Threshold PKC

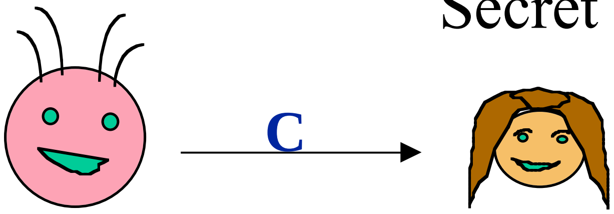
Shafi Goldwasser and
Ran Canetti

Public Key Encryption [DH]

A PKC consists of 3 PPT algorithms (G,E,D)

- $G(1^k)$ outputs public key e, and
  secret key d
- E(m, e) outputs cipher text c
- D(c, e, d) outputs m.

Public Key: e
Secret key: d



Active Adversary: Standard PKC [RS]

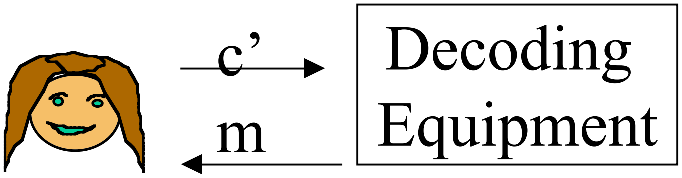
- Chosen Cipher-text Attacks (CCA)

  -Adversary chooses m_0 m_1

  -Adversary receives c either in $E(m_0)$ or $E(m_1)$ at random

  -Adversary may ask

  $c' \neq c$



A scheme is secure against CCA if adversary still cannot tell whether c in $E(m_0)$ or in $E(m_1)$ better than 50-50

comes up in protocols

Threshold Cryptography [D,DF]

An encryption or digital signature scheme where:

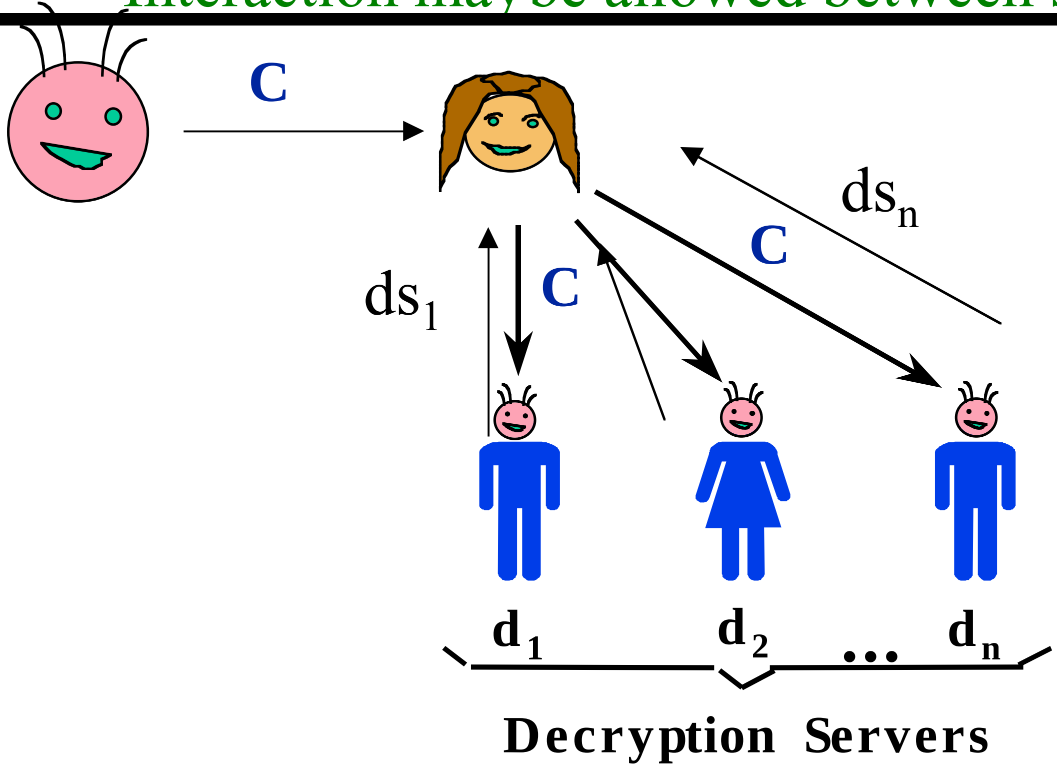
- Secret key is shared among trustees s.t.
- Trustees can decrypt or sign only if enough cooperate
- Faulty trustees can't prevent decryption or signature
- Faulty trustees can be detected if they act up (optional).

Threshold Public Key Cryptography [DF]

A Threshold PKC_n consists of 3 PPT algorithms (G,E,D)

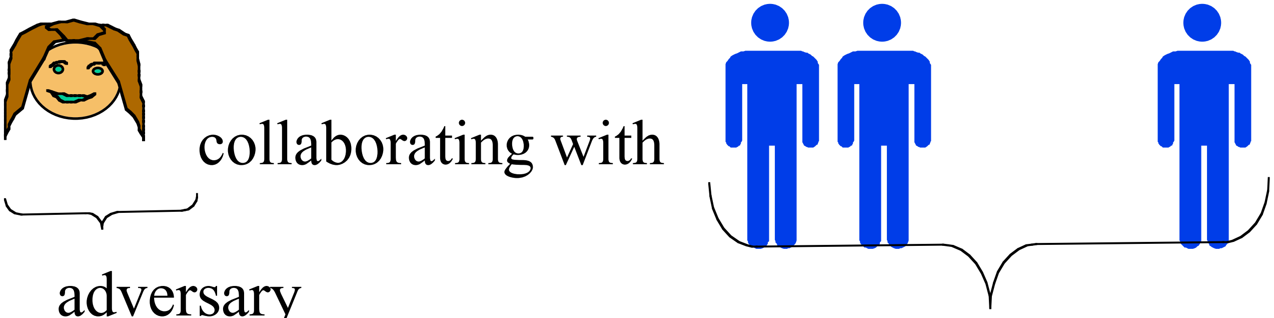
- $G(1^k)$ outputs public key e, and shares of secret key $d_1,\ldots,d_n$
- $E(m, e)$ outputs cipher-text c
- $D^* = (D_1, D_2)$ where $D_1(c, d_i)$ outputs decryption share ds_i
  $D_2(c, e, ds_1, \ldots, ds_n)$ outputs m.

* Interaction maybe allowed between servers and user.



Public Key: e
Secret Key Shares: d_i
distributed among servers

Security: Threshold PKC



collaborating with

adversary

t servers

While launching the CCA: the adversary has access to all the private data of collaborating servers

Say A Threshold Public Key Encryption Scheme is :

t-secure: a coalition of t curious but honest servers + adversary cannot break it.

t-robust: a coalition of t faulty servers cannot prevent user from decrypting (no denial of service).

Previous Work

- Gennaro-Shoup: under the assumption that Random Oracles exist and the DDH intractability assumption, show a *Threshold* PKC which is t-secure and t-robust for t< n/2 against CCA. (No interaction is necessary.)
- Dolev-Dwork-Naor: under the assumption trapdoor functions exist show *single server* PKC secure against CCA. Use NIZK for construction. (Prior [NY] LTA)
- Cramer-Shoup: under the DDH intractability assumption show a *single server* PKC secure against CCA. Quite Efficient.

New Threshold PKC

- KEY GEN: PK = $(g_1, g_2, a = g_1^{x1} g_2^{x2}, h = g_1^{z})$
  SK: each decryption server holds a share of x_1, x_2, y_1, y_2, z (using polynomial secret sharing,
  e.g. $x_{1i} = X_1(i)$ where $X_1(0) = x_1$, $\deg(X_1) = t$)

- ENC: Same as in single server case

- $\text{DEC}\underbrace{(SK,c)}_{(u_1,u_2,\, e,\, tag)}$: Let s be random and S a deg t polynomial s.t $S(0)=s$ and each server I has $S(i)=s_i$

- Server i computes $tag_i' = u_1^{x_{1i}} u_2^{x_{2i}}$ and sends the user

$$\left.\begin{array}{l} \boxed{g^{Q(i)}} = (tag/tag_i')^{s_i} h^{z_i} \\ \text{- User combines shares to obtain} \\ \boxed{g^{Q(0)}} = (tag/tag')^{s} h^{z} \quad \text{and} \quad \text{lets } m = e/(tag/tag')^{s} h^{z} \end{array}\right\} \textbf{HOW?}$$

Combine decryption shares by using Lagrange Interpolation?

- User received for all I,

  $\text{Share}_i = (tag/tag_i')^{si} h^{zi} = g^{Q(i)}$ where Q is some degree 2t polynomial s.t. $Q(0) = (tag/tag')^{s} h^{z}$,

  and needs $g^{Q(0)}$

Lagrange Interpolation: Gives λ_i s.t $Q(0) = \Sigma\, \lambda_i Q(I)$ for every 2t degree polynomial Q.

- To combine shares, user computes

  $$\Pi\, (\,\text{Share}_i\,)^{\lambda i} = \Pi\, (\,g^{Q(i)}\,)^{\lambda i} = g^{\Sigma\, \lambda i Q(I)} = g^{Q(0)}$$

Where do s_i come from for each decryption ?

1 Servers share in advance random poly's $S_1,\ldots S_k$ s.t. deg $(S_j) = t$ and $S_j(0)=s_j$. I.e server i holds $s_{ji}= S_j(i)$ for all j, to use for decrypting jth cipher text.

2 To avoid synchronization errors, servers can share in advance on a single 2-var polynomial $S(x,y)$ where $S(c,)$ is as above, I.e server i holds polynomial $S(x, i)$, and uses $s_i=S(c,I)$ for cipher text c.

EVOX 1.0 (current status)

- F.O.O. protocol: practical, scalable elections
- Simple implementation done in Java 1.1
- So far, 2 medium-size elections with relative success. Issues found:
    - Unintuitive user interface
    - Low Reliability
    - Some relatively obscure security bugs
- Numerous people (including 3 universities) have expressed interest in using EVOX.

EVOX 2.0 - 3.0 (this year)

- Coming Improvements
    - Multiple administrator servers (registrars) and threshold signature schemes to prevent single corruption point weakness in F.O.O. protocol.
    - Timing improvements through signature and verification batching (based on scheme by Amos Fiat), or delegation. Different schemes are currently being analyzed.
    - Improved UI, code security analysis, packaging of system to enable wider use.
    - Hoping for wider release of code (possible GPL?)
- Current contributors: Ben Adida, Brandon DuRette, Kevin McDonald
- http://theory.lcs.mit.edu/~cis/voting/voting.html