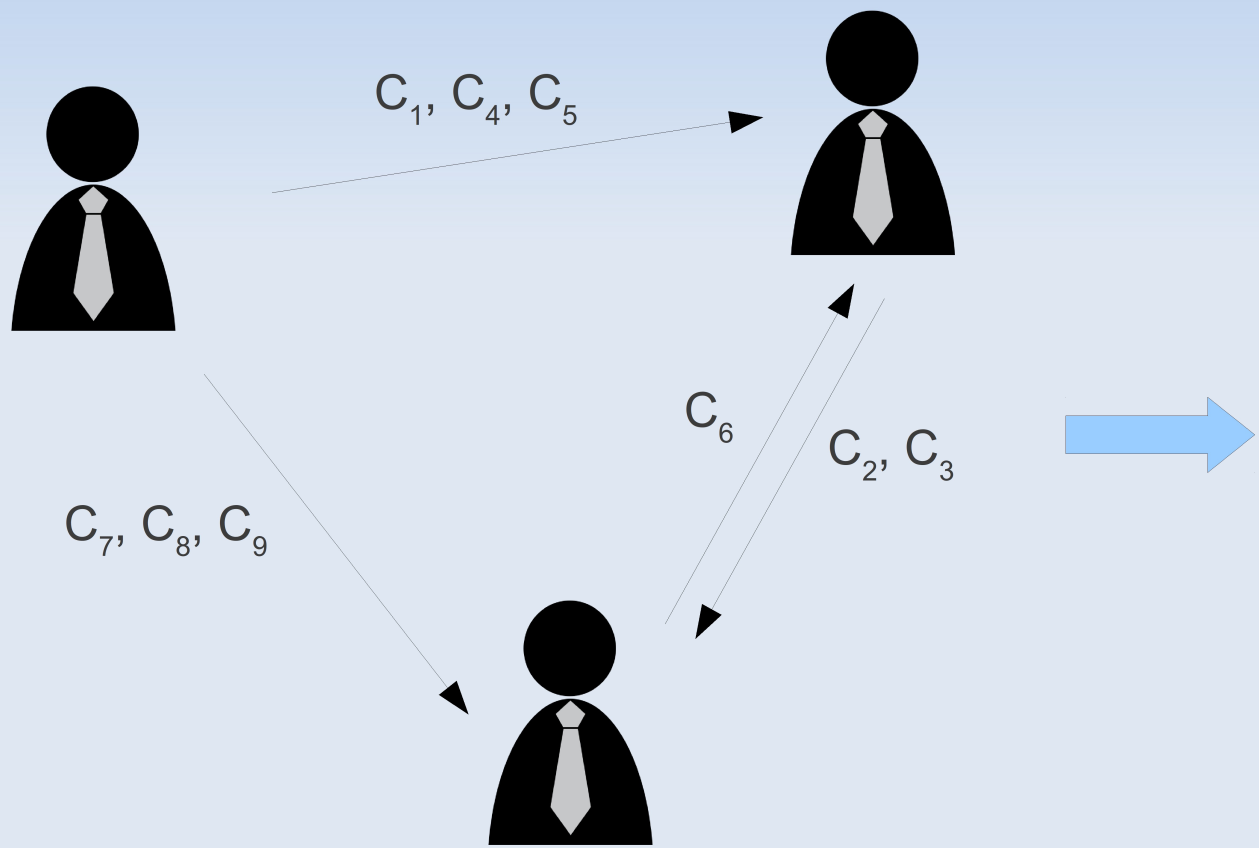
All-But-Many Lossy Trapdoor Functions

Dennis Hofheinz (Karlsruhe Institute of Technology)

Encryption: the "Real World"

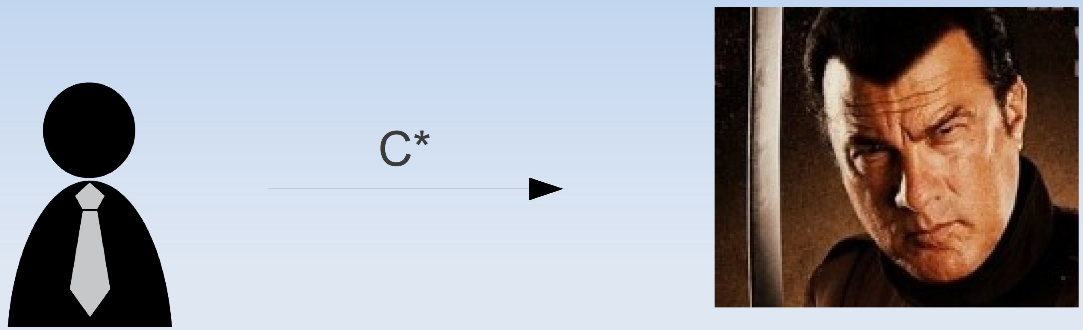
- Many parties, many ciphertexts



Adversary gets $C_1, \ldots, C_9$

A common simplification

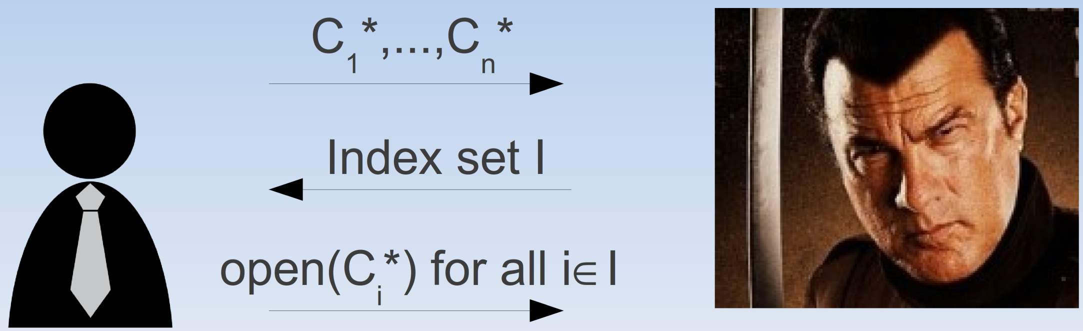
- **Simpler:** one user/sender, one challenge (e.g., IND-CCA)



Adversary gets C*

- **Justification:** usually, hybrid argument works
  - E.g., IND-CCA implies multi-user-multi-challenge-IND-CCA
- **But:** connection to real world not tight
- **And:** problematic in some cases (e.g., selective openings)

Example: Selective Openings



Adversary gets pk, $C_1^*,\ldots,C_n^*$,
openings of all C_i ($i \in I$)

- Intuition: adaptive corruption of multiple senders
- Security can be indistinguishability- or simulation-based
  - Intuition: adversary should not learn anything about unopened ciphertexts
  - **No hybrid argument, multiple challenges inherent**

Overview over this talk

All-But-Many Lossy Trapdoor Functions (ABM-LTFs)
A technical tool specifically designed for the multi-user-multi-challenge case

Construction of ABM-LTFs
A new look on Waters signatures

Next stop

All-But-Many Lossy Trapdoor Functions (ABM-LTFs)
A technical tool specifically designed for the multi-user-multi-challenge case

Recap: Lossy Trapdoor Functions

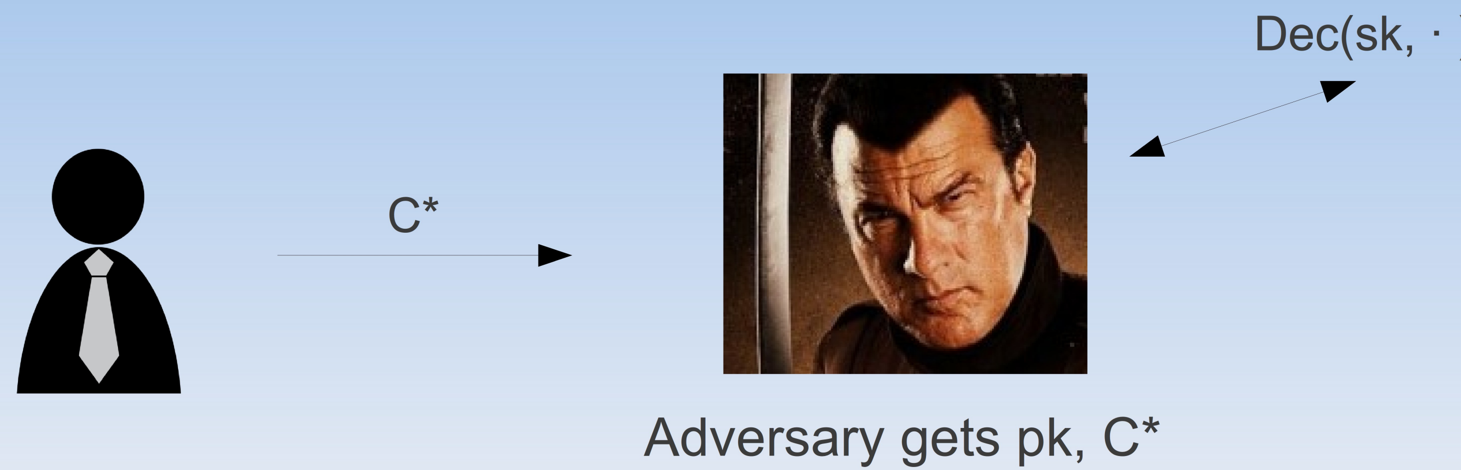
- (Keyed) function: $X \longrightarrow \boxed{F_{ek}} \longrightarrow F_{ek}(X)$
- Key can be ek (invertible mode) or ek' (lossy mode)
- Properties:
  - **Invertibility:** F_{ek} invertible using suitable trapdoor ik sampled with ek
  - **Indistinguishability:** ek $\approx$ ek'
  - **Lossiness:** image set $F_{ek'}(\mathbf{X})$ "much smaller" than $\mathbf{X}$
- Constructions from LWE, DDH, **DCR (efficient!):**

$$ek = (\ pk,\ C = E_{pk}(b)\ )$$

(Invertible mode: b=1, lossy mode: b=0)

$$\mathbf{F_{ek}(X) = C^X = E_{pk}(bX)}$$

Recap: PKE security from LTFs



- Intuition: pk = LTF key, C* contains LTF image
- Security: switch LTF to lossy mode, A gets (almost) no info on msg
- **Problem** with IND-CCA: cannot decrypt when in lossy mode
- **Solution:** All-But-One Lossy Trapdoor Functions [PW08]

T*

Does not work with many challenge ciphertexts!

All-But-N LTFs [HLOV11]

- Idea to cope with multi-challenge setting: many lossy tags!
- Construction based on Paillier/DJ encryption:

Pick degree-N polynomial $f(T) = \sum f_i T^i$ with zeros $T_1^*, \ldots, T_N^*$

$ek = (pk, C_0 = E_{pk}(f_0), \ldots, C_N = E_{pk}(f_N))$

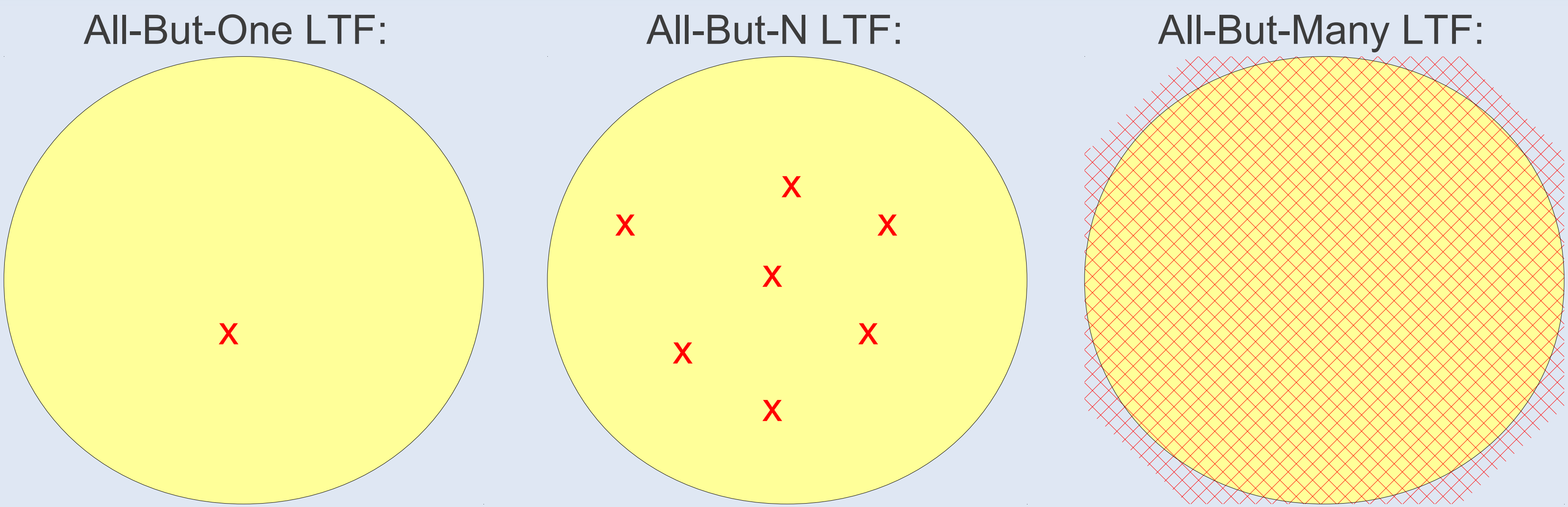
$\mathbf{F_{ek,T}(X) = (\prod C_i^{T^i})^X = E_{pk}(f(T)\, X)}$

- **Problem:** space complexity linear in the number of challenges
  - Actually, this is necessary to encode precisely N lossy tags
  - Yields SO-CCA secure PKE that depends on number of challenges
  - **Idea: each lossy tag T_i^* corresponds to a challenge ciphertext**
- **Our goal:** LTFs with many lossy tags!

All-But-Many LTFs

- Intuition/sketch of definition:
  - There are (superpoly) many lossy tags and (superpoly) many invertible tags
  - Lossy and invertible tags computationally indistinguishable

Tag sets (x marks lossy tags):



  - **Invertible** tags easy to sample, but **trapdoor** required to sample **lossy** tags
  - Syntactic similarity to **blinded signatures** (valid signature = lossy tag)

Next stop

Construction of ABM-LTFs
A new look on Waters signatures

First attempt

- Syntactic similarity to **”blinded signatures”** (valid sig = lossy tag)
- First attempt: so let's simply (Paillier/DJ-)encrypt signatures!

$$T = E(\mathrm{Sign}(H))$$

Something unique and public (e.g., chameleon hash)

- Evaluation ”magically” verifies signature inside encryption

...should end up with $C = E(0)$ **iff** sig is valid, then sets $Y{:=}C^X$

  - Sig valid $\Rightarrow$ $C = E(0)$ $\Rightarrow$ $F_{ek,T}(X) = C^X = E(0)$ lossy
  - Sig invalid $\Rightarrow$ $C = E(d)$ for $d \neq 0$ $\Rightarrow$ $F_{ek,T}(X) = C^X = E(dX)$ invertible
- Problem: (Paillier/DJ-)encryption only additively homomorphic
  - **How to evaluate signature using only addition in Z_N?**

Working with encrypted matrices

- **Idea 1:** use matrices instead of single elements (inspired by [PW08])

$$T \to E(M) = \begin{pmatrix} E(M_{1,1}) & E(M_{1,2}) & E(M_{1,3}) \\ E(M_{2,1}) & E(M_{2,2}) & E(M_{2,3}) \\ E(M_{3,1}) & E(M_{3,2}) & E(M_{3,3}) \end{pmatrix}$$

- Use "encrypted" matrix-vector multiplication:

$$F_{ek,T}(X) = E(M) \circ \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} \prod_j E(M_{1,j})^{Xj} \\ \prod_j E(M_{2,j})^{Xj} \\ \prod_j E(M_{3,j})^{Xj} \end{pmatrix} = E(M \cdot X)$$

- $F_{ek,T}$ lossy $\Leftrightarrow$ M non-invertible $\Leftrightarrow$ det(M)=0 (or non-invertible)
- **Payoff:** det(M) can be **cubic** in encrypted values
- **Use determinant to encode more complex computations**

Translating Waters signatures

- **Idea 2:** emulate Waters signatures in Z_N
  - Use encryption instead of exponentiation (g^a becomes E(a))
  - Pairing becomes Paillier/DJ multiplication **(encode verification into det(M)!)**
  - CDH in G becomes **"Paillier-No-Mult"**: E(a), E(b) → E(ab) hard
- **All-But-Many LTF construction (slightly simplified):**

ek = (A=E(a), B=E(b), H_i=E(h_i) (i=0,...,n)) (translated Waters public key)

T = (R=E(r), Z=E(z), CHF-rand) (translated Waters signature)

$$T \rightarrow E(M) = \begin{pmatrix} E(z) & E(a) & E(r) \\ E(b) & E(1) & E(0) \\ E(h) & E(0) & E(1) \end{pmatrix}$$

with $E(h) = H(t) = h_0 + \sum t_i h_i$
for t = CHF(R,Z;rnd)

$$\mathbf{F_{ek,T}(X) = E(M) \circ X = E(M \cdot X)}$$ (implicit Waters verification)

Note: det(M) = z – (ab+rh), **so:** T lossy ⇔ M singular ⇔ z = ab + rh

Last slide: applications

- **Efficient CCA-secure Selective Opening Security**
  - Many challenges, need to make exactly challenges lossy
  - Paillier-based ABM-LTFs give first efficient SO-CCA scheme
- **(Not very efficient) tight IND-CCA security for PKE**
  - Make all challenges lossy simultaneously (**tightly** secure ABM-LTF)
  - Different ABM-LTF required (not very efficient, based on q-SDDH)
- **CCA-secure Key-Dependent Message security**
  - Similar concepts, but more structured ABM-LTFs required **(upcoming)**
- **Leakage resilience?**