Extensions to Multi-Party Computations

Birgit Pfitzmann
Universität des Saarlandes
<pfitzmann@cs.uni-sb.de>

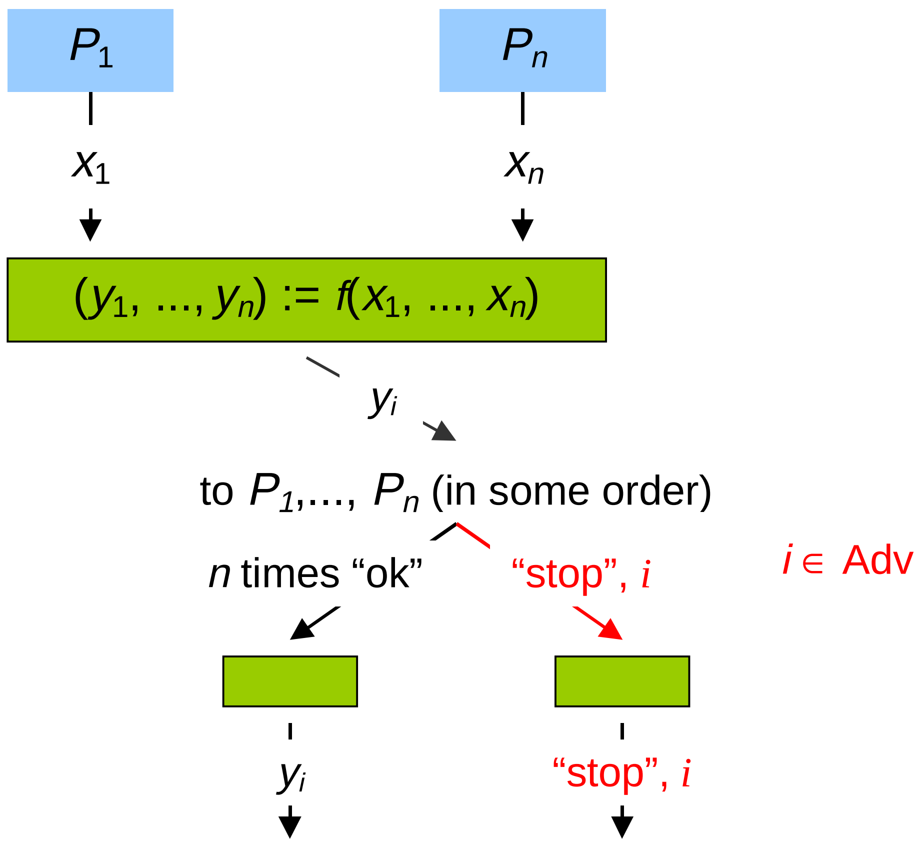
Michael Waidner
IBM Research Division, Zurich Research Lab
<wmi@zurich.ibm.com>

Table of Contents

IBM

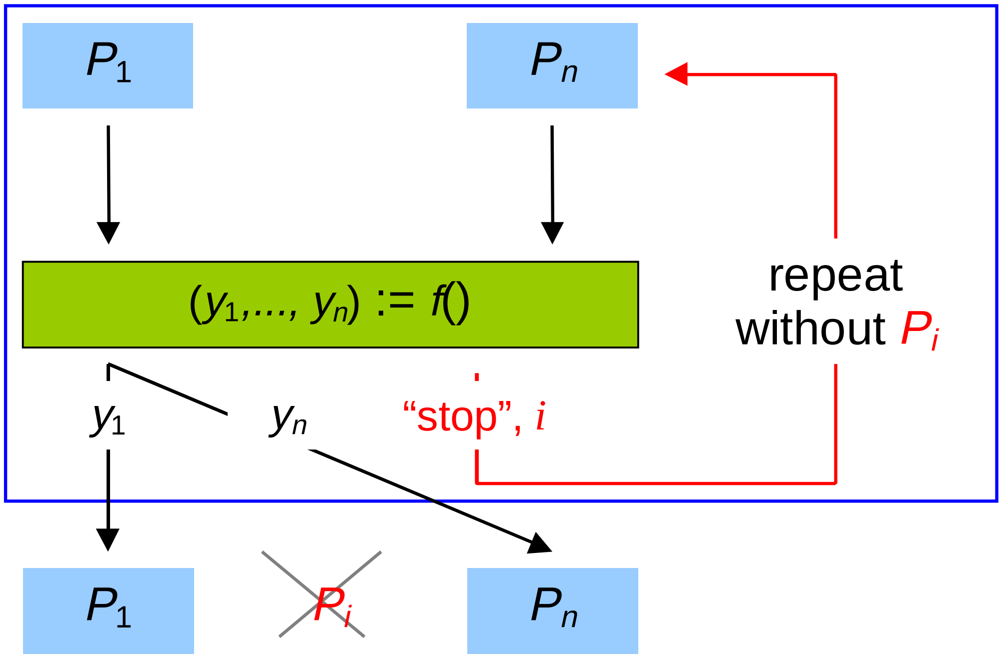
1 Use of Unfair Stopping

Motivation: Collective n-party key generation *without* honest majority (any $t < n$) [PW90, P96]



- Not fair: adversary learns output in any case
- Not in "standard" trusted host definitions
- Real protocol: [CDG88]
  - computational setting, any $t < n$
  - secrecy of inputs, even uncondional for P_1
  - either result correct or disrupter is identified

1.1 Repetition of MPCs with unfair stopping



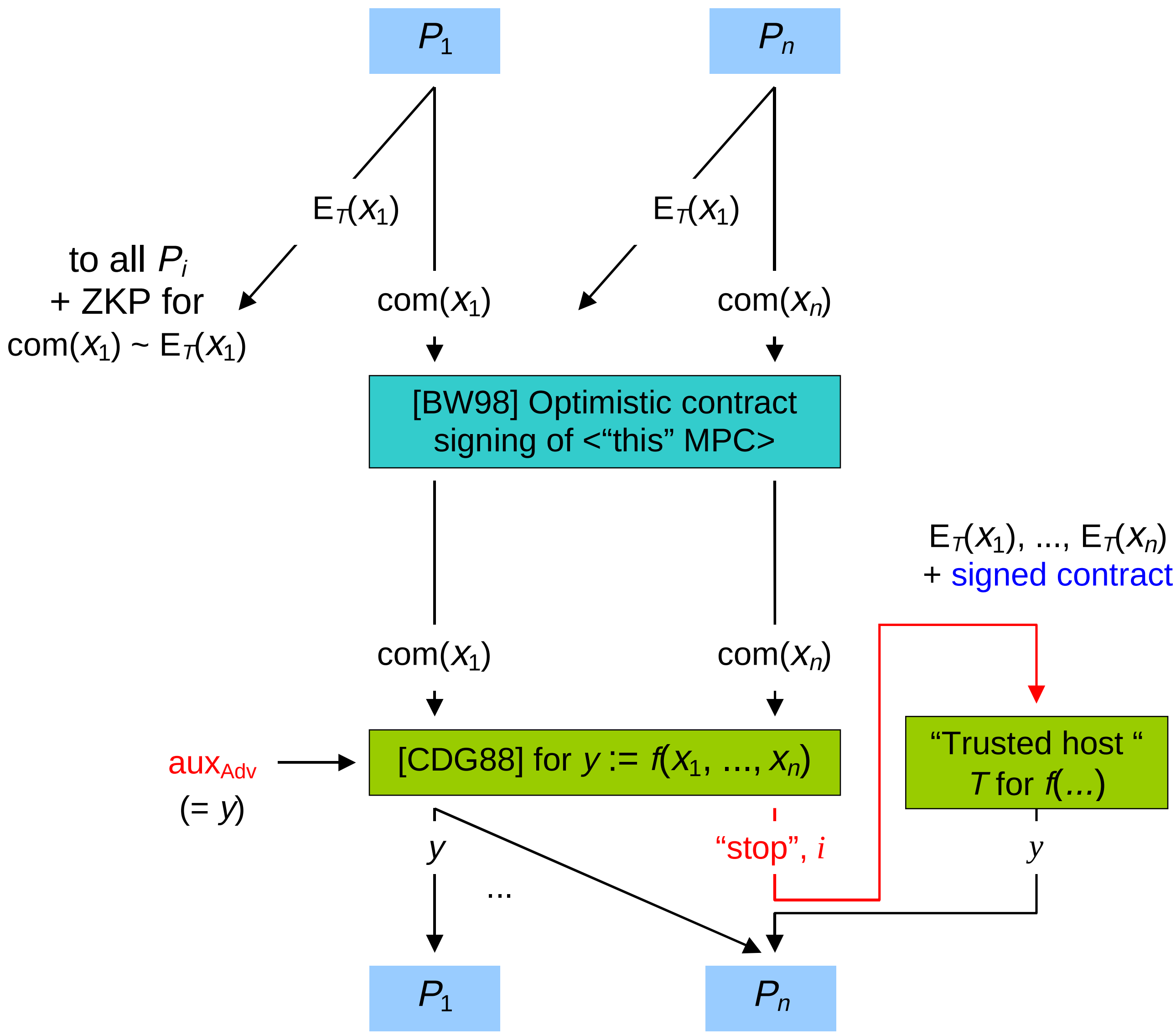
- Fail-stop signatures [PW90, P96]
  - No private inputs (only random values)
  - $y_1 = (sk, pk)$, $y_i = pk$ ($i = 2, \ldots, n$)
  - ≈ "adversary can select at most one out of n correct random keys" [sketch in PW90]
- In case of private inputs:
  - specific $f()$ only: don't need dishonest input
  - if "input commit" successful then repeat "compute $f()$" until it succeeds

2 Optimism

Motivation: Fair *n*-party contract signing with third party for exception handling [DLM82 → ASW96]

- Fair protocols using a trusted third party *T*
  - usually simple protocols (~ trusted host ...)
  - *T* might become a bottleneck if involved in all protocol runs
  - thus, be optimistic:
    - use *T* for exception handling only
    - less simple protocols
- Old concept
  - General approach [DLM82]
  - Fair exchange of “values” [BP90]
  - Contract signing [BGMR88]
  - Certified mail, contract signing [M97]
  - Generic fair exchanges [ASW96, ASW97, ...]
  - Multi-party protocols [ASW96, BW98]
- Can be applied to any MPC problem ...

Construction [BW98]



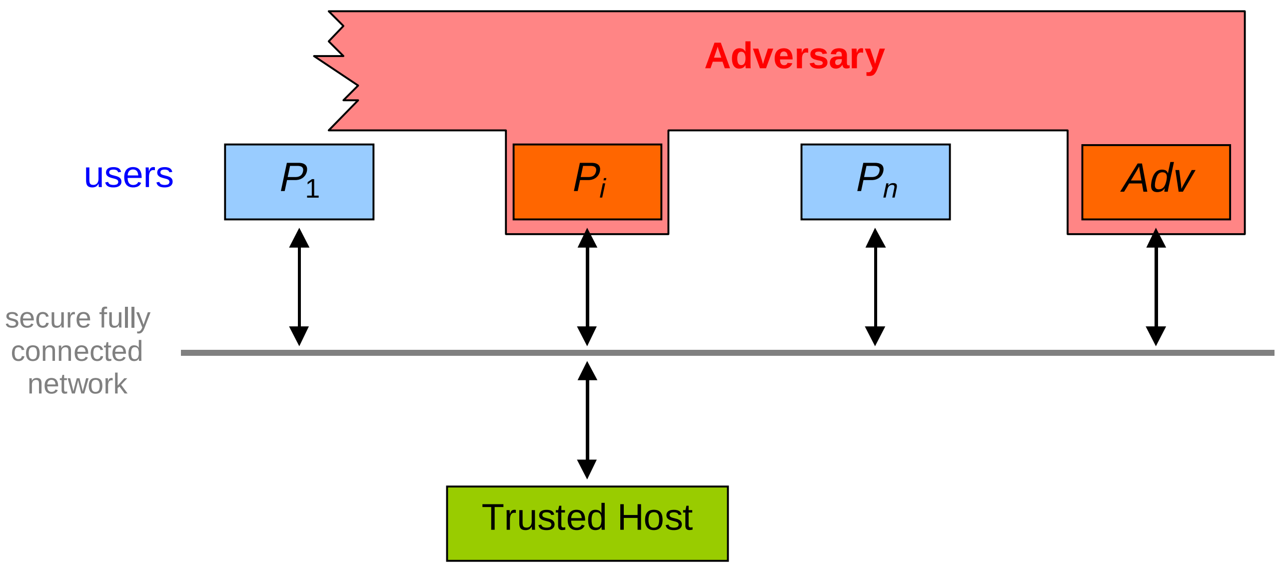
- Usual definition of MPC, but $t = n$-1
- Deterministically fair
  - needs third party [C86]
  - w/o third party: polynomially small advantage [GHY88, BG89, GL90]

3 Reactivity

Motivation: Games [GMW87], payment schemes, signature schemes, etc., are all reactive systems

- Main additions to the non-reactive setting
  - System has to keep state
    $\Rightarrow$ Internal variables: state
    $\Rightarrow$ External variables: user input / output
  - Behaviour of users must be considered
    $\Rightarrow$ Explicit "user machine," interacting with adversary
    - "Old" concept
      - "Message finder" [GM84]
      - Signature schemes secure against active attacks [GMR88]
      - General reactive systems [P93, PW94]
      - General MPC [G98, C98]

- User machine
  - Computationally as unrestricted as the adversary (necessarily not less ...)
  - Might interact *outside* the system
  - Might create most fortunate situation for the adversary
  - Thus: for *all* users ...



  - Sometimes minimally benign user behaviour is mandated

4 Individual Requirements

Motivation: Unify definitions for different kinds of services [P93, PW94, P96]

General topic

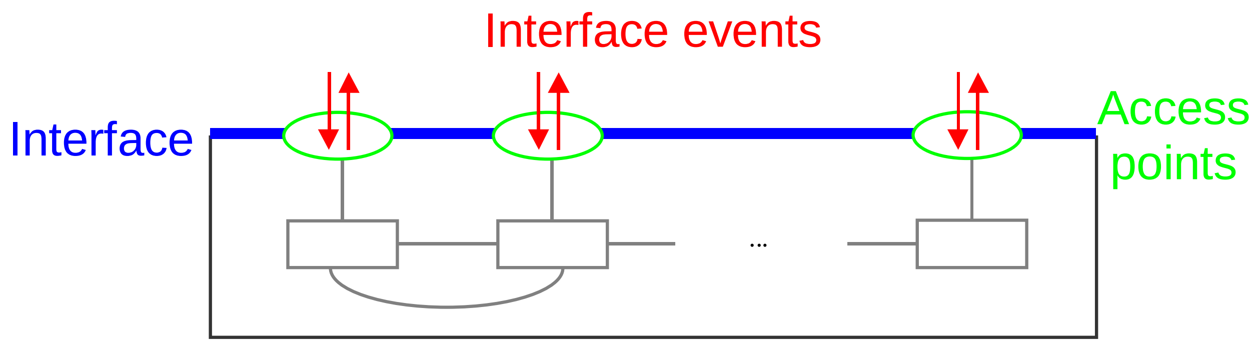
- Any MPC can be computed securely

Another way to look at the definitional problem

- Typical MPC problems
  - Signatures, payment, contract signing, ...
- Usually not completely specified
  - “Signatures” w/ or w/o “fail-stop property”
  - “Cash” w/ or w/o “offline transferability”
  - Trusted host always complete!
- Specific definitions for individual problems
  - [GM84, GMR88, N98, ...]; no trusted host models
- Our goal
  - Specify (minimal and optional) individual requirements on service only
  - Define the “ideal” service
  - Give a general cryptographic semantics to any such service

4.1 Security Requirements

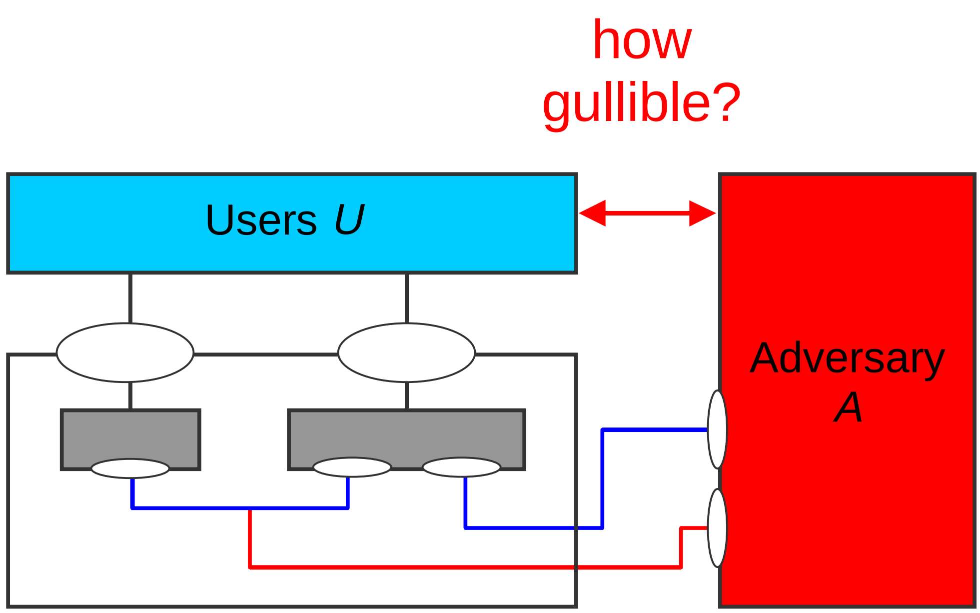
What is a service?



- Interleaved sequences of interface events
  - payer in: (“pay”, tid, 100$, to Birgit)
  - payee in: (“receive”, tid, 100$, from Michael)
  - bank_1 in: (“allow”, tid, 100$, from Michael, to Birgit)
  - bank_1 out: (“deduct”, tid, 100$, from Michael)
  - bank_2 out: (“add”, tid, 100$, to Birgit)
  - payer out: (“paid”, tid, 100$, to Birgit)
  - payee out: (“received”, tid, 100$, from Michael)
- Characterize set of allowed sequences, e.g., using some temporal logic (e.g., [PW96])
  - {payer}: “paid” not without “pay”
  - {payer, payee}: “pay” followed by “received”
  - {payer, bank_1}: “deduct” not without “pay”

4.2 Structure

- Req's independent of "implementation"!
- Structure of system providing service
  - Some model for interacting (poly) algs
  - ∀ users ...
  - Adversary *A*
    - Can interact with users *U* outside system
    - Altered system interface for *A*



  - In principle same problems & approach
    as for trusted host model for reactive systems
- For integrity requirements: *U* can be part of *A*

- Problems if one really tries it, e.g.,
  - structure might depend on sub-protocol, also in interface
  - identities and “tid” in interfaces
  - there is no complete, exact model of interacting machines usable for “trusted host” or our “structures”

4.3 Cryptographic semantics

- Principle

$\forall$ (poly) adversaries A (prob. interactive algo.)
$\forall$ (poly) users U "

$Prob_{wrong}(k)$ is “small”:

- $= 0$
- $< \text{poly}(sys\_pars) \cdot 2^{-k}$
- $< {}^{1}/\text{poly}(k)$

with

$Prob_{wrong}(k) :=$

$$\text{Prob}\Big[Run(Sys_{Rest}(k) \times A(k) \times U(k))|_{\text{Interface}} \notin Sem_{standard}(req)\Big]$$

4.5 Privacy

- Adversary does not get a specific information
  - Difficult to catch all hidden channels between users and adversary on the service level [S94]
  - General problem with all digital models ...
- System fulfills req's, and does nothing else
  - No additional interface events: complete specification
  - Specifies adversary's interface completely: leads naturally to trusted host model

$\Rightarrow$ Specification of secure systems in two steps:

  - Minimal service for integrity
    - Simple and intuitive approach
    - Definitions valid for larger classes, not for single services or systems only
- Complete system for privacy

5 Summary

Somewhat driven by the need to prove systems that use general MPCs as subprotocols ...

- We needed “non-standard” models of MPC
  - Unfair stopping
    $\rightarrow$ repetition for key generation
  - Optimistic computations
- Real-world systems are naturally reactive
  - Quantification over all users
  - Composition of reactive systems [PW92, PS96]

Service w/o privacy can be specified as “usual”

- Use some temporal logic
- Attach cryptographic semantics to requirements

Work in progress

- Proofs for the non-standard models
- Composability of reactive systems?
- How to deal with privacy requirements in service specifications

References

ASW96 N. Asokan, M. Schunter, M. Waidner: Optimistic Protocols for Multi-Party Fair Exchange; IBM Research Report RZ 2892, IBM Zurich Research Laboratory, Nov 1996

BG89 D. Beaver, S. Goldwasser: Multiparty Computation with Faulty Majority; 30th Symposium on Foundation of Computer Science (FOCS) 1989, IEEE Computer Society, 1989, 468-473

BGMR90 M. Ben-Or, O. Goldreich, S. Micali, R. L. Rivest: A Fair Protocol for Signing Contracts; IEEE Transactions on Information Theory 36/1 (1990) 40-46

BP90 H. Bürk, A. Pfitzmann: Value Exchange Systems Enabling Security and Unobservability; Computers & Security 9/8 (1990) 715-721

BW98 B. Baum, M. Waidner: Multi-Party Contract Signing and Certified Consensus; IBM Research Report RZ xxxx, IBM Zurich Research Laboratory, June 1998

C86 R. Cleve: Limits on the Security of Coin Flips When Half the Processors are Faulty; 18th Symposium on Theory of Computing (STOC) 1986, ACM, New York 1986, 364-369

C98 R. Canetti: Security and Composition of Multi-Party Cryptographic Protocols; June 5, 1998

CDG88 D. Chaum, I. B. Damgård, J. van de Graaf: Multiparty Computations ensuring privacy of each party's input and correctness of the result; Crypto '87, LNCS 293, Springer-Verlag, Berlin 1988, 87-119

DLM82 R. A. DeMillo, N. A. Lynch, M. J. Merritt: Cryptographic Protocols; 14th Symposium on Theory of Computing (STOC) 1982, ACM, New York 1982, 383-400

G98 O. Goldreich: Secure Multi-Party Computation; June 6, 1998

GHY88 Z. Galil, S. Haber, M. Yung: Cryptographic computation: secure fault-tolerant protocols and the public-key model; Crypto '87, LNCS 293, Springer-Verlag, Berlin 1988, 135-155

GL91 S. Goldwasser, L. Levin: Fair Computation of General Functions in Presence of Immoral Majority; Crypto '90, LNCS 537, Springer-Verlag, Berlin 1991, 77-93

GM84 S. Goldwasser, S. Micali: Probabilistic Encryption; Journal of Computer and System Sciences 28 (1984) 270-299

GMR88 S. Goldwasser, S. Micali, R. L. Rivest: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks; SIAM Journal on Computing 17/2 (1988) 281-308

GMW87 O. Goldreich, S. Micali, A. Wigderson: How to play any mental game - or - a completeness theorem for protocols with honest majority; 19th Symposium on Theory of Computing (STOC) 1987, ACM, New York 1987, 218-229

M96 S. Micali: Certified E-Mail with Invisible Post Offices; presented at RSA 97

N98 M. Naor: Talk on the different notions of secure encryption schemes; given at Monte Vertia Workshop on Cryptographic Protocols, March 1998.

P93 B. Pfitzmann: Sorting Out Signature Schemes; 1st ACM Conference on Computer and Communications Security, acm press 1993, 74-85

P96 B. Pfitzmann: Digital Signature Schemes — General Framework and Fail-Stop Signatures; LNCS 1100, Springer-Verlag, Berlin 1996

PS96 B. Pfitzmann, M. Schunter: Asymmetric Fingerprinting; Eurocrypt ‘96, LNCS 1070, Springer-Verlag, Berlin 1996, 84-95

PW90 B. Pfitzmann, M. Waidner: Formal Aspects of Fail-stop Signatures; Interner Bericht 22/90 der Fakultät für Informatik, Universität Karlsruhe, Dec. 1990

PW92 B. Pfitzmann, M. Waidner: How to Break and Repair a "Provably Secure" Untraceable Payment System; Crypto '91, LNCS 576, Springer-Verlag, Berlin 1992, 338-350

PW94 B. Pfitzmann, M. Waidner: A General Framework for Formal Notions of "Secure" Systems; Hildesheimer Informatik-Berichte 11/94, April 1994

PW96 B. Pfitzmann, M. Waidner: Properties of Payment Systems: General Definition Sketch and Classification; IBM Research Report RZ 2823, IBM Zurich Research Laboratory, May 1996

S94 M. Schunter: Spezifikation von Geheimhaltungseigenschaften für reaktive kryptologische Systeme; Diplomarbeit am Institut für Informatik, Universität Hildesheim, Januar 1994

Our reports: <http://www.semper.org/sirene>