Applications of the Dual System Encryption Approach to Threshold Cryptography

Benoît Libert\(^1\) and Moti Yung\(^2\)

\(^1\)Université catholique de Louvain, Crypto Group – F.N.R.S.
\(^2\)Google Inc. and Columbia University

June 30, 2011

Dagstuhl Seminar
Threshold Cryptography

- Introduced by Desmedt-Frankel (Crypto’89) and Boyd (IMA’89)
- Split private keys into \( n \) shares \( SK_1, \ldots, SK_n \) so that knowing strictly less than \( t \leq n \) shares is useless to the adversary.
- At least \( t \leq n \) shareholders must contribute to private key operations.
  - Decryption requires the cooperation of \( t \) decryption servers.
  - Signing requires at least \( t \) servers to run a joint signing protocol.
- **Robustness**: up to \( t - 1 \leq n \) malicious servers cannot prevent a honest majority from decrypting/signing.
Static vs Adaptive corruptions

- **Static corruptions**: adversary corrupts servers *before* seeing the public key.

Robust threshold cryptosystems with IND-CCA2 security:

- Shoup-Gennaro (Eurocrypt’98): in the ROM.
- Canetti-Goldwasser (Eurocrypt’99): requires interaction or storage of many pre-shared secrets; robust and adaptively secure for \( t = O(n^{1/2}) \).
- Dodis-Katz (TCC’05): generic constructions; ciphertexts of size \( O(n) \).
- Boneh-Boyen-Halevi (CT-RSA’06): no interaction needed for robustness.
- Wee (Eurocrypt’11): generic constructions from (threshold) extractable hash proof systems.
Static vs Adaptive corruptions

- Adaptive corruptions: adversary corrupts up to $t - 1$ servers \textit{at any time}.
  
  - Canetti \textit{et al.} (Crypto’99): shares refreshed after each private key operations.
  
  
  - Jarecki-Lysyanskaya (Eurocrypt’00): no need for erasures, but interaction.
  
  - Lysyanskaya-Peikert (Asiacrypt’01): adaptively secure signatures with interaction.
  
  - Abe-Fehr (Crypto’04): adaptively secure UC-secure threshold signatures and encryption with interaction.
Despite more than 10 years of research, adaptive security has not been achieved with:

- CCA2-security for encryption and CMA-security for signatures.
- Non-interactive schemes
- Robustness against malicious adversaries
- Optimal resilience \( t = (n - 1)/2 \)
- No erasures for shareholders
- Share size independent of \( t, n \)
- Proof in the standard model
CCA2-Secure Non-interactive Threshold Encryption

Our contribution (ICALP'11):

- The first adaptively secure non-interactive threshold cryptosystem providing
  - CCA2 security and robustness w/o random oracles
  - Short (i.e., $O(1)$-size) private key shares

The construction

- Builds on the dual system encryption approach (Waters, Crypto’09) and the Lewko-Waters techniques (TCC’10).
- Handles adaptive corruptions by instantiating Boneh-Boyen-Halevi (CT-RSA’06) in bilinear groups of order $N = p_1p_2p_3$.

- Gives adaptively secure non-interactive threshold signatures
CCA2-Secure Non-interactive Threshold Encryption

An alternative approach:

- **Combination between**
  
  - Universal hash proofs (simulator knows private keys in reduction).
  
  - Simulation-sound proofs of ciphertext validity.

- **Gives constructions in prime order groups**
  
  - Based on Groth-Sahai proofs (Decision Linear or Symmetric eXternal Diffie-Hellman assumptions).

  - Or, more efficiently, based on DDH and the random oracle model.

  ⇒ Gives an adaptively secure version of the Shoup-Gennaro system.
Chosen-ciphertext (IND-CCA) security:

1. Challenger generates \( PK, \ SK = (SK_1, \ldots, SK_n) \) and gives \( PK \) to \( A \).

2. \( A \) makes adaptive queries
   - Corruption \( i \in \{1, \ldots, n\} \): \( A \) receives \( SK_i \) (up to \( t - 1 \) queries allowed).
   - Decryption \((i, C)\): \( A \) receives \( \mu_i = \text{Share-Decrypt}(PK, i, SK_i, C) \)

3. \( A \) chooses \( M_0, M_1 \) and gets \( C^* = \text{Encrypt}(PK, M_\beta) \) for some \( \beta \overset{R}{\leftarrow} \{0, 1\} \).

4. \( A \) makes further queries with restrictions.

5. \( A \) outputs \( \beta' \in \{0, 1\} \) and wins if \( \beta' = \beta \).
Security of Non-interactive Threshold Encryption

Chosen-ciphertext (IND-CCA) security:

1. Challenger generates $PK$, $SK = (SK_1, \ldots, SK_n)$ and gives $PK$ to $A$.
2. $A$ makes adaptive queries
   - Corruption $i \in \{1, \ldots, n\}$: $A$ receives $SK_i$ (up to $t - 1$ queries allowed).
   - Decryption $(i, C)$: $A$ receives $\mu_i = \text{Share-Decrypt}(PK, i, SK_i, C)$
3. $A$ chooses $M_0, M_1$ and gets $C^* = \text{Encrypt}(PK, M_\beta)$ for some $\beta \overset{R}{\leftarrow} \{0, 1\}$.
4. $A$ makes further queries with restrictions.
5. $A$ outputs $\beta' \in \{0, 1\}$ and wins if $\beta' = \beta$.
Chosen-ciphertext (IND-CCA) security:

1. Challenger generates $PK, SK = (SK_1, \ldots, SK_n)$ and gives $PK$ to $A$.
2. $A$ makes adaptive queries
   - Corruption $i \in \{1, \ldots, n\}$: $A$ receives $SK_i$ (up to $t-1$ queries allowed).
   - Decryption $(i, C)$: $A$ receives $\mu_i = \text{Share-Decrypt}(PK, i, SK_i, C)$
3. $A$ chooses $M_0, M_1$ and gets $C^* = \text{Encrypt}(PK, M_\beta)$ for some $\beta \xleftarrow{R} \{0, 1\}$.
4. $A$ makes further queries with restrictions.
5. $A$ outputs $\beta' \in \{0, 1\}$ and wins if $\beta' = \beta$. 
Security of Non-interactive Threshold Encryption

- Chosen-ciphertext (IND-CCA) security:

1. Challenger generates $PK, SK = (SK_1, \ldots, SK_n)$ and gives $PK$ to $A$.
2. $A$ makes adaptive queries
   - Corruption $i \in \{1, \ldots, n\}$: $A$ receives $SK_i$ (up to $t-1$ queries allowed).
   - Decryption $(i, C)$: $A$ receives $\mu_i = \text{Share-Decrypt}(PK, i, SK_i, C)$
3. $A$ chooses $M_0, M_1$ and gets $C^* = \text{Encrypt}(PK, M_\beta)$ for some $\beta \overset{R}{\leftarrow} \{0, 1\}$.
4. $A$ makes further queries with restrictions.
5. $A$ outputs $\beta' \in \{0, 1\}$ and wins if $\beta' = \beta$. 
Security of Non-interactive Threshold Encryption

- Chosen-ciphertext (IND-CCA) security:

1. Challenger generates \( PK, SK = (SK_1, \ldots, SK_n) \) and gives \( PK \) to \( A \).

2. \( A \) makes adaptive queries
   - Corruption \( i \in \{1, \ldots, n\} \): \( A \) receives \( SK_i \) (up to \( t - 1 \) queries allowed).
   - Decryption \( (i, C) \): \( A \) receives \( \mu_i = \text{Share-Decrypt}(PK, i, SK_i, C) \)

3. \( A \) chooses \( M_0, M_1 \) and gets \( C^* = \text{Encrypt}(PK, M_\beta) \) for some \( \beta \overset{R}{\leftarrow} \{0, 1\} \).

4. \( A \) makes further queries with restrictions.

5. \( A \) outputs \( \beta' \in \{0, 1\} \) and wins if \( \beta' = \beta \).
Consistency:

1. Challenger generates $PK, SK = (SK_1, \ldots, SK_n)$ and gives $PK$ to $A$.

2. $A$ makes adaptive queries
   - Corruption query $i \in \{1, \ldots, n\}$: $A$ receives $SK_i$.
   - Decryption query $(i, C)$: $A$ receives $\mu_i = \text{Share-Decrypt}(PK, i, SK_i, C)$

3. $A$ outputs a ciphertext $C$ and sets $S = \{\mu_1, \ldots, \mu_t\}, S' = \{\mu'_1, \ldots, \mu'_t\}$ of shares such that
   - $C$ is a valid ciphertext.
   - $S$ and $S'$ are sets of valid shares.
   - $\text{Combine}(PK, C, S) \neq \text{Combine}(PK, C, S')$. 
Construction: Intuition

- Construction based on groups \((G, G_T)\) of order \(N = p_1p_2p_3\) with a bilinear map (a.k.a. pairing) \(e : G \times G \rightarrow G_T:\)

\[ e(g_i, g_j) = 1_{G_T} \text{ for any } g_i \in G_{p_i} \text{ and } g_j \in G_{p_j} \text{ s.t. } i \neq j. \]

- Applies dual system encryption techniques (Waters, Crypto’09).
  - Use of semi-functional ciphertexts and decryption shares (i.e., with \(G_{p_2}\) components)

- Applies the Canetti-Halevi-Katz transform (Eurocrypt’04) to the Lewko-Waters IBE (TCC’10)

- …with additional tricks to deal with adaptive corruptions
Construction

- **Keygen**$(\lambda, t, n) : (1 \leq t \leq n)$

  - Choose bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of order $N = p_1 p_2 p_3$ and group elements $g, h, u, v \overset{R}{\leftarrow} \mathbb{G}_{p_1}$, $X_3 \overset{R}{\leftarrow} \mathbb{G}_{p_3}$.
  
  - Choose a polynomial $P[X] = \alpha + \alpha_1 X + \cdots + \alpha_{t-1} X^{t-1} \overset{R}{\leftarrow} \mathbb{Z}_N[X]$.
  
  - Choose a collision-resistant hash function $H : \{0, 1\}^* \rightarrow \mathbb{Z}_N$.

  - Define private key shares $\{SK_i = h^{P(i)} \cdot Z_{3,i}\}_{i=1}^n$ with $Z_{3,i} \overset{R}{\leftarrow} \mathbb{G}_{p_3}$ and set

    $$PK = \left( g, h, u, v \in \mathbb{G}_{p_1}; \ X_3 \in \mathbb{G}_{p_3}; \ e(g, h)^\alpha, \ \{VK_i = e(g, h)^{P(i)}\}_{i=1}^n; \ H \right).$$

- **Encrypt**$(PK, M)$ : choose $s \overset{R}{\leftarrow} \mathbb{Z}_N$ and compute

  $$(C_0, C_1, C_2) = \left( M \cdot e(g, h)^{\alpha \cdot s}, \ g^s, (u^\theta \cdot v)^s \right)$$

  with $\theta = H(C_0, C_1) \in \mathbb{Z}_N$. 
Construction

- **Keygen(λ, t, n)**: $(1 \leq t \leq n)$
  - Choose bilinear groups $(G, \hat{G})$ of order $N = p_1 p_2 p_3$ and group elements $g, h, u, v \leftarrow \hat{G}_{p_1}$, $X_3 \leftarrow \hat{G}_{p_3}$.
  - Choose a polynomial $P[X] = \alpha + \alpha_1 X + \cdots + \alpha_{t-1} X^{t-1} \leftarrow \mathbb{Z}_N[X]$. 
  - Choose a collision-resistant hash function $H : \{0, 1\}^* \rightarrow \mathbb{Z}_N$.
  - Define private key shares $\{SK_i = h^{P(i)} \cdot Z_{3,i}\}_{i=1}^n$ with $Z_{3,i} \leftarrow \hat{G}_{p_3}$ and set
    $$PK = \left( g, h, u, v \in \hat{G}_{p_1}; \ X_3 \in \hat{G}_{p_3}; \ e(g, h)^\alpha, \ \{VK_i = e(g, h)^{P(i)}\}_{i=1}^n; \ H \right).$$

- **Encrypt**(PK, M): choose $s \leftarrow \mathbb{Z}_N$ and compute
  $$(C_0, C_1, C_2) = (M \cdot e(g, h)^{\alpha \cdot s}, \ g^s, \ (u^\theta \cdot v)^s)$$
  with $\theta = H(C_0, C_1) \in \mathbb{Z}_N$. 

Construction

- **Share-Decrypt** \((PK, i, SK_i, C)\): parse \(C\) as
  \[
  (C_0, C_1, C_2) = (M \cdot e(g, h)^{\alpha \cdot s}, g^s, (u^\theta \cdot v)^s).
  \]
  Return \(\bot\) if \(e(C_1, u^\theta \cdot v) \neq e(g, C_2)\) or if \(C_1\) or \(C_2\) has a \(\mathbb{G}_{p_3}\) component.
  Otherwise, pick \(r \leftarrow \mathbb{Z}_N\) \(R_{i,1}, R_{i,2} \leftarrow \mathbb{G}_{p_3}\) and return
  \[
  \mu_i = (D_{i,1}, D_{i,2}) = (SK_i \cdot (u^\theta \cdot v)^r \cdot R_{i,1}, g^r \cdot R_{i,2}).
  \]

- **Combine** \((PK, C, \{\mu_i\}_{i \in S})\): parses \(\mu_i\) as \((D_{i,1}, D_{i,2}) \in \mathbb{G}_{p_1 p_3}\) for each \(i\).
  Check that \(Share-Veify(PK, \mu_i) = 1\) for each \(i\). Compute
  \[
  (D_1, D_2) = (\prod_{i \in S} D_{i,1}^{\Delta_i, s(0)}, \prod_{i \in S} D_{i,2}^{\Delta_i, s(0)}) = (h^\alpha \cdot (u^\theta \cdot v)^\tilde{r} \cdot \tilde{R}_1, g^\tilde{r} \cdot \tilde{R}_2)
  \]
  Return
  \[
  M = C_0 \cdot e(C_1, D_1)^{-1} \cdot e(C_2, D_2).
  \]
Construction

- \textit{Share-Decrypt}(PK, i, SK_i, C) : parse C as

\[(C_0, C_1, C_2) = (M \cdot e(g, h)^{\alpha \cdot s}, g^s, (u^\theta \cdot v)^s).\]

- Return ⊥ if e(C_1, u^\theta \cdot v) \neq e(g, C_2) or if C_1 or C_2 has a \( \mathbb{G}_{p_3} \) component.

- Otherwise, pick \( r \overset{R}{\leftarrow} \mathbb{Z}_N \), \( R_{i,1}, R_{i,2} \overset{R}{\leftarrow} \mathbb{G}_{p_3} \) and return

\[\mu_i = (D_{i,1}, D_{i,2}) = (SK_i \cdot (u^\theta \cdot v)^r \cdot R_{i,1}, g^r \cdot R_{i,2}).\]

- \textit{Combine}(PK, C, \{\mu_i\}_{i \in S}) : parses \( \mu_i \) as \((D_{i,1}, D_{i,2}) \in \mathbb{G}_{p_1 p_3}\) for each \( i \). Check that \textit{Share-Verify}(PK, \mu_i) = 1 for each \( i \). Compute

\[(D_1, D_2) = (\prod_{i \in S} D_{i,1}^{\Delta_i, s(0)}, \prod_{i \in S} D_{i,2}^{\Delta_i, s(0)}) = (h^\alpha \cdot (u^\theta \cdot v)^{\tilde{r}} \cdot \tilde{R}_1, g^{\tilde{r}} \cdot \tilde{R}_2)\]

Return

\[M = C_0 \cdot e(C_1, D_1)^{-1} \cdot e(C_2, D_2).\]
Security

Theorem

The scheme provides **IND-CCA** security under adaptive corruptions if

- $H : \{0, 1\}^* \rightarrow \mathbb{Z}_N$ is a collision-resistant hash function.

- Assumptions 1, 2 and 3 hold in $(G, G_T)$.

1. Given $(g \in G_{p_1}, X_3 \in G_{p_3})$, $T \in_R G_{p_1}$ and $T \in_R G_{p_1p_2}$ are indistinguishable.

2. Given $(g \in G_{p_1}, X_3 \in G_{p_3}, X_1 X_2 \in G_{p_1p_2}, Y_2 Y_3 \in G_{p_2p_3})$, $T \in_R G_{p_1p_3}$ and $T \in_R G$ are indistinguishable.

3. Given $(g \in G_{p_1}, Z_2 \in G_{p_2}, X_3 \in G_{p_3}, g^s X_2 \in G_{p_1p_2}, g^\alpha Y_2 \in G_{p_1p_2})$, no PPT can distinguish $T = e(g, g)^{\alpha \cdot s} \in G_{T,p_1}$ from $T \in_R G_T$.

Theorem

The scheme provides **consistency** if Assumption 1 holds in $G$. 
The proof

- Uses semi-functional ciphertext and decryption shares (i.e., that contain $G_{p_2}$ components).

- Differences with Lewko-Waters (TCC’10):
  - Two kinds of semi-functional ciphertexts (with or without $G_{T,p_2}$ component)
  - Also uses semi-functional *private key shares* $SK_1, \ldots, SK_n$
  - Additional step needed in the proof
The proof in the IND-CPA case

Ciphertext is \((C_0, C_1) = (M \cdot e(g, h)^{\alpha \cdot s}, g^s)\):

- Uses two kinds of semi-functional ciphertexts with \(\tau, \theta \leftarrow \mathbb{Z}_{p_2}\)
  - Type I: \(C_0 = M^\beta \cdot e(g, h)^{\alpha \cdot s}, \quad C_1 = g^s \cdot g^\tau\).
  - Type II: \(C_0 = M^\beta \cdot e(g, h)^{\alpha \cdot s} \cdot e(g_2, g_2)^\theta, \quad C_1 = g^s \cdot g_2^\tau\).

- ... and semi-functional key shares: \(SK_i = h^{P(i)} \cdot R_{2,i} \cdot R_{3,i}\), with \(R_{2,i} \leftarrow \mathbb{G}_{p_2}\).

- Uses a sequence of games where Game_0 is the real game
  - Game_1: ciphertext is a Type I semi-functional ciphertext.
  - Game_2: — Ciphertext is a Type II semi-functional ciphertext
    — All private key shares are semi-functional.
  - Game_3: ciphertext is a Type II encryption of a random \(M \leftarrow \mathbb{G}_T\).
The proof in the IND-CPA case

Ciphertext is \((C_0, C_1) = (M \cdot e(g, h)^{\alpha \cdot s}, g^s)\):

- Uses two kinds of semi-functional ciphertexts with \(\tau, \theta \leftarrow \mathbb{Z}_{p_2}\)
  - Type I: \(C_0 = M_\beta \cdot e(g, h)^{\alpha \cdot s}, \quad C_1 = g^s \cdot g_2^\tau\).
  - Type II: \(C_0 = M_\beta \cdot e(g, h)^{\alpha \cdot s} \cdot e(g_2, g_2)^\theta, \quad C_1 = g^s \cdot g_2^\tau\).

- ... and semi-functional key shares: \(SK_i = h^{P(i)} \cdot R_{2,i} \cdot R_{3,i}\), with \(R_{2,i} \leftarrow \mathbb{G}_{p_2}\).

- Uses a sequence of games where Game_0 is the real game
  - Game_1: ciphertext is a Type I semi-functional ciphertext.
  - Game_2: ciphertext is a Type II semi-functional ciphertext
    - All private key shares are semi-functional.
  - Game_3: ciphertext is a Type II encryption of a random \(M \leftarrow \mathbb{G}_T\).
Extensions

- Robust and adaptively secure non-interactive threshold signatures in the standard model.

- Forward-security in threshold cryptography: past uses of private keys remain safe even after $t$ corruptions (Abdalla-Miner-Namprempre, CT-RSA’01)
  - Forward-secure threshold encryption (with short ciphertexts from the Lewko-Waters HIBE)
  - Forward-secure threshold signatures (by thresholdizing Boyen-Shacham-Shen-Waters, CCS’06)

...all with adaptive security in the standard model and w/o interaction for signing/decrypting or updates.
An alternative approach

Builds on Groth-Sahai proofs and the Decision Linear assumption:

- **Linear Problem**: given \((g, g_1, g_2, g_1^a, g_2^b, Z)\), decide if \(Z = g^{a+b}\).

- Equivalently, given

  \[
  \vec{g}_1 = (g_1, 1, g), \quad \vec{g}_2 = (1, g_2, g), \quad \vec{\phi} = (g_1^a, g_2^b, Z),
  \]

  decide whether \(\vec{g}_1, \vec{g}_2, \vec{\phi}\) are linearly dependent (i.e., \(\vec{\phi} = \vec{g}_1^a \cdot \vec{g}_2^b\)).

- To commit to \(x \in \mathbb{Z}_p\), set \(\vec{C} = \vec{\phi}^x \cdot \vec{g}_1^{t_1} \cdot \vec{g}_2^{t_2}\).

- **Dual mode commitments**:
  - Perfect binding commitments and perfectly sound proofs if \(\vec{\phi} \notin \text{span}(\vec{g}_1, \vec{g}_2)\).
  - Perfectly hiding commitments and WI proofs if \(\vec{\phi} \in \text{span}(\vec{g}_1, \vec{g}_2)\).
Construction in prime order groups

- Use Damgaard’s Elgamal with $PK = (g, g_1, g_2, X_1 = g_1^{x_1} g^z, X_2 = g_2^{x_2} g^z)$.

  $C_0 = M \cdot X_1^r \cdot X_2^s, \quad C_1 = g_1^r, \quad C_2 = g_2^s, \quad C_3 = g^{r+s}$

- Add a simulation-sound proof that $(C_1, C_2, C_3) = (g_1^r, g_2^s, g^{r+s})$ using a CRS $(\vec{g}_1, \vec{g}_2, (\varphi_1, \varphi_2, \varphi_3 \cdot g^{VK}))$ where $(SK, VK) \leftarrow G(\lambda)$ is a one-time key pair.

- Security proof works:
  - CRS is only WI for the challenge ciphertext and only the challenger can generate one fake proof.
  - Adversary can only prove true statements.
  - Simulator knows the decryption keys (as in HPS-based proofs).
Efficiency comparisons

- Estimations at the 128-bit security level

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Group order</th>
<th>Assumptions</th>
<th>Ciphertext overhead (# of bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual system</td>
<td>$N = p_1p_2p_3 &gt; 2^{3072}$</td>
<td>Assumptions 1-2-3</td>
<td>6144</td>
</tr>
<tr>
<td>NIZK proofs + HPS</td>
<td>$p &gt; 2^{512}$</td>
<td>DLIN</td>
<td>10240</td>
</tr>
<tr>
<td>NIZK proofs + HPS</td>
<td>$p &gt; 2^{256}$</td>
<td>SXDH</td>
<td>3328</td>
</tr>
</tbody>
</table>

**Figure:** Comparisons in terms of ciphertext overhead

- Under DLIN: 12 pairings to check ciphertexts (using batch-verification); sender computes 19 exponentiations.
- Under SXDH: only 6 pairings to check ciphertexts (with batch-verification); sender computes 7 exponentiations.
Conclusion

- We described
  - CCA2-secure robust and non-interactive threshold cryptosystems secure against adaptive corruptions (with short private key shares)
    - Using the dual system technique
    - ... or a combination of hash proof systems with publicly verifiable (one-time) simulation-sound proofs
  - The first non-interactive threshold signature w/o random oracles in the adaptive corruption setting

- Open problems:
  - Can we achieve proactive security?
  - Have a distributed key generation protocol