Secure and Efficient Metering

Moni Naor and Benny Pinkas

Eurocrypt '98

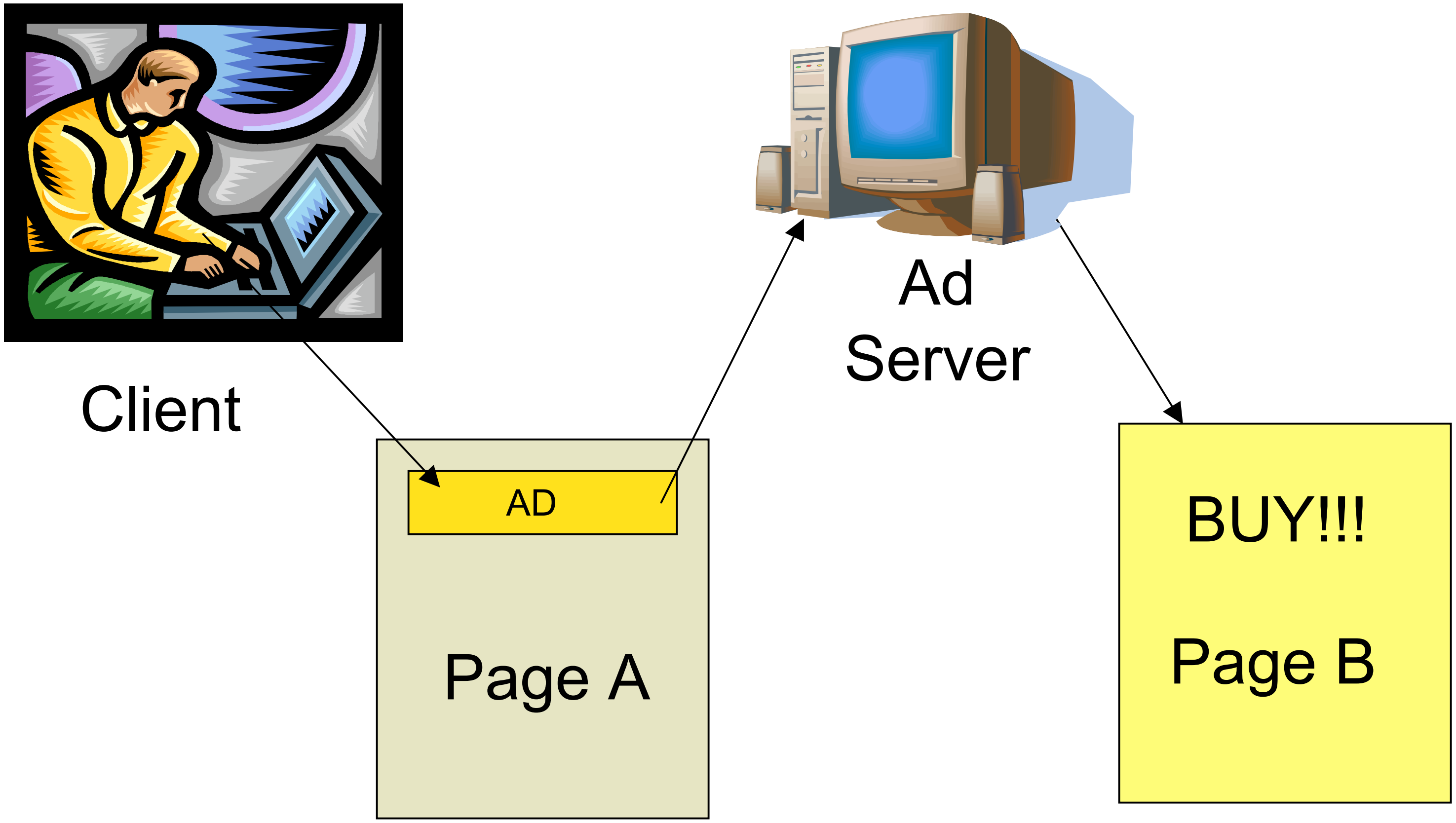
Contents

- Motivation
- One approach
- Lightweight Security
- Secure and Efficient Metering

Motivation

- Advertising
  - Webpage popularity
  - Cost
- Measure server & client interaction
- Royalties payment

Pay-Per-Click Scheme



Client

AD

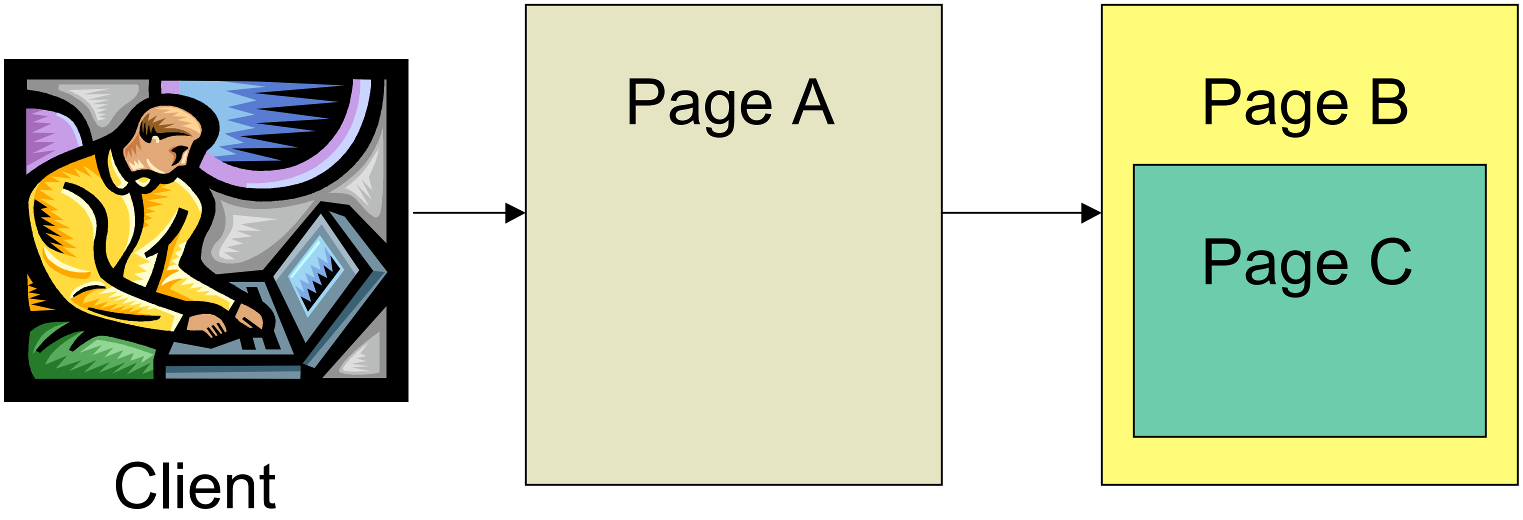
Page A

Ad Server

BUY!!!

Page B

Hit Inflation



Client

Page A

Page B

Page C

- Alternatives
  - Pay-per-sale
  - Pay-per-lead

SAWM: A Tool for Secure and Authenticated Web Metering

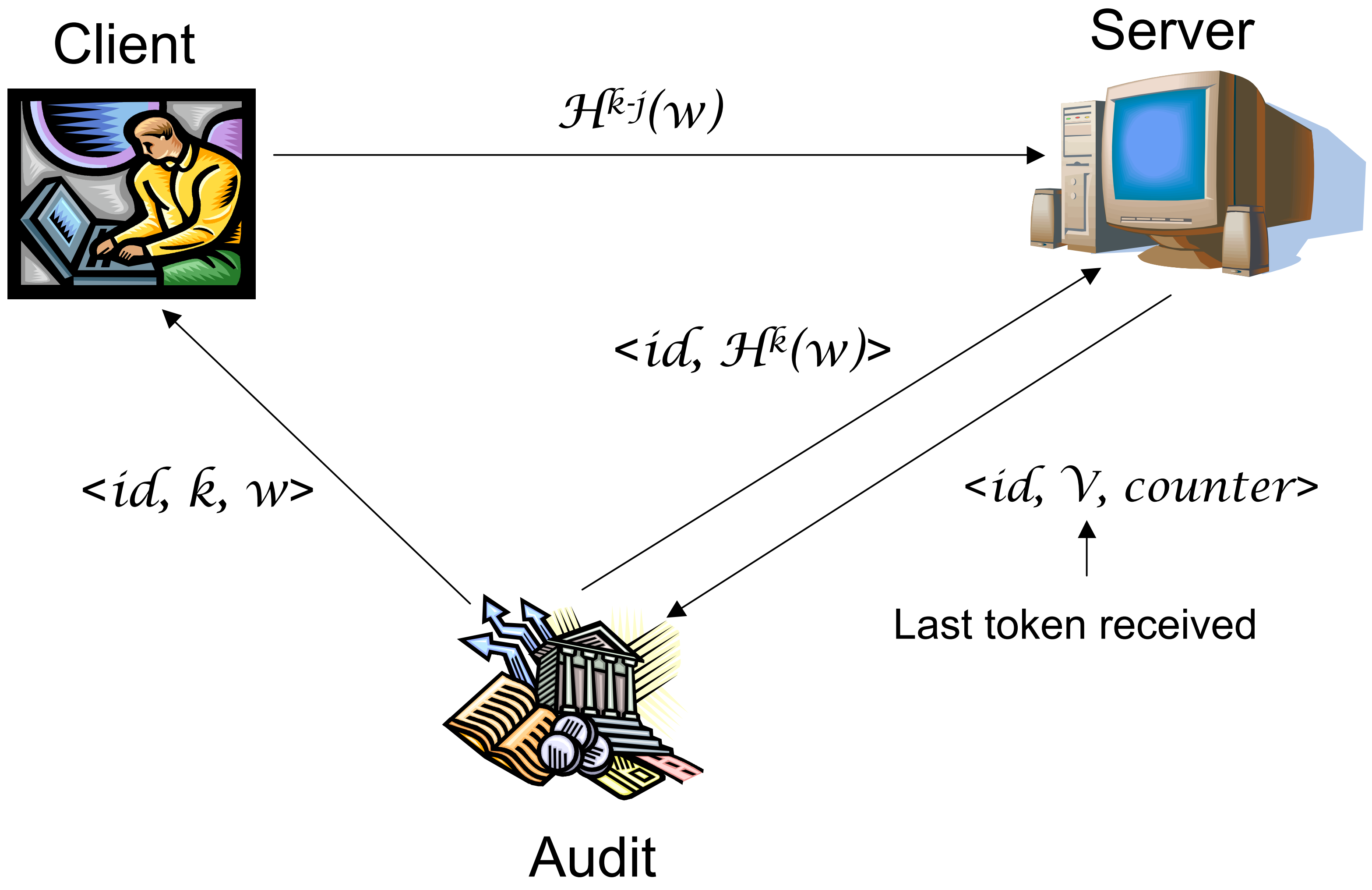
Blundo and Cimato

Proceedings of the 14th International Conference on Software engineering and knowledge engineering 2002

SAWM: A Tool for Secure and Authenticated Web Metering

- Hash chaining
- Three participants
  - Audit Agency
  - Client
  - Server
- Parameters
  - Random seed w
  - Hash function $\mathcal{H}$
  - Client identifier id
  - Number of applications k

SAWM Protocol



Client

Server

$\mathcal{H}^{k\text{-}j}(w)$

$<id, \mathcal{H}^{k}(w)>$

$<id, k, w>$

$<id, \mathcal{V}, counter>$

Last token received

Audit

Shortcomings

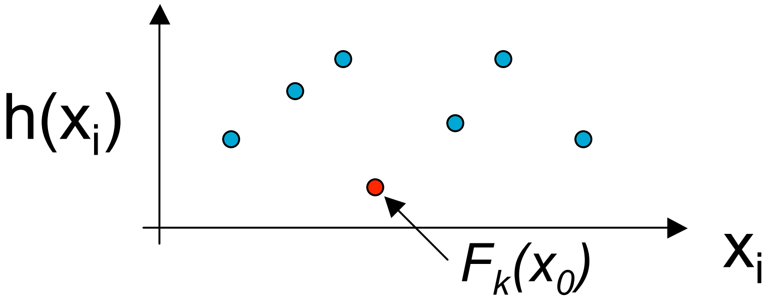
- Requires client & audit agency interaction
- Client and server can collude
- Corrupt servers can share client tokens
- Fake servers can collect tokens

Auditable Metering with Lightweight Security

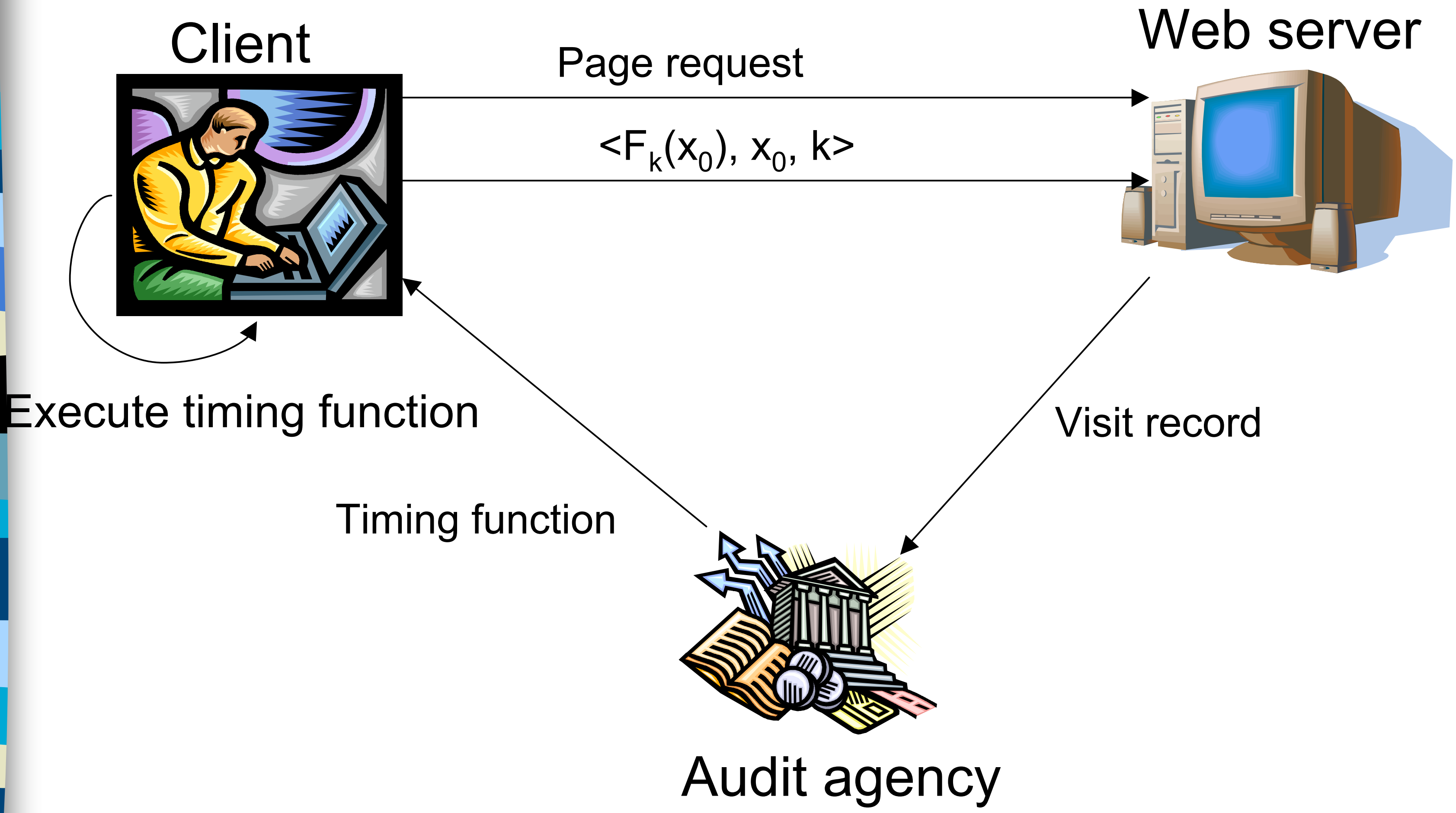
Franklin and Malkhi
Financial Crypto 1997

Auditable Metering with Lightweight Security

- Hash function h
- Timing function F
  - Apply hash function iteratively k times to x_0 such that $x_{j+1} = h(x_j)$
  - $F_k(x_0) = min\{x_j\}$, where $0<j\leq k$



Auditable Metering with Lightweight Security



Client

Page request

$<F_k(x_0), x_0, k>$

Web server

Execute timing function

Visit record

Timing function

Audit agency

Lightweight Security Auditing

- Method 1
  - Determine low probability visit records $<F_k(x_0),x_0,k>$
  - Verify these values
- Method 2
  - $y = F_k(x_0)$
  - Estimator function $\mu(y)$ that estimates k'
  - Check if estimator function approximates timing function

Lightweight Security Shortcomings

- Client can cheat server
- Client can collude with server
- Does not take into account different processing power of clients
- Costly verification
- Security based on statistical probabilities

Secure and Efficient Metering

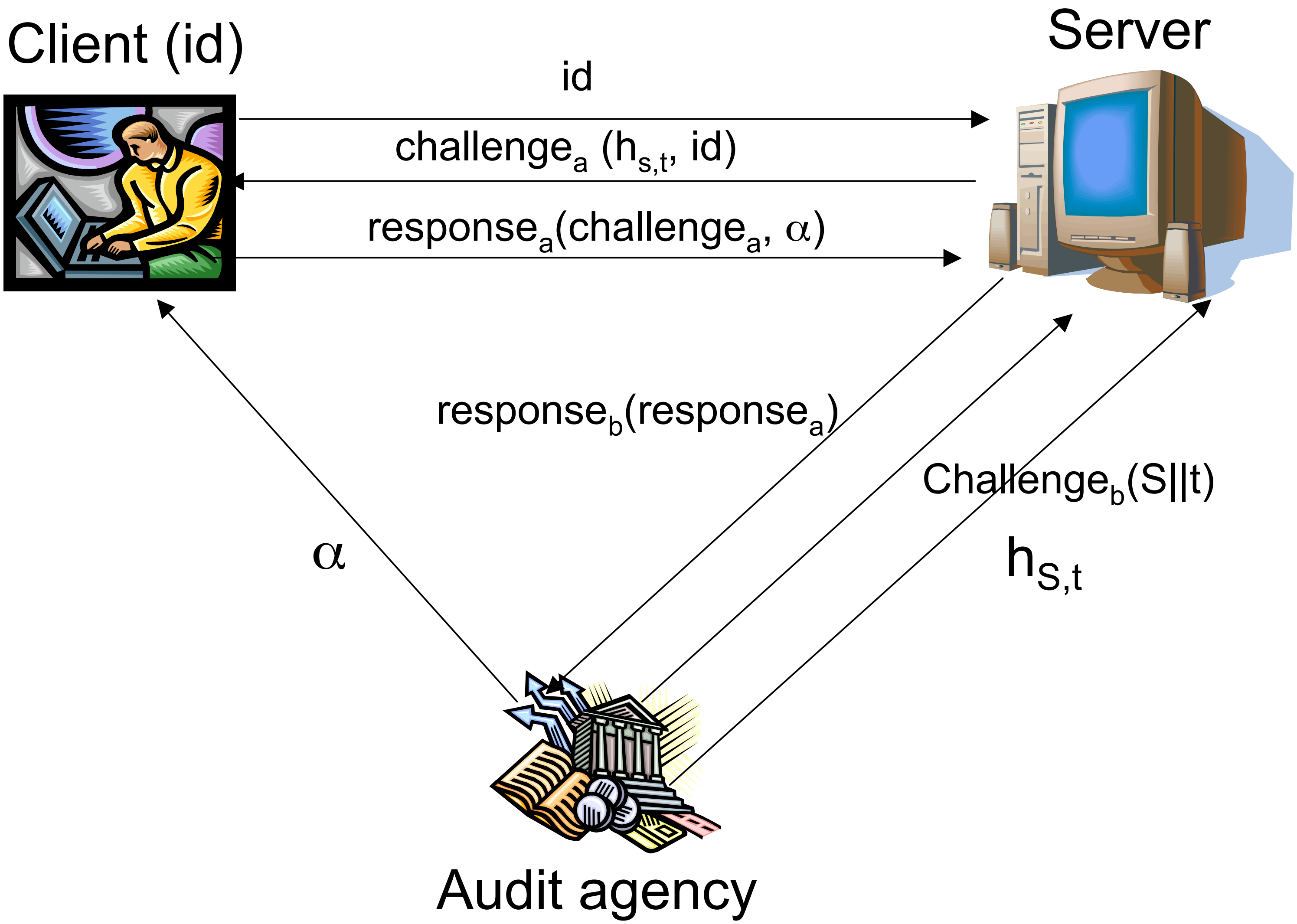
Naor and Pinkas

EuroCrypt ‘98

Secure and Efficient Metering

- Uses variant of Shamir secret sharing scheme
- Cryptographically secure scheme
- Requirements
  - Security
  - Efficiency
  - Accuracy
  - Privacy
  - Turnover

General Metering Scheme



Client (id)

Server

id

challenge$_a$ ($h_{s,t}$, id)

response$_a$(challenge$_a$, α)

response$_b$(response$_a$)

Challenge$_b$(S||t)

α

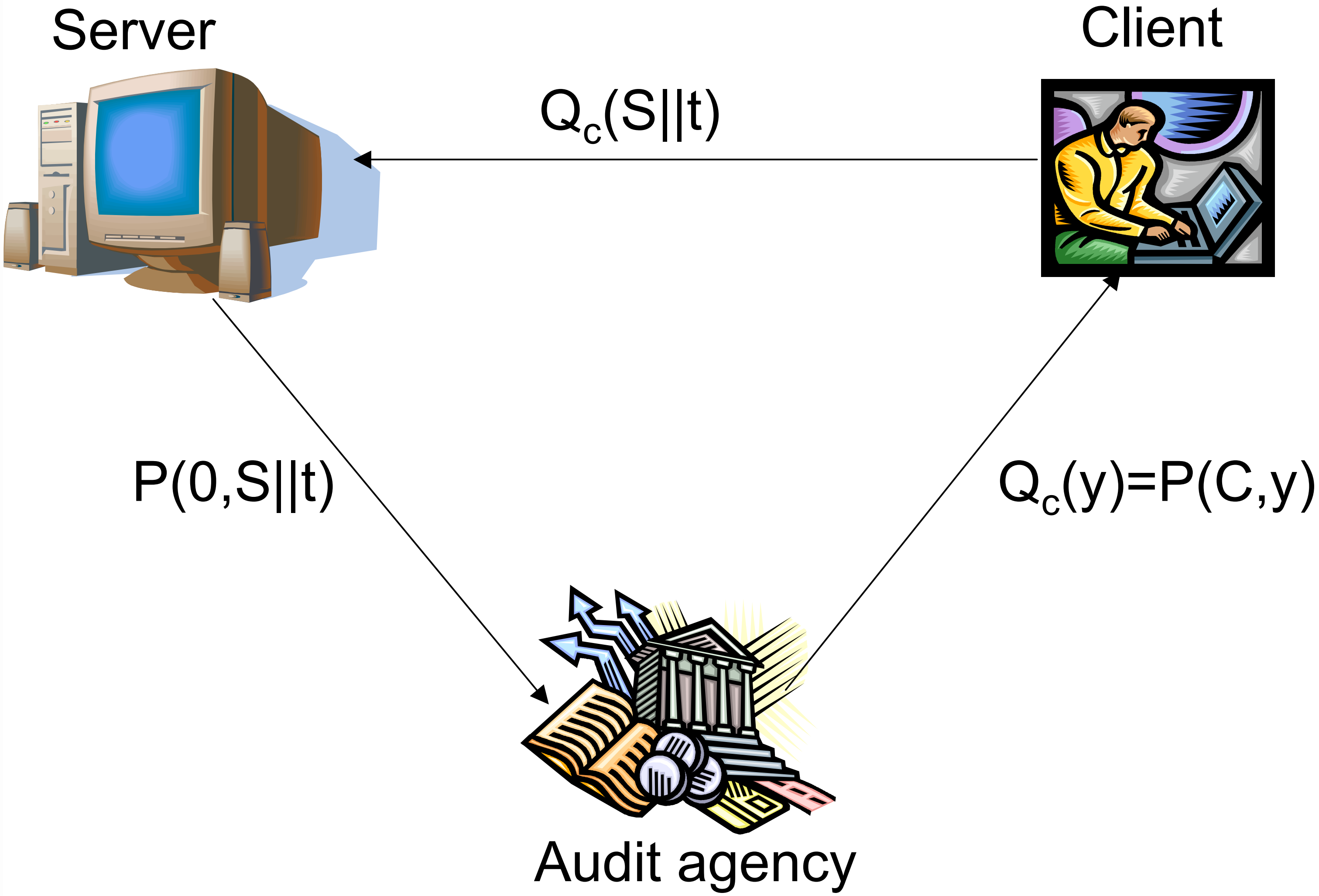
$h_{S,t}$

Audit agency

Secure & Efficient Metering Parameters

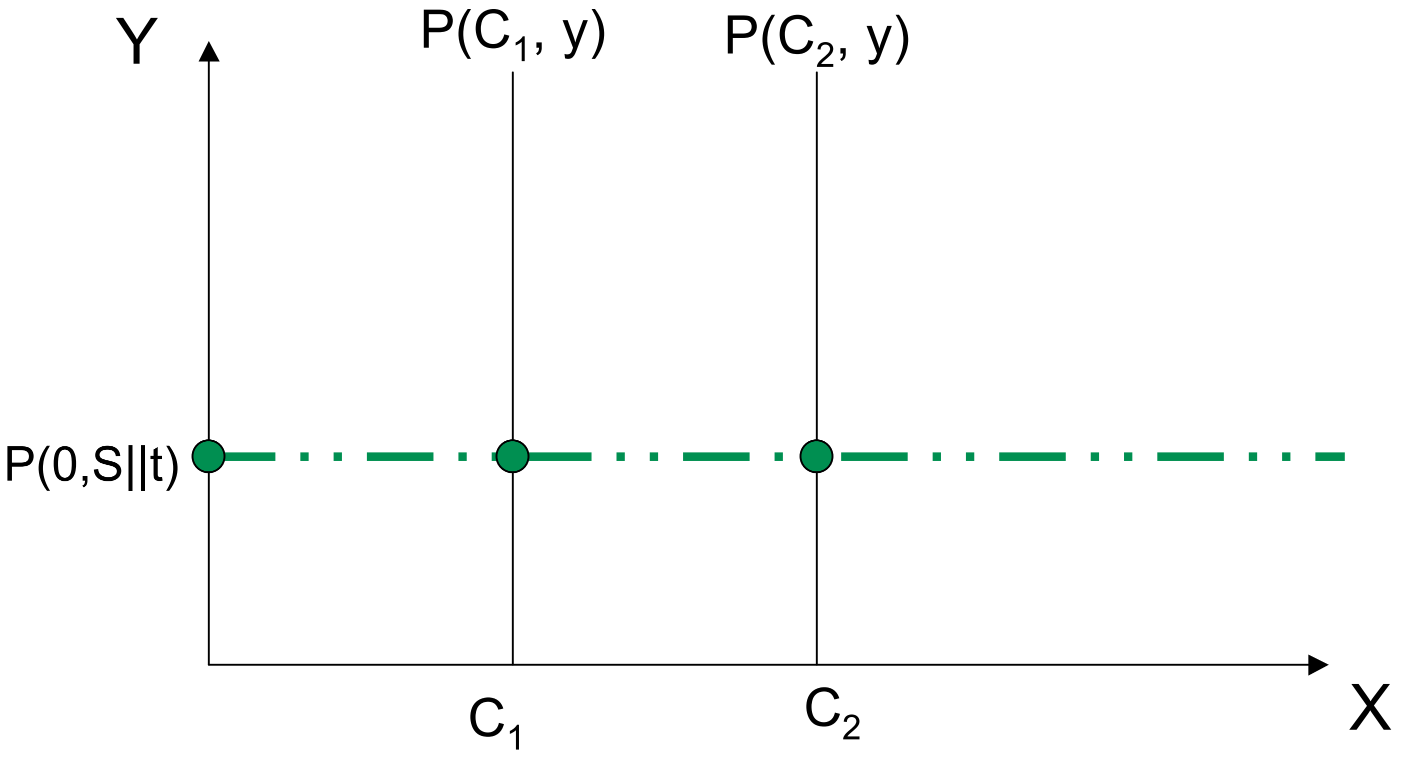
- Bivariate polynomial: *$P(x,y)$*
  - Degree $k-1$ in x
  - Degree $d-1$ in y
  - Finite field Z_p
  - Selected by audit agency
- Client value: *C*
- Server value: *S*
- Time frame: *t*

Secure and Efficient Metering Scheme



Calculating P(0, S||t)

- Use Lagrange interpolation



Security Analysis

- Without *k* visits, server has *1/p* chances of finding *P(0, S||t)*
- Corrupt clients can collude with servers
- Corrupt servers can donate client information from previous time frames
- Polynomial *P* replaced every *d* times frames

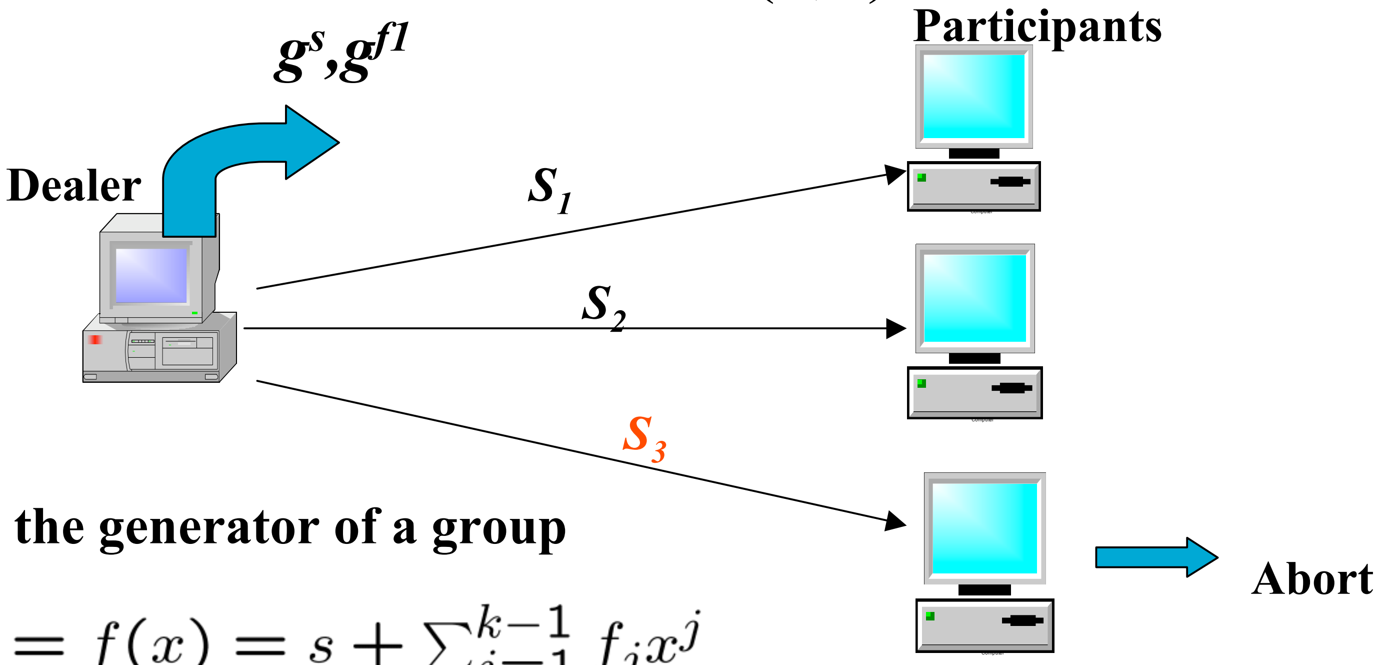
Robustness

- Corrupt clients can give the server wrong values
- Even with wrong values, a server should still be able to prove it had *k* visits
- Non-interactive verifiable secret sharing

Robustness

- Verifiable Secret Sharing for Shamir's scheme [Feldman87]

(2,3) VSS scheme



Participants

g^s, g^{f1}

Dealer

S_1

S_2

S_3

Abort

g is the generator of a group

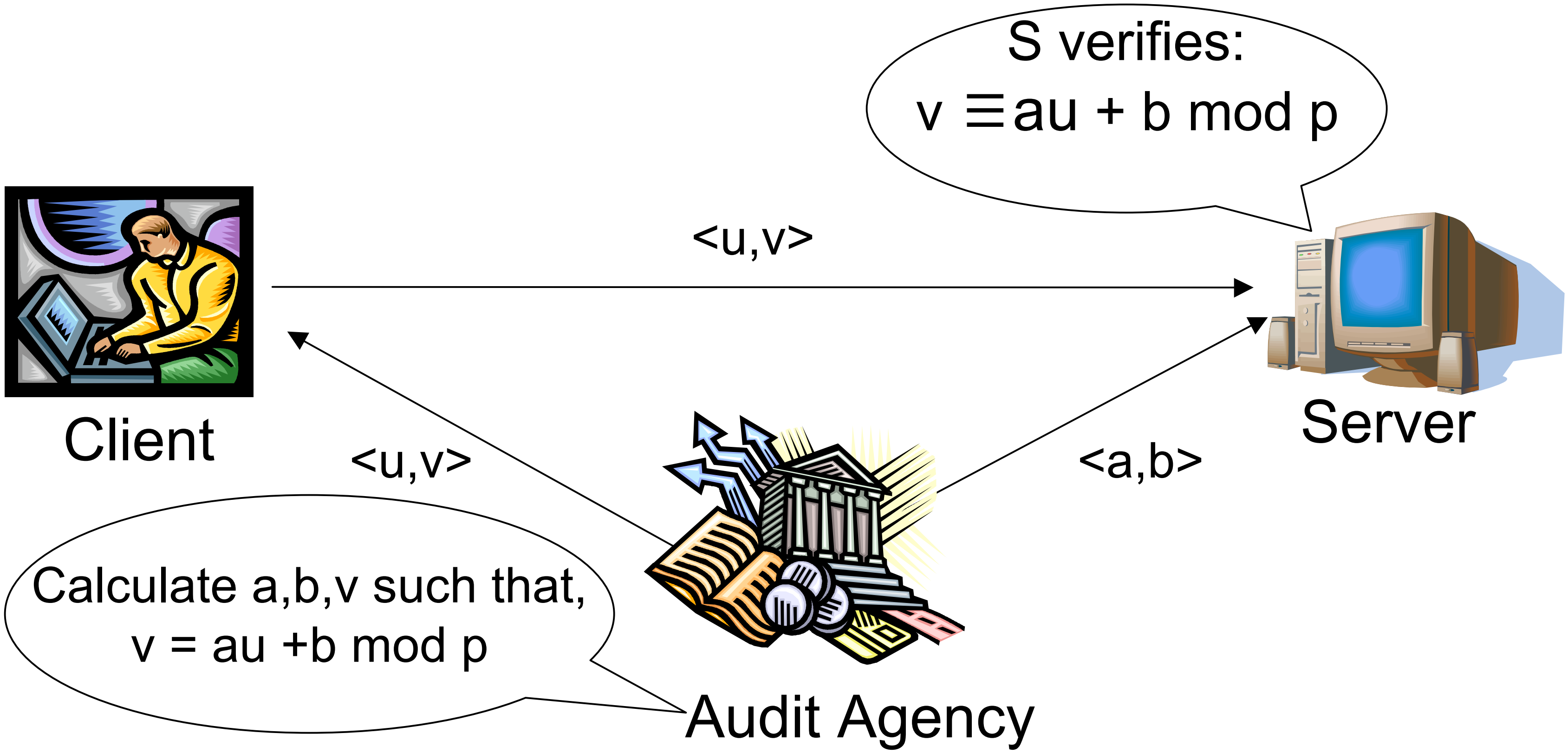
$$y = f(x) = s + \sum_{j=1}^{k-1} f_j x^j$$

$$s_i = (x_i, y_i)$$

$$g^{y_i} = g^s (g^{f_1})^{x_i}$$

Robustness: Alternate Method

- Audit agency wants the client to tell the server *u*.



S verifies:

$v \equiv au + b \bmod p$

Calculate a,b,v such that,

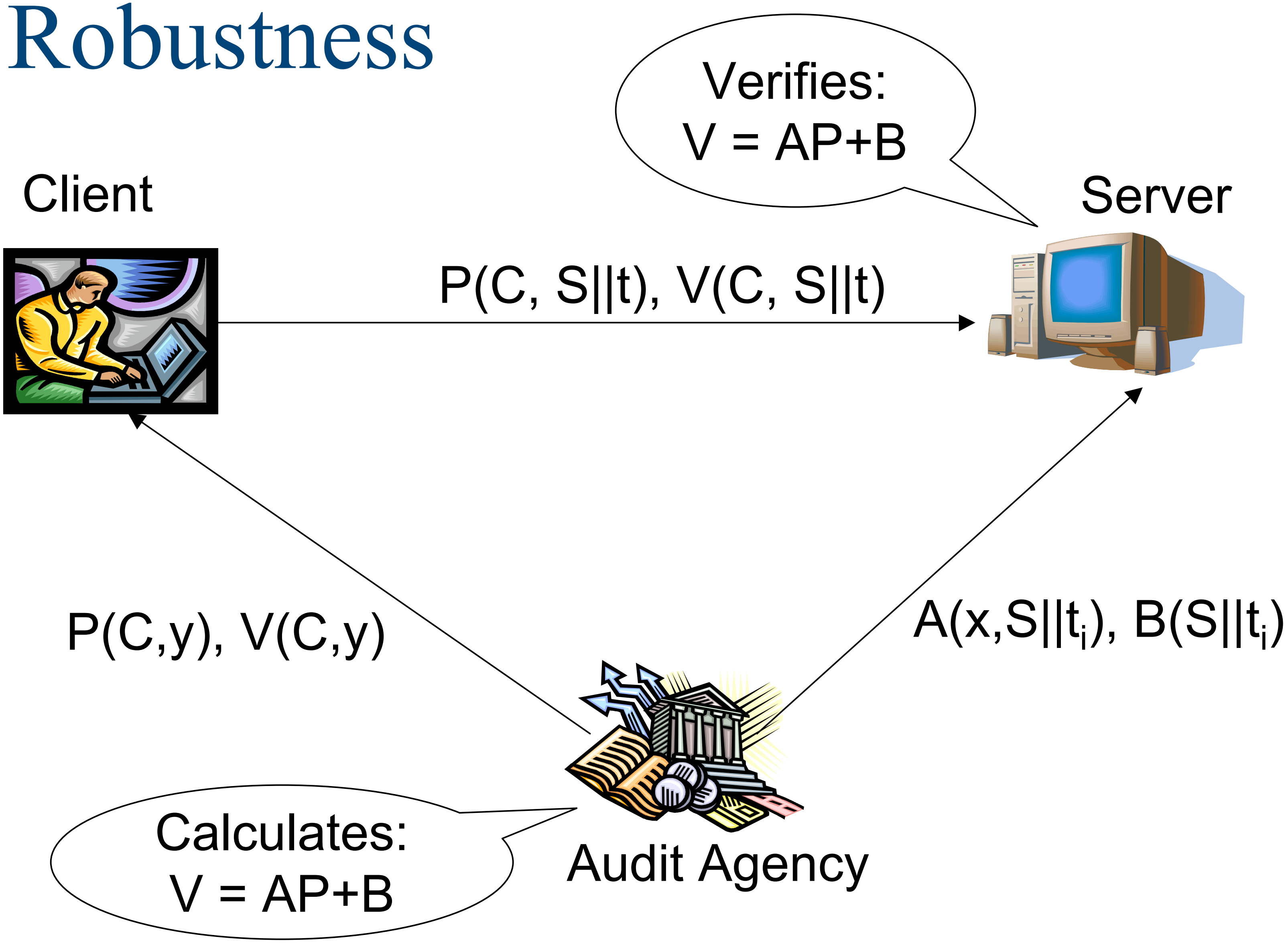
$v = au + b \bmod p$

Robustness

- P(x,y): degree *k-1* in *x*, degree *d-1* in *y*
- A(x,y): degree *a* in *x* and *b* in *y*
- B(y): degree *b* in *y*
- Audit Agency calculates:

$$V(x,y) = A(x,y) \cdot P(x,y) + B(y)$$

Robustness



Verifies:
V = AP+B

Client

Server

P(C, S||t), V(C, S||t)

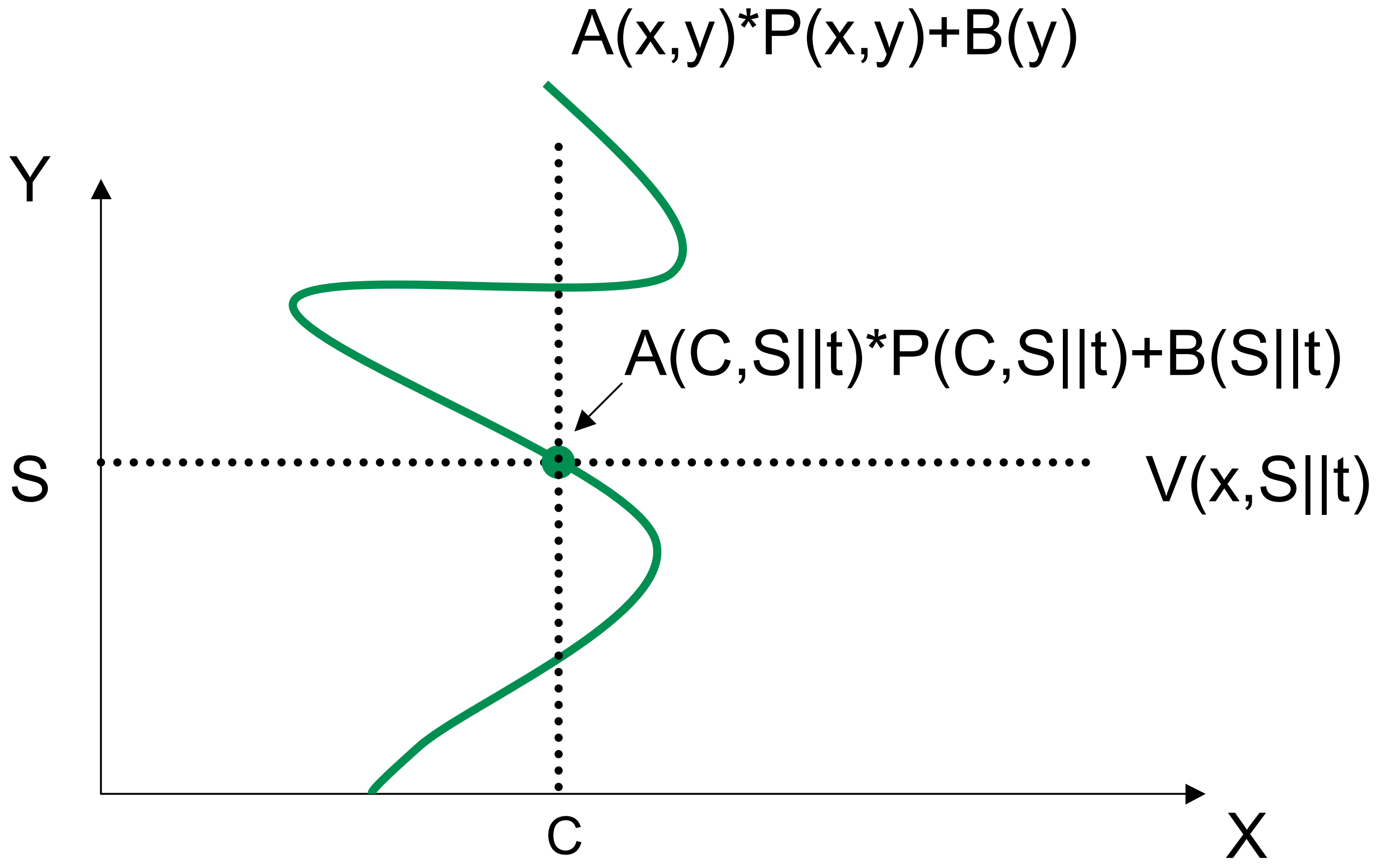
P(C,y), V(C,y)

$A(x,S||t_i), B(S||t_i)$

Calculates:
V = AP+B

Audit Agency

Robustness



Robustness

- Audit agency must compute *V*, *A* and *B*
- Server must store *A* and *B* for all time frames *t*
- Server must compute *A* and *B* for each client that visits
- Server must check *V*=*AP*+*B*
- Client must evaluate *V* for each server and time frame
- Additional communication overhead

Increasing Efficiency

- Divide k into n classes

  $n = k/k'$

- n random polynomials: $P_1(x,y)\ldots P_n(x,y)$
- Map clients randomly to $\{1,\ldots,n\}$
- Client gets respective polynomial $P_i(x,y)$
- Client sends class along with $P_i(C, S||t)$
- Server only needs k' clients from a class to interpolate

Increasing Efficiency

- Coupon Collector problem

Given a set of possible outcomes, what is the expected number of events before the entire set of possible outcomes occurs

Coupon Collector Example

- 3 toys: A,B,C
- Probability of obtaining any toy is 1/3
- Expected time to collect all 3

= E[waiting time for 1st toy] +

E[waiting time for 2nd toy] +

E[waiting time for 3rd toy]

$= 3/3 + 3/2 + 3/1$

= 5.5 tries

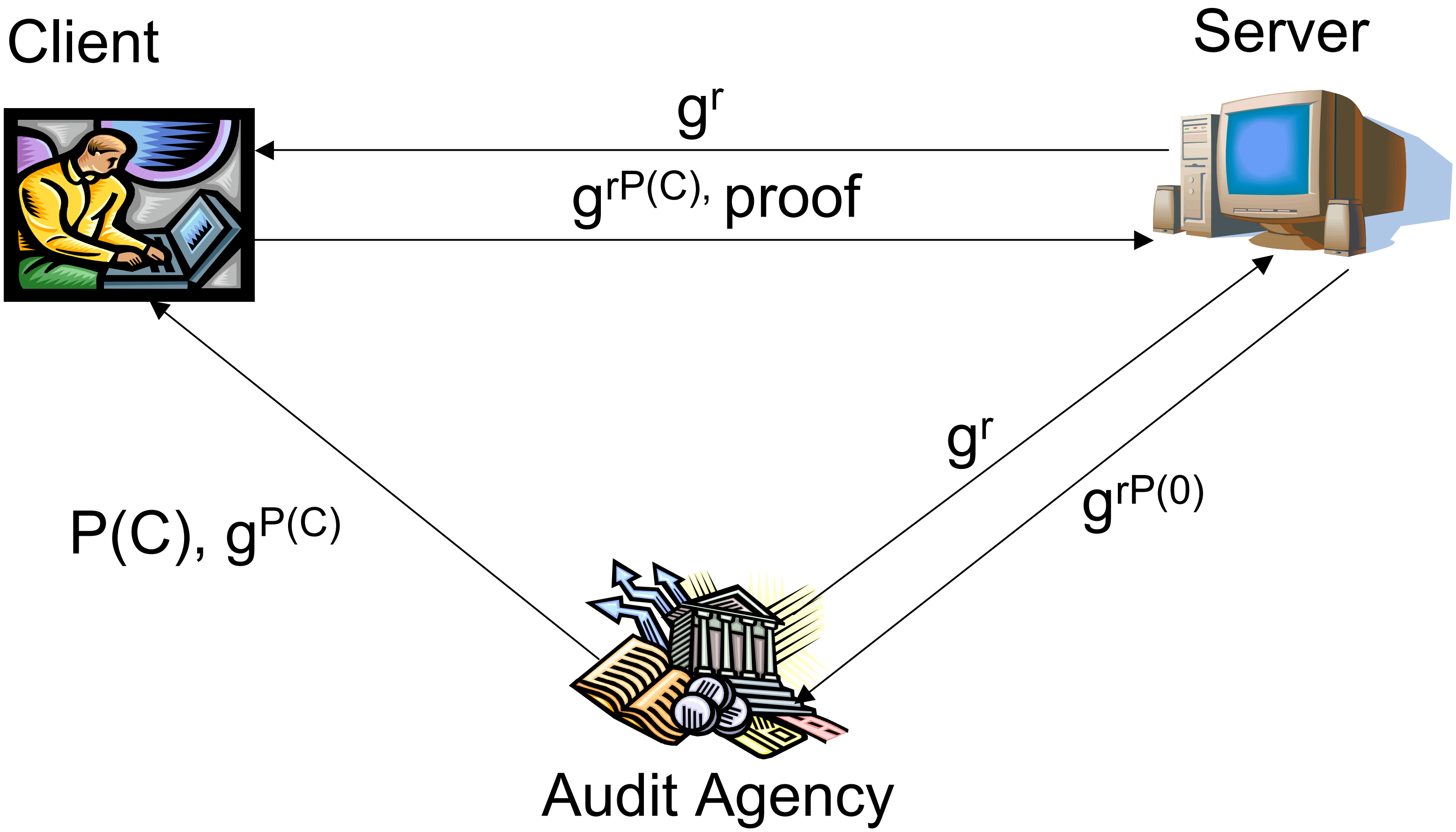
Increased Efficiency

- Audit agency must produce multiple polynomials
- Audit agency must map clients to polynomials and store the mapping
- Server must store the client's class as well as $P_i(C, S||t)$
- Client must store it's class with the polynomial P
- Probabilistic scheme rather than deterministic

Unlimited Use Scheme

- Basic scheme requires replacing *P* after *d* time frames
- Unlimited use scheme parameters
  - generator *g*
  - random value *r*

Unlimited Use Scheme



Client

Server

g^r

$g^{rP(C)}$, proof

g^r

$g^{rP(0)}$

$P(C)$, $g^{P(C)}$

Audit Agency

Unlimited Use Scheme

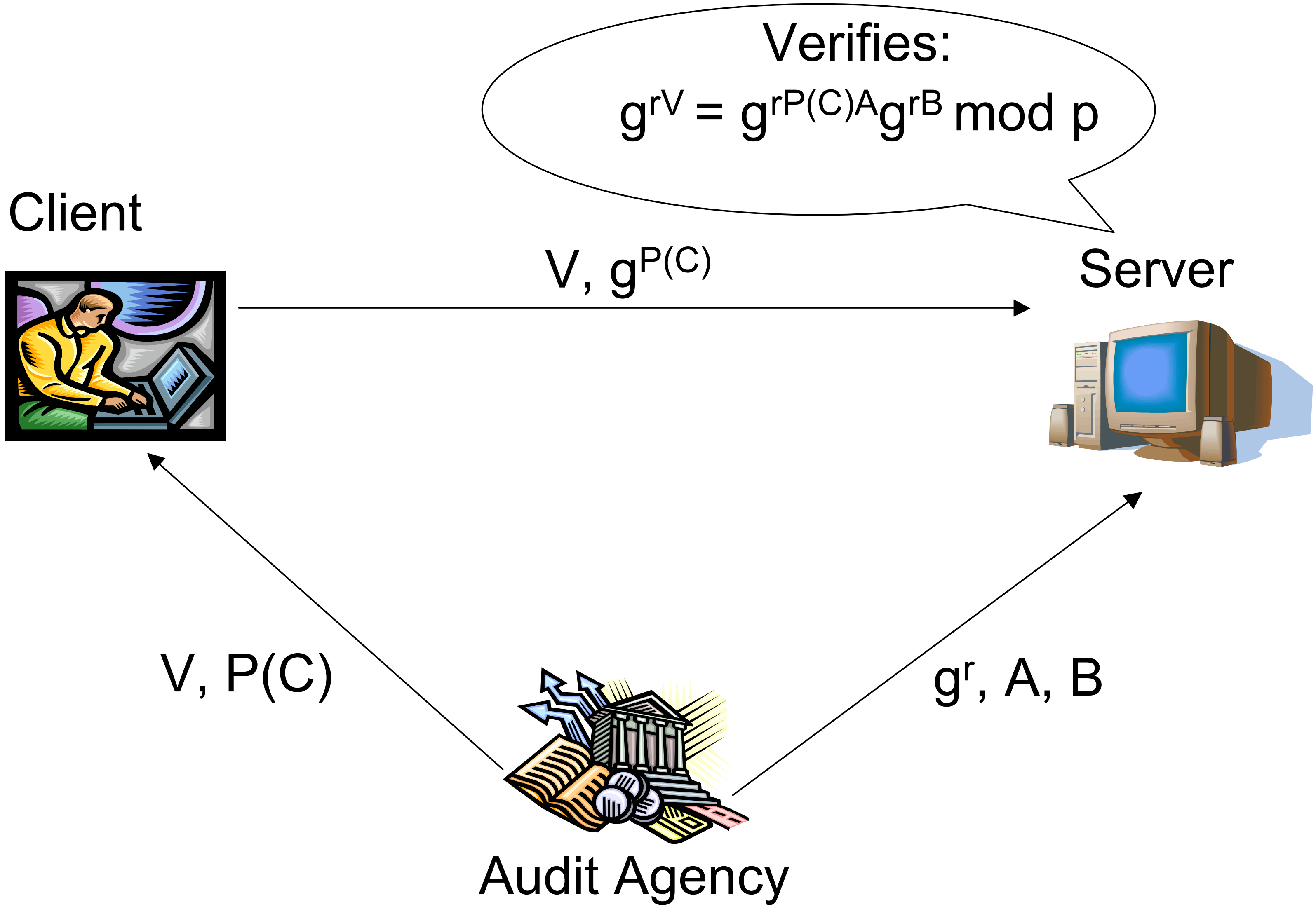
- Decisional Diffie-Hellman
  - – Given g^a, g^b, y, compute if $y == g^{ab}$
- Computational Diffie-Hellman
  - – Given g, g^a, g^b, compute g^{ab}
  - – In this case, the server has g, g^r and $g^{rP(Ci)}$, where $0< i < k$
  - – If it can calculate $g^{rP(0)}$ it can break CDH

Unlimited Use Scheme

- Client proof construction
  - Same as robustness scheme
  - Audit agency calculates $V(x,y)$, $A(x,y)$ and $B(y)$ such that when $x = C$ and $y = S$,

    $g^{rV} = g^{rP(C)A} g^{B} \mathit{mod}\ p$

Unlimited Use Scheme



Verifies:

$g^{rV} = g^{rP(C)A} g^{rB} \bmod p$

Client

$V, g^{P(C)}$

Server

$V, P(C)$

g^r, A, B

Audit Agency

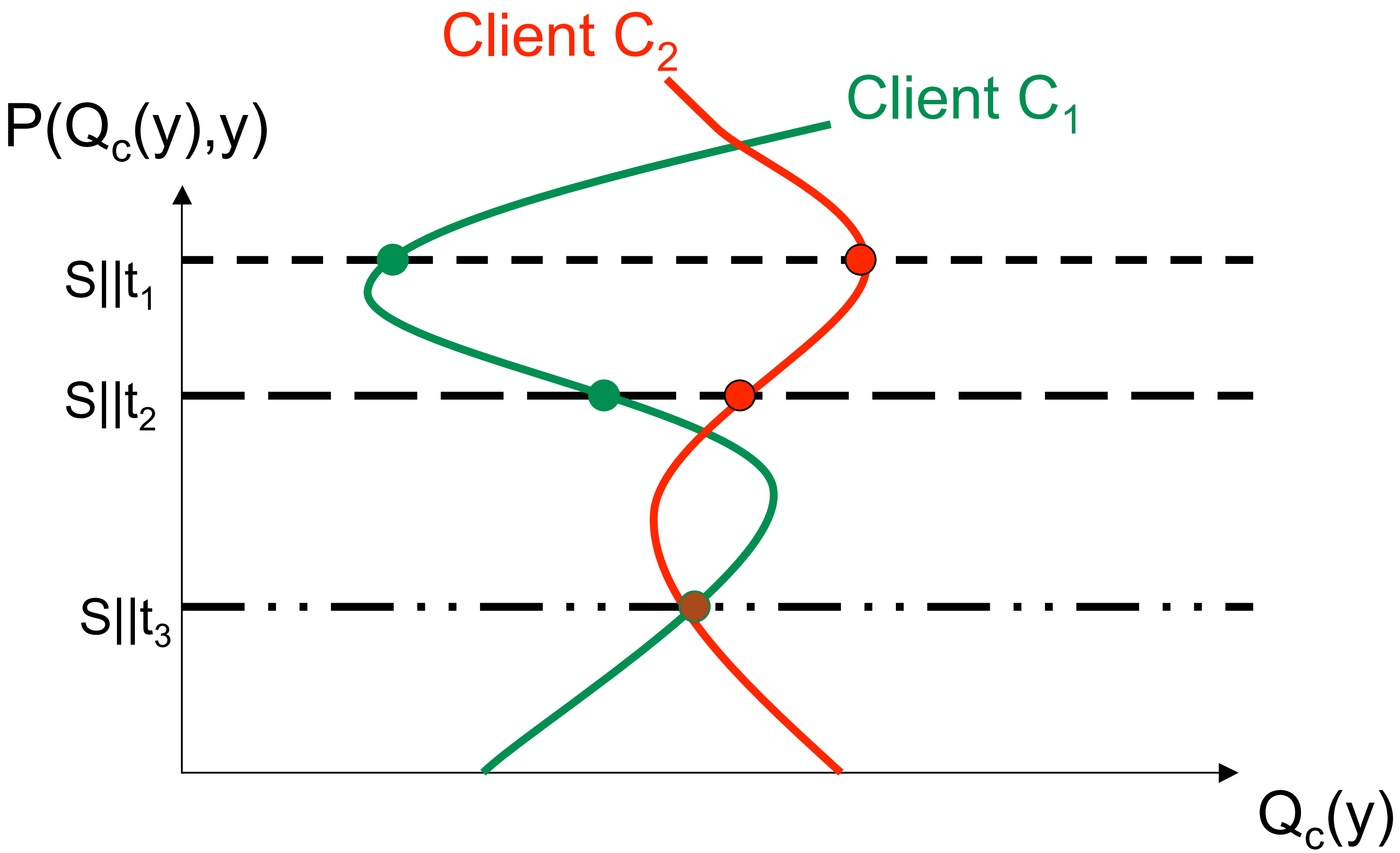
Unlimited Use Scheme

- Exponentiation of polynomials is computationally expensive
- Each time frame a new *r* is used and g^r must be calculated
- Additional communication overhead between audit agency and server
- Server must verify $g^{rV} = g^{rP(C)A} g^{rB} \bmod p$

Anonymity

- Preserves client privacy over multiple time periods
- Instead of $P(C,y)$, have $P(Q_c(y),y)$
  - $Q_c(y)$: random polynomial of degree u
    - where $y = S||t$
  - $Q_c(y)$ changes for each time period

Anonymity



Anonymity

- Audit agency must now generate $Q_c(y)$
- Clients must store $Q_c(y)$
- Clients must calculate $Q_c(y)$ for each visit
- Corrupt audit agencies can cooperate with servers to track client activity

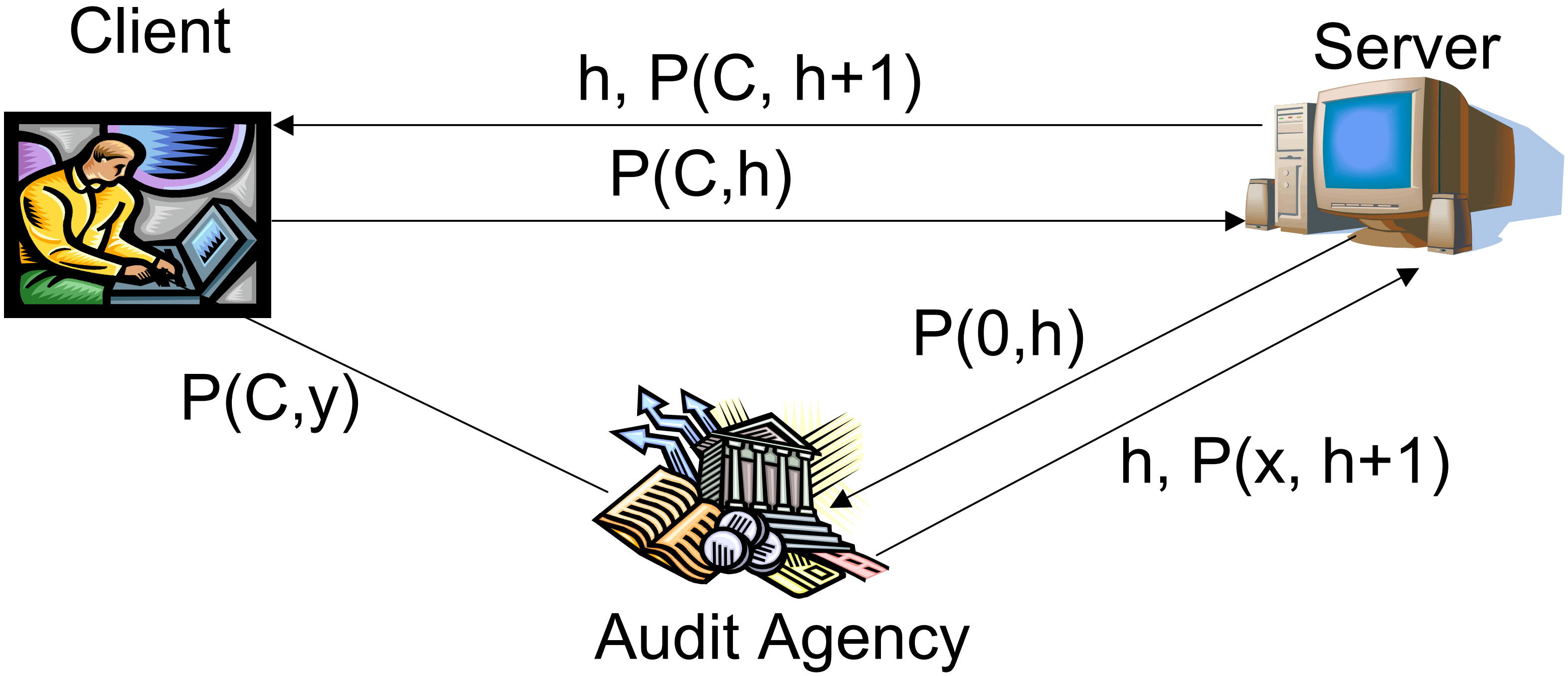
Variants

Variants: Metering Period

- Servers have varying amounts of traffic
- Replace timeframe *t* with challenge *h*
- Allows for variable metering periods
- Server now sends *h* to client when a page is requested

Variants: Metering Period

- Servers now send *h*
- Servers may try to send false *h* values



Variants: Client Turnover

- Advertising agencies may want to determine client loyalty
- Aids in developing payment schemes
- Detects corrupt servers

Variants: Client Turnover

- Audit agency sends server challenge *t* with domain *c*k* and hash function *h* with range *c*k*

- After receiving *c*k* new clients, server should find $g^{riP(C)}$ such that $h(g^{riP(C)}) = t$

Variants: Adaptability

- Servers with less traffic may never see *k* clients for a given time frame
- Decrease *k* to allow more fine grained measurements
- If server receives *k'<k*, ask for *k-k'* polynomial values to complete interpolation
- Server sets *k'*

Open Problems

- Efficient schemes limited usage times
- Unlimited use schemes inefficient
- Value for *k* must be preset
  - Cannot tolerate the number of clients changing
  - Even under adaptability scheme, *k* is still preset

Questions