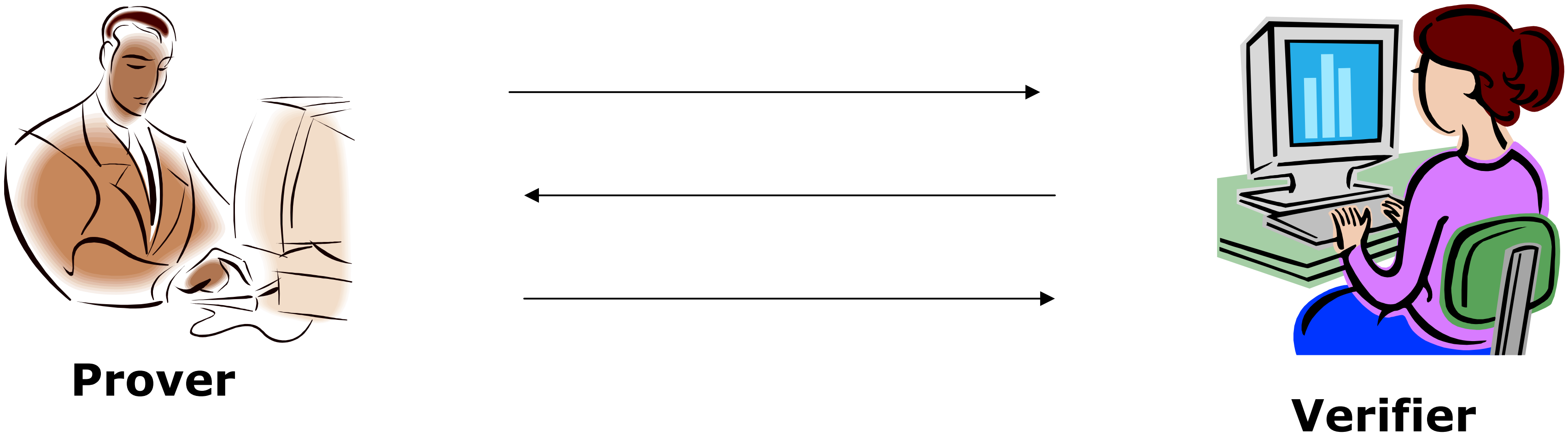
Batch Schnorr Id Scheme and Applications

Rosario Gennaro (IBM)
Darren Leigh (MERL)
Ravi Sundaram (NEU)
William Yerazunis (MERL)

Outline

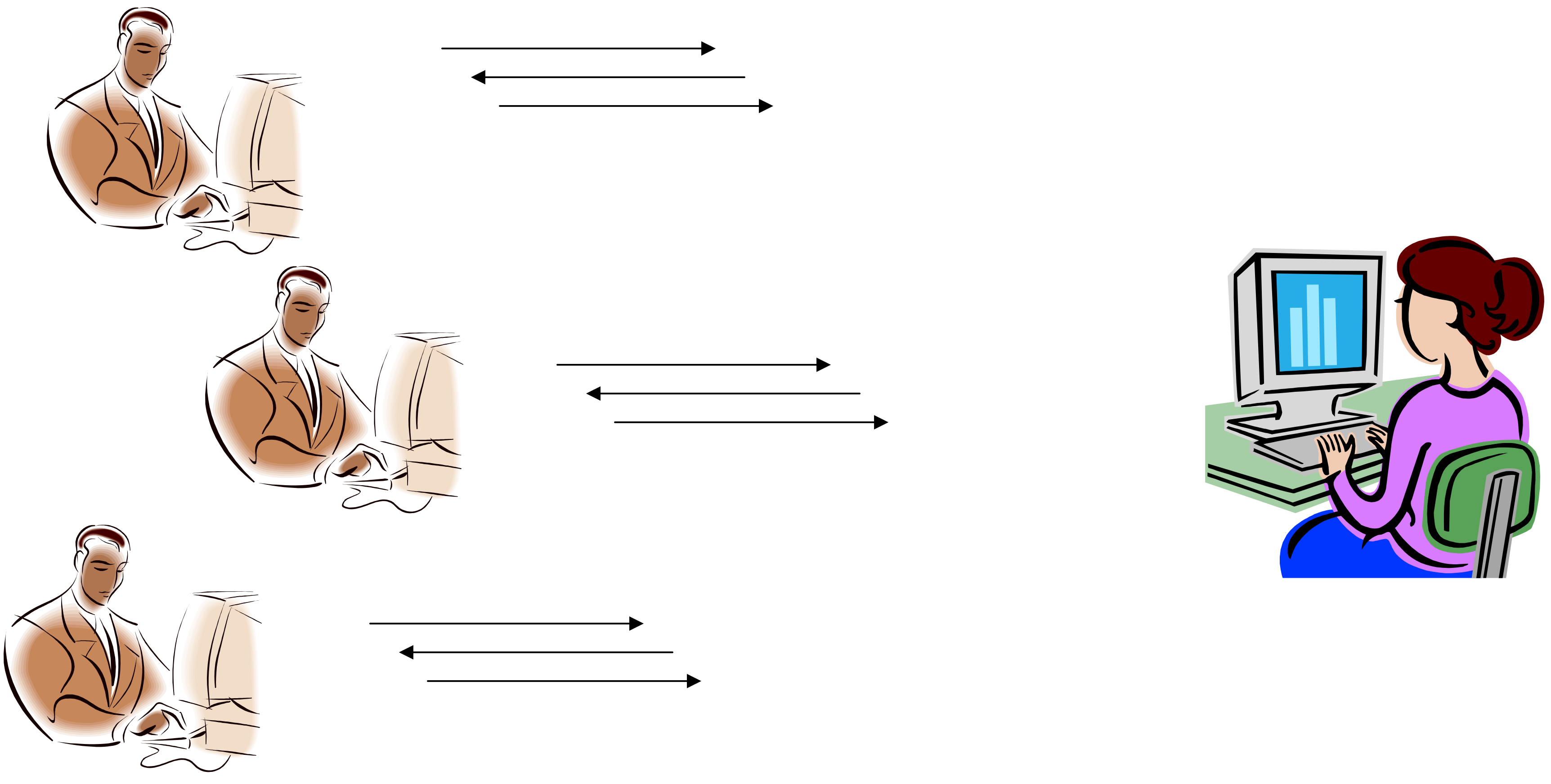
- Identification Schemes
  - Schnorr’s scheme based on discrete log - S’91
- Main Contribution:
  - *Batching:* Running many at the cost of one
- Applications
  - Privacy preserving authorization
  - Low Bandwidth Communication Devices
    - Implementation using a novel LED-based technology

Identification Schemes



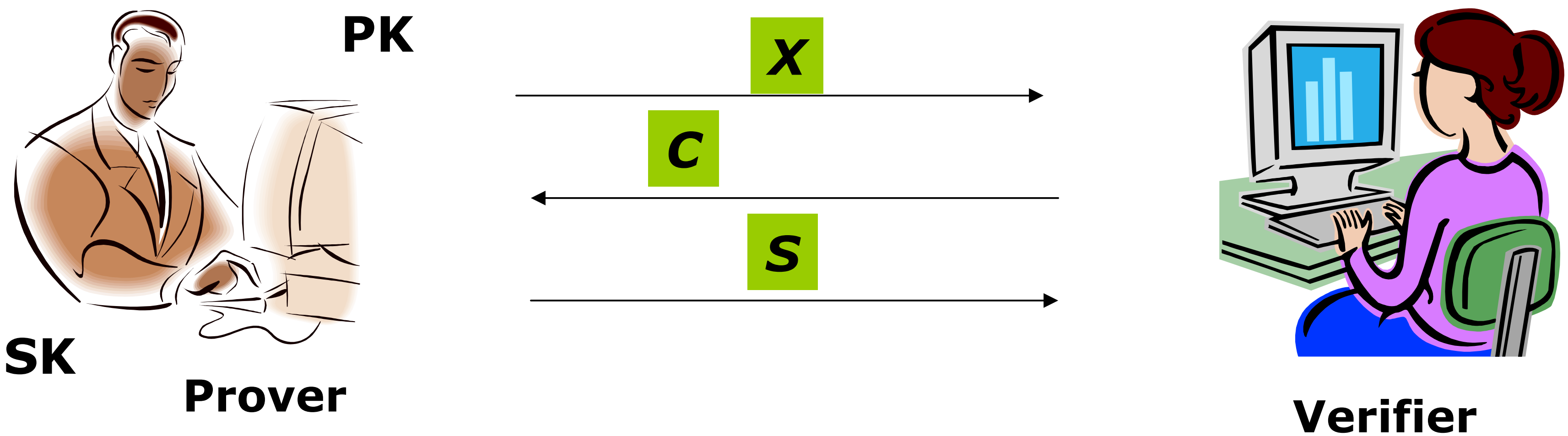
At the end of the interaction the Verifier knows she talked to the Prover, but she is not able to impersonate him

Concurrent Identification Schemes



Allow the Verifier to interact concurrently with many Provers, but still at the end she is not able to impersonate any of them

Proofs of Knowledge



Proof of Knowledge: Given oracle access to the Prover we can extract SK

Zero-Knowledge: Transcripts can be simulated without knowledge of SK → no information about SK → no impersonation

Schnorr's Proof for Discrete Log

$y=g^w$

w

Prover

$X=g^r$ →

C in $[1..2^t]$ ←

$S=r+Cw$ →

Verifier

$g^S \stackrel{?}{=} X\,y^C$

Proof of Knowledge: Given X,C,S and X,C',S we can compute $W=(S-S')/(C-C')$

Zero-Knowledge: Honest Verifier chooses C at random
Simulator: chooses C,S at random and sets $X=g^S\,y^{-C}$

What about bad Verifiers

- Proving both ZK and extraction is tricky
  - But can be done (CDM’00)
- Concurrent ZK is problematic
  - Simulation requires rewinding of Verifier
  - Can run in exp time (DNS’98)
  - Requires timing assumption to bound number of concurrent executions
- Still impersonation is hard (BP’02)
  - Schnorr is a secure concurrent ID scheme
  - Under the one-more inversion Dlog assumption
    - get ***k*** dlog input to invert
    - but can an query a dlog oracle ***k-1*** times

Proving Knowledge of *d* discrete logs

- Assume I have $y_1=g^{w1}\ldots y_d=g^{wd}$
  - Want to prove that I know $w_1\ldots w_d$
- Run Schnorr's Protocol ***d*** times
  - ***O(d)*** communication and cost for both parties
- Use batch exponentiation (BGR'98)
  - Run ***d*** copies of Schnorr's protocol
    - Verifier checks them all probabilistically with only one (more complicated) check
  - Still ***O(d)*** communication and computation for Prover
- Can we do better?

Yes! We can!

- Batch the whole Schnorr's protocol
  - Prover sends ***one*** commitment ***X***
  - Verifier sends ***one*** challenge ***C***
    - ***log d*** bits longer
  - Prover sends ***one*** answer ***S***
    - Which simultaneously verifies all y_i's
- Communication is virtually the same as in a ***single*** run of Schnorr's protocol
  - ***log d*** bits more are sent
- Computation is also improved
  - Prover:almost the same as a ***single*** execution
    - Only ***2d*** more multiplications
  - Verifier: ***d/2*** more work than a single run

Batch Schnorr

$y_1=g^{w1}\ldots y_d=g^{wd}$

Prover

$w_1\ldots w_d$

$X=g^r$ →

C in $[1..2^{t+logd}]$ ←

$S=r+Cw_1+C^2w_2+\ldots+C^dw_d$ →

Verifier

$g^S \mathrel{?=} X\, y_1^{C}\, y_2^{C2} \ldots y_d^{Cd}$

Security Properties

- Batch-Schnorr is
  - honest-verifier zero-knowledge
    - Simple to prove:
    - Simulator chooses ***C,S*** at random
    - Set ***X*** as in Verifier's verification equation
  - a proof of knowledge of $w_1 \ldots w_d$
  - a concurrently secure identification scheme
    - Proofs in next slides

Proof of Knowledge

- Ask ***d+1*** different challenges $C_1 \ldots C_{d+1}$
  - On the same commitment ***X***
- Get $S_1 \ldots S_{d+1}$
  - A linear system of ***d+1*** equations in the ***d+1*** unknowns $r, w_1 \ldots w_d$
  - Van der Monde matrix over $C_1 \ldots C_{d+1}$
    - → non-singular
  - Now find $w_1 \ldots w_d$

Concurrently Secure ID Scheme (1)

- Proof similar to BP'02
  - Uses the one-more inversion assumption
- Get ***d*** group elements $y_1 \ldots y_d$ to invert
  - Use them as the public key
- For each execution the adversary starts as a verifier
  - Ask for a group element ***X*** and use it as first message
  - To answer challenge ***C*** query dlog oracle
    - on $X\, y_1^{C}\, y_2^{C2} \ldots y_d^{Cd}$ to get the right answer ***S***
- Oracle queries:
  - We ask for ***k*** group elements
    - We need to invert them ***all***
  - We query Dlog oracle ***k-d*** times

Concurrently Secure ID Scheme (2)

- Now adversary runs an impersonation attack
- Use previous extraction to find $w_1 \ldots w_d$
  - Thus finding the dlog of d of the given group elements $y_1 \ldots y_d$
  - Then use verification equation to find the discrete log of the various X of the previous phase

Efficiency Comparison

- GQ-Protocol (GQ'88) is about 3 times more efficient than Schnorr's
  - For typical security parameters
- Thus when proving 3 or more identities simultaneously Batch-Schnorr is better than 3 executions of GQ
- Open Problem: an efficient batching for GQ

Applications: Privacy Preserving Authorization

- Access Control to resources (e.g. data)
  - Users have privileges
  - Access to a resource granted to users who own a specific subset of privileges
- Possible solution:
  - Each user is given a certified public key
  - Certificate specifies user's privileges
  - User runs ID protocol to access resources
- Shortcomings:
  - Impossible to delegate some privileges
  - When accessing a resource user reveals ***all*** his privileges
    - That resource may not require them all
    - Privacy violation
      - User reveals his security clearance when he only needs to prove his credit rating

Privacy Preserving Authorization

- Associate each privilege with a key
- Resource Access:
  - User proves the ***minimal*** set of privileges needed
  - Runs all the ID schemes in parallel
  - Use batching to improve efficiency
- Privacy
  - Verifier only learns that the user can access the given resource, not his other privileges
  - Assumes no collusions
    - Two colluding verifiers can reconstruct the union of the privileges used by a party
    - Group-signature based solutions (CL'02) guarantee privacy even with collusion
      - But they are less efficient
      - Batching techniques can be used to improve those solutions as well
        - They use simultaneous proofs of multiple ID's too

Implementation

- Implemented Batch-Schnorr
  - Suitable for low-bandwidth devices
- Use novel LED-based technology DYL’02
  - Light Emitting Diodes
  - Used as a bi-directional communication device
    - LEDs also “sense” incoming light
  - Another contribution of our work
    - We show that this technology is robust for crypto applications

Implementation Details

- Prover’s Device
  - A small microprocessor (smart-card)
    - 8-bit instruction words
    - 5 MIPS
    - 16KByte Storage
  - Connected to a LED
- Verifier’s Device
  - LED connected to a PC
- Communication
  - 250 bits/second
  - Range: just a few centimeters
- Full scale implementation
  - 200-bit prime order subgroup modulo a 1500 bit prime
  - Challenge length 95 bits
  - 32 identities proved in one Batch-Schnorr execution

Implementation Picture