

# Direct Anonymous Attestation (DAA): Ensuring privacy with corrupt administrators

Ben Smyth   Mark Ryan   Liqun Chen

School of Computer Science,  
University of Birmingham, UK  
{B.A.Smyth, M.D.Ryan}@cs.bham.ac.uk

HP Laboratories,  
Bristol, UK  
liqun.chen@hp.com

Tuesday, 3rd July 2007

# Trusted Computing

- ▶ Cryptographic guarantees about the state of a remote platform
- ▶ Allowing informed decisions as to whether to trust the platform
- ▶ Requires hardware device called a Trusted Platform Module (TPM)
- ▶ Example applications:
  - ▶ Mobile *ad hoc* networks
  - ▶ Grid computing
  - ▶ Corporate digital rights management

# Treachorous computing?

- ▶ Privacy advocates (and TCG members) voiced concerns
- ▶ Solution: server learns that a platform is trusted and not which particular one

## Privacy

- ▶ **Anonymity:** state of not being identifiable within a set of agents
- ▶ **Unlinkability:** inability to link actions performed by the same agent

# Direct Anonymous Attestation (DAA)

Developed in collaboration between Ernie Brickell (Intel), Jan Camenisch (IBM) & Liqun Chen (HP)

## TPM specification history

**v1.1:** Trusted third party

**v1.2:** Direct Anonymous Attestation (DAA)

## What's in a name?

**Direct** proof: without trusted third party

**Anonymous:** platform/user identity not disclosed

**Attestation:** membership claim (from TPM)

## Join protocol

- ▶ TPM chooses a secret message  $f$
- ▶ Obtain signature (aka attestation) on  $f$  from the *issuer*

## Sign/verify protocol

- ▶ Demonstrate knowledge of attestation to a *verifier*

## Terminology

**TPM:** A hardware device which is trustworthy

**Host/platform:** A device with an embedded TPM

**Issuer:** Responsible for managing membership

**Verifier:** An entity which remotely authenticates a host

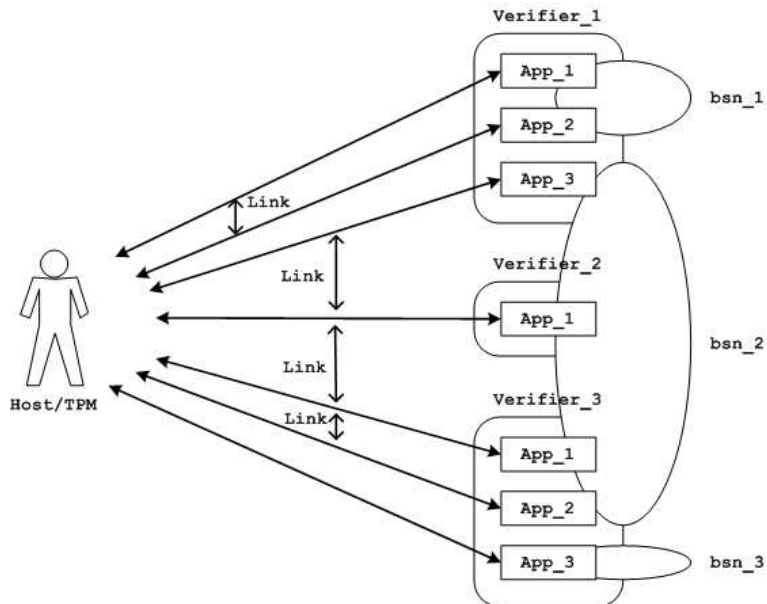
# DAA security properties

1. Only a trusted platform is able to authenticate
2. Privacy of a non-corrupt host is guaranteed by the sign/verify protocol:
  - 2.1 Interactions are anonymous
  - 2.2 Linkability (of transactions) is controlled by the user
3. Privacy is restored to a corrupted host if malicious software is removed
4. Mechanism to detect rogue members

## Provably secure

DAA has been shown to be secure in the provable security model

# Linkable transactions



# Inadequacy of provable security

**Provable security** aims to show the difficulty of breaking a scheme is as difficult as solving a supposedly difficult problem (e.g. integer factorisation)

Provable security on its own is insufficient:

- ▶ Effective attacks do not solve difficult problems, they discover weaknesses in the protocol

- ▶ Encryption

$$\{msg\}_K$$

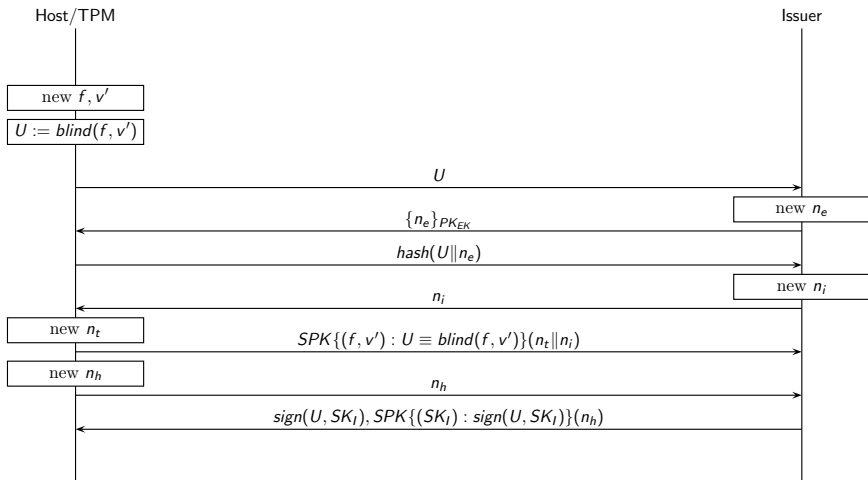
- ▶ Zero knowledge proofs of knowledge

$$PK\{(\alpha, \beta, \gamma) : y = g^\alpha h^\beta \wedge \hat{y} = \hat{g}^\alpha \hat{h}^\gamma \wedge \alpha \in [u, v]\}$$

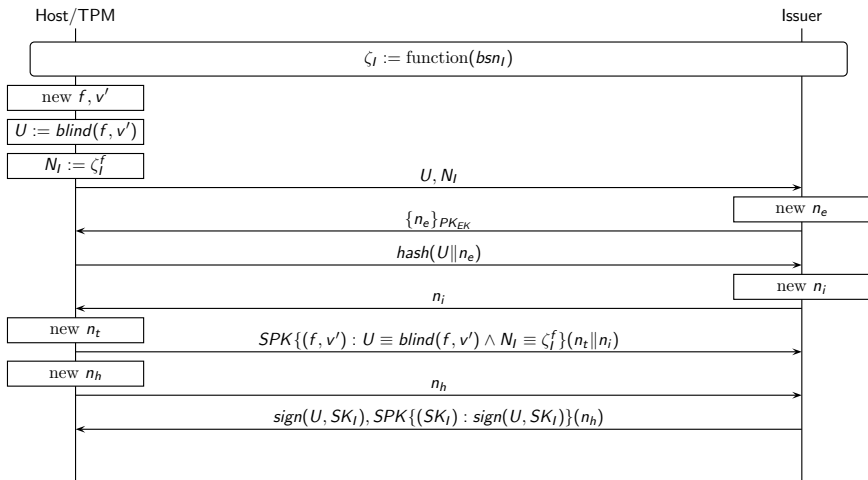
- ▶ Signature proofs of knowledge

$$SPK\{(\alpha, \beta, \gamma) : y = g^\alpha h^\beta \wedge \hat{y} = \hat{g}^\alpha \hat{h}^\gamma \wedge \alpha \in [u, v]\}(m)$$

# Join Protocol



# Join Protocol



# Sign/Verify Protocol

Host/TPM

Verifier

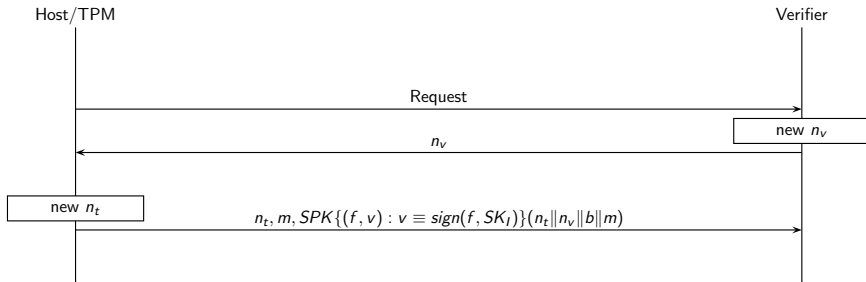
Request

new  $n_v$

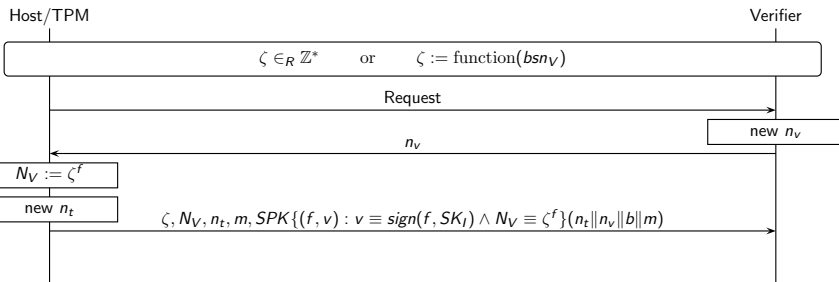
$n_v$

new  $n_t$

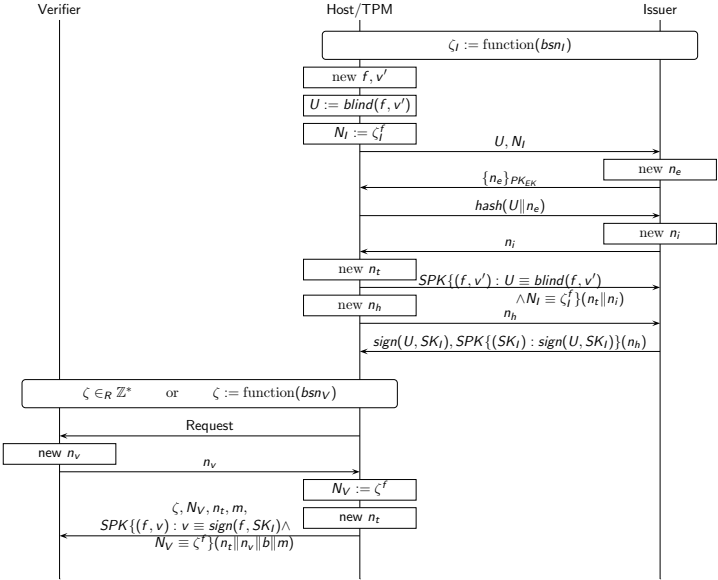
$n_t, m, SPK\{(f, v) : v \equiv \text{sign}(f, SK_f)\}(n_t \| n_v \| b \| m)$



# Sign/Verify Protocol



# DAA Protocol

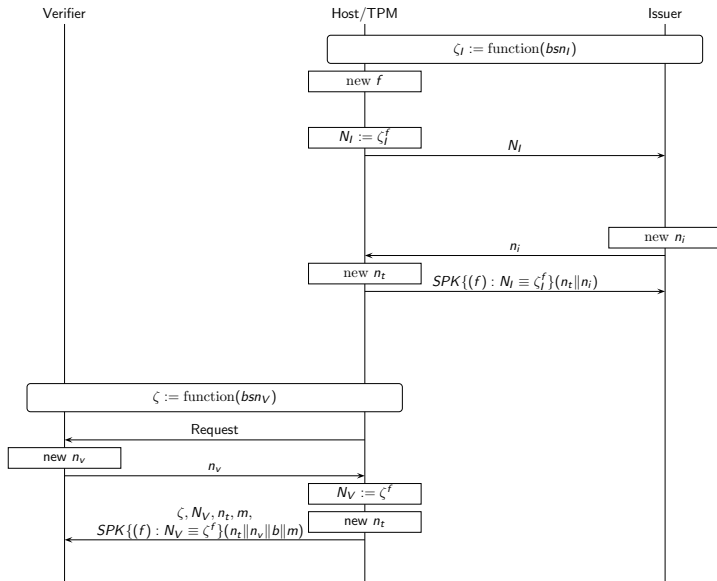


## Recall security properties

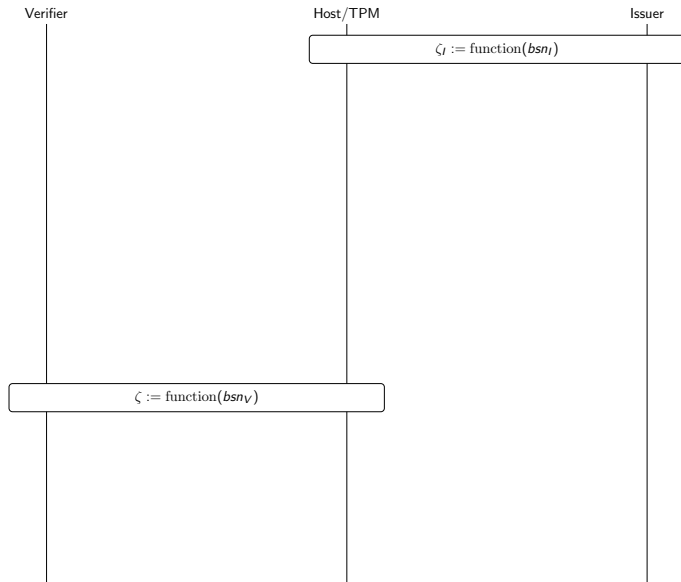
1. Only a trusted platform is able to authenticate
2. Privacy of a non-corrupt host is guaranteed by the sign/verify protocol:
  - 2.1 Interactions are anonymous
  - 2.2 Linkability (of transactions) is controlled by the user
3. Privacy is restored to a corrupted host if malicious software is removed
4. Mechanism to detect rogue members

A colluding issuer and verifier can violate security properties 2.1 & 2.2

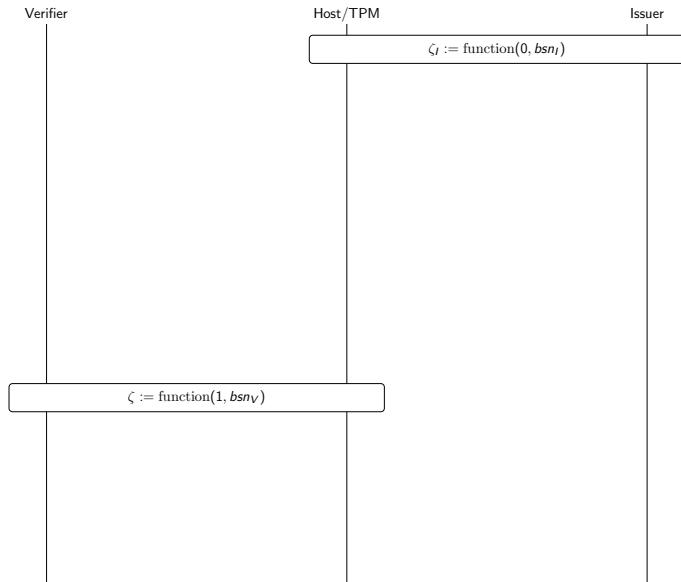
# Violating privacy



# Fix



# Fix



# Overcoming problems with DAA basenames

DAA does not make adequate provisions for the implementation and use of basenames.

1. **Ensuring user controlled linkability.** The user must be assured which verifier(s) will use a basename for what applications(s).
2. **Implementation.** The user must somehow manage a list of basenames. A naïve solution is to maintain a list of all basenames, however such a list is unfeasible.

# DAA basenames

Requirement: The host must be able to identify with which verifier/applications a basename should be used and the length of the basename list must be short.

Solution: Construct the basename from application/verifier/issuer specific data.

Application	DAA operation	Issuer/verifier data	Date	Other
1. Specification 2. URL 3. User ID 4. Password 5. Shared key 6. Other	1. DAA key issuing 2. PCR signing 3. AIK signing 4. External input signing 5. System input signing 6. Other	1. Issuer identity 2. Issuer public key 3. Verifier identity 4. Verifier public key 5. Auth request 6. Auth algorithm 7. Other	1. Start date 2. Expiry date 3. Other	1. Random data string* 2. Policy 3. Terms & conditions 4. Other

# Motivating authentication of the verifier

- ▶ Signing a declaration of membership leaks information
- ▶ Users may not wish to divulge their affiliations
- ▶ This motivates the authentication of verifiers

## Conclusion/further work

- ▶ Identified a weakness in the DAA protocol
  - ▶ Provided a fix which requires minimal alteration
  - ▶ The fix does not require any changes to the hardware TPM
- 
- ▶ Simpler solutions?
  - ▶ Finer-grained approach to privacy
  - ▶ Development of formal approaches for the verification of protocols such as DAA