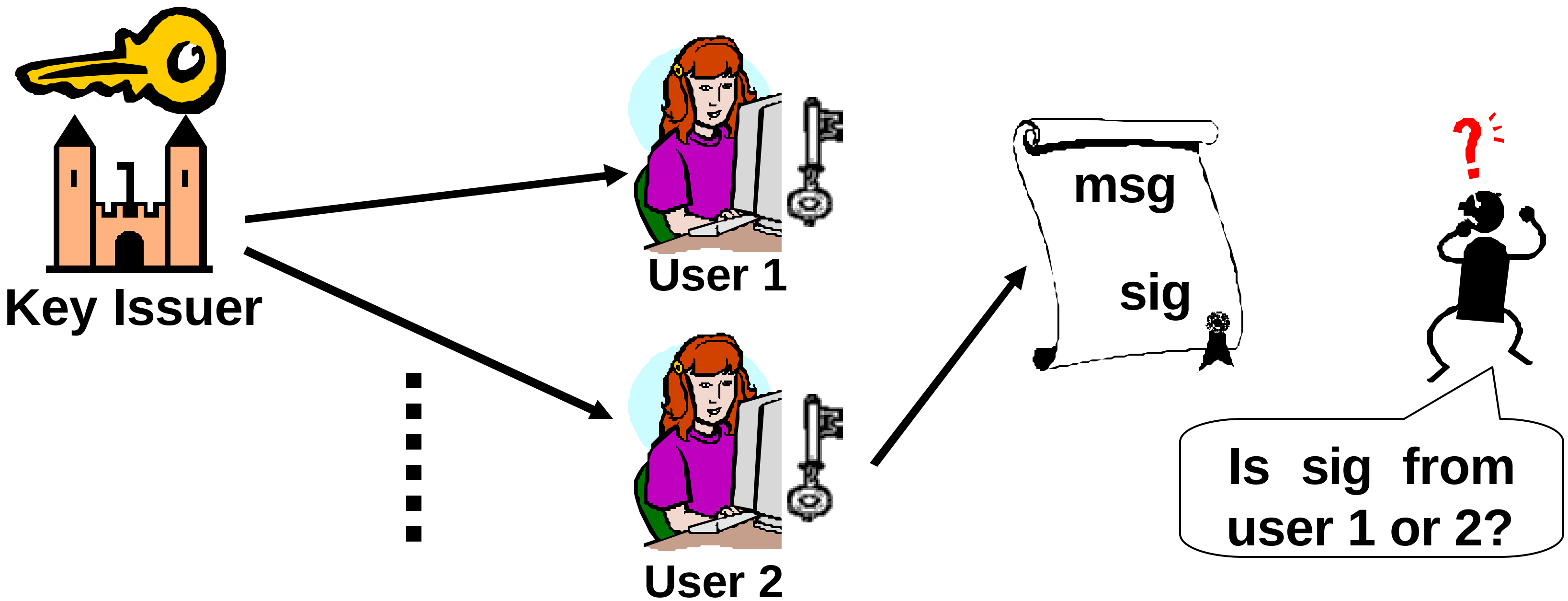
A New Life for Group Signatures

Dan Boneh

Stanford University

Group Signatures: intuition



- Simple solution: give all users same private key …
- … but, extra requirements:
  - Ability to revoke signers when needed.
  - Tracing Authority: trapdoor for undoing sig privacy.

History

- D. Chaum and E. van Heyst. [EC ’91]
- N. Baric and B. Pfitzman [EC ’97]
- **G. Ateniese, J. Camenisch, M. Joye, G. Tsudik** [EC ’00]
- J. Camenisch and A. Lysyanskaya. [Cr ’02]
- G. Ateniese, D. Song, and G. Tsudik [FC ’02]
- **M. Bellare, D. Micciancio, and B. Warinschi** [EC ’03]

⋮

This talk

- Recent real-world applications.
- Privacy definitions and models.
  - Zoology: 9 models for group sigs …

- New group sig constructions [BBS '04]
  - Very short. Very efficient.
  - Based on Strong-DH (using bilinear maps)

Basic group signatures [BMW'03]

Basic: tracing, but no revocation (static groups).

Group sig system consists of four algorithms:

- Setup(λ,n): λ = sec param. n = #users.
  Output: group-pub-key (GPK), (GSK_1 , …, GSK_n) ,
  group-tracing key (GTK)
- Sign(M, GSK_i): outputs group signature σ on M.
- Verify(M, σ, GPK): outputs `yes' or `no'
- Trace(M, σ, GTK): outputs i $\in$ {1,…,n} or `fail'

Precise security requirements: later …

Recent Applications for Group Sigs

- Two recent "real-world" applications:

  1. Trusted Computing (TCG, NGSCB)

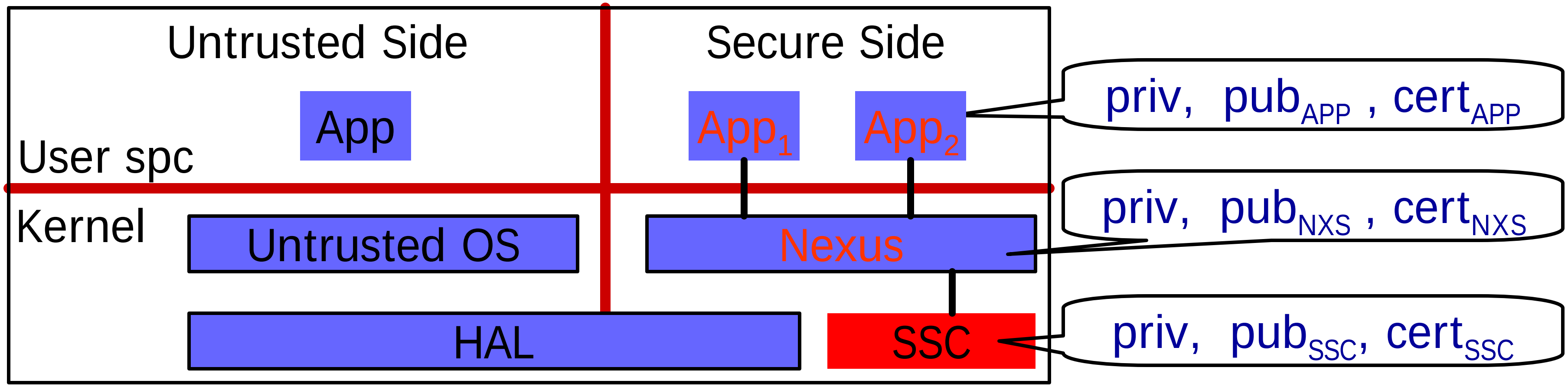
  2. Vehicle Safety Communications (VSC)

App. 1: Trusted Computing

- TCG: Trusted Computing Group (aka TCPA).
- NGSCB: Next Gen Secure Comp Base (aka Palladium)

- Provides new capability: Attestation.
  - Enables an application to authenticate its executable code to a remote server.
  - Uses: home banking, online games, … , DRM

(Very) High level architecture



- SSC: Security Support Component ("tamper resistant" chip)
  - Issues: $cert_{NXS}$ = [hash(nexus-code), nxs-pub-key, sig-ssc]
- Nexus: Protects and isolates apps on secure side.
  - Issues: $cert_{APP}$ = [hash(app-code), app-pub-key, sig-nxs]

- Attestation: app uses cert-chain = [$cert_{APP}$, $cert_{NXS}$, $cert_{SSC}$] in key exchange with remote server.

Privacy Problem

- SSC’s cert is sent to remote server on every attestation.
  - SSC’s cert identifies machine (recall Intel unique x86 ID’s)
  - Attestation breaks privacy tools (e.g. anonymizer.com)

➢ Initial TCG Solution: Privacy CA.

Trusted online service that anonymizes SSC’s cert.

➢ Simple Solution: give all SSCs same priv-key and $cert_{SSC}$.

Bad idea: no way to revoke compromised SSC.

➢ Better solution: group signatures. No online service [Brickell]

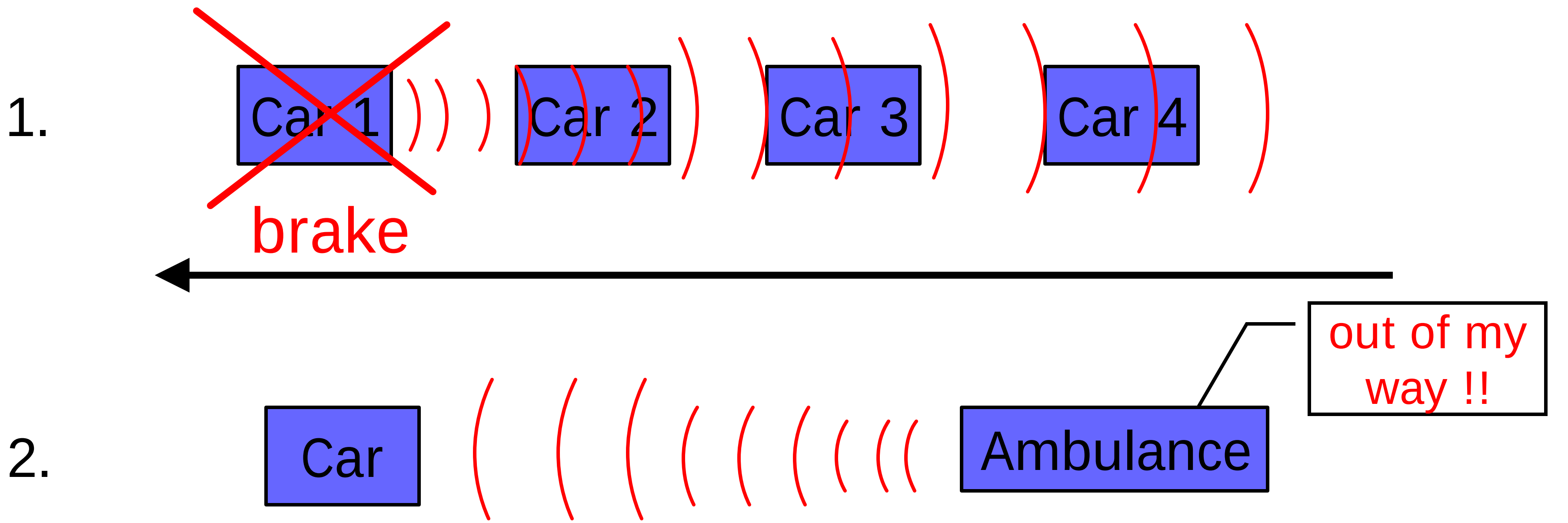
Group is set of all SSC.

Manufacturer embeds a group priv-key (GSK) in each SSC.

$cert_{NXS}$ issued by SSC does not reveal machine ID.

Trace and revoke SSC key in case of SSC compromise.

App. 2: Vehicle Safety Comm. (VSC)



- Require authenticated (signed) messages from cars.
  - Prevent impersonation and DoS on traffic system.
- Privacy problem: cars broadcasting signed (x,y, v).

- Clean solution: group sigs. Group = set of all cars.
- Project requirement: msg-size < 300 bytes

  $\Rightarrow$ Need short group signatures.

Characteristics of both applications

- Signing key in tamper resistant chip in user's hands.
  - Signing key embedded at manufacturing time.
- Revocation only needed for tamper resistance failure.
  - Infrequent. (unlike a private subscription service)
  - Tracing may or may not be needed.

Group signatures: basic definitions

[BMW'03]

- Def: A Basic Group Signature (static groups & tracing)

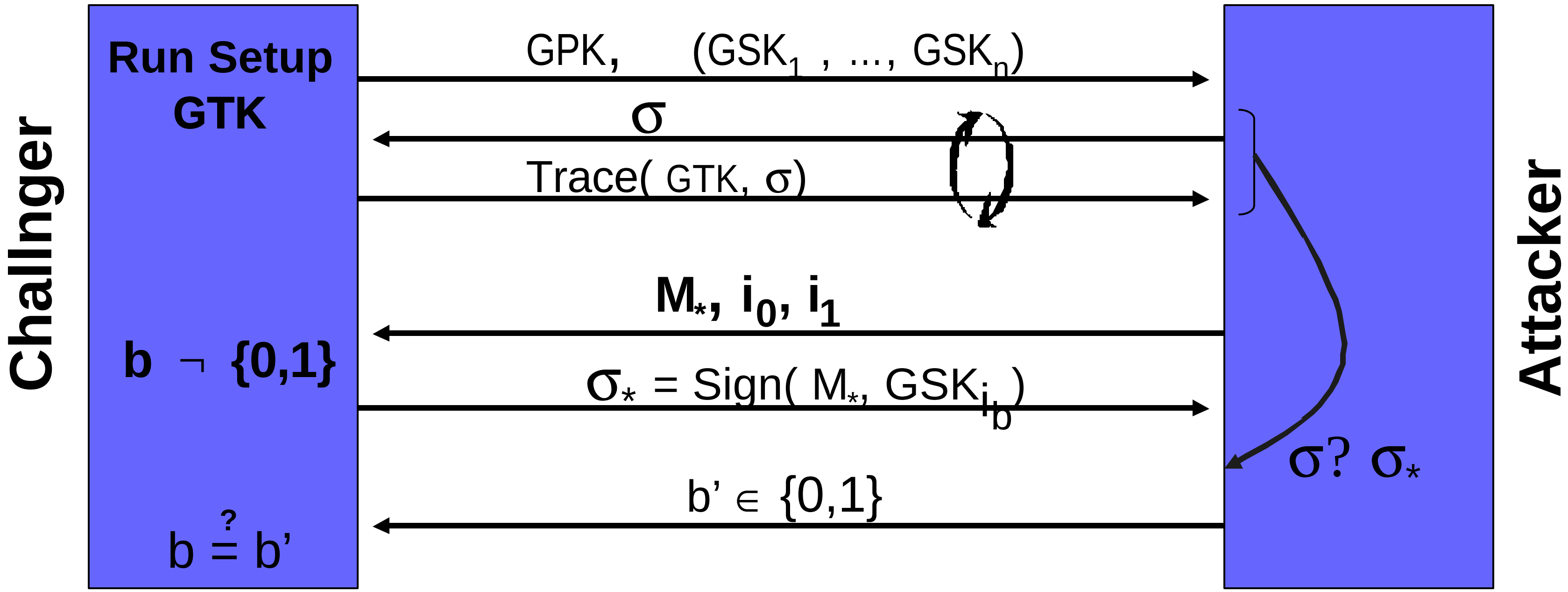
  (setup, sign, verify, trace)

  is secure if it has:

  1. full-privacy property, and
  2. full-traceability property.

(CCA) Full-Privacy

- No poly. time alg. wins the following game with non-negligible advantage:

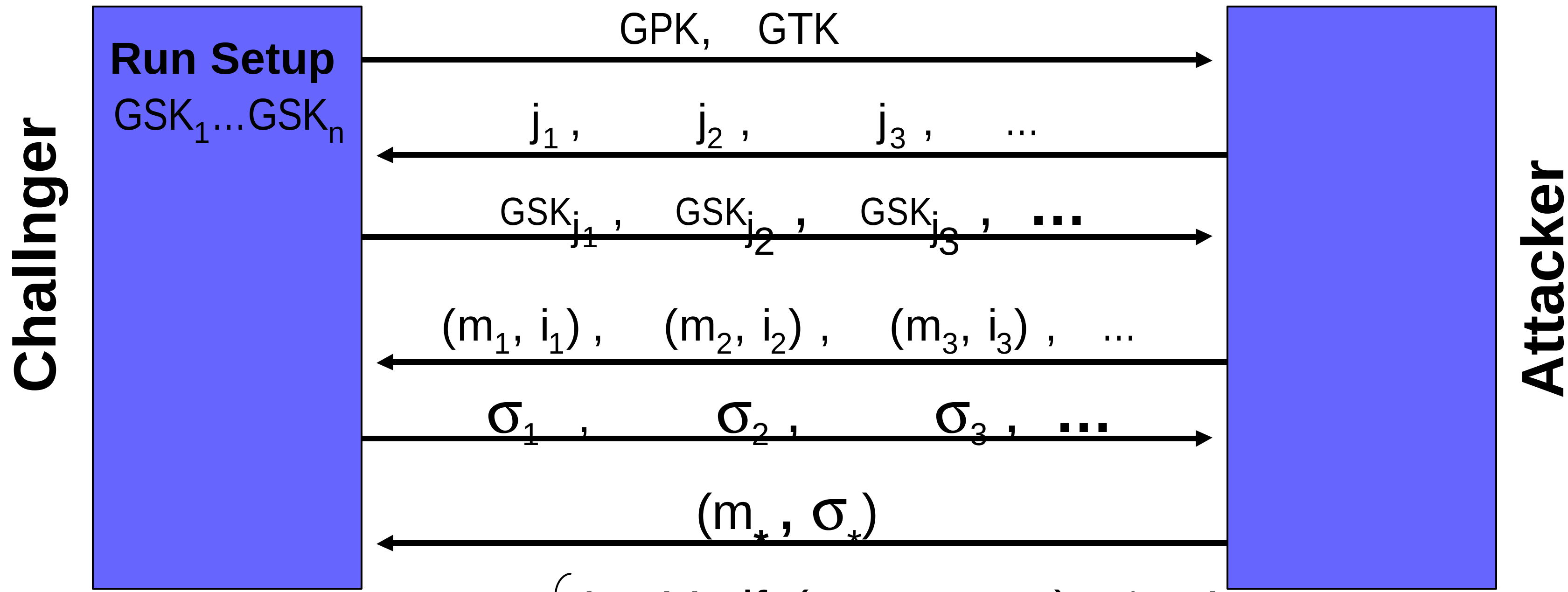


- Open problem: efficiently handle CCA2 tracing attack.

  Instead, will use: CPA-full-privacy

Full-Traceability

- No poly. time alg. wins the following game with non-negligible probability:



Attacker wins if :

1. $\text{Verify}(m_*, \sigma_*, \text{GPK}) = \text{'yes'}$
2. $(m_*, \sigma_*) \notin \{(m_1, \sigma_1), \dots\}$
3. $\text{Trace}(m_*, \sigma_*, \text{GTK}) \notin \{j_1, \dots\}$

Resulting properties (informal)

- Unforgeability. Group sig is existentially unforgeable under a chosen message attack.
- Unlinkable. Given two group sigs it is not possible to tell whether they were generated by same user.
- No Framing. A coalition of users cannot create a signature that traces to a user outside the coalition.
- Note: no exculpability. Key-Issuer might be able to forge signatures on behalf of a given user.
  - ACJT'00, BBS'04 provide exculpability.
  - May not be needed in real world (e.g., none in std. PKI)

Revocation Mechanisms

- Revocation goal (intuition):
  - After users $\{i_1, \ldots, i_r\}$ are revoked they cannot issue new valid group sigs.

- For now, ignore validity/privacy of old group sigs.

Revocation Mechanisms (easiest → hardest)

- Type 0: For each revocation event, generate new GPK. Give each unrevoked user its new private key.

- Type 1: For each revocation event, send a short broadcast message RL to all signers and all verifiers. (msg-len independent of group size)
  - Implementation: [CL'02]

    verifiers: $(\text{GPK}_{old}, \text{RL}) \rightarrow \text{GPK}_{new}$

    active user i: $(\text{GSK}_{i,old}, \text{RL}) \rightarrow \text{GSK}_{i,new}$

- Type 2: For each revocation, send msg to verifiers only.
  - Implementation: Verify(GPK, (m,σ), RL)
  - Note: old sigs of revoked users are no longer private.

Tracing Mechanisms (easiest $\rightarrow$ hardest)

- Type 0: No tracing possible.
- Type 1: Given a black box signing device, can identify at least one member of coalition that created device.
  - Note: $\text{Trace}^{\text{sig}(.)}$ (GTK) is now an oracle alg.
  - Definition: similar to full-traceability.
- Type 2: Full-traceability. Given a signature, can identify at least one member of coalition that created sig.

Zoology: Group signature types

- Each square below requires precise def (as for RT0-TT2)

<table>
<tr><th>revoke / Trace</th><th>RT0</th><th>RT1</th><th>RT2</th></tr>
<tr><td>TT0</td><td>Global Secret Key</td><td>Global key with NNL broadcast enc.</td><td rowspan="3">BBS’03
AST’02
(built in tracing)</td></tr>
<tr><td>TT1</td><td></td><td>BBS’04 Lite</td></tr>
<tr><td>TT2</td><td>BMW ‘03
ACJT ‘00</td><td>CL’02
BBS’04</td></tr>
</table>

[3rd dimension: exculpability (yes/no)]

Constructions:

- Construction from general primitives [BMW'03]
  - Uses public key encryption,
    Signature scheme,
    Non-Interactive Zero Knowledge.

- Specific constructions (using Fiat-Shamir heuristic) :
  - Based on the Strong-RSA assumption [ACJT'00, …]
  - New: Based on the Strong-DH assumption [BBS'04]
    - Much shorter sigs than Strong-RSA counter-part.

Strong Diffie-Hellman [BB '04, BBS '04]

- n-SDH problem: let G be a group of prime order p.
  - Input: $g, \; g^x, \; g^{(x^2)}, \; g^{(x^3)}, \; \ldots, \; g^{(x^n)} \in G$
  - Output: (A, e) s.t . $A^{x+e} = g$

    [Strong-RSA: given (N,s) output (A,e) s.t. $A^e = s$ (N)]

- n-SDH Assumption: "n-SDH problem is hard for rand x"

- Evidence n-SDH is a hard problem:

  Thm: An algorithm that solves n-SDH with prob. ε in a generic group of order p requires time $\Omega(\sqrt{\varepsilon p/n})$

App: Short sigs without RO [BB'04]

- Setup: $x,y \leftarrow Z_p$; $PK = (g, g^x, g^y)$; $SK = (x,y)$
- Sign(m, (x,y)): $r \leftarrow Z_p$; $\sigma = (g^{1/(x+ry+m)} , r)$
- Verify(m, σ=(h,r)): test $e(h, g^x \cdot (g^y)^r \cdot g^m) = e(g,g)$
- Thm: Signature scheme is existentially unforgeable under an n-chosen message attack, assuming (n+1)-SDH holds
- Signature is as short as DSA, but has a complete proof of security without random oracles.

Group sigs from SDH (RT1-TT2) [BBS '04]

- Setup(n): random $a, b, c \leftarrow \{1,\ldots,p-1\}$

  $GPK \leftarrow (g, h, h^a, h^b, g^c)$; $GTK \leftarrow (a,b)$

  $GSK_j \leftarrow (\, x_j\, ,\ A_j = g^{1/(c+x_j)}\ )$ for $j = 1,\ldots, n$

- Sign(m, GSK_j) = random $d, e \leftarrow \{1,\ldots,p-1\}$

  $T_1 = (h^a)^d$; $T_2 = (h^b)^e$; $T_3 = A_j \cdot h^{d+e}$

  Encryption of A_j

  Proof $\leftarrow ZKPK_m (\, d, e, x_j, dx_j, ex_j\,)$ satisfying 5 relations.

  sig = $[T_1, T_2, T_3,$ Proof] (9 elements)

- Trace(σ, (a,b)) = $T_3 / (T_1^{\,a} \cdot T_2^{\,b}) = A_i$

  Decryption

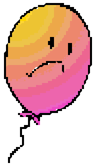
New group sig properties

- Security:
  - Full-Traceability: based on n-SDH
  - CPA-Full-Privacy: based on Decision Linear.
- Supports simple Type 1 revocation.
- Length:
  - $\approx$ same length as standard RSA signature.
  - In practice $\leq$ 200 bytes (!) for 1024-bit security.

Revocation (Type 1)

- Recall GPK $\leftarrow$ (g, h, h^a, h^b, g^c)
- To revoke $GSK_1 = (x_1 , A_1 = g^{1/(c+x_1)})$ do:
  - Publish GSK_1 in the clear.
  - $GPK_{new} \leftarrow (A_1, h, h^a, h^b, A_1^{\,c})$
  - $GSK_{i,new} \leftarrow (x_i , A_1^{1/(c+x_i)})$
- Main point: all unrevoked users can compute $GSK_{i,new}$.
  - Revoked user can no longer issue sigs (under SDH).

Conclusions

- Lots of group signature models.
  
  - Three tracing models. Three revocation models.
  - Use most efficient system that meets your needs …
- New constructions:
  - Short group signatures (same as std. RSA sigs).
  - Flexible: can be adapted to all trace/revoke models.
- Open problems:
  - Efficient group sigs (RT0-TT2) without random oracles.
  - Efficient CCA-full-privacy with/without random oracles.