Implementation and Evaluation of a Leakage-Resilient ElGamal KEM

David Galindo[1,2], Johann Großschädl[3], Zhe Liu[3], Praveen K. Vadnala[3], Srinivas Vivek[3]

[1] CNRS/Loria, France
[2] SCYTL Secure Electronic Voting, Spain
[3] University of Luxembourg

PROOFS 2014

Side-Channel Attacks

Use data leaked due to the physical nature of computation:

- running time
- power consumption
- electromagnetic-radiation leak
- acoustic emanation
- photons emissions
- ground electric potential
- fault attacks

Side-Channel Attacks Countermeasures

Aimed at specific attacks

Concrete implementations

Leakage model meaningful

Reasonably practical SCA-resistant primitives

SCA Countermeasures flow

input message

$K^\star$

target computation

$f(K^\star, \mathbf{T})$

leakage model

φ

noise

$\mathbf{N}$

actual leakage

$\mathbf{X} \overset{\mathbf{N}}{\approx} \varphi((K^\star, \mathbf{T}))$

distinguisher

$\mathcal{D}$

attack/non-attack

$\widehat{K} = \mathcal{D}(\mathbf{X}, \mathbf{T})$

Side-Channel Attacks Countermeasures

Aimed at specific attacks

Concrete implementations

Leakage model meaningful

Reasonably practical SCA-resistant primitives

However...

SCA Countermeasures flow

input message
$K^\star$
target computation
$f(K^\star, \mathbf{T})$
leakage model
φ
noise
$\mathbf{N}$
actual leakage
$\mathbf{X} \overset{\mathbf{N}}{\approx} \varphi((K^\star, \mathbf{T}))$
distinguisher
$\mathcal{D}$
attack/non-attack
$\widehat{K} = \mathcal{D}(\mathbf{X}, \mathbf{T})$

Side-Channel Attacks Countermeasures

- ☹ Aimed at specific attacks
- ☺ Concrete implementations
- ☺ Leakage model meaningful
- ☺ Reasonably practical SCA-resistant primitives

  A new attack $(\varphi, \mathbf{N}, \mathcal{D})$ might be discovered
- ☹ Endless? **cat-and-mouse** game

SCA Countermeasures flow

input message
$K^\star$
target computation
$f(K^\star, \mathbf{T})$
leakage model
φ
noise
$\mathbf{N}$
actual leakage
$\mathbf{X} \overset{\mathbf{N}}{\approx} \varphi((K^\star, \mathbf{T}))$
distinguisher
$\mathcal{D}$
security?
$\widehat{K} = \mathcal{D}(\mathbf{X}, \mathbf{T})$

SCA countermeasures		Leakage-Resilient Crypto	
☹	Aimed at specific attacks	☺	Aimed at generic attacks
☺	Concrete implementations	☹	No implementations
☺	Leakage model meaningful	☹	Leakage model generic
☺	Reasonably practical SCA-resistant primitives	☹	Not practical
	A new attack $(\varphi, \mathbf{N}, \mathcal{D})$ might be discovered		
☹	Endless? **cat-and-mouse** game	☺	Security reduction

Meaningful Leakage-Resilient Cryptography

- ☺ Aimed at general attacks
- ☺ Leakage model meaningful
- ☺ Reasonably practical SCA-resistant primitives
- ☺ Security reduction
- ☺ Concrete implementations

Meaningful Leakage-Resilient Cryptography

- ☺ Aimed at general attacks
- ☺ Leakage model meaningful
- ☺ Reasonably practical SCA-resistant primitives
- ☺ Security reduction
- ☺ Concrete implementations

In this work we take a step forward towards to this goal

Our contribution

Scytl
Innovating Democracy

- A **more reasonable** leakage modeling
- We depart from an existing practical ElGamal KEM and modify it using **practical motivations**
- We use the **theory and practice** of SCA to argue that it potentially meets the leakage bound
- We **implement** the scheme on an ARM Cortex M-3 processor

Stateful Key Encapsulation Mechanisms

A **stateful** KEM scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}_1, \mathsf{Dec}_2)$ consists of efficient algorithms:

- $\mathsf{KeyGen}(1^\kappa)$ outputs $(pk, (sk_0, sk'_0))$
- $\mathsf{Enc}(pk)$ outputs (K, C)
- $\mathsf{Dec}_1(sk_{i-1}, C)$ updates sk_{i-1} to sk_i and outputs intermediate state w_i
- $\mathsf{Dec}_2(sk'_{i-1}, w_i)$ updates sk'_{i-1} to sk'_i and outputs key K or $\perp$

Scytl
Innovating Democracy

- KG(κ): choose $x, t_0 \xleftarrow{\$} \mathbf{Z}_q$. Set $X = g^x, sk_0 = t_0, sk'_0 = x/t_0$. Return $(X, (sk_0, sk'_0))$
- Enc(pk) choose $r \xleftarrow{\$} \mathbf{Z}_q$. Compute $C = g^r$ and $K = X^r$; return (C, K)
- Dec1(sk_{i-1}, C) pick $t_i \xleftarrow{\$} \mathbf{Z}_q$, set $sk_i = sk_{i-1} \cdot t_i,\ Y_i = C^{sk_i}$. Return (t_i, Y_i)
- Dec2($sk'_{i-1}, (t_i, Y_i), C$) set $sk'_i = sk'_{i-1} \cdot t_i^{-1}$, and return $K = Y_i^{sk'_i}$.

CCA1 with Leakage – Stateful KEM

We consider **chosen-ciphertext and leakage security against lunch-time attacks** (CCLA1)

CCLA1 Experiment

KEM-CCLA1$_{\mathrm{KEM}}(\mathcal{A},\kappa,\lambda)$

$(pk,(sk_0,sk'_0)) \leftarrow \mathrm{KG}^*(\kappa,\lambda)$

$w \leftarrow \mathcal{A}^{O^{\mathrm{CCLA1}}(\cdot)}(pk)$

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

$(C^*,K_0) \leftarrow \mathrm{Enc}^*(pk)$

$K_1 \stackrel{\$}{\leftarrow} \mathcal{K}$

$b' \leftarrow \mathcal{A}(w,C^*,K_b)$

KEM-Leak-Oracle $O^{\mathrm{CCLA1}}(C,f_i,h_i)$

$(sk_i,w_i) \stackrel{r_i}{\leftarrow} \mathrm{Dec1}^*(sk_{i-1},C)$

$(sk'_i,K) \stackrel{r'_i}{\leftarrow} \mathrm{Dec2}^*(sk'_{i-1},w_i)$

$\Lambda_i := f_i(sk_{i-1},r_i)$

$\Lambda'_i := h_i(sk'_{i-1},r'_i,w_i)$

$i := i+1$

Return (K,Λ_i,Λ'_i)

CCA1 with Leakage – Stateful KEM

We consider **chosen-ciphertext and leakage security against lunch-time attacks** (CCLA1)

CCLA1 Experiment

KEM-CCLA1$_{\mathrm{KEM}}(\mathcal{A}, \kappa, \lambda)$

$(pk, (sk_0, sk_0')) \leftarrow \mathrm{KG}^*(\kappa, \lambda)$
$w \leftarrow \mathcal{A}^{O^{\mathrm{CCLA1}}(\cdot)}(pk)$
$b \xleftarrow{\$} \{0,1\}$
$(C^*, K_0) \leftarrow \mathrm{Enc}^*(pk)$
$K_1 \xleftarrow{\$} \mathcal{K}$
$b' \leftarrow \mathcal{A}(w, C^*, K_b)$

KEM-Leak-Oracle $O^{\mathrm{CCLA1}}(C, f_i, h_i)$

$(sk_i, w_i) \xleftarrow{r_i} \mathrm{Dec1}^*(sk_{i-1}, C)$
$(sk_i', K) \xleftarrow{r_i'} \mathrm{Dec2}^*(sk_{i-1}', w_i)$
$\Lambda_i := f_i(sk_{i-1}, r_i)$
$\Lambda_i' := h_i(sk_{i-1}', r_i', w_i)$
$i := i + 1$
Return $(K, \Lambda_i, \Lambda_i')$

Restriction on leakage functions f_i, h_i

$$\tilde{\mathbf{H}}_\infty(\mathfrak{t} \mid f_i(\sigma_{i-1}, r_i)) \geq \mathbf{H}_\infty(\mathfrak{t}) - \lambda \quad \forall \mathfrak{t} \in \sigma_{i-1} \cup r_i,$$
$$\tilde{\mathbf{H}}_\infty(\mathfrak{t} \mid h_i(\sigma_{i-1}', r_i', w_i)) \geq \mathbf{H}_\infty(\mathfrak{t}) - \lambda \quad \forall \mathfrak{t} \in \sigma_{i-1}' \cup r_i' \cup w_i.$$

- State of the art does not allow to give a security reduction with leakage
- If f_i, h_i leak $\lambda \geq 3/8 \log q$ bits of each share of the secret key, then there exists a heuristic attack [Galindo-Vivek,IPL 2014]
- Probably due to the fact that any **exponentiation algorithm** inherently leaks information about the **exponent**

- State of the art does not allow to give a security reduction with leakage
- If f_i, h_i leak $\lambda \geq 3/8 \log q$ bits of each share of the secret key, then there exists a heuristic attack [Galindo-Vivek,IPL 2014]
- Probably due to the fact that any **exponentiation algorithm** inherently leaks information about the **exponent**

Idea! Avoid placing secret data on your exponentiations' exponents...

Asymmetric Pairings

- Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be groups of prime order q
- $\mathbb{G}_1 =< g >, \mathbb{G}_2 =< G >$
- Pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$
  - bilinear: $e(g^a, g^b) = e(g, g)^{ab}, \forall a, b \in \mathbb{Z}$
  - non-degenerate: $\mathbb{G}_T =< e(g, G) >$

- KG(κ): choose $x, t_0 \xleftarrow{\$} \mathbf{Z}_q$. Set $X = g^x, sk_0 = g^{t_0}, sk_0' = g^{x-t_0}$, and $X_T = e(X, G)$. Return $(X_T, (sk_0, sk_0'))$
- Enc(pk) choose $r \xleftarrow{\$} \mathbf{Z}_q$. Compute $C = G^r$ and $K = X_T^r$; return (C, K)
- Dec1(C, sk_{i-1}) pick $t_i \xleftarrow{\$} \mathbf{Z}_q$, set $sk_i = sk_{i-1} \cdot G^{t_i}$, $Y_i = e(sk_i, C)$. Return (t_i, Y_i)
- Dec2$(sk_{i-1}', (t_i, Y_i), C)$ set $sk_i' = sk_{i-1}' \cdot G^{-t_i}$, and $Y_i' = e(sk_i', C)$. Return $K = Y_i \cdot Y_i' \in \mathbb{G}_T$

ElGamal KEM with Multiplicative Masking

Scytl
Innovating Democracy

- KG(κ): choose $x, t_0 \xleftarrow{\$} \mathbf{Z}_q$. Set $X = g^x$, $sk_0 = t_0$, $sk_0' = x/t_0$. Return $(X, (sk_0, sk_0'))$
- Enc(pk) choose $r \xleftarrow{\$} \mathbf{Z}_q$. Compute $C = g^r$ and $K = X^r$; return (C, K)
- Dec1(sk_{i-1}, C) pick $t_i \xleftarrow{\$} \mathbf{Z}_q$, set $sk_i = sk_{i-1} \cdot t_i$, $Y_i = C^{sk_i}$. Return (t_i, Y_i)
- Dec2($sk'_{i-1}, (t_i, Y_i), C$) set $sk'_i = sk'_{i-1} \cdot t_i^{-1}$, and return $K = Y_i^{sk'_i}$.

- KG(κ): choose $x, t_0 \xleftarrow{\$} \mathbf{Z}_q$. Set $X = g^x, sk_0 = g^{t_0}, sk_0' = g^{x-t_0}$, and $X_T = e(X, G)$. Return $(X_T, (sk_0, sk_0'))$
- Enc(pk) choose $r \xleftarrow{\$} \mathbf{Z}_q$. Compute $C = G^r$ and $K = X_T^r$; return (C, K)
- Dec1(C, sk_{i-1}) pick $t_i \xleftarrow{\$} \mathbf{Z}_q$, set $sk_i = sk_{i-1} \cdot G^{t_i}$, $Y_i = e(sk_i, C)$. Return (t_i, Y_i)
- Dec2$(sk_{i-1}', (t_i, Y_i), C)$ set $sk_i' = sk_{i-1}' \cdot G^{-t_i}$, and $Y_i' = e(sk_i', C)$. Return $K = Y_i \cdot Y_i' \in \mathbb{G}_T$

Security reduction in the Generic Bilinear Group Model if the leakage is **bounded in size**

Scytl Innovating Democracy

- KG(κ): choose $x, t_0 \stackrel{\$}{\leftarrow} \mathbf{Z}_q$. Set $X = g^x, sk_0 = g^{t_0}, sk_0' = g^{x-t_0}$, and $X_T = e(X, G)$. Return $(X_T, (sk_0, sk_0'))$
- Enc(pk) choose $r \stackrel{\$}{\leftarrow} \mathbf{Z}_q$. Compute $C = G^r$ and $K = X_T^r$; return (C, K)
- Dec1(C, sk_{i-1}) pick $t_i \stackrel{\$}{\leftarrow} \mathbf{Z}_q$, set $sk_i = sk_{i-1} \cdot G^{t_i}$, $Y_i = e(sk_i, C)$. Return (t_i, Y_i)
- Dec2$(sk_{i-1}', (t_i, Y_i), C)$ set $sk_i' = sk_{i-1}' \cdot G^{-t_i}$, and $Y_i' = e(sk_i', C)$. Return $K = Y_i \cdot Y_i' \in \mathbb{G}_T$

Security reduction in the Generic Bilinear Group Model if the leakage is **bounded in size**

Non-meaningful leakage model...

- $\mathrm{KG}(\kappa)$: choose $x, t_0 \xleftarrow{\$} \mathbf{Z}_q$. Set $X = g^x, sk_0 = g^{t_0}, sk_0' = g^{x-t_0}$, and $X_T = e(X, G)$. Return $(X_T, (sk_0, sk_0'))$
- $\mathrm{Enc}(pk)$ choose $r \xleftarrow{\$} \mathbf{Z}_q$. Compute $C = G^r$ and $K = X_T^r$; return (C, K)
- $\mathrm{Dec1}(C, sk_{i-1})$ pick $t_i \xleftarrow{\$} \mathbf{Z}_q$, set $sk_i = sk_{i-1} \cdot G^{t_i}$, $Y_i = e(sk_i, C)$. Return (t_i, Y_i)
- $\mathrm{Dec2}(sk_{i-1}', (t_i, Y_i), C)$ set $sk_i' = sk_{i-1}' \cdot G^{-t_i}$, and $Y_i' = e(sk_i', C)$. Return $K = Y_i \cdot Y_i' \in \mathbb{G}_T$

We did not get rid of exponentiations that place secret data on the exponent...

Scytl
Innovating Democracy

- $\mathrm{KG}(\kappa)$: choose $x, t_0 \xleftarrow{\$} \mathbf{Z}_q$. Set $X = g^x, sk_0 = g^{t_0}, sk_0' = g^{x-t_0}$, and $X_T = e(X, G)$. Return $(X_T, (sk_0, sk_0'))$
- $\mathrm{Enc}(pk)$ choose $r \xleftarrow{\$} \mathbf{Z}_q$. Compute $C = G^r$ and $K = X_T^r$; return (C, K)
- $\mathrm{Dec1}(C, sk_{i-1})$ pick $U_i \xleftarrow{\$} \mathbb{G}_1$, set $sk_i = sk_{i-1} \cdot U_i,\ Y_i = e(sk_i, C)$. Return (U_i, Y_i)
- $\mathrm{Dec2}(sk_{i-1}', (U_i, Y_i), C)$ set $sk_i' = sk_{i-1}' \cdot U^{-1}$, and $Y_i' = e(sk_i', C)$. Return $K = Y_i \cdot Y_i' \in \mathbb{G}_T$

Look, there is no need to exponentiate...

Second modification

- Computing random $u_i = g^{t_i}$ for $t_i \in \mathbb{F}_q$ leaks information on the fresh randomness used for decryption
- We do not know any exponentiation algorithm susceptible to meet the leakage bound
- We **do not need** knowledge of $t_i = \log_g u_i$
- We use an encoding $f : \mathbb{F}_p \mapsto E(\mathbb{F}_p)$ with good randomness preserving properties
- This encoding is **naturally** almost leakage-free

BEG-KEM+

- $\mathrm{KG}^{+}_{\mathrm{BEG}}(\kappa)$ choose $x, t_0 \xleftarrow{\$} \mathbf{Z}_q$. Set $X = g^x, sk_0 = g^{t_0}, sk'_0 = g^{x-t_0}$, and $X_T = e(X, G)^x$. Return $(X_T, (sk_0, sk'_0))$
- $\mathrm{Enc}^{+}_{\mathrm{BEG}}(pk)$ choose $r \xleftarrow{\$} \mathbf{Z}_q$, compute $C = G^r$ and $K = X_T^r$
- $\mathrm{Dec1}^{+}_{\mathrm{BEG}}(sk_{i-1}, C)$ choose $t_i, z_i \xleftarrow{\$} \mathbb{F}_p$, set $u_i = f(t_i) \cdot f(z_i)$, and compute $sk_i = sk_{i-1} \cdot u_i$ and $Y_i = e(sk_i, C)$. Return (u_i, Y_i)
- $\mathrm{Dec2}^{+}_{\mathrm{BEG}}(sk'_{i-1}, (u_i, Y_i), C)$ Set $sk'_i = sk'_{i-1} \cdot (u_i)^{-1}$ and $Y'_i = e(sk'_i, C)$. Return $K = Y_i \cdot Y'_i \in \mathbb{G}_T$

Require: A random number $t \in \mathbb{F}_p$

Ensure: Point $P \in E(\mathbb{F}_p)$

1: $w \leftarrow \sqrt{-3} \cdot t/(1 + b + t^2)$
2: $x_1 \leftarrow (-1 + \sqrt{-3})/2 - tw$
3: $x_2 \leftarrow -1 - x_1$
4: $x_3 \leftarrow 1 + 1/w^2$
5: $r_1, r_2, r_3 \xleftarrow{\$} \mathbb{F}_q^\star$
6: $\alpha \leftarrow \chi_p(r_1^2 \cdot (x_1^3 + b))$
7: $\beta \leftarrow \chi_p(r_2^2 \cdot (x_2^3 + b))$
8: $i \leftarrow [(\alpha - 1) \cdot \beta \mod 3] + 1$
9: **return** $P[x_i, \chi_p(r_3^2 \cdot t) \cdot \sqrt{(x_i^3 + b)}]$

- $p \equiv 3 \mod 4$
  $\chi_p(\cdot)$ is the Legendre symbol
- Use Extended Euclidean Algo to compute inverses as:
  $\frac{1}{x} = \frac{1}{x \cdot r} \cdot r$ for $r \xleftarrow{\$} \mathbb{F}_p$
- $\sqrt{x}$ for $x \in \mathbb{F}_p$ is computed as a fixed-exponent computation:
  $\sqrt{x} = x^{\frac{p+1}{4}}$

Require: A random number $t \in \mathbb{F}_p$

Ensure: Point $P \in E(\mathbb{F}_p)$

1: $w \leftarrow \sqrt{-3} \cdot t/(1 + b + t^2)$
2: $x_1 \leftarrow (-1 + \sqrt{-3})/2 - tw$
3: $x_2 \leftarrow -1 - x_1$
4: $x_3 \leftarrow 1 + 1/w^2$
5: $r_1, r_2, r_3 \xleftarrow{\$} \mathbb{F}_q^\star$
6: $\alpha \leftarrow \chi_p(r_1^2 \cdot (x_1^3 + b))$
7: $\beta \leftarrow \chi_p(r_2^2 \cdot (x_2^3 + b))$
8: $i \leftarrow [(\alpha - 1) \cdot \beta \mod 3] + 1$
9: **return** $P[x_i, \chi_p(r_3^2 \cdot t) \cdot \sqrt{(x_i^3 + b)}]$

- $p \equiv 3 \mod 4$
  $\chi_p(\cdot)$ is the Legendre symbol
- Use Extended Euclidean Algo to compute inverses as:
  $\frac{1}{x} = \frac{1}{x \cdot r} \cdot r$ for $r \xleftarrow{\$} \mathbb{F}_p$
- $\sqrt{x}$ for $x \in \mathbb{F}_p$ is computed as a fixed-exponent computation:
  $\sqrt{x} = x^{\frac{p+1}{4}}$

There are no branching instructions in the computation of the encoding

Pairings and Single Trace Attacks

Scytl
Innovating Democracy

- We present a security reduction in the Generic Bilinear Group Model if the leakage is **does not decrease** the min-entropy of the (intermediate) secret values “too much”...

Pairings and Single Trace Attacks

- We present a security reduction in the Generic Bilinear Group Model if the leakage is **does not decrease** the min-entropy of the (intermediate) secret values “too much”...

  par single trace!
- **Great bonus:** attacks that require multiple traces are **ruled out**
- Michael Scott in [Computing the Tate pairing, CT-RSA 2005] claims:

  > *"One might with reasonable confidence expect that the power consumption profile of (and execution time for) such protocols [against SPA attacks] will be constant and independent of any secret values."*

Pairings and Single Trace Attacks

[Unterluggauer-Wenger,ARES 2014] CPA attack with **1500 traces** in an ARM Cortex-M0 processor

[Ghosh-Roychowdhury,InfoSecHiComNet 2011] DPA attack with **2000 traces** in a Virtex-4 FPGA platform

no attacks known with single (or few) trace(s)!

Pairings and Single Trace Attacks

[Unterluggauer-Wenger,ARES 2014] CPA attack with **1500 traces** in an ARM Cortex-M0 processor

[Ghosh-Roychowdhury,InfoSecHiComNet 2011] DPA attack with **2000 traces** in a Virtex-4 FPGA platform

no attacks known with single (or few) trace(s)!

- “Intrinsically" more secure than e.g. exponentiation since the critical input data is a **secret group element** and not a secret scalar
- Operand-related SPA leakage from field-arithmetic operations is generally small (in large characteristic)

Implementation

- Barreto-Naehrig curve defined over a 254-bit prime field $\mathbb{F}_p$
- We implemented BEG-KEM+ in ANSI C
- MIRACL library for an efficient execution of the pairing evaluation
- Adruino Due microcontroller board with an ARM Cortex-M3 CPU

Table : Running times in 10^6 clock cycles

Operation	Time
Square root $\mathbb{F}_p$	0.7
Inversion $\mathbb{F}_p$	0.087
Encoding to $\mathbb{G}_2$	3.7
Exponentiation $\mathbb{G}_1$	4.5
Exponentiation $\mathbb{G}_2$	10.0
Exponentiation $\mathbb{G}_T$	27.1
Pairing	65.0

Table : Comparison of BEG-KEM and BEG-KEM+

Operation	BEG-KEM	BEG-KEM+
KeyGen	108	108
Encryption	34	34
Decryption	131	140

Conclusions

Scytl
Innovating Democracy

- We (would have liked to) contribute to bridge approaches for SCA resistance
  - SCA practice & countermeasures
  - provable security
- We provided a more reasonable leakage modeling
- We present a scheme and argue that it is susceptible to meet the leakage requirement
- We provided an implementation in an ARM Cortex-M3 processor
- Pairings have proven to be very useful in multiple contexts
  Maybe also for building SCA-resistant implementations?
- We continue exploring this approach

That s all folks! ☺