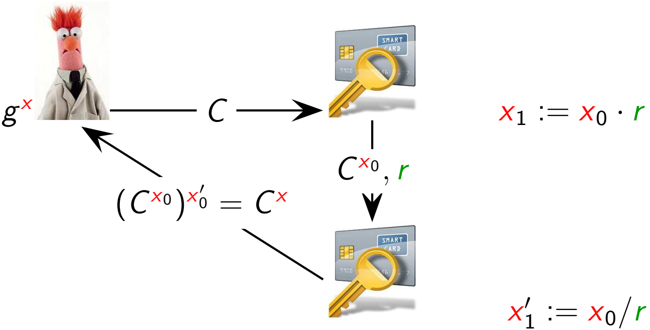
- Not Leakage-Resilient (learn x bit by bit.)
- Multiplicatively Secret-Share $x = x_0 \cdot x'_0$.

ElGamal KEM with shared key



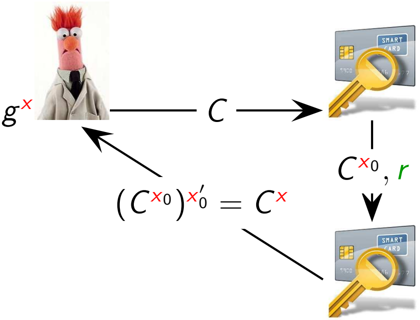
- Not Leakage-Resilient (learn x bit by bit.)
- Multiplicatively Secret-Share $x = x_0 \cdot x'_0$.

ElGamal KEM with shared key



- Not Leakage-Resilient (learn x bit by bit.)
- Multiplicatively Secret-Share $x = x_0 \cdot x'_0$.
- Re-Sharing: $x_{i+1} \leftarrow x_i \cdot r$, $x'_{i+1} \leftarrow x'_i / r$.

ElGamal KEM with shared key



$x_1 := x_0 \cdot r$

$x'_1 := x_0 / r$

- Not Leakage-Resilient (learn x bit by bit.)
- Multiplicatively Secret-Share $x = x_0 \cdot x'_0$.
- Re-Sharing: $x_{i+1} \leftarrow x_i \cdot r$, $x'_{i+1} \leftarrow x'_i / r$.
- i'th query:  adaptively chooses $f_i(.)$, $f'_i(.)$.
  Gets leakage $f_i(x_i, r)$, $f'_i(x'_i, r, C^{x_i})$.

Conjecture: ElGamal KEM (as on previous slide) is leakage-resilient if

- *the group order p is not smooth (i.e. $p-1$ has large prime factor.)*
- *Range of leakage functions is bounded to, say $\lambda = 0.25 \cdot \log(p)$ bits.*

[1] Howgrave-Graham, Nguyen, Shparlinski. *Hidden number problem with hidden multipliers, timed-release crypto, and noisy exponentiation. Math. Comput. 72(243): 1473-1485 (2003)*

Conjecture: ElGamal KEM (as on previous slide) is leakage-resilient if

- *the group order p is not smooth (i.e. $p-1$ has large prime factor.)*
- *Range of leakage functions is bounded to, say $\lambda = 0.25 \cdot \log(p)$ bits.*

- Attack exits if we use *additive* secret sharing, i.e. $x = x_i + x_i' \bmod p$ instead $x = x_i \cdot x_i' \bmod p$.

[1] Howgrave-Graham, Nguyen, Shparlinski. *Hidden number problem with hidden multipliers, timed-release crypto, and noisy exponentiation. Math. Comput. 72(243): 1473-1485 (2003)*

Conjecture: ElGamal KEM (as on previous slide) is leakage-resilient if

- *the group order p is not smooth (i.e. $p - 1$ has large prime factor.)*
- *Range of leakage functions is bounded to, say $\lambda = 0.25 \cdot \log(p)$ bits.*

- Attack exits if we use *additive* secret sharing, i.e. $x = x_i + x_i' \bmod p$ instead $x = x_i \cdot x_i' \bmod p$.
- Attack exists if $p - 1$ is smooth.

[1] Howgrave-Graham, Nguyen, Shparlinski. *Hidden number problem with hidden multipliers, timed-release crypto, and noisy exponentiation. Math. Comput. 72(243): 1473-1485 (2003)*

Conjecture: ElGamal KEM (as on previous slide) is leakage-resilient if

- *the group order p is not smooth (i.e. $p-1$ has large prime factor.)*
- *Range of leakage functions is bounded to, say $\lambda = 0.25 \cdot \log(p)$ bits.*

- Attack exits if we use *additive* secret sharing, i.e. $x = x_i + x_i' \bmod p$ instead $x = x_i \cdot x_i' \bmod p$.
- Attack exists if $p-1$ is smooth.
- Attack exists if $\lambda = 0.4 \cdot \log(p)$.[1]

[1] Howgrave-Graham, Nguyen, Shparlinski. *Hidden number problem with hidden multipliers, timed-release crypto, and noisy exponentiation. Math. Comput. 72(243): 1473-1485 (2003)*

Conjecture: ElGamal KEM (as on previous slide) is leakage-resilient if

- *the group order p is not smooth (i.e. $p-1$ has large prime factor.)*
- *Range of leakage functions is bounded to, say $\lambda = 0.25 \cdot \log(p)$ bits.*

- Attack exits if we use *additive* secret sharing, i.e. $x = x_i + x_i' \bmod p$ instead $x = x_i \cdot x_i' \bmod p$.
- Attack exists if $p-1$ is smooth.
- Attack exists if $\lambda = 0.4 \cdot \log(p)$.[1]
- Scheme if "lifted" to bilinear groups is secure in generic group model (next slides.)

[1] Howgrave-Graham, Nguyen, Shparlinski. *Hidden number problem with hidden multipliers, timed-release crypto, and noisy exponentiation. Math. Comput. 72(243): 1473-1485 (2003)*

Bilinear Groups

1. $\mathbb{G}$ is a (multiplicative) cyclic group of prime order p.
2. g is a generator of $\mathbb{G}$.
3. e is a bilinear map $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$
   1. $\forall a, b \in \mathbb{Z}, e(g^a, g^b) = e(g, g)^{ab}$
   2. $e(g, g) \stackrel{\text{def}}{=} g_T \neq 1$.

“lifted” ElGamal KEM

public parameter: $\mathbb{G}, \mathbb{G}_T$ of prime order p, $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$,

$$g = \langle \mathbb{G} \rangle,\ g_T \stackrel{\text{def}}{=} e(g, g)$$

"lifted" ElGamal KEM

public parameter: $\mathbb{G}, \mathbb{G}_T$ of prime order p, $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$,

$$g = \langle \mathbb{G} \rangle, \; g_T \stackrel{\text{def}}{=} e(g, g)$$

KG: $sk = g^{x}$, $pk = g_T^{x}$ where $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$

"lifted" ElGamal KEM

public parameter: $\mathbb{G}, \mathbb{G}_T$ of prime order p, $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$,

$$g = \langle \mathbb{G} \rangle, \; g_T \stackrel{\text{def}}{=} e(g, g)$$

KG: $sk = g^{x}$, $pk = g_T^{x}$ where $x \xleftarrow{\$} \mathbb{Z}_p$

Enc(pk): output $(C := g^r, K := g_T^{rx})$ where $r \xleftarrow{\$} \mathbb{Z}_p$

Dec(sk, C): output $e(C, g^{x})$

“lifted” ElGamal KEM

public parameter: $\mathbb{G}, \mathbb{G}_T$ of prime order p, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$,

$$g = \langle \mathbb{G} \rangle, \; g_T \stackrel{\text{def}}{=} e(g, g)$$

KG: $sk = g^x$, $pk = g_T^x$ where $x \xleftarrow{\$} \mathbb{Z}_p$

Enc(pk): output $(C := g^r, K := g_T^{rx})$ where $r \xleftarrow{\$} \mathbb{Z}_p$

Dec(sk, C): output $e(C, g^x) = g_T^{rx} = K$

“lifted” ElGamal KEM

public parameter: $\mathbb{G}, \mathbb{G}_T$ of prime order p, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$,

$$g = \langle \mathbb{G} \rangle,\ g_T \stackrel{\text{def}}{=} e(g, g)$$

KG: $\mathbf{sk} = \mathbf{g^x}$, $pk = g_T^{x}$ where $x \xleftarrow{\$} \mathbb{Z}_p$

Enc(pk): output $(C := g^r, K := g_T^{rx})$ where $r \xleftarrow{\$} \mathbb{Z}_p$

Dec(sk, C): output $e(C, g^x) = g_T^{rx} = K$

"lifted" ElGamal KEM

public parameter: $\mathbb{G}, \mathbb{G}_T$ of prime order p, $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$,

$$g = \langle \mathbb{G} \rangle,\ g_T \stackrel{\text{def}}{=} e(g, g)$$

KG: $\mathbf{sk} = \mathbf{g}^{\mathbf{x}}$, $pk = g_T^x$ where $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$

Enc(pk): output $(C := g^r, K := g_T^{rx})$ where $r \stackrel{\$}{\leftarrow} \mathbb{Z}_p$

Dec(sk, C): output $e(C, g^x) = g_T^{rx} = K$

Like for standard ElGamal, can define shared-key version $g^x = g^{x-r} \circ g^r$.

"lifted" ElGamal KEM

public parameter: $\mathbb{G}, \mathbb{G}_T$ of prime order p, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$,

$$g = \langle \mathbb{G} \rangle,\ g_T \stackrel{\text{def}}{=} e(g, g)$$

KG: $\mathbf{sk} = \mathbf{g^x}$, $pk = g_T^x$ where $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$

Enc(pk): output $(C := g^r, K := g_T^{rx})$ where $r \stackrel{\$}{\leftarrow} \mathbb{Z}_p$

Dec(sk, C): output $e(C, g^x) = g_T^{rx} = K$

Like for standard ElGamal, can define shared-key version $g^x = g^{x-r} \circ g^r$.

Theorem

In the bilinear generic group model the lifted, shared-key ElGamal KEM is Leakage-Resilient (CCA1).
The leakage per invocation can be $< .49|\log(p)|$ *bits.*


Questions?

Questions?

ICITS 2011, Amsterdam, The Netherlands, May 21 - 24, 2011
5th International Conference on Information Theoretic Security
Submission deadline: Dec 10, 2010

Invited Speakers:

Benny Applebaum, Alexander Barg, Imre Csiszar, Ivan Damgaard, Yuval Ishai, Renato Renner, Leonid Reyzin, Ronald de Wolf