How to Run the Automata in Ciphertext/Key Space Under the Leakage Situation
(CCCS2014@GUANGZHOU)

Mingwu Zhang

† School of Computer Sciences, Hubei University of Technology

GUANGZHOU, CHINA,
June 20-23, 2014
csmwzhang(at)gmail.com

The idea of this work is from:
Mingwu Zhang. New model and construction of ABE: achieving key resilient-leakage and attribute direct-revocation. ACISP’14, LNCS 8544, pp. 192-208

Mingwu Zhang. ACP-IrFEM: Functional encryption mechanism with automatic control policy in the presence of key leakage. ISPEC’14, LNCS 8434, pp. 481-495
Functional Encryption (FE): Generalized Public-key Encryption

- User’s key is encoded with an expressive policy $\Psi$
- Message receiver is encoded with an input (attribute) vector $\vec{w}$
- The decryption will succeed iff relation $R(\Psi, \vec{w})$ holds

Instances of FE

- **IBE**: $\Psi := \{0, 1\}^*$, $\vec{w} := \{0, 1\}^*$, $R$ is defined as equality test;
- **ABE**: $\Psi$ is defined an access structure (formally by LSSS $(A, \rho)$), $\vec{w}$ is attribute set, $R$ is defined by fitting of $\vec{w}$ and an LSSS matrix;
- **HVE**: $\Psi$ is a vector over $\{0, 1, \ast\}$, and $\vec{w}$ is a vector over $\{0, 1\}$, and $R$ is defined by $\forall i, \Psi_i \neq \ast, w_i = \Psi_i$;
- **IPE**: $\Psi$ and $\vec{w}$ are all vectors, and $R : \langle \Psi, w \rangle = 0$;
- **(Spatial) SE**: $\Psi$ is an affine space, and $\vec{w}$ is a point vector, and $R : \vec{w} \in \Psi$.

Our FE

- $\Psi$ is defined by a deterministic finite automata $M$
- $\vec{w}$ is an input string with arbitrary length over $\{0, 1\}$
- $R$ is defined by $M$ accepting $\vec{w}$
Provable security in the presence of leakage

- Traditionally, the provable security in FE is implemented that the receiver attributes are divided into two disjoint parts: set-A for key extraction and set-B for challenge.
- Adversary can gain any key from set-A (non-match key) but none from set-B (match key), as the match key can successfully decrypt the challenge ciphertext.
- What is the situation if the match key is partially leaked?
Introduction & Motivation 3/8: Leakage model

1. **Bounded leakage**: Key leakage is obtained from any efficiently computable (leakage) function \( f : \text{SK} \rightarrow \{0, 1\}^L \) of the key.

2. **Key leakage fraction**: The leakage fraction \( \gamma \) is defined as the relative leakage of a key \( \text{SK} \), i.e., \( \gamma = \frac{L}{|\text{SK}|} \). Obviously, \( L \leq |\text{SK}| \).

3. **Leakage oracle**: Taking a shrink function \( f \) and a secret key \( \text{SK} \) as inputs, and output \( f(\text{SK}) \) if amount output of \( f(\text{SK}) \leq L \), and output \( \emptyset \) otherwise.

4. The adversary can gain at most \( L \)-bit for any match key, and total key for any non-match key.
Introduction & Motivation 4/8: Leakage model and leakage function

**General Leakage (any input-shrinking function)**

- **Devil** chooses input and produces output.
- **SK** at the end of the process.
Definition: Deterministic finite automata (DFA)

DFA is a finite state machine that accepts/rejects finite strings of symbols and produces a unique run of the automation for each input string, which is defined as a 5-tuple \((X, \Sigma, \delta, q_0, Y)\):

- A finite set of states \(X\);
- A finite set of input symbols called the alphabet \(\Sigma\);
- A transition function \(\delta: X \times \Sigma \rightarrow X\);
- A start state \(q_0 \in X\);
- A set of accept states \(Y \subseteq X\).

\[
\begin{align*}
X &= \{S_0, S_1, S_2\} \\
\Sigma &= \{0, 1\} \\
\delta: & \\
Q_0 &= S_0 \\
Y &= \{S_1, S_2\}
\end{align*}
\]

Email Filtering System is a typical application by DFA technique.
Procedure of DFA $\mathcal{M}$ for all input $w_1w_2\cdots w_n$

Let $\vec{w} = w_1w_2\cdots w_n$ be a string over alphabet $\Sigma$. A $\mathcal{M}$ accepts the string $\vec{w}$ if a sequence of states, $r_0, r_1, \cdots, r_n$, exists in $X$ with

1. $r_0 = q_0$: set start state $q_0$;
2. $r_{i+1} = \delta(r_i, w_{i+1})$, for $i = 0, 1, \cdots, n - 1$: given each character of string $\vec{w}$, the machine $\mathcal{M}$ will transition from a state $r_i$ to another state $r_{i+1}$ according to the transition function $\delta$ on the input $w_{i+1}$;
3. $r_n \in Y$: the machine $\mathcal{M}$ accepts $\vec{w}$ if $r_n \in Y$. We write the machine $\mathcal{M}$ accepts (resp. rejects) the string $\vec{w}$ by $\text{Accept}(\mathcal{M}, \vec{w}) = 1$ (resp. 0).

Definition: $\text{Accept}(\mathcal{M}, \vec{w})$

$$\text{Accept}(\mathcal{M}, \vec{w}) = \begin{cases} 1, & i = 1, \cdots, n - 1, r_{i+1} = \delta(r_i, w_{i+1}); r_n \in Y \\ 0, & \text{otherwise} \end{cases}$$

How can we run DFA automata in Ciphertext/Key spaces?
Algorithm of ACP-lrFEM

A leakage-resilient functional encryption from finite automatic policy (ACP-lrFEM) is comprised of the following five algorithms.

1. \((PP, MK) \leftarrow \text{SysGen}(1^\kappa, \Sigma, L)\): system initialization algorithm
2. \(SK_M \leftarrow \text{KeyExt}(MK, M)\): key extraction algorithm
3. \(SK'_M \leftarrow \text{KeyUpd}(SK_M, M)\): key update algorithm
4. \(CT_{\vec{w}} \leftarrow \text{Enc}(M, \vec{w})\): encryption algorithm
5. \(M / \bot \leftarrow \text{Dec}(CT_{\vec{w}}, SK_M)\): decryption algorithm

Consistency

\[
\Pr \left[ \begin{array}{l}
(PP, MK) \leftarrow \text{SysGen}(1^\kappa, \Sigma, L); \forall M, \vec{w}, \text{s.t. } \text{Accept}(M, \vec{w}) = 1;
SK_M \leftarrow \text{KeyExt}(MK, M); \forall i, f_i \in F, \sum_i f_i(SK_M) \leq L;
SK'_M \leftarrow \text{KeyUpd}(SK_M, M); \forall i, g_i \in F, \sum_i g_i(SK'_M) \leq L;
CT_{\vec{w}} \leftarrow \text{Enc}(M, \vec{w}); \text{Dec}(CT_{\vec{w}}, SK'_M) \neq M.
\end{array} \right] = \varepsilon(\kappa)
\]
PK vs SK

1. A user is identified by an exclusive public key PK.
2. $PK \leftrightarrow SK$? One PK corresponds ONE SK? e.g., $SK = x$, $PK = g^x$.
3. If the secret key $x$ is partially leaked, ElGamal encryption is not secure since the DL assumption does not exist in this case.
4. If one PK has many corresponding SK, then we can select different SK in different period and obtain security when previous key is partially leaked. e.g., we periodically update our PSW (SK) for an email account (PK).
5. (Almost) Leakage-resilient ElGamal Encryption: $SK = (x_1, x_2)$, $PK = g^{x_1} g^{x_2}$. Update: $SK = (x_1 + r, x_2 - r \log_{g_2} g_1)$.

Definition: Security in continual leakage

The continual leakage resilience is guaranteed by equipping with a key update algorithm KeyUpd that outputs a re-randomized key from the same distribution generated by a fresh call to KeyExt algorithm. The security will yield resilience to continual leakage “for free” in different period.
Idea for Construction 1/3: Functionalities of subgroups

Let $G$ be a bilinear group of order $N = p_1 p_2 p_3$

$\hat{e} : G \times G \rightarrow G_t$ is a bilinear map

Subgroups $G_{p_1}, G_{p_2}, G_{p_3}$

- orthogonal under $\hat{e}$, e.g. $\hat{e}(G_{p_1}, G_{p_2}) = 1$

$G_{p_1}$ = main scheme

$G_{p_2}$ = semi-functional space

$G_{p_3}$ = randomization for keys

- Subgroup Decision Assumption (SDA):
- Subgroup Orthogonality: to tolerate leakage and to provide full security
Lemma:\ Let $m, l, d \in \mathbb{Z}^+$, $2d \leq l \leq m$ and $p$ be a prime. Let $A_1 \leftarrow R \mathbb{Z}_{p}^{m \times l}$ and $A_2 \leftarrow R \mathbb{Z}_{p}^{m \times d}$, and $T \leftarrow R \text{Rank}_{d}(\mathbb{Z}_{p}^{l \times d})$ (i.e., the rank of matrix $T$ is $d$). For any transformation $f : \mathbb{Z}_{p}^{m \times d} \rightarrow \Omega$, there exists

$$\Delta((A_1, f(A_1 T)), (A_1, f(A_2))) \leq \varepsilon(\cdot)$$

$$|\Omega| \leq 4(1 - 1/p) \cdot p^{l-2d+1} \cdot \varepsilon(\cdot)^2$$

By setting $d = 1$, $l = m - 1$ and $m = \omega$ ($\omega \geq 2$), we have the following claim.

**Claim:** Let $W, S \leftarrow \mathbb{Z}_p^\omega$ and $S'$ be selected uniformly randomly from the set of vector in $\mathbb{Z}_p^\omega$ which are orthogonal to $W$ under the inner product modulo $p$, i.e., $\langle W, S' \rangle \mod p = 0$. For any transformation $f : \mathbb{Z}_p^\omega \rightarrow \Omega$, then

$$\Delta((W, f(S)), (W, f(S'))) \leq \varepsilon(\cdot),$$

as long as $|\Omega| \leq 4p^{\omega - 3}(p - 1) \cdot \varepsilon(\cdot)^2$.

- This claim declares that: **given a vector $W$, it is hard to distinguish its (partial) orthogonal vector from a random one.**
- What is the relation between the input and the output $f(\cdot)$?
The Construction 1/5: SysGen(1\(^{\kappa}\), \(\Sigma\), \(L\))

1. \((p_1, p_2, p_3, G, \mathcal{H}, e) \leftarrow \text{GCP}(\kappa)\), where \(p_1, p_2\) and \(p_3\) are distinct primes.
2. Set subgroups \(G_1 = \langle P_1 \rangle\), \(G_2 = \langle P_2 \rangle\) and \(G_3 = \langle P_3 \rangle\) of orders \(p_1, p_2\) and \(p_3\) respectively.
3. Define \(G = G_1 \times G_2 \times G_3\) and set \(N = p_1p_2p_3\).
4. Select \(\tau \in \mathbb{R}^+\) such that \(\varepsilon = p_2^{-\tau}\) is negligible in \(\kappa\), and set \(\omega = \lceil 1 + 2\tau + L/|p_2| \rceil\).
5. Set system public key

\[
\text{PP} = \langle \Theta, Z, P_1, P_3, H_{st}, H_{end}, \forall \sigma \in \Sigma \ H_{\sigma}, (\overline{\alpha}_iP_1)_{i \in [\omega]}, (T_i)_{i \in \Sigma}, e(P_1, P_1)^{\gamma} \rangle
\]

6. Keep the master key \(\text{MK} = \langle (\overline{\beta}_iP_1)_{i \in [\omega]}, (\gamma + \langle \overline{\alpha}, \overline{\beta} \rangle)P_1 \rangle\)

In the setting, the parameter \(\omega\), mainly decided by \(L\), can be varied to achieve desired key leakage and size of keys/ciphertexts. Obviously, the larger \(L\), the larger of keys and ciphertexts.
The Construction 2/5: \( \text{Enc}(M, \vec{w} = w_1 w_2 \cdots w_n) \)

For encrypting a plaintext \( M \) to the receiver with designated input string \( \vec{w} = w_1 w_2 \cdots w_n \), the algorithm at random picks \( s_0, s_1, \cdots, s_n \in \mathbb{Z}_N \), and outputs the ciphertext

\[
\text{CT}_{\vec{w}} = \begin{pmatrix}
\vec{w}, & C_m \\
C_{st,1}, & C_{st,2} \\
C_{1,1}, & C_{1,2} \\
C_{2,1}, & C_{2,2} \\
\vdots & \vdots \\
C_{n,1}, & C_{n,2} \\
C_{\text{end},1}, & C_{\text{end},2} \\
\vec{C}_{\text{fin}}
\end{pmatrix} = \begin{pmatrix}
\vec{w}, & M \cdot e(P_1, P_1)^{ys_n} \\
s_0 P_1, & s_0 H_{st} \\
s_1 P_1, & s_1 H_{w_1} + s_0 Z \\
s_2 P_1, & s_2 H_{w_2} + s_1 Z \\
\vdots & \vdots \\
s_n P_1, & s_n H_{w_n} + s_{n-1} Z \\
s_n P_1, & s_n H_{\text{end}} \\
(\vec{C}_i^{s_n} P_1)_{i \in [\omega]} &
\end{pmatrix}
\]

- The ciphertext forms a chain-based structure and only the input with sequence \( w_1, w_2, \cdots, w_n \) can remove \( s_i \) for \( i = 1, \cdots, n - 1 \).
- \( \vec{C}_{\text{fin}} \) is to tolerate the leakage in orthogonal subgroup \( G_2 \times G_3 \) in the key.
- All components stay in \( G_1 \) except \( C_m \).
The Construction 3/5: \text{KeyExt}(\mathcal{MK}, \mathcal{M} = (X, \Sigma, q_0, \delta, Y))

1. At random select $D_0, D_1, \cdots, D_{|X|-1} \in G_1$, and encode each $q_i$ with $D_i$.

2. Choose $r_{st}, r_{end} \in \mathbb{Z}_N$ and $R_{st,1}, R_{st,2}, R_{end,1}, R_{end,2} \in G_3$ randomly, and for each transition $t \in T$ ($t$ is defined by a triple $(q_x, q_y, \delta) \in X \times X \times \Sigma$) select $r_t \in \mathbb{Z}_N$.

3. For $t \in T$, select $R_{t,1}, R_{t,2}, R_{t,3} \in G_3$ randomly.

4. Encode start state: $K_{st,1} = D_0 + r_{st} H_{st} + R_{st,1}$, $K_{st,2} = r_{st} P_1 + R_{st,2}$.

5. Encode the DFA: For all $t \in T$ with $t = (q_x, q_y, \delta)$, calculate $K_{t,1} = -D_x + r_t Z + R_{t,1}$, $K_{t,2} = r_t P_1 + R_{t,2}$ and $K_{t,3} = D_y + r_t H_\sigma + R_{t,3}$.

6. Encode the end state: For each $q_y \in Y$, at random select $r_{end,y} \in \mathbb{Z}_N$, $R_{end,y,1}, R_{end,y,2} \in G_3$, and calculate $K_{end,y,1} = (y + \langle \tilde{\alpha}, \tilde{\beta} \rangle) P_1 + D_y + r_{end,y} H_{end} + R_{end,y,1}$, and $K_{end,y,2} = r_{end,y} P_1 + R_{end,y,2}$.

7. Encode the leakage: For $i \in [\omega]$ at random pick $R_{fin,i} \in G_3$, and calculate $K_{fin,i} = \tilde{\beta}_i P_1 + R_{fin,i}$

\[
\text{SK}_\mathcal{M} = \langle (K_{st,1}, K_{st,2}), (K_{t,1}, K_{t,2}, K_{t,3})_{t \in T}, (K_{end,y,1}, K_{end,y,2})_{q_y \in Y}, (K_{fin,i})_{i \in [\omega]} \rangle
\]

start state DFA chain end state leakage resilience
The Construction 4/5: KeyUpd($SK_M, M = (X, \Sigma, q_0, \delta, Y)$)

1. Pick $r'_{st} \in \mathbb{Z}_N$ and $R'_{st,1}, R'_{st,2} \in G_3$, and update the start part key:
   
   \[ K'_{st,1} = K_{st,1} + r'_{st} H_{st} + R'_{st,1}, \quad K'_{st,2} = K_{st,2} + r'_{st} P_1 + R'_{st,2}. \]

2. For all $t \in T$ with $t = (q_x, q_y, \delta)$, select $r'_t \in \mathbb{Z}_N$, $R'_{t,1}, R'_{t,2}, R'_{t,3} \in G_3$, and update the transition key:
   
   \[ \forall t \in T \text{ with } t = (q_x, q_y, \delta), \text{ set } \]
   \[ K'_{t,1} = K_{t,1} + r'_t Z + R'_{t,1}, \quad K'_{t,2} = K_{t,2} + r'_t P_1 + R'_{t,2}, \]
   \[ K'_{t,3} = K_{t,3} + r'_t H_{\sigma} + R'_{t,3}. \]

3. For each $q_y \in Y$, select $r'_{end,y} \in \mathbb{Z}_N$, $R'_{end,y,1}, R'_{end,y,2} \in G_3$, and update the end part key:
   
   \[ K'_{end,y,1} = K_{end,y,1} + r'_{end,y} H_{end} + R'_{end,y,1}, \quad \]
   \[ K'_{end,y,2} = K_{end,y,2} + r'_{end,y} P_1 + R'_{end,y,2}. \]

4. Select $R_{fin,i} \in G_3$ for $i \in [\omega]$, update final part key:
   
   \[ K'_{fin,i} = K_{fin,i} + R'_{fin,i}. \]

5. Delete $SK_M$ and output the new key

- Only explicitly update the randomness in the key, and the new key has the same distribution.
- Achieve the resilience ability of key continual leakage in difference period.
- Update the key when the entropy loss of that key will draw near the threshold.
The Construction 5/5: Dec(CT_{\vec{w}}, SK_M)

1. At first calculate the initialization state:
\[
B_0 = \frac{e(C_{st,1}, K_{st,1})}{e(C_{st,2}, K_{st,2})} = e(D_0, C_{st,1}) = e(D_0, P_1)^{s_0}
\]

2. For \(i = 1\) to \(n\), calculate iteratively (perform the automata in cipertext/key space):
\[
B_i = B_{i-1} \cdot \frac{e(C_{i-1,1}, K_{t_i,1})e(C_{i,1}, K_{t_i,3})}{e(C_{i,2}, K_{t_i,2})} = e(D_{u_i}, C_{i,1}) = e(D_{u_i}, P_1)^{s_i}
\]

3. Check the end state As the automata \(\mathcal{M}\) accepts the string \(\vec{w}\), then the last state \(u_n\) must halt in \(Y\). That is, \(u_n = q_y\) for some \(q_y \in Y\) and \(B_n = e(D_y, P_1)^{s_n}\). Calculate:
\[
B_{end} = B_n \cdot \frac{e(C_{end,2}, K_{end_y,2})}{e(C_{end,1}, K_{end_y,1})} = e(P_1, P_1)^{-s_n(y + \langle \vec{\alpha}, \vec{\beta} \rangle)}
\]

4. Calculate \(B_{fin} = B_{end} \cdot e_n(\vec{C}_{fin}, \vec{K}_{fin}) = e(P_1, P_1)^{-y_{sn}}\)

5. Extract the plaintext from \(C_m\) by \(M \leftarrow C_m \cdot B_{fin} \).
## Security intuition: Indistinguishable games

<table>
<thead>
<tr>
<th>Games</th>
<th>Functionalities</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Λ₀</td>
<td>Real experiment</td>
<td>The keys and challenge ciphertext are all normal (w.o ( G_2 ))</td>
</tr>
<tr>
<td>Λ₁</td>
<td>update oracle replace by extraction oracle</td>
<td></td>
</tr>
<tr>
<td>Λ₂</td>
<td>Challenge ciphertext is converted to semi-functional</td>
<td></td>
</tr>
<tr>
<td>Λ₃,ₖ</td>
<td>First ( k-1 ) keys are type-2’s, ( k^{th} ) is type-1’s, and the rest keys are normal</td>
<td>( \Omega_E ) for non-match key, ( 1 \leq k \leq Q )</td>
</tr>
<tr>
<td>Λ₄,ₖ</td>
<td>The ( k^{th} ) is converted to nominal form</td>
<td>( \Omega_L ): Under the leakage of match key, the semi-functional key cannot convert into a nominal one</td>
</tr>
<tr>
<td>Λ₅</td>
<td>( C_0 ) is replaced by a random element from ( \mathcal{H} )</td>
<td>All components in keys and ciphertexts are semi-functional except ( C_m ) (( C_m ) is randomized)</td>
</tr>
</tbody>
</table>
Leakage bound:
\[ \omega = \lceil 1 + 2\tau + \frac{L}{|p_2|} \rceil \approx 1 + \frac{L}{|p_2|} \Rightarrow L = (\omega - 1)|p_2| \]

2. \(|p_1| = |p_2| = |p_3|, |N| = 3|p_2|, \)
   \# of the ciphertext \(|H| + (2n + 4 + \omega)|G| = (6n + 18 + 3\omega)|p_2|, \)
   \# of key \(3(2 + |T| + Q + \omega)|p_2| \)

Leakage fraction:
\[ \gamma = \frac{L}{|SK|} = \frac{\omega - 1}{3(2 + |T| + Q + \omega)} \approx \frac{1}{3(1 + \frac{|T| + Q}{\omega})} = \frac{1}{3} \]

Remarks
- If \(\omega = 1\), then \(L = 0\) and \(\tau = 0\), which means that the scheme is reduced to a non-leakage resilient encryption.
- The larger \(\omega\), the better leakage-resilience.
- The subgroup \(G_2\) has two functionalities: the hidden intractable order \(p_2\) from group \(G\) for adaptive security proof and the subspace orthogonality for leakage tolerance.
Performance and Application 2/3: Comparison

<table>
<thead>
<tr>
<th>scheme</th>
<th>Wat12</th>
<th>Ram13</th>
<th>ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>order</td>
<td>prime</td>
<td>composite</td>
<td>composite</td>
</tr>
<tr>
<td># accept states</td>
<td>≥ 1</td>
<td>1</td>
<td>≥ 1</td>
</tr>
<tr>
<td># of CT</td>
<td>6 + 2n</td>
<td>6 + 2n</td>
<td>6 + 2n + ω</td>
</tr>
<tr>
<td># of SK</td>
<td>2 + 3</td>
<td>T</td>
<td>+ 2</td>
</tr>
<tr>
<td>leakage fraction</td>
<td>∅</td>
<td>∅</td>
<td>33%</td>
</tr>
</tbody>
</table>

Using the transformation technique in Lew12\(^a\) and Fre10\(^b\), we can construct the scheme in prime order groups with cancelling map property.

\(^a\)Lewko A. “Tools for simulating features of composite order bilinear groups in the prime order setting”. EUROCRYPT’12, pp. 318-335 (2012)

\(^b\)Freeman D M. “Converting pairing-based cryptosystems from composite-order groups to prime-order groups.” EUROCRYPT’10, LNCS, pp. 44-61 (2010)
Performance and Application 3/3: Application in email filtering firewall

EMAIL FILTERING CHECK by DFA

(a)

KeyExt
Enc
Dec
EMAIL FILTERING
KeyUpd
SysGen
Filtering Rule
Filtering Rule

(b)
Our future research

- Implement the scheme in Dual System Groups [ePrint 2014/265] with weak orthogonality?
- Unconditionally secure from Orthogonal Vector Problem of leakage resilience [ePrint 2014/282]?
  **Maybe lattice will provide an alternative solution.**
- Run Turing Machine in Ciphertext/Key Spaces?

---

a Jie Chen and Hoeteck Wee, Dual System Groups and its Applications – Compact HIBE and More. ePrint 2014/265

b Ivan Damgård, Frédéric Dupuis, and Jesper Buus Nielsen, On The Orthogonal Vector Problem and The Feasibility of Unconditionally Secure Leakage Resilient Computation. ePrint 2014/282

What types of leakage?

- Randomness leakage (controlled PRF by the attacker)
- Unbounded leakage (captured the leakage by virus)
- Intermediate result leakage (memory/register monitor by the eavesdropper)
Thank the organizer of CCCS2014!

Thank you very much for your attention!

Questions & Comments?