On Tamper-Resistance from a Theoretical Viewpoint

The Power of Seals

Paolo Mateus   Serge Vaudenay

SQIG/IT and IST/TULisbon — Portugal

EPFL — Switzerland

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

http://lasecwww.epfl.ch/

The Impact of Setup Assumptions

- regular ZK protocol is well defined
  some ZK properties collapse in the CRS or RO model
  $\longrightarrow$ deniable ZK: Pass (Crypto'03)
- group signature provides anonymity when keys are well set up
  tricky things if key registered with/without proof-of-possession
  $\longrightarrow$ Ristenpart-Yilek (Eurocrypt'07)
- UC framework without setup assumptions is limited
  many issues using setup assumptions
  $\longrightarrow$ Barak-Canetti-Nielsen-Pass (FOCS'04)

☺ setup based on tamper resistance may ease things
$\longrightarrow$ Katz (Eurocrypt'07)

what is the real impact of tamper-resistance in setup assumptions?

1. **The Trusted Agent Model**
2. Zero-Knowledge in the Trusted Agent Model
3. Adding Key Registration
4. Some Attacks based on Trusted Agents

Trusted Agent: a Model for Tamper Resistance

- we add a special participant (tamper-resistant device)
- includes 1- a trusted boot loader, 2- a display, 3- an input port
- first input: a boot code (OS) C
- after boot complete: input/output defined by OS only
  C (or rather $h(C)$) concatenated to output
- input/output can be restricted by a participant (holder)
  holder can show the display to another participant
- if $[C : y]$ displayed by device, the reader is ensured that a Turing machine was initially set up with code C, then carried on some (unknown) interaction, and finally produced the output y

Commitment using a Trusted Agent — i

define code C:

1: receive x
2: pick a random sid
3: output receipt, sid
4: wait for new input
5: output open, sid, x

Commitment using a Trusted Agent — ii

commit protocol:

TA		**Alice**		**Bob**
load C	$\xleftarrow{C}$			
store x	$\xleftarrow{x}$			
	$\xrightarrow{[C\text{:receipt,sid}]}$	(show)	$\xrightarrow{[C\text{:receipt,sid}]}$	check, store sid

check means:

- check message comes from a TA
- check code C is as expected by the commitment protocol

Commitment using a Trusted Agent — iii

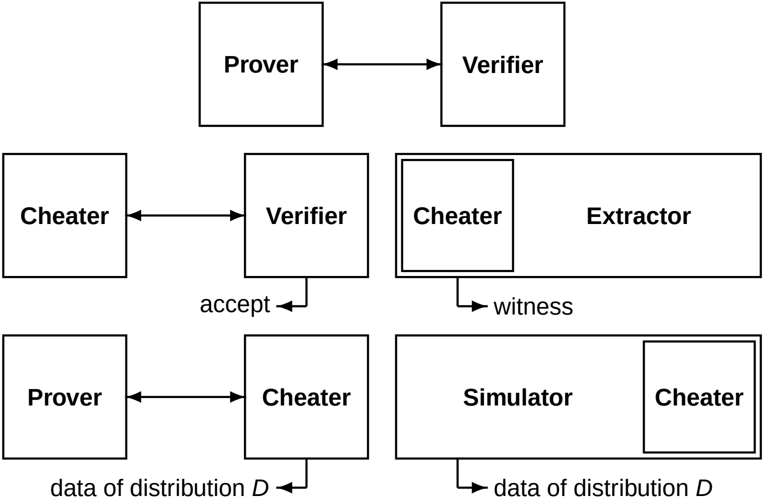
open protocol:

TA		**Alice**		**Bob**
	← dummy			
	[C:open,sid,x] →	(show)	[C:open,sid,x] →	check

check means:

- check message comes from a TA
- check code C is as expected by the commitment protocol
- check sid is the same

Zero-Knowledge



a proof of knowledge that leaks nothing that can later be used

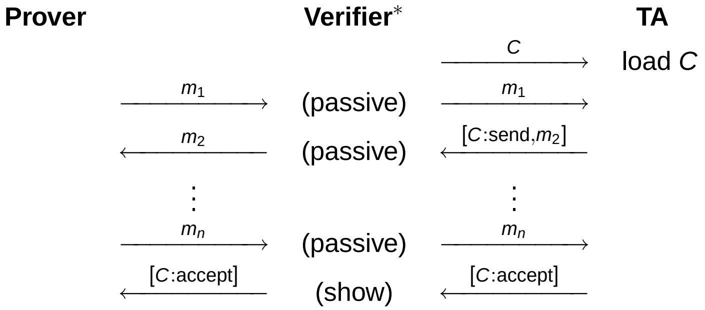
Trivial Zero-Knowledge for Relation R

TA		Prover		Verifier
load C	$\xleftarrow{C}$			
$b = R(x,w)$	$\xleftarrow{x,w}$			
	$\xrightarrow{[C:x,b]}$	(show)	$\xrightarrow{[C:x,b]}$	check

check means:

- check message comes from a TA
- check code C is as expected by the ZK protocol
- check x is as expected and $b = \text{true}$

Deniability loss in Regular ZK Protocols



final message cannot be simulated because it comes from a TA!
(TAs cannot be rewinded)

proof is offline transferable (thus not ZK)

Summary for the TA Model

- zero-knowledge becomes trivial if prover uses a TA
- when prover holds no TA:
  - regular ZK is no longer ZK (deniable) when malicious verifier uses TA
  - ZK survives if honest verifier can use a TA

Several Key Registration Models

- authority generates key pair and sends the public key to user! (key escrow)
- authority generates key pair and sends it to user (key escrow)
- user generates key pair and sends it to authority (key escrow)
- user generates key pair and sends the public key to authority and a self-signed certificate
- user generates key pair and sends the public key to authority and ZK-prove knowledge of secret key
- user generates key pair and sends the public key to authority (registered key may be a rogue key)

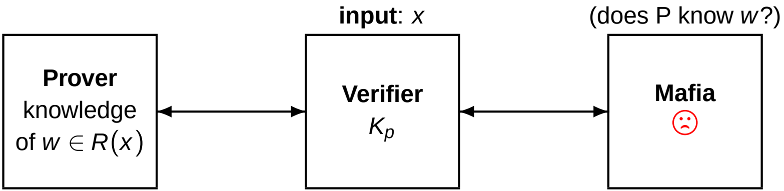
Key Registration with TA

- except for key escrow models, a TA could be used to register a key without giving the secret key
- registering users may later be able to prove ignorance of their secret key
- proof of ignorance can resurrect rogue key attacks

Two Non-Transferability Notions

- offline non-transferability (aka deniability):
  vulnerable to transfer attacks using a TA
- online non-transferability:
  vulnerable to rogue key registration (e.g. using a TA)

Mafia Fraud

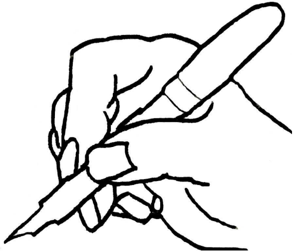


proof of knowledge of w

$\downarrow$

proof of knowledge of either w or a secret key attached to K_p

Invisibility Loss in Privacy-Enhanced Signatures



- signature verified through ZK protocols
  (e.g. undeniable signatures)

ZK proof for (in)valid signature can be transfered

Transferring Non-Transferable Proofs

- either a TA can be used to register a rogue key then prove ignorance of the secret key
- or key registration gives to the authority enough information to make a fake poof to the verifier

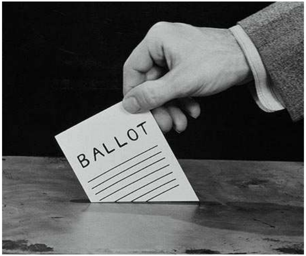
either transferability or key escrow!

Anonymity Loss in Group Signature

- either a TA can be used to register a rogue key then prove ignorance of the secret key
- or key registration gives to the authority enough information to impersonate a group member

either transferability or key escrow!

Selling Ballots



- use a TA to vote
- TA later proves vote (and get financial income for it)

Conclusion

- tamper-resistant device (if exist) can be maliciously used
- some cryptographic properties are more fragile than others
  - deniability in ZK (aka offline non-transferability)
  - (online) non-transferability
  - anonymity
  - receipt-freeness
- mind TAs when designing cryptographic protocols