Algorithmic Aspects of Secure Computation and Communication

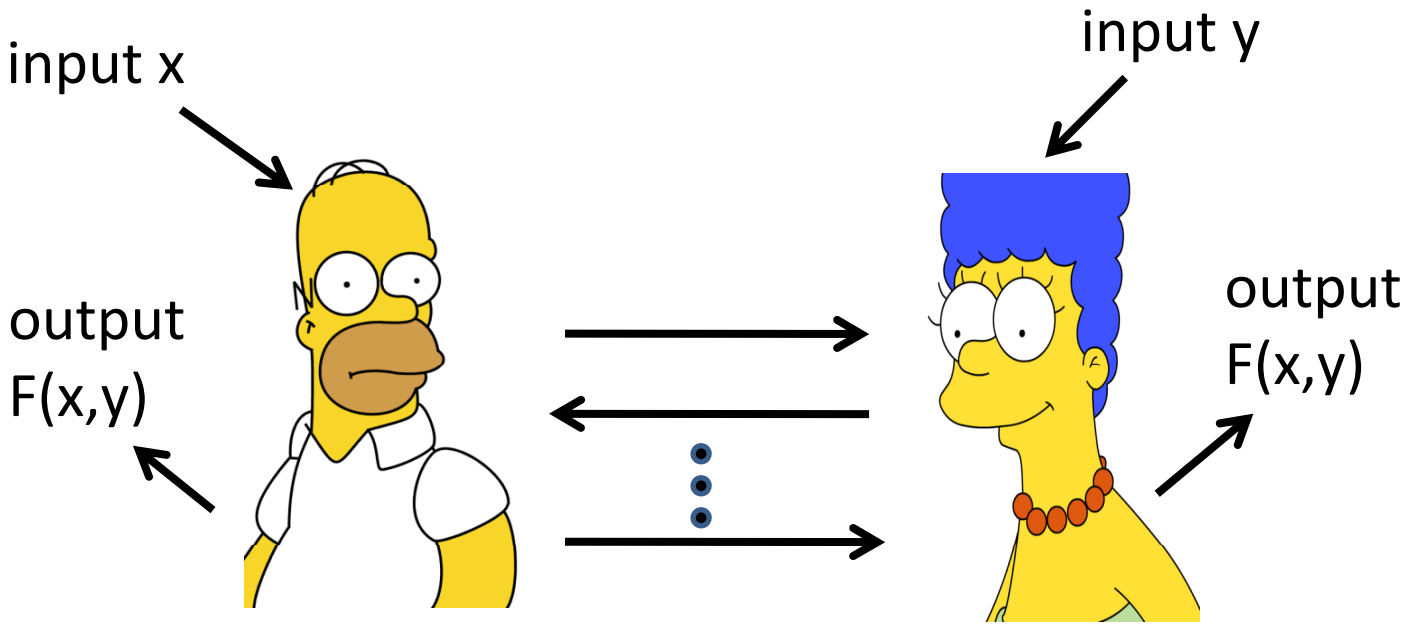
Matt Franklin

U. C. Davis

Algorithmic Aspects of …

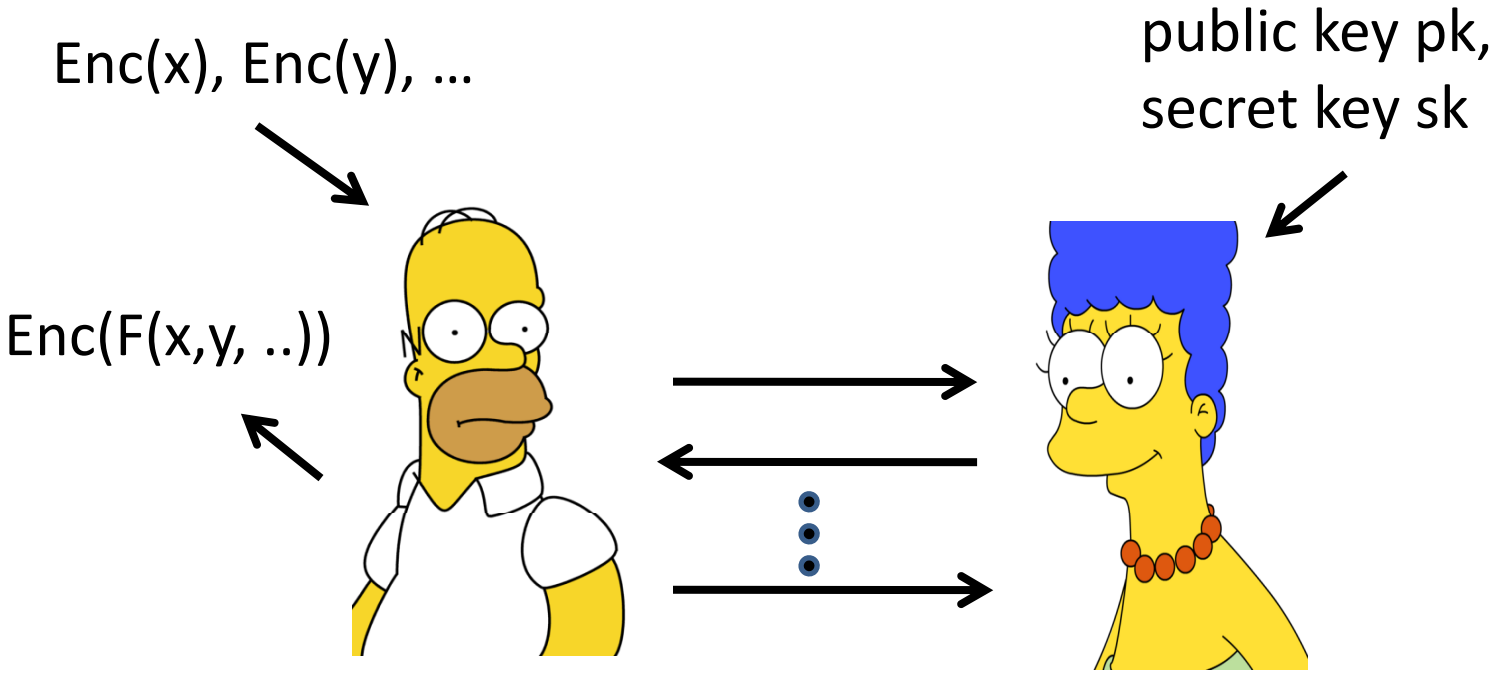
- Secure computation
  - Linear Algebra [KMWF 07]
    - Algorithmic idea: Linearly Recurrent Sequences
  - Longest Common Subsequence [FGM 09]
    - Algorithmic idea: Four-Russians speedup
- Secure communication
  - Reliable Message Transmission [BF 99, BM 05]
    - Algorithmic idea: graph connectivity variants

Secure Computation: Private 2-Party Setting



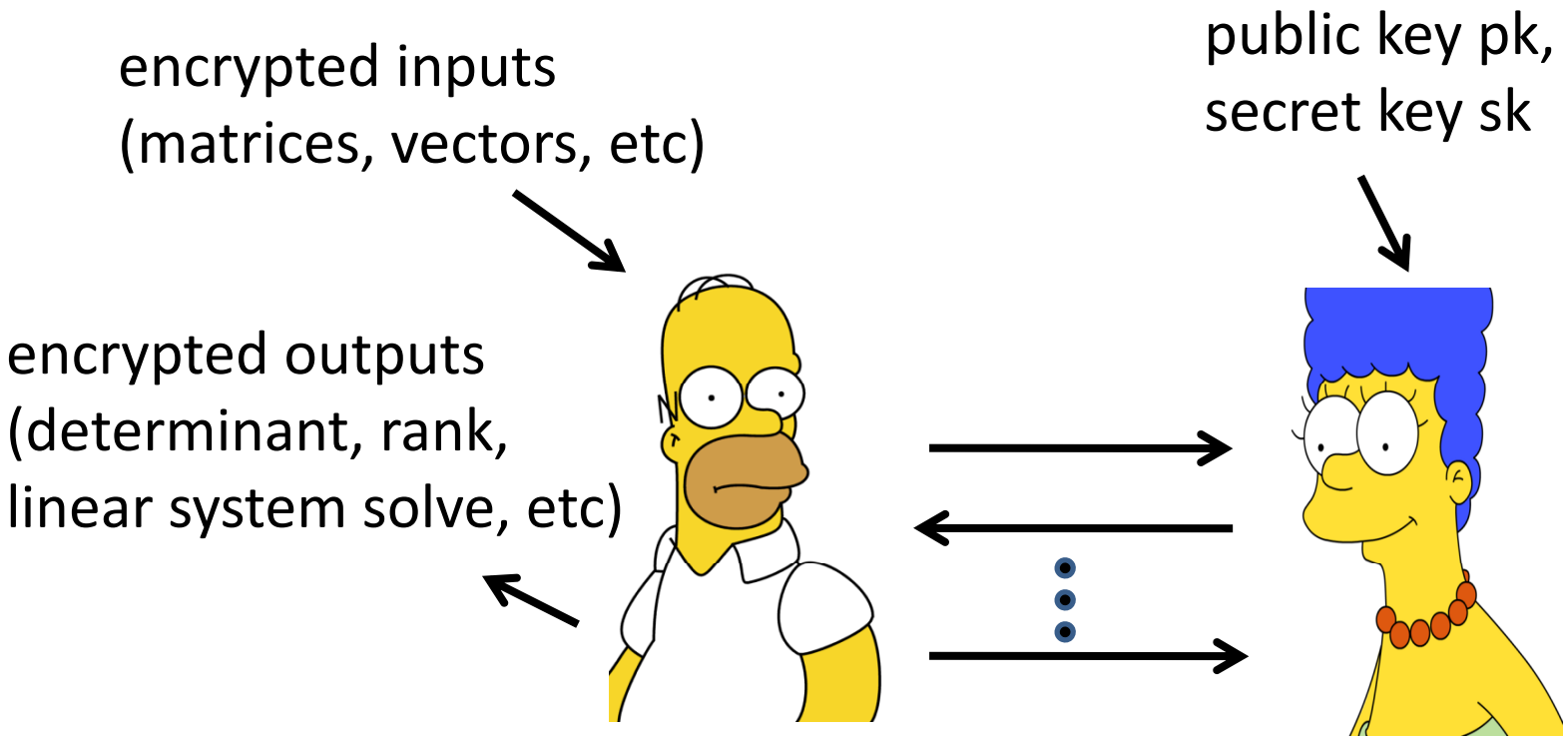
Compute function without leaking inputs.
semi-honest adversary (passive faults)
Classic problem [Yao 1986, Goldreich 2004, Canetti2000]

Secure Computation: Private 2-Party Variant



Essentially equivalent to the classic setting.

Private 2-Party Linear Algebra



Private 2-Party Linear Algebra

	Communication complexity	Round complexity
Yao86	$\tilde{O}(n^{2.38})$	O(1)
KMWF07	$\tilde{O}(n^2)$	O(log n)

Communication = # of encrypted values exchanged

inputs = n by n matrices, etc.

Private 2-Party Linear Algebra

	Communication complexity	Round complexity
Yao86	$\tilde{O}(n^{2.38})$	O(1)
KMWF07	$\tilde{O}(n^2)$	O(log n)
Gentry 09	$\tilde{O}(n^2)$	O(1)

fully homomorphic encryption

Linearly Recurrent Sequences for Solving Linear Systems

- Faster than Gaussian Elimination for:
  - sparse linear systems (Wiedemann 86)
    - Ay = x where A has few nonzero entries.
  - special form linear systems (Kaltofen-Sanders 91)
    - Ay = x where Av can be computed “fast” (for all v)
    - Sparse, Vandermonde, Sylvester, Toeplitz, etc.
- We apply to ordinary matrices

Linearly Recurrent Sequences of Field Elements

- The sequence of field elements $(a_i)_i$ is linearly recurrent if there exists field elements $f_0, \ldots, f_n$ such that $f_0 a_i + \ldots + f_n a_{i+n} = 0$ for all i.
  - $f(x) = f_n x^n + \ldots + f_0$ is a characteristic poly of $(a_i)_i$
- minimal poly = char poly of least degree
  - O(n polylog n) algorithm to compute min poly
  - Padé Approximation, Fast Extended GCD.

Linearly Recurrent Sequences of Matrices

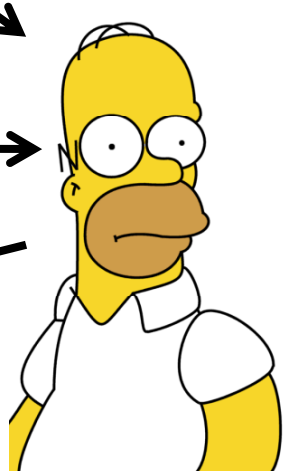
- The sequence of matrices $(m_i)_i$ is linearly recurrent if there exists field elements $f_0, \ldots, f_n$ such that $f_0 m_i + \ldots + f_n m_{i+n} = 0$ for all i.
  - $f(x) = f_n x^n + \ldots + f_0$ is a characteristic poly of $(m_i)_i$
- Min poly of $(m_i)_i$ = char poly of least degree
- Min poly of matrix M = min poly of $(M^i)_i$
  - Useful for computing rank, determinant, etc.

Additively Homomorphic Encryption

add-hom public key pk

Enc(x), Enc(y)

Enc(x+y)



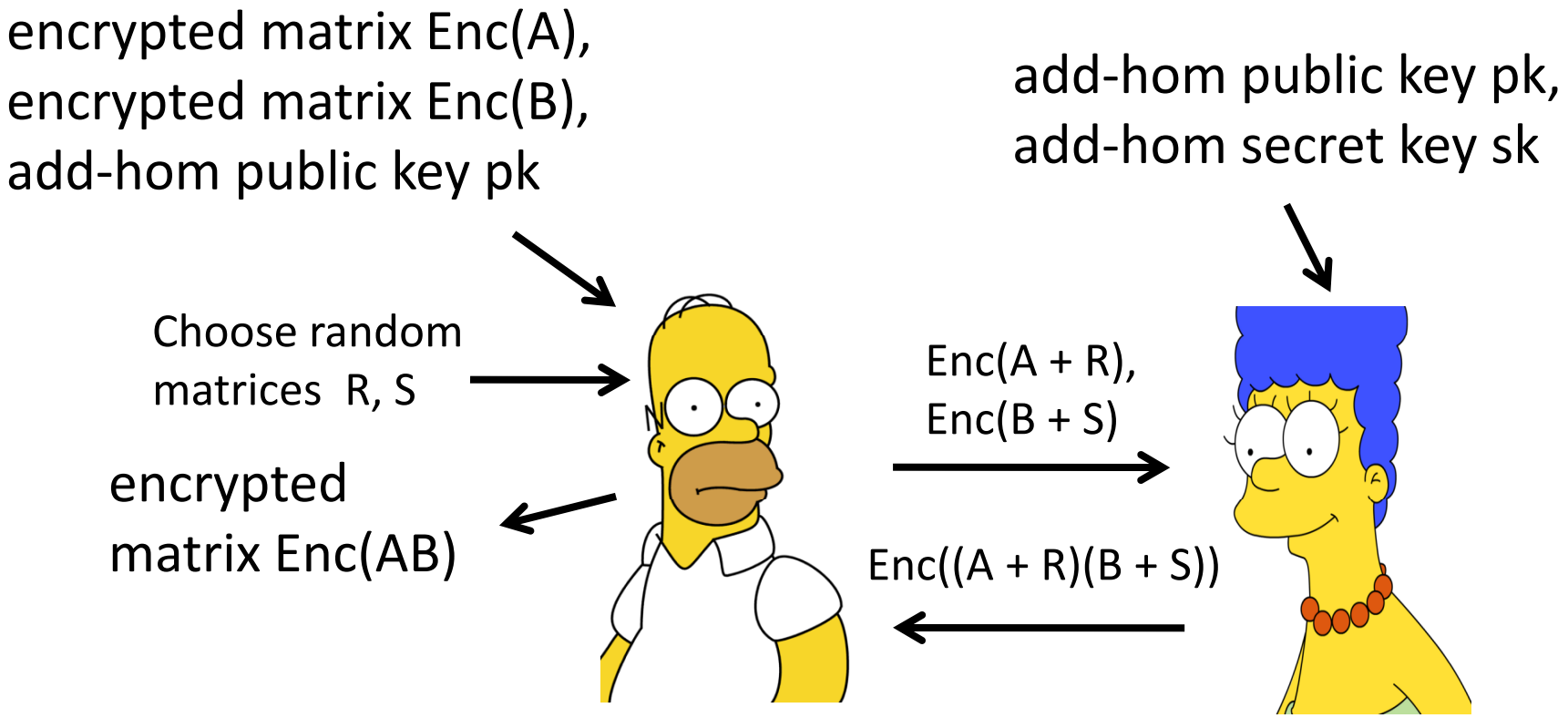
add-hom public key pk,
add-hom secret key sk

Relatively mild assumption (e.g., Paillier 1999)

Yao's Garbled Circuit Protocol [1986]

- Private 2-party computation for any function
- Communication complexity $O(G + \alpha + \beta)$:
  - G = size of Boolean circuit to compute function
  - α = number of inputs, β = number of outputs
- Our solutions use Yao as a sub-protocol
  - On functions with small boolean circuits

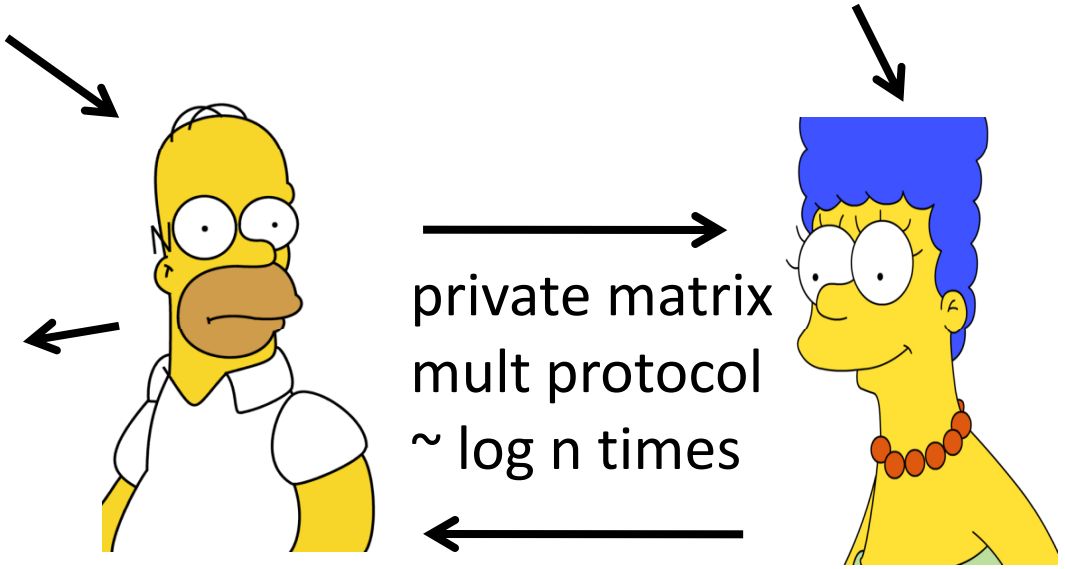
Efficient Private Matrix Multiply



Efficient Private Matrix Exponentiation

encrypted matrix Enc(A),
add-hom public key pk

add-hom public key pk,
add-hom secret key sk



encrypted matrices
$Enc(A^2)$, $Enc(A^4)$,
$Enc(A^8)$, …, $Enc(A^n)$

private matrix
mult protocol
~ log n times

Efficient Private Sequence Mult

encrypted matrix Enc(A),
unencrypted vector u, v
add-hom public key pk

add-hom public key pk,
add-hom secret key sk

private matrix
exponentiation

~ log n private
matrix mults
(carefully chosen)

encrypted field elements
$Enc(u^T A^i v)$, $0 \le i \le 2n-1$

Private MINPOLY Protocol

Input: enc(A) = encrypted n by n matrix

Output: encrypted min poly of A

1. Pick random vectors u, v.
2. Compute enc($u^T A^i v$) for i = 0, .., 2n-1 with Efficient Private Sequence Mult Protocol
3. Compute encrypted min poly of encrypted sequence of field elements from step (2), with Yao's protocol on "small" boolean circuit.

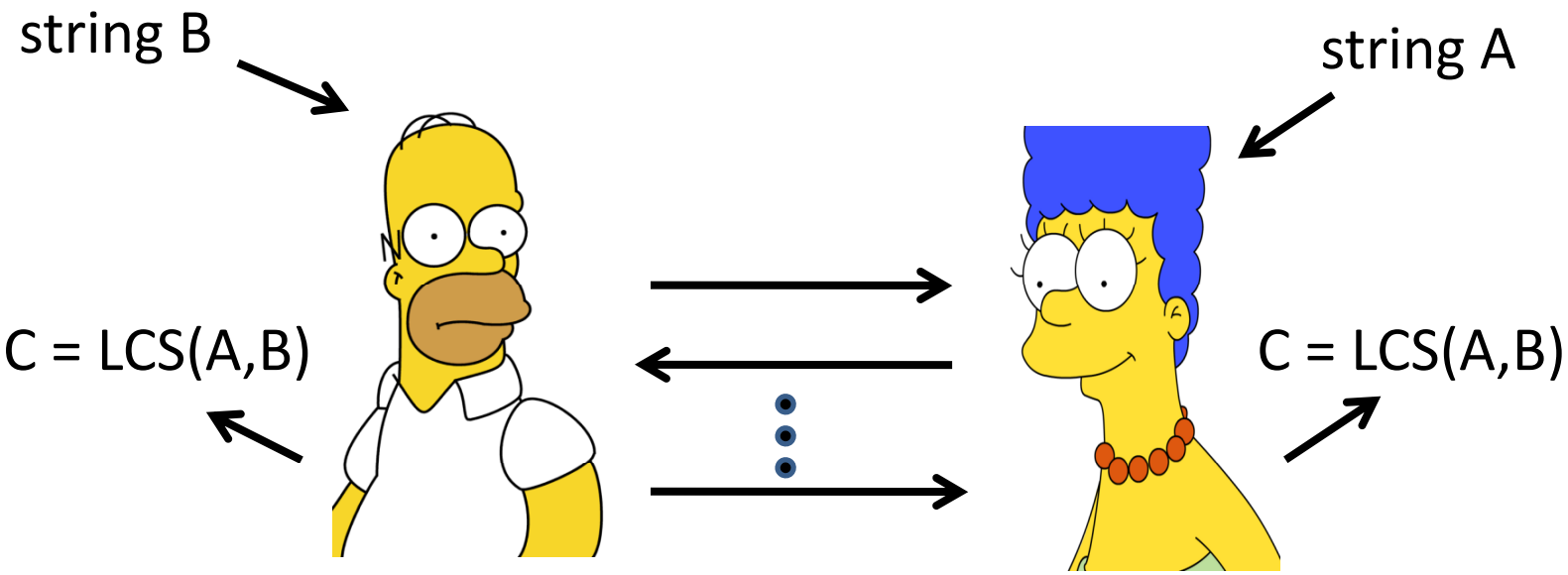
Private MINPOLY Protocol Analysis

- min poly m_A of matrix A = min poly of $(u^T A^i v)_i$ with prob $\geq 1 - (2 \deg(m_A) / |F|) \geq 1 - 2n/|F|$ when A is an n by n matrix.
- Private MINPOLY protocol computes $\mathrm{Enc}(m_A)$ from Enc(A) with prob $\geq 1 - 2n/|F|$, using
  - $O(n^2 \log n \log |F|)$ communication
  - O(log n) rounds.

Private 2-Party Linear Algebra: Other Functions

- Determinant, rank, linear system solution, etc
  - Same efficiency as Private MINPOLY
  - $O(n^2 \log n \log |F|)$ comm, $O(\log n)$ rounds
  - Similar ideas and techniques
- See paper for details [KMWF07].
- Open: $O(n^2 \text{polylog}(n, |F|))$ comm, $O(1)$ rounds
  - without using “big hammer”

Private 2-Party Longest Common Subsequence



Both parties learn LCS(A,B), while hiding inputs otherwise.
Today's talk: Both parties learn size of LCS.

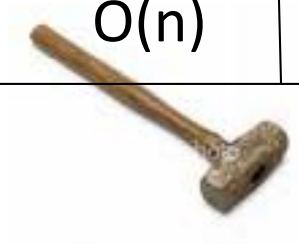
Private 2-Party LCS

	Comm complexity	Alice's Work	Bob's Work	Round complexity
Yao86	O(mn)	poly(m,n)	poly(m,n)	O(1)
FGM09	O(mn/t)	$O(2^t)$	O(mn/t)	O(m/t + n/t)

For strings of length m and n over constant-size alphabet.
For any t, $1 \leq t \leq \min(m,n)$.

Private 2-Party LCS

	Comm complexity	Alice's Work	Bob's Work	Round complexity
Yao86	O(mn)	poly(m,n)	poly(m,n)	O(1)
FGM09	O(mn/t)	$O(2^t)$	O(mn/t)	O(m/t + n/t)
Gentry 09	O(n)	O(n)	poly(m,n)	O(1)



fully homomorphic encryption

LCS Dynamic Programming

(Needleman-Wunsch, Smith-Waterman)

A \ B	a	f	b	c	a	a	a	d
a	1	1	1	1	1	1	1	1
b	1	1	2	2	2	2	2	2
f	1	2	2	2	2	2	2	2
c	1	2	2	3	3	3	3	3
e	1	2	2	3	3	3	3	3
a	1	2	2	3	4	4	4	4
d	1	2	2	3	4	4	4	5

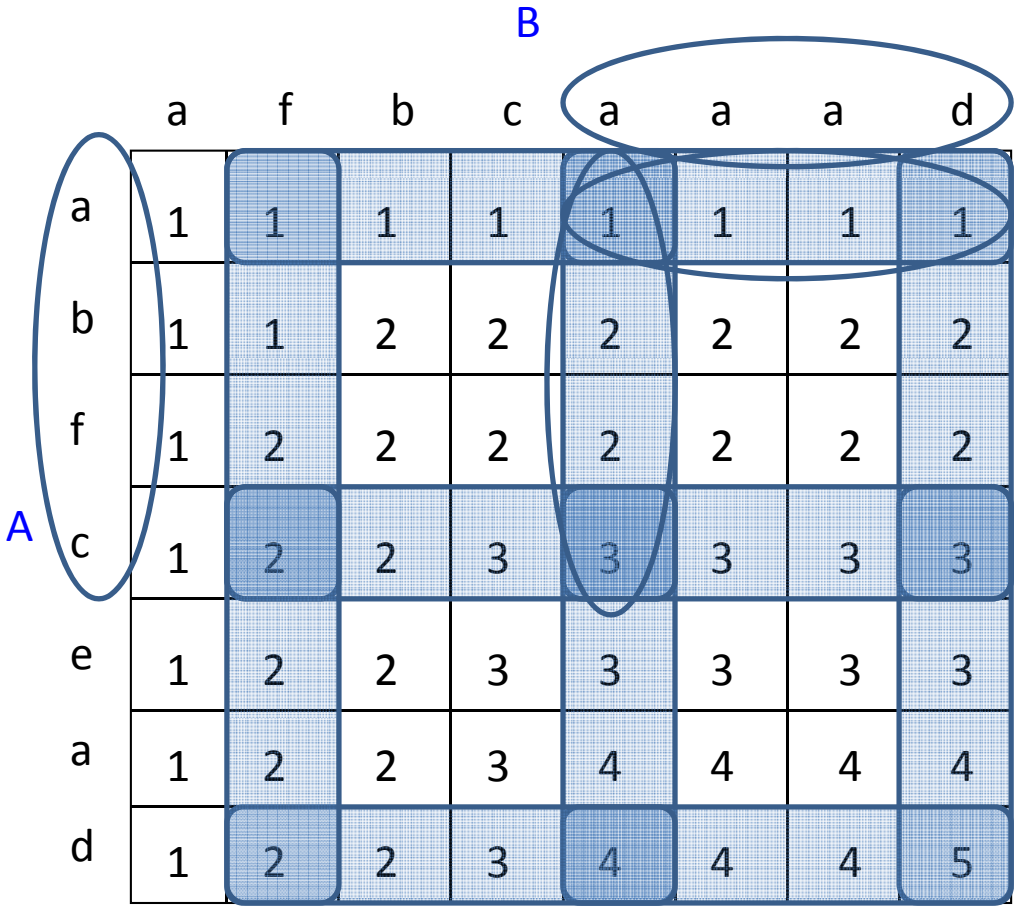
if A[i] = B[j] then
  L[i, j] = L[i-1, j-1] + 1
if A[i] ≠ B[j] then
  L[i, j] = max(L[i-1, j], L[i, j-1]

LCS-length(abfc, afbcaa)

LCS-length(A, B)

Four-Russians Speedup

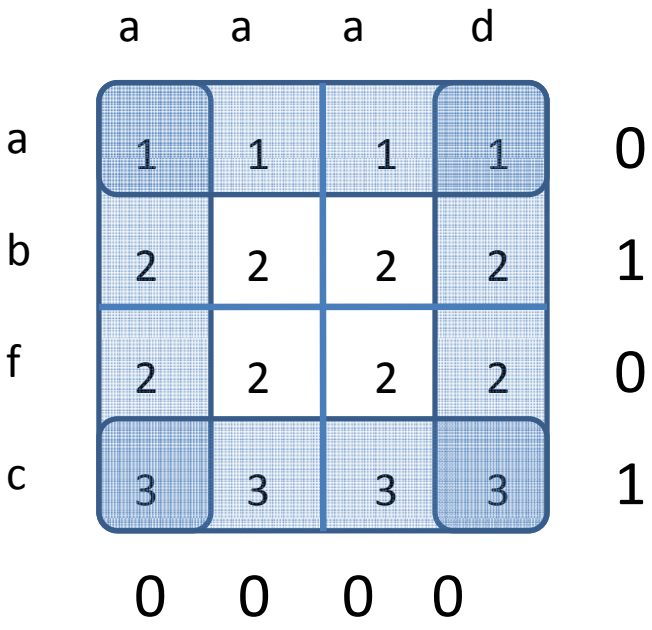
(Masek-Paterson)



Overlapping t by t blocks

Each block determined by:
top row of the block,
left column of the block,
length-t substring of A,
length-t substring of B.

Four-Russians Speedup



Offset vector = increments in row or column of block

Shift value = maximum increase within block (redundant but useful)

F(top row offset, left column offset, substrings of A and B)
= bottom row offset, right column offset, shift value

Four-Russians Speedup

- Pre-compute all possible t by t blocks
  - 2^{2t} possible offset vectors for top row, left column
  - $|\sum|^{2t}$ possible length-t substrings for A and B
  - (bottom row offset vector, right column offset vector, shift value) stored
- Fill in the block boundaries of the LCS table:
  - Look up the appropriate pre-computed block
  - Add offset vectors and shift value to LCS table.
  - Repeat (m/t) * (n/t) times

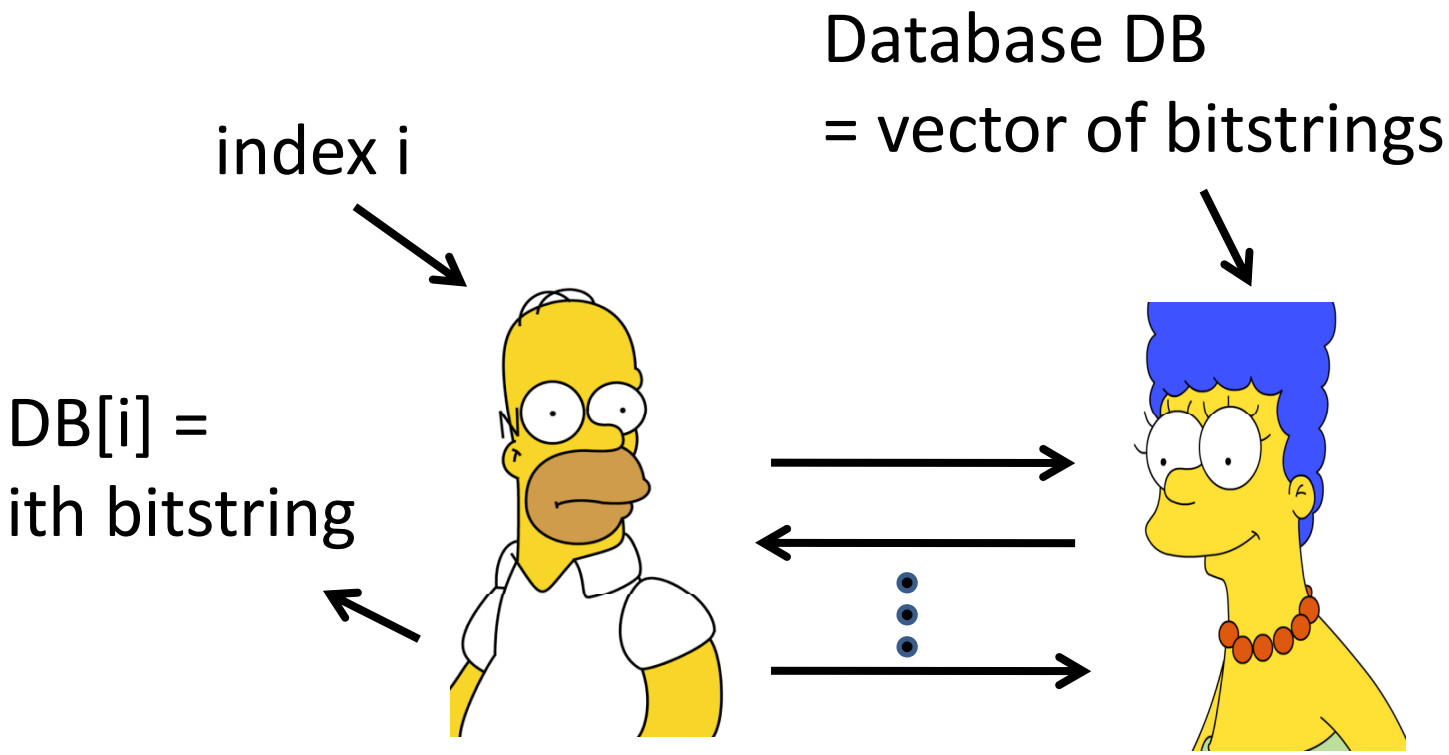
Four-Russians Table Look-Up

index	output
(i_1,i_2,i_3,i_4)	(offsets, shift value)
(j_1,j_2,j_3,j_4)	(offsets, shift value)
(k_1,k_2,k_3,k_4)	(offsets, shift value)

$O(2^t)$ entries of size $O(t)$

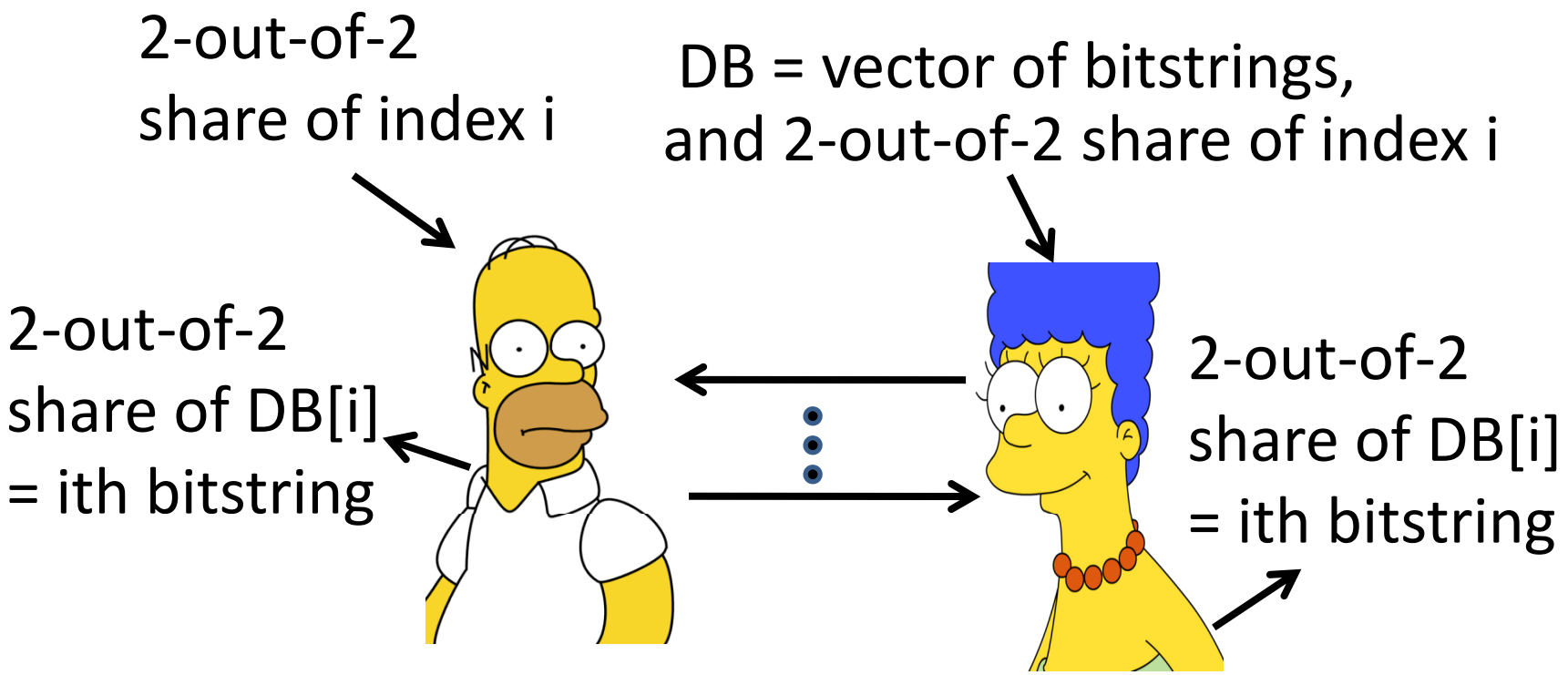
for alphabet of size $O(1)$

Private Block Retrieval



Index hidden from database owner.
Nothing hidden from index holder.

Private Indirect Indexing



Index hidden from database owner.
Database hidden from index holder.

Efficient Private Indirect Indexing

- Private Block Retrieval (Gentry-Ramzan):
  - O(k+d) comm for d-bit blocks (sec param k)
  - Hardness related to ϕ-Hiding Assumption
    - RSA modulus n "hides" small factors of $\phi(n) = (p-1)(q-1)$
- Private Indirect Indexing from any PBR:
  - General transform (Naor-Pinkas, Naor-Nissim)
  - No asymptotic increase in communication
  - Requires pseudorandom functions

Our Two-Party LCS Protocol

- Alice pre-computes Four-Russians table.
  - Indexed by top row (offset), left column (offset), length-t substrings of A and B.
  - Entries are bottom row offset vector, right column offset vector, shift value.
- Alice and Bob iterate Private Indirect Indexing:
  - compute shares of offset vectors, shift values
- L(m,n) recovered from shift value shares

Total Cost

mn/t^2 Private Indirect Indexings,

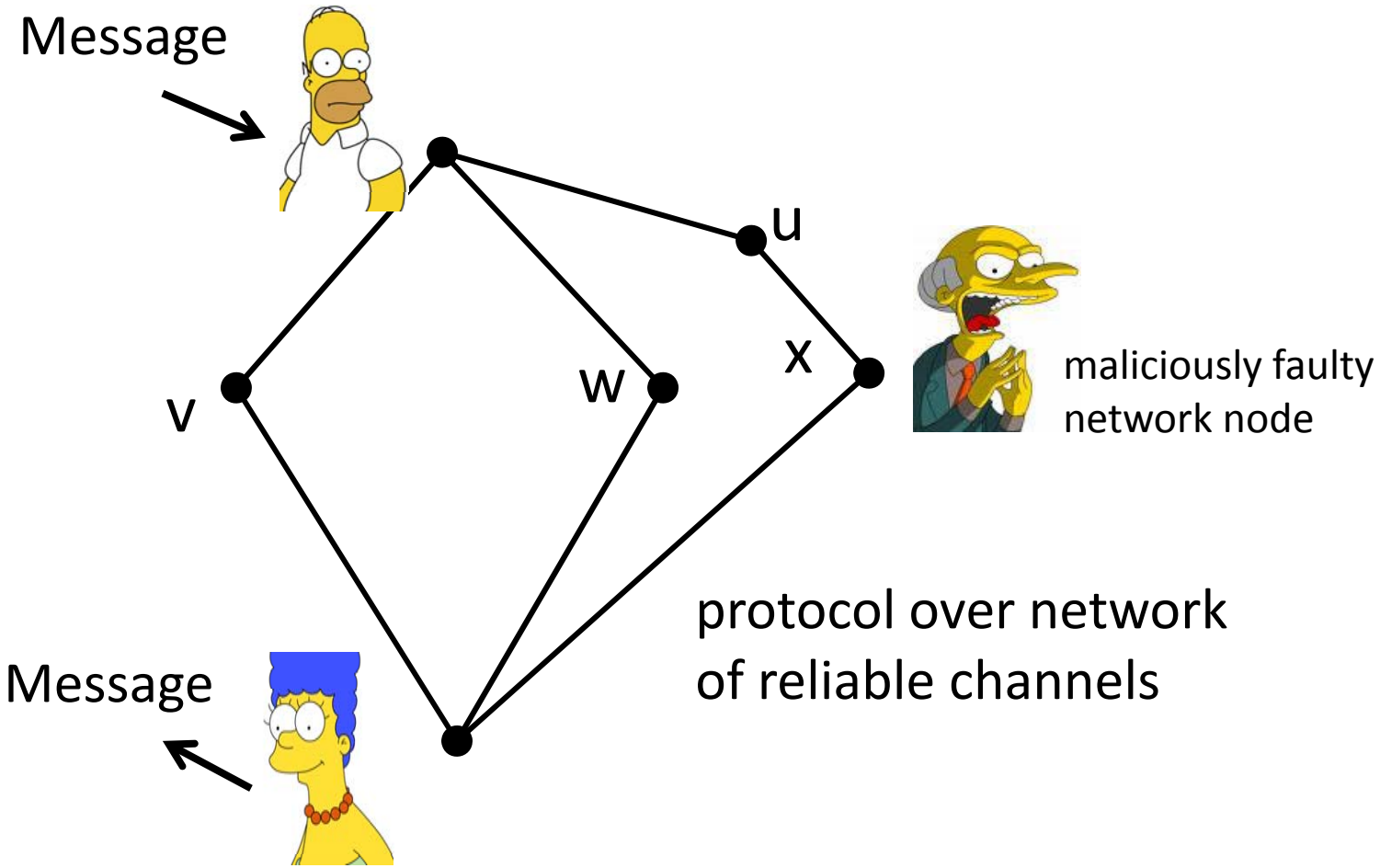
$m/t + n/t$ rounds (parallelizing)

See paper for details [FGM09]

Comm complexity	Alice's Work	Bob's Work	Round complexity
$O(mn/t)$	$O(2^t)$	$O(mn/t)$	$O(m/t + n/t)$

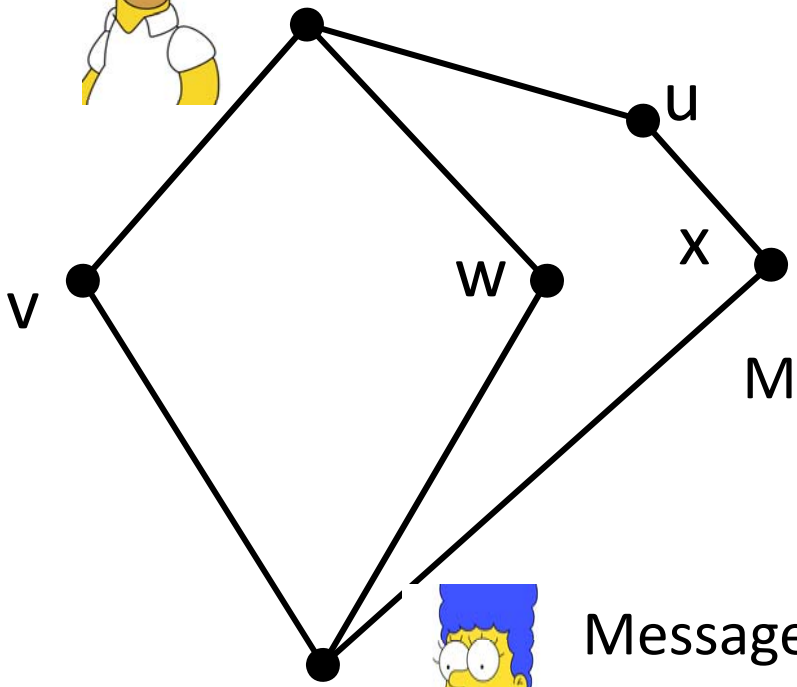
Open: Reduce comm and rounds with poly work (without using "big hammers").

Secure Communication: Reliable Message Transmission



Secure Communication: Reliable Message Transmission

Message sent on 3 paths



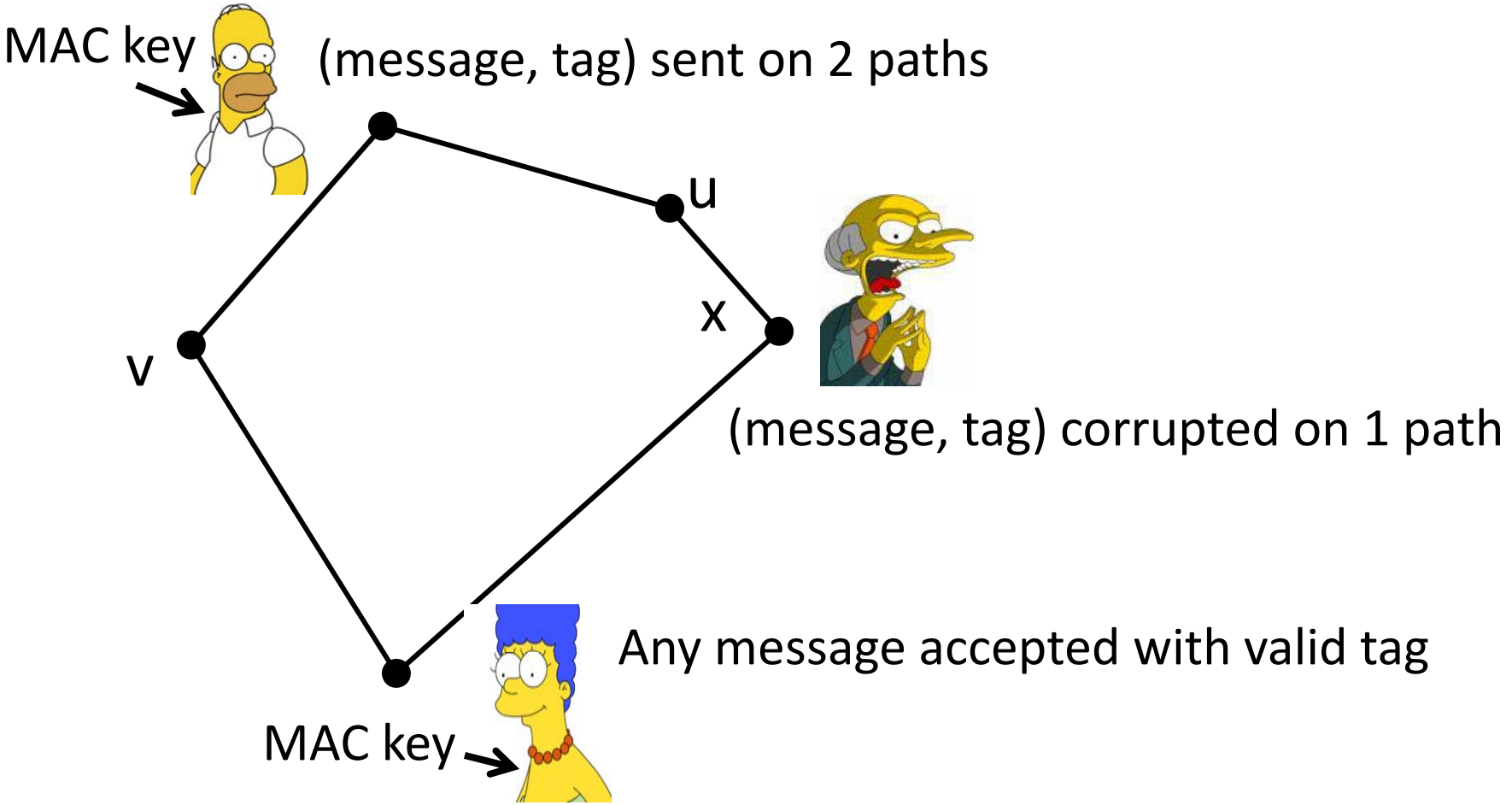
Message corrupted on 1 path

Message accepted if received on 2 paths

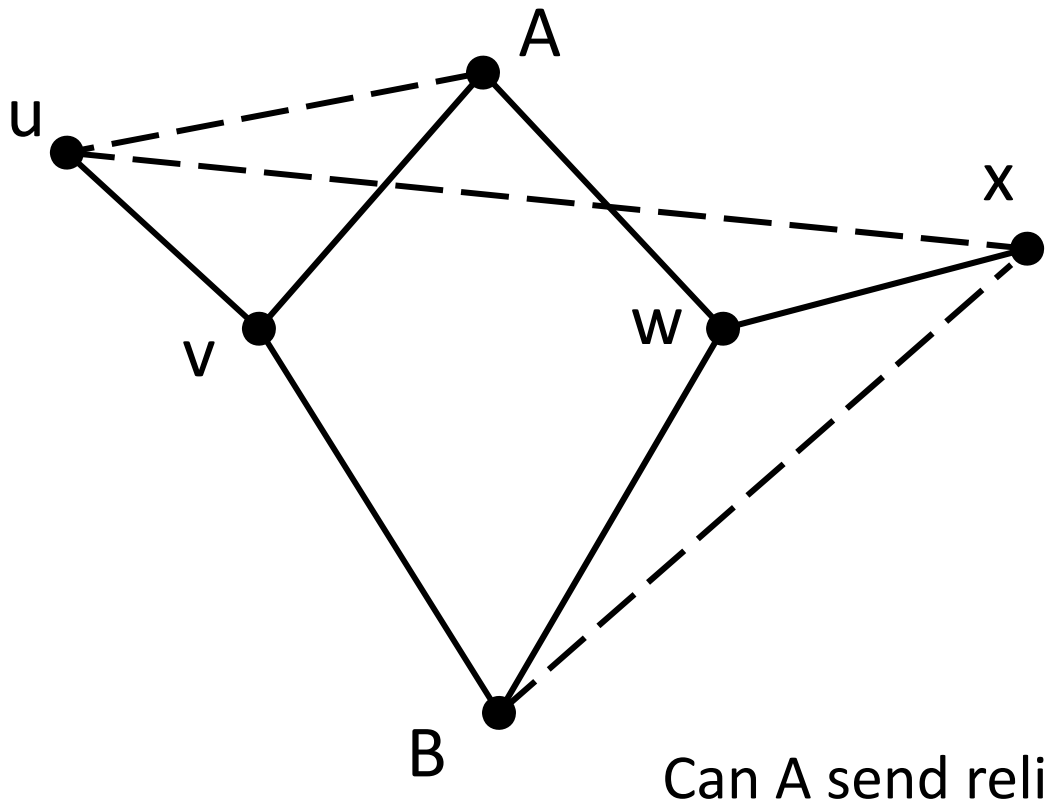
Reliable Message Transmission

- t malicious faults ⇔ (2t+1)-connected [DDWY]
  - Simple efficient protocol (“majority vote”)
  - Perfect reliability (failure probability 0)
- Can MAC’s help?
  - MAC = symmetric key analog of digital signature
  - MAC(message, key) = hard-to-forge “tag”

Reliable Message Transmission when Sender and Receiver Have MAC Key



Reliable Message Transmission with Arbitrary Distribution of MAC Keys



reliable channel

MAC key

Can A send reliably to B if u, v, w or x is malicious fault?

(t,ε)-Reliable Message Transmission with Arbitrary Distribution of MAC Keys

- Send message across network (with ε error)
  - Synchronous network of reliable channels
  - Pre-distributed MAC keys (pair-wise)
  - Adversary controls t malicious faulty parties
- Characterization [Beimel-Franklin 1999]
  - Complex recursive condition on comm/auth networks
  - Highly inefficient: $(n/t)^{O(t)}$ rounds [Beimel-Malka 2005]
  - Open: Find an efficient protocol (or counterexample)

Conclusion

- Rich algorithmic issues in crypto protocols
- Open Problems:
  - find algorithms that map to efficient protocols
  - Exploit relatively efficient “generic” methods
    - Yao: Boolean circuits
    - Naor-Nissim01: Bool circuits with look-up tables
    - BGN05, Gentry09, etc: somewhat-hom encryption
  - Consider protocols with small privacy leakage
- Thank you!