A New Life for Group Signatures

Dan Boneh, Xavier Boyen, and Hovav Shacham
{dabo,xb,hovav}@cs.stanford.edu

Related papers: [BBS04] in Crypto 2004;
[BS04] to appear in CCS 2004

Group Signatures

- There is one group public key.
- Each user has a private signing key.
- Any user can sign on behalf of the group
- Only the group manager can tell which user generated a particular signature.

Group Signature History

- D. Chaum and E. van Heyst. [EC '91]
- Camenisch and Stadler [Cr '97]
- Ateniese, Camenisch, Joye, Tsudik [EC '00]
- Camenisch and Lysyanskaya [Cr '02]
- Ateniese, Song, and Tsudik [FC '02]
- Bellare, Micciancio, and Warinschi [EC '03]
- Camenisch and Lysyanskaya [Cr '04]
- ...

Security Properties

- Correctness
- Unforgeability
- Anonymity
- Unlinkability
- Exculpability and non-framing
- Traceability
- Coalition-resistance
- Revocation?
- Forward secrecy?
- ...

What's the matter?

- Group signature security properties are many, ill-defined, and difficult to reason about.
- State-of-the-art group signature constructions (e.g., ACJT) are complicated and slow.
  - 32,000-bit signatures!
- There is no real-world need.

A New Life for Group Signatures

Within the last two years:

- New, simple security model [BMW03]
  (correctness, full-traceability, full-anonymity)
- Short, simple construction [BBS04]
  (based on Strong Diffie-Hellman [BB04])
- Real-world applications (TCPA, DSRC)

Trusted Computing (TCPA)

- Embed tamper-resistant hardware (SSC), with private key, in each PC
- Remote attestation: convince remote party that I am running particular software
  certificate chain with program hashes, rooted at SSC key
- Problem: now RIAA can follow my activities
- Solution: put SSCs in a group; root the certificate chain at a group signature.

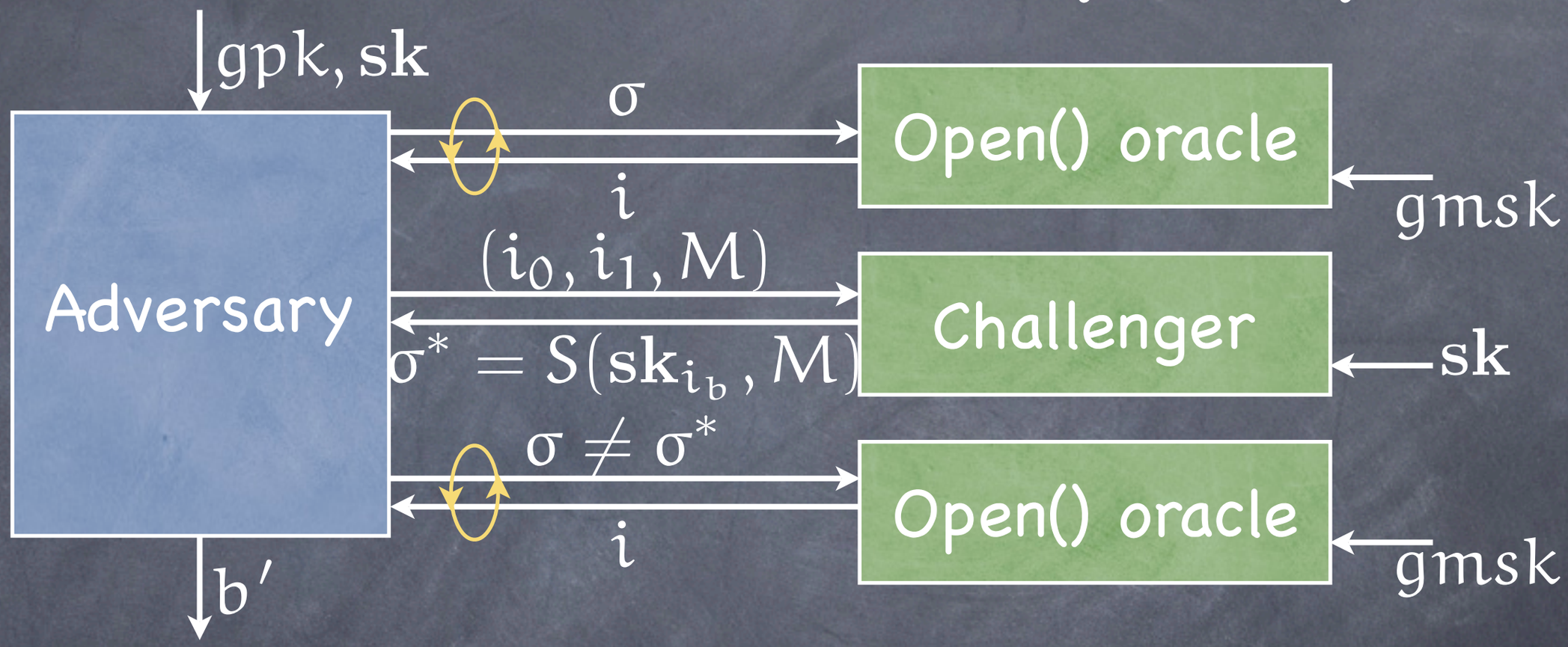
Vehicle Safety Ad-Hoc Neworks (DSRC)

- Short (<300-500 bytes) messages over wireless network
- Key embedded in tamper-resistant component
- Car-to-car: I am braking!
- Object to car: I am a stoplight
- Sign to avoid malicious messages
- Problem: Now the police can use my car's messages against me in court
- Solution: put all cars in a group; use group signatures for message authenticity

Group Signatures [BMW03]

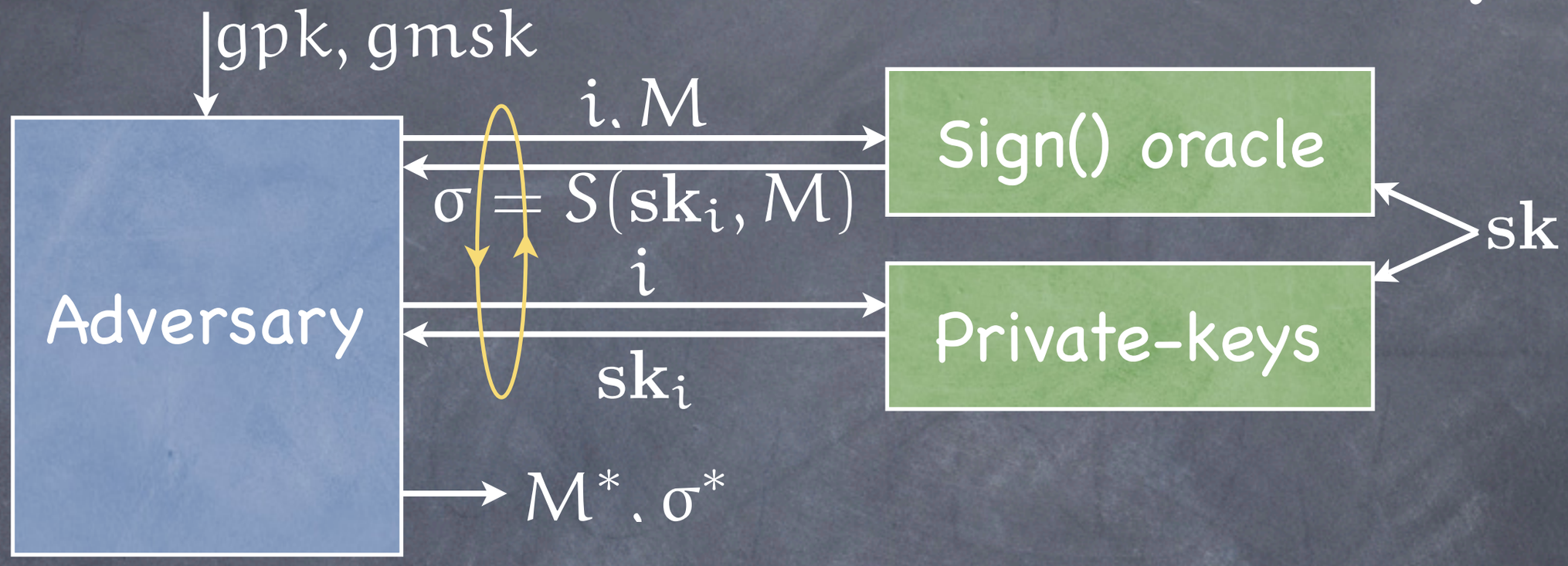
- Definition for static membership, no revocation
- Four algorithms
  - Setup(λ,n) — λ: security param; n: # users
    outputs public key gpk, tracing key gmsk, user keys sk_1, …, sk_n
  - Sign(sk_i,M) — outputs group signature σ on M
  - Verify(gpk,σ) — outputs 'yes' or 'no'
  - Open(gmsk, σ) — outputs $i \leq n$ or 'fail'
- Correctness: all properly-generated signatures verify, and trace to their signer.

[BMW03]: full-anonymity



- Adversary gets all user keys, access to tracing oracle (before and after challenge)
- Goal is to tell which user key created σ^*.
- Adversary wins if b = b′.

[BMW03]: full-traceability



- Adversary is given the tracing key, and can obtain signatures and private keys.
- His goal is to create a valid signature σ* that cannot be traced, or that traces to a user for whose private key he didn't ask.

Security Property Bingo

- Full-anonymity gives:
  Anonymity; Unlinkability (?)

- Full-traceability gives:
  Unforgeability; (Weak) exculpability; Non-framing; Traceability; Coalition-resistance

- To do:
  Strong exculpability; Revocation; Forward secrecy

- [Digression: do we need strong exculpability?]

Mathematical Setting [BLS01,...]

- Multiplicative groups G_1, G_2, G_T of order p; g_1 is the generator of G_1, g_2 of G_2.
- Map $\psi: G_2 \rightarrow G_1$ such that $\psi(g_2) = g_1$
- Bilinear map $e: G_1 \times G_2 \rightarrow G_T$ such that
  - $e(u^a,v^b) = e(u,v)^{ab}$
  - $e(g_1,g_2) \neq 1$
- [Digression: $G_1 \neq G_2$.]

Assumptions

- q-Strong Diffie-Hellman [BB04,MSK02]:
  Input: g_1, $w=g_2^{(\gamma)}$, $g_2^{(\gamma^2)}$, …, $g_2^{(\gamma^q)}$
  Output: (A,x) s.t. $A^{x+\gamma} = g_1$,
  i.e., s.t. $e(A,wg_2^{x})=e(g_1,g_2)$

- Linear [BBS04]:
  Input: $h,u,v \in G_1$, u^{α}, v^{β},
  Distinguish $h^{\alpha+\beta}$ from random.

Using SDH [BB04]

- Given q-SDH instance g_1, $w=g_2^{(y)}$, $g_2^{(y^2)}$, …, $g_2^{(y^q)}$ can compute
  - related parameters g_1',g_2',$w'=(g_2')^{(y)}$,
    - and q-1 SDH pairs (A_i,x_i) for these params (for any choice of x_i);
- But any additional SDH pair (A,x) for these params can be transformed into an SDH pair for the original params.
- This can be used to answer signing queries

Using Linear

Obtain ElGamal-style encryption:

- Keygen. pick $h \in G_1$, $x,y \in Z_p$;
  PK = $(u=h^{1/x}, v=h^{1/y})$; SK = (x,y)
- Encrypt $A \in G_1$: pick $\alpha,\beta \in Z_p$;
  output $(T_1=u^{\alpha}, T_2=v^{\beta}, T_3=h^{\alpha+\beta}A)$
- Decrypt (T_1,T_2,T_3): $A \leftarrow T_3/(T_1^{x}T_2^{y})$.

Secure under Linear, even when DDH is easy.

A ZKPK for SDH [BBS04]

- Shared params for SDH, Linear; prover also has SDH tuple (A,x).
- Prover first Linear-encrypts A:

$$T_1 \leftarrow u^{\alpha} \qquad T_2 \leftarrow v^{\beta} \qquad T_3 \leftarrow Ah^{\alpha+\beta}$$

- Prover and verifier then engage in ZKPK:

$$\text{ZKPK}\left[(\alpha,\beta,x): \begin{array}{cc} u^{\alpha}=T_1 & v^{\beta}=T_2 \\ e(T_3h^{-\alpha-\beta}, wg_2^x)=e(g_1,g_2) & \end{array}\right]$$

- With helper variables $\delta_1=\alpha x$, $\delta_2=\beta x$, this is a Schnorr PK of discrete logarithms:

$$\text{ZKPK}\left[\begin{array}{l}(\alpha,\beta,x,\delta_1,\delta_2): \\ \quad u^{\alpha}=T_1 \qquad v^{\beta}=T_2 \qquad T_1^x u^{-\delta_1}=1 \qquad T_2^x v^{-\delta_2}=1 \\ e(T_3,g_2)^x \cdot e(h,w)^{-\alpha-\beta} \cdot e(h,g_2)^{-\delta_1-\delta_2} = e(g_1,g_2)/e(T_3,w)\end{array}\right]$$

Group Sigs from SDH

- Derive SK of SDH pair from ZKPK above via Fiat-Shamir heuristic [FS86,AABN02]
  A signature is a 9-tuple $(T_1,T_2,T_3, c, s_\alpha,s_\beta,s_x,s_{\delta_1},s_{\delta_2})$
  (three elements of G_1, six of Z_p: 1533 bits)
- Group signature scheme:
  - Setup: $gpk = (g_1,g_2,h,u,v,w)$,
    $gmsk = (x,y)$, $sk_i = (A_i,x_i)$
  - Sign: output SK of (A_i,x_i) on message M
  - Verify: verify signature of knowledge
  - Open: decrypt (T_1, T_2, T_3) to recover A_i

Group Sig Properties

- Correct
- CPA-fully-anonymous (if Linear holds)
- Fully-traceable (if q-SDH holds)

- Short: 50% longer than (ordinary) RSA signature of comparable security
- Efficient: one pairing to verify, none to sign
- Extensible: revocation, verifier-local revocation, strong exculpability, ...

Extension: Revocation

- To revoke a user with key (A,x):
  - publish (A*,x), where $\psi(A^*) = A$.
  - update public key:

$$g_1' \leftarrow \psi(A^*) \qquad g_2' \leftarrow A^* \qquad w' \leftarrow g_2 \cdot (A^*)^{-x}$$

  - each user updates her private key (A_i, x_i):

$$A_i' \leftarrow \psi(A^*)^{1/(x_i - x)} / A_i^{1/(x_i - x)} \qquad x_i' \leftarrow x_i$$

- Note that revocation messages must be communicated to signers

Is DDH hard in G_1?

- Bilinear map gives algorithm for deciding co-DDH in (G1,G2)
- When there is a map Φ: G1 → G2, this is DDH in G1
- On MNT curves, there does not appear to be an efficiently computable map Φ.
- Using ElGamal instead of Linear encryption:
  1022-bit CPA-fully-anonymous group signatures
- Using Cramer-Shoup:
  1364-bit CCA2-fully-anonymous group signatures

Group Signatures with Verifier-Local Revocation [BS04]

- Three algorithms
  - Setup(λ,n) — λ: security param; n: # users
    outputs public key gpk, user keys sk_1, …, sk_n, <u>revocation tokens</u> rt_1, …, rt_n
    - Sign(sk_i,M) — outputs group signature σ on M
    - Verify(gpk,RL,σ) — outputs 'yes' or 'no'
      ('no' means invalid signature or revoked signer)
  - <u>Implicit tracing</u>: try Verify with RL as each $\{rt_i\}$
  - Correctness:

$$\text{Verify}(\text{gpk}, \text{RL}, \text{Sign}(\text{gpk}, \text{sk}_i, \text{M}), \text{M}) = \text{yes} \iff \text{rt}_i \notin \text{RL}$$

VLR Security: Traceability

- Adversary is given all revocation tokens in addition to public key
- He outputs a forged signature and a revocation list
- Goal is to create a valid signature σ^* that cannot be traced; or that traces to a user for whose private key he didn't ask; or to a revoked user

VLR Security: Selfless Anonymity

- Adversary is not given private keys; instead, has access to signing, private key, and revocation token oracles.
- Goal is to tell which of two users i_0 or i_1 created challenge signature σ^*.
- Adversary cannot request secret key or revocation token for either challenge user

A (non-ZK) PK for SDH

- Shared params for SDH;
  prover also has SDH tuple (A,x)
- Verifier picks random $u', v' \in G_2$, sends to prover
- Prover and verifier set $u = \psi(u')$, $v = \psi(v')$
- Prover ElGamal-"encrypts" A:

$$T_1 \leftarrow u^{\alpha} \qquad T_2 \leftarrow A v^{\alpha}$$

- Prover and verifier then engage in ZKPK:

$$\mathrm{ZKPK}\left[(\alpha, x): \quad u^{\alpha} = T_1 \quad e(T_2 v^{-\alpha}, w g_2^{x}) = e(g_1, g_2)\right]$$

- If A is encoded in (T_1,T_2) then $(u',v',T_1,T_2/A)$ is a co-Diffie-Hellman tuple

VLR Group Sigs from SDH

- Derive SK of SDH pair from ZKPK above
  A signature is a 7-tuple $(r, T_1, T_2, c, s_\alpha, s_x, s_\delta)$
  (r is an 80-bit nonce; signature is 1102 bits overall)

- Group signature scheme:
  - Setup: $gpk = (g_1, g_2, w)$, $sk_i = (A_i, x_i)$, $rt_i = A_i$
  - Sign: output SK of (A_i, x_i) on message M
  - Verify: verify signature of knowledge; ensure that no element of RL gives co-DH tuple

Conclusions

- Group signatures now have
  - rigorous definitions [BMW03]
  - real-world applications (TCPA,DSRC)
  - efficient constructions [BBS04]
- Verifier-local revocation [BS04] is appropriate when there are many signers, few verifiers
- Open problems:
  - Is DDH hard in G_1?
  - Practical group signatures without random oracles