Threshold Cryptography

When one cannot fully trust a unique person, but possibly a pool of individuals, the secret operation is distributed, so that authorized subsets only can perform it

- signature
- decryption

Threshold Cryptography

The access structure (authorized subsets) is defined by a threshold:

- any group of \( t \) players can perform the secret operation
- below this threshold, no power is provided to them
Threshold Public-Key Encryption

A ciphertext can be decrypted only if at least \( t \) users cooperate. Below this threshold, no additional information about the plaintext is leaked.

Many applications:
- electronic voting (decryption of the final result only)
- key-escrow
- identity-based cryptography (secret key extraction)
- etc

Classical Technique: ElGamal

\( \mathbb{G} = \langle g \rangle \) is a group of prime order \( p \)

**Lagrange Interpolation (Shamir’s Secret Sharing)**
- \( \mathcal{GM} \) generates a polynomial \( P \) of degree \( t - 1 \) over \( \mathbb{Z}_p \)
- each group member \( i \in \{1, \ldots, n\} \) receives \( sk_i = P(i) \)
- the group public key is \( PK = g^{sk} \), where \( sk = P(0) \)

\( t \) users can recover \( sk \), less than \( t \) users have no information.

**Threshold ElGamal Encryption**
- one can encrypt a message \( m \in \mathbb{G} \): \( c_1 = g^r, c_2 = PK^r \times m \)
- in order to decrypt, one has to compute \( a = PK^r = c_1^{sk} \):
  - each user \( i \) computes \( a_i = c_i^{sk_i} \)
  - with \( t \) values, \( a \) can be “interpolated”.
Limitations

At the key generation phase:

- the target group (or set) is fixed (the public key)
- the threshold $t$, to define the authorized subsets, is fixed

Dynamic Threshold Encryption

- any user can *dynamically* join the system as a future receiver
- the sender can *dynamically* choose the target set $S$
- the sender can *dynamically* set the threshold $t$

Related to

- Threshold broadcast encryption
  
  [Daza, Herranz, Morillo, Ràfols – ProvSec ’07]

  Ciphertext linear in $O(S)$

Outline

1. Formal Model
2. Our Construction
3. Conclusion
Robustness is achieved by **public** verification tools:

**ValidateCT**(EK, S, t, C). It checks whether C is a valid ciphertext with respect to EK, S and t

**ShareVerify**(VK, ID, uvk, C, σ). It checks whether σ is a valid decryption share with respect to uvk

**KEM-DEM methodology:**
- an ephemeral secret key K is first generated (KEM)
- a symmetric mechanism is used to encrypt the data (DEM)

**Encrypt**(EK, S, t). With the target set S (the public keys upk), and a threshold t, it outputs an ephemeral key K, and the key encapsulation material **HDR**
Security Model

**Correctness.** Valid encryptions should be correctly checked and decrypted, legitimate decryptions should be correctly verified, and should lead to the plaintext/ephemeral key.

**Robustness.** If $t$ shares are correctly checked with $\text{ShareVerify}$, then the $\text{Combine}$ algorithm outputs the correct key $K$.

**Privacy.** For any header $\text{HDR}$ encrypted for a target set $S$ of registered users with a threshold $t$, any collusion that contains less than $t$ users from this target set cannot learn any information about the ephemeral key $K$.

---

Security Model: Privacy

**Setup:** The challenger runs $\text{Setup}(\lambda)$ and the public parameters $(\text{EK}, \text{DK}, \text{VK}, \text{CK})$ are given to the adversary.

**Query phase 1:** The adversary $\mathcal{A}$ adaptively issues queries:
- Join queries (on a new user ID)
- Corrupt queries (on an existing user ID) to learn private keys
- ShareDecrypt queries (on an ID and a header $\text{HDR}$) to learn the partial decryption

**Challenge:** $\mathcal{A}$ outputs a set of users $S^*$ and a threshold $t^*$. The challenger randomly selects $b \leftarrow \{0, 1\}$, and gets $(K_0, \text{HDR}^*) = \text{Encrypt}(\text{EK}, S^*, t^*)$, and randomly chooses an ephemeral key $K_1$: it returns $(K_b, \text{HDR}^*)$ to $\mathcal{A}$.

**Query phase 2:** as Query phase 1

**Guess:** The adversary $\mathcal{A}$ outputs its guess $b'$ for $b$. 
Security Levels

With the natural restrictions on the oracle queries wrt. the target set and the threshold, the advantage of $A$ is defined as

$$\text{Adv}_A(\lambda) = \left| \Pr[b' = b] - \frac{1}{2} \right|.$$ 

As usual, $\text{Adv}(T, n, m, t, q_C, q_D)$ denotes the maximal value over the adversaries $A$ such that

- it runs within time $T$
- it makes at most
  - $n \text{ Join}$-queries
  - $q_C \text{ Corrupt}$-queries
  - $q_D \text{ ShareDecrypt}$-queries
- the size of $S^*$ is upper-bounded by $m$
- the value of $t^*$ is upper-bounded by $t$.

Security Level: the Basic one

**Non-Adaptive Adversary (NAA)**
We restrict the adversary to decide before the setup the set $S^*$ and the threshold $t^*$ to be sent to the challenger.

**Non-Adaptive Corruption (NAC)**
We restrict the adversary to decide before the setup the identities that will be corrupted.

**Chosen-Plaintext Adversary (CPA)**
We prevent the adversary from issuing $\text{ShareDecrypt}$-queries.

**$(n, m, t, q_C)$-IND-NAA-NAC-CPA security**
Non-adaptive adversary, non-adaptive corruption, and CPA.
Our **Combine** algorithm makes use of the **Aggregate** tool

[Delerablée, Paillier, and Pointcheval – Pairing ’07]

It allows to compute

\[ L = A^{(\gamma + x_1) \ldots (\gamma + x_t)} \in \mathbb{G}_T \]

given \( A \) and \( \Sigma = \{(x_j, a_j = A^{\gamma + x_j})\}_{j=1}^t \), but \( \gamma \) private, where the \( x_j \)'s are pairwise distinct.

---

**Our Construction: Setup**

\textbf{Setup}(\lambda). Given a bilinear setting, \( e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T \), with

- generators \( g \in \mathbb{G}_1 \) and \( h \in \mathbb{G}_2 \)
- \( \gamma, \alpha \leftarrow \mathbb{Z}_p^* \)
- \( \mathcal{D} = \{d_i\}_{i=1}^{m-1} \) of random values in \( \mathbb{Z}_p \), where \( m \) is the maximal size of a target set \( (\mathcal{D} \) corresponds to a set of public dummy users)
- \( u = g^{\alpha \cdot \gamma} \)
- \( v = e(g, h)^\alpha \)

- The master secret key: \( \text{MK} = (g, \gamma, \alpha) \)
- The encryption key: \( \text{EK} = (m, u, v, h^{\alpha \cdot \gamma}, \{h^{\alpha \cdot \gamma^i}\}_{i=1}^{2m-1}, \mathcal{D}) \)
- The decryption key: \( \text{DK} = \emptyset \)
- The combining key: \( \text{CK} = (m, h, \{h^{\gamma^i}\}_{i=1}^{m-2}, \mathcal{D}) \)
Our Construction: Join/Encrypt

**Join**(MK, ID). Given MK = (g, γ, α), and an identity ID, it randomly chooses a new \( x \in \mathbb{Z}_p \):

\[
\text{upk} = x \quad \text{usk} = g^{\gamma + x} \]

**Encrypt**(EK, S, t). Given a set \( S = \{ \text{upk}_1 = x_1, \ldots, \text{upk}_s = x_s \} \) and a threshold \( t \) (with \( t \leq s \leq m \)), **Encrypt** picks \( k \leftarrow \mathbb{Z}_p^* \), and sets \( \text{HDR} = (C_1, C_2) \) and \( K = v^k \):

\[
C_1 = u^{-k} \\
C_2 = h^{k \cdot \alpha \cdot \prod_{x_j \in S} (\gamma + x_j) \cdot \prod_{x \in D_{m+t-s-1}} (\gamma + x)}
\]

- a set of \( m + t - s - 1 \) dummy users + a set of \( s \) authorized users ⇒ a polynomial of degree \( m + t - 1 \) in the exponent of \( h \):
- \( m + t - 1 \leq 2m - 1 \): can be computed from EK
- the cooperation of \( t \) authorized users will decrease the degree of the polynomial in \( v \) to degree \( m - 1 \): **too high degree for CK**!

Our Construction: Decryption

**ShareDecrypt**(ID, usk, HDR). Given \( \text{HDR} = (C_1, C_2) \) and \( \text{usk} = g^{\gamma + x} \):

\[
\sigma = e(\text{usk}, C_2) = v^{k \cdot \prod_{x_j \in S \cup D_{m+t-s-1}} (\gamma + x_j) / (\gamma + x)}
\]

**Combine**(CK, HDR, T, Σ). Given a set \( \Sigma \) of \( t \) decryption shares:

\[
K = \left( e(C_1, h^{p(\gamma)}) \cdot \text{Aggregate}(v, \Sigma) \right)^{1/c}
\]

- \( c = \prod_{x \in S \cup D_{m+t-s-1} \setminus T} x \in \mathbb{Z}_p \)
- \( p(\gamma) = \frac{1}{\gamma} \cdot \left( \prod_{x \in S \cup D_{m+t-s-1} \setminus T} (\gamma + x) - c \right) \), a polynomial of degree \( m - 2 \), computable from CK
Our Construction: Decryption (Cont’d)

\[ K' = e\left(C_1, h^{p(\gamma)}\right) \cdot \text{Aggregate}(v, \Sigma) \]
\[ = e\left(g^{-k \cdot \gamma}, h^{p(\gamma)}\right) \cdot v^k \prod_{x \in S \cup D} \tau(\gamma + x) \]
\[ = v^{-k \cdot \gamma \cdot p(\gamma)} \cdot v^k (\gamma \cdot p(\gamma) + c) \]
\[ = v^{k \cdot c} = K^c. \]

\text{ValidateCT}(EK, S, t, HDR). \quad \text{Given } HDR = (C_1, C_2)

\[ C'_1 = u^{-1} \quad C'_2 = h^{\alpha \cdot \prod_{x \in S \cup D} (\gamma + x)} \]

\( HDR = (C_1, C_2) \) is valid with respect to \( S \) if and only if there exists a scalar \( k \) such that \( C_1 = C'_1 \cdot k \) and \( C_2 = C'_2 \cdot k \):

\[ e(C_1, C'_2) \overset{?}{=} e(C'_1, C_2) \]

Our Construction: Security Result

\textbf{Theorem}

\[ \text{Adv}(T, n, m, t, \ell, 0) \leq 2 \cdot \text{Adv}^{\text{mse} - \text{ddh}}(T', \ell, m, t). \]

\textbf{(\( \ell, m, t \)-Multi-Sequence of Exponents) DDH}

Let \( f \) and \( g \) be two random coprime polynomials, of respective orders \( \ell \) and \( m \), with pairwise distinct roots \( x_1, \ldots, x_\ell \) and \( y_1, \ldots, y_m \) respectively, as well as:

\[
\begin{align*}
g, g^{\gamma}, \ldots, g^{\gamma^{\ell+t-2}}, & \quad g^{k \cdot \gamma \cdot f(\gamma)}, \\
g^\alpha, g^{\alpha \cdot \gamma}, \ldots, g^{\alpha \cdot \gamma^{\ell+t}}, & \quad \\
h, h^{\gamma}, \ldots, h^{\gamma^{m-2}}, & \quad h^{k \cdot g(\gamma)}, \text{ and } T \in \mathbb{G}_T,
\end{align*}
\]

decide whether \( T \) is equal to \( e(g, h)^{k \cdot f(\gamma)} \) or not.
Our Construction: Security Result

**Lemma (Generic Security)**

For any probabilistic algorithm $A$ that makes at most $q$ queries to the group oracles, with $d = 4(\ell + t) + 6m + 2$

\[
\text{Adv}^{\text{mse-ddh}}(A, \ell, m, t) \leq \frac{(q + 4(\ell + t) + 6m + 4)^2 \cdot d}{2p}
\]

**Theorem (Generic Security)**

Our construction is secure
- against non-adaptive and generic adversaries
- under non-adaptive corruption
  and chosen-plaintext attacks

Our Construction: Efficiency

**Ciphertext Size**

Ciphertext: $C_1 = u^{-k}, C_2 = h^{k \cdot \alpha \cdot \prod_{x_i \in S}(\gamma + x_i) \prod_{x \in D^{m+t-s-1}}(\gamma + x)}$

The header has a constant size: two group elements

**Decryption**

Given $\text{HDR} = (C_1, C_2)$ and $\text{usk} = g^{\frac{1}{\gamma + x}}, \sigma = e(\text{usk}, C_2)$.

The user decryption is quite efficient: one pairing

**Non-Interactive Combination**

\[
K = \left( e\left(C_1, h^p(\gamma)\right) \cdot \text{Aggregate}(v, \Sigma) \right)^{1/c}
\]

The combination step does not need any interaction
Extensions: Random Oracle Model

All the previous properties are achieved in the standard model (under the MSE–DDH assumption)

Robustness

Easily achieved in the random oracle model, using Schnorr-like proof of equality of discrete logarithms

Identity-Based

It is simple to get an ID-based version in the random oracle model, by simply taking $u_{pk} = x = H(ID)$

Conclusion

- Security model for (dynamic) threshold public-key encryption (a.k.a. threshold broadcast encryption)
- Efficient and provably secure candidate the first with constant-size header

But still a lot of work on this topic:
- Use of a new non-standard assumption
- Secure against restricted adversaries only:
  - Chosen-plaintext attacks
  - Non-adaptive adversaries